

2. Praxistransferbericht

Refactoring einer Komponente zur Erstellung von Zuweisungen im Rahmen der
Durchlaufplanung von Studierenden und Auszubildenden

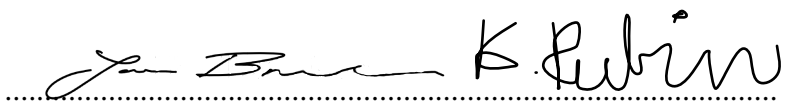
Name, Vorname: Bruckmoser, Jana || Rubin, Kristina
Matrikelnummer: 77220617105 || 77220681190
Ausbildungsbetrieb: Berliner Wasserbetriebe
Studienjahrgang: 2022
Fachbereich: Fachbereich 2 - Duales Studium
Studiengang: Informatik
Modul: IT2311
Betreuer:in Hochschule: Maria Stricker
Betreuer:in Unternehmen: Jeffrey Sonntag
Anzahl der Wörter: ca. 3300

Vom Ausbildungsleiter:in zur Kenntnis genommen:

21.08.2023



(Datum/Unterschrift)



(Datum/Unterschrift der Studierenden)

Kurzfassung / Abstract

Kurzfassung

Das Ziel dieser Arbeit ist es, die bestehende „mailing“-Komponente für die Studierenden und Auszubildenden Plattform der Berliner Wasserbetriebe zu refaktorisieren. Diese soll den aktuellen Sicherheitsstandards gerecht werden. Insgesamt bietet diese Arbeit einen Überblick darüber, wie **Refactoring** von objektorientiertem Code funktioniert und in der Praxis eingesetzt werden kann. Durch die Arbeit in der „mailing“- Komponente wird zudem gezeigt, wie **StAP**-Komponenten aufgebaut sind und welche Vorgaben dabei eingehalten werden müssen.

Abstract

The goal of this work is to refactor the existing mailingcomponent for the Students and Trainees Platform of Berliner Wasserbetriebe. It is intended to meet current security standards. Overall, this thesis provides an overview of how refactoring object-oriented code works and can be used in practice. By working in the mailingcomponent, it is also shown how **StAP** components are structured and which rules must be followed.

Inhaltsverzeichnis

Kurzfassung / Abstract	I
Inhaltsverzeichnis	II
Glossar	III
1 Einleitung	1
2 Moderne Softwareentwicklung	2
2.1 Grundlagen der objektorientierten Programmierung	2
2.2 Codeverbesserung durch Refactoring	5
2.3 Vorteile der objektorientierten Programmierung für das Refactoring	8
3 Refactoring einer Zuweisungserstellungskomponente	9
3.1 Ausgangslage der Studierenden und Auszubildenden Plattform	9
3.2 Analyse der bestehenden Komponente	11
3.3 Anforderungen an die Zuweisungserstellung	14
3.4 Umsetzung des Refactorings	16
3.5 Evaluation der erreichten Ergebnisse	22
4 Fazit	24
Abbildungsverzeichnis	A
Quellenverzeichnis	B
Ehrenwörtliche Erklärungen	G

Glossar

Axios ist ein Paket, das „http“-Nachfragen von Node.js. handelt. [NPM23]

BWB ist die Kurzform für Berliner Wasserbetriebe.

Client ist für das Versenden von Anfragen und Empfangen der Antworten des Servers im Client-Server-Modell zuständig. [Gui22a]

Client-Server-Modell ist ein Programmiermodell, welches in Clients und Server aufgeteilt ist.

createPDF ist eine Methode aus htmlPDF zur Erstellung von PDF Dokumenten.

CSS steht für "Cascading Style Sheets" und fungiert als Stylevorlage für HTML-Elemente. [Scha]

Framework ist ein Entwicklungsrahmen, welcher die grundlegende Architektur der Software vorgibt. [AGoD]

HTML steht für "Hyper Text Markup Language" und beschreibt die Struktur von Webseiten. [Schc]

NestJs ist ein Framework der Laufzeitumgebung Node.js, welches JavaScript-Dateien außerhalb eines Webbrowsers ausführen kann. [Was21c]

placeholders sind Platzhalter für eine Variable, diese stammen aus der „handlebars" Bibliothek. [BWBintern]

Refactoring steht in Englisch für Refaktorisierung.

screen ist ein "Media Type", dieser gibt Information aus, ob es sich um einen Computer-, Tablett- oder um einen Mobiltelefonbildschirm handelt und nicht zum Beispiel um einen Screenreader. [Schb]

Server ist das Computerprogramm, das die Anfragen des Clients annimmt und bearbeitet. [Gui22a]

SMTP steht für Simple Mail Transfer Protocol und ist das Standardprotokoll zum Übertragen von E-Mails im Internet.

StAP ist die Kurzform für Studierenden und Auszubildenden Plattform, dabei handelt es sich um eine Webanwendung der Berliner Wasserbetriebe zur Organisation der Studierenden und Auszubildenden.

Stuzubis ist die Kurzform für Studierende und Auszubildende.

TypeScript ist eine streng typisierte Programmiersprache, die auf der Skriptsprache JavaScript aufbaut. [TypoD]

Vite ist ein Tool, welches JavaScript-Dateien für die Browsernutzung zusammenzuführen und zu einer Datei bündeln kann. [VitoD]

Vue.js ist ein Framework, welches TypeScript nutzt zum Erstellen von Single-Page-Webanwendungen. [Was21a]

1 Einleitung

Im Rahmen des Studiengangs Informatik ist es für die dual Studierenden vorgesehen, innerhalb ihrer ersten drei Praxisphasen einen Praxistransferbericht zu verfassen.

Die Berliner Wasserbetriebe sind mit 4634 Mitarbeitern der größte Wasserver- und Abwasserentsorger Deutschlands. Mit einer 6,7 prozentigen Ausbildungsquote entsteht allein für die Ausbildung ein großer Verwaltungsaufwand. (vgl. [Was23c]) Dieser soll im Zeitalter der Digitalisierung fortwährend digitaler und auch einheitlicher für alle Organisationseinheiten werden. Aufgrund dessen wird eine Webanwendung, die genau diese Zwecke, Digitalisierung und Optimierung, bedient von den Berliner Wasserbetrieben entwickelt.

Dieser Praxistransferbericht befasst sich speziell mit einer Komponente der Webanwendung, die zu der Erstellung und Versendung einer Zuweisungs-PDF benötigt wird. Im Folgenden wird zunächst auf die Grundlagen der objektorientierten Programmierung und den Begriff des Refactorings eingegangen. Danach wird die Ausgangslage des Projekts beschrieben, die bereits bestehende Komponente wird analysiert und es werden die Anforderungen für die Komponente dargelegt. Anschließend folgt die Entwurfsrealisierung und Evaluation der Ergebnisse. Zum Ende der Arbeit wird kritisch ein Fazit gezogen.

2 Moderne Softwareentwicklung

2.1 Grundlagen der objektorientierten Programmierung

Die objektorientierte Programmierung (OOP), ist ein Programmiermodell, welches Daten und zugehörigen Code in sogenannten Objekten zusammenfasst. [Aug18]

Die ausschlaggebenden Bausteine der OOP sind:

- **Klassen**, welche benutzerdefinierte Datentypen sind und als Vorlage für individuelle Objekte, Attribute oder Methoden dienen. [Com22]
- **Objekte**, welche Instanzen einer Klasse sind und unter Verwendung von speziell festgelegten Daten erzeugt werden. Objekte können abstrakt sein oder realen Objekten entsprechen. [Com22]
- **Methoden**, welche Funktionen repräsentieren, die das Verhalten eines Objekts festlegen und innerhalb einer Klasse definiert werden. Sie werden zur Wiederverwertbarkeit verwendet oder um die jeweilige Funktionalität in einem Objekt zu kapseln. [Com22]
- **Attribute / Parameter**, welche in der Klassenvorlage definiert werden und den Zustand eines Objekts beschreiben. Die Daten der Objekte werden im Attributfeld gespeichert, Klassenattribute gehören jedoch zur Klasse selbst. [Com22]

