
Abschlussprüfung Sommer 2018

Fachinformatiker für Anwendungsentwicklung

Dokumentation der betrieblichen Projektarbeit

# Erweiterung des Lern- und Prüfungstools (LuPto)

## Hinzufügen eines virtuellen Belohnungssystems

Prüfungsbewerber: Heiko Greis

Prüfling-Nr.: 18028

Eibenstr. 35, 41239 Mönchengladbach

Ausbilder: WBS TRAINING AG

Hindenburgstraße 171, 41061 Mönchengladbach


Praktikumsbetrieb: DEKRA Media GmbH

Dahlener Str. 570, 41239 Mönchengladbach

# Inhalt

# 1 Einleitung

## 1.1 Vorstellung

Diese Projektdokumentation schildert den Ablauf der IHK-Abschlussarbeit im Rahmen einer Ausbildung zum Fachinformatiker in der Fachrichtung Anwendungsentwicklung. Ausbilder ist die WBS TRAINING AG, Standort Mönchengladbach.

Das betriebliche Praktikum wird durchgeführt in dem Unternehmen DEKRA Media GmbH mit derzeit 28 Mitarbeitern. Das Portfolio des Verlags umfasst unter anderem die Produktion von

- Sach- und Lehrbüchern,
- Trainingspräsentationen und Trainingshandouts,
- Web Based und Computer Based Trainings bis hin zu mobilen Lernformen/Apps,
- Lehr- und Produktfilmen sowie
- Learning- und Training-Management-Systemen.

## 1.2 Beschreibung der Ausgangssituation

Das in der Entwicklung stehende System *Lern- und Prüfungstool (LuPto)*, eine Teilumsetzung des Vorhabens *blink – berufsbezogen lernen inklusiv*, soll um ein virtuelles Motivations- und Belohnungssystem erweitert werden.

Das vom Bundesministerium für Bildung und Forschung geförderte Vorhaben *blink – berufsbezogen lernen inklusiv* zielt darauf ab, die Ausbildung im Beruf des/der Anlagenmechaniker/in für Sanitär-, Heizung- und Klimatechnik durch ein mediengestütztes ausbildungsbegleitendes Förderkonzept auch für die Zielgruppe von benachteiligten Jugendlichen zu öffnen. Im Rahmen eines Konzepts der assistierten Ausbildung entwickelt und erprobt das Projekt *blink* verschiedene Unterstützungsangebote, die es benachteiligten Jugendlichen ermöglichen sollen, die Ausbildung erfolgreich zu absolvieren. Der integrierte, mediengestützte Förderansatz kombiniert fachliche und sozialpädagogische Unterstützung.

Ein zentrales Element des Projekts ist das webbasierte Lern- und Prüfungsvorbereitungstool, das zum selbstständigen Lernen, Wiederholen und Vertiefen von Lerninhalten dient. Grundlage ist eine klassische Lernkartei, die mit den Funktionen eines modernen Learning-Management-Systems verklammert wird. Durch diese Verklammerung ist es möglich, dem prinzipiell auf Selbststudium angelegten Lernprozess mit einer Lernkartei eine kommunikative Dimension zu den pädagogischen Begleitern im virtuellen Raum zu eröffnen. Sozialpsychologische Trainer, Ausbildende sowie Berufsschullehrerinnen und Lehrer erhalten auf diese Weise die Möglichkeit, stets Einblick in die individuellen Lernprobleme bzw. den fachlichen Lernfortschritt nehmen zu können. Auf der Basis differenzierter Reportings können sie dann qualifiziert über geeignete Interventionsmaßnahmen entscheiden und diese auch auf individueller Ebene über die gesamte Ausbildungszeit steuern und dokumentieren.

## 1.3 Projektziel

Das übergeordnete Projekt *LuPto* dient dazu, Lerninhalte leicht verständlich aufzuarbeiten, zu präsentieren und sicherzustellen, dass diese verstanden und dauerhaft verinnerlicht werden.

Das Ziel dieses Teilprojekts ist eine Steigerung der Motivation der teilnehmenden Auszubildenden. Damit diese bereits zu Beginn ein Weiterkommen realisieren, sollen schon kleine Lernerfolge bzw. positives Verhalten eine Anerkennung erhalten.

Nach dem Einloggen gelangt der Auszubildende zu seinem virtuellen Schreibtisch, einer Übersicht der verfügbaren Module. Unter der Rubrik „Mein Lernfortschritt" sind die analytische Auswertung der bisher beantworteten Fragen und die grafische Darstellung der eingeloggten Tage abrufbar.

Darüber hinaus soll ein Bereich angelegt werden, in dem Erfolge, aber auch das bisherige Lernverhalten honoriert werden. Wie dies umzusetzen ist bzw. welches Verhalten belohnt wird, ist Teil dieses Projekts. Das bisher verwendete Design ist zu übernehmen bzw. weiterzuführen.

Als Vorlage dienen verschiedene Onlinespiele-Plattformen. So ist das erfolgreiche Abschließen eines Themenbereichs bzw. das Erreichen einer neuen Lernstufe vergleichbar mit dem Erreichen eines neuen Levels im Onlinespiel. In vielen Spielen kann man durch bestimmtes Verhalten, wie zum Beispiel die Interaktion mit anderen Spielern oder durch das Aufsammeln virtueller Werkzeuge und Schätze, Auszeichnungen (von Medaillen bis hin zur Freischaltung von Zusatzangeboten) erlangen. Analog sollen im *LuPto* ebenfalls virtuelle Auszeichnungen an bereits erlangte Erfolge erinnern und so eine ständige Motivation für die weitere Teilnahme sein.

Ausgehend vom virtuellen Schreibtisch zeigt das folgende Use-Case-Diagramm die Möglichkeiten, die einem Auszubildenden nach der Auswahl der Kachel „Ruhmeshalle" zur Verfügung stehen sollen:

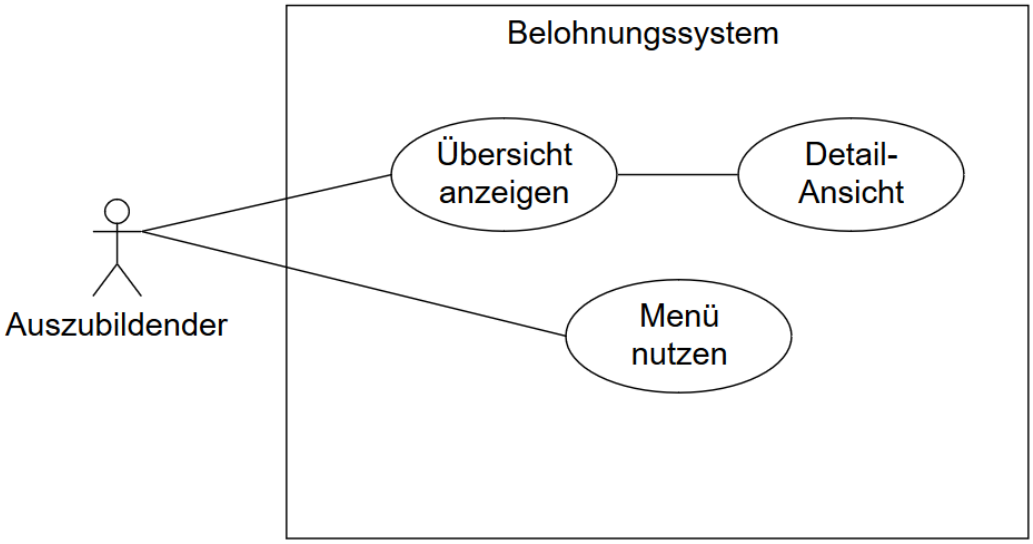


*Abbildung 1: Anwendungsfalldiagramm*

## 1.4 Projektabgrenzung

Aus pädagogischen Gründen ist ausschließlich positives Verhalten zu berücksichtigen. Jegliche negative Verhaltensweisen, z. B. eine Verschlechterung der Leistung oder mangelnde Beteiligung, werden in persönlichen Gesprächen mit den Trainern kommuniziert.

Eine rein statistische Auswertung und Darstellung der Ergebnisse ist bereits vorhanden.

Die erlangten Auszeichnungen dienen ausschließlich der Motivation der Auszubildenden. Daher soll der Bereich auch nur dieser Benutzergruppe zugänglich gemacht werden.

# 2 Ablauf- und Ressourcenplanung

## 2.1 Konzeptentwicklung

Dieses Projekt ist ein Teilprojekt. Damit die weitere Entwicklung und eventuelle Änderungen an übergeordneten Projektbestandteilen keinen Einfluss auf diese Projektarbeit haben, wurde der Stand vom 26.03.2018 zugrunde gelegt. Eine Implementierung findet später mittels GitHub statt, gehört aber nicht mehr zum Umfang dieser Projektarbeit.

Basierend auf den Anforderungen, die bereits unter Punkt 1.3 Projektziel und den Abgrenzungskriterien unter Punkt 1.4 Projektabgrenzung Erwähnung finden, wurde ein Pflichtenheft (Anhang A) erstellt.

