Abschlussprüfung Sommer 2024

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Importprofil-Tool

Verwaltung von Profilen für den Datenimport von Schülerdaten

Abgabedatum: Berlin, den 05.06.2024

Prüfungsbewerber:

Marco Garagna

Wigandstaler Straße 37

13086 Berlin

Betriebliches Praktikum:

ASCI GmbH

Alt-Friedrichsfelde 5A

10315 Berlin

Inhaltsverzeichnis

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

CD..... *Continuous Deployment*
CI ..... *Continuous Integration*
CSV..... *Comma Separated Values*
ERM ..... *Entity Relationship Model*
GUI..... *Graphical User Interface*
HTML ..... *Hypertext Markup Language*
JavaEE..... *Java Enterprise Edition*
JPA ..... *Jakarta Persistence API*
JSF..... *Jakarta Server Faces*
JSP ..... *Java Server Page*
MVC ..... *Model View Controller*
ORM..... *Object Relational Mapping*
SCM ..... *Source Code Management*
SQL..... *Structured Query Language*
SVN..... *Subversion*
UML ..... *Unified Modeling Language*
XML..... *Extensible Markup Language*

1 Einleitung

1.1 Projektumfeld

ASCI Systemhaus GmbH hatte vor einigen Jahren die webbasierte Anwendung SyABO für die Verwaltung von Fahrkartenabonnenten und Schülerverkehrsdaten im ÖPNV entwickelt. Dieses Programm ist bei verschiedenen Verkehrsunternehmen in Deutschland im Einsatz.

Jedes Verkehrsunternehmen hat mehrere Datenlieferanten, die im SyABO als Vertragspartner-Objekte dargestellt sind. Um einen problemlosen Datenaustausch durch die SyABO-Schnittstelle zu gewährleisten, muss jedes Verkehrsunternehmen mit jedem Vertragspartner bestimmte Konventionen vereinbaren. Die Einigung über diese Konventionen wurde unter anderem aufgrund der Eigenschaften der CSV-Datei getroffen.

1.2 Projektziel

Nach der Implementierung dieser neuen Funktionalität kann der Anwender jedem Vertragspartner ein eigenes konfigurierbares Importprofil zuweisen.

Die Erstellung der neuen Funktion besteht aus drei Teilen:

1. Die Entwicklung einer Benutzeroberfläche (UI) für die Profilerstellung und zur Verwaltung der Profile.

2. Die Erweiterung der bestehenden Benutzeroberfläche zur Verwaltung der Vertragspartner, um eine obligatorische Zuordnung eines Importprofils zum Vertragspartner zu ermöglichen.

3. Der dritte Teil ist die Überarbeitung des bestehenden Importassistent-Servlets, der für den Import der CSV-Datei selbst verantwortlich ist. Dieser muss angepasst und erweitert werden, um die Funktion korrekt zu implementieren und die Benachrichtigungen für Benutzer zu behandeln. Eine Benachrichtigung des Benutzers ist zum Beispiel dann erforderlich, wenn ein Import für einen Vertragspartner erfolgen soll, dem noch kein Importprofil zugewiesen wurde.

Jedes Importprofil enthält alle erforderlichen Informationen, die vom Importassistenten benötigt werden, um die Daten einzulesen und aufzubereiten. Dadurch, dass jeder Vertragspartner ein für ihn definiertes Importprofil nutzt, kann der Aufbau der CSV-Dateien von dem Vertragspartner bestimmt werden, solange die für das Programm erforderlichen Mindestdaten darin enthalten sind. Das spart Abstimmungsaufwand und auf der Seite der Vertragspartner den Anpassungsaufwand an die von SyABO vorgegebenen Strukturen.

1.3 Projektbegründung

Der Import der Daten erfolgt derzeit anhand von Schlüsselwörtern in der CSV-Datei, die einmalig im Programm festgelegt wurden. Dadurch ist eine Abstimmung des

Programmnutzers mit den unterschiedlichen Einrichtungen über das Format der CSV-Datei erforderlich. Um den Einsatz des Programms künftig flexibler zu gestalten, soll das Programm mit unterschiedlichen, auf den jeweiligen Datenlieferanten bezogenen Profilen für den Import der Daten aus den CSV-Dateien arbeiten, sodass beliebige Schlüsselwörter in den CSV-Dateien verwendet werden können.

1.4 Projektschnittstellen

Für das Deployment der gesamten Anwendung bleibt der Jenkins CI Server als Schnittstelle bestehen, der die Anwendung auf einem internen Server veröffentlicht. Eine zweite Schnittstelle ist spezifisch für das vorhandene Feature der Importassistent für CSV-Importe. Die Endbenutzer der Anwendung sind Mitarbeiter der Administrationsabteilung.

Während der Entwicklung der Funktionalitäten und Benutzeroberflächen wurde regelmäßiges Feedback von den Projektbetreuern eingeholt. Dies ermöglichte eine flexible Anpassung an die Anforderungen und könnte die Einführungsphase verkürzen. Die Benutzeroberflächen stehen allen Mitarbeitern mit den entsprechenden Rollen sofort im Anwendungsmenü zur Verfügung, ohne dass eine separate Installation erforderlich ist.

1.5 Projektabgrenzung

Die zeitliche Begrenzung auf 80 Stunden wurde von die IHK Berlin vorgegeben.

Bezüglich der in 1.2 Projektziel genannte Punkte: Punkt eins ist ein wesentlicher Bestandteil der neuen Implementierung dieses Projekts, nämlich das Hauptelement. Punkte zwei und drei sind Anpassungen bzw. Erweiterungen bestehender Komponenten von SyABO. Sie sind notwendig, damit Punkt 1 überhaupt funktionieren kann. Im Gegensatz zu Punkt 1 sind die Datenmodelle und Implementierungen bereits vorhanden.

2 Projektplanung

2.1 Projektphasen

Eine detailliertere Zeitplanung ist in Tabelle 5 in Anhang A1 zu sehen.

Projektphase	Geplante Zeit
Analyse	9 h
Entwurf	12 h
Implementierung	46 h
Abnahme	2 h
Einführung	2 h
Dokumentation	9 h
Gesamt	**80 h**

Tabelle 1: Grobe Zeitplanung

2.2 Abweichungen vom Projektantrag

Die Spaltennummern werden als Attribute in String-Variablen gespeichert, die sie dem Datenmodell "Schuelerdatenimportprofilspalte" zugeordnet sind. Ein "Schuelerdatenimportprofil" kann in seinem untergeordneten Datenmodell "Schuelerdatenimportprofilspalte" nur eindeutige Paare von Spaltenschlüsseln und Spaltennamen enthalten.

Auf der Benutzeroberfläche, wie im Anhang A4 auf Seite v dargestellt, befindet sich ein Textfeld mit dem Label "Spaltenname", in das der Benutzer alphanumerische Werte eingeben kann. Diese Werte folgen eng den Namenskonventionen, die während des ersten Kundengesprächs im Lastenheft grob zusammengefasst wurden.

Im Laufe des Projekts wurden die Anforderungen des Kunden leicht modifiziert, insbesondere hinsichtlich der Regelungen für Namenskonventionen und die Spalten, auf die sich die Regeln auswirken sollen.

Die regulären Ausdrücke bestimmen beispielsweise, mit welchen Zeichen ein Wert beginnen muss (z.B. ein Buchstabe) und welche Sonderzeichen erlaubt sind.

Beispielcode ist im Screenshoot A10 Seite viii zu sehen.

2.3 Ressourcenplanung

Anschließend wurden verwendete Ressourcen im Anhang A.2: Verwendete Ressourcen auf Seite ii aufgelistet. Neben allen Hard- und Softwareressourcen wurde auch das Personal aufgenommen. Im Hinblick auf anfallende Kosten wurde darauf geachtet, dass die Nutzung der Software kostenfrei ist oder die Lizenzen dem Unternehmen bereits zur Verfügung stehen. Dadurch konnten die Projektkosten auf einem Minimum gehalten werden. Unter anderem wurde für die Modellierung unterschiedlicher UML-Diagramme diagrams.net und als Anwendungsserver Apache Tomcat genutzt.

2.4 Entwicklungsprozess

Die Durchführung des Projektes wird testgetrieben durch kontinuierliches Review mit einem Projektbetreuer und Stakeholder, um sicherzustellen, dass alle Projektparteien mit dem aktuellen Entwicklungsstand des Features und dessen Funktionalität einverstanden sind, bevor eine Projektphase als abgeschlossen gilt.

Das erweiterte Wasserfallmodell ermöglicht es dem Projektteam jedoch, zu einer früheren Phase zurückzukehren, um z.B. nachträglich erfasste Verbesserungen zu berücksichtigen und schließlich Ergebnisse zu erzielen, die allen Anforderungen entsprechen.

3 Analysephase

3.1 Ist-Analyse

Für den Import von Schülerdaten aus CSV-Dateien in die SyABO-Datenbank existiert bereits eine Schnittstelle. Die Funktion, die die Importdaten aufbereitet, nennt sich Importassistent und wurde als Servlet-Komponente implementiert. Die Aufbereitung der Daten aus der CSV-Datei erfolgt anhand von Schlüsselwörtern, die in der ersten Zeile der CSV-Datei stehen. Die für die Verwendung definierten Schlüsselwörter und deren Beziehungen zu den Programmdaten sind in der Datei „importoptionen.properties“ gespeichert.

Auf einer grafischen Benutzeroberfläche kann der Nutzer die vom Programm erkannten Fehler in Datensätzen nachbearbeiten bzw. ergänzen. Nach der Korrektur werden die Daten aus den temporär angelegten Datenbanktabellen in die Arbeitsdatenbanktabellen gespeichert.

Die Anwendung durch Importassistent Servlet UI erlaubt einen Import von Schülerdaten mittel ein CSV-Datei. Daraus ergeben sich folgende Probleme (Ticket im Anhang …. Seite …):

- mehrere Datenlieferanten, die mit jeweilig eigenen Formaten die Dateien liefern können.
- nicht flexibel genug.
- mehrere Datenlieferanten, unterschiedlichen Spaltenbezeichnungen, Zeichensatz
- Es sind Anpassungen erforderlich, um die Importdateien in eine standardisierte Form zu bringen.

3.2 Wirtschaftlichkeitsanalyse

Aus der Behebung der in 3.1 Ist-Analyse genannten Probleme resultieren neben technischen Vorteilen auch reduzierte Verwaltungszeiten für die Führungskräfte. Der daraus entstehende finanzielle Vorteil soll im Folgenden dargelegt werden.

3.2.1 Make or Buy-Entscheidung

Da es sich beim ASCI-Systemhaus um kritische Infrastruktur des ÖPNV Deutschland handelt, die strengen Datenschutz- und Sicherheitsrichtlinien unterliegt, müsste eine eingekaufte Softwarelösung von Drittherstellern vor dem Einsatz sehr gründlich auf potenzielle Schwachstellen und Sicherheitsrisiken geprüft werden. Diese Prüfung würde weitaus mehr Kosten verursachen, als die eigenständige Entwicklung. Eine Eigenproduktion ist daher die sinnvollere Option.

