RHEINISCHE FACHHOCHSCHULE KÖLN

University of Applied Sciences

Fachbereich: Wirtschaft & Recht

Studiengang: Wirtschaftsinformatik II (B.SC.)

Projektarbeit

Autoteile NEAP

Projektarbeit von: Patrick Bartschewski, Mat.-Nr. 212172001
Andreas van Bonn, Mat.-Nr. 212172013
Emil Tomaszewski, Mat.-Nr. 212172011
Nadine Hegel, Mat.-Nr. 212172006

Dozent: Patric Steffen

Wintersemester 2018/19

Inhaltverzeichnis

1. Einleitung

Inhalt der Projektarbeit ist es, mit Hilfe bestimmter Techniken für eine konkrete Problem- oder Fragestellung eine Web-Anwendung zu entwickeln. Hierbei soll neben einer Anwendung für Desktop-Browser auch eine für mobile Endgeräte optimierte Variante entwickelt werden, um den speziellen Anforderungen hinsichtlich der Nutzbarkeit auf diesen Geräten gerecht zu werden. Unser Projektteam hat sich für die Aufgabenstellung „Autoteile NEAP“ entschieden und diese erarbeitet. Dabei soll die Inventarisierung von Autoteilen umgesetzt werden, in der verschiedene Merkmale verschiedener Objekte festgehalten und bearbeitet werden können.

Im ersten Schritt haben wir die Aufgabenstellung analysiert, d.h. die Anforderungen an die zu programmierende Web-Anwendung und App. Das Ergebnis haben wir in einem UML-Zustandsdiagramm erfasst und beschrieben, um dann einen Lösungsansatz zu entwerfen.
Das Analyse-Ergebnis unseres UML-Zustandsdiagrammes wurde für uns im Laufe der Erarbeitung der Projektarbeit zu unserem roten Faden und hat uns bei der Verwirklichung unserer Web-Anwendung immens geholfen.
Unser Webentwurf hat in den folgenden Schritten mehr und mehr an Form und Layout gewonnen. Als die Web-Seite „Autoteile NEAP“, entsprechend unseren Vorstellungen fertig kreiert war, haben wir diese mit konkreten Anfragen im letzten Schritt auf alle angeforderten Funktionen getestet und ausgeführt.
Danach wurde hierzu die entsprechende Mobile-App entwickelt.
Die erstellten Quellcode-Auszüge stehen exemplarisch für den vollständigen Code der Web- Anwendung.

2. Problem- bzw. Aufgabenstellung

Es bestanden im Hinblick auf die Entwicklung der Anwendung sowohl funktionale als auch nicht-funktionale Anforderungen. Somit waren mehrere wichtige Komponenten zu beachten. Die Aufgabe war es eine Web-Anwendung zu kreieren, welche im Intranet eines fiktiven Unternehmens einsetzt werden soll. Hierbei war als besondere nicht-funktionale Anforderung zum Beispiel möglichst hohe Benutzerfreundlichkeit gestellt.

Zusätzlich sollte darauf geachtet werden, dass die Anwendung nur von authentifizierten Benutzern genutzt werden kann. Sicherheitslücken, etwa durch SQL-Injection oder Cross-Site-Scripting, waren in diesem Projekt nicht zu beachten. Wichtig war jedoch, dass die Konfigurierbarkeit des Browsers insoweit eingeschränkt ist, als dass sich JAVA-Script nicht ausschalten lässt.

Alle Funktionen die den Datenbestand dauerhaft ändern konnten, sollten stets von dem jeweiligen Nutzer bestätigt werden, um einen versehentlichen Verlust der Daten zu verhindern.

Bei Ausführung dieser Funktionen soll sich ein eigenständiges Dialogfenster öffnen, damit der Nutzer diese Funktion in diesem editieren kann. Ein authentifizierter Nutzer charakterisiert sich durch seine Möglichkeit die Web-Anwendung zu ändern (Löschen, Bearbeiten, Hinzufügen). Sollte dieser eine nicht plausiblen Eingabe machen, so soll er hierauf hingewiesen werden und die Eingabe sollte von der Anwendung abgelehnt werden.

