IPA-Bericht  
Code-Generator



Autor Luan Caduff

Klasse ISO-20

Datum 03. Mai 2024

Firma Evernex IT Services Switzerland AG

Table of Contents

[1 Teil 1 4](#_Toc165651155)

[1.1 Dokumenteninformationen 4](#_Toc165651156)

[1.1.1 Historie 4](#_Toc165651157)

[1.1.2 Eigenschaften 4](#_Toc165651158)

[1.2 Aufgabenstellung 5](#_Toc165651159)

[1.3 Projektorganisation 7](#_Toc165651160)

[1.4 Projektmethode 7](#_Toc165651161)

[1.5 Deklaration der Vorkenntnisse 8](#_Toc165651162)

[1.6 Deklaration der Vorarbeiten 8](#_Toc165651163)

[1.7 Deklaration der benutzen Firmenstandards 8](#_Toc165651164)

[1.8 Organisation der Arbeitsergebnisse 9](#_Toc165651165)

[1.9 Zeitplan 10](#_Toc165651166)

[1.10 Arbeitsjournal 11](#_Toc165651167)

[1.10.1 Tag 1, 18. April 2024 11](#_Toc165651168)

[Tag 2, 19. April 2024 12](#_Toc165651169)

[1.10.2 Tag 3, 23. April 2024 13](#_Toc165651170)

[1.10.3 Tag 4, 24. April 2024 14](#_Toc165651171)

[1.10.4 Tag 5, 25. April 2024 15](#_Toc165651172)

[1.10.5 Tag 6, 26. April 2024 16](#_Toc165651173)

[1.10.6 Tag 7, 30. April 2024 17](#_Toc165651174)

[1.10.7 Tag 8, 01. Mai 2024 18](#_Toc165651175)

[1.10.8 Tag 9, 02. Mai 2024 19](#_Toc165651176)

[1.10.9 Tag 10, 03. Mai 2024 20](#_Toc165651177)

[1.10.11 Zusammenfassung Zeitplan 21](#_Toc165651178)

[2 Teil 2 22](#_Toc165651179)

[2.1 Management Summary (Kurzfassung des IPA-Berichts) 22](#_Toc165651180)

[2.1.1 Ausgangslage 22](#_Toc165651181)

[2.1.2 Vorgehen 22](#_Toc165651182)

[2.1.3 Ergebnis 22](#_Toc165651183)

[2.2 Informieren 23](#_Toc165651184)

[2.2.1 Aufgabestellung 23](#_Toc165651185)

[2.2.2 Technologien 23](#_Toc165651186)

[2.2.3 Use-Cases 24](#_Toc165651187)

[2.3 Planen 26](#_Toc165651188)

[2.3.1 GUI MockUps 26](#_Toc165651189)

[2.4 Entscheiden 29](#_Toc165651190)

[2.5 Realisieren 30](#_Toc165651191)

[2.5.1 Tabellen-Auswahl Frontend 30](#_Toc165651192)

[2.5.2 Config-Formular 31](#_Toc165651193)

[2.5.3 Grundaufbau Backend 32](#_Toc165651194)

[2.5.4 Config Datei 34](#_Toc165651195)

[2.5.5 Tabellen-Auswahl Backend 36](#_Toc165651196)

[2.5.6 Snippet-Auswahl Frontend 39](#_Toc165651197)

[2.5.7 Snippet-Generierung 42](#_Toc165651198)

[2.5.8 Klassendiagramm 60](#_Toc165651199)

[2.6 Kontrollieren 62](#_Toc165651200)

[2.6.1 Testfälle 62](#_Toc165651201)

[2.6.2 Testprotokoll 69](#_Toc165651202)

[2.7 Auswerten 70](#_Toc165651203)

[2.7.1 Projektauswertung 70](#_Toc165651204)

[2.7.2 Schlusswort 71](#_Toc165651205)

[2.8 Verzeichnisse 72](#_Toc165651206)

[2.8.1 Abbildungsverzeichnis 72](#_Toc165651207)

[2.8.2 Tabellenverzeichnis 73](#_Toc165651208)

[2.8.3 Links 74](#_Toc165651209)

[2.8.4 Glossar / Abkürzungen 75](#_Toc165651210)

[2.9 Anhang 76](#_Toc165651211)

# Teil 1

## Dokumenteninformationen

### Historie

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Gültig ab | Dokumentenhistorie / Änderungshinweis | Autor |
| 0.1.0.1 | 17.04.2024 | Erstellung der Grundstruktur | Luan Caduff |
| 0.2.0.2 | 18.04.2024 | Zeitplan, Informieren und Planen | Luan Caduff |
| 0.3.0.3 | 19.04.2024 | Informieren, Entscheiden | Luan Caduff |
| 0.4.0.4 | 23.04.2024 | Realisieren | Luan Caduff |
| 0.5.0.5 | 24.04.2024 | Realisieren | Luan Caduff |
| 0.6.0.6 | 25.04.2024 | Realisieren | Luan Caduff |
| 0.7.0.7 | 26.04.2024 | Realisieren | Luan Caduff |
| 0.8.0.8 | 30.04.2024 | Realisieren | Luan Caduff |
| 0.9.0.9 | 01.05.2024 | Realisieren, Kontrollieren | Luan Caduff |
| 0.10.0.10 | 02.05.2024 | Realisieren, Kontrollieren | Luan Caduff |
| 0.11.0.11 | 03.05.2024 | Realisieren, Auswerten | Luan Caduff |
| 1.0.0.12 | 03.05.2024 | Finalisierung für Abgabe | Luan Caduff |

Tabelle 1 - Dokumentenhistorie

Versionierung

A.B.C.D

A = Eine Veröffentlichung / bereit zum Druck / Abgabe  
B = Inhaltliche Änderungen am Dokument  
C = Korrekturen (keine inhaltlichen Änderungen)  
D = Laufnummer (wird bei jeder Änderung erhöht)

### Eigenschaften

|  |  |
| --- | --- |
| Bezeichnung | Detailinformationen |
| **Status** | Abgeschlossen |
| **Autor** | Luan Caduff |
| **Ausbildung zum** | Eidg. Dipl. Informatiker EFZ |
| **Fachrichtung** | Applikationsentwickler |
| **Version** | 1.0.0.12 |
| **Versionsdatum** | 03.05.2024 |
| **Seiten** |  |

Tabelle 2 - Dokumenteneigenschaften

## Aufgabenstellung

Die Firma S + O AG ist Partnerfirma der Evernex IT Services Switzerland AG und zuständig für den Betrieb und die Weitentwicklung deren Web-Applikationen. Die bestehende Applikation SWO (Simple Web Office) soll abgelöst werden. In einem ersten Schritt ist geplant, auf der bestehenden MariaDB Datenbank-Struktur neue Web-Views (CRUDs) für die Administration zu erstellen. Um diese Arbeit zu erleichtern und zu beschleunigen soll ein Code-Generator erstellt werden, welcher anhand der Struktur einer Datenbank-Tabelle Code-Snippets erstellt.

Der Code-Generator soll mit PHP sowie HTML, CSS und JavaScript umgesetzt werden. Es ist eine neue, stand-alone Applikation, welche entsprechend unabhängig läuft, ohne Einbettung in ein bestehendes Umfeld.

Da diese Applikation lediglich lokal von uns verwendet werden soll, benötigt es kein Authentifizierungssystem.

Der Benutzer dieses Code-Generators soll in einem Web-GUI den Datenbank-Server aus einer Selectbox auswählen können. Für die Auswahl des Datenbank-Servers sollen die zum Verbindungsaufbau nötigen Informationen aus einer Konfigurations-Datei gelesen werden (CSV, eine Zeile pro Datenbank-Server mit Strichpunkt getrennte Informationen wie Host, Login, Passwort).

Nach Auswahl des Datenbank-Servers werden dessen Datenbanken in einer weiteren Selectbox zur Auswahl angezeigt. Nachdem eine Datenbank gewählt wurde, werden dessen Tabellen ebenso zur Auswahl angezeigt. Nachdem eine Tabelle ausgewählt wurde, kann mit dem jeweiligen Button ein Code-Snippet erstellt werden.

Das generierte Code-Snippet soll im Web-GUI angezeigt werden und mittels eines Copy-Buttons in die Zwischenablage kopiert werden können.

Das Ziel dieser Web-Applikation ist es also ein sauber formatiertes (übliche Einrückungen, Zeilen-Abstände zur besseren Lesbarkeit) Code-Snippet in der Zwischenablage zur weiteren Verarbeitung bereit zu stellen.

Nachfolgende Code-Snippets sollen generiert werden können:

|  |  |
| --- | --- |
| **Snippet Art** | **Zweck** |
| PHP-Model-Klasse | Abbild der Tabelle (Attribute), Konstruktor, Getter/Setter,  JSON-Serialize, Objekt-Erstellung aus Daten |
| PHP-Gateway-Klasse | Hinzufügen bzw. Anpassen eines neuen Datensatzes anhand des Model-Objektes |
| ExtJS-Model | Laden bzw. mappen via JSON übertragener Datensätze |
| ExtJS-Grid-List | Liste der geladenen Datensätze darstellen |
| ExtJS-Create-Dialog | Eingabe-Formular für einen neuen Datensatz |
| ExtJS-Edit-Dialog | Eingabe-Formular für die Anpassung eines bestehenden Datensatzes |
| ExtJS-Info-Dialog | Darstellen aller Informationen (nicht editierbare Datenfelder) eines bestehenden Datensatzes |

Tabelle 3 - Code-Snippet Arten

Als Vorlage für die Code-Snippets dient das vorhandene Test-CRUD der Tabelle «erp\_article\_service», an dem ich in den letzten Monaten gearbeitet habe.

Diese Vorlagen werden aber hier nicht mitgeliefert bzw. hochgeladen, da es sich dabei um die gesamte Test-Applikation handelt und daraus allgemeine (bzw. für die gewählte Tabelle), sinnvolle Code-Snippets (nach obiger Auflistung) erstellt werden sollen. Die Auswahl für «sinnvoll» ist Bestandteil der IPA und soll auch entsprechend begründet werden.

Die Architektur soll MVC nach Firmenusanz abbilden: Klassen mit entsprechenden Funktionalitäten in entsprechenden Verzeichnissen. Dazu existieren keine dokumentierten Firmenstandards.

Die technische Dokumentation zum Aufbau der Applikation soll mittels Use Case und eines Klassen-Diagramms erstellt werden.

Das Code-Styling soll wie folgt sein: Die Namensgebung ist einfach gut gewählt. Die Struktur des Codes ist ebenfalls einfach übersichtlich gemacht. Es ist eine gewisse Einheit zu sehen in der Art und Weise, wie der Code strukturiert ist (d.h. es ist überall etwa gleich gemacht).

Die Applikation soll manuell, anhand von sinnvollen Testfällen getestet werden. Die Definition der Testfälle ist Bestandteil der IPA und die einzelnen Tests müssen dann auch entsprechend protokolliert werden.

## Projektorganisation

VF – Roman Born (Verantwortliche Fachkraft)

VF

FS

PL

TP

PL – Luan Caduff (Projektleiter)

FS – Luan Caduff (Fachspezialist)

TP – Luan Caduff (Testperson)

## Projektmethode

Dieses Projekt wird mit IPERKA durchgeführt. Dies ist eine simple Projekt-management-Methode zur strukturierten Planung und Umsetzung eines Projektes.

IPERKA ist ein Akronym und bedeutet folgendes:

**I** –Informieren – Relevante Informationen sammeln

**P** – Planen – Damit einen Plan erstellen

**E** – Entscheiden – Sich für eine spezifische Vorgehensweise entscheiden

**R** – Realisieren – Das Projekt durchführen

**K** –Kontrollieren – Endprodukt testen und überprüfen

**A** – Auswerten – Über den Arbeitsprozess und das Resultat reflektieren

Weitere Informationen zur IPERKA-Methode finden Sie unter <https://www.ict-berufsbildung-bern.ch/resources/Iperka_OdA_200617.pdf>

## Deklaration der Vorkenntnisse

Alle geplanten Tätigkeiten/Produkte/Techniken sind bekannt und wurden während der gesamten Praktikumszeit eingesetzt.

|  |  |
| --- | --- |
| **Technologie** | **Erfahrung** |
| PHP | Sehr gute Kenntnisse  Seit 2.5 Jahren aktiv genutzt. |
| HTML | Sehr gute Kenntnisse  Seit 3.5 Jahren aktiv genutzt. |
| JavaScript + jQuery | Ziemlich gute Kenntnisse  Seit 2.5 Jahren öfters genutzt. |
| CSS + Bootstrap | Ziemlich gute Kenntnisse  Seit 3.5 Jahren öfters genutzt. |
| MariaDB | Sehr gute Kenntnisse  Seit 3.5 Jahren aktiv genutzt. |

## Deklaration der Vorarbeiten

In den letzten Monaten habe ich ein Test-CRUD der Tabelle «erp\_article\_service» aufgebaut. Dieses CRUD dient als Grundlage für die Code-Snippets, die der Generator erstellen soll.

In direktem Zusammenhang mit dieser Arbeit habe ich ein GIT-Repository auf GitHub eingerichtet.

Des Weiteren habe ich die Dokumentenvorlage inklusive Verzeichnisstruktur am Vortag zum IPA-Start erstellt.

## Deklaration der benutzen Firmenstandards

Es existieren keine dokumentierten Firmenstandards.

Der Code wird nach der «PHP Standard Recommendation 1» aufgebaut, worin Grundlegende Coding Konventionen festgehalten werden.

Die Konventionen sind unter folgendem Link dokumentiert:

<https://www.php-fig.org/psr/psr-1/>

## Organisation der Arbeitsergebnisse

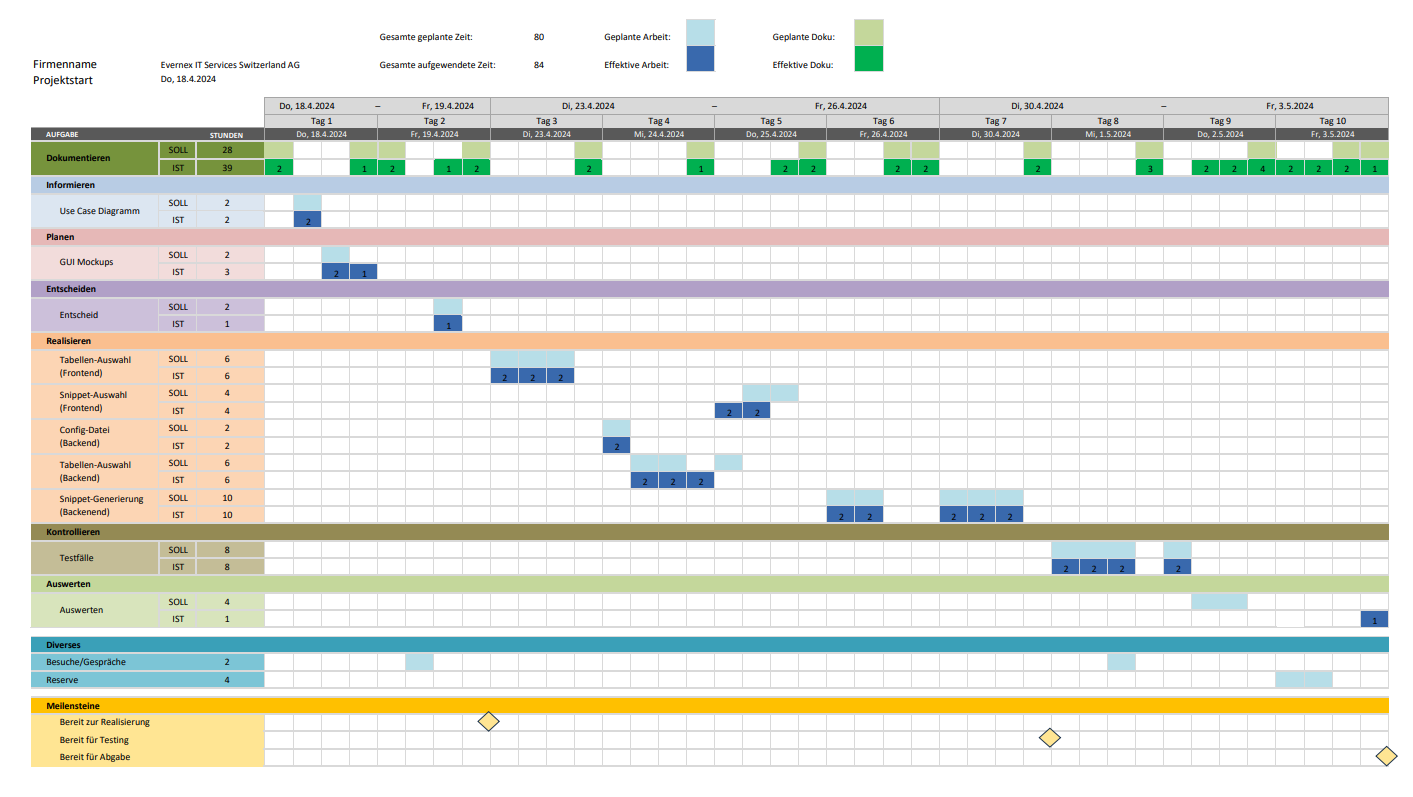
Der erarbeitete Code wird in das lokale GIT-Repository commited.

