# Inhaltsverzeichnis

Inhaltsverzeichnis ..... I

Abkürzungsverzeichnis ..... II

Abbildungsverzeichnis ..... II

Tabellenverzeichnis ..... II

1 Einleitung ..... 1

1.1 Motivation ..... 1

1.2 Unternehmensvorstellung ..... 2

1.3 Grundlagen ..... 3

1.3.1 Sicherheitsrelevante Begrifflichkeiten und Verfahren ..... 3

1.3.2 Einführung in die Cloud Taxonomie ..... 5

2 Hauptteil ..... 8

2.1 Analyse der vorhandenen Sicherheitskonzepte ..... 8

2.1.1 Ist-Zustand ..... 8

2.1.2 Soll-Zustand ..... 9

2.2 Kriterienkatalog für die STP Cloud ..... 9

2.3 Erkennung und Prävention von Gefahren ..... 10

2.3.1 Konzeption einer Testumgebung ..... 10

2.3.2 Mögliche Architektur mit ML.NET Proxy Service ..... 10

3 Schluss ..... 11

4 Literaturverzeichnis ..... 12

# Abkürzungsverzeichnis

SaaS – Software-as-a-Service

PaaS – Platform-as-a-Service

IaaS – Infrastructure-as-a-Service

# Abbildungsverzeichnis

Abbildung 1 JSON-Web-Token in kodierten und dekodierten Zustand ..... 4
Abbildung 2 Verantwortung über die genutzten Cloud Services. Grafik: Security Guidance v4.0, CSA ..... 6
Abbildung 3 Verwantwortung von Cloud Kunde und Cloud Betreiber Quelle: https://www.magenium.com/magenium/Magenium_Cloud_Services_Diagram.jpg ..... 7
Abbildung 4 Vereinfachte Darstellung der STP Cloud SaaS-Lösung ..... 8
Abbildung 5 Testumgebung für die Simulation der realen Anwendungslandschaft ..... 10
Abbildung 6 Beispiel-Architektur mit zusätzlichem Service für die Prüfung der eingehenden Requests mit Möglichkeit zum Throttling ..... 11

# Tabellenverzeichnis

# 1 Einleitung

## 1.1 Motivation

Gestützt durch Vorfälle aus jüngster Zeit ist die Sicherheit in der Informationstechnologie für Firmen auf der ganzen Welt zu einem immer mehr an Bedeutung gewinnenden Bestandteil geworden. Als Folge der aktuellen Angriffe wie zum Beispiel das unautorisierte Entwenden von mehr als 500 Millionen Nutzerdaten von Facebook (Duffy 2021) oder das Ausnutzen der Schwachstellen der Microsoft Exchange Server in zahlreichen Unternehmen (o.V. 2021a) nehmen immer mehr Firmen die IT-Sicherheit ihrer Produkte genauer in den Fokus. Dies gilt nicht nur für Software wie native Applikationen auf Mobiltelefonen oder Desktopanwendungen, sondern auch für Anwendungen, die ihren Nutzern als Clouddienste zur Verfügung stehen und öffentlich über das Internet erreichbar sind. Jedoch sind Kriterien, die für die Absicherung dieser Dienste gedacht sind, wie zum Beispiel des National Institut for Standards and Technology (NIST), der Cloud Security Alliance (CSA) oder das Bundesamt für Sicherheit in der Informationstechnik (BSI) nur wenig verbreitet und werden von den betreffenden Firmen mehr oder weniger umgesetzt. Des Weiteren ist festzustellen, dass sich die aufgezählten Kriterienkataloge immer wieder aufeinander beziehen, jedoch keine einheitliche Norm zur Regelung und Umsetzung der Sicherheitskriterien vorliegt. Als Richtlinie und Maßstab für zukünftige Entwicklungen an Cloudprodukten sollte jedes Unternehmen einen Katalog an bestehenden Sicherheitsmaßnahmen und Regularien entwerfen, der den Endnutzern ein gewisses Maß an Sicherheit garantiert.

