Digitales Klassenbuch Umwandler

von Christian Richter

—

Projektzeitraum: 14.06.2021 - 02.07.2021

Ausbildungsbetrieb
ThunderFrog MEDIA GmbH
Kiekenberg 10
45359 Essen

1. Einleitung

In dieser Projektdokumentation wird der Ablauf des Abschlussprojektes, welches im Rahmen der Umschulung zum Fachinformatiker für Anwendungsentwicklung durchgeführt wurde, erläutert. Das Projekt wurde ursprünglich in der youCcom GmbH durchgeführt, allerdings durch interne Umstrukturierungen an die ThunderFrog MEDIA GmbH abgetreten.

Der verantwortliche Entwickler wurde dementsprechend ebenfalls an die ThunderFrog MEDIA GmbH überstellt. Die ThunderFrog MEDIA GmbH ist ein seit 2008 bestehendes Unternehmen, welches zumeist in der Web- und Desktop App entwicklung tätig ist.

Ziel dieser Dokumentation ist es, die durchzuführenden Schritte des Projektes von der Planung bis zur Übergabe an den Vorgesetzten zu erläutern und dies mit UML-Diagrammen, Screenshots und Dokumenten zu unterstützen.

1.1 Projektbeschreibung

Aktuell erfolgt die Dokumentation der Unterrichtsinhalte auf zwei Wege.Zum einen digital für den Homeschooling Bereich und zum anderen analog bei Präsenzveranstaltungen.

Um diese beiden Wege zu kombinieren, da eine Papier Archivierung vorgeschrieben wird, muss das digitale Klassenbuch als genormte PDF ausdruckbar gemacht werden.

Der Produktverantwortliche für den Bereich IT-Ausbildung hat zusammen mit der IT-Abteilung den Auftrag gegeben ein entsprechendes System zu entwickeln.

1.2 Projektziel

Ziel des Projektes ist die Entwicklung eines Tools mit dem es möglich ist die nachträgliche Papierarchivierung zu vereinfachen, indem die digitalen Daten aus dem vorhandenen System exportiert und dann automatisch formatiert, als druckbare PDF ausgegeben werden kann. Die Daten werden als .csv mit Kommatrennung exportiert, mit dem Tool automatisch verarbeitet und dann als .pdf Format gespeichert.

1.3 Projektumfeld

ThunderFrog MEDIA GmbH ist ein mittelständiges Unternehmen mit dem Sitz in Essen. Mit 12 festen Mitarbeiter wird täglich daran gearbeitet Klein- und Mittelständige Unternehmen in Ihren täglichen Arbeitsalltag digital zu unterstützen.

Der Projektbetreuer ist der Ausbilder Herr Adler Schulten und der Projektverantwortliche ist Herr Christian Richter.

Die eingesetzte Programmiersprache ist Java in der aktuellen Version.

1.4 Projektbegründung

Durch die COVID-19 Pandemie wurden viele Unternehmen und auch Umschulungszentren dazu teilweise gezwungen ins HomeOffice bzw. HomeSchooling zu gehen.

Der Auftraggeber die BFZ Essen GmbH, ist eine führende Bildungseinrichtung für berufliche Erwachsenenbildung und bietet sowohl IHK-Berufsabschlüsse, zukunftsorientierte Fort- und Weiterbildungen sowie Firmenschulungen unter anderem in den Bereichen des Gesundheitswesens & IT.

Da die BFZ Essen GmbH ein Städtisches Unternehmen ist erfolgt die Archivierung von täglichen Schulungsinhalten streng nach Regeln und muss auf Papierbasis verwaltet werden.

Im aktuellen HomeSchooling System (Moodle) gibt es nur eine möglichkeit das komplette Klassenbuch eines Kurses in eine wenig Lesbare und schwer Druckbare CSV Datei zu exportieren.

Durch das entwickelte System soll es möglich werden die exportierte CSV Datei in ein vorgefertigtes Formular als PDF zu speichern, damit diese Datei einfacher ausgedruckt werden kann.

2. Projektplanung

2.1 Projektphasen

Für die Umsetzung des Projektes standen 70 Stunden zur Verfügung, welche vor dem Beginn des Projektes auf verschiedene Phasen aufgeteilt wurden.
Die Hauptphasen des Projektes und deren Zeiteinteilung lassen sich aus der untenstehenden Tabelle entnehmen.
Eine detaillierte Auflistung aller Phasen ist im Anhang A.1

Projektphase	Geplante Zeit
Analyse	6 h
Entwurf	8 h
Implementierung	33 h
Qualitätsprüfung	7 h
Rollout	6 h
Dokumentation	10 h
Gesamt	**70 h**

2.2 Ressourcenplanung

Diese beinhaltet Hardware und Software-Ressourcen sowie auch das benötigte Personal wurde in einem internen Auswahlverfahren dem Projekt zugeteilt.
Das Personal wurde nach deren Fähigkeiten und Erfahrungen mit ähnlichen Projekten ausgewählt, umso eine möglich schnelle und unproblematische Entwicklung zu gewährleisten.
Es wurde auch darauf geachtet, dass zur Umsetzung des Projektes nach Möglichkeit nur Software verwendet wird, die bereits im Betrieb vorhanden ist oder keine Kostenpflichtigen Lizenzen benötigt.
Somit können die Projektkosten möglichst niedrig gehalten werden.

Eine detaillierte Übersicht mit allen zur Umsetzung benötigten Ressourcen des Projektes ist im Anhang A.2.

2.3 Entwicklungsprozess

Bevor mit der Realisierung des Projektes begonnen werden kann, muss sich für einen geeigneten Entwicklungsprozess entschieden werden. Damit wird die Vorgehensweise festgelegt, nach der die Umsetzung erfolgen soll. Für das Projekt wird ein iterativer Entwicklungsprozess gewählt.
Das iterative Durchlaufen der Entwicklungsphasen ermöglicht dabei, schnell Teilergebnisse zu erzielen, die in kurzen Zyklen getestet und verfeinert werden können.

