



Betriebliche Projektarbeit

Auszubildende/r:

Johannes Meyerhoff
Sonnenweg 3
53229 Bonn

E-Mail:
john@r73.de

Ausbildungsbetrieb:

GESIS - Leibniz-Institut
für Sozialwissenschaften e.V.
Unter Sachsenhausen 6 - 8
50667 Köln

Ausbilder/in bzw. Projektbetreuer/in

Sascha Schüller
Sascha.Schueller@gesis.org

Identnummer: 5002305256
Prüflingsnummer: 18158

Vorschlag: 1
Beruf: Fachinformatiker Fachrichtung Anwendungsentwicklung
Prüfungstermin: Winter 2019/20
Prüfungsart: 50 - Abschlussprüfung

1 Projektbezeichnung (Auftrag/Teilauftrag):* [bearbeitet am 22.09.2019 um 21:14 Uhr]

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

1.1. Ausgangssituation* [bearbeitet am 22.09.2019 um 21:13 Uhr]

Die GESIS wird als ein Institut der Leibniz Gesellschaft regelmäßig auf Wirtschaftlichkeit geprüft. Für derartige Prüfungen, aber auch für administrative Entscheidungen, werden Daten zum Ressourcenverbrauch erfasst. Dies erfolgt derzeit dezentral von den einzelnen Mitarbeitern in Excel-Sheets. Diese werden anschließend von der Administration manuell zusammengeführt. Der Prozess ist anfällig für Fehler, unflexibel und mit erheblichem manuellem Aufwand verbunden. Änderungen der erfassten Daten sind schwierig.

Das QES-(Quartalsdaten-Erhebungs-System)-Projekt wurde mit dem Ziel, den dezentralen Prozess zu zentralisieren, zu automatisieren, zu vereinfachen und vor allem effizienter, weniger fehleranfällig und konfigurierbarer zu gestalten, gestartet.

QES ist eine Anwendung, die über einen Webbrowser aufgerufen werden kann. Sie ermöglicht es der Abteilung WTS (Wissenstechnologien für Sozialwissenschaften), die Auslastung der Teams in Bezug auf Projekte zu verfolgen. Es gibt Eingabemasken, Rollen und Gruppen. Die Datenstruktur

ermöglicht es dem Abteilungsleiter, zu verfolgen, wer wann in welchem Projekt gearbeitet hat und zu welchem Anteil der Arbeitszeit dies stattfand.

Mit QES werden Teams, Projekte, Kostenträger und Arbeitszeitmodelle definiert. Den Mitarbeitern werden Arbeitszeitmodelle und Projekte zugewiesen. Den Teams werden Mitarbeiter und Projekte zugewiesen. Für jedes Quartal werden Soll- und Ist-Stände der Mitarbeiter-Projekt-Verknüpfungen mit aktuellem Arbeitszeitmodell abgespeichert. Wenn ein Mitarbeiter das Team wechselt, wird dies aufgezeichnet, um im Nachhinein die Daten richtig interpretieren zu können.

Insgesamt können mit QES z.B. besonders zeitaufwändige Projekte besser identifiziert werden.

1.2. Zielsetzung*

[bearbeitet am 03.10.2019 um 18:58 Uhr]

Ziel ist es, der Abteilungsleitung und auch den Teamleitern eine tabellarische Excel-Übersicht über den tatsächlichen Ressourcenverbrauch pro Mitarbeiter und Projekt sowie Abweichungen von der Soll-Planung zu liefern.

Die Übersicht ist eine entscheidende Grundlage für die weitere Ressourcenplanung und soll dynamisch nach Auswahl eines Zeitraums aus den Daten in QES generiert werden.

Im QES soll 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 Nutzer wählt einen Zeitraum und 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.

1.3. Konsequenzen bei Nichtverwirklichung**

[bearbeitet am 22.09.2019 um 21:17 Uhr]

Abteilungsleitung und Teamleiter verfolgen den Ressourcenverbrauch gegenüber dem Soll aktuell auf Basis von Excel. Sofern der Export der QES-Daten in ein Excel-Format nicht gelingt, ist daher die gesamte Nutzung der Anwendung QES für Abteilungsleitung und Teamleiter infrage gestellt.

Es würde eine robuste und an aktuelle Bedarfe anpassbare Grundlage für die Ressourcenplanung fehlen.

Die Abteilungsleitung würde statt des neuen QES die bestehende dezentrale Lösung weiter verwenden, da mit dem fehlenden Excel-Export wichtige Ergebnisse für Nachfolgeprozesse fehlen würden.

2. Projektumfeld/Rahmenbedingungen*

[bearbeitet am 22.09.2019 um 21:13 Uhr]

GESIS - Leibniz-Institut für Sozialwissenschaften mit Sitz in Köln und Mannheim 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 Johannes Meyerhoff arbeitet, unterstützt die sozialwissenschaftlichen Fachabteilungen durch die Entwicklung innovativer Wissenstechnologien zur Verbesserung der Integration und Nutzerfreundlichkeit der digitalen GESIS-Angebote. Die Haupttätigkeit des Auszubildenden ist die Entwicklung des QES.

Der virtuelle Server für die Produktiv-Version ist auf das interne Firmennetz beschränkt. Der Auszubildende, sein Ausbilder und die IT-Abteilung haben administrative Rechte. Die MySQL-Datenbank für QES ist nur von dem Server, auf dem das System läuft, erreichbar

Die Anwendung QES läuft auf einem virtualisierten Ubuntu 16 System. 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.

Auf dem Computer des Auszubildenden 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. 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 Atom Editor, iTerm2, GitLab, phpMyAdmin, Mozilla Firefox und Google Chrome.

Auftraggeber ist der ehemalige kommissarische Abteilungsleiter der Abteilung WTS.

Ansprechpartner bei technischen oder organisatorischen Problemen ist Sascha Schüller.

Ansprechpartner bei geschäftsprozessbezogenen Rückfragen ist Thomas Knecht.

3. Projektplanung/Projektphasen/geplante Arbeitsschritte inklusive Zeitplanung* [bearbeitet am 03.10.2019 um 20:37 Uhr]

ggf. inklusive Angabe der Meilensteine

Gesamt 67 Stunden geplant, exklusive 3 Stunden Puffer.

Phase Projektinitialisierung:

Arbeitspakete:

- AP PI1: Kick-off Meeting mit dem Kunden durchführen (0,5 Stunden)
- AP PI2: Projektinitiierung/ -definition abschließen (0,5 Stunden)

Meilensteine:

- M1: Lastenheft (Ergebnis: AP PI1 und AP PI2)

Phase Projektplanung:

Arbeitspakete:

- AP P1: Risiken einschätzen und Kosten betrachten (1,5 Stunden)
- AP P2: Benötigte Funktionalitäten für Controller ermitteln (0,5 Stunden)
(Interaktionsübersichtsdiagramm/Struktogramm)
- AP P3: Verfügbare Komponenten und Lösungen am Markt untersuchen (für Laravel) (2 Stunden)

- AP P4: Arbeitspakete planen(2 Stunden)
- AP P5: Projektphasen, Tätigkeiten, Termine, Meilensteine, Zeiten,Projektablauf , Ressourcen, Kosten, Nutzen planen (3 Stunden)
- AP P6: Tests anlegen um Qualität zu planen (2 Stunden)

Meilensteine:

- M2: Risiko/Kostenbetrachtung erstellt(AP P1)
- M3: Struktogramm/Interaktionsübersichtsdiagramm für Controller erstellt (AP P2)
- M4: Marktübersicht verfügbarer Lösungskomponenten erstellt (AP P3)
- M5: Projektplan erstellt (AP P4 P5 P6)

Phase Projektdurchführung:

Analyse 2 Stunden

Arbeitspakete:

- AP A1: Machbarkeitsstudie durchführen (2 Stunden)

Meilensteine:

- M6: Machbarkeit bestätigt (AP A1)

Entwurf:

Arbeitspakete:

- AP E1: Aus dem Lastenheft das Pflichtenheft erstellen (2 Stunden)
- AP E2: Eigenschaften des Models festlegen (1 Stunde) (Klassendiagramm)
- AP E3: Formular mit Auswahlmöglichkeit zur Generierung entwerfen (4 Stunden) (Entwurfsdokument)
- AP E4: Menüpunkt und Controller entwerfen nach Model-View-Controller Konzept unter Beachtung der bestehenden Models(4 Stunden) (Interaktionsübersichtsdiagramm/Struktogramm)
- AP E5: Download von Excel-Fähiger Datei konzipieren (Im Stil von Mockup) (1 Stunde) (Entwurfsdokument vom Kunden bereitgestellt)
- AP E6: Layout mit Kunde absprechen (1 Stunde) (Entwurfsdokument mit Kunde gemeinsam erstellt)

Meilensteine:

- M7: Pflichtenheft erstellt (AP E1)
- M8: Entwurfsdokumente für die Implementierung erstellt (AP E2 E3 E4 E5)
- M9: Layout abgenommen (AP E6)

Implementierung:

Arbeitspakete:

- AP I1: Route zum Formular und zum Download anlegen (1 Stunde)
- AP I2: View-Oberfläche entwickeln und mit Kunde abstimmen (2 Stunden)
- AP I3: Mapping implementieren (Daten aus DB -> Excel wie vom Kunden vorgegeben) (1

Stunde) (Mit Hilfe der Entwurfsdokumente)

- AP I4: Controller erstellen (3 Stunden)
- AP I5: View anlegen (3 Stunden) (Mit Hilfe der Entwurfsdokumente)
- AP I6: Funktion implementieren, die die angefertigte Datei als Download bereitstellt (5 Stunden) (Anhand von Interaktionsübersichtsdiagramm/Struktogramm)
- AP I7: Qualitätssicherung durch automatisierte Tests (Test Driven Development) mit PHPUnit (4 Stunden)

Meilensteine:

- M10: Routing zwischen Views vollständig implementiert (AP I1)
- M11: Views funktional gleichwertig zum Mockup, der mit dem Kunden besprochen wurde (AP I2)
- M12: Datenbankinhalte korrekt zur View übertragen (AP I3)
- M13: View ist mit korrekten Inhalten erreichbar (AP I4 I5)
- M14: Download enthält Daten (AP I6)
- M15: Alle PHPUnit Tests bestanden (AP I7)

Testphase mit Nutzertests:

Arbeitspakete:

- AP T1: Funktionalität - Sind alle Funktionen abgedeckt? (2 Stunden)
- AP T1: Sicherheit - Werden unautorisierte Nutzer zurückgewiesen? (2 Stunden)
- AP T2: Integrität - Werden die richtigen Mitarbeiter für die Teamleiter ausgewählt? (1 Stunde)
 - Sind alle Teams für den Abteilungsleiter verfügbar? (1 Stunde)

Meilensteine:

- M13: Alle Nutzertests bestanden (AP T1 T2 T3)

Dokumentation erstellen (2 Stunden)

Phase Projektabschluss:

Abnahme: 2 Stunden

- Demonstration der Funktionalität
- Dokumentation an Kunden übergeben

Erstellung der Projektarbeit (Prozessorientierter Projektbericht) (11 Stunden)

4. Dokumentation/technische Unterlagen*

[bearbeitet am 03.10.2019 um 19:38 Uhr]


Welche technischen Unterlagen planen Sie ihrer Dokumentation später beizufügen?

Dokumentation:

- Anleitung für Abteilungsleitung
- Anleitung für Teamleiter
- Anleitung für Mitarbeiter
- Anleitung zur technischen Wartung
- Kommentierter Quellcode
- User-Stories

- Abnahmeprotokoll
- Prozessorientierter Projektbericht
- Aufwandsschätzung
- Pflichtenheft
- Projektplanungsdokumente
- Qualitätsmanagement- / Qualitätssicherungsdokumente
- Entwurfsdokumente
- Klassendiagramm
- Interaktionsübersichtsdiagramm/Struktogramm

Legende * = Pflichtfeld, ** = Freitext, *** = keine Eingabe erforderlich

From: thomas.rebbe@koeln.ihk.de 
Subject: Projektantrag genehmigt Meyerhoff, Johannes (Azubinummer: 5002305256 | Abschlussprüfung Winter 2019/20 | Beruf: Fachinformatiker)
Date: 8. October 2019 at 08:40
To: john@r73.de

Sehr geehrter Herr Meyerhoff,

Ihr Antrag für die/den betriebliche/n Projektarbeit/Fachaufgabe/Arbeitsauftrag wurde von der IHK Köln an den zuständigen Prüfungsausschuss zur weiteren Bearbeitung weitergeleitet.
Der Prüfungsausschuss hat den Antrag genehmigt.
Bitte beachten Sie die Hinweise des Prüfungsausschusses:

Sehr geehrter Herr Meyerhoff,

die Genehmigung Ihres Projektantrags wird ohne Auflage ausgesprochen.

Hinweise und Auflagen zur Erstellung des prozessorientierten Projektberichts:
Formale Vorgaben entnehmen Sie bitte dem Dokument "Formale Anforderung an die Dokumentation".
Umfassende Informationen zu den inhaltlichen Anforderungen entnehmen Sie bitte der "Handreichung FIAE".
Beide Dokumente finden Sie auf der Seite mit der Dokumentennummer 318 auf www.ihk-koeln.de.

Ein vom Auftraggeber unterzeichnetes Abnahmeprotokoll muss in den Anlagen zum Projektbericht enthalten sein.

Ihr Prüfer team wünscht Ihnen viel Erfolg

Weiterhin beachten Sie bitte folgende Hinweise:

Vertrauliche oder datenschutzrelevante Daten sind als solche zu kennzeichnen. Sofern es aus datenschutzrechtlichen Gründen erforderlich ist, können Teile der Information durch Schwärzen etc. unkenntlich gemacht werden. Es ist nicht gestattet, verfälschte (Pseudo-)Namen, unrichtige Datumsangaben etc. zu verwenden. Die Nachvollziehbarkeit einzelner Prozessschritte muss unbedingt gewährleistet sein.
Dokumente mit rechtlich verbindlichem Charakter sind, mit entsprechender Unterschrift versehen, eingescannt beizufügen.

Bitte beachten Sie unbedingt den Termin zur Einreichung der/des betriebliche/n Projektarbeit/Fachaufgabe/Arbeitsauftrag:

14.11.2019 12:00 Uhr

Bis zu diesem Termin müssen Sie Ihre/n Dokumentation/Report im pdf-Format von max. 6 MB in das Portal eingestellt haben.
Der vorstehende Termin ist eine Ausschlussfrist. Die Nichteinhaltung dieser Frist, sowie die Nichtbeachtung der formalen Vorgaben und Hinweise führt zu einer Wertung der/des Dokumentation/Report mit "0 Punkten / nicht bestanden".

Wir wünschen Ihnen viel Erfolg!

Industrie- und Handelskammer zu Köln
Im Auftrag

Thomas Rebbe
Geschäftsbereich Aus- und Weiterbildung

Unter Sachsenhausen 10-26 50667 Köln
Tel. +49 221 1640-6560
Fax +49 221 1640-6490
E-Mail: thomas.rebbe@koeln.ihk.de
Internet: www.ihk-koeln.de

Erklärung

Ich versichere durch meine Unterschrift, dass ich das betriebliche Projekt in der vorgegebenen Zeit durchgeführt und die dazugehörige Dokumentation selbstständig ohne fremde Hilfe erstellt habe. Alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, wurden von mir als solche kenntlich gemacht.

Die eingereichte betriebliche Projektarbeit hat in dieser Form weder bei der prüfenden noch oder einer anderen IHK vorgelegen.

Köln 7.11.2019

Meyheff

Ort:

Datum:

Unterschrift Antragsteller/-in (Prüfling)

Ich habe die obige persönliche Erklärung zur Kenntnis genommen und bestätige, dass die betriebliche Projektarbeit und die dazugehörige Dokumentation in der vorgegebenen Zeit in unserem Betrieb durch den Prüfling selbstständig bearbeitet und erstellt wurden. Die einzureichende Dokumentation liegt im Unternehmen vor.

Köln 13.11.2019

P. Jun

Ort:

Datum:

Stempel/Unterschrift Verantwortliche/r für die Projektarbeit

13.11.2019

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

Johannes Meyerhoff, Azubi-Nr.: 5002305256

FACHINFORMATIKER FÜR ANWENDUNGSENTWICKLUNG
GESIS LEIBNIZ INSTITUT FÜR SOZIALWISSENSCHAFTEN
UNTER SACHSENHAUSEN 6-8 50667 KÖLN

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	6
4	PROJEKTPHASEN	7
4.1	PLANUNGSPHASE	7
4.1.1	<i>Realisierungskonzept</i>	7
4.2	VORBEREITENDE AUFGABEN	7
4.2.1	<i>Testumgebung einrichten</i>	7
4.3	PROGRAMMREALISIERUNG	8
4.3.1	<i>Erstellung Model</i>	8
4.3.2	<i>Erstellung Controller</i>	10
4.3.3	<i>Erstellung View</i>	10
4.3.4	<i>Anpassung Zeilenausgabe</i>	11
4.4	QUALITÄTSSICHERUNG	11
4.4.1	<i>Unit-Tests</i>	11
4.4.2	<i>Feature-Tests</i>	11
4.4.3	<i>Performancetest</i>	11
4.4.4	<i>Fehlerbeseitigung</i>	12
4.5	ABNAHME UND DOKUMENTATION	12
4.5.1	<i>Erstellung der Anwenderdokumentation</i>	12
4.5.2	<i>Erstellung der Projektdokumentation</i>	12
4.5.3	<i>Übergabe und Abnahme</i>	12
5	PROBLEME UND LÖSUNGEN	13
5.1	ERSTELLUNG DES CONTROLLERS	13
5.2	FORMATIERUNG DER DATEN	13
6	ERGEBNISSE UND FAZIT	13
7	ANHANG	14
7.1	A1 INFORMATIONSQUELLENVERZEICHNIS	14
7.2	A2 ABBILDUNGSVERZEICHNIS	14
7.3	A3 ANLAGENVERZEICHNIS	14

1 Vorwort

Diese Projektdokumentation beschreibt die Durchführung meines Projekts zur Erweiterung des **Quartalsdaten-Erhebungs-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.

2 Projektbeschreibung

2.1 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 Peter Mutschke der Abteilung WTS. Ansprechpartner bei technischen oder organisatorischen Problemen ist mein Ausbilder Sascha Schüller. Fachlicher Ansprechpartner bei geschäftsprozessbezogenen Rückfragen ist Thomas Knecht.

2.2 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.

2.3 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.

3 Projektüberblick

3.1 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 Programmieren nicht durchgeführt.



Abbildung 1: Phasen des Vorgehensmodells Extreme Programming

3.2 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/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	AP A1	Machbarkeitsstudie durchführen	2	2
Analyse				
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

3.3 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

3.4 Sachmittel

In Tabelle 3 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 3: Sachmittel

4 Projektphasen

4.1 Planungsphase

4.1.1 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 (volle Größe im Anhang).

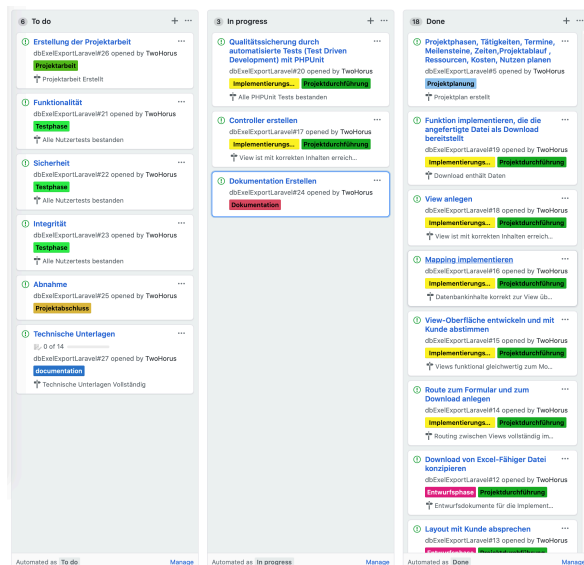


Abbildung 2: Kanban Board zur Planung

4.2 Vorbereitende Aufgaben

4.2.1 Testumgebung einrichten

Ich habe mit Bash einige Shell-Skripte 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.

4.3 Programmrealisierung

Das Model-View-Controller-Konzept (kurz MVC, siehe auch [MVCWIK]) wurde 1979 zunächst für Benutzeroberflächen in der Programmiersprache 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.

4.3.1 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 Selbstständige 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.

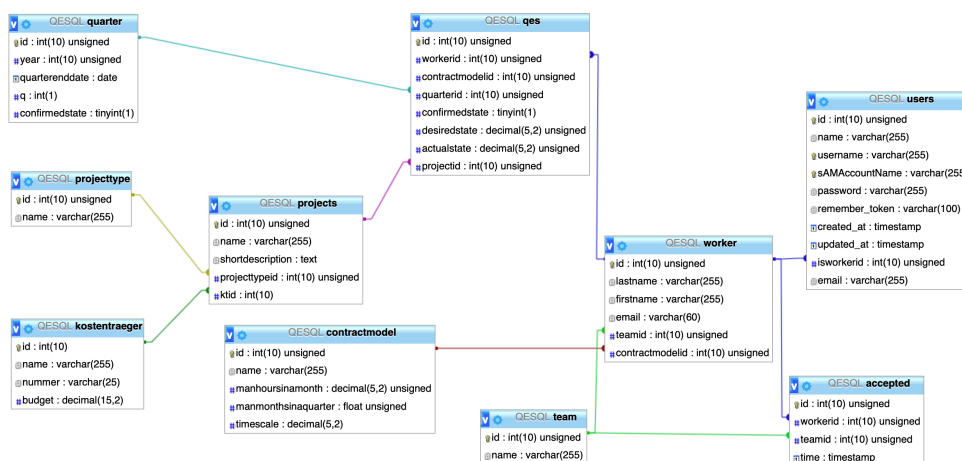


Abbildung 3: ERD QES

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 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.

```
public function project()
{
    return $this->belongsToMany('App\Project', 'projectid');
}
```

Abbildung 4: Beziehungsdefinition der Modelle

4.3.2 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

4.3.3 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.

4.3.4 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.

4.4 Qualitätssicherung

4.4.1 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.

4.4.2 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.

4.4.3 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.

4.4.4 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.

4.5 Abnahme und Dokumentation

4.5.1 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.

4.5.2 Erstellung der Projektdokumentation

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

4.5.3 Übergabe und Abnahme

Die Übergabe 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.

5 Probleme und Lösungen

5.1 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.

5.2 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 habe 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.

6 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.

7 Anhang

7.1 A1 Informationsquellenverzeichnis

[MVCWIK] https://en.wikipedia.org/wiki/Model_View_Controller

[XPWIK] https://en.wikipedia.org/wiki/Extreme_programming

7.2 A2 Abbildungsverzeichnis

Abbildung 1: Phasen des Vorgehensmodells Extreme Programming	4
Abbildung 2: Kanban Board zur Planung	7
Abbildung 3: ERD QES.....	8
Abbildung 4: Beziehungsdefinition der Modelle	9

7.3 A3 Anlagenverzeichnis

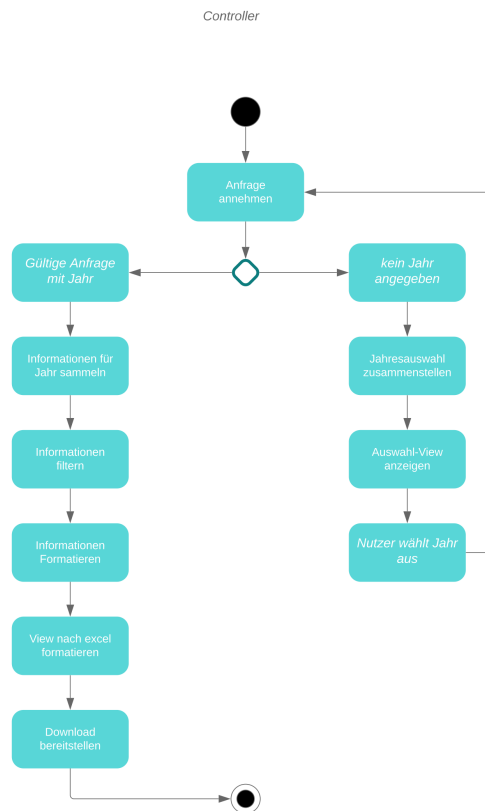
Anlage 1: Entwurfsdokumente

Anlage 2: Großes Kanban Board

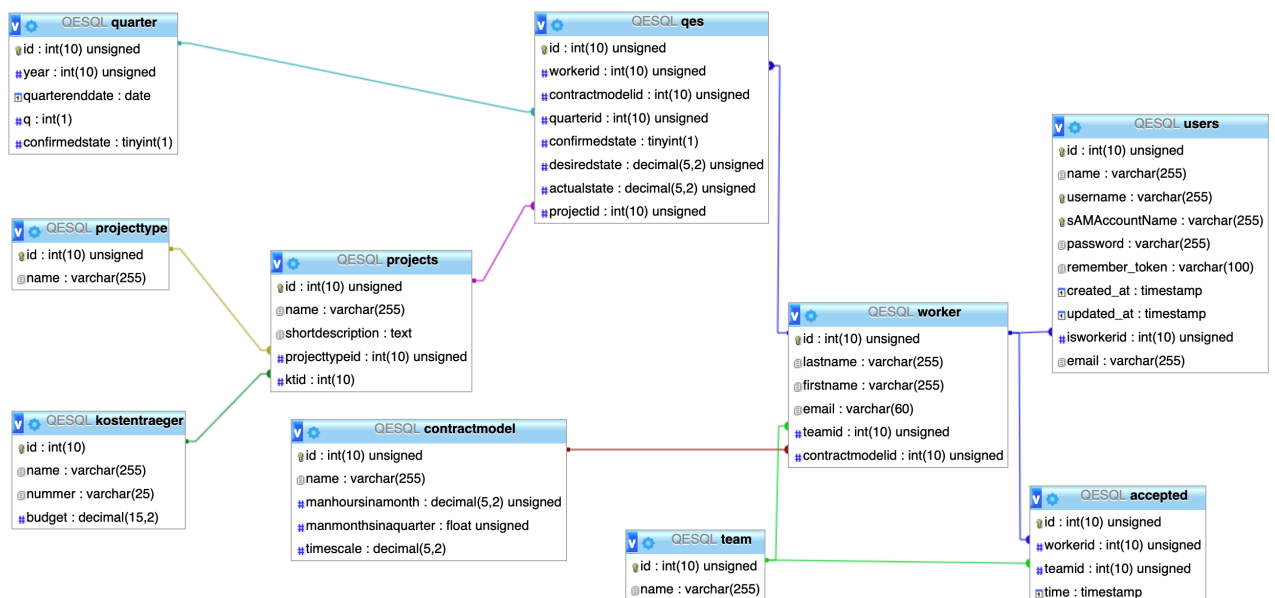
Anlage 3: Projektabnahme

Anlage 4: Auszüge aus dem Quelltext

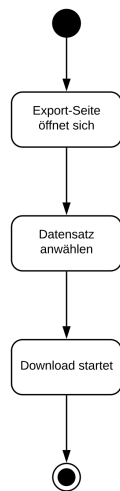
Ablauf des Controllers



Datenbankschema



Simple Nutzersicht



Anlage 2

6 To do

Erstellung der Projektarbeit

dbExelExportLaravel#26 opened by TwoHorus

Projektarbeit

Projektarbeit Erstellt

Funktionalität

dbExelExportLaravel#21 opened by TwoHorus

Testphase

Alle Nutzertests bestanden

Sicherheit

dbExelExportLaravel#22 opened by TwoHorus

Testphase

Alle Nutzertests bestanden

Integrität

dbExelExportLaravel#23 opened by TwoHorus

Testphase

Alle Nutzertests bestanden

Abnahme

dbExelExportLaravel#25 opened by TwoHorus

Projektabschluss

Technische Unterlagen

0 of 14

dbExelExportLaravel#27 opened by TwoHorus

documentation

Technische Unterlagen Vollständig

Automated as To do

Manage

3 In progress

Qualitätssicherung durch automatisierte Tests (Test Driven Development) mit PHPUnit

dbExelExportLaravel#20 opened by TwoHorus

Implementierungs... Projektdurchführung

Alle PHPUnit Tests bestanden

Controller erstellen

dbExelExportLaravel#17 opened by TwoHorus

Implementierungs... Projektdurchführung

View ist mit korrekten Inhalten erreich...

Dokumentation Erstellen

dbExelExportLaravel#24 opened by TwoHorus

Dokumentation

Automated as In progress

Manage

18 Done

Projektphasen, Tätigkeiten, Termine, Meilensteine, Zeiten, Projektablauf, Ressourcen, Kosten, Nutzen planen

dbExelExportLaravel#5 opened by TwoHorus

Projektplanung

Projektplan erstellt

Funktion implementieren, die die angefertigte Datei als Download bereitstellt

dbExelExportLaravel#19 opened by TwoHorus

Implementierungs... Projektdurchführung

Download enthält Daten

View anlegen

dbExelExportLaravel#18 opened by TwoHorus

Implementierungs... Projektdurchführung

View ist mit korrekten Inhalten erreich...

Mapping implementieren

dbExelExportLaravel#16 opened by TwoHorus

Implementierungs... Projektdurchführung

Datenbankinhalte korrekt zur View ü...

View-Oberfläche entwickeln und mit Kunde abstimmen

dbExelExportLaravel#15 opened by TwoHorus

Implementierungs... Projektdurchführung

Views funktional gleichwertig zum Mo...

Route zum Formular und zum Download anlegen

dbExelExportLaravel#14 opened by TwoHorus

Implementierungs... Projektdurchführung

Routing zwischen Views vollständig im...

Download von Excel-Fähiger Datei konzipieren

dbExelExportLaravel#12 opened by TwoHorus

Entwurfsphase Projektdurchführung

Entwurfsdokumente für die Implement...

Layout mit Kunde absprechen

dbExelExportLaravel#13 opened by TwoHorus

Entwurfsphase Projektdurchführung

Automated as Done

Manage

Anlage 3 (Scan)

Abnahmeprotokoll

Abnahmeerklärung

Endabnahme

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

Die Abnahme umfasst folgende Komponenten:

- Software zum Export der Daten
- Anwenderdokumentation

Johannes Meyerhoff hat die o.g. Komponenten am 07.11. betriebs- und abnahmebereit zur Durchführung der Abnahme an den Kunden übergeben. Der Kunde hat diese Komponenten geprüft und getestet.

Abnahme erteilt

13.11.2015 P. J.

Datum, Unterschrift

AUSSCHNITT AUS DER AUSGABEFUNKTION

```
<td <?php if ($finishedrow->drittmittel==1) {
    echo ('bgcolor="#C2C5CC"');
} ?>
    <?php if (is_numeric($finishedrow-
>desiredstate2)&&is_numeric($finishedrow->actualstate2)) {
        if (abs(($finishedrow->desiredstate2 - $finishedrow-
>actualstate2))>19) {
            echo ('style="color:#CD0000"');
        } else {
            echo('style="color:#000000"');
        }
    } ?>>
    <?php

    echo $finishedrow->actualstate2 ?? '&nbsp;';

//LOGIK ZUM NTSCHIEDEN OB Q1, Q2 oder Q3 ???
?></td>
```

```
<td <?php if ($finishedrow->drittmittel==1) {
    echo ('bgcolor="#C2C5CC"');
} ?>
    <?php if (is_numeric($finishedrow-
>desiredstate3)&&is_numeric($finishedrow->actualstate3)) {
        if (abs(($finishedrow->desiredstate3 - $finishedrow-
>actualstate3))>19) {
            echo ('style="color:#CD0000"');
        } else {
            echo('style="color:#000000"');
        }
    } ?>>
    <?php
```

```

        echo $finishedrow->desiredstate3 ?? '&nbsp;';

//LOGIK ZUM ENTSCHEIDEN OB Q1, Q2 oder Q3 ???
    ?></td>

<td <?php if ($finishedrow->drittmittel==1) {
    echo ('bgcolor="#C2C5CC"');
    } ?>
    <?php if (is_numeric($finishedrow-
>desiredstate3)&&is_numeric($finishedrow->actualstate3)) {
        if (abs(($finishedrow->desiredstate3 - $finishedrow-
>actualstate3))>19) {
            echo ('style="color:#CD0000"');
        } else {
            echo('style="color:#000000"');
        }
    } ?>>
    <?php

    echo $finishedrow->actualstate3 ?? '&nbsp;';

//LOGIK ZUM ENTSCHEIDEN OB Q1, Q2 oder Q3 ???
    ?></td>

```

CONTROLLER FÜR DEN EXPORT

```
<?php

namespace App\Exports;

use App\User;
use App\Worker;
use App\Qes;
use App\Kostentraeger;
use App\Project;
use App\Projecttype;

use Illuminate\Contracts\View\View;
use Maatwebsite\Excel\Concerns\FromView;

class UsersExport implements FromView
{

    private $year;

    public function __construct(int $year)
    {
        $this->year = $year;
    }

    public function view(): View
    {
        /*
        FEATURE FOR LATER
        $sumrow = (object)[
            'workerid' => null,
            'projectid' => null,
            'dent' => 'new', // NEW LINE
            'desiredstate1' => '=SUM(I1:I5)',
            'actualstate1' => '=SUM(J1:J5)',
            'desiredstate2' => '=SUM(K1:K5)',
            'actualstate2' => '=SUM(L1:L5)',
            'desiredstate3' => '=SUM(M1:M5)',
            'actualstate3' => '=SUM(N1:N5)',
        ];
        */
    }
}
```

```
'desiredstate4' => '=SUM(01:05)',
'actualstate4' => '=SUM(P1:P5)',
'projecttypename' => '',
'projectname' => ' ',
'funding' => '',
'drittmittel' => '',
'kt' => null,
'eg' => '',// LATER ADD AS FEATURE
'manhoursinamonth' => null,
'sender' => 'default',
'firstname' => ' ',
'lastname' => ' ',
'teamname' => ' ',
];
$emptyrow= (object)[
'workerid' => null,
'projectid' => null,
'dent' => 'new',// NEW LINE
'desiredstate1' => '',
'actualstate1' => '',
'desiredstate2' => '',
'actualstate2' => null,
'desiredstate3' => null,
'actualstate3' => null,
'desiredstate4' => null,
'actualstate4' => null,
'projecttypename' => '',
'projectname' => ' ',
'funding' => '',
'drittmittel' => '',
'kt' => null,
'eg' => '',// LATER ADD AS FEATURE
'manhoursinamonth' => null,
'sender' => 'default',
'firstname' => ' ',
'lastname' => ' ',
'teamname' => ' ',
];
```



```

*/
$objectthere=\DB::Table('qes')
->join('projects', 'qes.projectid', '=', 'projects.id')
->join('worker', 'qes.workerid', '=', 'worker.id')
->join('team', 'worker.teamid', '=', 'team.id')
->join('quarter', 'qes.quarterid', '=', 'quarter.id')
->join('projecttype', 'projecttypeid', '=', 'projecttype.id')
->join('contractmodel', 'worker.contractmodelid', '=',
'contractmodel.id')
->join('kostentraeger', 'projects.ktid', '=', 'kostentraeger.id')
->where('year', '=', $this->year)//this is the variable part
->orderBy('workerid')->orderBy('projectid')
->select('*', 'team.name as tname', 'projects.name as pname',
'projecttype.name as ptypename', 'kostentraeger.name as ktypename',
'contractmodel.name as eg')
->get();
$rows=[];
$currentworker=-1;
$currentproject=-1;
$firstmismatch=1;
foreach ($objectthere as $datapoint) {
    if ($currentworker==$datapoint->workerid) {//SAME WORKER
        if ($currentproject==$datapoint->projectid) {//SAME PROJECT
            //READDATATOROW
            switch ($datapoint->quarter->q ?? $datapoint->q) {
                case '1':
                    $row->desiredstate1 = $datapoint->desiredstate;
                    $row->actualstate1 = $datapoint->actualstate;
                    break;
                case '2':
                    $row->desiredstate2 = $datapoint->desiredstate;
                    $row->actualstate2 = $datapoint->actualstate;
                    break;
                case '3':
                    $row->desiredstate3 = $datapoint->
>desiredstate;
                    $row->actualstate3 = $datapoint->
>actualstate;
                    break;
            }
        }
    }
}

```

```

        case '4':
            $row->desiredstate4 = $datapoint->desiredstate;
            $row->actualstate4 = $datapoint->actualstate;
            break;
        default:
            break;
    }
} else {
    $currentproject=$datapoint->projectid;//UPDATE PROJECT
    //PUT ROW INTO OUR OBJECTLIST FIRST
    $rows[] = $row;
    $row = (object)[
        'workerid' => null,
        'projectid' => null,
        'dent' => 'new',// NEW LINE
        'desiredstate1' => null,
        'actualstate1' => null,
        'desiredstate2' => null,
        'actualstate2' => null,
        'desiredstate3' => null,
        'actualstate3' => null,
        'desiredstate4' => null,
        'actualstate4' => null,
        'projecttypename' => $datapoint->project->type->name ?? ' ',
        'projectname' => $datapoint->project->name ?? $datapoint->pname,
        'funding' => $datapoint->project->type->name ?? $datapoint->ptypename,
        'drittmittel' => 0,
        'kt' => $datapoint->project->kostentraeger->name ?? $datapoint->ktypename ?? ' ',
        'eg' => '',// LATER ADD AS FEATURE
        'manhoursinamonth' => 'X',
        'sender' => 'default',
        'firstname' => ' ',
        'lastname' => ' ',
        'teamname' => ' ',

```

```

    ];
    //READING Q DATA DYNAMICALLY
    switch ($datapoint->quarter->q ?? $datapoint->q) {
        case '1':
            $row->desiredstate1 = $datapoint->desiredstate;
            $row->actualstate1 = $datapoint->actualstate;
            break;
        case '2':
            $row->desiredstate2 = $datapoint->desiredstate;
            $row->actualstate2 = $datapoint->actualstate;
            break;
        case '3':
            $row->desiredstate3 = $datapoint->desiredstate;
            $row->actualstate3 = $datapoint->actualstate;
            break;
        case '4':
            $row->desiredstate4 = $datapoint->desiredstate;
            $row->actualstate4 = $datapoint->actualstate;
            break;
        default:
            break;
    }
}
} else {
    $currentworker = $datapoint->workerid; //WORKER IS WORKER NOW
    $currentproject=$datapoint->projectid; //PROJECT IS PROJECT
    NOW
    if ($firstmismatch==1) { //SPECIAL FIRST ROW ONLY
        $firstmismatch=0;
        //FIRST ROW IS NULL NOW
    } else {
        //PUT ROW INTO OUR OBJECTLIST FIRST
        $rows[] = $row;
        // $rows[] = $sumrow;
        // $rows[] = $emptyrow;
    }
    //THEN READ NEW DATA TO NEW ROW OBJECT
    $row = (object)[
        'workerid' => $datapoint->workerid,

```

```

        'projectid' => $datapoint->projectid,
        'dent' => 'new', // NEW LINE
        'desiredstate1' => null,
        'actualstate1' => null,
        'desiredstate2' => null,
        'actualstate2' => null,
        'desiredstate3' => null,
        'actualstate3' => null,
        'desiredstate4' => null,
        'actualstate4' => null,
        'projecttypename' => $datapoint->project->type->name ?? ' ',
        'projectname' => $datapoint->project->name ?? $datapoint->
>pname,
        'funding' => $datapoint->project->type->name ?? $datapoint->
>ptypename,
        'drittmittel' => 0,
        'kt' => $datapoint->project->kostentraeger->name ??
$datapoint->ktypename ?? ' ',
        'eg' => $datapoint->eg,
        'manhoursinamonth' => $datapoint->manhoursinamonth,
        'sender' => 'default',
        'firstname' => $datapoint->worker->firstname ?? $datapoint->
>firstname,
        'lastname' => $datapoint->worker->lastname ?? $datapoint->
>lastname,
        'teamname' => $datapoint->worker->team->name ?? $datapoint->
>tname,
    ];
    //READING Q DATA DYNAMICALLY
    switch ($datapoint->quarter->q ?? $datapoint->q) {
        case '1':
            $row->desiredstate1 = $datapoint->desiredstate;
            $row->actualstate1 = $datapoint->actualstate;
            break;
        case '2':
            $row->desiredstate2 = $datapoint->desiredstate;
            $row->actualstate2 = $datapoint->actualstate;
            break;
        case '3':

```

```
        $row->desiredstate3 = $datapoint->desiredstate;
        $row->actualstate3 = $datapoint->actualstate;
        break;
    case '4':
        $row->desiredstate4 = $datapoint->desiredstate;
        $row->actualstate4 = $datapoint->actualstate;
        break;
    default:
        break;
    }
}
}
$rows[] = $row;
return view('test', ['ges' => $rows, 'uienabled' => 'false' ]);
}
}
```

CONTROLLER FÜR DEN EXPORT

```
<?php

namespace App\Exports;

use App\User;
use App\Worker;
use App\Qes;
use App\Kostentraeger;
use App\Project;
use App\Projecttype;

use Illuminate\Contracts\View\View;
use Maatwebsite\Excel\Concerns\FromView;

class UsersExport implements FromView
{

    private $year;

    public function __construct(int $year)
    {
        $this->year = $year;
    }

    public function view(): View
    {
        /*
        FEATURE FOR LATER
        $sumrow = (object)[
```

```
'workerid' => null,  
'projectid' => null,  
'dent' => 'new',// NEW LINE  
'desiredstate1' => '=SUM(I1:I5)',  
'actualstate1' => '=SUM(J1:J5)',  
'desiredstate2' => '=SUM(K1:K5)',  
'actualstate2' => '=SUM(L1:L5)',  
'desiredstate3' => '=SUM(M1:M5)',  
'actualstate3' => '=SUM(N1:N5)',  
'desiredstate4' => '=SUM(O1:O5)',  
'actualstate4' => '=SUM(P1:P5)',  
'projecttypename' => '',  
'projectname' => ' ',  
'funding' => '',  
'drittmittel' => '',  
'kt' => null,  
'eg' => '',// LATER ADD AS FEATURE  
'manhoursinamonth' => null,  
'sender' => 'default',  
'firstname' => ' ',  
'lastname' => ' ',  
'teamname' => ' ',  
];  
$emptyrow= (object)[  
'workerid' => null,  
'projectid' => null,  
'dent' => 'new',// NEW LINE  
'desiredstate1' => '',  
'actualstate1' => '',  
'desiredstate2' => '',  
'actualstate2' => null,
```

```

'desiredstate3' => null,
'actualstate3' => null,
'desiredstate4' => null,
'actualstate4' => null,
'projecttypename' => '',
'projectname' => ' ',
'funding' => '',
'drittmittel' => '',
'kt' => null,
'eg' => '',// LATER ADD AS FEATURE
'manhoursinamonth' => null,
'sender' => 'default',
'firstname' => ' ',
'lastname' => ' ',
'teamname' => ' ',
];

```

```

*/

```

```

$objectthere=\DB::Table('ges')
->join('projects', 'ges.projectid', '=', 'projects.id')
->join('worker', 'ges.workerid', '=', 'worker.id')
->join('team', 'worker.teamid', '=', 'team.id')
->join('quarter', 'ges.quarterid', '=', 'quarter.id')
->join('projecttype', 'projecttypeid', '=', 'projecttype.id')
->join('contractmodel', 'worker.contractmodelid', '=', 'contractmodel.id')
->join('kostentraeger', 'projects.ktid', '=', 'kostentraeger.id')
->where('year', '=', $this->year)//this is the variable part
->orderBy('workerid')->orderBy('projectid')
->select('*', 'team.name as tname', 'projects.name as pname', 'projecttype.name as
ptypename', 'kostentraeger.name as ktypename', 'contractmodel.name as eg')
->get();

```



```

$rows=[];
$currentworker=-1;
$currentproject=-1;
$firstmismatch=1;
foreach ($objecthere as $datapoint) {
    if ($currentworker==$datapoint->workerid) {//SAME WORKER
        if ($currentproject==$datapoint->projectid) {//SAME PROJECT
            //READDATATOROW
            switch ($datapoint->quarter->q ?? $datapoint->q) {
                case '1':
                    $row->desiredstate1 = $datapoint->desiredstate;
                    $row->actualstate1 = $datapoint->actualstate;
                    break;
                case '2':
                    $row->desiredstate2 = $datapoint->desiredstate;
                    $row->actualstate2 = $datapoint->actualstate;
                    break;
                case '3':
                    $row->desiredstate3 = $datapoint->desiredstate;
                    $row->actualstate3 = $datapoint->actualstate;
                    break;
                case '4':
                    $row->desiredstate4 = $datapoint->desiredstate;
                    $row->actualstate4 = $datapoint->actualstate;
                    break;
                default:
                    break;
            }
        } else {
            $currentproject=$datapoint->projectid;//UPDATE PROJECT
            //PUT ROW INTO OUR OBJECTLIST FIRST

```

```

$rows[] = $row;
$row = (object)[
    'workerid' => null,
    'projectid' => null,
    'dent' => 'new', // NEW LINE
    'desiredstate1' => null,
    'actualstate1' => null,
    'desiredstate2' => null,
    'actualstate2' => null,
    'desiredstate3' => null,
    'actualstate3' => null,
    'desiredstate4' => null,
    'actualstate4' => null,
    'projecttypename' => $datapoint->project->type->name ?? ' ',
    'projectname' => $datapoint->project->name ?? $datapoint->pname,
    'funding' => $datapoint->project->type->name ?? $datapoint->ptypename,
    'drittmittel' => 0,
    'kt' => $datapoint->project->kostentraeger->name ?? $datapoint->ktypename ?? ' '
    ,

    'eg' => ' ', // LATER ADD AS FEATURE
    'manhoursinamonth' => 'X',
    'sender' => 'default',
    'firstname' => ' ',
    'lastname' => ' ',
    'teamname' => ' ',
];

//READING Q DATA DYNAMICALLY
switch ($datapoint->quarter->q ?? $datapoint->q) {
    case '1':
        $row->desiredstate1 = $datapoint->desiredstate;
        $row->actualstate1 = $datapoint->actualstate;

```

```

        break;
    case '2':
        $row->desiredstate2 = $datapoint->desiredstate;
        $row->actualstate2 = $datapoint->actualstate;
        break;
    case '3':
        $row->desiredstate3 = $datapoint->desiredstate;
        $row->actualstate3 = $datapoint->actualstate;
        break;
    case '4':
        $row->desiredstate4 = $datapoint->desiredstate;
        $row->actualstate4 = $datapoint->actualstate;
        break;
    default:
        break;
    }
}
} else {
    $currentworker = $datapoint->workerid; //WORKER IS WORKER NOW
    $currentproject=$datapoint->projectid; //PROJECT IS PROJECT NOW
    if ($firstmismatch==1) {//SPECIAL FIRST ROW ONLY
        $firstmismatch=0;
        //FIRST ROW IS NULL NOW
    } else {
        //PUT ROW INTO OUR OBJECTLIST FIRST
        $rows[] = $row;
        // $rows[] = $sumrow;
        // $rows[] = $emptyrow;
    }
    //THEN READ NEW DATA TO NEW ROW OBJECT
    $row = (object)[

```

```

'workerid' => $datapoint->workerid,
'projectid' => $datapoint->projectid,
'dent' => 'new', // NEW LINE
'desiredstate1' => null,
'actualstate1' => null,
'desiredstate2' => null,
'actualstate2' => null,
'desiredstate3' => null,
'actualstate3' => null,
'desiredstate4' => null,
'actualstate4' => null,
'projecttypename' => $datapoint->project->type->name ?? ' ',
'projectname' => $datapoint->project->name ?? $datapoint->pname,
'funding' => $datapoint->project->type->name ?? $datapoint->ptypename,
'drittmittel' => 0,
'kt' => $datapoint->project->kostentraeger->name ?? $datapoint->ktypename ?? ' ',
'eg' => $datapoint->eg,
'manhoursinamonth' => $datapoint->manhoursinamonth,
'sender' => 'default',
'firstname' => $datapoint->worker->firstname ?? $datapoint->firstname,
'lastname' => $datapoint->worker->lastname ?? $datapoint->lastname,
'teamname' => $datapoint->worker->team->name ?? $datapoint->tname,
];
//READING Q DATA DYNAMICALLY
switch ($datapoint->quarter->q ?? $datapoint->q) {
    case '1':
        $row->desiredstate1 = $datapoint->desiredstate;
        $row->actualstate1 = $datapoint->actualstate;
        break;
    case '2':
        $row->desiredstate2 = $datapoint->desiredstate;

```

```
        $row->actualstate2 = $datapoint->actualstate;
        break;
    case '3':
        $row->desiredstate3 = $datapoint->desiredstate;
        $row->actualstate3 = $datapoint->actualstate;
        break;
    case '4':
        $row->desiredstate4 = $datapoint->desiredstate;
        $row->actualstate4 = $datapoint->actualstate;
        break;
    default:
        break;
    }
}
}
$rows[] = $row;
return view('test', ['ges' => $rows, 'uienabled' => 'false' ]);
}
```

CONTROLLER FÜR DEN EXPORT

```
<?php

namespace App\Exports;

use App\User;
use App\Worker;
use App\Qes;
use App\Kostentraeger;
use App\Project;
use App\Projecttype;

use Illuminate\Contracts\View\View;
use Maatwebsite\Excel\Concerns\FromView;

class UsersExport implements FromView
{

    private $year;

    public function __construct(int $year)
    {
        $this->year = $year;
    }

    public function view(): View
    {
        /*
        FEATURE FOR LATER
        $sumrow = (object)[
```

```

'workerid' => null,
'projectid' => null,
'dent' => 'new',// NEW LINE
'desiredstate1' => '=SUM(I1:I5)',
'actualstate1' => '=SUM(J1:J5)',
'desiredstate2' => '=SUM(K1:K5)',
'actualstate2' => '=SUM(L1:L5)',
'desiredstate3' => '=SUM(M1:M5)',
'actualstate3' => '=SUM(N1:N5)',
'desiredstate4' => '=SUM(O1:O5)',
'actualstate4' => '=SUM(P1:P5)',
'projecttypename' => '',
'projectname' => ' ',
'funding' => '',
'drittmittel' => '',
'kt' => null,
'eg' => '',// LATER ADD AS FEATURE
'manhoursinamonth' => null,
'sender' => 'default',
'firstname' => ' ',
'lastname' => ' ',
'teamname' => ' ',
];
$emptyrow= (object)[
    'workerid' => null,
    'projectid' => null,
    'dent' => 'new',// NEW LINE
    'desiredstate1' => '',
    'actualstate1' => '',
    'desiredstate2' => '',
    'actualstate2' => null,

```

```

        'desiredstate3' => null,
        'actualstate3' => null,
        'desiredstate4' => null,
        'actualstate4' => null,
        'projecttypename' => '',
        'projectname' => ' ',
        'funding' => '',
        'drittmittel' => '',
        'kt' => null,
        'eg' => '', // LATER ADD AS FEATURE
        'manhoursinamonth' => null,
        'sender' => 'default',
        'firstname' => ' ',
        'lastname' => ' ',
        'teamname' => ' ',
    ];

```

*/

```

$objectthere=\DB::Table('ges')
->join('projects', 'ges.projectid', '=', 'projects.id')
->join('worker', 'ges.workerid', '=', 'worker.id')
->join('team', 'worker.teamid', '=', 'team.id')
->join('quarter', 'ges.quarterid', '=', 'quarter.id')
->join('projecttype', 'projecttypeid', '=', 'projecttype.id')
->join('contractmodel', 'worker.contractmodelid', '=', 'contractmodel.id')
->join('kostentraeger', 'projects.ktid', '=', 'kostentraeger.id')
->where('year', '=', $this->year)//this is the variable part
->orderBy('workerid')->orderBy('projectid')
->select('*', 'team.name as tname', 'projects.name as pname', 'projecttype.name as
ptypename', 'kostentraeger.name as ktypename', 'contractmodel.name as eg')
->get();

```



```

$rows=[];
$currentworker=-1;
$currentproject=-1;
$firstmismatch=1;
foreach ($objecthere as $datapoint) {
    if ($currentworker==$datapoint->workerid) {//SAME WORKER
        if ($currentproject==$datapoint->projectid) {//SAME PROJECT
            //READDATATOROW
            switch ($datapoint->quarter->q ?? $datapoint->q) {
                case '1':
                    $row->desiredstate1 = $datapoint->desiredstate;
                    $row->actualstate1 = $datapoint->actualstate;
                    break;
                case '2':
                    $row->desiredstate2 = $datapoint->desiredstate;
                    $row->actualstate2 = $datapoint->actualstate;
                    break;
                case '3':
                    $row->desiredstate3 = $datapoint->desiredstate;
                    $row->actualstate3 = $datapoint->actualstate;
                    break;
                case '4':
                    $row->desiredstate4 = $datapoint->desiredstate;
                    $row->actualstate4 = $datapoint->actualstate;
                    break;
                default:
                    break;
            }
        } else {
            $currentproject=$datapoint->projectid;//UPDATE PROJECT
            //PUT ROW INTO OUR OBJECTLIST FIRST

```

```

$rows[] = $row;
$row = (object)[
    'workerid' => null,
    'projectid' => null,
    'dent' => 'new', // NEW LINE
    'desiredstate1' => null,
    'actualstate1' => null,
    'desiredstate2' => null,
    'actualstate2' => null,
    'desiredstate3' => null,
    'actualstate3' => null,
    'desiredstate4' => null,
    'actualstate4' => null,
    'projecttypename' => $datapoint->project->type->name ?? ' ',
    'projectname' => $datapoint->project->name ?? $datapoint->pname,
    'funding' => $datapoint->project->type->name ?? $datapoint->ptypename,
    'drittmittel' => 0,
    'kt' => $datapoint->project->kostentraeger->name ?? $datapoint->ktypename ?? ' '
    ,

    'eg' => ' ', // LATER ADD AS FEATURE
    'manhoursinamonth' => 'X',
    'sender' => 'default',
    'firstname' => ' ',
    'lastname' => ' ',
    'teamname' => ' ',
];

//READING Q DATA DYNAMICALLY
switch ($datapoint->quarter->q ?? $datapoint->q) {
    case '1':
        $row->desiredstate1 = $datapoint->desiredstate;
        $row->actualstate1 = $datapoint->actualstate;

```

```

        break;
    case '2':
        $row->desiredstate2 = $datapoint->desiredstate;
        $row->actualstate2 = $datapoint->actualstate;
        break;
    case '3':
        $row->desiredstate3 = $datapoint->desiredstate;
        $row->actualstate3 = $datapoint->actualstate;
        break;
    case '4':
        $row->desiredstate4 = $datapoint->desiredstate;
        $row->actualstate4 = $datapoint->actualstate;
        break;
    default:
        break;
    }
}
} else {
    $currentworker = $datapoint->workerid; //WORKER IS WORKER NOW
    $currentproject=$datapoint->projectid; //PROJECT IS PROJECT NOW
    if ($firstmismatch==1) {//SPECIAL FIRST ROW ONLY
        $firstmismatch=0;
        //FIRST ROW IS NULL NOW
    } else {
        //PUT ROW INTO OUR OBJECTLIST FIRST
        $rows[] = $row;
        // $rows[] = $sumrow;
        // $rows[] = $emptyrow;
    }
    //THEN READ NEW DATA TO NEW ROW OBJECT
    $row = (object)[

```

```

'workerid' => $datapoint->workerid,
'projectid' => $datapoint->projectid,
'dent' => 'new', // NEW LINE
'desiredstate1' => null,
'actualstate1' => null,
'desiredstate2' => null,
'actualstate2' => null,
'desiredstate3' => null,
'actualstate3' => null,
'desiredstate4' => null,
'actualstate4' => null,
'projecttypename' => $datapoint->project->type->name ?? ' ',
'projectname' => $datapoint->project->name ?? $datapoint->pname,
'funding' => $datapoint->project->type->name ?? $datapoint->ptypename,
'drittmittel' => 0,
'kt' => $datapoint->project->kostentraeger->name ?? $datapoint->ktypename ?? ' ',
'eg' => $datapoint->eg,
'manhoursinamonth' => $datapoint->manhoursinamonth,
'sender' => 'default',
'firstname' => $datapoint->worker->firstname ?? $datapoint->firstname,
'lastname' => $datapoint->worker->lastname ?? $datapoint->lastname,
'teamname' => $datapoint->worker->team->name ?? $datapoint->tname,
];
//READING Q DATA DYNAMICALLY
switch ($datapoint->quarter->q ?? $datapoint->q) {
    case '1':
        $row->desiredstate1 = $datapoint->desiredstate;
        $row->actualstate1 = $datapoint->actualstate;
        break;
    case '2':
        $row->desiredstate2 = $datapoint->desiredstate;

```

```
        $row->actualstate2 = $datapoint->actualstate;
        break;
    case '3':
        $row->desiredstate3 = $datapoint->desiredstate;
        $row->actualstate3 = $datapoint->actualstate;
        break;
    case '4':
        $row->desiredstate4 = $datapoint->desiredstate;
        $row->actualstate4 = $datapoint->actualstate;
        break;
    default:
        break;
    }
}
}
$rows[] = $row;
return view('test', ['qes' => $rows, 'uienabled' => 'false' ]);
}
```