Westsächsische Hochschule Zwickau
University of Applied Sciences

flatexDEGIRO

Konzeption und Implementierung eines Verfahrens zur Abwehr von Angriffen bei der Kunden-Neuanmeldung bei einem Online-Broker

Bachelorarbeit

an der Fakultät Physikalische Technik- Informatik
(Westsächsische Hochschule Zwickau)

Betreuer an der Hochschule:
Prof. Dr. Laue, Ralf
Westsächsische Hochschule Zwickau
Kornmarkt 1
08056 Zwickau
ralf.laue@fh-zwickau.de

Betreuer an der Hochschule:
Prof. Dr. Geweniger, Tina
Westsächsische Hochschule Zwickau
Kornmarkt 1
08056 Zwickau
tina.geweniger@fh-zwickau.de

Betreuer beim Unternehmen:
Kuchs, Thomas
FlatexDEGIRO AG
Pölbitzer Str. 9
08058 Zwickau
thomas.kuchs@flatexdegiro.com

vorgelegt von:
Nurmanbetov, Bekbolot
Arndtstr. 11
08058 Zwickau
bekbolot.nurmanbetov.kx8@fh-zwickau.de

Abgabetermin: 3. April 2023

Selbstständigkeitserklärung gem. § 14 Absatz 5 BPO

Hiermit versichere ich, *Bekbolot Nurmanbetov*, dass die vorliegende Bachelorarbeit mit dem Titel
„Konzeption und Implementierung eines Verfahrens zur Abwehr von Angriffen bei der Kunden-Neuanmeldung bei einem Online-Broker“
selbstständig und nur unter Verwendung der von mir angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Zwickau, den 3. April 2023
Ort, Datum

Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Codeverzeichnis

Abkürzungsverzeichnis und Glossar

Akzeptanztests wird in [24] definiert als der Prozess der Evaluierung eines Systems oder seiner Komponente(n) im Hinblick auf die geschäftlichen und technischen Anforderungen, die dem Entwurf und der Entwicklung zugrunde liegen [Übers. durch den Autor]. 28–30

API Application Programming Interface. iv, v, vii, 18, 32

API-Schlüssel wird in [9] definiert als eine eindeutige Identifizierung, die zur Authentifizierung von Anfragen an eine API verwendet wird. API-Schlüssel werden üblicherweise von Entwicklern und Organisationen verwendet, um den Zugriff auf ihre APIs zu kontrollieren und die Nutzung und Abrechnung zu überwachen [Übers. durch den Autor]. 24

Basic-Authentifizierung wird in [20] definiert als ein einfacher Authentifizierungsmechanismus, der zum Schutz von Webressourcen verwendet wird, indem Benutzer vor dem Zugriff auf diese Ressourcen Anmeldedaten wie einen Benutzernamen und ein Kennwort angeben müssen [Übers. durch den Autor]. 24

Build wird in [14] definiert als eine Softwareanwendung, die den Prozess der Erstellung, des Testens und der Bereitstellung von Software automatisiert [Übers. durch den Autor]. 25, 26

CC Country Code for geographic areas wird in [36] definiert als die Kombination aus einer, zwei oder drei Ziffern, die ein bestimmtes Land, Länder in einem integrierten Nummerierungsplan oder ein bestimmtes geografisches Gebiet identifiziert [Übers. durch den Autor]. v, 12–18, 20, 24, 31

CCI Client Check-In ist ein Dienst von flatexDEGIRO, der im Onboarding-Prozess des Kunden verwendet wird. 1–5, 7, 8, 19, 24, 32

CI/CD wird in [11] definiert als eine Softwareentwicklungspraxis, die die häufige Integration von Codeänderungen in ein gemeinsames Code-Repository und die automatische Erstellung, Prüfung und Bereitstellung von Codeänderungen beinhaltet [Übers. durch den Autor]. 25, 26

Container-Orchestrierung wird in [15] definiert als die Verwaltung von Containern, in der Regel in einem Cluster von Hosts, um die Bereitstellung und Skalierung von Anwendungen zu ermöglichen. Eine Container-Orchestrierungs-plattform ist ein System, das die Tools und die Infrastruktur bereitstellt, die für die Verwaltung der Bereitstellung und des Betriebs von Containern in einem Cluster erforderlich sind [Übers. durch den Autor]. 25, 26

Counter oder Zähler wird in [25] definiert als eine kumulative Metrik, die einen einzelnen monoton ansteigenden Zähler darstellt, dessen Wert nur bei einem Neustart ansteigen oder auf Null zurückgesetzt werden kann [Übers. durch den Autor]. 31

CPU Central Processing Unit. 31

Cucumber wird in [4] definiert als ein quelloffenes Werkzeug, das die verhaltensgesteuerte Entwicklung (BDD) mit einer in Gherkin-Syntax geschriebenen Klartextbeschreibung des Softwareverhaltens unterstützt. Cucumber ermöglicht das automatisierte Testen von Web-, Mobil- und API-Anwendungen [Übers. durch den Autor]. 28, 30

Docker-Image wird in [5] definiert als ein leichtgewichtiges, eigenständiges, ausführbares Paket, das alles enthält, was zur Ausführung einer Software benötigt wird, einschließlich des Anwendungscodes, der Bibliotheken, der Abhängigkeiten und der Laufzeit [Übers. durch den Autor]. 26, 29

E.164 wird in [36] definiert als eine Standardisierung mit der Bezeichnung Internationaler Nummerierungsplan für den öffentlichen Fernmeldedienst [Übers. durch den Autor]. 10, 11

E.164-Nummer wird in [35] definiert als eine Folge von Dezimalziffern, die die drei in [ITU-T E.164] genannten Merkmale Struktur, Nummernlänge und Eindeutigkeit erfüllt. Die Nummer enthält die Informationen, die erforderlich sind, um den Anruf an den Endnutzer oder an einen Punkt weiterzuleiten, an dem ein Dienst bereitgestellt wird [Übers. durch den Autor]. vi, 10, 12

E.164-Nummernplan wird in [35] definiert als eine Art von Nummerierungsplan, der das Format und die Struktur der in diesem Plan verwendeten Nummern festlegt. Der Nummerierungsplan besteht in der Regel aus Dezimalziffern, die in Gruppen unterteilt sind, um bestimmte Elemente zu identifizieren, die für die Identifizierung, Leitweglenkung und Gebührenerhebung verwendet werden, z. B. um Länder, nationale Zielorte und Teilnehmer zu identifizieren [Übers. durch den Autor]. vi, 11, 12

EBC wird in [17] definiert als ein Software-Architekturmuster, das die Kerngeschäftslogik einer Anwendung (die Entitätsschicht) von der Schnittstellen- und Steuerungslogik (die Steuerungsschicht) und den externen Abhängigkeiten (die Begrenzungsschicht) trennt [Übers. durch den Autor]. 27

flatexDEGIRO ist der führende europäische Online Broker mit über 2 Millionen Kunden in 18 Ländern [7]. 1, 5–8, 17, 19, 23

Framework bzw. Software-Framework wird in [29] definiert als eine universelle, wiederverwendbare Softwareumgebung, die als Teil einer größeren Softwareplattform bestimmte Funktionen bereitstellt, um die Entwicklung von Softwareanwendungen, -produkten und -lösungen zu erleichtern. Ein Framework umfasst im Allgemeinen eine Sammlung von Bibliotheken und Modulen, Entwicklungswerkzeugen und Laufzeitumgebungen, die es Entwicklern ermöglichen, Softwareanwendungen und -systeme auf modulare, skalierbare und effiziente Weise zu erstellen [Übers. durch den Autor]. 22, 28, 30

Gauge wird in [25] definiert als eine Metrik, die einen einzelnen numerischen Wert darstellt, der beliebig nach oben und unten variieren kann [Übers. durch den Autor]. 31

GIC Group Identification Code wird in [36] definiert als ein einstelliger Identifikationscode, der einer Gruppe von Ländern zugeordnet ist [Übers. durch den Autor]. v

GoC Group of Countries wird in [36] definiert als mehrere von der ITU oder den Vereinten Nationen anerkannte Länder teilen sich denselben CC+GIC [Übers. durch den Autor]. 11

Graylog Open wird in [10] definiert als ein Dienst, der die wichtigsten zentralen Log-Management-Funktionen bietet, die zum Sammeln, Verbessern, Speichern und Analysieren von Daten benötigt werden [Übers. durch den Autor]. 32

Helm wird in [3] definiert als Paketmanager für Kubernetes, der eine Möglichkeit bietet, komplexe Kubernetes-Anwendungen zu definieren, zu installieren und zu aktualisieren [Übers. durch den Autor]. 26

IKT Informations- und Kommunikationstechnologie. 10, 11

Integrationstests wird in [28] definiert als eine Art von Softwaretest, der die Interaktion und Kommunikation zwischen verschiedenen Komponenten oder Systemen überprüft [Übers. durch den Autor]. 29

ITU International Telecommunication Union. 10, 11, 20

Jakarta-Bean-Validierung wird in [38] definiert als eine Java-Spezifikation, die es Entwicklern ermöglicht, Objektmodellen mithilfe von Annotationen Einschränkungen hinzuzufügen. Die Spezifikation bietet außerdem APIs für die Validierung von Objekten und Objektgraphen sowie von Parametern und Rückgabewerten von Methoden und Konstruktoren [Übers. durch den Autor]. 14

JSON oder JavaScript Object Notation wird in [26] definiert als ein leichtgewichtiges, textbasiertes, sprachunabhängiges Format für den Datenaustausch [Übers. durch den Autor]. 18

JUnit5 wird in [1] definiert als ein quelloffenes Testframework für Java, das für das Schreiben und Ausführen von Tests für Java 8 und höher konzipiert ist [Übers. durch den Autor]. vii, 28

Kubernetes-Cluster wird in [16] definiert als eine Gruppe von Computern, auf denen die Container-Orchestrierungsplattform Kubernetes läuft und die für die Bereitstellung und Verwaltung von containerisierten Anwendungen verwendet werden [Übers. durch den Autor]. 26

Link Mobility ist ein Anbieter von Kommunikationsplattformen als Service, der Kommunikationsprodukte und -dienste für Unternehmen bereitstellt [18] [Übers. durch den Autor]. 1

Micrometer-Tracing wird in [19] definiert als eine quelloffene Bibliothek, die eine einfache und konsistente Möglichkeit zum Sammeln und Exportieren von Tracing-Daten aus Java-Anwendungen bietet. Die Bibliothek baut auf der Mic-

rometer-Bibliothek auf, einer beliebten Java-Überwachungsbibliothek zur Erfassung von Metrikdaten [Übers. durch den Autor]. 32

Myra ist ein ansässiges Cybersicherheitsunternehmen in Deutschland, das eine Reihe von Sicherheitslösungen für Unternehmen und Organisationen anbietet. Zu den Angeboten von Myra gehören unter anderem Managed Security Services, Netzwerksicherheit, Cloud-Sicherheit und Datenschutz [21]. 6

N(S)N National (Significant) Number wird in [35] definiert als der Teil der internationalen E.164-Nummer, der auf die Ländervorwahl für geografische Gebiete folgt und in nationalen Nummerierungsplänen (NNP) festgelegt ist. Die nationale (signifikante) Nummer besteht aus der nationalen Zielvorwahl (NDC), falls vorhanden, und der Teilnehmernummer (SN). In einigen Fällen kann der NDC fehlen oder Teil der SN sein, und in diesem Fall fallen die N(S)N und die SN zusammen. Die Funktion und das Format der N(S)N werden auf nationaler Ebene festgelegt [Übers. durch den Autor]. vi, 12, 13, 18, 20

NDC National Destination Code wird in [35] definiert als ein nationales fakultatives Kennziffernfeld innerhalb des internationalen öffentlichen Telekommunikationsnummernplans (im Folgenden als 'internationaler E.164-Nummernplan' bezeichnet), das - in Verbindung mit der Teilnehmernummer (SN) - die nationale (signifikante) Nummer (N(S)N) der internationalen E.164-Nummer für geografische Gebiete bildet [Übers. durch den Autor]. vi, 12–14, 18, 20, 24

NNP Nationaler Nummernplan wird in [36] definiert als die nationale Umsetzung des internationalen E.164-Nummernplans [Übers. durch den Autor]. vi, 9

Nummernplan wird in [36] definiert als ein Plan, der das Format und die Struktur der in diesem Plan verwendeten Nummern festlegt. Sie besteht in der Regel aus Dezimalziffern, die in Gruppen unterteilt sind, um bestimmte Elemente für die Identifizierung, Leitweglenkung und Gebührenerhebung zu kennzeichnen, z. B. zur Identifizierung von Ländern, nationalen Zielvorwahlen und Teilnehmern [Übers. durch den Autor]. 10

Nummernübertragbarkeit wird in [12] definiert als die Möglichkeit, eine Telefonnummer von einem Telekommunikationsanbieter zu einem anderen zu übertragen oder zu portieren, wobei die gleiche Telefonnummer beibehalten wird. Auf diese Weise können Kunden den Anbieter wechseln, ohne ihre Telefonnummer ändern zu müssen, was für Freunde, Familie und Geschäftskontakte lästig sein und Verwirrung stiften kann [Übers. durch den Autor]. 35

ORM objekt-relationales Mapping wird in [2] definiert als eine Programmiertechnik zur Konvertierung von Daten zwischen Typsystemen unter Verwendung objektorientierter Programmiersprachen [Übers. durch den Autor]. 25, 26

OTLP Exporter oder OpenTelemetry Protocol Exporter wird in [22] definiert als eine quelloffene Bibliothek für den Export von Tracing- und Metrikdaten aus Anwendungen in ein Backend-System [Übers. durch den Autor]. 32

Plattform-Services ist eine Gruppe von Diensten von flatexDEGIRO, die eine eigene Datenbank haben. 8

REST API wird in [6] definiert als das Akronym für REpresentational State Transfer. Es ist ein Architekturstil für verteilte Hypermedia-Systeme [Übers. durch den Autor]. 2, 23, 24

SDK Software Development Kit. 18, 32, 35

SMS Short Message/Messaging Service. 1–4, 6, 7, 11, 13–18, 35

SMS-Gateway ist ein Dienst von flatexDEGIRO, der im Onboarding-Prozess des Kunden verwendet wird. 1–5, 7, 8, 11, 12, 23–25, 28, 32

SN oder Subscriber number wird in [35] definiert als der Teil der E.164-Nummer, der einen Abonnenten in einem Netz oder Nummerierungsbereich identifiziert [Übers. durch den Autor]. vi, 12, 13

Testcontainers wird in [31] definiert als eine quelloffene Java-Bibliothek, die die Integration von Docker-Containern in JUnit5-Tests ermöglicht. TestContainers bietet eine einfache Möglichkeit zur Verwaltung des Lebenszyklus von Docker-Containern, die für Tests benötigt werden, und unterstützt mehrere gängige Datenbanken, Nachrichtensysteme und andere Dienste [Übers. durch den Autor]. 29

Tracing wird in [23] definiert als eine Technik, die in verteilten Systemen verwendet wird, um den Fluss von Anfragen zu verfolgen, während sie sich durch verschiedene Dienste und Systeme ausbreiten. Beim Tracing werden Anwendungen instrumentiert, um Zeit- und Metadateninformationen zu sammeln, die dann zur Erstellung eines Traces des gesamten Anfragepfads verwendet werden [Übers. durch den Autor]. 30, 32, 33

Twilio Lookup API wird in [33] definiert als API, die es ermöglicht, Informationen über eine Telefonnummer abzufragen, so dass die Dienste eine vertrauenswürdige Interaktion mit dem Benutzer herstellen können [Übers. durch den Autor]. 20

VPN oder Virtual Private Network wird in [30] definiert als eine sichere und private Netzwerkverbindung, die es Benutzern ermöglicht, sich sicher und privat mit dem Internet zu verbinden. Das VPN verschlüsselt die Daten und schafft einen sicheren und privaten Tunnel zwischen dem Gerät des Nutzers und dem VPN-Server [Übers. durch den Autor]. 35

Zusammenfassung

Diese Studie befasst sich mit der Konzeption und Implementierung einer neuen Anwendung zum Schutz vor Angriffen bei der Anmeldung von Neukunden in einem Online-Brokerage-Plattformsystem in Europa. Die Integration sollte einfach sein, so dass das System wie bisher funktioniert, aber zusammen mit der neuen Struktur. Ziel ist es, die Kosten des Unternehmens und zu senken. Um dieses Problem zu lösen, hat der Autor ein Betrugserkennungssystem entwickelt und Tests durchgeführt. Das Erkennungsmuster hat sich als wirksam erwiesen, um betrügerisches Verhalten aufzudecken. Das Papier enthält eine schrittweise Beschreibung der Umsetzung.

Kapitel 1

Einleitung

1.1 Motivation

FlatexDEGIRO[1] ist ein Online-Broker, der seinen Kunden Finanzdienstleistungen anbietet. Das Unternehmen bietet Zugang zu verschiedenen Finanzprodukten und -Dienstleistungen, darunter Aktienhandel, ETFs, Investitionsfonds, Anleihen und andere Investitionsprodukte.

Bei flatexDEGIRO gab es ein erhebliches Sicherheitsproblem. Bei dem Problem handelt es sich um eine Sicherheitslücke in flatexDEGIRO, bei der Angreifer Bots einsetzen, um mehrere Angriffe zu starten. Diese Angriffe zielen darauf ab, die Kosten des Unternehmens durch den Versand einer großen Anzahl von SMS zu erhöhen. Für den Versand von Nachrichten nutzt flatexDEGIRO einen externen Anbieter. Die Angriffe beeinträchtigen das normale Funktionieren des Unternehmens und verursacht finanzielle Verluste. Das Unternehmen muss unverzüglich Maßnahmen ergreifen, um weitere Angriffe zu verhindern und seine Systeme vor künftigen Bedrohungen zu schützen. Der Exekutivdirektor der Zwickauer Abteilung des Unternehmens bot an, ein Projekt zur Lösung dieses Problems umzusetzen. Die Hauptgründe für die Wahl dieses Themas sind die Möglichkeit, das Projekt von Grund auf zu implementieren, und die Lösung dieses Problems dem Unternehmen in vielerlei Hinsicht helfen würde, z. B. in Bezug auf Finanzen, Sicherheit und Kundenorientierung.

