



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.....	I
Abkürzungsverzeichnis.....	IV
1 Einleitung	1
1.1 Projektumfeld	1
1.2 Projektziel	1
1.3 Projektbegründung.....	1
1.4 Projektschnittstellen	2
1.5 Projektabgrenzung	2
2 Projektplanung	2
2.1 Projektphasen	2
2.2 Abweichungen vom Projektantrag.....	2
2.3 Ressourcenplanung	3
2.4 Entwicklungsprozess.....	3
3 Analysephase.....	3
3.1 Ist-Analyse	3
3.2 Wirtschaftlichkeitsanalyse	4
3.2.1 Make or Buy-Entscheidung	4
3.2.2 Projektkosten	4
3.2.3 Amortisationsdauer	5
3.3 Nutzwertanalyse.....	5
3.4 Anwendungsfälle.....	5
3.5 Qualitätsanforderungen.....	6
3.6 Lastenheft/Fachkonzept.....	6
4 Entwurfsphase	6
4.1 Zielplattform	6
4.2 Architekturdesign	6
4.3 Entwurf der Benutzeroberfläche	7
4.4 Datenmodell.....	7
4.5 Geschäftslogik.....	8
4.6 Maßnahmen zur Qualitätssicherung.....	8
4.7 Pflichtenheft/Datenverarbeitungskonzept	9
5 Implementierungsphase	9
5.1 Implementierung der Datenstrukturen	9
5.2 Implementierung der Benutzeroberfläche.....	10
5.3 Implementierung der Geschäftslogik	10
5.3.1 Einleitung zu Implementierung	10

Inhaltsverzeichnis

5.3.2	Implementierung Transactionklasse	10
5.3.3	Implementierung Front-end Java Klasse	11
5.3.4	Anpassung der Vertragspartner Seite	12
5.3.5	Anpassung der Importassistent-Servlets	12
5.3.6	Zusammenfassung der Implementierung	13
6	Abnahmephase	13
7	Einführungsphase	14
8	Dokumentation	14
9	Fazit	15
9.1	Soll-/Ist-Vergleich	15
9.2	Lessons Learned.....	15
9.3	Ausblick.....	15
Anhang	i
A1	Detaillierte Zeitplanung.....	i
A2	Reg-Ex Code (Auszug)	ii
A3	Lastenheft (Auszug)	iii
A4	Verwendete Ressourcen	iv
A5	Ticket	v
A6	Amortisationsdiagramm.....	v
A7	Use-Case-Diagramm.....	vi
A8	Pflichtenheft (Auszug)	vii
A9	Oberflächenentwürfe.....	viii
A10	Datenbankmodell	ix
A11	Datenbankschema	x
A12	Aktivitätsdiagramm	xi
A13	Klassendiagramm	xii
A14	Unique Datenbank „Constraints“	xiii
A15	Screenshots der Schülerdatenimportprofil Seite.....	xiv
A16	Screenshots der Anwendung	xiv
A17	ColorPicker Script (JS).....	xv
A18	iBatis Framework Dokumentation (Auszug).....	xvi
A19	Klasse: TransactionSchuelerdatenimportprofil.....	xvii
A20	Architekturdiagramm (Auszug)	xix
A21	Schuelerdatenimportprofil.java Speichern Methode (Auszug)	xx
A22	Importassistent-Servlet Anpassung (Auszug).....	xx
A23	Testprotokoll (Auszug)	xxi
A24	Jenkins Build-Prozess GUI (Auszug).....	xxi

Inhaltsverzeichnis

A25	WIKI-Dokumentation (Auszug)	xxii
A26	Benutzerdokumentation (Auszug)	xxiii
A27	Mockup des künftigen Iteration-Zyklus	xxiv
Abbildungsverzeichnis		xxv
Tabellenverzeichnis		xxvi
Verzeichnis der Listings		xxvii
Quellenverzeichnis		xxix

Abkürzungsverzeichnis

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

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*), Logik und *Backend* 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.

Projektplanung

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 der IHK Berlin vorgegeben.

Bezüglich der in 1.2 Projektziel genannten 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 Seite i 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 A15 auf Seite xiv dargestellt, befindet sich ein *Textfeld* mit dem Label¹ "Spaltenname", in das der Benutzer alphanumerische Werte eingeben

¹ GUI-Element, Beschriftungstext nicht nur optisch mit der Entsprechende „Schwester“ GUI-Element verbunden, sondern auch programmatisch damit verknüpft. In den Fall mit Texteingabe.

Analysephase

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 Screenshot A2: *Reg-Ex Code* (Auszug) Seite ii zu sehen.

2.3 Ressourcenplanung

Anschließend wurden verwendete Ressourcen im Anhang A4: *Verwendete Ressourcen* auf Seite iv 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 Programmdateien 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 A5 Seite v):

- 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 im Kapitel 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:

$$\begin{aligned}
 8 \frac{\text{h}}{\text{Tag}} \cdot 220 \frac{\text{Tage}}{\text{Jahr}} &= 1.760 \frac{\text{h}}{\text{Jahr}} \\
 1.000 \frac{\text{€}}{\text{Monat}} \cdot 13,3 \frac{\text{Monate}}{\text{Jahr}} &= 13.300 \frac{\text{€}}{\text{Jahr}} \\
 \frac{13.300 \frac{\text{€}}{\text{Jahr}}}{1.760 \frac{\text{h}}{\text{Jahr}}} &\approx 7,56 \frac{\text{€}}{\text{h}}
 \end{aligned}$$

Es ergibt sich also ein Stundensatz von **7,56 EUR**. Die Durchführungszeit des Projekts beträgt 80 Stunden. Für die Nutzung von Ressourcen² 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 **2.084,80 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

² Räumlichkeiten, Arbeitsplatzrechner etc.

Analysephase

3.2.3 Amortisationsdauer

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

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

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 A6: *Amortisationsdiagramm* auf S. v 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*).

Die nicht-monetäre Vorzüge der Implementierung des neuen Features können genauer betrachtet werden, um ein umfassendes Verständnis für die Auswirkungen auf das Projekt zu erhalten. Hierbei sollten insbesondere die folgenden Aspekte berücksichtigt werden:

Verbesserte Effizienz: Durch die Möglichkeit, jedem Vertragspartner ein individuelles Importprofil zuzuweisen, können Arbeitsabläufe optimiert und die Effizienz gesteigert werden. Dies führt zu einer schnelleren und präziseren Verarbeitung von Daten.

Reduzierter Abstimmungsaufwand: Die Flexibilität, verschiedene Importprofile zu nutzen, reduziert den Abstimmungsaufwand zwischen dem Systemanbieter und den Datenlieferanten erheblich. Da die Vertragspartner ihre eigenen Profile konfigurieren können, entfällt die Notwendigkeit einer einheitlichen *Datenstruktur*, was den Abstimmungsprozess erheblich vereinfacht.

Bessere Anpassungsfähigkeit: Das neue Feature ermöglicht es den Vertragspartnern, ihre Importprofile entsprechend ihren individuellen Anforderungen anzupassen. Dies erhöht die Anpassungsfähigkeit des Systems und ermöglicht es den Benutzern, ihre Arbeitsabläufe effektiver zu gestalten.

Höhere Benutzerzufriedenheit: Durch die Verbesserung der Importfunktionalitäten und die Reduzierung des Abstimmungsaufwands wird die Benutzerzufriedenheit insgesamt gesteigert. Benutzer können effizienter arbeiten und sind weniger frustriert durch Probleme im Zusammenhang mit dem Import von Daten.

Langfristige Wettbewerbsfähigkeit: Die Implementierung dieses Features stärkt die Wettbewerbsfähigkeit des Systems, da es flexibler und besser an die individuellen Bedürfnisse der Kunden angepasst werden kann. Dies trägt dazu bei, langfristige Kundenbindungen aufzubauen und das System als bevorzugte Lösung im Markt zu etablieren.

3.4 Anwendungsfälle

Bei einem Treffen mit den Projektbeteiligten wurde ein *Anwendungsfalldiagramm* entwickelt, das die Hauptfunktionen der zu entwickelnde Anwendung darstellt. Dieses unter A7 *Use-Case-Diagramm* auf S. vi aufgeführte Diagramm kann des Weiteren zur Einteilung der Implementierung in einzelne Features herangezogen werden.

Entwurfsphase

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 A3: *Lastenheft* (Auszug) auf S. iii 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*, *iBatis ORM* 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 vermeiden³. 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 ist hauptsächlich für die Verwendung mit dem *Firefox-Browser* optimiert, da dies der *Standardbrowser* im Unternehmen des ASCI-Kunden 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

³ SyABO bietet DSGVO konforme Umsetzung der Anforderungen zum Datenschutz.

Entwurfsphase

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 *Frontendes* 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 Projektbetreuer 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 A9: Benutzeroberfläche *Mockup* Seite viii 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 A10: *ERM* auf Seite ix dargestellt. Um die Beziehung zwischen diesen beiden

Entwurfsphase

Entitäten darzustellen, wird ein *Fremdschlüssel*, die "SchuelerdatenimportprofilID", zur "Vertragspartner"-Tabelle hinzugefügt. Das "Schuelerdatenimportprofil" bildet den Kern des *Datenmodells* für diese neue Funktion.

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

Jedes Schuelerdatenimportprofil ist einer Liste von Spalten und Zuständen zugeordnet, wie im *Datenbankschema* in Anhang A11 auf Seite x dargestellt.

Die Zustände sind in der Tabelle „Schuelerdatenimportprofilstatus“ 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 „Schuelerdatenimportprofilspalte“ hat wie die Tabelle „Schuelerdatenimportprofilstatus“ 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

Das *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. Ein Auszug des *Aktivitätsdiagramms* in Anhang A12 Seite xi zeigt den grundsätzlichen Ablauf beim Einlegen eines Importprofil und Vertragspartner Zuweisung.

Der Entwurf der Logik der *Datenmodelle*, die dem Schuelerdatenimportprofil untergeordnet sind, schuelerdatenimportprofilspalte⁴ und schuelerdatenimportprofilstatus⁵ des Schuelerdatenimportprofils, hängt von der *Datenbanklogik* ab.

Zum besseren Verständnis der untergeordneten Struktur und der Beziehung zwischen den *Klassen* ist ein *Klassendiagramm* in Anhang A13 Seite xii verfügbar. Dieses Diagramm war natürlich die Grundlage für die Entwurfsphase und die Implementierung der erforderlichen *Datenstrukturen*.

Entsprechende *Datenmodellen* der *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 Teil dieser Dokumentation. Dennoch habe ich diese „*Constraints*“ in einem vereinfachten Auszug aus dem *Datenbankschema* Entwurf in Anhang A14 Seite xiii dargestellt.

