

# IPA-Bericht

## Code-Generator



Autor	Luan Caduff
Klasse	ISO-20
Datum	03. Mai 2024
Firma	Evernex IT Services Switzerland AG

## Table of Contents

<b>1</b>	<b>Teil 1.....</b>	<b>4</b>
1.1	Dokumenteninformationen .....	4
1.1.1	Historie.....	4
1.1.2	Eigenschaften .....	4
1.2	Aufgabenstellung.....	5
1.3	Projektorganisation.....	7
1.4	Projektmethode.....	7
1.5	Deklaration der Vorkenntnisse .....	8
1.6	Deklaration der Vorarbeiten.....	8
1.7	Deklaration der benutzen Firmenstandards.....	8
1.8	Organisation der Arbeitsergebnisse .....	9
1.9	Zeitplan .....	10
1.10	Arbeitsjournal .....	11
1.10.1	Tag 1, 18. April 2024.....	11
	Tag 2, 19. April 2024 .....	12
1.10.2	Tag 3, 23. April 2024.....	13
1.10.3	Tag 4, 24. April 2024.....	14
1.10.4	Tag 5, 25. April 2024.....	15
1.10.5	Tag 6, 26. April 2024 .....	16
1.10.6	Tag 7, 30. April 2024.....	17
1.10.7	Tag 8, 01. Mai 2024 .....	18
1.10.8	Tag 9, 02. Mai 2024.....	19
1.10.9	Tag 10, 03. Mai 2024 .....	20
1.10.11	Zusammenfassung Zeitplan .....	21
<b>2</b>	<b>Teil 2.....</b>	<b>22</b>
2.1	Management Summary (Kurzfassung des IPA-Berichts).....	22
2.1.1	Ausgangslage .....	22
2.1.2	Vorgehen .....	22
2.1.3	Ergebnis.....	22
2.2	Informieren.....	23
2.2.1	Aufgabestellung.....	23
2.2.2	Technologien.....	23
2.2.3	Use-Cases .....	24
2.3	Planen.....	26
2.3.1	GUI MockUps.....	26
2.4	Entscheiden .....	29

<b>2.5</b>	<b>Realisieren.....</b>	<b>30</b>
2.5.1	Tabellen-Auswahl Frontend.....	30
2.5.2	Config-Formular.....	31
2.5.3	Grundaufbau Backend.....	32
2.5.4	Datenbank Verbindungen (Config).....	34
2.5.5	Tabellen-Auswahl Backend.....	36
2.5.6	Snippet-Auswahl Frontend .....	40
2.5.7	Snippet-Generierung.....	43
<b>2.6</b>	<b>Kontrollieren.....</b>	<b>61</b>
2.6.1	Testfälle .....	61
2.6.2	Testprotokoll .....	68
<b>2.7</b>	<b>Auswerten .....</b>	<b>69</b>
2.7.1	Projektauswertung.....	69
2.7.2	Schlusswort.....	70
<b>2.8</b>	<b>Verzeichnisse.....</b>	<b>71</b>
2.8.1	Abbildungsverzeichnis.....	71
2.8.2	Tabellenverzeichnis.....	72
2.8.3	Links.....	73
2.8.4	Glossar / Abkürzungen .....	73
<b>2.9</b>	<b>Anhang.....</b>	<b>74</b>

# 1 Teil 1

## 1.1 Dokumenteninformationen

### 1.1.1 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
1.0.0.11	03.05.2024	Realisieren, Auswerten	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)

### 1.1.2 Eigenschaften

Bezeichnung	Detailinformationen
Status	Abgeschlossen
Autor	Luan Caduff
Ausbildung zum	Eidg. Dipl. Informatiker EFZ
Fachrichtung	Applikationsentwickler
Version	1.0.0.11
Versionsdatum	03.05.2024
Seiten	74

Tabelle 2 - Dokumenteneigenschaften

## 1.2 Aufgabenstellung

Die Firma S + O AG ist Partnerfirma der Evernex IT Services Switzerland AG und zuständig für den Betrieb und die Weiterentwicklung 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.

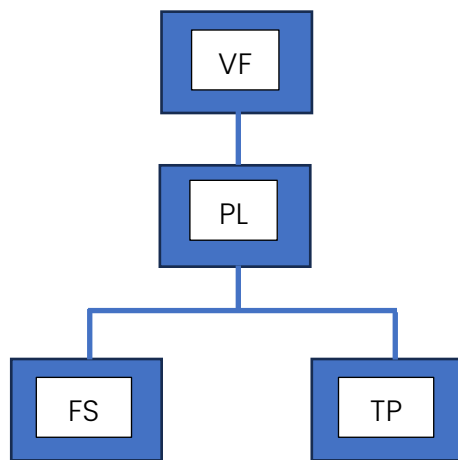
Die Architektur soll MVC nach Firmenusanz abbilden: Klassen mit entsprechenden Funktionalitäten in entsprechenden Verzeichnissen. Dazu existieren keine dokumentierten Firmenstandards.

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.

## 1.3 Projektorganisation



VF – Roman Born (Verantwortliche Fachkraft)

PL – Luan Caduff (Projektleiter)

FS – Luan Caduff (Fachspezialist)

TP – Luan Caduff (Testperson)

## 1.4 Projektmethode

Dieses Projekt wird mit IPERKA durchgeführt. Dies ist eine simple Projektmanagement-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](https://www.ict-berufsbildung-bern.ch/resources/Iperka_OdA_200617.pdf)

## 1.5 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.

## 1.6 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.

## 1.7 Deklaration der benutzten Firmenstandards

Es existieren keine dokumentierten Firmenstandards.

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

Die Spezifikationen sind unter folgendem Link dokumentiert:

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

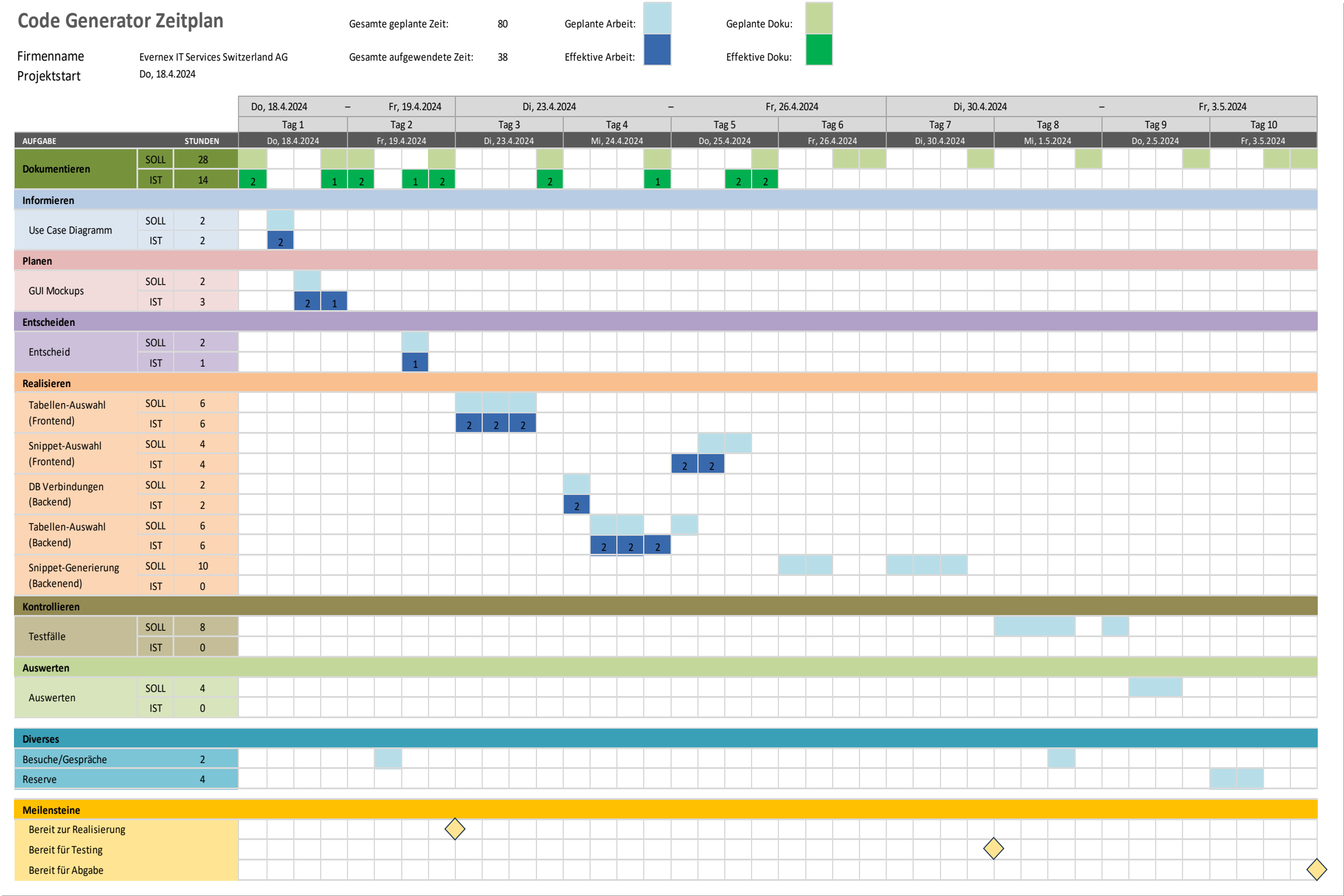


## 1.8 Organisation der Arbeitsergebnisse

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

Das Repository wird täglich (ebenfalls vor Arbeitsende) auf GitHub gepusht:  
<https://github.com/xivia/ipa-luan---code-generator>

