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 1

2 Projektbeschreibung 2

2.1 Projektumfeld und technische Infrastruktur 2

2.2 Ausgangssituation 2

2.3 Zielsetzung 3

3 Projektüberblick 3

3.1 Vorgehensmodell XP 3

3.2 Zeitaufwand 4

3.3 Kosten 6

3.4 Sachmittel 7

4 Projektphasen 8

4.1 Planungsphase 8

4.1.1 Realisierungskonzept 8

4.2 Vorbereitende Aufgaben 8

4.2.1 Testumgebung einrichten 8

4.3 Programmrealisierung 8

4.3.1 Erstellung Model 9

4.3.2 Erstellung Controller 11

4.3.3 Erstellung View 11

4.3.4 Anpassung Zeilenausgabe 12

4.4 Qualitätssicherung 12

4.4.1 Unit-Tests 12

4.4.2 Feature-Tests 12

4.4.3 Performancetest 12

4.4.4 Fehlerbeseitigung 13

4.5 Abnahme und Dokumentation 13

4.5.1 Erstellung der Anwenderdokumentation 13

4.5.2 Erstellung der Projektdokumentation 13

4.5.3 Übergabe und Abnahme 13

5 Probleme und Lösungen 14

5.1 Erstellung des Controllers 14

5.2 Formatierung der Daten 14

6 Ergebnisse und Fazit 14

7 Anhang 15

7.1 A1 Informationsquellenverzeichnis 15

7.2 A2 Abbildungsverzeichnis 15

7.3 A3 Anlagenverzeichnis 15

# Vorwort

Diese Projektdokumentation beschreibt die Durchführung meines Projekts zur Erweiterung des **Q**uartalsdaten-**E**rhebungs-**S**ystems 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 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.

# 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, Peter Mutschke, 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 in Bezug auf Projekte. Das aktuelle Verfahren basiert auf einer für jeden Mitarbeiter gleichen Excel Tabelle, welche erst unausgefüllt an den Mitarbeiter gesendet, und danach ausgefüllt 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. vier 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 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. Beispielsweise die täglichen Stand Up Meetings habe ich aufgrund der Tatsache, dass ich alleine programmiere, nicht durchgeführt.



Abbildung 1: Phasen des Vorgehensmodells Extreme Programming Quelle XPWIK

## Zeitaufwand

Die zeitliche Planung mit Hilfe von Arbeitspaketen und Meilensteinen ist 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 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. Die Summe setzt sich zusammen aus 57 Stunden für das Projekt (Tabelle 1) und 11 Stunden für die Erstellung dieses Prozessorientierten Projektberichts.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phase | Arbeitspaket | Name/Bezeichnung | 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

|  |  |  |  |
| --- | --- | --- | --- |
| NR | Phase | Meilenstein | Arbeitspaket(e) |
| M1 | Projektinitialisierung | Lastenheft | PI1 PI2 |
| M2 | Projektplanung | Risiko/Kostenbetrachtung erstellt | P1 |
| M3 | Projektplanung | Struktogramm/Interaktionsübersichtsdiagramm für Controller erstellt | P2 |
| M4 | Projektplanung | Marktübersicht verfügbarer Lösungskomponenten erstellt | P3 |
| M5 | Projektplanung | Projektplan erstellt | P4 P5 P6 |
| M6 | Projektdurchführung | Machbarkeit bestätigt | A1 |
| M7 | Projektdurchführung | Pflichtenheft erstellt | E1 |
| M8 | Projektdurchführung | Entwurfsdokumente für die Implementierung erstellt | E2 E3 E4 E5 |
| M9 | Projektdurchführung | Layout abgenommen | E6 |
| M10 | Projektdurchführung | Routing zwischen Views vollständig implementiert | I1 |
| M11 | Projektdurchführung | Views funktional gleichwertig zum Mockup, der mit dem Kunden besprochen wurde | I2 |
| M12 | Projektdurchführung | Datenbankinhalte korrekt zur View übertragen | I3 |
| M13 | Projektdurchführung | View ist mit korrekten Inhalten erreichbar | I4 I5 |
| M14 | Projektdurchführung | Download enthält Daten | I6 |
| M15 | Projektdurchführung | Alle PHPUnit Tests bestanden | I7 |
| M16 | Testphase | Alle Nutzertests bestanden | T1 T2 T3 |

Tabelle 2: Meilensteine und zugehörige Arbeitspakete

## 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 3 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 3: Kostenaufstellung

## Sachmittel

In Tabelle 4 sind die zur Umsetzung des Projektes verwendeten Software- und Hardware- Komponenten dargestellt.

|  |  |  |
| --- | --- | --- |
| Name/Bezeichnung | 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 4: Sachmittel

# Projektphasen

## Planungsphase

### Realisierungskonzept

Mit Hilfe 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 Ausschnitt einer Beispielsituation (größeres Beispiel im Anhang).