3.2.2 Projektkosten

Die Projektkosten setzen sich maßgeblich aus den Personalkosten, sowohl des Auszubildenden wie auch der beteiligten Mitarbeiter, sowie den Kosten für die Bereitstellung der benötigten Arbeitsmaterialien und des Arbeitsplatzes zusammen. Dabei kann für die Mitarbeiter ein Stundensatz von *40 EUR*. Zur Ermittlung des ungefähren Stundensatzes des Auszubildenden wurde folgende Rechnung genutzt:

$$8\frac{\text{h}}{\text{Tag}} \cdot 220\frac{\text{Tage}}{\text{Jahr}} = 1.760\frac{\text{h}}{\text{Jahr}}$$

$$1.000\frac{€}{\text{Monat}} \cdot 13,3\frac{\text{Monate}}{\text{Jahr}} = 13.300\frac{€}{\text{Jahr}}$$

$$\frac{13.300\frac{€}{Jahr}}{1.760\frac{\text{h}}{\text{Jahr}}} \approx 7,56\frac{€}{\text{h}}$$

Es ergibt sich also ein Stundensatz von *7,56 EUR*. Die Durchführungszeit des Projekts beträgt 80 Stunden. Für die Nutzung von Ressourcen[1] wird ein pauschaler Stundensatz von *15 EUR* angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundensatz von *25 EUR* angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 2 und sie betragen insgesamt *2.739,20 EUR*.

Vorgang	Zeit	Kosten / Stunde	Kosten
Entwicklung	80 h	*7,56 € + 15 € = 22,56 €*	*1.804,80 €*
Fachgespräch	3 h	*25 € + 15 € = 40,00 €*	*120,00 €*
Genehmigung	3 h	*25 € + 15 € = 40,00 €*	*120,00 €*
Abnahme	1 h	*25 € + 15 € = 40,00 €*	*40,00 €*
		Gesamt	***2.084,80 €***

Tabelle 2: Kostenaufstellung

[1] Räumlichkeiten, Arbeitsplatzrechner etc.

3.2.3 Amortisationsdauer

Bei einer Zeiteinsparung von ca. drei Stunden pro Monat, das entspricht 36 Stunden im Jahr. Daraus ergibt sich folgende Amortisationsgleichung:

$$\text{Amortisationszeit} = \frac{2.084{,}80\ €}{36\ \frac{h}{Jahr} * 40\ \frac{€}{h}} \approx 1{,}448\ Jahre\ \approx\ 17{,}5\ Monate$$

Nach ungefähr 17,5 Monaten sind die Kosten für die Entwicklung des neuen Features von den durch sie entstehenden Einsparungen gedeckt. Ein entsprechendes Diagramm liegt im Anhang A3 Amortisationsdiagramm auf S. iii vor.

3.3 Nutzwertanalyse

Neben den finanziellen Vorteilen überwiegen vor allem die nicht-finanziellen Vorteile. Diese ergeben sich aus der Beseitigung der in der Ist-Analyse (Kapitel 3.1) identifizierten Probleme sowie aus der Bewertung in der Entscheidungsmatrix gemäß Kapitel 4.2 (Architekturdesign).

3.4 Anwendungsfälle

Bei einem Treffen mit den Projektbeteiligten wurde ein Anwendungsfalldiagramm entwickelt, das die Hauptfunktionen der zu entwickelnde Anwendung darstellt. Dieses unter A4 Use-Case-Diagramm auf S. iv aufgeführte Diagramm kann des Weiteren zur Einteilung der Implementierung in einzelne Features herangezogen werden.

3.5 Qualitätsanforderungen

Um eine möglichst hohe Qualität der Importprofil-Tool sicherzustellen, wurden die einzelnen Funktionen mittels Komponenten- und Integrationstests überprüft. Hierdurch konnte zunächst die Korrektheit der einzelnen Komponenten bestätigt werden und weitergehend das Zusammenspiel mit anderen, voneinander abhängigen Komponenten.

In der weiter fortgeschrittenen Entwicklungsphase wurden Systemtests genutzt, um die gesamte Funktion zu überprüfen. Durch die Bereitstellung einer Testdatenbank des Kunden waren realistische Daten zum Testen vorhanden.

Die in Abschnitt 6 beschriebene Abnahmephase durch den Kunden stellt den Abnahmetest dar. Durch diese Teststufe konnte noch einmal die Korrektheit der Funktion in einer Kopie einer Produktiv-Datenbank des Kunden bestätigt werden.

3.6 Lastenheft/Fachkonzept

Das unter A2 Lastenheft (Auszug) auf S. ii aufgeführte Lastenheft entstand als Resultat aus der Analysephase in Kooperation mit dem Auftraggeber des Projekts und bildet die Grundlage für die nachfolgende Entwurfsphase des Projekts.

4 Entwurfsphase

4.1 Zielplattform

Bei der Auswahl der Zielplattform für das Projekt haben sich mehrere Bereiche ergeben, die berücksichtigt werden müssen.

Die Geschäftslogik im Backend wird mit *JavaEE* implementiert. Dies geschieht aus mehreren Gründen: Java ist die vorherrschende Programmiersprache im Unternehmen und es stehen alle erforderlichen Entwicklungstools zur Verfügung.

Zur Kommunikation mit der Datenbank wird JPA verwendet, ein Framework, das eine einfache Möglichkeit bietet, objektorientierte Datenmodelle in einer Datenbank zu speichern. Als Datenbank wird PostgreSQL verwendet, da sie eine robuste und weit verbreitete Open-Source-Datenbank ist. Die Datenbankserver sind bei jedem Kunden des ASCI-Systemhauses gehostet, um Datenschutzprobleme zu vermeiden. Der Server, auf dem das Backend gehostet wird, ist ein Apache Tomcat Server.

Für das Frontend steht das JSP, JSF und Spring Framework zur Verfügung, das mit entsprechenden Frontend-Bibliotheken verwendet wird, um HTML-Code abzubilden und die Benutzeroberfläche zu entwickeln.

Die Benutzeroberfläche wird hauptsächlich für die Nutzung mit dem Browser Firefox optimiert, da dies der Standardbrowser im Unternehmen ist.

4.2 Architekturdesign

Die Umsetzung des Projektes soll auf Basis des MVC-Konzepts erfolgen. Dieses sieht eine Trennung der Anwendung in das Datenmodell (Model), die Darstellung der Daten (View) und die Steuerung des Programmes (Controller) vor.

Diese Teilung erfolgt, um die spätere Bearbeitung und Auswertung der Daten sowie des Programmes zu vereinfachen und diese unabhängig voneinander zu ermöglichen. So können die jeweiligen Komponenten mit geringem Aufwand angepasst oder sogar ausgetauscht werden, ohne dass die anderen Bestandteile davon betroffen sind. Des Weiteren wird die Übersichtlichkeit und Wartbarkeit des Quellcodes durch die Nutzung des MVC-Konzepts verbessert.

Die Rolle der View soll im zu entwickelnden Programm von der im Frontend eingesetzten Webapplikation eingenommen werden. Diese ist lediglich für das Ausgeben von Daten aus dem Backend und das Annehmen von Benutzereingaben zuständig und kann daher jederzeit ausgetauscht werden.

Für den Controller werden Java Klassen im Backend eingesetzt, welche auf die Anfragen des Frondendes reagieren. Diese sind für die Berechnung beziehungsweise Abfrage der Ausgabeparameter sowie die Speicherung der vom User eingegebenen Daten im Model zuständig.

Basierend auf den Kriterien in Tabelle 3 wurde das Java Framework Spring als Implementierungsplattform für die Anwendung ausgewählt.

Eigenschaft	Gewichtung	*Struts*	*Hibernate*	*Spring*	Eigenentwicklung
Dokumentation	5	4	3	5	0
Reengineering	3	4	2	5	3
Generierung	3	5	5	5	2
Testfälle	2	3	2	3	3
Standardaufgaben	4	3	3	3	0
Gesamt	17	65	52	73	21
Nutzwert		**3,82**	**3,06**	**4,29**	**1,24**

Tabelle 3: Entscheidungsmatrix

4.3 Entwurf der Benutzeroberfläche

Die Hauptanforderung an alle Benutzeroberflächen der Anwendung ist, dass sie im gleichen Design, Look and Feel wie die bestehenden "Standard"-Seiten der Anwendung gestaltet werden. Bei den Importassistent Servlet-Elementen mussten keine GUI-Anpassungen vorgenommen werden, während auf der Vertragspartner Verwaltung lediglich ein Dropdown-Menü zur Auswahl hinzugefügt werden musste.

Die vollständige Neugestaltung der Hauptseite bedeutet in erster Linie, dass ein umfangreiches Design in enger Abstimmung mit dem zuständigen Berater entwickelt wird, wobei Skizzen als Hilfsmittel verwendet werden.

Durch iteratives Vorgehen und kurze Feedback-Meetings wurde eine nahezu endgültige Lösung vereinbart, die es ermöglichte, die erforderlichen GUI-Elemente zu identifizieren.

Diese sind in Abbildung 8: Benutzeroberfläche Mockup Seite v zu sehen.

Im Rahmen der Entwurf Konzept wurde insbesondere darauf geachtet, die Art und Anzahl der einzubindenden Eingabe- und Ausgabefelder in den jeweiligen Menüreitern sowie die Umsetzung des Corporate Designs des ASCI-Systemhauses entsprechend zu gestalten.

4.4 Datenmodell

Das aktuelle Datenmodell enthält bereits die Entitäten "Vertragspartner" und "Einrichtung", wie in Anhang A6 ERM auf Seite V dargestellt. Um die Beziehung zwischen diesen beiden Entitäten darzustellen, wird ein Fremdschlüssel, die "SchuelerdatenimportprofilID", zur "Vertragspartner"-Tabelle hinzugefügt. Das "Schuelerdatenimportprofil" bildet den Kern des Datenmodells für diese neue Funktion.

Ein Schulerdatenimportprofil kann einem aktiven Vertragspartner zugeordnet. Ein Profil kann jedoch auch ohne Zuordnung zu einem bestimmten Vertragspartner existieren.

Jedes Schülerdatenimportprofil ist einer Liste von Spalten und Zuständen zugeordnet, wie im Datenbankschema in Anhang A6, Tabellenmodell auf Seite vi dargestellt.

Die Zustände sind in der Tabelle „Schuelerdatenimportprofilstaus“ definiert. Jeder Status hat ein internes Identifikationsattribut namens "Status" (nicht die natürliche ID der Tabelle), das als *Integer* gespeichert wird, und eine Farbe, die als *Textvariable* im Hexadezimalformat gespeichert wird.

Die Tabelle "Spalte" hat wie die Tabelle "Status" eine interne Referenz, die durch ein *Integer-Attribut* mit dem Namen "Spalte" und ein Attribut für die Spaltenbezeichnung als *Text* dargestellt wird.

4.5 Geschäftslogik

Ein Aktivitätsdiagramm, das die Anforderungen des Projektleiters und des Auftraggebers darstellt, wurde in der Entwurfsphase erstellt und im Feedback-Gespräch für die meisten Funktionalitäten der Importprofilseite weitgehend gebilligt.

