

Bachelorarbeit
in
Medieninformatik
Zu Erlangung des Grades Bachelor of Science

**Entwicklung eines speziellen DevOps
Projekt-Konfigurator Werkzeuges für
Projektinitialisierung /– aktualisierung**

Referent: Herr Prof. Dr. Eisenbiegler
Korreferent: Herr Dr.-Ing- Christoph Rathfelder
Vorgelegt am: 31.12.2021
Vorgelegt von: Nick Stecker
25195
Haselstiegstr. 5/1
78052 Villingen
nick.stecker@hs-furtwangen.de

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

[Ort, Datum Name]

Abstract

[Englisch, 100 -120 Worte]

[Deutsch, 100 – 120 Worte]

Inhaltsverzeichnis

Eidesstattliche Erklärung	2
Abstract	3
Abbildungsverzeichnis	6
Tabellenverzeichnis	7
Abkürzungsverzeichnis	8
1. Einleitung	9
1.1 Nutzen dieser Bachelorarbeit	10
1.2 Die Problemstellung	10
1.2.1 Die Projekterstellung	11
1.2.2 Die Projektaktualisierung	11
1.3 Ziel dieser Bachelorarbeit	12
1.4 Weiterer Aufbau	12
2. Grundlagen	14
2.1 Softwareprojekte	14
2.2 Die Bedeutung von DevOps	14
2.3 Continuous Integration, Continuous Delivery (CI/CD)	15
2.4 Build-Tools	Fehler! Textmarke nicht definiert.
3. und Stand der Technik	17
3.1 Integrierte Entwicklungsumgebung (IDE)	17
3.1.1 Eclipse C/C++ Development Toolkit (CDT)	18
3.1.2 Visual Studio Code	24
3.1.3 CLion IDE	20
3.2 Das Problem der Projektaktualisierung ...	Fehler! Textmarke nicht definiert.
4. Lösungsansatz	28
5. Softwareentwurf	29
5.1 GUI Sketches	30
5.2 Architektur	37

5.3	Extrafunktionalitäten	37
6.	Implementierung und Evaluierung	38
7.	Zusammenfassung	39
7.1	Zusammenfassung	39
7.2	Das Problem	39
7.3	Mögliche Erweiterungspunkte	39
Literatur Fehler! Textmarke nicht definiert.	
Anhang	43

Abbildungsverzeichnis

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

Tabellenverzeichnis

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

Abkürzungsverzeichnis

CI/CD *Continuous Integration und Continuous Delivery*

DevOps Development und IT Operations

GUI Graphical User Interface

IDE Integrated Development Environment

JRE *Java Runtime Environment*

UML Unified Modeling Language

1. Einleitung

Die moderne Softwareentwicklung erfordert eine Vielzahl externer Programmierwerkzeuge (Tools), wie beispielsweise Integrated Development Environments (IDEs), Versionsverwaltungen, spezielle Build Systeme oder Frameworks, um den Softwareentwicklungsprozess professionell gestalten und verwalten zu können.

Dabei werden diese Tools nacheinander an bestimmten Stellen in dem Entwicklungsprozess eingesetzt, um spezifische Aufgaben zu lösen.

In der Praxis werden die Tools dabei zu einer sogenannten Werkzeugkette (Toolchain) miteinander verknüpft, um einzelne Prozesse in der Softwareentwicklung zu automatisieren.

...dadurch können einzelne Prozesse in dem gesamten Entwicklungsprozess automatisiert werden.

Allerdings erfüllen die Tools nicht immer allen Ansprüchen eines Softwareentwicklers. Eines dieser Probleme ist...

In vielen Berufsfeldern wäre die Arbeit ohne spezielle Werkzeuge undenkbar. Eines dieser Berufsfelder ist die Softwareentwicklung. Für eine professionelle Softwareentwicklung werden heutzutage nämlich viele verschiedene Programmierwerkzeuge (Tools) eingesetzt, die den Software-Entwicklungsprozess optimal unterstützen und beherrschbar machen.

In dieser Bachelorarbeit geht es um die Entwicklung eines solchen Tools. Der Projekt-Konfigurator ist ein spezielles Werkzeug für die Softwareentwicklung. Es soll Softwareentwicklern dabei helfen Softwareprojekte, basierend auf einem Projekt-Template zu erstellen und zu aktualisieren. Das Template-Projekt ist dabei einfach ein Ordner auf dem Computer mit einer gewissen Ordnerstruktur. Diese Ordnerstruktur setzt sich aus verschiedenen Unterordnern und Dateien zusammen.

Je nach Anwendungsfall können Toolchains aus unterschiedlichen Tools aufgebaut sein. Die größte Verwendung finden Toolchains jedoch in der Umsetzung der Continuous Integration und Continuous Delivery (CI/CD) Methode.

1.1 Nutzen dieser Bachelorarbeit

Diese Abschlussarbeit löst ein spezifisches Problem der Abteilung Anwendungsentwicklung von der Hahn-Schickard Gesellschaft in Villingen-Schwenningen. Das Problem bezieht sich hierbei auf die Initialisierung bzw. Aktualisierung eines bestehenden Projektes durch ein existierendes Template. Das Problem soll durch den in Abschnitt 1 beschriebene Projekt-Konfigurator gelöst werden.

1.2 Die Problemstellung

Die Entwicklung großer Softwareprojekte kann eine sehr schwierige Aufgabe sein, insbesondere wenn das Projekt nicht gut strukturiert ist. Darüber hinaus sind viele moderne Projekte aus modularen Teilen aufgebaut, die später für verschiedene Anwendungsfälle wiederverwendet werden können. Um die Wiederverwendbarkeit und Verbesserung dieser Module zu verbessern, werden sie oft selbst als Softwareprojekte mit eigenem Layout und zugehörigen Werkzeugen erstellt. Es ist kein unwahrscheinliches Szenario, dass sich das Layout oder die Tools dieser Projekte im Laufe der Zeit mit zunehmendem Entwicklungswissen ändern und daher zu einer Aktualisierung auffordern. Dies manuell zu tun, kann eine mühsame Aufgabe sein, insbesondere wenn es eine große Anzahl dieser Module gibt, sowie fehleranfällig. Um dieses Problem zu lindern, könnte eine grafische Benutzeroberfläche (GUI) verwendet werden, um die Projektlayoutänderungen auf einfache und leicht verständliche Weise auf jedes Modul anzuwenden. Diese Benutzeroberfläche sollte...

Die Abteilung Anwendungsentwicklung arbeitet täglich an vielen verschiedenen Softwareprojekten. Jedes Softwareprojekt, besitzt dabei, je nach Programmiersprache, eine exakte Vorgabe über die Basis der zu verwendenden Verzeichnisstruktur. Das bedeutet, dass jedes Softwareprojekt, einer Programmiersprache, nach der Initialisierung dieselbe Ordnerstruktur aufweist. Inwiefern die Ordnerstruktur jedes Softwareprojekts in ihrer Entwicklungszeit ausgebaut wird (wie viele Dateien also hinzugefügt und bearbeitet werden), ist für jedes Softwareprojekt spezifisch. Allerdings darf sich dabei die Grundstruktur des Projekts nicht ändern.

Wenn ein Mitarbeiter der Abteilung Anwendungsentwicklung nun ein neues Softwareprojekt erstellen will, muss dieser erst einmal wissen wie die Verzeichnisstruktur für die jeweilige Programmiersprache des Projekts aufgebaut ist.

Kennt der Mitarbeiter die Struktur, muss er diese auf dem Computer manuell nachbauen. Das heißt, der Mitarbeiter muss manuell alle nötigen Ordern und Dateien in der richtigen Ordnerstruktur erstellen. Kennt der Mitarbeiter die Struktur jedoch nicht, muss der Mitarbeiter diese erst herausfinden.

Um dieses Problem zu umgehen, besitzt die Abteilung für die meisten Softwareprojekte einer Programmiersprache ein vorgefertigtes Template-Projekt. Die Template-Projekte sind in den vorgegebenen Verzeichnisstrukturen aufgebaut und besitzen alle notwendigen Order bzw. Dateien. Die meisten Dateien sind so aufgebaut, dass erstellte Projekte diese ohne Veränderungen benutzen können. Allerdings beinhalten einige Dateien Platzhalter, die bei der Projekterstellung mit den Metadaten eines Softwareprojekts ersetzt werden müssen. Ohne diese Bearbeitungen, haben die Entwickler der Abteilung nur eine exakte Kopie eines Template-Projektes – das erstellte Softwareprojekt würde bis dato also noch kein reales Projekt repräsentieren.

1.2.1 Die Projekterstellung

Die Projekterstellung mithilfe des beschriebenen Template-Projektes ist das erste große Problem dieser Bachelorarbeit. Denn um ein neues Softwareprojekt zu erstellen, müssen die Mitarbeiter das passende Template-Projekt kopieren und die Platzhalter mit den Metadaten des Projekts bearbeiten. Jedoch müssen die Mitarbeiter vor der Bearbeitung wissen, welche Template-Dateien Platzhalter beinhalten und an welchen Stellen sie sich in der Template-Datei befinden. Wenn die Mitarbeiter dies nicht wissen, müssen sie das, vor der Bearbeitung herausfinden. Wenn sie es allerdings wissen, müssen sie jede zu bearbeitende Datei manuell öffnen und alle Platzhalter eigenständig ersetzen.