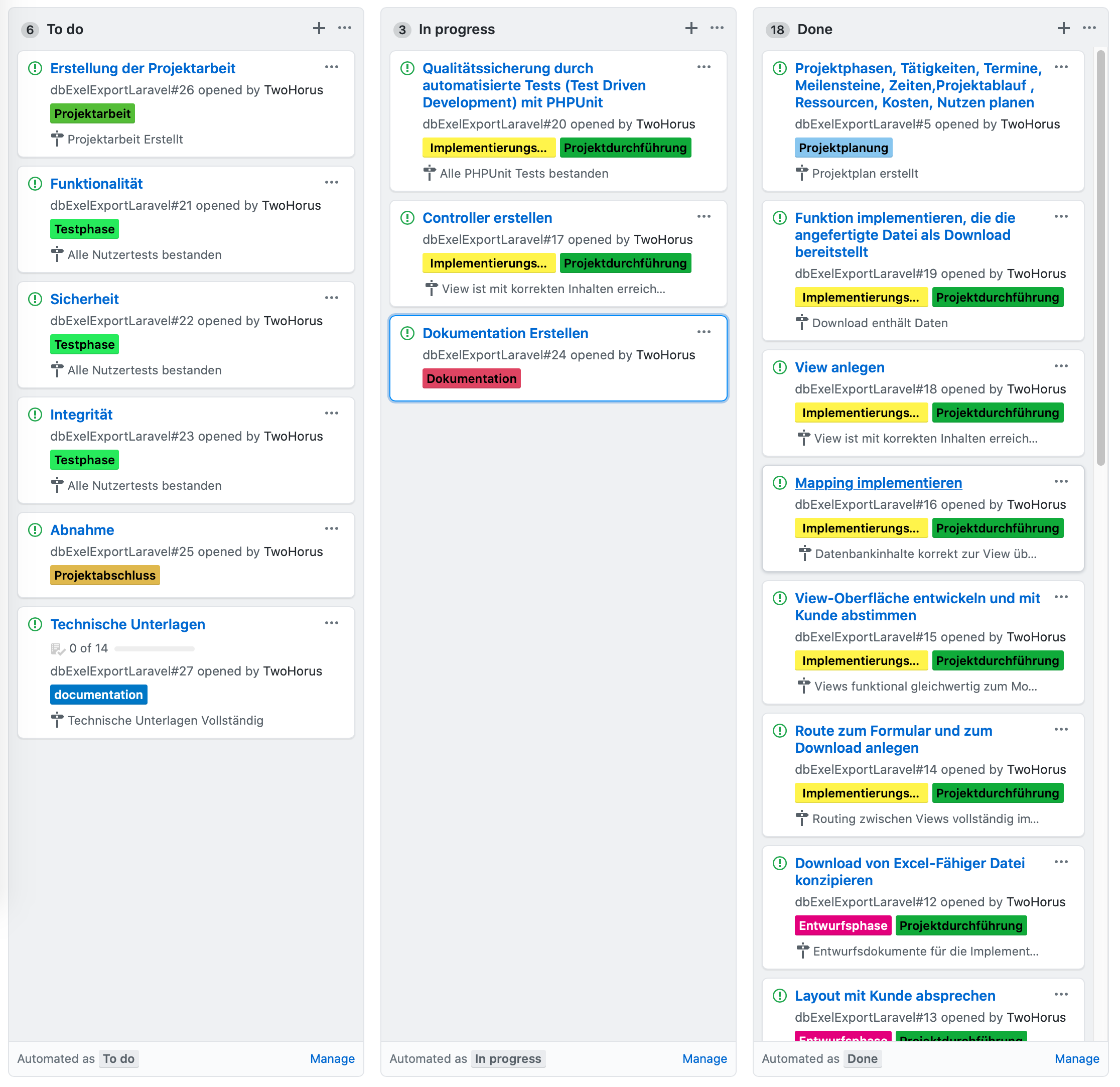


Abbildung 2: Kanban Board zur Planung

## 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 mit Hilfe 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 diese die einzige gut dokumentierte Exportbibliothek für Laravel ist. Es gibt eine weitere Bibliothek, „rap2hpoutre/fast-excel“. Diese 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 zu adressieren musste ich ein Modell für jede Tabelle der Datenbank, die für das Exportvorhaben relevant ist, erstellen. Abbildung 3 zeigt das relationale Datenbankmodell.