1.9 Zeitplan



## 1.10 Arbeitsjournal

### 1.10.1 Tag 1, 18. April 2024

Arbeiten	<p>Dokumentation</p> <ul style="list-style-type: none"> <li>▪ Zeitplan erstellt</li> <li>▪ Im IPA Dokument Teil 1 abgefüllt</li> </ul> <p>Informieren</p> <ul style="list-style-type: none"> <li>▪ Use Case Diagramm erstellt</li> </ul> <p>Planen</p> <ul style="list-style-type: none"> <li>▪ MockUps erstellt</li> </ul>
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	<p>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.</p> <p>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.</p>

Tag 2, 19. April 2024

Arbeiten	Erstes Gespräch mit HEX Dokumentation <ul style="list-style-type: none"><li>▪ Generell Dokumentation verschönert</li><li>▪ Use Case Beschreibungen verfasst</li></ul> Entscheiden <ul style="list-style-type: none"><li>▪ GUI-Varianten Entscheid gefällt</li></ul>
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.

## 1.10.2 Tag 3, 23. April 2024

Arbeiten	Dokumentation <ul style="list-style-type: none"><li>▪ Realisierung dokumentiert</li></ul> Realisieren <ul style="list-style-type: none"><li>▪ Frontend Tabellen-Auswahl</li></ul>
Probleme	Keine.
Hilfestellungen	Keine.
Überzeiten	Keine.
Ungeplante Arbeiten	Keine.
Erfolge	Alle Arbeiten wurden zufriedenstellend abgeschlossen.
Misserfolge	Keine.
Reflektion	<p>Ich konnte die geplanten 6 Stunden für die Tabellen-Auswahl effizient ausnützen.</p> <p>Für die verschiedenen Selectboxen musste ich Dummy-Daten hartkodieren, da noch kein Backend existiert.</p> <p>Zudem habe ich die Formularmaske zur Einrichtung neuer Datenbankverbindungen (Configs) erstellt.</p> <p>Ich finde die Fade-In und Fade-Out Effekte visuell ansprechend und habe sie implementiert, da ich noch genügend Zeit dafür hatte.</p> <p>Das Resultat entspricht dem MockUp soweit ziemlich genau.</p>

## 1.10.3 Tag 4, 24. April 2024

Arbeiten	Dokumentation <ul style="list-style-type: none"><li>▪ Realisierung dokumentiert</li></ul> Realisieren <ul style="list-style-type: none"><li>▪ Backend DB-Verbindungen und Tabellen-Auswahl</li></ul>
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	<p>Zuerst habe ich die Config-Datei inklusive Auslesung und Bearbeitung erstellt.</p> <p>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.</p> <p>In der Dokumentation bin ich ein wenig hinten, und sollte langsam aufholen.</p>

## 1.10.4 Tag 5, 25. April 2024

Arbeiten	Dokumentation <ul style="list-style-type: none"> <li>▪ Realisierung dokumentiert</li> </ul> Realisieren <ul style="list-style-type: none"> <li>▪ Frontend Snippet-Auswahl</li> </ul>
Probleme	Kopieren in die Zwischenablage komplizierter als gedacht.
Hilfestellungen	Keine.
Überzeiten	Keine.
Ungeplante Arbeiten	Keine.
Erfolge	Alle praktischen Arbeiten wurden zufriedenstellend abgeschlossen.
Misserfolge	Keine.
Reflektion	<p>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</p> <p>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.</p> <p>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.</p>

## 1.10.5 Tag 6, 26. April 2024

Arbeiten	Dokumentation <ul style="list-style-type: none"><li>▪ Realisierung dokumentiert</li></ul> Realisieren <ul style="list-style-type: none"><li>▪ Backend Snippet-Generierung PHP</li></ul>
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.



## 1.10.6 Tag 7, 30. April 2024

Arbeiten	Dokumentation <ul style="list-style-type: none"><li>▪ Realisierung dokumentiert</li></ul> Realisieren <ul style="list-style-type: none"><li>▪ Backend Snippet-Generierung ExtJs</li></ul>
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.

## 1.10.7 Tag 8, 01. Mai 2024

Arbeiten	Dokumentation <ul style="list-style-type: none"><li>▪ Kontrollieren dokumentiert</li><li>▪ Realisieren Dokumentiert</li></ul> Kontrollieren <ul style="list-style-type: none"><li>▪ Testfälle definiert.</li></ul>
Probleme	Keine.
Hilfestellungen	Keine.
Überzeiten	Keine.
Ungeplante Arbeiten	Keine.
Erfolge	Alle Arbeiten wurden zufriedenstellend abgeschlossen.
Misserfolge	Keine.
Reflektion	<p>Heute habe ich alle Testfälle definiert, und weiterhin die fehlenden Teile der Dokumentation in Bezug auf die Realisierung fertiggestellt.</p> <p>Für die Testfälle musste ich zuerst noch eine passende Tabelle erstellen.</p> <p>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.</p>

## 1.10.8Tag 9, 02. Mai 2024

Arbeiten	Dokumentation <ul style="list-style-type: none"><li>▪ Kontrollieren dokumentiert (Testprotokoll)</li><li>▪ Realisieren dokumentiert</li></ul> Kontrollieren <ul style="list-style-type: none"><li>▪ Testprotokoll erstellt.</li></ul>
Probleme	Keine.
Hilfestellungen	Keine.
Überzeiten	Heute habe ich zum Dokumentieren zwei Stunden Überzeit gemacht.
Ungeplante Arbeiten	Keine.
Erfolge	Alle 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.

## 1.10.9 Tag 10, 03. Mai 2024

Arbeiten	Dokumentation <ul style="list-style-type: none"><li>▪ Realisieren dokumentiert</li></ul> Auswertung <ul style="list-style-type: none"><li>▪ Projektauswertung und Schlusswort verfasst.</li></ul>
Probleme	Keine.
Hilfestellungen	Keine.
Überzeiten	Keine.
Ungeplante Arbeiten	Keine.
Erfolge	Alle Arbeiten wurden zufriedenstellend abgeschlossen.
Misserfolge	Keine.
Reflektion	

