

PROJEKTDOKUMENTATION CAMUNDA

TEAM NR. 04

Projektmitglieder

Fiebelkorn, Christian (PL)	s0558587
Fröhlich, Steffen	s0559875
Fidan, Handan	s0558029

INHALT

Ausgangssituation	3
Prozessbeschreibung	3
Ergebnisse der Spezifikation der WF-Anwendung	3
Formulardesign	3
Fachlicher Prozess mit Camunda	4
Testfälle	7
Dokumentation der Ergebnisse der WF-Implementierung	9
Anwenderdokumentation	9
Entwicklerdokumentation	11
Projektplan und –Abrechnung	21
Abbildungs- und tabellenverzeichnis	21
Abbildungsverzeichnis	21
Tabellenverzeichnis	21

1. AUSGANGSSITUATION

1.1. PROZESSBESCHREIBUNG

Zu Prozessbeginn gibt ein Sachbearbeiter die Mietvertragsnummer an, zu der die Miete erhöht werden soll. Sodann wird entsprechend §558 Abs. 1 BGB geprüft, ob die letzte Mieterhöhung mindestens 12 Monate zurück liegt. Ist dies nicht der Fall, wird der Prozess beendet. Im Anschluss wird die rechtlich maximal mögliche Mieterhöhung nach Rechtslage sowie nach "Bündnis für soziale Wohnungspolitik und bezahlbare Mieten" ermittelt. Sollte die niedrigste Begrenzung unter 5 Euro liegen, wird der Prozess beendet. Nun wird ein Mieterhöhungsverlangen über die geringst mögliche Mieterhöhung verfasst und dem Hauptmieter per E-Mail zugesandt. Nun wartet der Prozess auf das Eintreffen des Antwortschreibens in Papierform (rechtsgültige Unterschrift). Sollte diese nach 6 Wochen nicht eingetroffen sein, wird dem Mieter eine Erinnerung zugesandt. Trifft nach weiteren 4 Wochen keine Antwort ein, wird der Vorgang in Form eines neu gestarteten Prozesses an die Rechtsabteilung übergeben. Sendet der Mieter eine Ablehnung, wird der Vorgang ebenfalls der Rechtsabteilung übergeben. Trifft eine Zustimmung ein, wird die Mieterhöhung in der Datenbank verbucht und der Prozess ist beendet. Trifft ein Antrag auf Härtefallregelung nach Bündnis für soziale Wohnungspolitik ein, wird der Subprozess „Härtefallregelung bearbeiten“ gestartet. Im Subprozess „Härtefallregelung bearbeiten“ wird zunächst geprüft, ob eine Härtefallregelung möglich ist. Sollte dies nicht möglich sein, wird eine Ablehnung erstellt, dem Mieter zugesandt und der Subprozess beendet. Ansonsten wird die maximal mögliche Erhöhung nach Bündnis für soziale Wohnungspolitik bestimmt. Sollte diese unter 5 Euro liegen wird dem Mieter eine Information über die Einstellung des Mieterhöhungsverfahrens zugesandt. Ansonsten wird ein entsprechender Bescheid erstellt und dem Mieter zugesandt und der Subprozess beendet. Nun hat der Mieter erneut 4 Wochen Zeit, um die Antwort zu senden. Handelt es sich bei der Antwort um eine Zustimmung, wird die Mieterhöhung in der Datenbank verbucht und der Prozess ist beendet. Sendet der Mieter eine Ablehnung, wird der Vorgang der Rechtsabteilung in Form eines neu gestarteten Prozesses übergeben und der Prozess beendet.

2. ERGEBNISSE DER SPEZIFIKATION DER WF-ANWENDUNG

2.1. FORMULARDESIGN

Formular& Feldtyp	Variablennamen	Prozess-ID	Datentyp	Feldeinschränkung	Feldinhalt
1 Input Feld	Mietvertragsnummer	Mietvertragsnummer	String	keine	keine
2 Auswahl-feld	Art der Antwort des Mieters	Antworttyp	String	keine	"Zustimmung", "Ablehnung", "Härtefallregelung"
3 Datei Upload	Zustimmung einlesen	Zustimmung	File	Upload-Möglichkeit	hochgeladenes Dokument
3 Date Picker	Datum der Unterschrift	DatumDerUnterschriftZustimmung	String Date	Datum	Datum

4 Input Feld	Anzahl der Bewohner	anzahlBewohner	Integer	keine	Anzahl der Bewohner
4 Input Feld	davon Kinder	anzahlKinderfreibetraege	Integer	keine	Anzahl der Kinder
4 Input Felder	Anzahl der Einkommensbezieher	anzahlEinkommensbezieher	Integer	keine	Anzahl der Einkommensbezieher
4 Input Felder	Summe d. Bruttoeink. aller Bew. pro Jahr	bruttoHaushaltseinkommen	Integer	keine	Summe Brutto Haushaltseinkommen
4 Checkbox	Arbeitnehmerpauschale	arbeitnehmerpauschaleNachgewiesen	Boolean	keine	keine
4 Checkbox	Einkommenssteuer	einkommenssteuerNachgewiesen	Boolean	keine	keine
5 Auswahl-feld	Art der Antwort des Mieters	Antworttyp	String	keine	"Zustimmung", "Ablehnung"

Tabelle 1

2.2.FACHLICHER PROZESS MIT CAMUNDA

Präsentieren Sie hier ihr Prozessdiagramm zum fachlichen Prozess (mit allen Prozessvarianten) als BPD aus dem Camunda Modeler ggf. mit Kommentaren.

Das BPD ist in guter Auflösung und im Querformat auf google Drive zur Ansicht zu finden.
https://drive.google.com/file/d/19thN4qNml4M8mlklVXd_m3G80JwqCkPU/view?usp=sharing





2.3. TESTFÄLLE

Definieren Sie mindestens 3 Testfälle zu allen Prozessvarianten Ihres Workflows und beschreiben Sie diese nach folgendem Schema:

Datum des Tests: 13.01.2018
Tester: Handan Fidan
Bezeichnung: Abbruchtest wegen inaktivem Vertrag
Beschreibung (Testszenario): <ol style="list-style-type: none"> 1. Die Datenbank wird neu aufgesetzt. 2. Es wird ein Mietvertrag ausgewählt, dessen Status 'inactive' ist. 3. Die Mietvertragsnummer dieses Mietvertrages wird in das Start-Eingabeformular eingegeben. 4. Prüfen welches Ergebnis dem Benutzer als nächstes angezeigt wird.
Testdaten: Mietvertragsnummer: PV-20121212-005545-051
Vorbedingung: Die Datenbank enthält alle benötigten Daten im richtigen Stand.
Erwartetes Ergebnis: Eingabeformular mit dem Titel "Benutzer informieren" wird erwartet. Als Grund muss "Mietvertragsnummer nicht mehr aktiv" angezeigt werden
Ist-Ergebnis: Erwartetes Ergebnis ist eingetreten

Tabelle 2

Datum des Tests: 13.01.2018
Tester: Christian Fiebelkorn
Bezeichnung: Abbruchmeldung wegen nicht möglicher Erhöhung, da die letzte Erhöhung noch keine 12 Monate zurück liegt
Beschreibung (Testszenario): <ol style="list-style-type: none"> 1. Die Datenbank wird neu aufgesetzt. 2. Es wird ein Mietvertrag ausgewählt, dessen Mieterhöhung noch keine 12 Monate zurückliegt. 3. Die Mietvertragsnummer dieses Mietvertrages wird in das Start-Eingabeformular eingegeben. 4. Prüfen welches Ergebnis dem Benutzer als nächstes angezeigt wird.