4.6 Maßnahmen zur Qualitätssicherung

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

⁴ Klassennamen müssen nicht unbedingt den Regeln der deutschen Grammatik folgen.

⁵ Klassennamen.

Implementierungsphase

Darüber hinaus testet der Entwickler die Anwendung während der Entwicklung regelmäßig, um die Funktionalität der einzelnen Methoden sicherzustellen.

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 A8 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

Basierend auf den, in vorherigen Kapiteln erwähnten Artefakten wird zunächst ein *SQL-Skript* erstellt, um die drei neuen Tabellen hinzuzufügen und die Tabelle "Vertragspartner" anzupassen. Letztere ist bereits vorhanden und wird nun um die Zuweisung der "SchülerdatenimportprofilID" erweitert.

Das *SQL-Skript* wurde speziell für das *PostgreSQL-Datenbanksystem* entwickelt und wird mithilfe der *pgAdmin4-Software* verwaltet. Unter Anderem Es fügt die folgenden Tabellen hinzu:

1. *Schuelerdatenimportprofil*: Diese Tabelle enthält Informationen zu den Schülerdatenimportprofilen, einschließlich der Profilbezeichnung, des Zeichensatzes, des Trennzeichens und des Texttrennzeichens⁶.
2. *Schuelerdatenimportprofilspalte*: Hier werden die einzelnen Spalten eines Schülerdatenimportprofils gespeichert, einschließlich der Spaltennummer, der Spaltenbezeichnung und der Zuordnung zum entsprechenden Profil.
3. *Schuelerdatenimportprofilstatus*: Diese Tabelle enthält die verschiedenen Statusoptionen für Schülerdatenimportprofile, wie z. B. den Importstatus und die entsprechende Farbcodierung.

Dieses initiale *Skript* bildet den Ausgangspunkt für die Backend-Implementierung. Ein zweites *Skript* wird benötigt, um die Benutzeroberfläche zu testen oder relativ einfach zu implementieren. Dadurch kann sofort überprüft werden, ob das hinzugefügte *GUI-Element* korrekt dargestellt wird oder ob etwaige Bindungsprobleme (*Binding*⁷) schnell behoben werden können.

⁶ In LibreOffice heißt Texttrenner oder Zeichenfolgen-Trennzeichen und in Excel Textqualifizierer. Als Textqualifizierer werden Zeichen wie Anführungszeichen verstanden, die Text-Werte umschließen. Textqualifizierer für den Import von CSV-Dateien werden verwendet, um Textfelder zu kennzeichnen, die Sonderzeichen, wie z. B. Kommas oder Zeilenumbrüche, enthalten könnten. Durch die Verwendung von Textqualifizierern können diese Zeichen erkannt und die Integrität der Daten beim Import in Excel gewährleistet werden.

⁷ 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“.

Implementierungsphase

Eine zweite *SQL-Skript* dient, wie für andere neue Seite eine Hinzufügung einer neuen Menüpunkt. Genau Position der Menüeintrag, Name und andere Eigenschaften nicht Teil der 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 unterscheidbaren Elementen unterteilt werden, nämlich: oberer Teil für das Schuelerdatenimportprofil selbst, links blau umrahmter Bereich für die Schuelerdatenimportprofilspaltenelemente und rechts ebenfalls blau umrahmter Bereich für die Schuelerdatenimportprofilstatuselemente. Unterteilung auch in Screenshot A15 Schuelerdatenimportprofil-Webpage, Seite xiv zu sehen.

Bezüglich der unteren rechten 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 Importassistent 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 Screenshot Anhang A16 Seite xiv gezeigt. Screenshots von *Color Picker JavaScript Code* (Auszug von `SchuelerdatenimportprofilPage.jsp`) befinden sich im Anhang A17 Seite xv.

5.3 Implementierung der Geschäftslogik

5.3.1 Einleitung zu Implementierung

Das *iBatis ORM* bietet eine effektive Möglichkeit zur Durchführung von *CRUD*-Operationen (Create, Read, Update, Delete) für die Verwaltung von Datenbankaufrufen. Im Anhang A18 auf Seite xvi finden Sie einen Ausschnitt aus dem *Architekturdiagramm* des *iBatis ORM-Frameworks* (Begin, 2006), welches die Struktur und Funktionsweise dieses *ORM-Frameworks* veranschaulicht. Die Integration von *XML-Dateien* ermöglicht eine klare und strukturierte Beschreibung von Datenbankabfragen und -manipulationen, die dann von *iBatis* entsprechend umgesetzt werden.

5.3.2 Implementierung Transactionklasse

Als Beispiel für Beschreibung der Geschäftslogik ist die „`TransactionSchuelerdatenimportprofil`“ Klasse ausgewählt, die für die Datenverarbeitung im Zusammenhang mit `Schuelerdatenimportprofil.java` zuständig ist. Diese Klasse ist verantwortlich für das Laden, Löschen und Speichern von Schülerdatenimportprofilen⁸ sowie deren zugehörigen Spalten und Status.

⁸ Warum Schülerdatenimportprofil und nicht einfach Importprofil? In der Praxis existieren bereits andere Importprofile für andere Funktionalitäten oder können in Zukunft implementiert werden. Ein so spezifischer Name für das Hauptobjekt und die entsprechenden Klassen ist zwar nicht kurz und kompakt, dient aber dazu, dass seine Funktion und sein „*Scope*“ eindeutig im Namen enthalten und kodiert sind.

Implementierungsphase

Zu Beginn der Implementierung wird eine Instanz der *Klasse* erstellt und die erforderlichen *DAOs (Data Access Objects)*⁹ über *Setter-Methoden* gesetzt. Anschließend werden die Schülerdatenimportprofile aus der Datenbank geladen, wobei auch die zugehörigen Spalten und Status geladen und den entsprechenden Profilen zugeordnet werden.

Für das Löschen eines Schülerdatenimportprofils werden zunächst die zugehörigen Spalten und Status gelöscht, bevor das Profil selbst entfernt wird (*DB referentielle Integrität*). Beim Speichern eines Schülerdatenimportprofils 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.

Die Klasse `TransactionSchuelerdatenimportprofil` findet sich im Anhang A19 Seite xvii.

5.3.3 Implementierung Front-end Java Klasse

Um die Beziehung zwischen die *Klassen* besser zu verstehen, hat der Autor ein vereinfachtes *Architekturdiagramm* eingefügt. Die *Klassen* `SchuelerdatenimportprofilPage.jsp` und `SchuelerdatenimportprofilPage.java` sie werden hier aus Platz Gründe als *JSP-Klasse* und *Java Klasse* referenziert. Die *JSP-Klasse* wurde quasi parallel mit den *Java Klasse* entwickelt. Die Reihenfolge der Beschreibung in diesem Kapitel entspricht nicht die Entwicklung Reihenfolge, sondern die Architektur Reihenfolge (von *Backend* nach *Frontend*) wie zu sehen in Diagramm A20 Seite xix.

Designentscheidungen und Prinzipien: Bei der Entwicklung der *Klasse* wurden Prinzipien wie *Kapselung* und *Abstraktion* berücksichtigt, um die Interaktion mit den Importprofileinformationen zu vereinfachen und die Wiederverwendbarkeit zu fördern. Die *Klasse* folgt auch dem *SOLID10-Prinzip*, insbesondere dem *Single Responsibility Principle*, indem sie nur für die Verwaltung von Importprofileinformationen zuständig ist.

„Functions should do one thing. They should do it well. They should do it only.“ (Martin "Uncle Bob", 2009)

Testbarkeit und Wartbarkeit: Die *Klasse* ist gut testbar, da sie klar definierte Schnittstellen für den Zugriff auf Importprofileinformationen bietet. Die Methoden sind einzeln testbar und können leicht durch *Mock-Objekte* simuliert werden. Die Wartbarkeit der *Klasse* wird durch ihre modulare Struktur und die Verwendung von kohärenten Methoden verbessert.

Der Autor hat sich bewusst dafür entschieden, lediglich die *Transaction-Klasse* und einen Ausschnitt der *Speichermethode* in diesem Format zu reproduzieren, um Platz- und Formatierungsprobleme zu vermeiden. Er ist der Auffassung, dass die Formatierung und der saubere Code in einer integrierten *Entwicklungsumgebung (IDE)* besser dargestellt werden können und dass seine Aufmerksamkeit für Details möglicherweise nicht vermittelt wird. Insbesondere war

⁹ Das DAO repräsentiert eine Schnittstelle zur Datenbank in Ihrer Anwendung. Es agiert als Mittler zwischen Ihrer Anwendungslogik und der Datenbank. Das DAO ermöglicht Daten aus der Datenbank abzurufen, zu speichern oder zu ändern, ohne mit den komplexen 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.

¹⁰ Akronym, das für fünf Designprinzipien für die Entwicklung von Software beschreibt: Single Responsibility, Open/Close, Liskov Substitution, Interface Segregation and Dependency Inversion Principle.

Implementierungsphase

die Lesbarkeit der *Transaction-Klasse* stark eingeschränkt. Die Bildschirme und *IDEs*, in denen der Code geschrieben wird, sind horizontal orientiert, während dieses Format vertikal ist. Selbstverständlich kann der vollständige Quellcode auf Anfrage der Prüfer bereitgestellt werden. Der Autor Kontaktdaten sind der IHK Berlin bekannt. Der Abschnitt der Klasse `SchuelerdatenimportprofilPage.java` ist im Anhang A21 Seite xx zu finden.

5.3.4 Anpassung der Vertragspartner Seite

Die Vertragspartner Seite wurde um ein *Dropdown-Menü* und entsprechenden *Label* erweitert, mit dem ein Benutzer einem bestimmten Vertragspartner einfach durch Klicken auf eine der vorgegebenen Schülerdatenimportprofile zuweisen kann. Damit diese Änderung jedoch in der Datenbank übertragen werden kann, muss am Ende der Seite auf „Speichern“ geklickt werden. Erst nach dem Klick auf „Speichern“ ist ein Vertragspartner im Sinne eines Importprofils für externe Daten vollständig, und erst dann kann ein Import mithilfe einer anderen Seite (des Importassistent-Servlets) durchgeführt werden.

The screenshot shows a web form titled 'Vertragspartner-Seite'. At the top, there is a dropdown menu labeled '* Schuelerdatenimportprofil' with a blue arrow pointing to it. Below this, there is a section with a blue border containing a checkbox labeled 'Schule' with the text 'verwenden' next to it, a dropdown menu labeled '* Schultyp', and a dropdown menu labeled '* Feiertage und Ferien von'. At the bottom of the form, there are two buttons: 'Speichern' and 'Abbrechen'.