## 1.10.11 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	

Tabelle 4 - Zeitplan Zusammenfassung

Begründung → siehe Arbeitsjournal

## 2 Teil 2

### 2.1 Management Summary (Kurzfassung des IPA-Berichts)

#### 2.1.1 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.

#### 2.1.2 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.

#### 2.1.3 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.

## 2.2 Informieren

### 2.2.1 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.

#### **Begrifflichkeiten:**

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

### 2.2.2 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

## 2.2.3 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/>

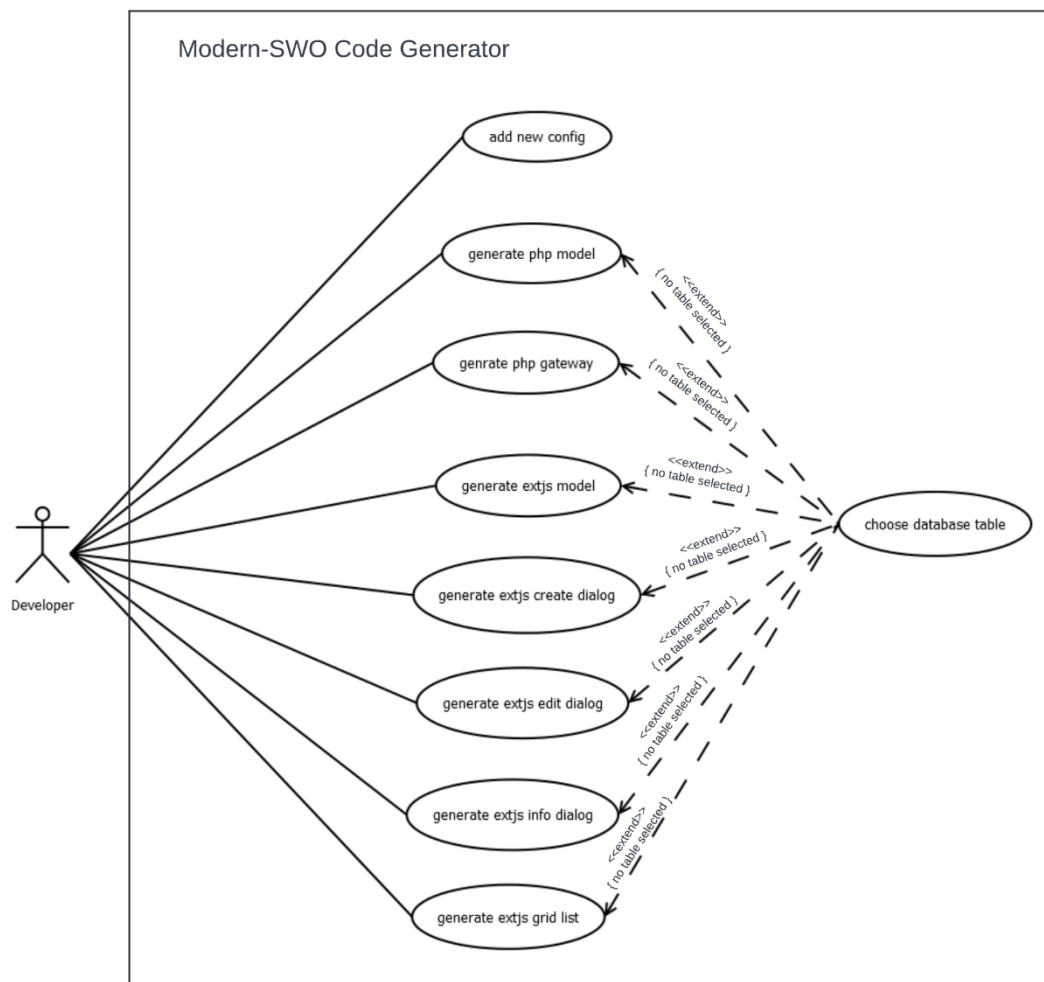


Abbildung 1 - Use-Case-Diagramm



## 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»

## 2.3 Planen

### 2.3.1 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

The mockup shows a web application titled "Modern-SWO Code Generator". On the left is a sidebar with three sections: "Config Label" containing a "<selectbox>" button, "DB Label" containing a "<selectbox>" button, and "Table Label" containing a "<selectbox>" button. Below these is a "new" button. The main area on the right has a title "Modern-SWO Code Generator". Under the title, there are two sections: "PHP" with two "gen" buttons, and "ExtJS" with five "gen" buttons. Below these is an "Output:" label. The output area is a large dark gray rectangle labeled "output box" in the center. A "copy" button is located in the top right corner of the output area.

Abbildung 2 - MockUp 1

## Entwurf 2

### Modern-SWO Code Generator

new

Config Label

<selectbox>

DB Label

<selectbox>

Table Label

<selectbox>

---

PHP

gen

gen

ExtJS

gen

gen

gen

gen

gen

Output:

copy

output box

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.

## 2.4 Entscheiden

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

Variante 1

oder

Variante 2

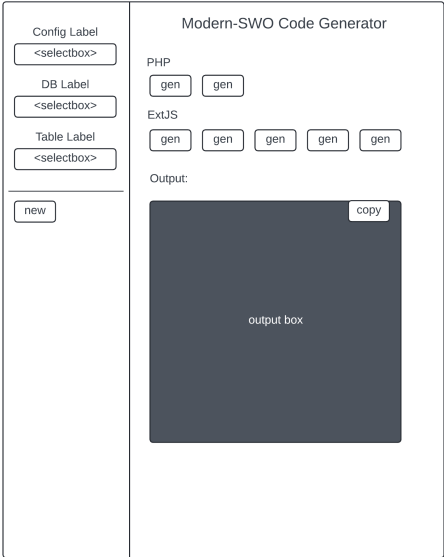


Abbildung 5 – MockUp Variante 1

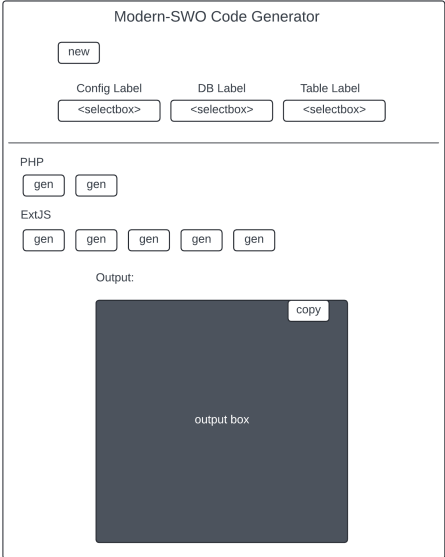


Abbildung 4 – MockUp Variante 2

Punktevergabe	Gewichtung	Variante 1		Variante 2	
0: nicht erfüllt	1: seltener Fall	Punkte	Gewichtete Punkte	Punkte	Gewichtete Punkte
1: erfüllt OK	2: häufiger Fall				
2: erfüllt gut					
3: erfüllt perfekt					
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

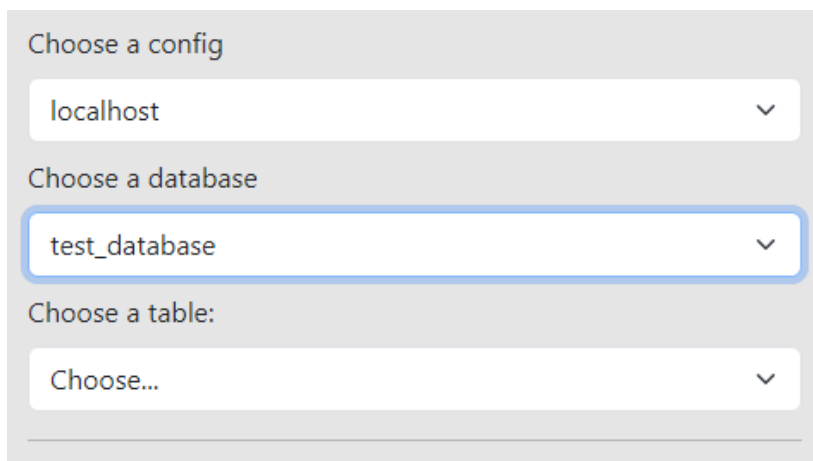
Dies war ein knapper Entscheid, beide Varianten sind eine valide Wahl, jedoch entscheide ich mich nun für Variante 1.

## 2.5 Realisieren

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

### 2.5.1 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.



Choose a config

localhost

Choose a database

test\_database

Choose a table:

Choose...

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.

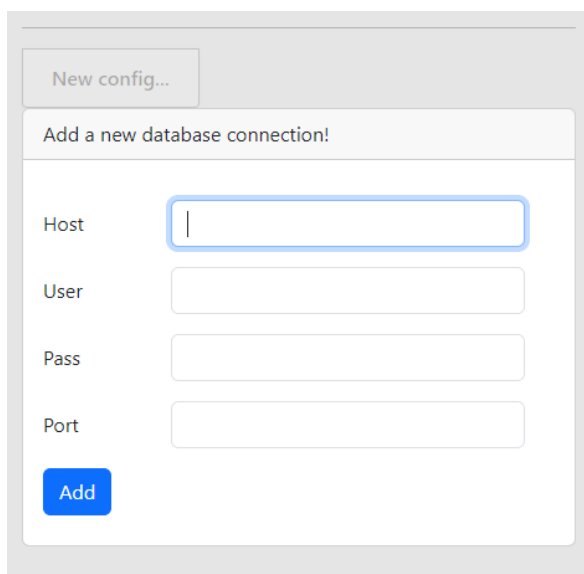
Falls im Backend beim Datenbankverbindungsversuch ein Fehler auftritt, wird eine Meldung unterhalb der Config Selectbox erscheinen.

## 2.5.2 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:



The image shows a modal window titled 'Add a new database connection!'. At the top left of the modal is a button labeled 'New config...'. The form contains four input fields: 'Host', 'User', 'Pass', and 'Port'. The 'Host' field is currently selected, indicated by a blue border and a vertical cursor. Below the input fields is a blue button labeled 'Add'.

Abbildung 8 – Config-Formular

## 2.5.3 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:

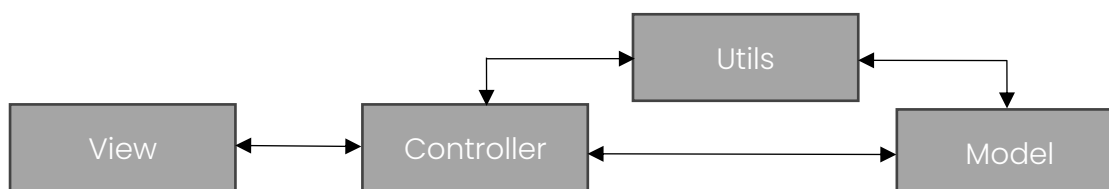


Abbildung 9 – Grundstruktur der Applikation

Im Controller habe ich eine PHP-Datei namens router.php erstellt. Diese ist die einzige Backend-Datei, welche keine Klasse ist. Sie ist die Kommunikationschnittstelle 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.

```

1  <?php
2  namespace Controller;
3
4  error_reporting(E_ALL);
5  ini_set('display_errors', 'On');
6
7  spl_autoload_register(function ($class_name) {
8      $class_name = str_replace('\\', DIRECTORY_SEPARATOR, $class_name);
9      require_once('..' . $class_name . '.php');
10 });
11
12
13
14 if(isset($_GET['action']) && $_GET['action'] == 'getConfigs') {
15
16     $controller = new ConfigController();
17     $controller->getConfigs();
18     exit();
19 }
20
21
22 if(isset($_GET['action']) && $_GET['action'] == 'getDatabases') {
23
24     $configId = isset($_GET['configId']) ? $_GET['configId'] : 0;
25
26     $controller = new ConfigController();
27     $controller->getDatabases($configId);
28
29     exit();
30 }
31
32 // . . .
33
34 echo(['msg': 'fallthrough']);
35
36 ?>
  