Testdaten: Mietvertragsnummer: PV-20171221-002156-001
Vorbedingung: Die Datenbank enthält alle benötigten Daten im richtigen Stand.
Erwartetes Ergebnis: Eingabeformular mit dem Titel "Benutzer informieren" wird erwartet. Als Grund muss "Miete kann zur Zeit nicht erhöht werden" angezeigt werden.
Ist-Ergebnis: Erwartetes Ergebnis ist eingetreten

Tabelle 3

Datum des Tests: 15.01.2018
Tester: Steffen Fröhlich
Bezeichnung: Abbruch wegen Erhöhung unter 5€
Beschreibung (Testszenario): <ol style="list-style-type: none"> 1. Die Datenbank wird neu aufgesetzt. 2. Es wird ein Mietvertrag ausgewählt, dessen Mieterhöhung für unter 5€ vorgesehen ist. 3. Die Mietvertragsnummer dieses Mietvertrages wird in das Start-Eingabeformular eingegeben. 4. Prüfen welches Ergebnis dem Benutzer als nächstes angezeigt wird.
Testdaten: Mietvertragsnummer: GV-20131221-001245-023
Vorbedingung: Die Datenbank enthält alle benötigten Daten im richtigen Stand.
Erwartetes Ergebnis: Eingabeformular mit dem Titel "Benutzer informieren" wird erwartet. Als Grund muss "Erhöhung wäre unter 5 Euro pro Monat" angezeigt werden
Ist-Ergebnis: Erwartetes Ergebnis ist eingetreten

Tabelle 4

Datum des Tests: 15.01.2018
Tester: Handan Fidan
Bezeichnung: Mieterhöhung eines Mietvertrages nach erfolgreich gestelltem Härtefallantrag
Beschreibung (Testszenario): <ol style="list-style-type: none"> 1. Die Datenbank wird neu aufgesetzt. 2. Es wird ein Mietvertrag ausgewählt, dessen Mieterhöhung unter einer Härtefallregelung erfolgt. 3. Die Mietvertragsnummer dieses Mietvertrages wird in das Start-Eingabeformular eingegeben.

4. Prüfen welches Ergebnis dem Benutzer als nächstes angezeigt wird. 5. Antrag auf Härtefallregelung einlesen. 6. Zustimmung einlesen.
Testdaten: Mietvertragsnummer: PV-20110115-001245-055
Vorbedingung: Die Datenbank enthält alle benötigten Daten im richtigen Stand.
Erwartetes Ergebnis: Die Miete wurde um 18,22 € erhöht und in die Datenbank eingetragen.
Ist-Ergebnis: Erwartetes Ergebnis ist eingetreten

Tabelle 5

3. DOKUMENTATION DER ERGEBNISSE DER WF-IMPLEMENTIERUNG

3.1. ANWENDERDOKUMENTATION

Der Prozess startet indem als erstes der Benutzer in dem Feld Mietvertragsnummer die Mietvertragsnummer einträgt , die er prüfen möchte, zum Beispiel die Mietvertragsnummer PV-20110115-001245-055.

Start process

Miete erhöhen

Über dieses Programm können Sie den Vorgang der Mietpreiserhöhung bestimmen. Geben Sie bitte dafür die Nummer des Mietvertrages ein.

Mietvertragsnummer

Back

Close

Start

Abb.3 Formular 1 Quelle: eigene Darstellung

Das System prüft in der Datenbank ob eine Mieterhöhung möglich ist und leitet den Benutzer bei einer möglichen Mieterhöhung weiter. Das System prüft außerdem die Mieterhöhungs-Spanne und verschickt bei relevanter Mieterhöhung ($\geq 5\text{EUR}$) automatisch eine eMail an den Mieter mit den entsprechenden Informationen auf Mieterhöhung. Der Mieter hat 6 Wochen Zeit, seine Zustimmung unterschrieben einzureichen. Wenn er dies innerhalb der 6 Monate jedoch nicht tut, wird ihm eine Erinnerung zugeschickt, die der Mieter dann innerhalb von 4 Wochen unterschrieben zurückschicken muss. (Wenn der Mieter auch auf diese Aufforderung nicht reagiert, wird der Bearbeitung an die Rechtsabteilung weitergeleitet.) Nachdem der Benutzer die Unterlagen erhalten hat, gibt er die Antwortart des Mieters ein, der annehmen,

ablehnen oder einen Härtefall beantragen kann. Die Antwort des Mieters wird von dem Benutzer eingetragen.

Brief auswerten

Über dieses Programm können Sie die Antwort des Kunden auswerten. Bei Briefeingang wählen Sie bitte "Brief eingegangen" und tragen die Antwortart des Kunden ein.

Art der Antwort des Mieters:

Antrag auf Härtefallregelung Save Complete

Abb.4 Formular 2 Quelle: eigene Darstellung

Wenn der Mieter einen Antrag auf Härtefall eingereicht hat, wird der Antrag eingesehen. Dabei gibt der Benutzer die Anzahl der Bewohner und Kinder an, sowie die Anzahl der Einkommensbezieher und die Summe der Bruttoeinkommen aller Bewohner pro Jahr. Außerdem füllt er aus, ob Werbungskosten/Einkommenssteuer/Krankenversicherung/Rentenversicherung zu berücksichtigen sind. Zuletzt gibt er das Datum der Unterschrift an und liest das Dokument ein.

Mieterhöhung

Set follow-up date

Set due date

demo

Demo Demo

Form

History

Diagram

Description

Antrag auf Härtefallregelung einlesen

Über dieses Programm können Sie die Daten eines Antrages auf Härtefallregelung einlesen.
Bitte laden Sie ebenfalls die gescannten Antragsunterlagen hoch.

Anzahl der Bewohner

2

- davon Kinder

1

Anzahl der Einkommensbezieher

1

Summe der Bruttoeinkommen aller Bewohner pro Jahr

6750.0

Arbeitnehmerpauschale (Werbungskosten) zu berücksichtigen?

☐

Einkommenssteuer zu berücksichtigen?

☐

Krankenversicherung zu berücksichtigen?

☐

Rentenversicherung zu berücksichtigen?

☐

Datum der Unterschrift

08.01.2018

Save

Complete

Abb.5 Formular 4 Quelle: eigene Darstellung

Wenn der Härtefall nicht zutreffend ist, wird eine Ablehnungs-eMail an den Mieter verschickt, der dann innerhalb von 4 Wochen diesem zustimmen oder ablehnen muss. Bei einer Überschreitung der 4 Wochen oder einer Ablehnung wird der Prozess an die Rechtsabteilung weitergeleitet. Wenn der Härtefall zutrifft, wird geprüft ob es eine bedingte Mieterhöhung gibt oder keine.