Abbildung 17: Vertragspartner-Seite (Auszug)

Wenn ein Importprofil, wie z.B. die Test-Importprofile, in der oberen *Listbox* „Importprofile“ (Schuelerdatenimportprofil Seite) in Abbildung A15 Seite xiv angezeigt werden, bedeutet dies, dass sie auch in diesem *Dropdown-GUI-Element* (Vertragspartner Seite) angezeigt werden. Alle logischen Prüfungen wurden bereits an die entsprechende Importprofilseite delegiert. Mit anderen Worten, wenn ein Profil in der *Dropdown-Liste* des Vertragspartners verfügbar und vollständig ist, bedeutet dies, dass in der Speichermethode (auf der Seite des Vertragspartners) keine zusätzlichen Prüfungen erforderlich sind.

5.3.5 Anpassung der Importassistent-Servlets

Das Importassistent-Servlet, hier einfach Importassistent genannt, war die komplexeste der drei Komponenten dieses Projekts, die erweitert werden musste. Dies lag daran, dass diese Seite nicht auf der Grundlage einer *JSP-HTML-Datei*, sondern mit Hilfe der *Java-Servlet-Technologie* entwickelt wurde. Einer der Gründe für diese Entscheidung war das Bedürfnis der Entwickler, bestimmte Datensätze mit Farben als Hintergrund darzustellen. Dieses Verhalten war mit der *JSP-JSF-Technologie* nicht realisierbar.

Die oben erwähnte Farbfunktionalität ist genau mit den hexadezimalen Farbwerten in der Datenbank verknüpft, die im entwickelten Schülerdatenimportprofil-Tool gespeichert sind (siehe den blauen Kasten unten rechts auf der Schülerdatenimportprofil-Seite Anhang A15 Seite xiv).

Der Importassistent verwendet eine ganz andere Syntax als eine *JSP-Seite* und war für den Autor ziemlich neu. Obwohl nur geringfügige Änderungen an der Logik erforderlich waren, bestätigt dies, was Onkel Bob sagte:

„Tatsächlich liegt das Verhältnis von Zeit, die mit Lesen gegenüber Schreiben verbracht wird, bei weit über 10 zu 1.“ (Martin "Uncle Bob", 2009)

In diesem Fall erwies sich diese Aussage als äußerst zutreffend.

Abnahmephase

Die alte Logik zur Auswahl der richtigen Farbe war fest in einer externen „properties“-Datei kodiert. Das bedeutet, dass die Farben nicht vom Benutzer ausgewählt werden konnten und für verschiedene Vertragspartner immer gleich blieben, ohne anpassbar zu sein. Es gab und gibt verschiedene Farben für verschiedene Zustände bestimmter Schülerdaten. Diese Zustände werden nun immer im rechten blauen Kasten auf der Seite "Schülerdaten-Importprofil" (Anhang A15 Seite xiv) angezeigt.

Letztendlich bestand die Anpassung darin, die Aufrufe der *hartcodierten* Werte in externen Dateien zu entkoppeln und stattdessen Aufrufe an die entsprechenden *Getter-Methoden* der Schülerdatenimportprofil und Schülerdatenimportprofilstatus-Objekte hinzuzufügen. Ein Beispiel ist in Abbildung 17 in Anhang A22 auf Seite xx zu sehen.

5.3.6 Zusammenfassung der Implementierung

Die Umsetzung der *Transactionklasse* sowie aller anderen neu erstellten *Klassen* der Anwendung, sowie die Anpassung der *Klassen*, die mit der Logik für den Importassistenten und die Verwaltung der Vertragspartner verbunden sind, folgten den Prinzipien von „Clean Code“ (Martin "Uncle Bob", 2009) und „Java by Comparison“ (Simon Harrer, 2018).

*„Clean code always looks like it was written by someone who cares.“
(Martin "Uncle Bob", 2009)*

Natürlich konnte der *Coding-Style* nicht vollständig nach persönlichen Vorlieben gestaltet werden. Vielmehr war es wichtig, eine flexible Anpassung vorzunehmen, indem der Autor seinen eigenen *Coding-Style* (als Praktikant noch nicht stark geprägt) mit den bereits vorhandenen Richtlinien und „good practices“ der Implementierung der Anwendung abglich. Obwohl dies zuweilen anspruchsvoll und zeitaufwendig war, stellte es eine äußerst wertvolle und lehrreiche Übung dar, bei der der Autor seine Vorlieben mit den bestehenden Vorgaben abwägen musste.

„Ein guter Code-Kommentar ist ein Kommentar, den man nicht schreiben musste, weil der Code selbsterklärend ist.“ (Simon Harrer, 2018)

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 A23: Auszug *Testprotokoll* Seite xxi zu entnehmen.

7 Einführungsphase

Nachdem der Code in zwei getrennten Abnahmeterminen überprüft und vom Projektbetreuer freigegeben wurde, kann er „committed“ werden. Für die Versionierung des Quellcodes wird *TortoiseSVN* verwendet.

Das neue Feature wurde wie üblich für andere SyABO *Commits*¹¹ über Jenkins¹² bereitgestellt. *Continuous Deployment* automatisiert den letzten Schritt der Bereitstellung des Codes an seinem endgültigen Bestimmungsort. In diesem Fall reduziert die Automatisierung die Anzahl der auftretenden Fehler, da die richtigen Schritte und *Best Practices* in Jenkins definiert sind.

Die Verwendung von Jenkins als Deployment-Server bietet zahlreiche Vorteile, darunter die flexible Anpassung der Prozesse an die Anforderungen der Anwendung. Ein *Jenkins-Server* ermöglicht es Entwicklern, Anwendungen zentral zu erstellen, zu testen und bereitzustellen.

Nach den Commit im Screenshot A24 Seite xxi ist wie *Jenkins-Verwaltung-GUI* aussehen kann.

8 Dokumentation

Neben der Projektdokumentation wurde auch eine betriebsinterne Entwicklungs- bzw. *Funktionsdokumentation* für das Firmen-Wiki von ASCI Systemhaus GmbH erstellt (Auszug in Screenshot A25 Abbildung 21-22 Seite xxii-xxiii). Darin sind die genauen Funktionen und Hinweise zur korrekten Implementierung auf einem internen *Informationsmanagementsystem* 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 sie (für SyABO-Anwender) selbsterklärend ist; durch die *Selbstbeschreibungsfähigkeit* wird unter anderem eines der Kernelemente der *Softwareergonomie* nach DIN (9241-220, 2019) aufgelistet. 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-Ausschnitt A26 auf Seite xxiii zu sehen ist.

Im Ausschnitt einer sogenannten "properties"-Datei, die Teil des *Frameworks* ist, werden die Meldungen in Form von Schlüssel-Wert-Paaren gespeichert. Diese *Map*¹³ ermöglicht eine vereinfachte Verwaltung und den Abruf der benötigten Werte, beispielsweise eines Satzes für Fehlermeldungen, von der *Oberflächen-Java-Klasse*. Die Meldungen, wie bereits erwähnt, sind integraler Bestandteil der *Benutzerdokumentation* und tragen zur Benutzerfreundlichkeit der Anwendung bei.

¹¹ Einreichen oder Übertragen von Änderungen an einer Subversion (SVN)-Repository mithilfe von TortoiseSVN. Ein Commit mit Tortoise ermöglicht es, lokal vorgenommene Änderungen in das zentrale Repository zu übertragen und sie anderen Teammitgliedern zur Verfügung zu stellen.

¹² Open-Source-Automatisierungsserver (aka Build Server oder CI-Server), für CI und CD. Jenkins ist eine Java Applikation, für die zahlreiche Plugins existiert.

¹³ Eine Map ist wie ein Wörterbuch aufgebaut. Jeder Eintrag besteht aus einem Schlüssel (key) und dem zugehörigen Wert (value). Es können beliebige Objekte hinzugefügt oder entfernt werden. Jeder Schlüssel darf in einer Map nur genau einmal vorhanden sein, wodurch jedes Schlüssel-Wert-Paar einzigartig 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 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 Designrichtlinien und die Benutzerfreundlichkeit besser eingehalten und gewährleistet, da sich eine ähnliche *UI-Struktur* auf einer anderen Seite der Anwendung befindet. Siehe Anhang A27: *Mockup* des zukünftige *Iteration-Zyklus* Seite xxiv.

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

Fazit

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 <i>JAVAdoc</i>	1 h
Gesamt	80 h

Tabelle 5: Detaillierte Zeitplanung

A2 Reg-Ex Code (Auszug)

```

/**
 * Verarbeitet die Aktion zum Hinzufügen von Importprofilspaltendaten.
 *
 * Diese Methode wird aufgerufen, wenn der Benutzer die Funktion zum Hinzufügen von Importprofilspaltendaten ausführt.
 * Regex Prüfung, ob Spaltenname mit '_' oder Buchstabe beginnt, Prüfung, ob enthält erlaubte Zeichen
 * und ob enthält nur Ziffern (nicht erlaubt).
 * Sie überprüft, ob alle erforderlichen Informationen vorhanden sind, und fügt dann eine neue Spaltenbezeichnung hinzu.
 * Nach dem Hinzufügen werden die Auswahl und die Eingabe für Spaltenname und Zustandsdaten zurückgesetzt.
 *
 * @see SchuelerdatenimportprofilBean#getSchuelerdatenimportprofil()
 * @see SchuelerdatenimportprofilBean#getSchuelerdatenimportprofilspalteMap()
 * @see Schuelerdatenimportprofilspalte
 */
public void buttonImportprofilspaltendatenUebernehmen_action()
{
    boolean ok = true;
    SchuelerdatenimportprofilBean bean = holeSchuelerdatenimportprofilBean();
    Schuelerdatenimportprofil schuelerdatenimportprofil = bean.getSchuelerdatenimportprofil();

    Integer spalte = (Integer) this.listBoxImportprofilspaltendaten.getSelected();
    String spaltenbezeichnung = (String) this.textFieldImportprofilspaltenname.getText();

    boolean gueltig = Pattern.matches(regex: "^([\\p{L}_\\d]*\\d*\\.?)\\s*[-]*[\\p{L}\\d_\\.\\s]*$", input: spaltenbezeichnung);

    if (!gueltig)
    {
        ok = false;
        this.textFieldImportprofilspaltenname.setValid(valid: false);
        FacesMessageUtils.error(bundleName: this.bundleMeldungen, key: "spaltenname_ungueltig", params: null);
    }
    else
    {
        if (spaltenbezeichnung.replaceAll(regex: "[\\p{L}]", replacement: "").isEmpty())
        {
            ok = false;
            this.textFieldImportprofilspaltenname.setValid(valid: false);
            FacesMessageUtils.error(bundleName: this.bundleMeldungen, key: "spaltenname_ohne_buchstaben", params: null);
        }
    }

    if (ok)
    {

```

Abbildung 1: Reg-Ex Code (Auszug)