```

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.

```
public function getDatabases($configId) {
    $response = new Response();

    $config = ConfigManager::getInstance()->getConfigById($configId);
    $databases = Database::list($config);

    $response->setStatus(Response::$STATUS_OK);
    $response->setMessage('');
    $response->setHttpStatusCode(Response::$HTTP_STATUS_OK);
    $response->setData($databases);

    $response->respond();
}
```

Abbildung 11 – Controller-Klasse Ausgabe

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

```
33 public function respond() {
34     $this->prepareResponse();
35     echo json_encode($this->response, JSON_UNESCAPED_UNICODE);
36 }
37
38 private function prepareResponse() {
39     http_response_code($this->httpResponseCode);
40
41     $this->response = [
42         'status' => $this->status,
43         'message' => $this->message,
44         'data' => $this->wrapData($this->data)
45     ];
46 }
47
48 private function wrapData(mixed $data): array {
49     // if data is not already an array put it in one
50     if(gettype($data) != 'array' && gettype($data) != 'object') {
51         return [$data];
52     } else {
53         return $data;
54     }
55 }
56 }
57
58 }
```

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.

```
1 <?php
2 namespace Utils;
3
4 class ErrorThrower {
5
6     public static function throw($msg, $data = []) {
7         $response = new Response(Response::$STATUS_ERROR, $msg, Response::$HTTP_STATUS_BAD_REQUEST);
8         $response->setData($data);
9         $response->respond();
10        exit();
11    }
12 }
```

Abbildung 13 – Utils-Klasse ErrorThrower

## 2.5.4 Datenbank Verbindungen (Config)

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:

```
public function getConfigs() {  
    $response = new Response();  
  
    $response->setStatus(Response::$STATUS_OK);  
    $response->setMessage('');  
    $response->setHttpStatusCode(Response::$HTTP_STATUS_OK);  
    $response->setData(ConfigManager::getInstance()->getConfigs());  
  
    $response->respond();  
}
```

Abbildung 14 - ConfigController getConfigs Funktion

## Hinzufügen einer Config

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

```
public function createConfig($host, $user, $password, $port) {
    $response = new Response();

    $manager = ConfigManager::getInstance();
    if($manager->createConfig($host, $user, $password, $port)) {
        $response->setStatus(Response::$STATUS_OK);
        $response->setMessage('');
        $response->setHttpStatusCode(Response::$HTTP_STATUS_OK);
    } else {
        $response->setStatus(Response::$STATUS_ERROR);
        $response->setMessage('Failed to add new config');
        $response->setHttpStatusCode(Response::$HTTP_STATUS_SERVER_ERROR);
    }

    $response->respond();
}
```

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:

```
private function writeToConfigFile(Config $config): bool {
    $configLine = chr(ASCII::$LINE_FEED);
    $configLine .= "{{$config->getId()}};{{$config->getHost()}};{{$config->getUsername()}};{{$config->getPassword()}};{{$config->getPort()}}";

    return (bool) file_put_contents($self::$configFilePath, $configLine, FILE_APPEND);
}

public function createConfig($host, $user, $password, $port): bool {
    $id = count($this->configs) + 1;
    $config = new Config($id, $host, $user, $password, $port);

    $this->configs[] = $config;
    return $this->writeToConfigFile($config);
}
```

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.

## 2.5.5 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 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.

```
public function getTables($configId, $databaseId) {  
    $response = new Response();  
  
    $config = ConfigManager::getInstance()->getConfigById($configId);  
    $database = Database::getById($config, $databaseId);  
    $tables = Table::list($database);  
  
    $response->setStatus(Response::$STATUS_OK);  
    $response->setMessage('');  
    $response->setHttpStatusCode(Response::$HTTP_STATUS_OK);  
    $response->setData($tables);  
  
    $response->respond();  
}
```

Abbildung 17 - ConfigController getTables Methode

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

```
public static function getById(Config $config, int $databaseId): Database {
    $mysqliConn = MysqliDB::getInstance()->getConnection($config);

    $res = $mysqliConn->query("SHOW DATABASES;");

    $object = (object) [];
    while ($record = $res->fetch_assoc()) {
        $object = new Database($record['Database'], $config);
        if($object->getId() == $databaseId) {
            break;
        }
    }

    return $object;
}
```

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.

```
public function getConnection(Config $config, Database $database = NULL): mysqli {
    $foundConnection = NULL;

    foreach ($this->connections as $connection) {
        if ($connection['config']->getId() == $config->getId()) {
            $foundConnection = $connection['mysqli'];
            break;
        }
    }
    // if we're here, mysqli was not found, so lets add it and get the new connection
    $foundConnection = $this->addConnection($config);

    // if database is set we have to use it
    if (!is_null($database)) {
        $foundConnection->select_db($database->getName());
    }

    return $foundConnection;
}
```

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.

```
public static function list(Database $database): array {  
    $config = $database->getConfig();  
  
    $mysqliConn = MysqliDB::getInstance()->getConnection($config, $database);  
  
    $res = $mysqliConn->query("SHOW TABLES;");  
  
    $key = "Tables_in_{$database->getName()}";  
    $objects = [];  
    while ($record = $res->fetch_assoc()) {  
        $objects[] = new Table($record[$key], $database);  
    }  
  
    return $objects;  
}
```

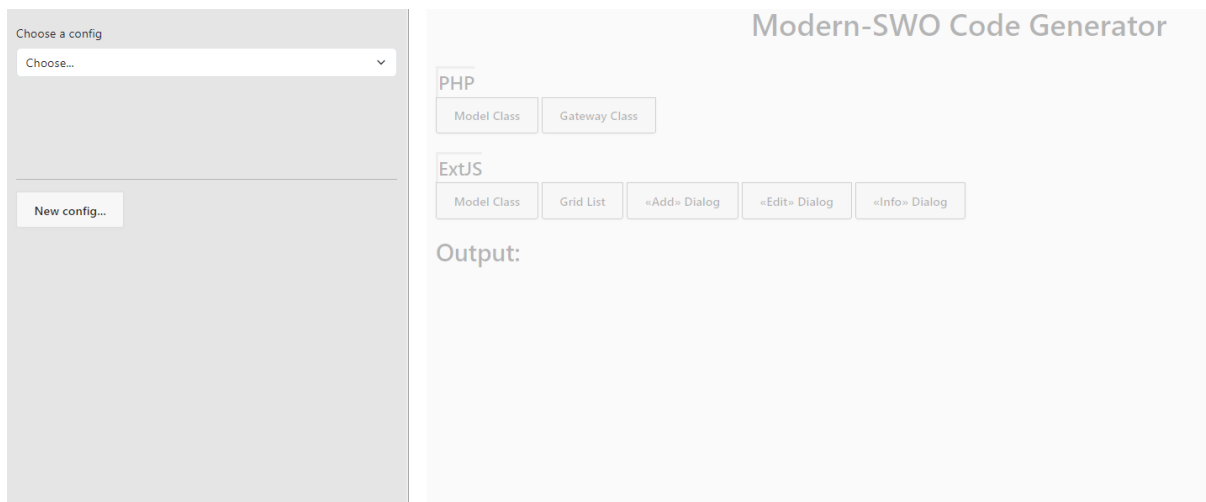
Wieder wird der MysqliDB Singleton verwendet, diesmal wird die Datenbank aber auch mitgegeben, damit diese gleich benutzt wird, und dann das SHOW TABLES Query auch funktioniert.

Im ConfigController werden die Tabellen dann ausgegeben.