Jedes Element darf von einem beliebigen authentifizierten Benutzer geändert werden, dennoch sollte die Anwendung Fehlersituationen erkennen, die bei gleichzeitiger Nutzung mehrere Nutzer entstehen könnten. Dies sollte mit einem Dialogfensterblock geschehen. Fehlerszenarien mussten weitestgehend duschgespielt werden um diese entsprechend zu verhindern.

Die Anwendung soll ein Single-Page-Interface besitzen. Es war darauf zu achten, die Standards des Hypertext Transfer Protocols weitestgehend auszuschöpfen. Das Layout und die Formatierung unserer Anwendung sollte mithilfe von CSS [1]erfolgen.

Abgesehen von den Funktionen Neuanlagen und Ändern waren alle genannten Anforderungen auch in der App Anwendung erforderlich.

Zusätzlich besteht die Anforderung, eine Installationsanleitung zu erstellen.

[1] Cascading Style Sheets

3. Objektorientierte Analyse

3.1. Allgemeines

Aufgrund der Komplexität der Aufgabenstellung wenden wir die objektorientierte Analyse an. Sie basiert auf miteinander in Beziehung stehenden und kommunizierenden Komponenten, den Objekten. Die Modellierung der Objekte erfolgt dabei in einer Anlehnung an die reale Welt, in dem untersucht wird, wie die Realität aussieht bzw. wie sie vereinfacht abgebildet werden kann.

Die zugrunde liegende Idee der Objektmodellierung ist es, Zustand (Daten) mit Verhalten (Funktionen auf diesen Daten) zu verbinden. Der Zustand (Status) eines Objektes wird durch die Werte der Attribute (Eigenschaften eines Objektes) bestimmt. Das Verhalten eines Objektes wird durch eine Reihe von Methoden beschrieben. Jedes Objekt kann von anderen Objekten stammende Funktionalitäten nutzen bzw. von anderen Objekten „benutzt“ werden.

Objekte mit gleichen Attributen, Verhalten und Assoziationen werden zu Klassen zusammengefasst. Klassen sind Baupläne bzw. Vorlagen für Objekte. Jedes Objekt ist eine Instanz ihrer Klasse.

Interaktionen zwischen realen „Systemen“ (Hotel, Kunde, Zimmer) kann als Interaktion zwischen Objekten modelliert werden. Zum Beispiel stellt das Objekt „Kunde“ eine Anfrage an Objekt „Hotel“ für ein Doppelzimmer. Damit bildet das abstrakte Modell sehr nahe die Realität ab.

3.2. Vorgehensweise bei „Autoteile NEAP“

In der Inventarisierung der Autoteile sind als Objekte die bestimmten Autoteile zu erfassen. Diesen müssen bestimmte Eigenschaften zugeordnet werden – im Fall dieser Anwendung sind dies:

- Titel / Name
- Notizen zum Teil
- Datum der Inventarisierung

- Farbe
- Bestand
- Preis
- Autor / Mitarbeiter

Diese müssen von einem Anwender angelegt, eingesehen und bearbeitet werden können.

Diese Funktionen müssen im Quelltext enthalten sein, sodass hier den Anforderungen entsprechend Objekte mit ihren Eigenschaften geplant und angelegt werden müssen. Die Übergabe von Informationen zwischen Datenbank, Java-Script, PHP und HTML wird über die Eigenschaften und Funktionen durchgeführt.

