

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 und Stand der Technik	14
2.1 Die Bedeutung von DevOps	14
2.2 Continuous Integration, Continuous Delivery (CI/CD)	14
2.3 Build-Tools	15
2.4 Integrierte Entwicklungsumgebung (IDE)	15
2.4.1 Eclipse C/C++ Development Toolkit (CDT)	16
2.4.2 Visual Studio Code	19
2.4.3 CLion IDE	21
2.5 Problem der IDEs	Fehler! Textmarke nicht definiert.
3. Lösungsansatz	27
4. Softwareentwurf	28
4.1 GUI Sketches	28
4.2 Architektur	28
4.3 Extrafunktionalitäten	28
5. Implementierung und Evaluierung	30

6. Zusammenfassung	31
6.1 Zusammenfassung	31
6.2 Das Problem	31
6.3 Mögliche Erweiterungspunkte	31
Literatur	Fehler! Textmarke nicht definiert.
Anhang	35

Abbildungsverzeichnis

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

Tabellenverzeichnis

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

Abkürzungsverzeichnis

CDT *C/C++ Development Toolkit*

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 und Stand der Technik

2.1 Die Bedeutung von DevOps

Bei der Bezeichnung DevOps handelt es sich um ein sogenanntes 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]. Die Softwareentwicklung arbeitet meist in **Projektform** an der Umsetzung von Kundenanforderungen. 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].

Ein weiteres Problem der klassischen Organisationsstruktur 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].

2.2 Continuous Integration, Continuous Delivery (CI/CD)

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-Pipelineprozess 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 aus der sie die Pipeline aufbauen möchten selbst aus [4].

2.3 Build-Tools

Ein Build-Prozess ist ein Vorgang bei dem eine ausführbare Software erzeugt wird [5]. Abgesehen von einfachen Programmen, besteht jede Software aus vielen Quelldateien und externen Elementen. Diese müssen vor der Ausführung der Software in ein lauffähiges Konstrukt umgewandelt werden. Typischerweise wird dabei der Quellcode zuerst in eine, für den Computer verständliche, Sprache übersetzt („Code-Kompilierung“) und anschließend den kompilierten Code an Bibliotheken „gelinkt“. Durch die Verlinkung werden die einzelnen Programmmodule zu einer eigenständigen, ausführbaren Software verbunden [6].

Früher haben Softwareentwickler diese Arbeitsschritte manuell gelöst. Doch mit der Zeit wurden Softwareprojekte immer umfangreicher, wodurch auch der Aufwand für die Kompilierung und Verknüpfung vieler verschiedener Dateien immer weiter angestiegen ist [7]. Aus diesem Grund werden bei komplexen Softwareprojekten heutzutage Build-Tools eingesetzt, um den gesamten Build-Prozess einfacher zu gestalten. Diese Tools sind dabei einzelne Softwarelösungen, die in dem Build-Prozess nacheinander eingesetzt werden, um bestimmte Prozesse zu automatisieren. Sie lesen dazu eine Konfigurationsdatei, in der allgemeine Rahmenbedingungen sowie spezielle Anweisungen für die einzelnen Schritte eines Build-Prozesses steht. Jeder Schritt kann dabei von verschiedenen Bedingungen abhängig sein, wie zum Beispiel die Aktualität oder Existenz bestimmter Dateien. Dadurch wird einerseits sichergestellt, dass die einzelnen Schritte in der richtigen Reihenfolge durchgeführt werden, andererseits kann dadurch unnötige Arbeit vermieden werden, indem noch aktuelle Teilergebnisse in weiteren Durchläufen wiederverwendet werden und nicht in jedem Arbeitsschritt neu generiert werden [6].

2.4 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 [8]. 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 [9]. 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 [8]

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 C++ Projekte 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.

2.4.1 Eclipse C/C++ Development Toolkit (CDT)

Eclipse CDT 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 [10]. Voraussetzung des Eclipse CDT ist ein vorinstallierter C++ Compiler, weitere Hilfsprogramme, wie make,

Id, cpp, sowie eine Java Runtime Environment (JRE), um den Betrieb von Eclipse zu gewährleisten [11].