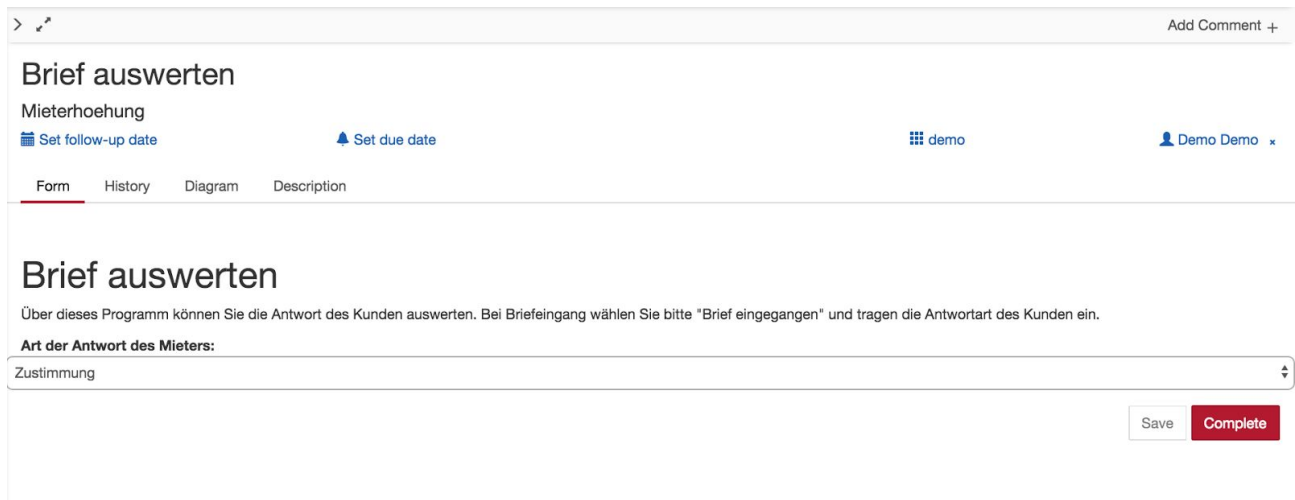


Abb.6 Formular 5 Quelle: eigene Darstellung

Bei einer Zustimmung wird die Zustimmung eingelesen und das Datum der Unterschrift notiert.

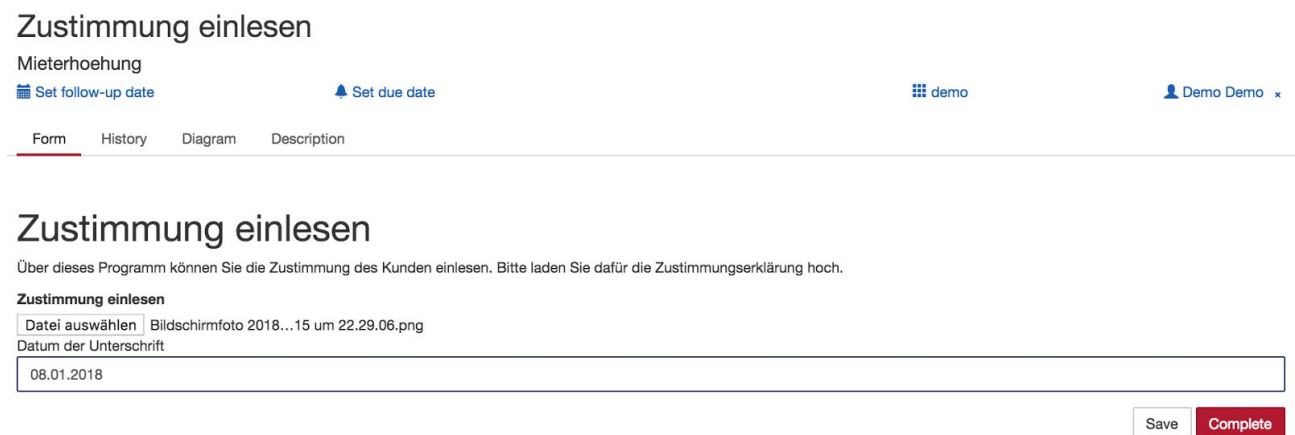


Abb.7 Formular 3 Quelle: eigene Darstellung

Danach wird die Miete erhöht und in die Datenbank eingetragen. Der Prozess wird beendet.

3.2. ENTWICKLERDOKUMENTATION

Startereignis: "Mietvertrag für Erhöhung auswählen", Formular formular1.html

Servicetask "Prüfen ob Erhöhung möglich", Klasse MAS_Projekt.mieteErhoehen.PruefenMitDB:

In der execute()-Funktion wird nach dem Laden der Mietvertragsnummer zunächst mit Hilfe der Methode BaueDBVerbindung.getConnection() eine Verbindung zur Datenbank (DB) aufgebaut. Anschließend wird über die Funktion pruefeMietvertragVorhanden() per SQL-Abfrage in der DB geprüft, ob ein entsprechender Mietvertrag vorhanden ist. Sollte dies nicht der Fall sein, wird eine entsprechende Abbruchmeldung gesetzt und die Kontrollvariable auf false gesetzt. Andernfalls werden über die Funktion leseDBZuMietvertrag() die Mietvertragsdaten und über die Funktion leseWohnungsdaten() die Wohnungsdaten aus der DB ausgelesen. Sollte der zugehörige Mietvertrag als inaktiv gekennzeichnet sein, wird die Kontrollvariable auf false gesetzt und eine entsprechende Abbruchmeldung gesetzt. Andernfalls werden die ermittelten Daten in die Datenstruktur des Prozeß-Objektes "execution" geschrieben und die Kontrollvariable auf true gesetzt. Anschließend wird über die Funktion vergleicheDatumDerLetztenErhoehung() per DB-Abfrage geprüft, ob die letzte Mieterhöhung zum gewählten Mietvertrag entsprechend den gesetzlichen Rahmenbedingungen mindestens 12 Monate zurück liegt (§558 Abs. 1 BGB) und bei Nichtzutreffen eine entsprechende Abbruchmeldung gesetzt und die Kontrollvariable auf false gesetzt.

Hauptcode der Klasse:

```

/**
 * Einstiegs-Methode zum Aufruf der Klasse, zur Steuerung der Klasse und zum
 * Speichern der geänderten Daten
 *
 * @param execution
 *   (DelegateExecution): Datenstruktur mit Daten des Vorgangs
 */
@Override
public void execute(DelegateExecution execution) throws Exception {
    String mietvertragsNr = (String) execution.getVariable("Mietvertragsnummer");

    LOGGER.info("\n\n ... LoggerDelegate invoked by PRUEFEN_MIT_DB" + "processDefinitionId="
        + execution.getProcessDefinitionId() + ", activityId=" + execution.getCurrentActivityId()
        + ", activityName=" + execution.getCurrentActivityName() + "" + ", processInstanceId="
        + execution.getProcessInstanceId() + ", businessKey=" + execution.getProcessBusinessKey()
        + ", executionId=" + execution.getId() + "\n\n");

    LOGGER.info("-> connection aufbauen...\n");
    connection = BaueDBVerbindung.getConnection();
    LOGGER.info("-> connection aufgebaut.\n");
    if (pruefeMietvertragVorhanden(mietvertragsNr)) {
        leseDBZuMietvertrag(mietvertragsNr);
        leseWohnungsdaten();
        if (statusmietvertrag.equals("inactive")) {
            execution.setVariable("Abbruchmeldung", "Mietvertragsnummer nicht mehr aktiv");
            execution.setVariable("erhoehungMoeglich", false);
        } else {
            execution.setVariable("Anrede", anrede);
            execution.setVariable("statusmietvertrag", statusmietvertrag);
            execution.setVariable("NachnameHauptmieter", nachnamehauptmieter);
            execution.setVariable("VornameHauptmieter", vornamehauptmieter);
            execution.setVariable("Abschluss", abschluss);
            execution.setVariable("E-Mail", email);
            execution.setVariable("StraßeHausnummer", straßeHausnummer);
            execution.setVariable("PLZOrt", plzOrt);
        }
    }
}