4. Objektorientierter Entwurf

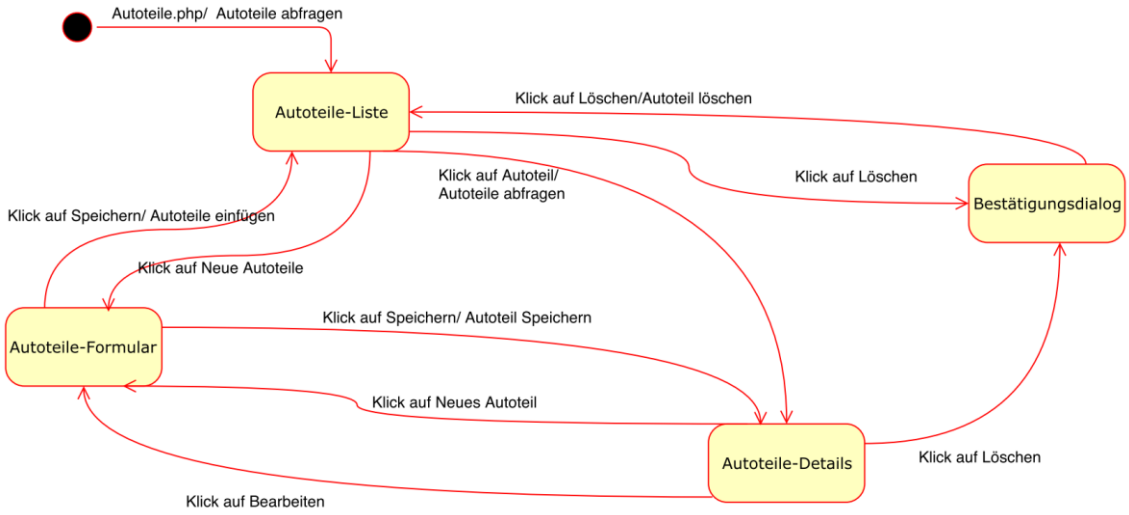
4.1. Allgemeines

UML-Zustandsdiagramme beschreiben alle beteiligten Funktionen mit ihren entsprechenden Ausführungen, Methoden und ihren Beziehungen zu anderen Funktionen.
Die Unified Modeling Language (UML) ist eine grafische Modelliersprache und unterstützt die Kommunikation zwischen den Beteiligten eines Projektes. Das UML-Zustandsdiagramm bildet die statische Struktur des Projekts in grafischer Form ab.

In der UML-Notation werden Klassen oder ein Zustand als Rechtecke mit abgerunderten Ecken angegeben. Befindet sich ein Objekt in einem Zustand, so können alle sogenannten inneren Aktivitäten auf diesem Objekt ausgeführt werden, die in diesem Zustand spezifiziert sind.

Das UML-Diagramm beschreibt die Beziehungen zwischen den Zuständen. Assoziationen zwischen Klassen werden mit einer durchgezogenen Linie plus Beziehungsname und Pfeil für die Leserichtung dargestellt. In unserem UML-Diagramm haben wir die Funktionen unseres Programmes in den Rechtecken dargestellt, die Pfeile entsprechend der Ausführung nach Klick auf eine bestimmte Funktion ausgerichtet.

4.2. Unser UML-Zustandsdiagramm „Autoteile NEAP“



5. Objektorientierte Programmierung

5.1. Allgemein

5.2. Programmierung „Autoteile NEAP“

Die Stuktur der Webanwendung sieht vor, dass der HTML-Code mithilfe von PHP, CSS, Javascript und einer SQL-Datenbank so ergänzt wird, dass die notwendigen Informationen dynamisch dargestellt werden können. Ajax ermöglicht es, während eine HTML-Seite angezeigt wird http-Anfragen auszuführen und die Seite zu verändern, ohne dass diese bei jeder Anfrage neu geladen wird. Bei der reinen Nutzung von http ist sind solche One-Page-Anwendungen nicht möglich, da die einzelnen Interaktionen ein erneutes Laden einer Seite zur Folge haben.

Bei der Umsetzung wurde auf eine Aufteilung zwischen Mobile-Client, Web-Client und Web-Service vorgenommen. So ist ein strukturiertes Arbeiten an den einzelnen Modulen möglich und eine technisch-thematische Aufteilung ist sichtbar. Die beiden Clients sind

unabhängig voneinander. Sie rufen über jQuery[2] und AJAX[3] die benötigten Daten vom Web-Service und darüber aus der Datenbank ab. So wird das Stellen von HTTP-Anfragen ermöglicht, während eine HTML-Seite angezeigt wird. Ohne die gesamte Seite neu zu laden werden Inhalte geändert.

