HTL - Perg
Höhere Abteilung für Informatik

Diplomarbeit

BLH-Templates

Projektteam: Markus Prandstätter
Gregor Haderer

Projektbetreuer: Prof. OStR Mag. Gabriela Danner

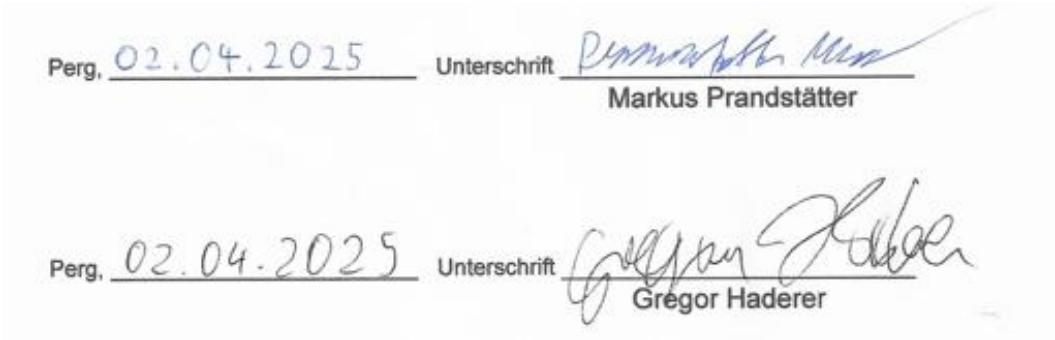
In Zusammenarbeit mit Kon-Cept Management Information Services GmbH
Betreuer Herr Philipp Starzer

Bearbeitungszeitraum: 01.10.2024. – 04.04.2025.

HTL Perg | Machlandstr. 48 | A-4320 Perg | Tel.: +43 (0) 7262 53926

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.
Diese Versicherung umfasst auch in der Arbeit verwendete bildliche Darstellungen, Tabellen, Skizzen und Zeichnungen.



Signiert von:	Markus Prandstätter
Datum:	03.04.2025 22:58:53

TRUST

Dieses Dokument ist digital signiert!

Dieses mit einer qualifizierten elektronischen Signatur versehene Dokument hat gemäß Art. 25 Abs. 2 der Verordnung (EU) Nr 910/2014 vom 23. Juli 2014 ("eIDAS-VO") die gleiche Rechtswirkung wie ein handschriftlich unterschriebenes Dokument.

Prüfinformation: Informationen zur Prüfung der elektronischen Signatur finden Sie unter: www.a-trust.at/pdf

Signiert von:	Gregor Haderer
Datum:	03.04.2025 23:03:39

TRUST

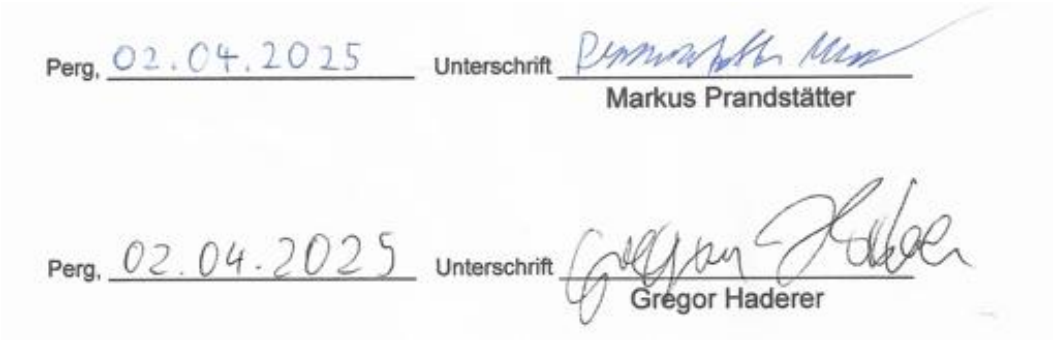
Dieses Dokument ist digital signiert!

Dieses mit einer qualifizierten elektronischen Signatur versehene Dokument hat gemäß Art. 25 Abs. 2 der Verordnung (EU) Nr 910/2014 vom 23. Juli 2014 ("eIDAS-VO") die gleiche Rechtswirkung wie ein handschriftlich unterschriebenes Dokument.

Prüfinformation: Informationen zur Prüfung der elektronischen Signatur finden Sie unter: www.a-trust.at/pdf

Gendererklärung

Im Sinne der besseren Lesbarkeit werden in dieser Diplomarbeit personenbezogene Bezeichnungen, die sich zugleich auf Frauen und Männer beziehen, generell nur in der im Deutschen üblichen maskulinen Form angeführt. Dies soll jedoch keinesfalls eine Geschlechterdiskriminierung oder eine Verletzung des Gleichheitsgrundsatzes zum Ausdruck bringen.



Danksagung

Unser Dank gilt allen Personen der HTL Perg und der Kon-Cept Management Information Services GmbH, die diese Diplomarbeit ermöglicht haben.

Frau Prof. OStR Mag. Gabriela Danner gebührt ein spezieller Dank. Sie half uns sowohl bei den theoretischen als auch bei den praktischen Aspekten der Arbeit. Zudem unterstützte sie uns bei der Auswahl des Vorgehensmodells und dessen Realisierung.

Wir möchten uns auch bei unserem Betreuer, Herrn Philipp Starzer von der Firma Kon-Cept Management Information Services GmbH, bedanken. Er half uns während dieser Kooperation bei der Entwicklung der Arbeit.

Zusammenfassung

Diese Arbeit zielt darauf ab, eine Lösung für die Firma Koncept zu finden, um beim Erstellen neuer Programme das Kopieren von altem Code in die neuen Programme zu verhindern. Um das Ergebnis nicht selber warten zu müssen, war das Verwenden einer bereits bestehenden Technologie Voraussetzung. Über diese Technologie soll beim Erstellen neuer Programme die Anzahl der Klassen auswählbar sein. Jede dieser Klassen muss wieder individuell konfigurierbar sein.

Ein Großteil unserer Arbeit war das Eruieren von passenden Technologien. Im Rahmen der Arbeit wurde Vieles getestet und Koncept vorgestellt um die richtige Technologie zu finden, welche den Ansprüchen entsprach.

Sowie die passende Technologie gefunden wurde, wurde auch gleich ein Prototyp erstellt. Daran wurde immer weitergearbeitet. Seitens unseres Betreuers der Firma, wurden reichlich Verbesserungsvorschläge und auch einige Ideen für Erweiterungen eingeworfen. Der Prototyp wurde so lange erweitert, bis es keine weiteren Vorschläge mehr gab und somit das Endprodukt entstanden war.

Das Endprodukt kann mit dem Programmierer per Kommandozeile interagieren und aufgrund der Eingaben oder Auswahlmöglichkeiten dynamisch ein neues Projekt erstellen, in welches man keinen alten Code kopieren muss. Zusätzlich zu der Kommandozeile wurde auch ein grafisches Benutzerfenster erstellt.

Da die ausgewählte Technologie keine Möglichkeit bietet, grafische Benutzeroberflächen zu erstellen, wurde diese mit Csharp erstellt.

Abstract

This work aims to find a solution for the company Koncept to prevent the copying of old code into new programs when creating them. To avoid having to maintain the result manually, the use of an already existing technology was a prerequisite. This technology should allow the selection of the number of classes when creating new programs. Each of these classes must be individually configurable.

A large part of our work involved evaluating suitable technologies. Throughout the project, many technologies were tested and presented to Koncept to find the right one that met the requirements.

As soon as the appropriate technology was found, a prototype was created. This prototype was continuously improved. Our company supervisor provided numerous suggestions for improvements as well as some ideas for extensions. The prototype was expanded until no further suggestions were made, resulting in the final product.

The final product allows the programmer to interact via the command line and dynamically create a new project based on inputs or selection options, ensuring that no old code needs to be copied. In addition to the command line interface, a graphical user interface was also developed.

Since the selected technology does not provide the capability to create graphical user interfaces, this was implemented using Csharp.

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangslage

In der Firma Koncept werden täglich neue Projekte erstellt. Die meisten dieser Projekte gehören zum firmeninternen Produkt "BLH"(Business Logic Host). Da diese Projekte immer einen ähnlichen Grundaufbau haben, kopieren viele Mitarbeiter Programmcode aus älteren Projekten. Jedoch entstehen oft Fehler bei diesem Kopiervorgang. Einerseits werden Programmteile überflüssig eingefügt oder vergessen, was zur Ressourcenverschwendung führt. Andererseits kann es passieren, dass kopierter Code nicht vollständig oder an falschen Stellen verwendet wird. Diese Faktoren führen zu einer zunehmenden Inkonsistenz der Projekte. Zusätzlich kann sich das Problem mit den Jahren verschlimmern. Wenn immer wieder Code aus älteren Programmen in neue kopiert wird, entsteht eine Kettenreaktion. Entwickler übernehmen Codefragmente aus bestehenden Projekten, ohne genau zu wissen, welcher Teil wofür zuständig oder verantwortlich ist. Zusätzlich hinterfragen sie nicht, ob der Code den aktuellen Formen entspricht. Auch die Qualitätssicherheit wird deutlich erschwert.

Ein weiteres Problem, das sich aus der Praxis ergibt, ist die fehlende Struktur der Projekte. Da immer Code aus älteren Programmen kopiert wird, verändert sich die Struktur ständig. Es gibt keine Standardstruktur. Dadurch leidet die Qualität und auch die Lesbarkeit des Codes stark. Aus diesem Grund müssen Teams oder neue Entwickler viel Zeit und Aufwand in die Einarbeitung von Projekten stecken. Darüber hinaus leidet auch die Skalierbarkeit der Projekte. Durch das unkontrollierte Kopieren wird es schwieriger, neue Funktionen zu integrieren oder bestehenden Code zu aktualisieren. Noch ein Problem von älteren Codes ist, dass dieser oft nicht mit neuen Ansätzen oder Technologien kompatibel ist, weshalb man beim Kopieren Fehler bekommt.

1.2 BLH

Zitat:

> ”MMS ist die Abkürzung für ”Manufacturing Management System”. MMS ist eine Lösung entwickelt von der Kon-Cept Gmbh, welche es ermöglicht, alle Informationsprozesse am Shop-Floor in elektronischer Form zu implementieren. MMS wurde speziell für den Einsatz in der Automobilindustrie entwickelt, wobei der Schwerpunkt auf Linienfertigung mit hoher Variantenvielfalt liegt, also z.B. Fahrzeugendmontage und Endausfertigung, Karosserierohbau und Vormontagen. MMS als Shop-Floor IT-Lösung ist eine Drehscheibe für den Datenaustausch in Echtzeit zwischen IT-Systemen und Anlagen am Shop-Floor. Eine essentielle Komponente ist dabei der sogenannte Business Logik Host (BLH). Wie der Name verrät, ist er dafür zuständig, jegliche Art von Business Logik bereitzustellen. Dies erfolgt verteilt via sogenannten Transaktionsmodulen (TMs) und ermöglicht so einen redundanzfreien und hoch-skalierbaren Systemaufbau.
>
> Es steht eine Vielzahl an vordefinierten Transaktionsmodulen bereit mit denen Laufzeitdaten empfangen, verarbeitet, geloggt, visualisiert und/oder an Fremdsystem weitergeleitet werden können. Zudem kann der Umfang stets um neue Logiken und Datenschnittstellen erweitert werden, sodass sich das System nahtlos in jede IT-Systemlandschaft einfügt.”
>
> -Herr Philipp Starzer - Kon-Cept Management Information Services GmbH

1.3 Projektumfeld

Diese Diplomarbeit entstand im Rahmen eines einmonatigen Praktikums in enger Zusammenarbeit mit der Firma Koncept. Während dieser Zeit wurde Hand in Hand mit dem Unternehmen gearbeitet, um die oben bereits erwähnten Probleme zu analysieren und um eine bestmögliche Lösung zu entwickeln.

In der direkten Zusammenarbeit mit den Mitarbeitern von Koncept wurde das Problemfeld intensiv analysiert und dokumentiert. Durch Gespräche mit Mitarbeitern wurde es möglich, an einer idealen Lösung zu arbeiten. Dieser enge Austausch zwischen Praktikanten und Unternehmen legte den Grundstein der Diplomarbeit.

Die Auswahl der verwendeten Technologien lag vollständig in der Verantwortung der Diplomanden. Verschiedene Technologien mussten ausgetestet und miteinander verglichen werden. Ziel war

es, Technologien hinsichtlich ihrer Stärken und Schwächen mit anderen zu vergleichen, um am Ende die bestmögliche Lösung zu entwickeln und dabei die effizienteste Technologie zu nützen. Bei diesem Testvorgang wurden alle Tests dokumentiert. Aber auch die Technologien wurden ausreichend getestet und schließlich mit Vor- und Nachteilen niedergeschrieben, damit für die Mitarbeiter von Koncept nachvollziehbar ist, wieso bestimmte Technologien nicht verwendet wurden.

1.4 Zielsetzung

Die Zielsetzung besteht darin, die immer wieder auftretenden Probleme bei der Entwicklung von BLH-Projekten zu lösen. Ein Hauptziel ist es, die Qualität und die Konsistenz dieser Projekte zu steigern. Auch soll der Code für die Entwickler besser lesbar werden.

Die Entwicklung und Implementierung von Templates ist ein entscheidender Bestandteil, um diese Ziele zu erreichen. Die Templates sind eine grundlegende Voraussetzung für die Entwicklung eines BLH-Projekts. Ihr Design bietet eine eindeutige und einheitliche Ausgangsbasis. So wird sichergestellt, dass die Projekte von guter Qualität sind und sich gut lesen lassen. Zudem verringern Templates das Risiko von Fehlern, indem sie unkontrolliertes Kopieren von Programmcode aus bestehenden Projekten verhindern.

Ein weiteres Ziel besteht darin, die langfristige Skalierbarkeit der Projekte zu erhöhen und fortlaufend zu optimieren. Es soll dabei verhindert werden, dass in Projekten widersprüchlicher oder überflüssiger Code kopiert wird. Anstelle dessen soll eine einheitliche Basis geschaffen werden, um zukünftige Erweiterungen und Optimierungen des Codes bestmöglich zu unterstützen und zu fördern.

Des Weiteren soll es allen Entwicklern möglich sein, jeden Projektcode zu verstehen. Dabei soll die Einarbeitungszeit auf ein Minimum reduziert werden. Templates geben eine eindeutige Anweisung, wo welcher Code platziert werden soll und wo nicht. Ebenfalls wird ein gemeinsamer Nenner für sämtliche Projekte erschaffen.

Letzten Endes soll die Projektarbeit so gestaltet werden, dass Zeit und Ressourcen gespart werden. In der Tat ist es auch Ziel, dass es nicht zu ineffizienten oder fehlerhaften Arbeitsprozessen kommt. Die Entwicklung neuer Funktionen und Systemen soll mit der eingesparten Zeit und Energie vorangetrieben werden. Die Vorlagen ermöglichen eine zügige und effiziente Projektentwicklung, ohne dass dabei die hohe Qualität in Gefahr gerät.

1.5 Projektinhalt

Der Inhalt des Projekts umfasst mehrere aufeinander aufbauende Phasen. Das Ziel dieser Phasen ist es, eine nachhaltige und praktische Lösung zu entwickeln.

Im ersten Schritt ist es wichtig, verschiedene Technologien und Werkzeuge zu finden und ihre Eigenschaften zu dokumentieren und analysieren. Dabei werden die jeweiligen Vorteile, aber auch die Nachteile bewertet. Die Anforderungen und Voraussetzungen der Firma Koncept müssen dabei im Auge behalten werden. Einerseits sollten die Technologien die Einfachheit und die Effizienz beziehungsweise auch die Qualität der Templates unterstützen, andererseits durften die Technologien nicht veraltet sein.

Nach der Auswahl der geeignetsten Technologien begann die zweite Phase. In diesem Schritt startete die Entwicklung der Templates. Durch gute und enge Zusammenarbeit mit den Mitarbeitern wurden die Probleme der BLH-Projektentwicklung identifiziert und in den Templates integriert. Dabei wurden viele wiederkehrende Muster und Anforderungen von Konzept berücksichtigt, um am Ende eine praxisorientierte Lösung zu entwickeln.

In den letzten Phasen wurden die entwickelten Templates auf Effizient und Brauchbarkeit getestet. Die Tests wurden oft von Mitarbeitern begleitet, um die Benutzerfreundlichkeit, aber auch die Qualität der Templates zu prüfen. Durch diese Rückmeldungen und Absprachen mit Mitarbeitern der Firma Koncept konnten wesentliche Verbesserungs- und Optimierungsvorschläge in die Entwicklung mit einbezogen werden. Es lässt sich nicht leugnen, dass das Ziel war, ein Werkzeug zu schaffen, das gerne und oft von Programmierern genützt wird um Projekte zu erstellen.

1.6 Phasenmodell

Um die Inhalte dieser Diplomarbeit klarer und nachvollziehbarer darzustellen, werden die nächsten vier Kapitel in einem Phasenmodell beschrieben.

In der Umsetzung startete jede Phase mit einem Meeting, in welchem besprochen wurde, was die nächsten Schritte waren und welche Technologien noch interessant waren. Beendet wurden Phasen auch mit einem Meeting, in welchem noch einmal besprochen wurde, zu welchen Ergebnissen die Diplomanden gekommen waren.

Jede Phase ist ein essenzieller und wichtiger Bestandteil dieser Arbeit. Durch diese Darstellung ist es leichter, die Zusammenhänge zwischen den Technologien besser zu verstehen. In jeder Phase werden Ideen, Überlegungen und Entscheidungen detailliert beschrieben.

Ziel dieser Struktur ist es, die Herangehensweise Schritt für Schritt anzugehen. Dabei werden nicht nur die Ergebnisse jeder Phase präsentiert, sondern auch die Überlegungen, wie man zu diesem Ergebnis gekommen ist. Darüber hinaus werden Vor- und Nachteile jeder Technologie beschrieben. Die Entscheidungsprozesse werden somit noch klarer. Jede Technologie und jedes Werkzeug wird nicht nur hinsichtlich ihrer theoretischen Eigenschaften bewertet, sondern auch über ihren praktischen Einsatz in der Firma, wodurch mehr über ihre Eigenschaften und ihre Skalierbarkeit herausgefunden wurde. Des Weiteren wird nicht nur die technische Umsetzung dokumentiert, sondern auch die Entwicklungsprozesse des Templates.

2 Erste Phase

2.1 Inhalt

In der ersten Phase lag der Fokus auf der Suche und Bewertung verschiedener Technologien. Diese Phase gehört zu den wichtigsten Teilen dieser Diplomarbeit. Einerseits war es essenziell, die bestmögliche und qualitativ hochwertigste Technologie zu finden und zu testen, andererseits durften die Anforderungen der Firma Koncept nicht vernachlässigt werden.

Im Rahmen dieser Phase wurden verschiedenste Tools und Frameworks recherchiert. Alle relevanten Werkzeuge wurden ausreichend und detailliert dokumentiert und getestet. Besonders wurde bei den Tests auf Qualität und Effizienz der Technologien geachtet. Diese umfassende Analyse dient als Entscheidungsgrundlage für weitere Schritte.