A3 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.
2. Mit Hilfe der zu entwickelnden Importprofilauswahl (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 (Profil):
 - Profilbezeichnung.
 - Zeichencodierung der Importdateien.
 - Verwendetes Trennzeichen.
 - Verwendetes Text Trennzeichen.
 - 3.2. Zweite Tabelle (Spalten):
 - Referenz auf das Hauptprofil.
 - Spaltenbezeichnung.
 - Interne Referenz.
 - Datum Format und Zahlen Format (nice to have)¹⁴.
 - 3.3. Dritte Tabelle (Status):
 - Referenz auf das Hauptprofil.
 - Farbe.
 - Interne Referenz.
4. Mit Hilfe der zu entwickelnden Importassistent-Servlet (Seite Ergänzung) müssen...
 - 4.1. Importe von CSV-Dateien werden durchgeführt, wenn der ausgewählte Vertragspartner eine Zuordnung zu einem vorgelegten Importprofil hat.
 - 4.2. Importe von CSV-Dateien werden nicht durchgeführt, wenn der ausgewählte Vertragspartner keine Zuordnung zu einem vorgelegten Importprofil hat.
 - 4.3. In diesem Fall (4.2) wird eine Fehlermeldung generiert und dem Benutzer angezeigt.
5. Sonstige Anforderungen
 - 5.1. Die Anwendung muss um Webbrowser Mozilla Firefox lauffähig sein.
 - 5.2. Da sich der Administrationsabteilung und externen Vertragspartnern auf diese Daten verlassen, muss die Anwendung korrekte Daten liefern.
 - 5.3. Ein Single-Sign-On sollte an der Anwendung möglich sein.

¹⁴ Aus Zeitgründen wurde es nicht in die erste Iteration Implementiert.

Fazit

A4 Verwendete Ressourcen

Software

- Microsoft Windows 11 Pro
- NetBeans IDE
- Firefox, Microsoft Edge
- SyABO
- Jenkins
- Microsoft Azure DevOps
- Notepad++
- pgAdmin4
- Interne Dokumentationen: Redmine ©
- Versionsverwaltung: TortoiseSVN
- Ticketsystem/Projektmanagement: Redmine ©
- Interne Kommunikation: Jitsi Meet

Hardware

- Notebook
- All in One Home pc
- Tastatur, Maus
- Personal
- Auszubildender (Umsetzung)
- Ausbilder (Festlegung der Anforderungen)
- Auftragsgeber/Projektleiter

Programmiersprachen, Frameworks, APIs

- Java
- jQuery
- Spring
- (SQL)

Hilfsunterlagen

- Spring Dokumentation (siehe Literaturverzeichnis)
- iBatis Dokumentation (siehe Quellenverzeichnis)

Fazit

A5 Ticket

Feature #2248



Import von Schülerdaten - Importprofil

Von Lutz Langerwisch vor mehr als 2 Jahren hinzugefügt. Vor 26 Tagen aktualisiert.

Status:	Neu	Beginn:
Priorität:	Niedrig	Abgabedatum:
Zugewiesen an:	 Marco Garagna	% erledigt:
Kategorie:	allg. Programmerweiterung	Geschätzter Aufwand:
Zielversion:	-	Totalhours:
Estimatedhours:		
Hours:		

Beschreibung

Der Import von Schülerdaten müsste m.E. flexibler gestaltet werden. Derzeit wird die Festlegung welche Spalten der Importdatei wie zu interpretieren ist in der Datei importoptionen.properties gespeichert. Das erscheint unter anderem unter dem Aspekt, dass es mehrere Datenlieferanten, die mit jeweilig eigenen Formaten die Dateien liefern können, nicht flexibel Identifikation und den den Import vorgesehen sind, sollte es auch mehrere Importkonfiguration mit identischen oder unterschiedlichen Spaltenbezeichnungen. Definition der Spalten in der übergebenen Datei in einer Tabelle mit der Einrichtung als Datenlieferant verbunden und mit der Information über den verwendete. Gibt es auch die Möglichkeit die CSS-Farbdefinitionen für die zuverwendenden Farben zur Markierung der Daten für die Nachbearbeitung in einer Tabelle zu s Anwendung zu bringen? Wenn ja, sollte es gemacht werden. Dann sollte man jedem Datenlieferanten ein mindestens ein Importprofil anlegen können, das man dann beim Datenimport aus einer Liste der Lieferanten m

Unteraufgaben

Abbildung 2: Ticket

A6 Amortisationsdiagramm

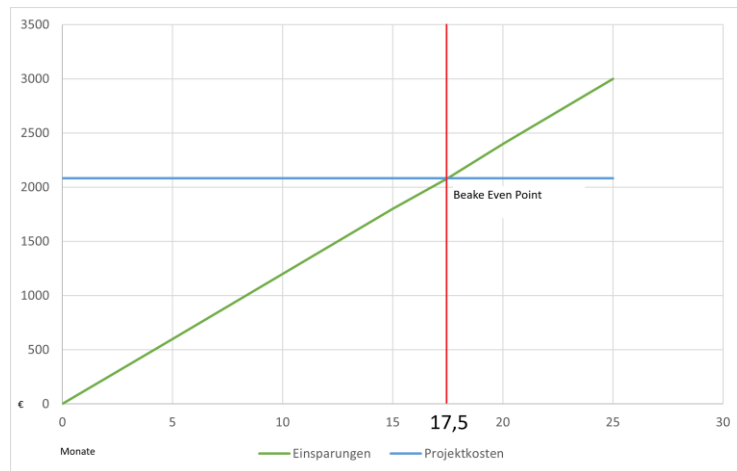


Abbildung 3: Amortisationsdiagramm

A7 Use-Case-Diagramm

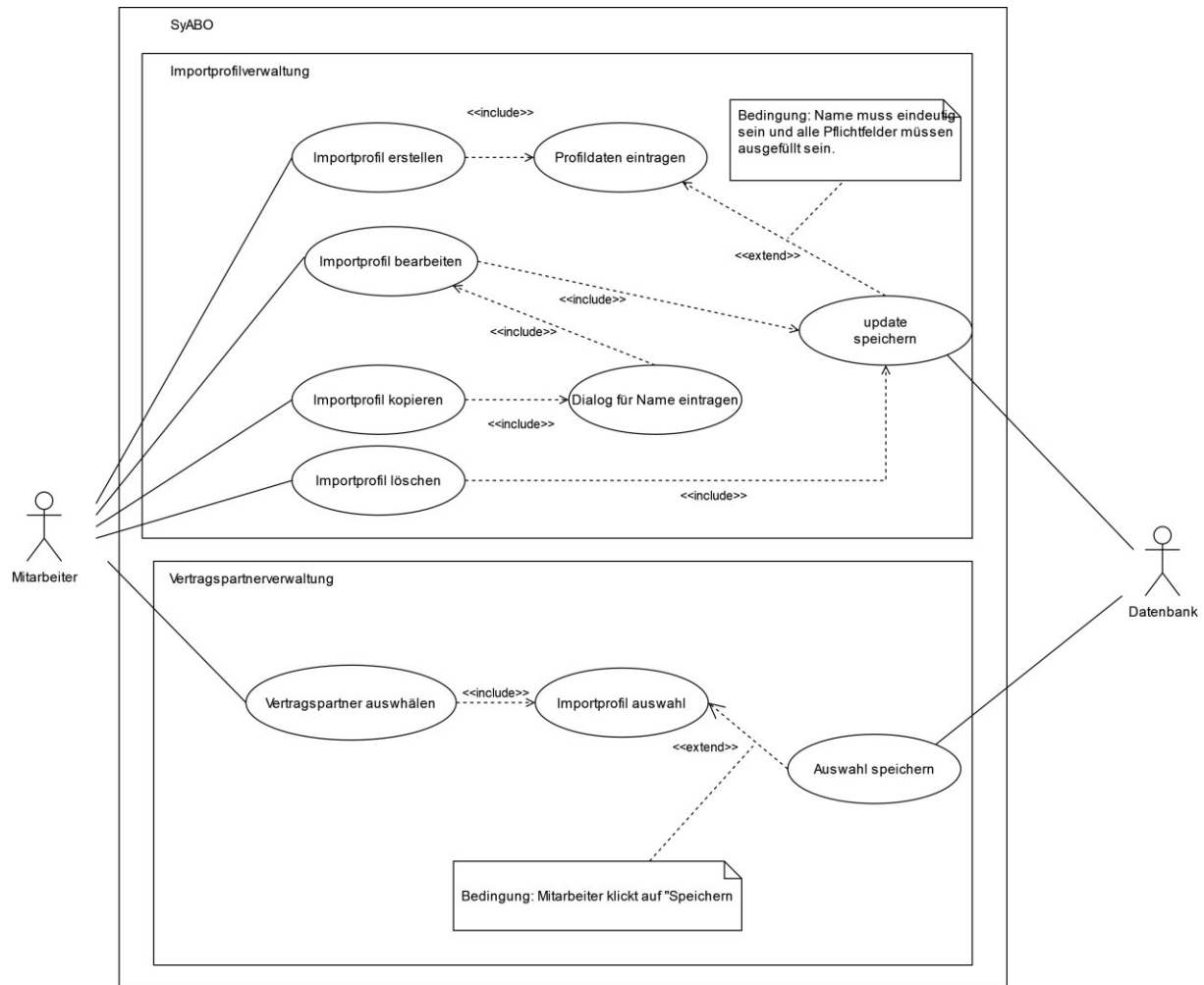


Abbildung 4: Use-Case-Diagramm

A8 Pflichtenheft (Auszug)

Zielbestimmung

1. Plattform

- 1.1. Zur Entwicklung der Anwendung wird mit NetBeans für *Java EE* eingesetzt.
- 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 JPA.
- 2.2. *Dabei wird mit der Implementierung ORM iBatis gearbeitet.*
- 2.3. Die Daten werden in einer *PostgreSQL* Datenbank gespeichert.

3. Benutzeroberflächen

- 3.1. Die Benutzeroberflächen werden mit HTML5 - CSS3 umgesetzt.
- 3.2. Um dies zu generieren, wird die JSF-Technologie genutzt.
- 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.

