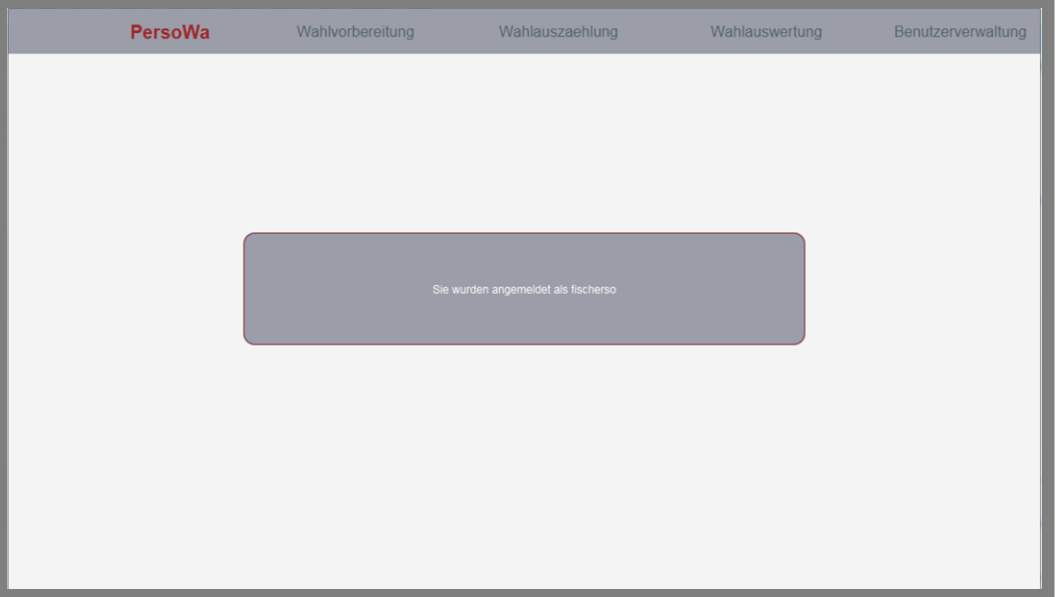
2016

PersoWa

Erweiterung der Anwendung „PersoWa“ um Funktionalitäten der Wahlauswertung zur Unterstützung der Personalratswahlen

Projektdokumentation – Abschlussprüfung Fachinformatiker für Anwendungsentwicklung Winter 2016/2017



XXXX

Inhalt

Kundendokumentation

1 Projektbeschreibung

Die folgende Projektdokumentation wurde im Rahmen des IHK Abschlussprojektes für Fachinformatiker Fachrichtung Anwendungsentwicklung von XXX im Zeitraum vom 31.Oktober 2016 bis 11.November 2016 erstellt.

1.1 Projektumfeld

Die Firma XXX, ist ein XXX. Ihr Hauptsitz liegt in XXX. Zu den Trägern gehören die XXXXMomentan beschäftigt XXX ca. XXX Mitarbeiter verteilt über XXX Standorte. XXX ist zudem das einzige Unternehmen deutschlandweit, XXX.

Der Auftraggeber dieses Projekts ist XXX selbst. Das Projekt wird in den Räumlichkeiten XXX unter der Anleitung der Abteilung „XXX“ durchgeführt.

1.2 Ausgangslage

Die Personalratswahlen finden alle vier Jahre statt. Der Prozess der Wahlvorbereitung, Wahlauszählung und Wahlauswertung ist stets mit einem hohen Einarbeitungsaufwand verbunden, da meist der Wahlvorstand variiert und sich dementsprechend Kenntnisse bzgl. der Wahlvorschriften angeeignet werden müssen. Die Auszählung findet bis dato händisch statt und muss für jeden Standort zusammengetragen werden. Die erforderlichen Dokumente wie Wahlniederschrift werden mittels Microsoft Word ohne Vorlage erstellt.
Um Kosten und Zeit zu sparen, sowie den Arbeitsaufwand bei jeder Wahl zu verringern, wurde sich dafür entschieden ein Programm zur Erleichterung der Wahl zu erstellen. Dieses wurde bereits in der Sprache Java als Desktopanwendung bis zur Wahlauswertung entwickelt. Durch Erweiterung der fachlichen Kenntnisse und Rücksprache mit erfahrenen Entwicklern wurde deutlich, dass zusätzlich zu der Entwicklung des Moduls der Wahlauswertung die bereits bestehenden Funktionalitäten im Rahmen eines Refactorings überarbeitet werden müssen.

1.3 Produktvision

In der Personalratswahl 2015 wurde entschieden, dass durch Auszubildende ein elektronisches, mehrbenutzerfähiges Wahlauswertungsprogramm erstellt werden soll. Der Einsatz des Programms beginnt mit dem Anlegen der Wahl. Dabei werden grundlegende Daten, wie zum Beispiel der Wahlgegenstand und das Jahr der Wahl, angegeben.
Anschließend wird das Wählerverzeichniss erstellt. Dieses wird aus dem Personalverwaltungsprogramm importiert. Die importierten Daten werden durch automatische Filter um die Merkmale Wahlberechtigung und Begründung ergänzt. Außerdem müssen die ANÜs manuell in das System eingegeben werden, da auch diese grundsätzlich wahlberechtigt sind.
Sollte eine gemeinsame Wahl durch Vorabstimmung beschlossen werden, so wird das entsprechende Merkmal für die Wahl im Programm umgestellt.
Geht ein externer Wahlvorschlag bei dem Wahlvorstand ein, so legt dieser den entsprechenden internen Wahlvorschlag an. Dafür kann er die auf dem Wahlvorschlag stehenden Bewerber aus dem

Wählerverzeichnis auswählen. Der Wahlvorschlag wird im System in verschiedenen Zuständen vorgehalten. So kann beispielsweise bei der Streichung eines Bewerbers auf Grund fehlenden passiven Wahlrechts eine Begründung im alten Wahlvorschlag angegeben werden. Bei der Prüfung eines Wahlvorschlags unterstützt das System inhaltlich nicht.
Ist die Prüfung aller eingegangenen Wahlvorschläge endgültig abgeschlossen und sind die Ordnungsnummern verteilt und eingegeben, erstellt das Programm den Stimmzettel. Nach der Stimmabgabe unterstützt das Programm bei der Auszählung. Bei der Auszählung werden händisch die Stimmzettel zunächst vorsortiert. Anschließend sind zwei Personen für die elektronische Erfassung zuständig. Hierfür kann ausgewählt werden, ob es sich bei der aktuellen Stimme um eine Listenstimme oder um eine Einzelstimme handelt. Anschließend können die Kreuze des Stimmzettels in das Programm eingegeben werden. Nach 50 eingegebenen Stimmzetteln weist das Programm darauf hin, dass ein Abgleich mit der manuellen Auszählung sinnvoll ist und bietet die Möglichkeit die eingegebenen Werte zu korrigieren. Ungültige Stimmen werden mit dem Grund der Ungültigkeit im Programm eingetragen.
Wenn die Auszählung abgeschlossen ist, kann mit der Auswertung begonnen werden. Für diese wird zunächst die Sitzverteilung ermittelt. Dies macht das Programm automatisch. Nur wenn eine Losentscheidung gefordert ist, meldet das Programm dies und erwartet die Eingabe der Losentscheidung. Mit der Sitzverteilung liegen anschließend alle notwendigen Daten vor und das Programm erzeugt die Wahlniederschrift.
Somit soll das Programm als Endprodukt bei der Vorbereitung der Wahl unterstützen, die Auszählung dokumentieren und fast vollständig die Auswertung mit der Wahlniederschrift übernehmen.

1.4 Projektabgrenzung

Das Programm „PersoWa“ wurde aufgrund des Umfangs bereits bis zu den Funktionalitäten der Wahlauswertung entwickelt. Hierbei wird sich an der bereits bestehenden Struktur des Programmes bedient und weiterentwickelt.

2 Projektplanung

2.1 Ist-Analyse

2.1.1 PersoWa Programmstruktur

Das Programm PersoWa wurde in der Sprache Java als Rich Client Desktopanwendung entwickelt und besteht aus insgesamt fünf Paketen. Im aufgeführten Diagramm erkennt man, dass das Programm nach dem etablierten MVC-Konzept aufgeteilt wurde. Um die Persistenz sicherzustellen, wird die relationale Datenbank JavaDB (Derby) auf einem zentralen Laufwerk genutzt. Die grafische Oberfläche wird mittels JavaFX dargestellt. Abgesehen von den MVC-üblichen Paketen, gibt es drei weitere:

- *test*

  Das Paket *test* beinhaltet eine Test-Suite, um die erstellten Klassen möglichst komfortabel mithilfe von JUnit-Tests zu testen.

- *guihandler*

  Das Paket *guihandler* dient zur Aufbereitung der Daten aus dem Controller, um sie visuell ansprechend darzustellen. Näheres siehe Punkt 2.1.3.

- *mapper*

  Das Paket *mapper* dient zur objektrelationalen Abbildung der Klassen. Diese werde in Punkt 2.1.2 näher erläutert.

Ein Überblick der gesamten Klassenstruktur und der Systemkontext befinden sich im Anhang. (Anhang 1 und 4)

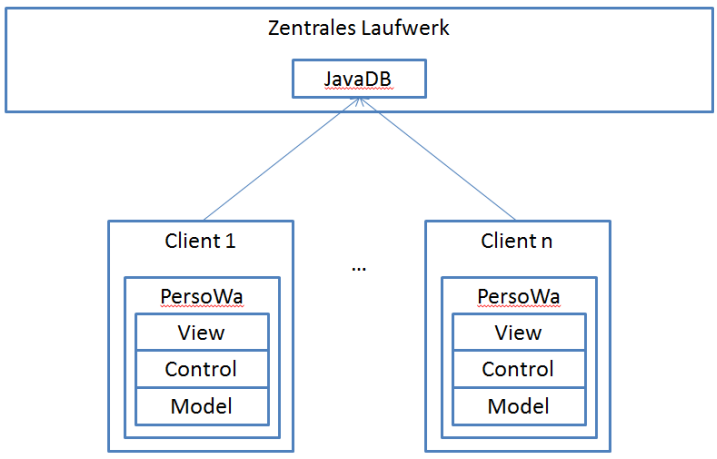


Abbildung 1: Architektur

2.1.2 Datenhaltungsschicht: Datenbank

Die objektrelationale Abbildung findet bei PersoWa nicht wie üblich über ein frei verfügbares Framework wie Hibernate statt, sondern wird über das Paket *mapper* mittels händisch erstellten Data Access Methoden behandelt. Dieses Paket beinhaltet für jede Klasse des Pakets *model* eine Klasse mit dem gleichen Namen mit dem Zusatz „Mapper“. In diesen Klassen befinden sich prepared SQL-Statements, um die erforderlichen Daten nach dem CRUD-Prinzip zu verarbeiten. Ein Beispiel hierfür findet sich im Anhang unter dem Abschnitt Quellcodeauszüge Punkt 5.3. Diese Art des ORM ist nicht mehr state-of-the-art und ist verbesserungswürdig. Ursprünglich wurde sie in dieser Form aus Lernzwecken implementiert. Aufgrund der zeitlichen Begrenzung des Projekts kann die Datenhaltungsschicht nicht komplett überarbeitet werden. Hierfür dient dann ein späteres Teilprojekt. Außerdem ist die Sicherheit der Daten aufgrund eines veralteten Verschlüsselungsalgorithmus des Passworts nicht mehr sicher. Dieser muss zusätzlich ersetzt werden.