Eclipse CDT ist eine **projektbasierte** IDE. Das bedeutet, dass alles was in Eclipse CDT 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 im 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 CDT nach Bedarf auch weitere Workspaces eingerichtet werden. In der geöffneten IDE kann daraufhin zu den eingerichteten Workspaces umgeschaltet werden [12].

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 [12].

[Screenshot des neuen Projektes]

Wie in der Abbildung (**Nummer der Abbildung**) gezeigt, werden neue Projekte in Eclipse CDT 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 verändert bzw. erweitert werden. Dadurch können in Eclipse CDT weitere Projektformate hinzugefügt werden, welche bei dem „New Project“-Dialog (**Nummer der Abbildung**) oder dem Popup-Menü der „New C/C++ Project“-Taste ausgewählt werden können. Aus diesen Projektformaten könne wiederum neue Projekte mit einem Wizard erstellt werden. Die Umsetzung dieser Möglichkeit ist allerdings kompliziert und aufwändig. Außerdem würde die Beschreibung der Vorgehensweise den Rahmen dieser Bachelorarbeit überschreiten. Dessen ungeachtet gibt es in Eclipse CDT eine weitere Möglichkeit neue Projekte aus selbst erstellten Vorlagen zu erstellen. Dazu muss vorab ein fertiges Template-Projekt, mit notwendigen Unterordnern und Dateien, in Eclipse erstellt und in einem beliebigen Workspace gespeichert werden (wie in den letzten Abschnitten dieses Kapitels erklärt). Das gespeicherte Projekt kann daraufhin mit einem Import in jedes beliebige Workspaces kopiert werden. Um dies zu erreichen, muss in dem Menü **File > import...** ausgewählt werden, um ein „Import“-Dialog zu öffnen.

[Screenshot des Dialogs]

In diesem Dialog muss nun **General > Existing Projects into Workspace** ausgewählt und die „Next >“-Taste gedrückt werden. Dadurch wird die nächste 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 man in der „Projects“-Liste das zu importierende Projekt auswählen und auf die „Finish“-Taste drücken. Dadurch wird das Projekt in den aktuellen Workspace kopiert. Die Kopie kann daraufhin umbenannt und erweitert werden. Dieser

Import Prozess kann in einem Workspace beliebig oft durchgeführt werden, solange es bei den Kopien zu keinen Namenskonflikten kommt.

2.4.2 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 [13].

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 Unterschied zu richtigen IDEs ist, dass VSCode nicht Projektbasiert arbeitet, sondern auf Basis von Dateien sowie Orderbasierten Projekten. Die Art der Datei spielt dabei keine Rolle, da VSCode fast mit jeder modernen Programmiersprache erweiterbar ist. In dem Tab „Erweiterungen“ muss dafür schlichtweg die jeweilige Sprache gesucht und anschließend installiert werden [13]. Ein weiterer Unterschied ist, dass VSCode keinen integrierten Compiler besitzt. Für die Entwicklung mit Programmiersprachen bei denen Compiler benötigt werden, müssen diese manuell heruntergeladen, installiert und mit VSCode verlinkt werden [14].

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 wurde, kann darin ein neuer Ordner angelegt werden, welcher das neue Projekt repräsentieren soll.

[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 [15].

Eine weitere Möglichkeit ein neues Projekt zu erstellen ist bietet das integrierte Terminal in VSCode. Dazu kann unter **Terminal > Neues Terminal** ein neues Terminal geöffnet werden.

[Screenshot des Menüs]

In diesem Terminal können dann alle notwendigen Befehle eingegeben werden, um ein neuen Ordner unter dem gewünschten Verzeichnis zu erstellen und diesen danach mit notwendigen Quelldateien zu füllen [15].

Diese Vorgehensweisen 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 [16].

2.4.3 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 [17].

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 [18]. 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 [19]. 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 [20–23]. 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 [24].

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 [24].

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 [25].

[Screenshot der CMakeLists.txt]

Im Vergleich zu den anderen vorgestellten IDEs gibt es bei CLion keine Möglichkeit das intern verwendete Template zu verändern, oder neue Template-Projekte anzulegen, auf dessen Basis neue CMake-Projekte erstellt werden können. Das bedeutet, dass die Verzeichnisstruktur eines erstellten CMake-Projekts manuell verändert werden muss, wenn einige Dateien/Ordner benötigt bzw. nicht benötigt werden.


