

Bericht über das   
Praktische Studiensemester

vom 01.03.2022 bis 31.08.2022

bei SEITENBAU

Abteilung Software Developement

and Services

**Praktikant/in: Jannis Liebscher**

Matr,. Nr.: 301645

Studiengang: AIN

Hochschulbetreuer/in: Titel, Name

Fimenbetreuer/in: Developer,

Kathin Wolpers

Datum der Abgabe: dd.mm.yyyy

**Inhalt**

[Vorwort IV](#_Toc116123294)

[Formel- und Abkürzungsverzeichnis V](#_Toc116123295)

[Abbildungsverzeichnis VI](#_Toc116123296)

[Tabellenverzeichnis VII](#_Toc116123297)

[1 Einleitung 1](#_Toc116123298)

[2 Der Betrieb 2](#_Toc116123299)

[3 Arbeitsweise 3](#_Toc116123300)

[4 Bearbeitete Aufgaben / Projekte 4](#_Toc116123301)

[4.1 Aufgabe 1 5](#_Toc116123302)

[4.1.1 Problembeschreibung 5](#_Toc116123303)

[4.1.2 Aufgabenstellung und Zielsetzung 5](#_Toc116123304)

[4.1.3 Theoretische Grundlagen 5](#_Toc116123305)

[4.1.4 Realisierung 6](#_Toc116123306)

[4.1.5 Nachweis/Bewertung der Funktionsfähigkeit 7](#_Toc116123307)

[4.1.6 Zusammenfassung 8](#_Toc116123308)

[4.2 Aufgabe 2 9](#_Toc116123309)

[4.2.1 Aufgabenbeschreibung und Zielsetzung 9](#_Toc116123310)

[4.2.2 Theoretische Grundlagen 9](#_Toc116123311)

[4.2.3 Realisierung 10](#_Toc116123312)

[4.2.4 Nachweis/Bewertung der Funktionsfähigkeit 11](#_Toc116123313)

[4.2.5 Zusammenfassung 11](#_Toc116123314)

[4.3 Aufgabe 3 12](#_Toc116123315)

[4.3.1 Aufgabenbeschreibung und Zielsetzung 12](#_Toc116123316)

[4.3.2 Theoretische Grundlage 12](#_Toc116123317)

[4.3.3 Realisierung 12](#_Toc116123318)

[4.3.4 Nachweis/Bewertung der Funktionsfähigkeit 13](#_Toc116123319)

[5 Persönliche Eindrücke und Erfahrungen 14](#_Toc116123320)

[Literaturquellen VI](#_Toc116123321)

[Anhang VIII](#_Toc116123322)

[Anhang A (Titel) VIII](#_Toc116123323)

[Anhang B (Titel) VIII](#_Toc116123324)

[Anhang C (Titel) VIII](#_Toc116123325)

[Weitere Hinweise IX](#_Toc116123326)

Vorwort

Dies ist eine Vorlage zur Erstellung des Praxissemester-Berichts (PSS-Bericht). Sie sollten die Vorlage nutzen um damit Ihren eigenen PSS-Bericht zu schreiben.

Die Vorlage enthält Formatierungsfunktionen, die Ihnen bei der Erstellung behilflich sein werden (Automatische Verzeichnisse mit Seitenangaben, automatische Text-Formatierung über Formatvorlagen, etc..). Grundsätzlich entsprechen die verwendeten Funktionen jenen, die sie bereits aus der Vorlage für Laborberichte kennen.

Erläuterungen zur Nutzung der Formatierungs-Funktionen finden Sie im “**Handbuch zur Vorlage 2016**“. Das Handbuch ist ebenfalls im Moodle-Kurs des Praxissemesters abgelegt.

Formel- und Abkürzungsverzeichnis

Keine Indexeinträge gefunden.

Hinweise:   
Im Abkürzungsverzeichnis werden die verwendeten Abkürzungen getrennt aufgelistet. Für Formelzeichen und Abkürzungen ist eine getrennte Liste zu führen.

Für **Formelzeichen** sind wie folgt aufzulisten:

*Formelzeichen phys. Einheit* Benennung z.B   
*g m/s2 Erdbeschleunigung*

Formelzeichen müssen im Text bei der ersten Nen­nung ausgeschrieben werden, z.B. „Die Federkonstante *c* beträgt …“. Im weiteren Text darf die Abkürzung verwendet werden: „*c* hängt ab von …“

Im **Abkürzungsverzeichnis** sind alle verwendeten Abkürzungen die nicht in Formeln verwendet werden aufzulisten z.B.

*Abt. CK – Abteilung Controlling und Koordination*

Nicht angegeben werden allgemeingebräuchliche Abkürzungen wie „usw.“, „z. B.“, „o. Ä.“ oder „d. h.“ sowie wissenschaftlichen Standardabkürzungen wie „ebd.“, „s.“, „S.“, „vgl.“

Abbildungsverzeichnis

Abb. 1 Beispielhafter Fehlerbericht von PAC 2021 5

Abb. 2 Das fox:alt Attribut in einem Grafik Element des Stylesheets 6

Abb. 3 Ausschnitt der Fonts-Sektion der Apache FOP Konfigurationsdatei 6

Abb. 4 Die Tabelle der Mantelbogen PDF 7

Abb. 5 Die Tabelle der Mantelbogen PDF als Screenreader Ansicht in PAC3

7

Abb. 6 Ausschnitt aus der Datei docker-compose-yaml 9

Abb. 7 Ausschnitt aus docker-compose.yaml, Konfigurations der Volumes

von edok-informationsdienst 10

Abb. 8 Ausschnitt aus docker-compose.yaml, Port Konfiguration des Proxys

11

Abb. 9 Das Alte Layout der PKP Startseite 12

Abb. 10 Das neue Layout der PKP Startseite 12

Abb. 11 Thymeleaf Fragment, welches einen Quicklink darstellt 13

Abb. 12 Ausschnitt aus der SVG Datei (Links) mit den Kommandos die zu

dem Pfeil Icon (Rechts) führen 13

# Einleitung

Die vorliegende Arbeit beschäftigt sich mit meiner Arbeit an den Projekten eDok (Elektronische Dokumente) und PKP (Gemeinsames Planungs- und Kabinettsmanagement Programm) der Firma SEITENBAU, im Rahmen meines Praxissemesters. Bei beiden handelt es sich um webbasierte Verwaltungsprogramme im Rahmen der E-Gesetzgebung (egg). In beiden kommt Java als (Haupt-)Programmiersprache, sowie Spring(Boot) als Grundlegendes Framework zum Einsatz.

Bei eDok handelt es sich um ein älteres Projekt. Hier werden hauptsächlich Bugfixes, Security Patches und kleinere Anpassungen vorgenommen. eDok umfasst die zwei Systeme Verteilung und Einbringung. Bei eDok Verteilung geht es um die elektronische Verteilung von sog. Drucksachen. Dabei handelt es sich um „eine Vorlage des [Deutschen Bundestages](https://de.wikipedia.org/wiki/Deutscher_Bundestag) zur Verteilung an dessen [Mitglieder](https://de.wikipedia.org/wiki/Mitglied_des_Bundestages), an die [Mitglieder des Bundesrates](https://de.wikipedia.org/wiki/Bundesrat_(Deutschland)) und an die [Ministerien](https://de.wikipedia.org/wiki/Bundesministerium_(Deutschland))“. WIKIPEDIA https://de.wikipedia.org/wiki/Bundestagsdrucksache Dagegen dient eDok Einbringung der elektronischen Einbringung von Gesetzinitiativen.

