Philipps-Universität Marburg

Fachbereich 12 - Mathematik und Informatik

Siemens Marburg

Analyse und Vergleich verschiedener Softwaretechnologien zur Entwicklung einer neuen Technologieplattform für die Siemens Applikation: „Opcenter Execution Pharma“

von Jonas Borgerding

Erklärung
Ich, Jonas Borgerding (Informatikstudent an der Philipps-Universität Marburg, Matrikelnummer 3111336), versichere an Eides statt, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Die hier vorliegende Bachelorarbeit wurde weder in ihrer jetzigen noch in einer ähnlichen Form einer Prüfungskommission vorgelegt.

Stadtallendorf, 29. Dezember 2020

Jonas Borgerding

Inhaltsverzeichnis

1 Einleitung

In dieser Bachelorarbeit wird untersucht, welche Technologien existieren und welche am besten für die Neuentwicklung des bestehenden Systems, „Opcenter Execution Pharma“ (siehe 3), unter mehreren Aspekten am besten geeignet ist. Hierzu ist die Bachelorarbeit viergegliedert. Im ersten Teil wird eine Technologieplattform ausgewählt, in welcher Entwickelt werden soll (siehe 4). Der zweite Teil ist die Auswahl von geeigneten Programmiersprachen für die gewählte Technologieplattform (siehe 5). Im dritten Teil werden für die gewählten Programmiersprachen passende Frameworks herausgesucht (siehe 6). Der letzte Teil wird eine Neuentwicklung eines Moduls des bestehenden Systems in wenigen Frameworks beinhalten 7. Anschließend wird eine Empfehlung für ein Framework gegeben (siehe 8).

Der Quellcode zu den Performanztests und den Modulen ist in folgenden Repositories zu finden:
`https://segit.mathematik.uni-marburg.de/Borgerdi/Bachelorarbeit_Jonas_Borgerding`
`https://github.com/JonasBorgerding/Bachelorarbeit_Jonas_Borgerding`

2 Grundlagen

2.1 Technologieplattformen

Für die Technologieplattformauswahl werden folgende Technologien betrachtet:

1. Native Anwendungen
2. Cross-Platform Anwendungen
3. Web Anwendungen
4. Hybride Anwendungen

Jede dieser Technologien erhält eine kleine Einleitung, was diese Technologie überhaupt ist. Außerdem werden hier einige Vor- und Nachteile zu den einzelnen Technologien erwähnt.
Als Informationssuche dient die Suche auf „Google“ nach den jeweiligen Technologien, gefolgt von „definition“, „advantages“ und „disadvantages“.

2.1.1 Native-Entwicklung

Was ist eine native Entwicklung?
Unter einer nativen Entwicklung wird verstanden, dass eine Anwendung für ein bestimmtes Betriebssystem geschrieben wird, diese ist dann auch nur auf diesem Betriebssystem direkt ausführbar.
Hierfür wird die Anwendungen in Programmiersprachen geschrieben, die überwiegend für das Betriebssystem verwendet wurden. Beispielsweise werden für Android Java und Kotlin, für iOS werden Objective C und Swift verwendet.
Native Anwendungen werden direkt auf den Computer des Benutzers installiert.[NAD] Als Computer werden hier alle Geräte bezeichnet, die eine Benutzeroberfläche zur Interaktion haben und auf denen eigene Programme installiert werden können (Z.B. Desktop PC, Laptop, Smartphone).

Vorteile einer nativen-Entwicklung
Native Anwendungen können eine große Menge an Funktionen bieten, da diese auf Komponenten vom Betriebssystem, wie Beispielsweise den Standort oder die Kamera, zugreifen können.
Weil Programmiersprachen verwendet werden, mit denen das Betriebssystem geschrieben wurden, ist die geschriebene native Anwendung schnell von der Performanz.
Bei der Gestaltung der Anwendung können die Betriebssystem spezifischen UI Komponenten verwendet werden, wodurch die Anwendung dem Nutzer vertrauter vorkommt. [NAD] UI bedeutet „User Interface“(dt. Benutzeroberfläche).
Die Anwendung kann sehr lange „leben“, d.h. unter anderem auf dem Rechner laufen und genutzt werden. Das Lebensende kann teils erst auftreten, wenn das Betriebssystem nicht mehr weiterentwickelt wird.

Eine Offline-Ausführung der nativen Anwendung ist möglich, wenn keine Daten von einem Server geholt werden müssen.
Außerdem kann bei der Entwicklung die neueste SDK für das Betriebssystem verwendet werden, wodurch neue Änderungen direkt verwendet werden können.[BNA] Ein SDK (Software Development Kit) ist eine Sammlung von Entwicklungswerkzeugen, wie einen Debugger und Compiler, die die Entwicklung von Anwendungen vereinfacht.

Nachteile einer nativen-Entwicklung
Der wohl größte Nachteil ist, dass keine Plattformübergreifende Anwendung mit der nativen Entwicklung erziehlt werden kann, wodurch für jede zu unterstützende Plattform eine eigene Anwendung geschrieben werden muss. [NAD]
Durch die verschiedenen Betriebssysteme kommt das Problem, dass für jedes Betriebssystem bei einem Update alles geschrieben und getestet werden muss. Auch muss die Kompatibilität zu den einzelnen Betriebssystemversionen sichergestellt werden.[DANA]
Hierdurch entstehen hohe Entwicklungskosten und Wartungskosten.
Weiterhin muss die Anwendung auf jedem Computer installiert werden, was zu hohen Installationskosten führt.

2.1.2 Cross-Platform-Entwicklung

Was ist eine Cross-Platform-Entwicklung?
Die Idee hinter der Cross-Platform Entwicklung ist eine Anwendung für mehrere Betriebssysteme zur Verfügung zu stellen, ohne für jedes Betriebssystem eine eigene Anwendung zu programmieren.
Dafür wird die Anwendung einmal programmiert und für die verschiedenen Betriebssysteme kompiliert. Es lassen sich dennoch die API von den Betriebssystemen verwenden, um einen Teil der Anwendung speziell für dieses Betriebssystem anzupassen. [CPD] Eine API (Application Programming Interface) wird verwendet, um eine Kommunikation zwischen verschiednen Software-Anwendungen zu ermöglichen.

Vorteile einer Cross-Platform-Entwicklung
Dadurch, dass die Anwendung nur einmal programmiert werden muss und dann für die verschiedenen Betriebssysteme kompiliert wird, werden Entwicklungszeit und -kosten gespart.
Ebenso werden Zeit und Kosten bei Updates gesparrt, denn die Updates müssen nur noch einmal geschrieben werden und können direkt für die Betriebssysteme kompiliert werden. [CPDA]
Für die Entwicklung ist es nicht notwendig die Technologien der verschiedenen Betriebssysteme zu lernen. [CPDDA]

Nachteile einer Cross-Platform-Entwicklung
Der Vorteil, die Anwendung nur einmal schreiben zu müssen und für verschie-

dene Betriebssysteme zu kompilieren, bringt auch Nachteile mit sich.
Ein Nachteil ist, dass die Anwendung auf nicht optimal mit dem Betriebssystem kommunizieren kann, was zu einer langsameren Performanz als eine native Anwendung führen kann.
Auch ist es schwierig alle Komponenten eines Betriebssystems zu nutzen, denn die jeweiligen Betriebssysteme entwickeln sich weiter, jedoch dauert es oft, wenn Komponenten überhaupt intergriert werden, lange, bis diese in den Cross-Platform Technologien verfügbar sind.
Weiterhin gibt es einen Nachteil der UI, denn die Cross-Platform Anwendungen haben zwar auf den Betriebssystemen das gleiche Aussehen, jedoch ist es gegenüber einer nativen Anwendung nicht dem Aussehen des Betriebssystems angepasst. [CPDD]

2.1.3 Web-Entwicklung

Was ist eine Web-Entwicklung
Die Webentwicklung beschäftigt sich mit dem Entwickeln von Websites. Hierzu zählt das Entwickeln vom Design, der hinter der Website stehenden Logik und eventuell noch das Verbinden mit Datenbanken.
Zu dem Entwickeln des Designs(Front-End) wird in der Regel HTML, CSS und JavaScript eingesetzt, wobei HTML und CSS für das Aussehen zuständig sind und JavaScript für eine Interaktion mit dem Benutzer zuständig ist.
Die hinter der Website stehenden Logik (Back-End) kann mit sehr vielen Sprachen, wie Beispielsweise Python, Java und PHP, entwickelt werden. Hier wird eventuell auch eine Verbindung zu einer/mehreren Datenbank(en) hergestellt.
Websites werden über das Internet oder Intranet gehostet, damit andere Rechner auf diese Website zugreifen können. [WDD]

Vorteile einer Web-Entwicklung
Der große Vorteil von Web-Anwendungen ist, dass diese auf jedem Gerät laufen, welche einen Browser haben. Dies beinhaltet Linux, Windows, Mac, iOS und Android.
Ein weiterer Vorteil ist, dass nur ein Webserver vorhanden sein muss. Dies führt dazu, dass das Installieren oder Upgraden der Website nur einen einzigen Rechner betrifft.
Da der Server die Hauptarbeit leisten kann, brauchen Endnutzergeräte nicht eine starke Rechenleistung.[BWD] Durch die Browser, welche auf den Betriebssystemen sogar oft standardmäßig installiert sind, ist kein weiterer Download notwendig.
Web-Entwicklung zählt zu den günstigeren Entwicklungen, denn es lassen sich mehrere Links zwichen der Anwendung und der URL erstellen, welche einfach miteinander verbunden werden können.[ADWD]

Nachteile einer Web-Entwicklung
Es gibt sehr viele Browser, die die Standarts für Browser in der Regel

implementieren. Dennoch kann es vorkommen, dass eine Browserspezifische Funktion benutzt wird. Dadurch kann die Webseite in anderen Browsern anders funktionieren, als es gewünscht ist.[WDF]
Bei der Entwicklung muss darauf geachtet werden, dass die Website responsive, d.h. sich an den Bildschirmgrößen anpasst, entwickelt wird. Ansonsten kann es nicht benutzerfreundlich aussehen.
Wenn die Website einen Fehler enthält, kann dies dazu führen, dass die gesamte Website nicht funktioniert.[DWD]
Die Endgeräte müssen immer eine aktive Verbindung zum Server haben, ansonsten funktioniert die Web-Anwendung nicht mehr.
Außerdem ist der Zugriff auf bestimmte Funktionen der Hardware eingeschränkt oder komplett untersagt, was die Entwicklungsmöglichkeiten einschränkt. So lässt sich z.B. nicht direkt die Kamera eines Smartphones benutzen.[ADWD]

2.1.4 Hybride-Entwicklung

Was ist eine Hybride-Entwicklung
Hybride-Anwendungen sind eine Mischung aus nativen-Anwendungen und Web-Anwendungen. Bei dieser Technologie wird die Web-Anwendung in eine native-Anwendung eingebettet.
Durch diese Lösung kann per Web-Anwendung auch auf systemspezifische Funktionen zugegriffen werden.[HD]

Vorteile einer Hybriden-Entwicklung
Für eine Hybride Anwendung ist kein Web Browser, wie für eine Web-Anwendung, notwendig.
Dadurch, dass Hybride Anwendungen einen nativen Teil beinhalten, ist der Zugriff auf Betriebssystemspezifische APIs möglich.
Der Hauptteil der Anwendung besteht aus der Web-Anwendung. Diese kann für die verschiedenen Plattformen wiederverwendet werden.[HD]

Nachteile einer Hybriden-Entwicklung
Durch den Web-Anteil in den Hybriden Anwendungen ist diese langsamer als eine native Anwendung, da der Hauptteil nicht in einer vom Betriebssystem direkt unterstützten Sprache geschrieben wurde.
Es muss ein Wrapper für den Web-Anteil geschrieben werden, damit die Anwendung funktionieren kann.[HD]

2.2 Programmiersprachen

2.2.1 TIOBE Index

Der TIOBE Index ist ein Ranking von einigen Programmiersprachen. Das Ranking erfolgt nach der Anzahl der Ergebnisse für die Suche „<Programmiersprache> programming“ auf verschiedenen Suchmaschinen, die unterschiedlich gewichtet werden. Als Programmiersprache werden nur die Spra-

chen gewertet, die 1. einen Wikipedia Eintrag haben und von Wikipedia ausdrücklich als Programmiersprache betitelt werden, 2. muss die Programmiersprache Turing Complete sein und 3. muss die genannte Suchanfrage mindestens 5000 Ergebnisse auf Google liefern.
Mit dem TIOBE Index kann geschaut werden, wie aktuell die gelernten Programmiersprachen sind und für Unternehmen, welche Programmiersprachen gerade im Trend sind und daher empfohlen werden. [TI]

2.2.2 GitHut 2.0

Der GitHut 2.0 Index ist ein Ranking von Programmiersprachen, welches mithilfe von GitHub, dem größten Host für Programmierprojekte, Daten erstellt wird. Dazu wird die GitHub API verwendet, von welcher Daten für Pull Requests, Stars, Issues und Pushes genommen werden.
Über GitHut kann herausgefunden werden, wie viel eine Programmiersprache tatsächlich verwendet wird.[GH2.0]

2.2.3 Time Machine / Wayback Machine

Mithilfe der Wayback Machine kann man sehr viele Webseiten von einem beliebigen Zeitpunkt der Vergangenheit besuchen. Dadurch ist es möglich veränderte Artikel von einem bestimmten Zeitpunkt zu lesen, falls Teile überarbeitet wurden.

2.2.4 Top 20 Programmiersprachen aus dem TIOBE Index und GitHut 2.0 Index

Für das Heraussuchen der Programmiersprachen aus dem TIOBE Index wurde die Wayback Machine verwendet, um auf die vorherigen Monate zuzugreifen. Dies ist notwendig, da die Webseite jeden Monat aktualisiert wird.
In dem GitHut Index musste keine Wayback Machine verwendet werden, da die letzten Quartale gespeichert sind. **Programmiersprachen aus dem TIOBE Index**

Monat Jahr	**Programmiersprachen**	**Link**
Dezember 2019	Java, C, Python, C++, C♯, Visual Basic .NET, JavaScript, PHP, SQL, Swift, Ruby, Delphi/Object Pascal, Objective-C, Assembly Language, Go, R, MATLAB, D, Visual Basic, Perl	[TI1219]
Januar 2020	Java, C, Python, C++, C♯, Visual Basic .NET, JavaScript, PHP, Swift, SQL, Ruby, Delphi/Object Pascal, Objective-C, Go, Assembly Language, Visual Basic, D, R, Perl, MATLAB	[TI0120]

Februar 2020	Java, C, Python, C++, C♯, Visual Basic .NET, JavaScript, PHP, SQL, Swift, Go, Assembly language, R, D, Ruby, MATLAB, PL/SQL, Delphi/Object Pascal, Perl, Objective-C	[TI0220]
März 2020	Java, C, Python, C++, C♯, Visual Basic .NET, JavaScript, PHP, SQL, Go, R, Assembly language, Swift, Ruby, MATLAB, PL/SQL, Perl, Visual Basic, Objective-C, Delphi/Object Pascal	[TI0320]
April 2020	Java, C, Python, C++, C♯, Visual Basic, JavaScript, PHP, SQL, R, Swift, Go, Ruby, Assembly language, PL/SQL, Perl, Objective-C, MATLAB, Classic Visual Basic, Scratch	[TI0420]
Mai 2020	C, Java, Python, C++, C♯, Visual Basic, JavaScript, PHP, SQL, R, Swift, Go, MATLAB, Assembly language, Ruby, PL/SQL, Classic Visual Basic, Perl, Scratch, Objective-C	[TI0520]
Juni 2020	C, Java, Python, C++, C♯, Visual Basic, JavaScript, PHP, R, SQL, Swift, Go, Ruby, Assembly language, MATLAB, Perl, PL/SQL, Scratch, Classic Visual Basic, Rust	[TI0620]
Juli 2020	C, Java, Python, C++, C♯, Visual Basic, JavaScript, R, PHP, Swift, SQL, Go, Assembly language, Perl, MATLAB, Ruby, Scratch, Rust, PL/SQL, Classic Visual Basic	[TI0720]
August 2020	C, Java, Python, C++, C♯, Visual Basic, JavaScript, R, PHP, SQL, Go, Swift, Perl, Assembly language, Ruby, MATLAB, Classic Visual Basic, Groovy, Objective-C, Rust	[TI0820]
September 2020	C, Java, Python, C++, C♯, Visual Basic, JavaScript, PHP, R, SQL, Go, Swift, Perl, Assembly language, Ruby, MATLAB, Groovy, Rust, Objective-C, Dart	[TI0920]
Oktober 2020	C, Java, Python, C++, C♯, Visual Basic, JavaScript, PHP, R, SQL, Perl, Groovy, Ruby, Go, MATLAB, Swift, Assembly language, Objective-C, Classic Visual Basic, PL/SQL	[TI1020]

Im TIOBE Index waren folgende Programmiersprachen unter den top 20 in den Monaten von Dezember 2019 bis Oktober 2020:
Java, C, Python, C++, C♯, Visual Basic .NET, JavaScript, PHP, SQL, Swift, Ruby, Delphi/Object Pascal, Objective-C, Assembly Language, Go, R,

MATLAB, D, Visual Basic, Perl, Assembly language, PL/SQL, Classic Visual Basic, Scratch, Rust, Groovy und Dart. **Programmiersprachen aus dem GitHut Index**

Quartal	Programmiersprachen	Link
4/2019	JavaScript, Python, Java, C++, PHP, Go, Ruby, C♯, TypeScript, Shell, C, Scala, Rust, Swift, Kotlin, Perl, Objective-C, Groovy, R, Vim script	[GH419]
1/2020	JavaScript, Python, Java, C++, PHP, Go, C♯, TypeScript, Ruby, Shell, C, Scala, Rust, Swift, Kotlin, Perl, Objective-C, Groovy, Vim script, Lua	[GH120]
2/2020	JavaScript, Python, Java, C++, PHP, TypeScript, Go, Shell, Ruby, C♯, C, Scala, Rust, Swift, Kotlin, Perl, Objective-C, Groovy, Lua, Vim script	[GH220]
3/2020	JavaScript, Python, Java, C++, PHP, TypeScript, Go, Shell, Ruby, C, C♯, Scala, Rust, Swift, Kotlin, Perl, Groovy, Objective-C, Dart, Lua	[GH320]

Im GitHut Index waren folgende Programmiersprachen unter den top 20 in den Quartalen 04/2019 bis 03/2020:
JavaScript, Python, Java, C++, PHP, Go, Ruby, C♯, TypeScript, Shell, C, Scala, Rust, Swift, Kotlin, Perl, Objective-C, Groovy, R, Vim script, Lua und Dart.

Programmiersprachen der Indizes kombiniert
Java, C, Python, C++, C♯, Visual Basic .NET, JavaScript, PHP, SQL, Swift, Ruby, Delphi/Object Pascal, Objective-C, Assembly Language, Go, R, MATLAB, D, Visual Basic, Perl, Assembly language, PL/SQL, Classic Visual Basic, Scratch, Rust, Groovy, Dart, TypeScript, Shell, Scala, Kotlin, Vim script und Lua.

2.2.5 Programmiersprache ist eine Hochsprache

Das Level gibt an, wie hardwarenahe eine Programmiersprache ist. Dabei wird zwischen „Low Level“, was sehr hardwarenahe ist (z.B. Assembler), „Middle Level“, was schon abstrakter ist, wodurch oft weniger Code notwendig ist (z.B. C) und „High Level“, was eine noch höhere Abstraktion ist (z.B. Java) unterschieden.
Um das Level der Programmiersprachen herauszusuchen wird für jede Programmiersprache aus Top 20 Programmiersprachen aus dem TIOBE Index und GitHut 2.0 Index folgendes bei der Suchmaschine „Google“ gesucht:
„<Programmiersprache> programming language level“.

Nachfolgend wird für jede der Programmiersprachen angegeben, welches Level dieser zugeordnet wird:

Java ist laut Oracle [JL1] und einem Medium Artikel [JL2] eine Hochsprache.

C war laut Medium [CL2] eine Hochsprache, es wird aber nicht gesagt, wie die heute steht. Nach Course Report [CL1] zufolge war C eine Hochsprache, wird heutzutage aber als Tiefesprache angesehen.
C++ war laut Course Report [CL1], wie C, eine Hochsprache, wird aber als Tiefsprache heutzutage betrachtet. Laut GeeksForGeeks [C++L] ist C++ eine Mittelsprache.
Python ist nach der offiziellen Website [PL1] und hackr.io [PL2] eine Hochsprache.
C♯ wird sowohl auf Medium [C♯L1] als auch auf Career Karma [C♯L2] als Hochsprache betrachtet.
Visual Basic .NET wird auf Researchgate [VBNETL1] und Acseduonline [VBNETL2] als Hochsprache betrachtet.
JavaScript ist laut Pluralsight [JSL1] und Tutsplus [JSL2] eine Hochsprache.
PHP ist nach EDUCBA [PHPL] eine Hochsprache.
SQL wird von Oracle Patches [SQLL] als eine Hochsprache eingestuft.
Swift wird von bestprogramminglanguagefor.met [SWL1] und Infoq [SWL2] als Hochsprache eingestuft.
Ruby ist nach bestprogramminglanguagefor.me [RL1] und isleofruby [RL2] als Hochsprache betitelt.
Delphi/Object Pascal wurde von embarcadero [DL2] und kolmck [DL2] als Hochsprache eingestuft.
Objective-C ist nach gsdh [OCL] eine Hochsprache.
Assembly Language ist eine Niedrigsprache nach Tutorialspoint [ASL1] und educba [ASL2].
Go wird nach einem Medium Artikel [GL] als Niedrigsprache mit modernen Hochsprachnefeatures betrachtet.
R ist laut Codementor [RL] eine Niedrigsprache.
MATLAB ist zwar laut Springerprofessional [MLL] eine Hochsprache.
D wird von der offiziellen Webseite [MLL] und Computerhope [MLL] als Hochsprache angesehen.
Visual Basic hat die letzte Versione 6.0. Genaueres später in der Entscheidung (siehe 5.1.3)
Perl ist nach der offiziellen Dokumentation [VBL] und GeeksForGeeks [VBL] eine Hochsprache.
PL/SQL ist wie SQL, nur mit mehr Erweiterung, also eine Hochsprache.
Classic Visual Basic bezeichnet die Visual Basic Versionen von 6.0 und früher. Für genaueres siehe 5.1.3
Scratch ist eine visuelle Programmiersprache, die Entwickelt wurde, damit Kinder das Programmieren leichter lernen können.
Rust wird von dev.to [RL] als Niedrigsprache mit hochsprachigen Konzepten bezeichnet.
Groovy ist laut cs.com [GL] eine Hochsprache.
Dart ist nach codecarbon [DTL1] und Javatpoint [DTL2] eine Hochsprache.
TypeScript ist JavaScript mit Typen [TS].
Shell wird in der Kommandozeile verwendet und nicht für eine Anwendungsentwicklung.

Scala ist eine Hochsprache nach der offiziellen Webseite [SCL1] und auch nach Toptal [SCL2].
Kotlin läuft wie Java auf der JVM. Da Kotlin unter anderem von Java, Python und C♯ beeinflusst ist, zähle ich Kotlin zu den Hochsprachen. Auch dafür spricht die noch geringere LOC Anzahl, um z.B. eine Klasse mit Attributen zu erstellen, die eine getter und setter Methode haben [KL]. LOC bedeutet Lines of Code, womit die Codezeilen eines Programms gemeint sind, die keine Kommentare sind.
Vim script ist die Skriptsprache, die in den Texteditor Vim eingebaut wurde.
Lua ist nach einem Medium [LL] Artikel eine Hochsprache. Lua ist, wie im Artikel angegeben, nicht nur für Kindet geeignet. Lua wurde schon in vielen Spielen und elektronischen Geräten wegen seiner kleinen Größe verwendet.

2.2.6 Programmiersprachen mit Web Framework und GUI Designer

Das Herausfiltern von Programmiersprachen, die kein Web Framework bereitstellen oder eins bereitstellen, aber keinen UI Designer direkt oder über eine Erweiterung bereitstellen, erfolgt, indem für jede noch übrige Programmiersprache jeweils nach „<Programmiersprache> Web Framework visual Designer“, „<Programmiersprache> Web Framework wysiwyg“ und „<Programmiersprache> Web Framework Drag and Drop“ gesucht wird. Es werden nur Programmiersprachensuchen mit mindestens einem Web Framework, für welches es einen Designer gibt, übernommen. Falls in keinem der Suchergebnisse von Seite 1 bis 3 ein Web Framework mit einem Designer erwähnt wird, nehme ich an, dass es keines gibt.

Java: Für Java gibt es mindestens ein Framework mit einem Designer. Ein Framework ist Vaadin [Vaadin]. Dieses bietet in der Pro Versionen einen Designer, welcher den dazugehörigen HTML Code und die Java Klassen direkt generiert.
Python: Auch hier gibt es ein Framework mit einem Designer. Dieses ist [Anvil], bei welchem es einen Designer gibt. Für die Webseite soll keine weitere Sprache außer Python notwendig sein. Bei dem Framework werden aus dem Designer lauter Python-Dateien generiert.
C♯: In C♯ gibt das ein Framework namens Blazor. Dieses Framework selbst bietet keine Designer, jedoch gibt es eine Erweiterung namens Radzen [Radzen], welche einen Designer mitliefert. Auch hier wird der dazugehörige Code generiert, welcher aus .razor Dateien besteht.
Visual Basic .NET: Microsoft bietet hier das Framework WebForms [WF1]. In diesem Framework lässt sich mit Visual Basic .NET und C♯ entwickeln. Nach der offiziellen Webseite [WF2] hat WebForms einen Designer.
JavaScript: Hier gibt es ein Frontend Framework namens Webix [WX]. Dieses bietet einige Komponenten und einen Designer auf der Webseite. Jedoch ist Webix nur ein Frontend Framework und daher nicht für die Neuentwicklung von sehr großen Nutzen, da eine Datenbankverbindung nur von einem Server aus möglich ist.

PHP: Auch für PHP gibt es eine Framework, welches einen Designer über die Webseite bereitstellt. Der Name ist Impresspages [IP].
Swift: Für Swift gibt es zwar Web Frameworks, aber diese bieten keinen Designer an.
Ruby: Auch in Ruby gibt es kein Web Framework mit einem Designer. Ferner gibt es keine Extension für das beliebte Ruby on Rails Framework.
Delphi/Object Pascal: Für Delphi/Object Pascal gibt es ein Web Framework mit einem Designer namens Raudus [RS]. Dieses ist für Delphi und Lazarus geschrieben, der Designer mit Endresultat sieht jedoch nach älteren UIs aus.
D: Es gibt zwar sehr wenige Web Frameworks für D, jedoch ist keines dabei, welches einen Designer bereitstellt.
Perl: Ebenfalls gibt es Web Frameworks für Perl, jedoch ohne Designer.
Groovy: Vaadin unterstützt auch Groovy, da Groovy auf der JVM läuft. Dieser Artikel erklärt es zwar für Scala, jedoch steht dabei, dass es mit Groovy ebenfalls gilt: [SV]
Dart: Für Dart gibt es ein großes Framewor, Flutter, jedoch bietet dieses keinen Designer. Auch gibt es scheinbar keine anderen Frameworks, die Dart als Programmiersprache verwenden.
TypeScript: Bei den Ergebnissen gab es kein Web Framework mit einem Designer.
Scala: Mit Scala lässt sich für Vaadin entwickeln, da Scala wie Java auf der JVM läuft. Dieser Artikel erklärt, wie es geht: [SV]
Kotlin: Wie auch Groovy und Scala läuft Kotlin auf der JVM. Daher ist Vaadin auch hier ein passendes Web Framework: [KV]
Lua: Leider gibt es kein Web Framework für die Skriptsprache Lua.

2.2.7 Programmiersprachen Einleitung und letztes Update

Hier wird folgendes für jede Programmiersprache gesucht: „<Programmiersprache> history“ für Informationen über die Programmiersprache und „<Programmiersprache> latest release date“, um das Datum vom letzten Update herauszufinden.
Java wurde 1995 von dem „Green Team“, einem kleinen Team von Sun Ingenieuren, veröffentlicht. James Gosling gilt als „Vater“ von Java, zusammen mit Mike Sheridan und Patrick Naughton hat er 1991 mit der Entwicklung der Sprache angefangen.
Die Intention von Java war für die Entwicklung von interaktiven Fernsehgeräten. Jedoch war die Technologie der digitalen Kabelfernsehertechnologie noch nicht fortgeschritten genug. Allerdings war Java für die Programmierung im Internet passend. Zurzeit wird Java unter anderem für die Entwicklung im Internet, Mobilen Geräten und Spiele eingesetzt. [JH]
Das letzte Update, JDK 15, wurde am 15. September 2020 veröffentlicht. [JLR]

Python wurde von Guido Van Rossum, am „Centrum Wiskunde & In-

formatica“, im Dezember 1989 angefangen zu Entwickeln. Die erste, öffentliche Version, erschien im Jahr 1991. Python wurde hauptsächlich dafür Entwickelt, eine Programmiersprache zu haben, die Lesbar ist. Die Syntax von Python erlaubt den Entwicklern Konzepte in weniger Codezeilen auszudrücken als mit Java oder C++. [PH1]
Python ist eine general purpose Programmiersprache, denn Python wird für so ziemlich alles eingesetzt. Ein Paar verwendungen von Python sind für die Mobile Entwicklung, Spieleentwicklung, Webentwicklung, IoT und Datascience. [PH2] IoT bedeutet Internet of Things. Hierbeit kommunizieren Computer mit anderen Maschinen, wobei ein ständiger Datenaustausch stattfindet.
Die letzte Version ist Python 3.9, welche am 05. Oktober 2020 veröffentlicht wurde. [PLR]

C♯ wurde von Anders Hejlsberg bei Microsoft im Jahr 2000 entwickelt. Die Programmiersprache ist sehr ähnlich zu Java, was daran liegt, dass Microsoft Änderungen an Java vornehmen wollte, Sun es jedoch verboten hat. Das war die Geburtsstunde von C♯. Sie wurde für die Entwicklung im .NET Framework erstellt und braucht auch die .NET Umgebung um Lauffähig zu sein. Sie ist vor allem sehr gut für die Desktop Entwicklung unter Windows und der Spieleentwicklung geeignet. Jedoch lassen sich mit C♯ auch Web Anwendungen und Mobile Anwendungen erstellen. [CSH]
Die letzte stabile Version von C♯ ist die Version 8.0, zu welcher von Microsoft am 07. April 2020 ein Beitrag erstellt wurde, was in der Sprache neu ist. [CSLR1]
Dazu kommt eine Vorschau zu C♯ 9.0, welche am 20.Mai 2020 veröffentlicht wurde. [CSLR2]

Visual Basic .NET wurde, wie C♯, von Microsoft entwickelt und 2002 mit dem .NET Framework veröffentlicht. Visual Basic .NET beruht auf Visual Basic, diese können aber nicht miteinander interagieren. Auch mit Visual Basic .NET kann für Windows Dektop, für das Web, Mobile Anwendungen und Office entwickeln. [VBNETH]
Visual Basic .NET scheint zuletzt am 24. Oktober 2018 ein Update erhalten zu haben. [VBNETLR1]
Geplant ist die Unterstützung von weiteren Tools in .NET 5, jedoch keine Weiterentwicklung der Programmiersprache selbst. [VBNETLR2]

PHP wurde 1994 von Rasmus Lerdorf entwickelt. Anfangs war PHP ein Tool, um die Besucherzahlen von seinem online Lebenslauf zu zählen. In Laufe der Zeit hat Rasmus mehr Funktionalitäten hinzugefügt, bis er PHP schließlich neu schrieb. Im Laufe der Zeit erhielt PHP unter anderem eine eingebaute Unterstützung für einige Datenbanksysteme, eigene Funktionen und Cookies. PHP wird Hauptsächlich für die Webentwicklung eingesetzt. [PHPH]
Die letzten Veröffentlichungen von PHP waren am 01. Oktober 2020. [PHPLR]

Delhpi/Object Pascal: Delphi basiert auf Object Pascal. Pascal wurde

1971 von Niklaus Wirth und 1973 entwickelt. Mit Pascal war es möglich, neue Datenstrukturen aus bestehenden zu bilden. Auch wurde dynamische Datenstrukturen unterstützt, so war es möglich Objekte im Laufe des Programms vergrößern oder verkleinern zu können.
Pascal wurde dafür entwickelt, eine Lehrsprache für Schüler in Programmierklassen zu sein.
Delphi ist eine Umwandlung von Pascal in eine Objektorientierte Programmiersprache, die auch Datenbankzugriffe erlaubt. [DH]
Die letzte Version von Delphi wurde, laut dem Freepascal Wikipedia, im November 2018 veröffentlich. [DLR]

Groovy wurde das erste Mal im Jahr 2003 von James Strachan in seinem Blog erwähnt. Die ersten Veröffentlichungen kamen in den Jahren 2004-2006. Nach einer Versionsumbenennung erschien Groovy 1.0 im Jahr 2007. Leider ist nicht mehr zu der Intention hinter Groovy bekannt, auch in anderen Artikeln. Groovy läuft auf der JVM, wodurch alle Bibliotheken, die unter anderem in Java geschrieben wurden, auch on Groovy verwendet werden können. [GH]
Die letzte Version von Groovy erschien am 26. September 2020 auf dem GitHub Repository. [GLR]

Scala wurde 2001 von Martin Odersky angefangen zu Entwickeln und 2004 veröffentlicht. Da Odersky bei der Entwicklung von Javac (primärer Java Compiler) und Generic Java dabei war, ist Scala ähnlich zu Java. Scala läuft, wie Java und Groovy, ebenfalls auf der JVM. Sie wurde entwickelt, um eine bessere Sprache zu sein. [SH]
Die neueste Version von Scala, 2.12.12, wurde am 13. Juli 2020 veröffentlich. [SLR]

Kotlin wurde von Jetbrains 2010 angefangen zu entwickeln. Mit Kotlin kann für Androit, JavaScript, Native und JVM entwickelt werden. Die Programmiersprache ist nicht nur Objektorientiert, sondern auch Funktional. Es werden auch Mischungen erlaubt. Sie wurde entwickelt, um kompakter Programmieren zu können. So lassen sich Programme mit bis zu 40% weniger Code schreiben. [SH]
Die Letzte Version von Kotlin wurde am 07. September 2020 veröffentlicht. [GLR]

2.2.8 Ausführungsgeschwindigkeit von einigen Programmiersprachen

Die Suche nach „Programming language execution performance“ wurde für einen groben Performanzvergleich verwendet. Folgende Resultate wurden in den Ergebnissen angezeigt:

Ein Artikel auf Newstack.io beschäftigt sich mit dem Stromverbrauch von 27 verschiedenen Programmiersprachen. Es ist nicht nur der Stromverbrauch, sondern auch der Speicherverbrauch und vor allem die Ausführungszeit von den Programmiersprachen dargestellt. Unter den 27 Programmiersprachen ist Java, C♯ und Python. Groovy, Scala und Kotlin sind leider nicht mit betrachtet.
Für den Vergleich in dem Artikel wurden, von 6 Portugisichen Forschern, 10 Probleme von den „Computer Language Benchmarks Game“ [CLBG] genommen.
Nach den Tests zufolge ist, von den übrigen Programmiersprachen aus dem vorherigen Schritt, folgende Reihenfolge der Performanz (Erst genannte Programmiersprachen sind schneller als letzter genannte):

1. Java
2. C♯
3. Python

[PU] Dieses Ergebnis ist nicht wirklich verwunderlich, denn Java und C♯ werden beide in ein Zwischenformat kompiliert und dann ausgeführt, während Python interpretiert wird. Interpretierte Programmiersprachen sind meistens langsamer als kompilierte Programmiersprachen und Programmiersprachen mit einer Zwischenschicht.
In dem GitHub Repository von attractivechaos wurden verschiedene Algorithmen mit unterschiedlichen Programmiersprachen und Compilern entwickelt. Die Resultate wurden in einer Tabelle dargstellt. [EP]
Auf dem ersten Blick ist zu erkennen, dass verschiedene Compiler verschiedene Ausführungsgeschwindigkeiten bei den Programmiersprachen haben.

Überblick über die verschiedenen Compiler:
C♯:
Mono-2.10.1 ist eine Open Source Variante des .NET Frameworks, welches auf Unix, Windows, macOS und anderen Betriebssystemen lauffähig ist. [MCS]
Java:
`JRE-1.6.0_25` ist das Java Runtime Environment von Oracle, welche Java offiziell verwalten. [JREJ]
Python:
PyPy-1.4.1 ist eine JIT Variante von Python, welche meistens schneller läuft, als die standart Python Variante. [PPP]
CPython-3.2 und CPython-2.7.1 sind die Interpreter von der offiziellen Python Software Foundation. Die Nummern geben die Version an. [CP]
Mono-2.10.1 bietet auch einen Python Interpreter. Durch IronPython ist es möglich Python mit .NET zu verwenden. [IP]
`JRE-1.6.0_25` erlaubt die Interaktion von Python und Java. Python kann somit von Java ausgeführt und Java von Python ausgeführt werden. [J]
GCC-4.3.2 mit ShedKin-09 ist ein Experiment um Python in C++ umzuwandeln. Es werden allerdings nicht so viele Bibliotheken unterstützt. [SS]

Für Python wird der CPython Interpreter mit der Version 3.2 betrachtet, da dieser von der Offiziellen Webseite ist und im Vergleich die neuere Version ist.

Nachfolgend werden die 4 auf der GitHub Weiseite betrachteten Programmiersprachen nach der Ausführungszeit und Algorithmus geordnet. A > B bedeutet, dass Programmiersprache A schneller als Programmiersprache B abgeschnitten hat.
sudoku:t: Java > C♯ > Python
matmul:t: Java > C♯ > Python
matmul:m: C♯ > Java > Python
patmch:1t: Python > Java > C♯
patmch:2t: Java > Python > C♯
dict:t: Python > C♯ > Java
dict:m: C♯ > Python > Java

2.2.9 Entwicklungsgeschwindigkeit

Als Suchkriterien für die Datenbank und INI Bibliotheken dienen die Suchen „<Programmiersprache> database connection low level“ und „<Programmiersprache> ini file library“.
Folgende Bibliotheken wurden für die Programmiersprachen gefunden und für den Vergleich genommen:

Programmiersprache	**Datenbank**	**INI Datei**
Java	JDBC	ini4j
Python	SQLAlchemy	configparser
C♯	SqlConnection(in System.Data)	ini-parser
Groovy	groovy-sql	ini4j
Scala	JDBC	ini4j
Kotlin	JDBC	ini4j

2.3 Web Frameworks

2.3.1 Frameworks mit einem GUI Designer

Für die Suche nach Web Frameworks mit einem GUI Designer wird auf der Suchmaschine Google nach folgenden Suchbegriffen gesucht:
„<Programmiersprache> Web Framework“, „<Programmiersprache> Web Framework GUI Designer“ und „<Programmiersprache> Web Framework Drag and Drop“. Es werden nur die Ergebnisse aus den ersten 3 Seiten betrachtet.
Als Web Framework mit GUI Designer zählen auch die Web Frameworks, deren Erweiterung einen GUI Designer bereitstellen.

Folgende Ergebnisse wurden für die Programmiersprachen gefunden:
Java:
1. JSF (Java Server Faces): Hier können Komponenten per drag and drop

verwendet werden. Es ist kein bis kaum HTML, CSS und JavaScript notwendig. [JWF1]
2. Vaadin: Mit dem Vaadin Designer (kostenplfichtig) lassen sich Webseiten per drag and drop designen. [JVD]
Scala:
1. TODO

3 Vorstellung von Opcenter Execution Pharma

Das System „Opcenter Execution Pharma“ (OEP) ist ein MES-Lösung für die pharmazeutische Industrie, welche in VB6 geschrieben wurde und als Desktop Anwendung bereitgestellt wird. MES bedeutet „Manufacturing Execution System“, oft als Produktionsleitsystem bezeichnet, und ermöglicht die Führung, Lenkung, Steuerung oder Kontrolle von Produktionen in Echtzeit.
Mit OEP wird eine papierlose Fertigung und vollelektronische Chargenerfassung ermöglicht. Es werden außerdem erweiterte Funktionen für die manuelle oder automatisierte Entwicklung, Optimierung und Verwaltung von Produktionsvorgängen und -prozessen.
OEP gibt durch verschiedene Fertigungsressourcen, inklusive Personen, Anlagen, Produkte und Prozesse, eine effiziente Produktfertigung.
Genauere Informationen zu verschiedenen Teilen können hier gefunden werden: [OEP].

Für die Neuentwicklung bestehen folgende Anforderungen:

1. Plattformunabhängig

   Einleitung: Ein Programm kann sehr viel verwendet werden, wenn es auf möglichst vielen Plattformen lauffähig ist. Auch in der pharmazeutische Industrie ist es wichtig, dass ein System auf möglichst vielen Plattformen genutzt werden kann, sodass jederzeit ein Zugriff auf das System möglich ist.
   Aktuelles System: Bisher kann Opcenter Execution Pharma nur auf Windows Systemen ausgeführt werden, da die Anwendung in VB6 geschrieben wurde.
   Anforderung an die neue Plattform: Das neuentwickelte System soll auf allen gängigen Betriebssystemen verwendet werden können. Dazu zählen vor allem Windows, macOS, iOS und Android.

2. Keine App-Store Abhängigkeiten

   Einleitung: Für den privaten Gebrauch werden sehr viele Anwendungen über App-Stores installiert, da es viel bequemer ist, als sich eine Kopie für den eigenen Computer holen zu müssen. Allerdings sind die Computer von den Kunden meist nicht mit dem Internet verbunden und haben somit keinen Zugriff auf App-Stores. Deshalb ist es wichtig, dass das neue System komplett offline installiert werden kann und keine Abhängigkeiten zu beispielsweise Google Play notwendig sind.
   Aktuelles System: Bei dem aktuellen System gibt es keine App-Store Abhängigkeiten, denn bei der Installation und bei Updates wird das Programm auf jeden einzelnen Computer einzeln installiert.

Anforderung an die neue Plattform: Auch das neu entwickelte System soll keine Abhängigkeiten zum App-Store haben, da die Computer nicht mit dem Internet verbunden sind. Allerdings sind alle Computer in einem Lokalen Netzwerk (Intranet) miteinander verbunden, somit ist eine Web Entwicklung möglich.

3. Effiziente Wartung und Pflege

   Einleitung: Bei ersten Installation und bei der Wartung und Pflege beim Kunden vor Ort geht viel Zeit verloren, wenn jeder Computer das Update einzeln erhalten muss.
   Aktuelles System: Bei dem aktuellen System muss mit jedem Update jeder einzelne Computer das Update manuell erhalten. Dies kostet sehr viel Zeit und damit Geld.
   Anforderung an die neue Plattform: Die Erstinstallation und jedes Update soll in möglichst kurzer Zeit beim Kunden installiert werden können. Somit können Kosten bei der Wartung und Pflege gespart werden.

4. Zukunftssicherheit

   Einleitung: Es gibt hunderte von Programmiersprachen und tausende von Frameworks, die für eine Neuentwicklung verwendet werden können. Da Programmiersprachen und Frameworks jedoch altern und der Support irgendwann eingestellt wird, ist es wichtig eine Programmiersprache und ein Framework zu wählen, welches eine Zukunftssicherheit hat.
   Aktuelles System: Das aktuelle System ist in VB6 geschrieben. Leider wurde der Support für VB6 2005 eingestellt, wodurch die Programmiersprache keine Updates mehr erhält.
   Anforderung an die neue Plattform: Es soll Programmiersprache und ein Framework gefunden werden, die mindestens die nächsten 15 Jahre Updates erhalten, denn es bestehen sehr strenge Qualitätsanforderungen bei der Systementwicklung, wodurch eine Neuentwicklung sehr lange dauert.

5. Schnelle Ausführungsgeschwindigkeit

   Einleitung: Bei einem Programm spielt die Ausführungsgeschwindigkeit eine sehr wichtige Rolle, denn kein Nutzer von einem Programm möchte lange auf eine Antwort warten.
   Aktuelles System: Das aktuelle System hat eine ziemlich geringe Ausführungseschwindigkeit. So werden Daten innerhalb von einer Sekunde aus der Datenbank gelesen und dem Nutzer angezeigt. Auch das wechseln von einem Modul in ein anderes ist in weniger als einer halben Skunde erreicht.
   Anforderung an die neue Plattform: Auch in der neuen Plattform soll die Ausführungseschwindigkeit nicht zu hoch sein. Das Auslesen und Anzeigen von Daten soll keine Sekunde dauern. Ebenfalls soll das wechseln zwischen Modulen sehr schnell sein.

6. Geringe Entwicklungszeit

   Einleitung: Ein sehr wichtiger Punkt bei der Entwicklung ist, dass die Entwicklungszeit so gering wie möglich ist, sodass kosten gespart werden können.
   Aktuelles System: Aktuell wird das System in VB6 entwickelt und es wird nur ein Programm erstellt, da nur Windows als Betriebssystem verwendet wird.
   Anforderung an die neue Plattform: Bei der neuen Plattform sollen die Entwicklungkosten so gering wie möglich gehalten werden bei gleichzeitiger Plattformunabhängig. Von der Entwicklung ausgeschlossen sind Low Code Plattformen, denn die Projekte auf der Prozessebene sind viel zu komplex. Mit Low Code Plattformen würde eine weitere Komplexität hinzukommen, vor allem bei der Dokumentation.

7. Anforderungen an die Programmiersprache

   Einleitung: Die Programmiersprache sollte eine bestimmte beliebtheit haben, sodass sichergestellt werden kann, dass es eine Dokumentation zu dieser gibt. Weiterhin sollte die Programmiersprache grundsätzlich nicht zu lange zum Entwickeln brauchen, bestmöglich eine Hochsprache. Denn mit Hochsprachen muss nicht alles betrachtet werden, wie Speichermanagement. Außerdem soll die Programmiersprache für die Web Entwicklung geeignet sein und das Framework selbst oder eine Erweiterung soll einen Designer zum Gestalten von Webseiten bereitstellen.
   Aktuelle Programmiersprache: VB6 war zu der Zeit eine sehr beliebte Programmiersprache. Heutzutage wird Visual Basic zwar noch verwendet, jedoch ist die Beliebtheit nicht mehr als zu hoch.
   Anforderung an die neue Programmiersprache: Die Programmiersprache soll unter den Top 20 Programmiersprachen aus dem TIOBE Index von Dezember 2019 bis Oktober 2020 und dem GitHut Index vom Quartal 4/2019 bis 03/2020 sein. Weiterhin soll die Programmiersprache eine Hochsprache sein und anhand des Ergebnisses von 4.3 ein Web Framework bereitstellen.

4 Auswahl einer geeigneten Technologieplattform

In diesem Kapitel wird untersucht, welche Technologieplattform für die Neuentwicklung am besten geeignet ist.

4.1 Auswahlkriterien

Für die Auswahl der Technologie werden die Kriterien 1 (Plattformunabhängig), 2 (Keine App-Store Abhängigkeiten), 3 (Effiziente Wartung und Pflege) und 6 (Geringe Entwicklungszeit) von Kapitel 3 verwendet.

4.2 Vorgehen bei der Auswahl

Für jedes Kriterium wird untersucht, welche der in den Grundlagen vorgestellten Technologien das jeweilige Kriterium erfüllt. Am Ende jedes Kriteriums wird untersucht, welche Technologieplattform(en) dieses Kriterium am besten erfüllt. Nach der Untersuchung der Kriterienerfüllung wird eine begründete Entscheidung für eine Technologieplattform getroffen.

4.3 Entscheidung

Vorab:
Im Grunde reicht eine blose Entscheidung mithilfe der Grundlagen aus 2.1 nicht aus, sondern es müsste in jeder Technologieplattform mindestens ein Projekt geschrieben werden, um einen Vergleich zu haben. Jedoch gibt es innerhalb einer Technologieplattform auch unterschiedliche Frameworks, die unterschiedliche Entwicklungszeiten benötigen. Deshalb wird bei der Entscheidung davon ausgegangen, dass die Frameworks der Technologieplattformen die gleiche Entwicklungszeit für ein Projekt benötigen, was in der Praxis natürlich nicht zutrifft.

Plattformunabhängigkeit:
Im Grunde decken alle 4 Technologieplattformen alle Betriebssysteme ab. Jedoch gibt es einen Unterschied vom Aufwand her:

Bei der nativen Entwicklung muss für jedes Betriebssystem jeweils eine Anwendung geschrieben werden, was zu erhöhten Entwicklungskosten führt.

Bei der Cross-Platform Entwicklung werden mit einem einzigen Projekt mehrere Betriebssysteme abgedeckt. Hier ist das schreiben von einer bis sehr wenigen Anwendungen notwendig.

In der Web Entwicklung muss das Projekt nur einmal geschrieben werden und ist dann auf allen Betriebssystemen über einen Browser verfügbar.

Auch bei der Hybriden Entwicklung werden alle Betriebssysteme abgedeckt.

Es muss für jedes einzelne Betriebssystem eine native Anwendung erstellt werden, in welche eine Website eingebunden wird.

⇒ Hier haben die Cross-Platform und Web Entwicklung einen deutlichen Vorteil gegenüber der hybriden und nativen Entwicklung.

Keine App-Store Abhängigkeiten:
Hier gibt es einen kleinen Unterschied:

Auf Desktop Geräten (PC, Laptop) können Programme z.B. einem USB-Stick installiert werden. Somit ist eine Offline Installation für alle 4 Plattformtechnologien auf einem Desktop Geräten ohne Probleme möglich.

Jedoch ist dies bei mobilen Geräten (Smartphone, Tablet) nicht so einfach möglich. Denn dort werden Apps in der Regel über den jeweiligen Store installiert. Es gibt aber Möglichkeiten Apps ohne den Store zu installieren. Diese sind jedoch nicht so trivial wie für Desktop Geräten.

⇒ Hier liegt der Vorteil bei den Desktop-Geräten, da hier problemlos die Anwendung installiert werden kann. Bei mobilen Geräten ist dies unter umständen Möglich, jedoch nicht elegant.

Effiziente Wartung und Pflege
Hier gibt es einen klaren Sieger:

Da bei der nativen, hybriden und cross Plattform Entwicklung jeweils eine Desktop-Anwendung entstehen, muss diese auf jedem Client Computer installiert werden und bei jedem Update muss das Update bei allen Computern installiert werden. Damit entsteht ein hoher Zeitaufwand und damit auch hohe Kosten.

Ganz anders sieht es bei der Web Entwicklung aus. Hier entsteht zwar auch ein Programm, jedoch muss dieses nur auf einem Rechner, dem Server, installiert werden. Auch bei Updates muss nur noch der Server aktualisiert werden. Dadurch werden einen Menge Installations- und Wartungskosten gespart, da viel weniger Zeit für die Installtion und Updates verwendet werden muss.

⇒ Bei der Web Entwicklung wird am wenigsten Zeit für die Erstinstallation und Wartung von Updates benötigt.

Geringe Entwicklungszeit
Die Entwicklungszeiten sind ziemlich unterschiedlich:

Mit der nativen Entwicklung muss für jedes zu unterstützende Betriebssystem eine eigene Anwendung geschrieben werden. Bei mehreren unterstützten Betriebssystemen muss das System mehrfach entwickelt werden, was zu langen und hohen Entwicklungskosten führt.

Mit der Cross-Platform Entwicklung werden mit einem Projekt mehrere Betriebssysteme erreicht. Dadurch müssen nicht viele verschiedene Projekte gestartet werden, sondern im besten Fall 1. Dadurch sind die Entwicklungskosten wesentlich geringer.

Die Web Entwicklung braucht nur ein Projekt, da die gängigen Betriebssysteme einen Browser zur Verfügung stellen und die Webseite im Intranet

erreicht wird.

Für die Hybride Entwicklung ist erstmal für jedes zu unterstützende Betriebssystem eine Anwendung notwendig, jedoch wird der Hauptteil nur einmal geschrieben und wird dann in den Container integriert. Dadurch sind die Entwicklungskosten sehr niedrig.

⇒ Die kürzeste Entwicklungszeit haben die Web und Cross-Platform Entwicklung, etwas länger und kostenspieliger ist die Hybride Entwicklung, jedoch nur bis zu einem gewissen Teil.

⇒ Zusammengefasst ist die Web-Entwicklung die beste Option für eine Neuentwicklung. Sie ist zwar oft etwas etwas langsamer als eine native Anwendung, jedoch liegt dies unter anderem an der Kommunikation zwischen dem Server und dem Client. Die Verbindung zwischen dem Client und dem Server wird im Intranet aufgebaut, wodurch keine Informationen ins Internet gelangen. Allerdings sind nicht starke Rechner für die Clients notwendig, sondern „nur“ ein starker Server. Dieser kann die Website schneller erstellen und dadurch etwas Geschwindigkeit aufholen. Vor allem wird durch die Web-Entwicklung eine Plattformunabhängigkeit geschaffen, die alle möglichen Betriebssysteme mit einem Browser direkt unterstützt. Mit der Plattformunabhängigkeit kommen geringere Entwicklungskosten als bei nativen und hybriden Entwicklungen. Auch sind die Kosten für die Installation und Wartung beim Kunden viel geringer, da nur 1 Rechner die neuere Version braucht. Ein Offline-Installtion ist auch möglich, da der Server in der Regel ein Desktop Computer ist.

5 Auswahl geeigneter Programmiersprachen

Im letzten Teil habe ich mich für die Web Entwicklung als Technologieplattform entschieden (siehe 4.3).
Dieser Teil der Bachelorarbeit befasst sich mit der Auswahl von geeigneten Programmiersprachen für die Web Entwicklung. Hierfür wird dieser Teil geteilt. Im ersten Abschnitt werden Programmiersprachen mithilfe von Binären Kriterien herausgesucht und gefiltert, im zweiten Abschnitt werden die resultierenden Programmiersprachen durch nicht binäre Kriterien weitergefiltert.

5.1 Binäre Filterung

5.1.1 Auswahlkriterien

Für die binäre Filterung wird das Kriterium 7 (Anforderungen an die Programmiersprache) verwendet. Dieses beinhaltet die Punkte „Programmiersprache ist unter den Top 20 vom TIOBE und GitHut Index“, „Programmiersprache ist eine Hochsprache“ und „Programmiersprache stellt eine Web Framework mit einem Designer bereit“.

5.1.2 Vorgehen bei der Auswahl

Die Kriterien werden nacheinander behandelt. Folgendes Vorgehen gilt für die Auswahl der einzelnen Kriterien:
Programmiersprache istunter den Top 20 vom TIOBE und GitHut Index
Um auf die vergangenen TIOBE Indizes (siehe 2.2.1) zugreifen zu können verwende ich die Wayback Machine (siehe 2.2.3).
Für die Auswahl der Programmiersprachen werden der TIOBE Index und GitHut Index verwendet, da einer alleine nicht aussagekräftig genug ist. Denn nur die Anzahl der Suchanfragen zu berücksichtigen zeigt mir nicht, wie viel die Programmiersprache im Verhältnis zu anderen verwendet wird. So kann sehr viel nach einer Programmiersprache gesucht werden, weil diese beispielsweise neu ist oder sehr viel nachgeschlagen wird.
Den GitHut Index alleine zu verwenden ist auch nicht aussagekräftig, denn ältere Programmiersprachen sind meistens in mehr Projekten vertreten als neuen Programmiersprachen.
Die Mischung aus beiden Indizes berücksichtigt die Programmiersprachen, die mehr gesucht werden und die Programmiersprachen, die viel verwendet werden.

Programmiersprache ist eine Hochsprache
Es werden nur eindeutige „Hochsprachen“, das heißt die Quellen geben das selbe Level an, für die Framework-Suche betrachtet, da diese eine höhere Abstraktion haben und dadurch einfacher zu lernen sind und es einfacher

ist, in diesen etwas zu Entwickeln. „Hochsprachen“ werden mehr für die Web Entwicklung eingesetzt als „niedrigere“ Programmiersprachen, da es meistens nicht notwendig ist die letzte kleine Performanz rauszuholen. [HLLL]
Programmiersprache stellt eine Web Framework mit einem Designer bereit Betrachtet werden nur Programmiersprachen mit mindestens einem Web Framework, welche einen GUI Designer haben. Der Grund dafür ist, dass ein Designer die Entwicklung sehr beschleunigt, da der Code nicht selbst geschrieben werden muss für die Oberfläche und das Endresultat direkt sichtbar ist.

5.1.3 Entscheidung

Für die einzelnen Kriterien wurden folgende Entscheidungen getroffen, welche Programmiersprachen weiter betrachtet werden:
Programmiersprache unter Top 20 im TIOBE Index und unter Top 20 im GitHut Index
Da es hier keine Interpretation gibt, werden folglich alle Programmiersprachen aus den Indizes übernommen (siehe 2.2.4)
Programmiersprache ist eine Hochsprache
Einordnung des Levels steht in den Grundlagen (siehe 2.2.5). Nicht überall ist das Level eindeutig, deshalb wird die Entscheidung hier aufgegriffen:
C: Da C als Hochsprache angesehen wurde, jetzt jedoch als Tiefsprache, würde ich C als Mittelsprache einordnen und daher nicht weiter betrachten.
SQL: SQL wird zwar als Hochsprache eingeordnet, jedoch ist SQL keine Programmiersprache im eigentlichen Sinne. Daher wird SQL nicht weiter betrachtet.
Objective-C: Zwar wurde Objective-C als Hochsprache eingeordnet, jedoch ist sie eine Mischung aus C und Smalltalk. Da C eine Mittelsprache ist wird Objective-C nicht für die weiteren Schritte betrachtet.
MATLAB: Wird zwar als Hochsprache angesehen, jedoch wird sie für die Datenanalyse verwendet. Daher wird MATLAB nicht weiter betrachtet.
Visual Basic: Da Visual Basic nicht mehr unterstützt wird, wird sie nicht weiter betrachtet.
PL/SQL: Ist eine Erweiterung von SQL und wird daher nicht weiter betrachtet.
Classic Visual Basic: Beinhaltet die Verionen von 6.0 und früher. Da schon VB 6.0 nicht unterstützt wird, wird Classic Visual Basic nicht weiter betrachtet.
Scratch: Ist eine Visuelle Programmiersprache, die für Kinder zum Programmieren lernen entwickelt wurde. Daher wird Scratch nicht weiter betrachtet.
Shell: Wird in der Kommandozeile verwendet und daher nicht weiter betrachtet.
Vim Script: Wurde für die Entwicklung für den Texteditor Vim entwickelt. Daher wird Vim Script nicht weiter betrachtet.

Folgende Programmiersprachen wurden eindeutig als Hochsprache eingeordnet und werden daher weiterhin betrachtet:

Java, Python, C♯, Visual Basic .NET, JavaScript, PHP, Swift, Ruby, Delphi/Object Pascal, D, Perl, Groovy, Dart, TypeScript, Scala, Kotlin und Lua.

Programmiersprache hat ein Web Framework mit einem GUI Designer
Da es auch hier keine Interpretation gibt, ob eine Programmiersprache ein Web Framework hat und wenn ja mit einem GUI Designer, sind folgende Sprachen für die weitere Betrachtung übriggeblieben (siehe 2.2.6):
Java, Python, C♯, Visual Basic .NET, PHP, Delphi/Object Pascal, Groovy, Scala und Kotlin

5.2 Nicht-Binäre Filterung

5.2.1 Auswahlkriterien

Für die Nicht-Binäre Filterung werden die Kriterien 4 (Zukunftssicherheit), 5 (Schnelle Ausführungsgeschwindigkeit) und 6 (Geringe Entwicklungszeit) verwendet.

5.2.2 Vorgehen bei der Auswahl

Auch hier werden Kriterien nacheinander abgearbeitet. Dabei wird folgendes Vorgehen für die einzelnen Schritte betrachtet:
Zukunftssicherheit der Programmiersprachen
Für jede Programmiersprache wird geguckt, wann sie das letzte Mal ein Update erhalten hat. Wenn das letzte Update länger als 18 Monate zurückliegt, wird die Programmiersprache als nicht schnell wachsend angesehen und nicht weiter betrachtet.
Für die Programmiersprachen, die die letzten 18 Monate ein Update erhalten haben wird der Verlauf der letzten 5 Jahre im TIOBE Index und im GitHut Index angesehen. Wenn die Beliebtheit gleich geblieben ist oder steigt, wird die Programmiersprache als Zukunftssicher angesehen.
Für die Visualierung der Beliebtheit wird der Graph mithilfe des Python Bibliothek „Matplotlib“ gezeichnet. Für den TIOBE Index habe ich einen Script geschrieben, der die Zahlen automatisch zieht. (siehe TIOBEBeliebtheit.py)
Ausführungsgeschwindigkeit
Es werden allgemeine Performanzvergleiche betrachtet. Dort sollten mindestens 2 übrige Programmiersprachen aus dem ersten Kriterium drin sein.
Falls kein Eindeutiges Ergebnis vorliegt, welche Programmiersprache schneller als eine andere ist, werden eigene Performanztests geschrieben. Ansonsten werden für alle übrigen Programmiersprache Ausführungszeiten herausgesucht und versucht diese miteinander zu vergleichen. Dies würde mit der Entwicklungsgeschwindigkeit testen zusammen passieren.
Entwicklungsgeschwindigkeit
Im Falle von eines eindeutigen Geschwindigkeitsvergleichs werden auch für die Entwicklungsgeschwindigkeit Quellen zum Vergleich herausgezogen.

Andernfalls werden die Ausführungsgeschwindigkeitstests mit den Entwicklungsgeschwindigkeitstest verbunden.
Da Bibliotheken die Entwicklung enorm beschleunigen, gelten folgende Kriterien für die Entwicklungsgeschwindigkeit:

1. Viele Bibliotheken, insbesondere eine für Datenbankverbindungen bereitgestellt werden. Der Grund ist, dass möglichst viele Funktionalitäten bereits implementiert wurden und nicht noch selbst entwickelt werden müssen.
2. Lesen und Schreiben von Dateien, denn oft werden Konfigurationen z.B. in INI-Dateien gespeichert. Diese müssen eingelsen werden. Bestmöglich gibt es eine Bibliothek oder es lassen sich Plattformspezifische Funktionen dafür aufrufen.

Anschließend werden die Performanzdaten ausgewertet und ein Ranking der Programmiersprachen erstellt. Die besten 5 Programmiersprachen werden übernommen.

Eigene Tests für die Ausführungsgeschwindigkeits- und Entwicklungsgeschwindigkeitstests von Programmiersprachen

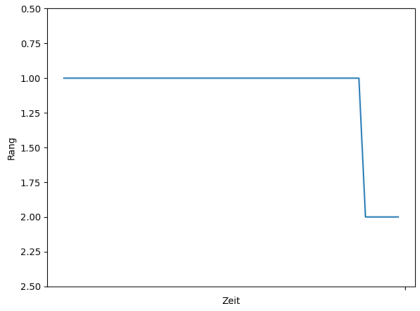
Welche Forschnungsfrage soll beantwortet werden?
Mithilfe dieser Tests soll untersucht werden, in welchen Programmiersprachen etwas potenziell schnell entwickelt werden kann und die schnell die Ausführungsgeschwindigkeiten der jeweiligen Programmiersprachen sind.
Was ist das Set Up?
Das Set Up der Tests besteht aus der Kommandozeile, einem Text Editor (Visual Studio Code) und den Kompilern für die jeweiligen Programmiersprachen.
Wie wird die Evaluation durchgeführt?
Es werden pro Programmiersprache jeweils 3 Dateien erstellt. Die eine Datei enthält die main Methode, die für die Geschwindigkeit messen und Ergebnisse ausgeben zuständig ist. Diese verwendet die anderen beiden Dateien. Die zweite Datei beinhaltet eine Klasse, die für eine Datenbankabfrage verwendet wird. Dafür wird eine Testdatenbank in SQLite bereitgestellt. Diese beinhaltet eine Tabelle mit 4 Spalten und 10.000 Einträgen. Im Programm wird die ganze Tabelle per Datenbankabfrage gelesen und die Daten in einer eigenen, inneren Klasse gespeichert. Die dritte Datei wird für das Einlesen einer INI Datei verwendet. Alle Kategorien und deren Einträge werden gelesen, in einer eigenen, inneren Klasse gespeichert und der jeweils letzte Eintrag verändert in „Testing around“, in die Datei geschrieben, wieder in den eigentlichen Wert verändert und erneut in die Datei geschrieben.
Für den Ausführungsgeschwindigkseitsvergleich werden die beiden Testklassen jeweils 500 mal erstellt und dessen Testmethode ausgeführt. Die Zeit wird für jede Ausführung gemessen und der Durchschnitt genommen.
Für den Entwicklungsgeschwindigkeitsvergleich werden die Programme selbst betrachtet. Dabei wird vor allem auf die notwendigen Codezeilen geschaut (ohne Leerzeilen, jedoch mit Lesbarkeit nach den Codestyleempfehlungen) und wie die Syntax der Sprache ist, vor allem die Lesbarkeit.

5.2.3 Entscheidung

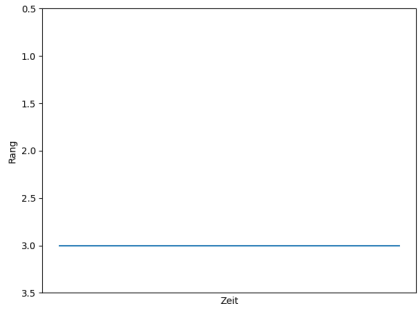
Zukunftssicherheit der Programmiersprachen
Hier werden die Gefundenen letzten Update Daten von den Programmiersprachen ausgewertet. Diese sind in 2.2.7 zu finden.

Java
Da das letzte Update in der 18 Monate Grenze liegt, wird der Verlauf der Beliebtheit ist folgend graphisch dargestellt:

(a) Java TIOBE-Index



(b) Java GitHut 2.0-Index

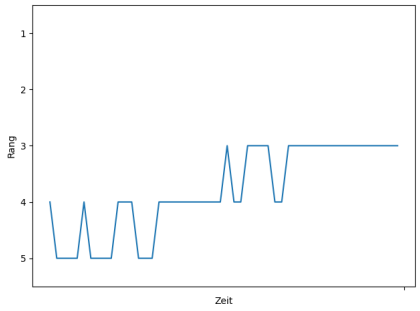


Der Verlauf im TIOBE Index deutet an, dass nach Java immer weniger gesucht wird. Das kann daran liegen, dass die Sprache nicht mehr so modern und beliebt ist, oder dass sehr viele Entwickler bereits Java kennen. Auf GitHub ist Java seit dem Quartal 03/2015 auf Platz 3.
Letztendlich würde ich sagen, dass Java nicht mehr so gefragt ist, wie es mal war. Da jedoch sehr viele Unternehmen noch Java verwenden, und Java bei Android als noch eine große Rolle spielt, wird Java die nächsten Jahre bis Jahrzehnte weiterhin bestehen.

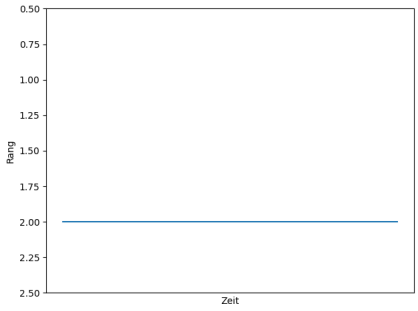
Python
Auch hier liegt das letzte Update in der 18 Monate Grenze. Daher wird der Verlauf der Beliebtheit auch hier graphisch angegeben:
Wie bei Java ist auch bei Python die Beliebtheit auf GitHub gleich geblieben, nur dass Python mit Platz 2 beliebter ist als Java.
Im TIOBE Index hat die beliebtheit am Anfang zwischen Platz 4 und 5 geschwankt, bi dem späteren Verlauf wurde Python jedoch beliebter, bis sie seit einigen Monaten auf Platz 3 liegt. Ein Grund dürfte die, zur Zeit, sehr hohe Nachfrage nach Python sein.
Da Python immer mehr auf Suchmaschinen gesucht wird und die Beliebtheit bei Pull-Requests auf GitHub die zweithöchste ist, würde ich Python als sehr Zukunftssicher einstufen.

C♯
Da die Vorschauversion keine 18 Monate entfernt veröffentlicht wurde, wird der

(a) Python TIOBE-Index

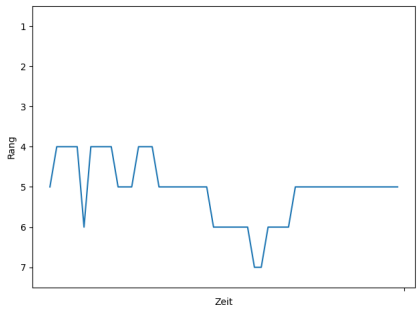


(b) Python GitHut 2.0-Index

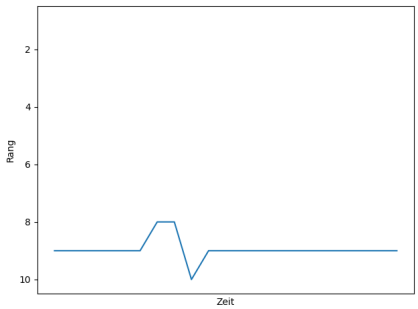


Beliebtheitsverlauf von C♯ folgend graphisch dargestellt:

(a) C♯ TIOBE-Index



(b) C♯ GitHut 2.0-Index



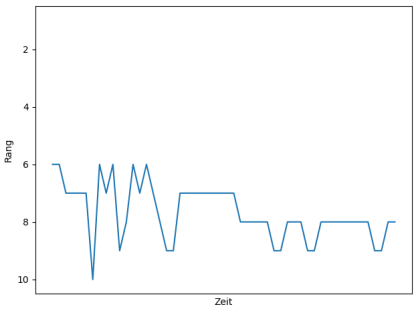
C♯ ist zwar Syntax-Technisch sehr ähnlich zu Java, jedoch nicht so beliebt. Nach dem TIOBE Index hat C♯ bis vor wenige Monate einige Schwankungen von den Suchanfragen her. Es ist bis vor wenige Monate ein ständiges Steigen und Fallen, jedoch hält sich C♯ insgesamt ungefähr auf Platz 5. Anders sieht es auf GitHub aus. Hier sind sehr wenige Schwankungen. Auch hier bleibt insgesamt der Rang von C♯ ziemlich gleich, auf Platz 9.
Da C♯ nicht unbeliebter wird und vor allem sehr viel Verwendung im .NET Framework für die Desktop, Mobile, Web und Spieleentwicklung hat, schätze ich C♯ als sehr Zukunftssicher ein. Ein Hauptfaktor hierfür ist noch, dass Microsoft, als ein gigantisches Unternehmen, hinter C♯ steht.

Visual Basic .NET
Visual Basic .NET wird nicht mehr weiter betrachtet, da das letzte Update 2 Jahre her ist und die Programmiersprache nicht mehr neue Features bekommt.

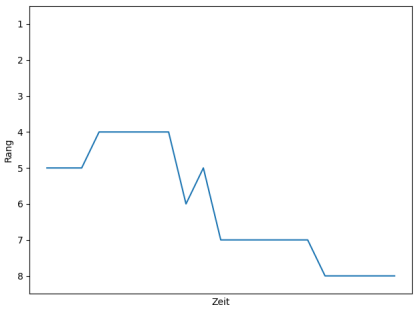
PHP
PHP ist in der 18 Monatsgrenze des letzten Updates und der Verlauf der

Beliebtheit wird auch hier graphisch angezeigt:

(a) PHP TIOBE-Index



(b) PHP GitHut 2.0-Index

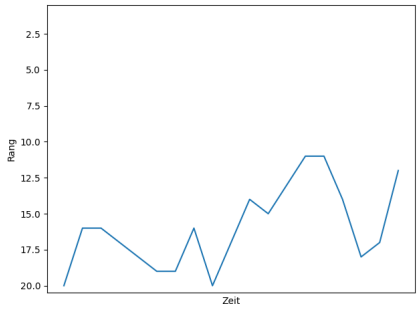


Bei PHP gibt es im TIOBE Index anfangs starke Schwankungen der Beliebtheit. Nur sind diese Schwankungen bei PHP stärker. PHP fing an auf Platz 6, und schwankt dann zwischen Platz 6 und 10, bis sie letztendlich zwischen Platz 8 und 9 schwankt. Sichtbar ist hier eine sinkende Tendenz der Beliebtheit.
Auch auf GitHub kann sich PHP immer weniger auf hohen Plätzen halten. Die Sprache ist erst leicht beliebter geworden, bis sie letztendlich von Platz 5 am Anfang auf Platz 8 im letzten Quartal abgerutscht ist.
Allerdings wird PHP sehr viel als Backend Sprache auf den Servern eingesetzt. Ich stufe PHP nicht als ganz Zukunftssicher ein. Sie wird wahrscheinlich noch sehr lange Zeit Updates erhalten, da scheinbar aber immer weniger Entwickler mit PHP arbeiten wollen, sehe ich keine sehr große Zukunft für PHP.

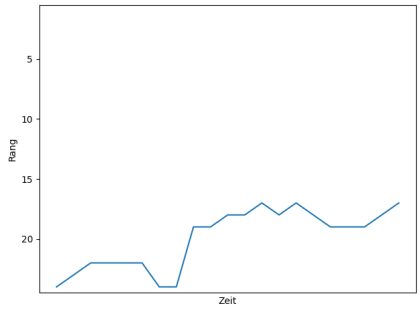
Delhpi/Object Pascal
Da das letzte Update über 18 Monate her ist wird Delhpi/Object Pascal nicht mehr vom Beliebtheitsverlauf betrachtet.

Groovy
Groovy liegt in der 18 Monate Grenze seit dem letzten Update und der Beliebtheitsverlauf von Groovy wird folgend graphisch angegeben:
Bei Groovy ist der Trend nicht ganz erkennbar.
Der TIOBE Index zeigt am Anfang einen Anstieg der Suchen nach Groovy, danach werden die Anfragen jedoch wieder weniger. Später erreicht Groovy seinen Höhepunkt auf Platz 11, jedoch fällt die Nachfrage direkt wieder und steigt dann wieder auf Platz 12.
Auch auf GitHub steigert Groovy sich vom Platz und fällt wenig später wieder. Dann steigt Groovy wieder, fällt und steigt.
Ich würde dennoch Groovy als Zukunftssicher einstufen, denn insgesamt wurde die Beliebtheit von Groovy nicht schlechter als sie am Anfang war. Sie steigert sich sogar im Vergleich vor 4-5 Jahren. Nach dem TIOBE Index sind es zwar extreme Unterschiede, jedoch ist der Anstieg auf GitHub stärker als der Fall.

(a) Groovy TIOBE-Index

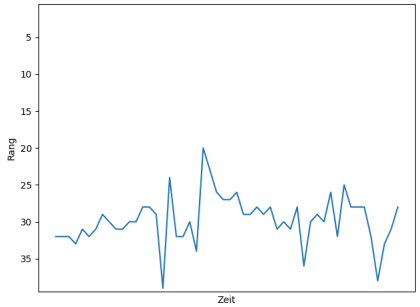


(b) Groovy GitHut 2.0-Index

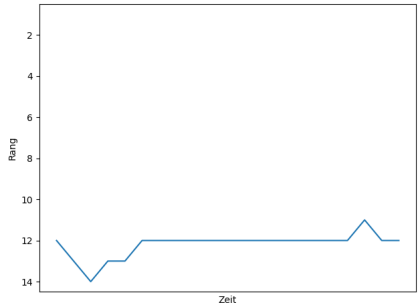


Scala
Scala liegt mit dem letzten Update in der 18 Monate Grenze. Der Beliebtheitsverlauf wird folgend graphisch dargestellt:

(a) Scala TIOBE-Index



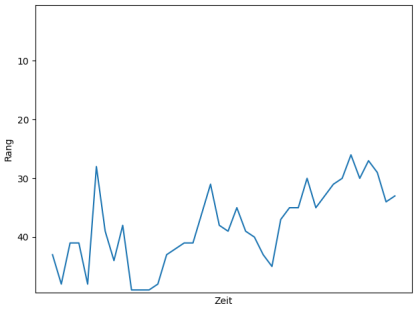
(b) Scala GitHut 2.0-Index



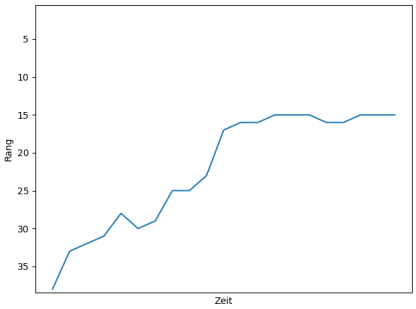
In dem TIOBE Index schwankt der Rang von Scala als hin und her.
Anfangs wird Scala beliebter von den Suchanfragen her, dann fällt die Beliebtheit. Dann steigt die Beliebtheit noch höher, und fällt direkt wieder. Zurzeit steigt die Beliebtheit wider, wie lange der Anstieg anhält, ist jedoch nicht klar. Nach dem TIOBE Index ist Scala nicht die beliebteste Sprache, jedoch bleibt der Rang auf die Zeit betrachtet ziemlich gleich.
Scala bewegt sich in dem GitHut Index auf die ganze Zeit gesehen meist auf Platz 12. Es gibt wenige Abweichungen. Damit ist Scala ziemlich stabil von den Pull Requests.
Ich stufe Scala auch als Zukunftssicher ein. Der Grund ist die hohe Stabilität auf GitHub. Die Suchanfragen nach Scala bringen die Sprache zwar ins Schwanken, aber letztendlich ist Scala auch im TIOBE Index Trendmäßig gleichgeblieben.

Kotlin
Auch Kotlin liegt in der 18 Monate Grenze seit dem letzten Update und der graphische Verlauf der Beliebtheit wird graphisch dargestellt:

(a) Kotlin TIOBE-Index



(b) Kotlin GitHut 2.0-Index



Wie bei Scala ist auch bei Kotlin ein dauerhaftes Schwanken im TIOBE Index. Bei Kotlin hingegen ist der Rang letztendlich wesentlich höher als am Anfang. Auf GitHub erfreut sich Kotlin immer größerer Beliebtheit. Im Gegensatz zum Anfang hat sich der Rang mehr als verdoppelt und stabilisiert sich letztendlich um den Rang 15.
Durch den letztendlich höheren Rang im TIOBE Index und den viel höheren Rang im GitHut Index stufe ich Kotlin als Zukunftssicher ein.

Folgende Programmiersprachen wurden als Zukunftssicher eingeschätzt und werden für die Performanzvergleiche betrachtet:
Java, Python, C♯, Groovy, Scala und Kotlin.

Ausführungsgeschwindigkeit
Wie in 2.2.8 zu erkennen ist, sind ist jede der Programmiersprachen mindestens einmal schneller als die anderen 3.
Da die Ausführungszeit hier unterschiedlich ist und nicht der Ausführungszeitreihenfolge aus der ersten Quelle direkt nachkommt, werde ich eigene Tests schreiben. Diese Tests werden im nächsten Abschnitt genauer beschrieben, nachdem nach bestimmten Bibliotheken gesucht wurde.

Entwicklungsgeschwindigkeit
Anhand der gefundenen Bibliotheken aus 2.2.9 werden dann Performanzvergleiche geführt. Dazu wird eine Datenbank eingerichtet, welche 10.000 Einträge in einer Tabelle enthält. Anschließend wird diese Tabelle ausgelesen und die Datenbankeinträge in einer Liste gespeichert. Weiterhin wird eine INI Datei mit 4 Kategorien und jeweils 5 Einträgen pro Katergorie erstellt. Diese wird eingelesen, jeder Eintrag in einer Liste gespeichert, der letzte Eintrag jeder Kategorie verändert und wieder zurückverändert.
Die Ausführungsgeschwindigkeit wird mithilfe von der Systemzeit gemessen. Dazu wird die aktuelle Systemzeit vor dem Codeblock gespeichert und diese Zeit von der Systemzeit nach dem Codeblock subtrahiert. Die Programme werden jeweils 50x ausgeführt und der Mittelwert der Ergebnisse genommen.

Für Entwicklungsgeschwindigkeit wird anhand der notwendigen Codezeilen und dem notwendigen Overhead beurteilt.

Ergebnisse Ausführungsgeschwindigkeits- und Entwicklungsgeschwindigkeitstests
Was sind die Ergebnisse?
Die Tests wurden auf einem Linux Mint 20 Cinnamon Laptop ausgeführt, der folgende Hardware hat:
Prozessor: Intel© Core™ i5-7200U CPU @ 2.50GHz × 2
Arbeitsspeicher: 8GB
Grafikkarte: Intel Corporation HD Graphics 620

Die folgende Tabelle zeigt die Ergebnisse der Tests:

Programmiersprache	Datenbank	INI Datei	Codezeilen gesamt
Java	0,0071 s	0,0025 s	142
Python	0,0488 s	0,0264 s	67
C♯	0,0210 s	0,0013 s	134
Groovy	0,3625 s	0,0063 s	85
Kotlin	0,0086 s	0,0022 s	68
Scala	0,0084 s	0,0035 s	106

Weiterhin wurden die Tests auf einem Windows 10 Computer ausgeführt, welcher folgende Hardware hat:
Prozessor: Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz, 3408 MHz, 4 Kern(e), 4 logische(r) Prozessor(en)
Arbeitsspeicher: 16GB
Grafikkarte: NVIDIA GeForce GTX 1060 6GB

Hier sind die Ergebnisse der Geschwindigkeitstests:

Programmiersprache	Datenbank	INI Datei
Java	0,0062 s	0,0037 s
Python	0,0326 s	0,0191 s
C♯	0,0088 s	0,0029 s
Groovy	0,6326 s	0,0049 s
Kotlin	0,0066 s	0,0036 s
Scala	0,0071 s	0,0039 s

Interpretation der Ergsbnisse und beantwortung, welche Programmiersprachen am besten geeignet sind anhand eines Rankings
Zu erkennen ist, dass Kotlin und Python, bis auf eine Zeile Unterschied, am wenigsten Codezeilen benötigen. Das liegt daran, dass Python keine Sichtbarkeitsmodifier wie public und private hat und Kotlin automatisch getter und setter generiert, welche sich aber nach belieben ersetzen lassen. Groovy hat auch automatisch generierte getter und setter, im Unterschied zu Kotlin können die Felder aber nicht im Methodenaufruf Styl deklariert werden. Scala erlaubt getter Methoden zu schreiben, die keine Funktionsaufrufklammern haben, wodurch auch wenige Codezeilen eingespart werden, auch erlaubt Scala,

wie Kotlin, eine Funktionsaufrufklammern Schreibeweise für die Klassenparameter. Am meisten Codezeilen benötigen Java und C♯, wobei bei C♯ von der Codingconvention die Methodenklammern in einer neuen Zeile empfohlen werden, was insgesamt 22 Zeilen ausmacht. Diese unterstützen beide keine Funktionsaufrufklammern Schreibweise für Klassenparameter, auch wird keine automatische getter und setter Generierung unterstützt. Jedoch erlaubt C♯ eine Kurzschreibweise für getter und setter.
Von der Syntax her ist Python eine sehr lesbare Sprache, egal in welchen Programmen, denn die Syntax erzwingt dies. Die anderen haben keine Syntax, die das Einrücken erzwingt.
Einen kleinen, aber doch netter Unterschied, haben Java und C♯ zu den anderen 4 Programmiersprachen, denn in nur in Java und C♯ sind die Semikolons notwendig, in den anderen nicht. So entstehen weniger Syntaxfehler.
Java hängt wenigen sehr netten Features gegenüber den anderen Programmiersprachen zurück, dazu zählt die Stringinterpolation, also das Verwenden von Variablen und Funktionsaufrufe innerhalb des Strings, ohne eine String Format explizit machen zu müssen und das überschreiben der Operatoren.
Python, Kotlin und Groovy erlauben Higher Order Functions, also Funktionen, die außerhalb einer Klasse geschrieben werden können. Das ist ein nettes nice to have, denn so sind keine Klassen für einzelne Methoden notwendig.

Entwicklungstechnisch sortiere ich die 6 Programmiersprachen deswegen folgend an:
1. Python: Die Higher Order Functions und vor allem die Syntax machen Python zu einer sehr lesbaren Sprache. Neue Klassenvariablen brauchen nicht explizit deklariert werden, diese können auch in einer Methode deklariert werden. Die Kombination macht Python sehr angenehm zum Entwickeln.
2. Kotlin: Das automatische generieren von getter und setter Methoden durch die data class, die extension Methoden wie run, welche einen Codeblock ausführen lässt, als wäre dieser Teil der Objektklasse, machen es nochmal einfacher zu Entwickeln. Ebenfalls macht das weglassen von dem „new“ Schlüsselwort eine Objekterzeugung auch angenehmer.
3. Groovy: Die automatische getter und setter machen das Klassenschreiben angenehmer. Ebenfalls, wie in Kotlin, lässt die with Methode einen Codeblock ausführen, als wäre es Teil der Objektklasse. Groovy braucht leider ein Paar mehr Zeilen Code, wodurch es etwas später eingeordet wird.
4. C♯: Obwohl Scala weniger Codezeilen benötigt und keine Semikolons notwendig sind, ist C♯ angenehmer zu entwickeln. Die stark verkürzten getter und setter machen Klassen schreiben viel angenehmer.
5. Scala: Wegen dem notwenigen Schreiben von getter und setter macht es das Entwickeln langsamer im Gegensatz zu Platz 1 bis 4, wodurch Scala auf Platz 5 landet.
6. Java: Java hängt den Programmiersprachen in einigen Aspekten nach, darunter automatisch generierte getter und setter, Stringinterpolation und vor allem Operatorenüberschreibung. Deswegen landet Java auf Platz 6.

Nach den Ergebnissen der Ausführungszeit ordne ich die Programmiersprachen wie folgt an:
1. Java: Sowohl auf Linux als auch auf Windows werden die Datenbankeinträge schneller gelesen und in ein Objekt reingepackt. Bei dem einlesen der Ini Datei sind C♯ und Kotlin nur leicht schneller, jedoch erfolgen Datenbankabfragen häufiger als Dateien einlesen.
2. Kotlin: Kotlin ist nur leicht langsamer als Scala im Datenbanktest auf Linux und etwas schneller auf Windows. Im INI Test ist Kotlin jedoch beide Male schneller.
3. Scala: Scala ist nur leicht langsamer als Kotlin im Ini Test und doch wesentlich langsamer als C♯, jedoch ist Scala auf Linux im Datenbanktest schneller als C♯ und etwas langsamer auf Windows.
4. C♯: Auf Linux ist C♯ wesentlich langsamer als auf Windows bei dem Datenbanktest. Insgesamt ist C♯ bei diesem Test am Platz 4. Anders sieht es bei dem INI Test aus, denn dort ist C♯ auf beiden Betriebssystemen am schnellsten. Da jedoch mehr Daten abgefragt werden und die Ausführung auf Linux so lange dauert, was wahrscheinlich daran liegt, dass C♯ von Microsoft entwickelt wird und es bei Windows für die Entwicklung verwendet wird, wird C♯ hier eingeordnet.
5. Python: Python braucht zwar beim INI Test auf beiden Betriebssystemen am längsten, jedoch ist Python beim Datenbanktest noch wesentlich schneller als Groovy.
6. Groovy: Groovy braucht bei dem Datenbanktest am längsten, und das mit weitem Abstand. In der Praxis ist diese Zeit viel zu lange.

Insgesamt würde ich die Programmiersprachen wie folgt anordnen:
1. Kotlin: Kotlin ist bei der Ausführungsgeschwindigkeit und Entwicklungsgeschwindigkeit auf Platz 2. Kotlin ist nur minimal langsamer als Java, lässt sich aber viel schneller entwickeln.
2. Java: Java ist zwar bei der Entwicklung auf dem letzten Platz, jedoch ist Java von den ganzen Programmiersprachen die schnellste. Die Entwicklung ist zwar nicht so angenehm wie bei den anderen Prorgrammierpsrachen, jedoch sind die meisten Aspekte nice to have Sachen.
3. Scala: Scala ist auf Platz 3 der Ausführungseschwindigkeit und auf Platz 5 bei der Entwicklung. Da die Ausführungsgeschwindigkeit aber entscheidender ist als die Entwicklung, ordne ich Scala hier ein.
4. C♯: Bei beiden Performanzvergleichen ist C♯ auf Platz 4. Wenn der Server ein Windows Betriebssystem hat, ist C♯ sehr nahe an Scala von der Ausführungsgeschwindigkeit dran.
5. Python: Die Entwicklung in Python ist sehr sehr angenehm, jedoch dauert die Ausführung der Programme sehr lange. Das wird daran liegen, dass Python interpretiert wird. Deshalb wird Python hier eingeordnet.
6. Groovy: Die Entwicklung ist auch in Groovy sehr angenehm, jedoch ist die Ausführungsgeschwindigkeit noch langsamer als die von Python bei den Datenbanken, was in der Praxis sehr schlecht ist.

Anhand der Anordnung wären die Top 4 Programmiersprachen Kotlin, Java, Scala und C♯. Da jedoch Kotlin, Scala, Java und Groovy auf der JVM lauffähig sind, wird für den dritten Abschnitt noch Python mitbetrachtet neben den Top 4 Programmiersprachen. Der Grund ist, dass ein Python Framework eventuell etwas Geschwindigkeit aufbauen kann. *Entscheidungen, die für die Evaluation vorgenommen wurden, welche die Validität der Regebnisse gefährden können*
Folgende Entscheidungen wurde vorgenommen:

1. Version der Programmiersprache: Bei der Entwicklungszeit spielt die Version der Programmiersprache eine Rolle. Der Grund ist, dass bei neueren Versionen oft neue Features hinzukommen, die die Entwicklungszeit deutlich reduzieren können (z.B. foreach Schleifen anstatt klassische for Schleifen).
2. Betriebssystem: Das Betriebssystem spielt eine wichtige Rolle: Programmiersprachen, die für die Entwicklung des Betriebssystems verwendet werden, laufen in der Regel schneller als Programmiersprachen, die nicht bei der Betriebssystementwicklung verwendet wurden.
3. Compiler: Die Wahl des Compilers spielt eine entscheidente Rolle. Denn bei den meisten Skriptsprachen, wie Python, gibt es den Compiler, der offiziell bereitgestellt wird. Aber es gibt auch Compiler von anderen Organisationen, welche z.B. das Just in Time Kompilieren unterstützen. Der Compiler wirkt sich auf die Ausführungsgeschwindigkeiten aus.

6 Auswahl geeigneter Web-Frameworks

In diesem Kapitel werden für die übrigen Programmiersprachen Web Frameworks herausgesucht, welche einen GUI Designer bereitstellen.

6.1 Auswahlkriterien

Für die Web Framework Auswahl werden folgende Kriterien verwendet:

1. Web Framework basiert auf einer der resultierenden Programmiersprache aus dem letzten Schritt.
2. Das Web Framework oder eine Erweiterung hat einen GUI Designer.
3. Das Web Framework ist Zukunftssicher.
4. Geringe Ausführungsgeschwindigkeit des Web Frameworks.
5. Geringe Entwicklungszeit des Web Frameworks.

6.2 Vorgehen bei der Auswahl

Wie in den anderen Abschnitten, werden auch hier die Kriterien nacheinander abgearbeitet.
Folgendes Vorgehen wird für die einzelnen Schritte vorgenommen:
Web Framework basiert auf einer resultierenden Programmiersprache aus 5.2.3 und hat einen GUI Designer
Es werden Web Frameworks mit GUI Designer für die resultierenden Sprachen aus 5.2.3 gesucht. Es werden nur Frameworks akzeptiert, welche einen GUI Designer direkt bereitstellen oder eine Erweiterung des Frameworks einen Designer bereitstellt.
Das Web Framework ist Zukunftssicher
Für jedes gefundene Web Framework wird herausgesucht, ob das letzte Update in den letzten 12 Monaten war. Weiterhin wird geschaut, wer dieses Framework entwickelt und welche Unternehmen dieses Framework verwenden.
Anhand der gefundenen Informationen wird eine Prognose abgegeben, ob dieses Framework zukunftssicher scheint.
Ausführungs- und Entwicklungsgeschwindigkeit
Die Performanz für die übrigen Frameworks wird auch hier anhand von einem kleinen Programm getestet.
Hierfür wird eine neue Datenbank aufgesetzt, welche 3 Tabellen mit je 4 Spalten hat. Es wird weiterhin eine INI Datei geschrieben, welche verschiedene Konfigurationen speichert. Diese Konfigurationen enthalten einen Namen, eine Beschreibung und einen SQL Statement.
Auf der Webseite bekommt man die Option eine Konfiguration auszuwählen. Dann wird einem der Name und Beschreibung angezeigt.Ebenfalls werden die Daten aus der Datenbank ausgelesen und angezeigt. Es gibt die Option nur eine bestimmte Anzahl an Daten zu holen.

6.3 Entscheidung

Web Framework basiert auf einer resultierenden Programmiersprache aus 5.2.3 und hat einen GUI Designer
Das Web Framework ist Zukunftssicher
Ausführungs- und Entwicklungsgeschwindigkeit

7 Entwicklung eiens Moduls in verschiedenen Frameworks

8 Entscheidung für ein Framework

Literatur

[NAD] Native Entwicklung: https://searchsoftwarequality.techtarget.com/definition/native-application-native-app (23.10.2020)

[BNA] Vorteile nativer Anwendungenhttps://mlsdev.com/blog/native-app-development-vs-web-and-hybrid-app-development (26.10.2020)

[DANA] Vor- und Nachteile einer Nativen, Web und Hybriden Anwendunghttps://threefourteen.ie/blog/advantages-and-disadvantages-of-web-native-and-hybrid-apps/ (26.10.2020)

[CPD] Cross-Platform Definition: https://www.techopedia.com/definition/30026/cross-platform-development (26.10.2020)

[CPDA] Cross-Platform Vorteile: https://medium.com/successivetech/benefits-of-cross-platform-development-bfa5f708c0a4 (26.10.2020)

[CPDD] Cross-Platform Vorteile: https://www.focaloid.com/blog/the-pros-and-cons-of-cross-platform-apps (26.10.2020)

[CPDDA] Cross-Platform Vor- und Nachteile: https://codeburst.io/native-vs-cross-platform-app-development-pros-and-cons-49f397bb38ac (26.10.2020)

[WDD] Web Entwicklung Definition: https://www.techopedia.com/definition/23889/web-development (01.10.2020)

[WDF] Web Entwicklung Funktionsweise: https://networkencyclopedia.com/web-application/ (01.10.2020)

[BWD] Vorteile Web Entwicklung: https://www.nirvanacanada.com/businessonline/the-benefits-of-web-application-development-for-businesses/ (01.10.2020)

[ADWD] Vor- und Nachteile Web Entwicklung: https://en.yeeply.com/blog/advantages-and-disadvantages-of-web-app-development/ (06.10.2020)

[DWD] Nachteile Web Entwicklung: https://www.linkedin.com/pulse/advantages-disadvantages-web-app-development-luis-picurelli (01.10.2020)

[HD] Vorteile Web Entwicklung: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/ (06.10.2020)

[TI] TIOBE Index Beschreibung (über erste Tabelle): https://www.tiobe.com/tiobe-index/ (30.10.2020)

[GH2.0] GitHut 2.0 Beschreibung (unter PI-Chart): https://madnight.github.io/githut/#/pull_requests/2020/3 (30.10.2020)

[HLLL] „Hochsprachen“ und „Tiefsprachen“: https://careerkarma.com/blog/high-level-and-low-level-languages/ (30.10.2020)

[TI1219] TIOBE Index von Dezember 2019: https://web.archive.org/web/20191231193250/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI0120] TIOBE Index von Januar 2020: https://web.archive.org/web/20200131222240/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI0220] TIOBE Index von Februar 2020: https://web.archive.org/web/20200229165403/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI0320] TIOBE Index von März 2020: https://web.archive.org/web/20200328111140/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI0420] TIOBE Index von April 2020: https://web.archive.org/web/20200430164914/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI0520] TIOBE Index von Mai 2020: https://web.archive.org/web/20200531144216/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI0620] TIOBE Index von Juni 2020: https://web.archive.org/web/20200627171928/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI0720] TIOBE Index von Juli 2020: https://web.archive.org/web/20200731064026/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI0820] TIOBE Index von August 2020: https://web.archive.org/web/20200831103505/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI0920] TIOBE Index von September 2020: https://web.archive.org/web/20200929231011/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[TI1020] TIOBE Index von Oktober 2020: https://web.archive.org/web/20201028100442/https://www.tiobe.com/tiobe-index/ (30.10.2020)

[GH419] GitHut 2.0 vom Quartal 4/2019: https://madnight.github.io/githut/#/pull_requests/2019/4 (31.10.2020)

[GH120] GitHut 2.0 vom Quartal 1/2020: https://madnight.github.io/githut/#/pull_requests/2020/1 (31.10.2020)

[GH220] GitHut 2.0 vom Quartal 2/2020: https://madnight.github.io/githut/#/pull_requests/2020/2 (31.10.2020)

[GH320] GitHut 2.0 vom Quartal 3/2020: https://madnight.github.io/githut/#/pull_requests/2020/3 (31.10.2020)

[JL1] Java Sprachlevel Oracle: https://docs.oracle.com/javase/specs/jls/se8/html/jls-1.html (01.11.2020)

[JL2] Java Sprachlevel Medium: https://medium.com/@escalesolutions/java-tutorial-best-five-tutorials-to-master-java-programming-language-in-2018-9916987df4 (01.11.2020)

[CL1] C Sprachlevel Course Report: https://www.coursereport.com/blog/a-guide-to-low-level-programming-for-beginners (01.11.2020)

[CL2] C Sprachlevel Medium: https://medium.com/swlh/a-high-level-introduction-to-the-c-programming-language-f5ada5a5bd5d (01.11.2020)

[C++L] C++ Sprachlevel GeeksForGeeks: https://www.geeksforgeeks.org/introduction-to-c-programming-language/ (01.11.2020)

[PL1] Python Sprachlevel Python: https://www.python.org/doc/essays/blurb/ (01.11.2020)

[PL2] Python Sprachlevel hackr.io: https://hackr.io/blog/python-programming-language (01.11.2020)

[C♯L1] C♯ Sprachlevel Medium: https://medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb (01.11.2020)

[C♯L2] C♯ Sprachlevel Cateer Karma: https://careerkarma.com/blog/c-plus-plus-vs-c-sharp/ (01.11.2020)

[VBNETL1] Visual Basic .NET Sprachlevel Researchgate: https://www.researchgate.net/publication/267788607_CIL_Programming_Under_the_Hood_of_NET (01.11.2020)

[VBNETL2] Visual Basic .NET Sprachlevel Acseduonline: https://www.acseduonline.com/courses/information-technology-computers-5/visual-basic-net-bit101-442.aspx (01.11.2020)

[JSL1] JavaScript Sprachlevel PluralsightPluralsight: https://www.pluralsight.com/paths/javascript-core-language (01.11.2020)

[JSL2] JavaScript Sprachlevel Tutsplus: https://code.tutsplus.com/tutorials/what-is-javascript--cms-26177 (01.11.2020)

[PHPL] PHP Sprachlevel EDUCBA: https://www.educba.com/high-level-languages-vs-low-level-languages/ (02.11.2020)

[SQLL] SQL Sprachlevel Oracle Patches: https://oracle-patches.com/en/databases/sql/4191-sql-programming-language-and-oracle-novations (02.11.2020)

[SWL1] Swift Sprachlevel bestprogramminglanguagefor.me: https://www.bestprogramminglanguagefor.me/why-learn-swift (02.11.2020)

[SWL2] Swift Sprachlevel Infoq: https://www.infoq.com/news/2014/06/apple-swift/ (02.11.2020)

[RL1] Ruby Sprachlevel bestprogramminglanguagefor.me: https://www.bestprogramminglanguagefor.me/why-learn-ruby (02.11.2020)

[RL2] Ruby Sprachlevel isleofruby: http://isleofruby.org/learn-ruby-over-other-programming-languages/ (02.11.2020)

[DL2] Delphi/Object Pascal Sprachlevel isleofruby: http://docwiki.embarcadero.com/RADStudio/Sydney/en/Language_Overview (02.11.2020)

[DL2] Delphi/Object Pascal Sprachlevel isleofruby: https://kolmck.net/general-information-delphi-programming-language/ (02.11.2020)

[OCL] Objective-C Sprachlevel gsdh: https://www.gsdh.org/en/objectivec.html (02.11.2020)

[ASL1] Assembly Language Sprachlevel isleofruby: https://www.tutorialspoint.com/assembly_programming/index.htm (02.11.2020)

[ASL2] Assembly Language Sprachlevel gsdh: https://www.educba.com/what-is-assembly-language/ (02.11.2020)

[GL] Go Sprachlevel Medium: https://medium.com/@imdanielsp/go-programming-language-analysis-i-989eab66a34a (02.11.2020)

[RL] R Sprachlevel Codementor: https://www.codementor.io/@sayantinideb/r-vs-python-best-programming-language-for-data-science-and-analysis-te05xg (03.11.2020)

[MLL] MATLAB Sprachlevel Springerprofessional: https://www.springerprofessional.de/en/matlab-programming-for-numerical-analysis/1903816 (03.11.2020)

[MLL] D Sprachlevel Dlang: https://dlang.org/overview.html (03.11.2020)

[MLL] D Sprachlevel Computerhope: https://www.computerhope.com/jargon/d/dlang.htm (03.11.2020)

[VBL] Perl Sprachlevel Perl Dokumentation: https://perldoc.perl.org/perlfaq1 (03.11.2020)

[VBL] Perl Sprachlevel GeeksForGeeks: https://www.geeksforgeeks.org/perl-programming-language/ (03.11.2020)

[RL] Rust Sprachlevel dev.to: https://dev.to/lambdude/comment/1okc (03.11.2020)

[GL] Groovy Sprachlevel cs.com: https://wiki.c2.com/?GroovyLanguage (03.11.2020)

[DTL1] Dart Sprachlevel codecarbon: https://codecarbon.com/pros-cons-dart-language/ (03.11.2020)

[DTL2] Dart Sprachlevel javatpoint: https://www.javatpoint.com/dart-programming (03.11.2020)

[TS] Typescript offizielle Webseite: https://www.typescriptlang.org/ (03.11.2020)

[SCL1] Scala Sprachlevel offizielle Webseite: https://www.scala-lang.org/ (03.11.2020)

[SCL2] Scala Sprachlevel Toptal: https://www.toptal.com/scala/why-should-i-learn-scala (03.11.2020)

[KL] Kotlin offizielle Webseite: https://kotlinlang.org/ (03.11.2020)

[LL] Lua Sprachlevel Medium: https://medium.com/turing-ninjas/the-double-edged-programming-language-lua-easy-for-kids-to-learn-and-also-powerful-enou (03.11.2020)

[Vaadin] Java Vaadin Framework: https://vaadin.com/ (04.11.2020)

[Anvil] Python Anvil Framework: https://anvil.works/ (04.11.2020)

[Radzen] C♯ Radzen (Blazor) Framework: https://www.radzen.com/blog/build-web-applications-with-radzen-blazor-csharp-no-javascript-required/ (04.11.2020)

[WF1] Artikel über WebForms: https://docs.microsoft.com/en-us/archive/msdn-magazine/2001/may/asp-net-web-forms-let-you-drag-and-drop-your-way-to-powerful-web-apps (05.11.2020)

[WF2] WebForms Webseite: https://dotnet.microsoft.com/apps/aspnet/web-forms (05.11.2020)

[WX] Webix Webseite: https://webix.com/ (06.11.2020)

[IP] Impresspages: https://www.impresspages.org/ (06.11.2020)

[RS] Raudus Webseite: https://www.raudus.com/ (07.11.2020)

[TSW] TypeScript Webix: https://blog.webix.com/typescript-types-in-webix-ui-framework/ (08.11.2020)

[SV] Scala Vaadin: https://vaadin.com/docs/v8/framework/getting-started/getting-started-scala.html (08.11.2020)

[KV] Kotlin Vaadin: https://vaadin.com/kotlin (08.11.2020)

[JH] Java Gründung: https://www.javatpoint.com/history-of-java (11.11.2020)

[JLR] Java letztes Update: https://www.codejava.net/java-se/java-se-versions-history (11.11.2020)

[PH1] Python Gründung Teil 1: https://www.geeksforgeeks.org/history-of-python/ (12.11.2020)

[PH2] Python Gründung Teil 2: https://www.javatpoint.com/python-history (12.11.2020)

[PLR] Python letztes Update: https://www.python.org/downloads/ (12.11.2020)

[CSH] C♯ Gründung: https://medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb (12.11.2020)

[CSLR1] C♯ letztes Update 8.0: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-8 (12.11.2020)

[CSLR2] C♯ letztes Update 9.0: https://devblogs.microsoft.com/dotnet/welcome-to-c-9-0/ (12.11.2020)

[VBNETH] Visual Basic .NET Gründung: https://www.guru99.com/vb-net-introduction-features.html (12.11.2020)

[VBNETLR1] Visual Basic .NET letztes Update 16.0: https://docs.microsoft.com/en-us/dotnet/visual-basic/whats-new/#visual-basic-160 (12.11.2020)

[VBNETLR2] Visual Basic .NET weiteres Update: https://devblogs.microsoft.com/vbteam/visual-basic-support-planned-for-net-5-0/ (12.11.2020)

[JSH] JavaScript Gründung: https://careerkarma.com/blog/javascript-history/ (13.11.2020)

[JSLR] JavaScript letztes Update: https://www.freecodecamp.org/news/javascript-new-features-es2020/ (13.11.2020)

[PHPH] PHP Gründung: https://www.php.net/manual/en/history.php.php (13.11.2020)

[PHPLR] PHP letzte Veröffentlichung: https://www.php.net/releases/index.php (13.11.2020)

[DH] Delphi/Object Pascal Gründung: https://vakrio.medium.com/history-of-delphi-from-pascal-to-embarcadero-delphi-xe-2-c6809366ae45 (13.11.2020)

[DLR] Delphi/Object Pascal letzte Veröffentlichung: https://wiki.freepascal.org/Delphi (13.11.2020)

[GH] Groovy Gründung: https://www.cleverism.com/skills-and-tools/groovy/ (13.11.2020)

[GLR] Groovy letztes Update: https://github.com/apache/groovy/releases (13.11.2020)

[TSH] TypeScript Gründung: https://coderlipi.com/typescript/typescript-history (15.11.2020)

[TSLR] TypeScript letztes Update: https://github.com/microsoft/TypeScript/releases (15.11.2020)

[SH] Scala Gründung: https://www.educative.io/courses/learn-scala-from-scratch/39BnN6DMZxr (16.11.2020)

[SLR] Scala letztes Update: https://github.com/scala/scala/releases (16.11.2020)

[SH] Kotlin Gründung: https://kotlinlang.org/docs/reference/faq.html (16.11.2020)

[SLR] Kotlin letztes Update: https://kotlinlang.org/releases.html (16.11.2020)

[CLBG] Computer Language Benchmarks Game: https://en.wikipedia.org/wiki/The_Computer_Language_Benchmarks_Game (17.11.2020)

[PU] Stromvergleich von Thenewstack.io: https://thenewstack.io/which-programming-languages-use-the-least-electricity/ (19.11.2020)

[EP] Aufsührungszeit vergleich von attractivechaos: https://attractivechaos.github.io/plb/ (20.11.2020)

[MCS] Mono-2.10.1: https://www.mono-project.com/docs/about-mono/releases/2.10.1/ (20.11.2020)

[JREJ] JRE-1.6.0_25: https://blogs.oracle.com/ebstech/sun-jre-16025-certified-with-oracle-e-business-suite (20.11.2020)

[V8JS] V8-r8384: https://v8.dev/ (20.11.2020)

[JMJS] JaegarMonkey-a95d42642281: https://wiki.mozilla.org/JaegerMonkey (20.11.2020)

[PPP] PyPy: https://www.pypy.org/ (20.11.2020)

[CP] CPython: https://github.com/python/cpython (20.11.2020)

[IP] IronPython: https://ironpython.net/ (20.11.2020)

[J] Jython: https://www.jython.org/ (20.11.2020)

[SS] ShedSkin: https://github.com/shedskin/shedskin (20.11.2020)

[TD] TIOBE Definition: https://www.tiobe.com/tiobe-index/programming-languages-definition/ (29.11.2020)

[JWF1] Mehrere Java Frameworks Vorstellung: https://hackr.io/blog/java-frameworks (14.12.2020)

[JVD] Vaadin Designer: https://vaadin.com/designer (14.12.2020)

[JJC] Judo.codes: https://www.judo.codes/features (15.12.2020)

[JOSBP] Eclipse OSBP: https://www.eclipse.org/osbp/index.html (15.12.2020)

[OEP] Opcenter Execution Pharma Hauptseite: https://www.plm.automation.siemens.com/global/de/products/manufacturing-operations-center/simatic-it-ebr.html (21.12.2020)