Allerdings gibt es in CLion eine Möglichkeit Dateien aus Template-Dateien zu erstellen. Standardmäßig stellt CLion dafür eine Liste von vordefinierten Datei-Vorlagen zur Verfügung. Diese sind intern und können nicht gelöscht oder bearbeitet werden. Ungeachtet dessen kann die Liste mit eigenen Vorlagen erweitert 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 [26]. 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 [27].

2.5 Das Problem der Projektaktualisierung

Wie in dem letzten Kapitel gezeigt, stellen alle IDEs verschiedene Möglichkeiten bereit, um neue Projekte zu erstellen. Zusätzlich dazu können bei den meisten Entwicklungsumgebungen einige Einstellungen/Konfigurationen vorgenommen werden, um ein eigenes Template-Projekt einbinden zu können, damit auf dessen Basis weitere Projekte erstellt werden können. Durch die Problemstellung (Abschnitt 1.1) im ersten Kapitel wird allerdings deutlich, dass es nicht unwahrscheinlich, dass sich der Aufbau eines Templates im Laufe eines Entwicklungsprozesses aus unterschiedlichen Gründen ändern kann. Durch die Änderung eines Templates müssen im Nachhinein alle Projekte, die aus diesem Template entstanden sind, ebenfalls aktualisiert werden. Über eine solche Funktionalität verfügt jedoch keine der derzeit verfügbaren IDEs auf dem Markt. Deswegen wurde im Rahmen dieser Thesis ein Projekt-Konfigurator Tool entwickelt, der sich mit dieser Funktionalität von gewöhnlichen Entwicklungsumgebungen abgrenzt. Entwickler können mit dem Tool bereits erstellte Projekte im Dateisystem suchen und diese auf ihre Aktualität überprüfen lassen. Bei diesem Prozess werden die ausgewählten Projekte mit dem Template verglichen werden, aus dem diese entstanden sind. Falls das Template verändert wurde, nachdem das ausgewählte Projekt erstellt wurde, bietet das Tool den Entwicklern eine Möglichkeit das Projekt auf den neusten Template-Stand zu

aktualisieren. Die Aktualisierung erfolgt automatisch und muss nicht manuell von den Entwicklern durchgeführt werden.

3. 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. 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.

In den folgenden Kapiteln 4 und 5 werden alle Funktionen des Projekt-Konfigurators ausführlicher mithilfe einiger Diagramme erklärt. Darunter befinden sich auch drei Extrafunktionalitäten, die bisher noch nicht genannt wurden.

4. Softwareentwurf

Für die Umsetzung des Lösungsansatzes, wurde eine Software mit einer grafischen Benutzeroberfläche benötigt. Die integrierten Funktionalitäten wurden aus der Problemstellung (Kapitel 1.1) abgeleitet. Demnach sollte man mit der GUI folgende fünf Aktionen ausführen können:

- Projekterstellung auf Basis einer Vorlage
- Projektaktualisierung auf Basis einer Vorlage
- Erweiterung der Projekt Module
- Ausführung von utilities
- Suchungsmechanismus von bestehenden C++ Projekten

Diesbezüglich muss die Benutzeroberfläche aus fünf verschiedenen Fenstern (eng.: Screen) und Unterfenstern (eng.: Subscreen) aufgebaut sein. Jeder Screen / Subscreen ist dabei für eine spezifische Funktion zuständig.