1.2 Problemstellung

Derzeit betreibt flatexDEGIRO einen Dienst zum Versand von SMS über einen externen Anbieter namens Link Mobility [2]. Jede an Link Mobility gesendete Nachricht wird mit der monatlichen Rechnung von Link Mobility verrechnet, es entstehen also laufende Kosten für die Einrichtung und Nutzung des SMS-Dienstes. Ein Dienst, der das SMS-Gateway exzessiv nutzt, ist das CCI in ihrem Onboarding-Prozess. Eines der ersten Dinge, die ein neuer interessierter Kunde zu Beginn des Onboarding-Prozesses tun muss, ist die Eingabe seiner Handynummer, um sich zu verifizieren. Da das CCI öffentlich im Internet verfügbar ist, ist es ein beliebtes Ziel für Angreifer. Ein regelmäßiges Szenario für Angriffe ist es, die SMS-Kosten für Unternehmen zu erhöhen, indem die Formeln des Frontends für die Eingabe von zufälligen Handynummern verwendet werden, auch für Länder wie Ägypten oder den Iran, in denen

[1] flatexDEGIRO `https://flatexdegiro.com/de`

[2] LINK Mobility GmbH `https://www.linkmobility.de/`

flatex und DEGIRO gerade nicht aktiv sind. Das CCI prüft die Vorwahlen nicht und kann dies auch nicht tun, da die Kriterien für die Eröffnung eines flatex- oder degiro-Kontos nicht an die Handynummer gebunden sind. In der Vergangenheit gab es mehrere Angriffe mit Tausenden von verschiedenen Telefonnummern, so dass die monatliche SMS-Rechnung aufgrund dieser Angriffe leider anstieg.

1.3 Zielsetzung

Das Ziel des neuen Dienstes SMS-Gatekeeper ist es, das SMS-Gateway zu schützen. Ein weiteres Ziel ist es, die Sicherheit, Skalierbarkeit und Zuverlässigkeit der bearbeiteten Anfragen sowie die Flexibilität und Nachvollziehbarkeit der Verwaltung und Verfolgung der Anfrageflüsse zu gewährleisten. Der SMS-Gatekeeper entwickelt einen eigenen Algorithmus auf der Grundlage einer Konfiguration und filtert die Nachrichten anhand von Akzeptanzkriterien. Andere Dienste, die derzeit direkt mit dem SMS-Gateway verbunden sind, werden sich in Zukunft stattdessen mit dem SMS-Gatekeeper verbinden, müssen aber ihre Implementierung nicht ändern, da das SMS-Gatekeeper genau dieselbe REST API zur Verfügung stellen wird.

Kapitel 2

Vorgehensweise

2.1 Ist-Zustand des Projekts

Dieses Kapitel zielt darauf ab, einen Überblick über den aktuellen Stand des Projekts zu geben. Zunächst wird die derzeitige Architektur untersucht, gefolgt von einer Untersuchung der Situation bei Angriffen und den damit verbundenen Kosten für das Unternehmen. Darüber hinaus werden in diesem Kapitel die bereits umgesetzten Maßnahmen zur Abschwächung dieser Risiken sowie die laufenden Bemühungen zur Verbesserung der Sicherheit des Systems erörtert.

Ein solches Kapitel ist wichtig, weil es entscheidende Informationen liefert, die für die Bewertung des Projekterfolgs, die Ermittlung von Verbesserungsmöglichkeiten und das Verständnis des Gesamtzustands erforderlich sind. Nachfolgend wird die aktuelle Situation im Detail beschrieben.

Es ist bereits bekannt, dass es potenziell gefährliche Nutzer des SMS-Gateways gibt. Da das CCI und alle damit arbeitenden Dienste nicht geändert werden können, muss ein neuer Dienst zum Schutz des SMS-Gateways entwickelt werden.

Bei der Entwicklung des SMS-Gateways stand nicht die Sicherheit im Vordergrund. Stattdessen sollte das SMS-Gateway in erster Linie als funktionaler Vermittler zwischen dem CCI und dem externen SMS-Provider dienen. Dadurch ist das SMS-Gateway anfällig für böswillige Angriffe, da es nicht über die robusten Sicherheitsmaßnahmen verfügt, die normalerweise in einem eigenständigen System implementiert werden.

Aufgrund der fehlenden Sicherheitsmaßnahmen stellt der derzeitige SMS-Gateway Dienst ein erhebliches Risiko für das Unternehmen dar. Angreifer nutzen die Schwachstellen des SMS-Gateways für Cyberangriffe aus.

Trotz der Sicherheitsrisiken, die mit dem derzeitigen SMS-Gateway verbunden sind, hat dieser Dienst sich bei der Erfüllung der Anforderungen des Unternehmens als praktisch und effektiv erwiesen. Die Fähigkeit zur nahtlosen Integration mit dem CCI und externen SMS-Providern hat effiziente und optimierte SMS-Kommunikation möglich gemacht. Die Zweckmäßigkeit des Systems sollte jedoch nicht auf Kosten der Sicherheit gehen, und das Unternehmen sollte auf die Einführung geeigneter Sicherheitsmaßnahmen hinarbeiten, um die vom SMS-Gateway ausgehenden Risiken zu mindern.

Die vorgeschlagene Sicherheitsmaßnahme ist die Einführung eines neuen Dienstes namens SMS-Gatekeeper. Der SMS-Gatekeeper wird in den nächsten Kapiteln beschrieben.

2.1.1 Grundlegende Architektur

Derzeit gibt es mehrere Dienste, die mit SMS-Gateway zusammenarbeiten. Alle Dienste außer CCI sind interne Dienste des Unternehmens und für die Öffentlichkeit nicht über das Internet zugänglich. Das CCI hingegen ist von außen zugänglich, und Angriffe erfolgen über das CCI. Daher wird in der Grundarchitektur (siehe Abbildung 1) nur das CCI dargestellt. Auf dem Bild ist CCI in zwei Teile unterteilt, nämlich CCI-Frontend und CCI-Backend. In anderen Kapiteln werden beide jedoch als ein einziges CCI betrachtet.

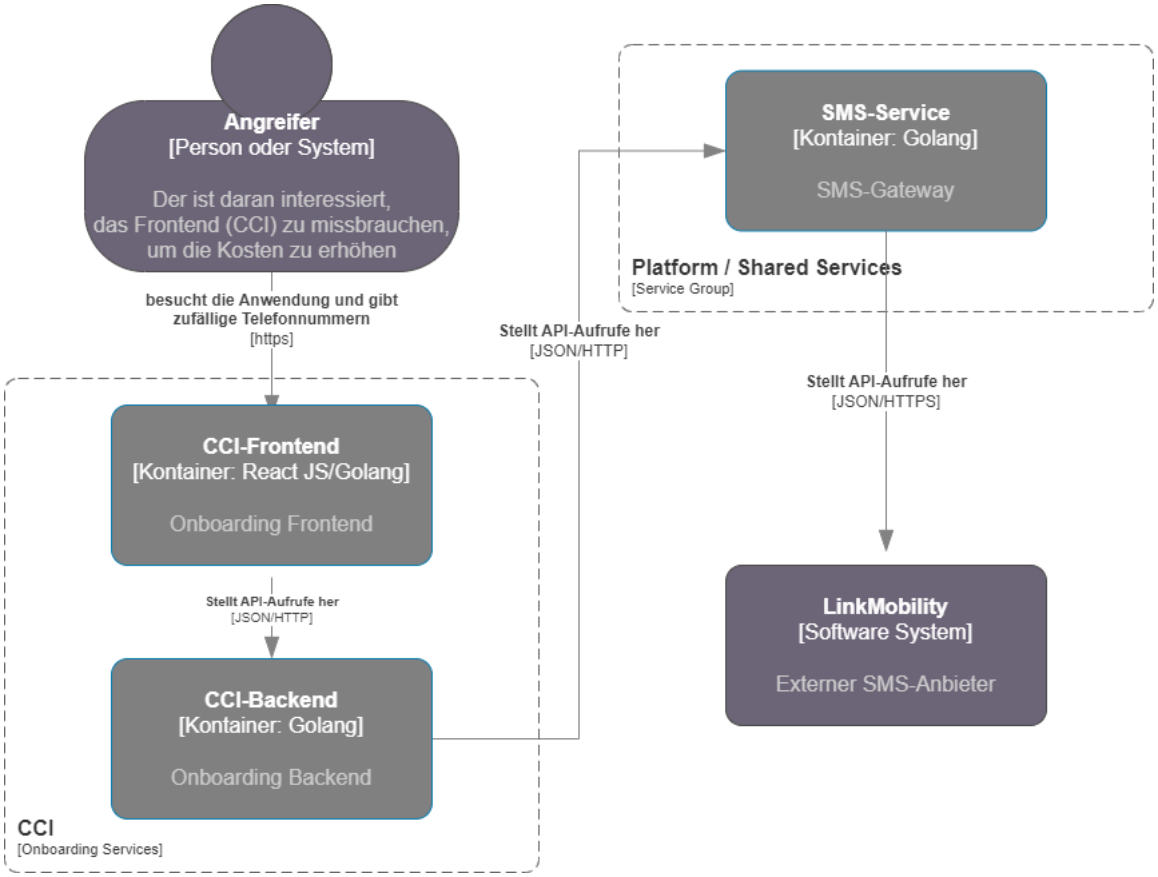


Abbildung 1: Grundlegende Architektur

Der Arbeitsablauf für den Versand von SMS über die Zielarchitektur beginnt damit, dass der Benutzer (entweder ein Angreifer oder ein potenzieller Kunde) seine Rufnummer in das CCI-Frontend eingibt. Das CCI-Backend fügt dann zusätzliche Informationen zur Anfrage hinzu, darunter den Autorisierungs-Header und den Namen des Dienstes, der die Anfrage sendet. Die Anfrage wird dann direkt an das SMS-Gateway weitergeleitet.

Nach Erhalt der Anfrage prüft das SMS-Gateway die Anfrage auf die Richtigkeit der Authentifizierungsinformationen und des Anfragekörpers. Wenn alles in Ordnung ist, leitet das SMS-Gateway die Anfrage an einen externen SMS-Provider weiter. Der SMS-Provider sendet dann die SMS-Nachricht an die Telefonnummer des Benutzers.

Die derzeitige Architektur für den Versand von SMS über die aktuelle Infrastruktur birgt erhebliche Sicherheitsrisiken, da es an robusten Verfahren zur Benutzervalidierung fehlt. Das System überprüft Benutzerinformationen wie Telefonnummern

nicht ausreichend und berücksichtigt keine Faktoren wie die Gültigkeit der Telefonnummer oder die Häufigkeit von Anfragen von derselben Landesvorwahl, demselben Anbieter, derselben Nummer oder derselben IP-Adresse. Infolgedessen ist das SMS-Gateway anfällig für Missbrauch und Ausnutzung durch Angreifer, was eine erhebliche Bedrohung für das Unternehmen darstellt.

2.1.2 Angriff und Kosten

Die vorliegende Studie dokumentiert die Maßnahmen, die flatexDEGIRO ergriffen hat, um bösartige Angriffe auf die Anwendung des Unternehmens, ein SMS-Gateway, abzuwehren. Die Analyse der während des Angriffszeitraums gesammelten Daten zeigt einen starken Anstieg der Zahl der Anfragen von dem CCI, wie in Abbildung 2 zu sehen ist.

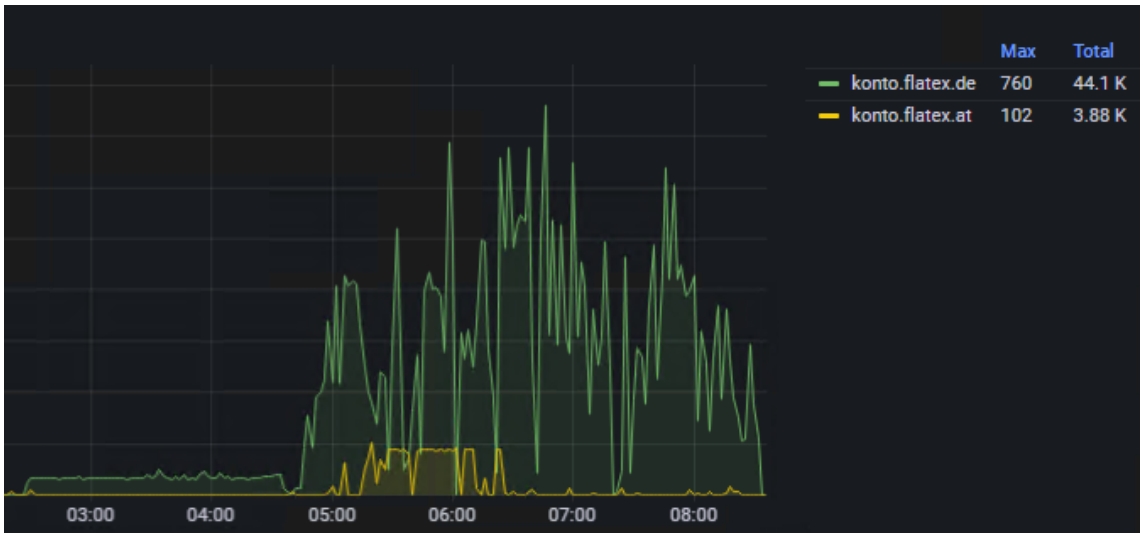


Abbildung 2: Beginn der Angriffe

Ein konkretes Beispiel für das Problem ist in Abbildung 3 zu sehen, wo ein Angreifer erfolgreich 18.000 Nachrichten nach Ägypten geschickt hat.

Abbildung 3: Ägypten

Dieses Beispiel ist nicht der einzige Fall von Angriffen. Insgesamt bestätigte die für das SMS-Gateway von flatexDEGIRO zuständige Abteilung, dass innerhalb eines

Monats 125.000 Nachrichten von den Angreifern verschickt wurden, was zu erheblichen finanziellen Verlusten für das Unternehmen führte. Die genauen Kostenangaben können aufgrund der Geheimhaltungsrichtlinien des Unternehmens nicht veröffentlicht werden. Die Kosten, die dem Unternehmen entstanden sind, lagen jedoch deutlich über den üblichen SMS-Preisen.

2.1.3 Bisherige Maßnahmen

Bislang hat das Unternehmen verschiedene Methoden ausprobiert, um das Problem der Cyberangriffe in den Griff zu bekommen, doch keine dieser Methoden hat eine dauerhafte oder zufriedenstellende Lösung gebracht.

Eine der von flatexDEGIRO durchgeführten Maßnahmen war die Aktivierung von Beschränkungen für IP-Adressen und Telefonnummern. Diese Begrenzung erlaubte den Versand von nur 5 Nachrichten pro Minute von einer IP-Adresse und maximal 10 Anfragen pro Tag von einer Telefonnummer. Die Einführung dieser Beschränkung hatte erhebliche Auswirkungen auf die Anzahl der von Angreifern gestellten Anfragen, wie in Abbildung 4 dargestellt. Der Unterschied zwischen der Anzahl der Anfragen vor und nach der Einführung des Limits ist deutlich erkennbar.

Darüber hinaus erhielt die IT-Sicherheitsabteilung Daten von der Myra [1]. In Abbildung 2 ist zu sehen, wann die Angreifer ihre Angriffe starten, wobei sie einige Bots einsetzen.

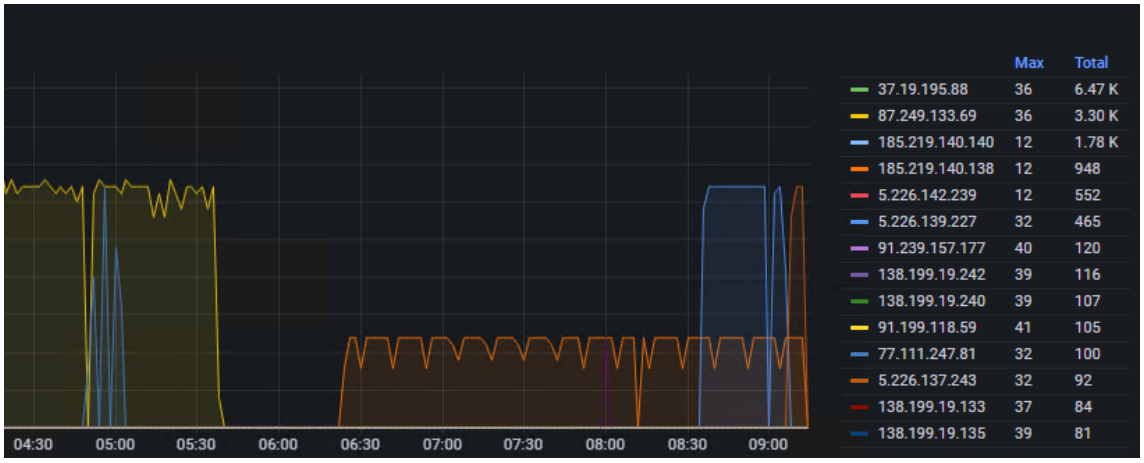


Abbildung 4: Anzahl der Angriffe bevor das IP-Limit

Trotz der Wirksamkeit dieser Maßnahme war diese Methode nur eine vorübergehende Lösung. Das Unternehmen erkannte, dass diese Lösung keinen vollständigen Schutz gegen potenzielle Angreifer bot. Daher beschloss das Unternehmen, ein neues Projekt zu implementieren, nämlich SMS-Gatekeeper.

[1]Myra `https://www.myrasecurity.com/de/`

2.2 Soll-Zustand des Projekts

Ziel dieses Kapitels ist es, einen Überblick über die Zielarchitektur und die geplanten Maßnahmen für das Projekt zu geben. In diesem Kapitel werden die Schritte beschrieben, die erforderlich sind, um den gewünschten Zustand des Projekts zu erreichen.

Die Zielarchitektur des Projekts umfasst die Implementierung eines sicheren und skalierbaren SMS-Gatekeeper-Dienstes. Der Abschnitt über die Zielarchitektur wird außerdem beschreiben, wie sich die Grundarchitektur ändern wird, welche neuen Komponenten hinzugefügt werden und welche der vorhandenen Komponenten ihr Verhalten ändern werden, ohne dass die Projekte selbst oder ihre Funktionalität verändert werden.

Im nächsten Abschnitt geplante Maßnahmen wird beschrieben, welche Maßnahmen zur Lösung des Problems ergriffen werden sollen. Die neuen Maßnahmen berücksichtigen die aktuelle Situation mit Angriffen und die bereits ergriffenen Maßnahmen, die im vorherigen Kapitel beschrieben wurden. Nachfolgend wird der Soll-Zustand im Detail beschrieben.

