Fakultät Informatik

Hochschule Reutlingen
Alteburgstraße 150
72762 Reutlingen

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science (B.SC.)
in Kooperation mit Novatec GmbH

API Security Testing mit OWASP Zap und RAML

Mert Yücel

Studiengang:	Wirtschaftsinformatik
Prüfer/in:	Prof. Dr. Martin Schmollinger, Prof. Dr. Muster Mustermann
Betreuer/in:	Anreas Falk, Jan Horak
Beginn am:	1. September 2018
Beendet am:	15. Januar 2019

Kurzfassung

REST-APIs sind heutzutage weit verbreitet und dank ihrer Einfachheit, Skalierbarkeit und Flexibilität werden sie weitgehend als Standardprotokoll für die Web-APIs angesehen. Es scheint plausibel anzunehmen, dass die Ära der Desktop-basierten Anwendungen kontinuierlich zurückgeht und im Zuge dessen, die Benutzer von Desktop- zu Web- und weiteren mobilen Anwendungen wechseln.

Bei der Entwicklung von REST-basierten Web-Anwendungen wird ein REST-basierter Web Service benötigt, um die Funktionalitäten der Web-Anwendung richtig testen zu können, da die gängigen Penetrationstest-Werkzeuge für REST-APIs nicht direkt einsatzfähig sind, wird die Sicherheit solcher APIs jedoch immer noch zu selten überprüft und das Testen dieser Arten von Anwendungen ist eine sehr große Herausforderung. Hauptsächlich für das erstmalige Testen ist es für den Betreiber von Webanwendungen sehr unüberschaubar. Verschiedene Werkzeuge, Frameworks und Bibliotheken sind dazu da, die Testaktivität automatisieren zu können. Die Nutzer wählen diese Dienstprogramme basierend auf ihrem Kontext, ihrer Umgebung, ihrem Budget und ihrem Qualifikationsniveau. Einige Eigenschaften von REST-APIs machen es jedoch für automatisierte Web-Sicherheitsscanner schwierig, geeignete REST-API-Sicherheitstests für die Schwachstellen durchzuführen.

Diese Bachelorarbeit untersucht wie die Sicherheitstests heutzutage realisiert werden und versucht qualitativ-deskriptiv aufzudecken, ob auf solche Sicherheitstests verlässlich sind. Es werden verschiedene existierende Tools verglichen, die das Testen von RESTful APIs unterstützen. Dann wird ihre Vor- und Nachteile herausgefunden und gegeneinander abgewägt. Es wird auch Gewißheit verschaffen, wie die technische Schwachstellen(wie z.B. Konzeptionsfehlern, Programmierfehlern und Konfigurationsfehlern) von Webanwendungen dargelegt. Parallel zu den Schwachstellen werden die jeweiligen Gefahren und Angriffspunkte für eine Webanwendung anhand von ”OWASP Top 10 - Risiken” erörtert.

Im Rahmen dieser Bachelorarbeit wird ein Plugin für das Open Source Werkzeug OWASP Zap entwickelt, um Schwachstellen automatisch zu untersuchen, um ein gewisses Maß an Sicherheit gewährleisten zu können. Dieses Plugin muss so implementiert werden, so dass die RAML-Dokumentationen importiert werden können, um automatisierte Penetrationstests ausführen zu können. Es wird auch eine Möglichkeit geschaffen, dass das Plugin in OWASP Zap sowohl über die UI, aber auch über Kommandozeile ansprechbar ist, um dies auch in CI-Umgebungen einbinden zu können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

Abkürzungsverzeichnis

ER error rate. 22

FR Fehlerrate. 22

RDBMS Relational Database Management System. 22

1. Einleitung

In diesem Kapitel steht die Einleitung zu dieser Arbeit. Sie soll nur als Beispiel dienen und hat nichts mit dem Buch [WCL+05] zu tun. Nun viel Erfolg bei der Arbeit!