Das IPA-Dokument wird ebenfalls – jeweils vor Arbeitsende – im lokalen   
GIT-Repository in das dafür vorgesehene Verzeichnis (000\_Dokumentation) sowohl als Word-Dokument als auch als PDF kopiert und commited.

Das Repository wird täglich (ebenfalls vor Arbeitsende) auf GitHub gepusht:

<https://github.com/xivia/ipa-luan---code-generator>

## Zeitplan



## Arbeitsjournal

### Tag 1, 18. April 2024

|  |  |
| --- | --- |
| **Arbeiten** | Dokumentation   * Zeitplan erstellt * Im IPA Dokument Teil 1 abgefüllt   Informieren   * Use Case Diagramm erstellt   Planen   * MockUps erstellt |
| **Probleme** | Ich habe die MockUps erstellt, jedoch wurden diese nicht richtig gespeichert und sind verloren gegangen. Aus diesem Grund habe ich sie dann erneut erstellen müssen. |
| **Hilfestellungen** | Zeitplan kurz mit VF besprochen. |
| **Überzeiten** | Aufgrund meines Problems habe ich für die MockUps eine Stunde mehr als geplant aufgewendet. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Ich habe die MockUps wie bereits erwähnt mit demselben Web-Tool erstellt, wie zuvor die Use Cases. Mit diesem Tool konnte ich die MockUps effizient zusammensetzen. Leider habe ich aus Versehen meinen Browser geschlossen, bevor das Tool die Änderungen speichern konnte. Zukünftig werde ich besser darauf Acht geben, Dateien öfters abzuspeichern.  Da ich nun etwas in Verzug bin, da die Use Case Beschreibungen noch nicht dokumentiert wurden, muss ich morgen im geplanten Dokumentieren-Block als erstes diese noch verfassen, um so wieder auf Kurs zu kommen. |

### Tag 2, 19. April 2024

|  |  |
| --- | --- |
| **Arbeiten** | Erstes Gespräch mit HEX  Dokumentation   * Generell Dokumentation verschönert * Use Case Beschreibungen verfasst   Entscheiden   * GUI-Varianten Entscheid gefällt |
| **Probleme** | Keine. |
| **Hilfestellungen** | Vorbereitung mit VF auf HEX-Gespräch. |
| **Überzeiten** | Keine. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Die Use-Case Beschreibungen waren schnell gemacht, da das Diagramm bereits existiert und ich die Use-Cases bereits gut kenne.  Die Entscheidung zwischen den beiden GUI-MockUps war schwerer als gedacht, da die Vergleichskriterien schwierig zu finden sind bzw. nicht immer objektiv sind. Vorerst wurde der Vergleich textuell gemacht, aber dies sollte zukünftig noch in eine Entscheidungsmatrix/Tabelle integriert werden. |

### Tag 3, 23. April 2024

|  |  |
| --- | --- |
| **Arbeiten** | Dokumentation   * Realisierung dokumentiert   Realisieren   * Frontend Tabellen-Auswahl |
| **Probleme** | Keine. |
| **Hilfestellungen** | Keine. |
| **Überzeiten** | Keine. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Ich konnte die geplanten 6 Stunden für die Tabellen-Auswahl effizient ausnützen.  Für die verschiedenen Selectboxen musste ich Dummy-Daten hartkodieren, da noch kein Backend existiert.  Zudem habe ich die Formularmaske zur Einrichtung neuer Datenbankverbindungen (Configs) erstellt.  Ich finde die Fade-In und Fade-Out Effekte visuell ansprechend und habe sie implementiert, da ich noch genügend Zeit dafür hatte.  Das Resultat entspricht dem MockUp soweit ziemlich genau. |

### Tag 4, 24. April 2024

|  |  |
| --- | --- |
| **Arbeiten** | Dokumentation   * Realisierung dokumentiert   Realisieren   * Backend DB-Verbindungen und Tabellen-Auswahl |
| **Probleme** | Keine. |
| **Hilfestellungen** | Keine. |
| **Überzeiten** | Eine Stunde Überzeit zur Fertigstellung der Tabellen-Auswahl. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Zuerst habe ich die Config-Datei inklusive Auslesung und Bearbeitung erstellt.  Danach habe ich die Grundstruktur des Backends aufgebaut, um die Kommunikation der verschiedenen Module zu organisieren.  Ich konnte einen flüssigen Übergang zur Tabellen-Auswahl machen, und habe diese heute vorfristig abgeschlossen, in dem ich sie anstelle der geplanten Dokumentation gemacht habe, und dafür eine Stunde Überzeit zum Dokumentieren investiert habe. Somit bin ich nun vorne im Zeitplan, und kann morgen bereits mit der Snippet-Auswahl beginnen.  In der Dokumentation bin ich ein wenig hinten, und sollte langsam aufholen. |

### Tag 5, 25. April 2024

|  |  |
| --- | --- |
| **Arbeiten** | Dokumentation   * Realisierung dokumentiert   Realisieren   * Frontend Snippet-Auswahl |
| **Probleme** | Kopieren in die Zwischenablage komplizierter als gedacht. |
| **Hilfestellungen** | Wegen Probleme mit kopieren mit VF gesprochen und danach eine Lösung aus dem Internet geholt. |
| **Überzeiten** | Keine. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle praktischen Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Heute habe ich das Frontend für die Snippet-Generation umgesetzt. Zur Erläuterung der Eingaben Reihenfolge (zuerst DB-Tabelle wählen) habe ich zusätzlich eine visuelle Hilfe (rechter Teil der Applikation grau überdeckt) eingebaut  Das Kopieren des Code-Snippets in die Zwischenablage musste ich mit einer Lösung aus dem Internet umsetzen, da mir keine Alternative bewusste war, als meine Vorgehensweise nicht funktionierte.  Das Dokumentieren konnte ich heute dank grösserer Zeitinvestition ein wenig aufholen, jedoch möchte ich morgen auch wieder mehr Zeit investieren, da ich merke, dass ich immer weiter nach hinten falle, wenn ich jetzt nicht aufhole. |

### Tag 6, 26. April 2024

|  |  |
| --- | --- |
| **Arbeiten** | Dokumentation   * Realisierung dokumentiert   Realisieren   * Backend Snippet-Generierung PHP |
| **Probleme** | Keine. |
| **Hilfestellungen** | Keine. |
| **Überzeiten** | Keine. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle praktischen Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Heute habe ich im Backend die PHP Snippets fertiggestellt.  Dies wurde noch ein bisschen stressig da zuerst noch alle grundlegende String-Manipulationsfunktionen in einer Basis-Klasse definiert werden mussten, um die Snippets alle mit den gleichen Funktionen zu erstellen.  Ich habe wieder mehr Zeit in die Dokumentation gesteckt. |

### Tag 7, 30. April 2024

|  |  |
| --- | --- |
| **Arbeiten** | Dokumentation   * Realisierung dokumentiert   Realisieren   * Backend Snippet-Generierung ExtJs |
| **Probleme** | Keine. |
| **Hilfestellungen** | Keine. |
| **Überzeiten** | Keine. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle praktischen Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Heute habe ich die ExtJs Snippets, und somit den «Realisieren» Teil der Arbeit, abgeschlossen. Das Dokumentieren ist zeitaufwändiger als gedacht, und ich bin mit meinem Fortschritt nicht zufrieden. |

### Tag 8, 01. Mai 2024

|  |  |
| --- | --- |
| **Arbeiten** | Dokumentation   * Kontrollieren dokumentiert * Realisieren dokumentiert   Kontrollieren   * Testfälle definiert. |
| **Probleme** | Keine. |
| **Hilfestellungen** | Keine. |
| **Überzeiten** | Keine. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle praktischen Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Heute habe ich alle Testfälle definiert, und weiterhin die fehlenden Teile der Dokumentation in Bezug auf die Realisierung fertiggestellt.  Für die Testfälle musste ich zuerst noch eine passende Tabelle erstellen.  Die verschiedenen Tests für die Applikation zu erfinden war gar nicht mal so einfach, den ich wusste zuerst nicht, was ich überhaupt wirklich alles Testen kann, abgesehen von den Muss-Kriterien. Dann habe ich aber gemerkt, dass ich auch meine Fehlermeldungen testen kann, und habe diese auch noch implementiert. |

### Tag 9, 02. Mai 2024

|  |  |
| --- | --- |
| **Arbeiten** | Dokumentation   * Kontrollieren dokumentiert (Testprotokoll) * Realisieren dokumentiert   Kontrollieren   * Testprotokoll erstellt. |
| **Probleme** | Keine. |
| **Hilfestellungen** | Keine. |
| **Überzeiten** | Heute habe ich zum Dokumentieren zwei Stunden Überzeit gemacht. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle praktischen Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Heute habe ich als erstes alle definierten Testfälle durchgeführt und protokolliert. Als nächstes im Zeitplan käme jetzt die Auswertung, jedoch habe ich mich dazu entschlossen, diese auf Morgen zu verschieben, da noch ein Grossteil der Realisierung undokumentiert ist und ich die Reflexion erst schreiben möchte, wenn ich auch wirklich mit der Arbeit fertig bin.  Somit habe ich den gesamten restlichen Tag, mit zwei zusätzlichen Überstunden, dokumentiert. |

### Tag 10, 03. Mai 2024

|  |  |
| --- | --- |
| **Arbeiten** | Dokumentation   * Realisieren fertig dokumentiert * Entscheidungsmatrix MockUps   Auswertung   * Projektauswertung und Schlusswort verfasst. |
| **Probleme** | Keine. |
| **Hilfestellungen** | Rücksprache mit VF über Bewertungskriterien für die Entscheidungsmatrix. |
| **Überzeiten** | Keine. |
| **Ungeplante Arbeiten** | Keine. |
| **Erfolge** | Alle Arbeiten wurden zufriedenstellend abgeschlossen. |
| **Misserfolge** | Keine. |
| **Reflektion** | Heute konnte ich die Realisierung fertig dokumentieren und danach die Auswertung in nur einer Stunde durchführen, da keine Zeit mehr übrig war.  Ausserdem habe ich das Dokument nochmals überprüft und sichergestellt, dass alle relevanten Teile richtig sind. |

### Zusammenfassung Zeitplan

Hier zählen wir alle Stunden pro Tag zusammen um einen guten, übersichtlichen Soll-/Ist- Vergleich zu erhalten.

|  |  |  |
| --- | --- | --- |
| Datum | Soll [h] | Ist [h] |
| Tag 1, 18. April 2024 | 8 | 8 |
| Tag 2, 19. April 2024 | 8 | 8 |
| Tag 3, 23. April 2024 | 8 | 8 |
| Tag 4, 24. April 2024 | 8 | 9 |
| Tag 5, 25. April 2024 | 8 | 8 |
| Tag 6, 26. April 2024 | 8 | 8 |
| Tag 7, 30. April 2024 | 8 | 8 |
| Tag 8, 1. Mai 2024 | 8 | 9 |
| Tag 9, 2. Mai 2024 | 8 | 10 |
| Tag 10, 3. Mai 2024 | 8 | 8 |
| Total | 80 | 84 |

Tabelle 4 - Zeitplan Zusammenfassung

# Teil 2

## Management Summary (Kurzfassung des IPA-Berichts)

### Ausgangslage

Die bestehende Applikation SWO (Simple Web Office) soll abgelöst werden. In einem ersten Schritt ist geplant, auf der bestehenden MariaDB Datenbank-Struktur neue Web-Views (CRUDs) für die Administration zu erstellen. Um diese Arbeit zu erleichtern und zu beschleunigen, soll ich einen Code-Generator erstellen, welcher anhand der Struktur einer Datenbank-Tabelle Code-Snippets erstellt.

### Vorgehen

Ich habe mit PHP und HTML, CSS und JavaScript eine Web-Applikation aufgebaut, in welcher verschiedene Datenbank-Zugangsdaten in einer CSV-Datei gespeichert werden können.

Ich habe eine PHP-Datei erstellt, welche alle Anfragen des Frontends entgegennimmt und dann an die entsprechenden Controller der Backend MVC-Struktur weitergibt. In der MVC-Struktur existiert auch noch ein Utils Ordner, welcher Aufgaben wie einheitliche Antworten oder Datenbankverbindungen löst.

Zur Generierung der Code-Snippets wird über die Liste der Felder iteriert.

Somit ist es eine Art von Object Relation Mapping.

### Ergebnis

Die Zugangsdaten können mittels Selectbox ausgewählt werden, was mögliche Datenbanken und Tabellen zur Auswahl stellt.

Wenn eine Tabelle selektiert ist, wird die Generation von verschiedenen Code-Snippets freigeschaltet. Diese sind als Buttons über einer Ausgabebox dargestellt.

Wird ein Button geklickt, wird das entsprechende Code-Snippet in der Ausgabebox dargestellt und zur Kopie in die Zwischenablage per Button bereitgestellt.

## Informieren

### Aufgabestellung

Siehe Kapitel 1.2 Aufgabenstellung

Nach Analyse dieser Aufgabenstellung habe ich bemerkt, dass nicht definiert ist, wie diese Konfigurations-Datei abgefüllt werden soll. Nach Rücksprache kann sie manuell bearbeitet werden. Ich habe mich dann aber dazu entschlossen ein kleines Formular zum Hinzufügen eines Eintrages umzusetzen, da dies keinen grossen Zusatzaufwand mit sich trägt, jedoch einen Mehrwert bietet.

Weil die Applikation auf Englisch umgesetzt wird, werden auch alle technischen Diagramme und Dateinamen Englisch benannt.

**Begrifflichkeit:**

Die Konfigurations-Datei wird fortan Config-Datei, und die darin enthaltenen Verbindungsinformationen Configs genannt.

### Technologien

Als Vorgabe sind folgende Technologien zu benützen:

* PHP – Personal Home Page Hypertext Preprocessor (rekursives Akronym)
  + Verwendet wird die Version 8.2
* HTML – Hypertext Markup Language
  + Verwendet wird HTML5
* JavaScript mit der jQuery Library
  + Hier kommen ECMAScript 2023 und jQuery Version 3.7.1 zum Einsatz
* CSS mit dem Bootstrap Framework – Cascading Style Sheets
  + Hier verwenden wir auch die aktuellsten Versionen der jeweiligen Technologien; CSS 3 & Bootstrap 5.3.2

### Use-Cases

Mithilfe des Web-Tools «Lucidchart» habe ich das Use-Case-Diagramm erstellt.

Mehr zum Tool finden Sie unter <https://www.lucidchart.com/pages/>

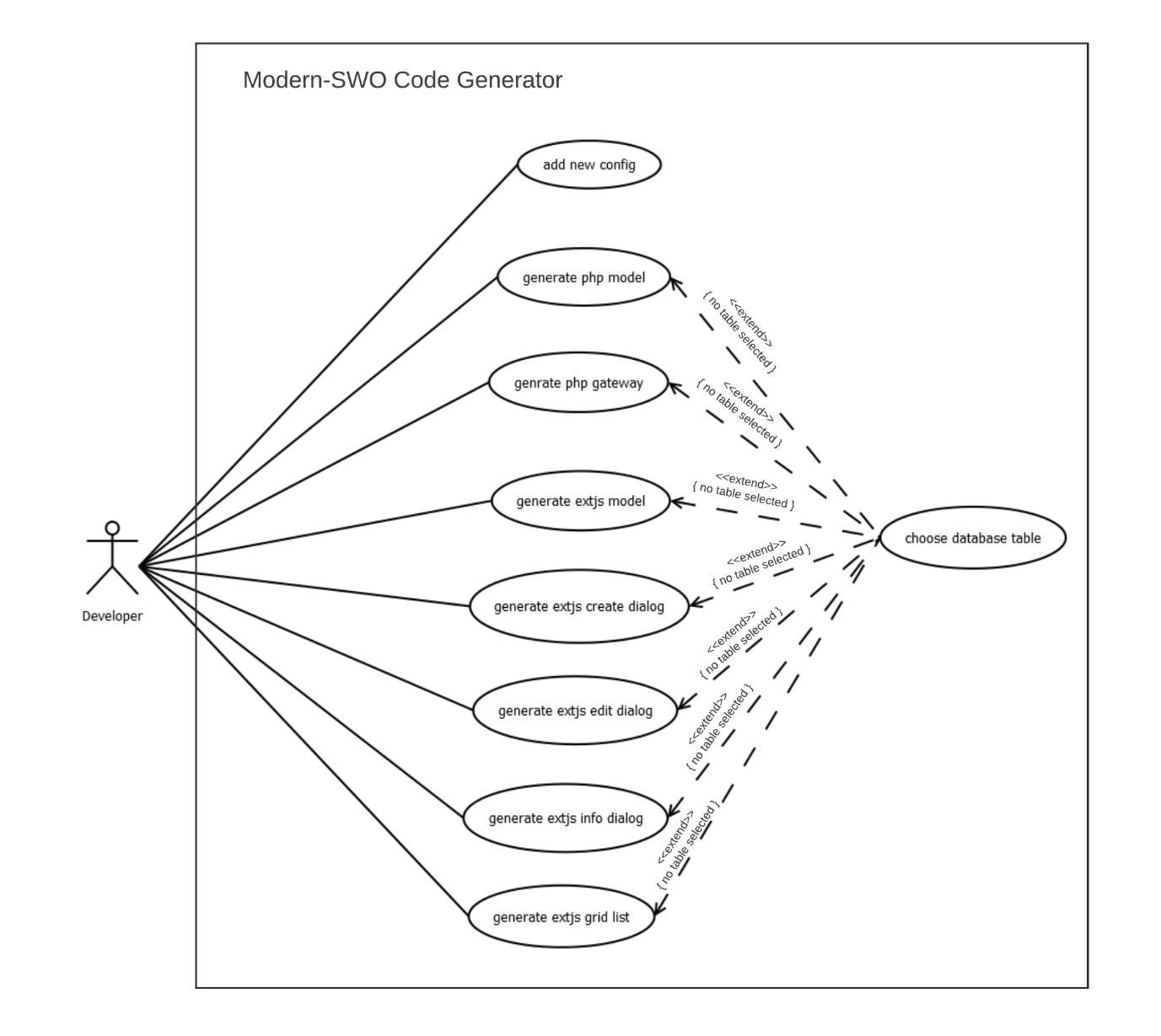


Abbildung 1 - Use-Case-Diagramm

**Beschreibungen:**

|  |  |
| --- | --- |
| Use Case | add new config |
| Akteur | Developer |
| Vorbedingung | Keine |
| Beschreibung | Durch Klick auf einen Button öffnet sich ein Formular, in welchem eine neue Datenbankverbindung (Eintrag in der Config-Datei) hinzugefügt werden kann. Es müssen Host, User, Passwort und der Port angegeben werden. Danach werden diese in der Config-Datei als neue Zeile gespeichert. |
| Alternative | Kann die Config nicht hinzugefügt werden, wird eine Fehlermeldung ausgegeben. |
| Nachbedingung | Eine neue Config wurde in der Config-Datei angelegt. |

