**Verwendung von MVC**

Der Ansatz der Anwendung entspricht dem MVC Prinzip, welches ein Architekturprinzip aus Model, View und Controller.

Das Model einer MVC-Anwendung dient zur Speicherung von Teilen des aktuellen Zustands der Anwendung. Es übernimmt Aufgaben wie die Gewährung des Zugriffs auf Zustandsdaten, die Information anderer Module über Änderungen, die Umsetzung der Komponentenlogik und Kommunikation mit externen Datenquellen für die Datensynchronisation.

Eine View kann bei Daten-Änderungen in den zugehörigen Modellen der Anwendung die View-Repräsentation dieser Daten automatisch anpassen, Benutzeraktionen, die über grafische Eingabeelemente erfolgen, an einen Controller weiterleiten und Status- und Fehlermeldungen von Controllern visualisieren. Views sind für die aktuelle Darstellung der Eingangs- und Ausgangsdaten in den jeweiligen Anzeigenobjekten verantwortlich. Durch ein Model können mehrere Views visualisiert werden, welche unterschiedliche Sichtweisen des Models darstellen.

Der Controller dient der Steuerung einer MVC-Anwendung. Dazu nimmt er Eingaben, wie beispielsweise Daten, die ein Benutzer über eine beliebige Benutzer-Schnittstelle wie eine Tastatur oder eine Maus eingibt, entgegen und an ein Model weiter. Der Controller hingegen ist verantwortlich für die Steuerung der Anwendung durch den Benutzer. Er überwacht alle Eingaben, wertet die Eingabedaten aus und leitet sie weiter. Änderungen der Modelldaten werden also vom Controller eingeleitet. Der Controller und die View bilden zusammen die Benutzungsoberfläche.



Abbildung MVC-Architektur

Die ControllerDateien unserer Anwendungen befinden sich im Ordner „core“ und beinhalten Methoden wie „addProduct.php“. Die Views sind die HTML-Dateien. Sie bestehen aus HTML-Code und JavaScript Coding. In der „app.js“ sind alle AJAX-Funktionen implementiert. Die HTML-Datei ist sehr einfach gehalten und eher „passiv“, da durch das JavaScript die Inhalte wie die Produktliste zur View hinzugefügt werden. Passiv daher, weil das JavaScript beim Laden der Produktliste einen erheblichen Teil beim Manipulieren des DOMs übernimmt. Das Model die Datenbank. Preise und Produkte werden wie die anderen Daten in der Datenbank abgelegt.

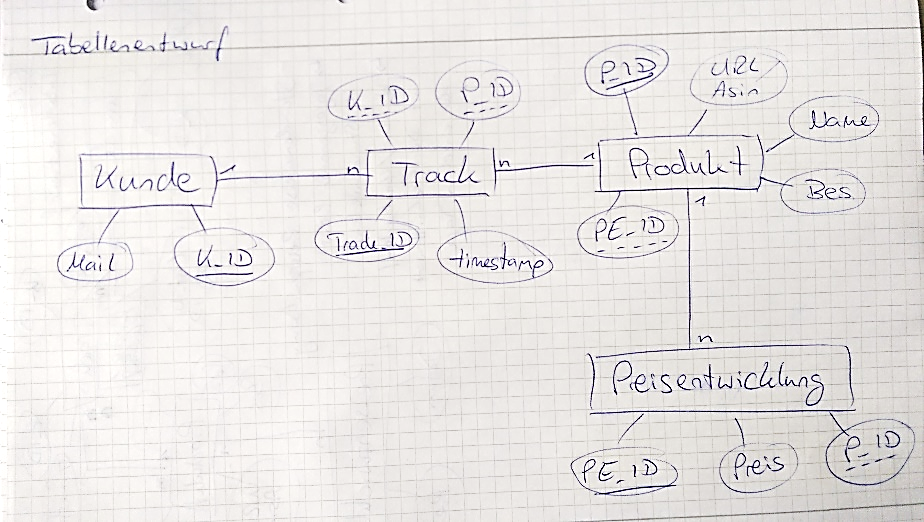


Abbildung - Tabellenentwurf der Datenbank

Der Benutzer kann sich mit seiner E-Mail-Adresse anmelden und sieht dann die unter seinem Benutzer individuelle Produktliste. Der Liste kann er Produkte hinzufügen, diese favorisieren und löschen. Weiter muss er bei Erstellung des Produkts in der Liste einen Preis angeben, bei dem er eine Benachrichtigung bei Unterschreitung seines Wunschpreises eine E-Mail erhalten will.

Die PHP-Dateien im Ordner „core“ sind der Controller, welcher für die oben beschriebene Benutzerinteraktion zuständig ist.