PKP wird aktiv entwickelt und umfasst eine Vielzahl von einzelnen Modulen, die von den Abgeordneten? des Bundestages verwendet werden. Unter anderem können Vorhaben eingebracht und an die entsprechenden Stellen weitergeleitet werden. Des Weiteren können Fragen/Anfragen in elektronischer Form gestellt und beantwortet werden. Der Politische Kalender ermöglicht eine Übersicht über Tagesordnungspunkte wie z.B. Sitzungen, während es die eMappe den Abgeordneten ermöglicht ihre Unterlagen in elektronischer Form mitzuführen und so während einer Sitzung Zugriff auf diese zu haben ohne große Aktenordner mitzuschleppen. Dazu kommt eine umfassende Verwaltung in der System Admins die Fraktionen, Berechtigungen und zum Teil auch einzelne Benutzer anpassen können.

Eng mit PKP verbunden ist das Identity and Access Managemant (IAM), in welches die Benutzerverwaltung ausgelagert wurde um es zu ermöglichen sich mit einem einzigen Account nicht nur bei PKP sondern bei allen an das IAM angebundene Systeme anzumelden.

# Der Betrieb

SEITENBAU wurde 1996 von Stefan Eichenhofer, Jan Bauer und Florian Leinberger gegründet. Mittlerweile zählt das Unternehmen über 200 Mitarbeiter, die hauptsächliche am Hauptsitz in Konstanz arbeiten. Eine kleinere Niederlassung gibt es außerdem in Köln. SEITENBAU entwickeln maßgeschneiderte Software für seine Kunden, zu denen vor allem einige Bundesämter, aber auch andere Unternehmen, wie z.B. die Telekom gehören. https://www.seitenbau.com/kunden

Im Unternehmen gibt es nur flache Hierarchien, die einzelnen Teams sind zum Großteil selbstverantwortlich für ihre Projekte. Auch in den Teams selbst gibt es keinen richtigen „Chef“, Entscheidungen werden nach gemeinsamer Beratung zusammen getroffen.

Ich wurde dem Team zugeteilt, dass sich um die Projekte eDok und PKP kümmert. Bei meinen Eintritt bestand das Team aus vier Entwicklern, sowie einem Scrum Master, einem Technical Consultant und der Projektleiterin. Drei Monate nach Beginn meines PSS wurde ein weiterer Entwickler eingestellt, wodurch das Team auf neun Leute wuchs.

Das Team kümmert sich um alles rund um die beiden o.g. Projekte. In der Entwicklung beinhaltet das Front- und Backend, das Testen und Ausliefern neuer Versionen sowie technischen Support rund um die Anwendung. Hinzu kommem Verwaltungsaufgaben wie die Ausschreibung um das Projekt nach Vertragsende weiter zu führen, oder die Rechnungsstellung.

# Arbeitsweise

Unser Team arbeitet nach dem Scrum-Prinzip. Dazu gehören regelmäßige (Online) Meetings mit den Kunden um den Projektfortschritt zu präsentieren sowie Feedback einzuholen. Zudem rekapitulieren wir die zweiwöchigen Sprints am Ende in einer Retrospektive. Im Rahmen des sog. „Refinements“ wird der Arbeitsaufwand für neue, vom Kunden gewünschte, Features abgeschätzt. Diese Umsetzungsaufgaben mit allen weiteren Aufwänden wie z.B dem Testen der neuen Funktionen werden als Story bezeichnet.

Im Sprint selbst werden diese Stories dann nochmal in einzelne Tickets unterteilt, d.h. kleinere Schritte, die im Idealfall parallel von mehreren Entwicklern bearbeitet werden können. Nach diesem „Task Breakdown“ kann sich jeder selbstständig als Bearbeiter eines Tickets eintragen und mit dessen Umsetzung beginnen. Dies geschieht im Management Programm Jira, in welchem unter anderem die Stories des aktuellen Sprints einsehbar sind. So bekommt man einen Überblick darüber wie weit einzelne Stories umgesetzt sind und wer gerade an was arbeitet. Dabei halten wir ständig Rücksprache mit den anderen Teammitgliedern. Das hilft zum einen dabei einen einheitlichen verständlichen Stil einzuhalten. Zum anderen gibt es aber auch neuen Mitgliedern die Möglichkeit Fragen zu stellen und erleichtert so die Einarbeitung. Gerade am Anfang, als auch bei neuen Themengebieten, wird auch Pair-Programming eingesetzt. Dass funktioniert sowohl vor Ort im Büro, als auch remote mittels Bildschirmübertragung sehr gut.

Zur Versionsverwaltung kommt Git in Verbindung mit Bitbucket als remote Repository zum Einsatz. Letzteres befähigt uns in Verbindung mit Jira z.B. einheitliche Git-Branches für einzelne Stories zu erstellen. Alle 3 Systeme zusammen spielen außerdem im Review eine große Rolle. Dabei wird jedes abgeschlossene Ticket von einem anderen Teammitglied „überprüft“. Das beinhaltet vor allem ein Code-Review, sowie, wenn angemessen, einen kurzen Test, ob die Anwendung weiterhin compiliert und startet. Das stellt zum einen eine hohe Code Qualität sicher, zum anderen nimmt es Neulingen die Angst etwas falsch zu machen.

# Bearbeitete Aufgaben / Projekte

## 4.1 Aufgabe 1

Generierung von Barrierefreien PDF-Dateien

### Problembeschreibung

Die Anwendung eDok generiert drei verschiedenen Typen von PDF Dateien. Das wären zum einen einfache Unterschriftenblätter. Dazu kommen sog. Mantelbögen, welche Informationen zur Verfügbarkeit einer Drucksache enthalten. Zuletzt gibt es eine Drucksachenmappe, welche an sich nur ein Deckblatt mit Inhaltsverzeichnis darstellt, welchem die eigentlichen Drucksachen angefügt werden. Alle drei Dateien sind für Sehbehinderte Menschen teilweise nicht, oder nur schwer lesbar, da sie nicht “getagged“ sind.

Des Weiteren nutzt eDok zur PDF-Generierung die Bibliothek Apache FOP in der recht alten Version 1.0. Diese soll deshalb auf die aktuelle Version aktualisiert werden.

### Aufgabenstellung und Zielsetzung

Ziel dieser Story ist es, dass die von eDok generierten PDF-Dateien im PDF Accessibility Checker **LINK** keine Fehler enthalten. Außerdem soll die Bibliothek Apache FOP, die zur PDF-Generierung verwendet wird, auf die aktuelle Version aktualisiert werden.

### Theoretische Grundlagen

Als Build-Tool wird in eDok Maven eingesetzt. Das Updaten der Bibliothek Apache FOP geschieht somit durch das Anpassen der Versionsnummer in der pom.xml. Diese Datei enthält alle für den Bau des Projektes notwendigen Dependencies. Maven läd diese in das lokale Repository und nimmt sie beim Build in das WAR-Archiv auf. **(WAR genauer erklären?)** Im Anschluss muss geprüft werden ob, und falls ja welche, Codeanpassungen vorgenommen werden müssen. Erst wenn die Anwendung wieder startet und identische PDF-Dateien erstellt werden wie zuvor wird damit begonnen die eigentliche PDF-generierung anzupassen.

Die Barrierefreiheit wird mit dem PDF Accessibility Checker (PAC) überprüft. Dieser zeigt Fehler an, die behoben werden müssen, damit eine PDF als barrierefrei gilt. Ein Beispiel für einen Solchen Bericht ist in Abbildung 1 zu sehen.Wichtig für die Barrierefreiheit sind vor allem ein Strukturbaum an dem sich Screenreader orientieren können sowie Alternativtexte für Abbildungen. Allerdings gilt es alle der angezeigten Fehler zu beheben.

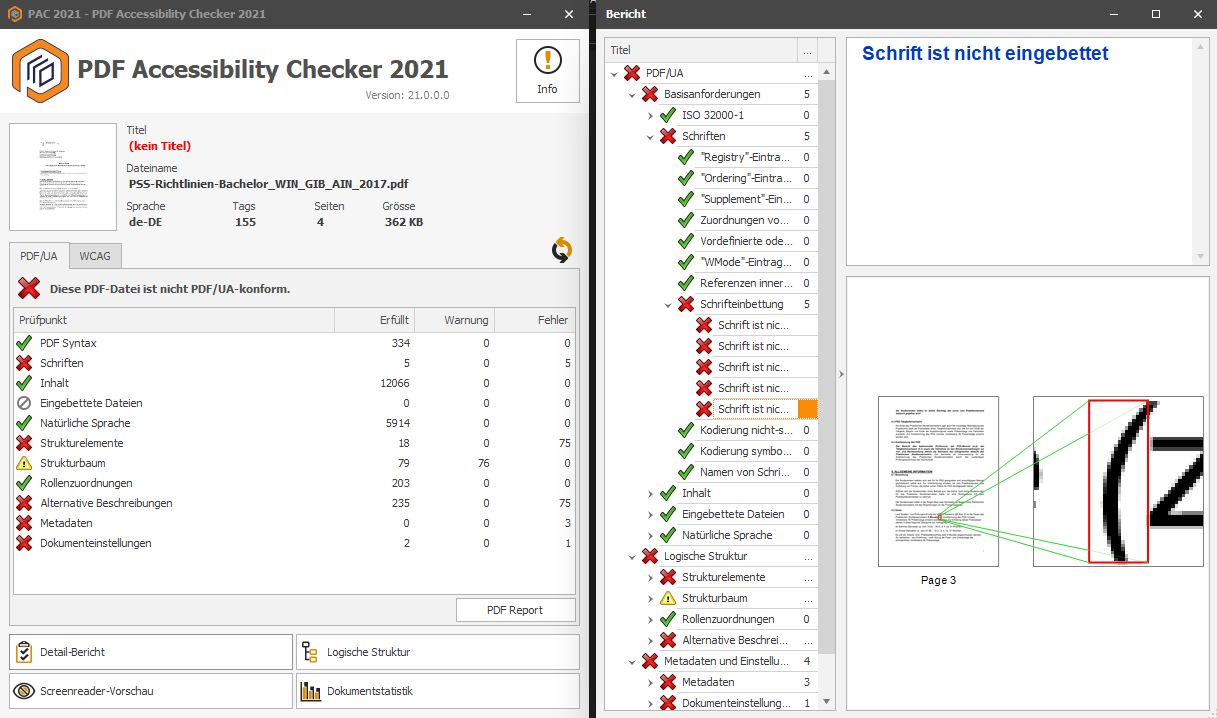


Abbildung 1 Beispielhafter Fehlerbericht von PAC 2021

Für die PDF-Generierung benötigt Apache FOP ein XSL-Stylesheet. Dies ist ein HTML-Ähnliches Template, welches die Struktur des Ausgabedokuments festlegt. Für Dynamische Daten gibt es Platzhalter. Bei der Erzeugung der PDF wird neben dem Stylesheet auch noch eine XML Datei eingelesen, welche die Daten enthält mit denen die Platzhalter ersetzt werden.

### Realisierung

Das Anheben der Version in der pom.xml führt wie erwartet zu Compile-Fehlern, hervorgerufen durch Syntaxänderungen in Apache FOP. Ein Blick in die Dokumentation und die Changelogs gibt Aufschluss darüber welche Anpassungen vorgenommen werden müssen. Im Zuge dieser Anpassung wird außerdem das Attribut Accessibility des „FOUserAgent“ auf true gesetzt. Der UserAgent ist in der Version 1.0 optional, wird nun aber benötigt um die PDF-Erstellung zu konfigurieren. Das Setzen dieses Attributs behebt einige Fehler die PAC3 anzeigt. Die übrigen Fehler müssen manuell behoben werden. Das geschieht durch Anpassung der Konfigurations-XML-Datei, sowie der XSL-Stylesheets.

Das Setzen ein eines Alternativtitels für eine Grafik geschieht im Stylesheet und beinhaltet lediglich das Setzen eines Attributs im entsprechenden Element, siehe [Abbildung 4.1](#AbblidungFoxAlt). Dadurch wird ein versteckter Text hinzugefügt, den Screenreader vorlesen können.

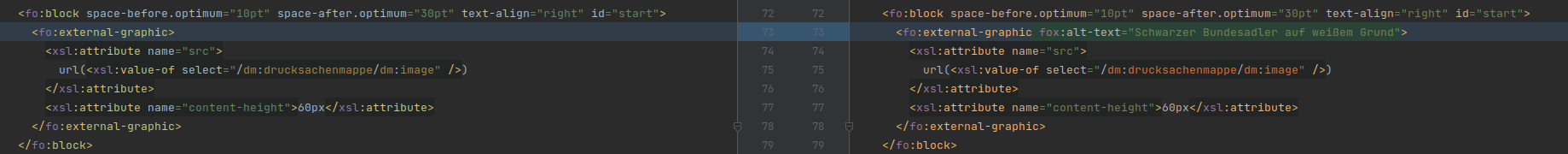


Abbildung 2 Das fox:alt Attribut in einem Grafik Element des Stylesheets

PAC3 bemängelt außerdem das Fehlen einer PDF/UA (Universal Accessibility) Kennzeichnung. Dieses wird hinzugefügt indem man in der Konfigurationsdatei das Element „<pdf-uy-mode>PDF/UA-1</pdf-ua-mode>“ hinzufügt.

Da bei normalen PDFs der Dateiname als Titel angezeigt wird muss zudem der Eintrag „DisplayDocTitel“ gesetzt werden **(**<https://www.barrierefreies-webdesign.de/knowhow/pdf-ua/07-001-pac-3-fehler-mit-adobe-acrobat-reparieren.html>). Bei dieser Gelegenheit wird auch noch die Sprache des Dokuments festgelegt. In diesem Fall werden die Entsprechenden Elemente im Stylesheet eingefügt.

Zur Bestehung der Prüfung müssen außerdem alle verwendeten Schriftarten ins Dokument eingebettet werden, sodass diese immer korrekt angezeigt werden, auch wenn die entsprechende Schriftart auf dem Zielrechner selbst nicht vorhanden ist. In der Konfigurationsdatei kann dafür ein „Fonts“ Block eingefügt werden. **ABBILDUNG** Die entsprechenden .ttf Dateien der Schriftarten müssen in der Dateistruktur des Projekts vorhanden sein und werden über eine relative URL aufgelöst.

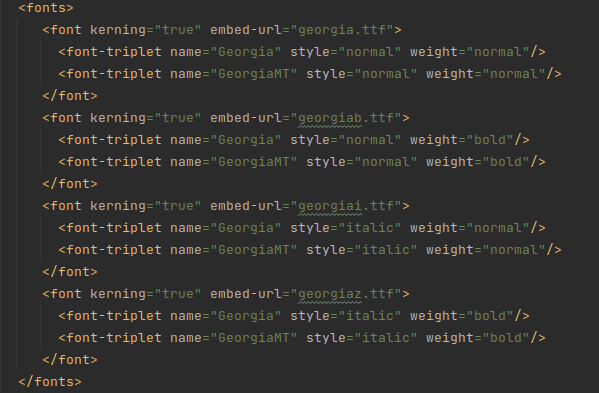


Abbildung 3 Ausschnitt der Fonts-Sektion der Apache FOP Konfigurationsdatei

Tabellen, wie etwa im Mantelbogen, können für Screenreader Nutzer schwer verständlich sein, wenn nicht klar ist ob eine Zelle eine Überschrift oder Daten enthält. Dazu kann eine Zelle als TH (Table Header) oder TD (Tabel Data) makiert werden. In Abbildung 4 sieht man die „Normalansicht“ der Tabelle, Abbildung 3 zeigt die Tabelle in der Screenreader Ansicht. Die Fettgedruckten Einträge zeigen, dass es sich um Überschriften handelt.

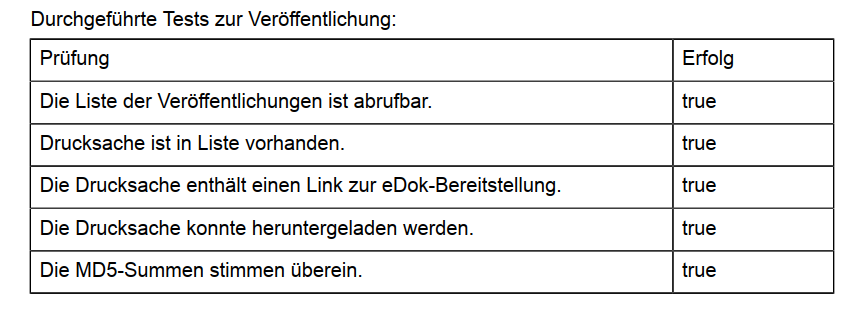


Abbildung 4 Die Tabelle der Mantelbogen PDF

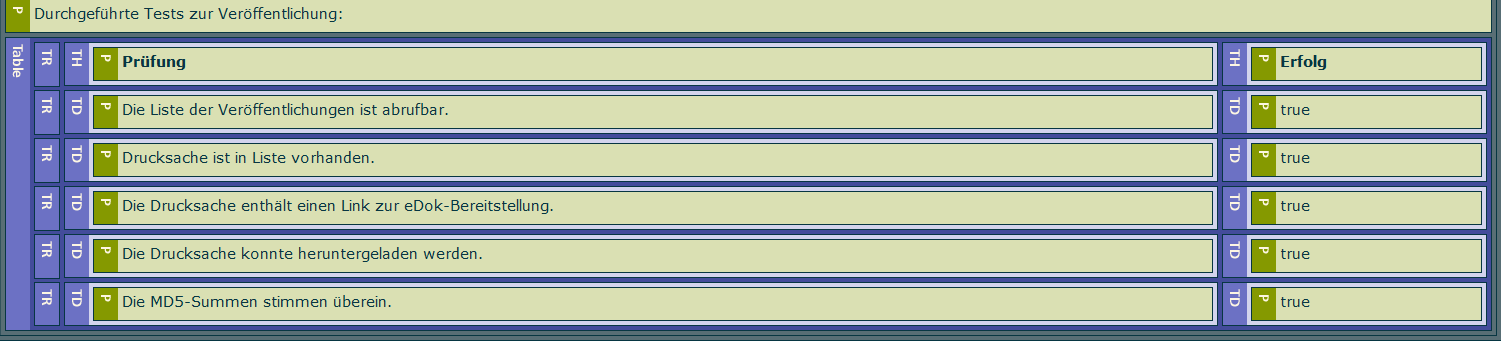


Abbildung 5 Die Tabelle der Mantelbogen PDF als Screenreader Ansicht in PAC3

### Nachweis/Bewertung der Funktionsfähigkeit

Neben den Standard Kriterien, wie der erfolgreiche Abschluss aller Unit-/Integration- und Systemtests sollen alle generierten PDFs in PAC3 keine Fehler erzeugen. Der Status der Tests kann in Jenkins eingesehen werden. Im Idealfall sieht man dort eine „grüne“, also erfolgreich durchgelaufene Pipeline. Außerdem werden manuelle Tests durchgeführt, bei denen die Wichtigsten Funktionen noch einmal durch ein Teammitglied kontrolliert werden. Nachdem eine Auswahl an erzeugbaren PDFs in PAC3 keine Fehler mehr aufweist werden diese zusätzlich noch Stichprobenartig durch einen Barrierefreiheitsspezialisten von SEITENBAU mit einem Screenreader überprüft.

### Zusammenfassung

Diese erste Aufgabe hat mir besonders dabei geholfen das Build Tool Maven besser zu verstehen mit dem ich hier zum ersten Mal in Kontakt kam. Außerdem war es Interessant zu sehen, wie man mit Java, unter Zuhilfenahme einer 3rd Party Bibliothek, PDF Dateien erstellen kann.

## Aufgabe 2

Dockerisierung von eDok

### Aufgabenbeschreibung und Zielsetzung

Die Anwendung eDok soll dockerisiert werden, d.h. alle Module, bzw. deren Komponenten sollen in Docker Containern ausgeführt werden. Auf die Datenbank trifft das schon zu, die restlichen Komponenten laufen aber als Webapps in einem Tomcat Webserver, welcher direkt auf dem Zielsystem installiert ist. Die Komponenten sollen nach der Umstellung jeweils in einem eigenen Dockercontainer ausgeführt werden, in welchen wieder ein Tomcat Webserver läuft. Des Weiteren sollen sich Images der einzelnen Container über einen einzelnen Befehl mittels Docker Compose gestartet werden können. Die automatisierte Test-Pipeline soll wie zuvor funktionieren.

### Theoretische Grundlagen

Um einen Container zu starten wird ein Image benötigt, welches die von eDok verwendete JVM Adoptium, sowie einen Tomcat Webserver enthält. Mit diesem Image können dann beliebig viele Container gestartet werden. Das fertige Image wird in die Docker Registry gepushed. Aus dieser wird dann das Image „gepulled“ wenn ein Container gestartet werden soll.

Docker Compose ist ein Tool, das darauf ausgelegt ist eine Anwendung zu starten die aus mehreren Docker Containern besteht. In einer .yaml-Datei wird dazu für jeden Container ein Service definiert, der neben dem zu startenden Image alle wichtigen Einstellungen enthält.

Jenkins ist eine webbasierte CI Software, die bei eDok zum Starten der Test-Pipeline eingesetzt wird. In dem sog. Jenkinsfile wird beschrieben wie das Projekt auf den Test VMs gebaut und wie die einzelnen Pipeline Stages ausgeführt werden sollen.

Da bei jedem Git-Push eine Test Pipeline läuft, muss außerdem der automatisierte Startup der Anwendung auf den Test VMs angepasst werden. Dazu wird das Automatisierungswerkzeug Ansible verwendet. In dem sog. Playbook kann dort beschrieben werden wie ein System aussehen muss, damit z.B. eine Webanwendung darauf gestartet werden kann. Ansible muss lediglich auf der verwaltenden Maschine installiert sein und führt die gewünschten Anweisungen dann per ssh auf dem Zielsystem aus. So werden automatisiert Verzeichnisse erstellt in denen Daten gespeichert werden sollen, Konfigurationsdateien an die vorgesehenen Stellen kopiert oder benötigte Prozesse durch z.B. Shell Befehle gestartet. Auch das Aufräumen von eventuell angefallen Testdaten kann durch Ansible vorgenommen werden. So wird sichergestellt, dass jeder Testdurchlauf auf derselben Grundlage erfolgt und reproduzierbar ist.

### Realisierung

Für das Bauen der Images verwenden wir Paketo Buildpacks. In einem Shell Skript wird Paketo ausgeführt. Dabei wird angegeben, wo sich das von Maven erstellte WAR-Archiv befindet und welche Buildpacks verwendet werden sollen, so dass im fertigen Image Tomcat und die JVM Adoptium vorhanden sind. Außerdem werden einige Konfigurationsdateien vom Build ausgeschlossen die nicht für den Produktiv-oder Test-Betrieb notwendig sind. Ist das Image erfolgreich erstellt soll es zunächst lokal eingebunden werden. Dafür wird die docker-compose.yaml, die bisher nur die Datenbank als Container startet, um weitere Services erweitert. Hier wird unter anderem das Image angegeben, aus dem der Container erstellt werden soll. Abbildung 6 zeigt den Anfang der Konfiguration des Containers edok-informationsdienst. Die Platzhalter in der Form „${}“ werden mit Daten aus der .env Datei ersetzt, wobei „.env“ der Name ist, nachdem Docker Compose default mäßig sucht um Platzhalter zu ersetzen. In dieser Datei stehen pro Zeile ein Platzhalter und der Wert durch den dieser ersetzt werden soll. Der Zwischenschritt über die Platzhalter ist nicht notwendig, erleichtert aber das ändern der Werte, da diese alle an einem Platz stehen, statt überall über die Docker-Compose Datei verteilt zu sein.

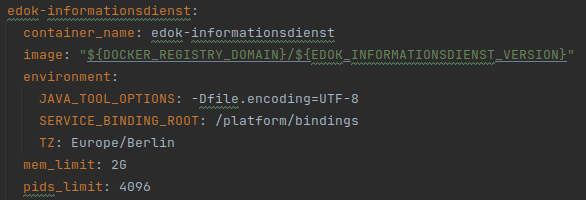


Abbildung 6 Ausschnitt aus der Datei docker-compose-yaml

Ein Image ist schreibgeschützt, ein Container kann darin also keine Daten speichern. Stoppt und löscht man einen laufenden Container so gehen ins Dateisystem geschriebene Daten wie z.B. Log Files verloren. Um das zu verhindern werden sog. Volumes erstellt. Diese „spiegeln“ das „virtuelle Verzeichnis„ eines laufenden Containers auf das des Hostsystems. Wird der Container gelöscht verbleiben die Dateien der Volumes auf dem Host-System. Dazu wird unter dem Punkt volumes definiert welches „echte“ Verzeichnis auf dem Host System welchem „virtuellen“ Verzeichnis im Container entsprechen soll. **ABBIDLUNG** Für die Container Verzeichnisse werden absolute Pfade angegeben, während die „echten“ Verzeichnisse relativ zur docker-compose.yaml angegeben werden. Dieser Teil ist recht fehleranfällig, da Typos in den Pfaden dazu führen, dass Leere Verzeichnisse erstellt werden, und kein Error geworfen wird. Daher ist es wichtig sich zu vergewissern, dass alle Volumes vorhanden sind, z.B. durch IDE Funktionen mit denen man in das Dateisystem eines Containers schauen kann.

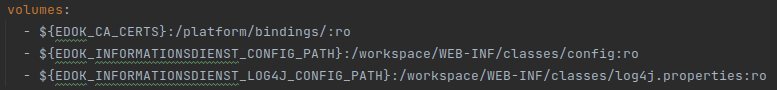


Abbildung 7 Ausschnitt aus docker-compose.yaml, Konfigurations der Volumes von edok-informationsdienst

Einige Komponenten greifen per http auf andere zu. Da bisher alle Komponenten auf demselben System liefen funktionierte das einfach über „<http://localhost>:port“. Eine Komponente die in einem Docker Container läuft ist darüber nun nicht mehr direkt erreichbar. Z.B. ist der NGINX Proxy der bisher auf Port 80 lief nun nur noch innerhalb des Containers darüber zu erreichen. Um das zu ändern können ähnlich wie zuvor die Volumes nun auch Ports gespiegelt werden. In Abbildung 8 sieht man, dass für den Proxy der Container Port 80 mit dem Port 80 des Host Systems verknüpft wird. Damit der Proxy nun aber die Weiterleitung vornehmen kann muss er mit den Containern der übrigen Komponenten kommunizieren. Dazu kann in docker-compose.yaml ein Netzwerk konfiguriert werden, dem dann alle Container hinzugefügt werden. Die Docker Engine sorgt dann dafür, dass aus Containersicht alle Komponenten weiterhin über localhost ansprechbar sind.

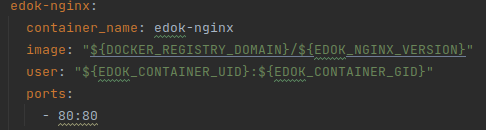


Abbildung 8 Ausschnitt aus docker-compose.yaml, Port Konfiguration des Proxys

Nachdem der Container lokal startet und die Anwendung wie gewohnt funktioniert ist der nächste Schritt dafür zu sorgen, dass auch auf den VMs ein Container gestartet wird. Der Bau des Images erfolgt weiterhin über das Shell Skript, allerdings wird dieses nun in der Build-Stage der Jenkins Pipeline aufgerufen. Außerdem wird das Image in die Firmeneigene Docker Registry gepusht, damit es von allen VMs aus gepulled werden kann. Dafür werden die entsprechenden Shell- und Docker Befehle in den Jenkinsfile geschrieben.

Nun muss eine VM für das Ausführen der Tests vorbereitet werden. Dafür wird in der entsprechenden Stage das Playbook aufgerufen. Dieses muss jetzt noch um die Schritte erweitert werden, die notwendig sind um die Container zu starten. Diese Schritte werden in die Datei docker.yml ausgelagert, welche über eine include Anweisung in das Playbook eingebunden werden kann. Dadurch lassen sich die Anweisungen z.B. Komponentenweise trennen, wodurch das Playbook übersichtlich bleibt. In docker.yml wird unter anderem festgelegt, wo Dateien wie docker-compose.yaml oder .env zu finden sind und in welches Verzeichnis sie auf der Test VM kopiert werden sollen. Alte Versionen dieser Dateien werden dabei überschrieben. Container die eventuell noch von einem früheren Testdurchlauf gestartet sind werden gelöscht. Anschließend werden über „docker-compose up“ wieder alle Container gestartet. **ABBILDUNG** Da in der Build stage die Images neu gebaut und in die Registry gepusht wurden werden nun auch diese neuen Images zum Starten der Container verwendet.

### Nachweis/Bewertung der Funktionsfähigkeit

In diesem Fall ist die Funktionsfähigkeit sehr leicht am Status der Testpipeline zu überprüfen. Sollte eine Komponente nicht starten oder keine Verbindung zu ihr möglich sein würden einige Tests einen Fehler werfen und die Pipeline wäre „rot“. Zudem liefern die Monitoring Seiten einiger Komponenten Informationen dazu, welche Verbindungen bereit sind.

### Zusammenfassung

Nicht nur die Arbeit mit Docker, auch das Thema rund um die Infrastruktur, bzw. wie man die neuen Container in diese integriert war sehr Interessant. Zudem

## Aufgabe 3

Anpassung der Startseite von PKP

### Aufgabenbeschreibung und Zielsetzung

Der Kunde von PKP hat sich ein neues Layout der Startseite gewünscht. Abbildung 9 und 10 zeigen das alte bzw. das Neue Layout. Neben Optischen Anpassungen sollen den einzelnen Kacheln, die jeweils ein Modul repräsentieren, sog. „Quicklinks“ hinzugefügt werden. Das sind Verknüpfungen auf Funktionen eines Moduls die direkt von der Startseite aus angesteuert werden können.

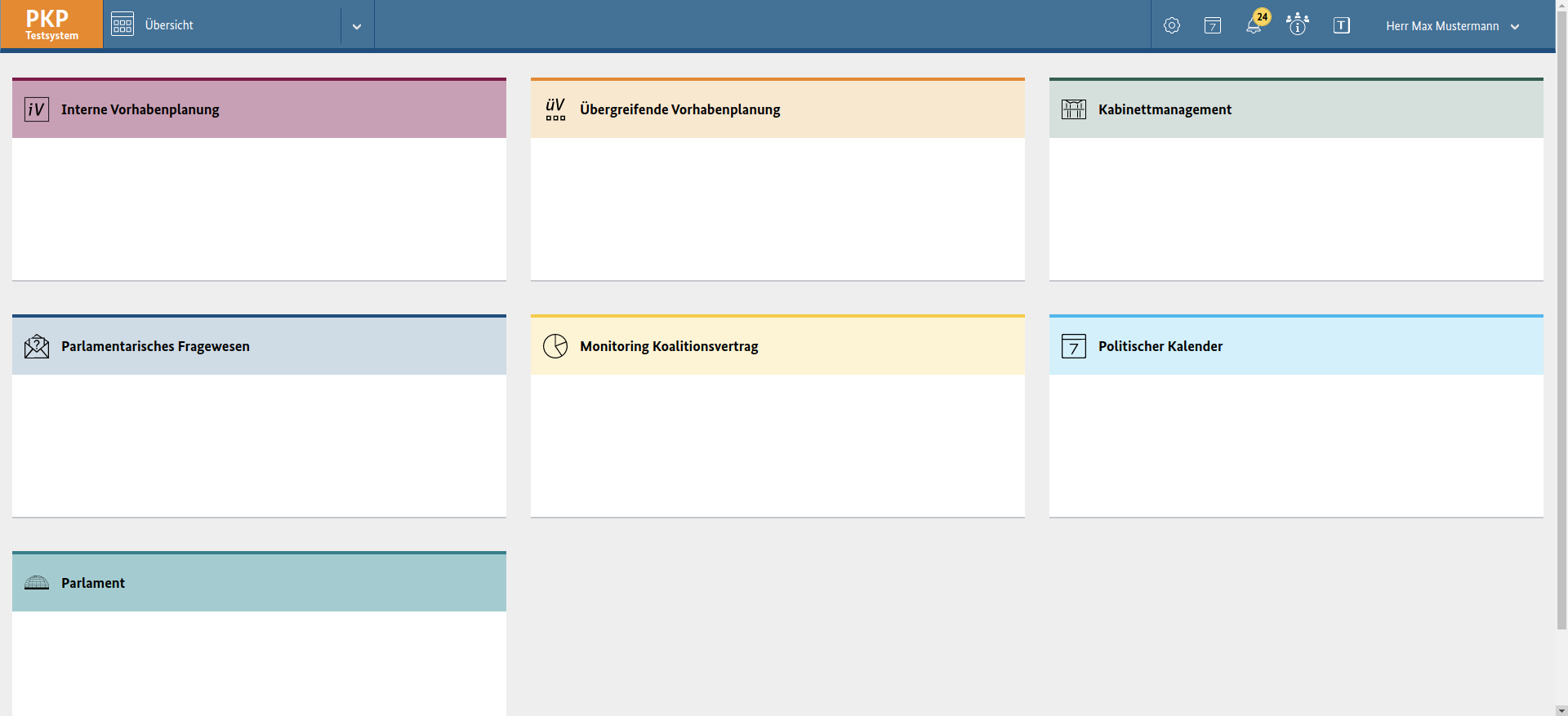


Abbildung 9 Das Alte Layout der PKP Startseite

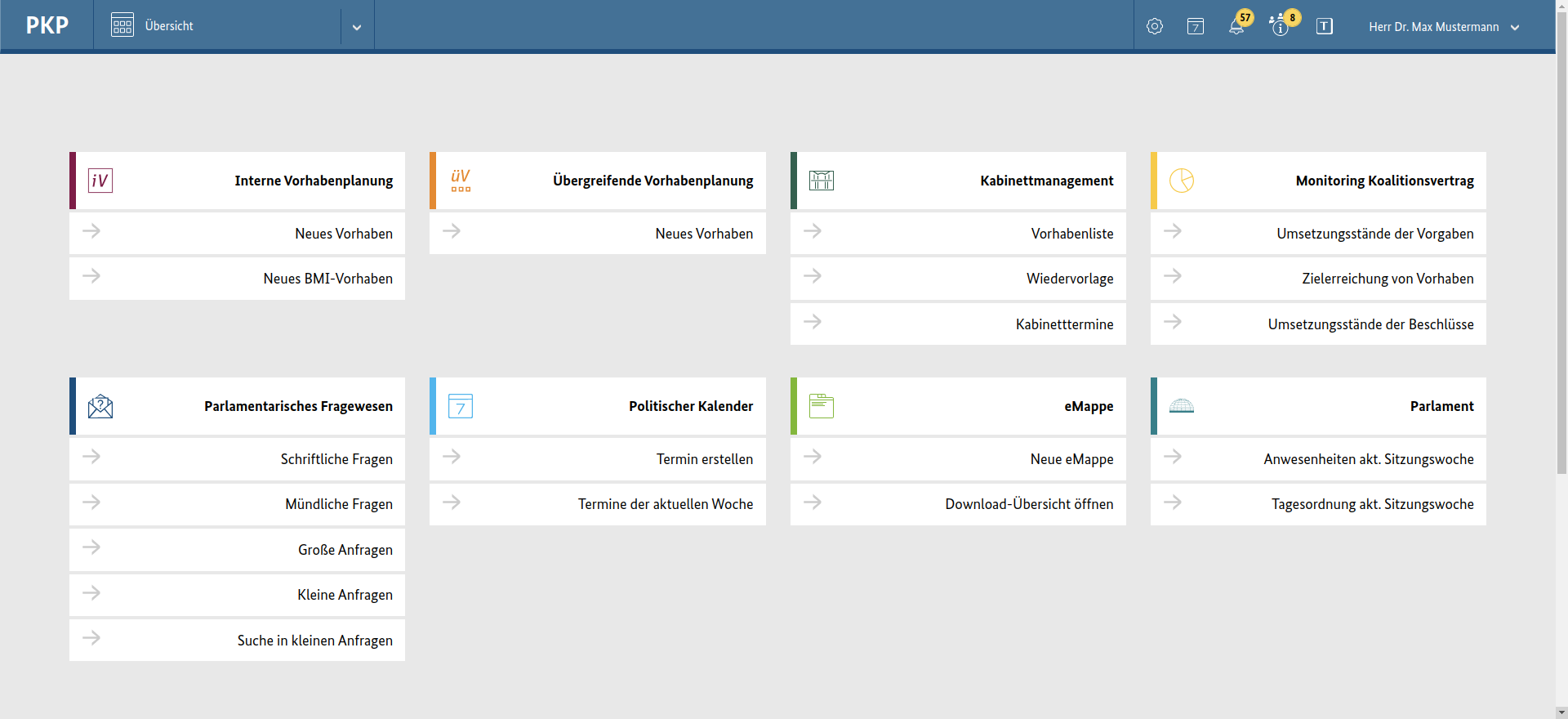


Abbildung 10 Das neue Layout der PKP Startseite

### Theoretische Grundlage

Für das Frontend kommt die Template Engine Thymeleaf zum Einsatz. In Verbindung mit dem Framework Spring erleichtert es die Darstellung der angeforderten Daten. Im Java Code wird ein Model-Objekt mit den Entsprechenden Daten befüllt die beim Aufruf einer Bestimmten URL benötigt werden. In den HTML-Dateien werden Variablen verwendet, die die Thymeleaf Engine dann durch die Daten aus dem Model-Objekt ersetzt.

### Realisierung

Um die Anzahl der Probleme einzugrenzen ist das erste Ziel das Layout so weit anzupassen, dass es der Desktop Version der Vorlage optisch ähnelt. Zunächst wird dafür das CSS angepasst um die Farbe der Icons zu ändern, den farbigen Rand zu verschieben und die Hintergrundfarbe zu ändern. Das geschieht indirekt über die Anpassung von .less Dateien. Less ist eine Stylesheet-Sprache die ähnlich zu benutzen ist wie css und auch in diese kompiliert wird.

Als nächstes wird mit Thymeleaf ein Quicklink-„Fragment„ definiert, zu sehen in Abbildung 11. Die URL des Links, sowie die Bezeichnung wird dabei als Parameter definiert. Dieses Fragment kann dann in den Kacheln mehrfach eingefügt werden. Da noch keine Links vorhanden sind werden diese Fragmente zunächst mit Dummy-Daten in die Kacheln eingesetzt.

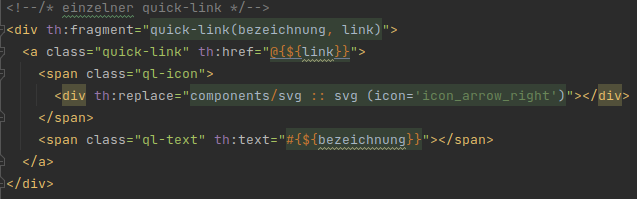


Abbildung 11 Thymeleaf Fragment, welches einen Quicklink darstellt

Wie in Abbildung 10 zu sehen ist soll außerdem ein kleiner Pfeil anzeigen, dass es sich um einen Link handelt. Dafür wird eine Scalable Vector Graphic (SVG) verwendet. In einem XML-Dokument wird hierbei das Icon definiert. Durch die Kommandos M (move) und L (line) bewegt sich ein virtueller Stift auf einem Koordinatenfeld oder zeichnet Linien, die sich zu einem Polygon verbinden lassen. Abbildung 12 zeigt, wie der Pfad in der XML Datei bzw. das fertige Icon aussieht.

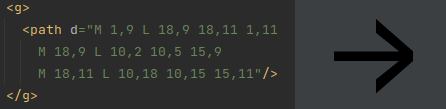


Abbildung 12 Ausschnitt aus der SVG Datei (Links) mit den Kommandos die zu dem Pfeil Icon (Rechts) führen

Nun müssen im Backend die Daten gesammelt werden die für die Quicklinks benötigt werden. Dafür wird dem Model-Objekt eine HashMap hinzugefügt, die als Key die Kurzbezeichnung der einzelnen Module enthält und als Wert eine Liste die die eigentlichen Links und deren Bezeichnungen enthält. Im HTML Dokument besteht dann die Möglichkeit mittels Thymeleaf Variablen auf diese zuzugreifen. Beim Befüllen der HashMap wird zudem der AccessContext des angemeldeten Benutzers als Parameter mitgegeben. Dieses Objekt enthält unter anderem die Rollen die dem Nutzer zugeordnet sind. Diese können dann beim Hinzufügen eines jeden Quicklinks abgeglichen werden. Links auf Funktionen, auf die der Nutzer keine Rechte hat, werden gar nicht erst in das Model-Objekt eingefügt und können auch nicht angezeigt werden.

Damit die Anzeigenamen der Links leicht geändert werden können werden diese in eine „message.properties“ Datei ausgelagert. Diese enthält Schlüssel-Wert Paare, die mittels Spring-Funktionen im Front- und Backend ausgelesen werden können. Durch Verwendung des Schlüssels lässt sich ein Anzeigetext in allen möglichen View Schichten wiederverwenden. Auch kann er an einer zentralen Stelle geändert werden.

Um nun in jedem Modul die richtigen Links anzuzeigen wird das Thymeleaf Attribut „th:each“ verwendet welches es ermöglicht in der HTML Datei über Java-Collections zu iterieren die sich im Model-Objekt befinden. Durch den Key des jeweiligen Moduls kann auf die Quicklinks zugegriffen werden.

Da sich ein Teil der Grafischen Oberfläche verändert hat müssen bei dieser Story bestehende Visual Regression Tests (VRT) angepasst werden. Bei diesen werden, nach dem Einspielen von Testdaten, verschiedene URLs aufgerufen. Anschließend wird ein Screenshot erstellt und dieser dann mit einer Vorlage abgeglichen. Dafür wird bei PKP BackstopJS eingesetzt, welches das automatische Erstellen und Vergleichen von Screenshots ermöglicht. Am Ende des Durchlaufs wird automatisch ein Bericht erstellt, der genau Zeigt, wo sich der Test Screenshot und die Vorlage unterscheiden. Da der VR Test für die Startseite schon existiert muss nun lediglich die Vorlage aktualisiert werden. Dafür wird das Szenario welches die Startseite beinhaltet mit der „reference“ Option aufgerufen. BackstopJS macht nun einen Screenshot und ersetzt damit die alte Vorlage.

### Nachweis/Bewertung der Funktionsfähigkeit

Der Erfolg dieser Story wird zum einen wieder am Status der Test Pipeline festgemacht. Zum anderen wird die Startseite aber auch noch manuell getestet. Dabei wird auf Anzeigefehler geachtet und die Links werden ausprobiert. Außerdem loggt man sich mit verschiedenen Berechtigungen ein um zu sehen, ob alle Elemente ausgeblendet werden die nicht zu sehen sein sollen.

# Persönliche Eindrücke und Erfahrungen

Hinweise:   
Fassen Sie in diesem Abschnitt die für Sie persönlich wichtigsten Erfahrungen, die sie während des Praxissemesters gesammelt haben zu zusammen, z.B.

* Welches waren die wertvollsten Erfahrungen, die Sie im Unternehmen gemacht haben?
* Was war besonders gut und lobenswert während Ihrer Zeit in diesem Betrieb (Betreuung, Thema, Betriebsklima etc.)? Was war nicht gut?
* Wo sehen Sie Verbesserungspotential für die Betreuung bzw. den Betrieb im Umgang mit Praktikanten und was sind Ihre Vorschläge?
* Was möchten Sie nachfolgende Praktikanten weitergeben? Wie ver­lief die Bewerbung? War der Betrieb behilflich bei der Zimmersuche?

Literaturquellen

**Im aktuellen Dokument sind keine Quellen vorhanden.**

Bearbeitungshinweise:   
**Bibliografische Angaben**

Die Darstellung der im Text erwähnten Quellen erfolgt im Literaturverzeichnis in alphabetischer Ordnung. Folgende Arten von Quellen sind zu unterscheiden:

1. Zeitschriftenaufsätze

BEITRAGSVERFASSERNAME, Vorname: „Beitragstitel“. In: *Zeitschriftentitel,* Bandzählung (Erscheinungsjahr), Heftnummer, Anfangsseite–Endseite

1. Beiträge in Sammelwerken

BEITRAGSVERFASSERNAME, Vorname: „Beitragstitel“. In: Herausgebername, Vorname (Hg.): *Buchtitel.* Erscheinungsort: Verlag, Erscheinungsjahr, Anfangs­seite–Endseite

1. Bücher

VERFASSERNAME, Vorname: *Buchtitel.* Erscheinungsort: Verlag, Erschei­nungsjahr

1. Internetseiten

BEITRAGSVERFASSERNAME, Vorname (falls nicht vorhanden, Domain als Verfasser): „Beitragstitel“. Erscheinungsjahr. In: Name der Webseite, URL: www… (ungekürzt), Abrufdatum

Kurzverweise im Text

Auf jedes aufgeführte Werk wird im Text Bezug genommen. Die Verfasser kön­nen im Text namentlich genannt werden, das Werk muss mit Kurzverweis ange­geben werden (als Fußnote):

1 Verfassername Erscheinungsjahr, zitierte Seite

Beispiel

Eintrag im Literaturverzeichnis:

Hans Martens: *Recyclingtechnik. Fachbuch für Lehre und Praxis,* Spektrum Akademi­scher Verlag, Heidelberg 2011

Originalauszug aus dem Buch:

Besonders günstige Voraussetzungen für das Recycling sind bei Produktions­abfällen gegeben.

Wörtliches Zitat in Ihrem Text:

Laut Martens sind „besonders günstige Voraussetzungen für das Recycling […] bei Produktionsabfällen gegeben“.2

Fußnote bei wörtlichem Zitat:

2 Martens 2011, S. 4

Sinngemäßes Zitat in Ihrem Text:

Laut Martens eignen sich Produktionsabfälle besonders gut für die Wieder­verwertung.3

Fußnote bei sinngemäßem Zitat:

3 vgl. Martens 2011, S. 4

Anhang

Hinweise:

Der Anhang bildet den letzten Teil eines Praxissemester-Berichts und enthält alle detaillierten In­formationen, die das Lesen des Berichtes beschwerlich machen würden, z. B. (Detail-)Berechnungen, Datenblätter, Detailauszüge und Bilder bei Modellierun­gen, Rohdatenkurven, Lösungsvarianten etc.

Auf den Anhang wird im Text der Arbeit immer dann verwiesen, wenn es für den Leser von Interesse sein könnte, weitere Informationen heranzuziehen („siehe Anhang B“).

Der Anhang kann aus mehreren Unterkapiteln bestehen, die auch als solche zu formatieren sind:

Anhang A (Titel)

Text

Text

Anhang B (Titel)

Text

Text

Anhang C (Titel)

Text

Text

Weitere Hinweise

Hinweise zu Messdaten

Messdaten werden immer ausgewertet. Dazu gehört:

1. Auftrag der Variablen in einem Diagramm (Excel).
2. Vergleich von Variablen zum Erkennen von Abhängigkeiten.
3. Statistik; Relevanz von Abweichungen gleicher Parameter verschiedener Messreihen; Vergleich mit der Messgenauigkeit.

Hinweise zum Inhalt

* Ein roter Faden in den dargebrachten Gedankengängen ist notwendig; Argu­mente und Informationen müssen in einer lückenlosen Kette wiedergegeben werden, Gedankensprünge sollten vermieden werden.
* Komplizierte Sachverhalte müssen in kleine überschaubare Probleme zerlegt werden.
* Der Text ist zu strukturieren. Nach einem abgeschlossenen Gedankengang wird ein neuer Absatz begonnen – und nicht etwa nach jedem Satz!
* Fachbegriffe, Abkürzungen und Formelzeichen müssen eingeführt oder definiert werden, bevor sie im weiteren Verlauf des Textes benutzt werden. Abkürzungen und Formelzeichen werden zusätzlich in der Nomenklatur aufgeführt.
* Jedes Kapitel der höchsten Gliederungsebene sollte auf einer neuen Seite be­ginnen. Für den Praxissemesterbericht sind ein bis drei Gliederungsebenen sinn­voll. Eine neue Unterebene wird nur eröffnet, wenn mindesten zwei Punkte auf dieser Ebene zu verzeichnen sind (z. B. 4.2.1 und 4.2.2).
* Falls der Betrieb eine Aufgabenstellung vorgibt, wird diese (als Originalblatt) nach dem Titelblatt eingefügt.

Hinweise zur Sprache

* Schreiben Sie kurz und präzise!
* Vermeiden Sie Ich-Formen („ich zeige“, „mein Versuch“), nutzen Sie Passiv-Kon­struktionen („es wird gezeigt“). Einzig das Kapitel „Eigene Eindrücke“ darf in Ich-Form verfasst werden.
* Wertende Ausdrücke wie „gut“, „schlecht“, „beeindruckend“, „bahnbrechend“ sind unwissenschaftlich. Die Darstellung sollte sich auf die Faktenlage beschränken. Qualitative oder quantitative Vergleiche ermöglichen eine sachliche Beurteilung.
* Verwenden Sie für allgemeingültige Aussagen das Präsens („Wasser siedet bei 100 °C.“).
* Verwenden Sie für vergangenes Geschehen das Präteritum („Die untersuchte Flüssigkeit siedete bei 86 °C.“).
* Vermeiden Sie Schachtelsätze mit mehr als zwei Nebensätzen.
* Vermeiden Sie wortgleiche Doppelungen ganzer Sätze, wenn Sie Schritte an­kündigen oder zusammenfassen. Langweilen Sie Ihr Publikum nicht!
* Vermeiden Sie mündliche Ausdrucksweisen wie Füllwörter („mal“, „irgendwie“), Übertreibungen („absolut“, „dramatisch“, „optimal“) und Unschärfen („quasi“, „sozusagen“, „etwa“).
* Auffällige Lieblingswörter oder ungewöhnliche Konstruktionen, die so häufig im Text auftauchen, dass sie als Marotte erscheinen, sind zu vermeiden.
* In Listen und Aufzählungen muss die grammatikalische Konstruktion aller Ein­träge einheitlich sein.

Hinweise zu grafischen Elementen

* Alle Bilder haben Unterschriften, die fortlaufend oder fortlaufend per Kapitel durchnummeriert werden: „Abb. 1: Becherwerk, Seitenansicht“; „Abb. 4.1: Schnittkraft in Abhängigkeit der Schnittgeschwindigkeit der Vertikalschneide, Lö­sungsvariante 1a“.
* Alle Tabellen haben Überschriften, die fortlaufend oder fortlaufend per Kapitel durchnummeriert werden.
* Die Fußnotenziffer steht am Ende der Bildunterschrift oder Tabellenüberschrift. In der Fußnote steht der Kurzverweis auf die Quelle: Verfassername Erschei­nungsjahr, zitierte Seite. Die Quelle wird im Literaturverzeichnis in gleicher Weise wie eine Textquelle angegeben.
* Auf alle grafischen Elemente und Tabellen muss im Text Bezug genommen wer­den: „In Tabelle 5 sind die Daten des ersten Siebvorganges aufgelistet. Die Messdaten des ersten und zweiten Siebvorgangs weichen nur innerhalb der Messgenauigkeit voneinander ab (vgl. Abb. 7).“