Die objektorientierte Programmierung ruht auf vier grundlegenden Prinzipien, welche ihre Struktur auszeichnet. Die **Kapselung** besagt, dass wesentliche Daten innerhalb des Objektes eingeschlossen sind und nur vereinzelte Daten nach außen sichtbar gemacht werden. Der Zustand und die Implementierung der Objekte werden in einer definierten Klasse privat gehalten, wodurch andere Objekte keinen Zugriff haben. Sie können nur eine Liste öffentlicher Funktionen oder Methoden aufrufen. Dies trägt zur Programmsicherheit und Verhinderung von Datenverfälschung bei. [Com22]

Der Begriff **Abstraktion** bedeutet, dass Objekte nur die inneren Abläufe preisgeben, die für die Interaktion mit anderen Objekten relevant sind. Die abgeleitete Klasse kann in Ihrer Funktionalität erweitert werden. Dies ermöglicht Entwicklern:innen im Verlauf des Projekts zusätzliche Änderungen oder Erweiterungen einfacher umzusetzen. [Com22]

Das Prinzip der **Vererbung** ermöglicht es Klassen den Code von anderen Klassen wiederzuverwenden. Entwickler:innen können eine gemeinsame Logik wiederverwenden und eine Hierarchie beibehalten, indem sie Beziehungen und Unterklassen zwischen den Objekten zuweisen. Dies erfordert eine umfassendere Datenanalyse, welche die Zeit der Entwicklung reduziert und ein höheres Maß an Präzision gewährleistet. [Com22]

Der **Polymorphismus** gewährt es Objekten ein gemeinsames Verhalten und mehr als eine Form annehmen zu können. Um Code-Redundanzen zu verringern kann das Programm bestimmen, welche Bedeutung oder Verwendung für jede Ausführung dieses Objekts aus einer übergeordneten Klasse erforderlich ist. Darauf folgend kann eine Kindklasse erstellt werden, welche die Funktionalität der Elternklasse erweitern kann. Durch Polymorphismus können unterschiedliche Objekttypen dieselbe Schnittstelle durchlaufen. [Com22]

Nach der Erläuterung der Grundprinzipien der OOP folgt nun eine Darstellung ihrer Vor- und Nachteile.

Vorteile	Nachteile
Die Gruppierung von Funktionalität durch Klassen verhindert eine lose Sammlung mehrerer Variablen und Methoden. [Gui22b]	Zu Beginn ist eine höhere Einarbeitungszeit nötig, damit sich die Entwickler:innen in die Konzepte einarbeiten können. [Gui22b]
Die Modellierung von objektorientierten Systemen ist angelehnt an die menschliche Denkweise. Es ist sowohl möglich physische Dinge als auch abstrakte Modelle als Objekt darzustellen. [Gui22b]	Das Prinzip der Kapselung führt zu Problemen, wenn Code parallelisiert wird. Diese Konflikte entstehen, da Objekte ihren Zustand zwischen den Methodenaufrufen verändern können, wenn diese von mehreren parallel ablaufenden Funktionen genutzt werden. [Gui22b]
Die Verwendung von Klassen-Hierarchien zur Vererbung reflektiert analoge Denkmuster, welche eine leicht verständliche Strukturierung des Codes begünstigt. [Gui22b]	Durch den Fokus auf dynamischen Code in der OOP wird die Programmpformance vernachlässigt. Dies liegt an weniger statischen Optimierungen und einer schwächeren Typisierung. Daraus folgend werden Fehler erst in der Laufzeit sichtbar. [Gui22b]

Im Anschluss wird die visuelle Darstellung von objektorientierter Programmierung betrachtet.

Die Unified Modeling Language (UML), auf Deutsch „vereinheitlichte Modellierungssprache“, wird zur Verbildlichung von Objekten, Zuständen und Prozessen innerhalb eines objektorientierten Systems genutzt. UML-Diagramme können als Vorlage zur Projektumsetzung genutzt werden, aber auch zur anschaulichen Erklärung des Projekts für möglicherweise Fachfremde. Sie hat zusätzlich eine strukturierte Semantik, was Missverständnisse in der Interpretation verringert. Mit UML können grafisch einzelne Objekte, deren Beziehungen, Klassen, Interaktionen mit Schnittstellen und komplexere Kombinationen von Aktivitäten dargestellt werden. [Gui18]

2.2 Codeverbesserung durch Refactoring

Refactoring ist ein systematischer Prozess zur Verbesserung von Code, ohne neue Funktionalitäten hinzuzufügen. Dieser Prozess wird durch verschiedene Refactoringtechniken umgesetzt und soll zu Clean Code führen. [RG23q] Clean Code ist laut Grady Booch „[...] *einfach und direkt* [...]“ [Mar13, S. 16]. Er erleichtert die Softwareentwicklung und erhöht die Qualität des resultierenden Produkts. Hierzu gibt es unterschiedliche Vorgaben, um ein allgemeines sauberes Ergebnis zu ermöglichen. Im Refactoring-Prozess treten sogenannte Code Smells auf, die Indikatoren für Probleme in der Implementierung sind. Sie selbst sind meist geringwertige Fehler, können aber Symptome für große Probleme oder wartungsintensiven Code sein. [RG23q]

Code Smells lassen sich in verschiedene Gruppen unterteilen. Diese sind

- „Bloaters“, Klassen und Methoden von unverhältnismäßiger Größe, die den Code dadurch „aufblähen“, [RG23b]
- die falsche Anwendung objektorientierter Prinzipien, [RG23o]
- starke Interdependenzen, welche bei kleinen Änderungen viele Korrekturen nach sich ziehen und [RG23c]
- irrelevanter, redundanter oder veralteter Code, der nicht mehr benötigt wird. [RG23i]

Zur Entfernung dieser Code Smells kommen diverse Refactoringtechniken zur Anwendung, welche sowohl die Les- und Veränderbarkeit verbessern. Dazu gehören

- die verbesserte Komposition von Methoden, [RG23e]
- das Verschieben von Methoden zwischen Objekten, [RG23n]
- die Organisation von Daten, [RG23p]
- das Vereinfachen und Aufbrechen bedingter Ausdrücke, um Verschachtelungen zu minimieren, [RG23s]
- das Vereinfachen von Methodenaufrufen und [RG23t]
- die Generalisierung. [RG23h]

Zu diesen Obergruppen folgt nun eine detaillierte Beschreibung bestimmter Code Smells und den jeweilig passenden Refactoringtechniken.

Code Smell	Refactoringtechnik
Eine unvollständige Bibliotheksklasse tritt auf, wenn Bibliotheken nicht mehr den Benutzeranforderungen entsprechen und keine neue Methode hinzugefügt werden kann. [RG23l]	Das Einfügen einer neuen Methode , das Ersetzen des vorherigen Algorithmus um sich der neuen Methode anzupassen und die Ergänzung fehlender Parameter können diesen Code Smell beheben. [RG23m] [RG23u] [RG23a]
Eine divergente Veränderung bezeichnet den Code Smell, wenn in einer Klasse viele nicht zusammenhängende Methoden angepasst werden müssen. Diese treten oft als Folge von anderen Refactoringprozessen auf. [RG23j]	Die gruppierbaren Codefragmente sind in eine neue Methode zu extrahieren . So soll das Beheben auftretender Fehler erleichtert werden. [RG23k]
Ein weiterer Code Smell, der als Folge auftreten kann ist der „ tote Code “. Dieser beschreibt Variablen, Methoden, Parameter oder Klassen, die nicht mehr genutzt werden. [RG23g]	Eine mögliche Lösung hierfür ist das Löschen von Parametern . [RG23r]
Ein allgemeiner Code Smell ist die überschüssige Nutzung von Kommentaren , dadurch wird der Code unübersichtlich und schlecht lesbar. [RG23d]	Kommentare können durch eine logische Benennung der Methoden eingespart werden. Zusätzlich erleichtert eine einheitliche Erläuterung der Methode in der Methodenbeschreibung die Lesbarkeit. [Was22a]