Aus dem Kapitel Ist-Zustand ist bereits bekannt, dass die Zielarchitektur um neue Komponenten, nämlich den neuen SMS-Gatekeeper-Dienst und dessen Datenbank, ergänzt werden muss. Andere Dienste, die mit dem SMS-Gateway interagieren, müssen stattdessen mit dem SMS-Gatekeeper interagieren.

Darüber hinaus muss bei der Entwicklung des SMS-Gatekeepers die Sicherheit im Vordergrund stehen. Im Gegensatz zu SMS-Gateway soll der neue Dienst nicht nur als Vermittler zwischen dem CCI und dem SMS-Gateway dienen, sondern auch für die Sicherheit verantwortlich sein. Daher müssen alle potenziellen Angriffe durch den SMS-Gatekeeper verhindert werden, ohne dass sie an andere Dienste weitergegeben werden.

Bei der Umsetzung von Maßnahmen zur Verhinderung von Angriffen werden auch die Kosten und die Ausfallsicherheit berücksichtigt. Externe, kostenpflichtige Dienste werden nur als letzter Ausweg genutzt, wenn die Überprüfung freier und lokaler Daten unzureichend ist.

Zusätzlich sollten einige Funktionen des SMS-Gateways, wie z.B. das Zwischenspeichern von Nachrichten und die Übermittlung statistischer Daten per E-Mail, an den SMS-Gatekeeper übertragen werden. Außerdem sollte der SMS-Gatekeeper nicht fest an einen bestimmten SMS-Provider gebunden sein, sondern flexibel für zukünftige Erweiterungen sein.

Insgesamt besteht das Ziel dieses Kapitels darin, einen klaren Ablaufplan für die Erreichung des gewünschten Projektzustands zu erstellen. Durch sorgfältige Planung, Implementierung von Sicherheitskontrollen und laufende Wartung soll ein zuverlässiger und sicherer Dienst bereitgestellt werden, der den Anforderungen von flatexDEGIRO und seinen Kunden entspricht.

2.2.1 Zielarchitektur

Ein neuer Dienst SMS-Gatekeeper muss implementiert und in die bestehende Infrastruktur integriert werden. Daher wird die grundlegende Architektur neu gestaltet, um die Integration des neuen Dienstes SMS-Gatekeeper zu berücksichtigen.

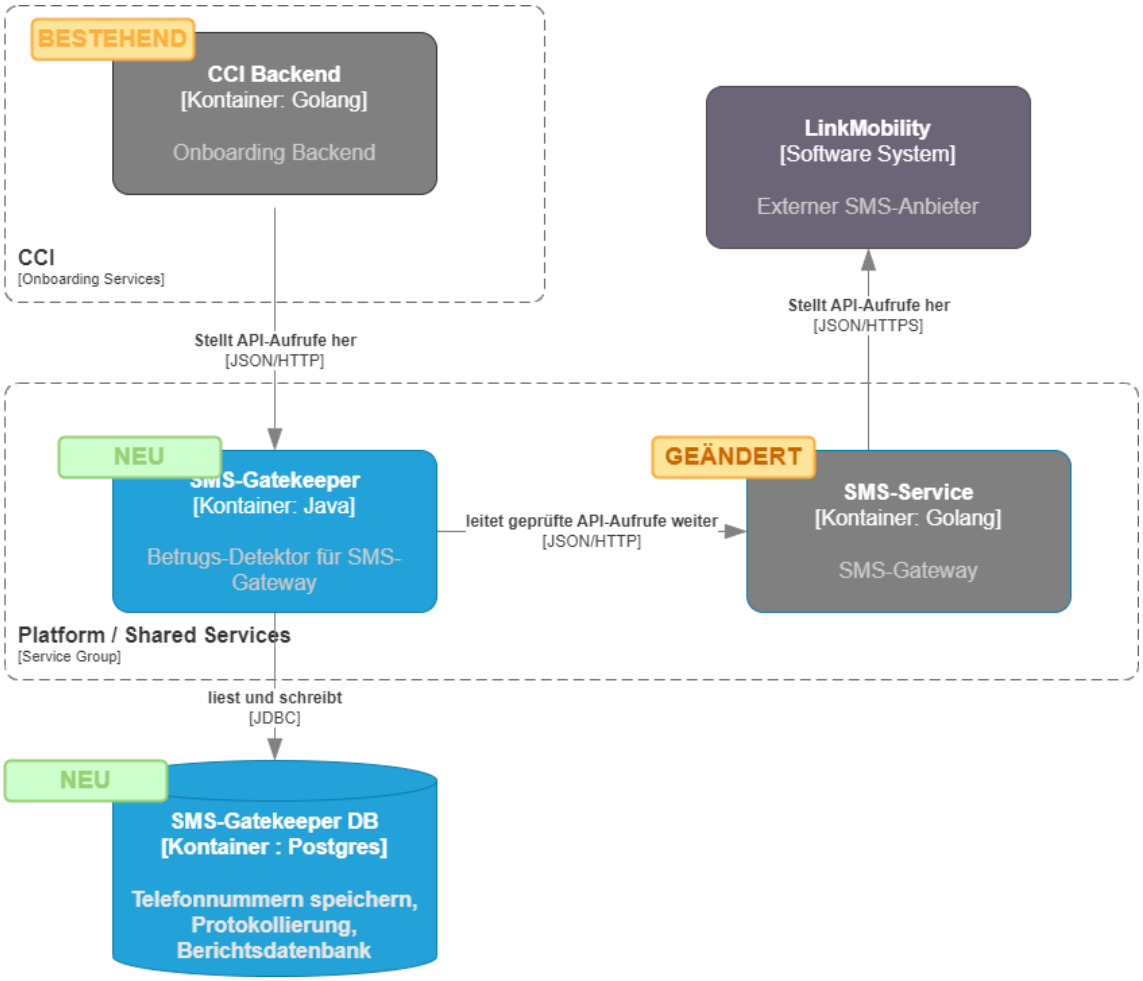


Abbildung 5: Zielarchitektur

Die vorgeschlagene Zielarchitektur (siehe Abbildung 5) für flatexDEGIRO umfasst eine neue Komponente namens SMS-Gatekeeper, die als Vermittler zwischen den Komponenten CCI und SMS-Gateway dient. Die Plattform-Services beinhaltet den SMS-Gatekeeper als eine ihrer Komponenten. Innerhalb dieser Plattform-Services hat jeder einzelne Dienst, einschließlich des SMS-Gatekeepers, seine eigene Datenbank. In der bisherigen Architektur (siehe Abbildung 1) wurden die von dem CCI gesendeten Anfragen direkt von SMS-Gateway bearbeitet. Aber in der neuen Architektur ist der SMS-Gatekeeper für die Bearbeitung von Anfragen von dem CCI zuständig und analysiert die Anfragen, bevor er entscheidet, ob die Anfragen an das SMS-Gateway weitergeleitet werden sollen. Der SMS-Gatekeeper bietet zusätzliche Sicherheits- und Filterfunktionen für das System. Durch die Analyse von Anfragen des CCI-Dienstes kann der SMS-Gatekeeper Angriffe von Bots oder anderen böswilligen Benutzern erkennen und verhindern, bevor die Angriffe das SMS-Gateway erreichen.

2.2.2 Voraussichtliche Maßnahmen

Die für das Projekt geplanten Maßnahmen sollen gewährleisten, dass das neue Sicherheitssystem effektiv und effizient umgesetzt wird. Zunächst wird eine umfassende Sicherheitsbewertung des derzeitigen Systems durchgeführt, um alle Schwachstellen zu ermitteln, die behoben werden müssen.

Um sicherzustellen, dass das neue Projekt alle notwendigen Anforderungen erfüllt, werden Akzeptanzkriterien entwickelt. Anhand dieser Kriterien wird die Wirksamkeit des neuen Sicherheitssystems bewertet und sichergestellt, dass es alle erforderlichen Standards erfüllt.

Bei der Einführung des neuen Sicherheitssystems werden die für die Umsetzung erforderlichen Kosten berücksichtigt. Ursprünglich war geplant, für die Bestimmung der Landesvorwahl, des Anbieters und der Art der Nummer kostenpflichtige Dienste zu nutzen. Um jedoch die Kosten zu senken, werden alternative Ansätze in Betracht gezogen. Dazu gehören die Nutzung der kostenlosen Bibliothek libphonenumber sowie die Erstellung einer lokalen Datenbank, die zunächst mit den Daten des NNPs gefüllt und dann im Laufe des Algorithmus mit neuen Daten erweitert wird. Sollte die Inanspruchnahme kostenpflichtiger Dienste erforderlich sein, so wird dies nur als letztes Mittel geschehen.

Mit diesen geplanten Maßnahmen wird das Unternehmen in der Lage sein, effektiv und effizient ein neues Sicherheitssystem zu implementieren, das alle notwendigen Anforderungen erfüllt. Darüber hinaus wird das Unternehmen durch die Verwendung alternativer Ansätze in der Lage sein, die Kosten zu senken und das Projekt kosteneffizienter zu gestalten.

2.3 Problemlösungsansatz

Das Kapitel Problemlösungsansatz stellt die Forschungsmethodik und den Prozess zur Lösung der Problemstellung vor. Dieser Teil ist in drei Abschnitte unterteilt, nämlich ITU, E.164 und Mustererkennung.

Der erste Abschnitt dieses Kapitels befasst sich mit der Internationalen Fernmeldeunion (ITU) und ihren grundlegenden Empfehlungen, die die Basis der modernen Telekommunikations- und Informationstechnologien bilden. Darüber hinaus wird in diesem Abschnitt die Bedeutung des Telekommunikation Standardisierung Sektors hervorgehoben und insbesondere die Relevanz der E.164-Empfehlung erklärt.

Der zweite Abschnitt dieses Kapitels befasst sich mit E.164, dem internationalen Standard für den Nummernplan, insbesondere mit E.164-Nummern für geografische Gebiete. In diesem Abschnitt wird auch die Struktur von E.164-Nummern beschrieben und warum diese Struktur für den SMS-Gatekeeper wichtig ist.

Der letzte Abschnitt dieses Kapitels befasst sich mit der Mustererkennung, insbesondere mit der Funktionsweise des Algorithmus. Wie der mehrstufige Algorithmus eingehende HTTP-Anfragen analysieren und wichtige Informationen extrahieren kann, die es dem Algorithmus ermöglichen, die richtigen Entscheidungen darüber zu treffen, wie die einzelnen Anfragen zu behandeln sind.

Diese drei Abschnitte sind für die Lösung der Problemstellung von wesentlicher Bedeutung, da sie ein gründliches Verständnis der technischen Grundlagen vermitteln, die für die Entwicklung der vorgeschlagenen Lösung erforderlich sind. Die detaillierte Untersuchung der Rolle der einzelnen Abschnitte wird in den Abschnitten selbst weiter erörtert.

Insgesamt bietet der Problemlösungsansatz einen umfassenden Überblick über die Forschungsmethodik und den Prozess zur Lösung der Problemstellung und zeigt die Bedeutung jedes Abschnitts bei der Entwicklung der vorgeschlagenen Lösung auf.

2.3.1 ITU

Die ITU ist die Sonderorganisation der Vereinten Nationen für Informations- und Kommunikationstechnologien (IKT). Die ITU wurde 1865 gegründet, um die ersten internationalen Telegrafennetze zu verwalten, aber im Laufe der Jahre hat sich ihre Tätigkeit erweitert.[34] Heutzutage hat die ITU drei Haupttätigkeitsbereiche, die in Sektoren unterteilt sind.[37]

Diese sind:

Funkkommunikation Sektor (ITU-R) - dieser Sektor ist für die Verwaltung des internationalen Funkfrequenzspektrums und der Ressourcen in der Satellitenumlaufbahn zuständig.[37]

Telekommunikationsentwicklung Sektor (ITU-D) - dieser Sektor bietet einen nachhaltigen und erschwinglichen Zugang zu IKT.[37]

Telekommunikation Standardisierung Sektor (ITU-T) - bildet die Grundlage für den Betrieb von IKT-Netzen.[37]

Nach Angaben der ITU haben die ITU-T-Empfehlungen einen fakultativen Status, bis sie in nationales Recht umgesetzt werden. Dennoch sind sie aufgrund ihrer internationalen Anwendbarkeit und der hohen Qualität, die vom ITU-T-Sekretariat und den Mitgliedern aus den weltweit führenden Unternehmen der IKT und den globalen Verwaltungen gewährleistet wird, in hohem Maße befolgt.[13]

Die ITU-T verfügt über mehr als 4000 Empfehlungen, von der Definition von Diensten bis hin zur Netzarchitektur und anderen, die die Grundlagen der heutigen IKT bilden.[13]

Die ITU-T verfügt zwar über eine umfangreiche Sammlung von Empfehlungen, die verschiedene Aspekte der Telekommunikations- und Informationstechnologien abdecken, doch für die Zwecke dieser Untersuchung liegt der Schwerpunkt speziell auf der E.164-Empfehlung. Wie bereits in den vorangegangenen Kapiteln erwähnt, beruht der Missbrauch des SMS-Gateways auf der Telefonnummer, indem gefälschte oder nicht validierte Telefonnummern für den Versand von SMS verwendet werden. Bei der Validierung von Telefonnummern kann der spezielle Standard E.164 der ITU nützlich sein, der die oben genannte Art des Missbrauchs verhindern kann.

Daher ist die E.164-Empfehlung die wichtigste ITU-Empfehlung für die Lösung des Sicherheitsproblems des SMS-Gateways. Andere ITU-Empfehlungen können für die Entwicklung und Implementierung zusätzlicher Sicherheitsmaßnahmen nützlich sein, aber für den Zweck dieser Arbeit wird der Schwerpunkt auf dem E.164-Nummernplan als primäre Lösung liegen.

Im nächsten Abschnitt wird die Umsetzung des E.164-Standards im Detail untersucht und seine Wirksamkeit bei der Lösung des Sicherheitsproblems des SMS-Gateways bewertet.

2.3.2 E.164

Der Abschnitt E der ITU-T-Empfehlungen bezieht sich auf den allgemeinen Netzbetrieb, den Telefondienst, den Dienstbetrieb und menschliche Faktoren. Davon bezieht sich E.160-E.169 auf den Nummerierungsplan für den internationalen Telefondienst. e164standart ist eine Standardisierung mit der Bezeichnung Internationaler Nummerierungsplan für den öffentlichen Fernmeldedienst.[36]

Die Empfehlung E.164 bietet die Struktur und Funktionalität von Nummern für fünf Kategorien von Nummern, die für die internationale öffentliche Telekommunikation verwendet werden. Diese Kategorien sind:

- Geographic Areas (Geografische Gebiete)
- Global Services (Globale Dienste)
- Networks (Netzwerke)
- Group of Countries (GoC) (Gruppe von Ländern)
- Resources for Trials (Ressourcen für Tests)

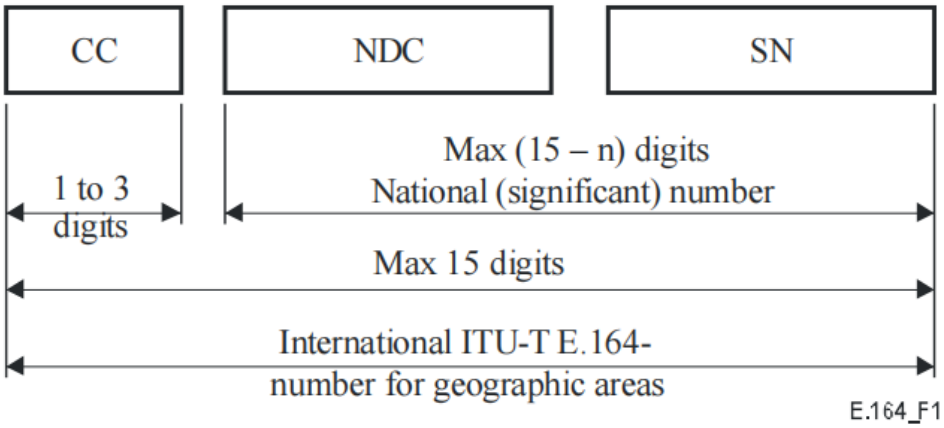
Für jede Kategorie werden die Komponenten der Nummerierungsstruktur und die Ziffernanalyse, die für eine erfolgreiche Anrufweiterleitung erforderlich sind, im Detail beschrieben.

Im Rahmen dieser wissenschaftlichen Arbeit werden weiterhin nur die internationalen ITU-T E.164-Nummern für geografische Gebiete betrachtet.

Der E.164-Nummernplan ist für geografische Gebiete besonders wichtig, da dieser Plan eine standardisierte und weltweit anerkannte Methode zur Identifizierung und Weiterleitung von Kommunikation an bestimmte Regionen bietet. Erreicht wird dies durch die Verwendung von Ländercodes (CC), die von der ITU-T vergeben werden und jedes Land eindeutig identifizieren, und von nationalen (signifikanten) Nummern N(S)N), die von den nationalen Telekommunikationsbehörden vergeben und verwaltet werden. Durch die Einhaltung der E.164-Empfehlungen für geografische Gebiete können die Telekommunikationsbetreiber sicherstellen, dass die Kommunikation unabhängig vom Standort des Absenders oder Empfängers ordnungsgemäß an die vorgesehenen Ziele weitergeleitet wird.

Nach der internationalen ITU-T E.164-Nummer darf die Länge der Nummer maximal 15 Ziffern betragen, wobei 1 bis 3 Ziffern den Ländercode für geografische Gebiete (CC) darstellen und der Rest die nationale (signifikante) Nummer N(S)N ist. Die N(S)N selbst kann in den nationalen Zielcode (NDC) und die Teilnehmernummer (SN) unterteilt werden.

Abbildung 6 zeigt die internationale ITU-T E.164-Nummern-struktur für geografische Gebiete.



CC Country Code for geographic area
NDC National Destination Code
SN Subscriber Number
n Number of digits in the country code

NOTE – National and international prefixes are not part of the international ITU-T E.164-number for geographic areas.

Abbildung 6: E.164 Nummernstruktur

Quelle: International ITU-T E.164-number structure for geographic areas [36]

Die internationale ITU-T E.164-Nummer ist für die Sicherheitsfragen des SMS-Gateways besonders relevant, da sie für die Mustererkennung von Telefonnummern im SMS-Gatekeeper-Dienst verwendet wird. Wie in den vorangegangenen Abschnitten erwähnt, bietet der E.164-Nummernplan ein standardisiertes Format für Tele-

fonnummern, das dem SMS-Gatekeeper die Erkennung und Extraktion von Informationen wie der Landesvorwahl (CC), die nationale signifikante Nummer (N(S)N), die nationale Zielvorwahl (NDC) und die Teilnehmernummer (SN) aus einer bestimmten Telefonnummer erleichtert.

Diese extrahierten Informationen werden dann für verschiedene Überprüfungen verwendet, wie z. B. die Überprüfung der Gültigkeit der Telefonnummer, die Überprüfung der Häufigkeit von Anfragen mit derselben Landesvorwahl, demselben Anbieter, derselben Nummer und derselben IP und die Sicherstellung, dass die Anfrage von einem potenziellen Nutzer und nicht von einem Angreifer stammt. Darüber hinaus werden diese Informationen verwendet, um die Telefonnummer in einem normalisierten Format für die künftige Verwendung in der Datenbank zu speichern und den entsprechenden Anbieter auf der Grundlage der CC, NDC und der NDC-Länge zu ermitteln. Der spezifische Sicherheitsmechanismus wird im folgenden Kapitel über die Mustererkennung näher erläutert.

2.3.3 Mustererkennung

Um die vordefinierten Anforderungen des Projekts zu erfüllen, wurde beschlossen, einen mehrstufigen Sicherheitsalgorithmus zu entwickeln. Er basiert auf der Idee, dreistufige Prüfverfahren zu konstruieren und je nach Prüfmodus die entsprechende Prüfung zu implementieren. Das Ziel des Algorithmus ist es, eine eingehende Anfrage zu analysieren und zu erkennen, ob es sich um eine gute oder schlechte Anfrage handelt. In diesem Zusammenhang definiert der Autor eine 'gute' Anfrage als eine gewöhnliche Anfrage von echten Kunden, während 'schlechte' Anfragen von gefälschten Benutzern mit böswilligen Absichten gesendet werden, um die Kosten des Unternehmens für SMS-Sendungen zu erhöhen. Bei der Filterung der Anfragen berücksichtigt der Algorithmus eine Reihe von Faktoren, wie z. B. die Häufigkeit der Anfragen, die IP-Adresse und insbesondere die Telefonnummer, an die die SMS gesendet werden soll. Die folgenden Faktoren sind in den Akzeptanzkriterien definiert, um sicherzustellen, dass der SMS-Gatekeeper die festgelegten Anforderungen erfüllt.

Die Akzeptanzkriterien umfassen die folgenden Punkte:

```
  Scenario: client makes call to POST /send
    Given a request body is passed
    When the client calls /send
    Then the message should not be empty
    Then If the cc is on the blacklist no SMS is sent
    Then If the number is in the locked list no SMS is sent
    Then If the cc is not on the black-/whitelist, a captcha is required
    Then If more than 5 SMS have been sent to the same number in the
↪   last 24 hours, a captcha is required
    Then If more than 5 SMS have been sent from the same IP in the last
↪   24 hours, a captcha is required
    Then the phone number must be valid
    Then If the number of SMS to a cc deviate by 30% compare to the
↪   daily average of the last n days, an alert is triggered
    Then the client receives status code of 200
```

Je nach den Akzeptanzkriterien kann folgendes Szenario definiert werden:

- `Schritt 1: Es wird ein Anfragekörper übergeben, der nicht leer sein sollte`
  - Dieser Punkt stellt sicher, dass der Inhalt der Nachricht vom Benutzer bereitgestellt wird und nicht leer ist. Dies ist wichtig, um sicherzustellen, dass der SMS-Gatekeeper den korrekten Anfragekörper ohne Datenverlust erhält.

    Aus Listing 6 (siehe Listing 6) ist erkennbar, dass der Anfragekörper anhand der Jakarta-Bean-Validierung mit entsprechenden Einschränkungen wie `@NotNull, @NotEmpty` validiert wird.
- `Schritt 2: Wenn der CC auf der Schwarzliste ist, wird keine SMS gesendet`
  - Dieser Punkt prüft, ob der CC des Empfängers auf der Schwarzliste steht. Wenn der CC auf der Schwarzliste steht, sollte keine SMS gesendet werden, um zu vermeiden, dass Nachrichten an Länder gesendet werden, die ein Risiko darstellen könnten oder in denen das Unternehmen nicht tätig ist.

    In der Datei `application.properties` gibt es eine Eigenschaft namens `SMS-GATEKEEPER_BLACKLISTED_COUNTRIES`, die später von Operatoren des Dienstes konfiguriert werden kann. Dieser Punkt wird in der folgenden Methode gegen diese Eigenschaft geprüft, wobei `SmsGatekeeperConfig-Klasse` (siehe Listing 8) eine Konfigurationsklasse für SMS-Gatekeeper-bezogene Eigenschaften ist:

```
    @Override
    public boolean isInBlackListedCountry(int cc) {
    return gatekeeperConfig.getBlackListedCCs().contains(cc);
    }
```

- `Schritt 3: Wenn der NDC in der Sperrliste steht, wird keine SMS gesendet`
  - Dieser Punkt prüft, ob die führenden Ziffern der Nummer auf der Sperrliste stehen. Wenn ja, sollte keine SMS gesendet werden, bis die Nummer wieder entsperrt wird, um einen Missbrauch des SMS-Gatekeepers zu verhindern.

    Bei der folgenden Methode werden die führenden Ziffern, d.h. NDC, mit der gesperrten Liste aus der Datenbank abgeglichen.

```
    @Override
    public boolean isInLockedListedNumbers(String leadingDigits)
↪   {
        var lockedNumbers = lockNumberService.getLockedNumbers();
        return lockedNumbers.contains(leadingDigits);
    }
```

- `Schritt 4: Wenn das CC weder auf der Weiß- noch auf der Schwarzliste steht, ist ein Captcha erforderlich`
  - Wenn der CC des Empfängers nicht auf der Weiß- oder Schwarzliste steht, wird der Benutzer aufgefordert, ein Captcha einzugeben. Damit soll sichergestellt werden, dass der Kunde ein echter Nutzer des Unternehmens ist und ein Missbrauch des SMS-Gatekeepers verhindert wird.

    Bei der folgenden Methode werden die CC gegen Schwarz- und Weißlisten geprüft.

```
    if (countryService.isInBlackListedCountry(cc))
      throw new BlackListedCountryException(msg);
    if (!countryService.isInWhiteListedCountry(cc))
      throw new CaptchaRequiredException(msg);
```

- `Schritt 5: Wenn in den letzten 24 Stunden mehr als 5 SMS an die gleiche Nummer gesendet wurden, ist ein Captcha erforderlich:`
  - Dieser Punkt prüft die Anzahl der SMS, die innerhalb der letzten 24 Stunden an eine bestimmte Nummer gesendet wurden. Wenn mehr als 5 SMS gesendet wurden, ist ein Captcha erforderlich, um Spamming oder Missbrauch des SMS-Gatekeeper zu verhindern.

    Darüber hinaus sind die Zahlen wie 5 SMS und 24 Stunden flexibel konfiguriert, so dass bei Bedarf auch andere Zahlen angegeben werden können.

```
    int rqstAmountFromSameNum = messageLogService
        .findAmountOfRequestsFromSameNumberInSpecificTimeInterval(
          gatekeeperConfig.getRequestTimeIntervalSeconds(),
          messageLogRequest.number()
        );

    if (rqstAmountFromSameNum != 0 && rqstAmountFromSameNum >=
↪   gatekeeperConfig.getAllowedRqstAmountInTimeIntervalSeconds()
        ){
            throw new CaptchaRequiredException("Do captcha!");
          }
```

- `Schritt 6: Wenn in den letzten 24 Stunden mehr als 5 SMS von der gleichen IP-Adresse gesendet wurden, ist ein Captcha erforderlich`
  - Dieser Punkt prüft die Anzahl der SMS, die innerhalb der letzten 24 Stunden von derselben IP versendet wurden. Wenn mehr als 5 SMS gesendet wurden, ist ein Captcha erforderlich, um Spamming oder Missbrauch des SMS-Gatekeeper zu verhindern.

    Darüber hinaus sind die Zahlen wie 5 SMS und 24 Stunden flexibel konfiguriert, so dass bei Bedarf auch andere Zahlen angegeben werden können.

```
    int rqstAmountFromSameIp = messageLogService
        .findAmountOfRequestsFromSameIpInSpecificTimeInterval(
          gatekeeperConfig.getRequestTimeIntervalSeconds(),
          messageLogRequest.ip()
        );

    if (rqstAmountFromSameIp != 0 && rqstAmountFromSameIp >=

↪   gatekeeperConfig.getAllowedRqstAmountInTimeIntervalSeconds()
        ){
            throw new CaptchaRequiredException("Do captcha!");
          }
```

- `Schritt 7: Die Telefonnummer muss gültig sein`
  - Dieser Punkt stellt sicher, dass die vom Benutzer angegebene Rufnummer gültig ist. Um sicherzustellen, dass die Nummer gültig ist, wird der mehrstufige Algorithmus aufgerufen. Der Prozess wird im Folgenden detailliert beschrieben.
- `Schritt 8: Wenn die Anzahl der SMS an den cc um 30% vom Tagesdurchschnitt der letzten 15 Tage abweicht, wird ein Alarm ausgelöst`
  - Dieser Punkt überwacht die Anzahl der an einen CC gesendeten SMS und vergleicht diese Anzahl mit dem Tagesdurchschnitt der letzten 15 Tage. Wenn die Anzahl der gesendeten SMS um 30% abweicht, wird ein Alarm ausgelöst. Dieser Alarm wird in Form von Logdateien erstellt, die in Grafana angezeigt werden. Der Zweck dieses Warnsignals ist es, den Operator über Unregelmäßigkeiten bei der Anzahl der versendeten SMS zu informieren.

    Außerdem sind die Zahlen wie 30% und 15 Tage flexibel konfiguriert, so dass bei Bedarf auch andere Zahlen angegeben werden können.

```
    @Override
    public boolean
↪   IsAverageAmountOfRequestsForSameCountryInLastNDaysDeviate(int
↪   n, int cc, double deviation) {
        var period =
↪   Timestamp.valueOf(LocalDateTime.now().minusDays(n));
        int amount =
↪   messageLogRepository.findAmountOfRqstsToSameCCInPeriod(period,
↪   cc).orElse(0);
        var avg = amount / n;
        var averageDailyLimitOfPastNdays = avg * deviation;
        var today = Timestamp.valueOf(LocalDateTime.now());
        var averageDailyLimitOfToday = messageLogRepository
            .findAmountOfRqstsToSameCCInPeriod(today, cc);
        return averageDailyLimitOfPastNdays <=
↪   Double.valueOf(averageDailyLimitOfToday.orElse(0));
    }
```

Die bereitgestellte Methode nimmt drei Parameter auf: int n, int cc und double deviation.

Die Methode berechnet zunächst den Zeitstempel des Beginns des Zeitraums, der n Tage vor dem aktuellen Datum und der aktuellen Uhrzeit liegt. Dann verwendet diese Methode ein messageLogRepository, das Zugriff auf die Datenbank hat, um die Anzahl der Anfragen abzurufen, die während dieses Zeitraums an denselben CC wie durch Parameter cc angegeben gestellt wurden.

Die Methode berechnet dann die durchschnittliche Anzahl der Anfragen pro Tag für diesen Zeitraum und multipliziert diese Zahl mit dem Prozentsatz der Abweichung, um das Tageslimit für diesen Zeitraum zu berechnen. Diese Methode ruft die Anzahl der Anfragen an denselben CC am aktuellen Tag ab und vergleicht diese Zahl mit dem für den Zeitraum berechneten Tageslimit. Wenn die Anzahl der Anfragen am aktuellen Tag kleiner oder gleich dem Tageslimit für den Zeitraum ist, gibt die Methode true zurück, was bedeutet, dass die durchschnittliche Anzahl der Anfragen für denselben CC in den letzten n Tagen nicht wesentlich vom Durchschnitt abweicht.

Zusammenfassend lässt sich sagen, dass diese Methode verwendet wird, um festzustellen, ob die Anzahl der Anfragen an denselben CC in den letzten n Tagen signifikant von der durchschnittlichen Anzahl der Anfragen pro Tag für diesen Zeitraum abweicht, basierend auf einem bestimmten Abweichungsprozentsatz. Dies kann nützlich sein, um Anomalien im Datenverkehr zu erkennen und mögliche Angriffe oder andere Probleme zu verhindern.

- `Schritt 9: Der Client erhält einen Statuscode von 200`
  - Dieser Punkt stellt sicher, dass der Benutzer einen Statuscode von 200 erhält, der anzeigt, dass der Aufruf von POST /send erfolgreich war. Dies ist für den Benutzer wichtig, um zu überprüfen, ob die Nachricht erfolgreich gesendet wurde.

In dem Schritt 7, in dem eine Zahl auf ihre Gültigkeit überprüft wird, wurde der mehrstufige Algorithmus erwähnt. Der mehrstufige Sicherheitsalgorithmus ist ein entscheidender Aspekt des entwickelten Verteidigungsmechanismus.

Der Algorithmus soll sicherstellen, dass mehrere Sicherheitsebenen implementiert werden, um einen verbesserten Schutz gegen potenzielle Angriffe zu bieten und die Kundenregistrierung auf der Grundlage der vorgegebenen Akzeptanzkriterien zu gewährleisten. Die Ebenen sind die folgenden:

Bibliothek libphonenumber - es handelt sich um die Bibliothek von Google zum Parsen, Formatieren und Validieren internationaler Telefonnummern.[8] Eine mögliche Verwendung von libphonenumber besteht darin, eine Vielzahl von Datenpunkten über eine Telefonnummer zu extrahieren, einschließlich der Landesvorwahl der Telefonnummer. Diese Informationen können genutzt werden, um festzustellen, ob flatexDEGIRO für den Empfang von SMS von einer bestimmten Telefonnummer offen ist, die mit einem erlaubten oder verbotenen Land verbunden ist.

Twilio SDK - es handelt sich um ein SDK, das von Twilio bereitgestellt wird, um mit der Twilio API zu interagieren.[33] Das Twilio SDK bietet eine einfache und intuitive API für eine Vielzahl von Programmiersprachen, mit der Entwickler schnell Kommunikationsanwendungen erstellen und bereitstellen können.

Lokaler Datensatz - es handelt sich um einen vom Autor gesammelten Datensatz. Dieser Datensatz wurde mit PostgreSQL, einem beliebten relationalen Open-Source-Datenbankmanagementsystem, erfasst und gespeichert. Der Zweck der lokalen Datenbank besteht darin, eine Reihe von nationalen Zielcodes (NDCs) und damit verbundene Details wie Ländercodes (CC), Anbieterinformationen, N(S)N-Länge und Nutzungsart (Mobilfunk oder Festnetz) zu enthalten. Die lokale Datenbank wird nicht nur verwendet, um sicherzustellen, dass die Nummern in der Anwendung korrekt den Anbieterländern zugeordnet sind, sondern die Datenbank dient auch als zusätzliche Sicherheitsebene für den Fall, dass die oben erwähnten externen Anwendungen ihr Verhalten ändern. Zu diesen Änderungen kann es gehören, dass der Zugang zu den Datenbanken der externen Anbieter gesperrt wird, dass Gebühren für den Zugang erhoben werden oder dass die Funktionslogik oder die Datenstruktur geändert wird. Durch eine lokale Version der erforderlichen Daten kann die Anwendung auch dann noch funktionieren, wenn diese externen Anwendungen nicht mehr verfügbar sind oder ihre Richtlinien ändern. Ein solcher Ansatz verbessert die allgemeine Widerstandsfähigkeit und Zuverlässigkeit der Anwendung und verringert das Risiko von Ausfallzeiten oder anderen Störungen, die sich auf die Benutzer auswirken könnten. Darüber hinaus wird die lokale Datenbank neben der Speicherung von Nummerninformationen auch zur Speicherung von Informationen über gesendete Anfragen und blockierte Nummern verwendet. Die Datenbank ist so konzipiert, dass Änderungen mithilfe von Liquibase-Tabellen nachverfolgt werden können, wodurch sichergestellt wird, dass das Datenbankschema in verschiedenen Umgebungen synchron bleibt. Die Struktur der Datenbank ist in Abbildung A.1 dargestellt.

Ein Endpunkt im SMS-Gatekeeper, der für die Bearbeitung von Anfragen zuständig ist, ist der Endpunkt `/send`. Es handelt sich um einen POST-Endpunkt, der JSON als Anforderungskörper annimmt, und zwar in Form (siehe Listing 6) von:

from - von wem die SMS kommt, Typ: String

to - die Rufnummer, an die die SMS gesendet werden soll

body - Inhalt der SMS

solution - Captcha-Lösung, Nullable: true

Da die Angriffe auf Telefonnummern beruhen, wird nur das Feld 'to', d. h. die Telefonnummer, näher betrachtet.

