Hochschule für Berufstätige Darmstadt

– Fachbereich Informatik –

Automatische DNS-Zonen und Eintragsverwaltung in einer verteilten Multi-Cloud-Microservice-Architektur

Abschlussbericht zur

Berufspraktischen Phase

vorgelegt von

Karsten Siemer

Hempberg 1
22848, Norderstedt
Matrikelnummer: 906507

Referent	:	Herr Prof. Dr. Jürgen Kühnlein
Abgabetermin	:	17.01.2024

Die berufspraktische Phase wurde in der Zeit vom 01.10.2023 bis zum 17.01.2024 bei der *SDA SE Open Industry Solutions* in Hamburg durchgeführt.

EHRENERKLÄRUNG

Ich, Karsten Siemer, versichere durch meine Unterschrift, dass ich die vorliegende Arbeit selbstständig erstellt habe. Andere als die angegebenen Hilfsmittel habe ich nicht verwendet.

Soweit ich fremde Gedankengänge oder Texte verwendet habe, sind diese von mir als solche kenntlich gemacht und dem Urheber eindeutig zuordenbar. Dazu zählen sowohl wörtliche als auch nicht wörtliche Übernahmen.

Ort, Datum	Karsten Siemer

ABSTRACT

A short summary of the contents in English of about one page. The following points should be addressed in particular:

- Motivation: Why did this work come about? Why is the topic of the work interesting (for the general public)? The motivation should be abstracted as far as possible from the specific tasks that may be given by a company.
- Content: What is the content of this thesis? What exactly is covered in the thesis? The methodology and working method should be briefly discussed here.
- Results: What are the results of this work? A brief overview of the most important results as a teaser to read the complete thesis.

BTW: A great guide by Kent Beck how to write good abstracts can be found here:

https://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html

ZUSAMMENFASSUNG

Kurze Zusammenfassung des Inhaltes in deutscher Sprache von ca. einer Seite länge. Dabei sollte vor allem auf die folgenden Punkte eingegangen werden:

- Motivation: Wieso ist diese Arbeit entstanden? Warum ist das Thema der Arbeit (für die Allgemeinheit) interessant? Dabei sollte die Motivation von der konkreten Aufgabenstellung, z.B. durch eine Firma, weitestgehend abstrahiert werden.
- Inhalt: Was ist Inhalt der Arbeit? Was genau wird in der Arbeit behandelt? Hier sollte kurz auf Methodik und Arbeitsweise eingegangen werden.
- Ergebnisse: Was sind die Ergebnisse der Arbeit? Ein kurzer Überblick über die wichtigsten Ergebnisse als Teaser, um die Arbeit vollständig zu lesen.

Eine großartige Anleitung von Kent Beck, wie man gute Abstracts schreibt, finden Sie hier:

https://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGS

ABKÜRZUNGSVERZEICHNIS

API Application Programming Interface

AWS Amazon Web Services

GCP Google Cloud Platform

RBAC Role-Based Access Control

DevOps Development and Operations

Azure Microsoft Azure

DNS Domain Name System

CAPI Cluster API

IaC Infrastructure as Code

API Application Programming Interface

Teil I

ABSCHLUSSBERICHT

1 FIRMA

Die SDA SE ist ein in Hamburg ansässiges Unternehmen, das sich auf die Entwicklung von Plattformen und Ökosystemen spezialisiert hat.

1.1 FIRMENCHRONIK

Die Gründung der SDA SE erfolgte 2016 auf Initiative von Prof. Dr. Markus Warg, der als Ideengeber der SDA und Leiter des Instituts für Service Design in Hamburg, sowie Vorstand der SIGNAL IDUNA eine entscheidende Rolle spielte. Die SDA SE wurde als Joint Venture zwischen der SIGNAL IDUNA, der MSG-Systems AG, ALLIANZ X und der IBM gegründet.

Diese wegweisenden Kooperationen führten zur Entstehung der SDA SE.

Im letzten Jahr wurde der ehemalige CEO Dr. Stephan Hans gegangen, was bis heute einige weitere interims CEO's und Umstrukturierungen zur Folge hatte.

1.2 ZIELE UND PRODUKTE

Die SDA SE bietet eine digitale Service-Plattform mit einer einzigartigen serviceorientierten Architektur. Diese Plattform wird kontinuierlich von einem weltweiten Netzwerk führender Servicewissenschaftler weiterentwickelt.

Die Kernprinzipien sind Offenheit und Zusammenarbeit, die es Unternehmen ermöglichen, sich über eigene digitale Ökosysteme mit Partnern und Start-ups zu vernetzen. Alle Services basieren auf Open-Source-Software und sind darauf ausgerichtet, die Anforderungen der modernen digitalen Welt zu erfüllen.