```

```

        execution.setVariable("Nettowohnflaeche", Math rint((nettowohnflaeche * 100.0) / 100.0);
        execution.setVariable("AnzahlZimmer", anzahlZimmer);
        execution.setVariable("aktuelle_Miete", Math rint((mietpreis * 100.0) / 100.0);
        execution.setVariable("erhoehungMoeglich", true);
        execution.setVariable("Mietvertrags_ID", mietvertrag_id);
        execution.setVariable("Wohnungs_ID", wohnungsId);
        execution.setVariable("Mietspiegel_ID", mietspiegelID);

        if (!vergleicheDatumDerLetztenErhoehung(mietvertrag_id)) {
            execution.setVariable("Abbruchmeldung", "Miete kann zur Zeit nicht erhoeht
werden");

            execution.setVariable("erhoehungMoeglich", false);

        } else {
            execution.setVariable("DatumLetzteErhoehung", datumLetzteErhoehung);
            execution.setVariable("aktuelles_Datum", heute);
        }
    } else {
        execution.setVariable("Abbruchmeldung", "Mietvertragsnummer nicht vorhanden");
        execution.setVariable("erhoehungMoeglich", false);
    }

    LOGGER.info("-> Ende: pruefenMitDB.execute\n");
}

```

Klasse MAS_Projekt.mieteErhoehen.BaueDBVerbindung:

Die Funktion getConnection() gibt eine DB-Verbindung zurück, die mit der Funktion baueDBVerbindung() aufgebaut wurde.

Hauptcode der Klasse:

```

private static Connection baueDBVerbindung() {
    Connection connection = null;
    final String driverClass = "oracle.jdbc.driver.OracleDriver";
    final String url = "jdbc:oracle:thin:@oradbs02.f4.htw-berlin.de:1521:orcl";
    final String user = "u558587";
    final String password = "558587p";

    try {
        Class.forName(driverClass);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }

    try {
        connection = DriverManager.getConnection(url, user, password);
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return connection;
}

```

}

Gateway "Erhöhung möglich":

Hier wird anhand der Kontrollvariable aus MAS_Projekt.mieteErhoehen.PruefenMitDB entschieden, ob der Prozeß wie vorgesehen durchlaufen oder über den Usertask "Benutzer informieren" mit dem Formular Infomeldung.html abgebrochen wird.

Servicetask "rechtlich max. mögliche Erhöhung bestimmen",

Klasse MAS_Projekt.mieteErhoehen.maximaleErhoehung:

In der execute()-Funktion werden - nach dem Laden der benötigten Daten aus und vor dem Festschreiben der geänderten Daten in der Prozeß-Datenstruktur (s.o.) - zunächst über die Funktion trageMietdatenInHM() benötigte Mietdaten aus der DB in eine Datenstruktur der Klasse eingelesen, dann über die Funktion anfangsmieteBestimmen() die Miete zu Beginn des Begrenzungszeitraumes nach §558 BGB bestimmt und anschließend über die Funktion maximaleErhoehungBestimmen() die rechtlich maximal mögliche Mieterhöhung berechnet (20% binnen 3 Jahren).

Hauptcode der Klasse:

```
/**
 * Einstiegs-Methode zum Aufruf der Klasse, zum Laden der benötigten Daten und
 * zum Speichern der ermittelten Mietdaten
 *
 * @param execution
 * (DelegateExecution): Datenstruktur mit Daten des Vorgangs
 */
@Override
public void execute(DelegateExecution execution) throws Exception {

    TreeMap<Date, Double> mietdaten = new TreeMap<Date, Double>();
    int mietvertrags_id;
    double aktuelleMiete; //
    double anfangsmiete; // Miete vor 3 Jahren, bzw am Abschlusdatum

    mietvertrags_id = (int) execution.getVariable("Mietvertrags_ID");
    aktuelleMiete = (double) execution.getVariable("aktuelle_Miete");

    anfangsmiete = anfangsMieteBestimmen(trageMietdatenInHM(mietvertrags_id));
    maximaleErhoehungBestimmen(aktuelleMiete, anfangsmiete);

    execution.setVariable("maximaleErhoehung", Math rint(maximaleErhoehung * 100.0) / 100.0);
    execution.setVariable("moeglicheNeueMiete", Math rint(maximaleNeueMiete * 100.0) / 100.0);
    execution.setVariable("DatumDerAnfangsmiete", datum);
    execution.setVariable("miete3JahreAlt", anfangsmiete);

}
```

Servicetask "Mietspiegelabgleich", Klasse MAS_Projekt.mieteErhoehen.ErhoehungMietspiegel:

Nach dem Laden der benötigten Daten aus und vor dem Festschreiben der geänderten Daten in der Prozeß-Datenstruktur (s.o.) - werden über die Funktion getMietspiegelDaten() die benötigten Mietdaten aus der DB ausgelesen.

Hauptcode der Klasse:

```
/**
 * Einstiegs-Methode zum Aufruf der Klasse
 *
 * @param execution (DelegateExecution): Datenstruktur mit Daten des Vorgangs
 */
@Override
public void execute(DelegateExecution execution) throws Exception {

    getExecutionVariablen(execution);
    getMietspiegelDaten(wohnungID);
    setExecutionVariablen(execution);
} //E execute
```

Servicetask "relevante Erhöhungs-Spanne festlegen".

Klasse MAS_Projekt.mieteErhoehen.RelevanteErhoehung:

Nach dem Laden der benötigten Daten aus und vor dem Festschreiben der geänderten Daten in der Prozeß-Datenstruktur (s.o.) - wird über die Funktion evaluiereRelevanteMiete() evaluiert, welche der ermittelten Mietpreiserhöhungs-Szenarien die Mieterhöhung am stärksten begrenzt und damit als für den Vorgang relevantes Begrenzung einzustufen ist.

Hauptcode der Klasse:

```
/**
 * Einstiegs-Methode zum Aufruf der Klasse
 *
 * @param execution (DelegateExecution): Datenstruktur mit Daten des Vorgangs
 */
@Override
public void execute(DelegateExecution execution) throws Exception {

    LOGGER.info("\n\n Start RelevanteErhoehung \n\n");

    getExecutionVariablen(execution);
    LOGGER.info("-> Ende: RelevanteErhoehung.getExecutionVariablen\n");
    erhoehungsDatum();
    evaluiereRelevanteMiete();
    LOGGER.info("-> Ende: RelevanteErhoehung.evaluiereRelevanteMiete\n");
    setExecutionVariablen(execution);
    LOGGER.info("-> Ende: RelevanteErhoehung.execute\n");

}

}
```

Gateway "relevante Mieterhoehung >= 5€ pM?"

In diesem Gateway wird geprüft ob die Mieterhöhung größer als 5 Euro ist, denn ansonsten lohnt sich eine Erhöhung nicht.

Ist die Erhöhung unter 5€ bekommt der Benutzer über den Usertask "Benutzer informieren" mit dem Formular Infomeldung.html abgebrochen wird.

Servicetask "Bescheid erstellen"

Klasse MAS_Projekt.mieteErhoehen.BescheidErstellen

In diesem Servicetask wird der Bescheid zur Mietpreiserhöhung erstellt, dabei wird eine txt-Datei erzeugt. Die erste Methode ruft wie bei den anderen Klassen die Camundavariablen ab, die letzte Methode schreibt die neuen Variablen in Camunda.