Im Folgenden soll der explizite Ablauf des Refactorings, wie in Abbildung 2.1 dargestellt, betrachtet werden. Der Refactoringprozess eines objektorientierten Quellcodes beginnt mit auftauchenden Code Smells. Daraufhin werden Codeabschnitte mit Refactoringbedarf festgelegt. Bevor diese durch die jeweils angebrachten Refactoringtechniken ausgebessert werden, muss zuerst geprüft werden, ob die gesamte Funktionalität des Codes gleich bleibt. Wenn dies nicht der Fall ist, muss das Refactoring neu geplant werden. Der selbe Prozess geschieht auch, wenn während des Refactorings neue Anforderungen an den Code gestellt werden und diese zusätzlich implementiert werden müs-

sen. Nach Anwendung des Refactorings, werden eventuelle Auswirkungen auf die Qualität des objektorientierten Codes geprüft und korrigiert. Dies ist erforderlich, um die Konsistenz zwischen vorhandenen und vom Refactoring betroffenen Komponenten sicherzustellen. [IDE16]

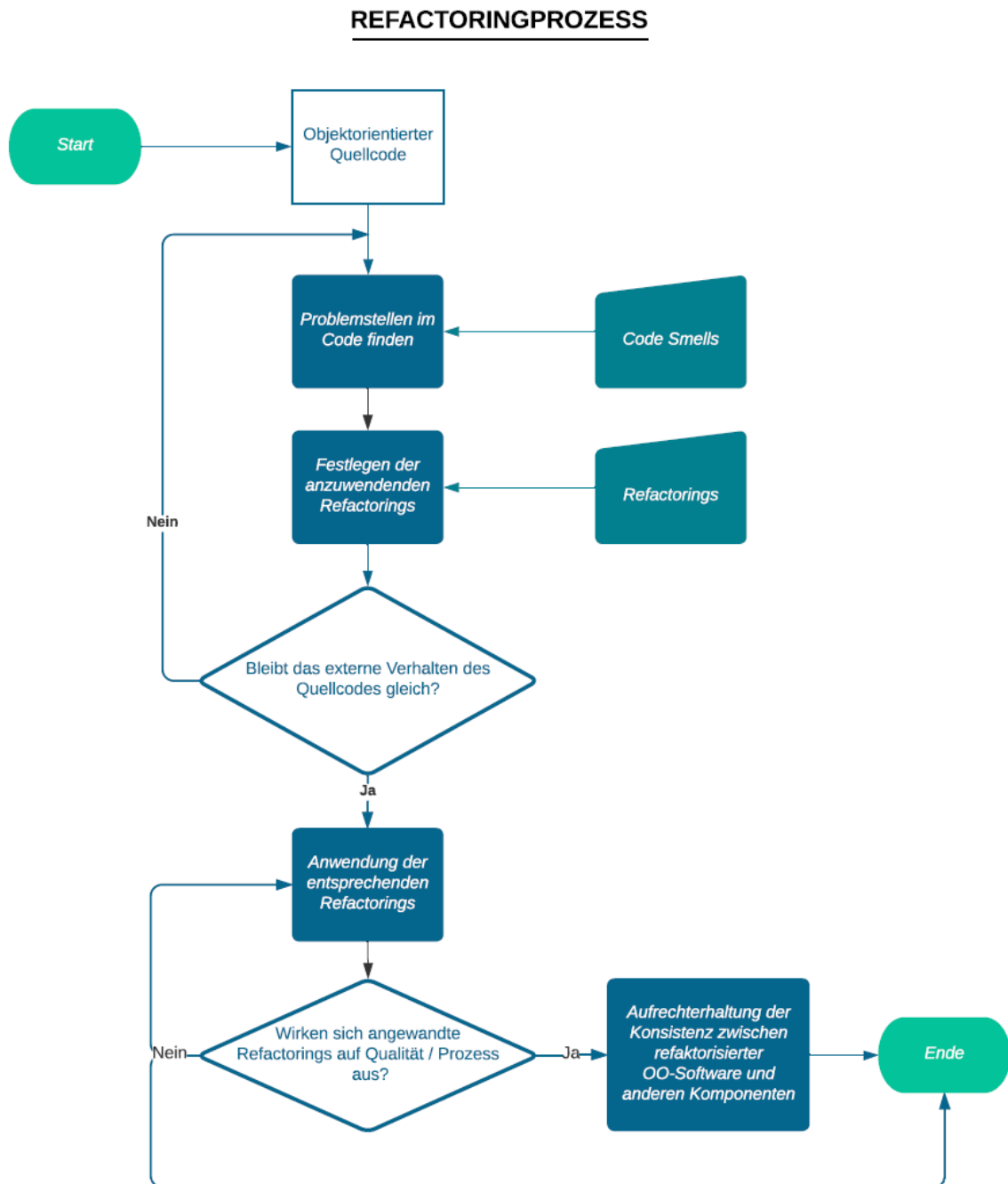


Abbildung 2.1: Refactoringprozess

2.3 Vorteile der objektorientierten Programmierung für das Refactoring

Die Anwendung objektorientierter Programmierung erleichtert im Rahmen ihrer in 2.1 aufgeführten Merkmale den Prozess des Refactorings.

So lassen sich durch die Datenkapselung Code Smells reduzieren, da grundsätzlich Coderedundanzen verringert und Refactoringprozesse vereinfacht werden. Die Komposition von Methoden lässt sich exemplarisch mit deutlich geringerem Aufwand durchführen. So simplifiziert die Aufteilung und damit einhergehende Kohärenz die Überarbeitung konkreter Prozesse im Code. Diese Problematik ist in Abbildung 2.2 anhand der durch OOP und Refactoring wegfallende Code Smells zu erkennen.

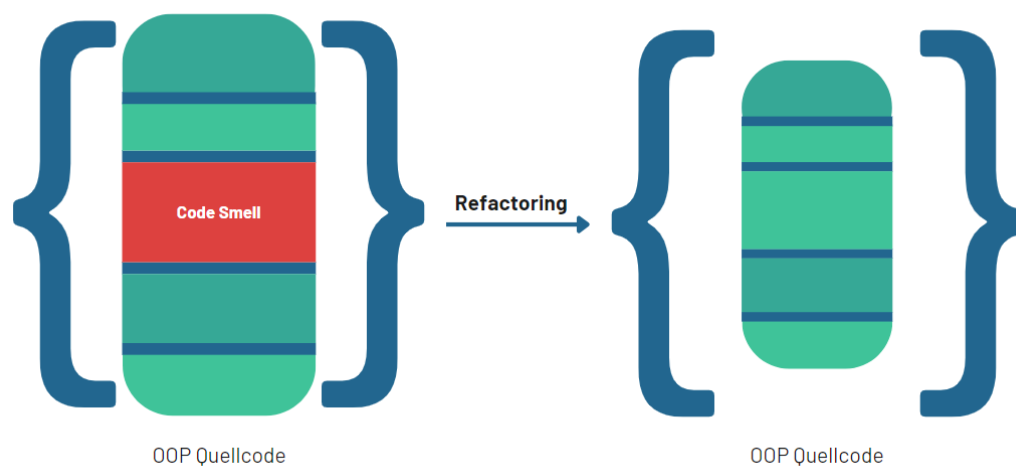


Abbildung 2.2: OOP und Refactoring

3 Refactoring einer Zuweisungserstellungskomponente

3.1 Ausgangslage der Studierenden und Auszubildenden Plattform

Die Studierenden und Auszubildenden Plattform (StAP) ist eine Webanwendung der **BWB**. Diese soll die Organisation von Terminen und Zuweisungen von Studierenden und Auszubildenden vereinfachen, da dies momentan von allen Abteilungen inhomogen organisiert wird. Betreut und entwickelt wird die Anwendung jahrgangsübergreifend von Studierenden und Auszubildenden der IT-Berufe.