1.3 BETRIEBSWIRTSCHAFTLICHE KENNZAHLEN

Leider können aktuelle betriebswirtschaftliche Kennzahlen aus vertraulichen Gründen nicht öffentlich zugänglich machen. Ehrlich gesagt sind diese auch persönlich gar nicht bekannt.

1.4 ORGANISATIONSSTRUKTUR

Die SDA SE ist ein Gemeinschaftsunternehmen von SIGNAL IDUNA, MSG, Allianz X, Debeka und HUK-COBURG. Im Aufsichtsrat finden sich Persönlichkeiten wie Prof. Dr. Markus Warg, Daniela Rode (Vorstand der SIGNAL IDUNA Gruppe), Carsten Middendorf (Head of Platforms, Acquisitions bei

Allianz X), Daniel Thomas (Vorstand der HUK-COBURG), Dr. Normann Pankratz (Vorstand der Debeka Versicherungsgruppe) sowie Dr. Jürgen Zehetmaier (Vorstand bei MSG-Systems).

Den heutigen Vorstand bildet Marco Ziegler.

1.5 PERSÖNLICHE AUFGABENSTELLUNG

Herr Siemer ist in seiner Position, als Development and Operations (DevOps) Engineer, für eine breite Palette von Aufgaben verantwortlich. Er entwickelt und implementiert eine Multicloud-Kubernetes-Plattform, die Amazon Web Services (AWS), Google Cloud Platform (GCP) und Microsoft Azure (Azure) umfasst. Diese Plattform wird von mehreren Teams und direkt von Kunden genutzt, um Microservices zu entwickeln und bereitzustellen.

Ziel ist es, eine effiziente Container-Orchestrierung und -Verwaltung über verschiedene Cloud-Plattformen sicherzustellen. Zusätzlich gestaltet und implementiert er eine umfassende Sicherheitsstrategie für die gesamte Infrastruktur.

Dabei werden spezielle Kubernetes-Sicherheitsmaßnahmen wie Netzwerkrichtlinien, Role-Based Access Control (RBAC) und Secrets Management eingesetzt. Er verantwortet auch das Management von Active Directory, die Autorisierung und Authentifizierung über verschiedene Flows von OAuth2. Herr Siemer arbeitet an der Entwicklung automatisierter Prozesse für Continuous Integration und Continuous Deployment und programmiert Cloud-native Software in den Sprachen Java, Python und Golang. Darüber hinaus ist er für die Erstellung von Microservice-Container-Images verantwortlich, um eine effiziente Bereitstellung und Skalierung zu ermöglichen.

Er implementiert Monitoring- und Alerting-Systeme und arbeitet an der Förderung einer Kultur der Zusammenarbeit und kontinuierlichen Verbesserung im Sinne von DevOps-Praktiken. Herr Siemer analysiert und behebt Infrastrukturprobleme und stellt die Systemverfügbarkeit und -leistung sicher. Erweitern des Cloud-Service-Angebots, die Optimierung von Cloud-Services und die Anpassung der Cloud-Infrastruktur an die Anforderungen des Unternehmens sind weitere Aufgaben.

2 PROJEKTBESCHREIBUNG

Der Titel des Projekts lautet „Automatische DNS-Zonen und Eintragsverwaltung in einer verteilten Multi-Cloud-Microservice-Architektur“ und wird im Folgenden beschrieben. Diese Arbeit soll sich auf das Kernthema der automatischen DNS-Zonen und Eintragsverwaltung in einer verteilten Multi-Cloud-Microservice-Architektur konzentrieren und nicht auf die Implementierung der Microservices selbst. Weiterhin werden verwendete Technologien welche nicht direkt mit dem Kernthema in Verbindung stehen nur oberflächlich behandelt.

2.1 IDEE

Die Projektidee entstand aus dem Bedarf der Prozesstrennung und Automatisierung zwischen den Abteilungen der Entwicklung und dem Betrieb.

2.1.1 *Ziel*

Die Abteilung des Betriebs besteht aus DevOps, welche sich um die Entwicklung und Bereitstellung von Infrastruktur kümmert. Die Abteilungen der Entwickler beschäftigen sich mit der Entwicklung von Microservices und der Bereitstellung dieser auf der Infrastruktur. Microservices bezeichnen einen Ausbau einer Programm-Architektur, bei der ein Programm in mehrere kleine Programme aufgeteilt wird, welche jeweils eine Aufgabe erfüllen und mit anderen Teilen des Programms über Application Programming Interface (API)s kommunizieren [Lim23].