In der mittleren Methode wird der eigentliche Bescheid erstellt

Hauptcode aus Klasse:

```
/**
 * Einstiegs-Methode zum Aufruf der Klasse
 *
 * @param execution (DelegateExecution): Datenstruktur mit Daten des Vorgangs
 */
public void execute(DelegateExecution execution) throws Exception {
    LOGGER.info("-> BescheidErstellen\n");

    getExecutionVariablen(execution);
    LOGGER.info("-> BEa\n");
    geschaeftslogikBescheidErstellen();
    LOGGER.info("-> BEb\n");
    setExecutionVariablen(execution);
    LOGGER.info("-> BEc\n");
}
```

Servicetask "Zustimmung erstellen"

Klasse MAS_Projekt.mieteErhoehen.BescheidZustimmungErstellen

Dieser Servicetask ist genauso aufgebaut wie der Servicetask "Bescheid erstellen"

Servicetask "Härtefallantrag erstellen"

Klasse MAS_Projekt.mieteErhoehen.HaertefallbescheidErstellen

Dieser Servicetask ist genauso aufgebaut wie der Servicetask "Bescheid erstellen"

Email-Send-Task "Bescheid versenden per Email"

Klasse MAS_Projekt.mieteErhoehen.EmailBescheid

Die erste Methode holt sich aus Camunda alle benötigten Variableninhalte. Die andere Methode, die von `execute()` erstellt den Inhalt der E-Mail aus mehreren Strings und ruft im Anschluss selber die Methode zum Versenden der EMail auf.

Hauptcode der Klasse:

```
/**
 * Einstiegs-Methode zum Aufruf der Klasse
 *
 * @param execution (DelegateExecution): Datenstruktur mit Daten des Vorgangs
 */
public void execute(DelegateExecution execution) throws Exception {
    LOGGER.info("-> ");
    getExecutionVariablen(execution);
    performGeschaeftslogik();
}
```

Usertask "Brief auswerten" formular2.html

An diesem Task hängt ein Abbruchtimer, dieser wird nach 60480 Minuten oder 6 Wochen automatisch ausgelöst.

Bei diesem Task wird ein String weitergegeben der an einem Gateway mit dem Namen "Antworttyp" ausgewertet wird.

Wenn der Abbruchtimer ausgelöst wird, geht es mit dem Servicetask "Erinnerung versenden" weiter.

Servicetask "Erinnerung versenden"

MAS_Projekt.mieteErhoehen.ErinnerungVersenden

Diese Klasse ist im Aufbau identisch mit "EMail-Send-Task "Bescheid versenden per Email"".

Usertask "Brief auswerten" formular2.html

An diesem Task hängt ein Abbruchtimer, dieser wird nach 4 Wochen automatisch ausgelöst.

Bei diesem Task wird ein String weitergegeben der an einem Gateway mit dem Namen "Antworttyp" ausgewertet wird.

Sollte dieser Abbruchtimer ausgelöst werden, geht es mit dem folgenden E-Mailtask weiter.

EMail-Send-Task "An Rechtsabteilung weiterleiten"

Klasse MAS_Projekt.mieteErhoehen.RechtsabteilungInformieren

Erst werden die Variablen aus Camunda ausgelesen.

Die `execute` - Methode ruft am Ende eine neue Methode auf, die sich um das Versenden der E-Mail kümmert. Diese nutzt eine feste E-Mailadresse für die Rechtsabteilung.

Hauptcode der Klasse:

```
/**
 * Diese Methode steuert die Ausfuehrung der Klasse
```

** Dabei liest sie Daten aus dem Camunda Prozess aus und erzeugt den Inhalt der Email*

**/*

```
public void execute(DelegateExecution execution) throws Exception {

    String vorgangsid = (String) execution.getProcessInstanceId();
    String mietvertragsnummer = (String) execution.getVariable("Mietvertragsnummer");

    String subject = "Probleme bei der Mietpreiserhöhung zu dem Mietvertrag mit der Nummer " + mietvertragsnummer;
    String mailtext = "Sehr geehrte Rechtsabteilung,\n" + "\n"
+ "\nLeider gibt es ein Problem bei der Mietpreiserhöhung zu dem Mietvertrag mit der Nummer " +
mietvertragsnummer
+ ", denn der Mieter hat nicht Fristgerecht auf die Erhöhung reagiert oder weigert sich, die Erhöhung anzuerkennen"
+ "\nAlle Infos zu diesem Vorgang finden Sie unter der ID " + vorgangsid + "Im System der Mietpreiserhöhung"
+ "\n"
+ "\nWir danken für Ihre Unterstützung." + "\n\nMit freundlichen Grüßen,\n Das Mietenmanagement";

    sendEmail(mailtext, subject);
}
```

Gateway "Antworttyp"

In diesem Gateway wird der Antwort des Mieters eingegeben und der Pfad bei einer "Annahme", bei einer "Ablehnung" oder bei einem "Antrag auf Härtefallregelung" jeweils der entsprechende Pfad genommen.

User Task "Zustimmung einlesen"

Dieser Task gibt das formular3.html aus. Dabei wird eine Datei hochgeladen und ein Datum eingesetzt und weitergegeben.

Service Task "Miete erhöhen"

Klasse MAS_Projekt.mieteErhoehen.MieteErhoehen

Diese Klasse trägt die vorgenommene Mietveränderung, sowie alle dazugehörigen Daten in die Datenbank ein.

Subprozess "Härtefallantrag bearbeiten"

Startereignis "Antrag einlesen" formular4.html

Hier wird der Benutzer aufgefordert diverse Eingaben zu machen, diese werden im Subprozess benötigt.

DMN Task "Härtefallregelung möglich?"

Hierbei wird von einer DMN Tabelle geprüft, ob der Antragssteller überhaupt das Anrecht auf eine Härtefallregelung hat. Dazu wird als Ausgangsvariable "haertefallregelungMoeglich" mit einem boolean gefüllt, der am nächsten Gateway ausgewertet wird.

Servicetask "Bescheid erstellen"

Klasse MAS_Projekt.mieteErhoehen.BescheidErstellen

Dieser Servicetask ist genauso aufgebaut wie der Servicetask “Bescheid erstellen”.

Servicetask “Zustimmungserklärung erstellen”

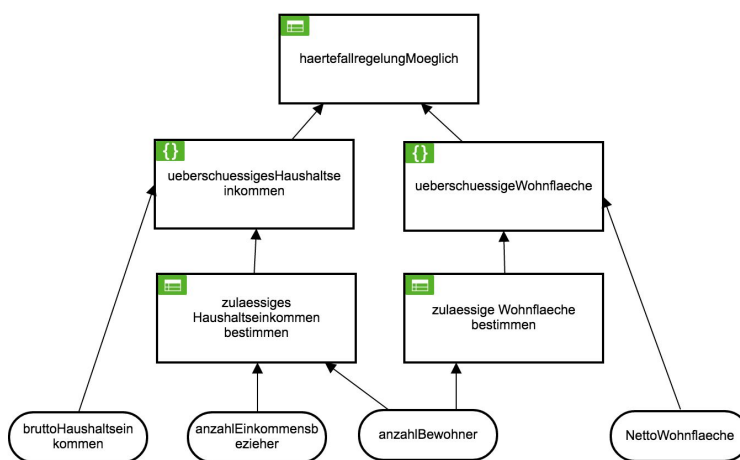
Klasse MAS_Projekt.mieteErhoehen.ZustimmungserklaerungErstellen