2.2 Technologie

In dieser Phase werden Technologien untersucht, getestet und analysiert. Die Technologien wurden gemeinsam mit dem Betreuer in der Firma ausgewählt. Auch durch eigenständige Recherche im Internet wurden weitere Technologien relevant. Alle Technologien in diesem Schritt werden mit Eigenschaften sowie mit Vor- und Nachteilen beschrieben.

2.2.1 .Net-Standardvorlagen

[1] [2] [3]

Eigenschaften

Die .Net-Standardvorlage bietet eine effiziente Möglichkeit, Templates zu erstellen. Außerdem ist es ein Bestandteil des .Net SDK. Diese Technologie bietet die Funktion `dotnet new` an. Mit diesem Mechanismus kann man Templates nach Belieben erstellen und bearbeiten. Um ein Projekt erstellen zu können, muss eine Standardvorlage bestehen. Aufgrund dieser wird ein neues Projekt erstellt. In der Standardvorlage können mittels geschwungenen Klammern

Platzhalter eingefügt werden, welche beim Erstellen des Projekts überschrieben werden. Mit welchen Werten die Platzhalter überschrieben werden sollen, wird in der template.json Datei festgelegt. Beispiel:

Listing 1: Beispiel um mit dotnet new Templates Platzhalter zu ersetzen

```
1         "defaultValue": "net7.0",
2         "replaces": "{TargetFramwork}"
```

Hier wird festgelegt, dass der Platzhalter "TargetFramework"mit net7.0 ersetzt werden soll. Während der Erstellung des Projekts, kann der Ersteller auch durch Trigger entscheiden, mit welchen Werten Platzhalter ersetzt werden sollen.

Listing 2: Beispiel für einen Trigger in einem dotnet new Template

```
1       "TriggerName": {
2           "type": "parameter",
3           "description": "Gib den Namen des Triggers ein",
4           "datatype": "string",
5           "defaultValue": "TriggerDb",
6           "replaces": "{TriggerName}"
7       }
```

Bei dem Trigger wird der Typ angegeben. Durch das Keyword ”Parameter” wird eine Benutzereingabe erstellt. Ebenfalls werden eine Beschreibung, der Datentyp und der Platzhalter, welcher in den Vorlagen zu ersetzen ist, angegeben.

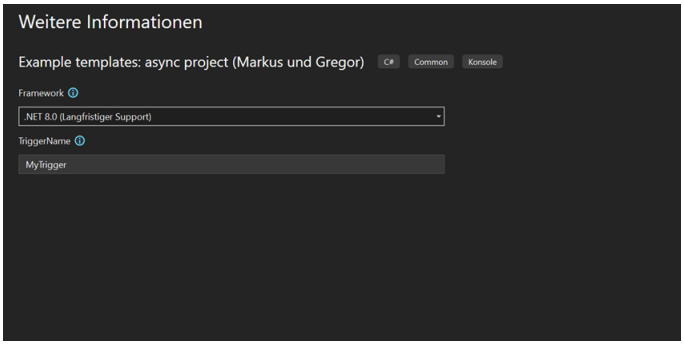


Abbildung 1: Der Trigger beim Erstellen eines Projekts

Vorteile

Der größte Vorteil dieser Technologie ist die einfache Implementierung in die .NET-Umgebung. Entwickler können schnell und effizient neue Projekte oder Templates erstellen, ohne zusätzliche Werkzeuge oder Tools zu benötigen. Da .NET bei Koncept bereits verwendet wird, müssen keine Änderungen der Infrastruktur vorgenommen werden. Darüber hinaus unterstützen die .Net-Standardvorlagen eine breite Menge an Projekttypen und bieten die Möglichkeit, die erstellten Templates im gesamten Unternehmen zu verwenden und zu implementieren.

Nachteile

Diese Technologie ist außerhalb der .NET-Umgebung fast nicht zu gebrauchen. Projekte können ausschließlich in Visual Studio erstellt werden. Dies stellt eine Einschränkung der Möglichkeiten der Technologie und der Projekte dar. Außerdem sind solche Templates nicht dynamisch. Individuelle Anforderungen des Entwicklers können nicht berücksichtigt werden. Es ist nur möglich, stumpfe Fragen zu stellen. Zum Beispiel könnte man mit dynamischen Fragen eine gewisse Anzahl an Klassen erstellen und für diese wiederum individuelle Fragen stellen. Stumpfe Fragen können zum Beispiel nur bestimmen, welcher Code im Projekt enthalten sein soll.

Aufgrund dieser Tatsachen ist diese Art der Technologie für die Firma Koncept nicht geeignet. Einerseits hat die Technologie viele nützliche Tools zu bieten, andererseits nützen diese Werkzeuge nichts, wenn sie nur in einem kleinen Rahmen verwendbar sind. Zusätzlich kann nicht auf individuelle Anforderungen und Bedürfnisse der Entwickler eingegangen werden.

2.2.2 Cookiecutter

[4] [5] [6] [7]

Eigenschaften

Cookiecutter bietet eine sehr einfache und schlichte Möglichkeit, Templates zu erstellen. Außerdem gibt es die Funktion, Cookiecutter direkt in Visual Studio zu implementieren. Da Koncept die meisten Projekte mit Visual Studio beziehungsweise in einer .NET-Umgebung programmiert, ist diese Technologie sehr naheliegend. Bei einer Implementierung Technologie in Visual Studio ist das Design des Templates irrelevant, da dann eine GUI in Visual Studio gezeigt wird. Mit dieser GUI lassen sich kleine Projekte einfach und schnell erstellen.

Abbildung 2: Cookiecutter

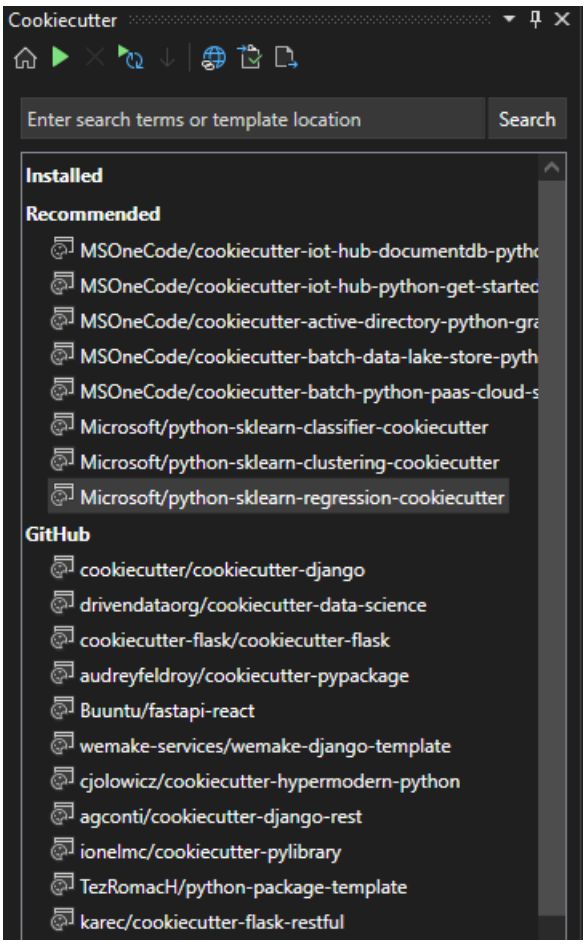


Abbildung 3: Cookiecutter-Browser

Im obigen Bild ist die Benutzeroberfläche von Cookiecutter dargestellt. Unter dem Reiter ”Installed” stehen die bereits installierten Templates von Cookiecutter, aber auch die eigens erstellten. Darunter befinden sich im Reiter ”GitHub” Vorlagen, welche online zur Verfügung gestellt werden. Hier ist ein Template auswählbar, welches am besten zu den Anforderungen des Projekts, das erstellt werden soll, passt.

Mittels Post-Generate Hooks, bei denen es sich um Python-Programme handelt, die nach dem Generieren der Dateien durchlaufen werden, lassen sich zusätzliche Funktionen zum Erstellen von Projekten hinzufügen. Insgesamt ist es ähnlich zu .Net-Standardvorlagen, da Cookiecutter ebenfalls mit einer JSON-Datei arbeitet. Auch in den Vorlagen, welche Cookiecutter verwendet, werden Platzhalter eingesetzt, um Benutzereingaben oder Sonstiges einzufügen.

Vorteile

Templates können ganz einfach nach Bedarf erstellt werden. Darüber hinaus ist Cookiecutter Open Source. Dadurch sind einige Testbeispiele oder Dokumentationen im Internet auffindbar. Zusätzlich ist Cookiecutter gut für die .NET-Umgebungen geeignet. Hier können die Templates

direkt implementiert oder installiert werden. Dies führt zu einer noch besseren Ressourcenaufteilung.

Nachteile

Es kann zu Ressourcenverschwendungen führen, wenn mit Cookiecutter zu große Projektvorlagen erstellt werden, da diese Technologie nicht für größere Projekte gedacht ist. Desto komplexer und größer das Template gestaltet wird, desto komplizierter wird es. Diese Technologie ist nur für kleinere Projektvorlagen geeignet. Bei größeren Projekten wird das Template zu unübersichtlich. Weitere Probleme können bei Wartungen oder Erweiterungen des Templates auftreten. Je größer ein Projekt ist, desto schwieriger wird es, Änderungen effizient umzusetzen. Darüber hinaus basiert Cookiecutter auf einer Verzeichnisstruktur und einer JSON-ähnlichen Syntax. Dadurch können Handlungen bei umfangreichen Projekten schnell unübersichtlich werden. Dies führt zu einem zusätzlichen Einarbeitungsaufwand für den Entwickler. Daher kommt Cookiecutter für die Fa. Koncept nicht in Frage. [4]

2.2.3 Gulp.js und Grunt.js

[8] [9] [10] [11] [12] [13]

Eigenschaften

Die Aufgaben von Gulp.js und Grunt.js bestehen darin, wiederkehrende Aufgaben zu automatisieren. Ebenfalls sind sie für die Frontend-Entwicklung gedacht. Grunt wird genauso wie Gulp über npm installiert und verwaltet. Sie sollen CSS und JavaScript-Dateien minifizieren oder mittels SASS oder LESS, SCSS zu CSS kompilieren. Weiters sind sie sogenannte Taskrunner. Das heißt, sie bekommen Aufgaben und arbeiten diese, eine nach der anderen, ab. Beide basieren auf Node.js und nutzen Plugins.

Abbildung 4: Gulp.js

Abbildung 5: Grunt.js

Grunt verwendet eine Konfiguration über eine JSON-Datei. Diese JSON-Datei enthält grundlegende Informationen über das Projekt. Dependencies werden auch in dieser Datei definiert. Ebenso gibt es eine JavaScript- oder CoffeeScript-Datei, welche dafür zuständig ist, Plugins zu laden und Tasks zu definieren. Hier ein Beispiel:

Listing 3: Beispiel für eine package.Json Datei

```
1      {
2    "name": "grunt-project",
3    "version": "1.0.0",
4    "description": "Ein Beispielprojekt mit Grunt",
5    "scripts": {
6      "start": "grunt"
7    },
8    "devDependencies": {
9      "grunt": "^1.6.1",
10     "grunt-contrib-cssmin": "^4.0.0",
11     "grunt-contrib-uglify": "^5.0.1",
12     "grunt-contrib-watch": "^1.1.0",
13     "grunt-contrib-sass": "^2.0.0",
14     "grunt-browser-sync": "^2.2.0"
15   }
16 }
```

Diese JSON-Datei beinhaltet die grundlegenden Informationen über ein Projekt. Mit devDependencies werden die Plugins zur Automatisierung angegeben und die Grunt-Version. In Scripts können Befehle angegeben werden. Hier wird der Start-Befehl mit Grunt angegeben, um die definierten Grunt-Tasks zu starten.

Listing 4: Beispiel für eine Javascript-Datei

```
module.exports = function (grunt) {
    // Konfigurationsobjekt fuer Grunt
    grunt.initConfig({
        pkg: grunt.file.readJSON('package.json'),

        // SCSS zu CSS kompilieren und minifizieren
        sass: {
            dist: {
                files: {
                    'dist/css/style.css': 'src/scss/style.scss'
                }
            }
        },
        cssmin: {
            target: {
                files: {
                    'dist/css/style.min.css': ['dist/css/style.css']
                }
            }
        },

        // JS minifizieren
        uglify: {
            target: {
                files: {
                    'dist/js/script.min.js': ['src/js/script.js']
                }
            }
        },

        // Live-Server mit BrowserSync
        browserSync: {
            dev: {
                bsFiles: {
                    src: ['dist/css/*.css', 'dist/js/*.js', 'dist/*.html']
                },
                options: {
                    watchTask: true,
                    server: './dist'
                }
            }
        },

        // Beobachtet SCSS und JS-Dateien fuer Aenderungen
        watch: {
            css: {
                files: 'src/scss/**/*.scss',
                tasks: ['sass', 'cssmin']
            },
            js: {
                files: 'src/js/**/*.js',
                tasks: ['uglify']
            },
            html: {
                files: 'dist/*.html'
            }
        }
    });

    // Plugins laden
    grunt.loadNpmTasks('grunt-contrib-sass');
    grunt.loadNpmTasks('grunt-contrib-cssmin');
    grunt.loadNpmTasks('grunt-contrib-uglify');
    grunt.loadNpmTasks('grunt-contrib-watch');
    grunt.loadNpmTasks('grunt-browser-sync');

    // Standard-Task
    grunt.registerTask('default', ['sass', 'cssmin', 'uglify', 'browserSync', 'watch']);
};
```

In dieser Javascript-Datei wird zuerst die package.json eingelesen. Dies geschieht mit dem Befehl grunt.file.readJSON. Als erster Bearbeitungsschritt wird SCSS mittels SASS in CSS konvertiert. Hier wird zuerst die Funktion SASS aufgerufen. Darin wird das Verzeichnis der CSS-Datei angegeben, in welcher der konvertierte Code stehen soll. Mit einem Doppelpunkt

ist dieser Pfad von dem Pfad der Ursprungs-Datei getrennt, in welchem der SCSS-Code steht. Als nächstes wird die Funktion cssmin aufgerufen. Diese minifiziert die bereits konvertierte CSS-Datei. In diesem Funktionsaufruf wird wieder der Ursprungspfad sowie der Speicherpfad angegeben. Das Javascript des Projektes wird nun durch die Funktion uglify minifiziert um eine kleinere Dateigröße zu erzielen. Ebenfalls wird innerhalb dieses Funktionsaufrufs der Ursprungs- sowie der Zielpfad angegeben. Mit dem Aufrufen der Funktion browserSync wird dafür gesorgt, dass der Browser sich automatisch neu lädt, falls sich Dateien ändern. Innerhalb des Aufrufs wird mittels bsFiles angegeben, welche Dateien überwacht werden. In den options wird auch noch der Server angegeben, welcher lokal gestartet wird und die dist-Dateien bereitstellt. Als letzte Funktion wird die watch-Funktion aufgerufen, welche bei Änderungen in der SCSS oder der JSON-Datei die vorher beschriebenen Funktionen erneut laufen lässt. Hier wird auch festgelegt, dass bei Änderungen in der HTML-Datei der Browser neu geladen werden soll. Nun werden noch die Plugins geladen. Ganz zum Schluss wird nun der Task als Standard definiert. Falls nun ”grunt” im Terminal aufgerufen wird, wird Grunt.js ausgeführt.

Gulp basiert im Gegensatz zu Grunt auf Code und wird mit JavaScript geschrieben. Diese Technologie arbeitet mit Streams, was eine effizientere Verarbeitung ermöglicht. Hierbei werden Dateien durch einige Verarbeitungsschritte gezogen, jedoch werden die Dateien nicht zwischengespeichert, was Gulp effizienter macht. Im folgenden Codebeispiel sieht man eine JavaScript-Datei mit Beispiel-Code für Gulp.:

Listing 5: Beispiel-Code für Gulp.Js

```
const { src, dest, series, parallel, watch } = require('gulp');
const sass = require('gulp-sass')(require('sass'));
const concat = require('gulp-concat');
const uglify = require('gulp-uglify');
const cleanCSS = require('gulp-clean-css');
const browserSync = require('browser-sync').create();

// Aufgabe: SCSS in CSS umwandeln und minifizieren
function styles() {
    return src('src/scss/**/*.scss')
        .pipe(sass().on('error', sass.logError))
        .pipe(cleanCSS())
        .pipe(dest('dist/css'))
        .pipe(browserSync.stream());
}

// Aufgabe: JS-Dateien zusammenfuegen und minifizieren
function scripts() {
    return src('src/js/**/*.js')
        .pipe(concat('main.js'))
        .pipe(uglify())
        .pipe(dest('dist/js'))
        .pipe(browserSync.stream());
}

// Aufgabe: Live-Server starten
function serve() {
    browserSync.init({
        server: {
            baseDir: './dist'
        }
    });

    watch('src/scss/**/*.scss', styles);
    watch('src/js/**/*.js', scripts);
    watch('dist/*.html').on('change', browserSync.reload);
}

// Standard-Task
exports.default = series(parallel(styles, scripts), serve);
```

In diesem Javascript passiert nun Folgendes:

In der Methode styles werden zuerst die SCSS-Dateien eingelesen, mit der SASS-Funktion in CSS umgewandelt und mit cleanCSS minifiziert. Nach dem Speichern der fertigen CSS-Datei führt der BrowserSync ein Live-Reload durch. In der scripts-Funktion werden JS-Dateien eingelesen und mittels der concat-Funktion zu einer einzigen Datei zusammengefasst. Die ulify-Funktion minifiziert diese zusammengefasste Datei. Anschließend wird wieder ein Live-Reload durchgeführt. Serve startet einen lokalen Server für die Entwicklung mittels browserSync und überwacht die SCSS- und Javascript-Dateien per watch um bei Änderungen die entsprechenden Tasks auszuführen. Durch default werden die styles- und scripts-Funktionen nun parallel ausgeführt und der Entwicklungsserver wird mittels serve gestartet.

Vorteile

Gulp ist durch die Nutzung von Streams recht schnell. Unnötige Schreib- und Lesevorgänge werden vermieden. Während der Verarbeitung können erkannte Fehler behandelt werden. Zudem verfügt es über eine große Auswahl an Plugins.

Grunt ist einfach gehalten und übersichtlich. Es gibt viele Dokumentationen. Durch eine große Community muss nicht für jedes Problem eine eigene Lösung kreiert werden, sondern kann nach bereits vorhandenen Lösungen gesucht werden.

Nachteile

Beide Technologien sind für die Frontend-Entwicklung gedacht und daher unpassend für unser Problem. Koncept möchte unser Produkt nämlich zur Code-Generierung im Frontend sowie im Backend einsetzen. Weiters können Grunt und Gulp zwar Code generieren, jedoch nicht Projekte wie für unseren Anwendungsfall gedacht erstellen. Falls diese Technologien eingesetzt werden, obwohl sie eigentlich schon ausgeschlossen wurden, gibt es noch weitere Nachteile. Grunt hat eine langsame Verarbeitung, da jede Aufgabe gespeichert wird. Es entstehen unnötige Festplattenzugriffe. Grunt kann auch schnell unübersichtlich werden, wenn zu viele Plugins verwendet werden. Dadurch wird die JSON-Datei schnell voll und man braucht lange, um dafür ein Verständnis zu entwickeln. Ein weiteres Problem stellt dar, dass Probleme erst nach der Ausführung erkannt werden. Da Gulp viel mit Streams arbeitet, kann ein Programmierer, welcher noch nie in Berührung mit diesen gekommen ist, schnell an seine Grenzen stoßen. Auch viele Plugins unterstützen die Arbeit mit Streams nicht optimal. Bei Fehlern mit Streams braucht es meist eine tiefe Analyse, um den Ursprung des Fehlers zu finden. Ebenfalls sind beide Technologien leicht veraltet. Modernere Tools bieten meist eine bereits bessere Performance. Auch können diese mit modernen Entwicklungsumgebungen verwendet werden.

2.2.4 Spring Initializr

[14] [15] [16]

Eigenschaften

Spring Initializr dient zur Erstellung neuer Spring-Boot-Projekte über das Web, die Spring Framework nutzen. Dieses bietet Eigenschaften wie Dependency Injection, simplen Zugriff auf Datenbanken und es vereinfacht das Erstellen von Webanwendungen sowie Rest-APIs. Spring Initializr unterstützt verschiedene Systeme wie Maven und Gradle. Eine Auswahl unterschiedlicher Build-Tools und Java-Versionen steht zur Verfügung, jedoch können keine CSharp-Projekte erstellt werden, da die verwendeten Technolo-

Abbildung 6: Spring Initializr.js

gien nur in Java einsetzbar sind. Spring Initializr ist auch in verschiedensten IDEs wie IntelliJ IDE, Eclipse und Visual Studio Code integriert,wodurch beim Erstellen von Projekten die Weboberfläche nicht genutzt werden muss.

Abbildung 7: Web-GUI von SpringInitializr

Wie aus dem obigen Bild ersichtlich, stehen dem Entwickler einige Auswahlmöglichkeiten zur Verfügung. Zuerst wird die Art des Projektes festgelegt. Darunter ist die Programmiersprache auswählbar, sowie die Spring Version. Die Metadaten wie Projektname oder Beschreibung werden vom Entwickler angefordert. Nach der Entscheidung zwischen Jar oder War im packaging steht noch eine Java-Version bereit. Bevor man das Projekt generieren lässt, können noch Dependencies hinzugefügt werden.

Vorteile

Diese Technologie bringt eine große Zeitersparnis mit sich. Spring Initializr verfügt über eine große Auswahl an Dependencies, welche hinzugefügt werden können. Dadurch lassen sich Projekte erstellen, welche den Ansprüchen des Entwicklers entsprechen. Durch die Integration in verschiedene IDEs ist es relativ einfach darauf zuzugreifen und damit zu entwickeln. Zusätzlich kann jederzeit über das Web auf Spring Initializr zugegriffen werden.

Nachteile

Da mit dieser Technologie nur Java-Projekte erstellt werden können, wurde sie verworfen. Weiteres kann nicht dynamisch auf den Benutzer eingegangen werden, da die Auswahl schon vorgefertigt ist und keine benutzerdefinierten Einstellungen hinzugefügt werden können. Auch ist Spring Initializr nur auf einen Bereich spezialisiert, nämlich die Erstellung von Webanwendungen und Microservices. Da Koncept nicht auf Webanwendungen oder Microservices spezialisiert ist und auch nicht mit Java programmiert, stattdessen mit CSharp, ist diese Technologie vollkommen ungeeignet für das Vorhaben unserer Diplomarbeit.

2.2.5 Helm Charts

[17]

Eigenschaften

Helm Charts ist ein Werkzeug zur Verwaltung und Bereitstellung von Kubernetes-Anwendungen. Sie fassen notwendige YAML-Manifeste in wiederverwendbaren Paketen zusammen. Aufmerksam sind wir auf diese Technologie geworden, da man dynamische Kubernetes-Manifeste generieren lassen kann. Auch das Wort "Template" hat uns aufhorchen lassen. Mit Helm Charts können nämlich dank einer integrierten Template-Engine Konfigurationsdateien dynamisch erstellt werden. Die Konfiguration erfolgt hierbei über eine values.yaml-Datei, in welcher die für Deployment, Services, ConfigMaps und weitere, benötigten Kubernetes-Objekte definiert werden. Jedoch benötigt Koncept keine Kubernetes-Manifeste, da sie mit Kubernetes-Anwendungen nicht arbeiten.

Abbildung 8: Helm Charts

2.2.6 T4

[18] [19] [20]

Eigenschaften

T4 ist eine Template Engine, welches in Visual Studio integriert ist. Automatisiertes Code-generieren ist hiermit möglich. Der zu generierende Code wird in eine Datei notiert und dann definiert. Während des Compiliervorgangs wird an der Stelle, an der das Template eingefügt wurde, der gewünschte Code generiert. Standardisierte Codefragmente können erstellt werden. Das Template besteht aus statischem Text aber auch aus Kontrollstrukturen, die in CSharp oder VB.NET geschrieben werden. Durch die Kontrollstrukturen, die Codefragmente sind, kann auch etwas Logik in die Generierung von Code eingebaut werden. Sollen mehrere Klassen erstellt werden, so muss für jedes Codesegment ein Template, also eine Text-Datei, erstellt werden, welche später zu Code wird. Das dynamische Erstellen von Klassen ist daher nicht möglich.

Abbildung 9: T4

Vorteile

Da T4 bereits in Visual Studio integriert ist, müssen keine extra Programme ausgeführt werden, um Code einfügen zu können. Die Textdateien bilden eine genaue Übersicht, da der Code komprimierter ist. Boilerplate-Code wird nicht angezeigt , da dieser ja noch nicht generiert ist. Um eine andere Art von Code einzufügen, müssen lediglich die Templates ausgetauscht werden und somit können Projekte mit anderen Inhalten erstellt werden. Falls der gleiche Code benötigt wird, kann ein Template in mehrere Projekte eingefügt werden. Durch den integrierten Code in den Templates lassen sich auch komplexere Logiken in die Codegenerierung einbauen.

Nachteile

Dynamisch auf die Wünsche des Benutzers einzugehen, ist nicht möglich, jedoch kann eine Textdatei mehrmals als Vorlage in einem Projekt verwendet und somit mehrere Klassen mit dem selben Inhalt erzeugt werden. Der Inhalt ist natürlich immer gleich und beispielsweise kann der Inhalt für jede einzelne Klasse nicht mit dem selben Ursprungs-Template geändert werden. Also wird der Code nicht dynamisch erstellt. Das eigentliche Ziel der Diplomarbeit,

ganze Projekte von Grund auf generieren zu lassen, ist nicht möglich. Auch die Wartbarkeit bei größeren Templates ist schwierig, da in den Templates eine Mischung von Text, welcher bereits aussieht wie Code (dieser wird während dem Kompilieren im Projekt eingefügt), und echtem Code besteht. Weiters kann die Performance unter T4 leiden. Die Verarbeitung bei größeren Templates oder bei öfterem Kompilieren nimmt schon etwas Zeit in Anspruch. IntelliSense gibt es in diesen Template-Files nicht, weshalb schneller Fehler eingebaut werden. Hilfestellungen beim Programmieren gibt es daher nicht. T4 ist nur für Visual Studio verfügbar und kann dadurch in anderen Umgebungen wie IntelliJ Rider nicht verwendet werden.

2.3 Resümee

Im Rahmen dieser ersten Phase wurden verschiedenste Technologien für die Entwicklung und Optimierung von Templates untersucht und getestet. Alle Technologien werden genauestens auf ihre Fähigkeiten und Einsatzmöglichkeiten überprüft, wobei jede Technologie bestimmte Vor- und Nachteile mit sich bringt.

Auch sind wir auf Technologien gestoßen, welche zwar Code generieren könnten, aber für unseren Verwendungszweck nicht nützlich sind.

Besonders wichtig waren uns einerseits die unterschiedlichen Möglichkeiten beim Entwerfen eines Templates, andererseits die Analyse der Wartbarkeit der Ergebnisse. Technologien, die nur für kleine Vorlagen geeignet sind, wurden ausgeschlossen, da die Firma Koncept eine Lösung benötigt, die komplexe Strukturen abbilden kann. Durch die Recherche stellte sich heraus, dass einige Technologien zu stark mit anderen Systemen zusammenhängen. Auch diese Tools wurden verworfen.

Zusammenfassend lässt sich sagen, dass diese Analyse von verschiedensten Tools eine solide Entscheidungsgrundlage darstellte. Durch diese detaillierte Suche wurden viele Informationen zu den Technologien gesammelt. Diese Fakten spielten eine große Rolle für die richtige Technologiewahl. Leider wurden in dieser Phase keine passenden Technologien gefunden. Alle untersuchten Werkzeuge wiesen zu große Nachteile auf oder waren nicht für Projektvorlagen dieses Umfangs geeignet. Somit war die Suche nach Technologien noch nicht beendet. Daher beschäftigt sich auch die zweite Phase dieser Diplomarbeit mit der Auffindung verschiedenster Tools und Werkzeuge. In der Zwischenzeit fanden weitere Meetings statt, in denen die bisher gefundenen Technologien dem Auftraggeber vorgestellt und über weitere Möglichkeiten mit anderen Technologien diskutiert wurde.

3 Zweite Phase

3.1 Inhalt

Da die Suche nach der passenden Technologie nicht abgeschlossen war, beschäftigt sich auch die zweite Phase mit diesem Thema. Mit der Firma Koncept wurden die getesteten Technologien besprochen. Dadurch ergab sich, dass die Technologie .Net-Standardvorlagen weiter untersucht werden soll und Workarounds zu erarbeiten sind, um die Nachteile der Technologien einzuschränken.

3.2 Technologie

3.2.1 .Net-Standardvorlagen

[21] [22] [23]

Eigenschaften

Die Stärken und Schwächen wurden bereits in der ersten Phase ausführlich beschrieben. Nach zusätzlicher Recherche wurden sogenannte ”Post-Actions” gefunden. In Verbindung mit .Net-Standardvorlagen entstehen mehrere Möglichkeiten, diese Technologie einzusetzen. Der Einsatzbereich wird somit vergrößert. Mit Post-Actions können mehr Konfigurationen definiert und zusätzliche Befehle oder Schritte automatisch ausgeführt werden. Nach der Projekterstellung ermöglicht diese Technologie ein automatisiertes Erstellen von Konfigurationen. Mittels Post Actions können Befehle wie in einem Terminal ausgeführt werden.

Listing 6: Beispiel für Post-Actions in einer .Net-Standardvorlage

```
"postActions": [
    {
      "actionId": "12345678-1234-1234-1234-1234567890ab",
      "description": "Open the project in Visual Studio",
      "manualInstructions": [
        {
          "text": "To open the project in Visual Studio, run: code ${projectName}.sln"
        }
      ],
      "args": {
        "executable": "cmd",
        "args": "/c start ${projectName}.sln"
      }
    }
  ],
```

Hier wird zuerst ein Array namens postActions erstellt, in welchem die Post-Actions angelegt werden. Jede Post-Action verfügt über eine eigene ”actionId”, welche zur Identifikation dient. Ebenfalls besitzen Post-Actions eine Beschreibung. In diesem Beispiel lautet die Beschreibung ”Open the project in Visual Studio”. In den ”manuelInstructions” wird die Anweisung an den Benutzer eingetragen. Im letzten Schritt wird in ”args” die ”executable”, also das Terminal, mit welchem man das Script ausführen will, angegeben. In diesem Fall ist es die cmd(Commandline). Auch das Script wird in ”args” geschrieben. In diesem Beispiel wird die Solution mit dem Namen projectName.sln gestartet, wobei ”projectName” ein Platzhalter für den Projektnamen ist.

Vorteile

Der größte Vorteil der Technologie .Net-Standardvorlagen mit der Erweiterung von Post-Actions ist die Ausweitung der Einsatzbereiche. Durch diese Erweiterung ist es möglich, verschiedene Schritte automatisch durchzuführen. Darüber hinaus ist die Implementierung nicht komplex. Die Einbindung erfolgt lediglich über eine JSON-Datei.

Nachteile

Auf der anderen Seite werden die Nachteile der Technologie nicht behoben. Die Umgebung muss dabei immer noch .NET sein und kann nur schwer in anderen Systemen implementiert werden. Zusätzlich reagiert diese Technologie nicht dynamisch auf die Anforderungen der Benutzer. Dieses wesentliche Problem kann nicht mit Post-Actions gelöst werden.

3.2.2 Wizard

[24] [25]

Eigenschaften

Abbildung 10: Wizard

Wizard oder auch Visual Studio Wizard hilft dem Entwickler Projekte, Dateien oder Code Vorlagen zu erstellen. Dies macht er entweder mit vorgefertigten oder mit eigens erstellten Templates. Ein Wizard ist besonders hilfreich, wenn mehrmals Ähnliches oder dasselbe erstellt werden soll. Wizard ist auch ein fixer Bestandteil von Visual Studio, weshalb er auch nur CSharp Projekte erstellen kann. Durch eine vorgefertigte Struktur lässt Wizard neue Projekte anlegen. Beim Generieren werden Standardordner und Konfigurationsdateien automatisch erstellt. Die vorgefertigte Struktur soll die Dateien, den Code und extra Konfigurationsdateien und Einstellungen enthalten, welche sich später auch im Code befinden sollen. Die Konfigurationen werden über .vstemplate-Datein in XML geschrieben. Diese Dateien geben die Struktur vor und definieren das Verhalten des Templates. Auch in dieser XML-Datei wird beschrieben, was nach dem Starten passieren soll. Etwa kann man ein Fenster öffnen lassen, welches Benutzereingaben einfordert und mit diesen den Inhalt des erstellten Projekts verändert. Der Wizzard kommuniziert deshalb auch begrenzt mit dem Entwickler. Beim Erstellen eines neuen Projekts wird der vorgegebene Code ausgeführt. Da diese Technologie ein direktes Feature von Visual Studio ist, kann damit natürlich auch jede Art von Projekttyp erstellt werden, wie Konsolenanwendungen, Windows Forms- oder WPF-Anwendungen. Wizard-Templates lassen sich durch eine Visual Studio-Erweiterung auch einfach entwickeln. Somit ist die Erstellung eines eigenen Templates für jede Art von Projekt möglich. Hier ein Beispiel für eine .vstemplate-Datei in einem Wizard:

Listing 7: Beispiel-Code für .vstemplate-Datei

```
<?xml version="1.0" encoding="utf-8"?>
<VSTemplate Version="3.0.0" Type="Project"
    xmlns="http://schemas.microsoft.com/developer/vstemplate/2005"
    xmlns:sdk="http://schemas.microsoft.com/developer/vstemplate-sdkextension/2010">
  <TemplateData>
    <Name>MyProjectTemplate</Name>
    <Description>MyProjectTemplate</Description>
    <Icon>MyProjectTemplate.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <LanguageTag>csharp</LanguageTag>
    <RequiredFrameworkVersion>2.0</RequiredFrameworkVersion>
    <SortOrder>1000</SortOrder>
    <TemplateID>988d80f0-dfad-49d0-9463-9937542920b3</TemplateID>
    <CreateNewFolder>true</CreateNewFolder>
    <DefaultName>MyProjectTemplate</DefaultName>
    <ProvideDefaultName>true</ProvideDefaultName>
  </TemplateData>
  <TemplateContent>
    <Project File="ProjectTemplate.csproj" ReplaceParameters="true">
      <ProjectItem ReplaceParameters="true"
          TargetFileName="Properties\AssemblyInfo.cs">AssemblyInfo.cs</ProjectItem>
      <ProjectItem ReplaceParameters="true" OpenInEditor="true">Class1.cs</ProjectItem>
    </Project>
  </TemplateContent>
  <WizardExtension>
    <Assembly>MyProjectWizard, Version=1.0.0.0, Culture=Neutral,
        PublicKeyToken=cdfd6ccf2198833c</Assembly>
    <FullClassName>MyProjectWizard.WizardImplementation</FullClassName>
  </WizardExtension>
</VSTemplate>
```

Zuerst wird im Wurzelelement ”VSTemplate” die Version, der Typ und die Namespaces angegeben. Die Projektvorlage ist am Typ erkennbar. In ”TemplateData” werden die Metadaten angegeben, darunter Projekttype und LanguageTag. Mit diesen Parametern wird festgelegt, dass es sich um eine CSharp-Projektvorlage handelt und auch die Programmiersprache CSharp ist. Die Reihenfolge der Vorlagenauswahl wird mit SortOrder fixiert. Auch eine eindeutige TemplateID wird angelegt. CreateNewFolder legt fest, ob ein neuer Projektordner erstellt werden soll. Am Ende von ”TemplateData” wird noch der Standardname für das Projekt definiert. Die Projektsstruktur wird in TemplateContent festgelegt. Es wird die .csproj-Datei angegeben, welche als Basis dient. Dadurch, dass ReplaceParameters auf true gesetzt wird, werden Platzhalter in der .csproj ersetzt. ProjectItem gibt eine Klasse an: hier AssemblyInfo.cs und Class1.cs. Diese Klassen werden später im fertigen Projekt erstellt. Außerdem wird nach dem Erstellen des Projekts durch das Setzen von OpenInEditor Class1.cs automatisch geöffnet. Mithilfe von WizardExtension kann nun Logik angefügt werden. Assembly gibt die DLL-Datei an, die die Logik beinhaltet und FullClassName beschreibt die Klasse, in welcher die DLL-Datei enthalten ist. Diese Logik kann zum Beispiel ein Fenster sein, in welches Klassennamen eingetragen werden, um die zu erstellenden Klassen individuell zu benennen.

Vorteile

Dadurch dass diese Technologie dem ”normalen” Erstellen eines Projekts ähnelt, kann auch ein Laie relativ schnell mit dem Wizzard neue Projekte erstellen. Da die Templates schon

vor dem Erstellen des Codes vorhanden sind, wird natürlich das Risiko vermindert Fehler im Projekt einzubauen. Auch lassen sich Änderungen einfach vornehmen, da bloß die Dateien in den Templates verändert werden müssen und diese Änderungen dadurch automatisch in den zukünftigen Projekten eingebaut sind. Standard-Templates, welche von Microsoft bereitgestellt werden, können noch vom Entwickler verändert und somit angepasst werden. Manche Templates können nicht nur Code generieren, sondern auch kommentieren. Da der Wizzard ein direkter Bestandteil von Visual Studio ist, ist er natürlich auch ein ideales Tool, um CSharp-Projekte in dieser Umgebung zu erstellen, was auch die Aufgabe unserer Arbeit sein sollte. Ebenso müssen dadurch keine extra Technologien, Plugins oder Erweiterungen installiert werden.

Nachteile

Der generierte Code ist oft generisch und muss daher meistens manuell nachbearbeitet werden. Da manche Templates mehr Dateien als nötig erzeugen, kann es dazu kommen, dass das Endresultat zu komplex wird. Da Wizzard ja nur für Visual Studio verfügbar ist, kann die Projekterstellung nur in CSharp erfolgen. Ältere vorgefertigte Templates sind heutzutage nicht mehr mit modernen Technologien kompatibel oder enthalten Sicherheitslücken. Visual Studio Wizzard ist natürlich bis zu einem gewissen Grad dynamisch, jedoch reicht das für unsere Anforderungen nicht aus, da er keine vorher definierte Anzahl an Klassen erstellen kann, welche alle unterschiedliche Parameter bekommen oder zurückgeben. Der Wizzard muss beim Erstellen eines Projekts extra ausgewählt werden. Dabei öffnet sich ein weiteres Fenster, in welchem die Dialogfenster für die Kommunikation sichtbar sind. Dies ist natürlich kein Problem, welches das Ergebnis beeinflussen könnte, jedoch gefiel den Mitarbeitern von Koncept diese Vorgehensweise nicht.

3.3 Resümee

In der zweiten Phase wurde die Technologie .Net-Standardvorlagen noch genauer untersucht und auf Erweiterungen getestet. Das Ziel war es, dieses Werkzeug so anzupassen, dass es den Anforderungen der Firma Koncept gerecht wird. Eine der wichtigsten und vielversprechendsten Technologien waren die Post-Actions. Mit ihnen war es möglich, automatische Schritte beim Erstellen eines Projekts durchzuführen. Jedoch löste dieses Tool nicht eines der zentrale Probleme, nämlich die fehlende Möglichkeit, komplexe Konfigurationen innerhalb einer Vorlage zu erstellen. Zusätzlich gibt es keine praktische Umsetzung dieser Technologie, die es ermöglicht, dynamisch individuelle Projekte zu erstellen.

Aufgrund dieser Nachteile und der Forderungen von Koncept muss .Net-Standardvorlage endgültig ausgeschlossen werden.

Als weitere Technologie wurde Visual Studio Wizard Templates genauer unter die Lupe genommen. Bei dieser Technologie ist es möglich, Code während des Erstellens auszuführen, jedoch müssen für die zu erstellenden Klassen, Vorlagen bestehen, somit ist dafür keine Dynamik gegeben. Es besteht also das gleiche Problem wie bei den .Net-Standardvorlagen.

Daher müssen neue Werkzeuge, Technologien und Tools eruiert werden. Dieser Schritt findet in der dritten Phase statt.

4 Dritte Phase

4.1 Inhalt

In dieser Phase liegt der Fokus darauf, alternative Technologien zu testen und zu dokumentieren, um die Anforderungen von Koncept für die Templates zu erfüllen. Es ist wichtig, die geeignete Technologie zu finden, um das bestmögliche Ergebnis zu präsentieren. Im Rahmen dieser Phase wurden zwei wesentliche Tools erforscht und zahlreiche Prototypen entwickelt. Diese Technologien wurden erst so spät erforscht, da der Fokus zuvor darauf lag ein Workaround für .Net-Standardvorlagen zu finden. Mithilfe der Mitarbeiter und zahlreicher Meetings wurden verschiedene Faktoren bewertet.

Einerseits war Koncept die Benutzerfreundlichkeit der Gestaltung des Templates mit der Technologie wichtig, andererseits wurden die Eigenschaften der Technologie sowie die Vor- und Nachteile in den Bewertungskriterien miteinbezogen. Diese Fakten liefern eine wichtige Entscheidungsgrundlage.

4.2 Technologie

4.2.1 Yeoman

[26] [27] [28] [29]

Eigenschaften

Im Rahmen dieser Phase wurde Yeoman als potenzielles Werkzeug untersucht. Yeoman ist ein Open-Source-Tool und ist weit verbreitet. Als Grundlage wird eine Node.js Umgebung verwendet, in der bestimmte Konfigurationen bestimmt werden. Sie erstellt Vorgaben, die das Aussehen des Projekts beschreiben. Darüber hinaus verwendet Yeoman sogenannte Generatoren, die für die Erzeugung der Projekte verantwortlich sind. Mit ihnen lassen sich Boilerplate-Code sowie Projektstrukturen automatisch generieren.

Abbildung 11: Yeoman

In Yeoman gibt es eine Generator-Architektur. Durch diese stehen dem Entwickler Vorgefertigte Templates für jede Art von Projekten zur Verfügung. Natürlich kann auch ein eigenes Template erstellt werden. Yeoman ist Node basiert und arbeitet mit npm. Dadurch fungiert diese Technologie nur im Terminal und es gibt keine grafische Benutzeroberfläche. Eingabeaufforderungen stellen Informationen zum Projekt, welche in die generierten Dateien eingebaut werden, bereit. Die Eingabeaufforderungen werden mittels Javascript programmiert. Yeoman ersetzt wie andere Technologien Platzhalter, jedoch können durch die Programmierung in Javascript dynamisch Inhalte hinzugefügt werden.

Listing 8: Beispiel für Eingabeaufforderungen in Yeoman

```
async prompting() {
          var amount = "b";
          const nameAnswer = await this.prompt([
          {
            type: "input",
            name: "name",
            message: "Your project name",
            default: "testProj" // Default to current folder name
          }]);

          //wiederholen bis eine Zahl uebergeben wird
            while(isNaN(amount)){
              const answer= await this.prompt([
            {

            type: "input",
            name: "amount",
            message: "How many Inputs do you want"

            }]);
            amount = answer.amount;
          };
```

Abbildung 12: Eingabe in Yeoman

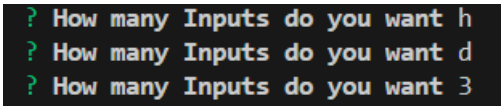


Abbildung 13: Schleife in Yeoman

In diesem Code-Teil ist die Implementierung zweier Prompts ersichtlich. Im ersten Prompt wird mit ”type” festgelegt, dass es sich um eine Eingabe handelt. Die Eingabe erhält für spätere Verwendungen den Namen ”name”. Zusätzlich erhält der Benutzer die Nachricht ”Your project name”, um den Projektnamen einzugeben. Auch der Default-Ordner wird angegeben. Darunter ist eine Schleife ersichtlich. Somit kann schon mehr Logik als in anderen Templates verwendet werden. Diese Schleife fragt den Nutzer sooft, wie viele Inputs er erstellen möchte, bis eine Zahl eingegeben wird. Dies wird mit der Funktion isNAN (is not a number) überprüft.

Listing 9: Beispiel für einen Trigger in einem dotnet new Template

```
1
2  const promts = await this.prompt([
3              {
4                type: 'checkbox',
5                name: 'function',
6                message: 'select function',
7                choices: [
8                    {
9                      name: 'input-function',
10                     value: 'input-function',
11                   },{
12                     name: 'output-function',
13                     value: 'output-function',
14                   },{
15                     name: 'work-function',
16                     value: 'work-function',
17                   }
18
19               ]
20             }
```

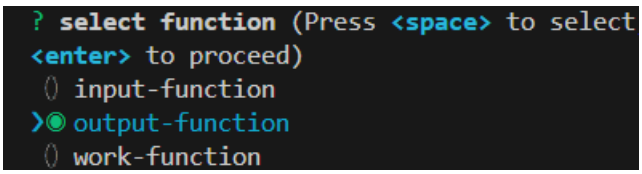


Abbildung 14: Auswahl in Yeoman

In diesem Codeteil wird eine Checkbox implementiert, also eine Liste von verschiedenen auswählbaren Attributen. Mittels ”type” wird zuerst festgelegt, dass es sich um eine Checkbox handelt. Auch ein Name wird zur späteren Verarbeitung vergeben. Im Array ”choices” sind die jeweiligen Auswahlmöglichkeiten angegeben. Jede dieser Auswahlmöglichkeiten besitzt einen Namen zur späteren Verarbeitung, welcher mittels ”name” deklariert wird. Der dem User angezeigte Text wird mit ”value” festgelegt.

Vorteile

Während der Analyse wurde Yeoman auf Benutzerfreundlichkeit, Flexibilität und Dynamik überprüft. Die Flexibilität zeigt sich als wesentlichster Vorteil.

Einer der größten Vorteile ist die dynamische Anpassung der Templates. Durch die Dynamik, die Yeoman anbietet, ist es möglich, auf die Bedürfnisse der Entwickler einzugehen. Im Gegensatz zu statischen Vorlagen können während des Generierungsvorgangs mit Yeoman interaktive Eingaben getätigt werden.

Nachteile

Trotz der zahlreichen Vorteile hat Yeoman auch seine Schattenseiten. Yeoman basiert vollständig auf Node.js., weswegen eine neue Umgebung in der Firma implementiert werden müsste. Zusätzlich bietet Yeoman keine eigene Benutzeroberfläche an und wird nur in der Command-Line ausgeführt.

Bei der Überprüfung von Yeoman haben wir festgestellt, dass seit mehreren Jahren auf dem Git-Repository nicht mehr gepusht wurde, was darauf schließen lässt, dass es keine Updates mehr geben wird. Dem Auftraggeber war dies ein zu hohes Sicherheitsrisiko.

4.3 Resümee

Oben angeführte Nachteile haben - nach mehreren Besprechungen mit dem Auftraggeber - zu der Entscheidung geführt Yeoman nicht als Technologie für die Template-Entwicklung zu verwenden. Die Begründung liegt in den Einschränkungen, die während dieser Phase festgestellt wurden. Yeoman bietet viele verschiedene Vorteile für die Entwicklung an.

Andererseits wurde entdeckt, dass Yeoman schon seit längerem keine wesentlichen Updates erhält. Während dieser Phase ergab sich, dass manche Tools und Features von Yeoman nicht mehr weiterentwickelt werden, was langfristig zu Kompatibilitätsproblemen und erhöhtem Wartungsaufwand führen wird. Erst vor kurzem wurde Yeoman von zwei neuen Entwicklern wieder aufgenommen. Zu dieser Zeit war die Diplomarbeit jedoch bereits fertiggestellt. Diese Fakten trugen dazu bei die Technologie Yeoman auszuschließen.

5 Vierte Phase

5.1 Inhalt

Da in der letzten Phase Yeoman aufgrund der Inaktivität der Entwickler abgelehnt wurde, ist in dieser Phase nach einer ähnlichen Technologie gesucht worden. Die gefundene Lösung erfüllt alle Anforderungen für die Firma Koncept und übertrifft sie sogar. Die Wahl fiel auf ”Plop”, eine leistungsstarke und dynamische Template-Engine, mit der es möglich ist, effizient Projektstrukturen zu erstellen.

Während in den letzten Phasen viele verschiedene Technologien auf ihre Stärken und Schwächen getestet wurden, wurde in der letzten Phase ”Plop” als beste Technologie für die Template-Entwicklung identifiziert. In dieser Phase wurden alle Eigenschaften detailliert beschrieben. Es wurden nicht nur theoretische Merkmale beschrieben, sondern auch die Ergebnisse der praktischen Implementierungen, die in Test-Umgebungen getestet worden waren, niedergeschrieben. Durch diese zahlreichen Tests konnte sichergestellt werden, dass ”Plop” wirklich die beste Technologie für diese Diplomarbeit ist.

5.2 Technologie

5.2.1 Plop

[30] [31] [32] [33]

Eigenschaften

Die Eigenschaften, welche Plop aufweisen musste, sind: die Benutzerfreundlichkeit, die Effizienz und die Dynamik. Plop ist ein leistungsfähiges CLI-Tool, das oft in der Entwicklung von Templates genutzt wird. Es ermöglicht den Entwicklern, effizient Projektstrukturen und Dateien zu generieren. Durch die hohe Skalierbarkeit des Tools kann das Template nach Belieben vergrößert und individuell angepasst werden. Die Generatoren und Vorlagen,

Abbildung 15: Plop

welche Plop benötigt, werden in der plopfile.js konfiguriert. Dies
ist eine Javascript-Datei, in welcher sich auch die Logik des Templates implementieren lässt. Da Plop Javascript verwendet, lässt sich auch dynamisch programmieren. Unsere Diplomarbeit ziehlt darauf ab, Klassen dynamisch zu erstellen, wodurch Plop ideal ist. Insgesamt arbeitet Plop ähnlich wie Yeoman und ersetzt auch Platzhalter. Ebenfalls kann mit dem Entwickler interagiert werden. Plop wurde zum Zeitpunkt unserer Praktikums im Gegensatz zu Yeoman regelmäßig weiterentwickelt.

Listing 10: Beispiel Benutzerinteraktion in Plop

```
const answers = await inquirer.prompt([
    {
        type: 'input',
        name: 'projectName',
        message: 'Enter your Projectname'
    },
    {
        type: 'list',
        name: `version`,
        message: 'Choose the version',
        choices: [
            { name: "net8.0", value: "net8.0" },
            { name: "net7.0", value: "net7.0" },
            { name: "net6.0", value: "net6.0" }
        ]
    }
]);
```

Abbildung 16: Eingabe in Plop

Abbildung 17: Auswahlmöglichkeiten in Plop

In diesem Beispiel sind zwei Arten der Benutzerinteraktion ersichtlich, ein einfaches Eingabefeld und eine Liste mit Auswahlmöglichkeiten.
Zuerst zu der Benutzereingabe: Ähnlich wie in Yeoman wird mittels ”type” definiert, dass es sich um eine Eingabe handelt. Auch ein Name wird mittels ”name” definiert. Zusätzlich wird mit ”message” eine Nachricht an den Benutzer ausgegeben.
Mittels des Types ”list” wird die zweite Benutzerinteraktion als Liste mit Auswahlmöglichkeiten definiert. Auch diese Liste besitzt einen Namen zur späteren Verarbeitung und eine Nachricht an den User. Die Auswahlmöglichkeiten der Liste wird im Array ”choices” wiedergegeben. Jede dieser Auswahlmöglichkeiten besitzt einen Namen und einen ”value”, welcher dem Benutzer angezeigt wird.

Vorteile

Besonders positiv fiel auf, dass Plop eine interaktive und dynamische Benutzerführung bietet, die es Entwicklern ermöglicht, individuell angepasste Templates mit minimalem Aufwand zu erstellen. Diese dynamische Herangehensweise erleichtert nicht nur die Arbeit mit Templates, sondern stellt auch sicher, dass die generierten Projekte stets den gewünschten Anforderungen entsprechen.

Ein weiterer großer Vorteil von Plop ist die einfache Integration in bestehende Entwicklungsumgebungen. Da Plop keine umfangreichen Abhängigkeiten benötigt, lässt es sich problemlos in bestehende Workflows einbinden, ohne dass größere Anpassungen an der Infrastruktur notwendig sind. Dies macht Plop besonders attraktiv für Unternehmen wie Koncept, die eine Lösung benötigen, die sich ohne hohe Implementierungskosten und aufwendige Schulungen in den Arbeitsalltag integrieren lässt. Zudem zeichnet sich Plop durch seine hohe Flexibilität aus. Entwickler haben die Möglichkeit, eigene Generatoren zu definieren und so individuell angepasste Strukturen zu erstellen, die exakt den Anforderungen des Unternehmens entsprechen.

Ein weiterer Aspekt, der für die Wahl von Plop sprach, war die Benutzerfreundlichkeit der Technologie. Während einige der zuvor getesteten Lösungen eine hohe Einarbeitungszeit erforderten, ließ sich Plop ohne großen Aufwand in die tägliche Entwicklungsarbeit integrieren. Die Konfigurationsmöglichkeiten sind verständlich aufgebaut, und die Nutzung von Plop ist intuitiv, sodass auch neue Mitarbeitende ohne lange Einarbeitungszeiten produktiv mit der Technologie arbeiten können.

Nachteile

Während dieser intensiven Testphase wurden keine relevanten Nachteile oder Schwächen entdeckt. Diese Technologie erfüllt exakt die Anforderungen der Firma Koncept. Tatsächlich ergab sich Plop als so effektiv und dynamisch, dass dieses Tool die geeignetste Technologie ist und auch langfristig eine innovative Lösung darstellt.

5.3 Resümee

Nach einer intensiven Suche nach der besten Technologie konnte in der letzten Phase die abschließende Technologie Plop gefunden werden. Durch die enge Zusammenarbeit der Firma Koncept und ihrer Mitarbeiter wurde Plop als die sinnvollste und technisch beste Wahl abgesegnet. Durch seine intuitive Bedienung, seine hohe Anpassungsfähigkeit und seine nahtlose Integration

in bestehende Workflows bietet Plop eine langfristige Lösung, die den Entwicklungsprozess bei Koncept nachhaltig verbessern wird.

Durch die abschließende Entscheidung der Technologie Plop wurde endlich die geeignetste Technologie gefunden. Alle Tools und Werkzeuge, die in den vier Phasen beschrieben wurden, dienten als entscheidende Grundlage. Die Verwendung und Implementierung von Plop bedeutete nicht nur eine Optimierung der Templates, sondern auch der Firma Konzept eine effektive Lösung zu präsentieren. Die zuvor bestehenden Probleme durch unstrukturierte oder fehleranfällige Template-Lösungen konnten durch den Einsatz von Plop erfolgreich eliminiert werden.

6 Implementierung

6.1 Funktionen

In diesem besonderen Abschnitt werden alle Technologien, Funktionen und Methoden dieser Diplomarbeit beschrieben. Zusätzlich werden die Umstände und Implementierungsvorgänge detailliert dargestellt. Die zentrale Suche der Technologie wird in diesem Abschnitt nicht analysiert. Dieses Thema wurde in den Kapiteln zuvor genug beschrieben.

6.1.1 Hauptprogramm

Das Template-Engine Plop diente als zentrale Technologie. Durch die Nutzung von Generatoren ermöglicht Plop die Erstellung von vordefinierten Projektvorlagen. Die Template-Engine wurde in Verbindung mit JavaScript und TypeScript ausprogrammiert. Dadurch entstand die Hauptkomponente beziehungsweise das Herzstück dieser Arbeit. Durch diese Kombination der Technologien war es möglich, die Anforderungen und Erwartungen der Firma Koncept zu erfüllen. Beim Starten der Applikation werden in der Kommandozeile die vordefinierten Fragen gestellt. Im Laufe der Entwicklung wurden gewisse Bereiche sichtbar. Die Hauptapplikation ist in drei Abschnitte unterteilt.

User Prompt Handling

In diesem Abschnitt des Hauptprogramms wurden die Eingabeaufforderungen und die Benutzerinteraktion deklariert. Anhand gezielt festgelegter Fragen für den Benutzer und deren Antworten werden die Projekte erstellt. Um Fehler in den generierten Projekten vorzubeugen, wurden verschiedenste Validierungen für die Antworten festgelegt. Einerseits wird auf die Groß- und Kleinschreibung Acht gegeben. Ein simples Beispiel dazu wäre, dass kein Name der Klassen mit einem Kleinbuchstaben beginnen darf. Andererseits werden keinerlei Sonderzeichen zugelassen. Eine weitere Fehlerbehandlung für die Antworten sind Zahlen in Methoden- oder Klassennamen. Generell dürfen Zahlen in Klassennamen vorkommen, sie dürfen aber nicht mit einer Zahl beginnen.

Der Sinn dieser Überprüfungen ist, dass die generierten Projekte eine einheitliche und standardisierte Namensgebung aufweisen. Dies dient nicht nur der besseren Lesbarkeit und Wartbarkeit, sondern auch der Vorbeugung potenzieller Laufzeitfehler oder Konflikte, die durch inkonsistente Bezeichnungen entstehen können. Um diese Aspekte zu gewährleisten, wurden viele verschiedene stringbasierte Validierungsmechanismen verwendet. Die Kriterien, ob der Klassenname zulässig ist, wurden oben beschrieben.

Um den Projekterstellern ein noch besseres und angenehmeres Gefühl bei der Nutzung der Templates zu geben, wird der Nutzer bei einer falschen Eingabe noch einmal aufgefordert, die Antwort zu überdenken. Zusätzlich wurde eine Funktion zur erneuten Beantwortung einer Frage implementiert. Um diesen Prozess zu ermöglichen, gibt es sogenannte Signalwörter. Eines davon ist das Wort ”back”. Mit diesem Schlagwort ist es dem Nutzer möglich, zu vorherigen Fragen zurückzukehren, falls eine getroffene Auswahl oder Eingabe korrigiert werden möchte. Dadurch wird vermieden, dass der Nutzer den ganzen Prozess von vorne beginnen muss.

Beim Verwenden des Templates werden viele verschiedene Fragen gestellt. Die Eingaben des Nutzers bestimmen dabei Konfigurationen bzw. welche Klassen erstellt werden. Es werden grundlegende, aber auch konkrete Fragen gestellt. Um besser zu verstehen, um welche Fragen es sich handelt, werden die wichtigsten Fragen beschrieben und ihr Zweck dargestellt. Zusätzlich wird auch der Code dahinter beschrieben und seine Eigenschaften dargestellt.

- grundlegende Fragen: Als erstes wird nach dem Projektnamen gefragt. Danach fällt die Entscheidung für die Version des Projektes. Verschiedene .Net Versionen sind auswählbar.

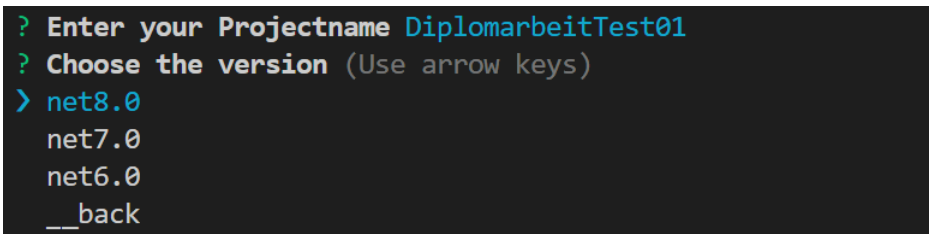


Abbildung 18: Auswahlmöglichkeit der Version

Im Code dahinter werden die Fragen in einem bestimmten Array gespeichert. Die erste Frage beschäftigt sich mit dem Projektnamen. Der Type dieser Frage ist ein ”Input”. Dieser Type beschreibt, dass man auf die Eingabe des Nutzers wartet. Darunter wird ein Variablennamen deklariert, wo die Antwort des Nutzers gespeichert wird. Zusätzlich wird in jeder Frage eine Validierungsmethode hinterlegt, um die Antworten auf ihre Korrektheit

zu überprüfen. Anders als in der ersten Frage, erhält der Nutzer bei der zweiten Frage eine Liste von Möglichkeiten. Hier kann zwischen drei verschieden .Net Versionen ausgewählt werden.

Listing 11: Code für die Basisfragen

```
prompts: async  () => {
            const questions = [
                {
                    type: 'input',
                    name: 'projectName',
                    message: 'Enter your Projectname',
                    validate: validationProjectname
                },
                {
                    type: 'list',
                    name: 'version',
                    message: 'Choose the version',
                    choices: [
                        { name: "net8.0", value: "net8.0" },
                        { name: "net7.0", value: "net7.0" },
                        { name: "net6.0", value: "net6.0" },
                        "__back"
                    ]
                },
            ]
    }
```

Zusätzlich wird nach der Solution, wo das Projekt hinzugefügt werden soll, gefragt. Um Fehler zu vermeiden, wird hier auf die Korrektheit und die Existenz der Solution Wert gelegt. Für diese Überprüfung verwendet das System einige Validierungsmethoden. Danach wird gefragt, wo das Projekt gespeichert werden soll. In der Koncept Ordnerstruktur gibt es für jede Solution explizite Dateispeicherorte. Das System erkennt alle Speicherorte und basierend auf der Solution, die vorher eingegeben wurde, schlägt das System den passenden Speicherort vor. Wenn der Nutzer mit diesem Vorschlag zufrieden ist, wird dies mit Enter bestätigt. Anderenfalls kann nach eigenen Vorstellungen eine individuelle Lösung angegeben werden. Dieser Vorschlag dient als Hilfestellung und ist kein Zwang.

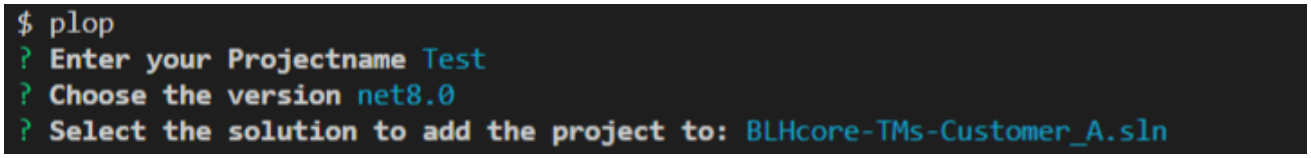


Abbildung 19: Beispiele für die Basisfragen

Eine Solution ist nichts anderes als eine Art Schublade voller Projekte. Projekte, die in einer Solution sind, können gemeinsam agieren und Klassen untereinander austauschen bzw. miteinander kommunizieren. Außerdem ist es möglich, Pakete in allen Projekten in dieser Schublade einfach zu implementieren und zu installieren.

- Inputs und Outputs: Im nächsten Schritt wird nach Inputs und Outputs gefragt. Die Inputs und Outputs dienen als Schnittstelle zu anderen Projekten und Paketen von Koncept. Soll

diese Schnittstellen in einem Projekt erstellt werden, muss lediglich der Name angegeben werden.

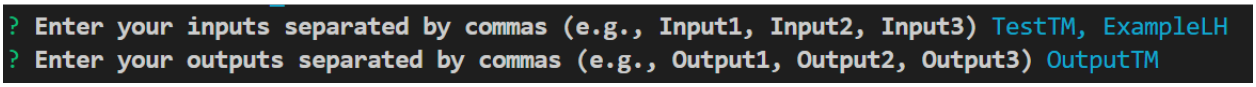


Abbildung 20: Beispiele für die Namensgebung

Durch die Angabe der Schnittstellen werden bestimmte Methoden in dem generierten Projekt erstellt. Die Methoden sind direkt mit den Eingaben des Nutzers verbunden und sorgen dafür, dass das Projekt die gewünschten Funktionen aufweist. Bei dem Generierungsvorgang werden die gewünschten Klassen automatisch erstellt, in denen die Methoden auffindbar sind.

- Konfigurationen: Die nächste Frage beschäftigt sich mit ganz bestimmten Konfigurationen, die in den Klassen eingerichtet werden. In den Hauptklassen, die durch das Template generiert werden, werden bestimmte Konstruktoren erstellt. In diesen Methoden werden bestimmte Variablen und Einstellungen gespeichert. Durch die Konfigurationsfrage des Templates kann der Nutzer diese Variablen erstellen lassen. Beispiele dafür wären Connections-Strings zu Datenbanken, API-Endpunkte zu anderen Projekten, Authentifizierungs- und Autorisierungseinstellungen und Logging- und Monitoring-Einstellungen.

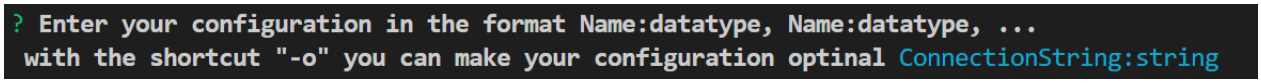


Abbildung 21: Beispiele für die Konfigurationen

Um diese Konfigurationen einzurichten, muss der Name und der Type der Variable angegeben werden. Leerzeichen dürfen dabei nicht gesetzt werden. Name und Datentyp werden mittels eines Doppelpunktes getrennt. Natürlich ist es auch möglich mehrere Einstellungen zu erstellen. Darüber hinaus kann mit dem Signalwort ”-o”, der Konfigurationsvariable eine Option zugewiesen werden, die als Standardwert oder zusätzliche Eigenschaft dient. Durch diese Erweiterung ist es möglich, bestimmte Werte oder Erweiterungen der Konfigurationsvariablen zu aktivieren.

- Optional Points: Die letzte und komplexeste Frage beschäftigt sich mit optionalen Features, die in den generierten Klassen eingefügt werden. Der Nutzer des Templates hat die Möglichkeit, aus drei verschiedenen Punkten auszuwählen.

Mit der ersten Option werden verschiedene Interfaces im Projekt erstellt. In diesen Interfaces befinden sich viele vordefinierte Methoden und Variablen. Durch diese Interfaces ist es möglich, in seinem Projekt noch mehr Struktur und Flexibilität des Codes einzubauen. Zusätzlich wird der Programmcode mit Interfaces wesentlich wartbarer. Durch diese Erweiterung ist es für den Entwickler deutlich einfacher, neue Methoden zu definieren und zu deklarieren.

Durch die zweite Option werden in verschiedenen Teilen des generierten Projektes Methoden für Fehlerbehandlungen implementiert. Diese besondere Form von Methoden können vom Nutzer überarbeitet oder erweitert werden. Durch diese Erweiterung des Codes können individuelle Anforderungen und Fehlerbehandlungsmethoden implementiert werden, ohne dass die Grundstruktur verändert werden muss. Diese spezielle Strategie dient als Fehlerhandler und wird automatisch in mehreren Teilen des Programms integriert.

Die letzte Variante der Optional Points macht es möglich, die oben bereits beschriebenen Inputs beziehungsweise Schnittstellen zu anderen Projekten in separate Klassen auszulagern. Diese Möglichkeit wird empfohlen, wenn die Schnittstelle zu einem anderen Projekt komplex oder aufwendig zu managen ist. Durch diese Option ist es den Entwicklern möglich, beliebig viele Methoden und Konfigurationen zu erstellen, ohne die Lesbarkeit oder Wartbarkeit der Klasse zu gefährden.

Klassendeklaration

Im zweiten Abschnitt des Hauptprogramms werden die Einstellungen für die Klassengenerierung erstellt. Die notwendigen Parameter für die automatische Klassenerstellung werden in diesem Segment definiert. Durch diese Konfiguration weiß die Technologie Plop sowohl welche Klassen als auch mit welchen Eigenschaften diese erstellt werden müssen. Wie bereits in anderen Kapiteln erwähnt, wird der Klasseninhalt durch dynamische Benutzereingaben bestimmt.

Listing 12: Konfigurationen für eine Klasse

```
actions = [
        {
            type: 'add',
            path: `${filepath}/${projectName}/${projectName}.cs`,
            templateFile: 'plop-templates/BlhTmTemplateTest.cs',
            data: {
                projectName: projectName,
                additionalNames: additionalNames,
                projectNameConfig: projectNameConfig,
                outputsName: outputs,
                configurationNames: configurationNames,
                dataInterface: dataInterface,
                rethrow: rethrow,
                separateInputs: separateInputs
            },
        }
]
```

Methoden

Der letzte Abschnitt beschäftigt sich mit einer Vielfalt von Methoden. Einige von ihnen sind Validierungsmethoden. In diesen werden Fehlerbehandlungen implementiert, welche im darüber liegenden Code verwendet werden. Ein simples Beispiel wären Validierungsmethoden für die Benutzereingabe. Diese Methoden geben vor, welche Eingaben akzeptiert und welche verworfen werden. Auch managen sie die Datenaufbereitung für andere Prozesse. Viele Eingaben des Nutzers müssen zuerst umformatiert werden, um sie zu verarbeiten. Natürlich werden auch diese Umformungen in den Methoden gehalten.

Listing 13: Beispiel für eine einfache Validierungsmethode

```
function validationProjectname(value) {
        if (value === "") {
            return 'You have to give your project a name'
        }
        if (value && /[!-/:-@[-`{-~]/.test(value.split('')[0])){
            return 'No symbols are allowed as first letter';
        }
        if (!isNaN(value.charAt(0)) || value.charAt(0) !== value.charAt(0).toUpperCase()) {
            return 'Each input must start with an uppercase letter and not begin with a
                number.';
        }
        return true;
    }
```

Darüber hinaus gibt es Methoden, die für die Modularisierung und Wiederverwendbarkeit des Codes sorgen. Ein Beispiel dafür wären Fehlermeldungsmethoden. Diese sorgen dafür, dass die Log-Meldungen im ganzen Programm gleich aussehen.

Listing 14: Exaptionmethode

```
function logError(message, error) {
    console.error(`[Error] ${message}:`, error);
}
```

6.1.2 JSON Data

Während der Entwicklung des Templates entstand eine essenzielle Erweiterung des Templates. Eine neue Funktion wurde hinzugefügt. Nach jeder Verwendung des Templates können die Eingaben der Nutzer gespeichert werden. Die Antworten auf die Templatefragen werden in einem besonderen Format in einer JSON-Datei hinterlegt. Das System wurde so überarbeitet, dass Projekte einerseits mit dem ganz normalen Frageprozess erstellt werden, andererseits ist es möglich, Projekte mit einer JSON-Datei statt der Fragen zu erstellen. Alle Einstellungen werden in einer Datei notiert. Basierend auf dieser Datei und ihrem Inhalt kann ein Projekt erstellt werden.

Listing 15: Exaptionmethode

```
{
  "projectName": "BLHExampleProject",
  "version": "net8.0",
  "solution": "BLHcore-TMs-Customer_A.sln",
  "filepath": "TM/Custom/Customer_A",
  "inputs": "MMSInput, BLcore",
  "outputs": "MMSOutpur",
  "configuration": "ConnectionMSSQL:string, ConnectionMongo:string",
  "comment": [
    "1", "Add dataInterface support?",
    "2", "Rethrow exceptions without output to Unprocessable?",
    "3", "Put Inputs into separate files using partial classes?"
  ],
  "optionalPoints": [
    "1",
    "2",
    "3"
  ],
  "openProject": "true"
}
```

Dadurch ist es möglich, dasselbe Projekt mit denselben Einstellungen, Konfigurationen und Schnittstellen noch einmal zu erstellen. Dies ist besonders nützlich, wenn mehrere Projekte mit ähnlichen Strukturen oder wiederkehrenden Konfigurationen erstellt werden sollen.

Durch diese Erweiterung werden alle Konfigurationen in externen Dateien gespeichert. So werden alle Projekte exakt dokumentiert und gespeichert. Eine genaue Bestimmung, wer welche Projekte erstellt hat und wann sie generiert wurden, ist dadurch gegeben. Mit dieser Möglichkeit wird vermieden, dass wichtige Parameter, wie Variablen, Schnittstellen und Methoden erneut manuell eingegeben werden müssen. Dies spart Zeit ein und beugt Fehlereingaben vor.

Zusätzlich wurde eine Art automatische Ladefunktion implementiert. Diese neue Funktion erkennt, ob die Projekteinstellungen bereits existieren. Falls eine solche Übereinstimmung zutrifft, wird gefragt, ob der Nutzer die frühere Konfiguration wiederverwenden möchte. Bei einer Zustimmung gibt es abermals die Möglichkeit, alle Einstellungen, Variablen und Schnittstellen zu überprüfen, um sicher zu stellen, dass diese Konfigurationen auch wirklich allen Ansprüchen gerecht wird. Nach dieser Bestätigung wird das Projekt erstellt.

6.1.3 JSON-Function

Im obigen Kapitel wurden die Vorteile und Eigenschaften der Auslagerung der Eigenschaften in JSON-Daten beschrieben. In diesem Abschnitt werden die technische Umsetzung und die verwendeten Technologien beleuchtet. Zusätzliche Codebeispiele lassen das Gelesene noch besser und klarer verstehen.

Bevor auf den Code genauer eingegangen wird, müssen die Einstellungen beziehungsweise die Pakete, die dafür gebraucht werden, erklärt werden. Es gibt in allen Node.js-Projekten eine zentrale Konfigurationsdatei. Auch diese Arbeit besitzt diese besondere Datei. Darin werden alle Metadaten und Einstellungen definiert. Sie beschreibt das Projekt und ihre Vielzahl an Paketen. Darüber hinaus bestimmt diese, wie die gesamte Plop-Anwendung gestartet und gebaut wird. Veränderungen in dieser Datei waren notwendig, um die Effizienz des Templates zu gewährleisten.

Listing 16: Exaptionmethode

```
1  "scripts": {
2      "plop": "npx plop",
3      "auto-plop": "node run-plopfile.js"
4    },
```

Durch diese Anpassung ist es nun möglich, das Template mit ”npm run auto-plop” zu starten. Zusätzlich zum Startbefehl muss der Dateipfad zu der JSON-Datei angegeben werden. Auf die komplette Eingabe folgt das Generieren des Projektes.

Nach dem Start beginnen die eigentlichen Schritte des Templates. Als allererstes wird mithilfe verschiedenster Methoden überprüft, ob der Nutzer überhaupt einen Pfad zu einer JSON-Datei angegeben hat. Diese Methoden wurden selbst implementiert und geschrieben. Wenn diese Fehlerbehandlung erfolgreich überwunden wird, beginnt der nächste Schritt. In diesem Teil wird die Existenz der Datei geprüft. Falls ein falscher Pfad angegeben wird, kümmert sich eine Exceptionmethode um eine saubere Fehlermeldung für den Nutzer. Nach der Existenzprüfung wird die JSON-Datei genauer untersucht. Das System öffnet die Datei und prüft, ob das Format des Dokuments auch korrekt ist. Falls dies nicht der Fall sein sollte, erzeugt das Programm eine angemessene und passende Fehlermeldung. Nach diesem Schritt erfolgt der eigentliche Prozess der Projekterstellung.

Der Prozess startet mit dem Auslesen der angegebenen Datei. Danach werden die Daten so aufbereitet, dass der Plop-Generator damit arbeiten kann. Der Generator hat die Funktion, aus diesem Dateiinhalt ein Projekt zu erstellen.

Listing 17: Prozzes der JSON-basierten Systems

```
1       // Load answers from JSON file
2       const answers = JSON.parse(fs.readFileSync(jsonFilePath, 'utf8'));
3
4       // Get the generator
5       const generator = plopInstance.getGenerator('component');
6
7       // Run the generator with answers
8       const result = await generator.runActions(answers);
```

Bevor der Generator ein Projekt generiert, werden noch einige Schritte durchlaufen. Der Generator selber muss sich versichern, dass alle Parameter auch wirklich vorhanden sind, die er braucht. Deshalb leitet er eine Reihe von Überprüfungen ein. Einerseits schaut er, ob das Format angemessen ist, andererseits wird überprüft, ob null oder undefined-Parameter übergeben worden sind.

Beispiel

Es wird eine JSON-Datei übergeben, in der das Format passt. Das setzt aber nicht voraus, dass alle Parameter einen gültigen Wert haben. Ist der Name des Projekts nicht lesbar oder auf null oder undefined gesetzt, kann das Projekt nicht generiert werden. Bei solch einem Fehler werden verschiedene Fehlermeldungen ausgegeben.

Natürlich werden nicht nur die Werte der Parameter überprüft, sondern auch ihre Datentypen. Wird bei einem Parameter ein Integer erwartet, aber ein String übergeben, kann das Projekt nicht erfolgreich erstellt werden. Auch hier erscheint eine Meldung.

Nach all den Validierungsmechanismen vom Generator wird das Projekt erstellt. Das System greift auf das Hauptprogramm zu und startet es. Nach dem Start werden alle Fragen basierend auf der korrekten JSON-Datei beantwortet. Dieser Schritt wird automatisch ausgeführt, ohne dass der Nutzer etwas davon erfährt. Anschließend wird das Projekt generiert.

6.1.4 PowerShell-Skript

Nach der fertigen Entwicklung der oben beschriebenen Applikationen und Programme gab es noch ein Problem zu lösen. Das Projekt muss immer manuell gestartet werden. Dieser große Nachteil erschwert es dem Nutzer, das Projekt zu erstellen, da bei jedem neuen Code-Template ein manueller Befehl namens ”plop” über die Konsole eingegeben und ausgeführt werden muss. Zusätzlich muss die Konsole in dem richtigen Verzeichnis geöffnet

Abbildung 22: PowerShell

werden, um überhaupt das Template mit dem Befehl ”plop” zu
starten. Daher ergibt sich eine Abhängigkeit der Ordnerstruktur.

Nach einigen Besprechungen und Diskussionen mit der Firma Koncept kam man zu der Lösung, ein PowerShell-Skript dafür einzusetzen. Anstatt ”plop” in der Konsole einzugeben, ermöglicht dieses Skript eine universelle Nutzung des Templates. Mit dem Start des PowerShell-Skripts kann ein Projekt nach eigenen Vorstellungen erstellt werden.

Das Ausführen des Skripts zieht die Fragestellung nach sich, ob bereits eine JSON-Datei besteht und das Projekt basierend auf dieser Datei erstellt werden soll oder ob alle Fragen manuell beantwortet werden. Nach der Beantwortung werden die bereits oben benannten Prozesse geladen und die Eingabe der Parameter erfolgt logischerweise über den Powershell-Prozess. Das PowerShell-Skript hat als einfache Aufgabe, die ganzen Programmteile miteinander zu verbinden. So dient das Skript aus der Sicht des Nutzers als Frontend und der eigentliche Template-Code als Backend. Zusätzlich wird durch diese übersichtliche Struktur die Skalierbarkeit der kompletten Template verbessert, da die Zuständigkeit des Prozesses klar erkennbar ist.

Nach der erfolgreichen Entwicklung des Skriptes wurden viele verschiedene Tests durchgeführt, um sicherzustellen, dass das Powershell-Script wirklich universell ausführbar ist. Die Tests bestätigten die perfekte Entwicklung des Skripts. Mit diesem Tool werden die Abhängigkeiten des Verzeichnisses erfolgreich gelöst.

6.1.5 User Interface

Auch wenn das PowerShell-Skript einen Teil des Problems gelöst hatte, gab es leider noch keine saubere Benutzeroberfläche. Seitens der Firma Koncept war dieses ”Problem” relativ irrelevant. Die Mitarbeiter der Firma Koncept, welche das Template nutzen, waren mit dem Powershell-Skript zufrieden. Trotzdem wurde vereinbart, eine Benutzeroberfläche zu entwickeln, um nach der Entwicklung die beiden User-Interfaces miteinander zu vergleichen.

Die neue Benutzeroberfläche wurde mit Csharp in Visual Studio programmiert. Nach einigen Recherchen und Absprachen der Mitarbeiter entschied man sich für eine WPF-Applikation. Windows Presentation Foundation (WPF) ist ein UI-Framework von Microsoft, das für grafische Benutzeroberflächen genutzt wird. WPF basiert auf mehreren XAML-Dateien. Diese werden zur Definition des User-Interface verwendet. Der Code dahinter wird meistens in Csharp programmiert. Das heißt, die WPF-Anwendung muss eine Kombination aus einer neuen Technologie, nämlich Csharp und der Plop Template-Engine ermöglichen.

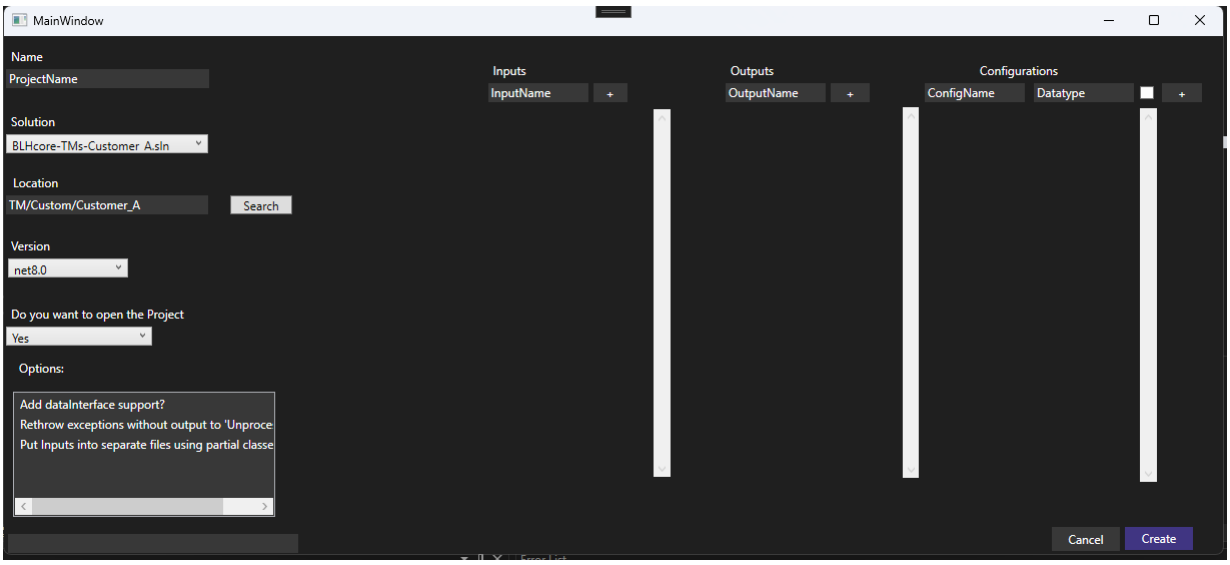


Abbildung 23: GUI

Das Ausprobieren verschiedener Designs sollte die Benutzerfreundlichkeit gewährleisten. Zuerst wurde ein Grundriss gezeichnet. Diese Grundidee des Designs wurde im Laufe der Zeit immer wieder verändert.

Nach der Fertigstellung des Designs folgte die Programmierung der Logik im Hintergrund. Der Mechanismus, einen Projektnamen festzustellen, ist ein simples Eingabefeld. Bei der Festlegung, wo das Projekt gespeichert werden soll, öffnet sich ein Windows-Explorer, in dem genau definiert werden kann, in welchem Pfad oder Ordner das Projekt gespeichert wird. Eine vordefinierte Liste lässt den Nutzer frei zwischen den Versionen entscheiden. Der Standardwert liegt aber bei Version .NET 8. Darüber hinaus wurden auch Eingaben für die Schnittstellen zu anderen Projekten errichtet. Diese sind unter den Namen Inputs und Outputs bekannt. Eine Werteliste macht die Schnittstellen nach dem Hinzufügen erkennbar. Dies erfolgt über den Plus-Button.

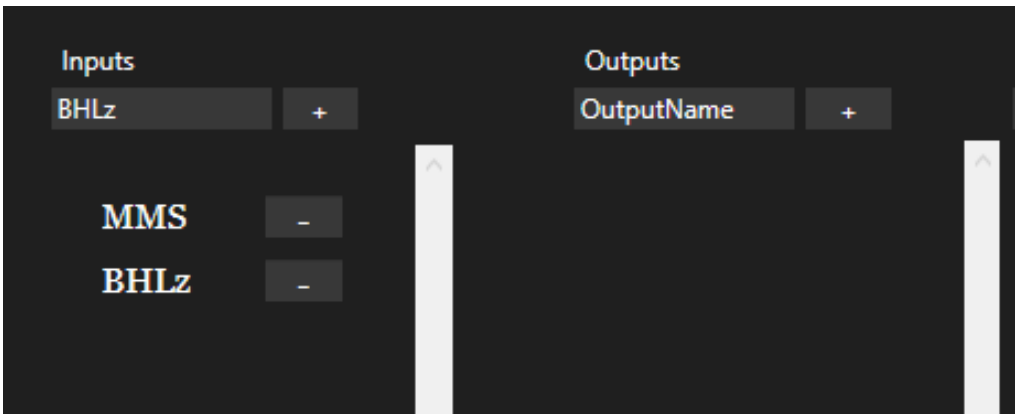


Abbildung 24: Beispiele für die Schnittstellen in der GUI

Ein hinzugefügter Delete-Button ermöglicht es, die eingegebenen Inputs oder Outputs zu entfernen. Damit kann ein Tippfehler schnell und bequem ausgebessert werden.

Ein in der GUI konfigurierbarer Parameter steht noch auf der Liste. Dies sind die Konfigurationsvariablen. Mithilfe von zwei Eingabefeldern kann einerseits der Name deklariert und andererseits mit dem zweiten Feld der Datentyp angegeben werden. Viele verschiedene Validierungsmethoden (siehe: Validierungen) verhindern auch hier, die einheitliche Namensgebung zu gefährden.

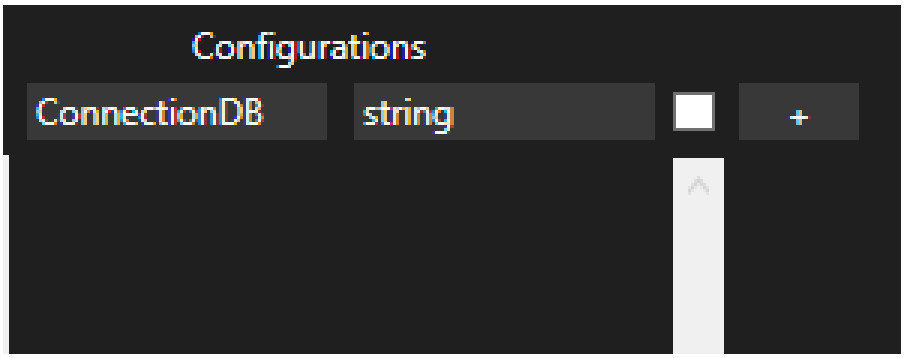


Abbildung 25: Beispiele für die Schnittstellen in der GUI

Die WPF-Anwendung agiert und basiert auf demselben Prinzip wie das PowerShell-Script. Durch das WPF werden alle Prozesse miteinander verbunden. Das WPF agiert nur als Frontend, also als die Oberfläche für den Nutzer. Das Frontend sammelt die Parameter, die eingegeben werden, und sendet sie zum Backend. Das Backend in diesem Fall sind die JavaScript-Anwendungen , die mit der Template-Engine Plop verbunden sind.

6.2 Intermediate Format Specification (IFS)

Das Generieren der Klassen erfolgt durch Generatoren, die auf IFS-Klassen zugreifen. Der Code der Klassen wird in Dateien hinterlegt, wobei das Hinzufügen verschiedenster Überprüfungen und Methoden möglich ist. Welcher Code generiert werden soll, wird in Wenn-Abfragen und Schleifen bestimmt. Die individuellen Benutzereingaben sendet der Generator zu den IFS-Klassen weiter. Darauf basierend werden die Klassen generiert.

Diese Technologie beinhaltet sogenannte Platzhalter, mit denen dynamische Namen zum Code hinzugefügt werden können. Diese Platzhalter werden bei der Generierung mit der Benutzereingabe überschrieben. So kann dynamisch und effizient auf die Bedürfnisse der Nutzer und auf ihre Namensgebung eingegangen werden.

Listing 18: IFS Example

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>{{version}}</TargetFramework>
    <RootNamespace>Koncept.MMS.BusinessLogicHost.TM.{{projectName}}</RootNamespace>
    <AssemblyName>{{projectName}}</AssemblyName>
    <Copyright>Copyright 2024 by Kon-Cept Management Information Services GmbH</Copyright>
    <Company>Kon-Cept Management Information Services GmbH</Company>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <Authors />
  </PropertyGroup>
  <ItemGroup>
  {{#if (eq dataInterface true)}}
    <ProjectReference
        Include="..\..\..\..\Common\MMS.DataInterface.Core\MMS.DataInterface.Core.csproj"
        />
  {{/if}}
    <ProjectReference
        Include="..\..\..\..\Manager\Messages\BLH.Messages.Impl\BLH.Messages.Impl.csproj"
        />
    <ProjectReference Include="..\..\..\BaseTM.Impl\BaseTM.Impl.csproj" />
  </ItemGroup>

</Project>
```

Hier ist ein exzellentes Beispiel für solch Intermediate Format Specification. Der Code zeigt eine Projektdatei für ein .NET-Projekt mit Platzhaltern und Bedingungsausdrücken. Die Platzhalter werden per geschwungener Klammer deklariert, die jene Variablen enthalten, welche beim Generieren ausgetauscht werden. Weiteres sind auch Bedingungsausdrücke und Abfragen vorhanden. Darin wird überprüft, ob eine bestimmte Variable einen bestimmten Wert aufweist. Auch Schleifen können in das Programm integriert werden. Durch ihre Verwendung in solchen IFS-Klassen ist es möglich, Methoden oder Variablen mit leichter Veränderung mehrmals zu erzeugen. Auch das Verwenden mehrerer IFS-Klassen bietet sich an. Im Rahmen dieser Arbeit wurden viele verschiedene Arten dieser besonderen Klasse angewandt.

6.3 UML Komponetendiagramm

Alle Komponenten dieser Arbeit wurden bereits beschrieben. Ein Diagramm, welches alle Komponenten zeigt und ihre Verbindungen miteinander definiert, schafft Klarheit. Eindeutig erkennbar ist das Herzstück dieser Arbeit, die Mainapplikation. Diese ist mit jedem anderen Teil verbunden. Unter anderem liegt der Zuständigkeitsbereich in der Kommunikation zwischen den einzelnen Prozessen. Verwendet eine Frontend-Applikation die Methoden und Funktionen der Nebenapplikation, ist ebenfalls das Herzstück für diesen Datenaustausch verantwortlich.

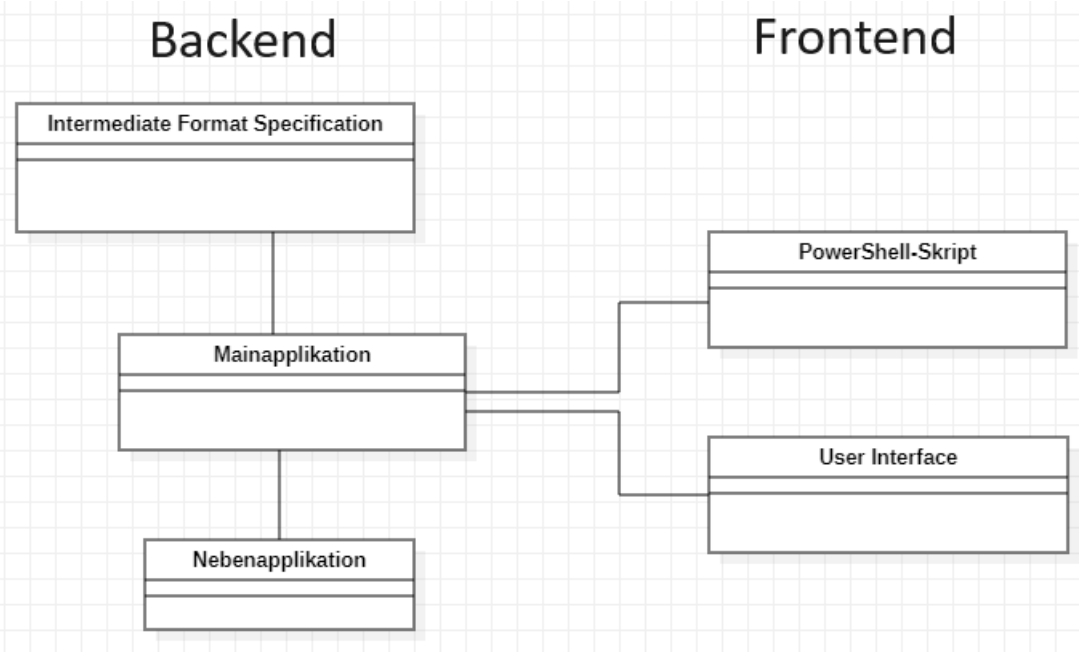


Abbildung 26: Zusammenhänge aller Komponenten

6.3.1 Backend

Das Backend besteht aus drei wesentlichen Bereichen, nämlich der Mainapplikation, der Nebenapplikation und der Intermediate Format Specification. Die Nebenapplikation, auch unter dem Kapitel JSON-Funktion beschrieben, ist eine Erweiterung zur Mainapplikation. (siehe oben: Mainapplikation).

6.3.2 Frontend

Auch das Frontend kann man in zwei Abschnitte unterteilen, nämlich das PowerShell-Skript und das User-Interface.

6.4 Organisation

6.4.1 Vorgehensmodell

Um diese Diplomarbeit bestmöglich umzusetzen, wurde ein Vorgehensmodell verwendet. Dieses Modell unterstützte die Entwicklung und den Fortschritt dieser Arbeit. Während der Entwicklung des Templates kam das Vorgehensmodell Scrum zum Einsatz. Damit wurden täglich Meetings mit den Mitarbeitern der Firma Koncept abgehalten. Im Scrum-Vorgehensmodell gibt es verschiedene Schritte, die eingehalten werden müssen.

Beispiel

- Sprint Planning: In dieser Phase wurden gemeinsam mit den Mitarbeitern der Firma Koncept Ziele und Vereinbarungen getroffen. In der Fachsprache wird dies als Meeting mit dem Product Owner bezeichnet. Alles wurde dokumentiert. In diesen Treffen wurde außerdem eine Aufwandsschätzung vorgenommen, um zu bestimmen, wie viel Zeit für das Erreichen des Ziels benötigt wird.
- Sprint: In diesem Schritt beginnt die eigentliche Programmierung und Entwicklung. Zusätzlich werden Daily Scrums abgehalten. Diese Meetings dienen dazu, offene Fragen zu klären und kurze Präsentationen durchzuführen, um den aktuellen Stand des Ziels zu definieren.
- Sprint Review: In diesem Abschnitt werden die erreichten Ergebnisse präsentiert. Der Product Owner beurteilt, ob die Umsetzung erfolgreich war. Zudem wird über das Design, die Performance und die Funktionalität gesprochen. Auch Verbesserungsvorschläge werden diskutiert – zum Beispiel, ob Funktionen fehlen, die für die Nutzer hilfreich wären, oder ob andere Aspekte unzureichend umgesetzt wurden.
- Sprint Retrospektive: Nach dem Meeting besprechen sich die Entwickler untereinander, was im Sprint gut lief und was verbessert werden sollte. Es wird nach möglichen Prozessanpassungen gesucht und diese werden umgesetzt. Dieser Schritt ist essenziell, da eine Verbesserung des Vorgehensmodells dazu führt, dass das Projekt noch effizienter und erfolgreicher umgesetzt werden kann.

Nach der Sprint-Retrospektive beginnt ein neuer Sprint. Alle weiteren Prozesse wiederholen sich und starten erneut.

6.4.2 Dokumentation

Alle Schritte dieses Vorgehensmodells wurden reichlich dokumentiert. Natürlich ist ein Ausschnitt aus der Doku auch ersichtlich. Im Anhang befindet sich ein Musterbeispiel, wie die Dokumentation der einzelnen Schritte aussehen kann. Daraus können viele verschiedene Texte, die farbig markiert sind, entnommen werden. Die dabei verwendeten Farben haben folgende Bedeutung:

- Gelb: Alle gelben Markierungen, sind Aufgaben, die noch erledigt werden müssen.
- Grün: Aufgaben, die schon gelöst sind. Auch der Lösungsweg wird beschrieben und ebenfalls mit Grün gekennzeichnet.
- Rot: Alle nicht erreichten Ziele und Aufgaben, die für den nächsten Tag oder die nächste Phase wichtig sind, werden mit Rot gekennzeichnet.

6.5 Sonstige Technologien

Zusätzlich zu den bereits beschriebenen Technologien kamen bei der Erstellung der Arbeit noch folgende Technologien zum Einsatz.

- **Teams:** Teams wurde für die Kommunikation mit der Firma Koncept verwendet. Über diese Plattform fand der Austausch von Informationen und Aufgaben statt. Auch zahlreiche Meetings wurden per Teams abgehalten.
- **Excel:** Um die Zeitaufzeichnung so genau wie möglich mitzuführen, wurde Excel verwendet. Die täglichen Arbeitszeiten an dieser Arbeit wurden mitgeschrieben. Dies war nicht nur wichtig für die Arbeit selbst, sondern auch relevant für die Firma Koncept. So konnten die Arbeitszeit, die Ressourcenplanung und die Projektabrechnung genau definiert werden.
- **OneNote:** Die Dokumentation selbst wurde in OneNote erstellt. Für jeden Tag wurde eine eigene Seite erstellt und mit einer Beschreibung von den Aufgaben und Erkenntnissen, die gemeistert oder gewonnen wurden, ergänzt.
- **Visual Studio und Visual Studio Code:** Die Entwicklung und die Tests der technischen Technologien wurden in Visual Studio Code und in Visual Studio unternommen. Diese beiden Plattformen bieten eine benutzerfreundliche Interaktion mit dem Programmcode.

7 Ergebnis

In diesem Kapitel wird das eigentliche Ergebnis präsentiert. Darüber hinaus werden noch einmal alle Technologien erwähnt, die wirklich Verwendung fanden. Auch die wichtigsten Ideen und Überlegungen werden angesprochen. Das Ende des Kapitels beschreibt die schwierigsten Herausforderungen und deren Überwindung. In diesem Abschnitt wird nur das Ergebnis präsentiert und nicht der technische Hintergrund. (siehe oben: Implementierung)

7.1 Frontend Ergebnis

Im Rahmen dieser Arbeit entstanden zwei verschiedene Benutzeroberflächen, die vom Nutzer bedient werden können.

7.1.1 User Interface (UI)

Bei dieser Benutzeroberfläche steht das Design und die Anordnung der einzelnen Komponenten im Fokus. Die Oberfläche zeichnet sich durch ihre klare Struktur aus. Darüber hinaus wurde bei dem User Interface auf moderne und benutzerfreundliche Optik geachtet. Zusätzlich wurden viele verschiedene Designarten und Versionen getestet. Daher entschied man sich bewusst für ein Visual Studio-ähnliches Design, da in der Firma Koncept Visual Studio ziemlich oft zum Programmieren verwendet wird. Der Nutzer soll dadurch ein bereits ihm vertrautes Gefühl dieser Umgebung bekommen. Bei dem Design war es wichtig, alle Anforderungen der Firma Koncept zu erfüllen und es so einfach wie möglich zu halten. Das User-Interface soll nicht zu verschachtelt oder zu kompliziert aufgebaut sein. Bei einem zu komplexen Aufbau kann es passieren, dass sich der Nutzer überfordert fühlt. Farblich wurde auf ein dezentes, einheitliches Design mit kontrastreichen Akzenten geachtet. Mit diesem Format wird einerseits die gute Lesbarkeit und andererseits die Orientierung im Template-Design gewährleistet.

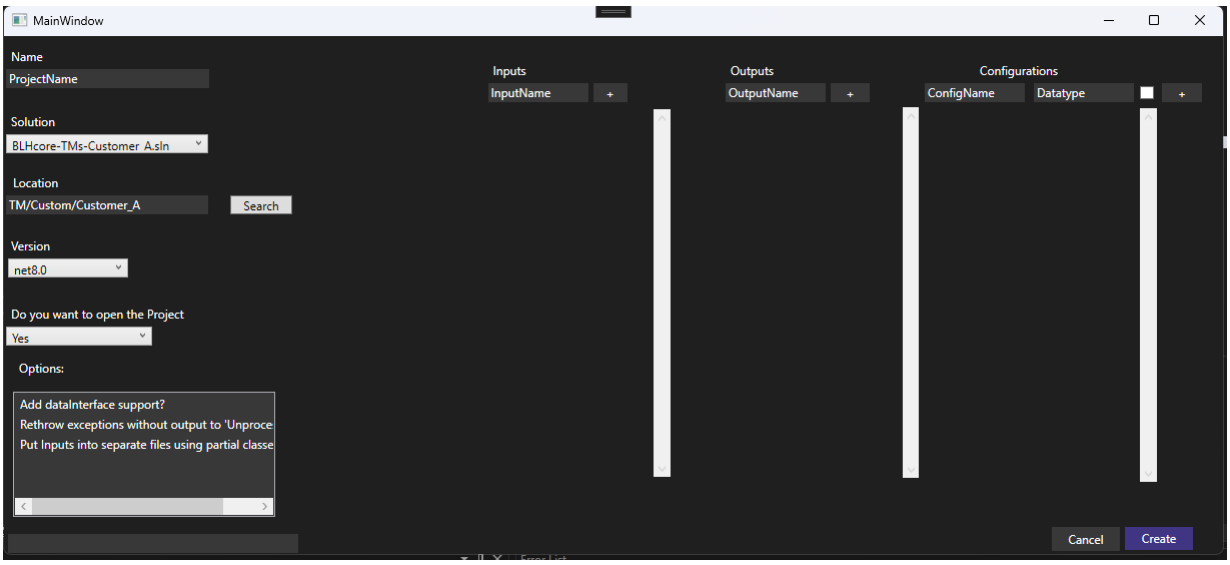


Abbildung 27: GUI

Am linken Rand des Fensters müssen alle Basisfragen beantwortet werden. Hier wird der Projektname, die Projektversion, der Speicherort, zu welcher Solution es hinzugefügt beziehungsweise verlinkt wird, und ob das Projekt nach dem Erstellen gleich geöffnet werden soll, angegeben. In der Mitte des Fensters bis zum linken Rand können weitere Einstellungen konfiguriert werden, welche in drei Bereiche unterteilbar sind. In den ersten zwei Bereichen findet sich ein Textfeld mit einem Add-Button. Der definierte Name im Textfeld wird hier mit dem Knopf zur Einstellungsliste hinzugefügt. Im dritten Bereich gibt es zwei verschiedene Textfelder. Das erste ist ebenfalls für den Namen der Konfiguration in den Klassen zuständig. Im zweiten Feld werden die dazugehörigen Datentypen definiert. Wie bei den anderen Bereichen kann mit dem Add-Button zur Konfigurationsliste hinzugefügt werden. Als guter Letzt wird durch das Drücken ”Create” rechts unten das Projekt erstellt.

7.1.2 Powershell-Skript

Neben dem User-Interface wurde auch ein Powershell-Skript erstellt. (Siehe: Skript). Bei diesem Skript stand nicht das Design, sondern die Funktionalität im Vordergrund. Powershell-Skript ermöglicht es, universell ein Projekt zu erstellen, ohne von der Ordnerstruktur abhängig zu sein. Das Skript funktioniert wie eine Konsolen-Applikation. Es werden nacheinander Fragen gestellt, um alle Informationen und Wünsche der Nutzer zu sammeln.

Während des Prozesses wird dem Nutzer durch eine Schritt-für-Schritt-Anleitung geholfen. Auch hier stellt sich wieder die Frage nach Projektnamen, Version und anderen Einstellungen. Am Ende des Prozesses wird das passende Projekt generiert und der Nutzer erhält die Option, das

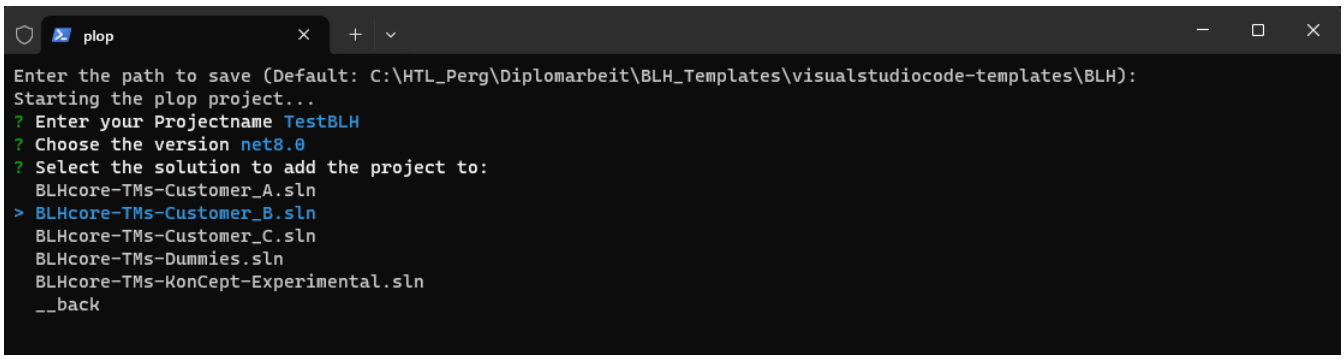


Abbildung 28: gestartetes Powershell-Skript

Projekt nach der Erzeugung in Visual Studio zu öffnen. Das direkte Öffnen des generierten Projektes spart wertvolle Zeit, da sofort mit dem Programmieren gestartet werden kann.

7.2 Backend Ergebnis

Im Rahmen dieser Arbeit entstanden zwei verschiedene Backends, wobei eines der beiden Teile nur eine Erweiterung ist. In diesem Kapitel werden die Hauptfunktionen allgemein erklärt. Der Code oder die technische Umsetzung wird dabei nicht berücksichtigt. (Siehe oben: Implementierung)

7.2.1 Hauptapplikation

In der Hauptapplikation sind alle Hauptfunktionen deklariert und ausprogrammiert. Alle Benutzeroberflächen müssen mit der Hauptapplikation des Backends verbunden sein. Durch dieses Programm ist es möglich, Projekte nach Belieben zu erstellen. Die Hauptapplikation bildet somit nicht nur das Herzstück der kompletten Anwendung, sondern ist auch für die Kommunikation zwischen dem User Interface und dem PowerShell-Skript verantwortlich. Sie beinhaltet die gesamte Logik und den zentralen Kern für die Generatoren. Durch diese Generatoren, welche die Technologie zur Verfügung stellt, ist es überhaupt möglich, Ordnerstrukturen und Projekte zu erstellen (siehe oben: Plop).

Eine Reihe unterschiedlicher Fragen wird gestellt. Zunächst grundlegende Fragen zum Projekt, zum Beispiel: Projektname, Version, Speicherort, Solution-Verzeichnis. Anschließend folgen weitere Fragen, deren Antworten für die Einstellungen oder Methodennamen des Projekts verwendet werden.

Der Code und die Applikation zeichnen sich nicht nur durch ihre Funktionalität aus, sondern auch durch ihre Skalierbarkeit. Durch die Skalierbarkeit der Applikation können in Zukunft

weitere Fragen und Methoden einfach in den Code integriert werden. Dieser Aspekt ist ein entscheidender Faktor für das Ergebnis dieses Abschnitts.

Außerdem übernimmt diese Applikation eine Reihe verschiedenster Validierungsmechanismen. Diese sind für die Korrektheit der Eingaben des Nutzers zuständig. Alle Eingabeparameter werden auf ihre Korrektheit getestet. Entspricht ein Wert nicht den Normen, wird der Nutzer aufgefordert, seine Eingabe zu überdenken.

7.2.2 Nebenapplikation

Die Nebenapplikation ist eine Erweiterung der Hauptapplikation und bietet neue Möglichkeiten und Funktionen an. Dieses Programm ermöglicht es, aus einer JSON-Datei mit einem bestimmten Format ein Projekt zu erstellen. Zeit, die normalerweise für das Fragenbeantworten anfällt, wird hiermit eingespart. Mit diesem Prozess werden alle Namen, Methoden und Einstellungen in einer Datei geschrieben, welche dann eingelesen und damit ein Projekt generiert wird.

Die Nebenapplikation ist hauptsächlich für fortgeschrittene Nutzer gedacht, die den Projektgenerierungsprozess weiter automatisieren möchten. Statt alle Informationen manuell und Schritt-für-Schritt einzugeben, können diese zentral und auf einmal in eine JSON-Datei geschrieben werden. Dies kann von großem Vorteil sein, wenn mehrere ähnliche Projekte erstellt werden sollen. Die wichtigsten Parameter dieser Datei sind leicht veränderbar, wodurch der manuelle Durchlauf der Fragen entfällt. Auch ist es möglich, mit dieser Applikation gleich mehrere Projekte auf einen Schlag zu generieren.

Es gibt zwei wesentliche Wege, diese Applikation und ihre Funktionen zu nutzen. Einerseits kann der Prozess über die Kommandozeile gestartet werden. Bei der Wahl des Startbefehls wurde auf Einfachheit und Effizienz geachtet (für genauere Informationen siehe: JSON-Funktion). Andererseits kann der Prozess auch mit dem PowerShell-Skript gestartet werden. Beide Optionen ermöglichen es dem Nutzer, universell ein Projekt zu erstellen.

7.3 Probleme und Herausforderungen

Während der Entwicklung entstanden einige Probleme, die bewältigt werden mussten. Eine der größten Herausforderungen war zweifellos die intensive Recherche. Dieser Abschnitt war mindestens genauso wichtig wie die eigentliche Entwicklungsarbeit. Der erste Schritt dieser Diplomarbeit bestand nicht im Programmieren, sondern in der Suche nach geeigneten Technologien.

Die bedeutende Herausforderung lag darin, eine Vielzahl von Technologien zu bestimmen und ihre Vorzüge und Schwächen im Detail zu untersuchen. Dies wurde durch eine Vielzahl von Tests erreicht, wodurch die wesentlichen Eigenschaften praktisch getestet und festgehalten werden konnten.

Das Ziel bestand darin, eine möglichst große Auswahl an geeigneten Technologien zu dokumentieren. Alle Technologien, unabhängig von ihrer Komplexität, mussten geprüft werden. Die gesammelten Erkenntnisse dienten als Basis für weitere Entscheidungen. Nach jeder Phase fanden zahlreiche und umfangreiche Meetings statt, um die Eigenschaften der jeweiligen Tools gemeinsam zu bewerten. Dieser Prozess wurde besonders detailliert und strukturiert durchgeführt.

Trotzdem darf nicht vergessen werden, dass auch dieser Abschnitt eine wesentliche Rolle in der Arbeit spielte und keinesfalls zu unterschätzen ist.

Neben der Recherche mussten auch die Anforderungen der Firma Koncept berücksichtigt werden. Es wurde angestrebt, eine Lösung zu entwickeln, die es ermöglicht, Projekte rasch und unkompliziert zu erstellen. Bei dieser Generierung sollten Methoden, Variablen und Klassen automatisch erstellt werden, basierend auf den Eingaben des Nutzers. Zudem sollte das Template sicherstellen, dass die Codestruktur einheitlich ist. Die Struktur der Klassen sollte immer gleich sein. Für die Auswahl geeigneter Technologien waren diese Anforderungen maßgeblich.

Außerdem musste die ausgewählte Technologie in der Lage sein, große und komplexe Projektstrukturen zu erzeugen, ohne dabei einen hohen Ressourcenaufwand zu verursachen. Abschließend sei erwähnt, dass das generierte Projekt in Csharp angelegt sein sollte und unmittelbar nach der Erstellung in Visual Studio geöffnet werden kann.

Des Weiteren war es notwendig, sämtliche Technologien samt ihrer Vor- und Nachteile zu protokollieren. Diese Dokumentation war nicht nur für diese Arbeit von zentraler Bedeutung, sondern auch für die Firma Koncept. Um die jeweiligen Merkmale der Technologien, vor allem deren Vorzüge und Nachteile, nachvollziehen zu können, benötigte sie Unterlagen sowie kleine Demobeispiele.

8 Resümee

8.1 Reflexion des Auftraggebers Fa. Koncept [11]

8.1.1 Einsatz des Produkts

”Im Rahmen ihres Praktikums haben die Praktikanten ein Produkt entwickelt, das für unseren Nutzen bereits weitgehend fertiggestellt ist. Allerdings sind noch einige Anpassungen erforderlich, damit es vollständig in unser System integriert werden kann. Aufgrund dieser offenen Punkte und begrenzter zeitlicher Ressourcen unsererseits konnte das Produkt bisher noch nicht produktiv eingesetzt werden. Nichtsdestotrotz stellt das entwickelte Produkt eine solide Grundlage dar und bietet viel Potenzial für eine zukünftige Nutzung.” -Herr Philipp Starzer - Kon-Cept Management Information Services GmbH

8.1.2 Zusammenarbeit