Tabelle 5 - Use-Case Beschreibung «add new config»

|  |  |
| --- | --- |
| Use Case | choose database table |
| Akteur | Developer |
| Vorbedingung | Keine |
| Beschreibung | Erstens kann eine Datenbankverbindung gewählt werden, woraufhin versucht wird, diese Verbindung herzustellen. Bei Erfolg kann als nächstes eine darin enthaltene Datenbank ausgewählt werden. Danach kann eine Tabelle der Datenbank ausgewählt werden. |
| Alternative | Kann die Datenbankverbindung nicht hergestellt werden, erscheint eine Fehlermeldung. |
| Nachbedingung | Es können Code-Snippets generiert werden. |

Tabelle 6 - Use-Case Beschreibung «choose database table»

|  |  |
| --- | --- |
| Use Case | generate Code-Snippet (php model, php gateway, extjs model, extjs grid, extjs add, extjs edit, extjs info) |
| Akteur | Developer |
| Vorbedingung | Es muss eine Datenbank-Tabelle ausgewählt sein. |
| Beschreibung | Durch Klick auf den entsprechenden Button wird das Code-Snippet generiert. |
| Alternative | Wenn in der Tabelle ein unbekannter Datentyp vorkommt, wird eine Fehlermeldung mit dem Namen des Datentyps ausgegeben. Sonst erscheint eine generische Fehlermeldung. |
| Nachbedingung | Das Code-Snippet ist im GUI ersichtlich und kopierbar. |

Tabelle 7 - Use-Case Beschreibung «generate Code-Snippet»

## Planen

### GUI MockUps

Ich möchte zwei Versionen eines möglichen Frontend-Aufbaus dieser Applikation entwerfen. Dazu verwende ich dasselbe Web-Tool wie bereits für die Use-Cases.

**Entwurf 1**



Abbildung 2 - MockUp 1

**Entwurf 2**



Abbildung 3 - MockUp 2

**Beschreibung der Elemente:**

Entwurf 1 hat ein horizontales Layout, wo die Selectboxen links ziemlich eng zusammen sind, und unterhalb dieser der «new» Button mit einem Trennstrich isoliert ist. Dies liegt daran, dass der Platz unter dem «new» Button dafür vorgesehen ist, ein Formular anzuzeigen, wenn der Button gedrückt wird.

Bei Entwurf 2 wäre solch ein Formular vom Platz her nicht umsetzbar, und man müsste ein modales Pop-Up verwenden.

Die Selectboxen gewährleisten jeweils die Auswahl der Config, der Datenbanken und der Tabellen.

Die Buttons, die mit «gen» gekennzeichnet sind, werden die Code-Snippets generieren und in dem Output-Fenster darstellen.

Um das Kopieren in die Zwischenablage zu erleichtern habe ich noch einen Copy-Button in das Output-Fenster integriert.

## Entscheiden

Nun wird sich für eine der beiden GUI-Varianten entschieden.

Variante 1 oder Variante 2



Abbildung 4 - MockUp Variante 2

Abbildung 5 - MockUp Variante 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Gewichtung | Variante 1 | | Variante 2 | |
| **Punktevergabe**  0: nicht erfüllt  1: erfüllt OK  2: erfüllt gut  3: erfüllt perfekt | 1: seltener Fall  2: häufiger Fall | Punkte | Gewichtete Punkte | Punkte | Gewichtete Punkte |
| Platz für Code-Snippet horizontal | 2 | 1 | 2 | 2 | 4 |
| Platz für Code-Snippet vertikal | 2 | 2 | 4 | 1 | 2 |
| Arbeitsprozess für die Erstellung einer neuen Config | 1 | 2 | 2 | 3 | 3 |
| Arbeitsprozess für die Erstellung von Code-Snippets | 2 | 2 | 4 | 1 | 2 |
| **Summe** |  |  | **12** |  | **11** |

Abbildung 6 – Mockup Entscheidungsmatrix

Dies war ein knapper Entscheid, beide Varianten sind eine valide Wahl, jedoch entscheide ich mich nun für Variante 1.

## Realisieren

Bei der Realisierung ist geplant, die Applikation schrittweise aufzubauen.  
Zuerst ein Abschnitt des Frontends und danach der entsprechende Backend Teil dazu.

### Tabellen-Auswahl Frontend

Um eine Datenbank Tabelle auswählen zu können, muss zuerst eine Datenbank ausgewählt sein. Um jedoch solch eine zu wählen, muss man eine Datenbankverbindung, also eine Config, auswählen.

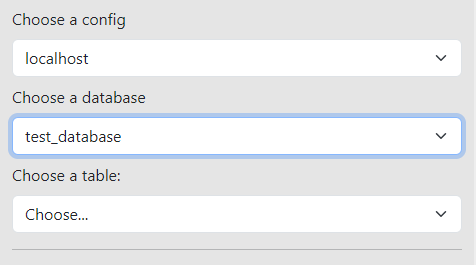


Abbildung 7 - Selectboxen

Dies habe ich mithilfe des JavaScript onChange Events umgesetzt.

Sobald die oberste Selectbox (Config) geändert wird, beginnt die automatische Abfüllung der nächsten Selectbox (Datenbank).

Bei der letzten Selectbox (Tabelle) ist keine onChange Event Funktion definiert.

Damit dies für den Benutzer klar ersichtlich und intuitiv ist, erscheint zuerst lediglich die oberste Selectbox. Erst so bald dort eine Option gewählt wurde, erscheint die Datenbank Selectbox. Das gleiche geschieht mit der Tabellen-Selectbox, sie erscheint erst, wenn eine Datenbank ausgewählt wurde.

Falls im Backend beim Datenbankverbindungsversuch ein Fehler auftritt, wird eine Meldung unterhalb der Config Selectbox erscheinen.

### Config-Formular

Zur erleichterten Anlegung einer neuen Config habe ich definiert, ein Formular dafür zu integrieren.

Dieses erscheint, nachdem man auf den «New config» Button drückt, und verschwindet, wenn der Fokus das Formular verlässt und kein Feld abgefüllt wurde oder eine Config hinzugefügt wurde.

Das Formular:

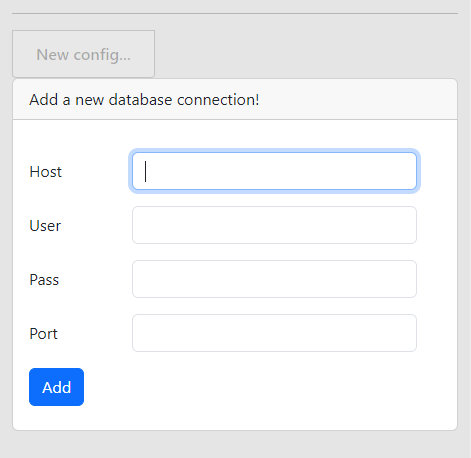


Abbildung 8 - Config-Formular

### Grundaufbau Backend

Grundsätzlich habe ich die Applikation nach MVC gestaltet. Alle Klassen, die nicht in dieses Schema passen, sind in einem vierten Verzeichnis namens «Utils» abgelegt.

Die verschiedenen Module kommunizieren folgendermassen:

Abbildung 9 – Grundstruktur der Applikation

View

Controller

Utils

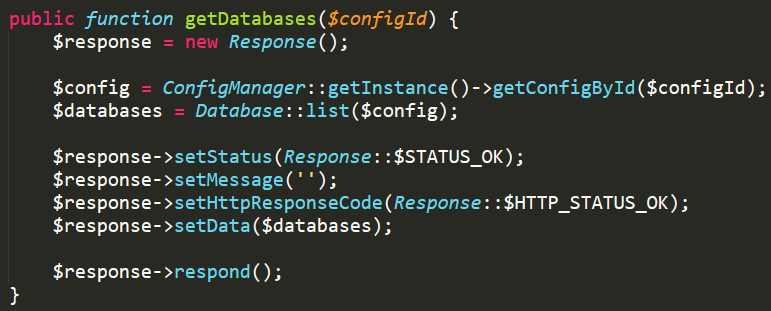
Model

Im Controller habe ich eine PHP-Datei namens router.php erstellt. Diese ist die einzige Backend-Datei, welche keine Klasse ist. Sie ist die Kommunikations­schnittstelle zwischen dem Front- und Backend, und wird auch verwendet, um einen Autoloader zu definieren und Error-Reporting einzuschalten.

Jede AJAX-Anfrage geht direkt an diese Datei, worin das entsprechende Controller Objekt anhand des übergebenen «action» Parameters erstellt wird und die gesendeten Daten (falls vorhanden) an eine Methode des erstellten Objekts übergeben werden.



Abbildung 10 - Backend router Ausschnitt

Jede Controller-Methode, welche die Sichtbarkeit public hat, sorgt für die Ausgabe (Antwort ans Frontend) der gewünschten Daten.

Mithilfe der Model- und/oder Utils-Klassen wird im Controller eine Antwort ausgearbeitet.

Abbildung 11 - Controller-Klasse Ausgabe

Zur Ausgabe habe ich die Utils-Klasse Response erstellt. Sie definiert eine Einheitliche Antwort im JSON-Format.



Abbildung 12 - Response-Klasse Ausschnitt

Die einzige Klasse, die kein Controller ist und eine Ausgabe ausführt, ist die Utils-Klasse ErrorThrower, welche lediglich eine statische Methode zur Fehlerausgabe und zum Ausführungsabbruch definiert.

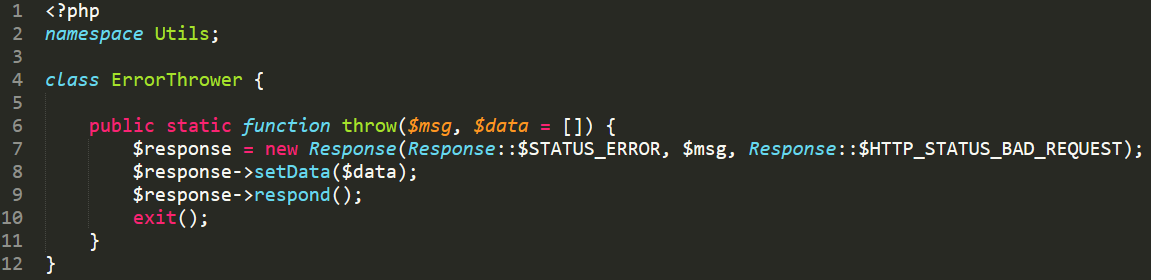


Abbildung 13 - Utils-Klasse ErrorThrower

### Config Datei

Um eine Datenbankverbindung herzustellen, werden Zugangsdaten benötigt.  
Diese werden in der Config-Datei «config.csv» im Ordner «config» im Root Verzeichnis der Applikation abgespeichert.

Die Config-Datei muss bei jeder Installation manuell erstellt werden, da sie aufgrund ihres heiklen Inhalts (Passwörter) in der gitignore-Datei ist, und das Formular „New config“ lediglich Config-Zeilen hinzufügen kann, jedoch die benötigte Datei nicht erstellt.

Zwei Zeilen in der Config-Datei würden folgendermassen aussehen:

1;mySQLServer.com;root;myPssword2024\*;3306

2;differentServer.ch;admin;admin; 1433

Der erste Wert ist die ID, welche unbedingt für jede Zeile inkrementell wachsen, und bei 1 anfangen muss. Dies liegt an der Art, wie eine neue Config hinzugefügt wird.

**Laden einer Config**

Die Utils-Klasse ConfigManager regelt alles, was direkt mit der Config-Datei zu tun hat. Die Klasse ist ein Singleton, und lädt die Config-Datei im Konstruktor.  
Dies bedeutet, dass die Datei nur ein einziges Mal und nur bei gebrauch geladen wird.

Mit laden ist gemeint, dass aus jeder Zeile der Config-Datei Config-Objekte erstellt werden und diese im ConfigManager in einem Array abgespeichert werden.

Wenn das Frontend gestartet wird, wird automatisch eine AJAX-Request gesendet, um die Config-Selectbox abzufüllen.

Diese wird an den ConfigController weitergeleitet, welcher die Configs mithilfe des ConfigManagers erhält und an das Frontend zurückgibt:

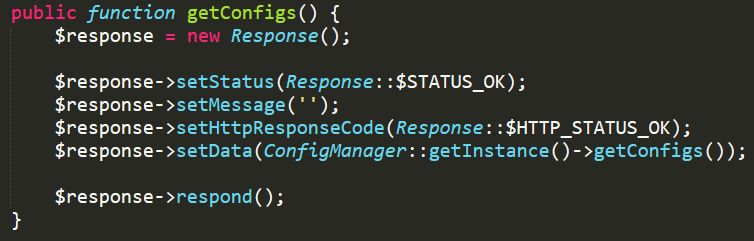


Abbildung 14 - ConfigController getConfigs Funktion

**Hinzufügen einer Config**

Wird das Formular zum Hinzufügen einer Config abgeschickt, kommt als erstes wieder der ConfigController zum Einsatz.

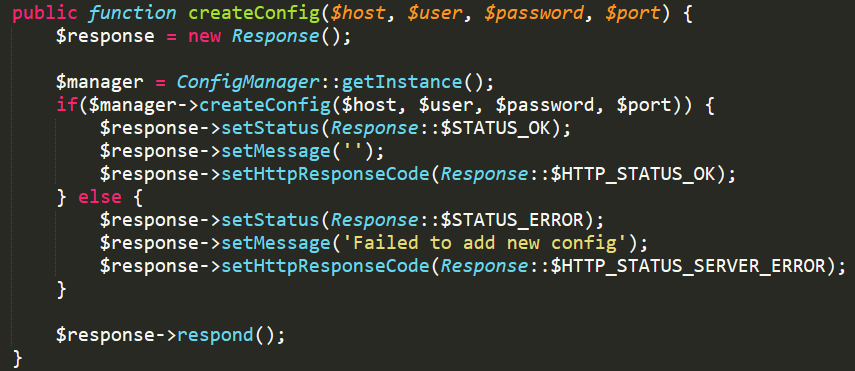


Abbildung 15 - ConfigController createConfig Methode

Wie man sehen kann, leitet der Controller die Aufgabe an den ConfigManager weiter, und gibt dem Frontend lediglich die entsprechende Antwort.

In diesem Bild sieht man, wie der ConfigManager die Dateibearbeitung durchführt:

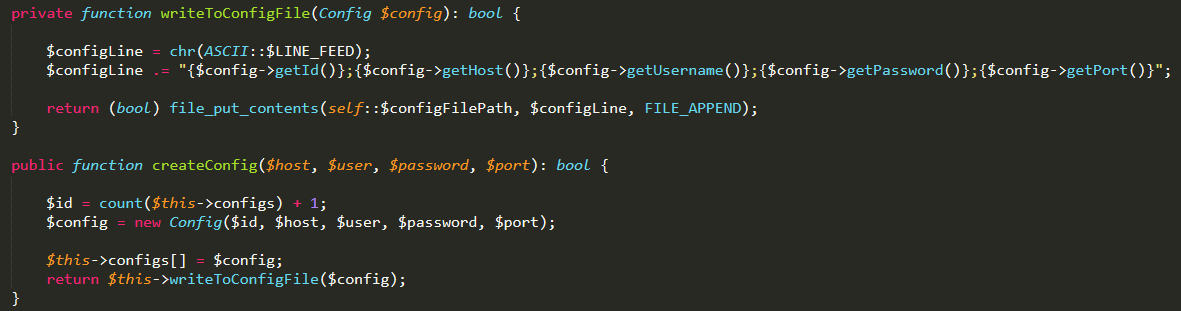


Abbildung 16 - ConfigManager Dateibearbeitung

In der writeToConfigFile Funktion kann man sehen, wie die Utils-Klasse ASCII verwendet wird, um mit der PHP-Funktion chr ein ASCII-Zeichen zu erzeugen.  
Diese Klasse hat lediglich statische Attribute für ein Mapping von Namen zu ASCII-Codes, und wird nur im ConfigManager verwendet

Unter anderem erkennt man, warum die ID in der Config-Datei Inkrementell wachsen, und bei 1 anfangen muss. Nämlich wird in der createConfig Funktion die

nächste ID anhand der Anzahl von Configs, die bereits abgespeichert sind, ausgerechnet.