A9 Oberflächenentwürfe

Aufgaben Beförderung Rechnungswesen Auswertungen Grunddaten Administration Einstellungen Fenster Abmelden PVGS

Importprofil

Importprofil

Profil_A
Profil_B
Profil_C

Neu Bearbeiten Löschen Kopieren

* Bezeichnung

* Zeichensatz

* Trennzeichen

Spaltenschlüssel	spalte	spaltenname	Zustand
	Schülernummer		
	Name		
	Vorname		
	Geburtsdatum		
	PLZ		
	Gemeinde		
	Ort		
	Straße		
	Hausnummer		
	Einstiegshaltestelle (Name)		
	Schule (Name)		
	Klasse		

Bearbeiten Entfernen

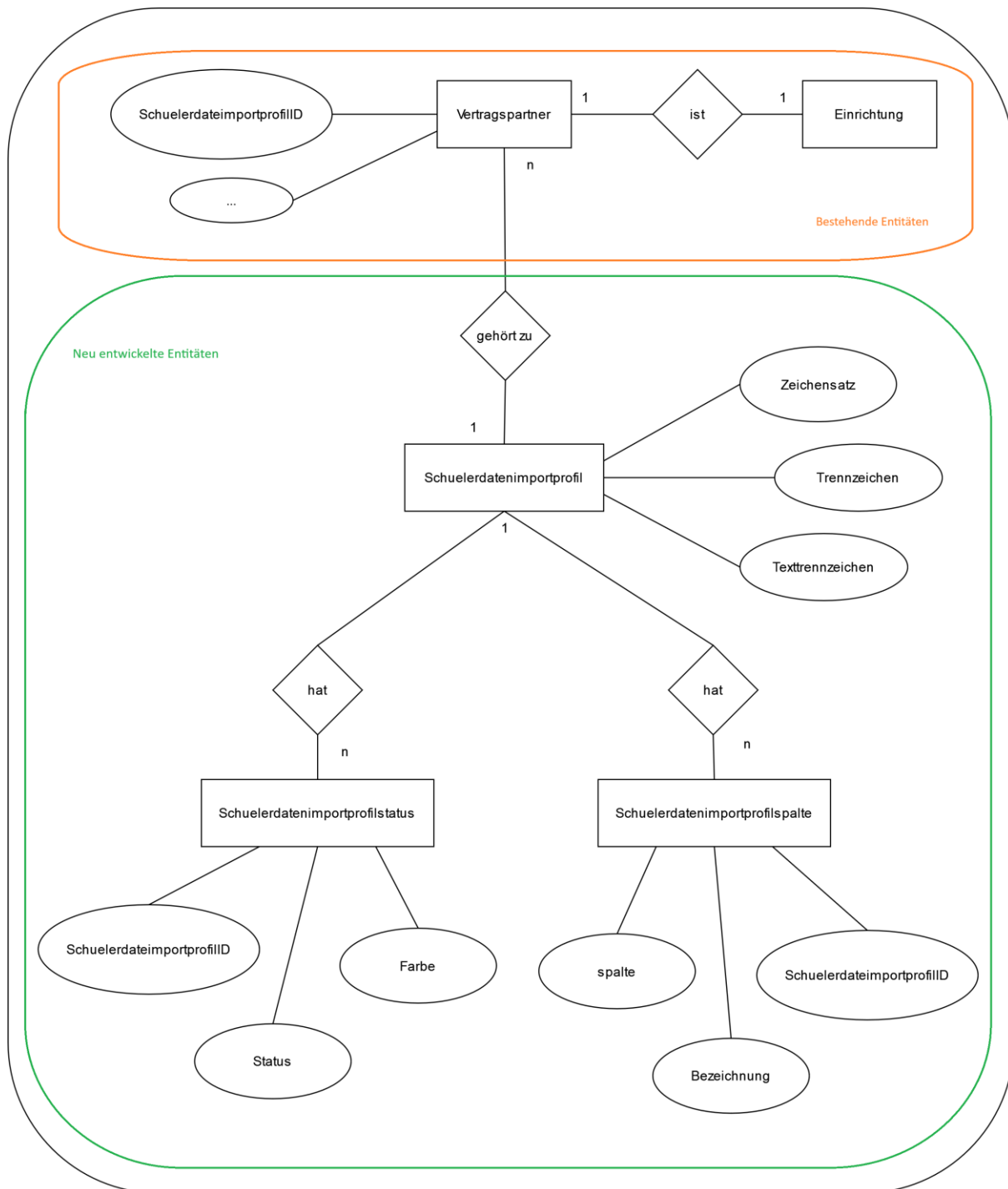
Spaltenname Hinzufügen

Speichern

* Farbe Übernehmen

Abbildung 5: Benutzeroberfläche Mockup

A10 Datenbankmodell



Jede Entität besitzt künstliche IDs, im oberen Diagramm waren sie nicht berücksichtigt.