Der Entwurf der Logik der Datenmodelle, die dem Schülerdatenimportprofil untergeordnet sind, schuelerdatenimportprofilspalte[2] und schuelerdatenimportprofilstatus[3] des Schülerdatenimportprofils, hängt von der Datenbanklogik ab.

[2] Klassennamen müssen nicht unbedingt den Regeln der deutschen Grammatik folgen.
[3] Klassennamen.

Zum besseren Verständnis der untergeordneten Struktur und der Beziehung zwischen den Klassen ist ein Klassediagramm in Anhang Axx Seite xx verfügbar. Dieses Diagramm war natürlich die Grundlage für die Entwurfsphase und die Implementierung der erforderlichen Datenstrukturen.

Die Klassen schuelerdatenimportprofilspalte und schuelerdatenimportprofilstatus vor der Implementierung in Java werden in der Datenbank mit bestimmten sogenannten „Constraints“ hinterlegt.

In der Datenbank-Fachsprache stellen solche „Constraints“ die Eindeutigkeit von Attributpaaren sicher, d.h. sie müssen als UNIQUE deklariert werden. Leider ist ein Auszug aus dem Datenbank-Script zum Hinzufügen der oben genannten Tabelle aus Sicherheitsgründen nicht in dieser Dokumentation enthalten.

Dennoch habe ich diese „Constraints“ in einem vereinfachten Auszug aus dem Datenbankschema in Anhang xx Seite xx dargestellt.

Ein Klassendiagramm, welches die Klassen der Anwendung und deren Beziehungen untereinander darstellt, kann im Anhang A14 eingesehen werden.

Die Klassendiagramm in A7 zeigt die Klassenstruktur des Features.

Die Aktivitätsdiagramm in Anhang **Fehler! Verweisquelle konnte nicht gefunden werden.** zeigt den grundsätzlichen Ablauf beim Einlegen eines Importprofil.

4.6 Maßnahmen zur Qualitätssicherung

Zur Sicherstellung der Qualität ist eine Testphase geplant. Hierbei werden vom Projektbetreuer verschiedene Testfälle erstellt, anhand derer der Berater die richtige Funktionalität testen und bestätigen kann. Findet der Berater hierbei Fehler, werden diese anschließend behoben.

Des Weiteren testet der Entwickler während der Entwicklung regelmäßig die Anwendung, um die Funktionalität einzelner Methoden zu gewährleisten.

Programmfehler werden vom IDE und Importassistent-Servlet automatisch geloggt, sowohl im Frontend Server für die Benutzeroberflächenlogik als auch im Backend Server für die Geschäftslogik. Somit lassen sich auch Fehler, die erst nach der Testphase auftauchen, nachträglich gut beheben.

4.7 Pflichtenheft/Datenverarbeitungskonzept

Im Entwurfsphase wurde ein Pflichtenheft erstellt. Ein Auszug für das auf dem Lastenheft (siehe Kapitel 3.5) aufbauende Pflichtenheft ist im Anhang A5 Auszug Pflichtenheft Seite vii zu finden. Das Pflichtenheft enthält detaillierte Pläne für die Umsetzung der fachlichen und technischen Anforderungen an die Anwendung aus Sicht des Auftragnehmers. Es dient als Leitfaden während der Implementierungsphase.

5 Implementierungsphase

5.1 Implementierung der Datenstrukturen

Ausgehend von den in den vorherigen Kapiteln erwähnten Artefakten wird zunächst ein SQL-Skript erstellt, um die 3 neuen Tabellen hinzuzufügen und die Tabelle Vertragspartner anzupassen. Diese ist bereits vorhanden und wird nun mit der SchülerdatenimportprofilID Zuweisung belegt.

Dieses erste Skript wird benötigt, um mit dem Rest der Implementierung in Backend zu beginnen. Ein zweites Skript wird benötigt, um die GUI zu testen bzw. relativ komfortabel zu implementieren. So kann man sofort sehen, ob das hinzugefügte GUI-Element problemlos dargestellt werden kann oder *Binding*[4]-Probleme schnell behoben werden können.

Eine zweite SQL-Script dient, wie für andere neue Seite eine Hinzufügung einer neue Menüpunkt. Genau Position der Menüeintrag, Name und andere Eigenschaften nicht Teil die Entwicklung dieser Feature.

5.2 Implementierung der Benutzeroberfläche

Basierend auf den verschiedenen GUI-Mockups und anderen Artefakten, wie z.B. dem Pflichtenheft, wurde eine Liste von GUI-Elementen erstellt. Diese Liste sowie eine Musterseite bildeten den Ausgangspunkt für die Implementierung der Benutzerschnittstelle.

Die Implementierung der „SchuelerdatenimportprofilPage“ kann letztlich in 3 deutlich unterscheidbare Unterelemente unterteilt werden, nämlich: oberer Teil für das Schuelerdatenimportprofil selbst, links blau umrahmter Bereich für die Schuelerdatenimportprofil-Spaltenelemente und rechts ebenfalls blau umrahmter Bereich für die Schuelerdatenimportprofil-Statuselemente. Unterteilung auch in Screenshot Axx Seite xx zu sehen.

Bezüglich die Untere rechts Rahmen der Oberfläche, für die Implementierung der Benutzerschnittstelle wurde eine Funktionalität zur Farbauswahl in das Programm integriert, die es dem Benutzer ermöglicht, mittels einer darauf aufbauenden Komponente, dem sogenannten „ColorPicker“, eine Hintergrundfarbe für jeden Status des spezifischen Studiendatei-Importprofils zuzuweisen. Diese Farben werden dann im Import-Assistenten angezeigt.

Nachdem der Benutzer eine Farbe für einen Status ausgewählt und auf Speichern geklickt hat, werden Sie in der Datenbank in Hexadezimalwerten gespeichert. Der Benutzer wählt jedoch keine hexadezimalen Werte aus, sondern klickt mit der Maus auf ein zusätzliches Popup GUI aus einer Farbpalette, wie im Screenshoot Anhang A10 gezeigt.

Screenshots von Color Picker JavaScript Code befinden sich im Anhang A10.

5.3 Implementierung der Geschäftslogik

CRUD-Methode anhand XML-Datei verwaltet von iBatis ORM für Datenbankaufrufe. Ein Auszug von (Begin, 2006) iBatis Framework Architektur Diagramm befinden sich in Anhang A17 Seite xx.

Als Beispiel für Beschreibung der Geschäft Logik ist die „TransactionSchuelerdatenimportprofil“ Klasse ausgewählt, die für die Datenverarbeitung im Zusammenhang mit dem Import von Schülerdatenprofilen zuständig ist. Diese Klasse ist verantwortlich für das Laden, Löschen und Speichern von Schülerdatenprofilen sowie deren zugehörigen Spalten und Status.

Zu Beginn der Implementierung wird eine Instanz der Klasse erstellt und die erforderlichen DAOs (Data Access Objects)[5] über Setter-Methoden gesetzt. Anschließend werden die Schülerdatenprofile aus der Datenbank geladen, wobei auch die zugehörigen Spalten und Status geladen und den entsprechenden Profilen zugeordnet werden.

[4] Verbindung, definierte Referenz zwischen dem GUI-Element in der JSP-Klasse und der entsprechenden GUI-Logik in der Java Klasse. Die beiden Klassen haben den gleichen Namen, gehören aber zu 2 verschiedenen Paketen und haben unterschiedliche Dateiendungen. Eine ist „.jsp“ und die andere ist „.java“.
[5] Das DAO repräsentiert eine Schnittstelle zur Datenbank in Ihrer Anwendung. Es agiert als Mittler zwischen Ihrer Anwendungslogik und der Datenbank. Das DAO ermöglich daten aus der Datenbank abzurufen, zu speichern oder zu ändern, ohne mit den komplexe internen Datenbank Details sich zu befassen. Es fördert eine saubere und geordnete Struktur in Ihrer Software, indem es die Datenzugriffslogik von anderen Teilen Ihrer Anwendung trennt.

Für das Löschen eines Schülerdatenprofils werden zunächst die zugehörigen Spalten und Status gelöscht, bevor das Profil selbst entfernt wird. Beim Speichern eines Schülerdatenprofils wird zunächst überprüft, ob es sich um ein neues Profil handelt oder ob es bereits in der Datenbank existiert. Entsprechend wird ein Einfügen oder Aktualisieren durchgeführt. Dabei werden auch die zugehörigen Spalten und Status entsprechend aktualisiert.

Die Klasse implementiert somit die logischen Abläufe für das Laden, Löschen und Speichern von Schülerdatenprofilen in der Anwendung und stellt sicher, dass die Daten konsistent in der Datenbank gespeichert werden.

Abkürzung CRUD

Die Klasse `TransactionSchuelerdatenimportprofil` findet sich im Anhang A13.

6 Abnahmephase

Nach Abschluss des Projekts fand ein Fachgespräch mit dem Entwicklungsleiter zur Abnahme statt. Das Projektziel wurde erfolgreich erreicht und entspricht vollständig den Anforderungen des Auftraggebers, was zu seiner vollsten Zufriedenheit führt.

Die Abnahme durch den Kunden verlief durch das iterative Vorgehen bei der Entwicklung sehr vorausschaubar. Der Auftraggeber wurde durch konstante Rücksprachen während der Entwicklung und dem Vorstellen vorläufiger Ergebnisse bereits früh eingebunden. Das Projekt war ihm somit bekannt und alle Funktionen waren vertraut sowie in erwartetem Umfang umgesetzt. Die finale Abnahme war hiermit erfolgreich und der Kunde hat die Zeiterfassung zufrieden entgegengenommen.

In der Diskussion mit dem Kunden wurden weitere neue Funktionen herausgefunden, die jedoch als neue Anforderungen gewertet werden. Es besteht somit Interesse an einer Weiterentwicklung der Importprofil-Tool.

Des Weiteren wurde vereinbart, dass in nächster Zeit regelmäßig Rücksprache über den Einsatz der Lösung gehalten wird. Gegebenenfalls wird es dann noch geringfügige Anpassungen geben.

Ein Auszug aus dem von Projektbetreuer erstellt Testprotokoll ist der Abbildung 14: Auszug Testprotokoll Seite XX zu entnehmen.

7 Einführungsphase

- Welche Schritte waren zum Deployment der Anwendung nötig und wie wurden sie durchgeführt (automatisiert/manuell)?
- Wurden Ggfs. Altdaten migriert und wenn ja, wie?
- Wurden Benutzerschulungen durchgeführt und wenn ja, Wie wurden sie vorbereitet?

8 Dokumentation

Neben der Projektdokumentation wurde auch eine betriebsinterne Entwicklungs- bzw. Funktionsdokumentation für das Firmen-Wiki von ASCI Systemhaus GmbH erstellt (Auszug in Screenshoot AXX Seite XX). In dieser sind die genauen Funktionen und Hinweise zur korrekten Implementation auf einem Kundensystem dokumentiert.