## 1.2 Unternehmensvorstellung

Die STP Informationstechnologie GmbH (nachfolgend kurz: STP GmbH) ist ein in Karlsruhe gegründetes IT-Unternehmen. Der Schwerpunkt der angebotenen Softwarelösungen und Informationssysteme zielt auf die Anwendungen in den Berufsgruppen im Rechtssektor wie Anwälte, Justizverwaltungen und weiteren fachnahen Institutionen ab. Gegründet wurde das Unternehmen im Jahr 1993 von Gunter Thies und Ralph Suikat als „Suikat-Thies + Partner GmbH“. Im Jahre 2001 wurde die Unternehmensform in eine Aktiengesellschaft umgewandelt (o.V. 2021b). Die STP AG im Rahmen eines Zertifizierungsprogramms durch SGS-International Certification Services GmbH nach DIN ISO 9001 zertifiziert. Seit November 2011 gilt dieses Zertifikat für die komplette STP Gruppe: STP Informationstechnologie AG, STP Holding GmbH, STP Portal GmbH und STP Solution GmbH. Seit dem 05. März 2021 ist die STP von einer Aktiengesellschaft in eine GmbH umfirmiert (o.V. 2021b).

Intern untergliedert sich die STP weiterhin in einzelne Abteilungen, Arbeitsbereiche und -gruppen, die mit verschiedenen Themen betraut sind. Der Schwerpunkt jeder einzelnen Abteilung liegt auf einem anderen Gebiet der Software- bzw. Produktentwicklung.

Der Fokus bei der Softwareentwicklung liegt jedoch primär auf der Umsetzung von Kundenlösungen mit dem .NET-Framework. Die Produktpalette umfasst neben On-Premise Lösungen, die lokal beim Kunden eingesetzt werden, auch Dienstleistungen wie Consulting- bzw. Beratungslösungen, die von internen Beratern bzw. Fachbearbeitern angeboten werden.

Der Wirkungsbereich der STP Informationstechnologie umfasst die gesamte DACH-Region. Dies bedeutet, es werden neben Kunden aus Deutschland auch Kunden aus der Schweiz, Standort der jüngsten Zweigstelle, und Österreich betreut. Die Niederlassung in Bulgarien fungiert in diesem Unternehmensverbund bisher als Zuarbeiter für spezielle Aufgaben der Entwicklung.

Das erarbeitete Projekt wurde hauptsächlich in der Abteilung Produktentwicklung (PDE) mit Betreuung durch Manuel Naujoks durchgeführt und erarbeitet. Der Fokus dieser Abteilung liegt auf der Evaluation und Verwendung von neuen Technologien im Ökosystem .NET, die bei der Entwicklung von neuen hauseigenen Produkten verwendet werden sollen.

## 1.3 Grundlagen

### 1.3.1 Sicherheitsrelevante Begrifflichkeiten und Verfahren

#### 1.3.1.1 Authentifizierung und Autorisierung

Aufgrund der Relevanz der Begriffe **Authentifizierung** und **Autorisierung** im Bereich der Informationssicherheit werden diese zunächst zum allgemeinen Verständnis in den Kontext eingeordnet und erläutert.

Beginnend mit dem Begriff der **Authentifizierung**. Für das nachfolgende theoretische Beispiel wird von einer Kommunikation zwischen einem menschlichen Nutzer mit einer Anwendung (Maschine) ausgegangen.

Die Authentifizierung des Nutzers bei einer Anwendung ist in vielen Fällen der erste Schritt, wenn der Nutzer Zugriff auf geschützte Ressourcen, wie zum Beispiel die Daten seines Profils in einem sozialen Netzwerk haben möchte. Hierzu wird er vom System bzw. der Anwendung zuerst aufgefordert den Usernamen und sein Passwort (nachfolgend Zugangsdaten) einzugeben. Diesen Vorgang der Authentifizierung nutzt die Anwendung, um zu prüfen, ob es sich bei dem vorliegenden Nutzer wirklich um den Nutzer handelt, der er vorgibt zu sein. Stimmen die Zugangsdaten, die meistens via Abgleich von verschlüsselten Werten in der Datenbank überprüft werden, mit den angegebenen Werten überein, ist der Nutzer erfolgreich authentifiziert. Ist dies nicht der Fall, wird dem Nutzer der Zugriff verwehrt.
Bei erfolgreicher Authentifizierung erhält der Nutzer in den meisten Fällen ein Token, der ihm als Beweis seiner Identität dient, um sich gegenüber dem System auszuweisen. Somit wird verhindert, dass bei erneuter Interaktion des Nutzers mit dem System, dieser sich erneut anmelden muss.

Bei der **Autorisierung** kommt der bereits genannte Token zum Einsatz. Möchte der Nutzer nun auf die Daten seines Profils zugreifen, sendet er neben dem Request noch seinen Token, im Header des Requests, mit. Dieser Token wird anschließend evaluiert und auf seine Gültigkeit geprüft. Stimmen die notwendigen Rollen mit denen im mitgelieferten Token überein, erhält der Nutzer den gewünschten Zugriff. Ansonsten wird ihm der Zugriff verwehrt.

Die Form und der Informationsgehalt dieser Tokens kann sich in den unterschiedlichen Authentifizierungs-Verfahren (nachfolgend auch Flows genannt) unterscheiden.
In den meisten Fällen handelt es sich jedoch um einen sogenannten JSON-Web-Token (JWT), der im Bearer Schema vorliegt. Diese Art von Tokens wird zur Übermittlung von Informationen genutzt und sind in den meisten Fällen signiert und verschlüsselt.

#### 1.3.1.2 JSON-Web-Token

Der grundlegende Aufbau des Tokens lässt sich in drei Bestandteile aufgliedern. Beginnend mit dem „Header“ oder auch Kopf, der den Typ des Tokens und den Algorithmus zur Signatur festlegt. Anschließend folgt durch einen Punkt getrennt der „Payload“, der userspezifische Informationen und die Berechtigungen des Users enthält. Als letzter Bestandteil folgt erneut durch einen Punkt getrennt die „Signature“ oder auch Signatur, mit der die Validität des Tokens überprüft werden kann.

Die gängigste Variante ist das Signieren der Tokens von Seiten des Identity Providers und eine Bereitstellung des Public Keys über einen öffentlichen Endpunkt. Mit Hilfe dieses Public Keys kann anschließend eine Client-Anwendung die Echtheit und die Validität des gesendeten Tokens überprüfen. Zugriff auf diese Informationen erhalten die Clients bzw. Anwendungen über spezielle Endpunkte, die die Identity Provider hierfür zur Verfügung stellen. Der derzeitige Standard zur Ausgabe dieser Informationen definiert eine URL in dem Format `<IdentityProviderDomain>/.well-known/openid-configuration`.

Das nachfolgende Beispiel zeigt einen JWT in seinem kodierten und dekodierten Zustand. Eine Kodierung mittels Base64 minimiert die Größe des Tokens und erleichtert somit die Übertragung auf HTTP-Ebene.

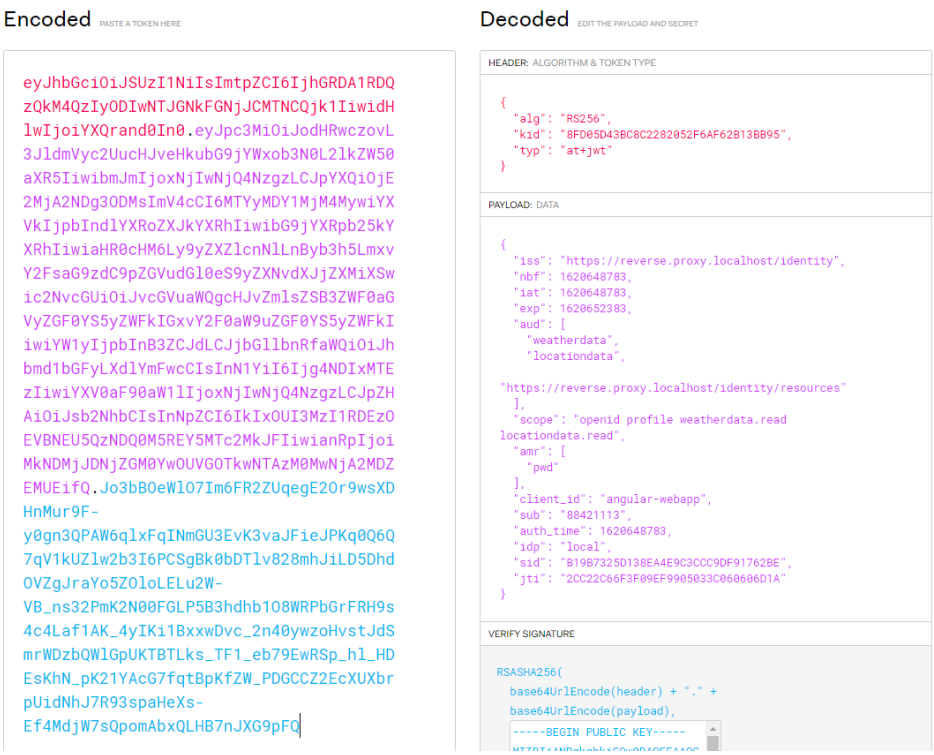


*Abbildung 1 JSON-Web-Token in kodierten und dekodierten Zustand*

#### 1.3.1.3 Authorization Code Flow with Proof-Key-of-Code-Exchange

Zuerst erfolgt einer Erläuterung des "**Authorization Code Flow + Proof Key of Code Exchange (PKCE)**". Diese Authentifizierungsmethode ist derzeit der Standard zur Authentifizierung mittels Webapplikationen, wie einer SPA bei Identity Providern. Der Vorgang besteht darin, dass der Client (hier eine Webapplikation) eine **Code_Challenge** an den Authorization Server (hier den Identity Provider, kurz AS genannt) schickt. Bei dieser **Code_Challenge** handelt es sich um eine Zeichenkette aus Base64-kodierten Werten. Diese Werte können Zeichen im Format von `a-z, A-Z, 0-9, . , ,,‘, ~` und - enthalten.

Anschließend wird diese Zeichenkette mittels SHA-256 verschlüsselt und an den AS geschickt. Nach Eingang der **Code_Challenge** speichert der AS den Wert ab und sendet eine **Code_Verification** zurück.

Möchte der Client nun einen Token zum Zugriff auf geschützte Bereiche des Ressource Server (kurz RS genannt), muss er neben der **Code_Challenge** zusätzlich die **Code_Verification** an den AS schicken, um sich zu authentifizieren (Nat Sakimura et al. 2015). Der Nachteil bei dieser Authentifizierungsvariante besteht darin, dass bei Entwendung des Tokens, dieser für die restliche Zeit seiner Gültigkeit weiterverwendet werden kann, ohne dass der Nutzer dies verhindern kann.

### 1.3.2 Einführung in die Cloud Taxonomie

Für die Einordung und das Verständnis der Cloud Taxonomie werden in diesem Kapitel der Thesis alle notwendigen Begriffe und deren Definition eingeführt. Die Ressourcen eines Cloud Computing Dienstes reichen von Software-Diensten bis zu Datenspeichern, Betriebssystemen und ganzen Hardware-Infrastrukturen. Basierend auf der Granularität des Dienstes kann in Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) und Software-as-a-Service (SaaS) unterschieden werden (Squicciarini et al. 2021, S. 5). Neben diesen bereits aufgeführten Service Modellen gibt es noch eine Bandbreite an weiteren Diensten, die über die Cloud angeboten werden können. Hierunter zählen zum Beispiel Communication-as-a-Service (CaaS), Compute-as-a-Service (CompaaS), Data-Storage-as-a-Service (DSaaS) bzw. Database-as-a-Service (DaaS) oder Network-as-a-Service (NaaS) (Stallings 2021, S. 17). In diesem Grundlagenkapitel bzw. im Rahmen der Thesis werden jedoch nur die ersteren drei Modelle genauer beschrieben. Sie sind im Rahmen der Cloud Sicherheit am besten für die Einordnung der Verantwortung des Kunden bezüglich der Wartung und Absicherung der in Anspruch genommenen Cloud Leistung geeignet.

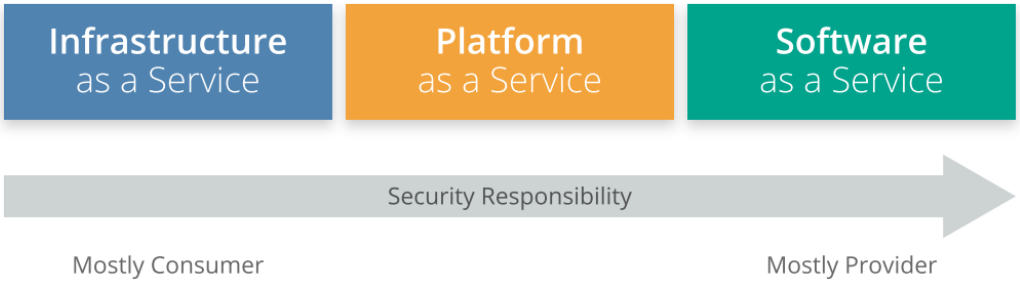


*Abbildung 2 Verantwortung über die genutzten Cloud Services. Grafik: Security Guidance v4.0, CSA*

#### 1.3.2.1 Infrastructure-as-a-Service

Durch die Verwendung von Infrastructure-as-a-Service hat der Cloud Kunde den kompletten Zugriff auf alle Komponenten (wie z.B. Betriebssysteme, Middleware, Laufzeit, Daten und Applikationen) des gemieteten Cloud Servers. Hierauf kann über vordefinierte grafische Benutzerschnittstellen oder VPN Verbindungen zugegriffen werden (Stallings 2021, S. 17). Dies garantiert neben einer Vielzahl an Konfigurationsmöglichkeiten eine große Verantwortung hinsichtlich der Wartung des Servers und den darauf ausgerollten Applikationen. Zusätzlich muss darauf geachtet werden, dass die neusten kritischen Sicherheits- und Betriebssystem-Updates installiert sind. Dies rundet die Konfiguration der Firewall, die den Server vor unbefugtem Zugriff schützt, ab.

#### 1.3.2.2 Platform-as-a-Service

Im Rahmen von Platform-as-a-Service werden dem Cloud Kunden unterschiedliche Plattformen für das Betreiben seiner Anwendungen zur Verfügung gestellt. In diesem Rahmen kann dieser seine Applikationen mit Hilfe von vorhandenen Entwickler-Tools und Laufzeitumgebungen ohne größeren Aufwand hinsichtlich der Konfiguration platzieren (Stallings 2021, S. 16–17).

#### 1.3.2.3 Software-as-a-Service

Der Cloud-Dienst Software-as-a-Service impliziert eine in der Cloud betriebene Anwendung, auf die der Kunde über einen Web-Browser zugreifen kann. Der Nutzer dieser Software hat somit den Vorteil, dass die Anwendung nicht lokal auf seinem System betrieben werden muss und somit keinerlei lokale Ressourcen verbraucht. Hiermit entfallen der Installationsprozess auf der lokalen Umgebung und der Kauf von Desktop- und Server-Lizenzen. Abgerechnet wird je nach Nutzungsdauer und der Anzahl der Nutzeraccounts, die für den jeweiligen Tenant angelegt wurden. Typische Anwendungen in diesem Service Modell sind E-Mail- und Dokumentenmanagement-Programme (Stallings 2021, S. 16).

Wie in der nachfolgenden Grafik nochmals genauer veranschaulicht ist innerhalb der einzelnen Service Modelle ein Gefälle an Verantwortung zu erkennen, die der Cloud Kunde selbst übernehmen muss. Von der kompletten Verwaltung von eigenen On-Premises Systemen bis hin zur fremdverwalteten SaaS-Lösung.

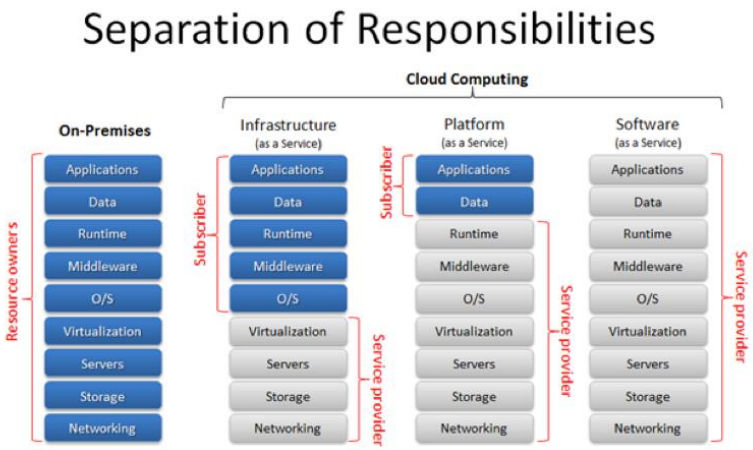


*Abbildung 3 Verwantwortung von Cloud Kunde und Cloud Betreiber*
*Quelle: https://www.magenium.com/magenium/Magenium_Cloud_Services_Diagram.jpg*

# 2 Hauptteil

## 2.1 Analyse der vorhandenen Sicherheitskonzepte

In den nachfolgenden Kapiteln wird der aktuelle Stand an Sicherheitskonzepten der Software-as-a-Service Lösung, auch „LEXolution.FLOW“ genannt, analysiert und einem möglichen Soll-Zustand gegenübergestellt.

### 2.1.1 Ist-Zustand

Die zu bewertende SaaS-Lösung der STP wird über einen Cloud Service Provider (nachfolgend auch CSP) zur Verfügung gestellt, bedeutet die notwendige Hardware wird nicht In-House betrieben, sondern von Außerhalb je nach Bedarf gebucht. Hierbei handelt es sich um ein deutsches Rechenzentrum, welches seine Daten ausschließlich in Standorten innerhalb von Deutschland speichert. In Folge des Beschlusses der amerikanischen Regierung ist es mittels des CLOUD Acts (Clarifying Lawful Overseas Use of Data) regierungsnahen Institutionen, wie zum Beispiel der NSA, CIA oder FBI, möglich sich ohne Einwilligung oder vorheriges Informieren der Cloud Nutzer Zugang zu den gespeicherten Daten der Cloud Provider zu verschaffen (Congress Government 2021). So wäre ein Hosting bei Microsoft, Amazon Web Services oder Google Cloud Platform mit der deutschen Rechtslage für die Zielgruppe der STP nicht vereinbar.

Bei der Analyse des Ist-Zustandes steht jedoch nicht die Bewertung der Regularien zum Betreiben einer Cloud Lösung oder die Sicherheitskriterien für die Rechenzentren im Mittelpunkt, sondern die Resistenz vor gängigen Angriffen auf Cloud Softwarelösungen und Ausfalltoleranz der Anwendung selbst. Hiermit wird eine Grenze zwischen der Software und der Plattform beziehungsweise der darunterliegenden Infrastruktur gezogen.

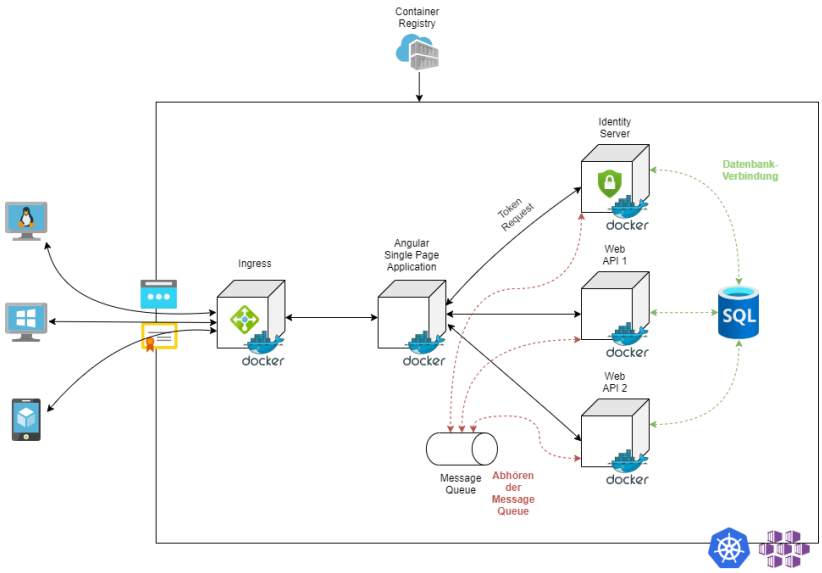


*Abbildung 4 Vereinfachte Darstellung der STP Cloud SaaS-Lösung*

Mithilfe der vorangegangenen vereinfachten Darstellung der STP Cloud Lösung soll nun die Bestandsaufnahme des Ist-Zustandes erfolgen. Der Anwendungsverbund besteht im übergeordneten Sinne aus unterschiedlichen Single Page Applikationen (nachfolgend auch SPA genannt), die die grafische Benutzeroberfläche der Anwendung repräsentieren. Hierbei wird je nach gewünschtem Menü über den vorgelagerten Ingress im Kubernetes Kluster die entsprechende SPA angesprochen. Somit verteilt sich die Last nicht nur auf eine einzelne Webanwendung. Über die Weboberfläche wird dem Kunden die Eingabe von für den Prozess relevanten Daten ermöglicht. Die Verarbeitung und Speicherung der Daten wird im Anschluss über Web Programmierschnittstellen (nachfolgend auch API genannt) gewährleistet. Die Implementierung der APIs ist mit dem .NET Framework und C# als Programmiersprache umgesetzt worden. Für die sichere Übertragung der Daten vom Frontend an das Backend wird mittels TLS 1.2 der ein- und ausgehende Datenstrom verschlüsselt. Dies verhindert im ersten Schritt das ungewünschte Mitschneiden und -lesen von Datenpaketen. Bezüglich der Verwaltung von Rechten wird im Backend ein IdentityServer in der Version 4 bereitgestellt. Hierbei handelt es sich um einen Service, der das OpenID Connect und das OAuth2 Protokoll umsetzt. Die Nutzung dieses Diensts gewährleistet eine zentrale Nutzerverwaltung für alle Tenants und die Vergabe von Rechten unterschiedlicher Granularität. Bevor der Nutzer Zugriff auf die SaaS-Lösung erhält, muss es sich initial beim Identity Provider (hier IdentityServer 4) authentifizieren. Stimmen Nutzername und Passwort mit den hinterlegten Anmeldedaten überein, erhält der Nutzer, die ihm zugewiesenen Rechte, er wird somit autorisiert. Im Hintergrund des Anmeldeprozesses findet der Authorization Code Flow with Proof-Key-of-Code-Exchange (nachfolgend auch Authorization Code Flow + PKCE genannt) statt. Hierbei handelt es sich um das aktuellste und sicherste im OAuth2 Protokoll spezifizierte Verfahren für den Austausch von Tokens zwischen Identity Providern und clientseitigen Webanwendungen, wie zum Beispiel SPAs.

### 2.1.2 Soll-Zustand

## 2.2 Kriterienkatalog für die STP Cloud

## 2.3 Erkennung und Prävention von Gefahren

### 2.3.1 Konzeption einer Testumgebung

Für die Evaluation und Umsetzung des ML.NET Proxies Dienstes wurde anhand der in Kapitel 2.1.1 beschriebenen Architektur der Anwendung eine beispielhafte Testumgebung erstellt. Diese besitzt die grundlegenden Eigenschaften des Originals und kann somit als Substitut für Testzwecke verwendet werden, ohne auf die eigentliche Anwendungslandschaft zugreifen zu müssen.

Wie im nachfolgenden Schaubild exemplarisch dargestellt, wird die Test-Lösung wie auch das Original mittels Containerisierung auf ein Kubernetes-Kluster ausgerollt. Auf die Nutzung einer Message Queue und einer relationalen Datenbank für die Persistierung der Daten wurde verzichtet. Der Fokus wird ausschließlich auf die Manipulation bzw. das Throttling der Anfragen an das Backend gelegt.

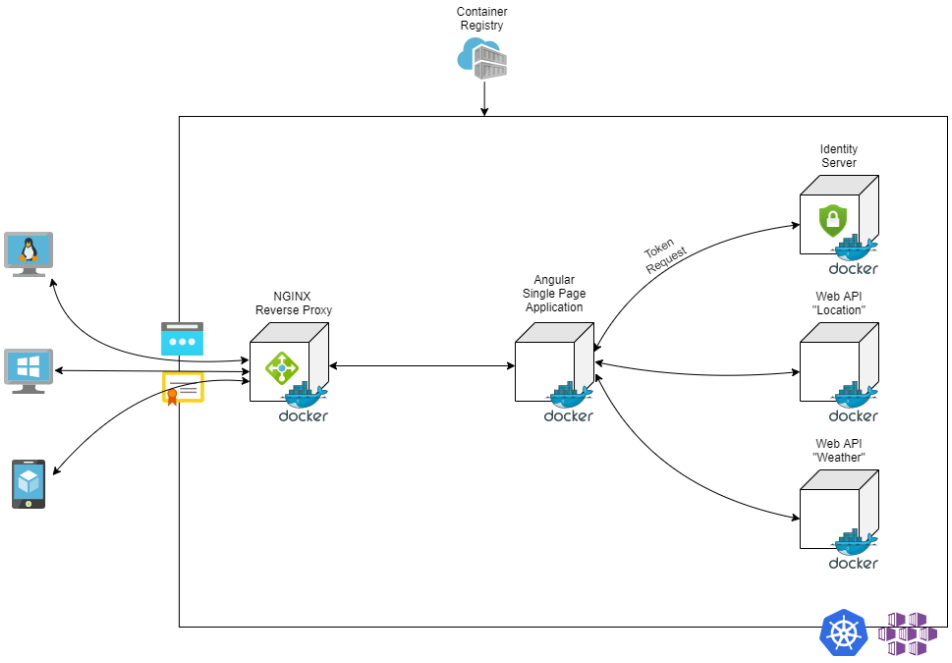


*Abbildung 5 Testumgebung für die Simulation der realen Anwendungslandschaft*

### 2.3.2 Mögliche Architektur mit ML.NET Proxy Service

Durch die Nutzung des ML.NET Proxy Service soll auf Auffälligkeiten in der Historie bzw. der aktuellen Anfragen von Außerhalb reagiert werden. Hierzu muss der Proxy jedoch zwischen Backend und dem Gateway des Klusters platziert werden. Eine mögliche Beispiel-Architektur ist im nachfolgenden Schaubild dargestellt.

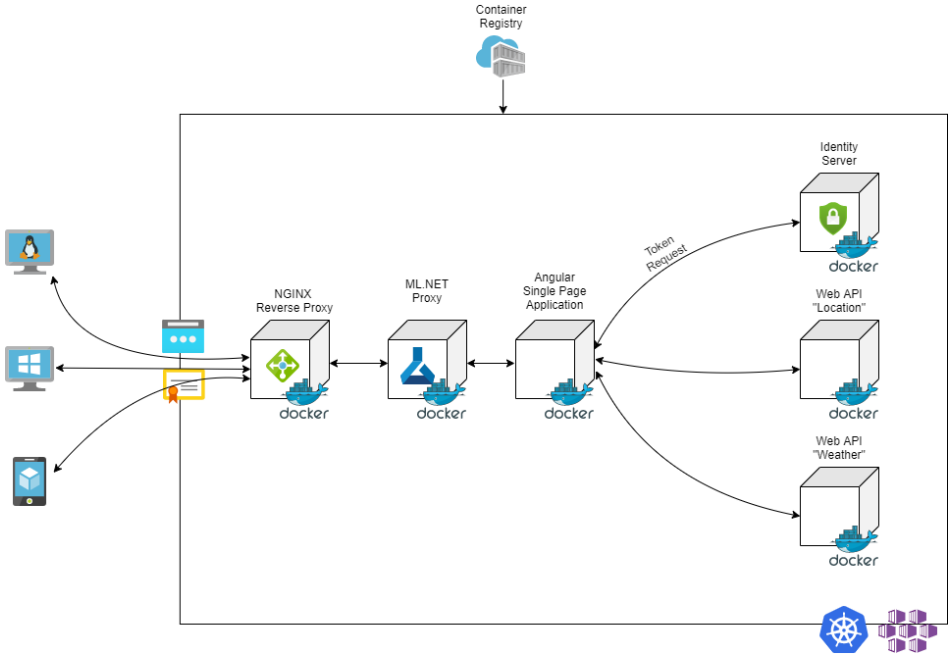


*Abbildung 6 Beispiel-Architektur mit zusätzlichem Service für die Prüfung der eingehenden Requests mit Möglichkeit zum Throttling*

# 3 Schluss

# 4 Literaturverzeichnis

Congress Government (2021): H.R.200 – 117th Congress (2021-2022): National Intersection and Interchange Safety Construction Program Act of 2021. Online verfügbar unter https://www.congress.gov/bill/115th-congress/house-bill/4943/text, zuletzt aktualisiert am 06.02.2018, zuletzt geprüft am 10.05.2021.

Duffy, Clare (2021): So you're one of 533 million in the Facebook leak. What now? CNN Business. Online verfügbar unter https://edition.cnn.com/2021/04/06/tech/facebook-data-leaked-what-to-do/index.html, zuletzt aktualisiert am 06.04.2021, zuletzt geprüft am 08.04.2021.

Nat Sakimura; John Bradley; Naveen Agarwal (2015): Proof Key for Code Exchange by OAuth Public Clients: RFC Editor (Request for Comments) (7636). Online verfügbar unter https://rfc-editor.org/rfc/rfc7636.txt.

o.V. (2021a): Mehrere Schwachstellen in MS Exchange. Bundesamt für Sicherheit in der Informationstechnik. Online verfügbar unter https://www.bsi.bund.de/SharedDocs/Cybersicherheitswarnungen/DE/2021/2021-197772-1132.pdf?__blob=publicationFile&v=4, zuletzt aktualisiert am 17.03.2021, zuletzt geprüft am 08.04.2021.

o.V. (2021b): STP Informationstechnologie AG. Stadtwiki Karlsruhe. Online verfügbar unter https://ka.stadtwiki.net/STP_Informationstechnologie_AG, zuletzt aktualisiert am 27.04.2021, zuletzt geprüft am 06.05.2021.

Squicciarini, Anna; Oliveira, Daniela; Lin, Dan (2021): Cloud Computing Essentials. In: John R. Vacca (Hg.): Cloud computing security. Foundations and challenges. Second edition. Boca Raton, FL: CRC Press, Taylor & Francis Group, S. 3–11.

Stallings, William (2021): An Overview of Cloud Computing. In: John R. Vacca (Hg.): Cloud computing security. Foundations and challenges. Second edition. Boca Raton, FL: CRC Press, Taylor & Francis Group, S. 13–29.