2.1.3 Grafische Darstellung der Anwendung

Die grafische Darstellung findet über das Framework JavaFX statt. JavaFX gilt als Nachfolger von Java Swing und wurde von Oracle als Standard zur Visualisierung gesetzt. Dementsprechend wurde in dem Projekt JavaFX als Framework zur Darstellung der Anwendung genutzt. Hierfür wurden in dem Projekt in dem Paket View die entsprechenden FXML Dokumente erstellt. Diese Dokumente sind an der Auszeichnungssprache XML orientiert und dienen zur reinen Anordnung und Strukturierung der Elemente. Um diese Elemente mit Daten zu füllen, wurde jeweils eine Klasse in dem Paket guihandler angelegt. Die Aufteilung von Strukturierung und Verarbeitung der Daten erweist sich bei Wünschen zur Veränderung des Layouts als sinnvoll, da hier nur die FXML Dateien bearbeitet werden müssen.

Ein Beispiel für eine FXML Datei und dem dazu entsprechenden GUI Handler befinden sich im Anhang unter Punkt 5.4 und 6.

2.1.4 Fehlende Komponenten

Um den Anspruch der Produktvision gerecht zu werden, müssen die fehlenden Funktionalitäten der Wahlauswertung implementiert werden. Dazu gehören die Implementierung der Logik zur Berechnung der Sitzverteilung und die Erstellung der Wahlniederschrift, sowie die Erstellung der Oberflächenkomponenten. Hierbei muss §23 der „PersRWahlV SH 2008“[1] eingehalten werden und alle geforderten Informationen müssen bei der automatischen Erstellung der Wahlniederschrift enthalten sein.

[1] Juris Abkürzung. Ausgeschrieben: „Landesverordnung über die Wahl der Personalräte“

2.2 Zielbestimmung

PersoWa ist in seinem jetzigen Zustand produktiv nicht einsetzbar. Wie in 2.1.4 beschrieben, müssen die fehlenden Komponenten ergänzt werden, sowie ein Refactoring des bestehenden Programms durchgeführt werden. Dabei soll die Produktqualität durch Verbesserung der Wartbarkeit, Zuverlässigkeit und Änderbarkeit nach ISO/IEC 9126 gesteigert werden. Hierfür müssen neue Testfälle geschrieben, sich wiederholender Code ausgelagert und wie in Punkt 2.1.2 dargestellt, der Verschlüsselungsalgorithmus der Passwörter ausgewechselt werden.

2.3 Projektphasen mit Zeitplanung

Der Zeitrahmen für das Abschlussprojekt beträgt 70 Stunden. Diese wurde bereits vor Beginn des Projektes wie folgt in mehrere Phasen aufgeteilt:

Projektphase	Voraussichtliche Dauer
Planung	13h
Refactoring	19h
Implementierung	20,5h
Präsentation und Abnahmegespräch	6,5h
Dokumentation	11h

Tabelle 1: Zeitplanung

2.4 Kostenplanung und Amortisationsdauer

2.4.1 Kostenplanung

Bei der Berechnung der Gesamtkosten für das Projekt darf aus datenschutzrechtlichen Gründen kein interner Verrechnungssatz für einen Auszubildenden verwendet werden. Deshalb werde ich in der folgenden Rechnung einen fiktiven Stundensatz von 50€ verwenden. Zusätzlich wird ein fiktiver Wert für die Nutzung der Büroräume und sonstiger Kosten wie Strom auf 5€ pro Stunde gesetzt. Für die Präsentation und das Abnahmegespräch, sowie die fachliche Begleitung bei Fragen durch einen festangestellten Entwickler wird ein fiktiver Stundensatz von 120€ verwendet.

Projektphase	Stundenzahl	Stundensatz	Gesamt
Planung	13h	55€	**715€**
Refactoring	19h	55€	**1045€**
Implementierung	20,5h	55€	**1127,50€**
Präsentation und Abnahmegespräch	6,5h	55€	**357,50€**
Dokumentation	11h	55€	**605€**
Lohnkosten Festangestellter	3h	120€	**360€**
Gesamt	70h (+3h)		**4210€**

Tabelle 2: Kostenplanung

Die Gesamtkosten für das Projekt betragen nach den veranschlagten Kosten ca. **4210,00€.**

2.4.2 Amortisationsdauer

$$t = \frac{Anschaffungsausgabe}{durchschnittlicher\ Rückfluss\ pro\ Jahr}$$

Formel zur Berechnung der Amortisationsdauer

Für die Amortisationsrechnung müssen die Anschaffungsausgaben, in diesem Fall 4210€ durch den durchschnittlichen Rückfluss pro Jahr dividieren. Da es sich bei diesem Projekt um ein internes Teilprojekt handelt, gibt es keinen direkten monetären Gewinn. Unabhängig davon kann man aber für die automatisierten Prozesse eine Zeitersparnis pro Personalratswahl evaluieren. Hierbei handelt es sich um Schätzungen, die auf Erfahrungswerte der letzten Wahlen basieren.

Automatisierte Wahlauswertung:

- Projektaufwand: 4210 €
- Kosten alle vier Jahre für die Wartung: 55 € (55€ Stundensatz des Mitarbeiters * eine Stunde)

Manuelle Wahlauswertung:

- Kostenersparnis alle vier Jahre für die Auswertung: 2400 € (errechnet aus der Differenz alte Bearbeitungszeit – neue Bearbeitungszeit (20 Stunden * Stundensatz (120€))

Amortisationsrechnung:
Amortisationszeit = Anschaffungskosten / Gewinn
Amortisationszeit = 4210€ / ((600€/Jahr)/12 – (13,75 €/Jahr)/12)
Amortisationszeit = 4210€ / 50€ - 1,15€
Amortisationszeit = ~86 Monate
Amortisationszeit = **7 Jahre und 2 Monate**

Ergebnis:

- Die automatisierte Auswertung lohnt sich unter dem finanziellen Aspekt bereits nach zwei Wahlperioden. Hierbei ist zu beachten, dass nur das Teilprojekt zur Berechnung genommen wurde.

Abgesehen von dem monetären Gewinn wird durch das Refactoring die Sicherheit erhöht (näher erläutert im Punkt 3.2.1) und der Organisationsaufwand durch den standortübergreifenden Zugriff verringert.

2.5 Anwendungsfälle und Backlog mit User-Stories

In der folgenden Tabelle werden die Software-Anforderungen als User-Story angegeben.

Priorität	Epic	User-Story	Status	Verantwortlicher
1	Wahlauswertung	Als Wahlvorstand kann ich die Sitzverteilung bestimmen, damit diese bekanntgegeben werden kann.	Offen	XXX
2	Wahlauswertung	Als Wahlvorstand kann ich das Wahlergebnis bestimmen, damit dieses bekanntgegeben werden kann	Offen	XXX
3	Wahlauswertung	Als Wahlvorstand kann ich die Wahlniederschrift als PDF erzeugen lassen, damit §23 der Wahlordnung umgesetzt werden kann	Offen	XXX

Tabelle 3: Backlog mit User-Stories

Aus diesen Anforderungen werden spezifische Anwendungsfälle erstellt. Im Anhang unter Punkt 2.1 und 2.2 finden sich der für das Projekt exemplarische Anwendungsfall „Sitzverteilung bestimmen“ und das zugehörige Aktivitätsdiagramm.

2.6 Entwurf der Benutzeroberfläche

Beim Entwurf der Benutzeroberfläche wird sich an der bereits bestehenden Oberfläche orientiert. Hierbei wird das Corporate Design von XXX in den Farben Grau und Rot eingehalten. Da das Programm für die interne Verwendung bestimmt ist, wird auf ein funktionales Design gesetzt. Dabei soll die Ausgabe der Sitzverteilung übersichtlich und das Ergebnis leicht erkennbar sein.
Im Anhang befindet sich ein Mockup der GUI unter Punkt 10, welches mithilfe von BalsamiQ erstellt wurde.

2.7 Datenmodell

Auch beim Datenmodell wird sich an das bereits bestehende Modell von PersoWa gehalten.
Die Daten, die man zur Wahlauswertung benötigt, befinden sich bereits in der Datenhaltungsschicht.
Die erforderlichen Daten werden aus der bereits bestehenden JavaDB, die bereits in Punkt 2.1.2 näher erläutert wurde, bezogen.
Unter Berücksichtigung dieser Punkte muss das Datenmodell nicht erweitert, sondern zur Verwertung der Daten genutzt werden.