Bei LaTeX werden Absätze durch freie Zeilen angegeben. Da die Arbeit über ein Versionskontrollsystem versioniert wird, ist es sinnvoll, pro *Satz* eine neue Zeile im `.tex`-Dokument anzufangen. So kann einfacher ein Vergleich von Versionsständen vorgenommen werden.

Die Arbeit ist in folgender Weise gegliedert: In Kapitel 5 werden die Grundlagen dieser Arbeit beschrieben. Schließlich fasst Kapitel 8 die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

1.1. Motivation

bla bla..

1.2. Zielsetzung

bla bla..

1.3. Aufbau der Arbeit

1. Vorerst werden in Kapitel 2..
2. In einer Anforderungsanalyse in Kapitel 3..
3. Kapitel 4 vermittelt..
4. Es folgt irgendwas in Kapitel 5. In diesem Teil..
5. Kapitel 6 dokumentiert
6. Die Gesamtarbeit..

2. Grundlagen

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

2.1. Informationssicherheit

2.2. Technologien des Projektes

2.2.1. Java

2.2.2. Microservice Architekturen

2.2.3. Spring Boot

2.2.3.1. Spring MVC Komponente

2.2.3.2. Spring Rest Docs

2.2.4. RESTful Web Services

2.2.5. RAML

2.2.6. Open Source Werkzeug OWASP Zap

2.2.7. RAML-Parser für Java

2.2.8. Test-Driven mit automatisierten Tests

3. Sicherheitsrisiken von Webanwendungen

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

3.1. Schwachstellen

3.1.1. OWASP Top 10 Risiken

3.1.2. Weitere Risiken

3.1.3. Common Vulnerability Scoring System

3.1.4. Common Vulnerability Exposures (CVE)

3.2. Technische Schwachstellen

3.2.1. Programmierfehler

3.2.2. Konfigurationsfehler

3.2.3. Konzeptionsfehler

3.2.4. Fehler resultierend aus menschlichem Verhalten

4. Penetrationstest

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

4.1. Grundlegendes Konzept

4.1.1. Black-Box

4.1.2. White-Box

4.1.3. Gray-Box

4.2. Kriterien für Penetrationstests

4.3. Ablauf eines Penetrationstest

4.4. Rechtliche Aspekte

4.5. Penetrationstest-Tools

4.5.1. ABC

4.6. Automatisierte Penetrationstest-Tools

4.6.1. ABC

4.7. Vor- und Nachteile zwischen manuelle und automatisierte Penetrationstest-Tools

5. Entwicklung einer Spring-Boot-Webanwendung mit Sicherheitslücken

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

6. Entwicklung von Plug-in für das OWASP-ZAP

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

7. Was ist die Vorteile von diesem Plug-in gegenüber anderen Plug-ins?

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

8. Fazit

Hier bitte einen kurzen Durchgang durch die Arbeit.

A. Benutzerhandbuch für das OWASP-Zap RAML Plug-in

> Probleme kann man niemals mit derselben Denkweise lösen, durch die sie entstanden sind.
>
> *(Albert Einstein)*

Pro Satz eine neue Zeile. Das ist wichtig, um sauber versionieren zu können. In LaTeX werden Absätze durch eine Leerzeile getrennt.

Folglich werden neue Abstäze insbesondere *nicht* durch Doppelbackslashes erzeugt. Der letzte Satz kam in einem neuen Absatz.

A.1. File-Encoding und Unterstützung von Umlauten

Die Vorlage wurde 2010 auf UTF-8 umgestellt. Alle neueren Editoren sollten damit keine Schwierigkeiten haben.

A.2. Zitate

Referenzen werden mittels `\cite[key]` gesetzt. Beispiel: [WCL+05] oder mit Autorenangabe: Weerawarana et al. [WCL+05].