Diese Infrastruktur wird auf Kundenwünsch hin auf verschiedenen Cloud-Providern bereitgestellt. Das Ziel der Arbeit ist den Entwicklern die Bereitstellung ihrer Microservices auf verschiedenen Cloud-Providern zu ermöglichen, ohne dass sie sich mit den Cloud-Providern beschäftigen müssen und ihnen eine adequate Abstraktionsebene zu bieten, welche die Komplexität der Cloud-Provider verbirgt und Prozesse außerhalb dieser Ebene automatisiert.

Fokus der Arbeit ist hierbei die automatisierung von Domain Name System (DNS). DNS ist ein System, welches Domainnamen in IP-Adressen auflöst und somit die Erreichbarkeit ermöglicht um Oberflächen oder APIs aufrufen zu können [Kle03].

Weiterhin müssen alle Ergebnisse dieser Arbeit dem neuesten Stand der Technik entsprechen und die Sicherheit der Infrastruktur gewährleisten. So wurden vorab die Anforderungen an Private Zonen und DNSSEC ermittelt, welche im Verlauf der Arbeit genauer beschrieben werden. Private Zonen

sind Zonen, welche nur innerhalb des Netzwerks erreichbar sind und somit die Sicherheit erhöhen, da IP-Adressen nicht öffentlich sichtbar sind. DNSSEC ist eine Erweiterung von DNS, welche die Authentizität und Integrität von DNS-Einträgen gewährleistet und somit die Sicherheit erhöht [Are+05].

2.1.2 *Motivation*

Für die Wahl der Abstraktionsebene gibt es viele Faktoren, welche nicht in gänze in dieser Arbeit behandelt werden können. Die Wahl fiel auf Kubernetes, da es sich um eine Open-Source Software handelt, welche Funktionen und Interfaces zur Verfügung stellt, um Microservices zu entwickeln und bereitzustellen. Kubernetes ist, nach Stand der Dinge, der Standard für Container Orchestration [Car22, p. 2]. Auf Kubernetes wird im Abschnitt 2.2.1 genauer eingegangen. Damit ein auf Kubernetes bereitgestellter Microservice von außerhalb des Kubernetes Clusters (z.B. aus dem Internet) erreichbar ist, muss ein DNS-Eintrag erstellt werden. Das sich nun stellende Problem besteht darin, dass die Microservices auf Kubernetes bereitgestellt werden, jedoch DNS-Einträge, welche für die Erreichbarkeit des Microservices sorgen, teil des Cloud-Providers sind und außerhalb von Kubernetes verwaltet werden. Dadurch entstand eine manuelle Tätigkeit, welche die Entwickler von ihrer eigentlichen Arbeit und Aufgabenbereich abhält und somit die Effizienz der Entwickler verringert. Schlussfolgernd ist für die SDA SE wünschenswert, den Zugriff der Entwickler so weit wie möglich auf lediglich Kubernetes zu beschränken und direkten Zugriff auf die Cloud-Provider zu vermeiden.

So entstand die Motivation, DNS-Einträge automatisch anhand der in Kubernetes enthaltenen Informationen zu erstellen und zu verwalten.

2.2 PLANUNG

Bei der Planung muss zwingend auf die Skalierbarkeit und Erweiterbarkeit der Lösung geachtet werden, da die SDA SE ein stark wachsendes Unternehmen ist und die Lösung auch in Zukunft noch verwendet werden soll. Außerdem muss die Lösung so gestaltet werden, dass sie auf verschiedenen Cloud-Providern funktioniert, da die SDA SE Microservices auf verschiedenen Cloud-Providern bereitstellt. Weiterhin ist das Lizenzmodell von verwendeten Lösungen zu beachten, da die SDA SE eine Open-Source first Policy verfolgt und somit Open-Source Lösungen bevorzugt werden.

2.2.1 *Umfeld*

Die Laufzeitumgebung der Microservices ist Kubernetes, welches auf den Cloud-Providern AWS, GCP und Azure als verwalteter Service gebucht wird. Kubernetes ist ein Container-Orchestrierungssystem, welches die Bereitstellung, Skalierung und Verwaltung von Containern ermöglicht. Die Microser-

vices werden in den Programmiersprachen Java, Python und Golang entwickelt und in Docker Container-Images verpackt, was die enthaltenen Programmzeilen, Bibliotheken und Abhängigkeiten kapselt und somit eine effiziente Bereitstellung, Skalierung, Unveränderbarkeit und Verbreitung auf Kubernetes ermöglicht [RBA17]. Ein Docker Container-Image ist eine Datei, welche alle Abhängigkeiten und Konfigurationen enthält, um ein Programm zu starten und ist prinzipiell einer virtuellen Maschine sehr ähnlich nur, dass der Kernel des Hosts mitverwendet wird [RBA17, p. 229].