Die einzelnen Autoteile und ihre Eigenschaften werden in einer SQL-Datenbank gespeichert. Diese beinhaltet für diesen Anwendungsfall eine Tabelle, die neben den in Kapitel 3.2 genannten Eigenschaften der Autoteile auch eine ID enthält, die eine eindeutige Unterscheidung der Teile ermöglicht. Außerdem findet sich in der Tabelle eine Spalte mit Versionsnummern, die sicherstellen sollen, dass immer der aktuellste Stand gespeichert wird. So wird vor dem endgültigen Bearbeiten eines Datensatzes geprüft, ob dieser in der Zwischenzeit eine neue Versionsnummer in der Datenbank erhalten hat und der Nutzer wird entsprechend informiert. So soll verhindert werden, dass sich Änderungen überschneiden und möglicherweise Informationen verloren gehen.

Die Zugriffskontrolle auf die Inventarisierungs-Plattform wird durch entsprechende Konfiguration in den Dateien .htaccess und .htusers realisiert. Diese sind Teil des WebServices, da sie sowohl für den Web- als auch den Mobilen Client gelten sollen.

In der .htaccess-Datei wird über die Zeile

```
AuthUserFile "c:\xampp\htdocs\inventur\WebService\.htusers"
```

auf die Datei verwiesen, die die autorisierten Nutzerkonten enthält.

5.2.1. Web-Client

Der Web-Client beinhaltet die Darstellung der Anwendung auf einem Desktop-Browser.

Vom Anwender wird über den Browser die Datei index.html aufgerufen, über die alle anderen Funktionen und Eigenschaften der Software auslösbar sind. So werden hier die

---

[2] Java-Script-Bibliothek

[3] Asynchronous JavaScript and XML

benötigten Java-Script-Objekte und die Optimierung der graphischen Darstellung durch CSS eingebunden. Dies findet im Header der Datei statt, der allgemeine Informationen und nicht einzelne Markup-Elemente enthält.

```
<head>
	<meta charset="utf-8"/>
	<title>Autoteile</title>
	<link rel="stylesheet" href="./css/jquery-ui-1.10.4.css" />
	<link rel="stylesheet" href="./css/application.css" />
	<script src="./javascript/jquery-1.10.2.js"></script>
	<script src="./javascript/jquery-ui-1.10.4.js"></script>
	<script src="./javascript/jquery.blockUI.js"></script>
	<script src="./javascript/jquery.ui.datepicker-de.js"></script>
	<script src="./javascript/autoteil.liste.js"></script>
	<script src="./javascript/autoteil.details.js"></script>
	<script src="./javascript/autoteil.errordialog.js"></script>
	<script src="./javascript/autoteil.deletedialog.js"></script>
	<script src="./javascript/autoteil.editdialog.js"></script>
	<script src="./javascript/autoteil.createdialog.js"></script>
	<script src="./javascript/autoteil.menubar.js"></script>
	<script src="./javascript/application.js"></script>
</head>
```

Im weiteren Teil der index.html werden sämtliche Objekte angelegt, die die Webseite potentiell anzeigen können soll. Dies beinhaltet sowohl die Auflistung der Autoteile mit den gewünschten Attributen, die Detailanzeige eines Autoteils mit allen sichtbaren Attributen sowie die Dialoge zum Löschen, Bearbeiten und Neuanlegen von Inventur-Einträgen. Auch eventuell aufkommende Fehlermeldungen werden hier abgefangen.

Die darin bestehenden Objekte – Textfelder sowie Schaltflächen, Label etc. – werden mithilfe einer im Header eingebundenen CSS-Datei formatiert. Außerdem definiert die CSS-Datei bestimmte für den gesamten HTML-Body geltenden Eigenschaften wie beispielsweise die Schriftart. Sie wurden hierfür Klassen oder IDs zugewiesen, um sie eindeutig mit den entsprechenden Bereichen der CSS-Datei zu verknüpfen. Der Unterschied zwischen Klassen und IDs besteht darin, dass eine ID nur einmalig verwendet werden darf und somit ein Objekt eindeutig kennzeichnet.