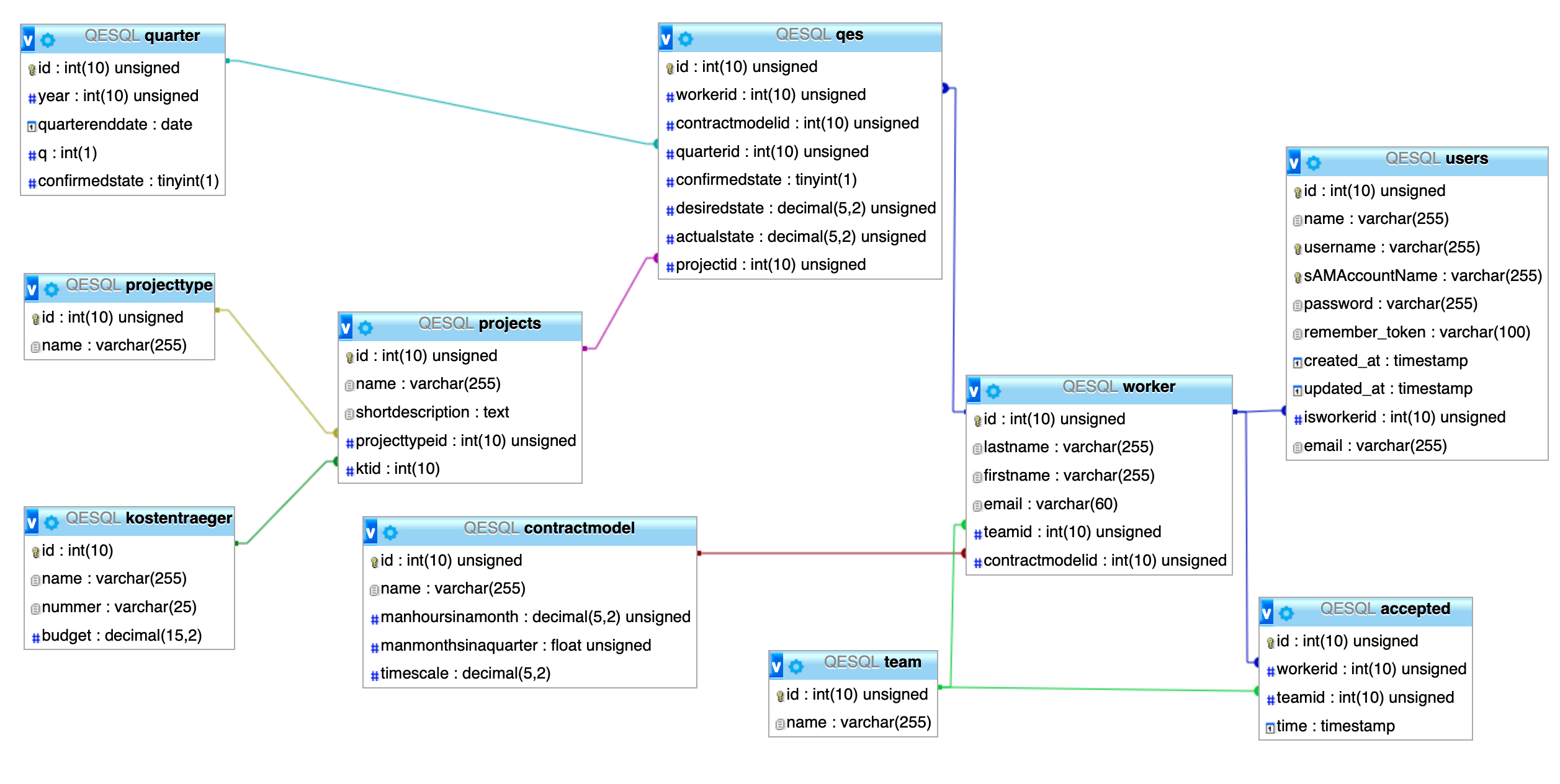


Abbildung 3: ERD QES

In der QES-Datenbank ist es möglich, zu erfassen, welcher Mitarbeiter wann an welchem Projekt gearbeitet hat. Das Kernstück der Datenbank ist die Tabelle „qes“. Die Tabelle „qes“ enthält einen Primärschlüssel, einen Fremdschlüssel für die Tabelle der Mitarbeiter, einen Fremdschlüssel für die Tabelle der Arbeitszeitmodelle, einen Fremdschlüssel für die Tabelle der Projekte, einen Fremdschlüssel für die Tabelle der Quartale und drei Nichtschlüsselattribute. Die Nichtschlüsselattribute sind der Soll-Zustand, der Ist-Zustand und der „confirmedstate“. Der „confirmedstate“ ist eine Variable, über die das Level der Bestätigung gesetzt wird. Der niedrigste Wert ist ein einfacher, vom Mitarbeiter eingetragener, Ist-Wert. Der nächsthöhere Wert wird gesetzt, wenn der Teamleiter des Mitarbeiters den Datensatz des Mitarbeiters eingesehen und bestätigt hat. Der höchste Wert wird gesetzt, wenn der Abteilungsleiter die Werte eingesehen und gesperrt hat. Somit wird dem Mitarbeiter die Möglichkeit gegeben, sich zu korrigieren, wenn er sich vertippt hat. Der Teamleiter kann sich jedoch nach der Bestätigung eines Datensatzes sicher sein, dass dieser nur in seinem Wissen wieder geändert werden kann. Dieses Konzept soll die Möglichkeit für Fehlerentstehung geringhalten.

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 (Object-Relational Mapping) ansprechen. Ein ORM hilft dem Anwendungsentwickler, Datenbankabfragen von der Datenbank zu abstrahieren, sodass von SQLite zu mySQL gewechselt werden kann ohne die Abfragen ändern zu müssen.



Abbildung 4: 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 mit Hilfe 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. Diese wird vom Parser in das Excel-Format überführt.

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. Nach dem MVC Pattern habe ich auch die Filterung und Sortierung im Controller implementiert. 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 4 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 20.000 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. Die Verbindung innerhalb des Firmennetzes ist jedoch so schnell, 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 die vorliegende Projektdokumentation erstellt.

### Übergabe und Abnahme

Die Übergabe der Ergebnisse fand am 07.11.2019 statt, die Abnahme wurde am 13.11.2019 bestätigt, da dem Kunden am ersten Übergabetermin die Daten, welche exportiert wurden, nicht genügten und neue Datensätze in das Basissystem eingetragen werden mussten. Am 13.11.2019 war der Kunde vollständig zufrieden mit der Handhabung und Funktionalität der Anwendung und Anwenderdokumentation.

# Probleme und Lösungen

## Erstellung des Controllers

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

## Formatierung der Daten

Es stellte sich anfangs schwierig dar, die Daten korrekt zu formatieren. Der Grund hierfür war die Menge an Sonderfällen. Nachdem ich logisch alle Sonderfälle aufgelistet hatte, konnte ich die Fehler im Code leichter finden, da die Probleme in der Ausgabe mehr Bezug zur Programmlogik haben. Wenn der Mitarbeiter des Eintrags sich ändert, muss ein neues Objekt erstellt werden und nicht nur das erste Objekt ausgegeben werden. Durch logische Strukturierung des Codes konnte ich die Fehleranfälligkeit der Ausgabe eliminieren.

# 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 Informationsquellenverzeichnis

[MVCWIK] https://en.wikipedia.org/wiki/[Model\_View\_Controller](https://de.wikipedia.org/wiki/Model_View_Controller) Stand 06.11.2019

[XPWIK] https://en.wikipedia.org/wiki/Extreme\_programming 06.11.2019

## A2 Abbildungsverzeichnis

[Abbildung 1: Phasen des Vorgehensmodells Extreme Programming 4](#_Toc24567076)

[Abbildung 2: Kanban Board zur Planung 8](#_Toc24567077)

[Abbildung 3: ERD QES 9](#_Toc24567078)

[Abbildung 4: Beziehungsdefinition der Modelle 10](#_Toc24567079)

## A3 Anlagenverzeichnis

Anlage 1: Entwurfsdokumente

Anlage 2: Goßes Kanban Board

Anlage 3: Projektabnahme

Anlage 4: Anwenderdokumentation

Anlage 5: Auszüge aus dem Quelltext

Anlage 1:

Ablauf des ControllersA picture containing screenshot

Description automatically generated

DatenbankschemaA screenshot of a social media post

Description automatically generated

Simple NutzersichtA close up of a logo

Description automatically generated

Anlage 2

A screenshot of a computer

Description automatically generated

Anlage 3 (Scan)