Das heisst, wenn ich bereits eine einzelne Config manuell mit der ID 2 abgespeichert habe, und dann eine zweite über das Formular hinzufüge, erhält die neu erstellte Zeile als ID auch die 2.

### Tabellen-Auswahl Backend

Für die weiteren zwei Selectboxen habe ich auch jeweils eine Model Klasse erstellt, Database und Table.

Wenn im Frontend eine Config oder Datenbank ausgewählt wird, muss eine Datenbankverbindung hergestellt werden, um die nächste Selectbox laden zu können.

Dazu habe ich die Utils Klasse MysqliDB erstellt.

Sie ist ein Singleton, der dazu dient, mysqli-Objekte zu erstellen und zu verwalteten, welche als Datenbankverbindung verwendet werden.

In PHP ist die mysqli-Klasse ein Datenbanktreiber der als Schnittstelle zu MySQL-Datenbanken verwendet wird.

Sie ist so konzipiert, dass mehrere Verbindungen gespeichert und wieder abgerufen werden können.

Dieses Verhalten ermöglicht es, während der Ausführung, eine unbegrenzte Anzahl an Verbindungen aufzubauen. Dies ist zwar keine Anforderung, jedoch erscheint es mir in diesem Projekt als sinnvoll, da es in der Zukunft dazu kommen könnte, dass wir über mehrere Datenbanken ein einziges Code-Snippet generieren möchten.

Am einfachsten ist die Funktionsweise erklärt, wenn der Ablauf nachfolgend demonstriert wird.

Wenn im Frontend die zweite Selectbox (Datenbanken) bereits abgefüllt wurde, kann mit der Auswahl einer Datenbank die Abfüllung der nächsten Selectbox (Tabellen) ausgelöst werden.

Das Frontend schickt die AJAX-Request mit der Config-ID und der ID der ausgewählten Datenbank.

Nachdem der router die Anfrage entsprechend weiterleitet wird im ConfigController als erstes die Config mithilfe des ConfigManagers geholt.

Danach kann das Datenbank Model mithilfe der erhaltenen Config, und der Datenbank-ID, geholt werden.

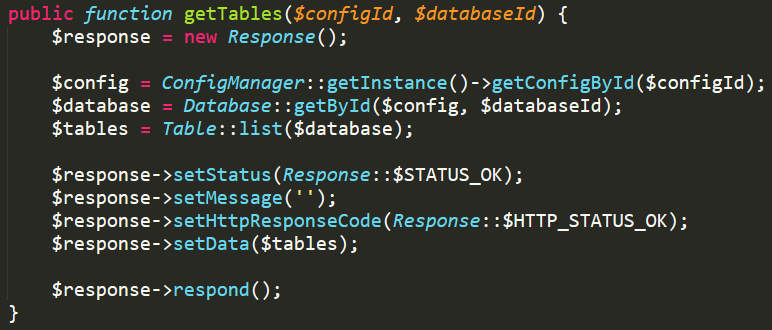


Abbildung 17 - ConfigController getTables-Methode

Im Database Model wird zuerst der MysqliDB Singleton geholt (getInstance), und dann das mysqli-Objekt (getConnection):

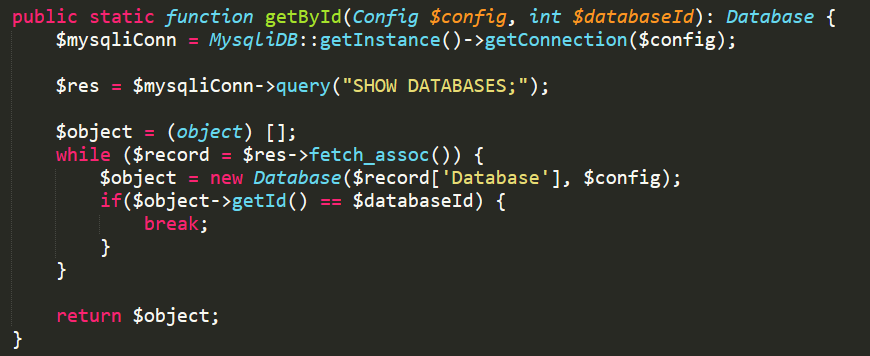


Abbildung 18 - Database getById-Methode

Die Methode getConnection braucht als Parameter das Config-Objekt, mit welchem es das dazugehörige mysqli-Objekt aus einer internen Liste von Verbindungen zurückgibt, oder falls dieses noch nicht vorhanden ist, ein neues erstellt, in der Liste abspeichert und dann zurückgibt.



Abbildung 19 - MysqliDB getConnection-Methode

Im Database Model wird nun die gewünschte Datenbank als Database-Objekt zurückgegeben. Dieses Objekt enthält als Attribute den Namen der Datenbank und die Config, von der Sie stammt.

Im ConfigController werden nun alle Tabellen mithilfe der List Methode vom Table Model geholt.

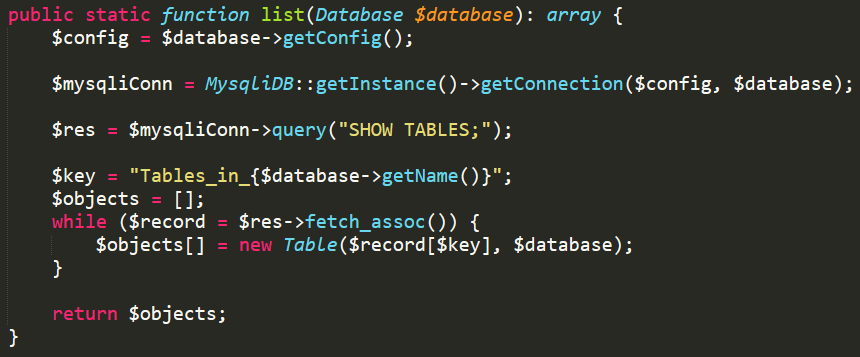


Abbildung 20 - Table list-Methode

Wieder wird der MysqliDB Singleton verwendet, diesmal wird die Datenbank aber auch mitgegeben, damit diese gleich benutzt wird, und dann das SHOW TABLES Query auch funktioniert.

Im ConfigController werden die Tabellen dann ausgegeben.

### Snippet-Auswahl Frontend

Die Snippet-Auswahl ist eine Reihe von Buttons, die erst benutzt werden können, wenn eine Tabelle ausgewählt wurde.

Um dies Visuell zu übermitteln habe ich dafür gesorgt, dass die rechte Seite des Layouts nicht verfügbar ist, wenn keine Tabelle ausgewählt wurde:

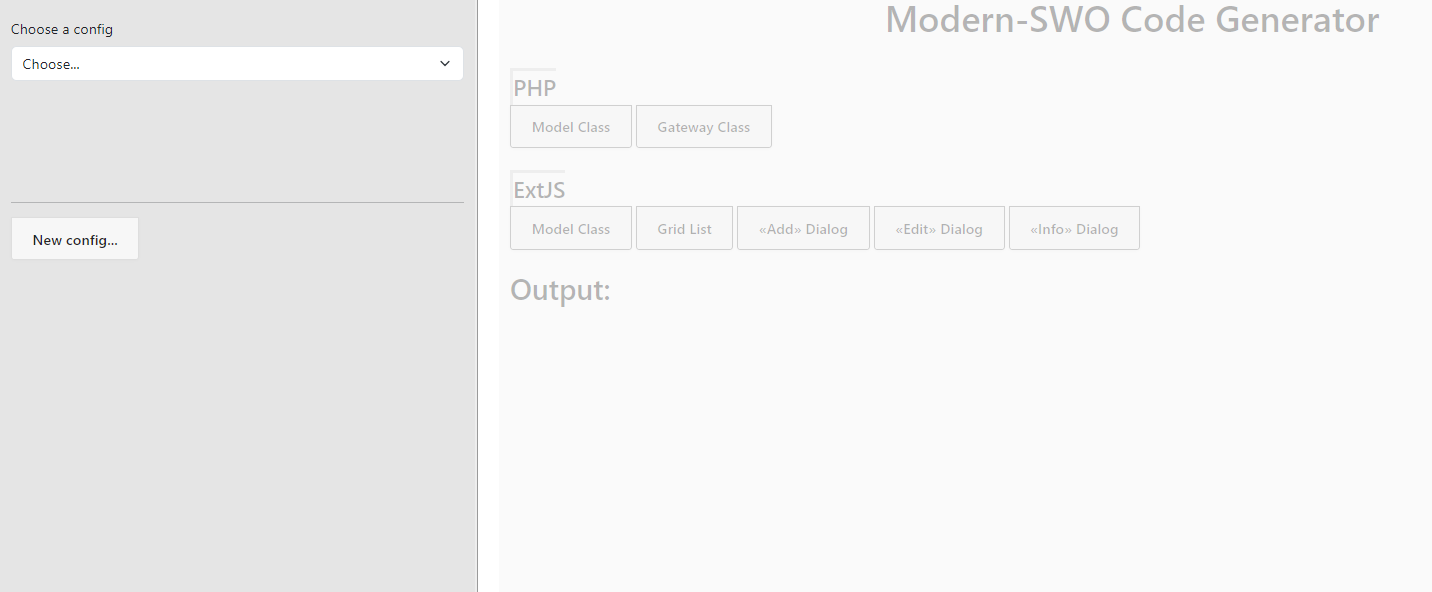


Abbildung 21 - Frontend abgedeckt

Die graue Überdeckung verschwindet erst, wenn eine Tabelle gewählt wurde:

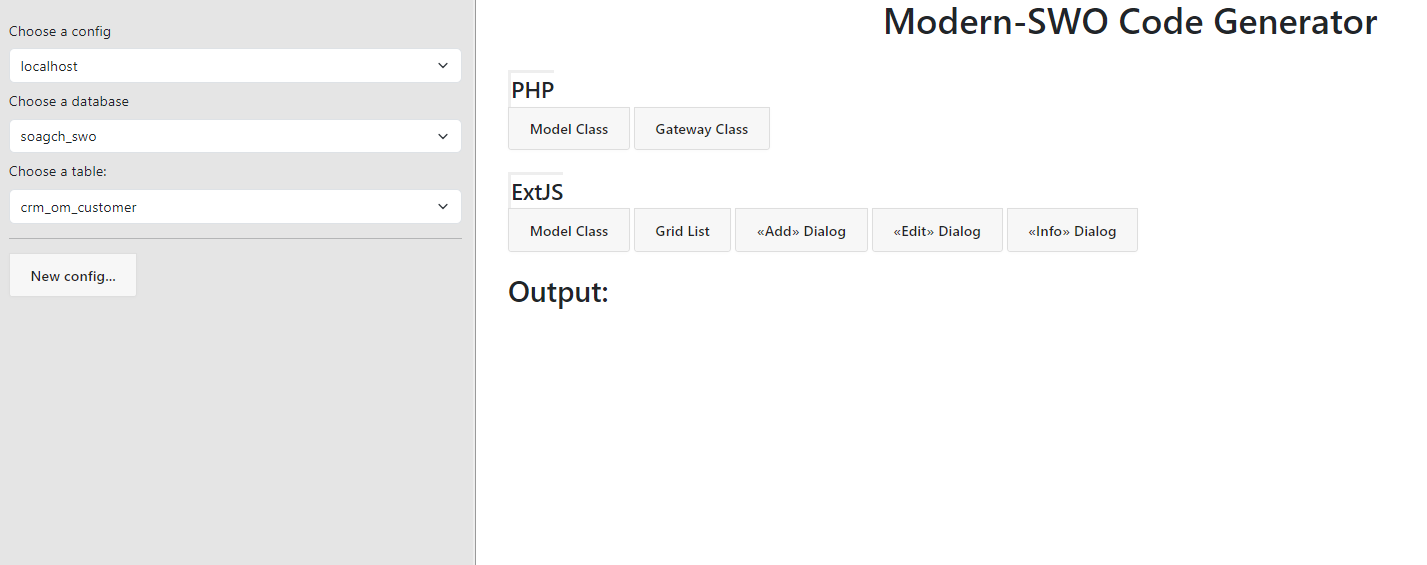


Abbildung 22 - Frontend freigeschaltet

Dies habe ich wieder mithilfe des onChange Events gemacht.

Wenn einer der Buttons geklickt wird, löst das die AJAX-Request aus, die das entsprechende Code-Snippet generiert.

Sobald das Snippet vom Backend zurückgegeben wird, erscheint es in einem schwarzen Output Fenster.

In diesem Fenster ist dann auch der Copy-Button, mitwelchem der Inhalt des Fensters kopiert werden kann:

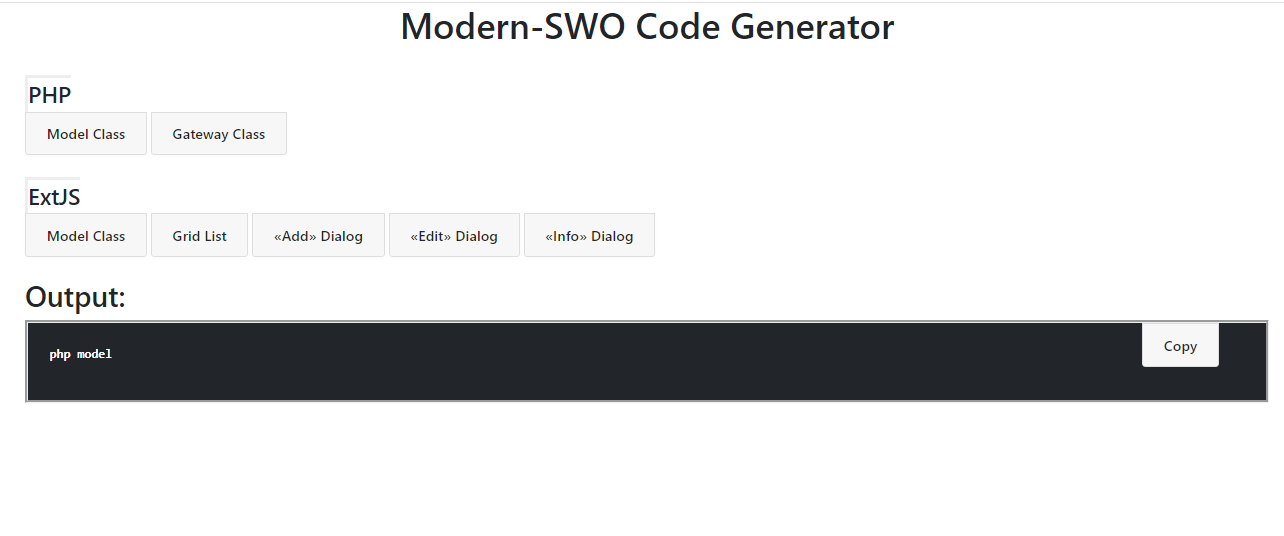


Abbildung 23 - Frontend Copy-Button

Bei der Umsetzung dieser Kopier-Funktion habe ich Zeit verloren, da ich dachte ich weiss bereits, wie man etwas in die Zwischenablage kopiert, jedoch habe ich bisher nur Inhalte aus Input-Feldern kopiert, und nicht Text aus einem Div-Element.

Es stellte sich heraus, dass man nicht, wie ich es mir gewohnt war, mithilfe der Funktion «navigator.clipboard.writeText» den Inhalt eines Divs in die Zwischenablage speichern konnte. Ansonsten hätte ich die Funktion ganze einfach so gestaltet:

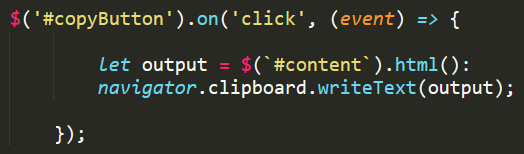


Abbildung 24 – Frontend copy-ClickEvent alt

Wieso dieser Weg nicht funktioniert, verstehe ich immer noch nicht genau, jedoch konnte ich im Internet einen Workaround finden.

Auf StackOverflow habe ich folgenden Beitrag gefunden:

<https://stackoverflow.com/questions/36639681/how-to-copy-text-from-a-div-to-clipboard>

Diesen habe ich mir zu nutzen gemacht, und die darin vorgeschlagene Lösung implementiert.

Es wird gezeigt, wie die createTextRange Funktion verwendet werden kann, um einen Inhalt auszuwählen und zu kopieren.

Ausserdem hat jemand in einem Kommentar erwähnt, dass man die Selektion wieder aufheben sollte, was ich dann auch gemacht habe.

Ich habe es in zwei Funktionen aufgeteilt:

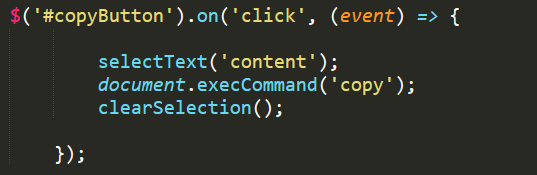


Abbildung 25 – Frontend copy-ClickEvent neu

Diese machen folgendes:



Abbildung 26 – Frontend copy-Funktionen

Nun funktioniert das Kopieren einwandfrei.

### Snippet-Generierung

Ich muss sieben verschiedene Code-Snippets generieren können.