Der folgende Satz demonstriert 1. die Großschreibung von Autorennamen am Satzanfang, 2. die richtige Zitation unter Verwendung von Autorennamen und der Referenz, 3. dass die Autorennamen ein Hyperlink auf das Literaturverzeichnis sind sowie 4. dass in dem Literaturverzeichnis der Namenspräfix „van der“ von „Wil M. P. van der Aalst“ steht. Reijers, Vanderfeesten und van der Aalst [RVA16] präsentieren eine Studie über die Effektivität von Workflow-Management-Systemen.

Der folgende Satz demonstriert, dass man mittels `label` in einem Bibliopgrahie-Eintrag den Textteil des generierten Labels überschreiben kann, aber das Jahr und die Eindeutigkeit noch von biber generiert wird. Die Apache ODE Engine [ASF16] ist eine Workflow-Maschine, die BPEL-Prozesse zuverlässig ausführt.

Wörter am besten mittels `\enquote{...}` „einschließen“, dann werden die richtigen Anführungszeichen verwendet.

Beim Erstellen der Bibtex-Datei wird empfohlen darauf zu achten, dass die DOI aufgeführt wird.

Listing A.1 lstlisting in einer Listings-Umgebung, damit das Listing durch Balken abgetrennt ist

```
<listing name="second sample">
  <content>not interesting</content>
</listing>
```

A.3. Mathematische Formeln

Mathematische Formeln kann man *so* setzen. `symbols-a4.pdf` (zu finden auf http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf) enthält eine Liste der unter LaTeX direkt verfügbaren Symbole. Z. B. $\mathbb{N}$ für die Menge der natürlichen Zahlen. Für eine vollständige Dokumentation für mathematischen Formelsatz sollte die Dokumentation zu `amsmath`, ftp://ftp.ams.org/pub/tex/doc/amsmath/ gelesen werden.

Folgende Gleichung erhält keine Nummer, da `\equation*` verwendet wurde.

$$x = y$$

Die Gleichung A.1 erhält eine Nummer:

$$x = y \tag{A.1}$$

Eine ausführliche Anleitung zum Mathematikmodus von LaTeX findet sich in http://www.ctan.org/tex-archive/help/Catalogue/entries/voss-mathmode.html.

A.4. Quellcode

Listing A.1 zeigt, wie man Programmlistings einbindet. Mittels `\lstinputlisting` kann man den Inhalt direkt aus Dateien lesen.

Quellcode im `<listing />` ist auch möglich.

A.5. Abbildungen

Die Abbildung A.1 und A.2 sind für das Verständnis dieses Dokuments wichtig. Im Anhang zeigt Abbildung A.4 auf Seite 24 erneut die komplette Choreographie.

Es ist möglich, SVGs direkt beim Kompilieren in PDF umzuwandeln. Dies ist im Quellcode zu latex-tipps.tex beschrieben, allerdings auskommentiert.

A.6. Tabellen

Tabelle A.1 zeigt Ergebnisse und die Tabelle A.1 zeigt wie numerische Daten in einer Tabelle representiert werden können.