Das StAP ist nach dem **Client-Server-Modell** aufgebaut. Dieses läuft unabhängig von sämtlichen Hardwareplattformen und auf allen Rechnerarchitekturen. Es besteht aus zwei Hauptkomponenten, dem **Client** und dem **Server**, welcher noch auf Datenbanken zugreifen kann. [Ben15] In der Anwendung kann es mehrere Clients geben, die Datenabfragen oder Funktionsausführungen an den Server senden. Im Server werden diese Anfragen mit Hilfe von Daten aus der Datenbank verarbeitet. Der Server erhält die Daten von der Datenbank und gibt diese als Antwort an den Client zurück oder veranlasst die Ausführung der Funktionen. [Ben14]

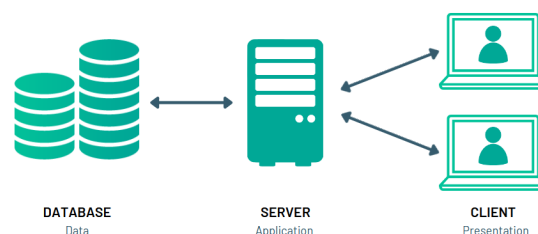


Abbildung 3.1: Client-Server-Modell

Der Client der StAP ist eine Single-Page-Application, also eine Webanwendung, bei der alles auf einer Seite umgesetzt wird, anstatt mehrere Seiten laden zu müssen, die mit dem **Framework Vue.js** erstellt wird. Der Quellcode ist durch Vue.js in viele **TypeScript**-Dateien aufgeteilt. Zur Steuerung der Kompilation und dem Zusammenführen der unterschiedlichen Dateien in eine einzelne JavaScript-Datei wird **Vite** genutzt . [Was21a]

Der Server verwendet das NodeJs-Framework **NestJs** . Er ist komplett in Typescript geschrieben und nutzt eine feste Applikationsstruktur, welche unter anderem über eine Kommandozeilenschnittstelle eingerichtet werden kann. [Was21c]

Die Datenbank der StAP ist eine relationale Datenbank mit PostgreSQL, das heißt sie ist in Form von Tabellen organisiert. Sie unterstützt komplexe Abfragen, Transaktionen und Beziehungen zwischen den Tabellen. [Was21b]

3.2 Analyse der bestehenden Komponente

Für die nachfolgende Analyse wird der StAP-Server betrachtet. Dieser Server teilt sich in verschiedene funktionsabhängige Komponenten auf. Im Blickpunkt der Analyse steht die „mailing“-Komponente. Die Hauptaufgabe dieser Komponente besteht darin, automatisch ein selbstgeneriertes Zuweisungsdocument per E-Mail zu versenden, sobald eine Zuweisung im StAP-Client erstellt wurde. Für eine bessere Übersicht teilt sich das Programm in die in Abbildung 3.2 abgebildete Ordnerstruktur.

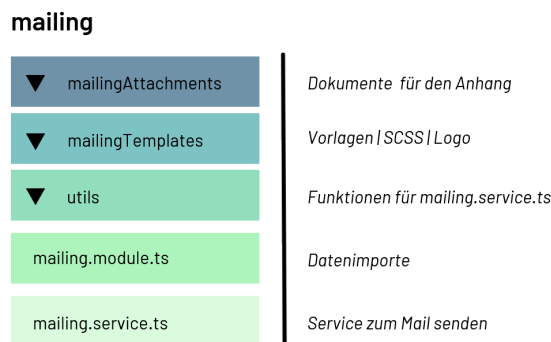


Abbildung 3.2: Ordnerstruktur der „mailing“-Komponente

Um diesen Prozess zu initiieren, muss ein:e Mitarbeiter:in aus dem Personalmanagement im StAP das passende Formular öffnen und eine Zuweisung erstellen. Daraus folgend wird im Server der „mailing“-Prozess beziehungsweise das „mailing.module“ gestartet. Die Abwicklung der angeforderten Komponente wird in der unteren Hälfte des Ablaufdiagramms dargestellt:

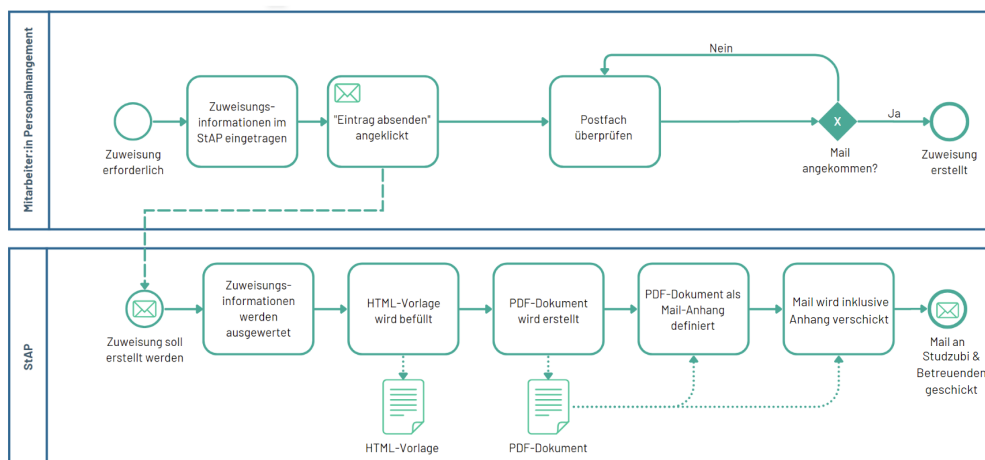


Abbildung 3.3: Prozessschaubild „mailing“

Innerhalb des Programmes wird damit der „mailing.service“ aufgerufen. Die Hauptmethode dieses Services, genannt „sendMail()“, nutzt importierte Methoden aus „mailing.utils“, um das Dokument zu generieren und zu versenden. Dieser Prozess ist in die folgenden Schritte aufgeteilt:

1. Zuweisungsinformationen werden ausgewertet

Mit Hilfe der „getMailingType()“-Methode wird eine von den drei subsequenten Methoden ausgewählt:

- „getBibDay()“ → für einen Bibliothekstag
- „getVacation()“ → für einen Urlaubsbeantragung
- „getWorking()“ → für die Zuweisungserstellung

Für das in dieser Arbeit betrachtete Refactoring ist nur die zuweisungserstellende Methode „getWorking()“ relevant. Diese definiert den Betreff der Mail, den Inhalt über einen Pfad zur **HTML**-Vorlage im „mailingTemplates“ Ordner und den eventuellen Anhang über einen verlinkten Pfad zum „mailingAttachments“ Ordner.

Anschließend werden in der „getMailingData()“-Methode die Daten des **Stuzubis** bereinigt, geladen und zwischengespeichert. Die Daten des Betreuenden werden separat in der „getInstructorHTML()“-Methode in einen HTML-String geladen. So können auch mehrere Betreuende pro Stuzubi angezeigt werden.

2. HTML-Vorlage wird befüllt

Die im vorherigen Schritt zwischengespeicherten Daten werden nun unter Verwendung von „**placeholders**“, sogenannten Platzhaltern, in die HTML-Vorlage eingesetzt.

3. PDF-Dokument wird erstellt

Die HTML-Vorlage wird nun mit der Methode „**createPDF()**“ in eine einzigartig benannte PDF unter den „mailingAttachments“ gespeichert.

4. E-Mail wird inklusive Anhängen verschickt

Im letzten Schritt des Programmes wird die E-Mail über die „sendMailWithAttachments()“-Methode über **Axios** an einen Test-SMTP Server geschickt. Dieser wird im Produktivsystem nicht zum Einsatz kommen, ist gerade jedoch aus sicherheitstechnischen Gründen notwendig.

Jetzt liegt es am Stuzubi und Betreuenden diese zu öffnen und somit den Prozess der Zuweisung zu beenden.