Es gibt zwei grundsätzlich verschiedene Snippet-Arten; PHP-Snippets und ExtJs-Snippets.

Deshalb habe ich die Controller für diese Snippets aufgeteilt, und von einer Parent-Klasse namens SnippetControllerBasic geerbt, welche alle Methoden definiert, die beide Snippet-Arten brauchen.

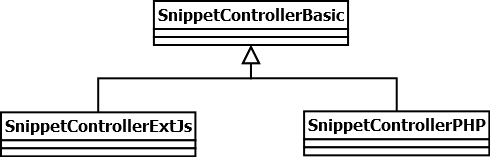


Abbildung 27 - Klassendiagramm Vererbung

Was zum Beispiel bei beiden Snippets gleich ist, ist die Namensgebung der entstehenden PHP- oder ExtJs-Klasse.

Alle Tabellen haben als Namen ein Präfix, ein zweites Präfix zur weiteren Unterteilung, und dann entweder noch mehr Präfixes oder dann der wirkliche Name der Tabelle.

Ist der Datenbank Tabellenname «erp\_article\_service»,

soll der Klassenname als «ERParticleService» definiert werden.

Also das erste Präfix alles gross, das zweite alles klein, und ab dem dritten immer nur der erste Buchstabe gross.

Somit habe ich im SnippetControllerBasic folgende Methode definiert:

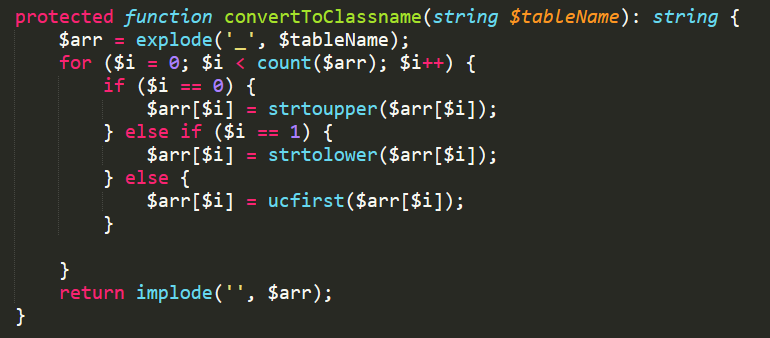


Abbildung 28 - SnippetControllerBasic convertToClassname-Methode

Wie man sieht, wird der Name der Tabelle übergeben, nach dem Bodenstrichzeichen aufgeteilt, jeder Teil bearbeitet und dann wieder zusammengesetzt und zurückgegeben.

Zur korrekten Darstellung von Abständen habe ich die Methode prepareFields definiert, welche immer vor der Snippet-Generation aufgerufen wird, um das Array worin die Tabelle abgebildet ist zu ergänzen und für jeden Feldnamen und jeden Dateityp so viele Non‑Breaking Space Charaktere hinzuzufügen, damit später alle Elemente mit den richtigen abständen benutzt werden können.



Abbildung 29 - SnippetControllerBasic prepareFields Methode

Wie man sieht, wird ebenfalls ein Array entgegengenommen, um ungewünschte Felder aus dem Tabellen-Array zu entfernen.

Ausserdem ist die Methode indent definiert, welche einen int Parameter entgegennimmt und dann soviel Non-Breaking Space Charaktere zurückgibt, wie man für das einrücken in einer Konstante definiert hat:

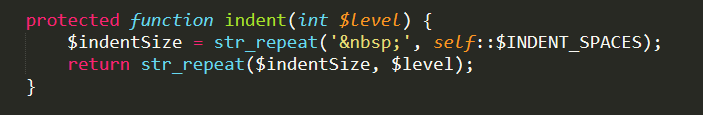


Abbildung 30 - SnippetControllerBasic indent-Methode

Die letzte Methode, die im SnippetControllerBasic definiert wird, heisst removeLastOccurrence und ist dafür gedacht, nach der Generation bei Array-Deklarationen usw. das letzte Komma, oder newLine-Zeichen zu entfernen:

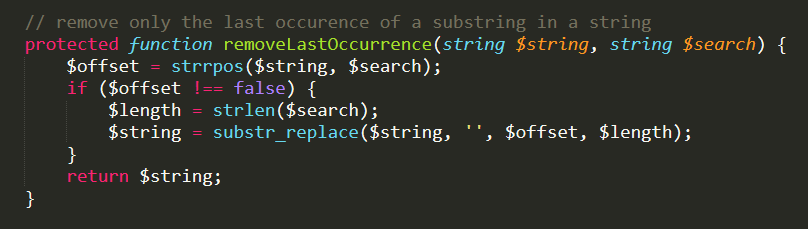


Abbildung 31 -SnippetControllerBasic removeLastOccurrence-Methode

#### PHP-Model-Klasse Snippet

Die Model-Klasse ist ein Abbild einer Datenbank Tabelle. Sie wird verwendet, um einen Datensatz der entsprechenden Tabelle zu laden und manipulieren.

In unserer neuen SWO-Software haben wir bereits mehrere solche Klassen erstellt, jedoch muss ich zuerst noch definieren, welche Teile der Klasse wirklich über alle Tabellen hinweg gleich sein müssen, und darf nicht generieren, was zu spezifisch für eine Einzelne oder wenige Tabellen ist.

Ich werde folgende Abschnitte generieren:



Attribute

JSON-Serialize

Getters & Setters

Objekterstellung

Abbildung 32 - PHP-Model-Klasse-Snippet Abschnitte

Wie man sehen kann, werden bei den Attributen und bei der Objekterstellung immer Standartwerte definiert. Diese sind pro Datentyp fix, und können somit im SnipetControllerPHP als Konstante abgebildet werden. Falls diese Standardwerte dann mal ändern, muss man sie nur an einem Ort ändern.



Abbildung 33 - SnippetControllerPHP DEFAULT\_VALUES

Die Generierung der Klasse ist in Methoden aufgeteilt. Pro Abschnitt gibt es eine Methode. Wir haben eine Methode mit Sichtbarkeit public, welche mithilfe von Member-Methoden die gesamte Klasse zusammensetzt und dann das Response-Objekt damit befüllt und ausgibt.

Alle nötigen Informationen zur Snippet-Generierung werden vom Table-Model geholt und dann jeder Member-Methode übergeben.

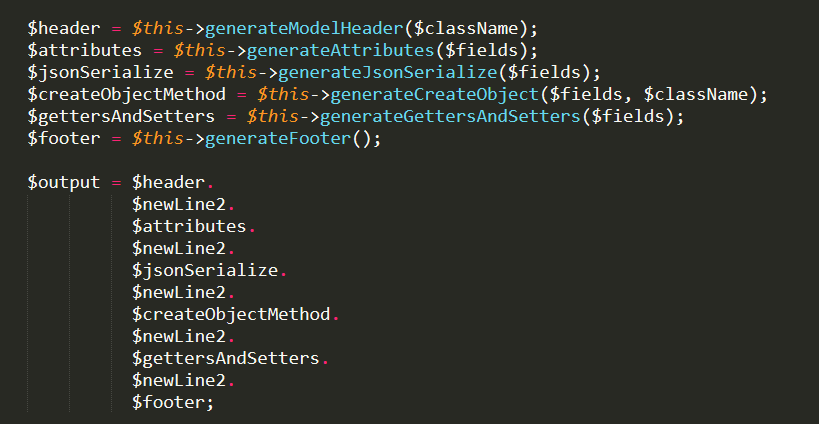


Abbildung 34 - SnippetControllerPHP Ausschnitt - Zusammensetzung der Klasse

In der Methode generateModelHeader wird lediglich der Anfang der PHP-Datei gemacht, sowie die Klassendefinition:

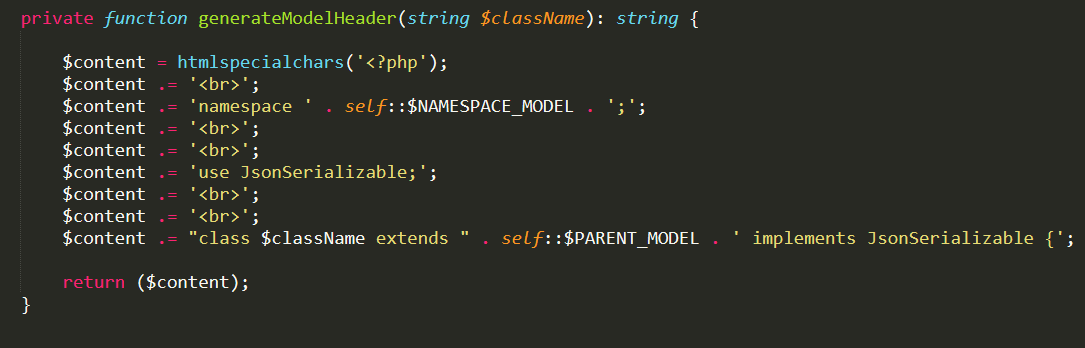


Abbildung 35 - SnippetControllerPHP Ausschnitt - generateModelHeader-Methode

Die Methode generateFooter schliesst lediglich den Scope der Klasse.

Zur Übersetzung von SQL-Datentypen in die gewünschten PHP-Datentypen habe ich die Parent-Methode prepareFields überschrieben:

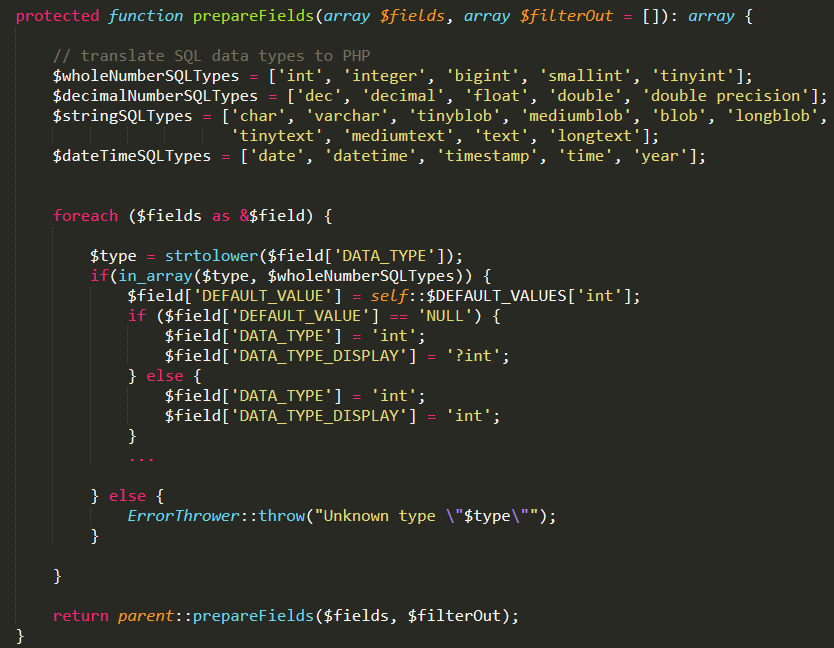


Abbildung 36 - SnippetControllerPHP Ausschnitt - prepareFields-Methode

Die Generation der einzelnen Abschnitte läuft immer sehr ähnlich ab.

Die Member-Methode generateJsonSerialize ist ein gutes Beispiel für die Code-Generation:

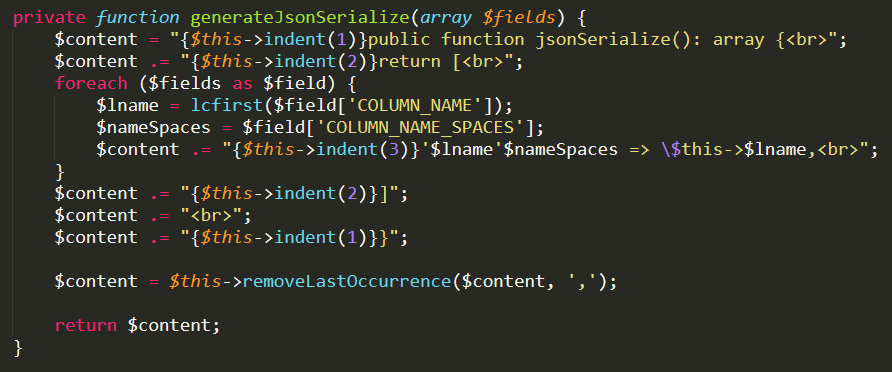


Abbildung 37 - SnippetControllerPHP Ausschnitt - generateJsonSerialize-Methode

So funktioniert jede Generation, es wird über die Felder iteriert, und daraus wird ein String zusammengesetzt.

Hier sieht man auch, wie einige Methoden aus der Parent-Klasse Verwendung finden, und bei der Generation eines sauberen Code-Snippets beitragen.

Als weiteres Beispiel zeige ich die Generation der Attribute, ergo die Member-Methode generateAttributes:

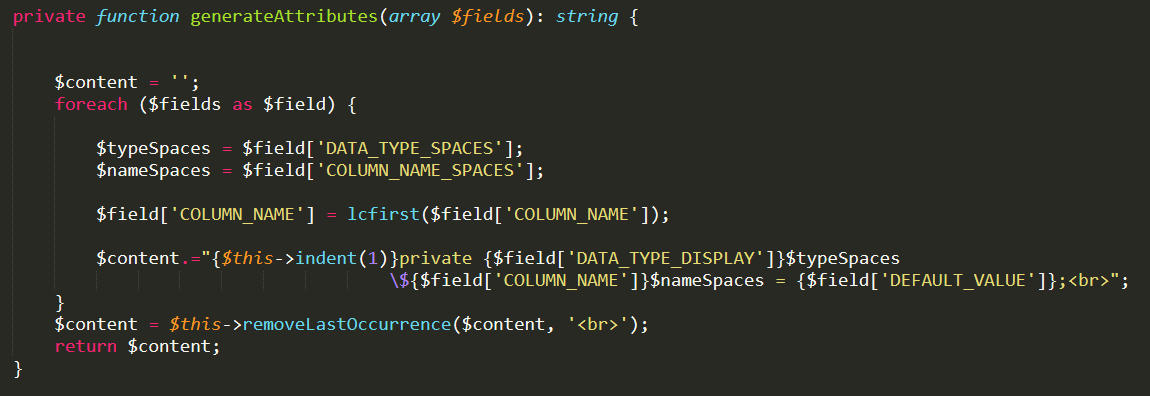


Abbildung 38 - SnippetControllerPHP Ausschnitt - generateAttributes-Methode

Man sieht wie die Space Charaktere, die in der prepareFields-Methode hinzugefügt worden sind, benutzt werden.

Die Hauptzeile dieser Methode, wo über die $fields Variable iteriert wird und die $content Variable abgefüllt wird, sollte auf einer Zeile sein, jedoch habe ich sie für eine bessere Lesbarkeit für dieses Bild umbrochen.

Ich habe diese Zeile im Code bewusst nicht umgebrochen, denn Sie repräsentiert auch wie eine Zeile nach der Generation aussehen wird, und so ist klarer, dass diese Operation pro Iteration nur eine Zeile generiert, auch wenn ich damit gegen die verwendete PHP-Code Konvention verstosse.

Das Resultierende Code-Snippet sieht im Frontend dann so aus:

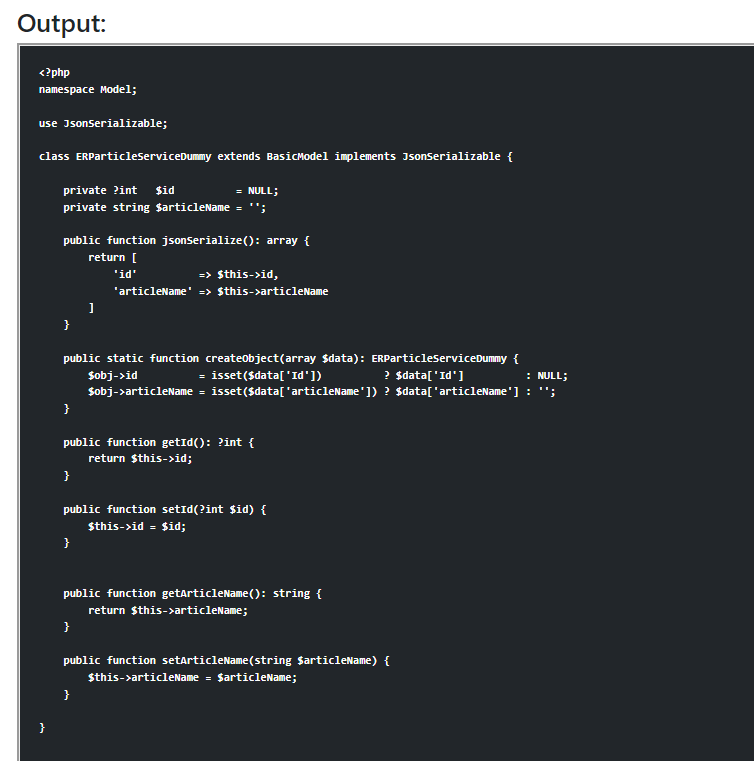


Abbildung 39 - PHP-Klasse Code-Snippet Resultat

Wie man sehen kann, habe ich eine Dummy-Tabelle für diese Demonstration verwendet, da alle anderen Tabellen zu viele Felder haben, und die Darstellung dann nicht mehr in einem angemessenen Stil möglich ist.

Zudem ist aus demselben Grund auch der Copy-Button des Output-Fensters nicht ersichtlich.

#### PHP-Gateway-Klasse Snippet

Die Gateway-Klasse ist die Schnittstelle zur Datenbank und regelt alle Abfragen.

Ich werde folgende Abschnitte generieren:



Methode insert

Konstruktor

Abbildung 40 - PHP-Gateway-Klasse-Snippet Abschnitte

Es existiert die gleiche Struktur, wie bei der Generation der Model-Klasse.

Die Hauptmethode setzt alle Teile zusammen und gibt sie dann aus:

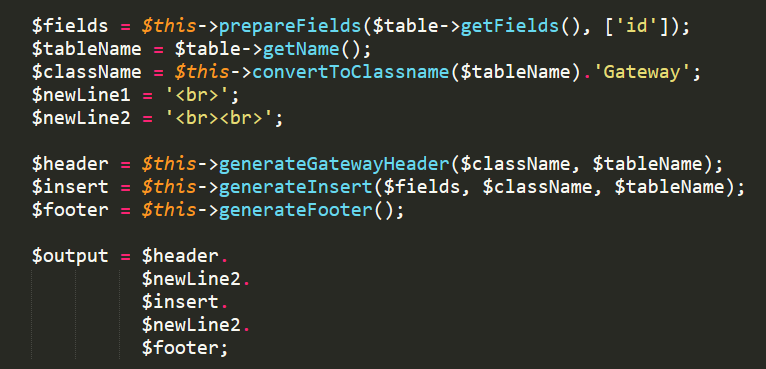


Abbildung 41 - SnippetControllerPHP Ausschnitt - Zusammensetzung des Gateways

Man sieht auch, wie im Aufruf der prepareFields-Methode das Feld „id“ aus der Tabelle ausgeschlossen wird.

Der Konstruktor wird in der generateGatewayHeader-Methode generiert, da er sehr klein ist, und ich eine eigene Methode als unnötig erachtet habe.

Die generateInsert-Methode ist um einiges Umfangreicher, und ein gutes Beispiel, wie die Generation aussieht, wenn es komplizierter wird:



Abbildung 42 - SnippetControllerPHP Ausschnitt - generateInsert-Methode

Das Resultierende Code-Snippet sieht im Frontend dann so aus:

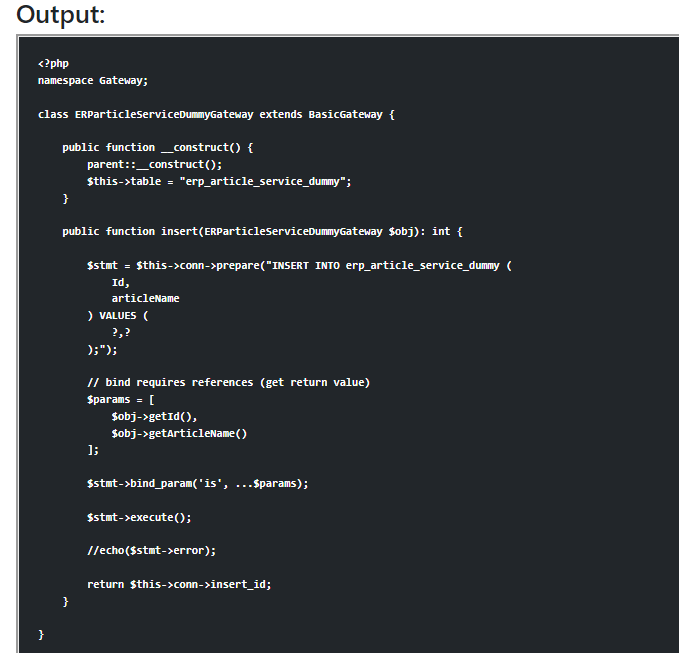


Abbildung 43 - PHP-Gateway Code-Snippet Resultat

#### ExtJs-Model

Der Aufbau des Controller SnippetControllerExtJs ist gleich, wie beim SnippetControllerPHP.

Die Methoden, die die Header und Footer generieren, sind jedoch einzigartiger und grösser. Auch die anderen Generations-Methoden sind um einiges grösser, da in den ExtJs-Dateien viele Sachen vorkommen, die fix immer drin sein müssen (mit Ausnahme des ExtJs-Models).

Das ExtJs-Model ist nämlich lediglich ein Mapping von Feldnamen und Datentypen:

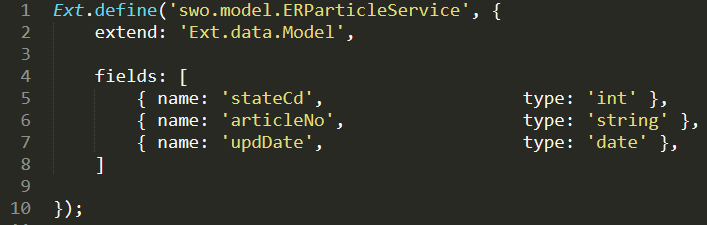


Abbildung 44 – ExtJs-Model-Snippet

Zuerst wird der Header generiert:

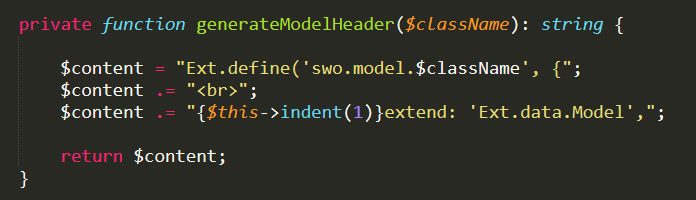


Abbildung 45 - SnippetControllerExtJs Ausschnitt - generateModelHeader-Methode

Und dann das Mapping:

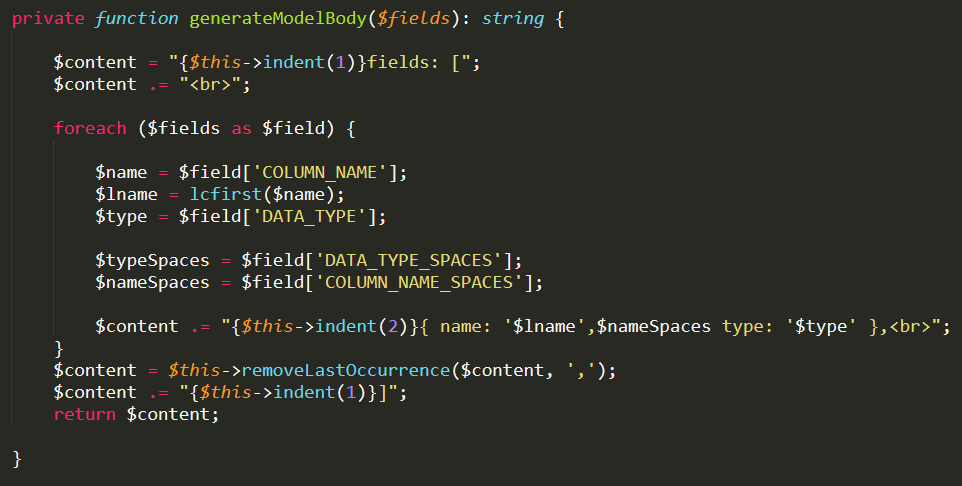


Abbildung 46 - SnippetControllerExtJs Ausschnitt - generateModelBody-Methode

Wie immer werden diese Teile durch die public Methode zusammengesetzt und ausgegeben:

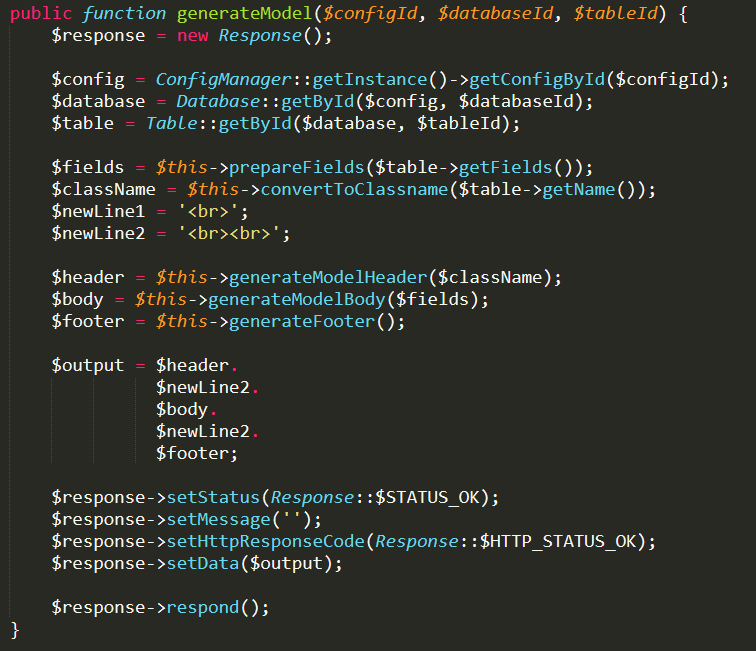


Abbildung 47 - SnippetControllerExtJs Ausschnitt - generateModel-Methode

Das Resultierende Code-Snippet sieht im Frontend dann so aus:

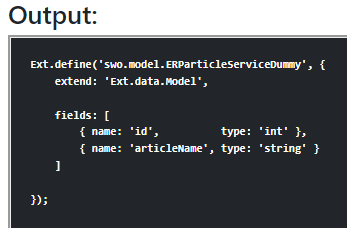


Abbildung 48 - ExtJs-Model Code-Snippet Resultat

#### ExtJs-Grid-List

Die Grid-Liste ist eine Art von ExtJs-Klasse, in der eine Liste definiert werden kann.

Sie wird in der neuen SWO-Software zur Auflistung und Bearbeitung ganzer Tabellen verwendet.

Sie hat einen grossen Header, in welchem unter anderem viele Abhängigkeiten definiert werden:

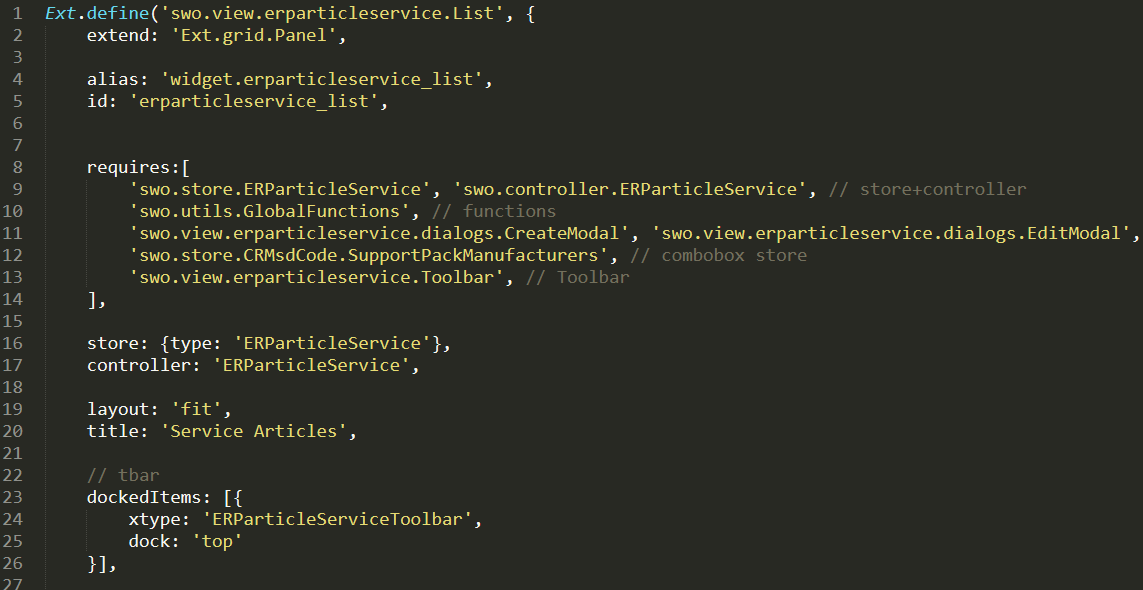


Abbildung 49 - ExtJs-Grid-List-Snippet Header

Danach kommen alle Spalten der Liste:

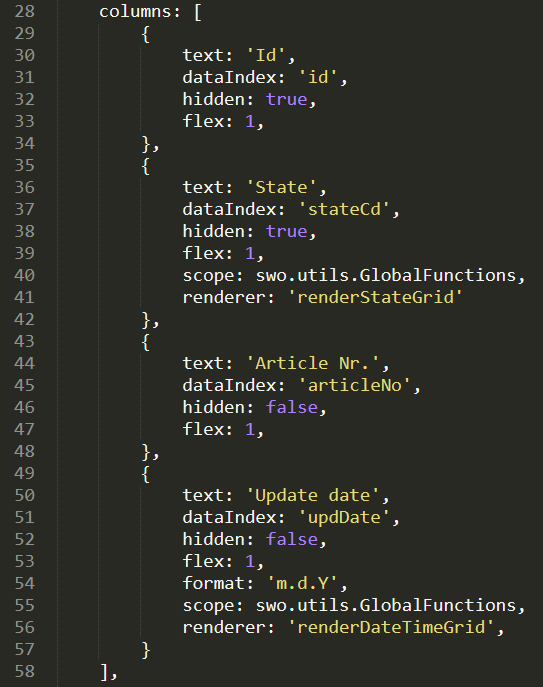


Abbildung 50 - ExtJs-Grid-List-Snippet Columns

Und am Ende noch eine Toolbar und ein Listener:

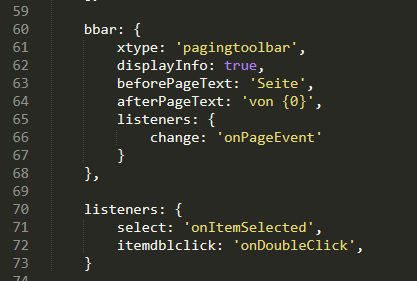


Abbildung 51 - ExtJs-Grid-List-Snippet Toolbar & Listener

Nur schon die generateGridListHeader-Methode ist zu gross, um auf einem Bild abzubilden. Hier ist ein Ausschnitt zur Veranschaulichung:



Abbildung 52 - SnippetControllerExtJs Ausschnitt - generateGridListHeader-Methode Ausschnitt

Die generateGridListBody-Methode ist dann wieder viel kleiner, da nur die Iterativen Operationen durchgeführt werden müssen:

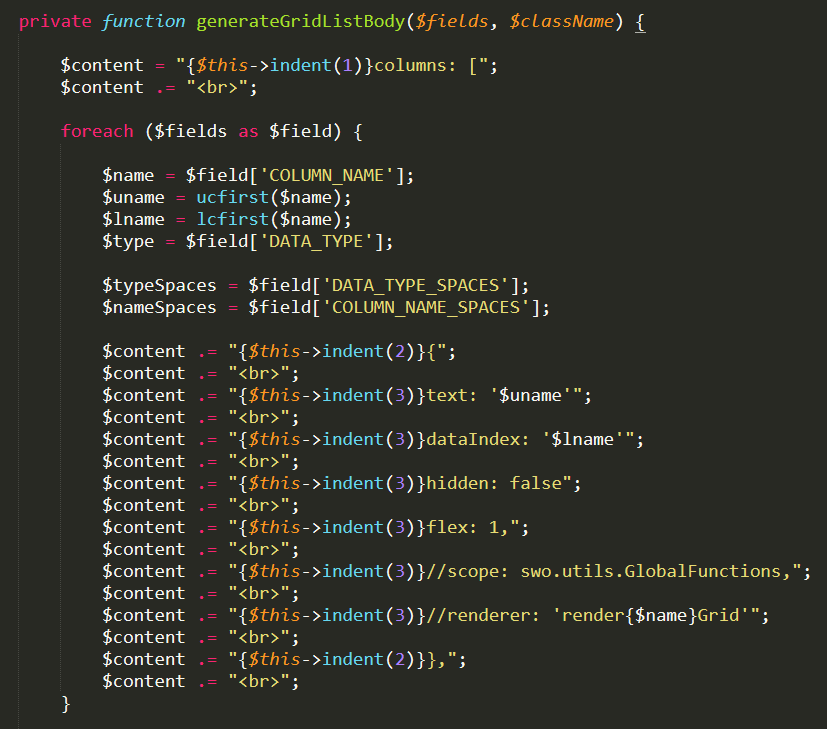


Abbildung 53 - SnippetControllerExtJs Ausschnitt - generateGridListBody-Methode Ausschnitt 1

Aber wächst am Ende wieder, da die Toolbar und der Listener noch hinzugefügt werden müssen:



Abbildung 54 - SnippetControllerExtJs Ausschnitt - generateGridListBody-Methode Ausschnitt 2

Das Resultierende Code-Snippet ist zu gross, um ein sinnvolles Bild davon erfassen zu können, deshalb habe ich den Header-Teil ausgeschlossen, da dieser Grösstenteils sowieso Hartkodiert ist.