Die implementierung mehrerer Cloud-Provider ist notwendig, da die Microservices aufgrund von Kundenanforderungen auf verschiedenen Cloud-Providern bereitgestellt werden müssen. Ein Kunde kann beispielsweise die Bereitstellung seiner Microservices auf AWS fordern, während ein anderer Kunde die Bereitstellung auf GCP oder Azure fordert. Die Wahl des Cloud-Providers ist für Kunden eine essenzielle Entscheidung, da Kunden komplexe Verträge mit den Cloud-Providern abschließen und diese nicht ohne weiteres wechseln können.

Das Bereitstellen von Kubernetes auf verschiedenen Cloud-Providern ist eine komplexe Aufgabe, da die Cloud-Provider gänzlich unterschiedliche Herangehensweisen und Anforderungen haben. Der Aufbau der Infrastruktur auf AWS unterscheidet sich grundlegend von dem Aufbau auf GCP oder Azure, was einer vollständigen Automatisierung nicht gerade in die Hände spielt [Kho20, p. 450]. Neue Initiativen wie Cluster API (CAPI) versuchen diese Problematik zu lösen, benötigen jedoch auch viel Provider-spezifischen Code, was die Komplexität dadurch nicht verringert. Außerdem wird für diesen Ansatz ein management Kubernetes Cluster benötigt, welcher sich nicht selbst bootstrappen kann und somit zumindest initial ein anderer Ansatz gewählt werden muss. Bei der SDA SE wurde sich für eine Infrastructure as Code (IaC) Herangehensweise entschieden, welche die Infrastruktur auf den Cloud-Providern mittels Terraform bereitstellt. IaC ist ein Ansatz, bei dem die Infrastruktur mittels Code deklarativ beschrieben wird und somit eine automatisierte Bereitstellung ermöglicht. Terraform ist ein Open-Source Tool, welches die Infrastruktur auf verschiedenen Cloud-Providern bereitstellen kann, wohingegen ein konkurrierendes Tool wie CloudFormation nur mit AWS funktioniert.

Trotz der Unterschiede bei den Cloud-Providern einen Kubernetes Cluster zu provisionieren, sind die Kubernetes Cluster implementationen bei den Cloud-Providern selbst beinahe identisch. Diesen Umstand will sich die SDA SE zunutze machen und Kubernetes als Abstraktionsebene zwischen den Cloud-Providern und weiterer Infrastruktur nutzen, um die Komplexität vor den Entwicklern zu verbergen. Die Entwickler sollen sich nicht mit allen Feinheiten der Infrastruktur beschäftigen müssen, sondern lediglich Kubernetes nutzen, um ihre Microservices zu entwickeln und bereitzustellen, um ihre Arbeitszeit effizienter nutzen zu können. Um dies zu erreichen und die Entwickler so gut wie möglich mit automatischen Produktionsstrassen zu unterstützen, kann Kubernetes in seiner Funktion mit weiteren

Programmen (auch Operatoren genannt) erweitert und ausgebaut werden. Ein solcher Ausbau wird häufig als Plattform bezeichnet.

So ist das Produkt der SDA SE eine Plattform, welche die Bereitstellung von Microservices auf verschiedenen Cloud-Providern mittels Kubernetes ermöglicht und erleichtert. Eine simultane Bereitstellung auf mehreren Cloud-Providern gleichzeitig ist möglich, jedoch im Standard nicht vorgesehen.

Eine Bestellung dieser Plattform enthält bis zu vier Kubernetes Clustern, welche auf dem gewählten Cloud-Provider bereitgestellt werden. Die Anzahl der Cluster ist abhängig von der Anzahl der gewünschten Umgebungen. So ist die kleinste Bestellung eine Umgebung mit zwei Clustern, einem „Tools“ und einem „Development“ Cluster. Der Tools Cluster ist für die Bereitstellung von Plattform-Tools zuständig, welche für die Entwicklung und Bereitstellung von Microservices benötigt werden. Die Umgebungscluster „Development“, „Staging“ und „Production“ sind einzig zuständig für das Bereitstellen von Firmen-Logik spezifischer Software.

2.2.2 *Plan*

Zur Projektplanung wurde ein iteratives Vorgehensmodell gewählt, da die Anforderungen und Auswirkungen des Projekts nicht vollständig bekannt sind und sich im Laufe des Projekts ändern können. Es wird außerdem angenommen, dass eine Konzeption sich später beim Test als unbefriedigend herausstellen könnte und somit ein Zurückkehren zur Konzeptionsphase notwendig sein könnte.