3 Projektdurchführung

3.1 Entwicklungsumgebung

Zur Entwicklung der Anwendung PersoWa wird das von XXX zur Verfügung gestellte Notebook mit folgender Hard- und Software verwendet:

- Intel Core i5-3340M CPU @2.70GHz
- 4 GB DDR RAM
- Windows 7 64-Bit Service Pack 1
- Eclipse Mars Release (4.5.0)
- Atlassian JIRA Project Management Software v.7.1.1 (bereitgestellt durch die interne Infrastruktur von XXX)
- Apache Derby (JavaDB)
- 100 Mbit/s LAN

Die hier ausgewählte Hard- und Software entspricht den XXX Standards. Die Nutzung anderer Software ist aufgrund von Sicherheitsstandards nicht ohne weiteres möglich. Hierfür wäre ein aufwendiger Genehmigungsprozess notwendig, dessen Nutzen dem dafür notwendigen Zeitaufwand weit unterliegt.
Eine Alternative wäre ein Desktoprechner mit folgender Hardware gewesen:

- Fujitsu-Siemens Celsius W340
- 2-4 GB DDR RAM
- Windows XP Professional SP3/Windows 7/Windows Server 2012

Dieser wird zu Testzwecken den Auszubildenden von XXX gestellt. Diese Testrechner dürfen aus Sicherheitsgründen keine Internetanbindung besitzen. Bei der Durchführung des Projekts ist es aber zwingend notwendig eine Internetanbindung zu haben, da während des Projekts Recherche und Lesen der Dokumentationen verschiedener Frameworks Alltag sind.
Da das zur Verfügung gestellte Notebook ohnehin als Arbeitswerkzeug genutzt wird und die Anforderungen des Projekts durch diese Hardware erfüllt werden, wird sich dafür entschieden, den Laptop zur Entwicklung zu nutzen.

3.2 Refactoring

3.2.1 Hash-Algorithmus der Datenbank

Der erste Teil des Refactorings besteht darin, den Hash-Algorithmus zur Verschlüsselung der Passwörter auszuwechseln. Bisher wird das Verschlüsselungsverfahren MD5 genutzt, welches mittlerweile als unsicher gilt. Das Problem bei MD5 ist der sogenannte Kollisionsangriff. Dabei kann der Angreifer versuchen, einen String mit dem gleichen Hashwert wie das gehashte Passwort zu erzeugen. Dabei muss sich der unverschlüsselte Inhalt nicht ähneln, sondern es geht nur darum einen String zu finden, der den gleichen Hashwert wie das Passwort hat. Mit diesem kann man sich dann unbefugten Zugriff zum System verschaffen.
Deshalb ist ein Wechsel des Hash-Algorithmus für den Datenbankcontroller unausweichlich, um weiterhin die Sicherheit zu gewährleisten.
Für diesen Zweck wird nach Empfehlung des BSI der SHA-2 Algorithmus mit einer Bit Länge von 256 ausgewählt.

„[…]*Andererseits werden die Verfahren bei der Verwendung längerer Schlüssel langsamer, so dass immer zu überlegen ist, welche Schlüssellänge unter Nutzen-/Leistungsgesichtspunkten angemessen ist. Als Faustregel für gute Verfahren (AES , HMAC - SHA -2/3, Serpent,...) und normalen Schutzbedarf gilt derzeit, dass die eingesetzten Schlüssel mindestens 100 Bit lang sein sollten.)“*[2]

Um diesen Hash-Algorithmus zu implementieren wird die Klasse *MD5Handler* überarbeitet. Hier befindet sich der Hash-Algorithmus zur Verschlüsselung der Passwörter. Java bietet über seine Security Klasse *MessageDigest* den SHA-256 Algorithmus an. Deshalb ist hier kein weiteres Framework notwendig. Nach Auswechseln des Algorithmus wird beim Start des Programmes folgender Fehler ausgegeben:

```
java.sql.SQLDataException: Beim Versuch, VARCHAR '5c496f9ce398c2b796a95184d6cb67c286b1747cacc64da3bc32a7adffa0&'
auf die Länge 32 zu kürzen, wurde ein Truncation-Fehler festgestellt.
```

Abbildung 2: Fehlermeldung

Nach Einsicht des Datenbankcontrollers wird ersichtlich, dass die Spalte *password* in der Datenbank auf eine Länge von 32 Zeichen beschränkt ist. Da sich die MD5 Hash Länge auf 32 Zeichen beschränkt, war das bisher ausreichend. Diese muss dann 64 Zeichen erweitert werden. Der Fehler ist damit behoben.

[2] https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02164.html

3.2.2 Erweiterung der Logging Funktion und der Test Suite

In PersoWa wird die von Java zur Verfügung gestellte Logger Klasse verwendet. Diese dient dazu, die Handlungen der Wahlbearbeiter zeitlich nachzuvollziehen. Insbesonders bei der Wahlauswertung bzw. Wahlvorbereitung ist dies wichtig, um Fehler bzw. Manipulationen aufzudecken. Die Logging Funktion wird über den *PersoWaController* gesteuert, welcher eine Datei im Quellordner anlegt. Diese Datei wird aus Sicherheitsgründen schreibgeschützt angelegt und kann nicht ohne Administratorenrechte verändert werden.

Die Logging Funktion wird bisher im Programm unzureichend genutzt und wird in Folge des Refactorings erweitert. In den Klassen *SHA256Handler*, *StimmzettelErzeuger* und *BekanntmachungWahlvorschlaegeErzeuger* werden Methodenaufrufe mit entsprechenden Meldungen des Loggers hinzugefügt. Ein Beispiel für die Ausgabe des Loggers findet sich im Anhang unter Punkt 9.

Um die Software PersoWa zu testen, befindet sich im Package *test* eine Test Suite. Diese Test Suite kopiert die aktuelle Datenbank und errichtet eine Verbindung zu dieser. Dadurch wird verhindert, dass beim Testen der Datenbankfunktionen (Create, Read, Delete und Update) kein Datenverlust entsteht. Nach Beendigung dieser Tests wird die Testdatenbank wieder vollautomatisch gelöscht. Diese Test Suite dient als Basis, um eine größtmögliche Testabdeckung zu ermöglichen. Zu dieser Test Suite werden einige neue Testfunktionen hinzugefügt, welche im Kapitel 3.4.1 näher erläutert werden.

3.3 Implementierung

3.3.1 Erstellung der Menüs und Oberflächenelemente

Hierfür werden in der Oberfläche die drei Elemente *Zwischenstand anzeigen*, *Sitzverteilung anzeigen* und *Wahlniederschrift erzeugen* zu dem Oberpunktmenü *Wahlauswertung* hinzugefügt. Für dieses Ziel werden in der *Rahmen* FXML die einzelnen Menüitems hinzufügt.

```
<MenuItem fx:id="zwischenStandAnzeigen" text="Zwischenstand anzeigen" />
<MenuItem fx:id="sitzverteilungAnzeigen" text="Sitzverteilung anzeigen" />
<MenuItem fx:id="wahlniederschriftErzeugen" text="Wahlniederschrift erzeugen" />
```

Anschließend werden im *RahmenController* die Elemente über die FXID deklariert und ActionHandler der Klasse übergeben.

```
wahlniederschriftErzeugen.setOnAction(this::wahlniederschriftErzeugen);

zwischenStandAnzeigen.setOnAction((event) -> {
                    parentController.wechsleZuScene("ZwischenStandAnzeigen");
            });
sitzverteilungAnzeigen.setOnAction((event) -> {
                    parentController.wechsleZuScene("SitzverteilungAnzeigen");
            });
```

Für die Szenen *ZwischenStandAnzeigen* und *SitzVerteilungAnzeigen* werden eigene Szenen erstellt, da diese aufwendigere Darstellungen benötigen. Um die Wahlniederschrift zu erzeugen, reicht eine einfache Meldung, die bereits für die Methoden *StimmzettelErzeugen* und *WählerverzeichnisErzeugen* genutzt wurden.

Für die beiden eigenen Szenen werden zwei GuiHandler angelegt, welche mit TableViews die Sitzverteilung bzw. den Zwischenstand ansprechend anzeigen. Ein Beispiel hierfür findet sich im Anhang unter dem Punkt 3 und Punkt 5.4 Quellcodeauszüge.

3.3.2 Implementierung der Sitzverteilung

Bei der Implementierung der Logik muss sich zuerst mit der Berechnung der Sitzverteilung auseinandergesetzt werden. Diese findet laut Wahlordnung über das D'Hondt Verfahren (s. D'Hondt Verfahren Wikipedia)[3] statt. Zuerst muss die Anzahl der Stimmen aus den einzelnen Stimmzetteln zusammengerechnet und anschließend (absteigend nach der Größe) sortiert werden. Dafür wird eine ArrayList<Double> erstellt. Da diese nur die zusammengerechneten Stimmen beinhaltet, wird zur Zuordnung der Parteien eine weitere ArrayList<ArrayList<Double>> erstellt.

```
ArrayList<ArrayList<Double>> list_parteien = new ArrayList<ArrayList<Double>>();
ArrayList<Double> list_stimmenanzahl = new ArrayList<Double>();
```

Durch diese zweidimensionale Liste, kann den einzelnen zusammengezählten Stimmen pro Person eine Partei zugeordnet werden. Nachdem dies geschehen ist, muss diese ArrayList nach dem D'Hondt Verfahren aufgeteilt werden. Hierfür werden die einzelnen Stimmergebnisse iterativ durch eine aufsteigende Reihe natürlicher Zahlen (1..n, wobei n für die Gesamtzahl der Sitze steht) geteilt. Zur Veranschaulichung dient die Tabelle 4:

Anzahl Sitze = 12;

Divisor	Partei A	Partei B	Partei C
1	416 (1*)	338 (2*)	246 (3*)
2	208 (4*)	169 (5*)	123 (7*)
3	138,7 (6*)	112,7 (8*)	82 (11*)
4	104 (9*)	84,5(10*)	61,5
5	83,2 (12*)	67,6	61,5

Tabelle 4: Beispielberechnung nach dem D'Hondt Verfahren

*Sitznummer

Auf die Darstellung weiterer Iterationen wurde aus Platzgründen verzichtet. Im „worst-case" müssen 12 Iterationen durchgeführt werden, wenn beispielsweise Partei A mit einem extrem hohen Stimmvorsprung führt

Sobald dies geschehen ist, muss die innere Liste absteigend sortiert werden, um die Sitze den einzelnen Parteimitgliedern herauszuarbeiten. Das geschieht über eine Sortiermethode, die sich an der Sortiermethode Insertionsort orientiert.

Im Anhang befinden sich unter den Punkt 5.1 Quellcodeauszüge und 5.2 Codeausschnitte.