Hierdurch kann die Funktion des Programmes zum Beispiel im Fall von einer Fehlfunktion auch von anderen Mitarbeitern nachvollzogen werden. Außerdem ist es so anderen Kollegen möglich, die Importprofile für Neukunden zu implementieren.

Die Benutzerdokumentation wurde in etwas unkonventioneller Weise behandelt. Statt sie einem bereits vorhandenen Benutzerhandbuch hinzuzufügen oder ein komplett neues Handbuch speziell für diese Funktionen zu erstellen, wurde mehr Wert daraufgelegt, wie sich die Funktionen auf die andere Seite der Anwendung auswirken. Insbesondere wurde die Funktionalität des Importprofil-Tools so gestaltet, dass es sich selbst erklärt (für erfahrene Anwender von SyABO). Alle Meldungen und Fehlerbehandlungsmechanismen befinden sich auf der anderen Seite, wodurch die Benutzerfreundlichkeit gewährleistet ist. Die Sammlung und Gestaltung der Meldungen sind Teil der Benutzerdokumentation, wie im Code-Snippet AXX auf Seite X zu sehen ist.

9 Fazit

9.1 Soll-/Ist-Vergleich

Im Folgenden wird der geplante Zeitbedarf dem tatsächlichen Zeitbedarf gegenübergestellt.

Während der Implementierungsphase wurde festgestellt, dass für die Anpassung der Importassistentenkomponente und den Austausch der alten Logik durch neue Logik (siehe Kapitel 9.2 Lessons Learned) wesentlich mehr Zeit benötigt wurde als ursprünglich geplant. Die zusätzliche Zeitersparnis wurde durch eine effizientere Analyse und Erstellung der Dokumentation erreicht. Wie in Tabelle 4 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

Phase	Geplant	Tatsächlich	Differenz
Analyse	9 h	8 h	-1 h
Entwurf	12 h	12 h	
Implementierung	46 h	49 h	+3 h
Abnahme	2 h	1 h	
Einführung	2 h	1 h	
Dokumentation	9 h	7 h	-2 h
Gesamt	**80 h**	**80 h**	

Tabelle 4: Soll-/Ist-Vergleich

9.2 Lessons Learned

Der Zeitaufwand für den Umbau alter Logik durch neue Funktionalität wurde stark unterschätzt. Wenn es sich um eine monolithische Architekturanwendung handelt, ist zu erwarten, dass, wenn alte Funktionalität entfernt oder angepasst werden muss, viel Zeit, Überlegung und Information darüber, wie die Funktionalität ursprünglich gedacht war, aufgewendet werden muss. Auch die besondere Rolle der Stakeholder Kommunikation im Bereich der Softwareentwicklung wird während der Ausführung des Projektes deutlich. So ist schnell klar, dass ohne ausreichende Rücksprache und gemeinsame Planung kein Produkt entstehen kann, welches den Ansprüchen der Auftraggeber*innen entspricht.

Ebenfalls wird die Bedeutung von intensiver Planung von der gegebenen Projektstruktur betont. In diesem kann das Projekt nach der Meinung des Autors und dessen Ausbilder*innen

als großer Erfolg bezeichnet werden, welcher sowohl einen didaktischen als auch einen praktischen Mehrwert liefert.

9.3 Ausblick

Während der Abnahme schlug der Projektmanager im Rahmen einer zweiten Iteration des Features eine Verbesserung der GUI vor.

Nach kurzen Gesprächen auf Managementebene und im Entwicklungsteam wurde beschlossen, diese Erweiterung im nächsten Iterationszyklus durchzuführen. Es betrifft die Farbkomponenten der Schülerdatenimportprofil-Seite. Die Modifizierung sieht vor, dass jeder Eintrag Zustand nicht mehr in einer Listbox, sondern in einer vertikalen Reihe mit jeweils einem eigenen Color Picker Button angezeigt wird. Mit dieser Version werden die Gestaltungsrichtlinien und die Benutzerfreundlichkeit besser respektiert und gewährleistet. Mockup in Abbildung….?

Quellenverzeichnis

Begin, C., 2006. *iBatis Data Access Objects Developer Guide Version 2.0.* s.l.:s.n.

Grashorn, D., 2010. *Entwicklung von NatInfo – Webbasiertes Tool zur Unterstützung der Entwickler,* Vechta: s.n.

ISO/IEC 9126-1, 2001. *Software-Engineering – Qualität von Software-Produkten – Teil 1: Qualitätsmodell.* s.l.:s.n.

Anhang

A1 Detaillierte Zeitplanung

Analysephase			**9 h**
1. Durchführung der Ist-Soll-Analyse (vorhandenen Importassistent)		3 h	
2. Wirtschaftlichkeitsprüfung und Amortisationsrechnung des Projektes		2 h	
3. Unterstützung des Fachbereichs bei der Erstellung des Lastenheftes		2 h	
4. Prüfung der technischen und organisatorischen Machbarkeit		1 h	
5. Erstellen eines Use-Case-Diagramms		1 h	
Entwurfsphase			**12 h**
1. Nutzwertanalyse zur Auswahl des Frameworks		1 h	
2. Datenbankentwurf		3 h	
2.1. ER-Modell erstellen	2 h		
2.2. Konkretes Tabellenmodell erstellen	1 h		
4. Benutzeroberflächen entwerfen und abstimmen		3 h	
5. Erstellen eines UML-Komponentendiagramms der Anwendung		2 h	
6. Erstellen des Pflichtenhefts		3 h	
Implementierungsphase			**46 h**
1. Implementierung Importprofil Verwaltung Komponente		25 h	
1.1. Implementierung Datentyp Klasse, DAO und Transaction Klassen	6 h		
1.2. Implementierung UI des Importprofils	5 h		
1.3. Implementierung der Logik für Profilverwaltung (CRUD)	7 h		
1.4. Implementierung der Eingabevalidierung	3 h		
1.5. Testen der Funktionalität der einzelnen Elemente des Importprofils	4 h		
2. Anpassung Vertragspartner Verwaltung (vorhandenes Element)		10 h	
2.1. Anpassung UI der Logik für Importprofil-Auswahl	4 h		
2.2. Implementierung der Logik für Importprofil Auswahl	6 h		
3. Anpassung Importassistent (vorhandenes Element)		5 h	
3.1. Die mit Externen „properties“ verbundene alte Logik entkoppeln	1 h		
3.2. Neue Logik mit Profil-Unterelementen verbinden	4 h		
4. Datenbanktabelle und Pflege		6 h	
Abnahme und Deployment			**4 h**
1. Code Review mit Ausbilder und Geschäftsführer		2 h	
2. Abnahme durch Ausbilder und PM		1 h	
3. Commit des Features und Deployment mit Jenkins		1 h	

Erstellen der Dokumentation		**9 h**
1. Erstellen der Benutzerdokumentation	1 h	
2. Erstellen der Projektdokumentation	7 h	
3. Programmdokumentation	1 h	
3.1. Generierung durch *JAVAdoc*	1 h	
Gesamt		**80 h**

Tabelle 5: Detaillierte Zeitplanung

A2 Lastenheft (Auszug)

Es folgt ein Auszug aus dem Lastenheft mit Fokus auf die Anforderungen:

Die Anwendung muss folgende Anforderungen erfüllen.

1. Mit Hilfe der zu entwickelnden Importprofilverwaltung (neue Seite) müssen…
   1.1. Importprofile hinzugefügt und mit einem Namen versehen.
   1.2. Importprofile bearbeitet.
   1.3. Importprofile entfernt.
   1.4. Ein Mitarbeiter kann Importprofile hinzufügen.
   1.5. Ein Mitarbeiter kann Importprofile bearbeiten.
   1.6. Ein Mitarbeiter kann Importprofile entfernen.
2. Mit Hilfe der zu entwickelnden Importprofiauswahl (Seite Ergänzung) müssen…
   2.1. Importprofil für einen bestimmten Datenlieferanten ausgewählt.
   2.2. Importprofil für einen bestimmten Datenlieferant gespeichert.
3. Folgende Informationen müssen bezüglich einer Importprofil allgemein speicherbar sein:
   3.1. Erste Tabelle:
      - Profilbezeichnung.
      - Zeichencodierung der Importdateien.
      - Verwendetes Trennzeichen.
      - Verwendetes Text Trennzeichen.
   3.2. Zweite Tabelle:
      - Referenz auf das Hauptprofil.
      - Spaltenbezeichnung.
      - Interne Referenz.
      - Datum Format und Zahlen Format (nice to have)[6].
   3.3. Dritte Tabelle:
      - Referenz auf das Hauptprofil.
      - Farbe.
      - Interne Referenz.

[6] Aus Zeitgründen wurde es nicht in die erste Iteration Implementiert.