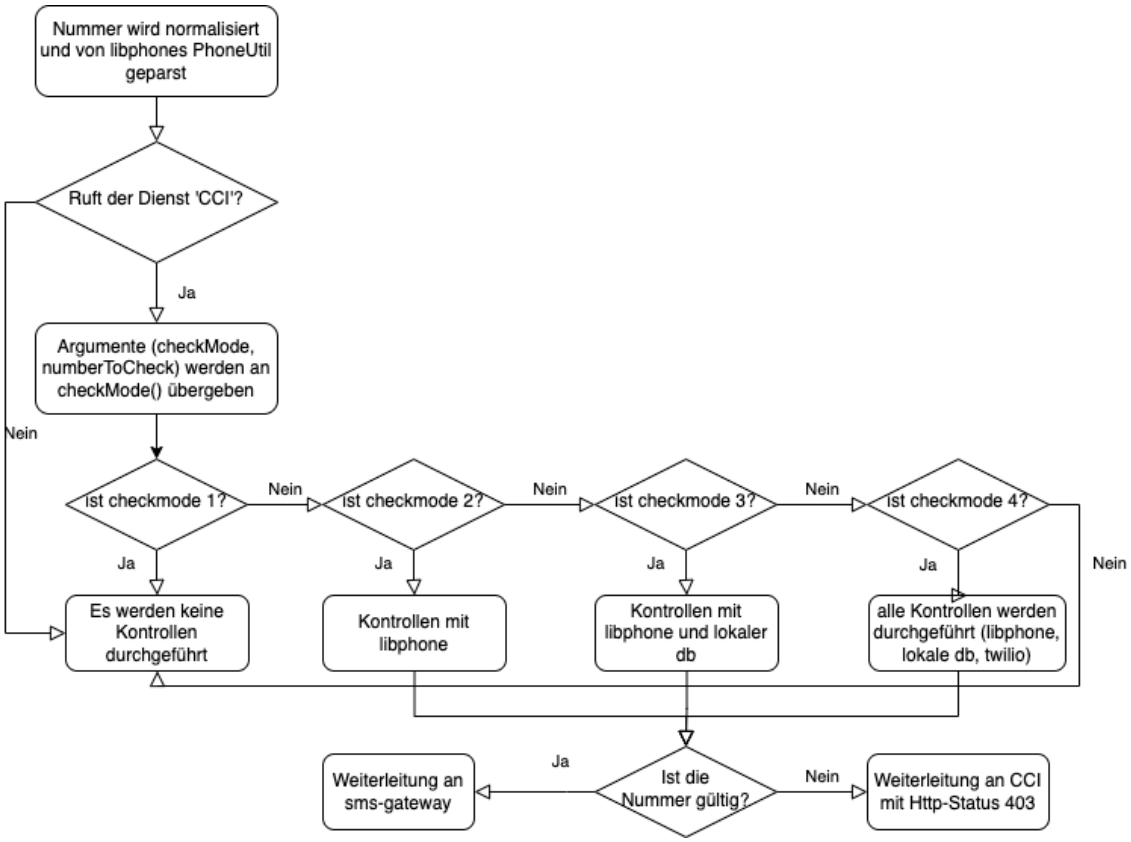


Abbildung 7: Prüfverfahren

Der Gesamtablauf des Algorithmus beginnt mit der Normalisierung der Telefonnummer in eine Form, die PhoneNumberUtil aus der libphonenumber-Bibliothek analysieren kann, und der Bildung einer statischen `PhoneNumber-Klasse` (siehe Listing 1). Der Algorithmus prüft dann, für welchen Dienst die Methode aufgerufen wird. Es gibt zwei Optionen, ob der aufrufende Dienst CCI oder ein interner Dienst von flatexDEGIRO ist. Wenn der Aufrufer das CCI ist, werden zwei Argumente an die `checkMode-Methode` (siehe Listing 9) übergeben, nämlich der Prüfmodus aus den Eigenschaften und die zu prüfende formatierte Nummer selbst. Die Schritte können in Abbildung 7 nachgelesen werden.

Der nächste Schritt besteht darin, den von den Anwendungsoperatoren über Umgebungsvariablen bereitgestellten Prüfmodus zu untersuchen, der je nach den Bedürfnissen der Organisation konfiguriert werden kann. Der Autor verwendet 4 Prüfmodi, beginnend mit 1, der schwächsten Sicherheitsüberprüfung, bis 4, der stärksten Sicherheit. Die Einzelheiten der einzelnen Prüfmodi werden im Folgenden beschrieben.

Wenn der Prüfmodus gleich 1 ist, führt die Anwendung keine Prüfungen durch (siehe Listing 9). Diese Option wird benötigt, um diejenigen herauszufiltern, die von internen Anwendungen des flatexDEGIRO aufgerufen werden. Der Grund für diese Prüfung ist, dass interne Projekte nicht unbedingt geprüft werden sollen, sondern nur, wenn der SMS-Gatekeeper vom CCI aufgerufen wird.

Der nachfolgende Prüfmodus, der als Modus 2 bezeichnet wird, ruft die Bibliothek libphonenumber wie oben beschrieben auf. Genauer gesagt wird die Methode `checkTargetNumber()` in libPhoneRuleSetService ausgeführt (siehe Listing 3), die

```
    // other parts are omitted
    case 2 -> libPhoneRuleSetService.checkTargetNumber(phoneNumber);
```

die `isValidNumber()-Methode` der Bibliothek verwendet. Diese Funktion führt eine gründliche Validierung durch, die sowohl das Präfix als auch die Längeninformationen der Telefonnummer überprüft. Diese Überprüfung garantiert, dass die Telefonnummer eine gültige Nummer ist, d.h. dass eine solche Nummer existiert. Bei erfolgreicher Validierung gibt die Funktion ein `NumberResponse-Objekt` (siehe Listing 7) zurück, das Informationen wie den Gültigkeitsstatus, den Nummerntyp (Mobilfunk oder Festnetz), die Landesvorwahl (CC), der NDC und den Dienstanbieter enthält. Alle diese Daten werden von der libphonenumber-Bibliothek bereitgestellt.

Wenn der Prüfmodus 3 zugewiesen wird, wird folglich die gleiche Methode wie im Prüfmodus 2 (siehe Listing 9) aufgerufen, `checkTargetNumber()`.

```
    case 3 -> {
              boolean isValid =
↪ libPhoneRuleSetService.checkTargetNumber(phoneNumber).isValid();
              if (isValid)
              numberResponse =
↪ dbRuleSetService.checkTargetNumber(phoneNumber);}
```

Wenn die Methode `checkargetNumber()` von libphoneRuleSetService zurückgibt, dass die Nummer gültig ist, werden die Daten zusätzlich mit Informationen in der lokalen Datenbank abgeglichen, die Daten aus ITU, libphonenumber und Twilio Lookup API enthält.

Die lokale Datenbank wird ständig mit neuen Daten erweitert. Die `checkTargetNumber-Methode` in dbRuleSetService (siehe Listing 4) verwendet die libphonenumber-Bibliothek, um die Nummer zu analysieren und die Länge des NDC, N(S)N, des Nummerntyps zu ermitteln. Mit extrahierten Informationen wird die Methode `findNumberByCountryCodeAndLeadingDigitsOfNSN()` aufgerufen, die daraufhin eine Anfrage an die Datenbank sendet, um Informationen über die abgefragte Nummer zu erhalten. Wenn eine solche Nummer in der Datenbank nicht gefunden wird, wird eine neue Nummer erstellt, und die Datenbank wird auf diese Weise erweitert und aktualisiert (siehe Listing 4).

Der Prüfmodus 4 verwendet beide oben genannten Sicherheitsüberprüfungsmethoden (siehe Listing 9), geht aber zusätzlich über Twilio SDK, um Informationen über eine Telefonnummer mit Twilio Lookup API abzufragen (siehe Listing 2).

```
    case 4 -> {
              boolean isValid = isTargetNumberValid(phoneNumber);
              numberResponse = ModifiableNumberResponse.create()
                    .countryCode(phoneNumber.getCountryCode())
                    .nationalNumber(phoneNumber.getNationalNumber())
                    .isValid(isValid);}
```

Der Unterschied zwischen dem Prüfmodus 4 und den anderen Prüfmodi besteht darin, dass im Prüfmodus 4 die Methode `checkTargetNumber()` der einzelnen Dienste nicht einzeln aufgerufen wird (siehe Listing 9). Stattdessen wird die Methode `checkTargetNumber()` des Hauptdienstes aufgerufen, der später jede Implementierung des RuleSetService durchläuft (siehe Listing 2).

```
    //other parts are omitted
      for (RuleSetService r : ruleSetService)
          validNumber = r.checkTargetNumber(numberProto).isValid();
```

Das heißt, im Prüfmodus 4 dient der Hauptdienst als Orchestrator, der die Aufgabe der Überprüfung der Zielnummer an jede Implementierung des RuleSetService delegiert. Dieser Ansatz kann effizienter und skalierbarer sein als der Aufruf der `checkTargetNumber()-Methode` jedes einzelnen Dienstes, insbesondere bei einer großen Anzahl von Diensten.

Insgesamt verdeutlicht dieser Implementierungsunterschied die Flexibilität und die Anpassungsmöglichkeiten des Prüfverfahrens, die es erlauben, je nach den spezifischen Bedürfnissen und Anforderungen der Anwendung unterschiedliche Ansätze zu verwenden.

2.4 Entwurf und Entwicklung

Im folgenden Kapitel wird der Entwicklungsprozess vorgestellt, einschließlich der Erstellung des Projekts und der Anforderungen, die das Projekt erfüllen muss. Das Kapitel ist in vier Abschnitte unterteilt, nämlich die Erstellen des SMS-Gatekeepers, Projektinfrastruktur und Datenbank, Testen sowie Berichterstattung und Überwachung.

Der erste Abschnitt dieses Kapitels zeigt die Erstellung des SMS-Gatekeepers auf, einschließlich der verwendeten Technologie und der Anforderungen, die die erstellte Infrastruktur und die Anwendung selbst erfüllen müssen.

Der zweite Abschnitt dieses Kapitels konzentriert sich auf die Projektinfrastruktur. Vom Webserver und der Datenbank bis hin zum Deployment auf Kubernetes. Außerdem wird in diesem Abschnitt beschrieben, welche Pakete und Verzeichnisse SMS-Gatekeeper enthält.

Der nächste Abschnitt dieses Kapitels zeigt den Testprozess der Anwendung. Die wichtigsten Details wie das verwendete Testframework, die Vorbereitung der Testumgebung und die Konfiguration der Testcontainer werden erläutert.

Der letzte Abschnitt dieses Kapitels befasst sich mit der Evaluierung des neuen Dienstes, insbesondere mit dem Berichterstattungs- und Überwachungsverfahren. Die wichtigsten Komponenten wie Spring Boot Metrics, Micrometer Tracing mit Opentelemetry und Logging werden besprochen.

Diese vier Abschnitte sind wesentlich für den Aufbau einer zuverlässigen Umgebung für den SMS-Gatekeeper, um den Dienst testen, überwachen und melden zu können, da diese Komponenten ein klares Ergebnis des erreichten Ziels liefern. Die detaillierte Untersuchung der Rolle der einzelnen Abschnitte wird in den Abschnitten selbst weiter erörtert.

Insgesamt war die Entwurf und Entwicklung der SMS-Gatekeeper-Komponente ein entscheidender Teil dieser Studie. Durch die Verwendung branchenüblicher Sicherheitspraktiken, eines robusten Rahmens und einer skalierbaren Infrastruktur war es möglich, einen sicheren und zuverlässigen SMS-Gatekeeper zu entwickeln, der die Anforderungen des Unternehmens erfüllt.

2.4.1 Erstellen des SMS-Gatekeepers

Für die Erstellung des neuen Dienstes wurde das Framework Spring Boot[2] gewählt. Die Gründe für die Wahl von Spring Boot werden im Folgenden erläutert.

Spring[3] ist ein Open-Source-Framework, das zur Entwicklung von Enterprise Java-Anwendungen verwendet wird. Es wurde im Jahr 2003 veröffentlicht. Das Hauptziel von Spring Framework war es, den Entwicklungsprozess zu vereinfachen. Innerhalb des Spring Frameworks gibt es eine Reihe weiterer Projekte. Eines dieser Projekte

[2]Spring Boot `https://spring.io/projects/spring-boot`

[3]Spring Framework `https://spring.io/`

ist Spring Boot. Die Geschichte von Spring Boot begann 2013 mit einem in JIRA erstellten Ticket [4].

Die wichtigsten Eigenschaften von Spring Boot sind die Bereitstellung von Starter-Abhängigkeiten zur Vereinfachung der Build-Konfiguration und die Möglichkeit, Standalone-Anwendungen mit eingebetteten Webservern zu erstellen.

Aber Spring Boot hat nicht nur Vorteile, sondern auch Nachteile. Einer davon ist, dass Spring Boot Abhängigkeiten automatisch konfiguriert und es daher schwierig sein kann, die inner Funktionsweise von Spring zu verstehen.

Was flatexDEGIRO betrifft, so erfüllt Spring Boot alle Anforderungen, die das Unternehmen stellt.

- Da Spring Boot stand-alone, produktionsreife Anwendungen bietet, kann es einfach in die bestehende Entwicklungsinfrastruktur integriert werden.
- Da Spring Boot eine hervorragende Unterstützung für den Aufbau von RESTful Web Services bietet, wäre es einfach, eine Verbindung zu anderen Diensten herzustellen, die dieselbe REST API anbieten, ohne ihre Implementierung zu ändern.
- Da Spring Boot verschiedene Sicherheitsprotokolle und -standards wie OAuth2 und JSON Web Tokens (JWT) unterstützt, gibt es viele Optionen zur Verfügung.
- Da Spring Boot zahlreiche Sicherheitskonfigurationsoptionen und -werkzeuge auf Klassen- und Methodenebene bietet, um die Anwendung zu sichern, eignet es sich gut für die Unternehmensanforderungen.
- Außerdem ist das Spring-Framework flexibel, schnell und produktiv – genau die Eigenschaften, die für das Unternehmen wichtig sind.

Der Hauptfaktor für die Wahl der Technologie für die neue Anwendung war, dass der neue Dienst ein Vermittler zwischen SMS-Gateway und anderen Diensten sein musste. Es wird erwartet, dass das endgültige Projekt eine stand-alone, produktionsreife Anwendung sein wird. Aus den oben genannten Gründen erfüllt Spring Boot alle Anforderungen.

Zur Initialisierung eines neuen Spring-Projekts wurde das vorinitialisierte Projekt namens Spring Initializr [5] verwendet. Eine ZIP-Datei des Projekts mit den entsprechenden Abhängigkeiten wurde erstellt und zur weiteren Verwendung heruntergeladen.

Nachdem die Anforderungen für das Framework festgelegt sind, wurden in einem nächsten Schritt die Anforderungen für die Anwendung selbst ermittelt. Diese Anforderungen wurden mit dem Ziel entwickelt, sicherzustellen, dass die Anwendung

[4]JIRA Ticket `https://jira.spring.io/browse/SPR-9888?redirect=false`

[5]Spring Initializr `https://start.spring.io/`

alle vordefinierten Akzeptanzkriterien erfüllt. Der neue Dienst, im Folgenden SMS-Gatekeeper genannt, muss die folgenden Anforderungen erfüllen:

- Funktionalität
  - Der SMS-Gatekeeper muss die gleiche REST API wie das SMS-Gateway bereitstellen.
  - Der SMS-Gatekeeper muss Anfragen auf der Grundlage der oben genannten Akzeptanzkriterien filtern und mit geeigneten Antworten weiterleiten.
  - Der SMS-Gatekeeper sollte die Nachrichten anhand der Telefonnummer im Zwischenspeicher halten können und entsprechende Antworten geben.
  - Der SMS-Gatekeeper soll die Suche nach Telefonnummern, Providern und Nachrichten ermöglichen.
- Sicherheit
  - Der SMS-Gatekeeper sollte nur autorisierten Benutzern den Zugriff auf das System und die Durchführung bestimmter Aktionen ermöglichen.
  - Der SMS-Gatekeeper muss ebenfalls dieselbe Authentifizierung verwenden wie das SMS-Gateway.
  - Der SMS-Gatekeeper muss über separate sichere Endpunkte für die Operatoren des SMS-Gatekeepers verfügen, um die erforderlichen Daten zu ändern (z. B. Sperren eines Bereichs von Nummern auf der Grundlage von CC und NDC).
  - Der SMS-Gatekeeper soll Endpunkte je nach Rolle schützen. Wenn der Benutzer CCI ist, dann werden die Anmeldedaten mit Basic-Authentifizierung überprüft, wenn der Benutzer Operator der Anwendung ist, dann mit API-Schlüssel.
- Skalierbarkeit
  - Der SMS-Gatekeeper soll fähig sein, eine große Anzahl von eingehenden Anfragen ohne Leistungseinbußen zu verarbeiten.
  - Der SMS-Gatekeeper sollte über Leistungsüberwachungsfunktionen verfügen, um eine Echtzeitüberwachung der Systemleistung zu ermöglichen und mögliche Leistungsengpässe zu ermitteln.
- Verlässlichkeit
  - Der SMS-Gatekeeper muss hochverfügbar und widerstandsfähig sein und sich schnell von Ausfällen erholen können.
  - Die Tests des SMS-Gatekeepers sollten unabhängig von Umgebungsänderungen sein und ohne Fehler laufen.
  - Der SMS-Gatekeeper soll fähig sein, mit Fehlern umzugehen, wenn Fehler auftreten, und Benachrichtigungen an die Operatoren zu senden.

- Flexibilität
  - Die aktuellen Dienste, die mit dem SMS-Gateway zusammenarbeiten, müssen ohne Implementierungsänderungen an den SMS-Gatekeeper gebunden werden.
  - Der SMS-Gatekeeper sollte über Anwendungseigenschaften angepasst und konfiguriert werden können, um spezifische Geschäftsanforderungen zu erfüllen.
  - Der SMS-Gatekeeper sollte für die weitere Unterstützung und Entwicklung des Dienstes einfach einzustellen und zu verwalten sein.
- Auditierbarkeit
  - Der SMS-Gatekeeper sollte Logging- und Auditing-Fähigkeiten besitzen, um eine effektive Verfolgung und Analyse der Anfrageflüsse zu ermöglichen.
  - Der SMS-Gatekeeper sollte verteiltes Tracing verwenden.
  - Der SMS-Gatekeeper sollte nicht eng an bestimmte Tracing- und Logging-Dienstleister gebunden sein, sondern unabhängig sein und ein Standardsystem verwenden.

Es ist wichtig, darauf hinzuweisen, dass dies nicht die einzigen Anforderungen an den SMS-Gatekeeper sind, und dass je nach den spezifischen Anforderungen des Projekts zusätzliche Anforderungen erforderlich sein können.

2.4.2 Projektinfrastruktur und Datenbanken

Der SMS-Gatekeeper als Spring Boot-Anwendung besteht aus einer Reihe von Schlüsselkomponenten, darunter ein Webserver, eine Datenbank, ein ORM-Werkzeug. Darüber hinaus stützt sich die Anwendung auf ein ein Build-Werkzeug und ein System zur kontinuierlichen Integration und Bereitstellung (CI/CD), um den Entwicklungsprozess effizient zu verwalten. Darüber hinaus wird die Container-Orchestrierung-Plattform für die Bereitstellung und Verwaltung der Anwendung in einer Produktionsumgebung verwendet.

Der Webserver ist unter anderem für die Bearbeitung eingehender Anfragen zuständig. Als Webserver wurde ein eingebetteter Tomcat-Server verwendet, der in das Spring Boot-Framework integriert ist und keine separate Installation erfordert.

Die Datenbank wird zum Speichern und Verwalten der Daten für die Anwendung verwendet. Als Datenbank wurde PostgreSQL[6] verwendet, ein leistungsstarkes und weit verbreitetes quelloffenes Datenbankmanagementsystem. Für die Verwaltung von Änderungen am Datenbankschema wurde Liquibase [7] verwendet, das dabei hilft, Datenbankänderungen in einer konsistenten und organisierten Weise zu verfolgen und anzuwenden.

---

[6]PostgreSQL `https://www.postgresql.org/`

[7]Liquibase `https://www.liquibase.org/`

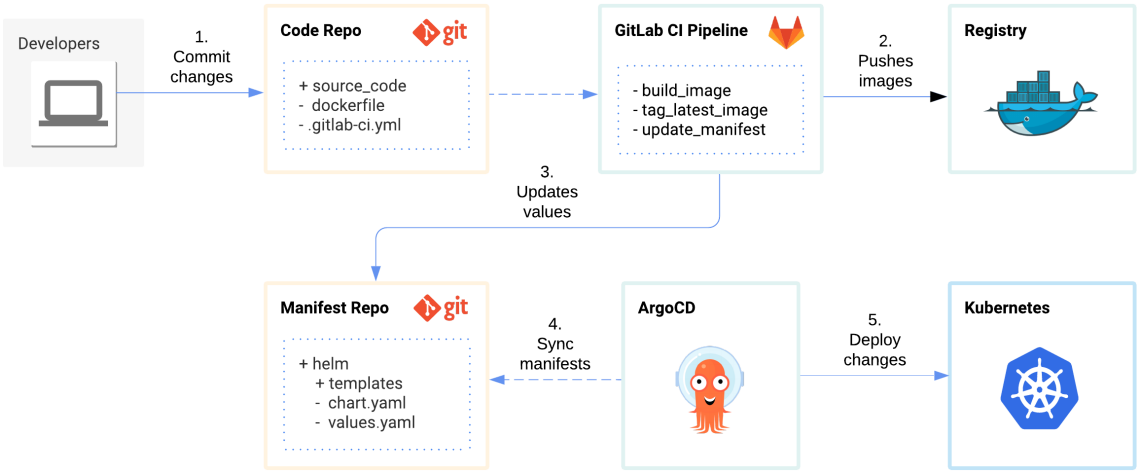
Das ORM-Werkzeug wird verwendet, um Java-Objekte auf Datenbankeinträge abzubilden und umgekehrt. Als ORM-Werkzeug wurde Hibernate [8] gewählt, da es auf objektorientierten Grundsätzen und bewährten Verfahren basiert, ohne sich um die Datenbanksemantik zu kümmern.

Das Build-Werkzeug dient der Verwaltung von Abhängigkeiten, dem Build und dem Testen der Anwendung. Als Build-Manager wurde Maven verwendet, da es die Verwaltung von Abhängigkeiten durch Kontrolle der Versionen der verwendeten Abhängigkeiten ermöglicht.

Zur Automatisierung des Build- und Deployment-Prozesses wurde GitLab[9] CI/CD eingesetzt, das sich in das Versionskontrollsystem des Unternehmens (GitLab) integriert, um SMS-Gatekeeper automatisch zu bauen, zu testen und zu verteilen, sobald neuer Code in das Repository eingestellt wird.

Zur Containerisierung von SMS-Gatekeeper wurde Docker[10] verwendet, das es ermöglicht, die Anwendung und ihre Abhängigkeiten in ein in sich geschlossenes Docker-Image zu verpacken, das leicht in jeder Umgebung bereitgestellt werden kann. Und für die Verwaltung von Bereitstellungen in Kubernetes[11] wurde Helm verwendet, das einen vorlagenbasierten Ansatz für die Definition und Bereitstellung komplexer Anwendungen im Kubernetes-Cluster bietet.

Schließlich dient Kubernetes als Container-Orchestrierung-Plattform für SMS-Gatekeeper. Kubernetes bietet eine hochskalierbare und zuverlässige Infrastruktur für die Bereitstellung und Verwaltung von containerisierten Anwendungen. Der vollständige Arbeitsablauf ist in Abbildung 8 zu sehen.



Quelle: How to use GitOps with ArgoCD [40]

Abbildung 8: GitOps Arbeitsablauf

[8]Hibernate `https://hibernate.org/`

[9]GitLab `https://about.gitlab.com/`

[10]Docker `https://www.docker.com/`

[11]Kubernetes `https://kubernetes.io/`

Die Projektinfrastruktur für SMS-Gatekeeper ist in einer Reihe von Paketen und Verzeichnissen organisiert. Zu den wichtigsten Paketen für die Anwendung gehören:

- com.flatex.pfs.smsgatekeeper
  - Dies ist das Hauptpaket für die Anwendung, das die Hauptklasse und alle anderen Klassen und Ressourcen der obersten Ebene enthält.
- com.flatex.pfs.smsgatekeeper.config
  - Dieses Paket enthält Konfigurationsklassen für die Anwendung, einschließlich Beans, Eigenschaften und andere Einstellungen.
- com.flatex.pfs.smsgatekeeper.(class)
  - Die Klassen (wobei class der letzte Ordner in der Hierarchie ist) werden in gleichnamigen Ordnern abgelegt. Jeder Ordner ist wiederum in drei Hauptordner unterteilt: Boundary (Benutzeroberfläche, Controller-Klassen), Service (Service-Klassen) und Domain (Entity-Klassen und Speicherschnittstellen), entsprechend dem Entity-Boundary-Controller (EBC)-Muster.

Neben der Pakethierarchie umfasst die Projektinfrastruktur eine Reihe weiterer Verzeichnisse, darunter:

- `src/main/java`
  - Dieses Verzeichnis enthält den Java-Quellcode für die Anwendung.
- `src/main/resources`
  - Dieses Verzeichnis enthält die Ressourcen für die Anwendung, wie z. B. Konfigurationsdateien, das Änderungsprotokoll der Migration und statische Assets.
- `src/test/java`
  - Dieses Verzeichnis enthält den Testcode für die Anwendung.
- `target`
  - Dieses Verzeichnis wird durch den Build-Prozess erzeugt und enthält den kompilierten Anwendungscode und andere Build-Artefakte.

Die Infrastruktur bietet eine robuste und skalierbare Grundlage für das Projekt, um Daten effizient zu verwalten und zu speichern, Datenbankänderungen zu verfolgen, die Anwendung zu erstellen und bereitzustellen und Container zu orchestrieren, wenn die Forschungsziele dieses Forschungspapiers erreicht sind.

2.4.3 Testen

Um zu evaluieren, ob der SMS-Gatekeeper die spezifizierten Anforderungen erfüllt und wie vorgesehen funktioniert, wurden Akzeptanztests durchgeführt. Die Akzeptanztests wurden mit dem Cucumber-Framework implementiert, das ein Werkzeug zum Schreiben und Ausführen von automatisierten Akzeptanztests ist.

Um Cucumber und JUnit5 für die Akzeptanztests zu konfigurieren, wurden die Cucumber- und JUnit5-Bibliotheken und alle erforderlichen Abhängigkeiten zur Build-Datei des Projekts hinzugefügt. Außerdem wurden eine Testkonfigurationsdatei (siehe Listing 12) und eine Konfigurationsklasse (siehe Listing 11) eingerichtet, die die Cucumber-Optionen und die JUnit5-Konfiguration für die Tests festlegen.

Die Akzeptanzkriterien für den SMS-Gatekeeper wurden definiert, um sicherzustellen, dass die Anwendung die festgelegten Anforderungen erfüllt. Zu den Akzeptanzkriterien für die Tests gehörten die folgenden:

- Der SMS-Gatekeeper sollte gesund und bereit sein, so dass die an `/healthz` und `/readiness` gesendeten Anfragen den HTTP-Status 200-OK zurückgeben sollten.
- Der SMS-Gatekeeper sollte in der Lage sein, alle Anfragen zu bearbeiten, die zuvor an das SMS-Gateway-Projekt gesendet wurden, und die Anfragen nach den im Kapitel Mustererkennung genannten Akzeptanzkriterien zu filtern.
- Der SMS-Gatekeeper sollte kubernetes-ready sein.
- Der SMS-Gatekeeper sollte vor unberechtigtem Zugriff geschützt sein und alle relevanten Sicherheitsstandards und -vorschriften erfüllen.

Die definierten Akzeptanzkriterien dienten als Leitfaden für die Entwicklung und das Testen des SMS-Gatekeeper und wurden während der Testphase bewertet. Es ist wichtig, darauf hinzuweisen, dass dies nicht die einzigen Anforderungen an den SMS-Gatekeeper sind, und dass je nach den spezifischen Anforderungen des Projekts zusätzliche Testanforderungen erforderlich sein können.

Die Testszenarien wurden in einer natürlichsprachlichen Syntax namens Gherkin [12] geschrieben. Die Testszenarien wurden in Funktionsdateien organisiert, die dann in Schrittdefinitionen verwendet wurden.

Die Testszenarien umfassten Folgendes:

- Health and Readiness Test: Dieses Testszenario wurde entwickelt, um zu überprüfen, ob der SMS-Gatekeeper einsatzbereit ist und so funktioniert, wie er soll. Das Testszenario wurde in Gherkin-Syntax wie folgt geschrieben:

```
Feature: Health and Readiness can be handled
Scenario: client makes call to GET /healthz
  When the client calls /healthz
  Then the client receives status code of 200
```

[12] Gherkin `https://cucumber.io/docs/gherkin/reference/`

- Test des Anfrageflusses zu `/send`: Mit diesem Test sollte überprüft werden, ob der Sms-Gatekeeper die eingehenden Anfragen anhand der Akzeptanzkriterien verarbeiten und filtern kann. Das Testszenario wurde in Gherkin-Syntax wie folgt geschrieben:

```
    Scenario: client makes call to POST /send
      Given a request body is passed
      When the client calls /send
      Then the message should not be empty
      Then If the country is on the blacklist no SMS is sent
      Then If the number is in the locked list no SMS is sent
      Then If the country is not on the whitelist, nor on the
↪  blacklist, a captcha is required
      Then If more than 5 SMS have been sent to the number in the
↪  last 24 hours, a captcha is required
      Then If more than 5 SMS have been sent from the requesting IP
↪  in the last 24 hours, a captcha is required
      Then the phone number must be valid
      Then If the number of SMS to a cc deviate by 30% compare to
↪  the daily average of the last n days, an alert is triggered
      Then the client receives status code of 200
```

Um einige Testfälle zu testen, war es notwendig, eine Datenbank und eine echte Webumgebung einzurichten. Es handelte sich also nicht nur um Akzeptanztests, sondern um eine Mischung aus Akzeptanztests und Integrationstests. Aus diesem Grund wurden die Testcontainer verwendet, um eine Datenbank und eine spezielle Einrichtung der Testumgebung zu starten.

Der Zweck der Verwendung von Testcontainers bestand darin, eine konsistente und zuverlässige Umgebung für die Tests zu schaffen und das Testen der Anwendung mit unterschiedlichen Konfigurationen und Abhängigkeiten zu erleichtern.

Um Testcontainers für die Tests zu konfigurieren, wurden die Testcontainers-Bibliothek und alle erforderlichen Abhängigkeiten zur Build-Datei des Projekts hinzugefügt. Außerdem wurde eine Testkonfigurationsdatei erstellt, in der die Docker-Images und -Container angegeben wurden, die für die Tests verwendet werden sollten.

Bei der Konfiguration war es schwierig, Container so zu konfigurieren, dass sie das richtige Docker-Image aus dem Nexus-Repository des Unternehmens ziehen. Die Docker-Image-Namenssubstitution muss verwendet werden. Nach der Dokumentation von Testcontainers [32] wurde das `hub.image.name.prefix` in der Datei `testcontainers.properties` (siehe Listing 13) konfiguriert, um das richtige Docker-Image für die Initialisierung des postgreSQL-Containers zu ziehen.

```
hub.image.name.prefix=NEXUS_LIBRARY_PostgreSQL
```

```
@Container
    private static final PostgreSQLContainer<?> postgresqlContainer =
            new PostgreSQLContainer<>("postgres");
```

Um die Testumgebung zu konfigurieren, war die zusätzliche Konfiguration der `CucumberAcceptanceTests-Klasse` (siehe Listing 12) notwendig. Die Test-Annotation `@SpringBootTest` startet automatisch nicht die webEnvironment, um die Testausführung weiter zu verfeinern. Es gibt mehrere Optionen [27]:

1. `MOCK(Standard)`
2. `RANDOM_PORT`
3. `DEFINED_PORT`
4. `NONE`

Die Option `RANDOM_PORT` lädt einen WebServerApplicationContext und bietet eine echte Webumgebung. Der eingebettete Server wird gestartet und lauscht auf einem zufälligen Port.

Zur Ausführung der Akzeptanztests wurde ein maven-surefire Plugin verwendet, da Maven standardmäßig (als Teil des Standard-Builds) Cucumber-Tests überspringt. Die Ergebnisse der Tests wurden in einem für Menschen lesbaren Format sowohl im Terminal als auch im Cucumber-Report (siehe Abbildung A.2) angezeigt, einschließlich aller aufgetretenen Ausfälle oder Fehler. Insgesamt haben die mit dem Cucumber-Framework implementierten Akzeptanztests gezeigt, dass der SMS-Gatekeeper die spezifizierten Anforderungen erfüllte und wie vorgesehen funktionierte.

2.4.4 Berichterstattung und Überwachung

Zur Überwachung der Leistung und Nutzung von SMS-Gatekeeper wurde eine Kombination von Spring Boot-Metriken mit Actuator, Prometheus, Tempo, Loki und Grafana verwendet.

Actuator - ist ein Spring Boot-Modul, das verschiedene Endpunktdienste wie Health Checks, Auditing und Metriken bereitstellt, die zur Überwachung und Verwaltung von Anwendungen verwendet werden können.

Prometheus - ist ein beliebtes quelloffenes Überwachungs- und Alarmierungstool, das Metriken aus verschiedenen Quellen, einschließlich Actuator-Endpunkten, abrufen und speichern kann.

Tempo - ist ein quelloffenes, verteiltes Tracing-Backend, das hoch skalierbar, einfach zu bedienen und herstellerneutral ist. Es bietet Beobachtbarkeit und Leistungsüberwachungsfunktionen für große verteilte Systeme und Microservices-Architekturen.

Loki - ist ein quelloffenes, horizontal skalierbares Log-Aggregationssystem, das für die Speicherung und Abfrage großer Mengen von Log-Daten entwickelt wurde. Es ist für Kubernetes und andere Cloud-native Architekturen optimiert und in die Grafana-Plattform zur einfachen Visualisierung und Analyse von Protokolldaten integriert.

Grafana - ist ein Visualisierungstool, mit dem Dashboards und Diagramme auf der Grundlage der von Prometheus erfassten Metriken erstellt werden können.

Der SMS-Gatekeeper wurde so konfiguriert, dass Actuator-Endpunkte offengelegt werden, so dass Prometheus diese Endpunkte nach Metriken durchsuchen kann.

Der Metrikbereich von Spring Boot bietet nicht nur standardmäßig eingebaute Metriken wie die Anzahl der empfangenen HTTP-Anfragen, die Anzahl der ausgelösten Ausnahmen und die Menge des verwendeten Speichers und der CPU, sondern auch die Möglichkeit, benutzerdefinierte Metriken zu konfigurieren.

Um benutzerdefinierte Metriken in einer Spring Boot-Anwendung zu erstellen, kann die `MeterRegistry` (siehe Listing 10) verwendet werden, die Methoden zum Erstellen und Registrieren verschiedener Arten von Metriken, wie Counter oder Gauge, bereitstellt. Diese Metriken sind in der Lage, bestimmte Ereignisse oder Werte in der Anwendung zu verfolgen, wie z.B. die Anzahl der Aufrufe einer bestimmten Methode oder die Zeit, die für die Ausführung einer Aufgabe benötigt wird.

Im SMS-Gatekeeper wurde ein Counter erstellt, der die Anzahl der Anfragen pro Land zählt. In dieser Methode wird ein neuer benutzerdefinierter Counter mit dem Namen „`NAMESPACE`_name“ und dem Tag „country“ erstellt.

```
    private Counter initNewCountryCounter(String tag) {
    return Counter.builder(NAMESPACE + "country_code")
            .tag("country", tag)
            .description("Total number of requests per country_code")
            .register(meterRegistry);
    }
```

Die `countCountry-Methode` erhält einen CC und initialisiert bei jedem Aufruf einen neuen Länderzähler oder erhöht den vorhandenen Counter, so dass die Anzahl der Anfragen pro Land verfolgt werden kann.

```
    @Override
    public void countCountry(int countryCode) {
    Counter cCounter = initNewCountryCounter(String.valueOf(countryCode));
    cCounter.increment();
    }
```

In den Metriken ist zu erkennen, dass beispielsweise von CC 49 zwei Anfragen und von CC 43 eine Anfrage gesendet wurde

```
# TYPE sms_gatekeeper_country_code_total counter
sms_gatekeeper_country_code_total{application="sms-gatekeeper",country="43"
,} 1.0
sms_gatekeeper_country_code_total{application="sms-gatekeeper",country="49"
,} 2.0
```

Außerdem wurde eine Reihe von Metriken erstellt, die messen, wie viele Anfragen pro IP und pro spezifischem Mobilfunkanbieter gesendet wurden. Die ausgewerteten

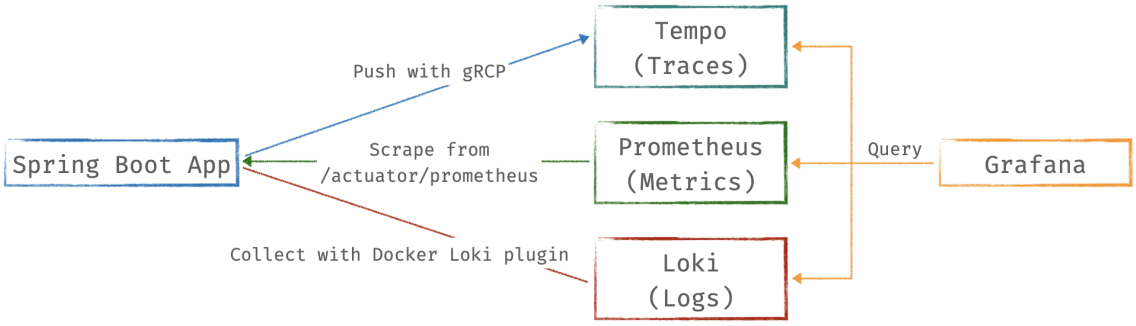
Ergebnisse wurden in das Grafana Dashboard übertragen, so dass sie visuell gut dargestellt werden können.

Zurzeit wird in der Firma für die Tracing und das Loggen Graylog Open verwendet. Diese Lösung war ineffektiv und nicht benutzerfreundlich. Infolgedessen war die Überwachung der Leistung und des Verhaltens der Systeme des Unternehmens eine Herausforderung. Deshalb bestand die Absicht, auf eine bessere Lösung umzusteigen, insbesondere für den Zugriff auf die Logdateien. Während des Teamleiter-Treffens des Unternehmens wurde beschlossen, OpenTelemetry als neuen Standard für alle Bereiche (Traces, Metriken und Logs) zu verwenden.

OpenTelemetry - ist eine Reihe von quelloffenen Werkzeugen und APIs für die Erfassung, Verarbeitung und den Export von Telemetriedaten, einschließlich Tracing, Metriken und Logs. OpenTelemetry bietet eine standardisierte Methode zur Instrumentierung von Anwendungen und Diensten für die Ablaufverfolgung und ermöglicht es Entwicklern, Ablaufverfolgungsdaten aus verschiedenen Quellen auf einheitliche Weise zu sammeln und zu analysieren. OpenTelemetry unterstützt eine Reihe von Sprachen und Plattformen, darunter Java, Go, Python und Node.js, und bietet eine Vielzahl von Integrationen mit anderen Überwachungs- und Beobachtungstools.

Im SMS-Gatekeeper wurde Micrometer-Tracing mit OpenTelemetry implementiert, um Tracing-Daten über das verteilte System, einschließlich CCI und SMS-Gateway, zu sammeln. OpenTelemetry wurde verwendet, um jeden Dienst innerhalb der Anwendung zu instrumentieren, so dass das Unternehmen detaillierte Tracing-Daten für jede Anfrage sammeln kann, während die Anfrage durch das System läuft. Diese Tracing-Daten werden durch das OpenTelemetry SDK gesammelt, das die Daten dann in einen Standard-OTLP Exporter exportiert, der es später ermöglicht, die Daten an eine Vielzahl von Überwachungs- und Beobachtungstools wie Grafana und Prometheus zu senden.

Das Ziel war es, den folgenden Aufbau (siehe Abbildung 9) zu erreichen, der am Ende zu einer Spring Boot Observability mit drei Hauptkomponenten führte.



Quelle: Spring Boot with Observability [39]

Abbildung 9: Spring Boot Observability

Durch die Implementierung von OpenTelemetry war es möglich, tiefere Einblicke in das Verhalten des SMS-Gatekeepers zu gewinnen und Probleme und zu beheben, sobald sie auftreten. Die Verwendung von Standard-OTLP Exportern stellt außerdem

sicher, dass die Tracing-Daten mit einer breiten Palette von Überwachungs- und Beobachtungstools kompatibel sind, so dass die Tools verwendet werden können, die den Anforderungen des Unternehmens am besten entsprechen.

In der Abbildung 10 ist die Dauer des `/send` Endpunkts von Tempo in der lokalen Testumgebung zu sehen. Die Daten werden zu Grafana exportiert.

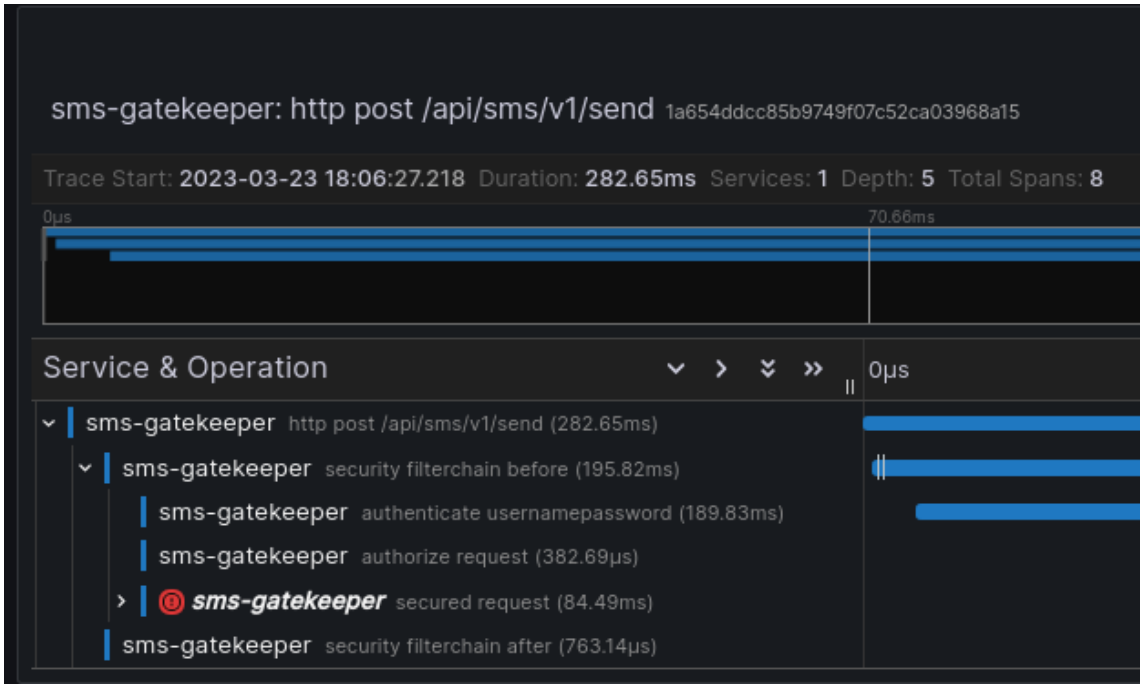


Abbildung 10: Trace-Daten aus dem Tempo visualisiert in Grafana

Insgesamt ist der Einsatz von verteiltem Tracing mit OpenTelemetry eine wertvolle Ergänzung der Überwachungs- und Beobachtungsstrategie des Unternehmens, die dazu beiträgt, die Zuverlässigkeit und Leistung verteilter Anwendungen zu erhalten.

Darüber hinaus umfasst die Berichtsfunktion vom SMS-Gatekeeper die Erstellung von statistischen Berichten, die auf Anforderung des Operators oder nach einem festgelegten Zeitplan im csv-Format exportiert werden. Der Prozess der Erstellung und des Exports dieser Berichte läuft wie folgt ab:

1. Die Berichtsfunktion wird durch einen geplanten Job ausgelöst, der in einem bestimmten Intervall läuft, z. B. einmal pro Woche oder einmal im Monat.
2. Wenn der Job läuft, sammelt er die erforderlichen Daten aus verschiedenen Quellen innerhalb des SMS-Gatekeepers, z. B. aus Logdateien und Leistungsmetriken.
3. Die gesammelten Daten werden verarbeitet und analysiert, um statistische Berichte zu erstellen, die Aufschluss über die Nutzung und Leistung des Dienstes im Laufe der Zeit geben. Dazu können Kennzahlen wie die Anzahl der bearbeiteten Anfragen, die Antwortzeit für jede Anfrage und die Anzahl der aufgetretenen Fehler gehören.
4. Sobald die Berichte erstellt sind, werden sie mit Hilfe einer internen Bibliothek von flatexDEGIRO in das PDF-Format umgewandelt.

5. Schließlich werden die PDF-Berichte an eine E-Mail angehängt und an eine bestimmte Liste von Empfängern, z. B. Service-Administratoren oder Geschäftsinteressenten, gesendet.

Außerdem ermöglicht die Berichtsfunktion die Erstellung benutzerdefinierter Berichte für bestimmte Kunden innerhalb eines bestimmten Zeitraums. Diese Funktionalität wird dadurch erreicht, dass der Operator Parameter wie den Kundennamen, das Jahr und den Monat eingeben kann, um einen auf seine Bedürfnisse zugeschnittenen Bericht zu erstellen.

Insgesamt sind Berichterstattung und Überwachung wichtige Werkzeuge für das Verständnis der Nutzung des Dienstes und der Leistung von SMS-Gatekeeper. Durch die Implementierung von Berichts- und Überwachungstools wie E-Mail-Warnungen und statistische Berichte können Administratoren Probleme schnell erkennen und Korrekturmaßnahmen ergreifen, um eine optimale Serviceleistung zu gewährleisten.

Kapitel 3

Fazit

3.1 Zusammenfassung

Abschließend wird in dieser Arbeit eine Lösung für das Sicherheitsproblem der Angriffe auf das System der Online-Brokerage-Plattform in Europa vorgestellt. Die Studie konzentriert sich auf die Konzeption und Implementierung einer neuen Anwendung, genannt SMS-Gatekeeper, zum Schutz vor betrügerischem Verhalten während des Onboarding-Prozesses von Neukunden. Der neue Dienst zielt darauf ab, die Kosten des Unternehmens zu senken, indem er Angriffe aufspürt und verhindert, die die SMS-Rechnung des externen Anbieters für den Versand von Nachrichten erhöhen.

Der SMS-Gatekeeper wurde so konzipiert, dass er sich leicht in das bestehende System integrieren lässt und gleichzeitig einen wirksamen Schutz bietet. In dieser Arbeit wird der Implementierungsprozess Schritt für Schritt beschrieben, einschließlich der Entwicklung eines Algorithmus zur Filterung von Anfragen auf der Grundlage der Zielnummer. Der mehrstufige Sicherheitsalgorithmus ist ein entscheidender Aspekt des entwickelten Schutzmechanismus. Der Algorithmus umfasst die Verwendung der Bibliothek libphonenumber, des Twilio SDK und einer lokalen Datenbank, um sicherzustellen, dass mehrere Sicherheitsebenen implementiert werden, um einen verbesserten Schutz gegen potenzielle Angriffe zu bieten und die Kundenregistrierung auf der Grundlage der vorgegebenen Akzeptanzkriterien zu gewährleisten.

Einer der Schwachpunkte der Anwendung ist, dass es keine Möglichkeit gibt, zu überprüfen, ob die Nummernübertragbarkeit zu einer Telefonnummer genutzt wurde oder nicht. Die Lösung dieses Problems könnte ein weiterer Schritt für die Weiterentwicklung des Projekts sein. Ein weiterer Punkt, über den der Autor besorgt ist, ist die Unfähigkeit, genaue IP-Adressen zu überprüfen, wenn die Nummer, die die SMS sendet, ein VPN verwendet, was ebenfalls weitere Entwicklung erfordert.

Eine der Schwierigkeiten, auf die man bei der Entwicklung der Anwendung stieß, war die Beschaffung von Daten, um die lokale Datenbank zu füllen. Diese Schwierigkeit erwies sich jedoch auch als Vorteil der Anwendung, da die Anwendung Nummern aus fast der ganzen Welt überprüfen kann, wodurch der SMS-Gatekeeper auch für einen größeren Markt skalierbar ist.

Insgesamt bietet der SMS-Gatekeeper eine effektive Lösung für das Sicherheitsproblem der Angriffe auf das System der Online-Broker-Plattform. Diese Anwendung erhöht die Sicherheit des Unternehmens und verringert gleichzeitig die finanziellen Verluste aufgrund von betrügerischem Verhalten. Darüber hinaus ist die Anwendung flexibel und skalierbar, und es könnten weitere Verbesserungen vorgenommen werden, um ihre Fähigkeit für zukünftige Anforderungen zu verbessern.

Literaturverzeichnis

[1] Stefan Bechtold u. a. *JUnit 5 User Guide*. Version 5.9.2. 2023. URL: https://junit.org/junit5/docs/current/user-guide/ (besucht am 30. 03. 2023) (siehe S. v).

[2] Gavin King Christian Bauer. *Java Persistence with Hibernate*. Manning Publications, 2006 (siehe S. vi).

[3] Cloud Native Computing Foundation. *Helm*. 2020. URL: https://www.cncf.io/reports/helm-project-journey-report/ (besucht am 26. 12. 2022) (siehe S. v).

[4] Cucumber. *What is Cucumber?* 2023. URL: https://cucumber.io/docs/guides/overview/#what-is-cucumber (besucht am 30. 03. 2023) (siehe S. iv).

[5] Docker. *Docker Image*. Version n.d. 2022. URL: https://docs.docker.com/engine/reference/commandline/image/ (besucht am 26. 12. 2022) (siehe S. iv).

[6] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (besucht am 12. 12. 2022) (siehe S. vii).

[7] flatexDEGIRO Bank AG. *Homepage von flatexDEGIRO*. 2022. URL: https://flatexdegiro.com/de (besucht am 12. 12. 2022) (siehe S. iv).

[8] Google. *Bibliothek libphonenumber*. Version v8.13.6. 2023. URL: https://github.com/google/libphonenumber (besucht am 15. 12. 2022) (siehe S. 17).

[9] Google Cloud. *API keys*. Version n.d. 2023. URL: https://cloud.google.com/docs/authentication/api-keys (besucht am 13. 03. 2023) (siehe S. iii).

[10] Inc. Graylog. *Graylog*. 2023. URL: https://www.graylog.org/products/source-available/ (besucht am 13. 03. 2023) (siehe S. v).

[11] Jez Humble und David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 2010 (siehe S. iii).

[12] ITU-T. *E.164 : The International Public Telecommunication Numbering Plan*. Techn. Ber. E.164 Suppl.2. International Telecommunication Union, 2016. URL: https://www.itu.int/ITU-T/worksem/ip-telecoms/e164/e164supp2.pdf (besucht am 30. 03. 2023) (siehe S. vi).

[13] *ITU-T Recommendations*. International Telecommunication Union - Telecommunication Standardization Sector. URL: https://www.itu.int/en/ITU-T/publications/Pages/recs.aspx (besucht am 12. 12. 2022) (siehe S. 11).

[14] Jenkins. *Build Automation*. Version n.d. 2023. URL: https://www.jenkins.io/doc/book/pipeline/jenkinsfile/#build-automation (besucht am 29. 01. 2023) (siehe S. iii).

[15] Brendan Burns Kelsey Hightower und Joe Beda. *Kubernetes: Up and Running*. O'Reilly Media, 2018 (siehe S. iii).

[16] Kubernetes. *Kubernetes-Cluster*. 2022. URL: https://kubernetes.io/docs/home/ (besucht am 26. 12. 2022) (siehe S. v).

[17] Paul Clements Len Bass und Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Professional, 2003 (siehe S. iv).
[18] Link Mobility Group ASA. *Homepage von Link Mobility*. 2022. URL: https://www.linkmobility.com/ (besucht am 12. 12. 2022) (siehe S. v).
[19] Micrometer. *Tracing*. Version n.d. 2023. URL: https://micrometer.io/docs/tracing (besucht am 30. 03. 2023) (siehe S. v).
[20] Mozilla Developer Network. *HTTP authentication*. Version n.d. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication (besucht am 13. 03. 2023) (siehe S. iii).
[21] Myra Security GmbH. *Myra*. 2022. URL: https://www.myrasecurity.com/de/ (besucht am 12. 12. 2022) (siehe S. vi).
[22] OpenTelemetry. *OpenTelemetry Protocol Specification*. Version n.d. 2023. URL: https://github.com/open-telemetry/oteps/blob/main/text/0035-opentelemetry-protocol.md (besucht am 30. 03. 2023) (siehe S. vi).
[23] OpenTelemetry. *Traces*. Version n.d. 2023. URL: https://opentelemetry.io/docs/concepts/observability-primer/#distributed-traces (besucht am 30. 03. 2023) (siehe S. vii).
[24] Martin Pol und Van Haren Ruud Teunissen. *Software Testing: A Guide to the TMap Approach*. Pearson Education, 2010 (siehe S. iii).
[25] Prometheus. *METRIC TYPES*. 2022. URL: https://prometheus.io/docs/concepts/metric_types/ (besucht am 20. 01. 2023) (siehe S. iii, iv).
[26] *The JavaScript Object Notation (JSON) Data Interchange Format*. Standard. Internet Engineering Task Force (IETF), 2014 (siehe S. v).
[27] Spring Boot Contributors. *Spring Boot API - SpringBootTest.WebEnvironment*. Version 3.0.0. 2022. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/context/SpringBootTest.WebEnvironment.html (besucht am 29. 12. 2022) (siehe S. 30).
[28] Techopedia. *Integration tests*. 2012. URL: https://www.techopedia.com/definition/7751/integration-testing (besucht am 29. 12. 2022) (siehe S. v).
[29] Techopedia. *Software Framework*. 2023. URL: https://www.techopedia.com/definition/14384/software-framework (besucht am 13. 03. 2023) (siehe S. iv).
[30] Techopedia. *VPN*. 2021. URL: https://www.techopedia.com/definition/4806/virtual-private-network-vpn (besucht am 30. 03. 2023) (siehe S. vii).
[31] Testcontainers. *Testcontainers*. 2023. URL: https://www.testcontainers.org/ (besucht am 30. 03. 2023) (siehe S. vii).
[32] Testcontainers Contributors. *Testcontainers Documentation: Image Name Substitution*. 2022. URL: https://www.testcontainers.org/features/image_name_substitution/ (besucht am 29. 12. 2022) (siehe S. 29).
[33] Twilio Inc. *Twilio SDK*. 2023. URL: https://www.twilio.com/docs/libraries/java (besucht am 15. 12. 2022) (siehe S. vii, 18).
[34] International Telecommunication Union. *About ITU*. 2022. URL: https://www.itu.int/en/about/Pages/default.aspx (besucht am 12. 12. 2022) (siehe S. 10).
[35] International Telecommunication Union. *ITU-T E.101*. 2009. URL: https://handle.itu.int/11.1002/1000/3583 (besucht am 12. 12. 2022) (siehe S. iv, vi, vii).

[36] International Telecommunication Union. *ITU-T E.164*. 2010. URL: https://handle.itu.int/11.1002/1000/10688 (besucht am 12. 12. 2022) (siehe S. iii–vi, 11, 12).
[37] International Telecommunication Union. *What does ITU do?* 2022. URL: https://www.itu.int/en/about/Pages/whatwedo.aspx (besucht am 12. 12. 2022) (siehe S. 10).
[38] Jakarta Bean Validation. *Jakarta Bean Validation*. Version 3.0. URL: https://beanvalidation.org/ (besucht am 13. 03. 2023) (siehe S. v).
[39] Liu Yi-Wei. *Spring Boot Observability*. 2023. URL: https://github.com/blueswen/spring-boot-observability (besucht am 28. 12. 2022) (siehe S. 32).
[40] Poom Wettayakorn. *GitOps in Kubernetes with GitLab CI and ArgoCD*. Aug. 2020. URL: https://medium.com/trendyol-tech/how-to-use-gitops-with-argocd-1782b8493cc3 (besucht am 28. 12. 2022) (siehe S. 26).