Ein Beispiel: Wenn ein Entwickler den Namen eines Projekts bearbeiten/ändern möchte, muss er den Projektnamen manuell in dem Readme-Titel, in den Buildscript-Variablen und der Dokumentierung des Projekts ändern. Das sind drei Dateien die der Entwickler nur für den Projektnamen öffnen und bearbeiten muss.

Dieses Vorgehen ist im gleichen Maße zeitaufwändig wie fehleranfällig.

1.2.2 Die Projektaktualisierung

Das zweite große Problem dieser Bachelorarbeit ist, dass ein Template-Projekte jederzeit aktualisiert werden kann. Die Entwickler können jederzeit bestimmte

Order/Dateien des Template-Projektes überarbeiten, oder ihm neue Order/Dateien hinzufügen.

Die Problematik hierbei ist, dass diese Änderungen im Anschluss auf alle bestehenden Softwareprojekte, die aus dem veränderten Template-Projekt entstanden sind, übertragen werden müssen. Dazu müssen die Entwickler jedes dieser Softwareprojekte manuell auf dem Computer öffnen und in diesen die gleichen Änderungen vornehmen, die sie davor im Template-Projekt durchgeführt haben.

Dieses Vorgehen ist wie im vorherigen Abschnitt extrem zeitaufwändig und fehleranfällig.

1.3 Ziel dieser Bachelorarbeit

Das Ziel dieser Bachelorarbeit ist die Konzeption und Implementierung eines Projekt-Konfigurators, der die beschriebenen Probleme in Abschnitt 1.2.1 und Abschnitt 1.2.2 löst.

Der Projekt-Konfigurator ist ein spezielles Programmierwerkzeug und besteht aus einer Software sowie einer grafischen Benutzeroberfläche (in Englisch, Graphical User Interface; abgekürzt GUI), die von den Entwicklern der Abteilung Anwendungsentwicklung ohne weitere Vorkenntnisse benutzt werden kann. Mithilfe der GUI können die Probleme wie folgt gelöst werden:

1.4 Weiterer Aufbau

In Kapitel 2 wird der aktuelle Stand der Technik bezüglich aktueller Möglichkeiten für die Projekterstellung /-aktualisierung beleuchtet. Bereits vorhandene Werkzeuge/Software und deren Eigenschaften werden mittels Recherche ermittelt.

Das dritte Kapitel behandelt den Lösungsansatz des beschriebenen Problems (in Abschnitt 1.2).

In Kapitel vier geht es um den Softwareentwurf des Projekts. Hier wird das geplante Aussehen der GUI anhand von Skizzen gezeigt und erklärt. Darüber hinaus werden an dieser Stelle die geplante Architektur, sowie alle Funktionalitäten mithilfe von UML Diagrammen erklärt und begründet. Ein wesentlicher Fokus liegt hier auf der Modularität des Quellcodes

Im fünften Kapitel geht es um die Implementierung und Evaluierung des Projekts. Zum einen wird hier auf die Umsetzung des Projekt-Konfigurators eingegangen. Es werden einzelne Teile des Quellcodes und verschiedene Konfigurationsdateien näher beschrieben. Zum anderen geht es in diesem Kapitel um die Evaluierung aller Funktionalitäten. Jedes Anwendungsbeispiel wird aus Anwendersicht mithilfe von GUI-Screenshots beschrieben. Implementierte Extrafunktionalitäten, die über die Aufgabenstellung hinausgehen werden, hier besonders erwähnt.

Im sechsten und letzten Kapitel geht es um eine Zusammenfassung der geleisteten Arbeit und um die Beantwortung der Frage inwiefern der Projekt-Konfigurator das beschriebene Problem (in Abschnitt 1.3) löst. Außerdem gibt dieses Kapitel noch einen Ausblick auf künftige Erweiterungspunkte.

2. Grundlagen

In diesem Kapitel werden die Grundlagen erklärt, die für das Verständnis aller weiteren Kapitel notwendig sind.


2.1 Softwareprojekte

[Hier erklären was Projekte und Softwareprojekte in dem Kontext dieser Bachelorarbeit sind]

2.2 Die Bedeutung von DevOps

Bei der Bezeichnung DevOps handelt es sich um ein Kofferwort, dass sich aus den Begriffen „Development (Entwicklung)“ und „IT Operations (IT-Betrieb)“ zusammensetzt. Damit sind die jeweiligen Organisationseinheiten im IT-Bereich gemeint, die in einer traditionellen Organisation klassischerweise getrennt sind und mit unterschiedlichen Zielsetzungen arbeiten [1, 2]. ~~Bei dieser Arbeit kommt es ohne den DevOps-Ansatz zu zwei großen Problemen. Das erste Problem ist, dass die Softwareentwicklung meist in Projektform an der Umsetzung von Kundenanforderungen arbeitet. Diese müssen schnell realisiert werden, damit das IT-Produkt und/oder –Service nicht an Akzeptanz verliert und weiterhin erfolgreich auf dem Markt bleibt. Der IT-Betrieb strebt dagegen eine entgegengesetzte Arbeitsweise an. Seine Ziele sind vor allem ein hohes Maß an Stabilität, Verfügbarkeit sowie Sicherheit [1].~~

Das ~~zweite~~ Problem betrifft die mangelnde Zusammenarbeit zwischen den beiden Bereichen. Dabei ist das Ziel der Softwareentwicklung die Bereitstellung der fertigen Software („Release“). Die anschließende Installation („Deployment“) ist hingegen der Startpunkt des IT-Betriebs. Wurden in der Entwicklung die Anforderungen des IT-Betriebs für die Installation nicht berücksichtigt, kann das zu Verzögerungen oder fehlschlagen des Deployments führen [3].

Angesichts derartiger Probleme ist DevOps entstanden. Es umfasst eine Reihe von Technologien, Prozessen und Werkzeugen, die darauf ausgelegt sind, typische Probleme in der Zusammenarbeit zwischen Entwicklungs- und Betriebseinheiten zu lösen und dadurch das Kundenerlebnis und die Kundenzufriedenheit zu verbessern. Entscheidend dafür sind Veränderungen in der Zusammenarbeit dieser Bereiche sowie eine möglichst große Automatisierung der täglichen Aufgaben [3]. 

~~2.3 Continuous Integration, Continuous Delivery~~

Continuous Integration und Continuous Delivery (CI/CD) ist ein grundlegender Prozess innerhalb von DevOps. Es umfasst eine Reihe von Tools, die nacheinander als eine Art Werkzeugkette (eng.: Toolchain) eingesetzt werden. Dieser Prozess wird auch CI/CD-Pipeline genannt. Die CI/CD-Pipeline ist im Wesentlichen ein Arbeitsablauf, mit dem der Prozess der Softwarebereitstellung automatisiert wird. Ohne die Pipeline müssten die DevOps Entwickler den Arbeitsablauf manuell durchführen, was zeitaufwändig und fehleranfällig wäre. Der Aufbau einer CI/CD-Pipeline ist nicht standardisiert, jedes DevOps-Team wählt die Tools selbst aus, mit denen sie die Pipeline aufbauen möchten. In Regel decken die verwendeten Tools aber folgende vier Phasen ab [4]:

Quellphase: In dieser Phase arbeiten Entwickler an dem Quellcode einer bestehenden Anwendung. Der Quellcode ist dabei in einem zentralen Repository wie GitHub / GitLab gespeichert. Um an dem Quellcode zu arbeiten, erstellen die Entwickler einen neuen Zweig im Repository und klonen (kopieren) diesen anschließend in einen lokalen Ordner auf dem Dateisystem. Anschließend können die Entwickler mit einer Entwicklungsumgebung in dem neuen Zweig eine neue Funktion einbinden oder bestehende Fehler beheben. Nachdem neuer Code geschrieben wurde, ~~führen die Entwickler lokale Test durch,~~ bevor sie den neuen Code in das zentrale Repository zurückübertragen (pushen) [4].

Build-Phase: Die Build-Phase ist ein Prozess bei dem eine ausführbare Software erzeugt wird [5]. Sie wird ausgelöst, wenn neuer Code in das zentrale Repository übertragen (gepusht) wird. Dabei werden die Codes der Repository-Zweige gesammelt und in eine, für den Computer verständliche, Sprache übersetzt („Code-Kompilierung“). anschließend den kompilierten Code an Bibliotheken „gelinkt“. Durch die Verlinkung werden die einzelnen Programmmodule zu einer eigenständigen, ausführbaren Software verbunden. Der Build-Vorgang wird dabei von einzelnen Build-Tools (teil der CI/CD-Pipeline) durchgeführt. Sie lesen dazu eine Anweisungsdatei, in der allgemeine Rahmenbedingungen sowie spezielle Anweisungen für die einzelnen Schritte eines Build-Prozesses steht [6]. Wenn der übertragene Code fehlerhaft ist, kann der Build-Prozess fehlschlagen. Durch die CI/CD-Pipeline erfahren die Entwickler an welcher Stelle der Build-Prozess abgebrochen wurde und welche Stelle im Code

dafür verantwortlich ist. Dadurch können die Entwickler so schnell wie möglich einen Fix implementieren [4].

