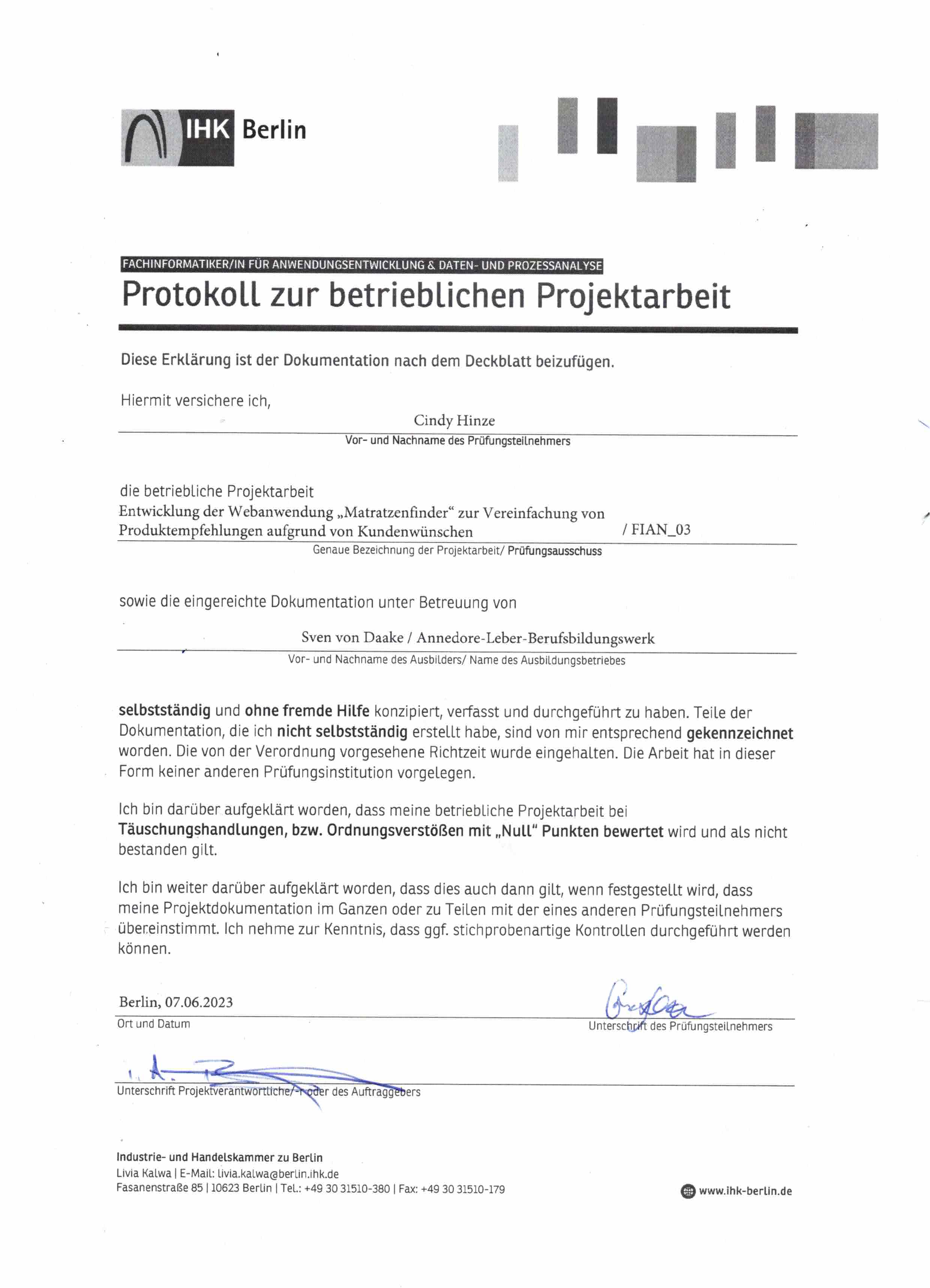
7. Juni 2023

Cindy Hinze

Prüfungsausschuss: FIAN\_03

Entwicklung der webanwendung „Matratzenfinder“

zur Vereinfachung von Produktempfehlungen aufgrund von Kundenwünschen



Inhalt

[1. Einleitung 1](#_Toc136951566)

[1.1. Projektumfeld 1](#_Toc136951567)

[1.2. Projektziel 1](#_Toc136951568)

[1.3. Projektbegründung 1](#_Toc136951569)

[1.4. Projektschnittstellen 1](#_Toc136951570)

[1.5. Projektabgrenzung 2](#_Toc136951571)

[2. Projektplanung 2](#_Toc136951572)

[2.1. Projektphasen 2](#_Toc136951573)

[2.2. Abweichungen von dem Projektantrag 2](#_Toc136951574)

[2.3. Ressourcenplanung 2](#_Toc136951575)

[2.4. Entwicklungsprozess 3](#_Toc136951576)

[3. Analysephase 3](#_Toc136951577)

[3.1. IST-Analyse 3](#_Toc136951578)

[3.2. Wirtschaftlichkeitsprüfung 3](#_Toc136951579)

[3.2.1. Projektkosten 3](#_Toc136951580)

[3.2.2. „Make or Buy“-Analyse 4](#_Toc136951581)

[4. Entwurfsphase 4](#_Toc136951582)

[4.1. SOLL-Konzept 4](#_Toc136951583)

[4.2. Benutzeroberfläche 5](#_Toc136951584)

[4.3. Datenbankmodell 5](#_Toc136951585)

[5. Implementierungsphase 5](#_Toc136951586)

[5.1. Einrichten der Datenbank 5](#_Toc136951587)

[5.2. Implementieren der Oberfläche 6](#_Toc136951588)

[5.3. Implementierung der Datenverarbeitung 6](#_Toc136951589)

[5.3.1. Implementieren des Frontends 6](#_Toc136951590)

[5.3.2. Implementieren des Backends 8](#_Toc136951591)

[6. Testphase 8](#_Toc136951592)

[6.1. Blackbox-Test 8](#_Toc136951593)

[7. Abnahmephase 9](#_Toc136951594)

[7.1. Einrichten der Anwendung 9](#_Toc136951595)

[7.2. Abnahme durch den Geschäftsleiter 9](#_Toc136951596)

[8. Fazit 9](#_Toc136951597)

[8.1. SOLL-/IST-Vergleich 9](#_Toc136951598)

[8.2. Lessons Learned 9](#_Toc136951599)

[8.3. Ausblick 10](#_Toc136951600)

[Glossar 11](#_Toc136951601)

[Anhang 12](#_Toc136951602)

[A.1. grobe Zeitplanung 12](#_Toc136951603)

[A.2. Projektkosten 12](#_Toc136951604)

[A.3. Ablaufplan 12](#_Toc136951605)

[A.4. Design Mock-Up 13](#_Toc136951606)

[A.5. Datenbankmodell 13](#_Toc136951607)

[A.6. SQL – Tabelle erstellen 14](#_Toc136951608)

[A.7. Einfügen der Datensätze 14](#_Toc136951609)

[A.8. HTML – Abschnitt Ausschnitt 14](#_Toc136951610)

[A.9. schowTab()-Funktion 15](#_Toc136951611)

[A.10. fixStepIndicator()-Funktion 15](#_Toc136951612)

[A.11. validateForm()-Funktion 16](#_Toc136951613)

[A.12. nextPrev()-Funktion Ausschnitt 16](#_Toc136951614)

[A.13. formSubmit()-Funktion Ausschnitt 17](#_Toc136951615)

[A.14. SQL-Abfrage 17](#_Toc136951616)

[A.15. Blackbox-Test Testfälle 18](#_Toc136951617)

[A.16. SOLL-/IST-Vergleich 18](#_Toc136951618)

Hinweis:

ausgegraute Wörter können im Abkürzungsverzeichnis nachgeschlagen werden.