2.2.3 *Wirtschaftlichkeitsbetrachtung*

Da zusätzliche Infrastrukturkosten derart gering sind, dass sie vernachlässigt werden können, wird sich die Wirtschaftlichkeitsbetrachtung auf die Arbeitszeit beschränken. So ist beispielhaft das Kostenmodell für DNS auf AWS nach Aufrufen gestaffelt und diese sind unabhängig vom Projekt gleich. Weiterhin sind die Kosten von Servern im Kubernetes auch unabhängig vom Projekt gleich, da ein Operator sehr geringe Lastkosten verursacht.

Die Projektlaufzeit wurde nach 2.1 auf 100 Stunden geschätzt. Die anfallende Arbeitszeit für die manuelle Erstellung DNS Einträge wurde auf 5 Minuten pro Eintrag geschätzt. Bei ungefähr 300 ausgerollten Microservices pro Tag (aus GitHub der SDA SE ausgelesen), ist der Break-Even-Point nach 4 Tagen erreicht.

2.2.4 *Alternativen*

Als alternative für den Ansatz welcher bereits in Teilen in dieser Arbeit präsentiert wurde, gibt es die Möglichkeit Wildcard DNS Einträge zu verwenden. Diese Einträge sind Platzhalter für alle Subdomains einer Domain und können auch zur Auflösung verwendet werden. Dieser Vorgang wäre nicht

Phase	Tätigkeiten	Dauer (Stunden)
Anforderungen	Ziel definieren	3
Analyse	Vergleich Kubernetes Operatoren	5
	Funktionsweise DNSSEC	5
	Funktionsweise Public/Private	3
Konzeption	DNS Hierarchie	5
	Security Öffentlich/Private Zonen	5
	Security DNSSEC	5
	Infrastruktur Übersetzung	2
Implementierung	DNS Hierarchie	10
	Security Öffentlich/Private Zonen	15
	Security DNSSEC	10
	Infrastruktur Übersetzung	5
	Kubernetes Operator	10
Test	Testen der Erstellung	10
	Testen der Erreichbarkeit	3
Übergabe	Übergabe per IaC	0
Gesamt		100

Tabelle 2.1: Projektplan

auf die Erstellung einzelner Einträge angewiesen, sondern würde alle Subdomains einer Domain auflösen.

Dieser Ansatz erschwert jedoch die gewünschte Trennung in öffentliche und private Zonen, da sonst eine Subdomain zur Unterscheidung zwischen öffentlich und privat verwendet werden müsste. Auf die gewünschte Struktur der Zonen und Begründungen dieser wird im Abschnitt 2.3 genauer eingegangen.

2.3 UMSETZUNG

Diese Sektion befasst sich dem Verlauf der Realisierung des Projekts, der Organisation und den dabei aufgetretenen Problemen.

2.3.1 *Organisation*

Diese Arbeit wurde eigenständig und ohne direkte Anleitung durchgeführt. Die Organisation der Arbeit und der Arbeitsergebnisse wurde durch wöchentliche Meetings mit dem Auftraggeber sichergestellt.

Analysen und Konzepte wurden in Form von Dokumenten festgehalten und dem Auftraggeber zur Verfügung gestellt. Die Implementierung wur-

de in Form von Code in einem GitHub Repository festgehalten und dem Auftraggeber zur Verfügung gestellt.

2.3.2 *Verlauf*

Es wurde ein iteratives Modell gewählt, welche die Phasen Anforderungen, Analyse, Konzeption, Implementierung, Test immer wieder durchläuft. Eine Anforderung wurde definiert, der Ist-Zustand analysiert, ein Konzept erstellt um sich der Anforderung so weit wie möglich anzunähern, dieses Konzept implementiert und getestet. Daraufhin wurde die Anforderung erneut betrachtet und gegebenenfalls angepasst. Beim zufriedenstellenden Ergebnis wurde die nächste Anforderung betrachtet und der Prozess wiederholt. Mit dem Abschließen der letzten Anforderung, wurden die Ergebnisse in ihrer Gänze betrachtet, getestet und gegebenenfalls angepasst. Am Ende der Arbeit wurde die Übergabe der Ergebnisse an den Auftraggeber durchgeführt.