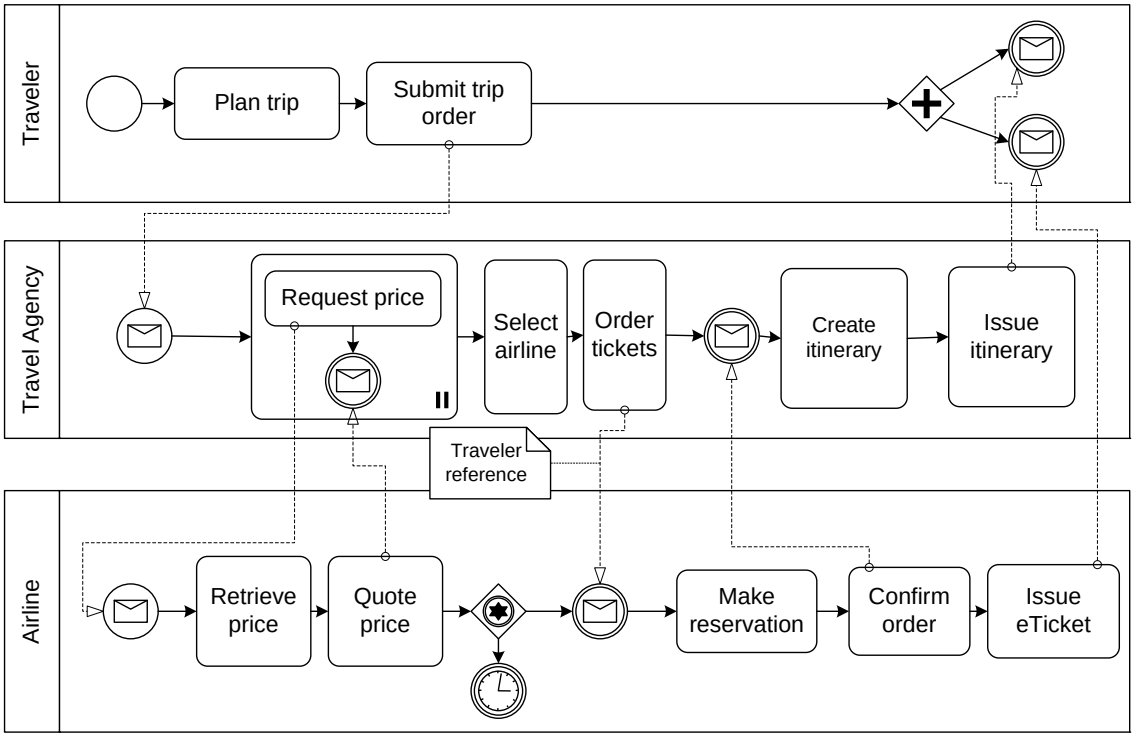


Abbildung A.1.: Beispiel-Choreographie

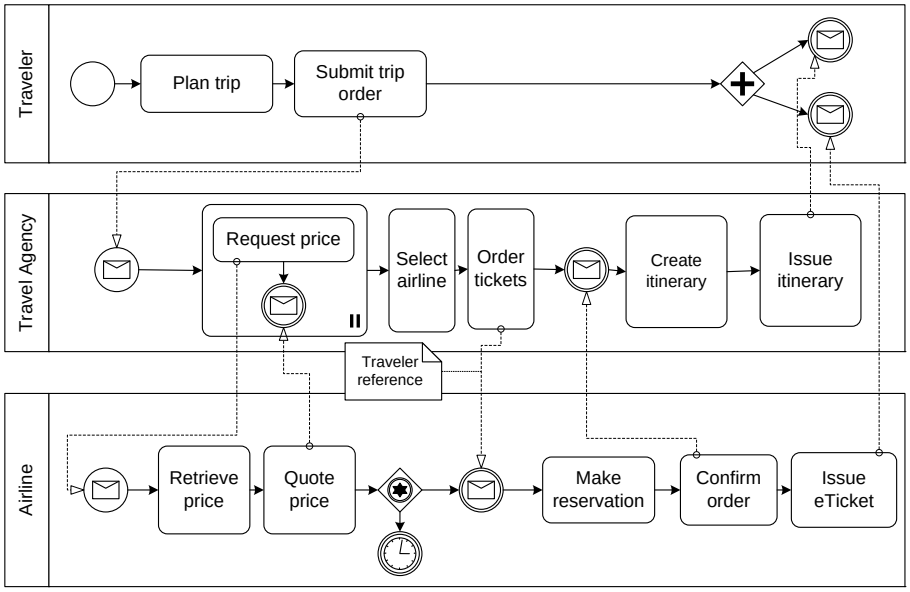
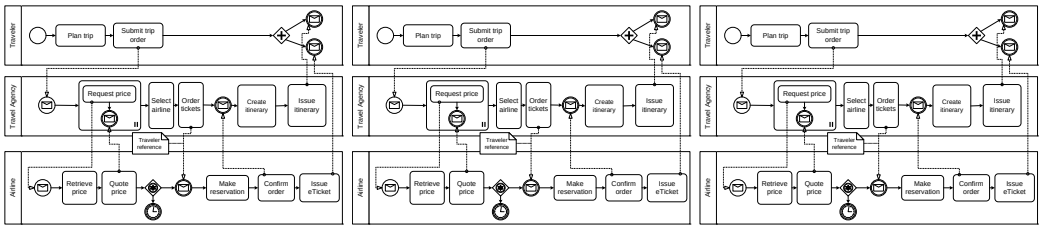