3. Analysephase

3.1 Ist-Analyse

Wie bereits in der Projektbegründung erwähnt, ist es aktuell nicht möglich das Ergebnis der exportierten Daten aus dem HomeSchooling System in ein leserliches Format für die Archivierung auszudrucken. Der momentane Ablauf wäre die Daten zu Exportieren und diese dann händisch in das entsprechende Formular zu übertragen damit es Ausgedruckt werden kann oder alternativ diese aus dem digitalen System abzuschreiben.
Dies soll mit dem entwickelten Tool vereinfacht werden.

3.2 Wirtschaftslichkeitsanalyse

3.2.1 “Make or Buy” - Entscheidung

Das zu entwickelnde Tool muss an das vorhandene System in seinen Anforderungen und Funktionalitäten sehr spezifisch sein. Dies ist zu individuell, als das es schon auf dem Markt erhältlich ist. Somit wird das Tool von der ThunderFrog MEDIA GmbH umgesetzt.

3.2.2 Projektkosten

Die Projektkosten, die während der Entwicklung des Programmes anfallen, werden im Folgenden kalkuliert. Dafür müssen sowohl die Personalkosten als auch die Aufwendungen für Ressourcen und andere Kosten in Form von Gemeinkosten berücksichtigt werden.

Für die Gemeinkosten wird von folgenden Stundensätze ausgegangen:

- ➔ Auszubildender 30€
- ➔ Entwickler 45€
- ➔ Entwicklungsleiter 55€
- ➔ Lohngemeinkosten 35%
- ➔ Verwaltungskosten 10%

Die Lohngemeinkosten enthalten neben den üblichen Gemeinkosten auch die Kosten, die durch die Nutzung der Hardware und Software entstehen.
Die Verwaltungsgemeinkosten decken die Kosten, die für die Verwaltung anfallen.
Eine detaillierte Auflistung der Personalkosten kann der nachfolgenden Tabelle entnommen werden:

Vorgang	Mitarbeiter	Zeit	Gesamt
Entwicklung	1x Auszubildender	70 h	2.100,00 €
Projektvorstellung	2x Entwickler 1x Entwicklungsleiter	1h	145,00 €
Abnahme	1x Entwicklungsleiter	1h	55,00 €
Gesamt			**2.300,00 €**

Die gesamten Personalkosten belaufen sich auf **2.300,00 €**

Lohngemeinkosten:	2.300,00 € * 0.35	= **805,00 €**
Verwaltungsgemeinkosten:	2.300,00 € * 0.1	= **230,00 €**

Daraus ergibt sich die folgende Kalkulation der gesamten Projektkosten:

Personalgesamtkosten	2.300,00 €
+ Lohngemeinkosten	805,00 €
+ Verwaltungsgemeinkosten	230,00 €
Gesamtkosten Projekt	**3.335,00 €**

Die kalkulierten Gesamtkostes des Projektes belaufen sich auf insgesamt **3.335,00 €**

3.3 Amortisation

Obwohl letztendlich das Projekt durchaus Zeit und somit Kosten bei der Entwicklung einsparen soll, ist es trotzdem schwierig, den sofortigen monetären Nutzen und somit die genaue Wirtschaftlichkeit des Projektes zu ermitteln.

Es handelt sich um ein internes Projekt, der Auszubildende Christian Richter den schulischen Teil der Ausbildung zum Fachinformatiker Anwendungsentwicklung bei der BFZ Essen GmbH absolviert. Daher werden die im Abschnitt 3.2.2 Projektkosten errechneten Projektkosten nicht durch einen Kundenauftrag im geschäftlichen Sinne gedeckt.

4. Entwurfsphase

4.1 Soll-Konzept

Das Abschlussprojekt wird, wie im Abschnitt 1.2 Projektziel erwähnt, als ein Tool zum Verarbeiten von CSV Daten aus dem vorhandenen System entwickelt.

Als Programmiersprache wird Java verwendet. Java gehört zu der am meist genutzten Programmiersprachen, da es fast auf jedem System funktioniert. Es wird keine Zusatzsoftware benötigt, um in Java entwickelte Programme zu starten. Nur eine Java-Laufzeitumgebung muss vorhanden sein, diese gibt es für alle gängigen Systeme.

Der Ablauf des Programmes wird mit Hilfe eines Aktivitätsdiagramms abgebildet und dies als Leitfaden für die benötigte Programm Funktionalitäten welches im Anhang zu finden ist.

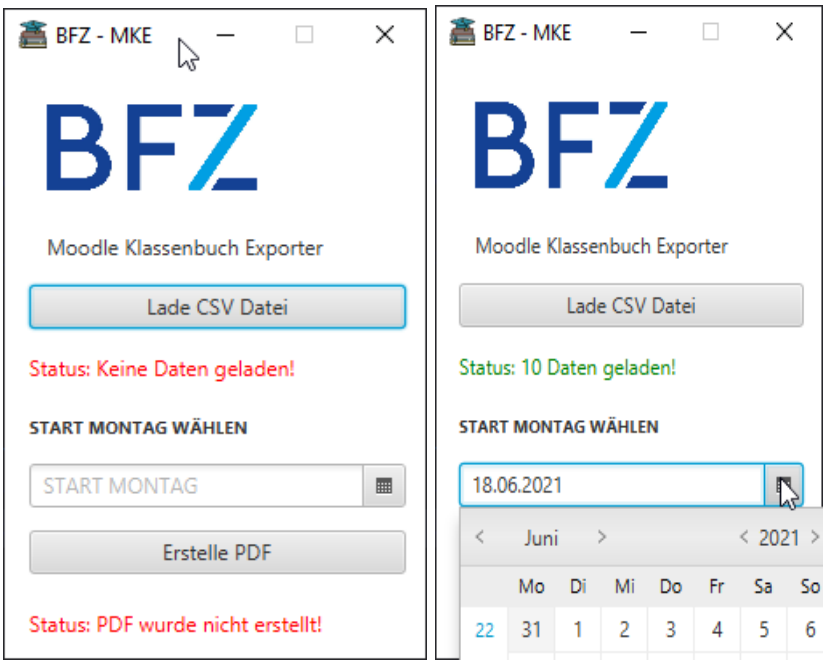
4.2 Projektstruktur

Das Projekt wird als Java Maven Projekt umgesetzt. Maven hat den Vorteil gegenüber anderen Projektstrukturen die Einbindung eines neuerem Front-End Framework, in diesem Projekt Fall JavaFX zur Gestaltung der Benutzeroberfläche. Weitere Bibliotheken werden einfach über die pom.xml der Projektstruktur hinzugefügt. Zum Parsen der CSV Datei wird Apache Commons verwendet, zum Erstellen des HTML Template Thymeleaf, das ersetzen der Platzhalter im HTML Template erfolgt mit Jsoup und zum Speichern der PDF Datei OpenHTMLtoPDF.

Das IHK-Projekt wird in dem Package src/main/java erstellt.

4.3 Benutzeroberfläche

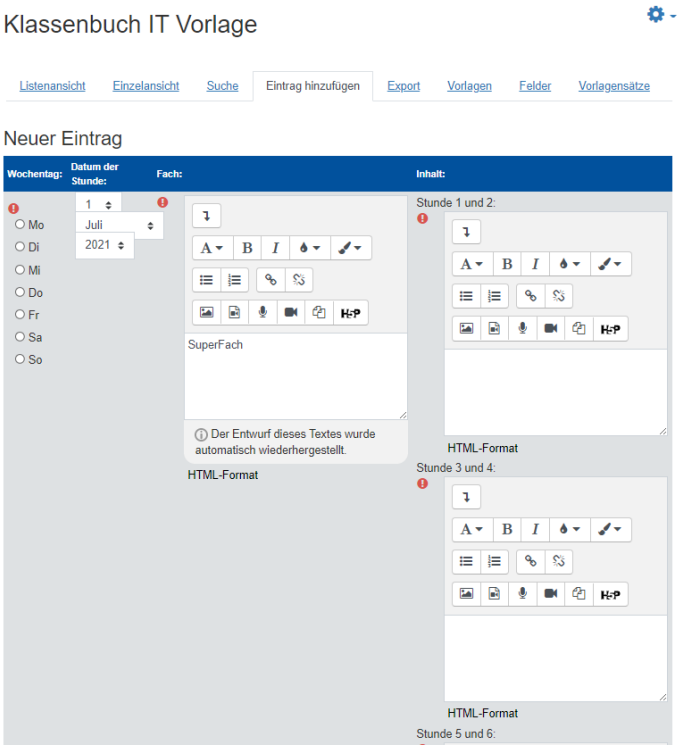
Für das Projekt wird eine Benutzeroberfläche mit JavaFX erstellt. Einfach und strukturiert gibt es auf der Benutzeroberfläche zwei Buttons, einmal zum Einlesen der CSV Daten sowie einmal zum Abspeichern als PDF-Datei. Da es wichtig ist das nur ein Montag bzw. nur 5 Tage als PDF gespeichert werden, wurde noch eine DatePicker erstellt damit ein bestimmtes Startdatum ausgewählt werden kann.



Dazu kommen noch zwei Labels die einen kleinen Status anzeigen.

4.4 Bearbeiten des vorhandenen System

Um die Daten erfolgreich als PDF zu speichern, muss das aktuelle HomeSchooling System (Moodle) angepasst werden. Die Eingabemaske sowie die Darstellung wurde angepasst, dadurch entstand eine Übersichtlichkeit und Daten können besser eingetragen werden.



Das alte System war darauf ausgelegt, dass das Fach und der Unterrichtsinhalt in großen Textfeldern mit HTML Support geschrieben werden kann.

Es fehlte auch noch zusätzlich die Information des Dozenten.

Problematisch wird hier der Export der Daten da diese mit HTML-Tags exportiert werden. Leider nicht immer und eher zufällig.

Mit Absprache des Auftraggeber wurde die Eingabemaske überarbeitet.

4.5 Pflichtenheft

Die Entwurfsphase wird mit dem Erstellen des Pflichtenheftes abgeschlossen. Es beschreibt, wie die herausgearbeiteten Anforderungen an das Programm umgesetzt werden. Es dient als Leitfaden für die Durchführung des Projektes.

5. Implementierungsphase

5.1 Bearbeitung des vorhandenen System

Damit wir beim Export von Daten keinen HTML Code bekommen der extra ausgefiltert werden muss, wurde mit Absprache des Auftraggeber die Eingabemaske überarbeitet.
Ein Vorher-Nachher vergleich ist als Screenshot im Anhang hinterlegt.
Es wurde die Textboxen durch Textfelder ersetzt, sowie ein Feld für Dozent hinzugefügt.

5.2 Programmablauf

Bevor mit der eigentlichen Implementierung begonnen werden kann, wird zunächst ein Programmablaufplan erstellt. In diesem werden die einzelnen Funktionsschritte und deren Reihenfolge Schematisch dargestellt.

5.3 Klasse MoodleData

Zum internen abspeichern der importierten Daten wird eine Klasse erstellt, die alle Datenfelder der CSV Datei hat. Mittels Getter und Setter kann dann auf diese Daten zugegriffen werden.

5.4 Controller Klasse

In der Controller Klasse werden alle Benutzerinteraktionen in Funktionen abgefangen. Der Button zum Einlesen der CSV Daten führt die Funktion openFile() aus.

Es wird ein FileChooser geöffnet in dem nur CSV Dateien ausgewählt werden können. Nachdem die Datei ausgewählt wurde wird gecheckt ob es sich wirklich um eine Datei handelt, wenn ja wird die Liste sortiert und geht dann zum Parsen der Daten, der Status im Label wird angepasst.

Die Funktion savePDF() wird beim drücken des Button “Speichere PDF” ausgeführt. Es wird die *Liste* mit *MoodleData* und dem gewähltem Datum aus dem DatePicker an die Funktion generate() übergeben. Bei erfolg oder fehler wird das Statuslabel angepasst.

5.5 Komponente zum Parsen der Export Daten

Um die exportierten Daten weiter bearbeiten zu können, müssen diese Sortiert und leserlich eingelesen werden. Dies geschieht mit Apache Commons.
Dazu wird eine statische Funktion getListfromCSV() in der Klasse **CSVImporter** mit dem Übergabeparameter *File* und als Rückgabewert eine *Liste* vom Typ *MoodleData*. Die eingelesenen Daten werden mittels for-each in ein Objekt (MoodleData) geschrieben. Das passiert mit allen Inhalte bis die export Datei fertig gelesen wurde. Sollte es zu einem Fehler kommen, wird dieser mit try-catch abgefangen und dem Anwender über die Statuslabel auf der GUI angezeigt.

5.6 Erstellen des HTML Template mit Thymeleaf und Jsoup

Um die Daten hinterher als PDF zu speichern, wird in der Klasse **PDFCreator** die Funktion generate() erstellt. Als Übergabeparameter wird die *List<MoodleData>*, sowie als String *date*. Mit dem Ausgewählten *date* wird aus den importierten Daten der Start Datensatz gewählt. Mittels for-schleife werden alle Daten durchgeschaut um den ersten Datensatz mit dem

passendem *date* zu finden, wurde der Datensatz gefunden werden die Platzhalter im HTML Template mit den gefundenen Daten ersetzt. Mittels Thymeleaf wird das HTML Template zusammengebaut und in einer Temp-Datei gespeichert.

5.7 Aus HTML Template wird PDF

Nachdem die Temp-Datei erstellt wurde wird diese zusammen mit einem Output Name an die Funktion htmlToPdf() übermittelt.
Der Output Name ist wichtig, damit wir einen Dateinamen nach der Erstellung des PDF`s haben.

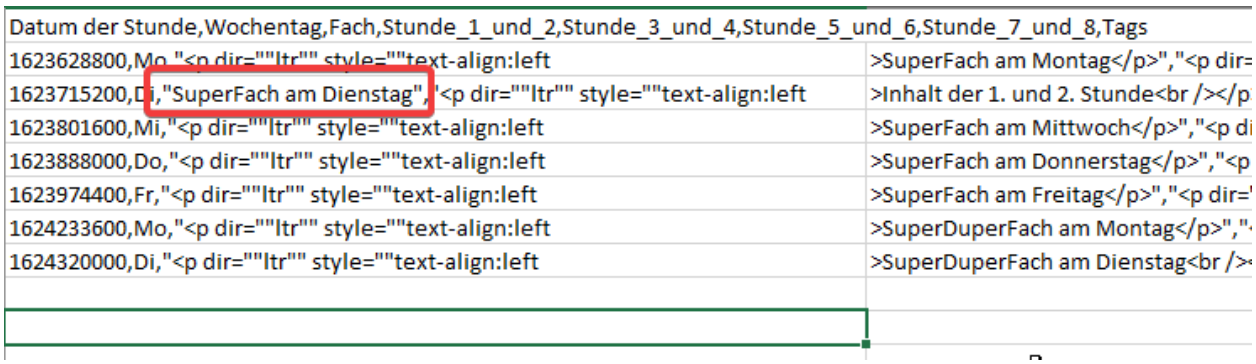
In der Funktion htmlToPdf() wird das vorher erstellte HTML Template in XHTML umgewandelt und mittels der builder Funktion von OpenHTMLToPDF in ein PDF umgewandelt. Nach erfolgreichem erstellen der PDF wird die Temp-Datei gelöscht und die PDF wird mit dem Output Name im Stammverzeichnis des Programms abgespeichert.

6. Qualitätsprüfung

6.1 Import Test mit Testdaten

Bevor mit realen Daten gearbeitet werden kann wurden Testdaten erstellt. Dies erfolgte noch mit der alten Eingabemaske. Dort kam es immer wieder zu Fehlern in der CSV Datei. Mal waren die Datensätze verrutscht, ein anderes mal gab es HTML-Tag und wieder nicht. Nachdem die Eingabemaske überarbeitet wurde, traten diese Fehler nicht mehr auf und so konnten brauchbare Daten zum Testen des Tools erstellt werden.
Eine Ausgabe der Daten erfolgten noch nicht als PDF sondern erstmal als Debug mittels Java Funktion System.out.println();

Nach erfolgreichem Test mit dem Import der Daten wurde das Programm erweitert zum verarbeiten der Daten.



Als Beispiel: SuperFach am Dienstag ohne HTML Tag und in der Spalte verrutscht.

6.2 Mit den Testdaten eine PDF erstellen

Nachdem die Daten erfolgreich eingelesen werden können und als Objekt im Programm vorliegen, wurde nun getestet wie die Daten auf dem PDF aussehen

Dazu wurde sich strikt an die vorgabe vom Auftraggeber gehalten. Nach mehreren Rücksprachen konnte das PDF so angepasst werden dass es für den Auftraggeber akzeptabel ist.

7. Rollout

Nachdem alle Funktionalitäten implementiert wurden und durch das Testverfahren deren richtige Funktionalität gewährleistet ist, wird das Projekt in einer Besprechung mit der IT-Abteilung der ThunderFrog MEDIA GmbH kurz vorgestellt und präsentiert.

Abschließend wird das Projekt noch vom Vorgesetzten Herrn Schulten begutachtet.

Als sichergestellt wurde, dass das Projekt allen Anforderungen entspricht, wurde dieses in an die IT-Abteilung des BFZ Essen GmbH übergeben.

7.1 Installation auf Kunden PC

Aufgrund dass das Programm in Java programmiert wurde, gab es für die Inbetriebnahme auf den PC-Systemen des BFZ-Essen GmbH nur die Einschränkung das eine Java Version, vorzugsweise die aktuellste, installiert ist. Sobald das Programm auf den Ziel-PC´s kopiert wurde, genügt ein doppelklick auf die .jar Datei um das Programm zu starten.

7.2 Schulung des Personals

Dem Personal welches das Programm nutzt wurde die GUI erklärt. Es wurde erklärt wo die fertige PDF nach dem Speichern zu finden ist. Ein wichtiger Hinweis wurde zum DatePicker gegeben das dort nur Montage ausgewählt werden können, da eine Schulwoche, so wie sie auf dem PDF steht, immer mit einem Montag anfängt.

8. Dokumentation

Die Dokumentation des Projektes zum speichern des Online Klassenbuch als PDF besteht aus der Projektdokumentation.

In der Projektdokumentation werden die einzelnen Phasen, die während der Umsetzung des Projektes durchlaufen wurden, beschrieben.

Für weitere Details der einzelnen Funktionen sind im Anhang die Klassendiagramme beigelegt, dort werden auch die einzelnen Übergabeparameter genauer erklärt.

9. Fazit

9.1 Soll-/Ist-Vergleich

Das IHK-Projekt wurde allen Anforderungen gerecht, gemäß Pflichtenheft und zusätzlichen Anforderungen vom Auftraggeber fertiggestellt. Der zum Beginn des Projektes erstellter Zeitplan konnte eingehalten werden. Die tatsächliche Zeit die das Projekt in Anspruch genommen hat wird in der unten stehende Tabelle ersichtlich.

Projektphase	Soll	Ist	Differenz
Analyse	6 h	8 h	+ 2 h
Entwurf	8 h	7 h	- 1 h
Implementierung	33 h	36 h	+ 3 h
Qualitätsprüfung	7 h	7 h	0 h
Rollout	6 h	2 h	- 4 h
Dokumentation	10 h	10 h	0 h
Gesamt	**70 h**	**70 h**	**70 h**

9.2 Wertvolle Erfahrungen

Über die gesamte Projektdauer hinweg konnten wertvolle Erfahrungen bezüglich der Planung und Durchführung eines eigenen Projektes gesammelt werden.

Besonders deutlich wurde die Wichtigkeit eines durchdachten Entwurfes, um die Implementierungsphase strukturiert durchführen zu können.

Die Realisierung des Projektes stellt somit nicht nur einen Erfolg für das Unternehmen dar, sondern hat auch persönlich wichtige und lehrreiche Bereicherungen mit sich ziehen lassen.

Wichtig ist noch anzumerken dass sobald das Projekt gestartet wurde der Auftraggeber nicht zu vergessen ist. Eine stetige Kommunikation sorgt dafür das frühzeitig bemerkte Fehler, Anpassungen oder Verbesserungen schneller mit dem Auftraggeber besprochen und umgesetzt werden können.

Auch erhält der Auftraggeber immer einen Einblick in den aktuellen Status vom Projekt.

9.3 Ausblick

Alle Anforderungen aus dem Lastenheft wurden realisiert. In Zukunft kann das Programm noch um weitere Funktionen erweitert werden.

Zum Beispiel kann eine Vorschaufunktion, ein Dialog zum Speichern mit individuellen Dateinamen und/oder ein Login um den zugriff Unbefugte zu verhindern.

ANHANG

A.1 - Detaillierte Projektphasen

Projektphase	Geplante Zeit
Analyse	6 h
Ist-Analyse	2 h
Wirtschatlichtsprüfung	1 h
Erstellen des Lastenheft	3 h
Entwurf	8 h
UML Diagramm erstellen	2 h
Nutzwertanalyse zur Auswahl des Frameworks	1 h
Benutzeroberfläche entwerfen	2 h
Pflichtenheft erstellen	3 h
Implementierung	33 h
Import von .csv Daten	4 h
Verarbeiten von .csv Daten	4 h
Vorschau der Daten erstellen	5 h
Speichern der verarbeiteten Daten als PDF	4 h
Formatieren der Daten	3 h
Benutzeroberfläche umsetzen	7 h
Testdaten erzeugen	6 h
Qualitätsprüfung	7 h
Import Test mit Testdaten	3 h
Ausdrucken der PDF mit Testdaten	4 h
Rollout	6 h
Installation auf End-PC-System	2 h
Schulung des Personal	4 h
Dokumentation	10 h
Projektdokumentation	6 h
Entwicklerdokumentation	4 h

A.2 - Verwendete Ressourcen

Hardware

- Büroarbeitsplatz
  - PC-System
  - Tastatur
  - Maus
  - 3 Monitore
  - Bürostuhl

Software

- Windows 10 Betriebssystem
- Intellij IDEA - Entwicklungsumgebung
- Java Open JDK 15
- MS Excel / OpenOffice zum überprüfen der Exportdaten

Personal

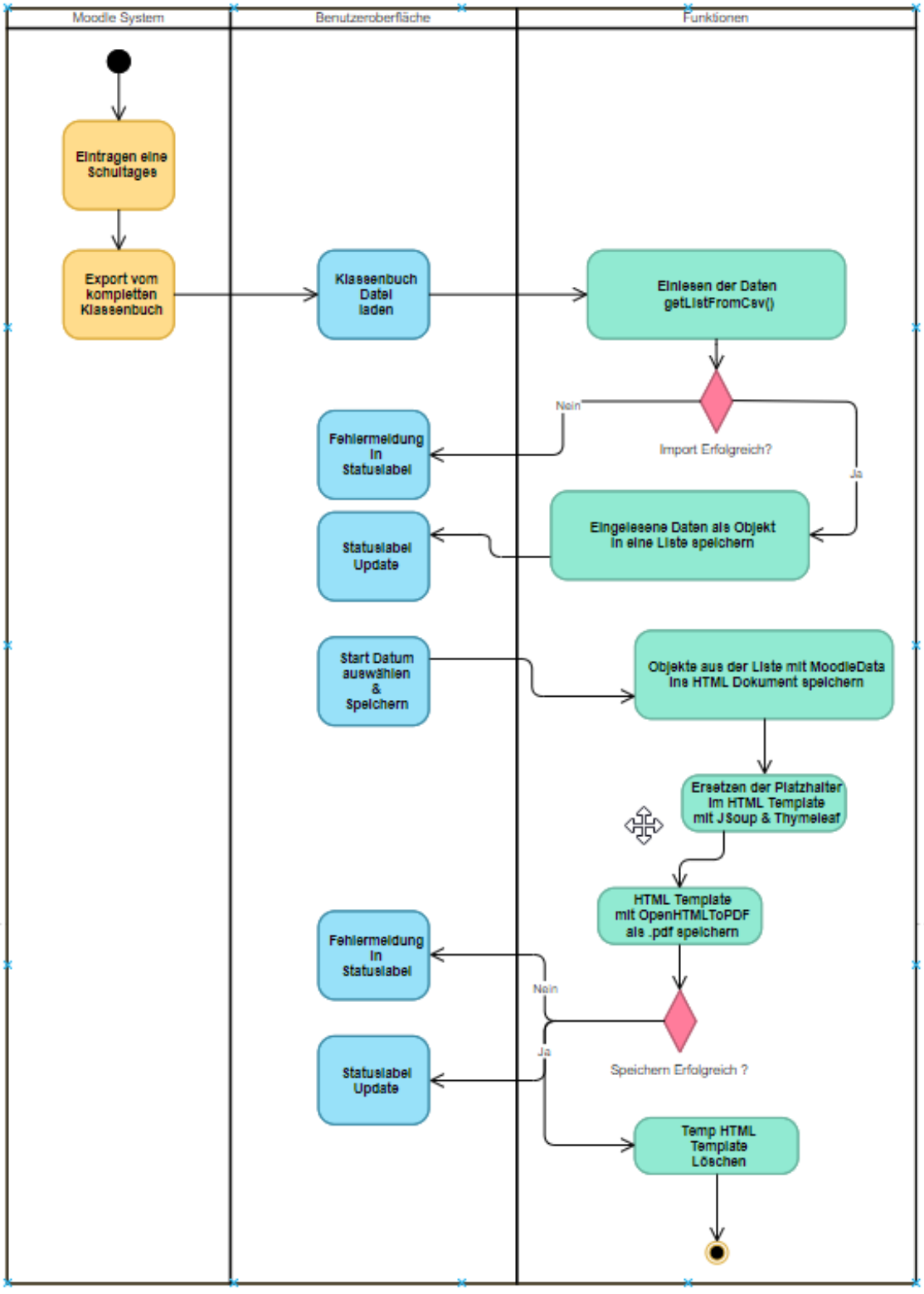
- Entwickler - Umsetzung des Projektes
- Anwendungsentwickler*innen - Zur Überprüfung
- Abteilungsleiter - Zur Abnahme

A 2.1 - Auszug aus dem Pflichtenheft

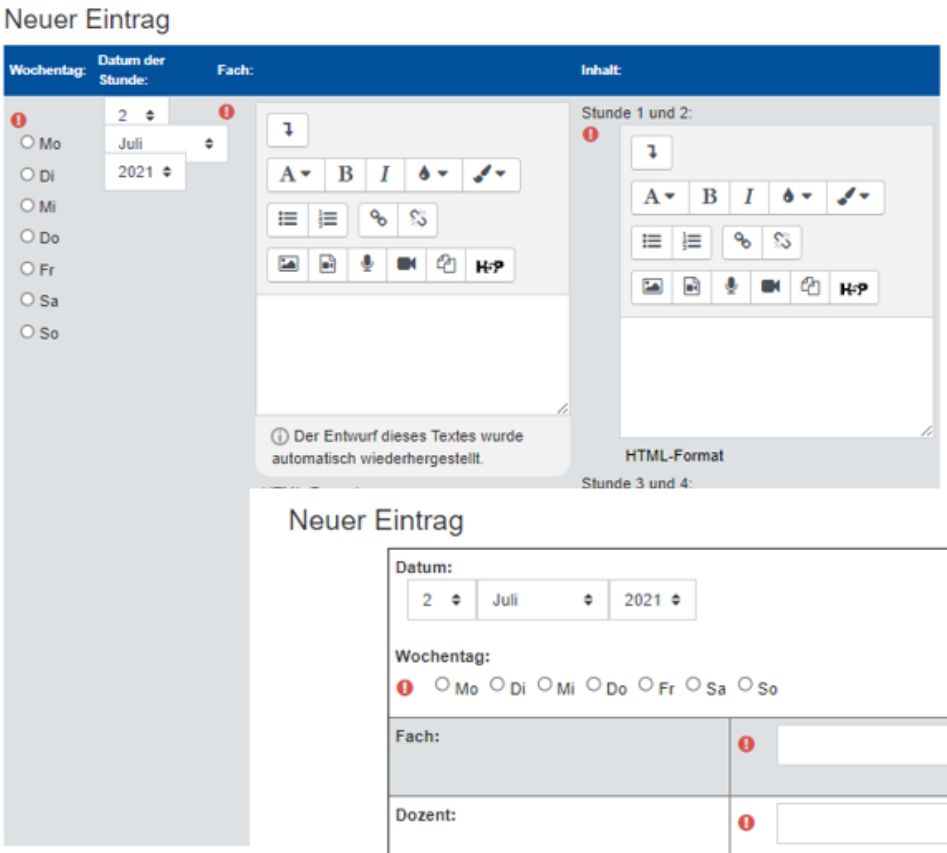
In folgendem Auszug aus dem Pflichtenheft wird die geplante Umsetzung der im Lastenheft definierten Anforderungen beschrieben:

- Es muss ein eigenständiges Programm sein, kein PlugIn für Moodle.
- Als Programmiersprache soll Java verwendet werden. Welche Hilfsmittel man dazu nimmt, ist den Entwickler*innen überlassen.
- Das Klassenbuch Formular kann in Absprache mit dem Auftraggeber angepasst werden, wenn notwendig
- Die Daten, die Umgewandelt werden sollen, werden aus dem Moodle-Portal exportiert und lokal gespeichert
- Die zu konvertierende CSV-Datei soll vom Nutzer eigenständig frei auswählbar sein
- Die PDF soll optisch dem bisherigen Klassenbuch gleichen
- Eine Dokumentvorlage wurde vom Auftraggeber zur Verfügung gestellt

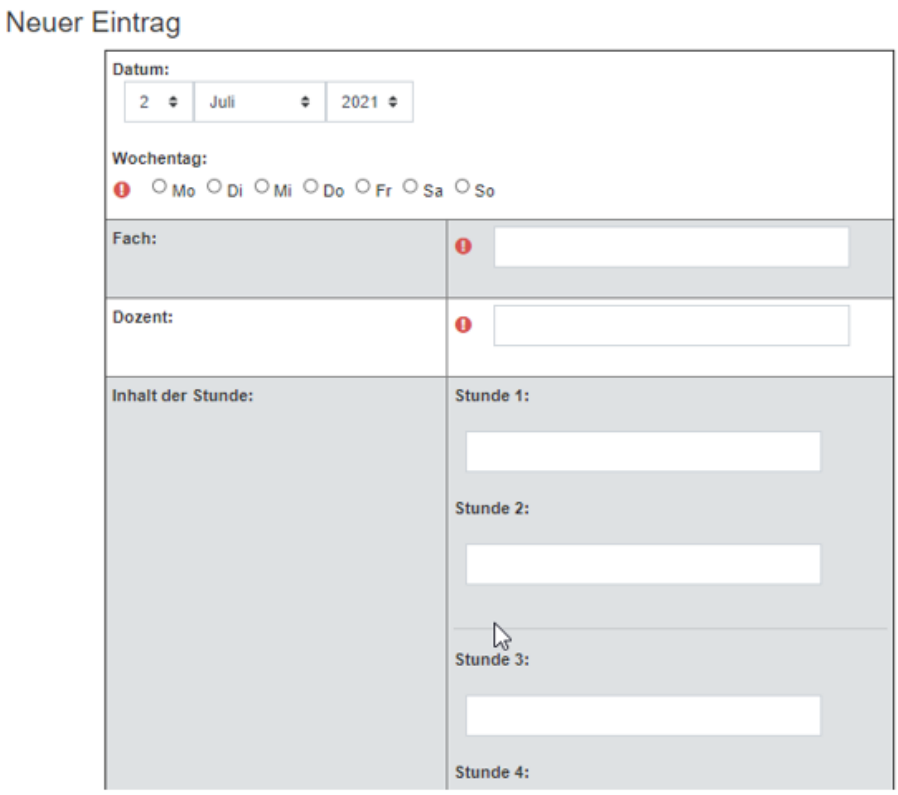
A.3 - Aktivitätsdiagramm



A.4 - Vorher/Nachher Vergleich

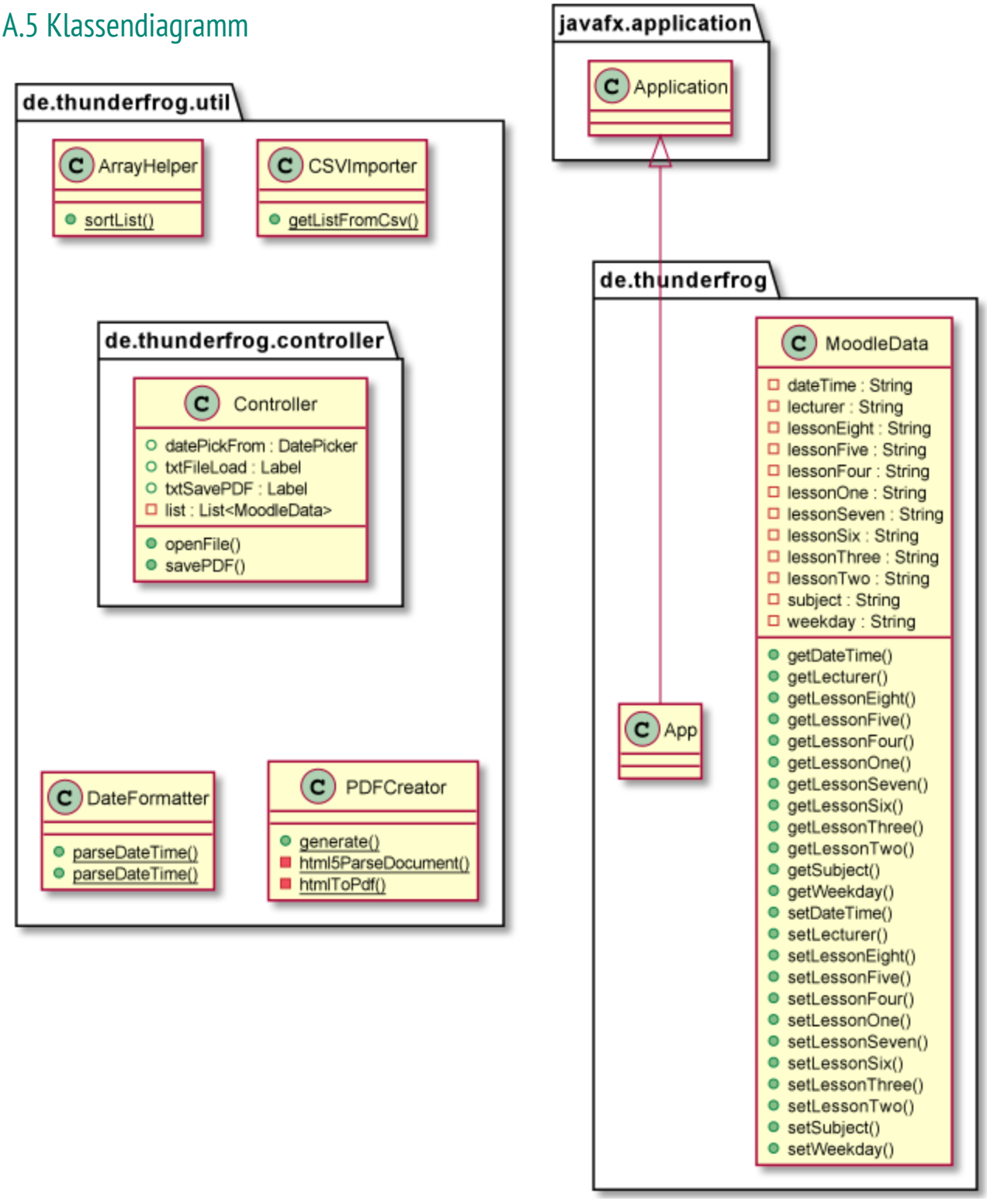


<= **Alte Eingabemaske**



Neue Eingabemaske =>

A.5 Klassendiagramm



A.6 - Quellcode Klasse CSVImporter

```
public class CSVImporter {
    /**
     * CSV aus einer Datei auslesen und als MoodleData in einer ArrayList speichern
     * @param filename
     * @return
     * @throws IOException
     */
    public static List<MoodleData> getListFromCsv(File filename) throws IOException {
        System.out.println("Datei wird gesucht & geladen");
        try {
            //  FileReader und Apache Commons initialisieren
            Reader in = new FileReader(filename);
            Iterable<CSVRecord> records = CSVFormat.DEFAULT.withDelimiter(',').withHeader().parse(in);

            //  ArrayList mit MoodleData Datentyp
            ArrayList<MoodleData> list = new ArrayList<>();

            //  Für jeden eintrag in der CSV (Zeile) ein neues MoodleData Objekt erstellen
            for (CSVRecord record : records) {
                MoodleData moodleData = new MoodleData();
                moodleData.setDateTime(record.get("Datum der Stunde"));
                moodleData.setWeekday(record.get("Wochentag"));
                moodleData.setSubject(record.get("Fach"));
                moodleData.setLessonOne(record.get("Stunde_1"));
                moodleData.setLessonTwo(record.get("Stunde_2"));
                moodleData.setLessonThree(record.get("Stunde_3"));
                moodleData.setLessonFour(record.get("Stunde_4"));
                moodleData.setLessonFive(record.get("Stunde_5"));
                moodleData.setLessonSix(record.get("Stunde_6"));
                moodleData.setLessonSeven(record.get("Stunde_7"));
                moodleData.setLessonEight(record.get("Stunde_8"));
                moodleData.setLecturer(record.get("Dozent"));

                // MoodleData Objekt in die ArrayListe schreiben
                list.add(moodleData);
            }
            System.out.println("Datei geladen " + list.size());
            return list;
        } catch (FileNotFoundException ex) {
            System.out.println("Datei nicht gefunden!");
        }
        return null;
    }
}
```

A.7 - Quellcode DateFormatter

```
public class DateFormatter {
    /**
     * Epoch Time Format in lesbares Datum umwandeln
     * @param dateTime
     * @return String parsedDate
     */
    public static String parseDateTime(String dateTime){
        LocalDate parsedDate = LocalDate.ofInstant(Instant.ofEpochSecond(Long.parseLong(dateTime)), ZoneId.systemDefault());
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
        return parsedDate.format(formatter);
    }

    /**
     * Date Format in deutsches Datum Format umwandeln
     * @param date
     * @return
     */
    public static String parseDateTime(LocalDate date){
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
        return date.format(formatter);
    }
}
```

A.8 - Context aus PDFCreator.java

```
  //  Context in das Template schreiben
  Context ctx = new Context();
  int j = 0;
  for (int i = 0; i < list.size(); i++) {
      if(DateFormatter.parseDateTime(list.get(i).getDateTime()).compareTo(date) >= 0){
          //System.out.println("j = " + j);
          ctx.setVariable("WEEKDAYSTART", date);

          //  Selected Date plus 4 Tage
          String nDate = date;
          SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd.MM.yyyy");
          Calendar c = Calendar.getInstance();
          c.setTime(simpleDateFormat.parse(nDate));
          c.add(Calendar.DATE,4);
          nDate = simpleDateFormat.format(c.getTime());


          ctx.setVariable("WEEKDAYEND", nDate);
          ctx.setVariable("WEEKDAY_" + j, list.get(i).getWeekday());
          ctx.setVariable("STARTDATE_" + j, DateFormatter.parseDateTime(list.get(i).getDateTime()));
          ctx.setVariable("SUBJECT_" + j, list.get(i).getSubject());
          ctx.setVariable("LESSONONE_" + j, list.get(i).getLessonOne());
          ctx.setVariable("LESSONTWO_" + j, list.get(i).getLessonTwo());
          ctx.setVariable("LESSONTHREE_" + j, list.get(i).getLessonThree());
          ctx.setVariable("LESSONFOUR_" + j, list.get(i).getLessonFour());
          ctx.setVariable("LESSONFIVE_" + j, list.get(i).getLessonFive());
          ctx.setVariable("LESSONSIX_" + j, list.get(i).getLessonSix());
          ctx.setVariable("LESSONSEVEN_" + j, list.get(i).getLessonSeven());
          ctx.setVariable("LESSONEIGHT_" + j, list.get(i).getLessonEight());
          ctx.setVariable("LECTURER_" + j, list.get(i).getLecturer());
          j++;
      }else{
          System.out.println("Datum Ungleich!");
      }
  }
```