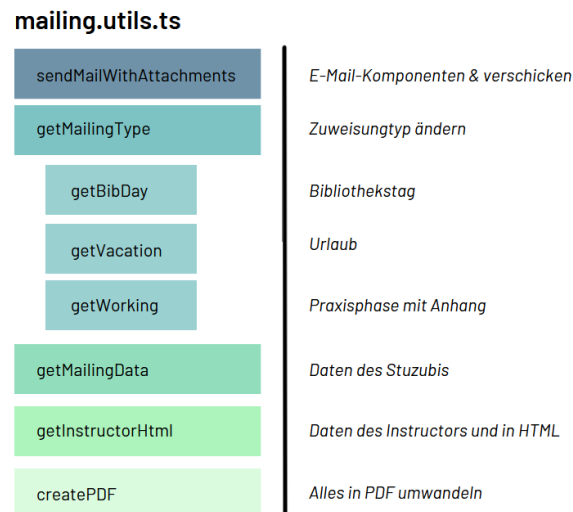


Abbildung 3.4: Zusammenfassung Methoden aus „mailing.utils“

3.3 Anforderungen an die Zuweisungserstellung

Im folgenden Kapitel werden die Anforderungen an das Refactoring der Zuweisungserstellung aus den vorhandenen Code Smells näher erläutert:

„html-pdf“ Paket ersetzen:

Unvollständige Bibliotheksklasse

Der letzte Stand vom Paket „html-pdf“ stammt aus dem Jahr 2021 und wird nicht mehr aktualisiert. Damit entspricht es nicht dem jetzigen Sicherheitsstandard der BWB. [Mar21] Es muss durch ein aktuelleres Paket ersetzt werden. Stattdessen soll die Bibliothek „puppeteer“ mit weiterlaufenden Aktualisierungen genutzt werden. Das Paket „puppeteer“ wird für automatisierte Client Tests verwendet. Es ist damit möglich Testwebseiten zu laden, zu bedienen und als PDF zu downloaden. Wichtig ist, dabei die gesamte Funktionalität der alten Methode beizubehalten. [Pup23a]

BWB-Design implementieren:

Divergente Veränderung

Alle Software-Anwendungen der BWB müssen sich an das Corporate Design halten. Dazu gehört ein Blau-Grünes Farbschema, feste Schriftarten und die korrekte Verwendung des Firmenlogos. [Was23a]

Clean Code–Regeln berücksichtigen:

Kommentare

Das StAP-Projekt hat eigene Clean Code–Vorlagen, die von den häufig wechselnden Programmierer:innen genutzt werden müssen. Damit soll die StAP besser nachvollziehbar werden und die Einarbeitungszeit möglichst gering sein. Es ist darauf zu achten, alle Vorlagen anzuwenden und diesen entsprechend zu kommentieren. [Was22b]

Zusätzlich wurden von der Personalabteilung neue funktionale Änderungen gewünscht, die zu implementieren sind. Der Fokus hierbei liegt auf der Trennung von Stuzubis in zwei unterschiedliche Zuweisungsdokumente und einen von der Organisationseinheit unabhängigeren Aufbau. [Was23b] Weitere Änderungsanforderungen der Personalabteilung sind in der folgenden Übersicht dargestellt.

Adressat und Inhalt der E-Mail ändern:	Die E-Mail soll nicht direkt von der StAP an die Stuzubis weitergeleitet werden. Stattdessen soll sie zuerst an die erstellende Person zur Kontrolle der Eingaben gesendet werden. Daraus folgt eine Anpassung des E-Mail Textes.
Toter Code / Ergänzung von Parametern	
E-Mail Anhänge entfernen:	Zudem haben die Abteilungen verschiedene eigene Anhänge für die E-Mail. Deswegen soll nur bei der Zuweisung das automatisch generierte Zuweisungsdokument mitgeschickt werden. Die restlichen Dokumente sollen manuell vom Abteilungspersonal hinzugefügt werden können.
Toter Code	
Unterschiedliche Formatvorlagen:	Abschließend soll das Dokument für Stuzubis aufgeteilt werden. Grund hierfür sind die variierenden Informationen für Auszubildende und Studierende.
Ergänzung von Parametern	

3.4 Umsetzung des Refactorings

Nachdem die Code Smells lokalisiert wurden, müssen die jeweils passenden Refactorings festgelegt und angewandt werden:

„html-pdf“ Paket ersetzen:

Das veraltete Paket „html-pdf“ wird in der Methode „createPDF()“ genutzt. Das bedeutet, dieser Algorithmus muss mit Hilfe des „puppeteer“-Paket ersetzt werden:

Dazu wird asynchron unter Verwendung des „puppeteer“-Pakets ein Browser geöffnet.	<i>puppeteer.launch()</i>
---	---------------------------

Auf diesem wird die HTML-Vorlage für die Zuweisung in eine neue Seite geladen und anschließend als PDF gespeichert. [Pup23d]	<i>browser.newPage()</i> <i>page.pdf()</i>
--	---

```
export const createPDF = (uniqueFilename: string, htmlContent: string) => {
  return new Promise(async (resolvePromise) => {

    const browser = await puppeteer.launch({
      headless: "new",
      ignoreHTTPSErrors: true,
    });

    const page = await browser.newPage();
    await page.setContent(htmlContent, {
      waitUntil: ["domcontentloaded", "networkidle0", "load", "networkidle2"],
    });

    resolvePromise(
      await page.pdf({
        path: `./exports/allocation/${uniqueFilename}.pdf`,
        margin: {top: "100px", right: "50px", bottom: "100px", left: "50px"},
        printBackground: true,
        format: "A4",
      })
    );
  });
};
```

Abbildung 3.5: Quellcode von „createPDF()“

BWB-Design implementieren:

Das Corporate Design wird mittels der CSS-Datei auf das HTML-Template angewandt. Um diese aufzurufen wird die „puppeteer“ „Page.addStyleTag()“-Methode verwendet. Sie verlinkt die CSS-Datei im HTML-Dokument mittels „<link rel=„stylesheet“>“. [Pup23b] Danach muss die „page.emulateMediaType()“-Methode verwendet werden, um den Medientyp korrekt auf „screen“ einzustellen. [Pup23c]

```
await page.addStyleTag({
  path: __dirname + "../mailingTemplates/" + "corporate.css",
});
await page.emulateMediaType("screen");
```

Abbildung 3.6: Quellcode von der „createPDF()“ CSS Ergänzung

Zu beachten ist, dass „puppeteer“ aus Sicherheitsgründen nicht auf lokale Dateien zugreift. Demzufolge wird in der HTML-Vorlage das Logo nicht geladen. Zur Lösung wird das Bild mittels der Base64-Kodierung als String eingefügt. Für einen übersichtlichen Quellcode wird die Kodierung in eine neue Methode „base64Encode()“ extrahiert.

```
function base64Encode(filepath) {
  return "data:image/gif;base64," + readFileSync(filepath, "base64");
}
```

Abbildung 3.7: Quellcode von „base64Encode()“

Clean Code – Regeln berücksichtigen [Was22b]:

DRY-Prinzip (Don't Repeat Yourself) – <i>Sich wiederholende Aufgaben sollen automatisiert werden.</i>	Durch die Auslagerung des Codes in Methoden innerhalb des „mailing.utils“-Ordners wird konsequent die Wiederholung von Code vermieden. Darüber hinaus hilft die OOP bei der Datenkapselung. Dies ermöglicht eine einfache Modifikation oder Ergänzung von Methoden, wie die Anpassung der „createPDF()“-Methode oder die Erstellung der „base64Encode()“-Methode.
Klare Funktionen – <i>Eine Funktion sollte nicht unübersichtlich lang sein und wenn möglich nur eine Aufgabe haben.</i>	Die Methode „createPDF()“ ist die geordnete Zusammenführung der essentiellen Schritte zur PDF-Generierung. Indem sie diese Aufgaben bündelt und organisiert, wird sichergestellt, dass die Funktionalitäten in einer leicht nachvollziehbaren Weise ausgeführt werden. Dadurch wird gewährleistet, dass der Code übersichtlich bleibt. So wird die Anpassung oder Erweiterung in Zukunft erleichtert.
Kommentare – <i>Zum besseren Verständnis und einer möglichst effizienten Einarbeitung soll der Code so gut wie möglich auskommentiert sein.</i>	Zur Dokumentation von Funktionen wird JSDoc genutzt. Bei diesem Verfahren wird ein mehrzeiliger Kommentarblock vor dem Funktionsblock geschrieben. In dem Kommentar wird zuallererst eine kurze Zusammenfassung der Funktion niedergeschrieben. Darauffolgend werden die übergebenen Parameter inklusive Typ beschrieben. Sollte die Funktion einen Rückgabewert haben, wird auch dieser im Kommentarblock beschrieben. Zusätzlich werden die Autoren:innen inklusive Kontaktdaten vermerkt.

```
* This function takes a subject and a string containing html-code. It then creates  
* a pdf-file named as the passed subject and saves it to ./export/allocation/  
* @author Kristina Rubin <kristina.rubin@bwb.de>  
* @author Jana Bruckmoser <jana.bruckmoser@bwb.de>  
* @param {string} uniqueFilename  
* @param {string} htmlContent  
*/
```

Abbildung 3.8: Quellcode von „createPDF()“

Adressat und Inhalt der E-Mail ändern:

Die neuen Anforderungen vom Personalmanagement sollen ein fehlersicheres Versandverfahren generieren und die E-Mail zuerst zurück an die erstellende Person schicken. Dazu muss der „send-MailWithAttachments()“-Methode eine andere Empfängeradresse gegeben werden. Diese wird mit Hilfe des importierten „bwbUserService“ über die Nutzeridentifikation nachgefragt und mitgegeben.

```
const mailingArgs = {  
  to: [(await this.bwbUserService.getBWBUserByID(this.request.user?.id)).email],
```

Abbildung 3.9: Quellcode von Adressatänderung

Der Text der E-Mail wird im „allocation-mail-template.html“ durch einen angepassten Text ersetzt. Zusätzlich muss ein neuer „placeholder“ für die Adressierung der erstellenden Person generiert und verwendet werden.