# Einleitung

## Projektumfeld

Mein Projekt fand in der Betten- und Matratzenfirma „BeLaMa“ statt. Die Firma besitzt eine Filiale in Berlin Treptow-Köpenick mit fünf Festangestellten. Hier wird zu Betten, Matratzen und Lattenroste beraten. Die Beratung findet dabei vor Ort oder über Telefon statt. Zudem betreibt die Firma einen Onlineshop, welches auf der JTL-Software basiert. Neben dem Onlineshop existieren ein Schlafblog und ein Schlaflexikon, welche Kunden online zu passenden Themen informieren sollen. Im Gegensatz zum Onlineshop werden der Blog und das Lexikon über WordPress betrieben. Mein Platz in dieser Filiale befand sich als Praktikantin im technischen Bereich, welcher sich um die jeweiligen Webseiten kümmert.

## Projektziel

In der Firma wird überwiegend zu Matratzen beraten, wodurch ein Online-Ratgeber, der sogenannte „Matratzenfinder“, entwickelt werden sollte. Diese Webapplikation sollte aus Kompatibilitätsgründen ein Plugin für WordPress werden und in den Schlafblog integriert werden.

Hierzu entwarf ich eine Applikation, welche aus mehreren HTML-Formularelementen besteht. Dieses HTML-Formular beinhaltet auswählbare Elemente zu bestimmten Anforderungen, wie zum Beispiel die Größe der Matratze. Nachdem alles ausgewählt wurde bekommt der Kunde bis zu vier passende Produktempfehlungen ausgegeben, welche dann zum Onlineshop verlinkt sind. Die Produkte, die ausgegeben werden können, befinden sich in einer eigenen Datenbank. Diese Datenbank habe ich im Rahmen des Projektes erstellt und mit den entsprechenden Daten befüllt.

Das Design habe ich mit CSS umgesetzt und optisch an das bestehende Blog- und Shop-Design angepasst.

## Projektbegründung

In der Filiale gibt es momentan nur drei Verkäufer bzw. Berater, worunter sich der Geschäftsleiter selbst ebenfalls befindet. Da die Beratung vor Ort meist umfangreicher ausfällt, ist es nicht immer möglich alle Kunden per Telefon beraten zu können. Mithilfe des „Matratzenfinders“ wird auf Entlastung der Mitarbeiter gehofft. Außerdem soll die Kundenzufriedenheit damit gesteigert werden, da diese nun nicht mehr auf das Telefon oder die Anfahrt in der Filiale angewiesen sein werden.

## Projektschnittstellen

Das Projekt wurde als Plugin für WordPress erstellt, worauf Kunden oder andere Internet-Nutzer, in einem Browser ihrer Wahl, über den Schlafblog der Firma zugreifen können.

Im Laufe der Entwurfsphase befand ich mich in Kommunikation mit dem Geschäftsleiter und einem aktuellen Mitarbeiter des Social-Media-Marketings. Gemeinsam entwickelten wir verschiedene Mock-Ups zum Design der Anwendung.

Nach der Implementierungsphase wurde die Anwendung als Plugin in WordPress integriert. Hierzu blieb ich in Kontakt mit einem externen Mitarbeiter, welcher zudem die ganzen Systeme der Firma administriert und dementsprechend die Integrierung des Projektes als Plugin übernahm. Nach der erfolgreichen Einrichtung wurde das fertige Projekt dem Geschäftsleiter vorgestellt.

## Projektabgrenzung

Die Integrierung des Projektes als Plugin war kein Bestandteil meines Abschlussprojekts. Diese übernahm der externe Mitarbeiter, wie in den Projektschnittstellen erwähnt. Ich war für die lokale Erstellung der Anwendung zuständig. Hier war es meine Aufgabe, das Design und eine dazugehörige Datenbank fertigzustellen, sowie alle Funktionalitäten zu gewährleisten.

# Projektplanung

## Projektphasen

Das Projekt habe ich in 80 Stunden durchgeführt. Eine grobe Zeitplanung befindet sich im Anhang [A.1. grobe Zeitplanung](#_A.1._grobe_Zeitplanung).

## Abweichungen von dem Projektantrag

Abweichungen haben sich bei dem Zeitplan ergeben. Im Laufe der Entwurfsphase ist mir aufgefallen, dass das gewählte Vorgehensmodell sich nicht in dem Zeitplan wiederspiegelte, welcher auch im Projektantrag hinterlegt war. Daraufhin wurde der ursprüngliche Zeitplan um die Testphase ergänzt und die Zeiten wurden dementsprechend angepasst.

Außerdem wurden keine Integrationstests durchgeführt, da die Zeit für die Lösung der Ergebnisse im Blackbox-Test benötigt wurde. Näheres zu den Zeitlichen Änderungen und den Anpassungen befinden sich im 8.Kapitel unter „SOLL-/IST-Vergleich“.

Eine kleinere Abweichung findet sich in zudem der Ist-Analyse des Projektantrags wieder. Hier wurde erwähnt, dass auch über E-Mail beraten wird. Allerdings finden keine Beratungen über E-Mail statt. Außerdem stimmt die Bezeichnung „Filialen-Seite“ nicht. Hierbei handelt es sich eher um den Schlagblog.

## Ressourcenplanung

Zur Durchführung des Projektes wurde ein Arbeitscomputer mit Windows 11 von der Firma gestellt und von mir genutzt. Als Programmiersoftware habe ich Visual Studio Code benutzt und diese mit meinem GitHub Repository, zur Versionsverwaltung, verbunden. Bis zur Integrierung der Anwendung in WordPress lief die Applikation lokal über XAMPP, welches den Apache Webserver, sowie phpMyAdmin beinhaltet.

Die Anwendung wurde als Plugin in WordPress integriert, was bereits zur Verfügung stand. WordPress läuft auf einem Apache Webserver und das Datenbanksystem von WordPress basiert auf phpMyAdmin. Für die Dateiübertragung an den Webserver wurde FileZilla genutzt.

Zur Verfassung der Projektdokumentation wurde Microsoft Word benötigt, worauf ich bereits durch die Berufsschule Zugriff hatte.

Zudem benötigte ich eine Liste mit allen Artikeln und den dazugehörigen Daten, welche für den „Matratzenfinder“ genutzt werden sollten, um diese dann in die Datenbank einpflegen zu können. Diese Liste erhielt ich vom Geschäftsleiter.

## Entwicklungsprozess

Als Vorgehensmodell habe ich mich für das erweiterte Wasserfallmodell entschieden, da die Bestandteile des Projekts bereits zu Beginn klar waren und somit im Voraus definierbar waren. Außerdem bot es mir die Möglichkeit eine Phase zurückgehen zu können, falls z.B. in der Testphase Fehler auftreten, die in der Implementierung verbessert werden müssen.

# Analysephase

## IST-Analyse

Wer sich zu Betten oder Matratzen beraten lassen möchte, kann sich bei BeLaMa an einen der drei verfügbaren Berater wenden. Die Beratung kann sowohl über Telefon, als auch persönlich vor Ort stattfinden. Hierbei gibt es Fragen, die bei jeder Beratung wichtig sind und dementsprechend bei jeder vorkommen. Aus diesen Fragen können sich bereits gewisse Produktempfehlungen erschließen, die dem Kunden dann vorgeschlagen werden.

Wenn im Rahmen der persönlichen Beratung vor Ort ein vorgeschlagener Artikel gekauft wird, werden die Personalien, wie Name und Adresse, des Kunden aufgenommen und eine Anzahlung erfolgt ebenfalls. Mit den nun vorhandenen Daten wird der gekaufte Artikel durch den Mitarbeiter beim Händler bestellt und die Anzahlung, sowie der Restbetrag werden erfasst.

Bei einer Beratung über Telefon verläuft die weitere Verkaufsabwicklung über den E-Mailverkehr.

## Wirtschaftlichkeitsprüfung

### Projektkosten

Für verwendete Software fallen für dieses Projekt keine Kosten an, da es sich entweder um frei nutzbare Software handelte (z.B. phpMyAdmin) oder diese bereits vor dem Projekt schon existierte und dementsprechend schon als bezahlt gilt (z.B. WordPress). Die zur Verfügung gestellte Hardware wurde ebenfalls vor dem Projekt bezahlt. Zur Ermittlung der Projektkosten gab es dementsprechend nur noch die Personalkosten. Aufgrund meines externen Ausbildungsverhältnisses wird bei mir kein Stundenlohn berechnet, sondern ein pauschaler Lohn für Arbeitsplatz, sowie Internet- und Stromkosten, welcher mit 5€ pro Stunde berechnet wird. Bei dem Mitarbeiter aus dem Social-Media-Marketing handelt es sich um einen Dual-Studenten, dessen Stundenlohn 5€ beträgt. Für den externen Mitarbeiter werden keine Kosten berechnet. Die Projektkosten belaufen sich daher auf 415€. Eine tabellarische Darstellung der Berechnung und den Kostenbetrag befindet sich im Anhang [A.2. Projektkosten](#_A.2._Projektkosten).

### „Make or Buy“-Analyse

Es gibt Anbieter, welche solche Plugins für WordPress anbieten und die andere Betten- und Matratzenverkäufer nutzen. Um diese in WordPress nutzen zu können, müsste der Onlineshop auf WordPress basieren und eine solche Umstellung wäre zu kosten- und zeitaufwendig. In diesem Punkt wäre die eigene Erstellung die bessere Lösung.

Es gibt allerdings auch Plugins für JTL, worauf der Onlineshop usw. basiert. Das Projekt betrug einmalige Kosten in Höhe von 415€. Ein Plugin, welches ich mir zum Vergleich herausgesucht habe, würde einmalig 75€ kosten. Zu dem Plugin gibt es ein Update-Abo, welches jährlich für 35€ verlängert werden kann, damit das Plugin immer auf dem neuesten Stand bleibt. In den einmaligen 75€ ist ein 12-monatiges Abo bereits enthalten. Wenn man das Update-Abo jährlich um 35€ verlängert, ergibt sich eine Armotisationszeit von ca. 10 Jahre. Aus finanzieller Sicht betrachtet würde sich das Plugin mit dem jährlichen Betrag von 35€ (Nach der ersten Zahlung von 75€) mehr lohnen.

Das zum Vergleich gewählte Plugin bietet die Möglichkeit, eine detailliertere „Merkmalsuche“ nutzen zu können. Hier ist kein Design enthalten und jegliche Konfiguration, wie z.B. Merkmale festlegen, müssen ebenfalls selbst übernommen werden. Installation und Einrichtung sind auch dem Anwender überlassen. Es sei denn, man zahlt zu Beginn weitere 75€ für die Installation des Plugins.

Fazit: Die Firma wäre in der Lage, die Installation, die Konfiguration und das Design zu übernehmen. Daher würden nur die einmaligen Kosten von 75€ und die darauffolgenden jährlichen Kosten von 35€ anfallen. Aus finanzieller Sicht gesehen wäre es daher eher eine „Buy“-Entscheidung. Allerdings ist die Individualität des Plugins unklar. Sollten die Möglichkeiten, das Plugin zu designen, zu Unzufriedenheit führen oder die Kompatibilität doch nicht gewährleistet werden können, wäre es eher eine „Make“-Entscheidung.

# Entwurfsphase

## SOLL-Konzept

Das Prinzip der Beratung sollte in Form einer Webseiten-Anwendung mit einer Multi-Step-Form umgesetzt werden. Hierbei werden dem Kunden nacheinander die wichtigsten Fragen gestellt, bei denen es sich unter anderem um die Matratzengröße, den Härtegrad und das Material handelt. Zu allen Fragen gibt es anklickbare Antworten, aus welchen eine ausgewählt wird und dann kann der Nutzer erst zur nächsten Frage weitergehen. Dem Anwender sollte es auch möglich sein zu den Fragen zurückgehen zu können, um eventuell etwas nachträglich zu ändern. Nachdem alle Fragen beantwortet wurden erfolgt der letzte Klick auf den „Fertig“-Button und maximal vier Artikel sollten erscheinen. Zu allen Produkten sollte ein Button erscheinen, welcher zur Onlineshop-Seite des Artikels verlinkt sein sollte. Außerdem sollte es auf der Ergebnis-Seite einen Button geben, welcher für den Kunden die Seite neu lädt, falls dieser den Prozess nochmal komplett von vorne beginnen möchte. Sofern es zu den ausgewählten Kriterien kein passendes Produkt gibt, sollte eine Ergebnislose Seite eingerichtet werden mit einem Button zum neu laden.

Um den Ablauf besser nachvollziehen zu können, unter anderem für die Benutzeroberfläche, wurde ein Ablaufplan erstellt, welcher im Anhang [A.3. Ablaufplan](#_A.3._Ablaufplan) zu finden ist.

## Benutzeroberfläche

Gemeinsam mit dem Geschäftsleiter und einem Mitarbeiter des Social-Media-Marketings habe ich Mock-Ups entworfen, um festzulegen, wie der „Matratzenfinder“ am Ende aussehen soll. Das Mock-Up, welches am Ende als Referenz diente, befindet sich im Anhang [A.4. Design Mock-Up](#_A.4._Design_Mock-Up). Die Basis bildete hier der Ablaufplan, da wir nun auf die einzelnen Schritte eingehen und entsprechende Antwortmöglichkeiten festlegen konnten. Ein Farbdesign wurde weder schriftlich noch im Mock-Up festgelegt. Diesbezüglich sollte ich mich an das Farbschema des Online Shops halten.

## Datenbankmodell

In der Entwurfsphase habe ich ein ER-Diagramm entworfen, um die Struktur der Datenbank festzulegen. Dieses ER-Diagramm finden Sie im Anhang unter [A.5. Datenbankmodell](#_A.5._Datenbankmodell). Hierzu habe ich den zuvor entworfenen Ablaufplan genutzt, um daraus die Tabellen generieren zu können. Welche Attribute und welche Datentypen die Tabellen benötigten, entnahm ich dem Mock-Up, welches ebenfalls zuvor erstellt wurde.

# Implementierungsphase

## Einrichten der Datenbank

Zu Beginn der Implementierungsphase richtete ich die Datenbank ein und befüllte sie mit Daten. Hierzu habe ich in phpMyAdmin eine Datenbank mit dem Namen „matratzenfinder“ erstellt. Als nächstes schrieb ich SQL-Befehle, um die einzelnen Tabellen zu generieren und ihnen dabei die erforderlichen Attribute und deren Datentypen zuzuweisen. Ein SQL-Befehl zum Erstellen einer Tabelle wurde als Beispiel dem Anhang [A.6. SQL – Tabelle erstellen](#_A.6._SQL_–) beigefügt. Da es beispielsweise eine Matratze in verschiedenen Größen geben kann oder eine Matratze besonders gut für mehrere Beschwerden sein kann, wurden sogenannte Kreuztabellen erstellt. PhpMyAdmin bietet hierzu die Möglichkeit, Fremdschlüssel zu erstellen, ohne SQL-Code schreiben zu müssen und diese Möglichkeit habe ich genutzt.

Als die Tabellen erstellt waren mussten die Daten eingefügt werden. Alle Daten, die ich benötigte, wurden in eine Word-Datei gespeichert. Ich habe aus diesen Daten alle INSERT-Befehle erfasst und damit alle Datensätze in den entsprechenden Tabellen eingefügt. Als Beispiel liegt ein Ausschnitt der INSERT-Befehle für die Kreuztabelle „size-mattress“ im Anhang [A.7. Einfügen der Datensätze](#_A.7._Einfügen_der) vor.

## Implementieren der Oberfläche

Für die Benutzeroberfläche habe ich mit HTML ein *form*-Element erstellt. In diesem *form* implementierte ich die verschiedenen Abschnitte als *div*-Elemente, welche die Fragen und Antworten enthalten, durch die sich der Anwender klicken kann. Jedes dieser *div*-Abschnitte beinhaltet *input*-Elemente des Typs „*radio*“, ausgenommen der Abschnitt für Allergien. Für *radio*-Eingaben habe ich mich entschieden, da es immer nur eine Antwortmöglichkeit geben kann und muss. Der Allergien-Abschnitt besitzt *checkbox*-Eingaben und hier besteht die Möglichkeit, mehrere Antworten auswählen zu können. Ein Ausschnitt des „Härtegrad“-Abschnittes wurde als Beispiel in dem Anhang [A.8. HTML – Abschnitt Ausschnitt](#_A.8._HTML_–) hinterlegt.

Die Eingabe-Elemente wurden überall, zusammen mit dem Text, in *label*-Elemente eingebunden. Das gab mir die Option, die Antwortmöglichkeiten individueller zu gestalten, z.B. farbige Kästchen draus zu machen. Außerdem ermöglichte es mir die *input*-Elemente zu verstecken, sodass nur noch die Kästchen sichtbar waren und trotzdem anklickbar bleiben, ohne ihre Funktion dabei zu verlieren. Nach dem letzten *div*-Abschnitt ist ein neues *div*-Element zu finden, welches zwei Buttons und *span*-Elemente beinhaltet. Die Buttons sind da, um zwischen den verschiedenen Schritten vor und zurück zu gehen. Die Funktion hierfür wurde mit JavaScript umgesetzt. Die *span*-Elemente habe ich so designt, dass sie die einzelnen Schritte anzeigen. Damit weiß der Anwender, in welchem der Abschnitte er sich befindet, und sieht auch, welche bereits erledigt wurden und wieviel noch kommt.

Jegliche Attribute wie „*name*“, „*data-value*“ und „*data-size*“ wurden für JavaScript benötigt. Sie ermöglichten die weitere Verarbeitung der ausgewählten Werte, welche über JavaScript und PHP erfolgt. „*oninput*“ und „*onclick*“ sind JavaScript-Events für die Abschnitte. Näheres hierzu ist im Kapitel „Implementieren des Frontends“ zu lesen.

Nach dem *form*-Element kommt ein letzter Abschnitt, in welchem die Ergebnisse angezeigt werden und der einen Button enthält, um das ganze neu laden zu können. Die Generierung der einzelnen Elemente für die entsprechenden Ergebnisse erfolgt über JavaScript. Sollte es kein Ergebnis geben, befindet sich im letzten Abschnitt eine „kein-Ergebnis“-Seite und ebenfalls der Button zum neu laden.

Das Design der Benutzeroberfläche habe ich zum Großteil mit CSS umgesetzt. Hier habe ich das Layout für das *form*-Element und die einzelnen Abschnitte als *Grid* eingestellt, um die einzelnen Elemente besser und dynamischer anordnen zu können. Zudem wurden die Antwort-Optionen als blaue Kästchen mit weißer Schrift entworfen. Wenn man über diese gleitet, werden die Kästchen heller. Das liegt an der Transparenz, die dabei runtergeht. Außerdem wurden mit JavaScript einzelne Optionen „*disabled*“ und die hellere Farbe, um zu kennzeichnen, dass diese Elemente nicht auswählbar sind, habe ich ebenfalls über CSS geregelt.