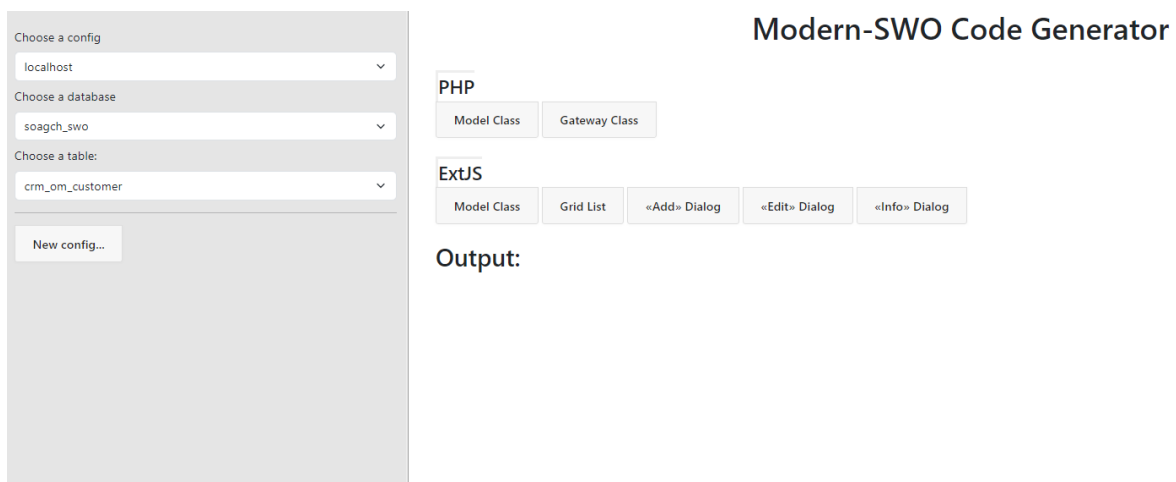
## 2.5.6 Snippet-Auswahl Frontend

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:



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



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:

### Modern-SWO Code Generator

PHP

Model ClassGateway Class

ExtJS

Model ClassGrid List«Add» Dialog«Edit» Dialog«Info» Dialog

Output:

php model

Copy

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:

```
$('#copyButton').on('click', (event) => {  
    let output = $('#content').html();  
    navigator.clipboard.writeText(output);  
});
```

Abbildung 18 – 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:

```
$('#copyButton').on('click', (event) => {  
    selectText('content');  
    document.execCommand('copy');  
    clearSelection();  
});
```

Abbildung 19 – Frontend copy-ClickEvent neu

Diese machen folgendes:

```
function selectText(containerid) {  
    if (document.selection) { // IE  
        var range = document.body.createTextRange();  
        range.moveToElementText(document.getElementById(containerid));  
        range.select();  
    } else if (window.getSelection) {  
        var range = document.createRange();  
        range.selectNode(document.getElementById(containerid));  
        window.getSelection().removeAllRanges();  
        window.getSelection().addRange(range);  
    }  
}  
  
function clearSelection() {  
    if (window.getSelection) {  
        window.getSelection().removeAllRanges();  
    } else if (document.selection) {  
        document.selection.empty();  
    }  
}
```

Abbildung 20 – Frontend copy-Funktionen

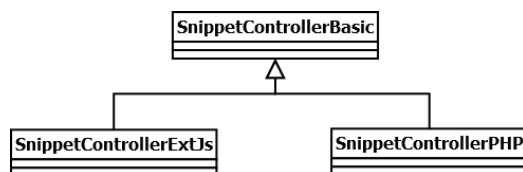
Nun funktioniert das Kopieren einwandfrei.

## 2.5.7 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.



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 Tabellename «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:

```

protected function convertToClassname(string $tableName): string {
    $arr = explode('_', $tableName);
    for ($i = 0; $i < count($arr); $i++) {
        if ($i == 0) {
            $arr[$i] = strtoupper($arr[$i]);
        } else if ($i == 1) {
            $arr[$i] = strtolower($arr[$i]);
        } else {
            $arr[$i] = ucfirst($arr[$i]);
        }
    }
    return implode('', $arr);
}
  
```

Abbildung 21 - 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.

```
protected function prepareFields(array $fields, array $filterOut = []): array {
    // filter out unwanted fields
    foreach ($fields as &$field) {
        if (in_array($field['COLUMN_NAME'], $filterOut)) {
            $key = array_search($field['COLUMN_NAME'], $fields);
            unset($fields[$key]);
        }
    }

    // get length of longest DATA_TYPE and COLUMN_NAME
    $types = array_map(fn($e) => $e['DATA_TYPE_DISPLAY'], $fields);
    $maxTypeLen = max(array_map('strlen', $types));

    $names = array_map(fn($e) => $e['COLUMN_NAME'], $fields);
    $maxNameLen = max(array_map('strlen', $names));

    // add new keys to subarrays of $fields for correct spacing later
    foreach ($fields as &$field) {
        $field['DATA_TYPE_SPACES'] = str_repeat('&nbsp;', $maxTypeLen - strlen($field['DATA_TYPE_DISPLAY']));
        $field['COLUMN_NAME_SPACES'] = str_repeat('&nbsp;', $maxNameLen - strlen($field['COLUMN_NAME']));
    }
    return $fields;
}
```

Abbildung 22 – 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:

```
protected function indent(int $level) {
    $indentSize = str_repeat('&nbsp;', self::$INDENT_SPACES);
    return str_repeat($indentSize, $level);
}
```

Abbildung 23 – 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:

```
// remove only the last occurrence of a substring in a string
protected function removeLastOccurrence(string $string, string $search) {
    $offset = strrpos($string, $search);
    if ($offset !== false) {
        $length = strlen($search);
        $string = substr_replace($string, '', $offset, $length);
    }
    return $string;
}
```


Abbildung 24 – SnippetControllerBasic `removeLastOccurrence`-Methode

### 2.5.7.1 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:



```

<?php
namespace Model;

use JsonSerializable;

class KLASSENNAME extends BaseModel implements JsonSerializable {

    private ?int      $id      = NULL;
    private string    $articleNo = '';
    private ?DateTime $validBegin = NULL;
    private ?DateTime $validEnd  = NULL;
    .....

    public function jsonSerialize(): array {
        return [
            'id'           => $this->id,
            'articleNo'    => $this->articleNo,
            'validBegin'   => $this->validBegin,
            'validEnd'     => $this->validEnd,
            .....
        ];
    }

    public static function createObject(array $data): ERParticleService {
        $obj->id           = isset($data['id']) ? $data['id'] : NULL;
        $obj->articleNo    = isset($data['ArticleNo']) ? $data['ArticleNo'] : '';
        $obj->validBegin   = isset($data['ValidBegin']) ? $data['ValidBegin'] : NULL;
        $obj->validEnd     = isset($data['ValidEnd']) ? $data['ValidEnd'] : NULL;
        .....
    }

    public function getId(): ?int {
        return $this->id;
    }

    public function setId(?int $id) {
        $this->id = $id;
    }

    .....
}
  
```

**Attribute**

**JSON-Serialize**

**Objekterstellung**

**Getters & Setters**

Abbildung 25 - PHP-Model-Klasse-Snippet Abschnitte

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

```
private static array $DEFAULT_VALUES = ['int' => 'NULL', 'float' => 'NULL', 'string' => '\\\\', 'DateTime' => 'NULL'];
```

Abbildung 26 - 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.

```
$header = $this->generateModelHeader($className);
$attributes = $this->generateAttributes($fields);
$jsonSerialize = $this->generateJsonSerialize($fields);
$createObjectMethod = $this->generateCreateObject($fields, $className);
$gettersAndSetters = $this->generateGettersAndSetters($fields);
$footer = $this->generateFooter();

$output = $header.
    $newline2.
    $attributes.
    $newline2.
    $jsonSerialize.
    $newline2.
    $createObjectMethod.
    $newline2.
    $gettersAndSetters.
    $newline2.
    $footer;
```

Abbildung 27 - SnippetControllerPHP Ausschnitt - Zusammensetzung der Klasse

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

```
private function generateModelHeader(string $className): string {
    $content = htmlspecialchars('<?php');
    $content .= '<br>';
    $content .= 'namespace ' . self::$NAMESPACE_MODEL . ';';
    $content .= '<br>';
    $content .= '<br>';
    $content .= 'use JsonSerializable;';
    $content .= '<br>';
    $content .= '<br>';
    $content .= "class $className extends " . self::$PARENT_MODEL . ' implements JsonSerializable {';
    return ($content);
}
```

Abbildung 28 - 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:

```
protected function prepareFields(array $fields, array $filterOut = []): array {
    // translate SQL data types to PHP
    $wholeNumberSQLTypes = ['int', 'integer', 'bigint', 'smallint', 'tinyint'];
    $decimalNumberSQLTypes = ['dec', 'decimal', 'float', 'double', 'double precision'];
    $stringSQLTypes = ['char', 'varchar', 'tinyblob', 'mediumblob', 'blob', 'longblob',
        'tinytext', 'mediumtext', 'text', 'longtext'];
    $dateTimeSQLTypes = ['date', 'datetime', 'timestamp', 'time', 'year'];

    foreach ($fields as &$field) {
        $type = strtolower($field['DATA_TYPE']);
        if (in_array($type, $wholeNumberSQLTypes)) {
            $field['DEFAULT_VALUE'] = self::$DEFAULT_VALUES['int'];
            if ($field['DEFAULT_VALUE'] == 'NULL') {
                $field['DATA_TYPE'] = 'int';
                $field['DATA_TYPE_DISPLAY'] = '?int';
            } else {
                $field['DATA_TYPE'] = 'int';
                $field['DATA_TYPE_DISPLAY'] = 'int';
            }
            ...
        } else {
            ErrorThrower::throw("Unknown type \"$type\"");
        }
    }

    return parent::prepareFields($fields, $filterOut);
}
```

Abbildung 29 – 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:

```
private function generateJsonSerialize(array $fields) {
    $content = "{$this->indent(1)}public function jsonSerialize(): array {<br>";
    $content .= "{$this->indent(2)}return [<br>";
    foreach ($fields as $field) {
        $lname = lcfirst($field['COLUMN_NAME']);
        $nameSpaces = $field['COLUMN_NAME_SPACES'];
        $content .= "{$this->indent(3)}'$lname'$nameSpaces => \\$this->$lname,<br>";
    }
    $content .= "{$this->indent(2)]]";
    $content .= "<br>";
    $content .= "{$this->indent(1)}}";

    $content = $this->removeLastOccurrence($content, ',');

    return $content;
}
```

Abbildung 30 – 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:

```
private function generateAttributes(array $fields): string {  
  
    $content = '';  
    foreach ($fields as $field) {  
  
        $typeSpaces = $field['DATA_TYPE_SPACES'];  
        $nameSpaces = $field['COLUMN_NAME_SPACES'];  
  
        $field['COLUMN_NAME'] = lcfirst($field['COLUMN_NAME']);  
  
        $content.="{$this->indent(1)}private {${field['DATA_TYPE_DISPLAY']}}$typeSpaces  
                \${$field['COLUMN_NAME']}$nameSpaces = {${field['DEFAULT_VALUE']}};<br>";  
    }  
    $content = $this->removeLastOccurrence($content, '<br>');  
    return $content;  
}
```

Abbildung 31 - 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:

## Output:

```
<?php
namespace Model;

use JsonSerializable;

class ERParticleServiceDummy extends BaseModel implements JsonSerializable {

    private ?int $id = NULL;
    private string $articleName = '';

    public function jsonSerialize(): array {
        return [
            'id' => $this->id,
            'articleName' => $this->articleName
        ]
    }

    public static function createObject(array $data): ERParticleServiceDummy {
        $obj->id = isset($data['Id']) ? $data['Id'] : NULL;
        $obj->articleName = isset($data['articleName']) ? $data['articleName'] : '';
    }

    public function getId(): ?int {
        return $this->id;
    }

    public function setId(?int $id) {
        $this->id = $id;
    }

    public function getArticleName(): string {
        return $this->articleName;
    }

    public function setArticleName(string $articleName) {
        $this->articleName = $articleName;
    }

}
```

Abbildung 32 – 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.

### 2.5.7.2 PHP-Gateway-Klasse Snippet

Die Gateway-Klasse ist die Schnittstelle zur Datenbank und regelt alle Abfragen.  
Ich werde folgende Abschnitte generieren:

```
1 <?php
2 namespace Gateway;
3
4 use mysqli;
5 use mysqli_result;
6 use Utils\DB;
7 use Model\ERParticleService;
8
9 class ERParticleServiceGateway extends BasicTableGateway {
10
11     public function __construct() {
12         parent::__construct();
13
14         $this->table = "erp_article_service";
15     }
16
17
18     public function insert(ERParticleService $obj): int {
19
20         $stmt = $this->conn->prepare("INSERT INTO erp_article_service (
21             StateCd,
22             ArticleNo,
23             updDate,
24             ....
25         ) VALUES (
26             ?,?, ....
27         );");
28
29         // bind requires references (get returns value)
30         $params = [
31             $obj->getStateCd(),
32             $obj->getArticleNo(),
33             $obj->formatDate($obj->getUpdDate())
34         ];
35
36         $stmt->bind_param('iss ....', ...$params);
37
38         $stmt->execute();
39
40         //echo($stmt->error);
41
42         return $this->conn->insert_id;
43     }
44 }
45
46 }
```

Konstruktor

Methode insert

Abbildung 33 – 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:

```
$fields = $this->prepareFields($table->getFields(), ['id']);
$tableName = $table->getName();
$className = $this->convertToClassname($tableName).'Gateway';
$newLine1 = '<br>';
$newLine2 = '<br><br>';

$header = $this->generateGatewayHeader($className, $tableName);
$insert = $this->generateInsert($fields, $className, $tableName);
$footer = $this->generateFooter();

$output = $header.
        $newLine2.
        $insert.
        $newLine2.
        $footer;
```

Abbildung 34 – 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:

```
private function generateInsert(array $fields, string $className, string $tablename) {

    $content = "{$this->indent(1)}public function insert($className \\$obj): int {<br><br>";
    $content .= "{$this->indent(2)}\\$stmt = \\$this->conn->prepare(\"INSERT INTO $tablename (<br>";

    foreach ($fields as $field) {
        $name = $field['COLUMN_NAME'];
        $content .= "{$this->indent(3)}$name,<br>";
    }

    $content = $this->removeLastOccurrence($content, ',');
    $content .= "{$this->indent(2)}) VALUES (<br>";
    $content .= "{$this->indent(3)}".str_repeat('?',',', count($fields))."<br>";
    $content = $this->removeLastOccurrence($content, ',');
    $content .= "{$this->indent(2)})\";";
    $content .= "<br><br>";
    $content .= "{$this->indent(2)}// bind requires references (get return value)<br>";
    $content .= "{$this->indent(2)}\\$params = [<br>";

    $bindTypes = '';
    foreach ($fields as $field) {
        $name = $field['COLUMN_NAME'];
        $uname = ucfirst($name);
        $dataType = $field['DATA_TYPE'];
        $bindTypes .= self::$BIND_TYPES[$dataType];
        if ($field['DATA_TYPE'] == 'DateTime') {
            $content .= "{$this->indent(3)}\\$obj->formatDate(\\$obj->get$uname()),<br>";
        } else {
            $content .= "{$this->indent(3)}\\$obj->get$uname(),<br>";
        }
    }

    $content = $this->removeLastOccurrence($content, ',');
    $content .= "{$this->indent(2)})\";";
    $content .= "<br><br>";
    $content .= "{$this->indent(2)}\\$stmt->bind_param('$bindTypes', ...\\$params);";
    $content .= "<br><br>";
    $content .= "{$this->indent(2)}\\$stmt->execute();";
    $content .= "<br><br>";
    $content .= "{$this->indent(2)}//echo(\\$stmt->error);";
    $content .= "<br><br>";
    $content .= "{$this->indent(2)}return \\$this->conn->insert_id;<br>";
    $content .= "{$this->indent(1)}}";

    return $content;
}
```

Abbildung 35 – SnippetControllerPHP Ausschnitt – generateInsert-Methode

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

## Output:

```
<?php
namespace Gateway;

class ERParticleServiceDummyGateway extends BasicGateway {

    public function __construct() {
        parent::__construct();
        $this->table = "erp_article_service_dummy";
    }

    public function insert(ERParticleServiceDummyGateway $obj): int {

        $stmt = $this->conn->prepare("INSERT INTO erp_article_service_dummy (
            Id,
            articleName
        ) VALUES (
            ?,?
        );");

        // bind requires references (get return value)
        $params = [
            $obj->getId(),
            $obj->getArticleName()
        ];

        $stmt->bind_param('is', ...$params);

        $stmt->execute();

        //echo($stmt->error);

        return $this->conn->insert_id;
    }

}
```

Abbildung 36 – PHP-Gateway Code-Snippet Resultat

### 2.5.7.3 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:

```

1  Ext.define('swo.model.ERParticleService', {
2      extend: 'Ext.data.Model',
3
4      fields: [
5          { name: 'stateCd',           type: 'int' },
6          { name: 'articleNo',        type: 'string' },
7          { name: 'updDate',          type: 'date' },
8      ]
9
10 });

```

Abbildung 37 – ExtJs-Model-Snippet

Zuerst wird der Header generiert:

```

private function generateModelHeader($className): string {

    $content = "Ext.define('swo.model.$className', {";
    $content .= "<br>";
    $content .= "{$this->indent(1)}extend: 'Ext.data.Model','";

    return $content;
}

```

Abbildung 38 – SnippetControllerExtJs Ausschnitt – generateModelHeader-Methode

Und dann das Mapping:

```

private function generateModelBody($fields): string {

    $content = "{$this->indent(1)}fields: [";
    $content .= "<br>";

    foreach ($fields as $field) {

        $name = $field['COLUMN_NAME'];
        $lname = lcfirst($name);
        $type = $field['DATA_TYPE'];

        $typeSpaces = $field['DATA_TYPE_SPACES'];
        $nameSpaces = $field['COLUMN_NAME_SPACES'];

        $content .= "{$this->indent(2)}{ name: '$lname',$nameSpaces type: '$type' },<br>";
    }
    $content = $this->removeLastOccurrence($content, ',');
    $content .= "{$this->indent(1)}]";
    return $content;
}

```

Abbildung 39 – SnippetControllerExtJs Ausschnitt – generateModelBody-Methode

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

```
public function generateModel($configId, $databaseId, $tableId) {
    $response = new Response();

    $config = ConfigManager::getInstance()->getConfigById($configId);
    $database = Database::getById($config, $databaseId);
    $table = Table::getById($database, $tableId);

    $fields = $this->prepareFields($table->getFields());
    $className = $this->convertToClassname($table->getName());
    $newline1 = '<br>';
    $newline2 = '<br><br>';

    $header = $this->generateModelHeader($className);
    $body = $this->generateModelBody($fields);
    $footer = $this->generateFooter();

    $output = $header.
        $newline2.
        $body.
        $newline2.
        $footer;

    $response->setStatus(Response::STATUS_OK);
    $response->setMessage('');
    $response->setHttpStatusCode(Response::HTTP_STATUS_OK);
    $response->setData($output);

    $response->respond();
}
```