Abbildung A.2.: Die Beispiel-Choreographie. Nun etwas kleiner, damit `\textwidth` demonstriert wird. Und auch die Verwendung von alternativen Bildunterschriften für das Verzeichnis der Abbildungen. Letzteres ist allerdings nur Bedingt zu empfehlen, denn wer liest schon so viel Text unter einem Bild? Oder ist es einfach nur Stilsache?



(a) **(b)** **(c)** Subcaption if needed

Abbildung A.3.: Beispiel um 3 Abbildung nebeneinader zu stellen nur jedes einzeln referenzieren zu können. Abbildung A.3b ist die mittlere Abbildung.

zusammengefasst		**Titel**
Tabelle	wie	in
tabsatz.pdf	empfohlen	gesetzt
Beispiel	ein schönes Beispiel für die Verwendung von „multirow“	

Tabelle A.1.: Beispieltabelle – siehe http://www.ctan.org/tex-archive/info/german/tabsatz/

A.7. Pseudocode

Algorithmus A.1 zeigt einen Beispielalgorithmus.

	Parameter 1		**Parameter 2**		**Parameter 3**		**Parameter 4**	
Bedingungen	**M**	**SD**	**M**	**SD**	**M**	**SD**	**M**	**SD**
W	1,1	5,55	6,66	,01				
X	22,22	0,0	77,5	,1				
Y	333,3	,1	11,11	,05				
Z	4444,44	77,77	14,06	,3				

Tabelle A.2.: Beispieltabelle für 4 Bedingungen (W-Z) mit jeweils 4 Parameters mit (M und SD). Hinweiß: immer die selbe anzahl an Nachkommastellen angeben.

Algorithmus A.1 Sample algorithm

procedure SAMPLE(a,v_e)
  parentHandled $\leftarrow (a = \text{process}) \vee \text{visited}(a'), (a', c, a) \in \text{HR}$
    // $(a', c'a) \in \text{HR}$ denotes that a' is the parent of a
  if parentHandled $\wedge (\mathcal{L}_{in}(a) = \emptyset \vee \forall l \in \mathcal{L}_{in}(a) : \text{visited}(l))$ **then**
    $\text{visited}(a) \leftarrow \text{true}$
    $$\text{writes}_{\circ}(a, v_e) \leftarrow \begin{cases} \text{joinLinks}(a, v_e) & |\mathcal{L}_{in}(a)| > 0 \\ \text{writes}_{\circ}(p, v_e) & \exists p : (p, c, a) \in \text{HR} \\ (\emptyset, \emptyset, \emptyset, false) & \text{otherwise} \end{cases}$$
    if $a \in \mathcal{A}_{basic}$ **then**
      HANDLEBASICACTIVITY(a,v_e)
    else if $a \in \mathcal{A}_{flow}$ **then**
      HANDLEFLOW(a,v_e)
    else if $a = \text{process}$ **then** // Directly handle the contained activity
      HANDLEACTIVITY(a',v_e), $(a, \perp, a') \in \text{HR}$
      $\text{writes}_{\bullet}(a) \leftarrow \text{writes}_{\bullet}(a')$
    end if
    for all $l \in \mathcal{L}_{out}(a)$ **do**
      HANDLELINK(l,v_e)
    end for
  end if
end procedure

Und wer einen Algorithmus schreiben möchte, der über mehrere Seiten geht, der kann das nur mit folgendem **üblen** Hack tun:

Algorithmus A.2 Description

code goes here
test2

A.8. Abkürzungen

Beim ersten Durchlauf betrug die Fehlerrate (FR) 5. Beim zweiten Durchlauf war die FR 3. Die Pluralform sieht man hier: error rates (ERs). Um zu demonstrieren, wie das Abkürzungsverzeichnis bei längeren Beschreibungstexten aussieht, muss hier noch Relational Database Management Systems (RDBMS) erwähnt werden.