Anhänge entfernen:

Ein klassisches Beispiel von totem Code sind die E-Mail Anhänge, welche mit den neuen Vorgaben entbehrlich wurden. Diese müssen aus dem Quellcode und dem Backend gelöscht werden. Hierzu wird der Parameter „attachments“ aus den Methoden „getBibDay()“, „getVacation()“ und „getWorking()“ entfernt. Die Dokumente werden manuell aus dem Ordner entfernt.

Unterschiedliche Formatvorlagen:

Die Entwicklung einer zweiten HTML-Vorlage für Azubis ist als Ergänzung fehlender Parameter zu sehen. Hierbei wird eine neue HTML-Datei auf dem Server erstellt. Diese Vorlage wird gestaltet und strukturiert, wobei die grundlegenden Elemente für die Anzeige von Informationen implementiert werden. Zusätzlich gibt es eine Auswahl, die den automatischen Wechsel zwischen den beiden Vorlagen ermöglicht. Diese sind in folgenden Abbildungen zu sehen:

Zuweisung dual Studierende:r

16.08.2023
Geoff Wochenende

Zuweisung für die praktische Ausbildung bei den Berliner Wasserbetrieben und bei der Berlinwasser Gruppe im Rahmen des Studiums an der Hochschule für Wirtschaft und Recht Berlin (HWR), Fachhochschule Potsdam und Beuth Hochschule:

Studierende:r:	Ingo Fröhlich
Studiengang:	Informatik , Bachelor of Science
Semester:	noch zu ergänzen
User-ID:	BWB211
Ausbildungsbeginn:	01.10.2021
Ausbildungsende:	30.09.2024
Studiengangsbeauftragte:r:	noch zu ergänzen
Praxisbetreuer:in:	Florian Bruder
Zuweisungsbeginn:	01.10.2021
Zuweisungsende:	31.03.2022
Bemerkungen:	keine Bemerkungen

Der bzw. die Studierende wird sich eine Woche vor Beginn dieser Zuweisungsphase mit dem bzw. der Praxisbetreuer:in in Verbindung setzen um Ort und Termin des Einführungsgesprächs abzustimmen!

Die Aufgabenstellung(en), die Ausbildungsziele und -inhalte werden im Einführungsgespräch vereinbart und sollten in einem Protokoll zum Einführungsgespräch festgehalten werden. Im Einführungsgespräch wird ferner das Thema für den Praxistransferbericht skizziert.

Der bzw. die Studierende soll bis zum Termin des Kompetenzeinschätzungsgesprächs, welches zum Ende der Praxisphase geführt wird, einen Tätigkeitsbericht bei dem bzw. der Praxisbetreuer:in und bei PM-A/K vorlegen.

Der bzw. die Studierende ist verpflichtet alle Studienunterlagen nach Beendigung der Praxisphase bei PM-A/K einzureichen.

Der bzw. die Studierende wird den Urlaub während der Zuweisungszeit nehmen und ihn mit dem bzw. der Ausbildungsbeauftragten abstimmen.

Hinweis zu Unterweisungen: Die Studierenden erhalten durch ihre für sie zuständigen Ausbilder:innen im Ausbildungszentrum 2x pro Ausbildungsjahr (½ jährlich, bezogen auf das Ausbildungsjahr) eine allgemeine Unterweisung. Die entsprechende Nachweisführung erfolgt im Ausbildungszentrum. Die für die Umsetzung von Arbeitsaufgaben im Zuweisungsbereich notwendigen arbeitsspezifischen Unterweisungen sind durch den/die Beauftragten vor Ort zu vermitteln und nachweislich zu dokumentieren.

Abbildung 3.10: Zuweisungsdocument Duales Studium

Zuweisung Auszubildende:r

16.08.2023
Geoff Wochenende

Zuweisung für die praktische Ausbildung bei den Berliner Wasserbetrieben.
Der bzw. die Auszubildende soll vor Beginn der Ausbildung in dem Bereich über deren Arbeitsweise und organisatorische Eingliederung in das Unternehmen unterrichtet werden.

Auszubildende:r:	Max I. Miyan Aster
Ausbildungsberuf:	Fachinformatik für Anwendungsentwicklung
Ausbildungsjahr:	noch zu ergänzen
User-ID:	BWB217
Ausbildungsbeginn:	01.10.2021
Ausbildungsende:	30.09.2024
Ausbildungsbeauftragte:r:	noch zu ergänzen
Praxisbetreuer:in:	Florian Bruder
Zuweisungsbeginn:	01.10.2021
Zuweisungsende:	31.03.2022
Blockwoche:	noch zu ergänzen
Bemerkungen:	keine Bemerkungen

Der bzw. die Auszubildende wird sich eine Woche vor Beginn dieser Zuweisungsphase mit dem bzw. der Ausbildungsbeauftragten in Verbindung setzen um Ort und Termin des Einführungsgesprächs abzustimmen!

Der bzw. die Auszubildende wird den Urlaub während der Zuweisungszeit nehmen und ihn mit dem bzw. der Ausbildungsbeauftragten abstimmen.

Die Aufgabenstellung(en), die Ausbildungsziele und -inhalte werden im Einführungsgespräch vereinbart und sollten in einem Protokoll zum Einführungsgespräch festgehalten werden.

Hinweis zu Unterweisungen: Die Auszubildenden erhalten durch ihre für sie zuständigen Ausbilder:innen im Ausbildungszentrum 2x pro Ausbildungsjahr (½ jährlich, bezogen auf das Ausbildungsjahr) eine allgemeine Unterweisung. Die entsprechende Nachweisführung erfolgt im Ausbildungszentrum. Die für die Umsetzung von Arbeitsaufgaben im Zuweisungsbereich notwendigen arbeitsspezifischen Unterweisungen sind durch den/die Beauftragten vor Ort zu vermitteln und nachweislich zu dokumentieren.

Abbildung 3.11: Zuweisungsdocument Ausbildung

3.5 Evaluation der erreichten Ergebnisse

Die Umsetzung des Refactorings der Mailing-Komponente hat zu einer zeitgemäßen und effizienten Lösung geführt, die den Anforderungen der BWB gerecht wird. In diesem Abschnitt werden die durchgeführten Änderungen evaluiert, um die Effektivität und den Mehrwert des Refactorings von objektorientiertem Code zu bewerten.

„html-pdf“ Paket ersetzen und BWB-Design implementieren:

Die ursprüngliche Problemstellung des veralteten „html-pdf“ Pakets wurde erfolgreich durch die Integration von „puppeteer“ gelöst. Dadurch entspricht die Komponente den aktuellen Sicherheitsstandards der BWB. Zudem wurde das BWB Corporate Design konsistent angewandt. Damit erfüllt das „Mailing-Modul“ allen Grundanforderungen.

Zusätzliche Anforderungen hinzufügen:

Die Erfüllung der zusätzlichen funktionalen Anforderungen seitens des Personalmanagements verdeutlicht die Flexibilität der implementierten Lösung. Der Anwendungsbereich wurde erfolgreich durch die Trennung von Stuzubis in unterschiedliche Zuweisungsdokumente erweitert. Auch die Anpassung an abteilungsspezifische Bedürfnisse ist in der aktuellen Version möglich.