[3] D'Hondt-Verfahren: https://de.wikipedia.org/wiki/D%E2%80%99Hondt-Verfahren

3.3.3 PDF-Generierung der Wahlniederschrift

Zur PDF-Generierung der Wahlniederschrift wird das bereits eingebundene Framework Apache FOP genutzt, welches die Möglichkeit besitzt XSL Dateien in das gängige PDF Format zu konvertieren. Bisher wird das Framework zur Erstellung der Wahlbekanntmachung und der Stimmzettel genutzt.
Als erstes muss hierfür ein Layout im XSL Format für die Wahlniederschrift erstellt werden. Hierbei müssen auf jeden Fall die wahlordnungsrelevanten Informationen enthalten sein, da die Wahl ansonsten für ungültig erklärt werden muss. Bei der grafischen Gestaltung des Dokuments wird sich an ein einfaches, aber übersichtliches Format gehalten, damit die wichtigen Informationen erkannt werden. Ein Ausschnitt dieser XLS-Datei befindet sich im Anhang unter Punkt 6.
Zusätzlich zu der XSL Datei wird eine neue Klasse mit dem Namen *WahlniederschriftErzeuger* in dem Controller Paket erstellt, welches die erforderlichen Daten sich aus der Datenbank/den anderen Klassen holt und dann den *FOPGenerator* zur Erstellung des PDF's übergibt.

```
<fo:block font-size="10pt">Zu wählen waren: <xsl:value-of select="AnzahlMitglieder"/> Personalratsmitglieder </fo:block>
```

Zur Datenübergabe wird das XSL Attribut value-of genutzt, die dann im Controller mithilfe des Select-Tags befüllt werden kann.
Auch hier wird die Logging Funktion hinzugefügt, die je nach Ausgang der Generierung eine entsprechende Meldung ausgibt.

3.4 Qualitätssicherung

3.4.1 Test-Suite

Die Test-Suite erstellt eine Testumgebung, um die Komponenten nicht nur einzeln, sondern auch im Integrationstest miteinander zu testen. Hierbei wird die Datenbank kopiert, um dessen Daten in einer realistischen Testumgebung testen zu können. Außerdem erstellt die Test-Suite Instanzen von jedem Mapper im Programm. Über die Test-Suite können so die einzelnen Testklassen auf die unterschiedlichen Mapper und deren Methoden zugreifen. Das ist notwendig, um die Interaktion zwischen den einzelnen Komponenten im Programm testen zu können. Die einzelnen Testklassen beinhalten einfache JUnit-assert Methodenaufrufe, wo die zu erwartenden mit den aufgetretenen Zuständen verglichen werden.
Unten abgebildet befindet sich eine solche Testklasse, in der man erkennt, wie über die TestSuite der Rollenmapper aufgerufen wird.

```
package de.dataport.persowa.test;

import static org.junit.Assert.*;

public class RolleMapperTest {

    @Test
    public void testFuegeRolleHinzu () throws SQLException {
        TestSuite.getRmapper().fuegeRolleHinzu("TestRolle");
        Rolle vergleich = TestSuite.getRmapper().gibAlleRollen().get(TestSuite.getRmapper().gibAlleRollen().size() - 1);

        assertEquals("TestRolle", vergleich.getRollenbezeichnung());
    }

    @Test
    public void testGibRolleNachID () throws SQLException {

        Rolle vergleich = TestSuite.getRmapper().gibRolleNachID(1);

        assertEquals("Beschaeftigter", vergleich.getRollenbezeichnung());
    }

}
```

Abbildung 3 Quellcode einer Testklasse

3.4.2 Anwendertest/Entwicklertest

Die JUnit-Tests können leider keine vollständige Testabdeckung anbieten. Gerade die Elemente in der View können nur schwer (wenn überhaupt) über das JUnit-Framework getestet werden. Dafür werden neben dem Testen mithilfe von JUnit-Tests zusätzlich industrieübliche Anwendertests durchgeführt. Hierbei wird vor allem die korrekte Funktionsweise der GUI-Elemente getestet, aber auch die Kriterien der Usability überprüft.
Auch bei der Programmierung des D‘Hondt Auswahlverfahrens wird die Sitzverteilung händisch ausgerechnet und das Ergebnis mit dem aus dem Programm verglichen. Hierbei wird dann schnell deutlich, ob sich in der Programmierlogik Fehler eingeschlichen haben.

3.4.3 Logging

Wie bereits in Punkt 3.2.2 erwähnt, werden wichtige Ereignisse im Programmverlauf geloggt. Dabei wird mithilfe der Logging-Klasse von der JavaSDK eine Text Datei im Quellordner erstellt, welche bestimmte Ereignisse mit Zeitstempel hinterlegt. Dies ist einerseits zum Debuggen hilfreich, da man erkennt in welchen Programmteil es Probleme gab, und bietet andererseits den Administratoren bei Streitfragen die Möglichkeit Handlungen nachzuvollziehen und zu reproduzieren. Zusätzlich werden die Meldungen mit einem Log-Level versehen, sodass man die Fehler nach Dringlichkeit unterscheiden kann. Hierbei werden folgende Log-Level genutzt:

- INFO – Ist eine reine Information und muss nicht näher betrachtet werden.
- SEVERE – Schwerwiegender Fehler; bitte prüfen!
- WARNING – Ein Fehler ist aufgetreten, der aber den regulären Programmfluss nicht stört.

Durch diese Aufteilung kann der teils umfangreiche Log schnell nach den wichtigen Meldungen sortiert werden.

4 Projektabschluss

4.1 Abnahme

PersoWa wurde in der fertigen Form am 15.11.2016 abgenommen. Dabei wurde die Testsuite ausgeführt, um die einzelnen Komponenten einzeln und im Verbund zu testen. Das Testergebnis findet sich unter Punkt 7 im Anhang. Zusätzlich wurde das Programm händisch von dem Projektverantwortlichen überprüft. Hierbei traten keine Fehler auf und die Erwartungen an PersoWa wurden erfüllt.

4.2 Soll-Ist Vergleich

Projektphase	erwartete Stundenzahl	tatsächliche Stundenzahl
Planung	13h	11h (-2h)
Refactoring	19h	17h (-2h)
Implementierung	20,5h	24,5h (+4h)
Präsentation und Abnahmegespräch	6,5h	4,5g (-2h)
Dokumentation	11h	13h (+2h)
Gesamt	70h	70h

Tabelle 5: Soll-Ist Vergleich

Die beiden Projektphasen Planung und Refactoring haben weniger Zeit in Anspruch genommen als erwartet. Hierbei spielte bei der Planung die Erstellung der Artefakte eine große Rolle, weil die einzelnen Tabellen und Modelle schneller zu erstellen waren als erwartet. Beim Refactoring gestaltete sich das Auswechseln des Hash-Algorithmus einfacher als gedacht. Bei der Abnahme wurde weniger Zeit für das Change Management geplant als benötigt. Die hier eingesparte Zeit wurde aber in der

Implementierungsphase zusätzlich verbraucht, da sich die Implementierung der Geschäftslogik zeitintensiver als geplant gestaltete. Außerdem wurde für die Dokumentation zwei Stunden zusätzlich an Aufwand erbracht. Insgesamt konnte die Zeitplanung eingehalten werden.

Priorität	Epic	User-Story	Status	Verantwortlicher
1	Wahlauswertung	Als Wahlvorstand kann ich die Sitzverteilung bestimmen, damit diese bekanntgegeben werden kann.	Erledigt	XXX
2	Wahlauswertung	Als Wahlvorstand kann ich das Wahlergebnis bestimmen, damit dieses bekanntgegeben werden kann	Erledigt	XXX
3	Wahlauswertung	Als Wahlvorstand kann ich die Wahlniederschrift als PDF erzeugen lassen, damit §23 der Wahlordnung umgesetzt werden kann	Erledigt	XXX

Tabelle 6: Backlog

Die im Backlog hinterlegten User-Stories sind mit Projektabschluss als Erledigt zu kennzeichnen, da alle gewünschten Funktionalitäten der Wahlauswertung erfolgreich implementiert wurden.

4.3 Dokumentation

Zusätzlich zu dieser Projektdokumentation wurde für die Anwender des Programmes eine Anwenderdokumentation zur Benutzung der Wahlauswertungskomponente angefertigt. Diese befindet sich im Anhang zu dieser Dokumentation.

Durch die ausreichende Dokumentation des Codes, kann über JavaDoc automatisch eine Entwicklerdokumentation generiert werden. Diese beinhaltet detaillierte Informationen zu den einzelnen Klassen und Methoden.

Ein Auszug dieser Dokumentation befindet sich im Anhang unter Punkt 8.

4.4 Fazit

Abschließend wurde das Teilprojekt „Erweiterung der Anwendung „Persowa“ um Funktionalitäten der Wahlauswertung zur Unterstützung der Personalratswahlen“ erfolgreich durchgeführt. Hierbei wurden inhaltliche und zeitliche Anforderungen erfüllt.

Durch das Projekt wurden sich weitere Kenntnisse über das Unit-Testing, Protokollieren von Programmereignissen, sowie den Auszeichnungssprachen XML/XSLT angeeignet. Außerdem wurden die bereits vorhandenen Kenntnisse in der OOP und diversen Softwarepatterns wie MVC und Singleton vertieft.

Das Ziel des Projekts, die Software PersoWa fertigzustellen, ist hiermit erreicht.

In der Zukunft sind mögliche Erweiterungen denkbar. Da das Programm strikt nach dem MVC Pattern erstellt wurde, sind Änderungen an dem Programm ohne größeren Aufwand möglich.

Glossar

ANÜ	Arbeitnehmerüberlassung – Externe Mitarbeiter von XXX
Artefakt	Arbeitsergebnisse innerhalb eines Projekts wie z.b. Use-Case Diagramm
BSI	Bundesamt für Sicherheit in der Informationstechnik
Corporate Design	Erscheinungsbild eines/r Unternehmens/Organisation
CRUD	Create-Read-Update-Delete; Grundlegende Datenbankoperationen
Epic	Beschreibung einer Anforderung auf übergeordneter Ebene
Framework	Programmiergerüst
FXML	XML basierte Auszeichnungssprache für JavaFX
GUI	Graphical User Interface – Grafische Benutzerschnittstelle
Hibernate	Open Source ORM Framework für Java
ISO/IEC 9126	Norm um Softwarequalität sicherzustellen
Java	Eine moderne Programmiersprache
Java Swing	Veraltetes Framework zur Darstellung von Programmen
JavaDB	Relationale Datenbank
JavaDoc	Ist Bestandteil der Java SDK und bietet eine automatische Generierung der Entwicklerdokumentation
JavaFX	Framework zur Darstellung von Programmen
JavaSDK	Java Software Development Kit – Von Oracle bereitgestelltes zur Entwicklung mit Java
JIRA	Software zur Vorgangs- und Projektverfolgung
JUnit	Framework zum Testen einer in Java geschriebenen Software
Mockup	Visueller Entwurf einer Oberfläche
MVC	Model-View-Controller ist ein Software Design Pattern, der aktuell als Standard in der Softwareprogrammierung gilt
ORM	object-relational mapping ist eine Technik der Softwareentwicklung, um Objekte in einer Datenbank ablegen zu können
PDF	Plattformunabhängiges Datenformat (meist für Dokumente)
Refactoring	Strukturverbesserung von Quelltexten unter Beibehaltung des beobachtbaren Programmverhaltens
String	Zeichenkette
Test-Suite	Bestandteil des JUnit-Frameworks. Bietet die Möglichkeit mehrere Test Fälle auf einmal durchzuführen
User-Story	Drücken im Rahmen der agilen Softwareentwicklung eine Software-Anforderung aus
XML	Extensible Markup Language – Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten
XSLT	Programmiersprache zur Transformation von XML Dokumenten
BalsamiQ	Tool zur Erstellung von Mockups

Tabellenverzeichnis

Abbildungsverzeichnis

Quellenverzeichnis

Literatur:

- BSI - https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02164.html - Letzter Zugriff am 09.11.2016

- Wikipedia –
  https://de.wikipedia.org/wiki/JavaFX - Letzter Zugriff am 09.11.2016
  https://de.wikipedia.org/wiki/Model_View_Controller - Letzter Zugriff am 09.11.2016
  https://de.wikipedia.org/wiki/Objektrelationale_Abbildung - Letzter Zugriff am 09.11.2016
  https://de.wikipedia.org/wiki/Wasserfallmodell - Letzter Zugriff am 09.11.2016
  https://docs.oracle.com/javase/7/docs/api/ - Letzter Zugriff am 09.11.2016
  https://de.wikipedia.org/wiki/Amortisationsrechnung - Letzter Zugriff am 09.11.2016

- *Andreas Spillner, Tilo Linz: Basiswissen Softwaretest. Dpunkt.verlag, 2004*

Software:

- Eclipse
  https://eclipse.org/downloads/

- Jira
  https://de.atlassian.com/software/jira

- FOP
  https://xmlgraphics.apache.org/fop/

Eidesstattliche Erklärung

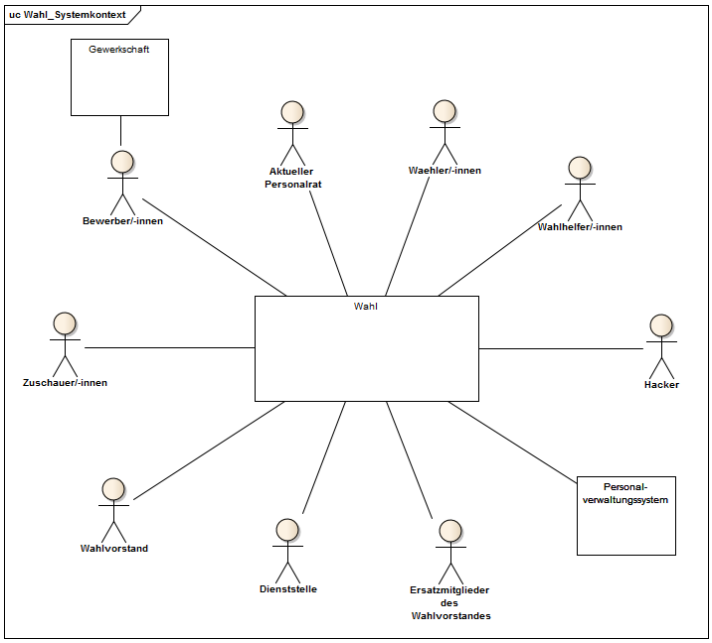
Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig und nur unter Zuhilfenahme der ausgewiesenen Hilfsmittel angefertigt habe.
Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach anderen gedruckten oder im Internet verfügbaren Werken entnommen sind, habe ich durch genaue Quellenangaben kenntlich gemacht.

Ort, Datum

XXX

9 Anhang

Anhang 1 Dokumentation des Systemkontexts der PR-Wahl



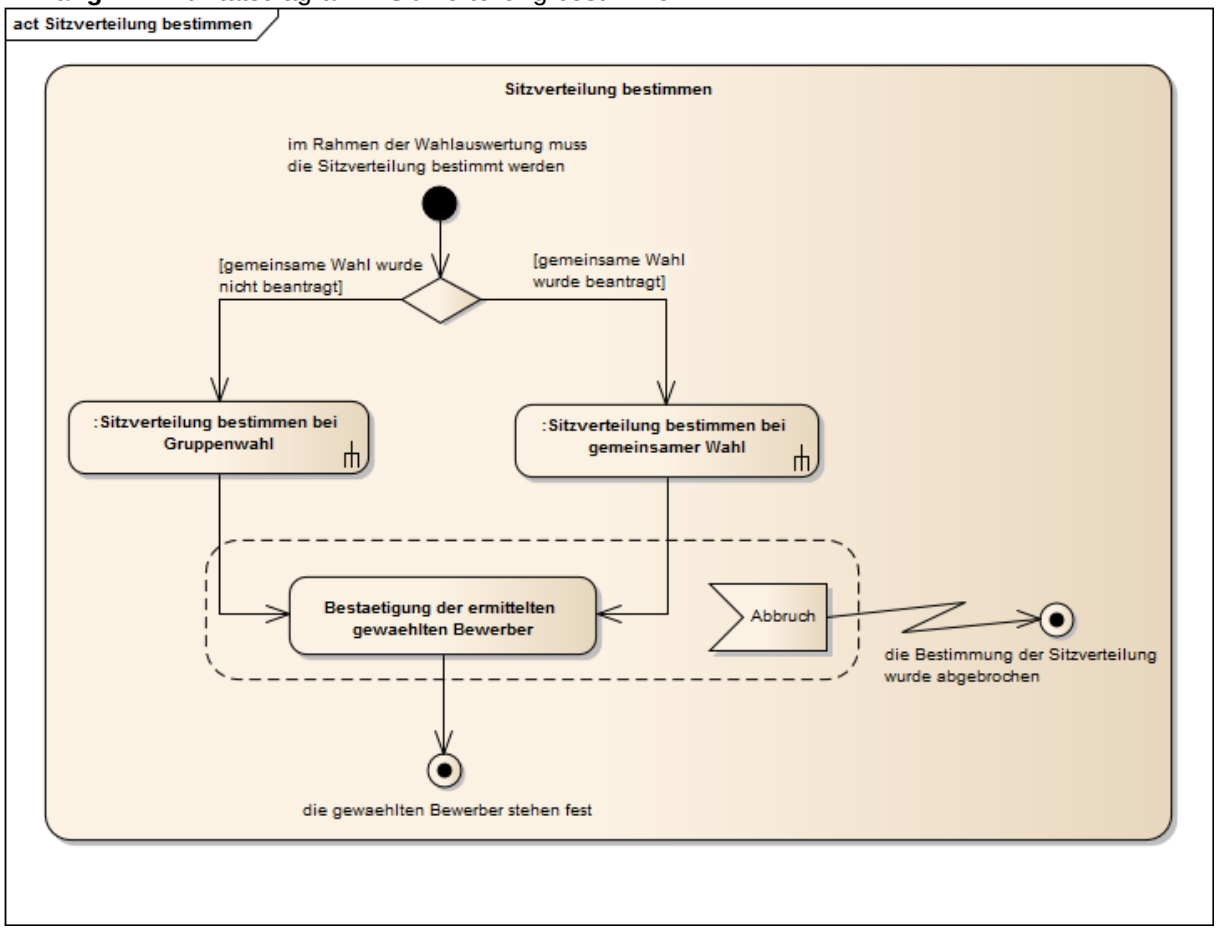
Die Auszählung erfolgt durch die Wahlhelfer, den Wahlvorstand und die Ersatzmitglieder des Wahlvorstandes, indem alle Stimmzettel auf 2n + 1 Stapel sortiert werden. Die Stapel sind dabei n Stapel für Einzelstimmen (einen für Stimmzettel mit Stimmen für Wahlvorschlag 1, einen für Stimmzettel mit Stimmen für Wahlvorschlag 2 und ohne Stimmen für Wahlvorschlag 1, einen für Stimmzettel mit Stimmen für Wahlvorschlag 3 und ohne Stimmen für Wahlvorschlag 1 und 2,…), n Stapel für Listenstimmen (einen Stapel pro Wahlvorschlag) und ein Stapel für die Stimmzettel, die den Verdacht der Ungültigkeit erregen. Anschließend können die Stimmzettelergebnisse in das System eingegeben werden. Die Eingabe kann gegebenenfalls korrigiert werden. Nach 50 Stimmzetteln soll korrigiert werden können. Die Stimmengültigkeit darf nur vom Wahlvorstand bewertet werden. Ebenso führt der Wahlvorstand die Auswertung durch. Dabei bestimmt er die Sitzverteilung und erzeugt die Wahlniederschrift.

Anhang 2 Anwendungsfälle und Aktivitätsdiagramme

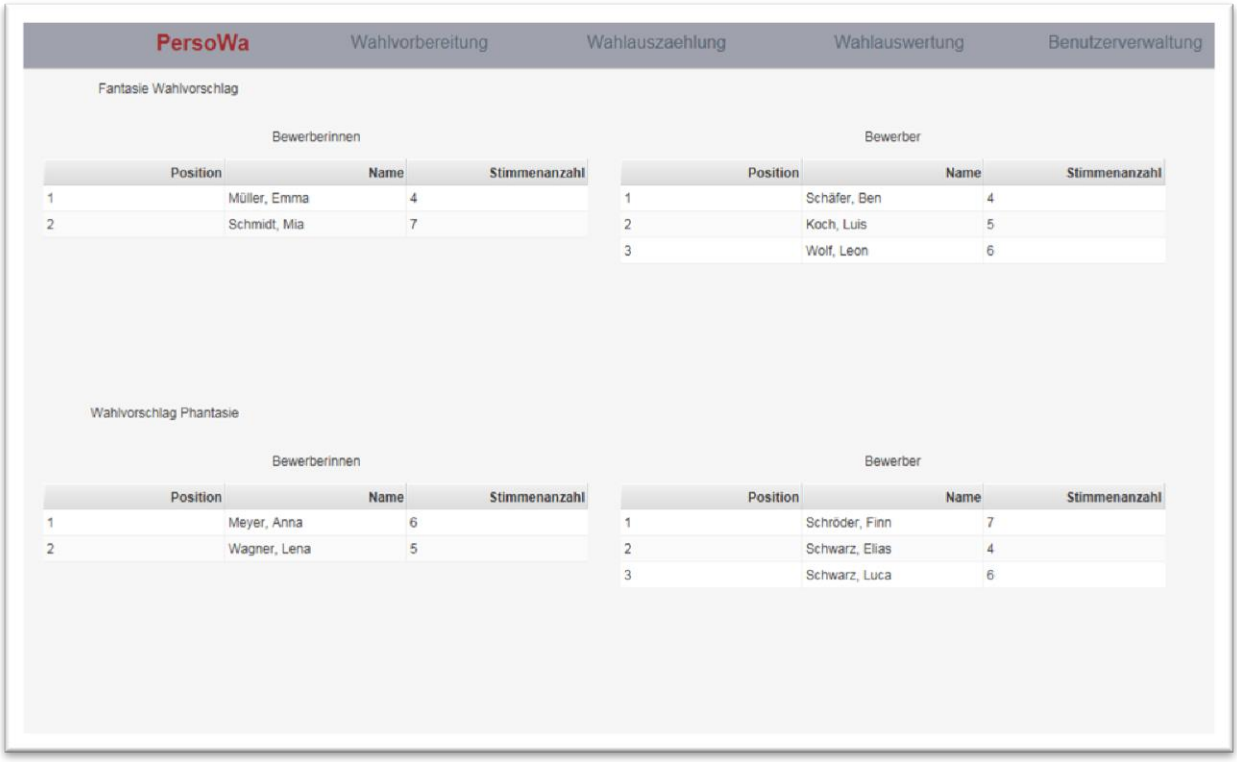
Anhang 2.1 Use-Case Sitzverteilung bestimmen

Name des Use-Cases:	Sitzverteilung bestimmen
Kurzbeschreibung:	Die Besetzung des Personalrats bei einer gemeinsamen Wahl wird ermittelt
Akteure:	Wahlvorstand, Ersatzmitglieder des Wahlvorstandes
Vorbedingungen:	Die Wahlauszählung ist abgeschlossen.
Fachlicher Auslöser:	Die gemeinsame Wahl wurde beantragt
Eingehende Daten:	Ergebnis der Wahl
Normalfall (essenzielle Schritte):	Die Sitzverteilung bei Gruppenwahl wird ermittelt
Ausnahmefälle:	Die gemeinsame Wahl wurde beantragt: Die Sitzverteilung bei gemeinsamer Wahl wird ermittelt
Nachbedingung:	Die Sitzverteilung wurde bestimmt und liegt im System vor.
Nicht-funktionale Aspekte	Netzwerkübergreifend, hohe Sicherheit (passwortgeschützt / Log-In) Hohe Performance nicht dringend notwendig, zuverlässige Funktionalität hat höchste Priorität.

Anhang 2.2 Aktivitätsdiagramm Sitzverteilung bestimmen



Anhang 3 Screenshot



Anhang 4 Klassenstruktur

Hier war ein Klassenstrukturauszug

Anhang 5 Quellcodeauszüge

Anhang 5.1 Zusammenzählung der Stimmen und Berechnung der Sitze nach D'Hondt Verfahren

```
public ArrayList<Double> berechneSitze(List<Stimmzettelergebnis> parteien) throws SQLException {
    ArrayList<Double> gesamtzahlstimmen = new ArrayList<Double>();
    double stimmenanzahl = 0;
    ArrayList<Integer> id = new ArrayList<Integer>();
    ArrayList<Wahlvorschlag> validierteWahlvorschlaege = (ArrayList<Wahlvorschlag>) getController().getwMapper()
            .gibValidierteWahlvorschlaege(getController().isBeamte());
    for (Wahlvorschlag v : validierteWahlvorschlaege) {
        List<Bewerber> bewerberAufWahlvorschlag = v.getBewerberliste();
        id.add(v.getId());
        for (Bewerber aktuellerBewerber : bewerberAufWahlvorschlag) {
            stimmenanzahl += aktuellerBewerber.getBeschaeftigter().getStimmenkorrektur();
            for (Stimmzettelergebnis aktuellerStimmzettel : parteien) {
                if (getController().getseMapper().istBewerberAufStimmzettel(
                        aktuellerBewerber.getBeschaeftigter().getId(), aktuellerStimmzettel.getId())) {
                    stimmenanzahl++;
                }
            }
        }
        gesamtzahlstimmen.add(stimmenanzahl);
        stimmenanzahl = 0;
    }
    return dHondt(gesamtzahlstimmen, sitzeanzahl, id);
}

public ArrayList<Double> dHondt(ArrayList<Double> list_gesamtzahl, int anzahl_sitze, ArrayList<Integer> id) {
    ArrayList<ArrayList<Double>> list_parteien = new ArrayList<ArrayList<Double>>();
    ArrayList<Double> list_stimmenanzahl = new ArrayList<Double>();
    for (int i = 0; i < list_gesamtzahl.size(); i++) {
        for (int a = 1; a <= anzahl_sitze; a++) {
            ArrayList<Double> div = new ArrayList<Double>();
            div.add((double) id.get(i));
            div.add((double) (list_gesamtzahl.get(i).doubleValue() / a));
            list_parteien.add(div);
        }
    }
    list_parteien = sort(list_parteien);
    for (int i = 0; i < list_gesamtzahl.size(); i++) {
        list_stimmenanzahl.add(0.0);
    }
    for (int i = list_parteien.size() - 1; i >= ((list_parteien.size()) - anzahl_sitze); i--) {
        list_stimmenanzahl.set((int) (list_parteien.get(i).get(0) - 1),
                list_stimmenanzahl.get((int) (list_parteien.get(i).get(0) - 1)) + 1);
    }
    return list_stimmenanzahl;
}
```

Anhang 5.2 Sortierverfahren einer zweidimensionalen ArrayList

```
private ArrayList<ArrayList<Double>> sort(ArrayList<ArrayList<Double>> list_stimmen) {
    ArrayList<Double> temp = new ArrayList<Double>();
    for (int i = 1; i < list_stimmen.size(); i++) {
        temp = list_stimmen.get(i);
        int j = i;
        while (j > 0 && list_stimmen.get(j - 1).get(1) > temp.get(1)) {
            list_stimmen.set(j, list_stimmen.get(j - 1));
            j--;
        }
        list_stimmen.set(j, temp);
    }
    return list_stimmen;
}
```

Anhang 5.3 BeschaeftigtenMapper

```
public void fuegeBewerberHinzu(Beschaeftigter bewerber) throws SQLException {
    int nextFreeID = -1;
    Statement statement = getConnection().createStatement();
    ResultSet resultSet = statement.executeQuery("SELECT MAX(id) FROM Beschaeftigter");
    if (resultSet.next()) {
        nextFreeID = resultSet.getInt(1) + 1;
    }
    PreparedStatement preparedStatement = getConnection().prepareStatement(
            "INSERT INTO Beschaeftigter (id,vorname,nachname,geschlecht,gruppe,betriebsstaette,geburtsdatum,"
                    + "stimmenkorrektur,rollenID, pasWahlrecht, aktWahlrecht, kurzzeichen, personalnummer, "
                    + "bemerkung) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

    preparedStatement.setInt(1, nextFreeID);
    preparedStatement.setString(2, bewerber.getVorname());
    preparedStatement.setString(3, bewerber.getNachname());
    preparedStatement.setString(4, bewerber.getGeschlecht());
    preparedStatement.setString(5, bewerber.getGruppe());
    preparedStatement.setString(6, bewerber.getBetriebsstaette());
    preparedStatement.setDate(7, new java.sql.Date(bewerber.getGeburtsdatum().getTimeInMillis()));
    preparedStatement.setInt(8, bewerber.getStimmenkorrektur());
    preparedStatement.setInt(9, bewerber.getRolle().getId());
    preparedStatement.setBoolean(10, bewerber.isPasWahlrecht());
    preparedStatement.setBoolean(11, bewerber.isAktWahlrecht());
    preparedStatement.setString(12, bewerber.getKurzzeichen());
    preparedStatement.setString(13, bewerber.getPersonalnummer());
    preparedStatement.setString(14, bewerber.getBemerkung());

    preparedStatement.execute();
    preparedStatement.close();
}
```

Anhang 5.4 GuiHandler ZwischenStandAnzeigen

```
public void fuegeWahlvorschlagHinzu(Wahlvorschlag wahlvorschlag, List<Stimmzettelergebnis> stimmzettel, int id) {
    Label wahlvorschlagName = new Label();
    wahlvorschlagName.setText(wahlvorschlag.getName());
    boxFuerTableViews.getChildren().add(wahlvorschlagName);

    HBox hbox = new HBox();
    hbox.getStyleClass().addAll("hbox");

    TableView<BeschaeftigterIntegerInteger> tableViewBewerberinnen = new TableView<BeschaeftigterIntegerInteger>();
    tableViewBewerberinnen.setPlaceholder(new Label(""));
    tableViewBewerberinnen.setMaxHeight(150);
    tableViewBewerberinnen.setSelectionModel(null);
    tableViewBewerberinnen.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
    try {
        fuelleTableViews(tableViewBewerberinnen, wahlvorschlag, stimmzettel, id);
        spaltenbeschriftungen(tableViewBewerberinnen);

    } catch (SQLException e) {
        getController().wechsleZuScene("Fehler: Die Tabelle konnte nicht gefüllt werden.", false);
        getController().getLog().log(Level.INFO, "Die Tabelle konnte nicht gefüllt werden.");
    }

    VBox bewerberinnen = erstelleVBoxMitLabelUndTableView("Sitzverteilung", tableViewBewerberinnen);

    hbox.getChildren().add(bewerberinnen);
    boxFuerTableViews.getChildren().add(hbox);

}

private VBox erstelleVBoxMitLabelUndTableView(String label, TableView<BeschaeftigterIntegerInteger> tableView) {
    VBox bewerber = new VBox();
    bewerber.getStyleClass().addAll("vbox");
    Label lbl_bewerber = new Label();
    lbl_bewerber.setText(label);
    lbl_bewerber.setId("overTable");
    bewerber.getChildren().add(lbl_bewerber);
    bewerber.getChildren().add(tableView);
    return bewerber;
}
```

Anhang 6 XSL-Stylesheet Auszug

Hier war ein XSL Auszug

Anhang 7 Testergebnis

Hier war ein Testergebnis

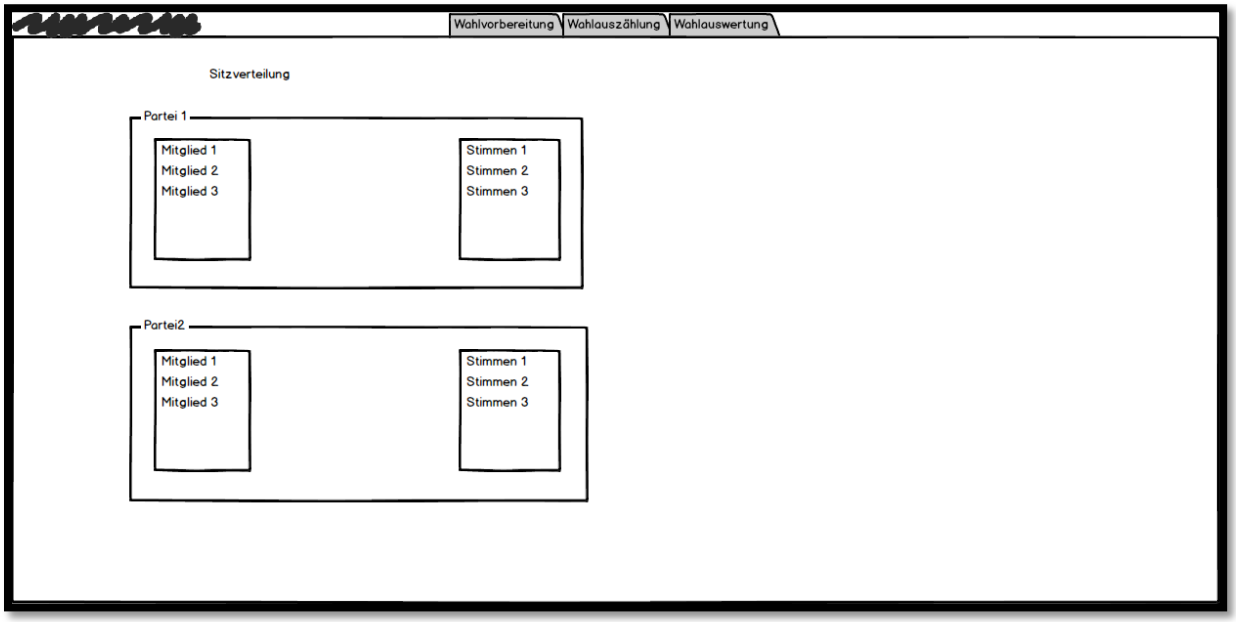
Anhang 8 JavaDoc Auszug

Hier war ein Javadoc Auszug

Anhang 9 Logger Auszug

Hier war ein Logger Auszug

Anhang 10 Mockup der GUI



PersoWa – Wahlauswertung

Benutzerdokumentation

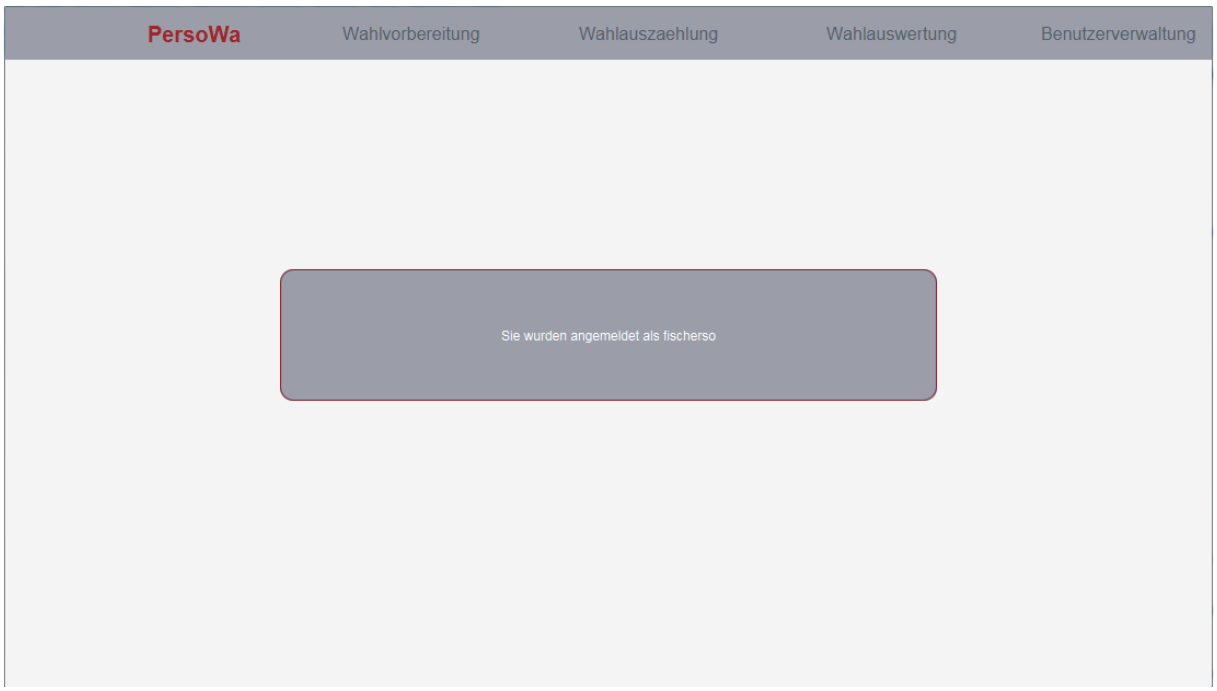
1 Einleitung

Dieser Teil der Benutzerdokumentation richtet sich nur an die Anwendergruppe **Wahlvorstand und Administratoren.**

Die Wahlauswertung findet erst bei erfolgreichem Abschluss der Wahlauszählung statt.

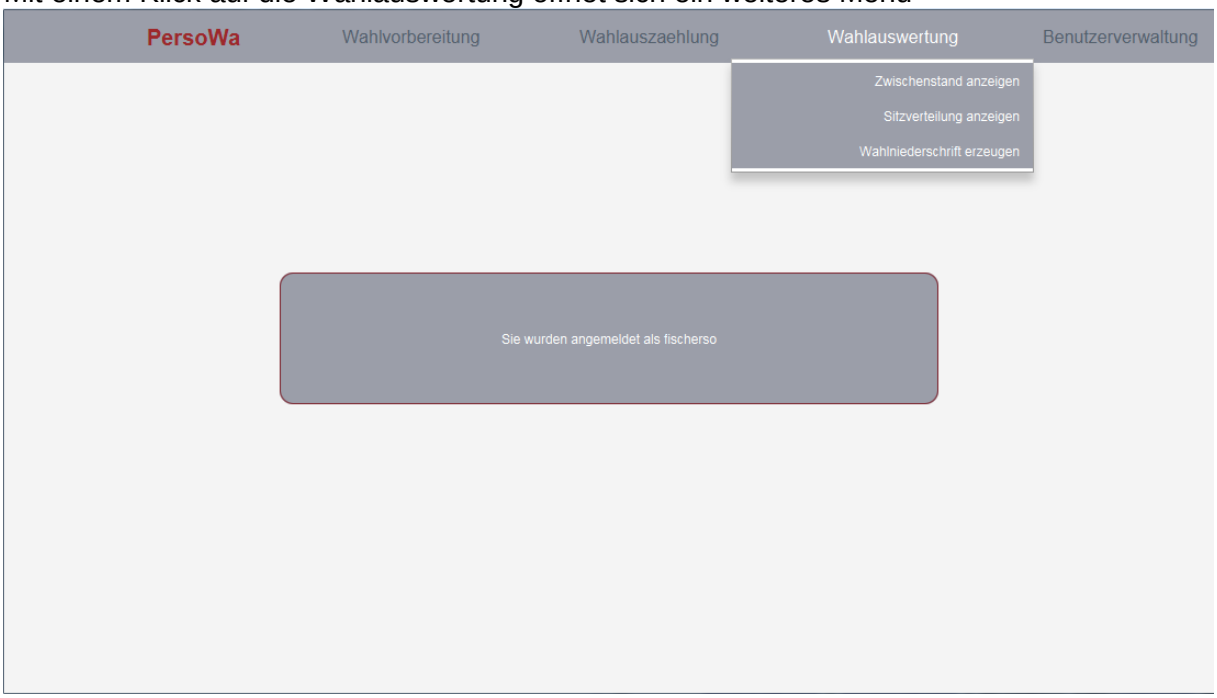
2 Oberfläche

Um zu der Oberfläche der Wahlauswertung zu gelangen, starten Sie PersoWa wie gewohnt und loggen sich mit ihren gültigen Zugangsdaten ein.

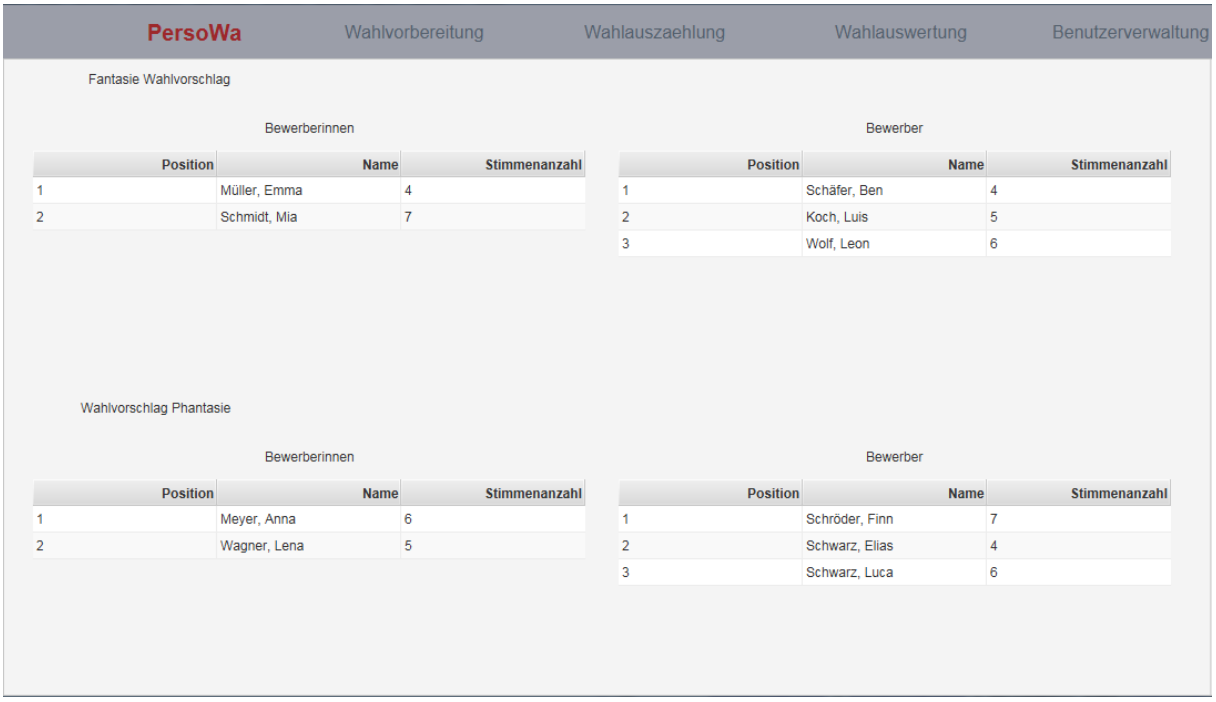


Sobald die Anmeldemaske erscheint, ist oben rechts der Reiter Wahlauswertung verfügbar (Achtung! Dies gilt nur für den Wahlvorstand und Administratoren. Als Wahlhelfer sehen Sie diesen Menüpunkt nicht!)

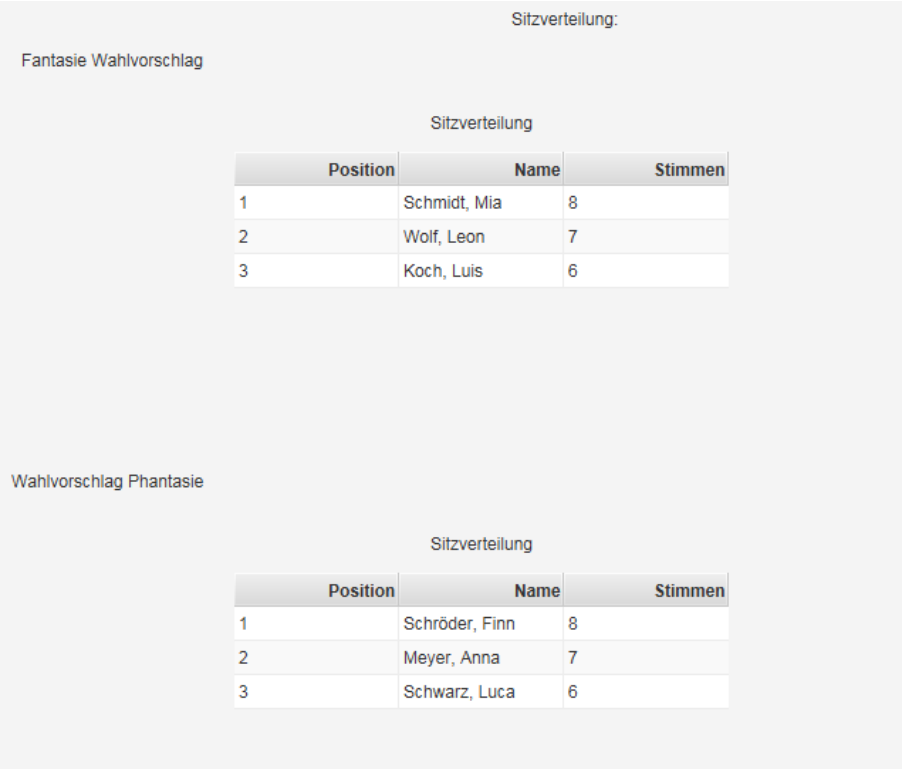
Mit einem Klick auf die Wahlauswertung öffnet sich ein weiteres Menü



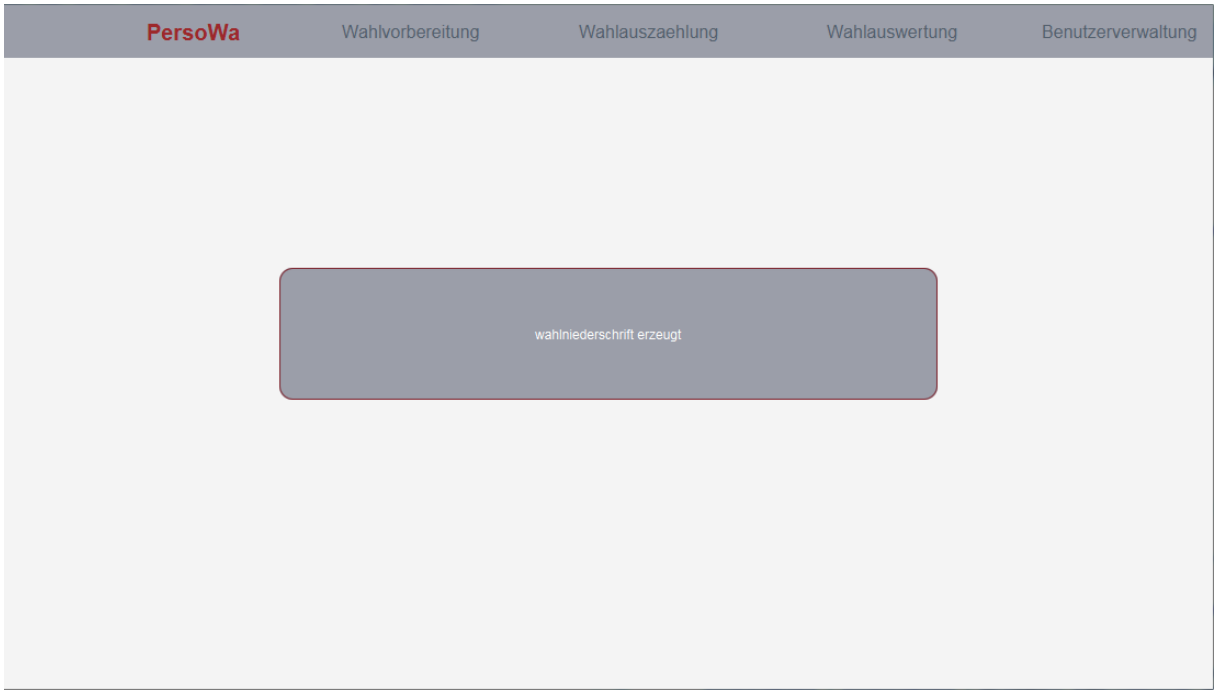
Dort haben Sie die Unterpunkte „Zwischenstand anzeigen“, „Sitzverteilung anzeigen“ und „Wahlniederschrift erzeugen“ zur Verfügung. Mit einem weiteren Klick auf die Fläche Zwischenstand Anzeigen, bekommen sie den aktuellen Zwischenstand der Wahl mit der Anzahl der Stimmen zurück.



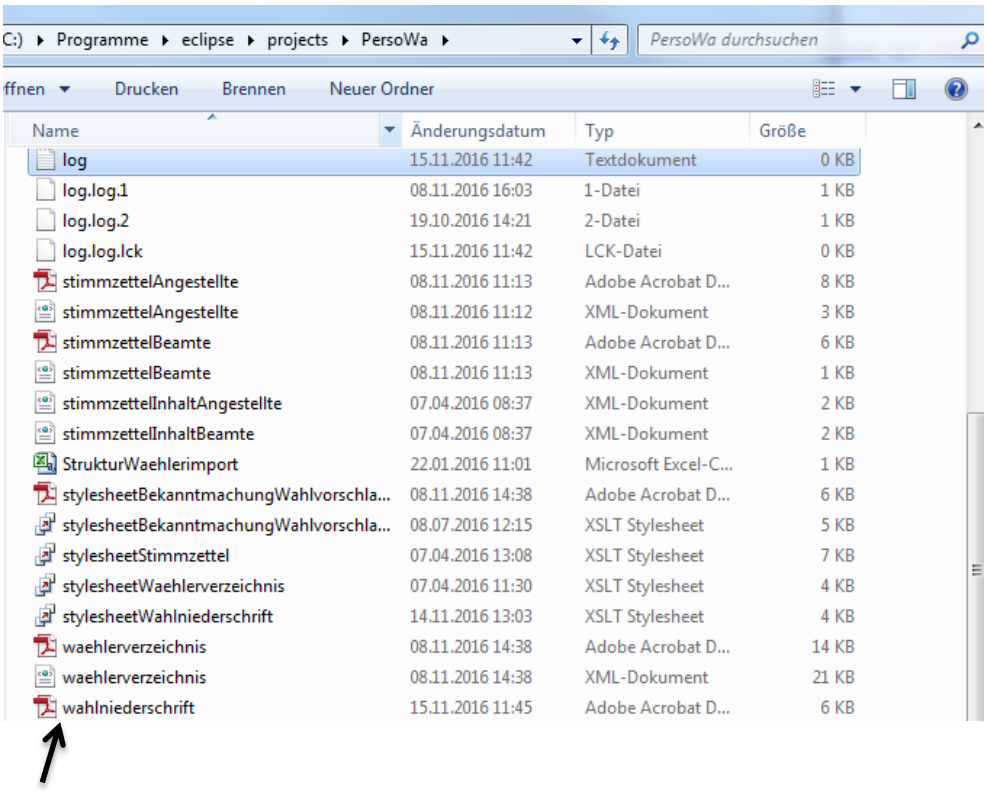
Sollte die Auszählung abgeschlossen sein und mit der händischen Auszählung übereinstimmen, können Sie sich nun die Sitzverteilung anzeigen lassen.



Hier wird Ihnen die Sitzverteilung ausgerechnet nach dem d'Hondt Verfahren angezeigt. Wenn Sie nun aus den aktuellen Daten ihre Wahlniederschrift erzeugen lassen möchten, öffnen Sie das Wahlauswertungsmenü und klicken die Schaltfläche „Wahlniederschrift erzeugen"



Die erzeugte Wahlniederschrift befindet sich nun in dem Quellordner Ihres Programms.



Wahlniederschrift

3 Fehlerbehandlung

Meldung – Wahlniederschrift konnte nicht erzeugt werden

Wenn Sie die Meldung „Wahlniederschrift kann nicht erzeugt werden“ bekommen, überprüfen Sie ob das Dokument bereits geöffnet ist. Falls ja, schließen Sie das Dokument und versuchen Sie erneut über das Menü die Wahlniederschrift zu erzeugen. Wenn das Problem weiterhin besteht, wenden Sie sich bitte an Ihren Administrator.