Hochschule für
Technik und Wirtschaft
Dresden
University of Applied Sciences

Entwicklerdokumentation

OPAL DRUCKSERVICE

Project Manager: Francesco Ryplewitz (FR)

Analyst: Paul Rogge (PR)

Architect: Felix Müller (FM)

Developer: Paul Jannasch (PJ), Annabelle Zimmer (AZ), Felix Müller (FM)

Tester: Paul Rogge (PR)

Deployment Engineer: Jan Heelemann (JH)

Technical Writer: Jan Heelemann (JH)

INHALTSVERZEICHNIS

1 EINFÜHRUNG

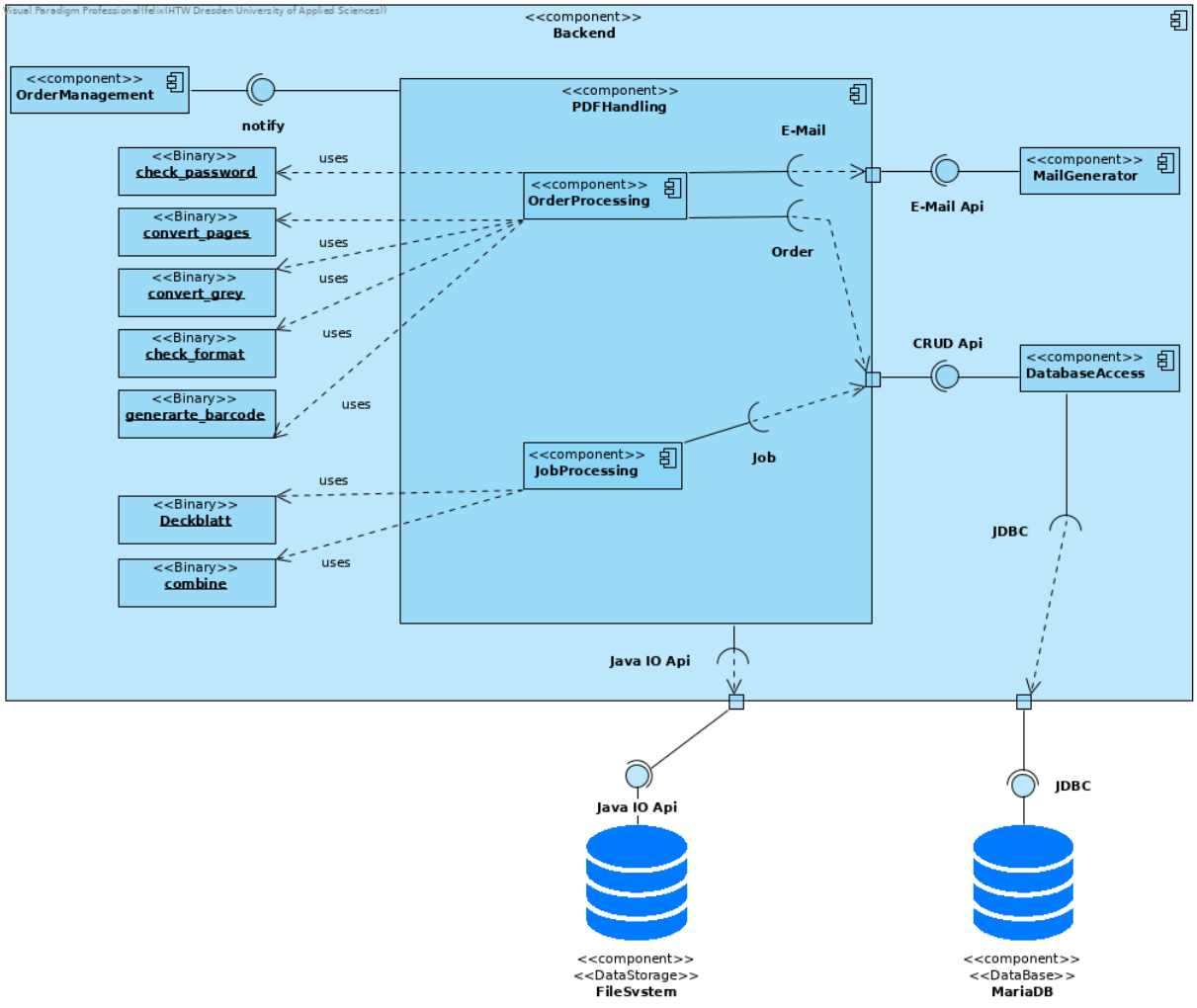
Dieses Dokument beinhaltet alle wichtigen Informationen über den Aufbau, die Wirkungsweise und den Inhalt des Quellcodes für das Projekt Opal Druckservice. Diese Informationen dienen dazu, die Software zu warten, zu modifizieren oder gegebenenfalls weiterzuentwickeln.

2 ARCHITEKTUR FRAMEWORK

2.1 HINTERGRUNDSYSTEM

Das Hintergrundsystem wird in drei wesentliche Komponente strukturiert:

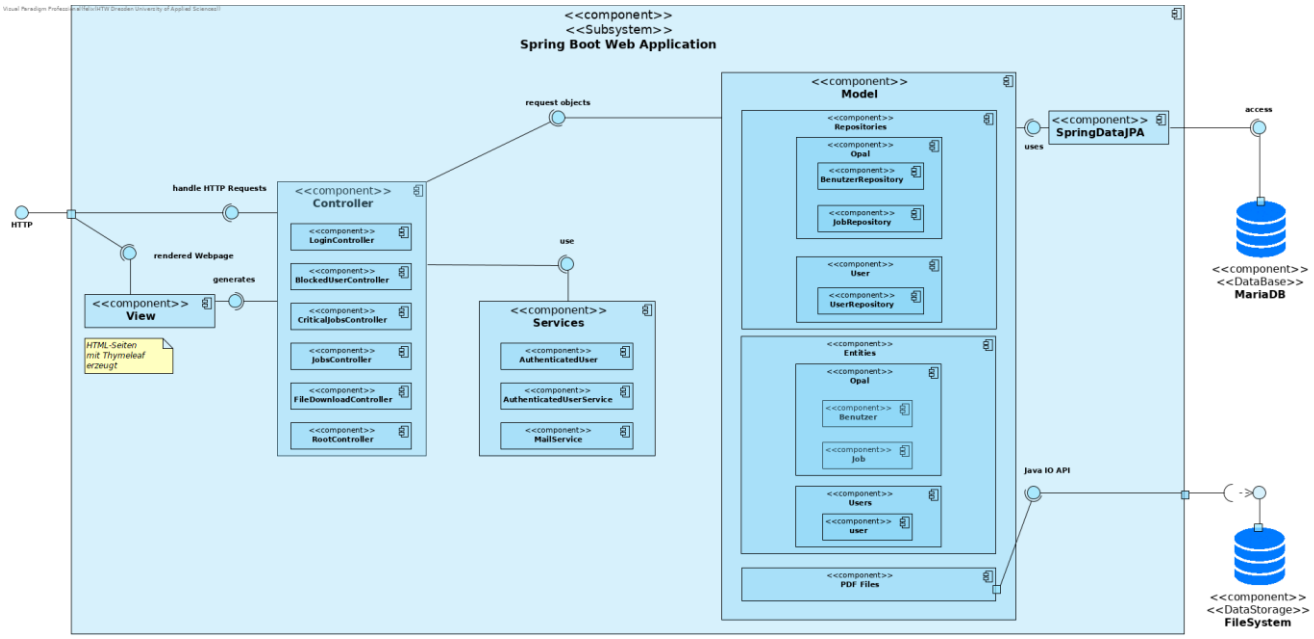
- OrderManagement: dient der Abholung neuer Aufträge (E-Mail-Kommunikation)
- Logik-Schicht: implementiert die Geschäftslogik zur Umwandlung der Dokumente und zum Versenden von E-Mails
- Daten-Schicht: dient dem persistenten Abspeichern von Dokumenten, Benutzerinformationen, Druckaufträgen und -jobs (Datenpersistenz)



2.2 Zuständigkeiten der Komponenten

- OrderManagement: holt Aufträge ab
- PDF-Handling: steuert die Abarbeitung der Bash-Skripte, behandelt Fehler, speichert umgewandelte Dokumente ab und aktualisiert DB-Einträge
    1. Preprocessing: Graustufen und Blattseiten
    2. Processing: Aufträge zu Druckjob zusammenfassen und Deckblatt erstellen
- DataAccessManagement: zuständig für alle DB-Anfragen, kapselt DB-Zugriff, stellt CRUD API bereit
- MailGenerator: Generieren und Versenden von E-Mails (E-Mail-Kommunikation)

3 Web-Anwendung



1. Webanwendung basiert auf klassischem MVC Pattern
2. Als Webframework wird Thymeleaf verwendet
3. Zusammenhänge zwischen einzelnen Klassen und Methoden lassen sich in der angebundenen JavaDoc Dokumentation nachvollziehen

4 LOGISCHE SICHT

Beschreibung von Schnittstellen und Paketstrukturen.

4.1 SCHNITTSTELLE ZWISCHEN OPAL-SERVER UND HINTERGRUNDSYSTEM

- Dokument zu Druckauftrag: Anhang der E-Mail
- Druckeinstellungen: im E-Mail-Body (XML Format)
- Benutzerinformationen: im E-Mail-Body (XML Format)
- diese Schnittstelle soll perspektivisch durch eine REST-API des Opal-Servers und eine entsprechende Komponente im Hintergrundsystem abgelöst werden.

Darstellung	
Benutzerinformationen	**Bibliotheksnummer**
	Name
	Vorname
Auftragsinformationen	**Dateiname** (ID aus Datenbank)
	Druckeinstellungen
	Preis

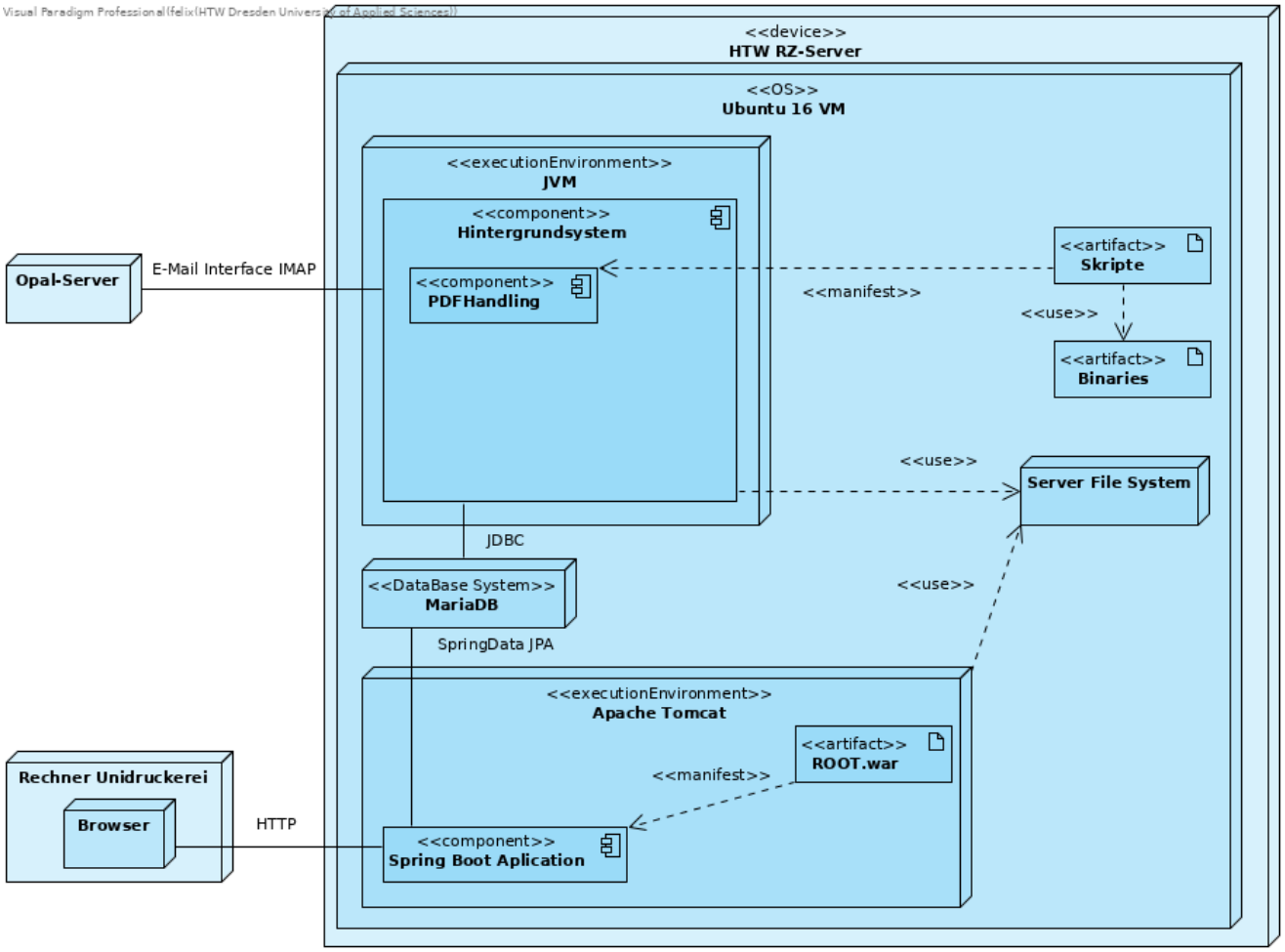
4.1.1 Schnittstelle zwischen Hintergrundsystem und Webanwendung

- PDF-Dokumente und Datenbank (siehe Datenpersistenz)

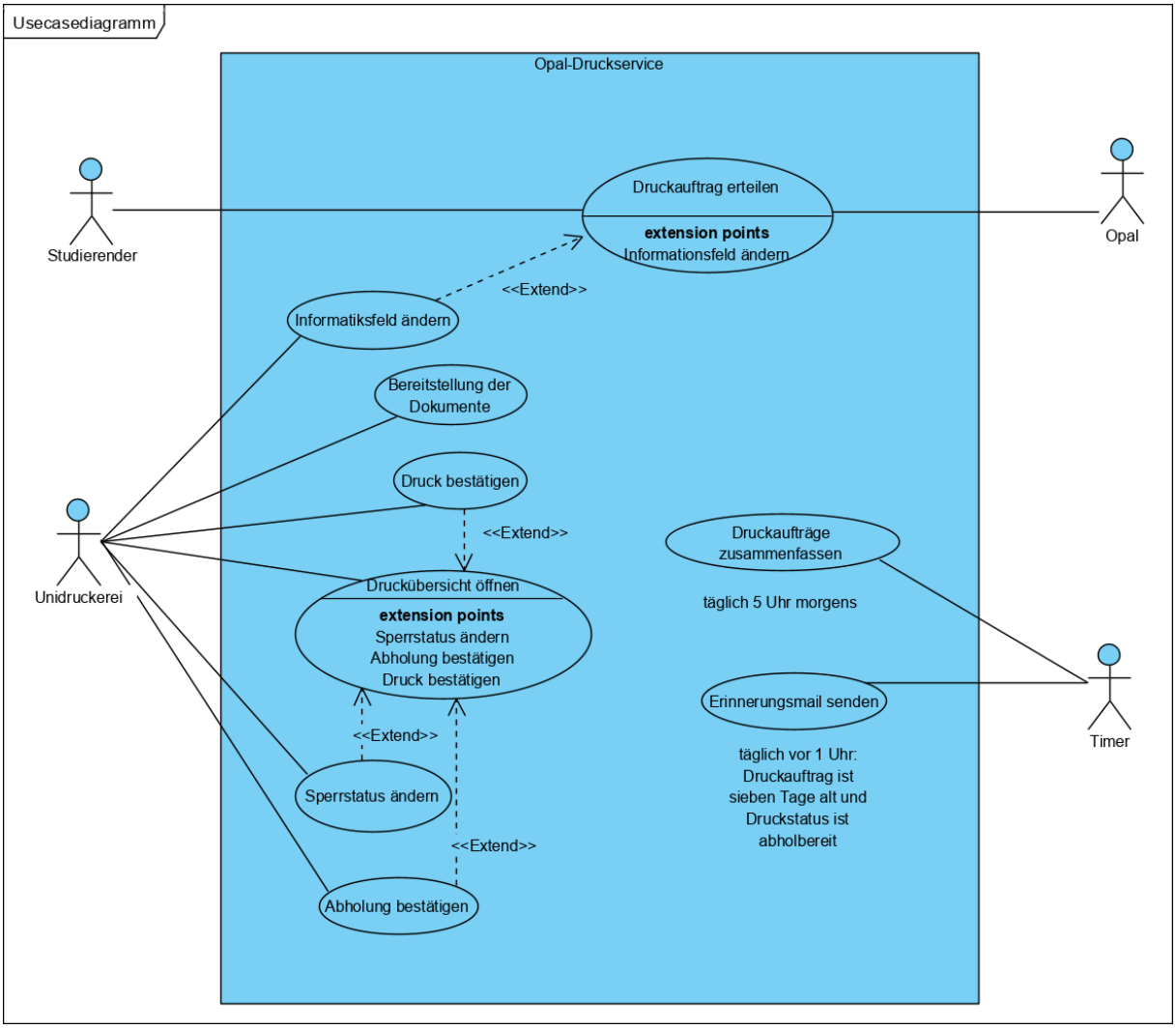
4.1.2 Schnittstelle zwischen Webanwendung und UNI-Druckerei

- HMTL, (CSS und JavaScript)
- PDF und Zip

4.2 Physische Sicht (Betriebssicht)



4.3 Use Cases



4.4 Datenbankschema

5 KLASSENÜBERSICHT

5.1 PROZESSSTEUERUNG

5.1.1 Run.java

Die Run Klasse bildet das Gesamtprogramm ab, welches alle anderen Klassen mit deren Funktionalitäten nutzt. Es ist die einzige Klasse mit einer Main-Funktion.

- Main() (<- pub.stat.void.main()) bildet das gesamte Programm ab und ruft die Funktionalitäten direkt oder untergeordnet über eine Funktion auf
    - While(true) ist die Programmdauerschleife - in ihr passieren alle folgenden Aktivitäten
    - Wenn die Anzahl an Mails im Postfach größer 0 ist, werden alle Mails einzeln mittels processFiles() verarbeitet
        - processFile() führt die DB Inserts/Abfragen aus und führt über den ProcessHandler die Shell Skripte aus, die die PDFs umwandeln und verarbeiten
        - Hier wird also jede Mail und deren Inhalt einzeln verarbeitet, die Funktion wird iterativ für jede Mail aufgerufen
    - Wenn es vor 1:00 Uhr ist, werden die Jobs für den neuen Tag wieder auf nicht erledigt gesetzt und Erinnerungsmails an alle Studenten, die vor genau 7 Tagen einen Auftrag aufgegeben und nicht abgeholt haben gesendet
    - Nach 5:00 Uhr werden alle Jobs für den Tag mittels der DailyTask Klasse erledigt und die Jobs für den Tag auf erledigt gesetzt

5.1.2 ProcessHandler.java

- Diese Klasse besitzt nur die Funktion runScript() welche mittels des ProcessBuilder die Shell Skripte mit unterschiedlichen Parametern aufruft und somit die Konvertierung der Druckaufträge steuert
- Der Rückgabewert der Skripte wird abgefangen und in Run und DailyTask verglichen - wenn dieser nicht korrekt ist, wird eine Fehlermail versendet

5.1.3 DailyTask.java

Hier findet die tägliche Zusammenfassung der Druckaufträge eines Studenten zu einem Druckjob statt. Pro Druckjob wird zudem ein Deckblatt erstellt.

- In der Methode run() startet der DailyTask durch Aufruf aus der Klasse Run. Sie ruft generateJobs() auf. Nachdem die Druckjobs verarbeitet wurden, werden alle eventuell noch vorhandenen Restdateien gelöscht. Die Druckjoberzeugung endet und Run wird wieder gestartet
- In generateJobs() werden die Druckjobs entsprechend der s-Nummern der Aufträge in der Datenbank erstellt und einzeln zur PDF-Verarbeitung an processJobs() weitergegeben
- processJobs() verarbeitet reihenweise die einzelnen Dateien eines jeden Druckjobs. Die zum Druckjob dazugehörigen Dateien werden gesucht (Aufruf getFiles()) und ein Abgleich zwischen Datenbank und Ordner bezüglich der zusammenzufassenden Aufträge durchgeführt. Ein Barcode wird generiert und danach ein Deckblatt erstellt. Dieses enthält die Daten des Druckjobs und den Barcode. Anschließend wird das Deckblatt mit einer leeren Seite kombiniert (Rückseite). Zum Schluss werden das Deckblatt und die Druckaufträge jeweils immer paarweise zusammengefügt
- getFiles() sucht nach vorverarbeiteten Dateien. Es wird nach Dateianfängen entsprechend dem übergebenen Muster gesucht und die vollen Namen der gefundenen Dateien zurückgegeben

5.2 DATENBANKANBINDUNG

5.2.1 DBInterface.java

- Gibt nur generelle Datenbankfunktionen an, die in jeder Klasse mehr oder weniger implementiert sind, um Redundanzen zu vermeiden
- Funktionen sind insert(), delete(), update(), toString()
- insert() fügt das Objekt „this“ ein
- delete() entfernt je nach Bedarf das Objekt oder die gesamte Tabelle
- update() ändert ein Objekt
- toString() gibt alle Attribute eines Objekts zurück
- getObject()/getAll() gibt das Objekt selber oder alle Objekte der Tabelle zurück

5.2.2 DBConnection.java

- connectToDB() stellt die Datenbankverbindung über JDBC her
- disconnectFromDB() baut die Verbindung wieder ab

5.3 TABELLENKLASSEN

5.3.1 Benutzer.java

- exists() gibt zurück, ob der Benutzer bereits in der Tabelle vermerkt ist
- getSperrstatus() schaut, ob der Benutzer gesperrt ist (wenn exists() == true)
- sperren() sperrt den Nutzer -> Webanwendung (kommt in Serveranwendung nicht vor, ist aber vorhanden)

5.3.2 Druckauftrag.java

- SetID() setzt die ID des Benutzer-Objekts, da diese über die Datenbank automatisch generiert wird
- DateiCount() gibt die Anzahl an Dateien für den Druckjob eines bestimmten Nutzers zurück
- SeitenzahlSumme() gibt die Seitenanzahl eines Druckjobs aus (alle Dateien mit gleicher Snr)
- getAuftragBibNr() gibt alle Benutzer mit Aufträgen als String Array zurück
- getSeitenzahl() liefert die Seitenanzahl eines Druckauftrages
- getDateiname() liefert den Dateinamen eines Druckauftrages
- deleteElement() löscht den Druckauftrag (delete() aus DBInterface löscht die ganze Tabelle)
- getAuftragID() gibt die AuftragsIDs zu einer Snr zurück
- countDateinameBenutzer() gibt die Anzahl einer bestimmten Datei von einem bestimmten Studenten zurück

5.3.3 Druckjob.java

- setID() setzt die durch die Datenbank generierte ID für das Objekt
- getBibNr() gibt alle Bibliotheksnummern mit Jobs als String Array zurück
- getAllOverdueJobs() gibt alle Nutzer mit nichtabgeholten Druckjobs = 7 Tage zurück
- updateSeitenzahl() ändert die Seitenzahl eines Druckjobs bei fehlerhafter Verarbeitung einer Datei
- updatePreis() ändert den Preis eines Druckjobs bei fehlerhafter Verarbeitung einer Datei

5.4 EMAILANBINDUNG

5.4.1 MailClient.java

- Der MailClient dient zum Auf- und Abbau der Connection zum Mailserver Exchange und die Verwaltung der verschiedenen Mailinhalte wie Daten zum Auftraggeber, Druckeinstellungen und die angehängten PDFs
- initSend() initialisiert den Client für das Senden von Mails an den Nutzer (Bestätigung, Erinnerung, Fehler, …)
- initListen() initialisiert den Client, um die Inbox auszulesen
- prepareMessage() bereitet eine Mail Message für das Versenden vor
- getSnr() und getEinstellungen() parsen empfangene Mails nach der S-Nummer der Benutzers und den gewählten Einstellungen für den Druckauftrag
- getMailCount() wird verwendet um die Anzahl an Mails in der Inbox zu ermitteln um über diese zu iterieren
- sendMail() versendet eine Nutzermail
- getMailContent() ermittelt den Inhalt der Mail (sowohl Text als auch Anhang)
- IMAPListener() liest die Inbox aus und ermittelt mittels getMailContent() den Inhalt rekursiv durch die verschiedenen Body-/Multiparts
- replaceName() passt den Namen der angehängten PDF an, falls Sonderzeichen vorkommen
- setCustomContent() setzt den Inhalt nach Mailtyp für die Mails an die Studenten (Bestätigung, Fehler, …)

5.5 SKRIPTE

5.5.1 check_password

- ermittelt mittels gs ob die PDF passwortgeschützt ist
- exit 0 -> kein Passwort, exit 1 -> Passwort

5.5.2 check_format

- ermittelt die Ausrichtung der Folien in der PDF
- exit 0 -> Querformat (landscape), exit 1 -> Hochformat (portrait)

5.5.3 convert_grey

- Umwandlung der PDF in Graustufen mittels gs

5.5.4 convert_pages

- Konvertierung der PDF in das gewünschte Format (1x1, 1x2, 2x2) und richtige Drehung der Folien mit Nutzung von check_format

5.5.5 Combine

- Kombiniert mittels pdfnub immer zwei PDFs zu einer einzelnen
- Nutzung für die Erstellung der Druckjobs -> kombiniert immer eine PDF mit der nächsten rekursiv bis alle Dokumente eines Studenten den fertigen Druckjob ergeben
- Nutzung für die Erstellung des Deckblattes -> kombiniert generiertes Deckblatt mit leerer Seite (Vermeidung Deckblatt und erster Auftrag auf einem Blatt)
- Nutzung für die Anpassung von Druckaufträgen mit ungerader Seitenzahl -> kombiniert Dokument mit ungerader Seitenzahl mit leerer Seite (Vermeidung zwei Druckaufträge auf einem Blatt)

5.5.6 generate_barcode

- generiert den Barcode mittels des Befehles „barcode“ für den berechneten Preis als *.ps
- danach erfolgt die Umwandlung in eine PDF mittels ps2pdf

5.5.7 generate_deckblatt

- gegeben ist das Deckblattformular (form_template.pdf), welches auszufüllen ist
- mittels der Parameter, welche übergeben werden, wird eine Form.fdf Datei erstellt, die dann das Formular mit den Parametern als Werten befüllt (mit pdftk)
- mit pdfjam wird dann der Barcode als PDF-Datei auf das Deckblatt gestempelt

6 JAVADOC

Für den Java Quellcode existiert auch eine mittels JavaDoc erzeugte Dokumentation. Diese wird diesem Dokument beigefügt.