Abbildung 6: Entity-Relationship-Model

A11 Datenbankschema

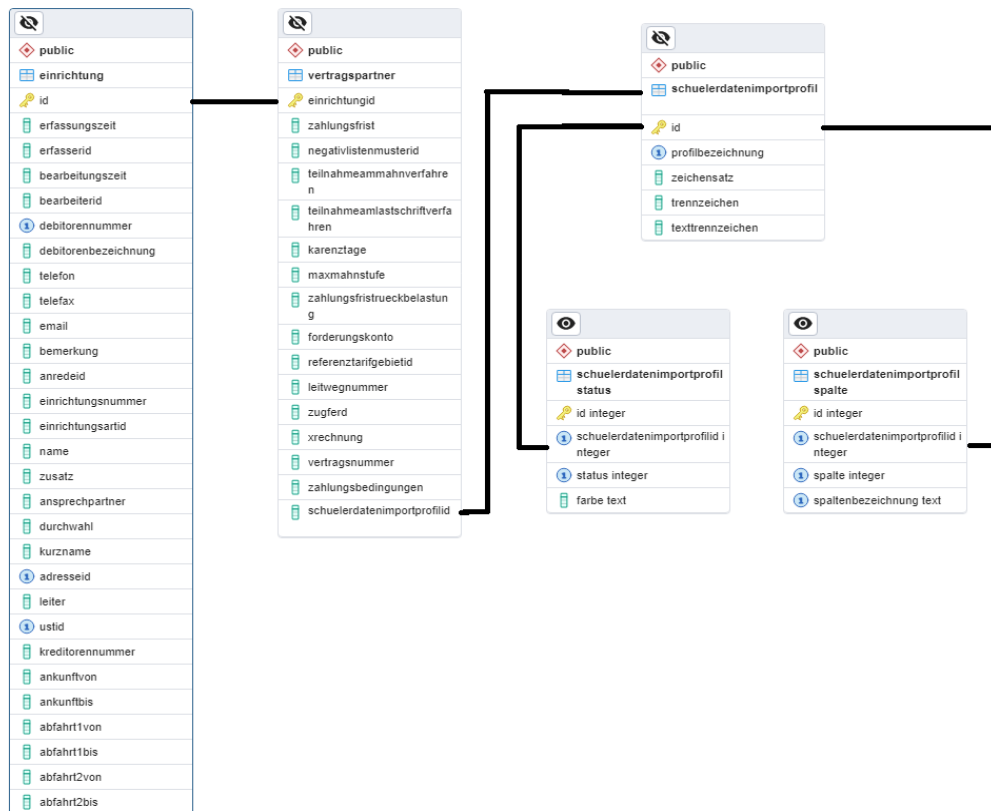


Abbildung 7: Datenbankschema

A12 Aktivitätsdiagramm

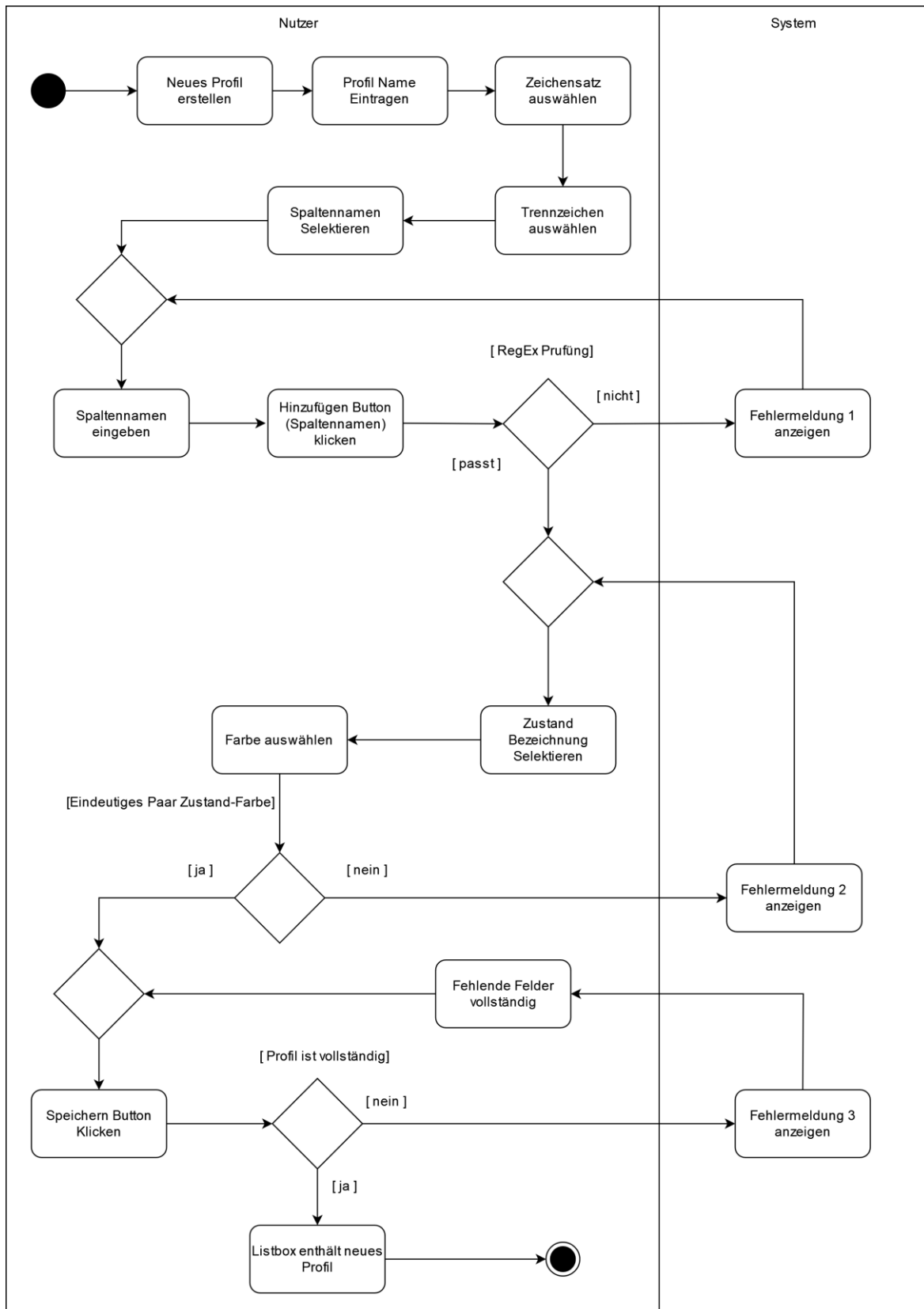


Abbildung 8: Verfahren zum Einfügen eines Importprofils

A13 Klassendiagramm

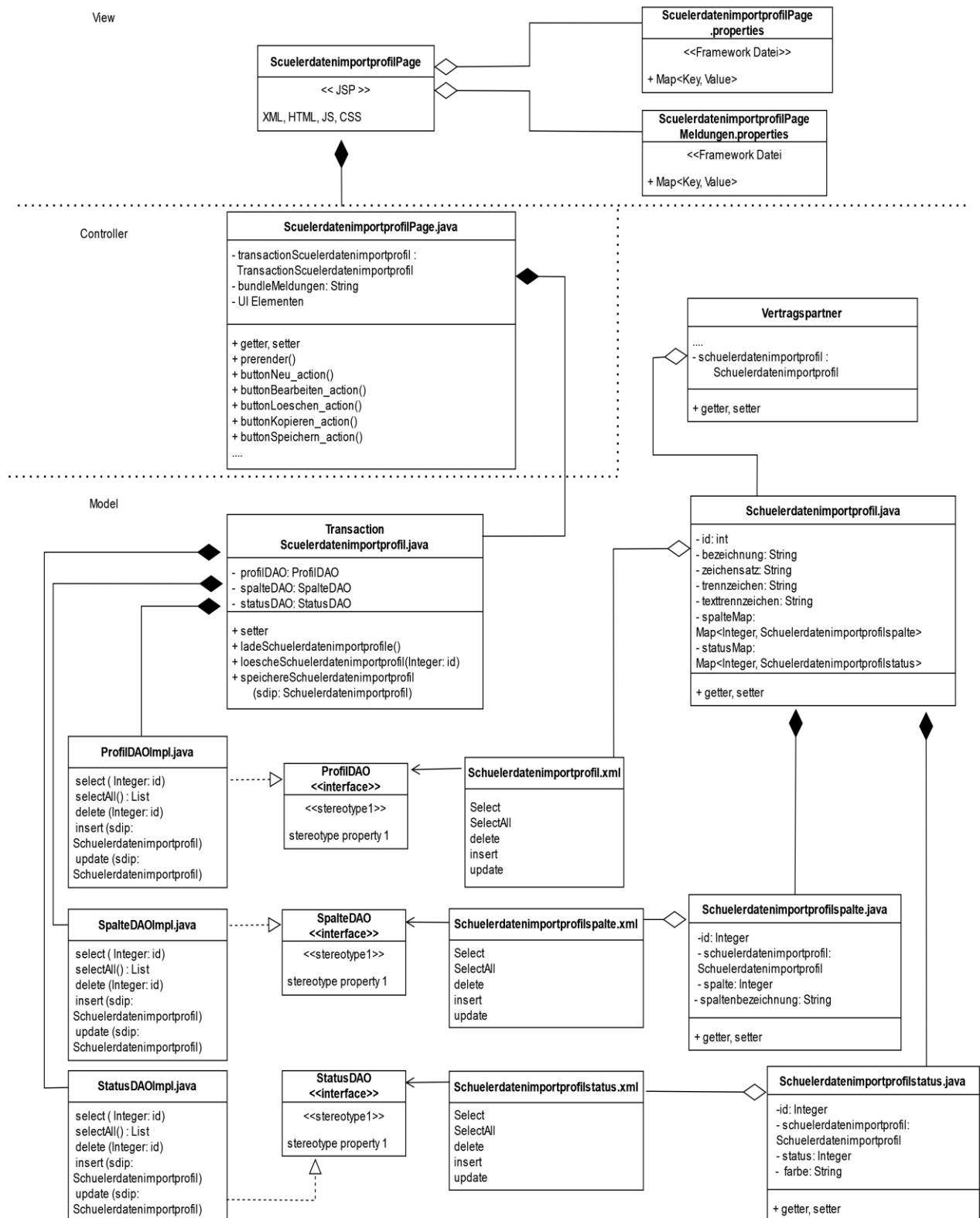


Abbildung 9: Klassendiagramm

A14 Unique Datenbank „Constraints“

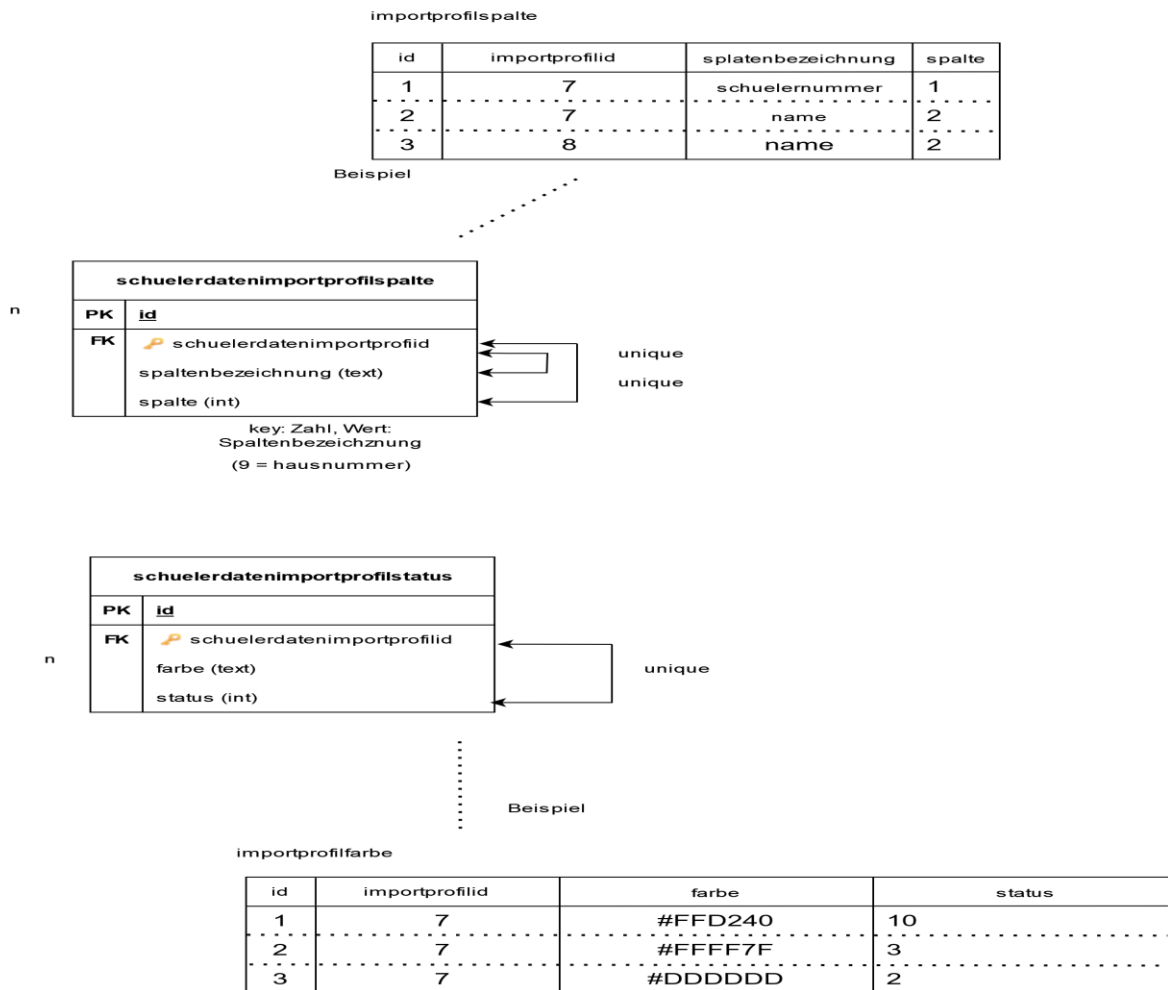


Abbildung 10: Unique Datenbank „Constraints“

IMPORTPROFIL-TOOL

Verwaltung von Profilen für den Datenimport von Schülerdaten

Fazit

A15 Screenshots der Schülerdatenimportprofil Seite

The screenshot displays the 'Schülerdatenimportprofil' window. It features an 'Importprofil' list with 'Profiltest_A', 'Profiltest_B', and 'Profiltest_C'. Below this are buttons for 'Neu', 'Bearbeiten', 'Löschen', and 'Kopieren'. A 'Bezeichnung' field is set to 'Profiltest_A'. The 'Zeichensatz' is 'UTF-8', 'Trennzeichen' is ';', and 'Texttrennzeichen' is '"'. The 'Spaltendaten' section shows a mapping of columns (e.g., 'Adressezusatz', 'Bemerkung') to a 'Spaltenschlüssel' and 'Spaltenname'. The 'Zustandsdaten' section shows a list of states (e.g., 'Abweichend', 'Aktualisiert') with 'Vorhanden' and 'Definition' columns. A 'Farbe' field is also present.