Testphase: Nachdem ein fehlerfreier Build erstellt wurde, durchläuft der Build mehrere Tests, um sicherzustellen, dass der Code das macht, was er machen muss. Unterandrem gehören dazu meist Komponententest (eng.: „Unit-Tests“), bei denen der Code in kleine Einzelteile zerlegt wird, die daraufhin einzeln auf verschiedene Funktionen getestet werden. Des Weiteren können in dieser Phase auch Benutzerakzeptanz-, Sicherheits-, Last- oder andere Test durchgeführt werden. Jede Tests haben dabei andere Testkriterien [4].

Verteilungsphase: Die Verteilungsphase ist die letzte Phase der Pipeline. Während dieser Phase wird eine fertige Anwendung verteilt, bis alle Zielsysteme über die aktualisierte Version der Anwendung verfügen. Für dieses Vorgehen gibt es verschiedene Verteilungsstrategien, die angewendet werden können. Die Auswahl der besten Strategie richtet sich nach Art der Anwendung, dem Grad der Überarbeitung, der Zielumgebung und anderen Faktoren [4].



3. Stand der Technik

In diesem Kapitel wird die allgemeine Funktion einer integrierten Entwicklungsumgebung näher erläutert. Anschließend werden bestimmte Funktionen von drei speziellen Entwicklungsumgebungen analysiert und begründet, warum diese nicht genügen, um das Problem dieser Bachelorarbeit zu beheben.



3.1 Integrierte Entwicklungsumgebung (IDE)

Um einfache Programme entwickeln zu können, benötigen Softwareentwickler grundsätzlich nur zwei Werkzeuge. Einen Code-Editor, um den Quellcode zu schreiben und einen Compiler bzw. Interpreter, um die Programme in den für den Computer verständlichen, ausführbaren Maschinencode zu übersetzen. Für eine effiziente und professionelle Entwicklung von komplexeren Anwendungen sind allerdings weitere Werkzeuge notwendig, um den Entwicklern einzelne Arbeitsschritte abzunehmen oder zu erleichtern [7]. Damit bei der Entwicklung einer Anwendung nicht jedes Tool einzeln verwendet werden muss, kamen in der ersten Hälfte der 1980er Jahre sogenannte integrierte Entwicklungsumgebungen (engl.: „integrated development environment“, IDE) auf den Markt [8]. Das Besondere der IDEs ist, dass diese die einzelnen Tools, in einer gemeinsamen Benutzeroberfläche vereinen. Dadurch können verschiedene Aufgaben in der Softwareentwicklung möglichst ohne Medienbrüche bearbeitet werden. Die wichtigsten Funktionalitäten, die die meisten modernen IDEs gemeinsam haben sind z.B.:

- Direkter Aufruf des Compilers/Interpreters
- Funktionen zur effizienten Bearbeitung des Programmcodes (bspw. Syntax Highlighting oder Autovervollständigung)
- Funktionalität zur Fehlersuche und –behebung
- Versionsverwaltung der Dateien basierend auf Git
- Funktionen zur Erstellung und Verwaltung ganzer Projekte mit mehreren Dateien [7]

Letztere ist zugleich auch die wichtigste Funktionalität in Bezug auf diese Bachelorarbeit. Im weiteren Verlauf dieses Kapitels wird diese Funktionalität anhand der drei meistgenutzten IDEs für Softwareprojekte näher analysiert. Im Besonderen werden folgende Funktionen untersucht:

- Einfache Projekterstellung

- Integrierung von eigenen Template-Projekten
- Projekterstellung anhand von Template-Projekten
- Erstellung von Platzhaltern in Template-Dateien



~~Nachfolgend wird mittels der Analyse deutlich erläutert, warum IDEs nicht ausreichen, um das Problem dieser Bachelorarbeit zu lösen.~~

3.1.1 Eclipse IDE für C/C++

Eclipse ist eine kostenlose integrierte Entwicklungsumgebung für C und C++ auf Basis der Eclipse Plattform. Die IDE ist plattformübergreifend auf den Betriebssystemen Windows, Linux und macOS verfügbar [9]. Voraussetzung der Eclipse IDE ist ein vorinstallierter Java Runtime Environment (JRE), um den Betrieb von Eclipse zu gewährleisten [10].

Eclipse ist eine projektbasierte IDE. Das bedeutet, dass alles was in Eclipse passiert, im Kontext eines Projektes passiert. Diese Projekte sind dabei lediglich Ordner, in denen sich weitere Unterordner und Quelldateien befinden. Die erstellten Projekte werden standardmäßig unter einem vorgegebenen Pfad gespeichert, welcher von Eclipse als Workspace, zu Deutsch Arbeitsbereich genannt wird. Der Workspace ist dementsprechend ein Ordner in Dateisystem, in dem alle Projekte und deren Einstellungen abgelegt werden. Zusätzlich dazu werden in dem Workspace auch die Einstellungen der Plattform selbst gespeichert. Allerdings können in Eclipse nach Bedarf auch weitere Workspaces eingerichtet werden. In der geöffneten IDE kann daraufhin zu den eingerichteten Workspaces umgeschaltet werden [11].

Um ein neues Projekt zu erstellen, muss in dem Menü **File > New > Project...** ausgewählt werden. Daraufhin öffnet sich ein „New Project“-Dialog, in dem das Format des Projektes ausgewählt werden muss. Wichtig hierbei ist nur die Auswahl in dem „C/C++“ Ordner. Darin kann unter anderem zwischen einem C- bzw. C++-Projekt ausgewählt werden (In diesem Beispiel wurde ein C++ Projekt ausgewählt).

[Screenshot des Dialogs]

Nachdem einer dieser zwei Projektformate ausgewählt und auf die „Next >“-Taste gedrückt wurde, öffnet sich ein Wizard, der nach dem Namen des Projekts fragt. Alternativ zu dieser Vorgehensweise kann auch im Popup-Menü der „New C/C++

Project“-Taste der Eintrag „C Project“ bzw. „C++ Project“ ausgewählt werden, um den Wizard zu öffnen.

[Screenshot des Wizards]

Als erstes muss der Name des Projekts in dem Feld „Project Name“ eingegeben werden. Daraufhin kann entschieden werden, ob das Projekt in dem geöffneten Workspace gespeichert werden soll oder unter einem anderen Pfad. Falls das Projekt nicht in dem Workspace gespeichert werden soll, muss der Haken bei „Use default location“ entfernt werden und der entsprechende Pfad unter „Location“ eingegeben werden. Als nächstes muss der Projekttyp in dem Feld „Project type“ ausgewählt werden (In diesem Beispiel wurde ein leeres Executable Projekt ausgewählt). Als letztes muss nur noch die „Finish“-Taste gedrückt werden, um alle Angaben zu bestätigen und den Wizard zu schließen. Nun sollte in dem „C/C++ Projects“-View ein neues Projekt mit dem eingegeben Namen erscheinen [11].

[Screenshot des neuen Projektes]

Wie in der Abbildung (**Nummer der Abbildung**) gezeigt, werden neue Projekte in Eclipse nicht als leere Ordner, sondern mit einer gewissen Verzeichnisstruktur erstellt. Diese Struktur stammt aus intern gespeicherten Informationen, aus denen hervorgeht welche Unterordner und Dateien das Projekt enthalten soll. Diese Informationen können allerdings nicht ohne weiteres verändert bzw. erweitert werden. Um in Eclipse trotzdem Projekte aus einer eigenen Vorlage erstellen zu können gibt es zwei Möglichkeiten. Die erste Möglichkeit ist eine eigene Eclipse-Erweiterung zu programmieren, die es erlaubt ein neues Projektformat in Eclipse zu integrieren. Zusätzlich muss mit der Erweiterung ein neuer Projekt-Wizard aufrufbar sein, mit dessen Hilfe ein neues Projekt aus dem neuen Projektformat erstellt werden kann. Die zweite Möglichkeit ist der Import eines zuvor mit Eclipse erstellten Projektes, in den aktuellen Workspace. Dafür muss in dem Hauptmenü **File > import...** ausgewählt werden, um ein „Import“-Dialog zu öffnen.

[Screenshot des Dialogs]

Auf der ersten Seite muss **General > Existing Projects into Workspace** ausgewählt und bestätigt werden. Dadurch wird die zweite Seite des Dialogs aufgerufen, auf der

in „Select root directory“ das Verzeichnis der Projekt-Vorlage ausgewählt werden muss.

[Screenshot der neuen Seite des Dialogs]

Als letztes muss das richtige Projekt in der „Projects“-Liste ausgewählt und bestätigt werden. Dadurch wird das Projekt in den aktuellen Workspace kopiert.

Beide Möglichkeiten sind nicht ausreichend, um das Problem dieser Bachelorarbeit zu lösen. Für die Umsetzung der ersten Möglichkeit benötigt man spezielle Kenntnisse über die Architektur von Eclipse, sowie fortgeschrittene Programmierkenntnisse, um eine eigene Erweiterung für Eclipse zu entwickeln. Außerdem können mit einer solchen Erweiterung Projekte nur erstellt und nicht aktualisiert werden. Das bedeutet, dass trotz der Erweiterung in Eclipse keine Möglichkeit besteht existierende Projekte mit der Vorlage, aus der das Projekt hervorgeht, zu vergleichen und zu aktualisieren, falls die Vorlage nach der Erstellung des Projekts verändert oder erweitert wurde. Die zweite Möglichkeit ist zwar die einfachere, aber auch die schlechtere Option, da bei dem Import Befehl lediglich ein existierendes Projekt in den aktuellen Workspace kopiert wird. Das bedeutet, dass bei dieser Methode keine richtige Projekterstellung mittels einer Projekt-Vorlage abläuft, sondern nur eine einfache Kopie erstellt wird, bei der alle Metadaten im Nachhinein manuell angepasst werden müssen.

3.1.2 CLion IDE

CLion ist eine plattformübergreifende IDE von JetBrains für C/C++. Anders als Eclipse (Abschnitt 2.4.1) und VSCode (Abschnitt 2.4.2) ist CLion kostenpflichtig mit einer kostenlosen 30-tägigen Testphase [12].

Wie Eclipse ist CLion eine projektbasierte IDE, in der alles was in CLion gemacht wird, im Kontext eines Projektes passiert. Das Projekt ist eine Organisationseinheit mit einer gewissen Verzeichnisstruktur und stellt dadurch ein komplettes Softwareprojekt dar [13]. CLion unterstützt dabei vier verschiedene Projektformate: Cmake, Gradle, JSON-Kompilierungsdatenbank (dieses Format ist im Rahmen dieser Bachelorarbeit nicht weiter relevant und wird deshalb nicht weiter beachtet), GNU Makefile [14]. Allerdings können durch CLion nur Cmake-Projekte neu erstellt werden. Projekte mit den anderen drei Projektformaten müssen extern initialisiert werden und können anschließend in CLion geöffnet und bearbeitet werden [15–18]. Um ein Projekt zu öffnen, muss in CLion die Menüpunkte **File > Open...** ausgewählt werden.

[Screenshot des Menüs]

Dadurch öffnet sich ein „Pfad auswählen-Dialog“. Im Falle eines CMake-Projektes, reicht es, in diesem Dialog das Verzeichnis des Projekts auszuwählen, wenn sich in diesem Projekt eine „CMakeLists.txt“ Datei befindet. Nach dem Bestätigen dieses Dialogs, öffnet sich ein zweiter Dialog, in dem „Trust Project“ ausgewählt werden muss, um den Inhalten des Projekts zu vertrauen.

[Screenshot des Dialogs]

Als drittes öffnet sich der letzte Dialog, in dem man entscheiden kann, ob das Projekt in dem aktuellen oder einem neuen Fenster geöffnet werden soll.

[Screenshot des Dialogs]

Wenn alle drei Dialoge bestätigt wurden, sollte sich das CMake-Projekt in dem ausgewählten Fenster öffnen [19].

Für die Projekte mit anderen Formaten, muss in dem „Pfad auswählen-Dialog“ der Pfad zur Build-Datei des Projektes ausgewählt werden. Für Gradle-Projekte ist das die „build.gradle“ Datei und für Make-Projekte die „Makefile“ Datei. Nachdem die entsprechende Datei ausgewählt wurde und der „Pfad auswählen-Dialog“ bestätigt wurde, erscheint ein neuer Dialog, mit dem man die Datei mit der Taste „Open as a Project“ als Projekt öffnen kann.

[Screenshot des Dialogs]

Nach der Bestätigung dieses Dialogs folgen die gleichen zwei Dialoge wie bei dem Öffnen des CMake-Projektes (**hier die Nummer der Abbildungen**), bei denen dieselben Tasten bestätigt werden müssen, um das Projekt in dem ausgewählten Fenster zu öffnen [19].

Wie in dem zweiten Absatz dieses Kapitels erklärt, können in CLion nur CMake-Projekte neu erstellt werden. Dazu muss in dem Menü **File > New > Project...** ausgewählt werden, um das „New Project“-Dialog zu öffnen.

[Screenshot des Dialogs]

In diesem Dialog muss nun die Sprache (C oder C++) und der Zieltyp (executable oder library) des Projektes ausgewählt werden. Nachdem eine Auswahl für beide Optionen

getroffen wurde, muss in dem Eingabefeld auf der rechten Seite des Dialogs die Lage und der Name des Projekts eingegeben werden (In dem folgenden Beispiel wurde ein C++ Library Projekt ausgewählt).

[Screenshot des Eingabefelds]

Diese Information kann entweder manuell eingegeben werden oder durch Drücken der Durchsuchen-Taste, innerhalb des Eingabefelds.

[Screenshot des Buttons]

Dadurch öffnet sich nämlich ein Ordner-Dialog, mit dessen Hilfe das richtige Verzeichnis ausgewählt und bestätigt werden kann.

[Screenshot des Dialogs]

Als nächstes muss der Sprachstandard (eng.: „Language standard“) des Projektes ausgewählt werden. Mit der „Language standard“ Dropdown-Taste kann dazu eine Liste geöffnet werden, in der der gewünschte Sprachstandard ausgewählt werden muss (In diesem Beispiel wurde C++ 14 ausgewählt).

[Screenshot der Liste]

Zum Schluss muss für alle Library-Projekte ein Bibliothekstyp (eng.: „Library Type“) ausgewählt werden (Für Executable-Projekte existiert diese Auswahl nicht). Mit der Dropdown-Taste unter „Library Type“ kann hierfür zwischen „static“ und „shared“ Bibliothekstypen ausgewählt werden (In diesem Beispiel wurde shared ausgewählt).

[Screenshot der Liste]

Nachdem alle Konfigurationen durchgeführt wurden, muss die Create-Taste gedrückt werden, um ein neues CMake-Projekt zu erstellen. Dieses Projekt wird standardmäßig auf Basis eines internen Templates generiert. Das bedeutet, das Projekt besteht nach dessen Erstellung aus einer bestimmten Verzeichnisstruktur mit vordefinierten Ordnern und Dateien. Eine dieser Dateien ist beispielsweise die Root-CMakeLists.txt Datei, welche, basierend auf den bereitgestellten Informationen (Projekt Name, Sprachstandard, Bibliothekstyp), automatisch generiert wird [20].

[Screenshot der CMakeLists.txt]


Ähnlich wie Eclipse, verwendet CLion eine intern gespeicherte Projekt-Vorlage, um neue Projekte zu erstellen. Verglichen mit Eclipse, gibt es in CLion jedoch keine Möglichkeit die interne Projekt-Vorlage zu verändern oder eine neue Projekt-Vorlagen hinzuzufügen. Allerdings können in CLion selbsterstellte Datei-Vorlagen integriert werden. Dafür muss zuerst eine Datei in dem Editor geöffnet werden. Anschließend muss in dem Hauptmenü **File > Save File as Template** ausgewählt werden, um ein „Save File as Template“-Dialog zu öffnen.

[Screenshot des Dialogs]

In diesem Dialog kann daraufhin ein neuer Template Name für das Template vergeben werden. Zum Schluss müssen die Eingaben mit der „OK“-Taste bestätigt werden, um die Datei als neue Vorlage zu speichern.

Um eine Datei aus einer Vorlage zu erstellen, müssen mit der Kombination **Strg+Alt+S** die Einstellungen der DIE geöffnet werden.

[Screenshot der Einstellungen]

Darin muss zu **Editor > File and Code Templates** navigiert werden, um die Liste aller Datei Vorlagen zu öffnen. Nachdem eine Vorlage in der Liste ausgewählt wurde, muss anschließend das  -Symbol angeklickt werden. In dem neu erschienen Bereich kann dann der Name, die Dateierweiterung und wenn nötig der Textkörper der Vorlage verändert werden.

[Screenshot des Bereichs]

Zum Schluss müssen die Einstellungen mit der „OK“-Taste bestätigt werden, um die Einstellungen zu schließen und die neue Datei zu erstellen [21]. Die selbst erstellten Vorlagen können zudem auch Platzhalter beinhalten, die bei der Erstellung der realen Datei mit projektspezifischen Daten ersetzt werden. CLion bietet dafür eine Reihe von vordefinierten Platzhaltern an. Allerdings können auch selbst erstellte Platzhalter verwendet werden. Dazu müssen in den Template-Dateien das Schlüsselwort „#set“ benutzt werden [22].

Wie auch Eclipse, bietet CLion keinen ausreichenden Funktionsumfang bereit, der das Problem dieser Abschlussarbeit hinreichend lösen würde. Je nach Projektformat wird jedes neue Projekt wird in CLion nämlich, mittels einer intern gespeicherten Projekt-

Vorlage erstellt, die nicht veränderbar oder erweiterbar ist. Zudem gibt es keine Möglichkeit weitere Projekt-Vorlagen hinzuzufügen. Dadurch ist ausgeschlossen, dass das Problem der Projektaktualisierung mit CLion behoben werden kann. Auch wenn es eine Möglichkeit gäbe existierende Projekte mit der internen Vorlage zu vergleichen und diese bei unterschieden zu aktualisieren, würde sie nicht helfen, da man keinen Einfluss auf die Änderungen der internen Vorlage hat. Die einzige Option mit eigenen Vorlagen zu arbeiten, ist die erklärte Dateierstellung mittels Datei-Vorlagen. Dadurch können zwar alle benötigten Dateien, die nicht in der internen Projekt-Vorlage vorhanden sind, mittels einer Vorlage erstellt werden, doch diese Erstellungen würden dabei einzeln nacheinander passieren, was zeitaufwändig ist. Zudem gibt es auch bei den Dateien keine Möglichkeit diese zu aktualisieren.

3.1.3 Visual Studio Code

Visual Studio Code (abgekürzt VSCode) ist ein kostenloser und open source (github.com/microsoft/vscode) basierter Code-Editor von Microsoft. Es ist plattformübergreifend auf den Betriebssystemen Windows, Linux und macOS verfügbar [23].

Anders als Eclipse und CLion ist VSCode keine richtige IDE, sondern ein Quellcode-Editor, dessen Hauptzweck es ist das Schreiben von Quellcode zu vereinfachen. Der Hauptunterschied zu richtigen IDEs liegt darin, dass in VSCode viele Tools, die für die Entwicklung von Softwareprojekten wichtig sind, standardmäßig nicht integriert sind. Nichtsdestotrotz können diese Tools in dem Tab „Erweiterungen“ nachträglich in VSCode installiert werden. Zusätzlich kann VSCode auch mit weiteren Programmiersprachen erweitert werden [24].

Wie in Abschnitt 2.4 erwähnt, ist die wichtigste Funktionalität die, der Erstellung und Verwaltung ganzer Projekte mit mehreren Dateien. Um ein neues Projekt in VSCode anzulegen, muss unter **Datei > Ordner öffnen...** ein Verzeichnis ausgewählt werden, in welches ein neues Projekt angelegt werden soll.

[Screenshot des Menüs]

Nachdem ein Verzeichnis ausgewählt und geöffnet wurde, kann darin, mithilfe des VSCode Explorers, ein neuer Ordner angelegt werden. Dieser Ordner repräsentiert das Softwareprojekt.

[Screenshot des VSCode Explorers]

Diesem Ordner können im Nachhinein auf die gleiche Weise weitere Unterorder und Dateien, die für das Projekt notwendig sind, hinzugefügt werden [25].

Diese Vorgehensweise sind jedoch nicht dafür geeignet, wenn ein Softwareprojekt (wie in der Problemstellung (Abschnitt 1.1) erklärt) nach einer bestimmten Vorlage erstellt werden soll. Mit der Erweiterung „Project Templates“ von cantonios (marketplace.visualstudio.com/project-templates) kann dieses Problem allerdings gelöst werden. Nach der Installation dieser Erweiterung ist es zum einen möglich ein bestimmtes Projekt als ein Template-Projekt zu speichern. Dazu muss als erstes ein zuvor initialisiertes Projekt in VSCode geöffnet werden. Nachdem das Projekt geöffnet wurde, muss im Explorer ein Kontextmenü geöffnet werden, indem man mit der rechten Maustaste auf das Projekt klickt. Wenn die Erweiterung richtig installiert wurde, sollte nun „Save Project as Template“ als Auswahl in dem Kontextmenü erscheinen.

[Screenshot des Menüs]

Nachdem dieser Menüpunkt ausgewählt wurde, öffnet sich die Befehlspalette von VSCode, in der man einen beliebigen Namen für dieses Template-Projekt vergeben kann.

[Screenshot der Befehlspalette]

Zum Schluss muss der eingegebene Name mit der Eingabetaste auf der Tastatur bestätigt werden, um das Projekt automatisch als Template-Projekt zu speichern. Mit den Standardeinstellungen von „Project Templates“ werden alle Template-Projekte unter folgenden Pfaden gespeichert:

```
$HOME/.config/Code/User/ProjectTemplates      # Linux
$HOME/Library/Application Support/Code/User/ProjectTemplates # macOS
%APPDATA%\Code\User\ProjectTemplates          # Windows
```

Diese können aber in den Benutzereinstellungen unter **Verwalten > Einstellungen > VSCode Project Template Configuration > In „settings.json“ bearbeiten** geändert werden, indem man folgende Zeile addiert:

```
"projectTemplates.templatesDirectory": "path/to/my/templates"
```

Zum anderen kann die Extension dazu benutzt werden, um ein Projekt aus einem gespeicherten Template-Projekt zu erstellen. Dazu muss in VSCode ein Verzeichnis geöffnet werden, in dem ein leerer Ordner liegt. In diesem leeren Ordner wird das neue Projekt erstellt. Wie in dem Abschnitt zuvor muss man dafür mit der rechten Maustaste auf den leeren Ordner klicken, damit sich ein Kontextmenü öffnet. In dem Menü sollte sich der Menüpunkt „Create Project from Template“ befinden.

[Screenshot des Menüs]

Mit der Auswahl des Menüpunktes öffnet sich die Befehlspalette in VSCode mit einer Liste aller gespeicherten Template-Projekte.

[Screenshot der Befehlspalette]

Als letztes muss nur noch das gewünschte Template-Projekt ausgewählt werden. Dadurch wird der Inhalt (alle Unterordner und Dateien) des ausgewählten Template-Projektes in den leeren Ordner kopiert. Damit besitzt das neue Projekt am Ende des Vorgangs die gleiche Verzeichnisstruktur, wie das Template-Projekt.

[Screenshot des Projekts]

Darüber hinaus können in den Dateien eines Template-Projektes auch variable Platzhalter eingesetzt werden. Die Schreibweise sieht wie folgt aus:

```
Author: #{author}  
Title: #{title}
```

Wenn eine Datei aus einem Template mit Platzhaltern erstellt wird, wird man bei der Erstellung zu einer Eingabe eines Wertes aufgefordert. Platzhalter können außerdem auch in Dateinamen verwendet werden [26].

Auch die Extension „Project Templates“ ist für die Lösung des Problems dieser Arbeit nicht ausreichend. Es ist mit ihr zwar möglich ein neues Projekt anhand einer Projekt-Vorlage zu erstellen, allerdings sind die Platzhalter, die man in die Datei-Vorlagen einbinden kann, nur auf die Ersetzung durch Zeichenketten beschränkt. Es ist nicht möglich, dass Platzhalter eine Liste von vordefinierten Werten übergeben werden kann, aus der bei der Projekterstellung ein Wert ausgewählt werden kann. Des Weiteren können Platzhalter keine booleschen Werte annehmen, aus denen hervorgeht, ob manche Dateien der Projekt-Vorlage, nicht in dem erstellten Projekt

vorhanden sein sollen, diese also nicht miterstellt werden. Außerdem fehlt auch bei dieser Extension die Funktion ein Projekt mit deren Vorlage zu vergleichen und zu aktualisieren, falls die Vorlage nach der Erstellung des Projektes verändert / erweitert wurde.

~~3.2 Vergleich mit dem Projekt-Konfigurator~~

~~[Hier alle IDEs mit dem Projekt-Konfigurator vergleichen und dadurch eine Abgrenzung schaffen]~~



4. Lösungsansatz

Um das beschriebene Problem im **ersten Kapitel** zu lösen, wurde im Rahmen dieser Thesis ein spezielles DevOps Projekt-Konfigurator Tool, mit zwei Hauptfunktionen entwickelt. Die Hauptfunktionen des Tools erlauben es Projekte, mithilfe einer grafischen Benutzeroberfläche, zu erstellen und zu aktualisieren. Die Projekterstellung sowie -aktualisierung erfolgt dabei auf Basis eines intern gespeicherten Template-Projektes. Die Verzeichnisstruktur des Templates besteht aus bestimmten Unterordnern und Dateien, die für jedes neue Projekt erforderlich sind. Der Inhalt der Dateien besteht meist aus fixen Daten, in manchen befinden sich allerdings Platzhalter, die bei der Erstellung ersetzt werden müssen.

Um ein neues Projekt mit dem Projekt-Konfigurator zu erstellen, muss ein spezielles Formular in der GUI ausgefüllt werden.

[Hier noch beschreiben, dass es nicht nur Texteingabefelder gibt]

Die notwendigen Informationen des Formulars beziehen sich dabei auf die Metadaten des zu erstellenden Projektes. Nachdem das Formular ausgefüllt und bestätigt wurde, erstellt das Tool, mit Hilfe der eingegebenen Informationen und des Templates, automatisch ein neues Projekt. Die eingegebenen Informationen werden bei dem Erstellungsvorgang dafür verwendet, um die Platzhalter in den Template-Dateien, mit den entsprechenden Metadaten des Projekts zu ersetzen.

Die Aktualisierung bestehender Projekte ist die wichtigste Funktion des Projekt-Konfigurators. Mithilfe der GUI muss dafür ein bereits erstelltes Projekt im Dateisystem ausgewählt werden. Nachdem ein Projekt ausgewählt wurde, überprüft das Tool die Aktualität des Projekts, indem es das Projekt mit dem entsprechenden Template vergleicht. Wenn das Template, nach der Erstellung des ausgewählten Projektes verändert wurde, werden auf der GUI alle Unterordner und Dateien des Projekts aufgelistet, die aktualisiert werden müssen. Zusätzlich dazu beinhaltet die Auflistung auch alle Unterordner und Dateien, die dem Template neu hinzugefügt wurden und nicht in dem realen Projekt vorhanden sind. Alle Unterordner und Dateien, die sich in der Liste befinden können, anschließend automatisch durch einen Tastendruck auf der GUI aktualisiert werden.

5. Softwareentwurf

Für die Umsetzung des Lösungsansatzes, wurde eine Software mit einer grafischen Benutzeroberfläche benötigt. Die GUI ist dabei die Schnittstelle, über die die Entwickler mit dem Programmcode der Software und dem Computer interagierten. Dafür mussten als erstes die notwendigen Funktionen bestimmt werden, die zur Lösung der Problemstellung (Kapitel 1.1) unabdingbar sind. Demnach mussten die GUI am Ende dieser Bachelorarbeit folgende drei Hauptfunktionen ausführen können:

- Erstellung von Projekten auf Basis einer Vorlage
- Aktualisierung von Projekten auf Basis einer Vorlage
- Suchungsmechanismus von bestehenden Projekten

Mit letzterer können existierende Projekte gesucht und ausgewählt werden, um diese daraufhin aktualisieren zu können. Zusätzlich zu den Hauptfunktionen wurden noch zwei weitere Zusatzfunktionalitäten bestimmt, die ebenfalls in den Funktionsumfang der GUI integriert werden sollten:

- Erweiterung der Projekt Module
- Ausführung von Dienstprogrammen eines Projektes

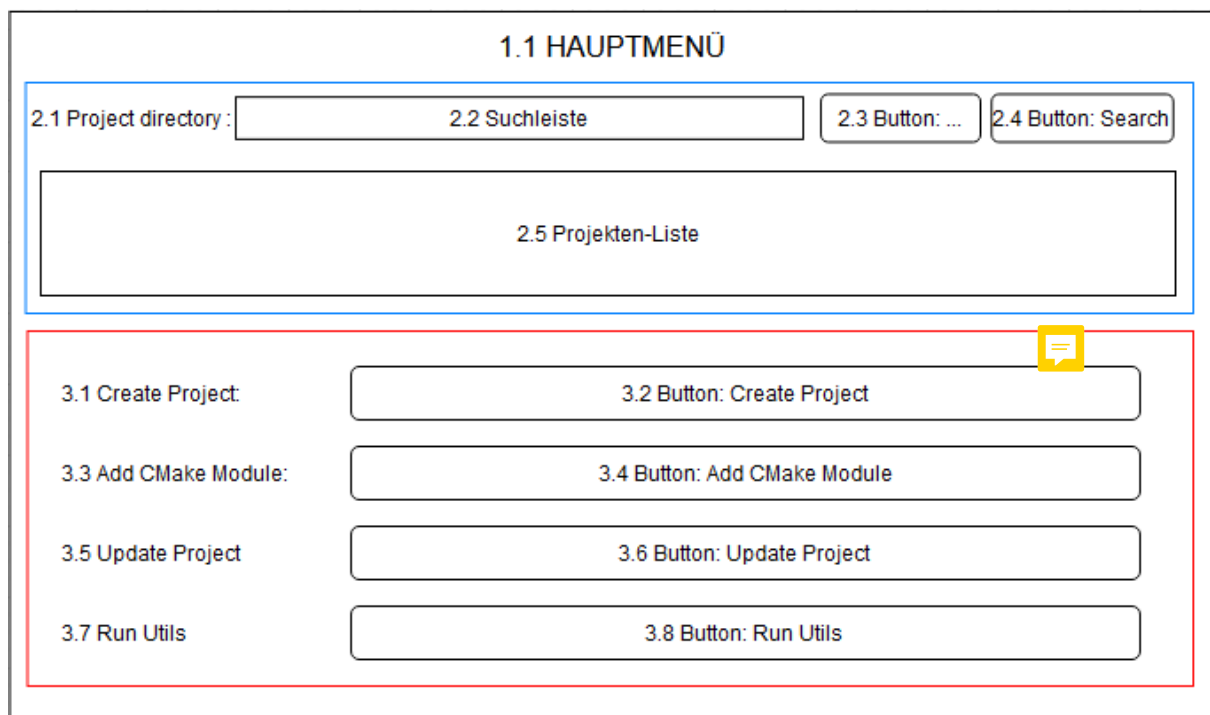
Entsprechend den Haupt- und Zusatzfunktionen musste die Benutzeroberfläche aus fünf verschiedenen Fenstern (eng.: Screen) und Unterfenstern (eng.: Subscreen) aufgebaut sein. Jeder Screen / Subscreen sollte dabei für eine bestimmte Haupt- oder Nebenfunktionalität zuständig sein. Um die Funktionen ausführen zu können, musste jeder Screen / Subscreen aus einer Reihe von bestimmten Steuerelementen (sogenannten Widgets) aufgebaut sein. Ein Widget ist dabei ein Interaktionselement der grafischen Benutzeroberfläche, wie bspw.: Ein Button, ein Textfeld, eine Dropdown-Liste oder ein Listefeld. Durch die Widgets können Entwickler mit der GUI interagieren und unter anderem bestimmte Prozesse starten, Werte eingeben, Werte auswählen oder Daten konfigurieren. Die Widgets jedes Screens / Subscreens wurden hinsichtlich ihrer auszuführenden Funktion ausgewählt. Diese Auswahl folgte an zweiter Stelle, nach der Bestimmung aller GUI-Funktionalitäten. Das Ergebnis war eine Liste von Widgets für jeden Screen / Subscreen. Mit den ausgewählten Widgets wurden anschließend verschiedene Skizzen bzw. Sketches jedes Screens / Subscreens gezeichnet, um den Aufbau jedes Fensters zu visualisieren. Die Skizzen dienten außerdem als Vorlage für die Implementierung. Im darauffolgenden Kapitel

~~werden die Skizzen der Screens / Subscreens (inklusive der Widgets) gezeigt und näher erläutert.~~ Nachdem alle Skizzen fertiggestellt wurden, wurden mithilfe der gesammelten Informationen (Funktionen, Skizzen, Widgets) ein Entwurf über die Architektur der Software, im Rahmen eines Klassendiagramms erstellt. Im übernächsten Kapitel werden das Klassendiagramm und alle enthaltenen Klassen und Methoden, sowie alle Beziehungen der Klassen zueinander, veranschaulicht und erklärt.

5.1 GUI Skizzen

Aus den herausgearbeiteten Funktionen geht hervor, dass fünf Screens / Subscreens für die GUI benötigt werden – eins für jede Funktion. Dementsprechend wurden fünf verschiedene Skizzen für jedes Fenster der GUI angefertigt. Im Folgenden wird jede Skizze gezeigt und erklärt.

5.1.1 Das Hauptmenü



~~Die erste Skizze zeigt das Hauptmenü der GUI.~~ Das Hauptmenü ist der Startbildschirm der GUI. Das bedeutet, dieses Fenster ist das erste, das die Entwickler bei der Benutzung des Projekt-Konfigurators sehen. In der Skizze ist zusehen, dass das Hauptmenü aus zwei verschiedenen Teilen (Blau und Rot) aufgebaut ist. Jeder Teil ist dabei für eine spezielle Funktion zuständig. Der erste Teil (Blau) ist für den Suchungsmechanismus von bestehen Projekten verantwortlich. In der Suchleiste (Widget 2.2 in der Skizze) kann dafür der Pfad zu dem Root-Ordner eingegeben werden, in dem ein oder mehrere Projekte liegen. Um den Pfad nicht manuell eingeben

zu müssen, kann alternativ dazu der „...“-Button (Widget 2.3 in der Skizze) benutzt werden, um einen Ordner-Dialog zu öffnen, mit dem der Root-Ordner ausgewählt werden kann. Der Pfad des Root-Ordners wird daraufhin automatisch in die Suchleiste übertragen. Anschließend kann durch das Drücken des „SEARCH“-Buttons (Widget 2.4 in der Skizze) jedes Projekt, das in dem Root-Ordner liegt, in der Projekten-Liste (Widget 2.5 in der Skizze) aufgelistet werden.

Der zweite Teil (Rot) des Hauptmenüs ist für das Wechseln auf die anderen GUI-Fenster zuständig. Jeder der vier Buttons (Widgets 3.2, 3.4, 3.6, 3.8 der Skizze), wechselt dabei auf ein anderes Fenster. Der „Create Project“-Button (Widget 3.2 in der Skizze) wechselt auf das Fenster für die Projekterstellung (Kapitel 5.1.2). Der „Add CMake Module“-Button (Widget 3.4 in der Skizze) wechselt auf das Fenster für die Erweiterung der Projektmodule (Kapitel 5.1.4). Der „Update Project“-Button (Widget 3.6 in der Skizze) wechselt auf das Fenster für die Projektaktualisierung (Kapitel 5.1.3). Der „Run Utils“-Button (Widget 3.8 in der Skizze) wechselt auf das Fenster für die Ausführung der Dienstprogramme (Kapitel 5.1.5). Die Buttons 3.4, 3.6, 3.8 sind allerdings standardmäßig inaktiv, was bedeutet, dass diese nicht gedrückt werden können. Erst wenn ein Projekt in der Projekten-Liste (Widget 2.5 in der Skizze) ausgewählt wird, werden diese drei Buttons aktiv und wieder klickbar. Der Grund ist dafür ist, dass für die Weiterarbeit in den Fenstern, auf die mithilfe der drei Buttons gewechselt wird, ein ausgewähltes Projekt benötigt wird.

~~Die sechs nicht erwähnten Widgets sind einfache Beschriftungen. Die Beschriftung 1.1 ist dabei die Hauptüberschrift des Fensters und die restlichen dienen lediglich als beschreibende Texte.~~

5.1.2 Fenster für die Projekterstellung

1.1 Formular

2.1 Project name (long):	2.2 Texteingabefeld
2.3 Project name (short):	2.4 Texteingabefeld
2.5 Dokumentation:	2.6 Texteingabefeld

3.1 Project Type:	3.2 Dropdown
3.3 Module name:	3.4 Texteingabefeld
3.5 License Type:	3.6 Dropdown
3.7 Conan Support:	3.8
3.9 Use Gtest:	3.10
3.11 Use Contribution Guide:	3.12
3.13 Linting and Formatting:	3.14
3.15 Use Gitignore:	3.16
3.17 Use GiLab CI/CD:	3.18

4.1 Authors:	4.2 Texteingabefeld	4.3 Enter
4.4 Autoren-Liste		

5.1 Project directory:	5.2 Texteingabefeld	5.3 ...
------------------------	---------------------	---------

6.1 BUTTON: Back to Menu

6.2 BUTTON: Create Project

Die zweite Skizze zeigt das Fenster, mit dem Projekte auf Basis einer Vorlage erstellt werden können. In der Skizze ist zu sehen, dass das Fenster aus vier verschiedenen Teilen (Blau, Rot, Grün und Pink), plus zwei Buttons aufgebaut ist. Alle Teile und deren Widgets tragen dazu bei Informationen über das Projekt zu sammeln, die (wie in Kapitel 4 erklärt) für die Projekterstellung benutzt werden. Der erste Teil (Blau) ist zuständig für den Projektnamen und die Projektbeschreibung. Der Projektname ist dabei in eine lange und eine kurze Version aufgeteilt. In dem ersten Eingabefeld (Widget 2.2 in der Skizze) muss der lange Projektname eingegeben werden und in dem zweiten Eingabefeld (Widget 2.4 in der Skizze) der kurze Projektname. In dem letzten Eingabefeld des Teils (Widget 2.6 in der Skizze) muss die Projektbeschreibung eingegeben werden. Dieser Teil des Fensters hat zwei Limitierungsfaktoren. Die erste Limitierung ist, dass der eingegebene lange Projektname nicht länger als 255 Zeichen sein darf. Die zweite Limitierung betrifft den eingegebenen kurzen Projektnamen. Dieser muss URL-Konform sein, was bedeutet, dass der kurze Projektname keine

Zeichen beinhalten darf, die in einer URL nicht enthalten sein dürfen. Die Projektbeschreibung enthält keinerlei Limitierungen.

Der zweite Teil (~~Rot~~) ist zuständig für das Projektmodul, die verwendeten Lizenzen und die externen verwendeten Tools. Für das Projektmodul muss zuerst ein Projekttyp ausgewählt werden. Dieser ist über die erste Dropdown-Liste (Widget 2.2 in der Skizze) auszuwählen. ~~Zur Auswahl stehen drei Projekttypen.~~ Anschließend muss ein Modulname vergeben werden. Dieser kann mithilfe des Eingabefeldes (Widget 2.4 in der Skizze) eingegeben werden. Als nächstes muss ein Lizenztyp über die zweite Dropdown-Liste (Widget 2.4 in der Skizze) ausgewählt werden. ~~Zur Auswahl stehen hierfür drei Lizenztypen.~~ Als letztes müssen alle externen Tools ausgewählt werden, die für das Projekt verwendet werden. ~~Zur Auswahl stehen sechs Tool,~~ die durch die Checkboxes (Widgets 3.6, 3.8, 3.10, 3.12, 3.14, 3.16 in der Skizze) ausgewählt werden können.

Der dritte Teil (~~Grün~~) ist für die Autoren des Projektes zuständig. In dem Eingabefeld (Widget 3.2 in der Skizze) kann dafür der Name eines Autors eingegeben werden. Mit dem Drücken des „Enter“-Buttons (Widget 3.3 in der Skizze) wird der eingegebene Name bestätigt und in die Autoren-Liste (Widget 3.4 in der Skizze) eingetragen. Dieser Vorgang kann für jeden Autor wiederholt werden. Das Ergebnis ist eine Liste bei dem jeder Listeneintrag ein Name eines Autors des Projekts ist.

Der vierte Teil (~~Pink~~) des Fensters ist für den Speicherort des erstellten Projekts verantwortlich. In dem Eingabefeld (Widget 4.2 in der Skizze) kann dazu der Pfad eingegeben werden, in dem das Projekt nach der Erstellung gespeichert werden soll. Alternativ dazu, kann (wie in Kapitel 5.1.1) der „...“-Button (Widget 4.3 in der Skizze) gedrückt werden, um ein Ordner-Dialog zu öffnen. In diesem Dialog kann anschließend der entsprechende Pfad ausgewählt und bestätigt werden. Durch die Bestätigung wird der Pfad automatisch in das Eingabefeld eingetragen (Widget 4.2 in der Skizze).

Als letztes befinden sich unten in dem Fenster noch zwei weitere Buttons (Widget 6.1, 6.2). Der „Back to Menu“-Button dient lediglich dazu die Fenster zu wechseln, um zurück auf das Hauptmenü (Kapitel 5.1.1) zu gelangen. Durch den „Create Project“-Button werden alle getätigten Eingaben in dem Fenster bestätigt und das neue Projekt

erstellt. Nachdem das Projekt erstellt wurde, wechselt die GUI wieder auf das Hauptmenü-Fenster.

~~Die 15 nicht erwähnten Widgets sind einfache Beschriftungen. Die Beschriftung 1.1 ist dabei die Hauptüberschrift des Fensters und die restlichen dienen lediglich als beschreibende Texte.~~

5.1.3 Fenster für die Projektaktualisierung

1.1 Update Project

2.1 Project Settings

2.2 Project Name (Long): 2.3 Texteingabefeld

2.4 Project Name (Short): 2.5 Texteingabefeld

2.6 License Type: 2.7 Dropdown

2.8 Conan Support: ☐ 2.9

2.10 Use Gtest: ☐ 2.11

2.12 Use Contribution Guide: ☐ 2.13

2.14 Linting and Formatting: ☐ 2.15

2.16 Use Gitignore: ☐ 2.17

2.18 Use GitLab CI/CD: ☐ 2.19

3.1 Updatable files

3.2 Liste: Aktualisierbare Dateien und Ordner

4.1.1 Button: Take Yours	4.2.1 Button: Accept Changes	4.2.2 Button: Reset Changes	4.3.1 Button: Take Theres
4.1.2 Liste: Projekt	4.2.3 Liste: Merged	4.3.2 Liste: Template	

5.1 Button: Back to Menu

5.2 Button: Update Project

[Beschreibender Text]

5.1.4 Fenster für die Erweiterung der Projektmodule

1.1 Add Cmake Modul

2.1 Select Module Type: 2.2 Dropdown

2.3 Module Name: 2.4 Texteingabefeld

2.5 Select local dependency: 2.6 Dependency Liste

2.7 External Dependency: 2.8 Texteingabefeld 2.9 Button: Add

3.1 Button: Back to Menu 3.2 Button: Add Cmake

Die vierte Skizze zeigt das Fenster, mit dem ein neues Projektmodul für das ausgewählte Projekt, aus der Projekte-Liste des Hauptmenüs (Widget 2.6 in der Skizze des Kapitels 5.1.1), erstellt werden kann. In der Skizze ist zusehen, dass das Fenster aus einem Teil (Blau), aus zwei weiteren Buttons besteht. In dem ersten und einzigen Teil werden alle Informationen über das neue Projektmodul gesammelt. Die erste Information, die benötigt wird, ist der Modultyp. Dieser kann mit der Auswahl eines Wertes aus der Dropdown-Liste (Widget 2.2 in der Skizze) bestimmt werden. Zur Auswahl stehen hierfür drei Modultypen. Als nächstes muss der Name des Moduls festgelegt werden. Dazu muss ein Modulname in das Texteingabefeld (Widget 2.4 in der Skizze) eingegeben werden. Anschließend kann eine Auswahl getroffen werden welche lokalen Abhängigkeiten dem Projektmodul hinzugefügt werden sollen. Dazu werden in der Liste (Widget 2.6 in der Skizze) alle lokalen Abhängigkeiten aufgelistet, die für das neue Projektmodul des ausgewählten Projektes ausgewählt werden können. Um eine lokale Abhängigkeit zu bestimmen, muss diese in der Liste bestätigt werden und angegeben werden, ob diese als private oder öffentliche Abhängigkeit hinzugefügt werden soll. Um das neue Projektmodul mit den eingegebenen

Informationen zu erstellen, muss der „Add Cmake“-Button (Widget 3.2 in der Skizze) gedrückt werden. Das Projektmodul wird dabei direkt in dem ausgewählten Projekt erstellt.

Mit dem „Back to Menu“-Button (Widget 3.1 in der Skizze) wechselt die GUI das Fenster zum Hauptmenü.

5.1.5 Fenster für die Ausführung der Dienstprogramme



Die fünfte Skizze zeigt das Fenster, mit dem die Dienstprogramme des ausgewählten Projektes, aus der Projekten-Liste des Hauptmenüs (Widget 2.6 in der Skizze des Kapitels 5.1.1), ausgeführt werden können. In der Skizze ist zusehen, dass das Fenster aus zwei Teilen (Blau und Rot) aufgebaut ist. Der erste Teil (Blau) besteht wiederum selbst auch aus zwei Unterteilen – die linke Seite (Widget 2.1 in der Skizze) und die rechte Seite (Widget 2.2 in der Skizze). Die linke Seite (Widget 2.1 in der Skizze) ist eine Liste, in der alle ausführbaren Dienstprogramme des ausgewählten Projektes aufgelistet werden. Die Dienstprogramme in der Liste können dabei angeklickt bzw. ausgewählt werden. Nachdem das passiert ist, wird das ausgewählte Dienstprogram mit dem Befehl „--help“ ausgeführt. Die resultierende Ausgabe wird anschließend in dem Text Browser auf der rechten Seite (Widget 2.2 in der Skizze)

ausgegeben. Eine weitere Funktion bietet der Text Browser nicht. Er dient lediglich als Bildschirm für die Ausgaben der ausgeführten Dienstprogramme.

Der zweite Teil (~~Rot~~) des Fensters, beinhaltet zwei Buttons (Widget 3.1, 3.4 in der Skizze) und ein Texteingabefeld (Widget 3.3 in der Skizze). Mit dem „Baack to Menu“-Button (Widget 3.1 in der Skizze) wechselt, wie in den Kapiteln davor, die GUI das Fenster zum Hauptmenü. In das Texteingabefeld (Widget 3.3 in der Skizze) können benutzerdefinierte Befehle für einen ausgewähltes Dienstprogram aus der Liste des ersten Teils (Widget 2.1 in der Skizze) eingegeben werden. Mit dem „Run“-Button (Widget 3.4 in der Skizze) kann das ausgewählte Dienstprogramm mit den eingegeben benutzerdefinierten Befehlen ausgeführt werden. Die resultierende Befehlsausgabe wird anschließend auf dem Text Browser ausgegeben.

5.2 Architektur der Software

Projekt Erstellung

Projekt Aktualisierung

Projekt Template Aktualisierung

5.3 Extrafunktionalitäten

Erweiterung der Projekt Module

Suchungsmechanismus von bestehenden C++ Projekten

Run utilities

6. Implementierung und Evaluierung

Internes Projekt Template

GUI Einstellungen

Projekt Erstellung

Suchungsmechanismus von bestehenden C++ Projekten

Erweiterung der Projekt Module

Projekt Template Aktualisierung

Run utilities

7. Zusammenfassung

7.1 Zusammenfassung

7.2 Das Problem

7.3 Mögliche Erweiterungspunkte

Literatur

- [1] E. Wolff, *Continuous delivery: Der pragmatische Einstieg*, 2. Aufl. Heidelberg, Germany: dpunkt.verlag, 2016. [Online]. Verfügbar unter: <https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4471176>
- [2] M. Hüttermann, *DevOps for developers*. New York, NY: Apress, 2012. [Online]. Verfügbar unter: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=1156065>
- [3] R. Alt, *Innovationsorientiertes IT-Management mit DevOps: IT im Zeitalter von Digitalisierung und Software-defined Business*. Wiesbaden: Springer Gabler, 2017.
- [4] Splunk, *Was ist eine CI/CD-Pipeline?* [Online]. Verfügbar unter: https://www.splunk.com/de_de/data-insider/what-is-ci-cd-pipeline.html (Zugriff am: 5. Januar 2022.576Z).
- [5] J. Rossberg, *Agile Project Management with Azure DevOps: Concepts, Templates, and Metrics*. Berkeley, CA: Apress L. P, 2019. [Online]. Verfügbar unter: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5771167>
- [6] S. Augsten, „Was ist ein Build?“, *Dev-Insider*, 27. Apr. 2018, 2018. [Online]. Verfügbar unter: <https://www.dev-insider.de/was-ist-ein-build-a-702737/>. Zugriff am: 24. November 2021.114Z.
- [7] J. L. Zuckarelli, *Programmieren lernen mit Python und JavaScript: Eine praxisorientierte Einführung für Einsteiger*, 1. Aufl. Wiesbaden: Springer Fachmedien Wiesbaden; Imprint Springer Vieweg, 2021.
- [8] Axel Bruns: *Die Geschichte des Computers - ebook - neobooks*. [Online]. Verfügbar unter: <https://link.springer.com/content/pdf/10.1007%2F978-1-4842-6901-5.pdf> (Zugriff am: 25. November 2021.268Z).
- [9] D. Schaefer, *Eclipse CDT | The Eclipse Foundation*. [Online]. Verfügbar unter: <https://www.eclipse.org/cdt/> (Zugriff am: 3. Januar 2022.849Z).
- [10] *C++ programmieren mit Eclipse CDT*. [Online]. Verfügbar unter: <https://www.edv-buchversand.de/productinfo.php?replace=false&cnt=productinfo&mode=2&type>

=2&id=dp-

196&index=2&nr=0&art=Blick%20ins%20Buch&preload=false&page=1&view=fit
&Toolbar=1&pagemode=none (Zugriff am: 3. Januar 2022.834Z).

- [11] S. Bauer, „Eclipse für C/C++-Programmierer – Handbuch zu den Eclipse C/C++ Development Tools (CDT)“.
- [12] JetBrains, *Intelligente Programmierunterstützung und Codeanalyse – Features | CLion*. [Online]. Verfügbar unter: <https://www.jetbrains.com/de-de/clion/features/> (Zugriff am: 29. November 2021.450Z).
- [13] CLion Help, *Quick start guide | CLion*. [Online]. Verfügbar unter: <https://www.jetbrains.com/help/clion/project.html> (Zugriff am: 8. Dezember 2021.446Z).
- [14] CLion Help, *File templates | CLion*. [Online]. Verfügbar unter: <https://www.jetbrains.com/help/clion/project-models.html> (Zugriff am: 8. Dezember 2021.086Z).
- [15] CLion Help, *File templates | CLion*. [Online]. Verfügbar unter: <https://www.jetbrains.com/help/clion/quick-cmake-tutorial.html> (Zugriff am: 8. Dezember 2021.931Z).
- [16] CLion Help, *File templates | CLion*. [Online]. Verfügbar unter: <https://www.jetbrains.com/help/clion/gradle-support.html> (Zugriff am: 8. Dezember 2021.412Z).
- [17] CLion Help, *File templates | CLion*. [Online]. Verfügbar unter: <https://www.jetbrains.com/help/clion/makefiles-support.html> (Zugriff am: 8. Dezember 2021.280Z).
- [18] CLion Help, *File templates | CLion*. [Online]. Verfügbar unter: <https://www.jetbrains.com/help/clion/compilation-database.html> (Zugriff am: 8. Dezember 2021.290Z).
- [19] CLion Help, *Open, reopen, and close projects | CLion*. [Online]. Verfügbar unter: <https://www.jetbrains.com/help/clion/opening-reopening-and-closing-projects.html> (Zugriff am: 10. Dezember 2021.803Z).

- [20] CLion Help, *Quick CMake tutorial | CLion*. [Online]. Verfügbar unter:
<https://www.jetbrains.com/help/clion/creating-new-project-from-scratch.html>
(Zugriff am: 21. Dezember 2021.578Z).
- [21] CLion Help, *File templates | CLion*. [Online]. Verfügbar unter:
<https://www.jetbrains.com/help/clion/using-file-and-code-templates.html> (Zugriff
am: 12. Januar 2022.637Z).
- [22] CLion Help, *Quick CMake tutorial | CLion*. [Online]. Verfügbar unter:
[https://www.jetbrains.com/help/clion/using-file-and-code-templates.html#create-](https://www.jetbrains.com/help/clion/using-file-and-code-templates.html#create-new-template)
[new-template](https://www.jetbrains.com/help/clion/using-file-and-code-templates.html#create-new-template) (Zugriff am: 21. Dezember 2021.379Z).
- [23] A. Del Sole, *Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux*, 2. Aufl. Berkeley, CA: Apress; Imprint Apress, 2021.
- [24] *Managing Extensions in Visual Studio Code*. [Online]. Verfügbar unter:
<https://code.visualstudio.com/docs/editor/extension-marketplace> (Zugriff am: 14.
Januar 2022.023Z).
- [25] *Visual Studio Code Tips and Tricks*. [Online]. Verfügbar unter:
https://code.visualstudio.com/docs/getstarted/tips-and-tricks#_files-and-folders
(Zugriff am: 29. November 2021.615Z).
- [26] *Project Templates - Visual Studio Marketplace*. [Online]. Verfügbar unter:
[https://marketplace.visualstudio.com/items?itemName=cantonios.project-](https://marketplace.visualstudio.com/items?itemName=cantonios.project-templates)
[templates](https://marketplace.visualstudio.com/items?itemName=cantonios.project-templates) (Zugriff am: 29. November 2021.523Z).

Anhang