Restliches Snippet:

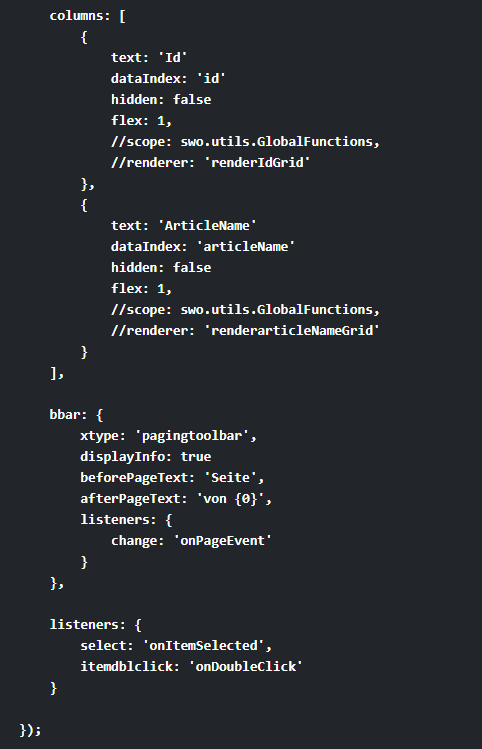


Abbildung 55 - ExtJs-GridList Code-Snippet Resultat

#### ExtJs Dialoge

Ich habe mich entschieden, die drei Dialoge aufgrund ihrer minimalen Differenzen zusammenzufassen.

Der unterschied zu allen anderen Snippets ist, dass die public Methoden, welche die Snippets zusammensetzen und ausgeben, den Member-Methoden entweder die Request-Method oder den Dialognamen mitgeben, um unterscheiden zu können, welcher Dialog genau gemacht werden muss:

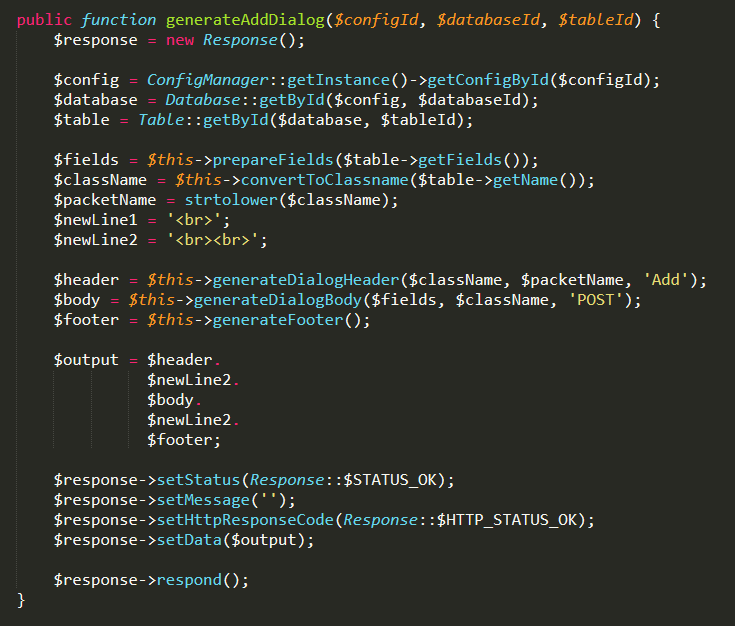


Abbildung 56 - SnippetControllerExtJs Ausschnitt - generateAddDialog-Methode

Der Header wird zum Beispiel zu generiert:



Abbildung 57 - SnippetControllerExtJs Ausschnitt - generateDialogHeader-Methode

Somit entsteht am Ende bei allen drei eine Ausgabe, die wenige unterschiede aufzeigt.

Im folgenden Bild habe ich den Add-Dialog generiert und markiert, was bei den anderen Dialogen im Header anders sein würde. Auch hier wird nur der Header gezeigt, da das Resultat zu gross ist:

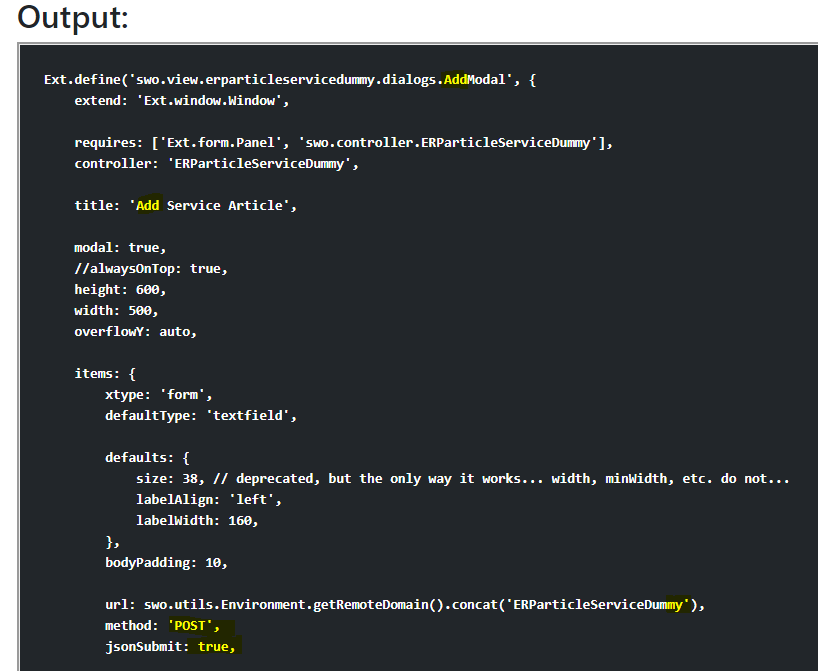


Abbildung 58 - ExtJs-Dialog Code-Snippet Resultat

### Klassendiagramm

Zur technischen Dokumentation habe ich ein Klassendiagramm erstellt.

Dieses habe ich in zwei Teile aufgeteilt, damit eine sinnvolle Darstellung möglich ist.

Es gibt den Controlling Teil, und den Model Teil.

Würde ich die Verbindungen dieser beiden Teile ausweisen, wäre das Diagramm zu unübersichtlich.

#### Controlling

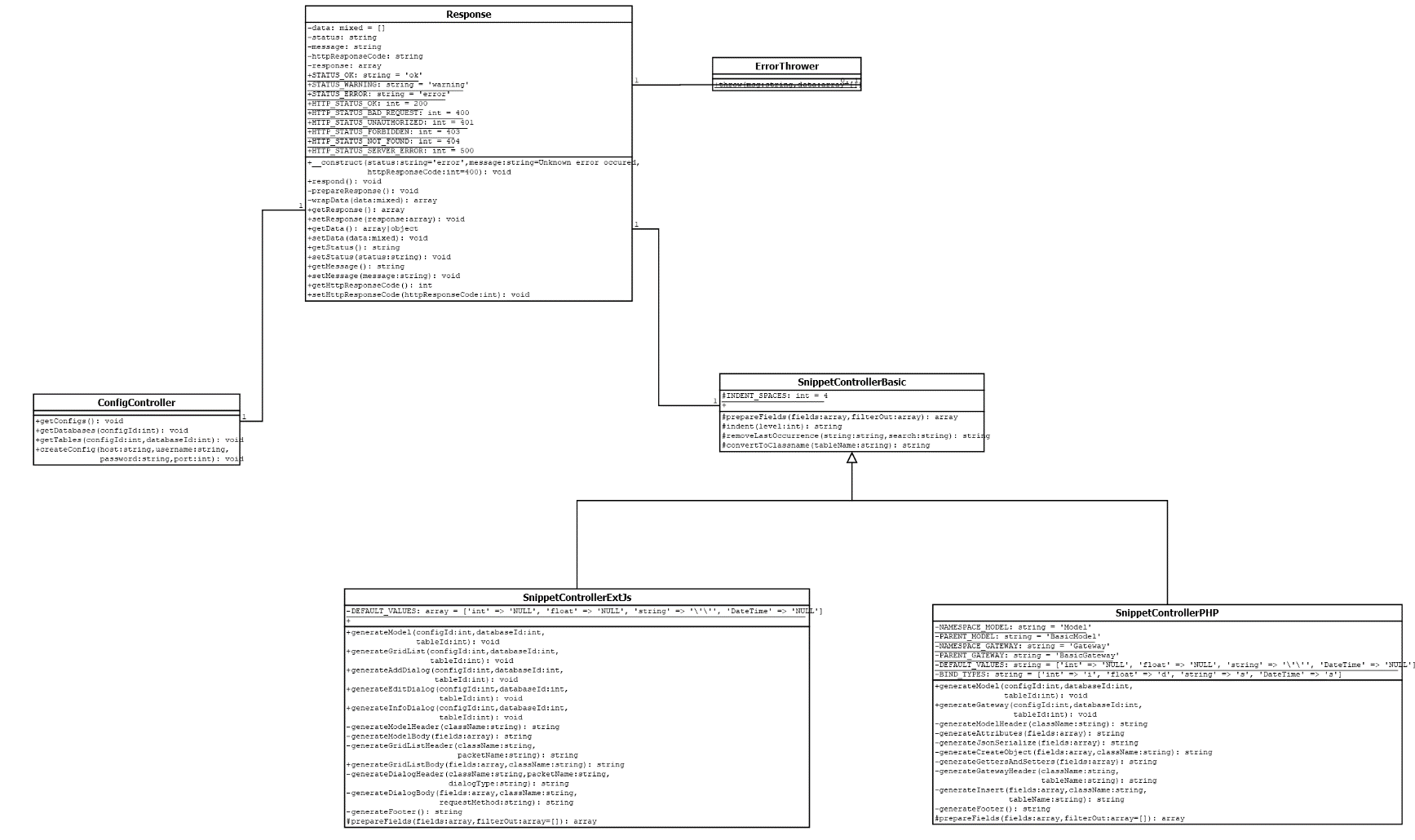


Abbildung 59 - Klassendiagramm Controlling

#### Model

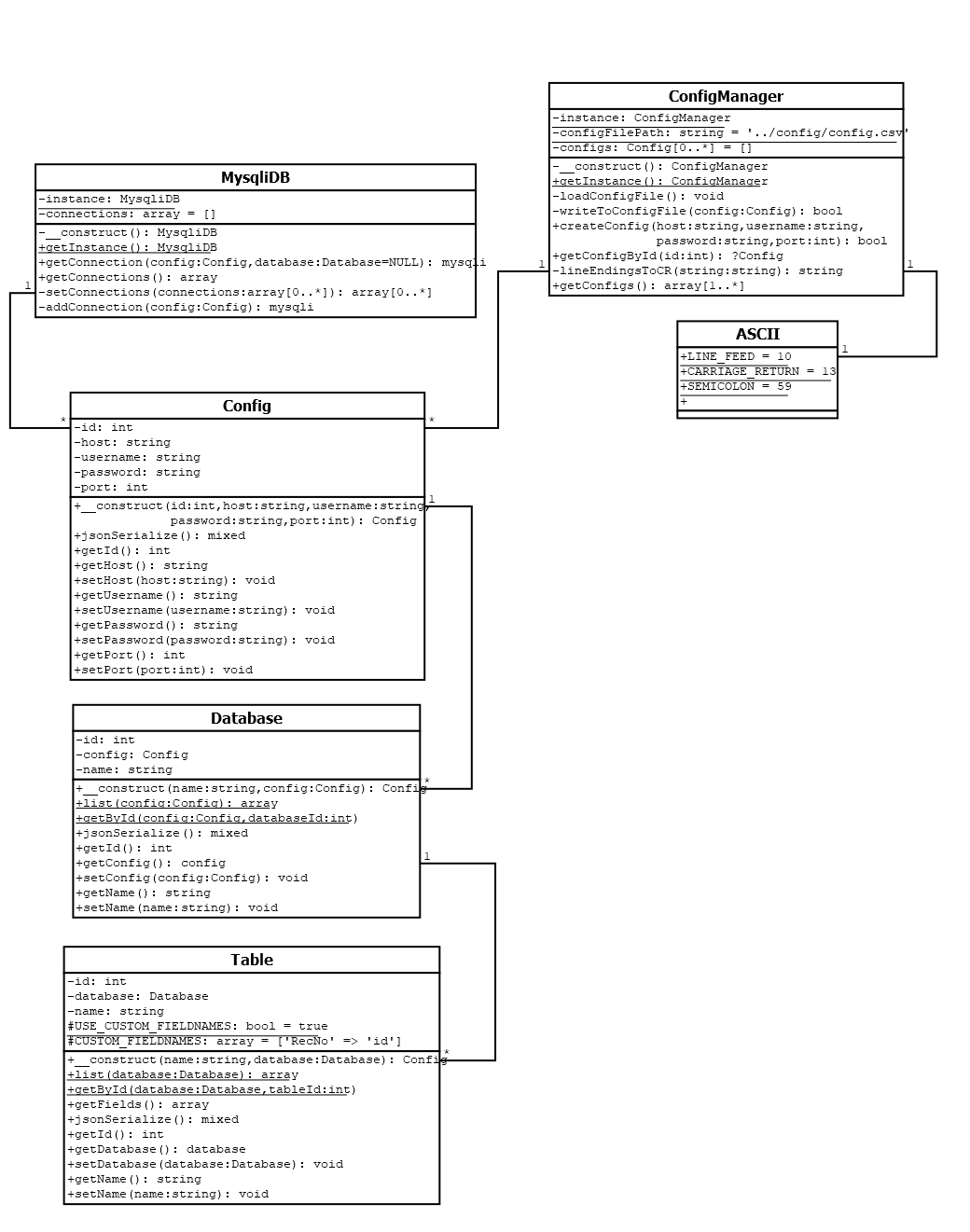


Abbildung 60 - Klassendiagramm Model

## Kontrollieren

Zur Qualitätssicherung ist folgendes Testprotokoll definiert, welches alle notwendigen Funktionen der Applikation abdeckt.

Die Tests werden manuell durchgeführt.

### Testfälle

|  |  |
| --- | --- |
| **ID** | T\_001 |
| **Beschreibung** | Eine neue Config über das Formular hinterlegen. |
| **Testvoraussetzung** | Keine. |
| **Testschritte** | 1. «New config» Button drücken 2. Formular mit Daten befüllen 3. Formular mittels «Add» Button abschicken |
| **Erwartetes Ergebnis** | Die Config wird richtig in der Config-Datei hinterlegt und steht in der Config-Selectbox zur Auswahl. |

Tabelle 8 - Testfall 1

|  |  |
| --- | --- |
| **ID** | T\_002 |
| **Beschreibung** | Eine Config aus der Config-Selectbox auswählen. |
| **Testvoraussetzung** | Es wurde mindestens eine Config hinterlegt. |
| **Testschritte** | 1. Auf der Selectbox mit Label «Choose a config» eine Config auswählen |
| **Erwartetes Ergebnis** | Unterhalb dieser Selectbox erscheint nun eine weitere, mit dem Label «Choose a database» |

Tabelle 9 - Testfall 2

|  |  |
| --- | --- |
| **ID** | T\_003 |
| **Beschreibung** | Eine Config aus der Config-Selectbox auswählen, mit welcher keine Datenbankverbindung aufgebaut werden kann. |
| **Testvoraussetzung** | Es wurde mindestens eine Config hinterlegt, auf welche eine Datenbankverbindung unmöglich ist (z.B. mit Host, der kein Server ist oder absichtlich falschem Passwort). |
| **Testschritte** | 1. Auf der Selectbox mit Label «Choose a config» eine fehlerhafte Config auswählen |
| **Erwartetes Ergebnis** | Unterhalb dieser Selectbox erscheint ein Ladezeichen, welches durch eine Fehlermeldung ersetzt wird, sobald der Server Antwort gibt. |

Tabelle 10 - Testfall 3

|  |  |
| --- | --- |
| **ID** | T\_004 |
| **Beschreibung** | Eine Datenbank aus der Datenbank-Selectbox auswählen. |
| **Testvoraussetzung** | Es wurde eine Config in der Config-Selectbox ausgewählt. |
| **Testschritte** | 1. Auf der Selectbox mit Label «Choose a database» eine Datenbank auswählen |
| **Erwartetes Ergebnis** | Unterhalb dieser Selectbox erscheint nun eine weitere, mit dem Label «Choose a table» |

Tabelle 11 - Testfall 4

|  |  |
| --- | --- |
| **ID** | T\_005 |
| **Beschreibung** | Eine Tabelle aus der Tabellen-Selectbox auswählen. |
| **Testvoraussetzung** | Es wurde eine Datenbank in der Datenbank-Selectbox ausgewählt. |
| **Testschritte** | 1. Auf der Selectbox mit Label «Choose a table» eine Tabelle auswählen |
| **Erwartetes Ergebnis** | Der Rechte Bereich des Layouts verliert die graue Überdeckung, und man kann damit interagieren (Buttons drücken). |

Tabelle 12 - Testfall 5

|  |  |
| --- | --- |
| **ID** | T\_006 |
| **Beschreibung** | Ein PHP-Model-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation bekannt sind. |
| **Testschritte** | 1. Button «Model Class» unter PHP klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem das Code-Snippet dargestellt wird. |

Tabelle 13 - Testfall 6

|  |  |
| --- | --- |
| **ID** | T\_007 |
| **Beschreibung** | Ein PHP-Model-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation unbekannt sind. |
| **Testschritte** | 1. Button «Model Class» unter PHP klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem eine Fehlermeldung und der Unbekannte Dateityp, der den Fehler ausgelöst hat, ausgegeben werden. |

Tabelle 14 - Testfall 7

|  |  |
| --- | --- |
| **ID** | T\_008 |
| **Beschreibung** | Ein PHP-Gateway-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation bekannt sind. |
| **Testschritte** | 1. Button «Gateway Class» unter PHP klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem das Code-Snippet dargestellt wird. |