Abbildung 11: Anzeige der GUI-Unterteilung Design

A16 Screenshots der Anwendung

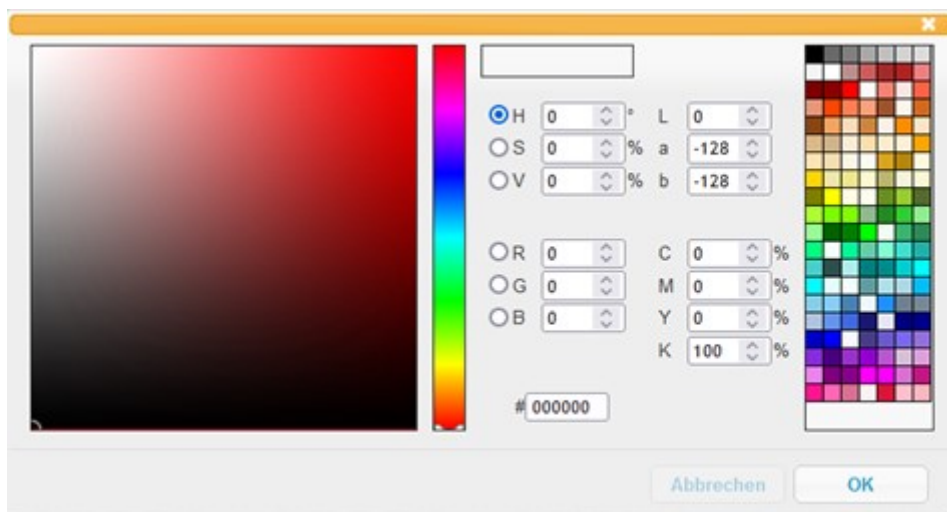


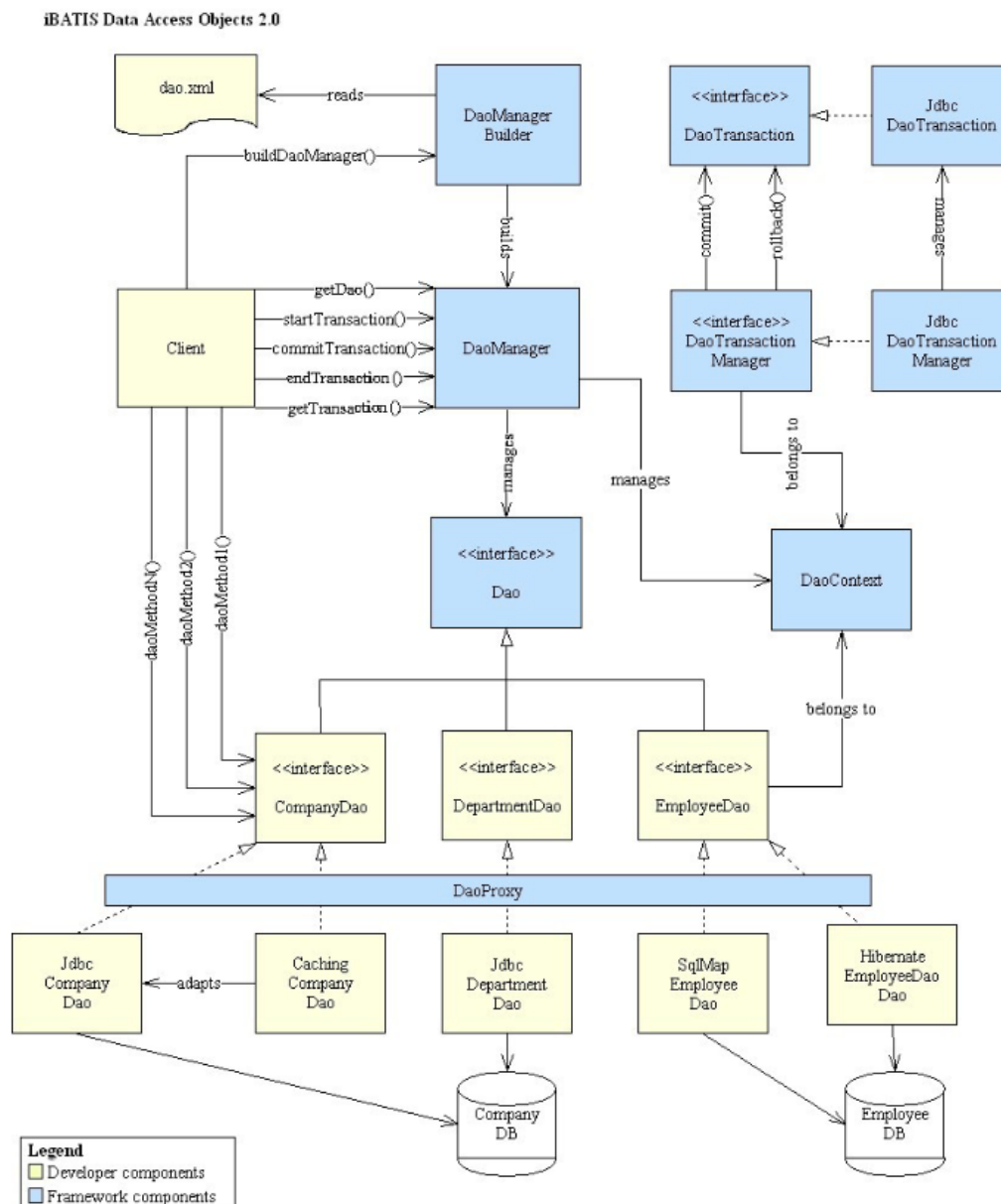
Abbildung 12: ColorPicker Popup Element

A17 ColorPicker Script (JS)

```
<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 13: ColorPicker Script (JS)

A18 iBatis Framework Dokumentation (Auszug)



<http://www.ibatis.com>

by Clinton Begin

Abbildung 14: iBatis Data Access Objects Developer Guide (Auszug)

A19 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());
```

Fazit

```
        schuelerdatenimportprofilstatus.setSchuelerdatenimportprofil(schuele  
rdatenimportprofil);  
  
        schuelerdatenimportprofil.getSchuelerdatenimportprofilstatusMap().put  
(schuelerdatenimportprofilstatus.getId(),  
        schuelerdatenimportprofilstatus);  
    }  
    return schuelerdatenimportprofilMap;  
}  
  
public void  
loescheSchuelerdatenimportprofil(Integer schuelerdatenimportprofilId)  
{  
    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());  
}
```

Fazit

```

    for (Schuelerdatenimportprofilstatus schuelerdatenimportprofilstatus :
        schuelerdatenimportprofil.getSchuelerdatenimportprofilstatusMap().values())
    {
        if (schuelerdatenimportprofilstatus.getId() == null)
        {
            Integer id =
                this.schuelerdatenimportprofilstatusDAO.selectNextId();
            schuelerdatenimportprofilstatus.setId(id);
        }

        this.schuelerdatenimportprofilstatusDAO.insert(schuelerdatenimportp
            rofilstatus);
    }
}

```

Listing 1: Klasse `TransactionSchuelerdatenimportprofil`

A20 Architekturdiagramm (Auszug)

Daten Lebenszyclus

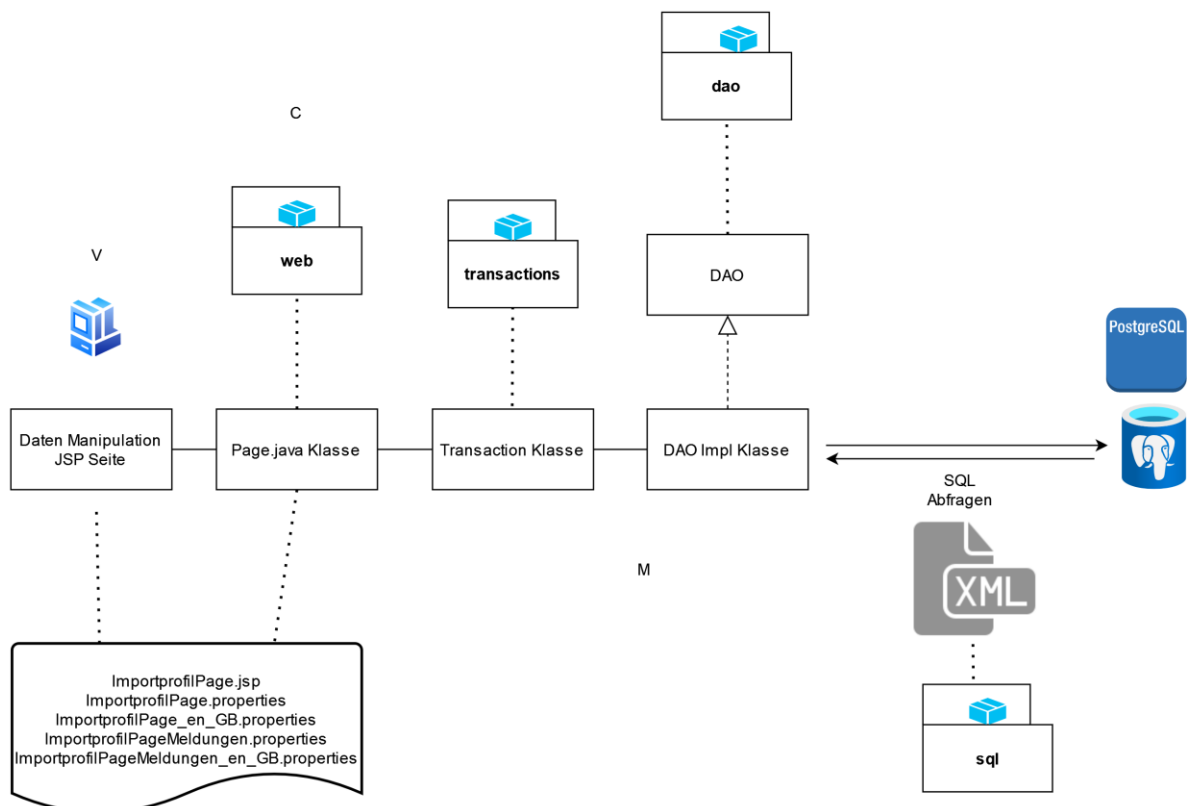


Abbildung 15: Auszug aus der Architekturdiagramm

A21 Schuelerdatenimportprofil.java Speichern Methode (Auszug)

```
public void buttonSpeichern_action()
{
    Locale locale = getSessionBean1().getLocale();
    boolean ok = true;
    SchuelerdatenimportprofilBean bean = holeSchuelerdatenimportprofilBean();
    Schuelerdatenimportprofil schuelerdatenimportprofil = bean.getSchuelerdatenimportprofil();

    String importprofilbezeichnung = (String) this.textFieldImportprofilbezeichnung.getText();
    String importprofilzeichensatz = (String) this.dropDownZeichensatz.getSelected();
    String importprofiltrennzeichen = (String) this.dropDownTrennzeichen.getSelected();
    String importprofiltexttrennzeichen = (String) this.dropDownTexttrennzeichen.getSelected();

    if (bean.getSchuelerdatenimportprofilspalteMap().isEmpty())
    {
        ok = false;
        FacesMessageUtils.error(bundleName: this.bundleMeldungen, key: "importprofilspalten_leer", params: null);
    }

    Map<Integer, String> daten = TextUtils.lokalisiere(bundleName: this.bundleLabel, locale, listName: "status");
    for (Integer key : daten.keySet())
    {
        if (! bean.getSchuelerdatenimportprofilstatusMap().containsKey(key))
        {
            ok = false;
            FacesMessageUtils.error(bundleName: this.bundleMeldungen, key: "importprofilzustaeude_unvollstaendig", params: null);
            break;
        }
    }

    if (ok)
    {
        for (Schuelerdatenimportprofil sdip : bean.getSchuelerdatenimportprofilMap().values())
        {
            if (! sdip.getId().equals(obj: schuelerdatenimportprofil.getId()) && sdip.getProfilbezeichnung().equalsIgnoreCase(
                ok = false;
                this.textFieldImportprofilbezeichnung.setValid(valid: false);
                FacesMessageUtils.error(bundleName: this.bundleMeldungen, key: "importprofilbezeichnung_eindeutig", params: null);
            }
        }
    }
}
```

Abbildung 16: Speichern Methode (Auszug)