2.3.3 *Modifikation*

blablabla

2.4 ERGEBNISSE

blablabla

2.4.1 *Darstellung*

blablabla

2.4.2 *Analyse*

(Soll-/ Ist-Vergleich)

2.4.3 *Wirtschaftlichkeitsbetrachtung*

(Soll-/ Ist-Vergleich)

2.5 SCHLUSSFOLGERUNG

blablabla

2.5.1 *Auswirkungen*

(Soll-/ Ist-Vergleich)

2.5.2 *Modifikationen*

(Soll-/ Ist-Vergleich)

2.5.3 *Bilanz*

(Soll-/ Ist-Vergleich)

Teil II

APPENDIX

A

INTRODUCTION TO THE CLASSICTHESIS STYLE

The ClassicThesis bundle for LaTeX has two goals:

1. Provide students with an easy-to-use template for their Master's or PhD thesis. (Though it might also be used by other types of authors for reports, books, etc.)

2. Provide a classic, high-quality typographic style that is inspired by Bringhurst's "*The Elements of Typographic Style*" [Bri13].

Automatische DNS-Zonen und Eintragsverwaltung in einer verteilten Multi-Cloud-Microservice-Architektur version 0.1

The bundle is configured to run with a *full* MiKTeX or TeXLive[1] installation right away and, therefore, it uses only freely available fonts. (Minion fans can easily adjust the style to their needs.)

People interested only in the nice style and not the whole bundle can now use the style stand-alone via the file `classicthesis.sty`. This works now also with „plain" LaTeX.

As of version 3.0, `classicthesis` can also be easily used with LyX[2] thanks to Nicholas Mariette and Ivo Pletikosić. The LyX version of this manual will contain more information on the details.

This should enable anyone with a basic knowledge of LaTeX 2_ε or LyX to produce beautiful documents without too much effort. In the end, this is my overall goal: more beautiful documents, especially theses, as I am tired of seeing so many ugly ones.

The whole template and the used style is released under the GNU General Public License.

If you like the style then I would appreciate a postcard:

André Miede
Detmolder Straße 32
31737 Rinteln
Germany

The postcards I received so far are available at:

`http://postcards.miede.de`

A well-balanced line width improves the legibility of the text. That's what typography is all about, right?

So far, many theses, some books, and several other publications have been typeset successfully with it. If you are interested in some typographic details behind it, enjoy Robert Bringhurst's wonderful book.

1 See the file `LISTOFFILES` for needed packages. Furthermore, `classicthesis` works with most other distributions and, thus, with most systems LaTeX is available for.

2 `http://www.lyx.org`

IMPORTANT NOTE: Some things of this style might look unusual at first glance, many people feel so in the beginning. However, all things are intentionally designed to be as they are, especially these:

- No bold fonts are used. Italics or spaced small caps do the job quite well.
- The size of the text body is intentionally shaped like it is. It supports both legibility and allows a reasonable amount of information to be on a page. And, no: the lines are not too short.
- The tables intentionally do not use vertical or double rules. See the documentation for the `booktabs` package for a nice discussion of this topic.[3]
- And last but not least, to provide the reader with a way easier access to page numbers in the table of contents, the page numbers are right behind the titles. Yes, they are *not* neatly aligned at the right side and they are *not* connected with dots that help the eye to bridge a distance that is not necessary. If you are still not convinced: is your reader interested in the page number or does she want to sum the numbers up?

Therefore, please do not break the beauty of the style by changing these things unless you really know what you are doing! Please.

YET ANOTHER IMPORTANT NOTE: Since `classicthesis`' first release in 2006, many things have changed in the LaTeX world. Trying to keep up-to-date, `classicthesis` grew and evolved into many directions, trying to stay (some kind of) stable and be compatible with its port to LyX. However, there are still many remains from older times in the code, many dirty workarounds here and there, and several other things I am absolutely not proud of (for example my unwise combination of KOMA and `titlesec` etc.).

An outlook into the future of `classicthesis`.

Currently, I am looking into how to completely re-design and re-implement `classicthesis` making it easier to maintain and to use. As a general idea, `classicthesis.sty` should be developed and distributed separately from the template bundle itself. Excellent spin-offs such as `arsclassica` could also be integrated (with permission by their authors) as format configurations. Also, current trends of `microtype`, `fontspec`, etc. should be included as well. As I am not really into deep LaTeX programming, I will reach out to the LaTeX community for their expertise and help.

A.1 ORGANIZATION

A very important factor for successful thesis writing is the organization of the material. This template suggests a structure as the following:

You can use these margins for summaries of the text body...

3 To be found online at http://mirror.ctan.org/macros/latex/contrib/booktabs/.

- `Chapters/` is where all the „real“ content goes in separate files such as `Chapter01.tex` etc.
- `FrontBackMatter/` is where all the stuff goes that surrounds the „real“ content, such as the acknowledgments, dedication, etc.
- `gfx/` is where you put all the graphics you use in the thesis. Maybe they should be organized into subfolders depending on the chapter they are used in, if you have a lot of graphics.
- `Bibliography.bib`: the BibTEX database to organize all the references you might want to cite.
- `classicthesis.sty`: the style definition to get this awesome look and feel. Does not only work with this thesis template but also on its own (see folder `Examples`). Bonus: works with both LATEX and PDFLATEX. . . and LYX. Great tool and it's free!
- `ClassicThesis.tex`: the main file of your thesis where all gets bundled together.
- `classicthesis-config.tex`: a central place to load all nifty packages that are used.