Dieser Servicetask ist genauso aufgebaut wie der Servicetask “ZustimmungserklaerungErstellen”

EMail-Send-Task “Bescheid versenden”

Klasse MAS_Projekt.mieteErhoehen.EmailBescheidVersenden

Dieser EMail-Send-Task ist genauso aufgebaut wie der EMail-Send-Task “Bescheid versenden”

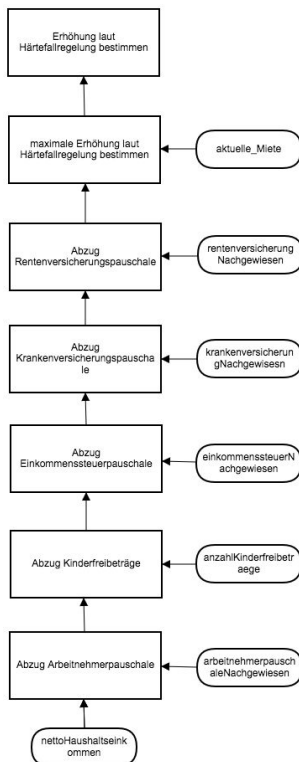


Gateway “Härtefallregelung möglich”

Nun wird die Ausgangsvariable vom letzten DMN ausgewertet.

DMN Task “maximal mögliche Erhöhung bestimmen”

Hier wird ermittelt, wie hoch denn die vertretbare Mieterhöhung wäre. Dies wird unter der Variablen “erhoehungHaertefallregelung” zurückgegeben und im Anschluss im Gateway ausgewertet.



User Task “Benutzer informieren”

Dieser Task gibt das Formular Infomeldung.html aus. Dabei wird eine die Mietvertragsnummer und der Grund der Ablehnung angegeben.

Email-Send-Task “Mieter informieren”

Klasse MAS_Projekt.mieteErhoehen.EmailMieterInformieren

Erst werden die Variablen aus Camunda ausgelesen.

Die execute - Methode ruft am Ende eine neue Methode auf, die sich um das Versenden der E-Mail kümmert. Diese nutzt die E-Mailadresse des Mieters.

Email-Send-Task “Ablehnung versenden”

Klasse MAS_Projekt.mieteErhoehen.EmailAblehnungVersenden

Erst werden die Variablen aus Camunda ausgelesen.

Die execute - Methode ruft am Ende eine neue Methode auf, die sich um das Versenden der E-Mail kümmert. Diese nutzt die E-Mailadresse des Mieters, die die Ablehnung des Härtefallantrags versendet.

Gateway “Erhöhung?”

In diesem Gateway wird geprüft ob eine Erhöhung des Mietpreises stattfindet. Wenn eine Erhöhung stattfindet, wird der Prozess an den User Task “Brief auswerten” weitergeleitet. Wenn keine Erhöhung des Mietpreises stattfindet, dann wird der Prozess beendet.

Usertask "Brief auswerten" formular5.html

An diesem Task hängt ein Abbruchtimer, dieser wird nach 4 Wochen automatisch ausgelöst.

Bei diesem Task wird ein String weitergegeben der an einem Gateway mit dem Namen "Antworttyp" ausgewertet wird. Bei Annahme wird der Prozess weitergeleitet an das Gateway "Annahme", bei Ablehnung oder Abbruch wird der Prozess an die Rechtsabteilung weitergeleitet.

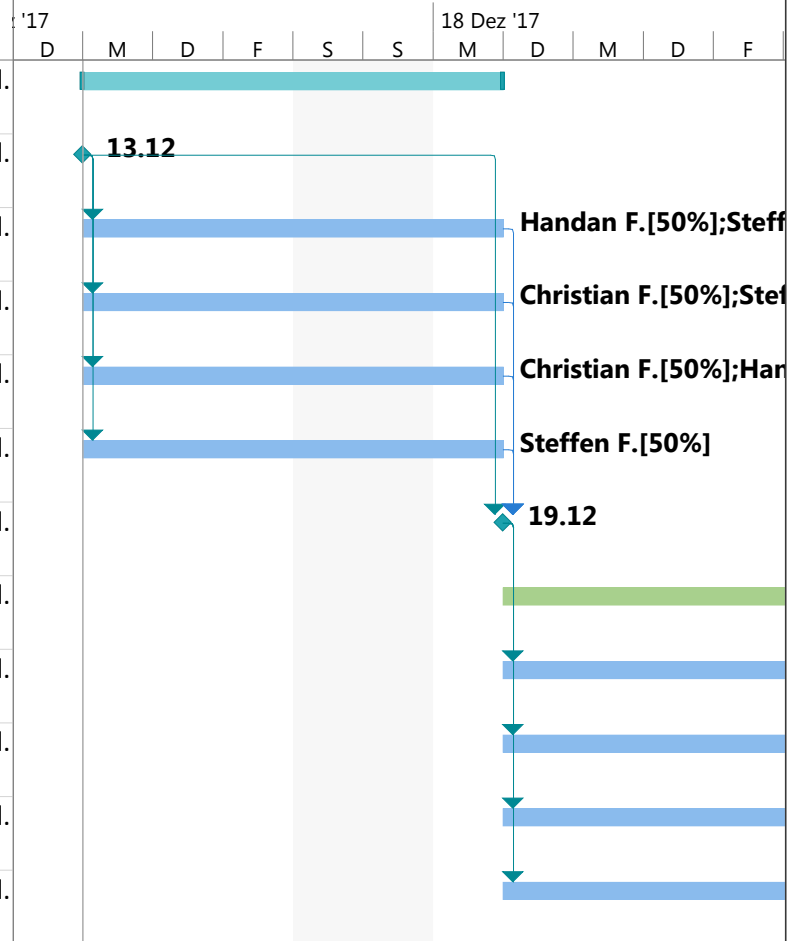
Gateway "Annahme"

In diesem Gateway wird über bei einer Annahme des Mieters zum Mietpreises der User Task "Zustimmung einlesen" erreicht. Bei einer Ablehnung wird der Prozess an die Rechtsabteilung weitergeleitet.

4. PROJEKTPLAN UND –ABRECHNUNG

Auf den nächsten Seiten sind ist die Abbildung aus MS Project zu sehen. Diese stellt den Projektplan da.

Nr.	Vorgang	Vorgangsname	Dauer	Anfang	Ende	Vorgänger	Ressource	Arbeit	
1		Projekt-Vorarbeit	4 Tage	Mit 13.12.17	Mon 18.12.17			0 Std.	
2		Projektstart	0 Std.	Mit 13.12.17	Mit 13.12.17			0 Std.	
3		Erstellung Prozeßbeschreibung	4 Tage	Mit 13.12.17	Mon 18.12.17	2	Handan F.[50%];Steffen F.[50%]	2,8 Std.	
4		Erstellung BPMN-Diagramm	4 Tage	Mit 13.12.17	Mon 18.12.17	2	Christian F.[50%];Steffen F.[50%]	3,85 Std.	
5		Erstellung Formulare Spezifikation	4 Tage	Mit 13.12.17	Mon 18.12.17	2	Christian F.[50%];Handan F.[50%]	2,9 Std.	
6		Erstellung Projektplanung	4 Tage	Mit 13.12.17	Mon 18.12.17	2	Steffen F.[50%]	2 Std.	
7		1. Abgabetermin	0 Tage	Die 19.12.17	Die 19.12.17	2;3;6;4;5	Christian F.	0 Std.	
8		Projekt-Durchführung	13 Tage	Die 19.12.17	Die 09.01.18			0 Std.	
9		Formulare erstellen	13 Tage	Die 19.12.17	Die 09.01.18	7	Handan F.[50%];Steffen F.[50%]	9,75 Std.	
10		DMNs erstellen	13 Tage	Die 19.12.17	Die 09.01.18	7	Handan F.[50%]	6,5 Std.	
11		Tasks programmieren	13 Tage	Die 19.12.17	Die 09.01.18	7	Christian F.[75%];Steffen F.[25%]	19,5 Std.	
12		DB erstellen	13 Tage	Die 19.12.17	Die 09.01.18	7	Christian F.	3,25 Std.	



Projekt: MAS-Projektplan
Datum: Mon 15.01.18

Vorgang		Inaktiver Sammelvorgang		Externe Vorgänge	
Unterbrechung		Manueller Vorgang		Externer Meilenstein	
Meilenstein		Nur Dauer		Stichtag	
Sammelvorgang		Manueller Sammelrollup		In Arbeit	
Projektsammelvorgang		Manueller Sammelvorgang		Manueller Fortschritt	
Inaktiver Vorgang		Nur Anfang			
Inaktiver Meilenstein		Nur Ende			

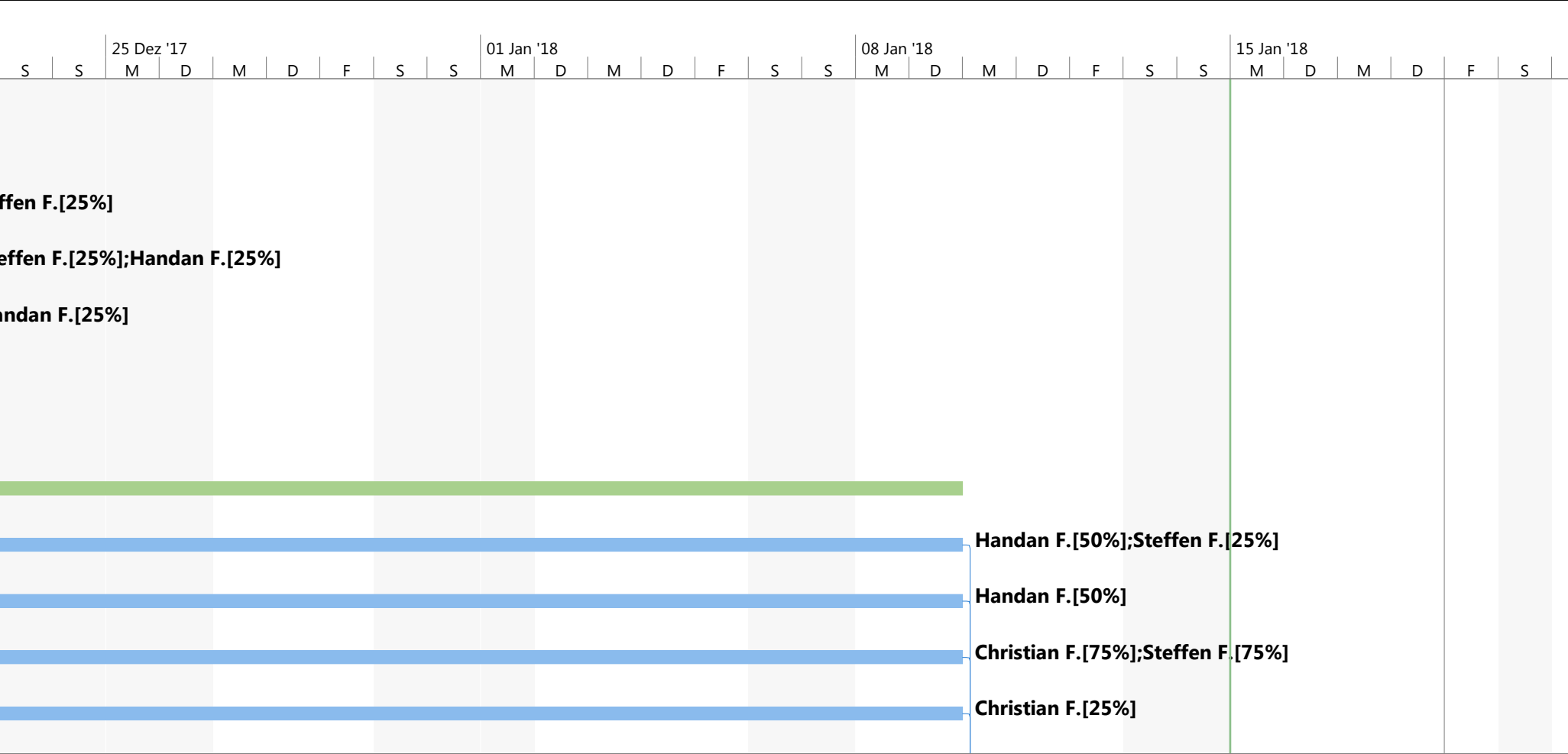
Nr.		Vorga	Vorgangsname	Dauer	Anfang	Ende	Vorgänger	Ressource	Arbeit												17					18 Dez '17				
										D	M	D	F	S	S	M	D	M	D	F										
13			Abgabe Workflow	0 Tage	Mit 10.01.18	Mit 10.01.18	9;11;12;10		0 Std.																					
14			Projekt-Dokum	6 Tage	Mit 10.01.18	Mit 17.01.18			0 Std.																					
15			Dokumentation erstellen	3,5 Tage	Mit 10.01.18	Mon 15.01.18	13	Christian F.;Handa	7 Std.																					
16			Abgabe Projektdokume	0 Tage	Mon 15.01.18	Mon 15.01.18	15	Handan F.	0 Std.																					
17			Projektabrechn	5 Tage	Mit 10.01.18	Die 16.01.18	13	Steffen F	5 Std.																					
18			Abgabe Projektabrechn	0 Tage	Mit 17.01.18	Mit 17.01.18	17	Steffen F.	0 Std.																					
19			Projektpräsent	0,5 Tage	Mit 17.01.18	Mit 17.01.18	16	Christiar	126,5 S...																					
20																														
21			geleistete Arbeitszeit, gesamt						125 Std.																					
22			Christian Fiebelkorn						55 Std.																					
23			Handan Fidan						30 Std.																					