- Für Lösung wird GUI mit verschiedenen Seiten benötigt
- Jede Seite soll eine bestimmte Aufgabe erledigen
- Aus Aufgaben wurden alle benötigten Funktionalitäten abgeleitet
- Anschließend wurden den Funktionen passende Widgets zugeteilt
- Aus diesen Informationen wurden Sketches aller GUI Seiten gemacht
- Daraus wurde ein Klassendiagramm mit allen Klassen und Funktionen gefertigt
- Dann Klassendiagramm mit Python und PyQt umgesetzt

4.1 GUI Sketches

4.2 Architektur

Projekt Erstellung

Projekt Aktualisierung

Projekt Template Aktualisierung

4.3 Extrafunktionalitäten

Erweiterung der Projekt Module

Suchungsmechanismus von bestehenden C++ Projekten

Run utilities

5. Implementierung und Evaluierung

Internes Projekt Template

GUI Einstellungen

Projekt Erstellung

Suchungsmechanismus von bestehenden C++ Projekten

Erweiterung der Projekt Module

Projekt Template Aktualisierung

Run utilities

6. Zusammenfassung

6.1 Zusammenfassung

6.2 Das Problem

6.3 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] P. D. Crutcher, N. K. Singh und P. Tiegs, *Essential Computer Science: A Programmer's Guide to Foundational Concepts*, 1. Aufl. Berkeley, CA: Apress; Imprint Apress, 2021.
- [8] J. L. Zuckarelli, *Programmieren lernen mit Python und JavaScript: Eine praxisorientierte Einführung für Einsteiger*, 1. Aufl. Wiesbaden: Springer Fachmedien Wiesbaden; Imprint Springer Vieweg, 2021.
- [9] 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).
- [10] D. Schaefer, *Eclipse CDT | The Eclipse Foundation*. [Online]. Verfügbar unter: <https://www.eclipse.org/cdt/> (Zugriff am: 3. Januar 2022.849Z).

- [11] *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).
- [12] S. Bauer, „Eclipse für C/C++-Programmierer – Handbuch zu den Eclipse C/C++ Development Tools (CDT)“.
- [13] A. Del Sole, *Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux*, 2. Aufl. Berkeley, CA: Apress; Imprint Apress, 2021.
- [14] B. Ayodeji, „How to compile your C++ code in Visual Studio Code“, *freeCodeCamp.org*, 7. Okt. 2019, 2019. [Online]. Verfügbar unter:
<https://www.freecodecamp.org/news/how-to-compile-your-c-code-in-visual-studio-code/>. Zugriff am: 5. Januar 2022.278Z.
- [15] *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).
- [16] *Project Templates - Visual Studio Marketplace*. [Online]. Verfügbar unter:
<https://marketplace.visualstudio.com/items?itemName=cantonios.project-templates> (Zugriff am: 29. November 2021.523Z).
- [17] JetBrains, *Intelligente Programmierunterstützung und Codeanalyse – Features | CLion*. [Online]. Verfügbar unter: <https://www.jetbrains.com/de-de/clion/features/> (Zugriff am: 29. November 2021.450Z).
- [18] CLion Help, *Quick start guide | CLion*. [Online]. Verfügbar unter:
<https://www.jetbrains.com/help/clion/project.html> (Zugriff am: 8. Dezember 2021.446Z).
- [19] CLion Help, *File templates | CLion*. [Online]. Verfügbar unter:
<https://www.jetbrains.com/help/clion/project-models.html> (Zugriff am: 8. Dezember 2021.086Z).

- [20] CLion Help, *File templates* | *CLion*. [Online]. Verfügbar unter:
<https://www.jetbrains.com/help/clion/quick-cmake-tutorial.html> (Zugriff am: 8. Dezember 2021.931Z).
- [21] CLion Help, *File templates* | *CLion*. [Online]. Verfügbar unter:
<https://www.jetbrains.com/help/clion/gradle-support.html> (Zugriff am: 8. Dezember 2021.412Z).
- [22] CLion Help, *File templates* | *CLion*. [Online]. Verfügbar unter:
<https://www.jetbrains.com/help/clion/makefiles-support.html> (Zugriff am: 8. Dezember 2021.280Z).
- [23] CLion Help, *File templates* | *CLion*. [Online]. Verfügbar unter:
<https://www.jetbrains.com/help/clion/compilation-database.html> (Zugriff am: 8. Dezember 2021.290Z).
- [24] 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).
- [25] 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).
- [26] 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).
- [27] CLion Help, *Quick CMake tutorial* | *CLion*. [Online]. Verfügbar unter:
<https://www.jetbrains.com/help/clion/using-file-and-code-templates.html#create-new-template> (Zugriff am: 21. Dezember 2021.379Z).

Anhang