C:\Users\haller\Desktop\Logo_HFU.tif

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](#_Toc88043930)

[Abstract 3](#_Toc88043931)

[Abbildungsverzeichnis 6](#_Toc88043932)

[Tabellenverzeichnis 7](#_Toc88043933)

[Abkürzungsverzeichnis 8](#_Toc88043934)

[1. Einleitung 9](#_Toc88043935)

[1.1 Nutzen dieser Bachelorarbeit 10](#_Toc88043936)

[1.2 Die Problemstellung 10](#_Toc88043937)

[1.2.1 Die Projekterstellung 11](#_Toc88043938)

[1.2.2 Die Projektaktualisierung 11](#_Toc88043939)

[1.3 Ziel dieser Bachelorarbeit 12](#_Toc88043940)

[1.4 Weiterer Aufbau 12](#_Toc88043941)

[2. Grundlagen und Stand der Technik 14](#_Toc88043942)

[2.1 Die Bedeutung von DevOps 14](#_Toc88043943)

[2.2 Continuous Integration, Continuous Delivery (CI/CD) 14](#_Toc88043944)

[2.3 Build-Management-Tools 14](#_Toc88043945)

[2.4 Frameworks 14](#_Toc88043946)

[2.5 Integrierte Entwicklungsumgebung (IDE) 14](#_Toc88043947)

[2.5.1 Eclipse IDE 15](#_Toc88043948)

[2.5.2 Visual Studio Code IDE 15](#_Toc88043949)

[2.5.3 CLion IDE 15](#_Toc88043950)

[3. Lösungsansatz 16](#_Toc88043951)

[4. Softwareentwurf 17](#_Toc88043952)

[4.1 GUI Sketches 17](#_Toc88043953)

[4.2 Architektur 17](#_Toc88043954)

[4.3 Extra Funktionalitäten 17](#_Toc88043955)

[5. Implementierung und Evaluierung 18](#_Toc88043956)

[6. Zusammenfassung 19](#_Toc88043957)

[6.1 Zusammenfassung 19](#_Toc88043958)

[6.2 Das Problem 19](#_Toc88043959)

[6.3 Mögliche Erweiterungspunkte 19](#_Toc88043960)

[Literaturverzeichnis 20](#_Toc88043961)

[Anhang 21](#_Toc88043962)

# Abbildungsverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

# Tabellenverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

# Abkürzungsverzeichnis

**DevOps** Development und IT Operations

**GUI** Graphical User Interface

**IDE** Integrated Development Environment

**UML** Unified Modeling Language

# 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.

## 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.

## 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 notwenigen 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.

### 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.

### 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.

## 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:

## Weiterer Aufbau

In Kapitel 2 wird der aktuelle Stand der Technik bezüglich aktuellen 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.

# Grundlagen und Stand der Technik

## 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 (vgl. Wolff 2016; vgl. Hüttermann 2012). 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 (vgl. Wolff 2016).

Ein weiteres Problem der klassischen Organisationstruktur 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 (vgl. Alt 2017).

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 (vgl. Alt 2017).

## Continuous Integration, Continuous Delivery (CI/CD)

Damit der DevOps-Ansatz erfolgreich umgesetzt werden kann, werden hierzu viele einzelne Werkzeuge (Tools) für bestimmte Aufgaben eingesetzt. Diese Tools müssen allerdings intelligent miteinander kombiniert werden, damit der DevOps-Prozess fehlerfrei funktioniert. Der Projekt-Konfigurator, ist eines dieser speziellen Werkzeuge.

## Build-Tools

Ein Build-Prozess ist ein Vorgang bei dem eine ausführbare Software erzeugt wird (vgl. Rossberg 2019). 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 (vgl. Augsten 2018).

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 (vgl. Crutcher et al. 2021). Aus diesem Grund werden Build-Prozesse bei komplexen Softwareprojekten mit Build-Tools automatisiert. Diese Tools 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 (vgl. Augsten 2018).

## Frameworks

## 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 (vgl. Zuckarelli 2021). 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 (vgl. Bruns 2015). 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
* Funktionen zur Erstellung und Verwaltung ganzer Projekte mit mehreren Dateien (vgl. Zuckarelli 2021)

Letztere ist zugleich auch die wichtigste Funktionalität in Bezug auf diese Bachelorarbeit. Im Folgenden wird diese Funktionalität anhand der drei meistgenutzten IDEs für C++ Projekte näher erläutert und deutlich gemacht, warum IDEs nicht ausreichen, um das Problem dieser Bachelorarbeit zu lösen.