Clean Code-Regeln berücksichtigen:

Die konsequente Anwendung der Clean Code-Prinzipien, wie zum Beispiel das DRY-Prinzip, hat zur Schaffung leicht verständlichen Codes beigetragen. Die einzelnen Methoden innerhalb des „mailing.utils“-Ordners sorgen für eine Reduktion von Code-Wiederholungen und erleichtern weitere Refactorings.

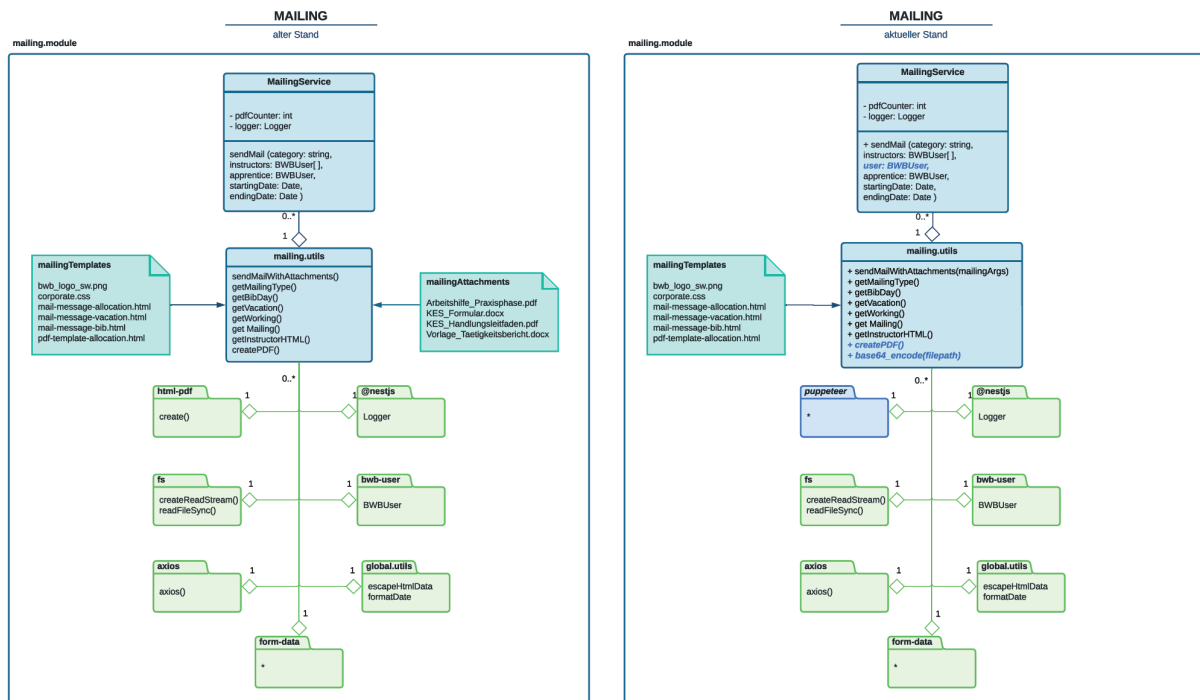


Abbildung 3.12: Vergleich der Klassendiagramme von der „mailing“-Komponente

Der Vorher-Nachher-Vergleich im dargestellten Klassendiagramm zeigt die Veränderungen in der Programmstruktur, wobei diese blau markiert sind. Auffallend ist die minimale Veränderung im Programmaufbau. Die OOP und die Clean Code-Regeln spielen dazu eine Schlüsselrolle im Refactoring. So wurde der Code gemäß des DRY-Prinzips in objektorientierte Methoden getrennt. Dadurch wird jederzeit eine einfache Anpassung und Erweiterung für alle Entwickler:innen ermöglicht.

4 Fazit

Diese wissenschaftliche Arbeit hat es sich zum Ziel gesetzt, die „mailing“-Komponente und damit die Zuweisungserstellung der StAP zu optimieren. Dabei wurden die Konzepte der OOP und des Refactorings angewandt. Die im theoretischen Teil herausgearbeiteten Vorteile der OOP für das Refactoring konnten auch in der praktischen Umsetzung genutzt werden. Durch die eindeutige Trennung in einzelne Klassen und Methoden wurde die Identifikation von Code Smells erleichtert und eine zielgerichtete und schrittweise Überarbeitung ermöglicht. Die hohe Modularität unterstützte die Einführung neuer Methoden, wie die Umsetzung des BWB-Designs oder die Erfüllung zusätzlicher Anforderungen des Personalmanagements. Insgesamt ist die „mailing“-Komponente nun universeller durch die angepassten Templates und Anhänge. Zudem wurde die Sicherheitslücke des Pakets behoben.

Die enge Verknüpfung von OOP und Codeverbesserung durch Refactoring stellt somit aus Sicht der Autorinnen eine optimale Grundlage für eine effektive und nachhaltige Softwareentwicklung dar. Die erfolgreiche Umsetzung der Anforderungen und die positiven Veränderungen im Klassendiagramm bestätigen diese Ansicht und zeigen die Bedeutung der Prinzipien für die Praxis.

Abbildungsverzeichnis

2.1	Refactoringprozess	7
2.2	OOP und Refactoring	8
3.1	Client-Server-Modell	9
3.2	Ordnerstruktur der „mailing“-Komponente	11
3.3	Prozesschaubild „mailing“	11
3.4	Zusammenfassung Methoden aus „mailing.utils“	13
3.5	Quellcode von „createPDF()“	16
3.6	Quellcode von der „createPDF()“ CSS Ergänzung	17
3.7	Quellcode von „base64Encode()“	17
3.8	Quellcode von „createPDF()“	18
3.9	Quellcode von Adressatänderung	19
3.10	Zuweisungsdocument Duales Studium	20
3.11	Zuweisungsdocument Ausbildung	21
3.12	Vergleich der Klassendiagramme von der „mailing“-Komponente	23

Quellenverzeichnis

- [AGoD] AG, Business S.: *Was ist ein Framework?* <https://bsh-ag.de/it-wissensdatenbank/framework/>. Version: o.D.. – [Online; Stand 18. August 2023]
- [Aug18] AUGSTEN, Stephan: Was ist OOP? In: *Dev-Insider* (2018), Sep. <https://www.dev-insider.de/was-ist-oop-a-677737/>. – [Online; Stand 19. August 2023]
- [Ben14] BENDEL, Günther: *Grundkurs Verteilte Systeme: Grundlagen und Praxis des Client-Server und Distributed Computing*. Wiesbaden, 2014
- [Ben15] BENDEL_U.A., Günther: *Masterkurs Parallele und Verteilte Systeme: Grundlagen und Programmierung von Multicore-Prozessoren, Multiprozessoren, Cluster, Grid und Cloud*. Wiesbaden, 2015. – ISBN 978 3 8348 1671 9
- [Com22] COMPUTERWEEKLY.DE, Redaktion: *Objektorientierte Programmierung (OOP)*. <https://www.computerweekly.com/de/definition/Objektorientierte-Programmierung-OOP>. Version: 2022. – [Online; Stand 14. August 2023]
- [Gui18] GUIDE, IONOS D.: *UML – eine grafische Modellierungssprache*. <https://www.ionos.de/digitalguide/websites/web-entwicklung/uml-modellierungssprache-fuer-objektorientierte-programmierung/>. Version: 2018. – [Online; Stand 14. August 2023]
- [Gui22a] GUIDE, IONOS D.: *Was ist das Client-Server-Modell?* <https://www.ionos.de/digitalguide/server/knowhow/client-server-modell/>. Version: 2022. – [Online; Stand 18. August 2023]
- [Gui22b] GUIDE, IONOS D.: *Was ist objektorientierte Programmierung (OOP)?* <https://www.ionos.de/digitalguide/websites/web-entwicklung/objektorientierte-programmierung-oop/>. Version: 2022. – [Online; Stand 14. August 2023]

- [IDE16] ISONG, Bassey ; DLADLU, Nosipho ; ELE, Bassey: Object-Oriented Code Metric-Based Refactoring Opportunities Identification Approaches: analysis, 2016
- [Mar13] MARTIN, Robert C.: Clean Code - Refactoring, Patterns, Testen und Techniken für sauberen Code: Deutsche Ausgabe. (2013). ISBN 9783826696381
- [Mar21] MARCBACHMANN.: *Releases · marcbachmann/node-html-pdf*. <https://github.com/marcbachmann/node-html-pdf/releases>. Version: 2021. – [Online; Stand 17. August 2023]
- [NPM23] NPMJS: *Axios*. [npm.https://www.npmjs.com/package/axios](https://www.npmjs.com/package/axios). Version: 2023. – [Online; Stand 17. August 2023]
- [Pup23a] PUPPETEER: *GitHub - Puppeteer/puppeteer: Node.js API for Chrome*. [GitHub-Puppeteer/puppeteer:Node.jsAPIforChrome](https://github.com/Puppeteer/puppeteer). Version: 2023. – [Online; Stand 17. August 2023]
- [Pup23b] PUPPETEER.: *Page.addStyleTag() method*. <https://pptr.dev/api/puppeteer.page.addstyletag>. Version: 2023. – [Online; Stand 17. August 2023]
- [Pup23c] PUPPETEER.: *Page.emulateMediaType() method*. <https://pptr.dev/api/puppeteer.page.emulatemediatype>. Version: 2023. – [Online; Stand 17. August 2023]
- [Pup23d] PUPPETEER.: *PuppeteerNode Class*. <https://pptr.dev/api/puppeteer.puppeteernode>. Version: 2023. – [Online; Stand 17. August 2023]
- [RG23a] REFACTORING-GURU: *Add Parameter*. <https://refactoring.guru/add-parameter>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23b] REFACTORING-GURU: *Bloaters*. <https://refactoring.guru/refactoring/smells/bloaters>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23c] REFACTORING-GURU: *Change Preventers*. <https://refactoring.guru/refactoring/smells/change-preventers>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23d] REFACTORING-GURU: *Comments*. <https://refactoring.guru/smells/comments>. Version: 2023. – [Online; Stand 15. August 2023]

- [RG23e] REFACTORING-GURU: *Composing Methods*. <https://refactoring.guru/refactoring/techniques/composing-methods>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23f] REFACTORING-GURU: *Couplers*. <https://refactoring.guru/refactoring/smells/couplers>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23g] REFACTORING-GURU: *Dead Code*. <https://refactoring.guru/smells/dead-code>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23h] REFACTORING-GURU: *Dealing with Generalization*. <https://refactoring.guru/refactoring/techniques/dealing-with-generalization>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23i] REFACTORING-GURU: *Dispensables*. <https://refactoring.guru/refactoring/smells/dispensables>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23j] REFACTORING-GURU: *Divergent Change*. <https://refactoring.guru/smells/divergent-change>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23k] REFACTORING-GURU: *Extract Method*. <https://refactoring.guru/extract-method>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23l] REFACTORING-GURU: *Incomplete Library Class*. <https://refactoring.guru/smells/incomplete-library-class>. Version: 2023. – [Online; Stand 16. August 2023]
- [RG23m] REFACTORING-GURU: *Introduce Foreign Method*. <https://refactoring.guru/introduce-foreign-method>. Version: 2023. – [Online; Stand 16. August 2023]
- [RG23n] REFACTORING-GURU: *Moving Features between Objects*. <https://refactoring.guru/refactoring/techniques/moving-features-between-objects>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23o] REFACTORING-GURU: *Object-Oriented Abusers*. <https://refactoring.guru/refactoring/smells/oo-abusers>. Version: 2023. – [Online; Stand 15. August 2023]


- [RG23p] REFACTORING-GURU: *Organizing Data*. <https://refactoring.guru/refactoring/techniques/organizing-data>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23q] REFACTORING-GURU: *Refactoring*. <https://refactoring.guru/refactoring>. Version: 2023. – [Online; Stand 13. August 2023]
- [RG23r] REFACTORING-GURU: *Remove Parameter*. <https://refactoring.guru/remove-parameter>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23s] REFACTORING-GURU: *Simplifying Conditional Expressions*. <https://refactoring.guru/refactoring/techniques/simplifying-conditional-expressions>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23t] REFACTORING-GURU: *Simplifying Method Calls*. <https://refactoring.guru/refactoring/techniques/simplifying-method-calls>. Version: 2023. – [Online; Stand 15. August 2023]
- [RG23u] REFACTORING-GURU: *Substitute Algorithm*. <https://refactoring.guru/substitute-algorithm>. Version: 2023. – [Online; Stand 15. August 2023]
- [Scha] SCHOOLS, W3: *CSS Introduction*. https://www.w3schools.com/Css/css_intro.asp. – [Online; Stand 18. August 2023]
- [Schb] SCHOOLS, W3: *CSS Media Queries*. https://www.w3schools.com/css/css3_mediaqueries.asp. – [Online; Stand 18. August 2023]
- [Schc] SCHOOLS, W3: *HTML Introduction*. https://www.w3schools.com/html/html_intro.asp. – [Online; Stand 18. August 2023]
- [TypoD] TYPESCRIPT: *TypeScript*. <https://www.typescriptlang.org/>. Version: o.D.. – [Online; Stand 18. August 2023]
- [VitoD] VITE: *Getting Started*. <https://vitejs.dev/guide/>. Version: o.D.. – [Online; Stand 18. August 2023]

- [Was21a] WASSERBETRIEBE, Berliner: *Client (Frontend)*. <https://aquanet.berlinwasser.de/confluence/pages/viewpage.action?pageId=711294977>. Version: 2021. – [Intern; Stand 18. August 2023]
- [Was21b] WASSERBETRIEBE, Berliner: *Datenbank*. <https://aquanet.berlinwasser.de/confluence/display/SAPS/Datenbank>. Version: 2021. – [Intern; Stand 18. August 2023]
- [Was21c] WASSERBETRIEBE, Berliner: *Server*. <https://aquanet.berlinwasser.de/confluence/display/SAPS/Server>. Version: 2021. – [Intern; Stand 18. August 2023]
- [Was22a] WASSERBETRIEBE, Berliner: *Coding Guidelines*. <https://aquanet.berlinwasser.de/confluence/display/SAPS/Coding+Guidelines>. Version: 2022. – [Intern; Stand 16. August 2023]
- [Was22b] WASSERBETRIEBE, Berliner: *Coding Guidelines*. <https://aquanet.berlinwasser.de/confluence/pages/viewpage.action?spaceKey=SAPS&title=Coding+Guidelines>. Version: 2022. – [Intern; Stand 17. August 2023]
- [Was23a] WASSERBETRIEBE, Berliner: *Berliner Wasserbetriebe Style Guide*. <https://bwb.frontify.com/d/dXY1NQej6Sfc/corporate-design-manual#/basics/farbklima>. Version: 2023. – [Online; Stand 17. August 2023]
- [Was23b] WASSERBETRIEBE, Berliner: *Besprechung Design Zuweisungen*. <https://aquanet.berlinwasser.de/confluence/display/SAPS/Besprechung+Design+Zuweisungen>. Version: 2023. – [Intern; Stand 17. August 2023]
- [Was23c] WASSERBETRIEBE, Berliner: *Wir als Arbeitsgeber*. <https://www.bwb.de/de/arbeitgeber.php>. Version: 2023. – [Online; Stand 24. März 2023]

Ehrenwörtliche Erklärungen

Hiermit erkläre ich ehrenwörtlich:


1. dass ich meine Studienarbeit selbständig verfasst habe,
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe
3. und dass ich meine Studienarbeit bei keiner anderen Prüfung vorgelegt habe.

20.08.2023 

Datum/Unterschrift der/des Studierenden

Hiermit erkläre ich ehrenwörtlich:

1. dass ich meine Studienarbeit selbständig verfasst habe,
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe
3. und dass ich meine Studienarbeit bei keiner anderen Prüfung vorgelegt habe.

 20.08.2023
.....

Datum/Unterschrift der/des Studierenden