Listing 2: Schuelerdatenimportprofil.java

A22 Importassistent-Servlet Anpassung (Auszug)

```
* 139 → this.transactionBestellung.bestellungenvertragspartner(bestellungBean, vertragspartnerId);
* 140 → Vertragspartner vertragspartner = bestellungBean.getVertragspartnerMap().get(vertragspartnerId);
* 141 → if (vertragspartner.getSchuelerdatenimportprofil() != null)
* 142 → {
* 143 →     HashMap<Integer, Bestelldetailstatus> statuswerte = bestellungBean.getStatuswerte();
* 144 →
* 145 →     Schuelerdatenimportprofil schuelerdatenimportprofil = vertragspartner.getSchuelerdatenimportprofil();
* 146 →     TreeMap<Integer, Schuelerdatenimportprofilstatus> schuelerdatenimportprofilstatusMap =
* 147 →         schuelerdatenimportprofil.getSchuelerdatenimportprofilstatusMap();
* 148 →     for (Schuelerdatenimportprofilstatus schuelerdatenimportprofilstatus : schuelerdatenimportprofilstatusMap.values())
* 149 →     {
* 150 →         Bestelldetailstatus bestelldetailstatus = statuswerte.get(schuelerdatenimportprofilstatus.getStatus());
* 151 →         bestelldetailstatus.setFarbe(schuelerdatenimportprofilstatus.getFarbe());
* 152 →     }
* 153 → }
154 → }
155 → else if (aktion.equals("bestellungennurunerledigte"))
156 → {
157 →     boolean nurunerledigtebestellungen = params.get("nurunerledigtebestellungen") != null;
158 →     this.transactionBestellung.bestellungennurunerledigte(bestellungBean, nurunerledigtebestellungen);
159 → }
160 → else if (aktion.equals("bestellungenneu"))
161 → {
* 162 →     Vertragspartner vertragspartner = bestellungBean.getVertragspartnerMap().get(bestellungBean.getVertragspartnerId());
* 163 →     if (vertragspartner.getSchuelerdatenimportprofil() != null)
* 164 →     {
* 165 →         this.transactionBestellung.bestellungenneu(bestellungBean);
* 166 →     }
* 167 →     else
* 168 →     {
* 169 →         errorMsg = "Bevor Sie diese Aktion ausführen können, müssen Sie dem ausgewählten Vertragspartner ein Schülerdatenimportprofil zuweisen.";
* 170 →     }
}
```

Abbildung 18: Auszug der Importassistent-Servlet Anpassung

Fazit

A23 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 19: Auszug Testprotokoll

A24 Jenkins Build-Prozess GUI (Auszug)



























Build	Datum ↑	Status
 syabo #533	54 Minuten	Stabil 
 asciutlis #79	57 Minuten	Stabil 
 raildispo #1826	1 Monat 6 Tage	Stabil 
 jsfapp #149	1 Monat 9 Tage	Stabil 
 ascitypes #422	1 Monat 24 Tage	Stabil 
 osm #149	3 Monate 7 Tage	Stabil 
 mobileclient #4	5 Jahre 5 Monate	Stabil 
 fahrplanung #70	7 Jahre 4 Monate	Stabil 
 pebbledruck #2	9 Jahre 2 Monate	Stabil 
 tpnSOAPClientStubs #1	10 Jahre	Stabil 
 pdfdruck #4	10 Jahre	Stabil 
 pdfdruck #3	10 Jahre	Stabil 
 glas #2	10 Jahre	Stabil 

Abbildung 20: Screenshot aus der Jenkins-Server-GUI

IMPORTPROFIL-TOOL

Verwaltung von Profilen für den Datenimport von Schülerdaten

Fazit

A25 WIKI-Dokumentation (Auszug)

Übersicht
Aktivität
Tickets
Neues Ticket
Gantt-Diagramm
Kalender
Dokumente
Wiki

WikiStart » TypesIndex »

WikiStart - TypesIndex

Name: Schuelerdatenimportprofil

Alias: sdip

Beschreibung:

Anwenderzugriff: ...

Attribute:

Name	Typ	NULL	Beschreibung
id	Integer		Generierter Primärschlüssel
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:

Wiki
Hauptseite
Seite

Abbildung 21: Auszug aus der WIKI-System

Fazit

Constructor Summary		
Constructors		
Constructor	Description	
<code>SchuelerdatenimportprofilPage()</code>		

Method Summary		
All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
void	<code>buttonBearbeiten_action()</code>	Verarbeitet die Aktion, wenn die Schaltfläche "Bearbeiten" unter Importprofil-ListBox geklickt wird.
void	<code>buttonImportprofilspaltendatenBearbeiten_action()</code>	Verarbeitet den Aktions-Event des "Bearbeiten"-Buttons für Importprofilspaltendaten.
void	<code>buttonImportprofilspaltendatenEntfernen_action()</code>	Verarbeitet die Aktion des Entfernens von Importprofilspaltendaten.
void	<code>buttonImportprofilspaltendatenUebernehmen_action()</code>	Verarbeitet die Aktion zum Hinzufügen von Importprofilspaltendaten.
void	<code>buttonImportprofilzustandsdatenUebernehmen_action()</code>	Verarbeitet den Action-Event des Übernehmen-Buttons für die Importprofilzustandsdaten.
void	<code>buttonKopieren_action()</code>	Erstellt eine Kopie des ausgewählten Schülerdatenimportprofils.
void	<code>buttonLoeschen_action()</code>	Löscht das ausgewählte Schülerdatenimportprofil, sofern es nicht bereits von einem Vertragspartner verwendet wird.
void	<code>buttonNeu_action()</code>	Setzt das Schülerdatenimportprofil und zugehörige Daten zurück auf den Anfangszustand und verwirft eingereichte Werte in Formularfeldern.
void	<code>buttonSpeichern_action()</code>	Verarbeitet den Speichern-Button-Event für ein Schülerdatenimportprofil.
void	<code>destroy()</code>	Callback method that is called after rendering is completed for this request, if <code>init()</code> was called (regardless of whether or not this was the page that was actually

Abbildung 22: Auszug aus der Entwicklerdokumentation mit *JavaDoc*

A26 Benutzerdokumentation (Auszug)

<code>importprofilspalten_leer</code>	=Es sind keine Spalten definiert.
<code>importprofilbezeichnung_eindeutig</code>	=Die Bezeichnung des Importprofils muss eindeutig sein.
<code>importprofilzustaende_unvollstaendig</code>	=Alle Importprofilzustände müssen definiert werden.
<code>spaltenname_eindeutig</code>	=Der Spaltenname muss eindeutig sein.
<code>spaltenname_ungueltig</code>	=Der Spaltenname ist ungültig. Der Spaltenname darf nur Buchstaben, Ziffern, '_' und '-' enthalten. De
<code>spaltenname_ohne_buchstaben</code>	=Der Spaltenname enthält keine Buchstaben. Der Spaltenname darf nur Buchstaben, Ziffern, '_' und

Abbildung 23: Auszug aus der Benutzerdokumentation

IMPORTPROFIL-TOOL

Verwaltung von Profilen für den Datenimport von Schülerdaten

Fazit

A27 Mockup des künftigen Iteration-Zyklus

Schülerdatenimportprofil

Importprofil

Neu

Bearbeiten

Löschen

Kopieren

* Bezeichnung

Zeichensatz

Trennzeichen

Texttrennzeichen

Spaltendaten

Spaltenschlüssel

Spaltenname

Zuordnung

Adresszusatz

Bemerkung

Eigenanteil

Einsteigehaltestelle (Name)

Einsteigsort

FSV-Info

Fahrkarten-Gültigkeitsbereich von

Fahrkarten-Gültigkeitsbereich bis

Geburtsdatum

Geltungsbereich nach

Geltungsbereich von

Gemeinde

Gemeinde des Sorgeberechtigten

Hausnummer

Hausnummer des Sorgeberechtigten

Klasse

Bearbeiten

Entfernen

Spaltenname

Übernehmen

Speichern

Zustandsdaten

* Abweichend

* Aktualisiert

* Duplikat

* Fahrgast gelöscht

* Schüler erzeugbar

* Schüler nicht erzeugbar

* Ticket erzeugt

* Ticket gelöscht

* Vertrag endet

* Verworfen

Abbildung 24: Zukünftige GUI-Mockup

Abbildungsverzeichnis

Abbildung 1: Reg-Ex Code (Auszug).....	ii
Abbildung 2: Ticket (Auszug).....	v
Abbildung 3: Amortisationsdiagramm	v
Abbildung 4: Use-Case-Diagramm	vi
Abbildung 5: Benutzeroberfläche Mockup	viii
Abbildung 6: Entity-Relationship-Model	ix
Abbildung 7: Datenbankschema	x
Abbildung 8: Verfahren zum Einfügen eines Importprofils	xi
Abbildung 9: Klassendiagramm	xii
Abbildung 10: Unique Datenbank „Constraints“	vi
Abbildung 11: Anzeige der GUI-Unterteilung Design	xiv
Abbildung 12: ColorPicker Popup Element.....	xiv
Abbildung 13: ColorPicker Script (JS).....	xv
Abbildung 14: iBatis Data Objects Developer Guide (Auszug).....	xvi
Abbildung 15: Auszug der Architekturdiagramm	vix
Abbildung 16: Speichern Methode (Auszug).....	xx
Abbildung 17: Vertragspartner-Seite (Auszug).....	vi
Abbildung 18: Auszug der Importassistent-Servlet Anpassung.....	vi
Abbildung 19: Auszug Testprotokoll	vi
Abbildung 20: Screenshot aus der Jenkins-Server-GUI.....	vi
Abbildung 21: Auszug aus der WIKI-System	vi
Abbildung 22: Auszug aus der Entwicklerdokumentation mit <i>JavaDoc</i>	xxiii
Abbildung 23: Auszug aus der Benutzerdokumentation.....	xxiii
Abbildung 24: Zukünftige GUI-Mockup	viv

Tabellenverzeichnis

Tabelle 1: Grobe Zeitplanung	2
Tabelle 2: Kostenaufstellung	4
Tabelle 3: Entscheidungsmatrix.....	7
Tabelle 4: Soll-/Ist-Vergleich.....	15
Tabelle 5: Detaillierte Zeitplanung	ii

Verzeichnis der Listings

Listing 1: Klasse <code>TransactionSchuelerdatenimportprofil</code>	xix
Listing 2: <code>Schuelerdatenimportprofil.java</code>	xx

Fazit



Quellenverzeichnis

9241-220, D. E. I., 2019. *DIN-Normenausschuss Ergonomie (NAErg)*. s.l.:s.n.

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

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

Martin "Uncle Bob", R. C., 2009. *Clean Code*. s.l.:s.n.

Sedlacek, A. R. - R., 2022. *Cleverer Tipps für die Projektarbeit*. s.l.:u-form.

Simon Harrer, J. L. L. D., 2018. *Java by Comparison*. s.l.:s.n.

ColorPicker JQuery Dokumentation

<https://www.redprinting.co.kr/assets/js/colorpicker-master/index.html>