Da dieses Projekt von nur einer Person durchgeführt wird, ist keine Absprache mit der Fachabteilung nötig. Die einzelnen Entwicklungsschritte bauen meist aufeinander auf. Somit bietet sich das Wasserfallmodell als geeignetes Vorgehensmodell an. Für eine bessere Übersicht und Einhaltung der angedachten Durchführungszeiten wurden im Projektablaufplan (Tabelle 1) die Phasen nicht nur aufgelistet, sondern auch gleich in die anfallenden Arbeitsschritte unterteilt.

## 2.2 Ressourcenplanung

Unter Punkt 3.2 sind alle notwendigen Ressourcen detailliert aufgelistet. Diese stehen dem Verfasser für die gesamte Projektdauer uneingeschränkt zur Verfügung. Eine Planung bzw. Absprache mit anderen Personen entfällt somit.

Bei der Umsetzung fallen keine besonderen Anforderungen an die Hardware an. Da überwiegend kostenfreie Programme verwendet werden, ist keine Beachtung von Lizenzen oder Ähnlichem nötig – ein Standard-EDV-Arbeitsplatz ist ausreichend.

Eingesetztes Personal ist ausschließlich der Verfasser dieser Dokumentation.

## 2.3 Projektphasen

Zur Umsetzung des Projekts stehen 70 Stunden (Vorgabe der IHK) zur Verfügung. Diese wurden vor Projektbeginn auf verschiedene Phasen verteilt, die während der Softwareentwicklung durchlaufen werden:

*Tabelle 1: Projektablaufplan*

| **Projektdauer** | **70 Stunden** |
|---|---|
| **Analyse** | **9 Stunden** |
| Ist-Analyse/Bestandsanalyse | 3 Stunden |
| Analyse des bestehenden Quellcodes | 3 Stunden |
| Analyse der Wirtschaftlichkeit | 3 Stunden |
| **Entwurf** | **10 Stunden** |
| Use-Case-Diagramm | 2 Stunden |
| Entity-Relationship-Modell | 2 Stunden |
| Programmablaufplan | 3 Stunden |
| Pflichtenheft | 3 Stunden |
| **Realisierung** | **22 Stunden** |
| Installation/Konfiguration | 2 Stunden |
| Modell „Erfolge“ erstellen und Datenbank migrieren | 2 Stunden |
| Menü-Auswahl „Erfolge“ hinzufügen | 1 Stunde |
| Erstellung: Ansicht „Erfolge“ | 2 Stunden |
| Erstellung: Detailansicht | 2 Stunden |
| Algorithmen zur Erkennung von Lernerfolgen | 12 Stunden |
| Event zur Ausführung der Überprüfung und Speicherung | 1 Stunde |
| **Tests** | **12 Stunden** |
| Manipulation der Datenbank für Funktionstests | 3 Stunden |
| Durchführung und Protokollierung aller Eintrittsbedingungen | 4 Stunden |
| Fehlerbehebung | 5 Stunden |
| **Dokumentation** | **17 Stunden** |
| Projektdokumentation | 17 Stunden |

## 2.4 Zeitlicher Ablauf – Gantt-Diagramm

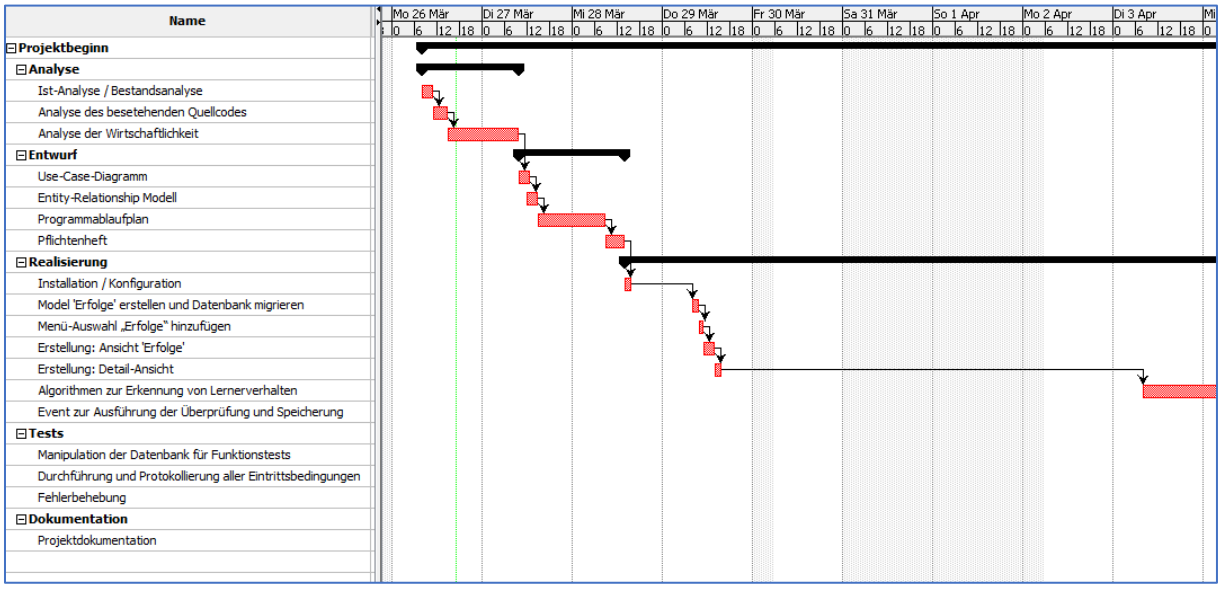


*Abbildung 2: Gantt-Diagramm a*

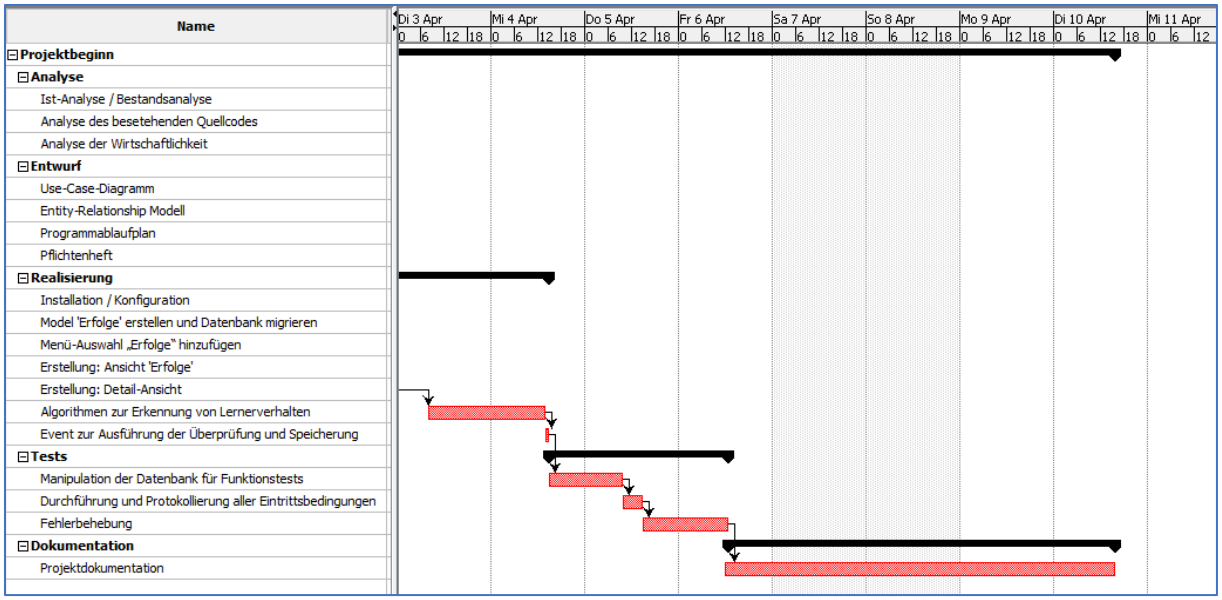


*Abbildung 3: Gantt-Diagramm b*

Arbeitsfreie Tage:
30.03.2018 – gesetzl. Feiertag (Karfreitag)
31.03.2018 – Samstag
01.04.2018 – Sonntag
02.04.2018 – gesetzl. Feiertag (Ostermontag)
07.04.2018 – Samstag
08.04.2018 – Sonntag

Tägliche Arbeitszeit: 7 Std.

# 3 Analysephase

## 3.1 Ist-Analyse

Die DEKRA Media AG wurde mit der Umsetzung einer webbasierten Anwendung zum selbstständigen Lernen, Wiederholen und Vertiefen von Lerninhalten in Verbindung mit einschlägigen LMS- und Dokumentationsfunktionen betraut. Aktuell befindet sich das Projektteam in der Kernentwicklungsphase, d. h., die grundlegenden Funktionen sind implementiert und lauffähig:

- Benutzerverwaltung
- Erstellung und Zuweisung der Inhalte
- Kommunikationssystem für Trainer und Teilnehmer
- Lernsystem (Single- und Multiple-Choice-Fragen, Freitextantworten)
- Als Vorlage dient das Karteikarten-Prinzip (https://de.wikipedia.org/wiki/Lernkartei).
- Auswertung der Lernerfolge

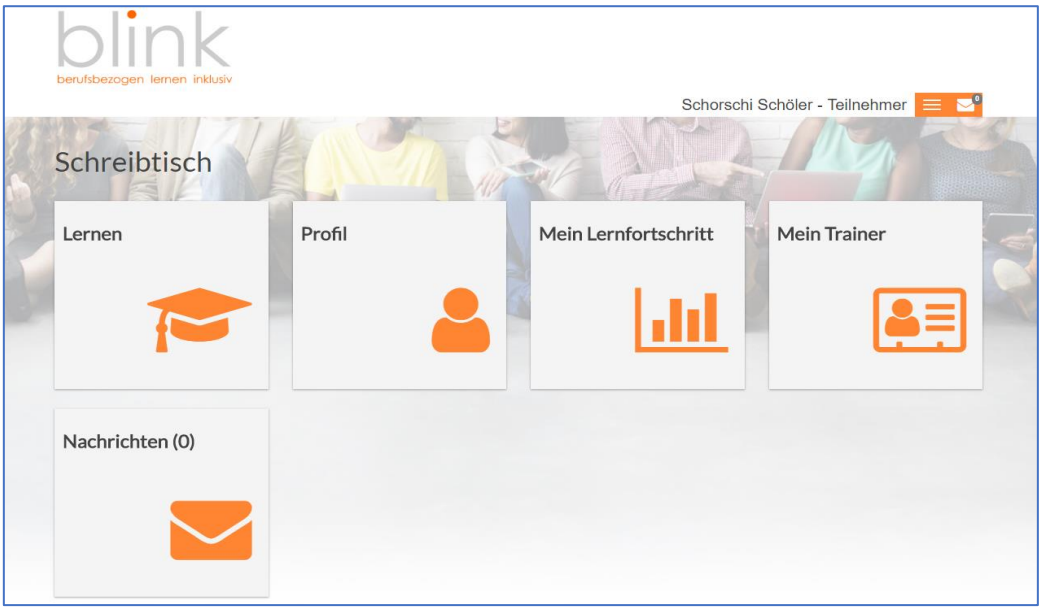


*Abbildung 4: Funktionsumfang vor Projektstart*

Kurze Beschreibung der Funktionsweise aus Anwendersicht: Der Auszubildende kann sich mit seinen ihm mitgeteilten Benutzerdaten einloggen. Sofern ihm sein Trainer bereits einen Fragenkatalog zugewiesen hat, kann er mit der Beantwortung beginnen. Wird eine Frage korrekt beantwortet, wandert sie in ein höheres Fach und wird zukünftig seltener gestellt.

Verwendet wird das PHP-Framework *Laravel*. Eine ausführliche Dokumentation ist auf der *Laravel*-Homepage https://laravel.com/docs/5.5 zu finden.

Um der Forderung, das bisherige Design weiterzuführen und doppelte Datenbankeinträge zu vermeiden, nachzukommen, ist eine intensive Studie des bisherigen Quellcodes notwendig.

## 3.2 Was ist zur Erfüllung der Zielsetzung erforderlich?

Die folgende Auflistung zeigt nicht die zwingenden Voraussetzungen für die Umsetzung des Projektziels, sondern beschreibt überwiegend einen Standard-EDV-Arbeitsplatz bei der DEKRA Media GmbH.

Hardware:

- PC mit Intel i3 530 CPU, 4 GB RAM, 500 GB Festplatte
- Peripherie: Monitor, Tastatur, Maus, Netzwerkkarte
- Internetzugang

Software:

- Betriebssystem: Microsoft Windows 10 Pro
- Office-Programme: Microsoft Office Home oder Pro 2016
- IDE/Texteditor mit Code-Vervollständigung: Sublime Text V3.0
- *Laravel*-Framework: Laragon V3.1 (beinhaltet u. a. Apache-Server, SQL-Datenbank)
- Internetbrowser: Mozilla Firefox V52.0.2, Google Chrome V63.0.3239.84 oder Microsoft Internet Explorer V11.726.15063.0 – im Browser muss JavaScript verfügbar sein.
- Project Libre Version 1.6.2
- Zur Erstellung von UML-Grafiken (UML = *Unified Modeling Language*) wurde das Online-Tool draw.io genutzt.
- DbVisualizer V10.0.8

## 3.3 Wirtschaftlichkeitsanalyse

### 3.3.1 Make-or-Buy-Entscheidung

Auf dem Markt gibt es einige Angebote zum Thema „Karteikartensystem-gestütztes Lernen". Die aktuellen Lösungsansätze basieren jedoch darauf, dass der Anwender selbst die Fragen und Antwortmöglichkeiten eingeben muss. Dies ist zum einen sehr zeitintensiv und zum anderen auch eine mögliche Fehlerquelle.

Hinzu kommt, dass im System *LuPto* sozialpädagogisch ausgebildete Trainer die Inhalte individuell zusammenstellen und über ein Kommunikationssystem für Rückfragen bereitstehen.

Ein solches Gesamtkonzept ist derzeit nicht erhältlich – ein Einkauf bietet sich daher nicht an.

### 3.3.2 Kosten

Dieses Teilprojekt findet im Rahmen einer Umschulung statt. Dem Praktikumsbetrieb DEKRA Media GmbH entstehen somit keine direkten Personalkosten, abgesehen von Kosten für den Zeitaufwand von Mitarbeitenden der Fachabteilung, die für Rückfragen bereitstehen. Für eine realistische Einschätzung wird der folgenden Berechnung ein Stundensatz eines Auszubildenden in Höhe von 25,00 €/Stunde (inklusive aller Lohnkosten- und Arbeitgeberanteile) zugrunde gelegt.

Die Kosten dieses Projekts setzen sich aus Personal- und Gemeinkosten zusammen. Zu den Gemeinkosten gehören unter anderem anteilig:

- Gebäudekosten (Miete, Heizung, Stromverbrauch, ...)
- Büroverbrauchsmaterialien
- Hardware-Nutzung eines EDV-Arbeitsplatzes
- Lizenzgebühren für Standard-Software (MS Windows, MS Office)

Eine genaue Aufschlüsselung der Teilkosten stand nicht zur Verfügung. Die DEKRA Media GmbH rechnet hier ebenfalls mit festen Stundensätzen. Aktuell betragen die Gemeinkosten 18,00 €/Stunde.

Zusätzliche Kosten, wie zum Beispiel die Anschaffung von Hardware, die ausschließlich für dieses Projekt benötigt wird, fallen nicht an. Durch die Verwendung von Open-Source-Software entstehen auch keine weiteren Kosten für beispielsweise Lizenzgebühren oder Ähnliches.

Die Arbeitszeit, die für die Endabnahme und die Implementierung in den aktuellen Stand des Gesamtprojekts anfällt, gehört nicht zur Aufgabe dieser Projektarbeit und wird somit bei der Berechnung der Kosten auch nicht berücksichtigt:

Gesamtkosten = Projektdauer * (Personalkosten + Gemeinkosten)

Gesamtkosten = 70 Stunden * (25,00 €/Stunde + 18,00 €/Stunde)

**Gesamtkosten = 3.010,00 €**

### 3.3.3 Amortisation

Die Umsetzung dieses Projekts zielt auf die Motivation von Auszubildenden ab, das Werkzeug *LuPto* häufiger zu nutzen. Dieses ist ein nicht monetärer Vorteil. Eine Möglichkeit, die Nutzungssteigerung durch diese Erweiterung messbar zu machen, bestünde darin, dass eine Kontrollgruppe ohne und eine Gruppe mit diesem Zusatz arbeitet. Aussagekräftige Ergebnisse würde man anschließend nach Durchlaufen einer gesamten Ausbildungsdauer erhalten.

Diese Auswertung ist jedoch ausdrücklich nicht erwünscht, denn die Kontrollgruppe würde bei der Durchführung benachteiligt.

Da sich der Nutzen nicht in geldwerten Vorteilen darstellen lässt, ist eine tatsächliche Berechnung der Amortisationsdauer nicht möglich. Stattdessen beschränke ich mich in dieser Dokumentation auf die Berechnung der anfallenden Kosten.

Für die Durchführung dieser Projektarbeit entstehen Kosten in Höhe von 3.010 €.

# 4 Entwurfsphase

## 4.1 Zielplattform

Die einzigen Anforderungen an die Hardware der Auszubildenden sollen ein Internetzugang und die Verwendung eines Standard-Browsers auf einem beliebigen Endgerät sein. Dieses Projekt wird in Form einer Web-Applikation umgesetzt, um plattformunabhängig zu sein.

Die bisher existierenden Programmteile wurden unter Verwendung des PHP-Frameworks *Laravel* entwickelt. Um bereits bestehende Klassen problemlos nutzen zu können, bietet sich hier die weitere Verwendung an. Ebenfalls bietet die Umgebung schon eine Benutzerverwaltung und ein integriertes Datenbankmanagement. Jegliche Rechnerleistung findet auf dem Server statt und entlastet somit die Hardware der Auszubildenden.

Diese Vorgaben erübrigen eine gesonderte Analyse der geeigneten Programmiersprachen oder Datenbanksysteme.

## 4.2 Architekturdesign

Das Projekt soll auf der Basis des MVC-Architekturmusters (*Model View Controller*) umgesetzt werden. Dies ergibt sich einerseits aus den Konventionen rund um *Laravel* und andererseits ist es ein sinnvolles Konzept, sobald mehrere Entwickler an einem Projekt beteiligt sind.

Das Model-View-Controller-Paradigma lässt sich wie folgt zusammenfassen: Jede Komponente eines Programms lässt sich einem der drei Bestandteile dieses Musters – *Model*, *View* oder *Controller* – zuordnen. Das *Model* beinhaltet die Daten und die zugehörige Verarbeitungslogik. Der *View* ist für die Darstellung von Daten zuständig und im *Controller* findet die Steuerung der Anwendung statt. Der *Controller* stellt das Bindeglied zwischen *Model* und *View* dar. Die Kapselung der Bestandteile erhöht die Wiederverwendbarkeit bzw. die Austauschbarkeit. Auch das Aufsplitten in Teilaufgaben bei Teamarbeit ist so übersichtlicher.

Das gesamte Datenbankhandling wird vom *Laravel*-Framework nach einmaliger Konfiguration und Migration der modellierten Struktur übernommen. Die Speicherung und das Zugreifen auf Daten erfolgen mittels einer Befehlslogik (*Object Relational Mapping*, ORM):

*Laravel*-Eloquent: https://laravel.com/docs/5.5/eloquent.

## 4.3 Programmablaufplan

Die zu programmierenden Funktionen wurden in einem Flussdiagramm abgebildet. Zur Vereinfachung bzw. aus Gründen einer besseren Übersicht wurden die zu überprüfenden Bedingungen und das Abspeichern erzielter Trophäen zusammengefasst.

Weiterhin ist im ganzen System *LuPto* ein Menü aufrufbar, über das man direkt zu anderen Modulen wechseln kann. Der Verfasser dieser Dokumentation hat versucht, sich auf die Funktionen, die für dieses Teilprojekt erforderlich sind, zu beschränken:

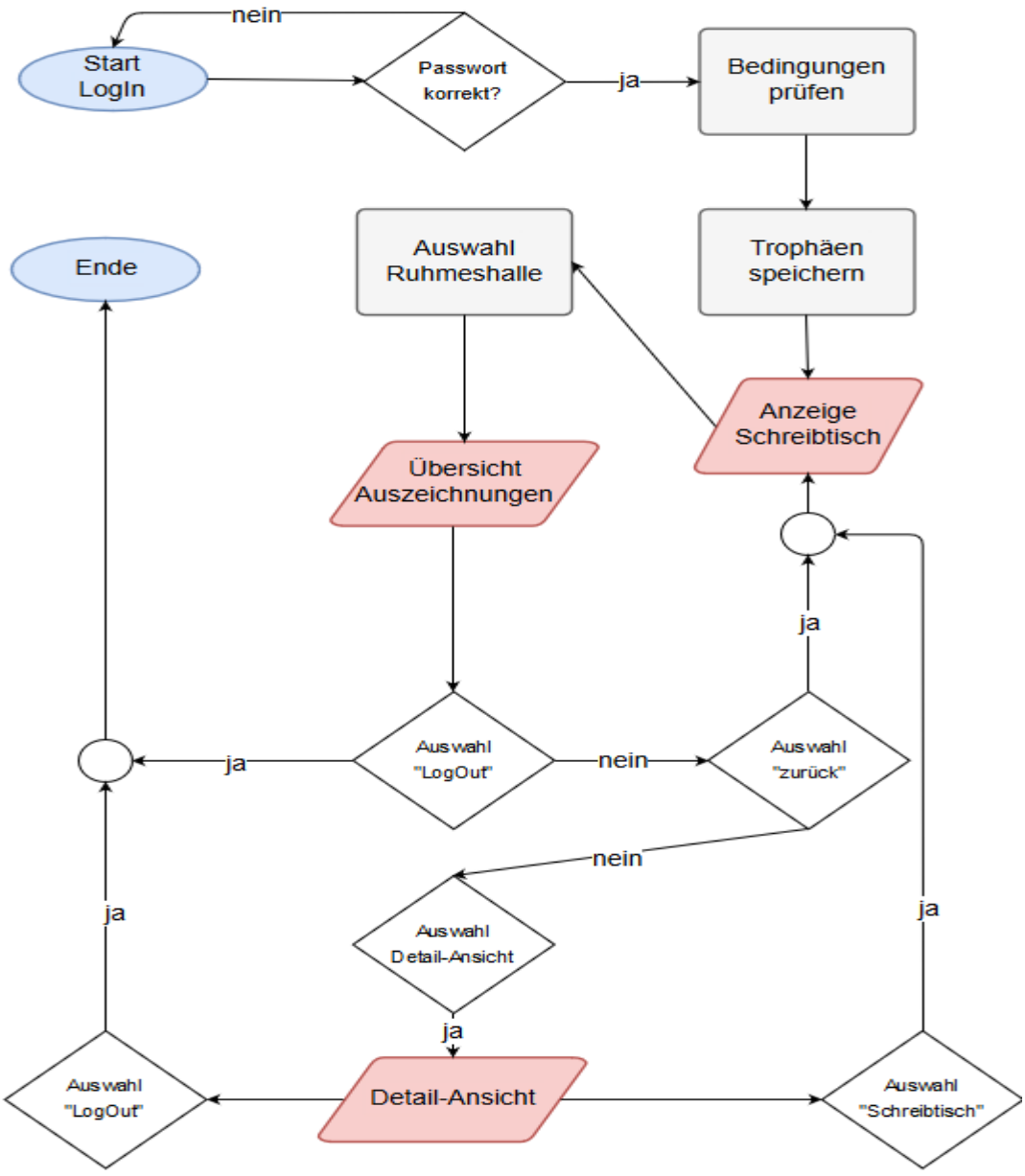


*Abbildung 5: Programmablaufplan*

## 4.4 Entwurf der Benutzeroberfläche

Das ganze Programm richtet sich an lernschwache Auszubildende. Eine intuitive, leicht verständliche Benutzeroberfläche ist eine zwingende Voraussetzung. Hierzu können alle Funktionen über ein zentrales Menü aufgerufen werden. Darüber hinaus wird der Teilnehmer nach dem Log-in automatisch zum „Schreibtisch" geleitet. Hier sind alle Informationen und Optionen übersichtlich mit leicht verständlichen Piktogrammen dargestellt.

Für die Piktogramme steht eine kostenfreie Bibliothek zur Verfügung: http://fontawesome.io/icons/, lediglich eine Farbanpassung wird vorgenommen.

Auch hier gilt, dass bereits existierende Klassen verwendet werden sollen. Ein großer Teil der Projektarbeit besteht also aus dem Auffinden und der Analyse bereits bestehender Code-Fragmente.

Auf dem „Schreibtisch" soll eine passende Verknüpfung angelegt werden. In Anlehnung an beliebte Sportarten wurde die Bezeichnung „Ruhmeshalle" gewählt. Nach der Auswahl wird eine kurze Erklärung der Rubrik generell angezeigt und alle erlangten Auszeichnungen werden mittels aussagekräftiger Piktogramme dargestellt. Nach Auswahl einer erhaltenen Trophäe wird der Teilnehmer zu deren Erklärung weitergeleitet.

Die Umsetzung der Oberfläche erfolgt über HTML, erweitert durch *Laravel Blade*. Hierbei handelt es sich um eine Template-Engine, also eine Software, die eine Vorlagen-Datei (engl. *template*) verarbeitet und mittels Platzhaltern die aktuellen, individuellen Inhalte ersetzt.

Aufbauend auf dem übergeordneten Design wurden bereits existierende Vorlagen verwendet und mit den entsprechenden Inhalten befüllt. In der folgenden Abbildung ist der vom Verfasser erstellte Bereich schwarz umrandet:


*Abbildung 6: Benutzeroberfläche*

## 4.5 Datenmodell

Welche Daten benötigt werden, hängt von den Bedingungen der jeweiligen Auszeichnung ab. Eine ausgiebige Analyse der bestehenden Modelle zeigt, welche Daten und Beziehungen bereits vorliegen und welche Klassen angelegt werden müssen.
Basierend auf den erlangten Erkenntnissen wurde ein Tabellen-Modell erstellt, das die relevanten Typen und Beziehungen darstellt (Abbildung 7).

Da der eigentliche Aufbau der Datenbank von *Laravel* mittels Migrations-Klassen erledigt wird, dient diese Grafik lediglich zur Veranschaulichung.

Eine genaue Dokumentation zu diesem Thema ist hier abrufbar: https://laravel.com/docs/5.5/migrations.