Die Modifikation der Daten innerhalb des HTML-Dokuments werden durch jQuery modifiziert.

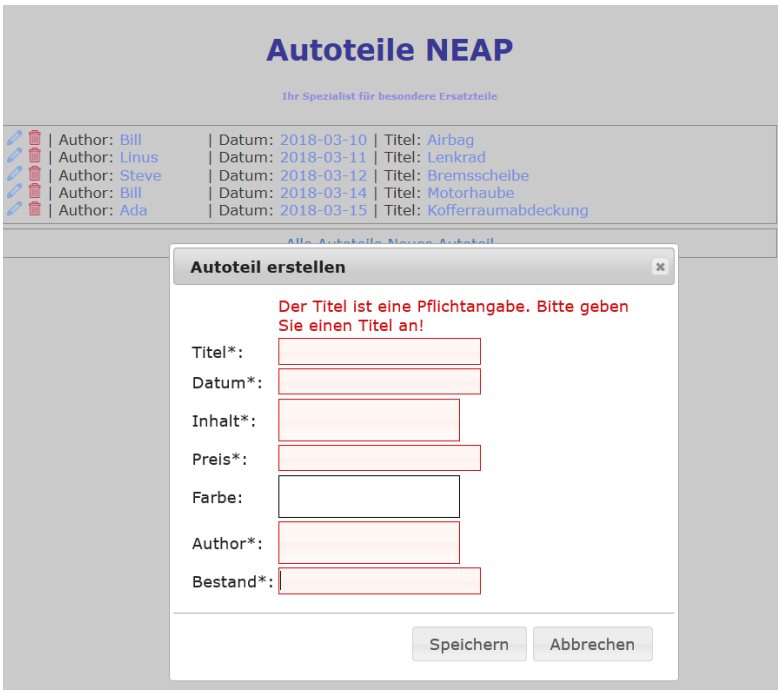
5.2.2. Web-Service

Der Web-Service übernimmt die Verwaltung der Daten sowohl für den Web-Client als auch für den Mobile-Client. Zentraler Punkt ist hier der RequestHandler, der aus den

Klassen die URLs für die http-Anfragen an den Server erstellt und durchführt. Es werden spezifische URLs generiert, die durch die unterschiedlichen Arten der http-Anfragen über den AutoteileService die Befehle zum Auslesen, neu Anlegen oder Ändern der Autoteile ausführt. Diese Klasse AutoteileService ist auch diejenige, die mit der SQL-Datenbank kommuniziert. Über eine Datenbank-Verbindung werden SQL-Statements ausgeführt und das Ergebnis dann als JSON-Objekt in den Response-Body eingefügt. Von dort kann der Client die Daten auslesen und im HTML-Dokument darstellen.

5.2.3. Funktion „Neue Autoteile hinzufügen“

Die Funktion, eine neues Autoteil hinzuzufügen wird aus der Übersicht der Autoteile heraus per Klick auf die Schaltfläche „Neues Autoteil“ ausgelöst. Hierbei wird zum einen das Formular geöffnet, um dem Anwender die Eingabe zu ermöglichen. Gleichzeitig muss beachtet werden, dass die auszufüllenden Felder nicht leer sind und der Anwender, falls nötig, darauf hingewiesen wird, die Pflichtfelder mit Inhalt versehen. Die Felder werden über die Klasse „ui-state-error“ rot hinterlegt. Diese Klasse wird von den Elementen entfernt, wenn sie mit Inhalt versehen sind.



Beim Klicken auf „Speichern“ wird eine POST-Anfrage an den Web-Service gesendet, der über den AutoTeile-Service dann das SQL-Statement zum Einfügen der Daten ausführt.

5.2.4. Funktion „Bearbeiten“

Beim Bearbeiten eines bestehenden Teils wird dieses über seine ID identifiziert und aus der Datenbank abgerufen, indem die URL für den http-GET-Befehl entsprechend vom WebService angepasst wird. Die dort bestehenden Inhalte werden vorgegeben, der Nutzer kann diese ändern. Es gelten dieselben Prüfbedingungen wie bei der Neuanlage, sodass Pflichtfelder nicht leer bleiben dürfen.

5.2.5. Funktion „Löschen“

Beim Löschen eines inventarisierten Autoteils wird die URL mit dem http-DELETE-Befehl um die ID des gewählten Teils ergänzt. Dieser bewirkt, dass als Sicherheitsabfrage eine Bestätigung des Users angefordert wird. Erst nach einem Klick auf „OK“ wird vom WebService das entsprechende SQL-Statement abgegeben; bei einem Klick auf „Abbrechen“ und zum Unterbrechen des Lösch-Vorgangs wird der Anwender auf die Liste aller Autoteile zurückgeführt.

5.2.6. Funktion „Autoteile anzeigen“

Die Anforderung, alle Objekte aufgelistet anzuzeigen wird damit umgesetzt, dass der RequestHandler die URL nicht wie beim einzelnen Autoteil um eine ID ergänzt, sondern sie nach „/autoteile“ beendet. Die Klasse AutoteileService führt damit die Methode readautoteile aus, was in der SQL-Datenbank eine Selektion und Ausgabe aller Objekte zur Folge hat.

Diese werden in der index.html – Datei als Tabelle formatiert angezeigt; für jeden Eintrag wird eine neue Zeile generiert. Zusätlich sind an dieser Stelle für jede Zeile, also für jedes Autoteil, die Buttons zum Löschen und Bearbeiten implementiert.

5.2.7. Programmbeschreibung der Mobile-App

Für die Umsetzung der mobilen Anwendung wurde auf das Sencha Touch – Framework zurückgegriffen. Dieses Java-Script-Framework eignet sich zur Erstellung von Web-Anwendungen, sich nutzerfreundlich auf mobilen Endgeräten benutzen lassen und sich ähnlich zu nativ installierten Apps verhalten. Dabei wird zum Beispiel auf die Bedienung über Touch-Oberflächen besonders Rücksicht genommen.

6. Fazit

Das Projekt wurde abgeschlossen, allerdings wurden einzelne Funktionen nicht gänzlich erfüllt. Sämtliche Bearbeitungsschritte funktionieren zwar, allerdings ließen sich einzelne, nicht die Basis-Funktionalität betreffende Dinge nicht umsetzen. Ein Beispiel hierfür ist, dass sich das DatePicker-Field öffnet, wenn beim Speichern eines bearbeiteten oder neuen Autoteils ein Pflichtfeld noch nicht ausgefüllt ist.

Durch die vorhergehende Planung der Vorgehensweise und strukturiertes Handeln nach einem Zeitplan konnten zeitliche Engpässe vermieden werden. Zudem waren die Fähigkeiten jedes Gruppenmitglieds gefordert, sodass jeder Anteil am Erfolg des Projektes hatte. Andererseits wurden Kompetenzen gefördert, da die einzelnen Arbeitsschritte im gegenseitigen Austausch innerhalb der Gruppe stattfanden. Damit wurde auch Wissen ausgetauscht und erweitert.

7. Eigenständigkeitserklärung

Hiermit erklären wir, dass wir die vorliegende Arbeit selbständig und ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfs-mittel angefertigt haben.

Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Dies gilt auch für Quellen aus eigenen Arbeiten.

Wir versichern, dass wir diese Arbeit oder nicht zitierte Teile daraus vorher nicht in einem anderen Prüfungsverfahren eingereicht haben.

Uns ist bekannt, dass unsere Arbeit zum Zwecke eines Plagiatsabgleichs mittels einer Plagiatserkennungssoftware auf ungekennzeichnete Übernahme von fremdem geistigem Eigentum überprüft werden kann.

Wir versichern, dass die elektronische Form unserer Arbeit mit der gedruckten Version identisch ist.

Patrick Bartschewski
Andreas van Bonn
Emil Tomaszewski
Nadine Hegel