A3 Use-Case-Diagramm

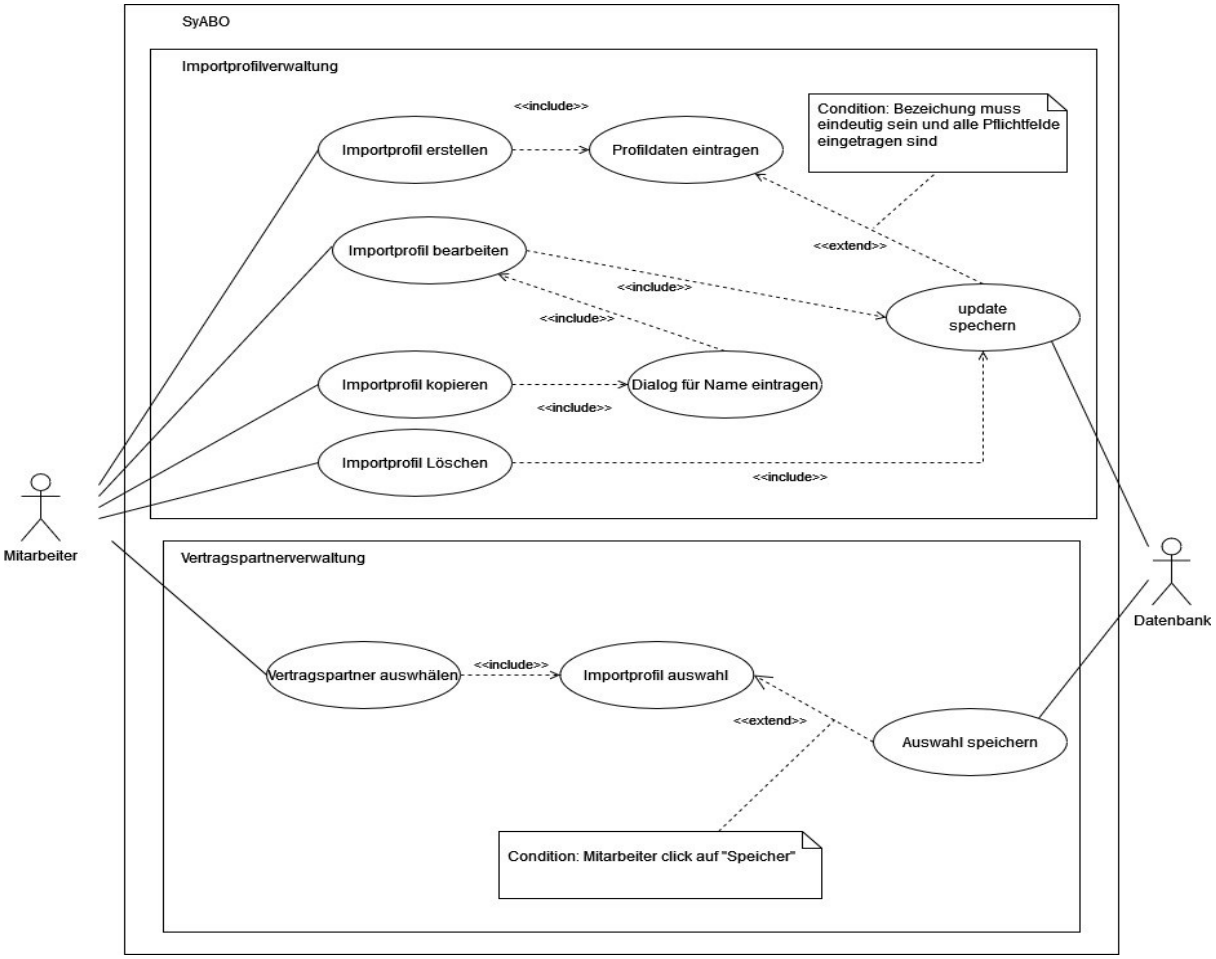

Abbildung 1: Use-Case-Diagramm

A4 Amortisationsdiagramm

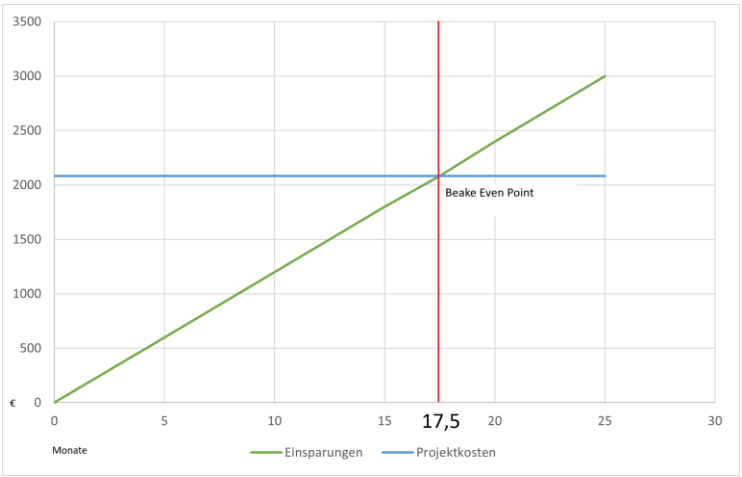

Abbildung 2: Amortisationsdiagramm

A5 Pflichtenheft (Auszug)

Zielbestimmung

1. Plattform
   1.1. Die Anwendung wird mit *Java* 5 implementiert.
   1.2. Die Entwicklung der Webkomponente geschieht mit Jakarta EE.
   1.3. Die Webanwendung wird nur im Intranet erreichbar sein.
   1.4. Die Webanwendung wird auf einem *Apache Tomcat* betrieben.
   1.5. *TortoiseSVN* wird für SCM genutzt.
   1.6. *Maven* wird für den automatischen Build-Prozess genutzt.
   1.7. Der *Jenkins-Server* übernimmt das CD.
2. Datenbank
   2.1. Die Datenbankanbindung erfolgt durch ORM (*iBatis).*
   2.2. Die Daten werden in einer *PostgreSQL* Datenbank gespeichert.
3. Benutzeroberflächen
   3.1. Die Benutzeroberflächen werden mithilfe von JSF realisiert.
   3.2. Die Benutzeroberflächen werden mit CSS 3.
   3.3. Grundlage zur weiteren Gestaltung ist JSP.
   3.4. Die Benutzeroberflächen werden nicht responsiv sein.
4. Zielgruppen
   4.1. Das Importprofilverwaltungstool wird lediglich von Mitarbeitern der Administrationsabteilung der Verschieden ASCI-Kunden genutzt.

A6 Datenbankmodell

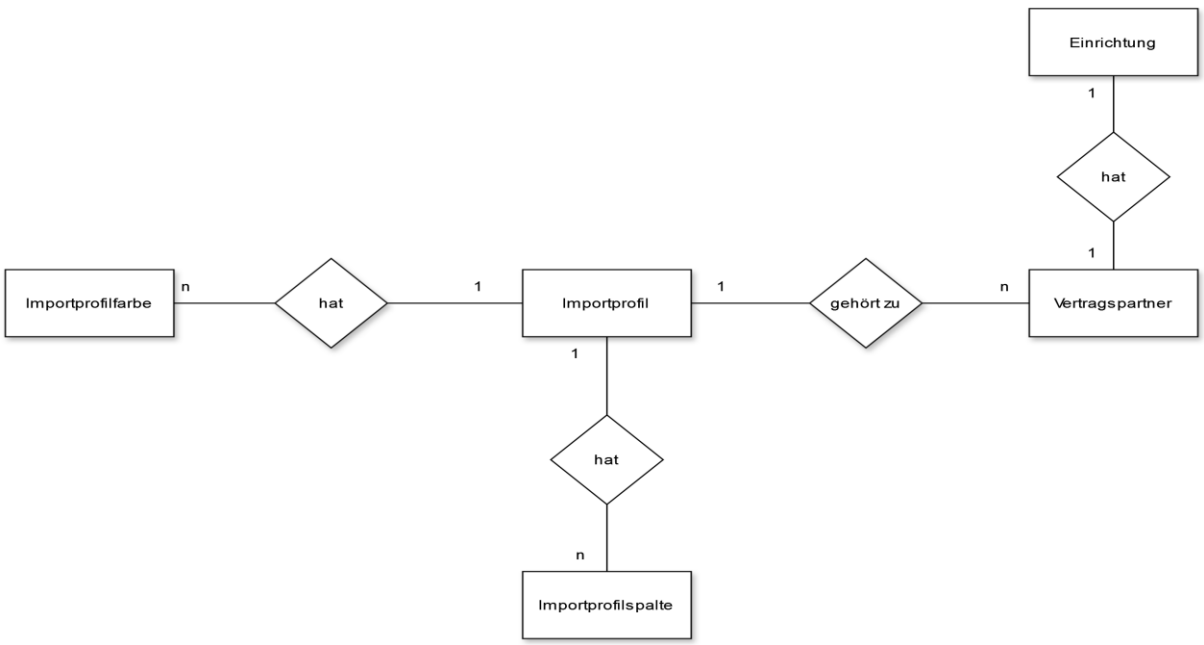

Abbildung 3: Entity-Relationship-Model

Anhang

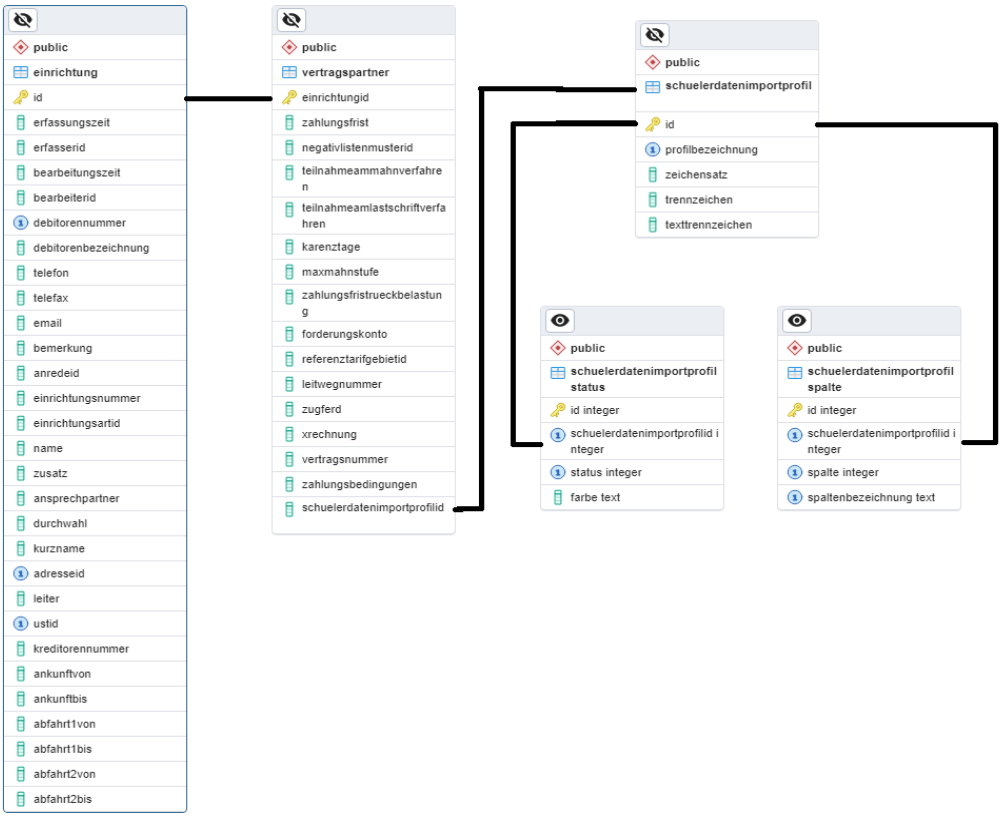

Abbildung 4: Tabellenmodell

A7 Aktivitätsdiagramm

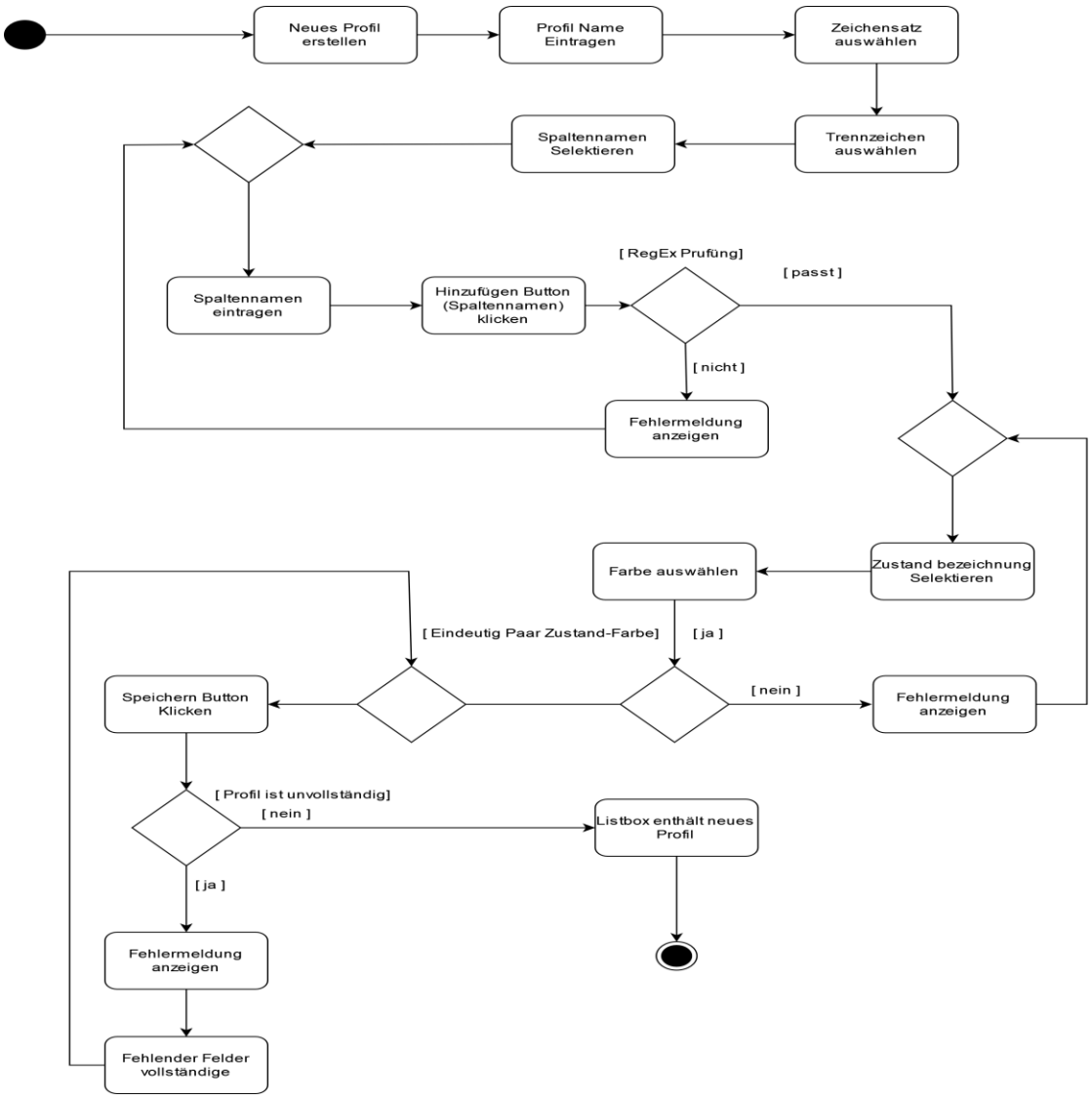

Abbildung 5: Prozess des Einlegen eines Importprofil

A8 Oberflächenentwürfe

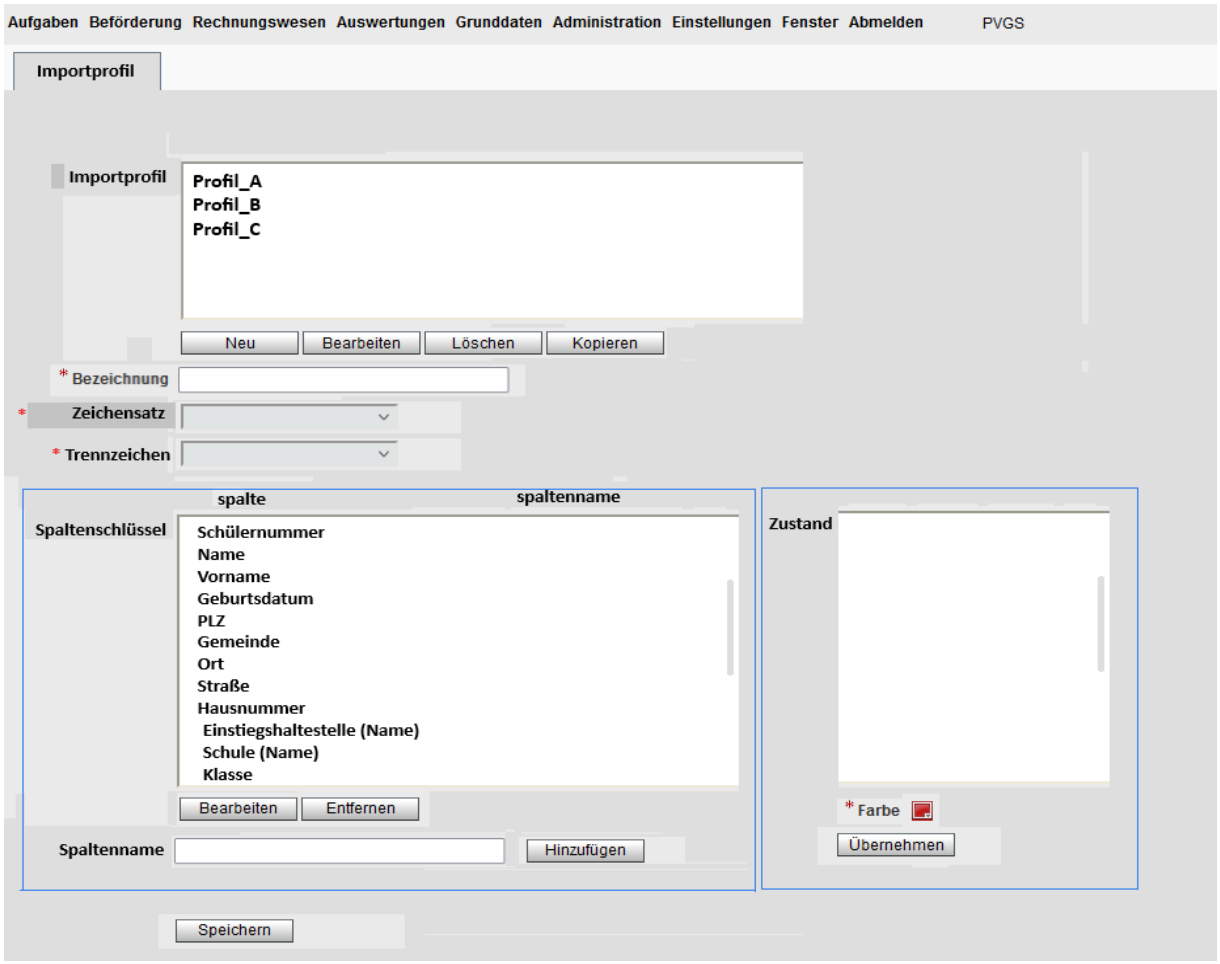

Abbildung 6: Benutzeroberfläche Mockup

A9 Screenshots der Schülerdatenimportprofil Seite

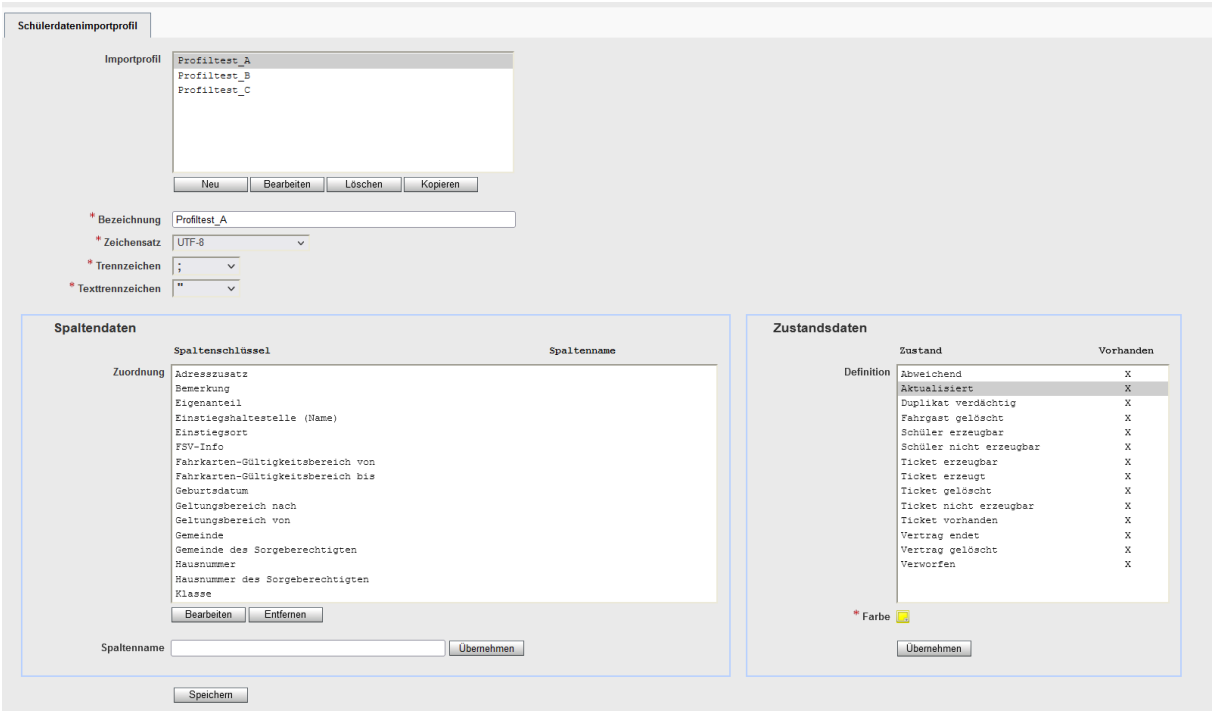

Abbildung 7: Anzeige der GUI-Unterteilung Design

A10 Screenshots der Anwendung

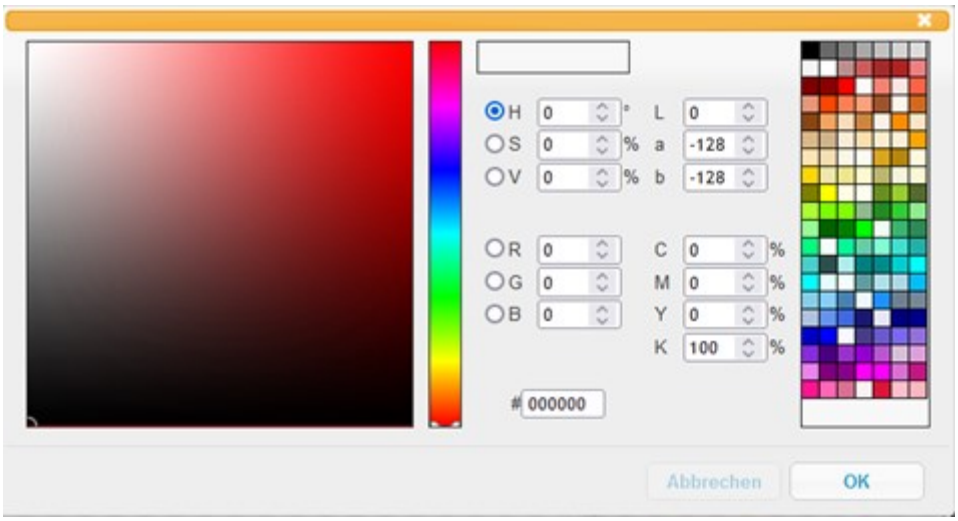

Abbildung 8: ColorPicker Popup Element

Anhang

```
<ui:script id="scriptColorpicker">
    <f:verbatim>
        $(function () {
                // Colorpicker modifiziert Skript von Fahrplandruckprofilcolorpicker: Was sich ändert ist folgendes:
                // Eigener Parser für cmyk hier nicht nötig
                // parts: 'full' war: parts: ['header', 'map', 'bar', 'hsv', 'rgb', 'cmyk', 'preview', 'footer'],'full'
                // colorFormat: '#HEX'
                $('.fahrplandruckprofilcolorpicker').colorpicker({
                    regional: 'de',
                    parts: 'full',
                    showOn: 'both',
                    colorFormat: '#HEX',
                    buttonImageOnly: true,
                    buttonColorize: true,
                    buttonImage: '../../resources/colorpicker/images/ui-colorpicker.png',
                    revert: true,
                    okOnEnter: true,
                    draggable: false,
                    modal: true,
                    title: ' ',
                    position: {
                        my: 'center',
                        of: window
                    }
                });
            });
            // Set the colorpicker input to be readonly
            $('.fahrplandruckprofilcolorpicker').prop('readonly', true);
    </f:verbatim>
</ui:script>
```

Abbildung 9: ColorPicker Script (JS)

A11 Entwicklerdokumentation (Auszug)

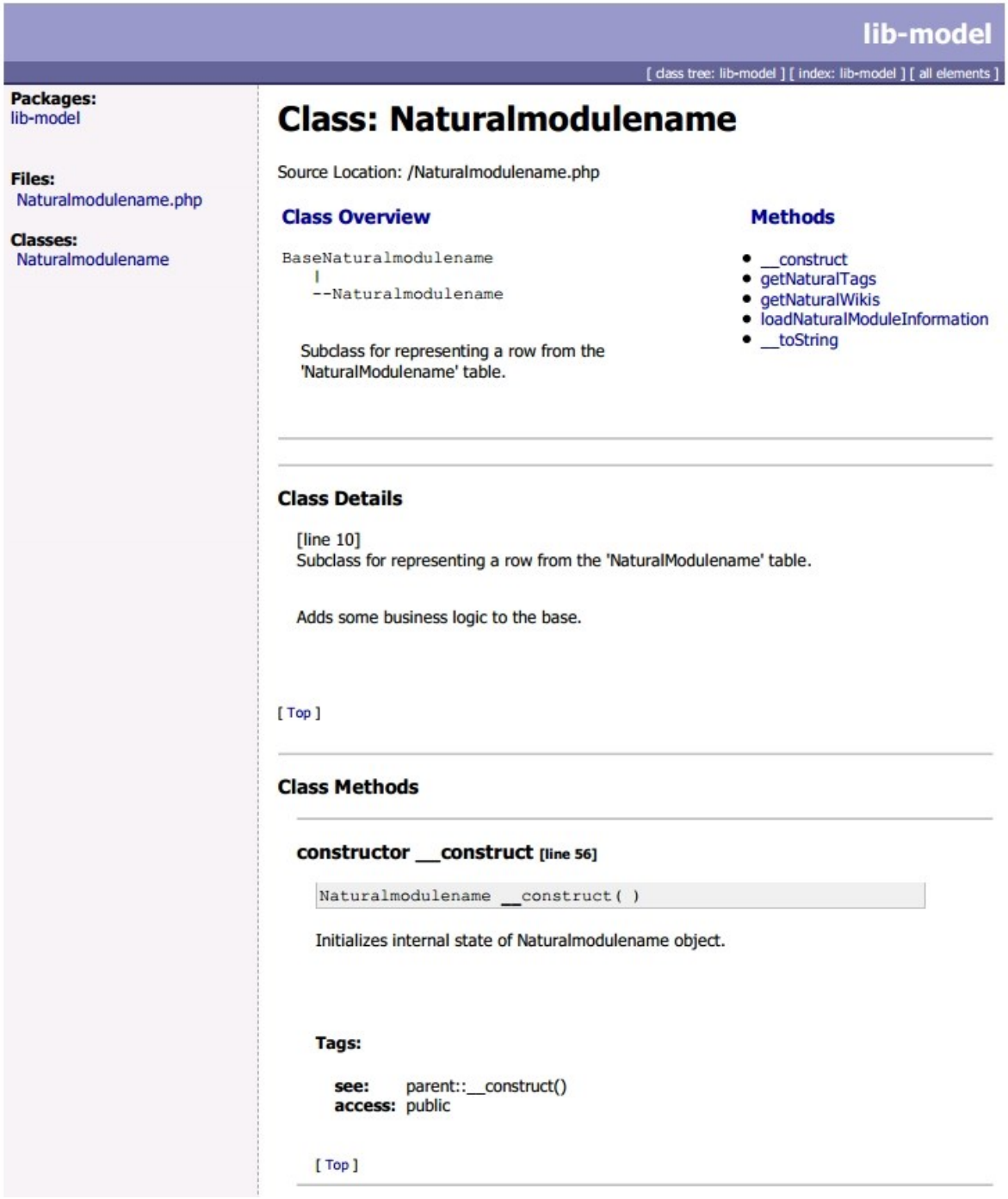

Abbildung 10: Auszug aus der Entwicklerdokumentation mit *PHPDoc*

A12 Testfall und sein Aufruf auf der Konsole

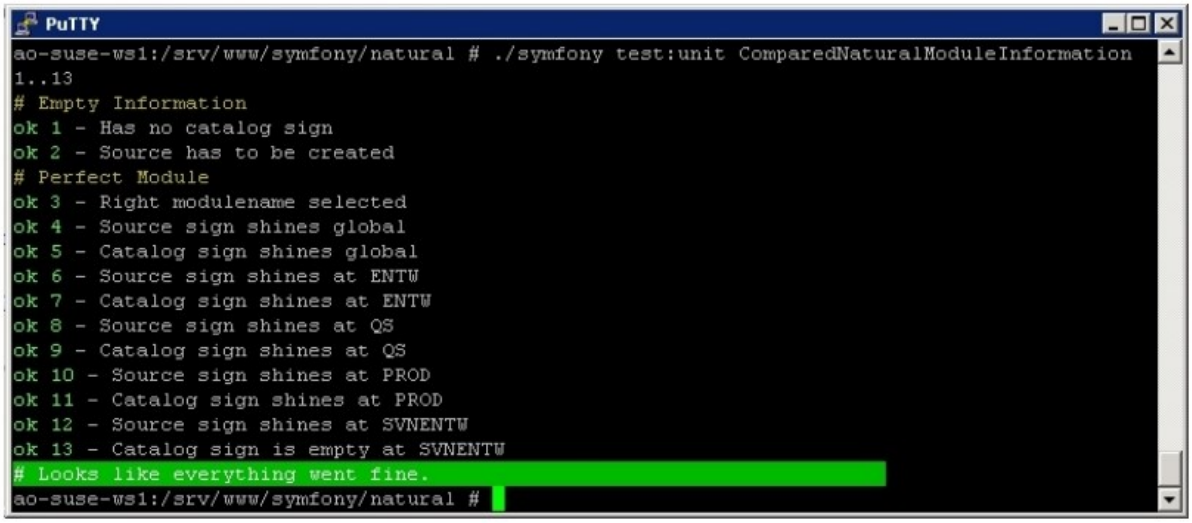

Abbildung 11: Aufruf des Testfalls auf der Konsole

Listing 1: Testklasse

A13 Klasse: TransactionSchuelerdatenimportprofil

Kommentare und simple Getter/Setter werden nicht gezeigt.

```
public class TransactionSchuelerdatenimportprofil
{
     private SchuelerdatenimportprofilDAO schuelerdatenimportprofilDAO;
     private SchuelerdatenimportprofilspalteDAO schuelerdatenimportprofilspalteDAO;
     private SchuelerdatenimportprofilstatusDAO schuelerdatenimportprofilstatusDAO;
     public Map<Integer, Schuelerdatenimportprofil>ladeSchuelerdatenimportprofile()
     {
          List<Schuelerdatenimportprofil> schuelerdatenimportprofile =
          this.schuelerdatenimportprofilDAO.selectAll();
          Map<Integer, Schuelerdatenimportprofil> schuelerdatenimportprofilMap =
          new HashMap<Integer, Schuelerdatenimportprofil>();
          for (Schuelerdatenimportprofil schuelerdatenimportprofil :
          schuelerdatenimportprofile)
          {
               schuelerdatenimportprofilMap.put(schuelerdatenimportprofil.getId(),
               schuelerdatenimportprofil);
          }
          List<Schuelerdatenimportprofilspalte> schuelerdatenimportprofilspalteList
          = this.schuelerdatenimportprofilspalteDAO.selectAll();
          for (Schuelerdatenimportprofilspalte schuelerdatenimportprofilspalte :
          schuelerdatenimportprofilspalteList)
          {
```

```
            Schuelerdatenimportprofil schuelerdatenimportprofil =
            schuelerdatenimportprofilspalte.getSchuelerdatenimportprofil();

            schuelerdatenimportprofil =
            schuelerdatenimportprofilMap.get(schuelerdatenimportprofil.getId());


            schuelerdatenimportprofilspalte.setSchuelerdatenimportprofil(schuele
            rdatenimportprofil);


            schuelerdatenimportprofil.getSchuelerdatenimportprofilspalteMap().put
            (schuelerdatenimportprofilspalte.getId(),
            schuelerdatenimportprofilspalte);

      }

      List<Schuelerdatenimportprofilstatus> schuelerdatenimportprofilstatusList
      = this.schuelerdatenimportprofilstatusDAO.selectAll();

      for (Schuelerdatenimportprofilstatus schuelerdatenimportprofilstatus :
      schuelerdatenimportprofilstatusList)

      {

            Schuelerdatenimportprofil schuelerdatenimportprofil =
            schuelerdatenimportprofilstatus.getSchuelerdatenimportprofil();

            schuelerdatenimportprofil =
            schuelerdatenimportprofilMap.get(schuelerdatenimportprofil.getId());


            schuelerdatenimportprofilstatus.setSchuelerdatenimportprofil(schuele
            rdatenimportprofil);


            schuelerdatenimportprofil.getSchuelerdatenimportprofilstatusMap().put
            (schuelerdatenimportprofilstatus.getId(),
            schuelerdatenimportprofilstatus);

      }

      return schuelerdatenimportprofilMap;

}

public void
loescheSchuelerdatenimportprofil(IntegerschuelerdatenimportprofilId)

{
      this.schuelerdatenimportprofilspalteDAO.deleteBySchuelerdatenimportprofil
      (schuelerdatenimportprofilId);

      this.schuelerdatenimportprofilstatusDAO.deleteBySchuelerdatenimportprofil
      (schuelerdatenimportprofilId);

      this.schuelerdatenimportprofilDAO.delete(schuelerdatenimportprofilId);

}

public void speichereSchuelerdatenimportprofil(Schuelerdatenimportprofil
schuelerdatenimportprofil)

{

      if (schuelerdatenimportprofil.getId() == null)

      {

            this.schuelerdatenimportprofilDAO.insert(schuelerdatenimportprofil);

      }

      else

      {
```

```
                this.schuelerdatenimportprofilDAO.update(schuelerdatenimportprofil);
            }
            this.schuelerdatenimportprofilspalteDAO.deleteBySchuelerdatenimportprofil
            (schuelerdatenimportprofil.getId());
            for (Schuelerdatenimportprofilspalte schuelerdatenimportprofilspalte :
          schuelerdatenimportprofil.getSchuelerdatenimportprofilspalteMap().values())
            {
                if (schuelerdatenimportprofilspalte.getId() == null)
                {
                    Integer id =
                    this.schuelerdatenimportprofilspalteDAO.selectNextId();
                    schuelerdatenimportprofilspalte.setId(id);
                }
                this.schuelerdatenimportprofilspalteDAO.insert(schuelerdatenimportp
                rofilspalte);
            }
            this.schuelerdatenimportprofilstatusDAO.deleteBySchuelerdatenimportprofil
            (schuelerdatenimportprofil.getId());
            for (Schuelerdatenimportprofilstatus schuelerdatenimportprofilstatus :
          schuelerdatenimportprofil.getSchuelerdatenimportprofilstatusMap().values())
            {
                if (schuelerdatenimportprofilstatus.getId() == null)
                {
                    Integer id =
                    this.schuelerdatenimportprofilstatusDAO.selectNextId();
                    schuelerdatenimportprofilstatus.setId(id);
                }
                this.schuelerdatenimportprofilstatusDAO.insert(schuelerdatenimportp
                rofilstatus);
            }
        }
}
```

Listing 2: Klasse `TransactionSchuelerdatenimportprofil`

A14 Klassendiagramm

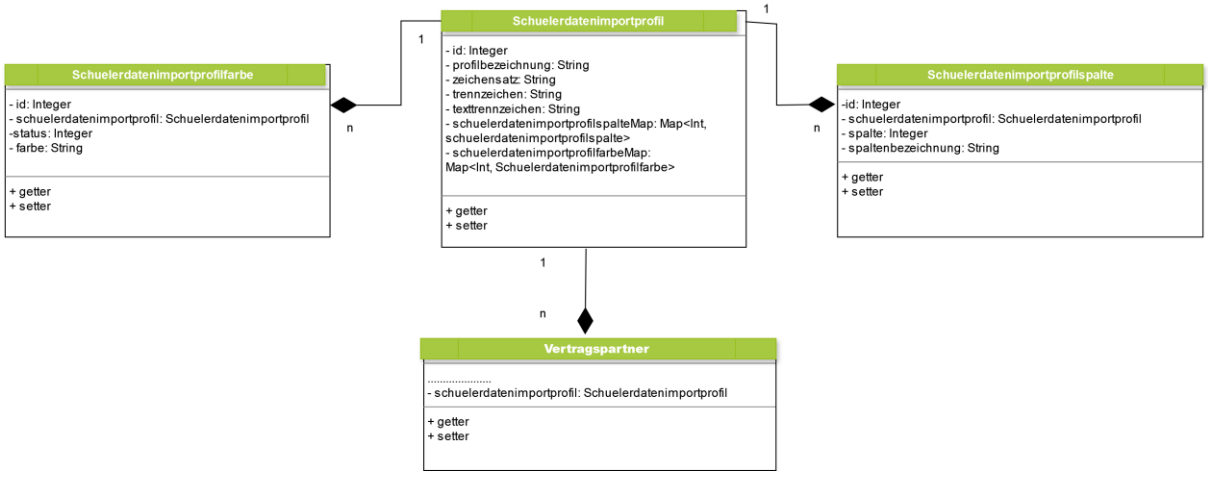

Abbildung 12: Klassendiagramm

A15 Benutzerdokumentation (Auszug)

Symbol	Bedeutung global	Bedeutung einzeln
	Alle Module weisen den gleichen Stand auf.	Das Modul ist auf dem gleichen Stand wie das Modul auf der vorherigen Umgebung.
	Es existieren keine Module (fachlich nicht möglich).	Weder auf der aktuellen noch auf der vorherigen Umgebung sind Module angelegt. Es kann also auch nichts übertragen werden.
	Ein Modul muss durch das Übertragen von der vorherigen Umgebung erstellt werden.	Das Modul der vorherigen Umgebung kann übertragen werden, auf dieser Umgebung ist noch kein Modul vorhanden.
	Auf einer vorherigen Umgebung gibt es ein Modul, welches übertragen werden kann, um das nächste zu aktualisieren.	Das Modul der vorherigen Umgebung kann übertragen werden um dieses zu aktualisieren.
	Ein Modul auf einer Umgebung wurde entgegen des Entwicklungsprozesses gespeichert.	Das aktuelle Modul ist neuer als das Modul auf der vorherigen Umgebung oder die vorherige Umgebung wurde übersprungen.

Abbildung 13: Auszug aus der Benutzerdokumentation

A16 Testprotokoll (Auszug)

Ergebnis

Testfall	Erwartet	Tatsächlich	Ergebnis
Importprofil an Vertragspartner zuweisen	OK	OK	OK
Importprofil an Vertragspartner wechseln	OK	OK	OK
Importprofil an Vertragspartner speichern	OK	OK	OK
Importassistent ohne Zuweisung starten	Scheitern	Scheitern	OK

Importassistent mit Zuweisung starten	OK	OK	OK
Neu Importprofil legen	OK	OK	OK
Bezeichnung ändern	OK	OK	OK
Zeichencodierung ändern	OK	OK	OK
Trennzeichen ändern	OK	OK	OK
Spaltenname Hinzufügen	OK	OK	OK
Zustand-Farbe zuweisen	OK	OK	OK
Importprofil unvollständig speichern	Scheitern	Scheitern	OK
Importprofil vollständig speichern	OK	OK	OK

Abbildung 14: Auszug Testprotokoll

A17 WIKI-Dokumentation (Auszug)

Übersicht | Aktivität | Tickets | Neues Ticket | Gantt-Diagramm | Kalender | Dokumente | Wiki

WikiStart » TypesIndex »

Bearbeiten Beobachten Umbenennen Löschen Historie

Wiki

Haup
Seite
Seite

WikiStart - TypesIndex

Name: Schuelerdatenimportprofil

Alias: sdip

Beschreibung:

Anwenderzugriff: ...

Attribute:

Name	Typ	NULL	Beschreibung
id	Integer		Generierter Primärschluessel
profilbezeichnung	String		
zeichensatz	String		
trennzeichen	String		
texttrennzeichen	String	X	

Fremdschlüssel:

Referenzierungen:

Tabelle	Spalte	Referenzspalte	ON UPDATE	ON DELETE
schuelerdatenimportprofilspalte	id	schuelerdatenimportprofilid	NO ACTION	CASCADE
schuelerdatenimportprofilstatus	id	schuelerdatenimportprofilid	NO ACTION	CASCADE
vertragspartner	id	schuelerdatenimportprofilid	NO ACTION	NO ACTION

Constraints:

- ix_schuelerdatenimportprofil_profilbezeichnung UNIQUE (profilbezeichnung)

Anmerkungen:

letzte Aenderungen:

Abbildung 15: Auszug aus der WIKI-System

A18 iBatis Framework Dokumentation (Auszug)

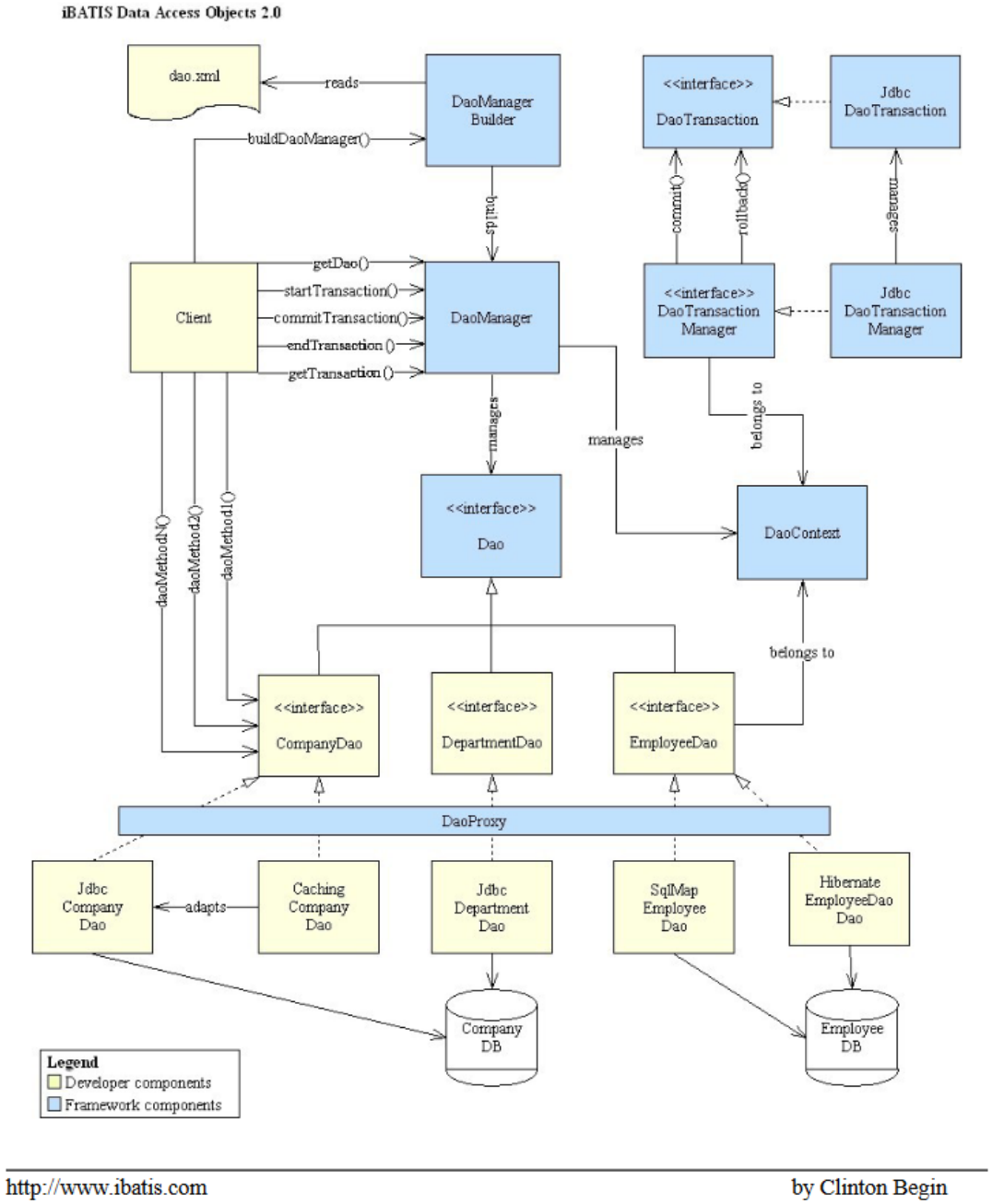

Abbildung 16: Auszug aus der iBATIS Data Access Objects Developer Guide