## Implementierung der Datenverarbeitung

### Implementieren des Frontends

Jeder *div*-Abschnitt, welcher für die einzelnen Schritte steht, erhielt die Klasse „*tab*“. In der JavaScript-Datei *functions.js* gibt es die erste Funktion *showTab()*. In dieser wird auf jedes HTML-Element mit der *tab*-Klasse zugegriffen und diese werden auch hier drüber als *grid* angezeigt. Die Funktion ist dafür da, damit der „Zurück“-Button im ersten Abschnitt nicht angezeigt wird. Sobald der zweite Abschnitt erreicht wird, ist der „Zurück“-Button zu sehen und nutzbar. In allen Abschnitten ist der zweite Button der „Weiter“-Button. In der *showTab()-*Funktion wird der Text im letzten Abschnitt von „Weiter“ zu „Fertig“ geändert, um zu signalisieren, dass dies der letzte Schritt ist. Der Code zur *showTab()*-Funktion befindet sich im Anhang [A.9. schowTab()-Funktion](#_A.9._schowTab()-Funktion).

Es wird ebenfalls auf die Funktion *fixStepIndicator()* hingewiesen, welche sich um die *span*-Elemente kümmert, die die aktuellen Schritte anzeigen. Bei jedem aktiven Abschnitt oder auch *tab*, wird dem zugehörigem *span*-Element die Klasse „*active*“ zugewiesen.

Wenn man sich also beim zweiten Abschnitt befindet, wird dem zweiten span-Element die *active*-Klasse hinzugefügt. Geht man in den dritten Abschnitt, wird die *active*-Klasse vom zweiten *span*-Element entfernt und durch „*finish*“ ersetzt. Die fixStepIndicator()-Funktion ist im Anhang unter [A.10. fixStepIndicator()-Funktion](#_A.10._fixStepIndicator()-Funktion) zu finden.

Für die „*finish*“-Klasse sorgt die *validateForm()*-Funktion. In dieser wird im aktuellen *tab* geschaut, ob mindestens ein input-Element ausgewählt wurde. Ist dies der Fall, bekommt das dazugehörige *span*-Element die Klasse „*finish*“. Sollte dies nicht der Fall sein, wird *valid* auf „*false*“ gesetzt. Ausnahme liegt hier im vierten Abschnitt, welcher für die Allergien ist. Da hier alles oder auch nichts ausgewählt werden kann, ist *valid* in diesem Abschnitt immer *true*. Im Anhang [A.11. validateForm()-Funktion](#_A.12._nextPrev()-Funktion_Ausschnit) finden Sie den Code zur eben beschriebenen Funktion. Diese ist wichtig, um zum nächsten Abschnitt gehen zu können.

Hier greift die Funktion *nextPrev*() ein. Sollte die *validateForm()*-Funktion ein *false*-Ergebnis liefern, wird *nextPrev()* nicht weiter ausgeführt. Im *true*-Fall wird der aktuelle Abschnitt versteckt und der Zähler für die *tabs* geht um eins hoch, um den nächsten Abschnitt anzuzeigen. In der *nextPrev()*-Funktion befinden sich zwei weitere *if()*-Anweisungen, welche auf die *data-size*-Attribute im HTML-Dokument zugreifen. Bei der ersten wird sichergestellt, dass die ersten beiden Matratzengrößen mit dem Attribut *data-size = 2* im dritten Abschnitt disabled werden, wenn im ersten Schritt „Erwachsene“ ausgewählt wurde. Ob im ersten Abschnitt „Erwachsene“ ausgewählt wurde, wird durch *querySelectorAll()* ermittelt, was sich auf die „*name = target“*-Attribute im HTML-Dokument bezieht. Die zweite *if()*-Anweisung funktioniert vom Prinzip wie die eben beschriebene. Sie *disabled* die Optionen mit dem „*data-size = latex“*-Attribut im fünften Schritt, wenn im vierten Abschnitt die *checkbox* mit dem „*data-value = 1“*-Attribut ausgewählt wurde. Somit werden alle Latex-Materialien disabled, wenn man angibt, eine Latex Allergie zu haben. Einen Ausschnitt des Codes der nextPrev()-Funktion wurde im Anhang [A.12. nextPrev()-Funktion Ausschnitt](#_A.12._nextPrev()-Funktion_Ausschnit) hinterlegt. Zum Schluss der Funktion wird geschaut, ob man sich im letzten Abschnitt befindet. Falls nicht, wird der korrekte Abschnitt angezeigt. Sollte es doch so sein, tritt die Funktion *formSubmit()* in Kraft.

*formSubmit()* enthält die ausgewählten Daten in Form eines *FormData-*Objektes. In diesem Objekt befinden sich die Ergebnis-Daten aus der *controller.php*-Datei, die dort mit SQL aus der Datenbank abgefragt und als *json* bereitgestellt, sowie von JavaScript dann abgefragt wurden. Steht das *json* bereit, wird das gesamte *HTML-form*-Element entfernt.

Sollte das *json* leer sein, weil es keine passenden Ergebnisse gibt, kommt man auf die „keine-Ergebnisse“-Seite. Ansonsten wird der Ergebnis-Abschnitt als *grid* angezeigt. Hier werden *div*-Elemente erzeugt, welche die Ergebnisse aus der Abfrage, die sich nun im *FormData-*Objekt befinden, enthalten. Angezeigt werden dabei der Name, das Bild, der Preis und der entsprechende Shop-Link zum jeweiligen Produkt. Einen Ausschnitt der Funktion formSubmit() finden Sie im Anhang [A.13. formSubmit()-Funktion Ausschnitt](#_A.13._formSubmit()-Funktion_Ausschn).

### Implementieren des Backends

In der *controller.php* wird zu Beginn geprüft, ob die Server-Request-Methode der *POST*-Methode entspricht. Wenn nicht, wird sofort abgebrochen und weder eine Datenbankverbindung kann aufgebaut werden, noch kann eine Datenverarbeitung stattfinden.

Des Weiteren wurden Variablen erstellt, in welche die einzelnen Datenpunkte aus dem *FormData*-Objekt, welches mit der *POST*-Request von der *functions.js* übergeben wurde, gespeichert werden. Dann wird eine Verbindung zur Datenbank hergestellt.

Als Letztes habe ich eine SQL-Abfrage erstellt, welche zum Ermitteln der Produkte aus der Datenbank zuständig ist. Hierbei werden die zuvor erstellten Variablen genutzt. Damit findet eine Datenbankabfrage statt und die abgefragten Daten werden mit der *fetch\_all()-*Funktion in ein *Array* gespeichert. Anschließend wird dieses Array in *json* umgewandelt und für die *functions.js* bereitgestellt. Die SQL-Abfrage befindet sich im Anhang unter [A.14. SQL-Abfrage](#_A.14._SQL-Abfrage).