Abbildung 40 – SnippetControllerExtJs Ausschnitt – generateModel-Methode

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

## Output:

```
Ext.define('swo.model.ERParticleServiceDummy', {
    extend: 'Ext.data.Model',

    fields: [
        { name: 'id',          type: 'int' },
        { name: 'articleName', type: 'string' }
    ]

});
```

Abbildung 41 – ExtJs-Model Code-Snippet Resultat

### 2.5.7.4 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:

```

1  Ext.define('swo.view.erparticlesservice.List', {
2      extend: 'Ext.grid.Panel',
3
4      alias: 'widget.erparticlesservice_list',
5      id: 'erparticlesservice_list',
6
7
8      requires:[
9          'swo.store.ERParticleService', 'swo.controller.ERParticleService', // store+controller
10         'swo.utils.GlobalFunctions', // functions
11         'swo.view.erparticlesservice.dialogs.CreateModal', 'swo.view.erparticlesservice.dialogs.EditModal',
12         'swo.store.CRMsdCode.SupportPackManufacturers', // combobox store
13         'swo.view.erparticlesservice.Toolbar', // Toolbar
14     ],
15
16     store: {type: 'ERParticleService'},
17     controller: 'ERParticleService',
18
19     layout: 'fit',
20     title: 'Service Articles',
21
22     // tbar
23     dockedItems: [{
24         xtype: 'ERParticleServiceToolbar',
25         dock: 'top'
26     }],
27

```

Abbildung 42 – ExtJs-Grid-List-Snippet Header

Danach kommen alle Spalten der Liste:

```

28     columns: [
29         {
30             text: 'Id',
31             dataIndex: 'id',
32             hidden: true,
33             flex: 1,
34         },
35         {
36             text: 'State',
37             dataIndex: 'stateCd',
38             hidden: true,
39             flex: 1,
40             scope: swo.utils.GlobalFunctions,
41             renderer: 'renderStateGrid'
42         },
43         {
44             text: 'Article Nr.',
45             dataIndex: 'articleNo',
46             hidden: false,
47             flex: 1,
48         },
49         {
50             text: 'Update date',
51             dataIndex: 'updDate',
52             hidden: false,
53             flex: 1,
54             format: 'm.d.Y',
55             scope: swo.utils.GlobalFunctions,
56             renderer: 'renderDateTimeGrid',
57         }
58     ],

```

Abbildung 43 – ExtJs-Grid-List-Snippet Columns

Und am Ende noch eine Toolbar und ein Listener:

```

59
60     bbar: {
61         xtype: 'pagingtoolbar',
62         displayInfo: true,
63         beforePageText: 'Seite',
64         afterPageText: 'von {0}',
65         listeners: {
66             change: 'onPageEvent'
67         }
68     },
69
70     listeners: {
71         select: 'onItemSelected',
72         itemdblclick: 'onDoubleClick',
73     }
74

```

Abbildung 44 – 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:

```

private function generateGridListHeader($className, $packageName) {

    $content = "Ext.define('swo.view.$packageName.List', {";
    $content .= "<br>";
    $content .= "{$this->indent(1)}extend: 'Ext.grid.Panel',";
    $content .= "<br>";
    $content .= "{$this->indent(1)}alias: 'widget.{$packageName}_list',";
    $content .= "<br>";
    $content .= "{$this->indent(1)}id: '{$packageName}_list'";
    $content .= "<br>";
    $content .= "{$this->indent(1)}requires: [";
    $content .= "<br>";
    $content .= "{$this->indent(2)}'swo.store.$className', 'swo.controller.$className', // store+controller";
    $content .= "<br>";
    $content .= "{$this->indent(2)}'swo.utils.GlobalFunctions', // functions";
    $content .= "<br>";
    $content .= "{$this->indent(2)}'swo.view.$packageName.dialogs.CreateModal', // dialogs";
    $content .= "<br>";
    $content .= "{$this->indent(2)}'swo.view.$packageName.dialogs.EditModal',";
    $content .= "<br>";
    $content .= "{$this->indent(2)}'swo.view.$packageName.dialogs.DeleteDialog',";
    $content .= "<br>";
    $content .= "{$this->indent(2)}// 'combobox stores, toolbar, etc.'";
    $content .= "<br>";
    $content .= "{$this->indent(1)}],";
}

```

Abbildung 45 – SnippetControllerExtJs Ausschnitt – generateGridListHeader-Methode Ausschnitt



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

```
private function generateGridListBody($fields, $className) {
    $content = "{$this->indent(1)}columns: [";
    $content .= "<br>";

    foreach ($fields as $field) {
        $name = $field['COLUMN_NAME'];
        $uname = ucfirst($name);
        $lname = lcfirst($name);
        $type = $field['DATA_TYPE'];

        $typeSpaces = $field['DATA_TYPE_SPACES'];
        $nameSpaces = $field['COLUMN_NAME_SPACES'];

        $content .= "{$this->indent(2)}{";
        $content .= "<br>";
        $content .= "{$this->indent(3)}text: '$uname'";
        $content .= "<br>";
        $content .= "{$this->indent(3)}dataIndex: '$lname'";
        $content .= "<br>";
        $content .= "{$this->indent(3)}hidden: false";
        $content .= "<br>";
        $content .= "{$this->indent(3)}flex: 1,";
        $content .= "<br>";
        $content .= "{$this->indent(3)}//scope: swo.utils.GlobalFunctions,";
        $content .= "<br>";
        $content .= "{$this->indent(3)}//renderer: 'render{$name}Grid'";
        $content .= "<br>";
        $content .= "{$this->indent(2)}},";
        $content .= "<br>";
    }
}
```

Abbildung 46 – SnippetControllerExtJs Ausschnitt – generateGridListBody-Methode Ausschnitt 1

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

```
$content = $this->removeLastOccurrence($content, ',');

$content .= "{$this->indent(1)},";
$content .= "<br>";
$content .= "<br>";
$content .= "{$this->indent(1)}bbar: {";
$content .= "<br>";
$content .= "{$this->indent(2)}xtype: 'pagingtoolbar'";
$content .= "<br>";
$content .= "{$this->indent(2)}displayInfo: true";
$content .= "<br>";
$content .= "{$this->indent(2)}beforePageText: 'Seite'";
$content .= "<br>";
$content .= "{$this->indent(2)}afterPageText: 'von {0}'";
$content .= "<br>";
$content .= "{$this->indent(2)}listeners: {";
$content .= "<br>";
$content .= "{$this->indent(3)}change: 'onPageEvent'";
$content .= "<br>";
$content .= "{$this->indent(2)}}";
$content .= "<br>";
$content .= "{$this->indent(1)},";
$content .= "<br>";
$content .= "<br>";
$content .= "{$this->indent(1)}listeners: {";
$content .= "<br>";
$content .= "{$this->indent(2)}select: 'onItemSelected'";
$content .= "<br>";
$content .= "{$this->indent(2)}itemdblclick: 'onDoubleClick'";
$content .= "<br>";
$content .= "{$this->indent(1)}}";

return $content;
}
```

Abbildung 47 – 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:

```
columns: [
  {
    text: 'Id'
    dataIndex: 'id'
    hidden: false
    flex: 1,
    //scope: swo.utils.GlobalFunctions,
    //renderer: 'renderIdGrid'
  },
  {
    text: 'ArticleName'
    dataIndex: 'articleName'
    hidden: false
    flex: 1,
    //scope: swo.utils.GlobalFunctions,
    //renderer: 'renderarticleNameGrid'
  }
],
bbar: {
  xtype: 'pagingtoolbar',
  displayInfo: true
  beforePageText: 'Seite',
  afterPageText: 'von {0}',
  listeners: {
    change: 'onPageEvent'
  }
},
listeners: {
  select: 'onItemSelected',
  itemdblclick: 'onDoubleClick'
}
});
```

Abbildung 48 – ExtJs-GridList Code-Snippet Resultat

### 2.5.7.5 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:

```
public function generateAddDialog($configId, $databaseId, $tableId) {
    $response = new Response();

    $config = ConfigManager::getInstance()->getConfigById($configId);
    $database = Database::getId($config, $databaseId);
    $table = Table::getId($database, $tableId);

    $fields = $this->prepareFields($table->getFields());
    $className = $this->convertToClassname($table->getName());
    $packageName = strtolower($className);
    $newLine1 = '<br>';
    $newLine2 = '<br><br>';

    $header = $this->generateDialogHeader($className, $packageName, 'Add');
    $body = $this->generateDialogBody($fields, $className, 'POST');
    $footer = $this->generateFooter();

    $output = $header.
        $newLine2.
        $body.
        $newLine2.
        $footer;

    $response->setStatus(Response::STATUS_OK);
    $response->setMessage('');
    $response->setHttpStatusCode(Response::HTTP_STATUS_OK);
    $response->setData($output);

    $response->respond();
}
```

Abbildung 49 – SnippetControllerExtJs Ausschnitt – generateAddDialog-Methode

Der Header wird zum Beispiel zu generiert:

```
private function generateDialogHeader($className, $packageName, $dialogType): string {
    $content = "Ext.define('swo.view.$packageName.dialogs.{ $dialogType }Modal', {";
    $content .= "<br>";
    $content .= "{$this->indent(1)}extend: 'Ext.window.Window',";
    $content .= "<br>";
    $content .= "{$this->indent(1)}requires: ['Ext.form.Panel', 'swo.controller.$className'],";
    $content .= "<br>";
    $content .= "{$this->indent(1)}controller: '$className',";
    $content .= "<br>";
    $content .= "{$this->indent(1)}title: '<b>$dialogType Service Article<b>',";
    $content .= "<br>";
    $content .= "{$this->indent(1)}modal: true,";
    $content .= "<br>";
    $content .= "{$this->indent(1)}//alwaysOnTop: true,";
    $content .= "<br>";
    $content .= "{$this->indent(1)}height: 600,";
    $content .= "<br>";
    $content .= "{$this->indent(1)}width: 500,";
    $content .= "<br>";
    $content .= "{$this->indent(1)}overflowY: auto,";
    $content .= "<br>";
    $content .= "};";

    return $content;
}
```

Abbildung 50 – 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:

## Output:

```
Ext.define('swo.view.erparticleservicedummy.dialogs.AddModal', {
    extend: 'Ext.window.Window',

    requires: ['Ext.form.Panel', 'swo.controller.ERParticleServiceDummy'],
    controller: 'ERParticleServiceDummy',

    title: 'Add Service Article',

    modal: true,
    //alwaysOnTop: true,
    height: 600,
    width: 500,
    overflowY: auto,

    items: {
        xtype: 'form',
        defaultType: 'textfield',

        defaults: {
            size: 38, // deprecated, but the only way it works... width, minWidth, etc. do not...
            labelAlign: 'left',
            labelWidth: 160,
        },
        bodyPadding: 10,

        url: swo.utils.Environment.getRemoteDomain().concat('ERParticleServiceDummy'),
        method: 'POST',
        jsonSubmit: true,
```

Abbildung 51 - ExtJs-Dialog Code-Snippet Resultat

## 2.6 Kontrollieren

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

Die Tests werden manuell durchgeführt.

### 2.6.1 Testfälle

ID	T_001
Beschreibung	Eine neue Config über das Formular hinterlegen.
Testvoraussetzung	Keine.
Testschritte	<ol style="list-style-type: none"><li>1. «New config» Button drücken</li><li>2. Formular mit Daten befüllen</li><li>3. Formular mittels «Add» Button abschicken</li></ol>
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	<ol style="list-style-type: none"><li>1. Auf der Selectbox mit Label «Choose a config» eine Config auswählen</li></ol>
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

## 2.6.2 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

## 2.7 Auswerten

### 2.7.1 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.

## 2.7.2 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.

## 2.8 Verzeichnisse

### 2.8.1 Abbildungsverzeichnis

Abbildung 1 - Use-Case-Diagramm.....	24
Abbildung 2 - MockUp 1 .....	26
Abbildung 3 - MockUp 2.....	27
Abbildung 4 - MockUp Variante 2 .....	29
Abbildung 5 - MockUp Variante 1.....	29
Abbildung 6 - Mockup Entscheidungsmatrix .....	29
Abbildung 7 - Selectboxen .....	30
Abbildung 8 - Config-Formular .....	31
Abbildung 9 - Grundstruktur der Applikation.....	32
Abbildung 10 - Backend router Ausschnitt.....	32
Abbildung 11 - Controller-Klasse Ausgabe .....	33
Abbildung 12 - Response-Klasse Ausschnitt .....	33
Abbildung 13 - Utils-Klasse ErrorThrower.....	33
Abbildung 14 - ConfigController getConfigs Funktion.....	34
Abbildung 15 - ConfigController createConfig Methode.....	35
Abbildung 16 - ConfigManager Dateibearbeitung .....	35
Abbildung 17 - ConfigController getTables Methode .....	37
Abbildung 18 - Frontend copy-ClickEvent alt .....	41
Abbildung 19 - Frontend copy-ClickEvent neu .....	42
Abbildung 20 - Frontend copy-Funktionen.....	42
Abbildung 21 - SnippetControllerBasic convertToClassname-Methode .....	43
Abbildung 22 - SnippetControllerBasic prepareFields Methode .....	44
Abbildung 23 - SnippetControllerBasic indent-Methode.....	44
Abbildung 24 - SnippetControllerBasic removeLastOccurrence-Methode.....	44
Abbildung 25 - PHP-Model-Klasse-Snippet Abschnitte.....	45
Abbildung 26 - SnippetControllerPHP DEFAULT_VALUES.....	46
Abbildung 27 - SnippetControllerPHP Ausschnitt - Zusammensetzung der Klasse..	46
Abbildung 28 - SnippetControllerPHP Ausschnitt - generateModelHeader-Methode .....	46
Abbildung 29 - SnippetControllerPHP Ausschnitt - prepareFields-Methode .....	47
Abbildung 30 - SnippetControllerPHP Ausschnitt - generateJsonSerialize-Methode .....	47
Abbildung 31 - SnippetControllerPHP Ausschnitt - generateAttributes-Methode .....	48
Abbildung 32 - PHP-Klasse Code-Snippet Resultat.....	49
Abbildung 33 - PHP-Gateway-Klasse-Snippet Abschnitte.....	50
Abbildung 34 - SnippetControllerPHP Ausschnitt - Zusammensetzung des Gateways .....	51
Abbildung 35 - SnippetControllerPHP Ausschnitt - generateInsert-Methode .....	51

Abbildung 36 – PHP-Gateway Code-Snippet Resultat .....	52
Abbildung 37 – ExtJs-Model-Snippet .....	53
Abbildung 38 – SnippetControllerExtJs Ausschnitt – generateModelHeader-Methode.....	53
Abbildung 39 – SnippetControllerExtJs Ausschnitt – generateModelBody-Methode .....	53
Abbildung 40 – SnippetControllerExtJs Ausschnitt – generateModel-Methode.....	54
Abbildung 41 – ExtJs-Model Code-Snippet Resultat .....	54
Abbildung 42 – ExtJs-Grid-List-Snippet Header.....	55
Abbildung 43 – ExtJs-Grid-List-Snippet Columns .....	55
Abbildung 44 – ExtJs-Grid-List-Snippet Toolbar & Listener.....	56
Abbildung 45 – SnippetControllerExtJs Ausschnitt – generateGridListHeader-Methode Ausschnitt .....	56
Abbildung 46 – SnippetControllerExtJs Ausschnitt – generateGridListBody-Methode Ausschnitt 1.....	57
Abbildung 47 – SnippetControllerExtJs Ausschnitt – generateGridListBody-Methode Ausschnitt 2.....	57
Abbildung 48 – ExtJs-GridList Code-Snippet Resultat .....	58
Abbildung 49 – SnippetControllerExtJs Ausschnitt – generateAddDialog-Methode.....	59
Abbildung 50 – SnippetControllerExtJs Ausschnitt – generateDialogHeader-Methode.....	59
Abbildung 51 – ExtJs-Dialog Code-Snippet Resultat .....	60

## 2.8.2 Tabellenverzeichnis

Tabelle 1 – Dokumentenhistorie.....	4
Tabelle 2 – Dokumenteneigenschaften.....	4
Tabelle 3 – Code-Snippet Arten .....	6
Tabelle 4 – Zeitplan Zusammenfassung.....	21
Tabelle 5 – Use-Case Beschreibung «add new config».....	25
Tabelle 6 – Use-Case Beschreibung «choose database table» .....	25
Tabelle 7 – Use-Case Beschreibung «generate Code-Snippet» .....	25
Tabelle 8 – Testfall 1.....	61
Tabelle 9 – Testfall 2 .....	61
Tabelle 10 – Testfall 3.....	62
Tabelle 11 – Testfall 4.....	62
Tabelle 12 – Testfall 5.....	62
Tabelle 13 – Testfall 6 .....	63
Tabelle 14 – Testfall 7.....	63
Tabelle 15 – Testfall 8 .....	63
Tabelle 16 – Testfall 9 .....	64
Tabelle 17 – Testfall 10 .....	64



Tabelle 18 – Testfall 11 .....	64
Tabelle 19 – Testfall 12 .....	65
Tabelle 20 – Testfall 13 .....	65
Tabelle 21 – Testfall 14 .....	65
Tabelle 22 – Testfall 15 .....	66
Tabelle 23 – Testfall 16 .....	66
Tabelle 24 – Testfall 17 .....	66
Tabelle 25 – Testfall 18 .....	67
Tabelle 26 – Testfall 19 .....	67
Tabelle 27 – Glossar .....	73

## 2.8.3 Links

[https://www.ict-berufsbildung-bern.ch/resources/lperka\\_OdA\\_200617.pdf](https://www.ict-berufsbildung-bern.ch/resources/lperka_OdA_200617.pdf)

Website der ICT-Berufsbildung Bern

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

## 2.8.4 Glossar / Abkürzungen

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

Tabelle 27 – Glossar

## 2.9 Anhang

Besprechungsprotokolle

Quellcode

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