Anhang

Bibliothek libphonenumber

```
package com.google.i18n.phonenumbers;
import java.io.Serializable;

public final class Phonenumber {
    private Phonenumber() {
    }

    public static class PhoneNumber implements Serializable {
        private static final long serialVersionUID = 1L;
        private boolean hasCountryCode;
        private int countryCode_ = 0;
        private boolean hasNationalNumber;
        private long nationalNumber_ = 0L;
        private boolean hasExtension;
        private String extension_ = "";
        private boolean hasItalianLeadingZero;
        private boolean italianLeadingZero_ = false;
        private boolean hasNumberOfLeadingZeros;
        private int numberOfLeadingZeros_ = 1;
        private boolean hasRawInput;
        private String rawInput_ = "";
        private boolean hasCountryCodeSource;
        private Phonenumber.PhoneNumber.CountryCodeSource
↪   countryCodeSource_;
        private boolean hasPreferredDomesticCarrierCode;
        private String preferredDomesticCarrierCode_ = "";
}
//other part is omitted
```

Listing 1: PhoneNumber class des Bibliothek-libphonenumber

IsTargetNumberValid-Methode

```
    @Override
    public boolean isTargetNumberValid(Phonenumber.PhoneNumber
↪   numberProto) {
        log.debug("Is target number valid: {}", numberProto);
        boolean validNumber = true;
        for (RuleSetService r : ruleSetService) {
            validNumber = r.checkTargetNumber(numberProto).isValid();
        }
        return validNumber;
    }
```

Listing 2: IsTargetNumberValid-Methode

CheckTargetNumber-Methode in LibPhoneRuleSet-Service

```
    @Override
    public NumberResponse checkTargetNumber(Phonenumber.PhoneNumber
↪   phoneNumber) {
        log.info("Check targetNumber in LibPhoneRuleSetService");
        boolean isValid = phoneUtil.isValidNumber(phoneNumber);
        boolean isPossibleNumber = phoneUtil
                .isPossibleNumber(phoneNumber);
        var numberType = phoneUtil.getNumberType(phoneNumber);
        var carrier = carrierMapper.getNameForNumber(phoneNumber,
↪   Locale.ENGLISH);
        NumberResponse numberResponse = ModifiableNumberResponse.create()
                        .isValid(isValid)
                        .isPossibleNumber(isPossibleNumber)
                        .numberType(String.valueOf(numberType))
                        .countryCode(phoneNumber.getCountryCode())
                        .nationalNumber(phoneNumber.getNationalNumber())
                        .serviceProvider(carrier);
        log.debug("NumberResponse: {}", numberResponse);
        return numberResponse;
    }
```

Listing 3: CheckTargetNumber-Methode in LibPhoneRuleSetService

CheckTargetNumber-Methode in DbRuleSetService

```
    @Override
    public NumberResponse checkTargetNumber(Phonenumber.PhoneNumber
↪   phoneNumber) {
        int ndcLength = phoneUtil
                .getLengthOfNationalDestinationCode(phoneNumber);
        var ndc = phoneUtil
                .getNationalSignificantNumber(phoneNumber)
                .substring(0, ndcLength);
        ModifiableNumberResponse numberResponse;
        var cc = phoneNumber.getCountryCode();
        try {
            numberResponse = numberService
                .findNumberByCountryCodeAndLeadingDigitsOfNSN(cc, ndc);
        } catch (EntityNotFoundException e) {
            numberResponse = null;
        }
        if (numberResponse == null) {
            Number number = new Number();
            var usage = phoneUtil.getNumberType(phoneNumber);
            number.setUsage(usage);
            number.setCountryCode(cc);
            number.setLeadingDigits(Integer.valueOf(ndc));
            var carrier = carrierMapper.getNameForNumber(phoneNumber,
↪   Locale.ENGLISH);
            number.setServiceProvider(carrier);
            number.setNsnLength(
                phoneUtil
                    .getNationalSignificantNumber(phoneNumber).length());

            Number savedNumber = numberService.create(number);

            numberResponse = ModifiableNumberResponse.create()
                    .id(number.getId())
                    .isValid(phoneUtil.isValidNumber(phoneNumber))
                    .isPossibleNumber(
                        phoneUtil.isPossibleNumber(phoneNumber))
                    .countryCode(number.getCountryCode())
                    .leadingDigits(number.getLeadingDigits())
                    .serviceProvider(number.getServiceProvider())
                    .usage(number.getUsage())
                    .nsnLength(number.getNsnLength())
                    .updateCommunicationDatetime(
                        number.getUpdateCommunicationDatetime())
                    .rangeSpecialNote(number.getRangeSpecialNote());
        }
        return numberResponse;
    }
```

Listing 4: CheckTargetNumber-Methode in DbRuleSetService

CheckTargetNumber-Methode in TwilioRuleSetService

```
    @Override
    public NumberResponse checkTargetNumber(Phonenumber.PhoneNumber
↪   phoneNumber) {
        PhoneNumber twilioResponse;
        boolean isCarrierLookupEnabled =
↪   twilioConfig.isLookupCarrierEnabled();
        ModifiableNumberResponse response =
↪   ModifiableNumberResponse.create();
        var cc = phoneNumber.getCountryCode();
        //twilio initialisation is omitted
        String formatted = phoneUtil.format(phoneNumber,
↪   PhoneNumberUtil.PhoneNumberFormat.E164);
        var request = new com.twilio.type.PhoneNumber(formatted);
        if (!isCarrierLookupEnabled) {
            twilioResponse = PhoneNumber.fetcher(request).fetch();
            } else {
            twilioResponse = PhoneNumber.fetcher(request)
                    .setType(List.of("carrier")).fetch();}
        if (twilioResponse != null) {
            String ndc = numberService.returnNdcOfNumber(phoneNumber);
            try {
                response = numberService
                    .findNumberByCountryCodeAndLeadingDigitsOfNSN(cc,
↪   ndc);
                response.isValid(phoneUtil.isValidNumber(phoneNumber));
            } catch (EntityNotFoundException e) {
                Number number = new Number();
                //setters of number are omitted
                Number savedNumber = numberService.create(number);
                response = ModifiableNumberResponse.create();
                //setters of response are omitted
            }
            //if carrierLookupIsEnabled extra fields of response are set
        } else {
            response.isValid(false);
            response.isPossibleNumber(false);
        }
        return response;
    }
```

Listing 5: CheckTargetNumber-Methode in TwilioRuleSetService

MessageRequest-Klasse

```java
package com.flatex.pfs.smsgatekeeper.message.dto;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import jakarta.annotation.Nullable;
import jakarta.validation.constraints.NotEmpty;
import org.immutables.value.Value;

@Value.Immutable
@Value.Modifiable
@JsonSerialize(as = ImmutableMessageRequest.class)
@JsonDeserialize(as = ImmutableMessageRequest.class)
@Value.Style(set = "")
public interface MessageRequest {
    @NotEmpty(message = "Please provide from")
    String getFrom();

    @NotEmpty(message = "Please provide to")
    String getTo();

    @NotEmpty(message = "Please provide body")
    String getBody();

    @Nullable
    String getSolution();
}
```

Listing 6: MessageRequest-Klasse

NumberResponse-Klasse

```
package com.flatex.pfs.smsgatekeeper.number.dto;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import org.immutables.value.Value;

@Value.Immutable
@Value.Modifiable
@JsonSerialize(as = ImmutableNumberResponse.class)
@JsonDeserialize(as = ImmutableNumberResponse.class)
@Value.Style(set = "")
public interface NumberResponse {
    long getId();

    int getCountryCode();

    int getLeadingDigits();

    String getNsnLength();

    String getServiceProvider();

    String getUpdateCommunicationDatetime();

    @Value.Default
    default int getMcc() { return 0; }

    @Value.Default

    default int getMnc() { return 0; }

    String getUsage();

    String getRangeSpecialNote();

    @Value.Default
    default boolean isValid() { return false; }

    @Value.Default
    default boolean isPossibleNumber() { return false; }

    @Value.Default
    default String getNumberType() { return ""; }

    @Value.Default
    default long getNationalNumber() { return 0; }
}
```

Listing 7: NumberResponse-Klasse

SmsGatekeeperConfig-Klasse

```
@Validated
@ConfigurationProperties(prefix = "sms.gatekeeper")
public class SmsGatekeeperConfig
{
    private static String checkStatus = "CHECKS: ";
    @NotNull
    private int requestTimeIntervalSeconds;
    @NotNull
    private int allowedRequestAmountInTimeIntervalSeconds;
    @Min(0)
    @Max(4)
    private int checkMode;
    @NotNull
    private Set<Integer> whiteListedCountries;
    private Set<Integer> blackListedCountries;

    @NotBlank
    private String adminUsername;
    @NotBlank
    private String adminPassword;
    @NotNull
    private Set<String> authValidApiKeys;

    @NotBlank
    private String authHeader;

    @NotNull
    private double deviationPercentage;
    @Min(1)
    @Max(365)
    @NotNull
    private int countDeviationPercentageForLastNDays;
}
```

Listing 8: SmsGatekeeperConfig-Klasse

Die Datenbankstruktur von SMS-Gatekeeper

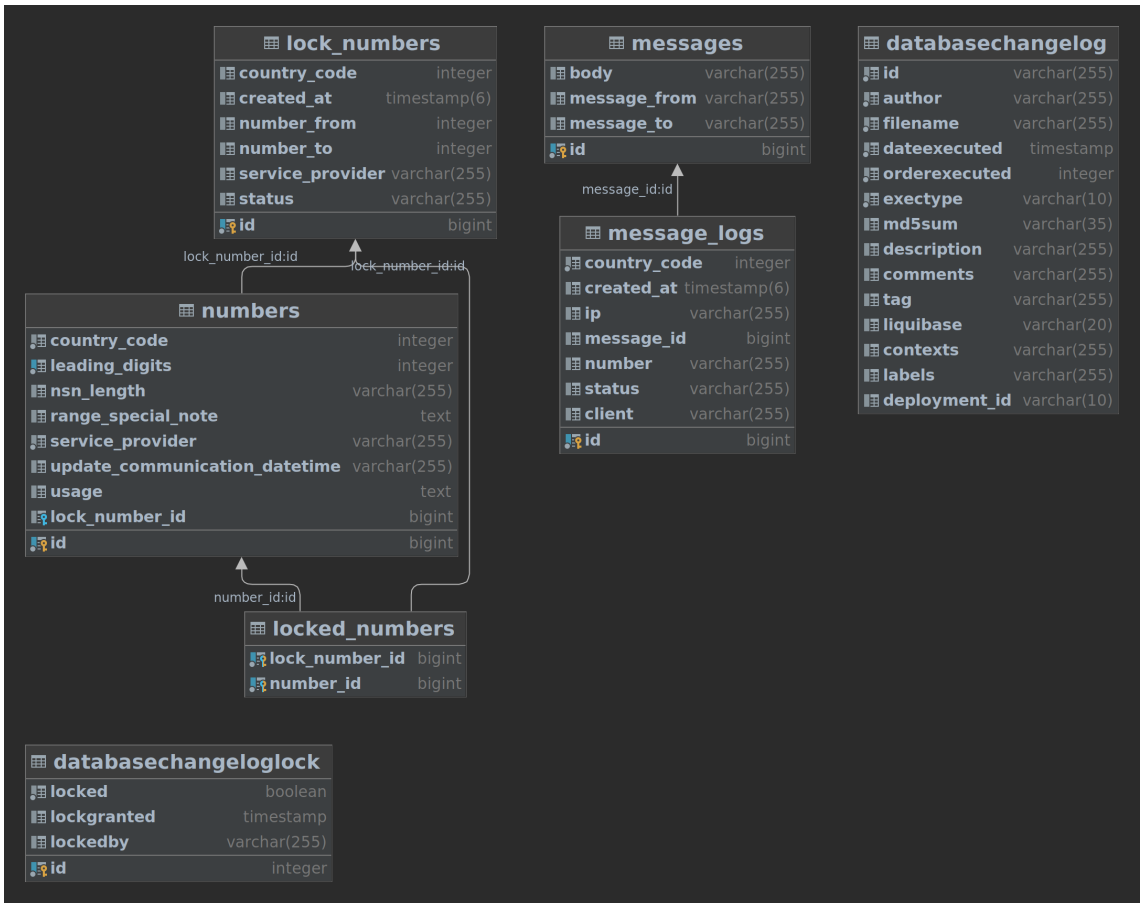


Abbildung A.1: Die Datenbankstruktur von SMS-Gatekeeper

Cucumber-Report

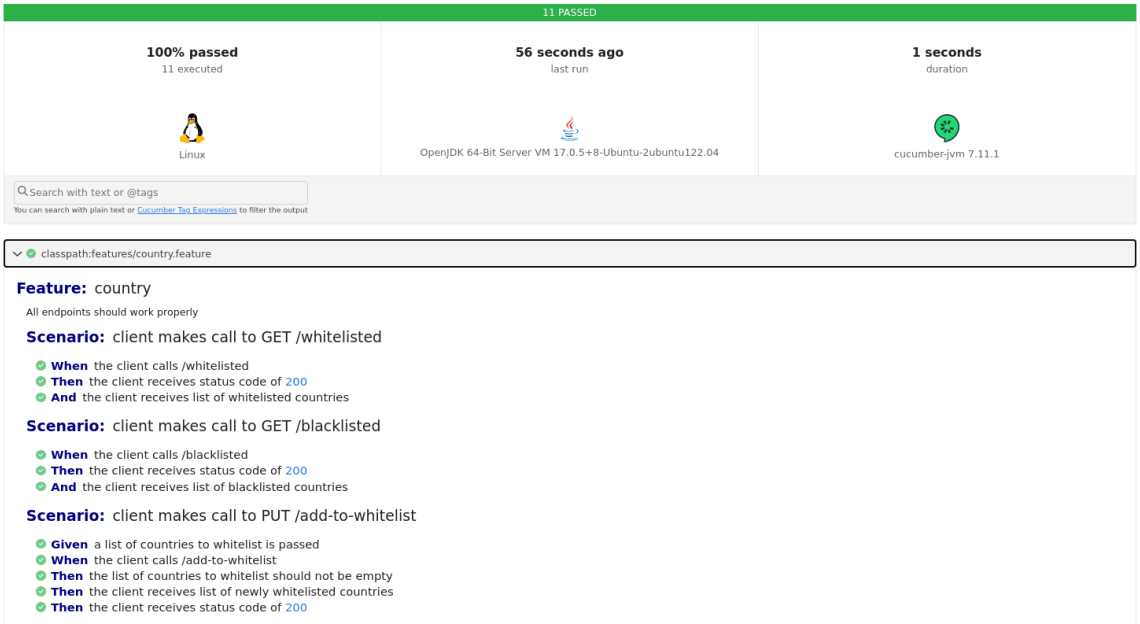


Abbildung A.2: Cucumber-Report(HTML)

CheckMode-Methode

```
    @Override
    public NumberResponse checkMode(int mode, Phonenumber.PhoneNumber
↪   phoneNumber) {
        NumberResponse numberResponse = NumberResponse.builder().build();
        switch (mode) {
            case 1 -> {
                numberResponse = ModifiableNumberResponse
                        .create()
                        .isValid(true)
                        .isPossibleNumber(true);
                return numberResponse;
            }
            case 2 ->
↪   libPhoneRuleSetService.checkTargetNumber(phoneNumber);
            case 3 -> {
                boolean isValid =
↪   libPhoneRuleSetService.checkTargetNumber(phoneNumber).isValid();
                if (isValid)
                    numberResponse =
↪   dbRuleSetService.checkTargetNumber(phoneNumber);
            }
            case 4 -> {
                boolean isValid = isTargetNumberValid(phoneNumber);
                numberResponse = ModifiableNumberResponse
                        .create()
                        .countryCode(phoneNumber.getCountryCode())
                        .nationalNumber(phoneNumber.getNationalNumber())
                        .isValid(isValid);
            }
            default -> throw new IllegalStateException("Bad value!");
        }
        return numberResponse;
    }
```

Listing 9: CheckMode-Methode

MeterRegistry-Klasse

```
    package io.micrometer.core.instrument;
    // imports

    public abstract class MeterRegistry {
    protected final Clock clock;
    private final Object meterMapLock = new Object();
    private volatile MeterFilter[] filters = new MeterFilter[0];
    private final List<Consumer<Meter>> meterAddedListeners = new
↪   CopyOnWriteArrayList();
    private final List<Consumer<Meter>> meterRemovedListeners = new
↪   CopyOnWriteArrayList();
    private final List<BiConsumer<Id, String>>
↪   meterRegistrationFailedListeners = new CopyOnWriteArrayList();
    private final MeterRegistry.Config config = new
↪   MeterRegistry.Config();
    private final MeterRegistry.More more = new MeterRegistry.More();
    private final Map<Id, Meter> meterMap = new ConcurrentHashMap();
    private final Map<Id, Set<Id>> syntheticAssociations = new HashMap();
    private final AtomicBoolean closed = new AtomicBoolean();
    private PauseDetector pauseDetector = new NoPauseDetector();
    @Nullable
    private HighCardinalityTagsDetector highCardinalityTagsDetector;
    private NamingConvention namingConvention;

    protected MeterRegistry(Clock clock) {
        this.namingConvention = NamingConvention.snakeCase;
        Objects.requireNonNull(clock);
        this.clock = clock;
    }

    protected abstract <T> Gauge newGauge(Id var1, @Nullable T var2,
↪   ToDoubleFunction<T> var3);

    protected abstract Counter newCounter(Id var1);

    //other part is omitted
    }
```

Listing 10: MeterRegistry-Klasse

CucumberAcceptanceTests-Klasse

```
        @ActiveProfiles(Profiles.DIT)
        @Suite
        @IncludeEngines("cucumber")
        @SelectClasspathResource("features")
        @ConfigurationParameter(key = GLUE_PROPERTY_NAME, value =
↪   "com.flatex.pfs.smsgatekeeper.stepdefs")
        @CucumberContextConfiguration
        @SpringBootTest(webEnvironment =
↪   SpringBootTest.WebEnvironment.RANDOM_PORT)
        @AutoConfigureMockMvc
        @Testcontainers
        public class CucumberAcceptanceTests {
            //other parts are omitted
        }
```

Listing 11: CucumberAcceptanceTests-Klasse

Testkonfigurationsdatei

```
    cucumber.publish.quiet=true
    cucumber.plugin=pretty,\
            html:target/cucumber-report/report.html,\
            json:target/cucumber-report/report.json,
```

Listing 12: junit-platform.properties

```
    ryuk.container.image=NEXUS_LIBRARY_TESTCONTAINERS
    ryuk.container.privileged = false
    hub.image.name.prefix=NEXUS_LIBRARY_PostgreSQL
    pull.pause.timeout=30
    check.disable=true
    docker.host=tcp://localhost:2376/
```

Listing 13: testcontainers.properties