# Testphase

## Blackbox-Test

Nach der Entwicklungsphase ging ich in die Testphase über. Hier habe ich im Rahmen von Blackbox-Tests verschiedene Mitarbeiter der Firma die Anwendung testen lassen. Dazu habe ich verschiedene Anwendungsfälle als Testfälle definiert, welche von den Mitarbeitenden ausgeführt werden sollten. Ich verfasste die verschiedenen Testfälle in einer Tabelle und notierte mir die erwarteten Ergebnisse, sowie die Ergebnisse, die tatsächlich rausgekommen sind.

Im ersten Durchlauf zeigte sich, dass die Abfrage, nach einem Ergebnis, ohne jegliche Allergien angegeben zu haben, nicht funktionierte und zu keiner weitere Ausführung des Codes führte. Daraufhin ging ich zurück zur Implementierungsphase und löste das Problem. Die *validateForm()-*Funktion hatte zu dem Zeitpunkt nichts was sie tun sollte, wenn der Fall eintritt, dass man sich im Allergie-Abschnitt befindet. Hierzu wurde dann die *if*-Funktion erstellt, die in genau dem Fall die valid-Variable auf *true* setzt, um zum nächsten Schritt weitergehen zu können. Außerdem zeigte sich ein Fehler im Backend bei der SQL-Abfrage mit dem Allergien-*Array*. Normalerweise sollte das *Array* nicht mit in die Abfrage genommen werden, wenn nichts ausgewählt wurde. Allerdings wurde das *Array* trotzdem mit einbezogen und das hat zu einem Fehler in der SQL-Syntax geführt. Dementsprechend musste die Funktion für dieses *Array* umgeschrieben werden, sodass das *Array* nicht in die Abfrage genommen wird, wenn es leer ist.

Danach erfolgte ein zweiter Durchlauf mit denselben Testfällen und die erwarteten Ergebnisse wurden erfüllt. Eine Tabelle mit den beiden Durchläufen und den Testfällen befindet sich im Anhang [A.15. Blackbox-Test Testfälle](#_A.15._Blackbox-Test_Testfälle).

# Abnahmephase

## Einrichten der Anwendung

Die Einrichtung der Anwendung erfolgte in Zusammenarbeit mit dem externen Mitarbeiter. Die Integrierung wurde von ihm übernommen, allerdings mussten ein paar Anpassungen vorgenommen werden. Beispielsweise musste das Design etwas an das verwendete WordPress-Theme angepasst werden. Dann habe ich eine Seite erstellt, welche das Plugin beinhalten sollte und zu welcher dann verlinkt werden kann.

## Abnahme durch den Geschäftsleiter

Nach der Fertigstellung der Anwendung wurde diese dem Geschäftsleiter vorgeführt und die grundlegende Funktionsweise erklärt. Danach war die erfolgreiche Abnahme durch den Geschäftsleiter abgeschlossen.

# Fazit

## SOLL-/IST-Vergleich

Wie in den Abweichungen des Projektantrags erwähnt, wurden die eingeplanten Zeiten etwas abgeändert, da die Testphase hinzugefügt wurde. Zu Beginn wurden sowohl für die Blackbox-Tests als auch die Integrationstests insgesamt 6 Stunden eingeplant. Von diesen 6 Stunden wurden 5 Stunden der Entwurfsphase abgezogen und die Abnahme durch den Geschäftsleiter um eine Stunde reduziert. Die Zeit wurde hauptsächlich in der Entwurfsphase gekürzt, da ich für die Datenbankentwürfe und die Mock-Ups weniger Zeit benötigte als ursprünglich geplant.

Nach den Blackbox-Tests wurde mehr Zeit für die Implementierung benötigt, wodurch die 4 Stunden für die Integrationszeit entfielen und auf die Implementierungen des Backend, sowie des Frontend aufgeteilt wurden.

Der SOLL-/IST-Vergleich ist als Tabelle im Anhang [A.16. SOLL-/IST-Vergleich](#_A.16._SOLL-/IST-Vergleich) zu finden.

## Lessons Learned

Das von mir verwendete Wasserfallmodell zur Durchführung des Projektes stellte sich grundsätzlich als gute Wahl heraus, da ich mit diesem Modell einen deutlichen, roten Faden hatte, dem ich gut folgen konnte. Allerdings hätte ich im Nachhinein mich bevorzugt für ein Wasserfallmodell mit agilen Elementen entschieden, um beispielsweise während der Implementierungsphase in Rücksprache mit dem Geschäftsleiter bleiben zu können, z.B. in Bezug auf das Design.

## Ausblick

Als Ausblick wurde bereits über die Möglichkeit eines eigenen CMS für diese Anwendung gesprochen. Dieses CMS soll das Einfügen oder Löschen von Daten für Nicht-Programmierer ermöglichen. Damit soll es möglich werden, dass z.B. der Geschäftsführer selbst Matratzen für den Matratzenfinder einfügen oder entfernen kann. Ob dafür ein eigenes CMS entworfen wird oder ob es andere Möglichkeiten, wie weitere Plugins, gibt, ist bisher unklar.

# Abkürzungsverzeichnis

JTL **J**anusch Lisson, **T**homas **L**isson, E-Commerce-Softwarehersteller

HTML **H**ypertext **M**arkup **L**anguage

CSS **C**ascading **S**tyle **S**heets

XAMPP **A**pache, **M**ySQL, **P**erl, **P**HP, das **X** bezieht sich auf verschiedene Betriebssysteme, die genutzt werden können

ER **E**ntity-**R**elationship

SQL **S**tructured **Q**uery **L**anguage

PHP Backronym für PHP:Hypertext Preprocessor

js **J**ava**S**cript

json **J**ava**S**cript **O**bject **N**otation

POST **p**ower-**o**n-**s**elf-**t**est

CMS **C**ontent **M**anagement **S**ystem

# Anhang

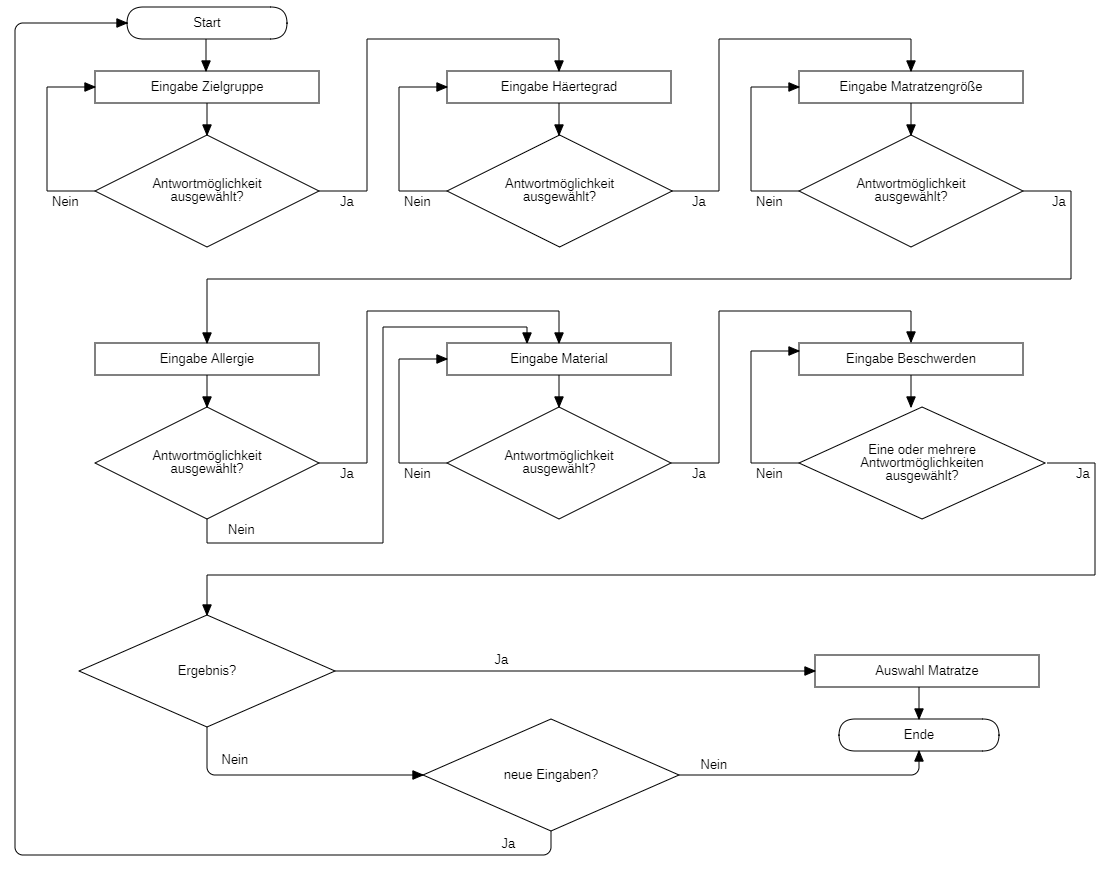
## A.1. grobe Zeitplanung

|  |  |
| --- | --- |
| **Phasen** | **Zeit in Stunden** |
| Analyse | 8 |
| Entwurf | 12 |
| Implementierung | 41 |
| Einrichtung und Abnahme | 8 |
| Dokumentation | 11 |
| **Gesamt** | **80** |

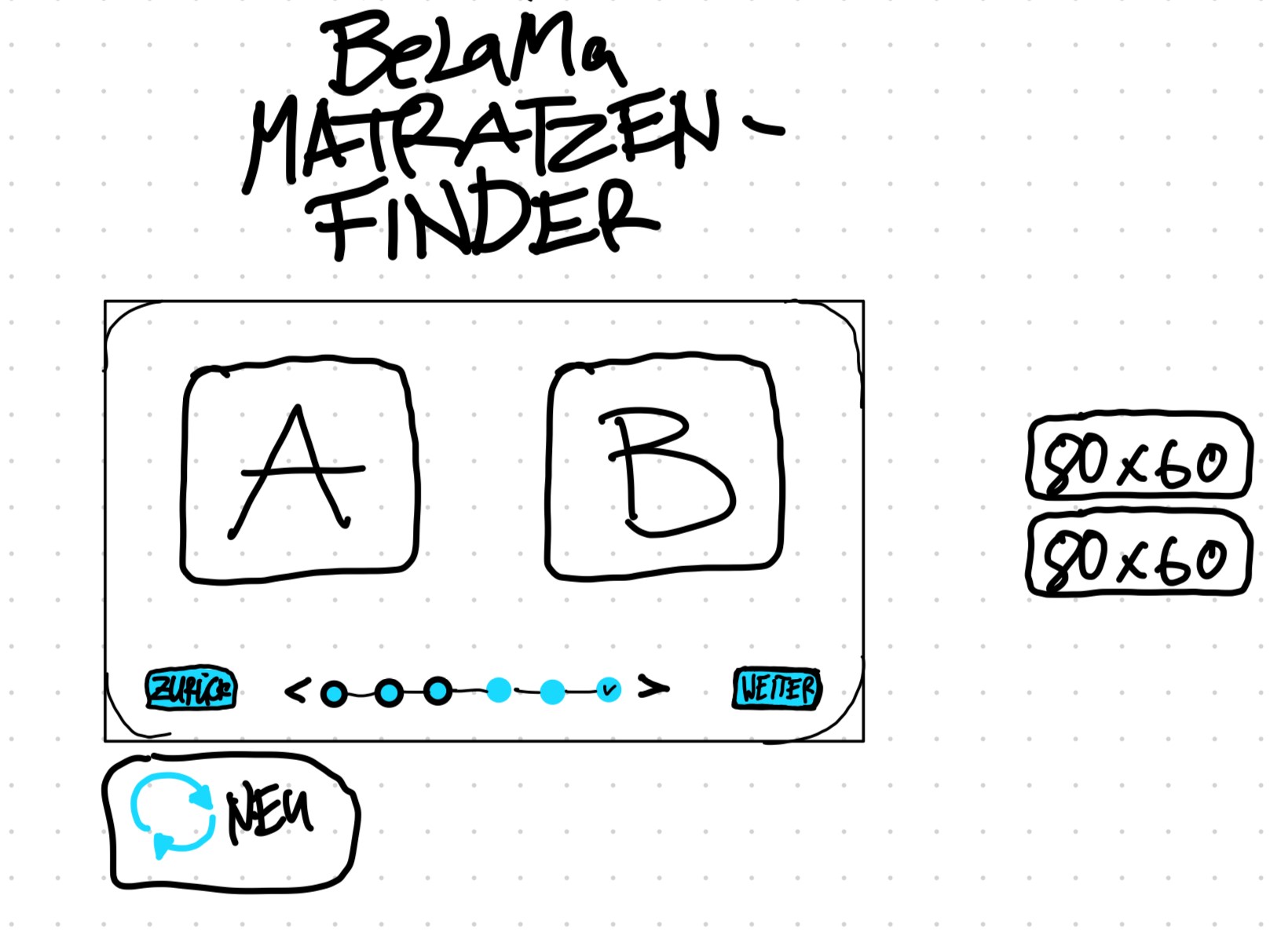
## A.2. Projektkosten

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vorgang** | **Personal** | **Zeit** | **Kosten pro Stunde** | **Gesamt** |
| Entwicklung | Auszubildende | 80 h | 5 € | 400 € |
| Designentwurf | Mitarbeiter Social-Media-Marketing | 3 h | 5 € | 15 € |
|  |  |  |  | 415 € |

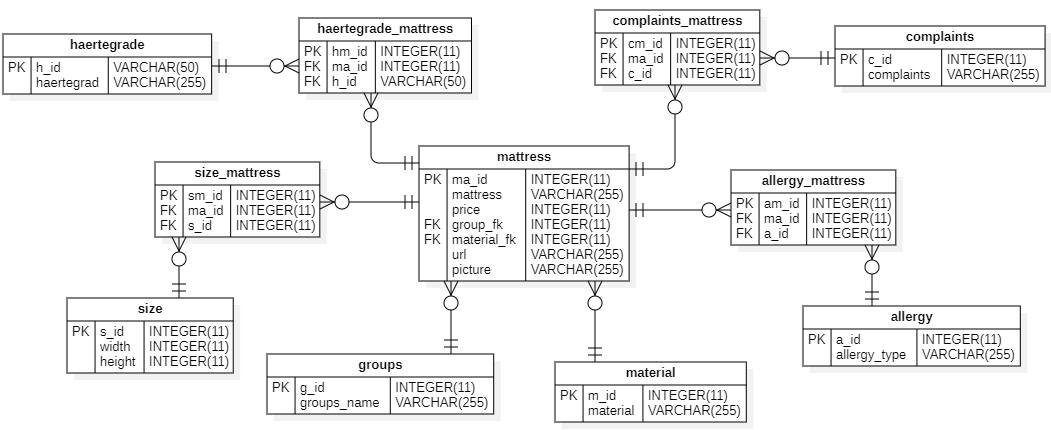
## A.3. Ablaufplan



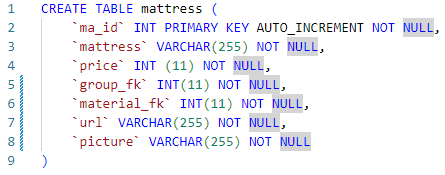
## A.4. Design Mock-Up



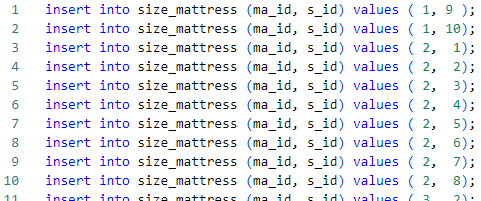
## A.5. Datenbankmodell



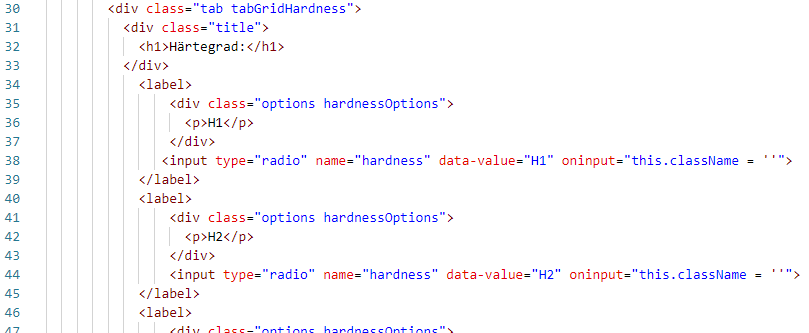
## A.6. SQL – Tabelle erstellen



## A.7. Einfügen der Datensätze



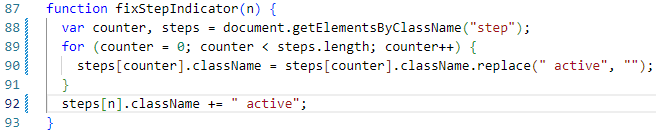
## A.8. HTML – Abschnitt Ausschnitt



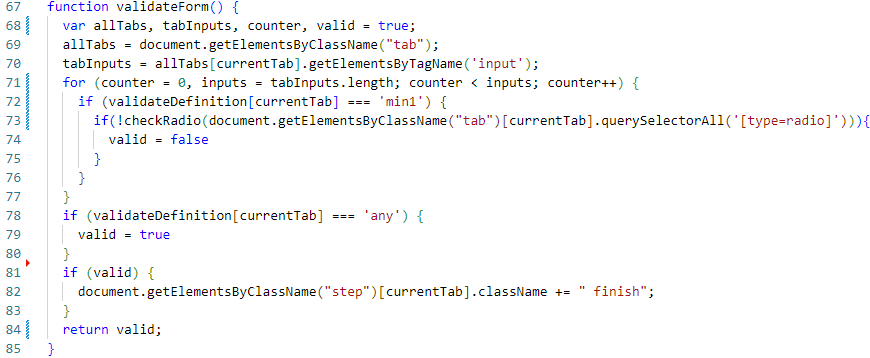
## A.9. schowTab()-Funktion



## A.10. fixStepIndicator()-Funktion



## A.11. validateForm()-Funktion



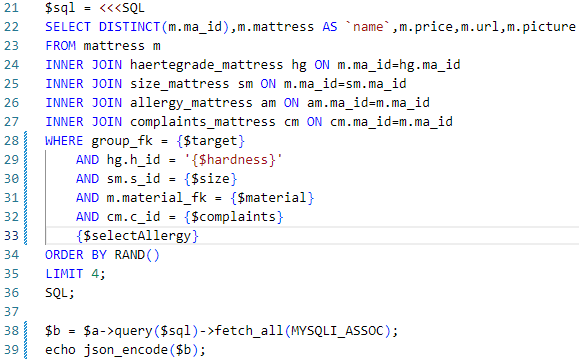
## A.12. nextPrev()-Funktion Ausschnitt



## A.13. formSubmit()-Funktion Ausschnitt



## A.14. SQL-Abfrage



## A.15. Blackbox-Test Testfälle

Wird zuhause ergänzt

## A.16. SOLL-/IST-Vergleich

|  |  |  |
| --- | --- | --- |
| **Phasen** | **Soll** | **Ist** |
| **Analysephase** | **8h** | **8h** |
| * Ist-Analyse | 5h | 5h |
| * Wirtschaftlichkeitsanalyse | 3h | 3h |
| **Entwurfsphase** | **12h** | **7h** |
| * Benutzeroberfläche inkl. Mock-Ups | 6h | 3h |
| * Datenbankstruktur inkl. ER-Modell | 4h | 2h |
| * Ablaufplan | 2h | 2h |
| **Implementierungsphase** | **41h** | **45h** |
| * Einrichten der Datenbank | 6h | 6h |
| * Implementieren der Oberfläche inkl. Tests | 9h | 9h |
| * Implementierung der Datenverarbeitung in PHP und JavaScript | 26h | 30h |
| * + Implementieren des Frontends inkl. Tests | 12h | 14h |
| * + Implementieren des Backends (Empfehlungen) inkl. Tests | 14h | 16h |
| **Testphase** |  | **2h** |
| * Blackbox-Tests |  | 2h |
| **Einrichtung und Abnahme** | **8h** | **7h** |
| * Einrichten der Anwendung | 6h | 6h |
| * Abnahme durch den Geschäftsleiter | 2h | 1h |
| **Dokumentation** | **11h** | **11h** |
| * Erstellen der Projektdokumentation | 11h | 11h |
| **Gesamt** | **80h** | **80h** |