  Make your changes and adjustments here. This means that you specify here the options you want to load `classicthesis.sty` with. You also adjust the title of your thesis, your name, and all similar information here. Refer to Anhang A.3 for more information.

  This had to change as of version 3.0 in order to enable an easy transition from the „basic“ style to LYX.

In total, this should get you started in no time.

A.2 STYLE OPTIONS

There are a couple of options for `classicthesis.sty` that allow for a bit of freedom concerning the layout:

. . . or your supervisor might use the margins for some comments of her own while reading.

- General:
  - `drafting`: prints the date and time at the bottom of each page, so you always know which version you are dealing with. Might come in handy not to give your Prof. that old draft.
- Parts and Chapters:
  - `parts`: if you use Part divisions for your document, you should choose this option. (Cannot be used together with `nochapters`.)
  - `linedheaders`: changes the look of the chapter headings a bit by adding a horizontal line above the chapter title. The chapter number will also be moved to the top of the page, above the chapter title.
- Typography:
  - `eulerchapternumbers`: use figures from Hermann Zapf's Euler math font for the chapter numbers. By default, old style figures from the Palatino font are used.
  - `beramono`: loads Bera Mono as typewriter font. (Default setting is using the standard CM typewriter font.)
  - `eulermath`: loads the awesome Euler fonts for math. Palatino is used as default font.

Options are enabled via `option=true`

- Table of Contents:
  - `tocaligned`: aligns the whole table of contents on the left side. Some people like that, some don't.
  - `dottedtoc`: sets pagenumbers flushed right in the table of contents.
  - `manychapters`: if you need more than nine chapters for your document, you might not be happy with the spacing between the chapter number and the chapter title in the Table of Contents. This option allows for additional space in this context. However, it does not look as „perfect“ if you use `\parts` for structuring your document.
- Floats:
  - `listings`: loads the `listings` package (if not already done) and configures the List of Listings accordingly.
  - `floatperchapter`: activates numbering per chapter for all floats such as figures, tables, and listings (if used).

Furthermore, pre-defined margins for different paper sizes are available, e. g., `a4paper`, `a5paper`, and `letterpaper`. These are based on your chosen option of `\documentclass`.

The best way to figure these options out is to try the different possibilities and see what you and your supervisor like best.

In order to make things easier, `classicthesis-config.tex` contains some useful commands that might help you.

A.3 CUSTOMIZATION

This section will show you some hints how to adapt `classicthesis` to your needs.

The file `classicthesis.sty` contains the core functionality of the style and in most cases will be left intact, whereas the file `classicthesis-config.tex` is used for some common user customizations.

The first customization you are about to make is to alter the document title, author name, and other thesis details. In order to do this, replace the data in the following lines of `classicthesis-config.tex`:

Modifications in `classic-thesis-config.tex`

```
% ********************************************************
% 2. Personal data and user ad-hoc commands
% ********************************************************
\newcommand{\myTitle}{A Classic Thesis Style\xspace}
\newcommand{\mySubtitle}{An Homage to...\xspace}
```

Further customization can be made in `classicthesis-config.tex` by choosing the options to `classicthesis.sty` in a line that looks like this:

```
\PassOptionsToPackage{
  drafting=true,    % print version information on the bottom of the
      pages
  tocaligned=false, % the left column of the toc will be aligned (no
      indentation)
  dottedtoc=false,  % page numbers in ToC flushed right
  parts=true,       % use part division
  eulerchapternumbers=true, % use AMS Euler for chapter font (
      otherwise Palatino)
  linedheaders=false,       % chaper headers will have line above and
      beneath
  floatperchapter=true,     % numbering per chapter for all floats (i.
      e., Figure 1.1)
  listings=true,    % load listings package and setup LoL
  subfig=true,      % setup for preloaded subfig package
  eulermath=false,  % use awesome Euler fonts for mathematical
      formulae (only with pdfLaTeX)
  beramono=true,    % toggle a nice monospaced font (w/ bold)
  minionpro=false   % setup for minion pro font; use minion pro small
      caps as well (only with pdfLaTeX)
}{classicthesis}
```

Many other customizations in `classicthesis-config.tex` are possible, but you should be careful making changes there, since some changes could cause errors.