Projekt: MAS-Projektplan
Datum: Mon 15.01.18

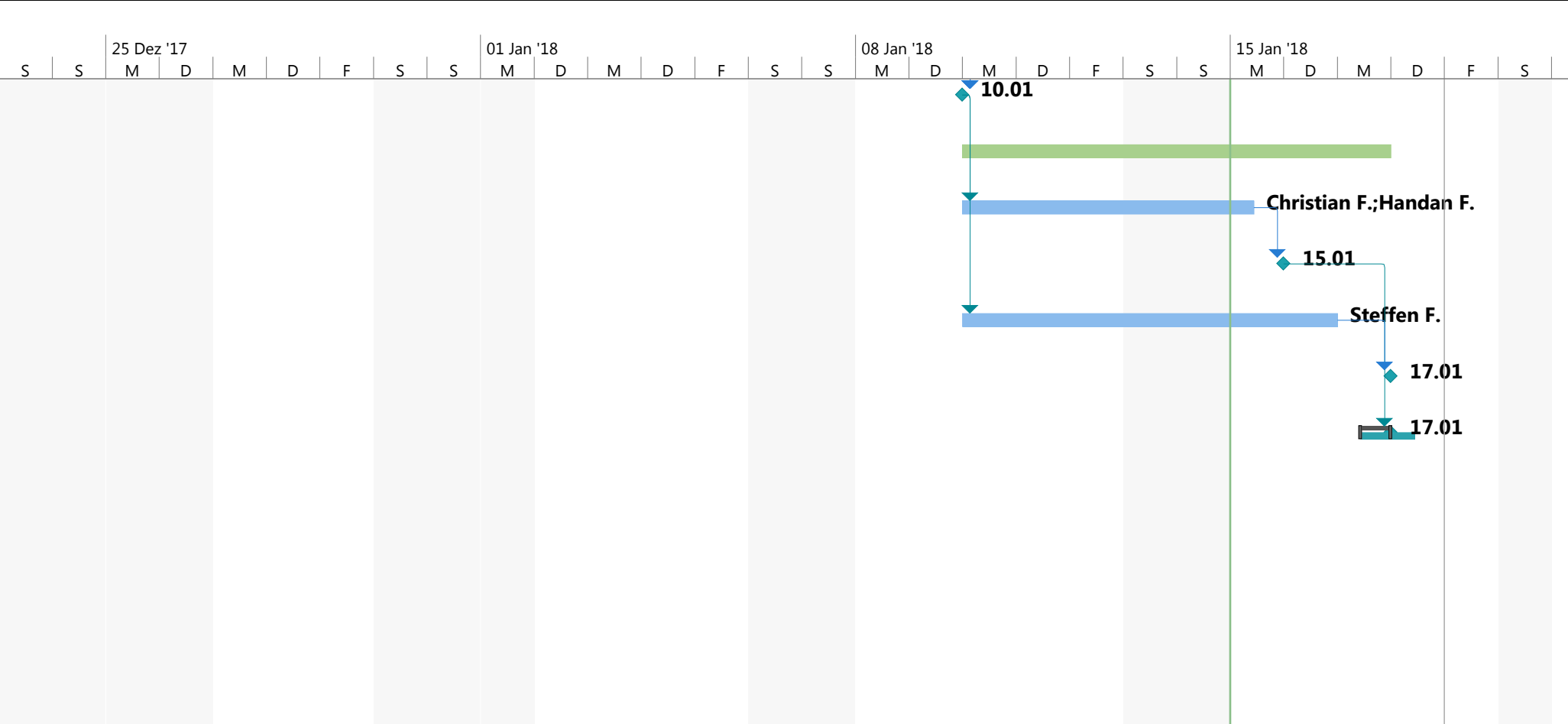
Vorgang		Inaktiver Sammelvorgang		Externe Vorgänge	
Unterbrechung		Manueller Vorgang		Externer Meilenstein	
Meilenstein		Nur Dauer		Stichtag	
Sammelvorgang		Manueller Sammelrollup		In Arbeit	
Projektsammelvorgang		Manueller Sammelvorgang		Manueller Fortschritt	
Inaktiver Vorgang		Nur Anfang			
Inaktiver Meilenstein		Nur Ende			

Nr.		Vorga	Vorgangsname	Dauer	Anfang	Ende	Vorgänger	Ressource	Arbeit														
										'17	D	M	D	F	S	S	18 Dez '17						
24			Steffen Fröhlich						40 Std.														

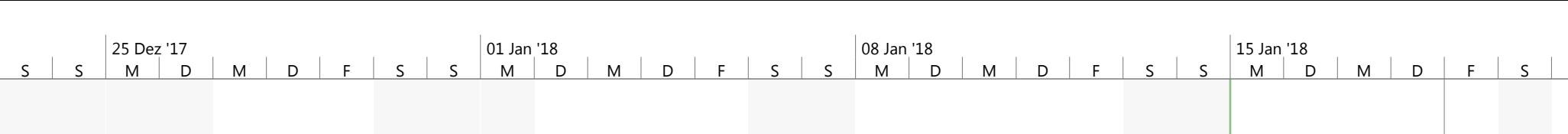
<div> <div>Projekt: MAS-Projektplan</div> <div>Datum: Mon 15.01.18</div> </div>	Vorgang	<div></div>	Inaktiver Sammelvorgang	<div></div>	Externe Vorgänge	<div></div>
	Unterbrechung	<div></div>	Manueller Vorgang	<div></div>	Externer Meilenstein	<div></div>
	Meilenstein	<div></div>	Nur Dauer	<div></div>	Stichtag	<div></div>
	Sammelvorgang	<div></div>	Manueller Sammelrollup	<div></div>	In Arbeit	<div></div>
	Projektsammelvorgang	<div></div>	Manueller Sammelvorgang	<div></div>	Manueller Fortschritt	<div></div>
	Inaktiver Vorgang	<div></div>	Nur Anfang	<div></div>		
	Inaktiver Meilenstein	<div></div>	Nur Ende	<div></div>		



Projekt: MAS-Projektplan Datum: Mon 15.01.18	Vorgang		Inaktiver Sammelvorgang		Externe Vorgänge	
	Unterbrechung		Manueller Vorgang		Externer Meilenstein	
	Meilenstein		Nur Dauer		Stichtag	
	Sammelvorgang		Manueller Sammelrollup		In Arbeit	
	Projektsammelvorgang		Manueller Sammelvorgang		Manueller Fortschritt	
	Inaktiver Vorgang		Nur Anfang			
	Inaktiver Meilenstein		Nur Ende			



Projekt: MAS-Projektplan Datum: Mon 15.01.18	Vorgang		Inaktiver Sammelvorgang		Externe Vorgänge	
	Unterbrechung		Manueller Vorgang		Externer Meilenstein	
	Meilenstein		Nur Dauer		Stichtag	
	Sammelvorgang		Manueller Sammelrollup		In Arbeit	
	Projektsammelvorgang		Manueller Sammelvorgang		Manueller Fortschritt	
	Inaktiver Vorgang		Nur Anfang			
	Inaktiver Meilenstein		Nur Ende			



Projekt: MAS-Projektplan Datum: Mon 15.01.18	Vorgang		Inaktiver Sammelvorgang		Externe Vorgänge	
	Unterbrechung		Manueller Vorgang		Externer Meilenstein	
	Meilenstein		Nur Dauer		Stichtag	
	Sammelvorgang		Manueller Sammelrollup		In Arbeit	
	Projektsammelvorgang		Manueller Sammelvorgang		Manueller Fortschritt	
	Inaktiver Vorgang		Nur Anfang			
	Inaktiver Meilenstein		Nur Ende			

5. ABBILDUNGS- UND TABELLENVERZEICHNIS

5.1. ABBILDUNGSVERZEICHNIS

Abb.1 BPMN Quelle: eigene Darstellung.....	S. 5
Abb.2 BPMN Quelle: eigene Darstellung.....	S. 6
Abb.3 Formular 1 Quelle: eigene Darstellung.....	S. 9
Abb.4 Formular 2 Quelle: eigene Darstellung.....	S. 10
Abb.5 Formular 4 Quelle: eigene Darstellung.....	S. 10
Abb.6 Formular 3 Quelle: eigene Darstellung.....	S. 11
Abb.7 Formular 5 Quelle: eigene Darstellung.....	S. 11
Abb.8 Projektplan Quelle: eigene Darstellung.....	S. 22

5.2. TABELLENVERZEICHNIS

Tabelle 1.....	S. 3
Tabelle 2.....	S. 7
Tabelle 3.....	S. 7
Tabelle 4.....	S. 8
Tabelle 5.....	S. 8