”Die Zusammenarbeit mit den Praktikanten verlief durchweg positiv. Sie zeigten sich von Anfang an hoch motiviert und engagiert und arbeiteten weitgehend selbstständig an ihren Aufgaben. Aufgaben und Anforderungen, die in gemeinsamen Meetings besprochen wurden, konnten sie strukturiert und zielgerichtet umsetzen. Darüber hinaus brachten sie eigene Ideen ein und gingen konstruktiv mit Feedback um. Ihr Engagement und ihre lösungsorientierte Denkweise haben einen positiven Eindruck hinterlassen.” -Herr Philipp Starzer - Kon-Cept Management Information Services GmbH

8.2 Persönliches Resümee

8.2.1 Perspektive Prandstätter

Im Rahmen dieser Arbeit lernte ich, wie wichtig und entscheidend die Technologiewahl ist. Jede Technologie hat ihre eigenen und besonderen Eigenschaften. Basierend auf den Umständen und den Anforderungen muss die geeignete Technologie ausgewählt werden.

Darüber hinaus heiße ich die Zusammenarbeit mit der Firma Koncept gut. Alle Meetings oder Besprechungen wurden mit gegenseitigem Respekt durchgeführt. Auch bei Fragen über Struktur, Code oder allgemeinen Sachen gab es immer Verlass auf die Mitarbeiter der Firma Koncept.

In Erinnerung bleibt mir einerseits die angenehme Arbeitsatmosphäre, die trotz der teils komplexen Aufgaben nie angespannt wirkte. Das offene und kollegiale Miteinander erleichterte die Einarbeitung in die Firmenstruktur und half beim gemeinsamen Lösen von Aufgaben.

Andererseits blieben mir auch die unterschiedlichsten Technologien und ihre Herausforderungen im Gedächtnis. In dieser Zeit konnte ich mein Wissen über die verschiedensten Softwarearchitekturen erweitern. Außerdem durfte ich hautnah miterleben, wie wichtig und unverzichtbar das Dokumentieren von Programmen ist.

Diese Arbeit war nicht nur eine fachliche, sondern auch eine persönliche Erfahrung.

8.2.2 Perspektive Haderer

Das Erstellen einer Diplomarbeit und die enge Zusammenarbeiten mit einer Firma war für mich eine ganz neue Erfahrung. Ich war sehr positiv überrascht, wie schnell wir ein passendes Vorgehensmodell gefunden hatten und wie gut die Zusammenarbeit funktionierte.

Das Eruieren einer passenden Technologie für eine Problemstellung eröffnete mir einen breiteren Blickwinkel. Vor allem bestätigte sich die Wichtigkeit der richtigen Technologiefindung bzw. zeigte sich mir wie viel Zeit wirklich dafür investiert werden muss.

Auch das Prinzip immer neue Prototypen zu erstellen und diese kritisch zu bewerten, weckte mein Interesse. Vor allem gefiel es mir, diese vor den Mitarbeitern von Koncept zu präsentieren und Verbesserungsvorschläge zu erhalten. Oftmals wurden Technologien auch ganz abgelehnt , wodurch wieder Motivation entstand, eine neue Lösung für ”Das Problem” zu finden.

Die Zusammenarbeit mit Koncept gab mir einen sehr guten Einblick in die ”richtige Arbeitswelt”. Den persönlichen, respektvollen Umgang miteinander und das Zusammenarbeiten bestätigte mein

positives Denken. Insgesamt war die Arbeitsatmosphäre sehr entspannt und mein konzentriertes Arbeiten gut möglich.

Das Verfassen dieser Arbeit empfand ich nicht ganz so positiv wie das Finden und Umsetzen einer Lösung. Vor allem war es für mich schwierig Technologien, welche ich im Kopf bereits als "nicht nützlich" abgespeichert hatte, wieder neu zu beschreiben. Teilweise musste ich mich in diese Themen wieder einlesen. Ich merkte schnell, dass ich viel lieber Neues lerne.

Insgesamt hat mir diese Diplomarbeit neue Erkenntnisse gebracht und Einblicke in die Arbeitswelt als Programmierer gegeben. Ich konnte einiges Neues lernen und bereits Gelerntes anwenden. Zusammengefasst sammelte ich nur gute Erfahrungen beim Erstellen dieser Arbeit.

Literaturverzeichnis

[1] Microsoft, „.NET-Standardvorlagen für dotnet new," letzter Zugriff am 12.11.2024. Online verfügbar: https://learn.microsoft.com/de-de/dotnet/core/tools/dotnet-new-sdk-templates

[2] ——, „.NET-Standard," letzter Zugriff am 12.11.2024. Online verfügbar: https://github.com/dotnet/standard

[3] U. A. User: Thomas, „How to create .NET Platform Standard project," letzter Zugriff am 15.11.2024. Online verfügbar: https://stackoverflow.com/questions/38755201/how-to-create-net-platform-standard-project

[4] Microsoft, „Verwenden Sie die Cookiecutter-Erweiterung," letzter Zugriff am 20.11.2024. Online verfügbar: https://learn.microsoft.com/de-de/visualstudio/python/using-python-cookiecutter-templates?view=vs-2022

[5] ookiecutter, „Cookiecutter," letzter Zugriff am 5.12.2024. Online verfügbar: https://www.cookiecutter.io/

[6] Timothy Edmund Crosley, „An introduction to LaTeX," letzter Zugriff am 8.12.2024. Online verfügbar: https://github.com/timothycrosley/cookiecutter-python

[7] The LATEX Project, „cookiecutter-python," letzter Zugriff am 20.12.2024. Online verfügbar: https://cookiecutter-python.readthedocs.io/en/latest/tutorial.html

[8] IONOS Redaktion, „Gulp vs. Grunt: Das zeichnet die beiden Task-Runner aus," letzter Zugriff am 4.1.2025. Online verfügbar: https://www.ionos.at/digitalguide/websites/web-entwicklung/gulp-vs-grunt-so-unterscheiden-sich-die-task-runner

[9] Microsoft, „Verwenden von Grunt in ASP.NET Core," letzter Zugriff am 4.1.2025. Online verfügbar: https://learn.microsoft.com/de-de/aspnet/core/client-side/using-grunt?view=aspnetcore-8.0

[10] Gulp, „Gulp," letzter Zugriff am 8.1.2025. Online verfügbar: https://gulpjs.com/

[11] ——, „Quick Start," letzter Zugriff am 10.1.2025. Online verfügbar: https://gulpjs.com/docs/en/getting-started/quick-start/

[12] Grunt, „Grunt The JavaScript Task Runner," letzter Zugriff am 10.1.2025. Online verfügbar: https://gruntjs.com/

[13] ——, „Documentation," letzter Zugriff am 18.1.2025. Online verfügbar: https://gruntjs.com/documentation

[14] Moritz Halbritter, „spring-io/initializr," letzter Zugriff am 18.1.2025. Online verfügbar: https://github.com/spring-io/initializr?

[15] Spring, „Spring Initializr Reference Guide," letzter Zugriff am 6.2.2025. Online verfügbar: https://docs.spring.io/initializr/docs/current/reference/html

[16] ——, „Spring Initializr," letzter Zugriff am 8.2.2025. Online verfügbar: https://start.spring.io/

[17] Helm, „Helm," letzter Zugriff am 8.2.2025. Online verfügbar: https://helm.sh/de/

[18] Microsoft, „Codegenerierung und T4-Textvorlagen," letzter Zugriff am 10.2.2025. Online verfügbar: https://learn.microsoft.com/de-de/visualstudio/modeling/code-generation-and-t4-text-templates?view=vs-2022

[19] ——, „Schreiben einer T4-Textvorlage," letzter Zugriff am 10.2.2025. Online verfügbar: https://learn.microsoft.com/de-de/visualstudio/modeling/writing-a-t4-text-template?view=vs-2022

[20] ——, „Laufzeittextgenerierung mithilfe von T4-Textvorlagen," letzter Zugriff am 10.2.2025. Online verfügbar: https://learn.microsoft.com/de-de/visualstudio/modeling/run-time-text-generation-with-t4-text-templates?view=vs-2022&tabs=csharp

[21] Vlada Shubina, Mike Lorbetske, Bekir Ozturk, David Karlaš und weitere, „Post Action Registry," letzter Zugriff am 12.2.2025. Online verfügbar: https://github.com/dotnet/templating/wiki/Post-Action-Registry

[22] ——, „Using Primary Outputs for Post Actions," letzter Zugriff am 16.2.2025. Online verfügbar: https://github.com/dotnet/templating/wiki/Using-Primary-Outputs-for-Post-Actions

[23] User: billy bud, User: Leif Swenson, „How to execute a postaction script from a dotnet new template," letzter Zugriff am 18.2.2025. Online verfügbar: https://stackoverflow.com/questions/75876420/how-to-execute-a-postaction-script-from-a-dotnet-new-template

[24] Microsoft, „Verwenden von Assistenten mit Projektvorlagen," letzter Zugriff am 2.3.2025. Online verfügbar: https://learn.microsoft.com/de-de/visualstudio/extensibility/how-to-use-wizards-with-project-templates?view=vs-2022

[25] ——, „IWizard Schnittstelle," letzter Zugriff am 5.3.2025. Online verfügbar: https://learn.microsoft.com/de-de/dotnet/api/microsoft.visualstudio.templatewizard.iwizard?view=visualstudiosdk-2022

[26] Addy Osmani, Sindre Sorhus, Pascal Hartig, Stephen Sawchuck, Simon Boudrias und weitere, „Yeoman," letzter Zugriff am 6.3.2025. Online verfügbar: https://yeoman.io/

[27] Microsoft, „Erstellen von Office-Add-In-Projekten mithilfe des Yeoman-Generators," letzter Zugriff am 6.3.2025. Online verfügbar: https://learn.microsoft.com/de-de/office/dev/add-ins/develop/yeoman-generator-overview

[28] Addy Osmani, Sindre Sorhus, Pascal Hartig, Stephen Sawchuck, Simon Boudrias und weitere, „Generators," letzter Zugriff am 10.3.2025. Online verfügbar: https://yeoman.io/generators/

[29] ——, „Yeoman," letzter Zugriff am 15.3.2025. Online verfügbar: https://github.com/yeoman

[30] Andrew Worcester, Corbin Crutchley, Nicolas Carlo, Carlos Alcaide Corvo, Armando Magalhães und weitere, „Learning to Plop : PLOP," letzter Zugriff am 15.3.2025. Online verfügbar: https://plopjs.com/documentation/

[31] ——, „plop.js/plop," letzter Zugriff am 20.3.2025. Online verfügbar: https://github.com/plopjs/plop

[32] ——, „plopjs - npm," letzter Zugriff am 22.3.2025. Online verfügbar: https://www.npmjs.com/package/plopjs

[33] Lorenz Weiß, „Add automated code templates for react components," letzter Zugriff am 25.3.2025. Online verfügbar: https://www.lorenzweiss.de/creating_components_with_plop/

Abbildungsverzeichnis

Quellcodeverzeichnis

Anhang

A Bilder

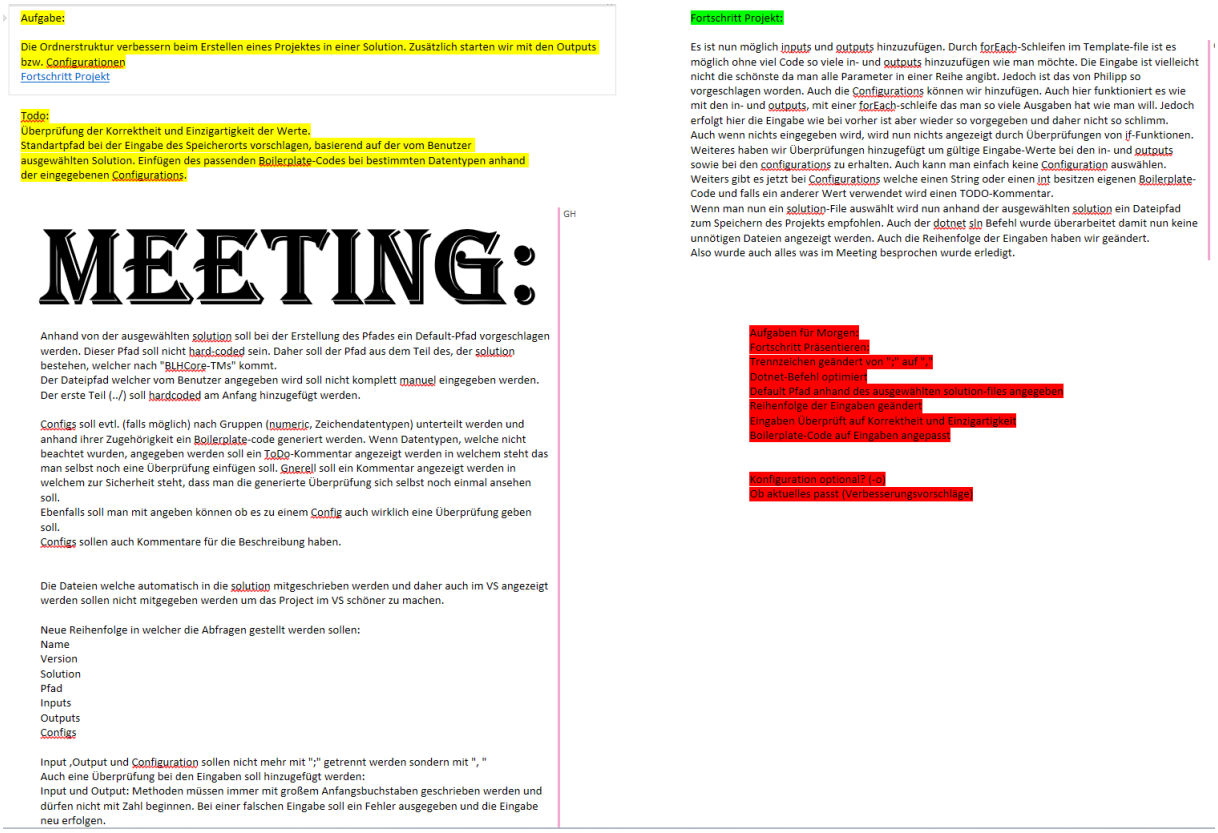


Abbildung 29: Dokuasuzug

B Quellenverzeichnis

Reflexion von Koncept Email von Phillip Starzer,Projektbetreuer unserer Diplomarbeit bei Koncept

C Glossar

Template

Ein Template ist eine Vorlage für die Erstellung von Dokumenten. Es enthält Platzhalter oder Strukturen, welche mit spezifischen Daten oder anderen Inhalten gefüllt sind. Templates sind nützlich, um wiederverwendbare Strukturen zu speichern und diese beim Erstellen von neuen Projekten ohne viel Aufwand einzubauen. Templates können für Code aller Art verwendet werden. Es gibt auch Templates für E-Mails um automatisierte Nachrichten zu erstellen.

Konstruktor

Ein Konstruktor ist eine Methode in einer Klasse, welche beim Erstellen eines Objekt aufgerufen wird. Somit können Objekte initialisiert werden, um etwa Variablen zu setzen oder auch bestimmte Resourcen bereitzustellen. Ein Konstruktor hat keinen Rückgabewert (nicht einmal void) und besitzt immer den gleichen Namen wie die Klasse.

.Net SDK

Das .Net SDK steht abgekürzt für .Net Software Development Kit und besteht aus einer Sammlung von Tools und Bibliotheken. Diese werden für das Entwickeln, Testen und Bereitstellen von .Net-Anwendungen benötigt.

Trigger

Trigger sind Bedingungen oder Ereignisse, welche eine bestimmte Aktion auslösen. Am bekanntesten sind Datenbank-Trigger. Wenn zum Beispiel in einer Datenbank eine neue Zeile eingefügt wird, kann durch einen Trigger in eine Log-Datei die Info, dass eine neue Zeile geschrieben wurde, eingefügt werden.

Open Source

Open Source bedeutet, dass der QuellCode einer Software öffentlich für jeden zugänglich ist.

npm

npm steht abgekürzt für Node Package Manager und ist ein Packetmanager für Javascript. npm kann Pakete installieren aktualisieren und deinstallieren. Ebenso verwaltet npm die Pakete in der Datei package.json, die für Javascript-Anwendungen benötigt wird. Es ist über den Befehl ”npm run” möglich vorgefertigte Skripts auszuführen.

minifizieren

Beim Minifizieren wird der Quellcode in einer Datei verkleinert und überflüssige Zeichen, wie Leerzeichen, Zeilenumbrüche und Kommentare, entfernt.

SASS

SASS ist die Abkürzung für Syntactically Awesome Stylesheets. Es ist eine Erweiterung für CSS und bietet zusätzliche Funktionen um Stylesheets effizienter zu schreiben.

LESS

LESS steht abgekürzt für Leaner Style Sheets. Es ist wie SASS eine CSS-Erweiterung, welche zusätzliche Funktionen hinzufügt. Less zielt darauf ab, Stylesheets wiederverwendbar und modularer zu gestalten.

SCSS

SCSS steht für Sassy CSS und bietet eine erweiterte Syntax von SASS. Es werden zusätzliche Funktionen wie Variablen und Verschachtelung geboten. Somit werden Stylesheets übersichtlicher und wiederverwendbarer.

CSS

CSS ist die Abkürzung für Cascading Style Sheets und ist eine Stylesheet-Sprache. Mithilfe von CSS lässt sich das Aussehen und Layout von HTML-Dokumenten gestalten. Eigenschaften von Texten wie die Farbe, Größe oder auch der Abstand können geändert werden. Es kann jedoch noch viel mehr gestylt werden. CSS kann direkt in der HTML-Datei einer Website

eingebunden werden. Empfehlenswert ist jedoch, das Design in eine eigene Datei auszulagern um alles übersichtlicher zu halten.

Plugins

Mittels Plugins können Funktionalitäten zu Programmen hinzufügt werden, ohne das Hauptprogramm zu ändern. Plugins sind sozusagen Erweiterungen oder Zusatzmodule.

Dependencies

Dependencies oder auch Abhängigkeiten sind Module oder Bibliotheken, welche eine Software benötigt, um zu funktionieren. Diese werden genutzt, um bestimmte Funktionen bereitzustellen ohne sie selbst programmieren zu müssen.

CoffeeScript

CoffeeScript ist eine Programmiersprache, welche eine kürzere und elegantere Syntax als Javascript bietet, aber nach JavaScript kompiliert.

Distribution-Dateien

Distrubution-Dateien oder abgekürzt auch Dist-Dateien sind Dateien, die zur Bereitstellung einer Anwendung verwendet werden. Oftmals ist in ihnen optimierter und minifizierter Code zu finden, welcher direkt in der Produktionsumgebung läuft.

Streams

Streams sind Datenströme, die große Datenmengen stückweise verarbeiten. Oftmals spricht man auch von Chunk für Chunk. Die Datenmengen werden nicht komplett im Speicher gehalten, wodurch sie effizienter für Dateien, Netzwerkkommunikation oder Datenverarbeitung sind.

Framework

Ein Framework ist eine vorgefertigte Struktur, welche zur Entwicklung von Software verwendet wird.

Dependency Injection

Bei der Dependency Injection werden Dependencies anstatt diese direkt in eine Klasse oder Modul einzufügen, von außen eingefügt. Dadurch wird der Code flexibler, testbarer und besser wartbar gemacht.

Rest-API

Rest-API ist die Abkürzung für Representation State Transfer API. Sie ist eine Schnittstelle, die das Senden von Daten zwischen Client und Server ermöglicht.

IDE

IDE steht für Integrated Development Environment und ist eine Entwicklungsumgebung. In dieser werden Werkzeuge zum Schreiben, Testen und Debuggen von Code geboten.

Microservice

Microservices sind Anwendungen, die aus mehreren kleinen, unabhängigen Diensten besteht, welche eigenständig entwickelt und bereitgestellt werden. Jeder Microservice erfüllt nur eine bestimmte Aufgabe. Sie kommunizieren meist über Rest-APIs oder Messaging Systeme.

Kubernetes

Kubernetes ist eine Plattform, welche die Bereitstellung, Skalierung und Verwaltung von Containern automatisiert. Dadurch werden Lasten verteilt. Kubernetes skaliert Container automatisch und behebt auch Fehler automatisch.

Boilerplate-Code

Boilerplate-Code ist Code, welcher immer wieder verwendet wird, aber wenig Funktionalität beinhaltet. Jedoch wird Boilerplate-Code benötigt und kann nicht einfach ersetzt werden.

IntelliSense

IntelliSense ist eine Automatisierungshilfe, welche mithilfe von Code-Vorschlägen das Programmieren einfacher und schneller gestaltet.

XML

XML ist die Abkürzung für eXtensible Markup Language. Es ist eine sogenannte Auszeichnungssprache. XML wird verwendet um Daten zu strukturieren, speichern und übertragen. Diese Sprache ist für Menschen sowie für Maschinen lesbar.

Backend

Als ein Backend wird der Code bezeichnet, der auf dem Server des Programms läuft und für die Verarbeitung der Daten und deren Speicherung sowie Logik verantwortlich ist. Es stellt die Schnittstelle für das Frontend zur Verfügung.

Frontend

Das Frontend ist der Teil eines Programms, der direkt mit dem Benutzer interagiert. Es ist verantwortlich, die Benutzeroberfläche und dessen Design bereitzustellen.

Kommandozeile

Die Kommandozeile, oder auch Terminal oder Konsole, ist ein Interface, mit dem der Benutzer nur über Text kommuniziert. In der Kommandozeile gibt es keine grafische Oberfläche, Knöpfe oder Sonstiges. Durch sie können Programme ausgeführt, Dateien verwaltet oder Aufgaben automatisiert werden.

GUI

GUI ist die Abkürzung für Graphical User Interface. Es bedeutet also grafische Benutzeroberfläche. Mit Hilfe dieser ist es Benutzern möglich eine Software über visuelle Elemente wie ein Fenster oder eine Schaltfläche zu bedienen.