Tabelle 15 - Testfall 8

|  |  |
| --- | --- |
| **ID** | T\_009 |
| **Beschreibung** | Ein PHP-Gateway-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation unbekannt sind. |
| **Testschritte** | 1. Button «Gateway Class» unter PHP klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem eine Fehlermeldung und der Unbekannte Dateityp, der den Fehler ausgelöst hat, ausgegeben werden. |

Tabelle 16 - Testfall 9

|  |  |
| --- | --- |
| **ID** | T\_010 |
| **Beschreibung** | Ein ExtJs-Model-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation bekannt sind. |
| **Testschritte** | 1. Button «Model Class» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem das Code-Snippet dargestellt wird. |

Tabelle 17 - Testfall 10

|  |  |
| --- | --- |
| **ID** | T\_011 |
| **Beschreibung** | Ein ExtJs-Model -Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation unbekannt sind. |
| **Testschritte** | 1. Button «Model Class» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem eine Fehlermeldung und der Unbekannte Dateityp, der den Fehler ausgelöst hat, ausgegeben werden. |

Tabelle 18 - Testfall 11

|  |  |
| --- | --- |
| **ID** | T\_012 |
| **Beschreibung** | Ein ExtJs-GridList-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation bekannt sind. |
| **Testschritte** | 1. Button «Grid List» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem das Code-Snippet dargestellt wird. |

Tabelle 19 - Testfall 12

|  |  |
| --- | --- |
| **ID** | T\_013 |
| **Beschreibung** | Ein ExtJs-GridList-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation unbekannt sind. |
| **Testschritte** | 1. Button «Grid List» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem eine Fehlermeldung und der Unbekannte Dateityp, der den Fehler ausgelöst hat, ausgegeben werden. |

Tabelle 20 - Testfall 13

|  |  |
| --- | --- |
| **ID** | T\_014 |
| **Beschreibung** | Ein ExtJs-Add-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation bekannt sind. |
| **Testschritte** | 1. Button «Add Dialog» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem das Code-Snippet dargestellt wird. |

Tabelle 21 - Testfall 14

|  |  |
| --- | --- |
| **ID** | T\_015 |
| **Beschreibung** | Ein ExtJs- Add -Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation unbekannt sind. |
| **Testschritte** | 1. Button «Add Dialog» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem eine Fehlermeldung und der Unbekannte Dateityp, der den Fehler ausgelöst hat, ausgegeben werden. |

Tabelle 22 - Testfall 15

|  |  |
| --- | --- |
| **ID** | T\_016 |
| **Beschreibung** | Ein ExtJs-Edit-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation bekannt sind. |
| **Testschritte** | 1. Button «Edit Dialog» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem das Code-Snippet dargestellt wird. |

Tabelle 23 - Testfall 16

|  |  |
| --- | --- |
| **ID** | T\_017 |
| **Beschreibung** | Ein ExtJs- Edit -Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation unbekannt sind. |
| **Testschritte** | 1. Button «Edit Dialog» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem eine Fehlermeldung und der Unbekannte Dateityp, der den Fehler ausgelöst hat, ausgegeben werden. |

Tabelle 24 - Testfall 17

|  |  |
| --- | --- |
| **ID** | T\_018 |
| **Beschreibung** | Ein ExtJs-Info-Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation bekannt sind. |
| **Testschritte** | 1. Button «Info Dialog» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem das Code-Snippet dargestellt wird. |

Tabelle 25 - Testfall 18

|  |  |
| --- | --- |
| **ID** | T\_019 |
| **Beschreibung** | Ein ExtJs- Info -Snippet generieren. |
| **Testvoraussetzung** | Es wurde eine Tabelle in der Tabellen-Selectbox ausgewählt, die Dateitypen beinhaltet, die der Applikation unbekannt sind. |
| **Testschritte** | 1. Button «Info Dialog» unter ExtJs klicken. |
| **Erwartetes Ergebnis** | Unterhalb des Titels «Output:» erscheint ein schwarzes Fenster, in dem eine Fehlermeldung und der Unbekannte Dateityp, der den Fehler ausgelöst hat, ausgegeben werden. |

Tabelle 26 - Testfall 19

### Testprotokoll

|  |  |  |  |
| --- | --- | --- | --- |
| **Testfall ID** | **Testperson** | **Testzeitpunkt** | **Testresultat** |
| T\_001 | Luan Caduff | 02.05.2024 um 08:47 | Erfolg |
| T\_002 | Luan Caduff | 02.05.2024 um 08:50 | Erfolg |
| T\_003 | Luan Caduff | 02.05.2024 um 08:51 | Erfolg |
| T\_004 | Luan Caduff | 02.05.2024 um 08:51 | Erfolg |
| T\_005 | Luan Caduff | 02.05.2024 um 08:52 | Erfolg |
| T\_006 | Luan Caduff | 02.05.2024 um 08:52 | Erfolg |
| T\_007 | Luan Caduff | 02.05.2024 um 08:53 | Erfolg |
| T\_008 | Luan Caduff | 02.05.2024 um 08:53 | Erfolg |
| T\_009 | Luan Caduff | 02.05.2024 um 08:54 | Erfolg |
| T\_010 | Luan Caduff | 02.05.2024 um 08:54 | Erfolg |
| T\_011 | Luan Caduff | 02.05.2024 um 08:54 | Erfolg |
| T\_012 | Luan Caduff | 02.05.2024 um 08:55 | Erfolg |
| T\_013 | Luan Caduff | 02.05.2024 um 08:55 | Erfolg |
| T\_014 | Luan Caduff | 02.05.2024 um 08:55 | Erfolg |
| T\_015 | Luan Caduff | 02.05.2024 um 08:56 | Erfolg |
| T\_016 | Luan Caduff | 02.05.2024 um 08:57 | Erfolg |
| T\_017 | Luan Caduff | 02.05.2024 um 08:58 | Erfolg |
| T\_018 | Luan Caduff | 02.05.2024 um 08:58 | Erfolg |
| T\_019 | Luan Caduff | 02.05.2024 um 08:58 | Erfolg |

## Auswerten

### Projektauswertung

Der Schwerpunkt dieses Projektes war die Realisierung der Applikation selbst, wofür dementsprechend am meisten Zeit aufgewandt wurde. Das Endprodukt ist ein Code-Generator, welcher alle Kriterien der Aufgabenstellung abdeckt, und sogar noch darüber hinaus geht und mehrere Features implementiert, welche nicht gefragt waren, jedoch die Qualität in meinen Augen um einiges erhöhen.

Die Applikation ist intuitiv zu benutzen und leicht zu erweitern. Es steht ein grosses Grundgerüst für die Entwicklung weiterer Code-Snippets zur Verfügung, welches hier in dieser Dokumentation beschrieben und erklärt wurde.

Die Applikation wird auf jeden Fall einen grossen nutzen in der Weiterentwicklung unserer SWO-Software erbringen und diesen Prozess erleichtern und beschleunigen.

Die dazugehörige Dokumentation ist meiner Meinung nach nicht so gut wie die Applikation selbst, da ich den Zeitaufwand, der damit verbunden ist, unterschätzt habe. Trotzdem konnten alle Tagesjournale aktuell gehalten werden, und die gesamte Applikation zusammen mit den dazugehörigen Arbeitsschritten sinnvoll dokumentiert werden. Auch allgemeine Dokumente-Form Kriterien wurden abgedeckt.

Als Ganzes ist das Projekt, und die daraus entstandene Applikation, durchaus ein Erfolg und wird zukünftig auch von unserer Firma benutzt werden.

### Schlusswort

Das Umsetzen dieses Projekts war für mich eine großartige Erfahrung. Natürlich war es nicht immer ganz einfach, immer so konzentriert zu arbeiten, jedoch konnte ich mich sehr gut zusammenreissen und wurde selten abgelenkt. Den Erfolg zu verspüren, wenn etwas nach langer Entwicklung so funktioniert, wie man es sich vorgestellt hat, ist grandios und gibt einem ein Gefühl von Stärke und Kompetenz.

Gegen Ende wurde mir jedoch bewusst, dass die Dokumentation nicht so ausführlich und exakt werden wird, wie ich mir das gewünscht habe. Die Realisierung forderte sehr viel Zeit auf, und das Dokumentieren habe ich nicht so gut im Griff wie das Programmieren, und somit bin ich in den letzten Tagen immer mehr in Zeitdruck geraten.

## Verzeichnisse

### Abbildungsverzeichnis

[Abbildung 1 - Use-Case-Diagramm 24](#_Toc165651212)

[Abbildung 2 - MockUp 1 26](#_Toc165651213)

[Abbildung 3 - MockUp 2 27](#_Toc165651214)

[Abbildung 4 - MockUp Variante 2 29](#_Toc165651215)

[Abbildung 5 - MockUp Variante 1 29](#_Toc165651216)

[Abbildung 6 – Mockup Entscheidungsmatrix 29](#_Toc165651217)

[Abbildung 7 - Selectboxen 30](#_Toc165651218)

[Abbildung 8 - Config-Formular 31](#_Toc165651219)

[Abbildung 9 – Grundstruktur der Applikation 32](#_Toc165651220)

[Abbildung 10 - Backend router Ausschnitt 32](#_Toc165651221)

[Abbildung 11 - Controller-Klasse Ausgabe 33](#_Toc165651222)

[Abbildung 12 - Response-Klasse Ausschnitt 33](#_Toc165651223)

[Abbildung 13 - Utils-Klasse ErrorThrower 33](#_Toc165651224)

[Abbildung 14 - ConfigController getConfigs Funktion 34](#_Toc165651225)

[Abbildung 15 - ConfigController createConfig Methode 35](#_Toc165651226)

[Abbildung 16 - ConfigManager Dateibearbeitung 35](#_Toc165651227)

[Abbildung 17 - ConfigController getTables-Methode 37](#_Toc165651228)

[Abbildung 18 - Database getById-Methode 37](#_Toc165651229)

[Abbildung 19 - MysqliDB getConnection-Methode 38](#_Toc165651230)

[Abbildung 20 - Table list-Methode 38](#_Toc165651231)

[Abbildung 21 - Frontend abgedeckt 39](#_Toc165651232)

[Abbildung 22 - Frontend freigeschaltet 39](#_Toc165651233)

[Abbildung 23 - Frontend Copy-Button 40](#_Toc165651234)

[Abbildung 24 – Frontend copy-ClickEvent alt 40](#_Toc165651235)

[Abbildung 25 – Frontend copy-ClickEvent neu 41](#_Toc165651236)

[Abbildung 26 – Frontend copy-Funktionen 41](#_Toc165651237)

[Abbildung 27 - Klassendiagramm Vererbung 42](#_Toc165651238)

[Abbildung 28 - SnippetControllerBasic convertToClassname-Methode 42](#_Toc165651239)

[Abbildung 29 - SnippetControllerBasic prepareFields Methode 43](#_Toc165651240)

[Abbildung 30 - SnippetControllerBasic indent-Methode 43](#_Toc165651241)

[Abbildung 31 -SnippetControllerBasic removeLastOccurrence-Methode 43](#_Toc165651242)

[Abbildung 32 - PHP-Model-Klasse-Snippet Abschnitte 44](#_Toc165651243)

[Abbildung 33 - SnippetControllerPHP DEFAULT\_VALUES 45](#_Toc165651244)

[Abbildung 34 - SnippetControllerPHP Ausschnitt - Zusammensetzung der Klasse 45](#_Toc165651245)

[Abbildung 35 - SnippetControllerPHP Ausschnitt - generateModelHeader-Methode 45](#_Toc165651246)

[Abbildung 36 - SnippetControllerPHP Ausschnitt - prepareFields-Methode 46](#_Toc165651247)

[Abbildung 37 - SnippetControllerPHP Ausschnitt - generateJsonSerialize-Methode 46](#_Toc165651248)

[Abbildung 38 - SnippetControllerPHP Ausschnitt - generateAttributes-Methode 47](#_Toc165651249)

[Abbildung 39 - PHP-Klasse Code-Snippet Resultat 48](#_Toc165651250)

[Abbildung 40 - PHP-Gateway-Klasse-Snippet Abschnitte 49](#_Toc165651251)

[Abbildung 41 - SnippetControllerPHP Ausschnitt - Zusammensetzung des Gateways 50](#_Toc165651252)

[Abbildung 42 - SnippetControllerPHP Ausschnitt - generateInsert-Methode 50](#_Toc165651253)

[Abbildung 43 - PHP-Gateway Code-Snippet Resultat 51](#_Toc165651254)

[Abbildung 44 – ExtJs-Model-Snippet 52](#_Toc165651255)

[Abbildung 45 - SnippetControllerExtJs Ausschnitt - generateModelHeader-Methode 52](#_Toc165651256)

[Abbildung 46 - SnippetControllerExtJs Ausschnitt - generateModelBody-Methode 52](#_Toc165651257)

[Abbildung 47 - SnippetControllerExtJs Ausschnitt - generateModel-Methode 53](#_Toc165651258)

[Abbildung 48 - ExtJs-Model Code-Snippet Resultat 53](#_Toc165651259)

[Abbildung 49 - ExtJs-Grid-List-Snippet Header 54](#_Toc165651260)

[Abbildung 50 - ExtJs-Grid-List-Snippet Columns 54](#_Toc165651261)

[Abbildung 51 - ExtJs-Grid-List-Snippet Toolbar & Listener 55](#_Toc165651262)

[Abbildung 52 - SnippetControllerExtJs Ausschnitt - generateGridListHeader-Methode Ausschnitt 55](#_Toc165651263)

[Abbildung 53 - SnippetControllerExtJs Ausschnitt - generateGridListBody-Methode Ausschnitt 1 56](#_Toc165651264)

[Abbildung 54 - SnippetControllerExtJs Ausschnitt - generateGridListBody-Methode Ausschnitt 2 56](#_Toc165651265)

[Abbildung 55 - ExtJs-GridList Code-Snippet Resultat 57](#_Toc165651266)

[Abbildung 56 - SnippetControllerExtJs Ausschnitt - generateAddDialog-Methode 58](#_Toc165651267)

[Abbildung 57 - SnippetControllerExtJs Ausschnitt - generateDialogHeader-Methode 58](#_Toc165651268)

[Abbildung 58 - ExtJs-Dialog Code-Snippet Resultat 59](#_Toc165651269)

[Abbildung 59 - Klassendiagramm Controlling 60](#_Toc165651270)

[Abbildung 60 - Klassendiagramm Model 61](#_Toc165651271)

### Tabellenverzeichnis

[Tabelle 1 - Dokumentenhistorie 4](#_Toc165651272)

[Tabelle 2 - Dokumenteneigenschaften 4](#_Toc165651273)

[Tabelle 3 - Code-Snippet Arten 6](#_Toc165651274)

[Tabelle 4 - Zeitplan Zusammenfassung 21](#_Toc165651275)

[Tabelle 5 - Use-Case Beschreibung «add new config» 25](#_Toc165651276)

[Tabelle 6 - Use-Case Beschreibung «choose database table» 25](#_Toc165651277)

[Tabelle 7 - Use-Case Beschreibung «generate Code-Snippet» 25](#_Toc165651278)

[Tabelle 8 - Testfall 1 62](#_Toc165651279)

[Tabelle 9 - Testfall 2 62](#_Toc165651280)

[Tabelle 10 - Testfall 3 63](#_Toc165651281)

[Tabelle 11 - Testfall 4 63](#_Toc165651282)

[Tabelle 12 - Testfall 5 63](#_Toc165651283)

[Tabelle 13 - Testfall 6 64](#_Toc165651284)

[Tabelle 14 - Testfall 7 64](#_Toc165651285)

[Tabelle 15 - Testfall 8 64](#_Toc165651286)

[Tabelle 16 - Testfall 9 65](#_Toc165651287)

[Tabelle 17 - Testfall 10 65](#_Toc165651288)

[Tabelle 18 - Testfall 11 65](#_Toc165651289)

[Tabelle 19 - Testfall 12 66](#_Toc165651290)

[Tabelle 20 - Testfall 13 66](#_Toc165651291)

[Tabelle 21 - Testfall 14 66](#_Toc165651292)

[Tabelle 22 - Testfall 15 67](#_Toc165651293)

[Tabelle 23 - Testfall 16 67](#_Toc165651294)

[Tabelle 24 - Testfall 17 67](#_Toc165651295)

[Tabelle 25 - Testfall 18 68](#_Toc165651296)

[Tabelle 26 - Testfall 19 68](#_Toc165651297)

[Tabelle 27 – Glossar 75](#_Toc165651298)

### Links

<https://www.ict-berufsbildung-bern.ch/resources/Iperka_OdA_200617.pdf>

Website der ICT-Berufsbildung Bern

Heruntergeladen am: 18.04.2024

<https://www.php-fig.org/psr/psr-1/>

Webseite der PHP Framework Interop Group

Heruntergeladen am: 18.04.2024

<https://www.lucidchart.com/pages/>

Web-Tool für MockUps

Erstellt am: 18.04.2024

<https://stackoverflow.com/questions/36639681/how-to-copy-text-from-a-div-to-clipboard>

Hilfestellung beim Copy-Problem

Angesehen am: 25.04.2024

### Glossar / Abkürzungen

|  |  |
| --- | --- |
| CRUD | Create, Read, Update, Delete |
| SWO | Simple Web Office |
| AJAX | Asynchronous JavaScript and XML |

Tabelle 27 – Glossar

## Anhang

Der Anhang beinhaltet lediglich den Quellcode, und wurde separat hochgeladen.