### Eclipse IDE

### Visual Studio Code IDE

### CLion IDE

# Lösungsansatz

Bei der Projekterstellung, führt der Projekt-Konfigurator vorgegebene Konfigurationen an dem Projekt-Template aus. Die Konfigurationsmaßnahmen werden von dem Softwareentwickler selbst bestimmt. Durch die Konfigurationen entsteht am Ende ein neues einzigartiges Projekt.

Bei der Projektaktualisierung prüft der Projekt-Konfigurator die Version eines vorgegebenen Softwareprojekts verglichen mit der Version des Projekt-Templates. Wenn die Version des Projekt-Templates neuer ist, teilt der Projekt-Konfigurator dem bedienenden Softwareentwickler mit welche Ordner und Dateien des Projekts aktualisierbar sind.

Das erste Problem löst der Projekt-Konfigurator, indem er den Entwicklern in der GUI ein Formular zu Verfügung stellt. In dieses Formular müssen die Mitarbeiter alle Metadaten des zu erstellenden Softwareprojekts eintragen. Auf Basis der eingegebenen Daten, erstellt der Projekt-Konfigurator ein neues konfiguriertes Projekt.

Das bedeutet, dass die Entwickler die Daten des neuen Softwareprojektes nicht mehr manuell ändern müssen, sondern alles automatisiert erstellt und angepasst wird. Alle notwendigen Änderungen/Anpassungen werden durch das Werkzeug gemacht und müssen nicht umständlich per Hand erfolgen. Dadurch werden die Projekte effizient und fehlerfrei erstellt.

Das zweite Problem löst der Projekt-Konfigurator, indem er die Template-version eines ausgewählten Softwareprojekts überprüft und daraufhin selbst entscheidet, ob das Projekt auf den neusten Template-Stand beinhaltet oder nicht. Wenn das Softwareprojekt nicht auf dem neusten Template-Stand ist, gibt der Projekt-Konfigurator den Entwicklern Möglichkeiten vor, um das Projekt zu aktualisieren. Dieser Mechanismus ist der Hauptteil dieser Abschlussarbeit.

# Softwareentwurf

## GUI Sketches

## Architektur

Projekt Erstellung

Projekt Aktualisierung

Erweiterung der Projekt Module

Projekt Template Aktualisierung

## Extra Funktionalitäten

Suchungsmechanismus von bestehenden C++ Projekten

Run utilities

# Implementierung und Evaluierung

Internes Projekt Template

GUI Einstellungen

Projekt Erstellung

Suchungsmechanismus von bestehenden C++ Projekten

Erweiterung der Projekt Module

Projekt Template Aktualiserung

Run utilities

# Zusammenfassung

## Zusammenfassung

## Das Problem

## Mögliche Erweiterungspunkte

Literaturverzeichnis

Alt, Rainer (2017): Innovationsorientiertes IT-Management mit DevOps. IT im Zeitalter von Digitalisierung und Software-defined Business. Wiesbaden: Springer Gabler (SpringerLink Bücher).

Augsten, Stephan (2018): Was ist ein Build? In: *Dev-Insider*, 27.04.2018. Online verfügbar unter https://www.dev-insider.de/was-ist-ein-build-a-702737/, zuletzt geprüft am 24.11.2021.114Z.

Crutcher, Paul D.; Singh, Neeraj Kumar; Tiegs, Peter (2021): Essential Computer Science. A Programmer’s Guide to Foundational Concepts. 1st ed. 2021. Berkeley, CA: Apress; Imprint Apress (Springer eBook Collection).

Hüttermann, Michael (2012): DevOps for developers. New York, NY: Apress. Online verfügbar unter https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=1156065.

Rossberg, Joachim (2019): Agile Project Management with Azure DevOps. Concepts, Templates, and Metrics. Berkeley, CA: Apress L. P. Online verfügbar unter https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5771167.

Wolff, Eberhard (2016): Continuous delivery. Der pragmatische Einstieg. 2., aktualisierte und erweiterte Auflage. Heidelberg, Germany: dpunkt.verlag. Online verfügbar unter https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4471176.

Zuckarelli, Joachim L. (2021): Programmieren lernen mit Python und JavaScript. Eine praxisorientierte Einführung für Einsteiger. 1st ed. 2021. Wiesbaden: Springer Fachmedien Wiesbaden; Imprint Springer Vieweg (Springer eBook Collection).

Bruns, Axel (2015): Die Geschichte des Computers. Wie es bis zur Form des heutigen ‚PC‘ kam. Berlin: neobooks Self-Publishing. Online verfügbar unter https://www.neobooks.com/ebooks/axel-bruns-die-geschichte-des-computers-ebook-neobooks-43533?toplistType=undefined

# Anhang