A.4 ISSUES

This section will list some information about problems using `classicthesis` in general or using it with other packages.

Beta versions of `classicthesis` can be found at Bitbucket:

https://bitbucket.org/amiede/classicthesis/

There, you can also post serious bugs and problems you encounter.

A.5 FUTURE WORK

So far, this is a quite stable version that served a couple of people well during their thesis time. However, some things are still not as they should be. Proper documentation in the standard format is still missing. In the long run, the style should probably be published separately, with the template bundle being only an application of the style. Alas, there is no time for that at the moment... it could be a nice task for a small group of LaTeXnicians.

Please do not send me email with questions concerning LaTeX or the template, as I do not have time for an answer. But if you have comments, suggestions, or improvements for the style or the template in general, do not hesitate to write them on that postcard of yours.

A.6 BEYOND A THESIS

The layout of `classicthesis.sty` can be easily used without the framework of this template. A few examples where it was used to typeset an article, a book or a curriculum vitae can be found in the folder `Examples`. The examples have been tested with `latex` and `pdflatex` and are easy to compile. To encourage you even more, PDFs built from the sources can be found in the same folder.

A.7 LICENSE

You should have received a copy of the GNU General Public License along with this program; see the file `COPYING`. If not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

CLASSICHTHESIS AUTHORS' NOTE: There have been some discussions about the GPL's implications on using `classicthesis` for theses etc. Details can be found here:

`https://bitbucket.org/amiede/classicthesis/issues/123/`

We chose (and currently stick with) the GPL because we would not like to compete with proprietary modified versions of our own work. However, the whole template is free as free beer and free speech. We will not demand the sources for theses, books, CVs, etc. that were created using `classicthesis`.

Postcards are still highly appreciated.

B

APPENDIX TEST

Lorem ipsum at nusquam appellantur his, ut eos erant homero concludaturque. Albucius appellantur deterruisset id eam, vivendum partiendo dissentiet ei ius. Vis melius facilisis ea, sea id convenire referrentur, takimata adolescens ex duo. Ei harum argumentum per. Eam vidit exerci appetere ad, ut vel zzril intellegam interpretaris.

More dummy text.

B.1 APPENDIX SECTION TEST

Test: Tabelle B.1 (This reference should have a lowercase, small caps A if the option `floatperchapter` is activated, just as in the table itself → however, this does not work at the moment.)

LABITUR BONORUM PRI NO	QUE VISTA	HUMAN
fastidii ea ius suscipit instructior	germano titulo	demonstratea personas
quaestio philosophia	facto	demonstrated

Tabelle B.1: Autem usu id.

B.2 ANOTHER APPENDIX SECTION TEST

Equidem detraxit cu nam, vix eu delenit periculis. Eos ut vero constituto, no vidit propriae complectitur sea. Diceret nonummy in has, no qui eligendi recteque consetetur. Mel eu dictas suscipiantur, et sed placerat oporteat. At ipsum electram mei, ad aeque atomorum mea. There is also a useless Pascal listing below: Listing B.1.

Listing B.1: A floating example (`listings` manual)

```
for i:=maxint downto 0 do
begin
{ do nothing }
end;
```

LITERATUR

[Are+05] Roy Arends, Rob Austein, Matt Larson, Dan Massey und Scott Rose. *RFC 4033: DNS security introduction and requirements*. 2005.

[Bri13] Robert Bringhurst. *The Elements of Typographic Style*. Version 4.0: 20th Anniversary Edition. Point Roberts, WA, USA: Hartley & Marks Publishers, 2013.

[Car22] Carmen Carrión. „Kubernetes as a Standard Container Orchestrator - A Bibliometric Analysis“. In: *Journal of Grid Computing* 20.4 (2022), S. 42. DOI: 10.1007/s10723-022-09629-8. URL: https://doi.org/10.1007/s10723-022-09629-8.

[Kho20] Aditi Rajan Khot. „A Comparative Analysis of Public Cloud Platforms and Introduction of Multi-Cloud“. In: *International Journal of Innovative Science and Research Technology* 5.9 (2020), S. 448–454.

[Kle03] John Klensin. *Role of the domain name system (dns)*. Techn. Ber. 2003.

[Lim23] Red Hat Limited. *Was sind Microservices?* https://www.redhat.com/de/topics/microservices/what-are-microservices. [Online; accessed 04-November-2023]. 2023.

[RBA17] Babak Bashari Rad, Harrison John Bhatti und Mohammad Ahmadi. „An introduction to docker and analysis of its performance“. In: *International Journal of Computer Science and Network Security (IJCSNS)* 17.3 (2017), S. 228.