Mit `\gls{...}` können Abkürzungen eingebaut werden, beim ersten Aufrufen wird die lange Form eingesetzt. Beim wiederholten Verwenden von `\gls{...}` wird automatisch die kurz Form angezeigt. Außerdem wird die Abkürzung automatisch in die Abkürzungsliste eingefügt. Mit `\glspl{...}` wird die Pluralform verwendet. Möchte man, dass bei der ersten Verwendung direkt die Kurzform erscheint, so kann man mit `\glsunset{...}` eine Abkürzung als bereits verwendet markieren. Das Gegenteil erreicht man mit `\glsreset{...}`.

Definiert werden Abkürzungen in der Datei *content ausarbeitung.tex* mithilfe von `\newacronym{...}{...}{...}`.

Mehr Infos unter: http://tug.ctan.org/macros/latex/contrib/glossaries/glossariesbegin.pdf

A.9. Verweise

Für weit entfernte Abschnitte ist „varioref“ zu empfehlen: „Siehe Anhang A.3 auf Seite 18“. Das Kommando `\vref` funktioniert ähnlich wie `\cref` mit dem Unterschied, dass zusätzlich ein Verweis auf die Seite hinzugefügt wird. `vref`: „Anhang A.1 auf Seite 17“, `cref`: „Anhang A.1“, `ref`: „A.1“.

Falls „varioref“ Schwierigkeiten macht, dann kann man stattdessen „cref“ verwenden. Dies erzeugt auch das Wort „Abschnitt“ automatisch: Anhang A.3. Das geht auch für Abbildungen usw. Im Englischen bitte `\Cref{...}` (mit großem „C“ am Anfang) verwenden.

A.10. Definitionen

Definition A.10.1 (Title)
Definition Text

Definition A.10.1 zeigt …

A.11. Fußnoten

Fußnoten können mit dem Befehl `\footnote{...}` gesetzt werden[1]. Mehrfache Verwendung von Fußnoten ist möglich indem man zu erst ein Label in der Fußnote setzt `\footnote{\label{...}...}` und anschließend mittels `\cref{...}` die Fußnote erneut verwendet[1].

A.12. Verschiedenes

Ziffern (123 654 789) werden schön gesetzt. Entweder in einer Linie oder als Minuskel-Ziffern. Letzteres erreicht man durch den Parameter `osf` bei dem Paket `libertine` bzw. `mathpazo` in `fonts.tex`.

Kapitälchen werden schön gesperrt...

I. Man kann auch die Nummerierung dank paralist kompakt halten
II. und auf eine andere Nummerierung umstellen

A.13. Weitere Illustrationen

Abbildungen A.4 und A.5 zeigen zwei Choreographien, die den Sachverhalt weiter erläutern sollen. Die zweite Abbildung ist um 90 Grad gedreht, um das Paket `pdflscape` zu demonstrieren.

[1] Diese Fußnote ist ein Beispiel.