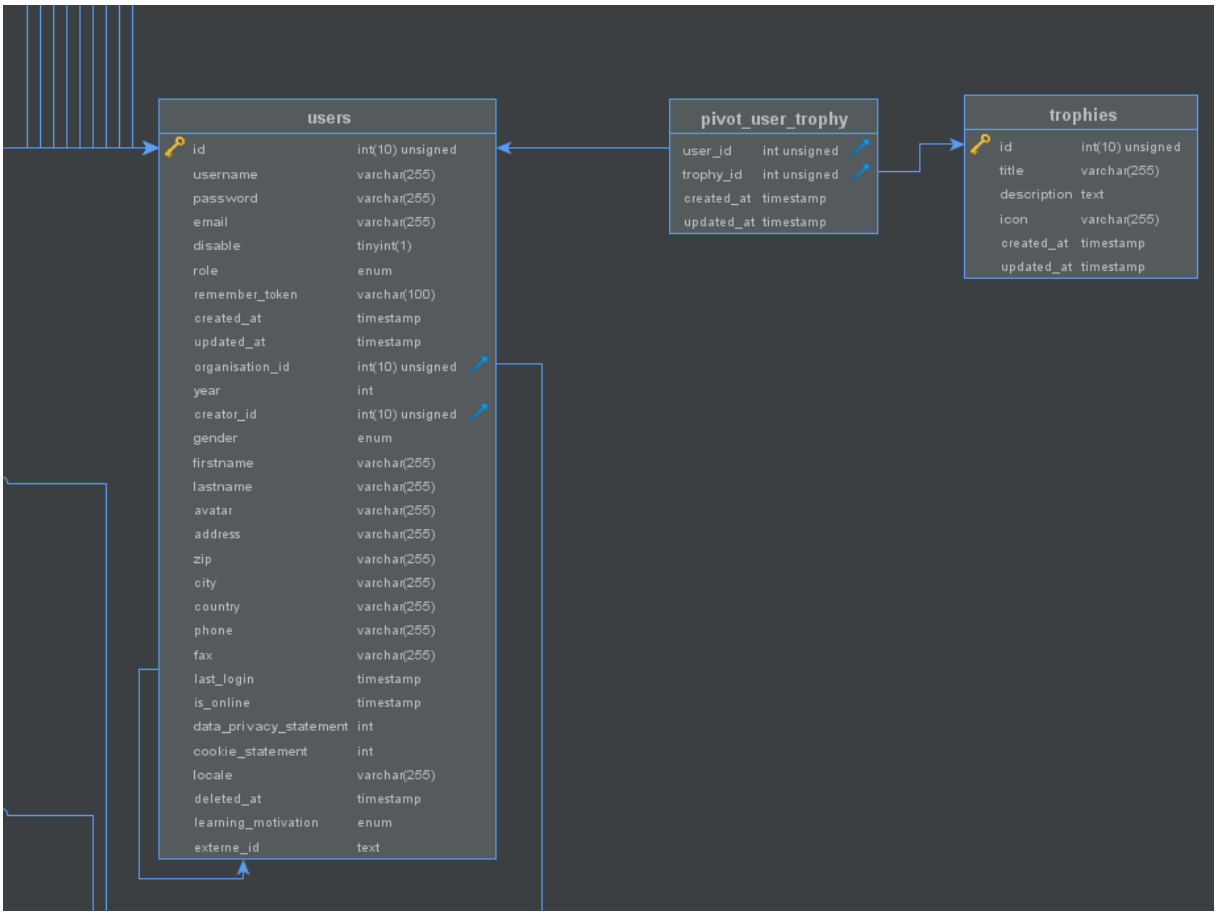


*Abbildung 7: Ausschnitt Tabellen-Modell*

## 4.6 Pflichtenheft

Zum Ende der Entwurfsphase wurde das Pflichtenheft erstellt. Dieses beschreibt, welche Anforderungen wie umzusetzen sind. Es dient somit als verbindlicher Leitfaden für die Realisierung dieses Projekts. Ein Auszug aus dem Pflichtenheft ist dieser Dokumentation beigefügt (Anhang 1).

# 5 Implementierungsphase

## 5.1 Programmierumgebung

Das ganze Projekt *LuPto* wird mittels des Frameworks *Laravel 5.5* umgesetzt. Die eigentliche Programmiersprache ist PHP, deren Bezeichnung ursprünglich ***P**ersonal **H**ome **P**age Tools* bedeutete. Heute steht PHP für ***P**HP: **H**ypertext **P**reprocessor* und beschreibt somit die grundlegende Funktionsweise dieser Sprache. Der Quellcode kann mit einem einfachen Texteditor, wie zum Beispiel Notepad, verfasst werden. Die Verarbeitung (Kompilierung) findet auf dem Server statt, der das Ergebnis an den Browser des Internetnutzers sendet. Übermittelt wird diese Anzeige in HTML (*Hypertext Markup Language*). Diese Vorgehensweise hat wesentliche Vorteile:

- Die Rechenleistung des Anzeigegeräts wird niedrig gehalten und eignet sich somit hervorragend für mobile Endgeräte.
- Sensible Daten, wie zum Beispiel die Anmelde-Informationen für das Datenbank-Management-System, werden erst gar nicht übertragen und sind somit auch nicht auslesbar.

Zur Umsetzung wird das Programm *Laragon* in der Version 3.1 genutzt. Dieses beinhaltet mehrere nützliche Werkzeuge, wie z. B.:

- einen lokalen Apache-Server zur Ausführung von PHP-Quellcode sowie
- ein Datenbank-Management-System (HeidiSQL V9.4.0.5125).

Die wesentlichen Vorteile des verwendeten Texteditors *Sublime* sind die automatische Code-Vervollständigung und das Hervorheben der PHP-Syntax.

## 5.2 Implementierung der Datenstruktur

*Laravel* übernimmt unter anderem auch das Anlegen von Tabellen für die Datenbank. Der Aufbau der Tabelle wird in einer Migrations-Datei definiert, die unter Verwendung eines Konsolenbefehls automatisch generiert wird. Neben der Tabelle „trophies", die u. a. die Bezeichnung, Beschreibung und das zugehörige Piktogramm der Auszeichnung enthält, wird eine Zwischentabelle benötigt. Viele Nutzer können viele Trophäen erlangen – dieses ist eine n:m-Beziehung. Die Befehle zur Erzeugung beider Tabellen lauten:

```
php artisan make:migration create_trophies_table
php artisan make:migration create_pivot_user_trophy_table
```

Der Inhalt dieser Migrations-Dateien ist im Anhang A5 einsehbar.

Nachdem beide Dateien entsprechend bearbeitet sind, kann die Migration mit folgendem Konsolen-Befehl gestartet werden:

```
php artisan migrate
```

## 5.3 Implementierung der Benutzeroberfläche

Da das bisherige Design weitergeführt werden soll, wurde der Aufbau existierender Blade-Dateien (so nennt man bei *Laravel* die Ansichts-Dateien) übernommen. Der Controller ruft diese HTML-basierten Ansichten auf und übergibt die darzustellenden Daten. Der genaue Quellcode hierfür ist im Anhang A4 hinterlegt.

Die Abbildung 8 zeigt die Ansicht eines Nutzers, der alle möglichen Auszeichnungen bereits erlangt hat. Nach dem Anklicken einer Kachel wird der Auszubildende zur Detailansicht weitergeleitet. Die verschiedenen Ansichten sind in der Benutzerdokumentation (Anhang A6) aufgelistet.

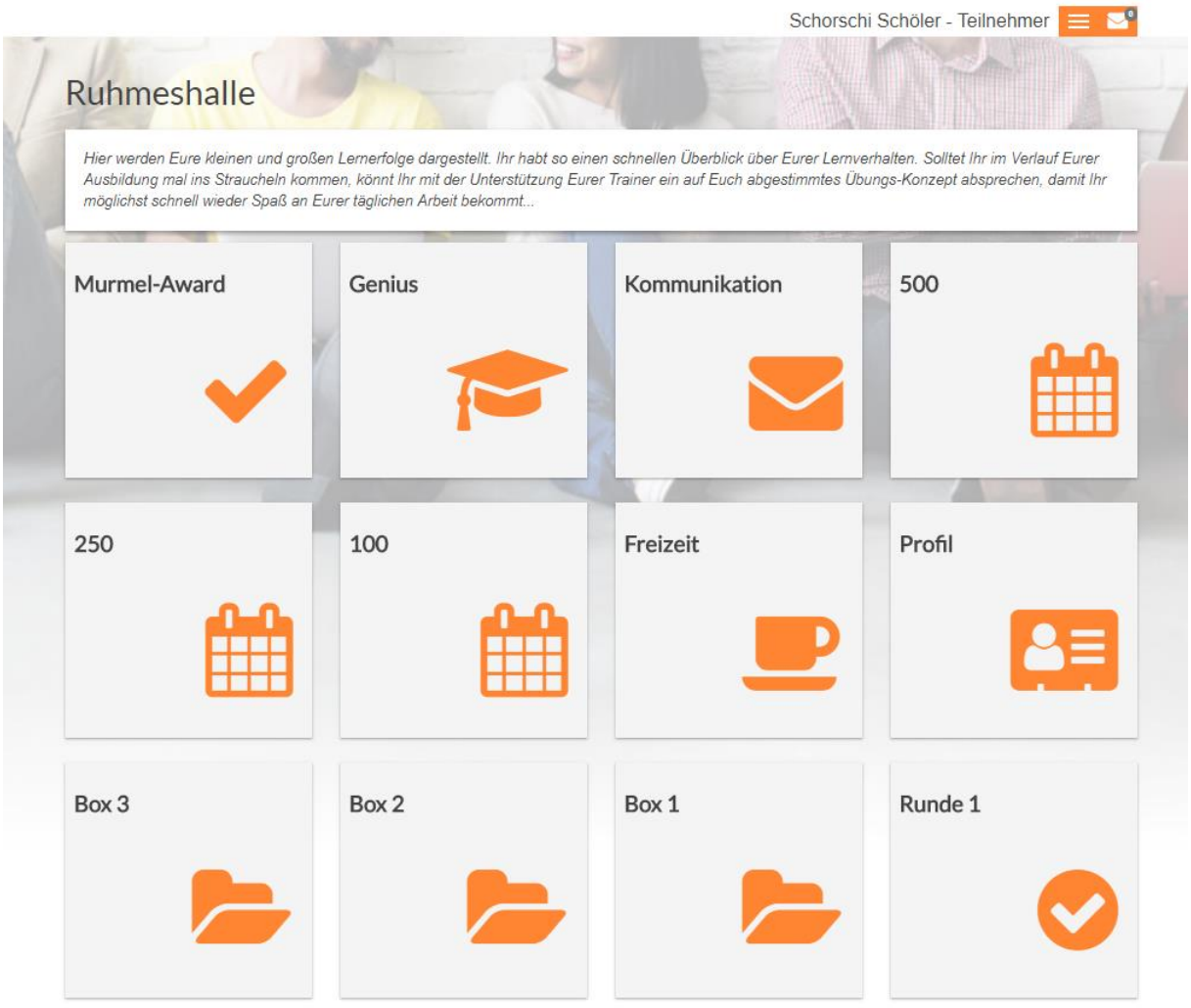


*Abbildung 8: Screenshot – Übersicht Auszeichnungen*

## 5.4 Tests

Die Prüfung der Eintrittsbedingungen für jede Auszeichnung wurde mittels einfacher Wenn-Dann-Sonst-Unterscheidungen realisiert. Um die korrekte Zuweisung zu überprüfen, wurde jede entscheidende Situation mittels einer manuellen Manipulation der Datenbank simuliert.

Die Trophäe „Murmel-Award" wird vergeben, wenn sich der Auszubildende an jedem Werktag der Vorwoche eingeloggt hat. Zur Überprüfung wurden folgende Situationen für die Vorwoche temporär hergestellt:

*Tabelle 2: Funktionstest für die Auszeichnung „Murmel-Award"*

| Login-Daten | Trophäe zugewiesen |
|---|---|
| Keine Daten vorhanden | Nein |
| Montag–Freitag | Ja |
| Montag, Dienstag, Donnerstag, Freitag | Nein |
| Montag, Dienstag, Donnerstag, Freitag, Samstag | Nein |
| Sonntag, Montag, Dienstag, Mittwoch, Donnerstag | Nein |
| Dienstag, Mittwoch, Donnerstag, Freitag, Samstag | Nein |

Diese Vorgehensweise wurde auf alle zu vergebenden Auszeichnungen angewandt. Besonderes Augenmerk lag hierbei auf den Grenzbereichen. Da die Bedingungen sehr unterschiedlich sind, wurde auf automatisierte Tests verzichtet.

# 6 Dokumentation

## 6.1 Projektdokumentation

In der Projektdokumentation sind die einzelnen Phasen während der Entstehung dieses (Teil-)Projekts ausführlich beschrieben.

## 6.2 Entwicklerdokumentation

Die Entwicklerdokumentation umfasst alle während dieses Projekts entstandenen bzw. geänderten Dateien inklusive einer kurzen Beschreibung der Funktionalität. Auszüge aus dieser Dokumentation sind in den Anhängen A4 und A5 hinterlegt.

## 6.3 Kundendokumentation

Die Kundendokumentation beinhaltet eine Funktionsbeschreibung aus Sicht eines Teilnehmers. Eingegangen wird ausschließlich auf den Bereich, der im Rahmen dieses Projekts entsteht. Ein Auszug, der Bildschirmfotografien der Detailansicht aller Auszeichnungen beinhaltet, ist im Anhang A6 beigefügt.

# 7 Fazit

## 7.1 Abnahme durch die Fachabteilung

Nach der Fertigstellung des Programms wurde ein Fachgespräch mit dem verantwortlichen Projektteam geführt.

Die Vorgaben aus dem Pflichtenheft wurden in Gänze umgesetzt und das Teilprojekt wird zu gegebener Zeit dem Gesamtprojekt *LuPto* zugeführt.

## 7.2 Soll-/Ist-Vergleich

Unter Punkt 2.3 wurde ein Projektphasenplan erstellt. Diese Einschätzungen basieren auf Erfahrungswerten des Entwicklers.

Da diese IHK-Projektarbeit Teil einer Umschulung ist, die in einem Praktikumsbetrieb durchgeführt wird, ist sicherlich nachvollziehbar, dass der Verfasser über keinen allzu großen Erfahrungsschatz verfügt. Zwar konnte die Vorgabe von einer Gesamtdauer von 70 Stunden eingehalten werden, jedoch war die Aufteilung der Teilaufgaben eine andere. So hat zum Beispiel die Erstellung der Dokumentation ca. 22 Stunden statt der geplanten 17 Stunden in Anspruch genommen. Im Gegensatz dazu konnte bei der Fehlerbehebung viel Zeit eingespart werden, da die Tests parallel zur Erstellung der Algorithmen durchgeführt wurden.

Bemerkenswert ist auch die Tatsache, dass die eigentliche Realisierung der Aufgabe nur einen geringen Teil der Projektarbeit ausmacht. Sicher ist der betriebene Aufwand für größere Projekte gerechtfertigt und vereinfacht die Zusammenarbeit mehrerer Kollegen an einem Projekt. Fraglich bleibt jedoch, ob potenzielle Kunden aus dem Mittelstand bei kleinen Programmieraufgaben durch den verhältnismäßig hohen Zeitaufwand für Dokumentation und den Entwurf von Grafiken sowie den damit verbundenen Kosten nicht eher abgeschreckt werden.

# A1 Pflichtenheft (Auszug)

In diesem Auszug aus dem Pflichtenheft wird die Umsetzung der Anforderungen an das Teilprojekt beschrieben:

⇨ Lobenswertes Lernverhalten bzw. Lernerfolge, die hervorzuheben sind:

- Ein vollständig ausgefülltes Benutzerprofil
- Wenn erstmalig jede Frage mindestens einmal beantwortet wurde
- Wenn sich erstmalig keine Frage mehr in dem ersten Karteikartenfach befindet
- Wenn sich erstmalig keine Frage mehr in dem zweiten Karteikartenfach befindet
- Wenn sich erstmalig keine Frage mehr in dem dritten Karteikartenfach befindet
- Tägliches Einloggen an Werktagen der Vorwoche – diese Auszeichnung ist temporär und die Bedingung dafür wird bei jeder Überprüfung erneut abgefragt.
- Das Einloggen in das System außerhalb der täglichen Arbeitszeit

⇨ Anforderungen an die Programmierung:

- Bestehende Klassen (z. B. Benutzer) sollen nach Möglichkeit erweitert werden.
- Die Speicherung der Auszeichnungen findet über das Framework statt (keine direkten Einträge in der Datenbank erzeugen, sondern dem bisherigen Paradigma folgen und erzeugte Klassen-Modelle abspeichern).
- Die Fall-Überprüfungen und das Speichern der Ergebnisse finden nach dem Einloggen des Benutzers statt. Hierzu soll ein *EventListener* angelegt werden, damit die Prüfung bei einem späteren Betrieb täglich zu einer festen Uhrzeit stattfinden kann.
- Quelle der zu verwendenden Piktogramme: [fontawesome.io/icons/](fontawesome.io/icons/)
- Darstellung: Das Design baut auf schon bestehenden Inhalten auf; hierzu sind bereits existierende Klassen erneut zu verwenden:

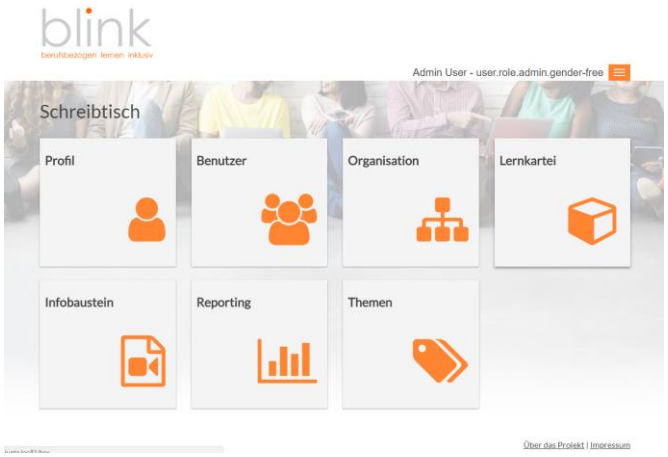


*Abbildung: Design-Beispiel*

# A2 Datenschutzbestimmungen

Erst nach einer Lesebestätigung bzw. Einverständniserklärung der Datenschutzbestimmungen kann das *LuPto* genutzt werden. Die Standarderklärung der DEKRA Media GmbH ist auch im Internet unter https://www.dekra-safety-web.eu/datenschutz/ abrufbar:

---

**DEKRA respektiert Ihre Privatsphäre**

Der Schutz Ihrer Privatsphäre bei der Verarbeitung der persönlichen Daten sowie die Sicherheit aller Geschäftsdaten ist uns ein wichtiges Anliegen, das wir in unseren Geschäftsprozessen berücksichtigen. Darüber hinaus ist es uns wichtig, dass Sie jederzeit wissen, wann wir welche Daten speichern und wie wir diese verwenden.

**Datenschutz**

Insbesondere der Schutz personenbezogener Daten wie Name, Adresse, Telefonnummer oder E-Mail-Adresse und der verantwortungsvolle Umgang mit den Informationen, die Sie uns anvertrauen, sind uns wichtig. Dazu gehören auch die Daten, die bei der Internetnutzung im Hintergrund anfallen. Werden Sie auf den einzelnen Webseiten um die Angabe persönlicher Informationen gebeten, so werden Sie dort über den beabsichtigten Verarbeitungszweck informiert und, falls erforderlich, um Ihre Zustimmung gebeten.

**Anwendbares Recht**

DEKRA sichert zu, dass Ihre Angaben entsprechend den geltenden deutschen datenschutzrechtlichen Bestimmungen vertraulich behandelt werden.

**Ihr Besuch auf der DEKRA Webseite/Verwendung der Nutzungsdaten**
Auch Ihr Besuch auf unseren Webseiten unterliegt dem Datenschutz. Unsere Webserver speichern standardmäßig den Namen Ihres Internet-Service-Providers, die Webseite, von der aus Sie uns besuchen, die Webseiten, die Sie bei uns besuchen, sowie das Datum und die Dauer des Besuchs. Durch die Aufrufe der Internetseiten erhält DEKRA Nutzungsdaten, die für Sicherungszwecke gespeichert werden. Die Nutzerdaten werden für statistische Zwecke ausgewertet. Die regelmäßige Auswertung der Zugriffsstatistiken der Webseiten zeigt die zeitliche Entwicklung der Besucherresonanz sowie den unterschiedlichen Erfolg verschiedener Angebote auf der Webseite auf. Sie ist unter anderem Basis für die Steuerung der weiteren Entwicklung der Internetseite. Eine Weitergabe von Nutzerdaten an Dritte, weder zu kommerziellen noch zu nicht kommerziellen Zwecken, findet nicht statt.

# A3 Nutzungsbedingungen

Die Nutzungsbedingungen beziehen sich auf das gesamte Projekt *LuPto* und befinden sich derzeit noch in der Entwicklung (nicht Bestandteil dieses Teilprojekts).

Vor dem Gebrauch muss jeder Nutzer der Seite die Nutzungsbedingungen bestätigen.

In diesen Bedingungen wird ausdrücklich darauf hingewiesen, dass sämtliche personenbezogenen Daten gesammelt und ausgewertet werden. Insbesondere die Log-in-Zeiten, sensible Daten wie das Lernverhalten und der aktuelle Wissensstand werden protokolliert; sie stehen jedoch nur dem zugewiesenen Trainer und keiner anderen Person zur Verfügung.

Die Teilnahme an diesem Programm ist absolut freiwillig und dient ausschließlich der Unterstützung der Auszubildenden.

# A4 Quellcode – Auszug

c:\laragon\www\lupto\routes\web.php ➜ wurde ergänzt
Hier werden die Weblinks aus der Browser-Adressleiste abgefangen und verarbeitet, d. h., es wird die zugehörige Controller-Funktion aufgerufen:

```
…

// ==========================================
// Trophy Routes
// ==========================================
Route::get('trophy', 'TrophyController@index')->name('trophy.index');
Route::get('trophy/{award}', 'TrophyController@details')->name('trophy.details')-
>middleware(['setredirect:modal.unauthenticated', 'role:trainee']);;

…
```

c:\laragon\www\lupto\app\Http\Controllers\TrophyController.php

Der Controller ist für die Steuerung des Programms zuständig. In diesem Fall werden die nötigen Daten gesammelt und an die Ausgabe weitergeleitet:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use App\Models\Trophy;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;

class TrophyController extends Controller
{
    public function index() {
        $user = Auth::user();
        $trophies = $user->trophy->sortByDesc('id');
        return view('trophy.index', compact('trophies'));
    }

//  Transfer of the award-details to the declaration-view
    public function details($award) {
        $trophy = Trophy::find($award);
        return view('trophy.declaration', compact('trophy'));
    }
}
```

c:\laragon\www\lupto\resources\views\trophy\index.blade.php

Diese Ansicht zeigt eine kurze Ansprache und eine Auflistung aller bereits erlangten Trophäen:

```
@extends('layouts.lupto')
@section('bgimage_overlay', true)
@section('page_title', 'Ruhmeshalle')

@section('content')
    <div class="gandalf__container">
        <i>
            Hier werden Eure kleinen und großen Lernerfolge dargestellt.
            Ihr habt so einen schnellen Überblick über Eurer Lernverhalten.
            Solltet Ihr im Verlauf Eurer Ausbildung mal ins Straucheln kommen,
            könnt Ihr mit der Unterstützung Eurer Trainer ein auf Euch
            abgestimmtes Übungs-Konzept absprechen, damit Ihr möglichst
            schnell wieder Spaß an Eurer täglichen Arbeit bekommt...
        </i>
    </div>
    <div class="row" >
        @foreach ($trophies as $trophy)
            @include('dashboard.partials.panel',
                ['url' => route('trophy.details', ['award'=>$trophy->id]),
                'title' => $trophy->title,
                'icon' => $trophy->icon])
        @endforeach
    </div>
@endsection
```

c:\laragon\www\lupto\resources\views\trophy\declaration.blade.php

In dieser Ansicht wird die Trophäe erläutert:

```
@extends('layouts.lupto')
@section('bgimage_overlay', true)
@section('page_title', 'Ruhmeshalle')

@section('content')
    <div class="gandalf__container">
        @include('dashboard.partials.panel',
            ['url' => route('trophy.index', ['award'=>'topten']),
            'title' => $trophy->title,
            'icon' => $trophy->icon])
        <i><br><br>{{ $trophy->description }}</i><br>
        @if ($trophy->title == 'Murmel-Award')
            <br>
            <iframe width="427" height="240"
                src="https://www.youtube.com/embed/zlepZJsoYcw"
                frameborder="0" gesture="media"
                allow="encrypted-media" allowfullscreen>
            </iframe>
        @endif
    </div>
@stop
```

c:\laragon\www\lupto\app\Models\Trophy.php

Die Klasse „Trophy" beinhaltet die Methoden zur Überprüfung der Eintrittsbedingungen aller Auszeichnungen. Bei positiver Rückmeldung werden die Auszeichnungen dem aktuellen Benutzer durch die Methode „trophy()" zugewiesen und in der Datenbank hinterlegt:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Auth;
```

```php
use App\Models\User;
use App\Models\Logitem;
use App\Models\BoxProgress;
use App\Models\CardProgress;

class Trophy extends Model
{
    protected $table = 'trophies';
    protected $fillable=[
        'title', 'description', 'icon',
    ];

    /**
     * check if every trophy is assigned, if not check conditions
     */
    public static function assign($user)
    {
        $inactiveCards = BoxProgress::where('user_id', $user->id)
            ->sum('inactive_cards_count');
        $activeCards = BoxProgress::where('user_id', $user->id)
            ->sum('active_cards_count');

        $trophies = Trophy::all();
        foreach ($trophies as $trophy) {
            if ($user->trophy()->find($trophy->id) == null) {
                $help = "check_".$trophy->id;
                if (function_exists($help)) {
                    self::$help($user, $inactiveCards, $activeCards);
                }
            }
        }
        self::check_12($user); //this award is temporary (Murmel)
        return;
    }

    /**
     * check the conditions for Trophy#1 'Round 1'
     * condition: if user has answered to every single question once
     */
    public static function check_1($user, $inactiveCards, $activeCards)
    {
        if ($inactiveCards == null and $activeCards != null) {
            $user->trophy()->attach(['trophy_id' => 1]);
        }
    }


    /**
     * check the conditions for Trophy#2 'Box 1'
     * condition: if user has no active card in box 1
     */
    public static function check_2($user, $inactiveCards, $activeCards)
    {
         $box1=CardProgress::where('user_id', $user->id)
            ->where('repetition_counter', 1)
            ->sum('repetition_counter');
        if ($box1 == null and $inactiveCards == null and $activeCards != null) {
            $user->trophy()->attach(['trophy_id' => 2]);
```

```
        }
    }

    /**
     * check the conditions for Trophy#3 'Box 2'
     * condition: if user has no active card in box 2 or less
     */
    public static function check_3($user, $inactiveCards, $activeCards)
    {
         $box2=CardProgress::where('user_id', $user->id)
            ->where('repetition_counter', '<', 3)
            ->sum('repetition_counter');
        if ($box2 == null and $inactiveCards == null and $activeCards != null) {
            $user->trophy()->attach(['trophy_id' => 3]);
        }
    }

    /**
     * check the conditions for Trophy#4 'Box 3'
     * condition: if user has no active card in box 3 or less
     */
    public static function check_4($user, $inactiveCards, $activeCards)
    {
         $box3=CardProgress::where('user_id', $user->id)
            ->where('repetition_counter', '<', 4)
            ->sum('repetition_counter');
        if ($box3 == null and $inactiveCards == null and $activeCards != null) {
            $user->trophy()->attach(['trophy_id' => 4]);
        }
    }


    /**
     * check the conditions for Trophy#5 'Profile'
     * condition: if user completed his profile
     */
    public static function check_5($user)
    {
        if ($user->email != null and
            $user->gender != null and
            $user->firstname != null and
            $user->lastname != null and
            $user->avatar != null and
            $user->address != null and
            $user->zip != null and
            $user->city != null)
        {
            $user->trophy()->attach(['trophy_id' => 5]);
        }
    }


    /**
     * check the conditions for Trophy#6 'Freetime'
     * condition: if user answered questions after 17:00
     */
    public static function check_6($user)
    {
```

```
        $logons = CardProgress::where('user_id', $user->id)->get();
        foreach ($logons as $logon) {
            $timeToProof = date('Hi', strtotime($logon->updated_at));
            if ($timeToProof > 1700) {
                $user->trophy()->attach(['trophy_id' => 6]);
            }
        }
    }

    /**
     * check the conditions for Trophy#7 'Logins'
     * condition: user logged in for 100 times
     */
    public static function check_7($user)
    {
        $logons = Logitem::where('user_id', $user->id)
            ->where('subtype', 'login')->count();
        if ($user->trophy()->find(7) == null and $logons >= 100) {
            $user->trophy()->attach(['trophy_id' => 7]);
        }
    }

    /**
     * check the conditions for Trophy#8 'Logins'
     * condition: user logged in for 250 times
     */
public static function check_8($user)
    {
        $logons = Logitem::where('user_id', $user->id)
            ->where('subtype', 'login')->count();
        if ($user->trophy()->find(8) == null and $logons >= 250) {
            $user->trophy()->attach(['trophy_id' => 8]);
        }
    }

    /**
     * check the conditions for Trophy#8 'Logins'
     * condition: user logged in for 500 times
     */
public static function check_9($user)
    {
        $logons = Logitem::where('user_id', $user->id)
            ->where('subtype', 'login')->count();
        if ($user->trophy()->find(9) == null and $logons >= 500) {
            $user->trophy()->attach(['trophy_id' => 9]);
        }
    }

    /**
     * check the conditions for Trophy#10 'Communication'
     * condition: if user sent more then 9 messages
     */
    public static function check_10($user)
    {
        $mails = $user->logitem()->where('type', 'message')->count();
        if ($user->trophy()->find(10) == null and $mails >= 10) {
            $user->trophy()->attach(['trophy_id' =>10]);
        }
```

```
    }

    /**
     * check the conditions for Trophy#11 'Genius'
     * condition: if more than 92% of all given answers
     * are correct
     */
    public static function check_11($user)
    {
        $box5 = CardProgress::where('user_id', $user->id)
            ->where('repetition_counter', 5)
            ->count();
        $activeCardsCount = BoxProgress::where('user_id', $user->id)
            ->sum('active_cards_count');
        if ($activeCardsCount != null) {
            $rate = $box5 / $activeCardsCount *100;
            if ($rate >= 92) {
                $user->trophy()->attach(['trophy_id' => 11]);
            }
        }
    }

    /**
     * check the conditions for Trophy#12 'Murmel'
     * condition: if user was logged in on every weekday
     * in the past week
     */
    public static function check_12($user)
    {
        $murmel = null;
        for ($i = 0; $i <= 4; $i++ ) {
            $dateToCheck = date('d.m.Y', strtotime('last monday + '.$i.' days'));
            foreach (Logitem::where('user_id', $user->id)->get() as $logon) {
                $logitem = date('d.m.Y', strtotime($logon->created_at));
                if ( $logitem == $dateToCheck) {
                    $murmel = $murmel + 1;
                    break;
                }
            }
        }
        if ($murmel >= 5) {
            $user->trophy()->attach(['trophy_id' => 12]);
        }
        else {
            $user->trophy()->detach(['trophy_id' => 12]);
        }
        return;
    }
}
```

c:\laragon\www\lupto\app\Models\User.php ➜ wurde ergänzt

Die bestehende~~n~~ Klasse „User" wurde um die Methode „trophy()" erweitert. Hier findet die eigentliche Zuweisung einer erlangten Auszeichnung mittels einer Zwischentabelle statt~~:~~.

```
    /**
     *
     * @return Illuminate\Database\Eloquent\Relations\hasMany
     */
    public function trophy()
    {
       // assign trophy to the user
       return $this->belongsToMany('App\Models\trophy',
            'pivot_user_trophy', 'user_id', 'trophy_id');
    }
```

c:\laragon\www\lupto\app\Providers\EventServiceProvider.php ➜ ergänzt

Dieser Service-Provider ruft beim Login eines Benutzers u. a. den *Listener* „Save Trophies" auf. Später wird die markierte Zeile verschoben, damit die Ausführung zu einer festgelegten Zeit stattfindet:

```
…
class EventServiceProvider extends ServiceProvider
{
    /**
     * The event listener mappings for the application.
     *
     * @var array
     */
    protected $listen = [
        'Illuminate\Auth\Events\Login' => [
            'App\Listeners\SuccessfulLoginListener',
            'App\Listeners\Logbook\SaveLogitemMiscLogin',
            'App\Listeners\SaveTrophies',
        ],
...
```

c:\laragon\www\lupto\app\Listeners\SaveTrophies.php

Hier wird die Zuweisung der Auszeichnungen aufgerufen:

```
<?php

namespace App\Listeners;

use Illuminate\Auth\Events\Login;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Contracts\Queue\ShouldQueue;
```

```php
use Illuminate\Support\Facades\Auth;
use App\Models\User;
use App\Models\Trophy;

class SaveTrophies
{
    /**
     * Create the event listener.
     *
     * @return void
     */
    public function __construct()
    {
        //
    }

    /**
     * Handle the event.
     *
     * @param  Login  $event
     * @return void
     */
    public function handle(Login $event)
    {
        $user = $event->user;
        Trophy::assign($user);
    }
}
```

# A5 Migrations-Dateien – Auszug

Die Objekte der Klasse „Trophy" und deren Attribute werden in der Tabelle „trophies" gespeichert. Der für die Erzeugung der Tabelle nötige Quellcode ist in folgender Datei hinterlegt:

c:\laragon\www\lupto\database\migrations\2018_03_29_090534_create_trophies_table.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTrophiesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('trophies', function (Blueprint $table) {
            $table->increments('id');
            $table->string('title');
            $table->text('description');
            $table->string('icon');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void     */
    public function down()
    {
        Schema::dropIfExists('trophies');
    }
}
```

Zwischen den Benutzern und den Auszeichnungen besteht eine n:m-Beziehung (viele Nutzer können viele Auszeichnungen erlangen). Damit diese Daten abgespeichert werden können, wird eine Zwischentabelle erzeugt:

c:\laragon\www\lupto\database\migrations\2018_03_29_105356_create_pivot_users_trophies_table.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePivotUsersTrophiesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('pivot_user_trophy', function (Blueprint $table) {
            $table->integer('user_id');
            $table->integer('trophy_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('pivot_users_trophies');
    }
}
```

# A6 Benutzerdokumentation – Auszug

In diesem Auszug aus der Benutzerdokumentation sind Bildschirmfotografien aller möglichen Auszeichnungen aufgelistet:


*Und täglich güßt das Murmeltier.... Du hast dich in der letzten Woche täglich eingeloggt*



*Aufgrund deiner Lern-Ergebnisse bist du ein Einser-Kandidat. Die Prüfungen kannst du somit selbstbewust auf dich zukommen lassen...*


*Auch Kommunikation gehört zu deinem Berufsbild. - Du hast bisher mehr als 10 Nachrichten mit deinem Trainer ausgetauscht... Nur keine Hemmungen - Weiter so!!!*


*Du hast dich an 500 Tagen eingeloggt*


*Du hast dich an 250 Tagen eingeloggt*


*Du hast dich an 100 Tagen eingeloggt*


*Ehrgeiz hat keinen Feierabend - Du hast auch außerhalb deiner normalen Arbeitszeit gelernt... BRAVO*


*Vollständig ausgefülltes Profil - Danke für deine Mitarbeit*


*Nachdem nun auch das dritte Fach leer ist, sollte das Lernen immer leichter fallen... WEITER SO*


*Auch das zweite Fach in deiner Lernkartei ist leer - GOOD JOB*


*Keine Karten mehr im ersten Karteifach - Ab hier musst du nur noch das Wissen vertiefen*


*Du hast die erste Runde geschafft! - Alle Fragen mindestens einmal beantwortet*

# A7 Tabellen-Modell