13.11.2019

Johannes Meyerhoff, Azubi-Nr.: 5002305256

Fachinformatiker fÜR Anwendungsentwicklung

GESIS Leibniz institut für sozialwissenschaften

UNTER SACHSENHAUSEN 6-8 50667 KÖLN

Erweiterung des Quartalsdaten-Erhebungs-Systems (QES) um die Funktionalität, erhobene Datensätze zu exportieren.

Inhaltsverzeichnis

1 Vorwort 2

2 Projektbeschreibung 3

2.1Projektumfeld und technische Infrastruktur 3

2.2Ausgangssituation 3

2.3Zielsetzung 3

3 Projektüberblick 5

3.1Vorgehensmodell XP 5

3.2Zeitaufwand 6

3.3Kosten 7

3.4Sachmittel 7

4 Projektphasen 8

4.1Planungsphase 8

4.1.1 Realisierungskonzept 8

4.2Vorbereitende Aufgaben 9

4.2.1 Testumgebung einrichten 9

4.3Programmrealisierung 9

4.3.1 Erstellung Model 9

4.3.2 Erstellung Controller 9

4.3.3 Erstellung View 9

4.4Qualitätssicherung 10

4.4.1 Unit-Tests 10

4.4.2 Feature-Tests 10

4.4.3 Performancetest 10

4.4.4 Fehlerbeseitigung 10

4.5Abnahme und Dokumentation 10

4.5.1 Erstellung der Anwenderdokumentation 10

4.5.2 Erstellung der Projektdokumentation 10

4.5.3 Übergabe und Abnahme 10

5 Probleme und Lösungen 11

5.1Erstellung des Controllers 11

6 Ergebnisse und Fazit 11

7 Anhang 11

7.1A1 Glossar 11

7.2A2 Informationsquellenverzeichnis 11

7.3A3 Abbildungsverzeichnis 11

7.4A4 Tabellenverzeichnis 11

7.5A5 Anlagenverzeichnis 11

# Vorwort

Diese Projektdokumentation beschreibt die Durchführung meines Projekts zur Erweiterung eines bestehenden Systems um die Funktionalität, Daten in ein spezifisches Excel-Format zu exportieren. Das Projekt habe ich im Rahmen der IHK Ausbildung zum Fachinformatiker für Anwendungsentwicklung in meinem Ausbildungsbetrieb GESIS durchgeführt. Meine Haupttätigkeit als Auszubildender bei der GESIS ist die Entwicklung des QES (**Q**uartalsdaten-**E**rhebungs-**S**ystem).

GESIS - Leibniz-Institut für Sozialwissenschaften mit Sitz in Köln und Mannheim (Hauptsitz) ist die größte deutsche Infrastruktureinrichtung für die Sozialwissenschaften.

Das Institut ist in fünf wissenschaftliche Abteilungen, die mit ihrem forschungsbasierten Service- und Produktangebot den Forschungsprozess der empirischen Sozialforschung in seiner gesamten Breite abdecken, unterteilt. Die Abteilung WTS (Wissenstechnologien für Sozialwissenschaften), für die ich arbeite, unterstützt die sozialwissenschaftlichen Fachabteilungen durch die Entwicklung innovativer Wissenstechnologien zur Verbesserung der Integration und Nutzerfreundlichkeit der digitalen GESIS-Angebote.

In dieser Dokumentation werden Abkürzungen und Fachbegriffe genutzt, welche über das Grundwissen der Informatik hinausgehen. Sofern diese nicht im Kontext erläutert werden, werden diese bei der erstmaligen Verwendung durch eine orangene Farbgebung gekennzeichnet und im Glossar genauer erläutert.

# Projektbeschreibung

## Projektumfeld und technische Infrastruktur

Die Entwicklung erfolgt am Sitz der GESIS in Köln. Der virtuelle Server für die Produktiv-Version von QES ist auf das interne Firmennetz beschränkt. Ich als Auszubildender, mein Ausbilder und die IT-Abteilung haben administrative Rechte. Die MySQL-Datenbank für QES (**Q**uartalsdaten-**E**rhebungs-**S**ystem) ist nur von dem Server, auf dem die Anwendung läuft, erreichbar. Die Anwendung QES läuft auf einem virtualisierten Ubuntu 16 System. Auf meinem GESIS-Entwicklungs-Computer ist auch eine Testumgebung mit gleichen Versionen vorhanden. Diese wird jedoch nicht mit VMware, sondern mit Oracle VirtualBox virtualisiert.

Die Entwicklungsschritte werden im Firmeninternen Git versioniert. QES ist mit dem Laravel PHP Framework nach dem Model-View-Controller Prinzip aufgebaut und verwendet Laravel in der Version 5.4 und PHP mit Version 7.2.14. Für einige Funktionen sind bereits automatisierte Tests vorhanden. Hierzu wird die in Laravel 5.4 mitgelieferte Version 5.7.27 von PHPUnit verwendet.

Werkzeuge zur Entwicklung sind Microsoft Visual Studio Code, iTerm2, GitLab, phpMyAdmin, Mozilla Firefox und Google Chrome.

Auftraggeber für das Projekte ist der ehemalige kommissarische Abteilungsleiter der Abteilung WTS. Ansprechpartner bei technischen oder organisatorischen Problemen ist mein Ausbilder Sascha Schüller. Fachlicher Ansprechpartner bei geschäftsprozessbezogenen Rückfragen ist Thomas Knecht.

## Ausgangssituation

Die Leitung der Abteilung WTS erfasst in einem auf Excel-Tabellen basierenden Verfahren die Auslastung der Mitarbeiter im Bezug auf Projekte. Das aktuelle Verfahren basiert auf einer für jeden Mitarbeiter gleichen Excel Tabelle, welche erst unausgefüllt an den Mitarbeiter gesendet, und danach zurückgesendet wird. Dieses Verfahren zeichnet sich durch große Ineffizienz aus. Die Tabelle hat ca. 92 Zeilen und 15 spalten je Worksheet. Hiervon werden jedoch nur ca. 4 bis 15 Zellen durch den jeweiligen Mitarbeiter gefüllt, dies entspricht ca. 1% der gesamten Datei. Das neue System QES kann erst mit der in diesem Projekt entwickelten Export-Funktionalität vollständig in den Arbeitsablauf integriert werden. Erst mit dieser Funktion kann damit das alte Verfahren abgelöst werden und somit Zeit und damit Geld gespart werden.

## Zielsetzung

Um Personal richtig planen und führen zu können ist es notwendig die reale Auslastung und den tatsächliche Ressourcenverbrauch in jedem Projekt zu kennen. Eine Übersicht darüber ist die entscheidende Grundlage für die weitere Ressourcenplanung der gesamten Abteilung. Ziel des Projektes ist es, dem Abteilungsleiter und auch den Teamleitern die Möglichkeit zu geben, einen Zeitraum aus den Bestehenden Datensätzen auszuwählen und so Abweichungen von der Soll-Planung zu erkennen.

Im Detail soll im QES für die Teamleiter und den Abteilungsleiter ein neuer Menüpunkt ergänzt werden. Der Menüpunkt Export soll auf eine Seite führen, auf der der Export definiert und angefordert wird. Der Ablauf ist wie folgt geplant: Der Abteilungsleiter oder die Teamleiter wählen einen Zeitraum, anhand dieser Auswahl wird eine Datei erstellt. Dabei erhalten Teamleiter und Abteilungsleiter gemäß ihrer Rolle unterschiedliche Ergebnisse. Die Ergebnis-Dateien stellen den Ressourcenverbrauch pro Mitarbeiter und Projekt dar. Je Mitarbeiter wird je Projekt eine eigene Zeile mit dem Soll- und dem Ist-Wert angelegt. Autorisierte Nutzer können dann auf die Export-Dateien für eine bestimmte vorab festgelegte Zeit zugreifen. Verarbeitung, Konvertierung und Export der Daten auf dem Server müssen konzipiert, implementiert und getestet werden. Leitbild für Entwurf und Umsetzung ist das Model View Controller Konzept.

# Projektüberblick

## Vorgehensmodell XP

Als Vorgehensmodell habe ich Extreme Programming nach Kent Beck (kurz XP, siehe auch [XPWIK]) gewählt, da es beispielsweise im Vergleich zu Scrum weniger Rollen und Meetings benötigt. Extreme Programming hat außerdem einen iterativen Ansatz, der sehr schnell erste Ergebnisse produziert. Extreme Programming ist ein agiles Vorgehensmodell, welches ursprünglich dazu entwickelt wurde, die Kosten von Änderungen im laufenden Entwicklungsprozess gering zu halten. Extreme Programming verwendet das Konzept des Test Driven Developments (TDD). Das bedeutet, dass zur Entwicklung von Funktionen oder Funktionalitäten Tests geschrieben werden, welche bei korrekter Programmierung der Funktion erfolgreich durchlaufen werden. Die Tests werden für Laravel meist in PHPunit geschrieben. Auf den Pair-Programming Ansatz des Extreme Programmings wurde verzichtet, da ich alleine für die Umsetzung der Programmierung verantwortlich war.

Die in Abbildung 1 aufgezeigten Phasen des Vorgehensmodells können je nach Bedarf und Kompetenzen angepasst werden.



Abbildung : Phasen des Vorgehensmodells Extreme Programming

## Zeitaufwand

Die zeitliche Planung mithilfe von Arbeitspaketen und Meilensteinen tabellarisch in Tabelle 1 zu sehen. Arbeitspakete sind möglichst atomar gehalten und logisch gruppiert. Hierbei wurde jedoch nicht zwingend AP1 vor AP2 durchgeführt. Der Menüpunkt aus AP E4 wurde beispielsweise entworfen bevor die Eigenschaften des Models festgelegt wurden. Der Controller aus AP E4 wurde jedoch danach entworfen.

Die Durchführung fand im Zeitraum vom 07.10.2019 bis 12.11.2019 statt. Im gesamten Zeitraum wurden 68 Stunden an dem Projekt gearbeitet.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phase | Arbeitspaket | Name | Soll in Stunden | Ist in Stunden |
| Projektinitialisierung | AP PI1 | Kick-off Meeting mit dem Kunden durchführen | 0,5 | 0,5 |
|  | AP PI2 | Projektinitiierung/ -definition abschließen | 0,5 | 0,5 |
| Projektplanung | AP P1 | Risiken einschätzen und Kosten betrachten | 1,5 | 1 |
|  | AP P2 | Benötigte Funktionalitäten für Controller ermitteln | 0,5 | 1 |
|  | AP P3 | Verfügbare Komponenten und Lösungen am Markt untersuchen | 2 | 2 |
|  | AP P4 | Arbeitspakete planen | 2 | 2 |
|  | AP P5 | Projektphasen, Tätigkeiten, Termine, Meilensteine, Zeiten,Projektablauf , Ressourcen, Kosten, Nutzen planen | 3 | 4 |
|  | AP P6 | Tests anlegen um Qualität zu planen | 2 | 2 |
| Projektdurchführung  Analyse | AP A1 | Machbarkeitsstudie durchführen | 2 | 2 |
| Entwurf | AP E1 | Aus dem Lastenheft das Pflichtenheft erstellen | 2 | 2 |
| Entwurf | AP E2 | Eigenschaften des Models festlegen | 1 | 1 |
| Entwurf | AP E3 | Formular mit Auswahlmöglichkeit zur Generierung entwerfen | 4 | 4 |
| Entwurf | AP E4 | Menüpunkt und Controller entwerfen | 4 | 4 |
| Entwurf | AP E5 | Download von Excel-Fähiger Datei konzipieren | 1 | 1 |
| Entwurf | AP E6 | Layout mit Kunde absprechen | 1 | 1 |
| Implementierung | AP I1 | Route zum Formular und zum Download anlegen | 1 | 1 |
| Implementierung | AP I2 | View-Oberfläche entwickeln und mit Kunde abstimmen | 2 | 2 |
| Implementierung | AP I3 | Mapping implementieren | 1 | 1 |
| Implementierung | AP I4 | Controller erstellen | 3 | 4 |
| Implementierung | AP I5 | View anlegen | 3 | 3 |
| Implementierung | AP I6 | Funktion implementieren, die die angefertigte Datei als Download bereitstellt | 5 | 4 |
| Implementierung | AP I7 | Qualitätssicherung durch automatisierte Tests | 4 | 4 |
| Testphase mit Nutzertests | AP T1 | Funktionalität | 2 | 2 |
|  | AP T2 | Sicherheit | 2 | 2 |
|  | AP T3 | Integrität | 2 | 2 |
| Dokumentation |  | Dokumentation erstellen | 2 | 2 |
| Projektabschluss |  | Abnahme | 2 | 2 |

Tabelle 1: Aufwand mit Soll/Ist in Stunden

## Kosten

Kosten des Projekts setzen sich aus den Personalkosten und den Kosten für Sachmittel zusammen. Die genutzten Sachmittel sind jedoch bereits in den Personalkosten enthalten. Der interne Verrechnungssatz für einen Entwickler beträgt bei GESIS 60€. Die Kostenaufstellung kann Tabelle 2 entnommen werden.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Beschreibung | Kosten pro Stunde | Stunden | Gesamtkosten |
| Johannes Meyerhoff | Projektleiter  Entwickler | 60€ | 68 | 4080€ |
| Sascha Schüller | Ansprechpartner | 60€ | 4 | 240€ |
| **Summe** |  |  | **72** | **4320€** |

Tabelle 2: Kostenaufstellung

## Sachmittel

In Tabelle 3 sind die zur Umsetzung des Projektes verwendeten Software- und Hardware- Komponenten dargestellt.

|  |  |  |
| --- | --- | --- |
| Name | Art | Menge |
| BenQ BL2411 | Bildschirm | 2 |
| Dell Optiplex 7040 | Computer | 1 |
| Lenovo 54Y9414 | Tastatur | 1 |
| Logitech MX Master 2 | Maus | 1 |
| Windows 10 Enterprise | Betriebssystem | 1 |
| MS VS Code | Entwicklungsumgebung | 1 |

Tabelle 3: Sachmittel

# Projektphasen

## Planungsphase

### Realisierungskonzept

Mithilfe der in GitHub vorhandenen Projektansicht habe ich ein Kanban Board erstellt. In meinem Kanban Board sind alle Arbeitspakete hinterlegt. Sie sind unterteilt in drei vertikal getrennte Sektionen. Links ist der „Pool“ der möglichen Aufgaben, in der Mitte sind die Tickets, die aktuell in Bearbeitung sind und auf der rechten Seite sind die fertigen Aufgaben abgelegt.

Jedes Arbeitspaket ist als Ticket hinterlegt und einem Meilenstein zugeordnet. Jedes Ticket ist einem Meilenstein zugeordnet, der mehrere Tickets umfassen kann. Abbildung 2 zeigt einen beispielhaften Zwischenstand.

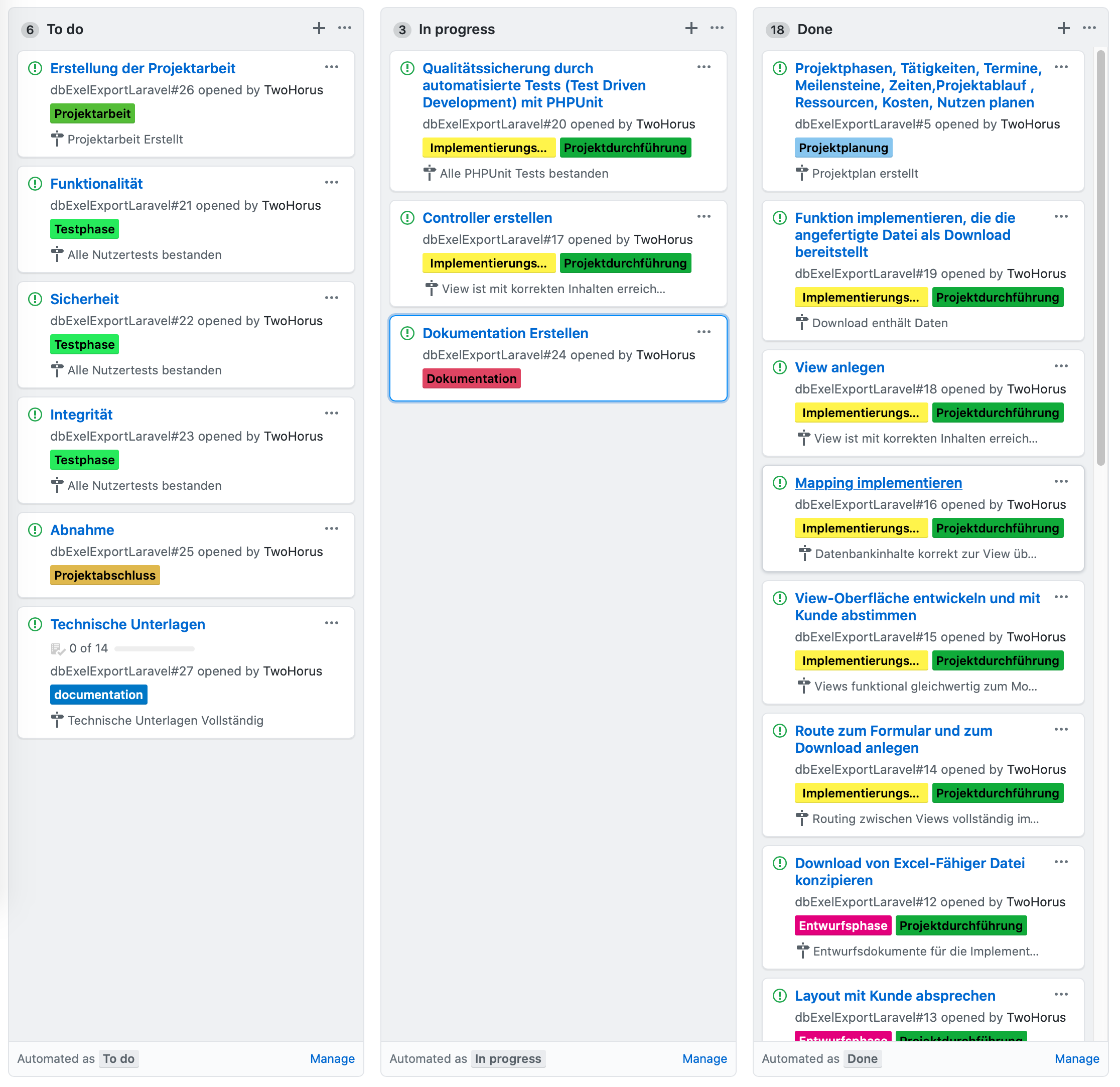


Abbildung : Kanban Board zur Planung

## Vorbereitende Aufgaben

### Testumgebung einrichten

Ich habe mit Bash einige Shell-Scripte zum Starten meiner Tests erstellt. Wenn ich nun ein Haupt-Skript ausführe, wird die Test-Datenbank neu erstellt und ihre Integrität mithilfe der im Projekt hinterlegten Modelle geprüft. Das bedeutet, es wird getestet, ob der Mitarbeiter, der sich hinter „isworkerid“ verbirgt, auch einen Eintrag mit der ID in der Tabelle „worker“ hat. Ist dies nicht der Fall, so wird auch direkt im Terminal, in dem das Skript läuft, eine Fehlermeldung angezeigt. Nach vollständiger Migration öffnet sich eine Entwickleransicht im Browser. Hier können für die Entwicklung nützliche Informationen und Formulare hinterlegt werden.

## Programmrealisierung

Das Model-View-Controller-Konzept (kurz MVC, siehe auch [MVCWIK]) wurde 1979 zunächst für Benutzeroberflächen in der Programmiersprach Smalltalk durch Trygve Reenskaug beschrieben. Reenskaug arbeitete an der Weiterentwicklung von Smalltalk. MVC gilt mittlerweile als De-facto-Standard für den Grobentwurf vieler komplexer Softwaresysteme. Für mein Projekt war dieses Konzept ein wichtiger Ansatz, um schnell zu einer sinnvollen Architektur zu kommen. Das Framework Laravel baut auch auf dem Model View Controller Konzept auf. Dementsprechend verwende ich das gleiche Grundkonzept wie mein Framework.

### Erstellung Model

Ich habe mich für die Verwendung der Bestehenden Marktlösung von „Maatwebsite/Laravel-Excel“ zum Export nach Excel entschlossen, da dies die einzige gut dokumentierte Exportbibliothek für Laravel ist. Es gibt eine weitere Bibliothek, „rap2hpoutre/fast-excel“. Dies ist eine schlankere Version von „Maatwebsite/Laravel-Excel“. In der Dokumentation von fast-excel wird zur Nutzung des Laravel-Excel Pakets geraten, wenn man mehr als die Grundfunktionen (beispielsweise farbliche Markierungen) verwenden möchte.

Genutzt wird die Bibliothek von Maatwebsite, da es mit enormen Kosten verbunden wäre die Bibliothek selbstständig zu programmieren. Eine Selbständige Programmierung einer Bibliothek auf Basis von php-excel ist möglich, jedoch kann ich die fertige Bibliothek verwenden, ohne selber den Entwicklungsaufwand erneut zu betreiben, da „Maatwebsite/Laravel-Excel“ php-excel als Basis nutzt.

Um die Daten richtig zu adressieren musste ich ein Modell für jede Tabelle der Datenbank erstellen, die für das Exportvorhaben relevant ist. Abbildung 3 zeigt das relationale Datenbankmodell.

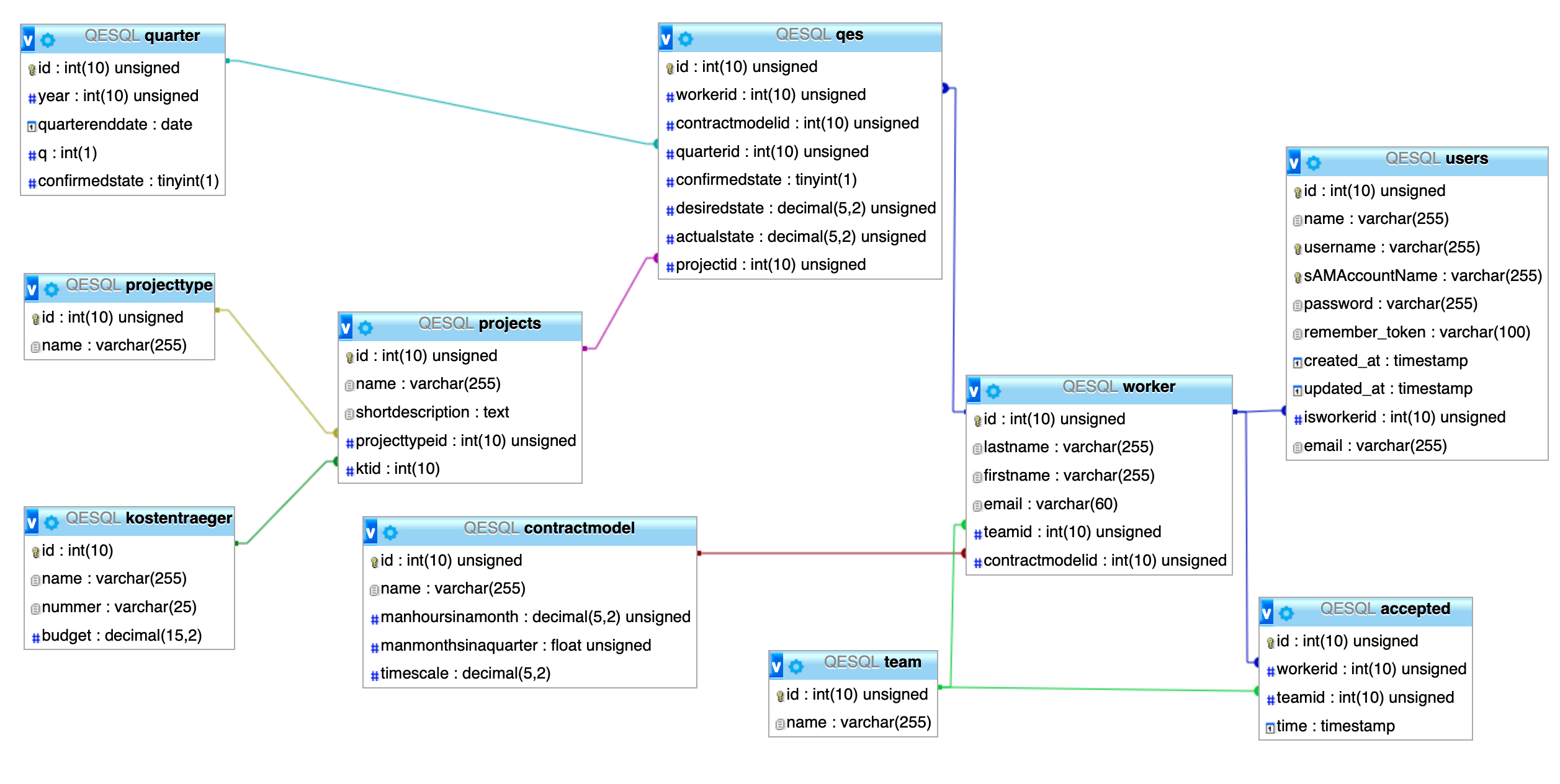


Abbildung : ERD QES

WAS DATEN

WAS NUTZEN

WAS VERKNÜPFUNGDie Datenbank schlüssel auf QES VERKNÜPFT PROJEKTE DADURCH LÄSST SICH SCHLIESSEN VON AUF!!!!!!

Die Modelle, die ich erstellt habe, enthalten Informationen über die Daten, mit denen die Anwendung arbeitet. Tabellennamen sind in einer Variablen angegeben, Attribute und die Bedingungen, welche an diese Attribute gebunden sind, werden in einem Array angegeben. Fremdschlüssel werden über die Beziehung angegeben. Die Beziehung zwischen zwei Modellen wird in beiden Modellen als Funktion hinterlegt, welche bei der Beziehung zwischen QES und Projects wie in Abbildung 4 gezeigt aus dem Namen der Beziehung („project“), der Kardinalität der Beziehung („belongsTo“) und der Referenz (in grün) besteht. Die Referenz besagt hierbei, dass das Modell „App\Project“ in der in der Spalte „projectid“ der hinterlegten Tabelle referenziert wird. Dadurch kann man die Daten über das Laravel ORM ansprechen. Ein Object-Relational Mapping hilft dem Anwendungsentwickler, Datenbankabfragen von der Datenbank zu abstrahieren, sodass von SQLite zu mySQL gewechselt werden kann, ohne die Abfragen ändern zu müssen.



Abbildung : Beziehungsdefinition der Modelle

### Erstellung Controller

Der Controller nimmt in Laravel die Anfrage vom Routing entgegen. Hierbei kann man die Parameter einer Anfrage mit Laravel validieren und mit Parametern angeben, ob der Parameter ignoriert wird, sofern er nicht das richtige Format hat, oder man die Anfrage zurückweist. Letzteres Benennt Laravel „bail“. Der Controller nimmt einen Aufruf des Nutzers als erste Anfrage entgegen, um dem Nutzer eine Liste an möglichen Optionen in einem Formular aufzuzeigen. Die Optionen sind in einer Select-Box aufgelistet. Somit ist eine fehlerhafte Eingabe seitens des Nutzers nur durch Manipulation des HTML-Formulars möglich. Nachdem der Nutzer einen Zeitraum für den Export gewählt hat, werden ihm die Informationen als Download bereitgestellt. Den Download habe ich mithilfe der „FromView“-Eigenschaft über die Bibliothek zum Export realisiert. Hierfür musste ich eine Datenbankabfrage formulieren, welche die benötigten Daten aus der Datenbank in einem Objekt ablegt, welches ich vom Controller an die View weitergebe. Die View muss dann so über das Objekt iterieren, dass die Daten im richtigen Format in einer HTML-Tabelle landen, welche vom Parser nach Excel formatiert werden kann.

Hierbei schaue ich in jeder Iteration wie sich das aktuelle Element verändert hat, um festzustellen, ob der Datensatz zu einem neuen Mitarbeiter gehört oder nicht. Wenn dieser zum gleichen Mitarbeiter gehört prüft die View, ob der Datensatz auch zum gleichen Projekt gehört. In einer frühen Version des Programms hatte ich diese Formatierungsfunktionalität in der View, habe jedoch gemerkt, dass auch die Filterung und Sortierung im Controller geschehen sollte. Ich erstelle im Controller ein großes Array aus mehreren Objekten. Jede Zeile ist ein Objekt, es werden so viele Daten übergeben, wie nötig, aber so wenige wie möglich. Das bedeutet, dass ich Informationen wie den Namen des Mitarbeiters nur mitgebe, wenn er relevant ist (ausgegeben wird). Der Code für die Logik zum Erstellen des Arrays ist im Kommentierten Quellcode im Anhang enthalten

### Erstellung View

Die View habe ich so erstellt, dass sie eine einzige große Tabelle anzeigt. Bei einem neuen bzw. anderen Projekt wird eine neue Zeile ausgegeben - bei einem neuen bzw. anderen Mitarbeiter wird der Name des Mitarbeiters am Anfang der Zeile ausgegeben. Bei Änderung des Projekts wird bei gleichbleibendem Mitarbeiter eine Einrückung dort eingefügt, wo dessen Name und Team sonst stehen würden. Die Ausgabe der Zeilen habe ich in eine Funktion „printCurrentRow“ ausgelagert. Die Einrückung habe ich in eine Funktion „indent“ ausgelagert. „Indent“ wird mit einem Parameter aufgerufen, welcher angibt, wie viele leere Zellen ausgegeben werden sollen. So werden bei jedem neuen Mitarbeiter sein Name, seine Abteilung und andere mitarbeiterspezifische Informationen ausgegeben. Bei jedem weiteren Projekt für denselben Mitarbeiter bleibt der Platz für diese Informationen frei und es werden nur die Soll und Ist Datensätze in den zugehörigen Spalten für die Zeiträume ausgegeben.

### Anpassung Zeilenausgabe

Die ausgelagerte Funktion wird für Laravel in der Datei „APP/helpers.php“ implementiert. Laravel sorgt dann per „Bootstrapping“ dafür, dass die in der Datei deklarierten Funktionen innerhalb der Anwendung aufrufbar sind.

Die Ausgabe der Zeile war schwierig zu implementieren, da durch die Menge an Daten auch mit gut formatiertem Code der Programmfluss nicht direkt ersichtlich ist. Im Fazit wird auf diese Problemstellung noch einmal eingegangen.

## Qualitätssicherung

### Unit-Tests

Die Unit-Tests habe ich mit PHPUnit geschrieben, da dieses Test-Framework sehr gut in Laravel integriert ist und in den Tests mit den Models gearbeitet werden kann. Ein Beispiel für einen einfachen Unit-Test ist, die Erreichbarkeit einer Route mit der HTTP GET Methode zu prüfen.

### Feature-Tests

Feature Tests lassen sich auch mit PHPUnit umsetzen. Da ich jedoch einen Download einer Excel-Datei bereitstelle und die Inhalte einer Excel-Datei nicht von PHPUnit geprüft werden können, habe ich eine Test-View erstellt. Von dieser Test-View aus kann ich ein Jahr auswählen und mir die View anzeigen lassen, welche zum Erstellen der Datei verwendet wird oder die Datei herunterladen. Die View ist für den Nutzer irrelevant, kann jedoch mit PHPUnit geprüft werden. Somit muss ich nur die heruntergeladene Datei mit der View vergleichen und die Inhalte der View prüfen. Wenn die Datei der View gleicht, ist es implizit auch so, dass die richtigen Inhalte in der View auch in der Datei richtig sind.

### Performancetest

Bei realistischen Datenmengen, das heißt unter 1Mb habe ich im Test keine signifikanten Unterschiede bei den Antwortzeiten feststellen können. Dies habe ich mit bis zu 20000 Datensätzen getestet, wobei die Zeit zwischen Aufruf der Export-Funktion und Anfang des Downloads zwischen 0,5 und 1 Sekunde lag. Die Zeit, welche der Download benötigt ist natürlich von der Größe des ausgewählten Datensatzes abhängig, jedoch ist die Verbindung innerhalb des Firmennetzes schnell genug um die Datei so schnell herunterzuladen, dass der Download fertig ist, bevor der Nutzer die Downloadübersicht des Browsers öffnen kann um einen Fortschrittsbalken zu sehen.

### Fehlerbeseitigung

Mein Test-Framework PHPUnit gibt, wenn Fehler aufgetreten sind, die Informationen zum Fehler direkt in der Konsole aus. Wenn ich also eine „Method not allowed“ Exception erhalte besteht die Fehlerbeseitigung darin, zu prüfen ob der Test eine falsche Methode testet – Beispielsweise GET auf eine Funktion welche nur POST Anfragen entgegennimmt – oder ob die Funktion richtig aufgerufen wird aber der Fehler in der Implementierung der Funktion liegt. Die Tests haben mir auch geholfen, wenn eine Variable in der Datenbankabfrage noch eingelesen werden muss, welche ich in der Ausgabe schon verwende. Viele „Fehler“ wurden auch schon während der Programmierung vom PHP-Parser in meiner IDE bemängelt und direkt dort behoben.

## Abnahme und Dokumentation

### Erstellung der Anwenderdokumentation

Für den Auftraggeber, der bereits mit dem Basissystem vertraut ist, habe ich eine Dokumentation erstellt, welche aufzeigt, wie der Nutzer die Anwendung aufrufen kann und wie er innerhalb der Anwendung Daten exportiert. Auch der Aufbau und die Bedeutung der Farbgebung wird erklärt. Dies ermöglicht dem Anwender eine einfache Handhabung des Systems.

### Erstellung der Projektdokumentation

Im Rahmen der IHK-Abschlussprüfung zum Fachinformatiker für Anwendungsentwicklung habe ich eine Projektdokumentation erstellt.

### Übergabe und Abnahme

Die Übergabe fand am 07.11.2019 statt, die Abnahme wurde am 13.11.2019 bestätigt.

# Probleme und Lösungen

## Erstellung des Controllers

Der Controller ist sonst für alle Daten und den Datenfluss zuständig. Um jedoch die Daten iterativ in Reihen auszugeben musste ich die Datenbankabfrage so formulieren, dass die Sortierung nach Mitarbeiter und je Mitarbeiter nach Projekt durchgeführt wird.

# Ergebnisse und Fazit

Die zeitlichen Ansprüche an mich als Projektleiter und Entwickler waren schwer vorherzusehen, da ich bisher nur an viel größeren Projekten oder viel kleineren Teilaufgaben beteiligt war. Das Projekt war lehrreich für mein Verständnis des Projektmanagements und den Verantwortungen eines Projektleiters. Den Programmierungs-Teil des Projekts hätte ich mir leichter machen können, indem ich meine Zeit in der Planungsphase anders verteilt hätte. Die Planung für Menüpunkte und Modelle hat die Implementierung der betroffenen Komponenten unwesentlich schneller gemacht (im Verhältnis zu früheren Erfahrungswerten). Diese Zeit hätte ich zur Planung der Zeilenausgabelogik verwenden müssen, da diese doch schwieriger zu implementieren war, als es anfangs schien.

# Anhang

## A1 Glossar

## A2 Informationsquellenverzeichnis

[MVCWIK] https://en.wikipedia.org/wiki/[Model\_View\_Controller](https://de.wikipedia.org/wiki/Model_View_Controller)

[XPWIK] https://en.wikipedia.org/wiki/Extreme\_programming

## A3 Abbildungsverzeichnis

[Abbildung 1: Phasen des Vorgehensmodells Extreme Programming 6](#_Toc24330762)

[Abbildung 2: Kanban Board zur Planung 10](#_Toc24330763)

[Abbildung 3: ERD QES 12](#_Toc24330764)

[Abbildung 4: Beziehungsdefinition der Modelle 13](#_Toc24330765)

## A4 Tabellenverzeichnis

## A5 Anlagenverzeichnis