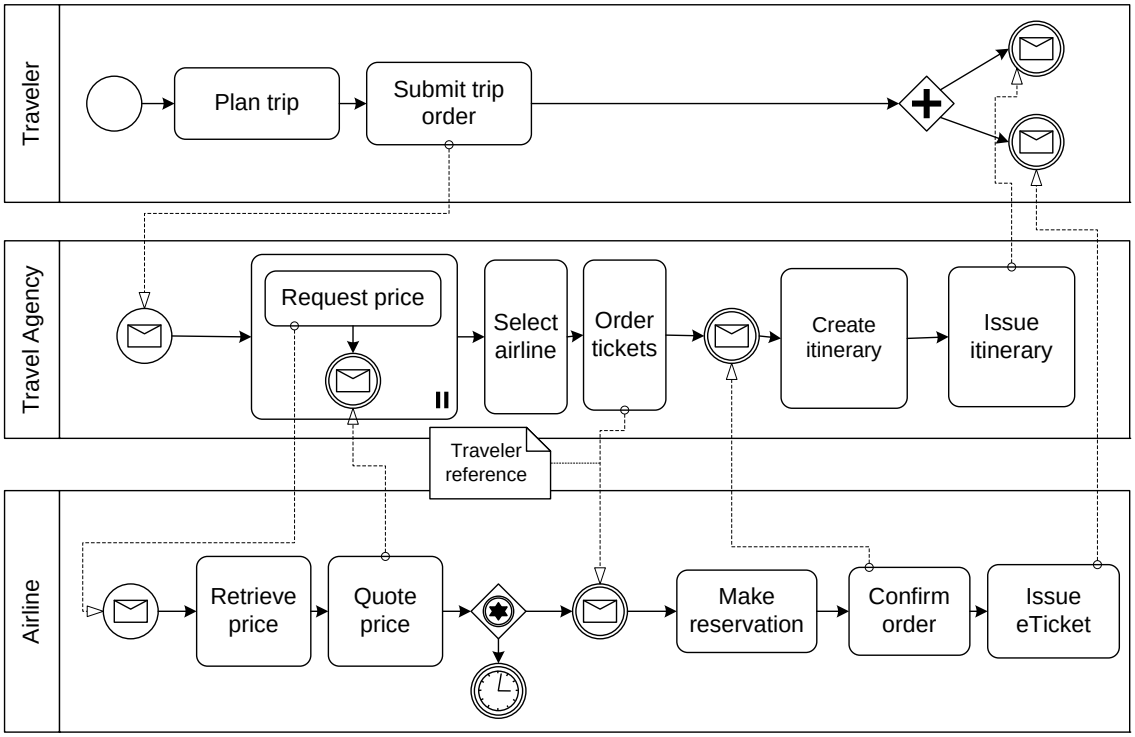


Abbildung A.4.: Beispiel-Choreographie I

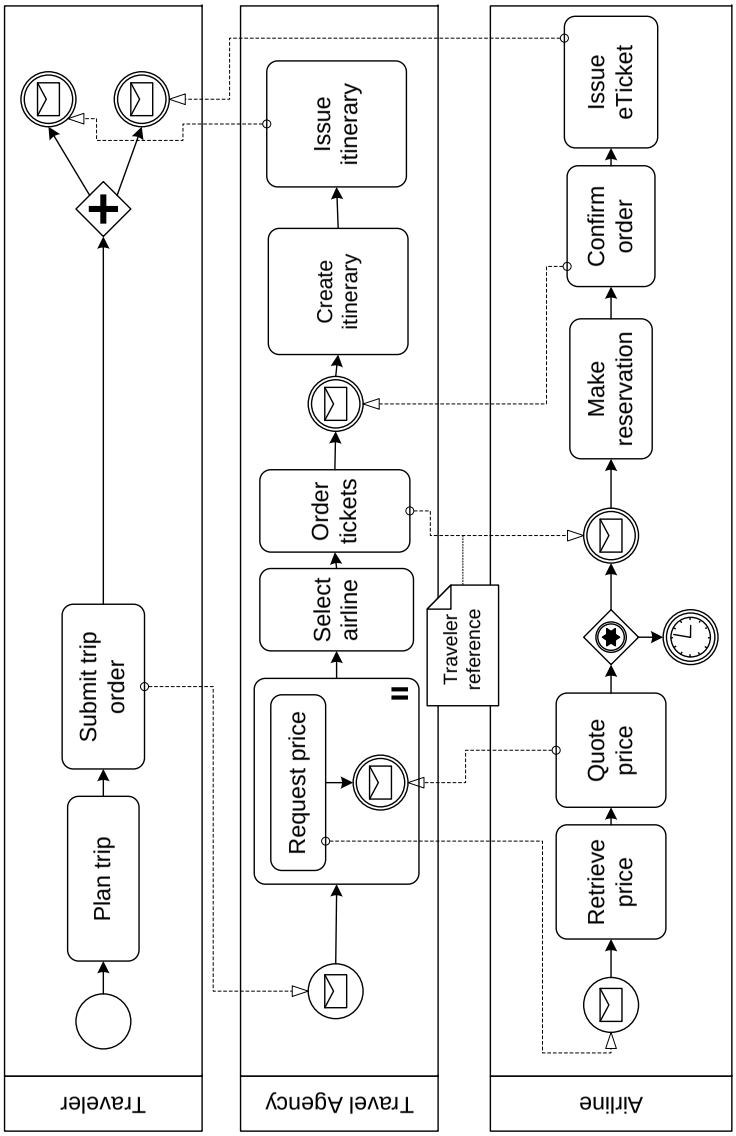


Abbildung A.5.: Beispiel-Choreographie II

A.14. Plots with pgfplots

Pgfplot ist ein Paket um Graphen zu plotten ohne den Umweg über gnuplot oder matplotlib zu gehen.

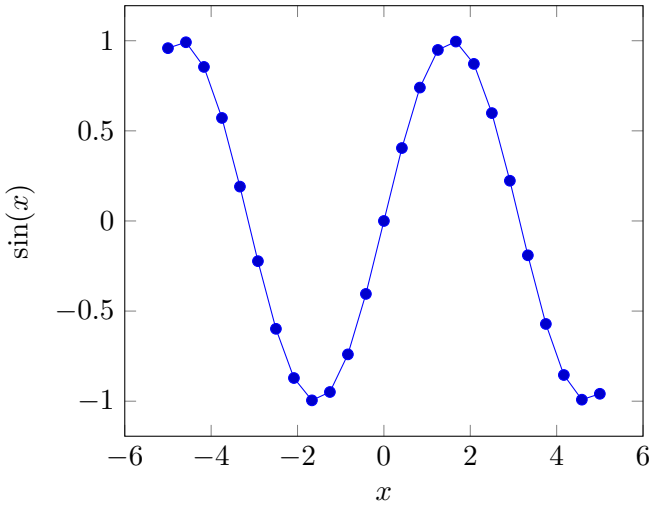


Abbildung A.6.: $\sin(x)$ mit pgfplots.

A.15. Figures with tikz

TikZ ist ein Paket um Zeichnungen mittels Programmierung zu erstellen. Dieses Paket eignet sich um Gitter zu erstellen oder andere regelmäßige Strukturen zu erstellen.

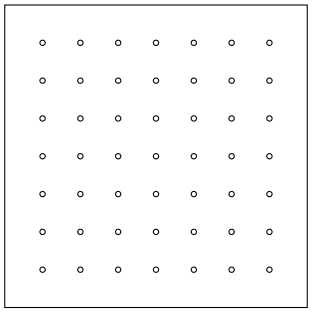


Abbildung A.7.: Eine tikz-Graphik.

Literaturverzeichnis

[ASF16] The Apache Software Foundation. *Apache ODE™ – The Orchestration Director Engine*. 2016. URL: http://ode.apache.org (zitiert auf S. 17).

[RVA16] H. Reijers, I. Vanderfeesten, W. van der Aalst. „The effectiveness of workflow management systems: A longitudinal study“. In: *International Journal of Information Management* 36.1 (Feb. 2016), S. 126–141. DOI: 10.1016/j.ijinfomgt.2015.08.003 (zitiert auf S. 17).

[WCL+05] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, D. F. Ferguson. *Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall PTR, 2005. ISBN: 0131488740. DOI: 10.1.1/jpb001 (zitiert auf S. 1, 17).

Alle URLs wurden zuletzt am 10. 01. 2019 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift