Red Room Discussion Channel

Software- und Systemarchitektur nach arc42

Bernhard Hartmann und Pavol Malo

1. Einführung und Ziele

Red Room Discussion Channel ist eine Anwendung für sicheren Informationsaustausch, die im Rahmen der Verhandlungen von Mergers & Acquisitions verwendet werden soll. Es sollte dabei gewährleistet werden, dass die Informationsflüsse auch für lokal vorhandene “Middlebox”-Software (wie zB Firewall oder vom jeweiligen Konzern verwendete Beobachtungssoftware zur Sicherstellung von Compliance) in keiner Hinsicht – inklusive der Metadaten – lesbar ist.

Das wirtschaftliche Ziel ist einerseits den Kundenkreis des Auftragsgebers um größere Unternehmen, bei denen persönlicher Informationsaustausch nicht möglich ist, zu erweitern; andererseits sollte der angebotene Leistungsumfang um eine SaaS erweitert werden.

Da die Daten lediglich auf der Server-Seite, die ausschließlich vom als Vermittler oder Transaktionsanwalt agierenden Auftraggeber verwaltet wird, wird gleichzeitig die Umsetzung der Non Disclosure Agreements erleichtert - insbesondere dahingehend, dass versehentliche Offenlegung von Informationen (wie bspw sorgloser Umgang mit den bei der Datenübertragung abgefangenen Metadaten aufgrund fehlender technischen Kenntnisse) vom Anfang an nicht möglich sein wird.

1.1. Aufgabenstellung

Die Aufgabe ist eine Webapplikation zu einem ultra-sicheren Informationsaustausch zu bauen.

Der Betrieb dieser Applikation muss auf jedem Rechner möglich sein, der online ist, Windows als Betriebssystem verwendet und die eingehende und ausgehende Kommunikation darf für andere Software, Middlebox und zwischengeschaltete Geräte (zB Router) inhaltlich nicht lesbar sein.

Diese Applikation soll künftig in der Form Software as a Service (SaaS) im Rahmen der Kaufabwicklung angeboten werden. Die primäre Funktion der Applikation ist die Datenvermittlung im Rahmen der Kaufabwicklung. Die sekundäre Funktion ist die Datenaufbewahrung für Beweiszwecke bis zur Verjährung der eventuellen Gewährleistungsansprüche (idR mindestens fünf Jahre), was den typischen Geschäftsverlauf des Auftraggebers teilweise von einmaligen Vermittlungsgeschäften in die Richtung von langfristigen Geschäftsverhältnissen schieben soll.

Siehe auch Anforderungsdokument SRS Red Room Discussion Channel vom 4. Dezember 2019, Version 0.2.

1.2. Qualitätsziele

Die primären Qualitätsziele lassen sich wie folgt erläutern:

Tabellarische Darstellung der Qualitätsziele		
Priorität	**Beschreibung**	**Szenario**
1	Datensicherheit	Die übermittelten Daten sind äußerst sensitiv. Es darf in keiner Hinsicht zu einer Offenlegung an Dritte kommen (zB Metadaten). Eine Offenlegung ist auch durch mangelhafte technische Kenntnisse hinsichtlich der von der Middlebox-Software (wie zB Firewall) in der lokalen Maschine angesammelten Daten oder durch "Interception" durch eine schädigende Person denkbar.
2	Datenaufhebung	Die Daten bilden Beweismaterial hinsichtlich der abgeschlossenen Verträge dar. Ein Datenverlust würde dementsprechend potentiell große Schaden verursachen können.
3	Datenübermittlung	Damit eine Information (zB eine Äußerung über die Ertragskraft des verkauften Unternehmens) überhaupt zu einem Bewegungsgrund für einen Vertragsabschluss werden kann, muss diese an die anderen Geschäftspartei übermittelt werden.

1.3. Stakeholder

In folgender Tabelle wird ein expliziter Überblick über die Stakeholder des Systems geliefert welche die Architektur kennen sollten oder von der Architektur überzeugt werden müssen. Zudem sind auch jene Stakeholder angeführt, welche mit der Architektur oder dem Code arbeiten (z.B. Schnittstellen nutzen), die Dokumentation der Architektur für ihre eigene Arbeit benötigen, oder Entscheidungen über das System und dessen Entwicklung treffen.

Rolle	Kontakt	Erwartungshaltung
Geschäftsführer der GeldGeldGeld AG	geld@geld.at	*Positive Erledigung des Projektes sowie gute Zusammenarbeit, sowohl im Team wie auch mit dem Projektleiter.*
Projektleiter	heimo.hirner@fh-campuswien.ac.at	*Research der neuen Technologie sowie der Implementierung einer Client-Server Applikation; Erstellung der Dokumentation über die einzelnen Schritte in geeigneter Form, sodass nach Ende des Projektes im Falle einer weiteren Beauftragung vom aktuellen Stand weitergearbeitet werden kann.*
Entwickler	pavol.malo@stud.fh-campuswien.ac.at	*Auseinandersetzen mit einer neuen Technologie, sowie das Anwenden von Erlerntem Wissen auf zukünftige Projekte. Primäres Ziel ist dabei jedoch der aktuelle Anwendungsfall sowie eine positive Erledigung der*

		Erwartungshaltung des Projektleiters und dessen Anforderungen.
Entwickler	bernhard.hartmann@stud.fh-campuswien.ac.at	*Auseinandersetzen mit einer neuen Technologie, sowie das Anwenden von Erlerntem Wissen auf zukünftige Projekte. Primäres Ziel ist dabei jedoch der aktuelle Anwendungsfall sowie eine positive Erledigung der Erwartungshaltung des Projektleiters und dessen Anforderungen.*

2. Randbedingungen

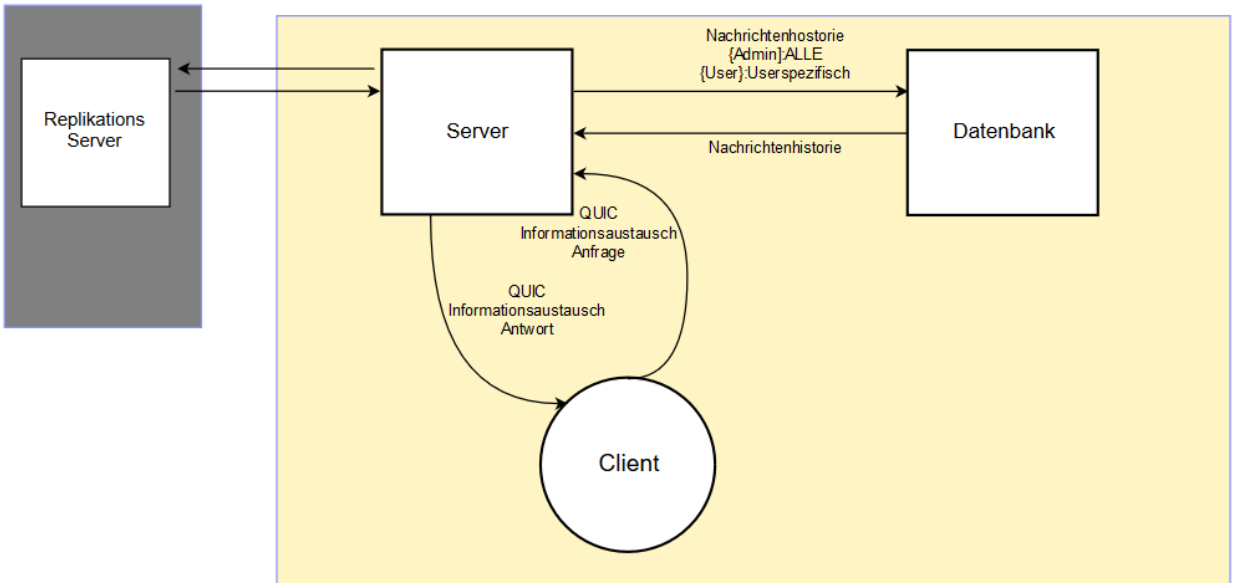
Zu den Randbedingungen und Vorgaben, welche die Freiheiten bezüglich des Entwurfs, Implementierung oder des Entwicklungsprozesses einschränken gehören einerseits die negativen Abgrenzungen der zu entwickelnden Funktionalitäten und andererseits politische und rechtliche Einschränkungen. Die zu beachtende Randbedingungen lassen sich wie folgt darstellen:

Randbedingungen		
Art	**Beschreibung**	**Definition**
Technisch	Telefonate	Es wird kein Verbindungsaufbau mit entsprechender Sprachübertragung über dieses Protokoll im Rahmen dieses Projektes stattfinden.
Technisch	Videokonferenzen	In weiterer Folge grenzen wir uns auch klar von jeglicher Art von Video ab. Es soll keine Art von Streaming implementiert werden.
Technisch	Bild	Eine weitere Art der Medienübertragung beinhaltet Bilder in jeglicher Form – hier wird auch klar abgegrenzt.
Technisch, politisch	Verhinderung von „Screenshots“ der einzelnen Nachrichten	Eine Funktion auf die in unserer Applikation nicht eingegangen wird ist die des Screenshots eines Devices oder OS. Hier wird von der Applikation nicht darauf Rücksicht genommen.
Politisch, organisatorisch	Ablage und Austausch von Dokumenten	Auch das Verhindern des Speicherns lokaler Dokumente mit vertrauten Daten wird von der Applikation nicht unterstützt .In diesem Zusammenhang auch nicht in der Kommunikation da sich diese rein auf den Austausch von Text konzentrieren wird.

Rechtlich	Non Disclosure Agreement	Jedes M&A-Geschäft ist mit einem Non Disclosure Agreement versehen, das komplette Verschwiegenheitspflicht verlangt. Sorgloser Umgang mit Daten löst Schadenersatzpflicht aus.
Rechtlich	Allgemeines Zivilrecht	Beweisvorlage und Nachweis der Authentizität hinsichtlich der erfolgten Kommunikation unterliegen den Bestimmungen des Zivilrechts
Rechtlich	Datenschutz-bestimmungen	Die Datenerhebung unterliegt den Datenschutzbestimmungen und verlangt idR eine Bewilligung des Betroffenen

Anm.: Definierte technische Randbedingungen, welche oben angeführt worden sind, dienen rein der Abgrenzung für den Scope in diesem Projekt. Ziel wäre natürlich viele dieser Abgrenzungen in einem Nachfolgeprojekt näher zu betrachten und Fokus auf deren Implementierung zu legen. Das Projektteam sieht in diesem Zusammenhang nämlich auch großes Potential. Die rechtlichen Randbedingungen sind dagegen unveränderbar.

3. Kontextabgrenzung



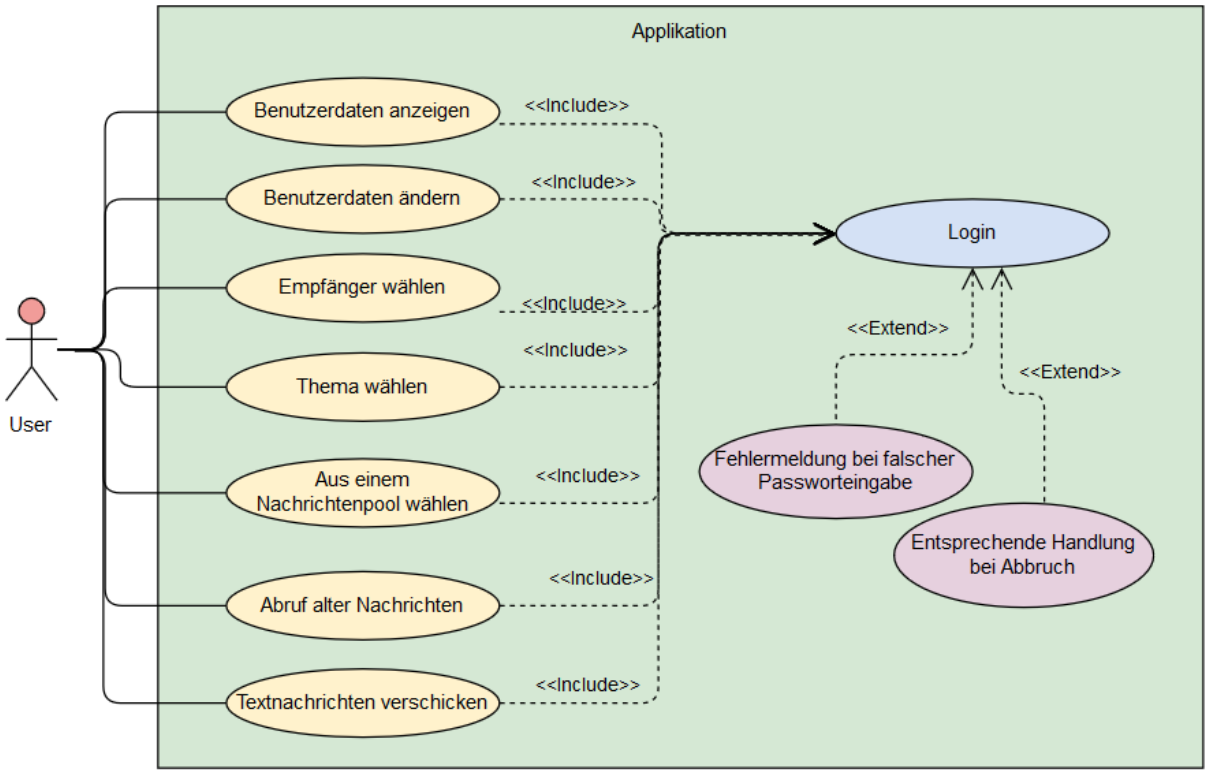
Da eine Client Server Verbindung zahlreiche Möglichkeiten mit sich bringt jedoch aufgrund der unerforschten Technologie Ungewissheiten vorherrschen, wurde in der Kontextabgrenzung, wie im nachstehenden Bild veranschaulicht, eine Grauzone definiert. Das primäre Ziel dieser Applikation ist eine Client Server Verbindung mittels des zu erforschenden QUIC Netzwerkprotokolls, um hier jedoch auch einen Mehrwert für den End User dieser Applikation bieten zu können wird auch eine Userverwaltung sowie eine Verwaltung einer Nachrichtenhistorie, welche von enormer Wichtigkeit ist, umgesetzt. In diesem Zusammenhang würde sich eine Datenbank eignen, welche sich um die Verwaltung kümmert.

Als weiteren Mehrwert, welcher sich jedoch in einer Grauzone außerhalb des Systems bewegt, ist, einen Replikationsserver zu implementieren.

Möglichkeiten welche mit IP-basierten Systemen möglich sind wie bspw. der Video oder Sprachenaustausch sehen wir nicht im Scope dieses Projektes und ist deshalb wie in dem der Grafik beschriebenen Informationsaustausch ausgeschlossen.

Listen von Kommunikationsbeziehungen mit deren Schnittstellen. Eine genauere Beschreibung findet sich in Kapitel 5 Bausteinsicht.

3.1. Fachlicher Kontext

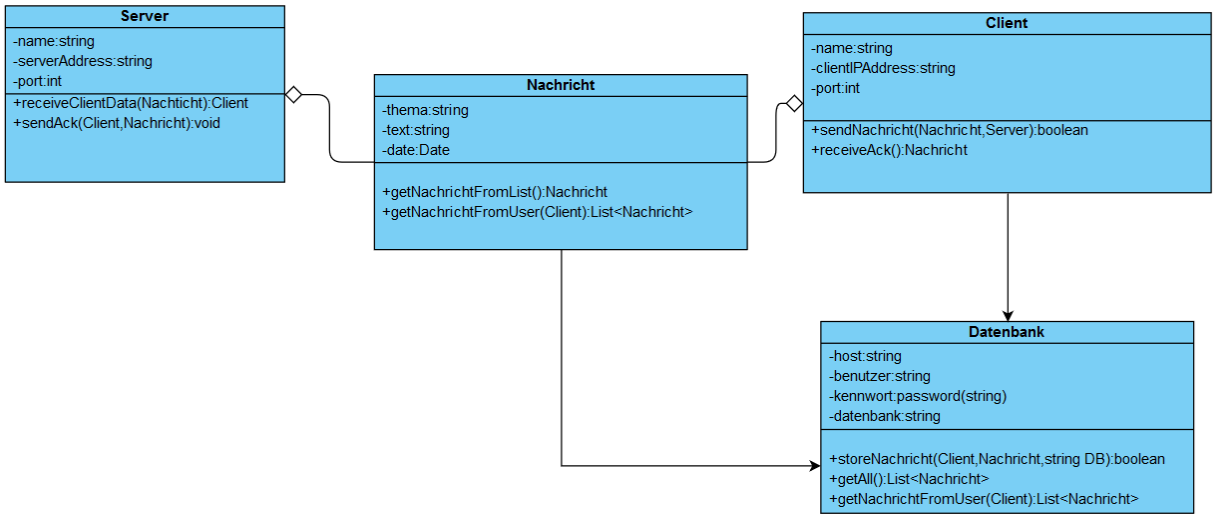


Auf der grafischen Benutzeroberfläche soll es dem User möglich sein seine Benutzerdaten anzuzeigen und auch ändern zu können. In weiterer Folge ist der User in der Lage einen Empfänger aus einer Liste zu wählen, um einen geeigneten Empfänger seiner zu erfassenden Nachricht hinzuzufügen. Zudem ist es möglich und in weiterer Folge auch verlangt ein Thema der Nachricht zu wählen, um diese einer Kategorie zuzuweisen. Es besteht auch die Möglichkeit aus einem Nachrichtenpool vorgefertigte Nachrichten auszuwählen, um Standardabfragen schnell und einfach zu bedienen. Ein zusätzliches wesentliches Featur, welches der Benutzer in der Oberfläche vorfindet ist das Abrufen alter Nachrichten um so wieder einen Kontext alter Konversationen zu bekommen. Die wesentliche Funktion die Nachricht

im Endeffekt auch erfolgreich abschicken zu können soll dem User dann nun auch nicht verwehrt bleiben.

Blackbox Sicht Als Black Box bezeichnet man ein System, von welchem im gegebenen Zusammenhang nur das äußere Verhalten betrachtet werden soll. Die innere Struktur auch wenn bekannt, wird in der Blackbox Sicht nicht bekanntgegeben. Man beschränkt sich bei der Untersuchung und Beschreibung auf die Messung der Input-Output-Beziehungen. Input in diesem Fall ist der Input der QUIC Client und der Output wird in der Grafik als Server wiedergespiegelt.

UML Klassendiagramm



Das UML Klassendiagramm stellt die Beziehungen der definierten Hauptklassen sowie deren Methoden und Attributen dar. Folgende Tabelle beschreibt diese Beziehungen untereinander.

Beziehungen:

Klasse1	Klasse2	Beziehung	Erklärung
Client	Nachricht	M:N	Mehrere Clients können mehrere Nachrichten verschicken
Client	Server	N:1 (M)*	*Beziehung ändert sich bei aktiver Replikation
Nachricht	Server	N:1 (M)*	*Beziehung ändert sich bei aktiver Replikation
Server	Datenbank	1:n*	*Beziehung ändert sich bei aktiver Replikation *Beziehung ändert sich bei aktiver Replikation Vorgesehen ist eine Datenbank pro Server die Möglichkeit hier aufzustocken ist jedoch nach einer Instanz eine leichte

3.2. Technischer Kontext

Technische Schnittstellen, mit denen dieses System in Verbindung kommt, ist unter anderem die Betriebssystem Firewall. Diese könnte bei Kommunikation nach außen Probleme machen und müsste gegebenenfalls konfiguriert oder gar abgeschalten werden. Ein und Ausgaben des Systems werden über das definierte Netzwerkprotokoll abgehalten oder über das Phyton Framework Django. Die Funktionsweise beider Systeme ist in Kapitel 5 Bausteinsicht genauer erläutert. Schnittstellen, welche im Ausmaß des Systems bezüglich Hardware definiert werden müssten, halten sich in Grenzen, somit sind auch keine besonderen Anforderungen gefordert. Bei aktiver Replikation müssen diese jedoch wieder erneut in Augenschein genommen werden. Eine MySQL Datenbank ist an das System angebunden und ist im Zusammenhang eines technischen Kontextes nicht unzulänglich.

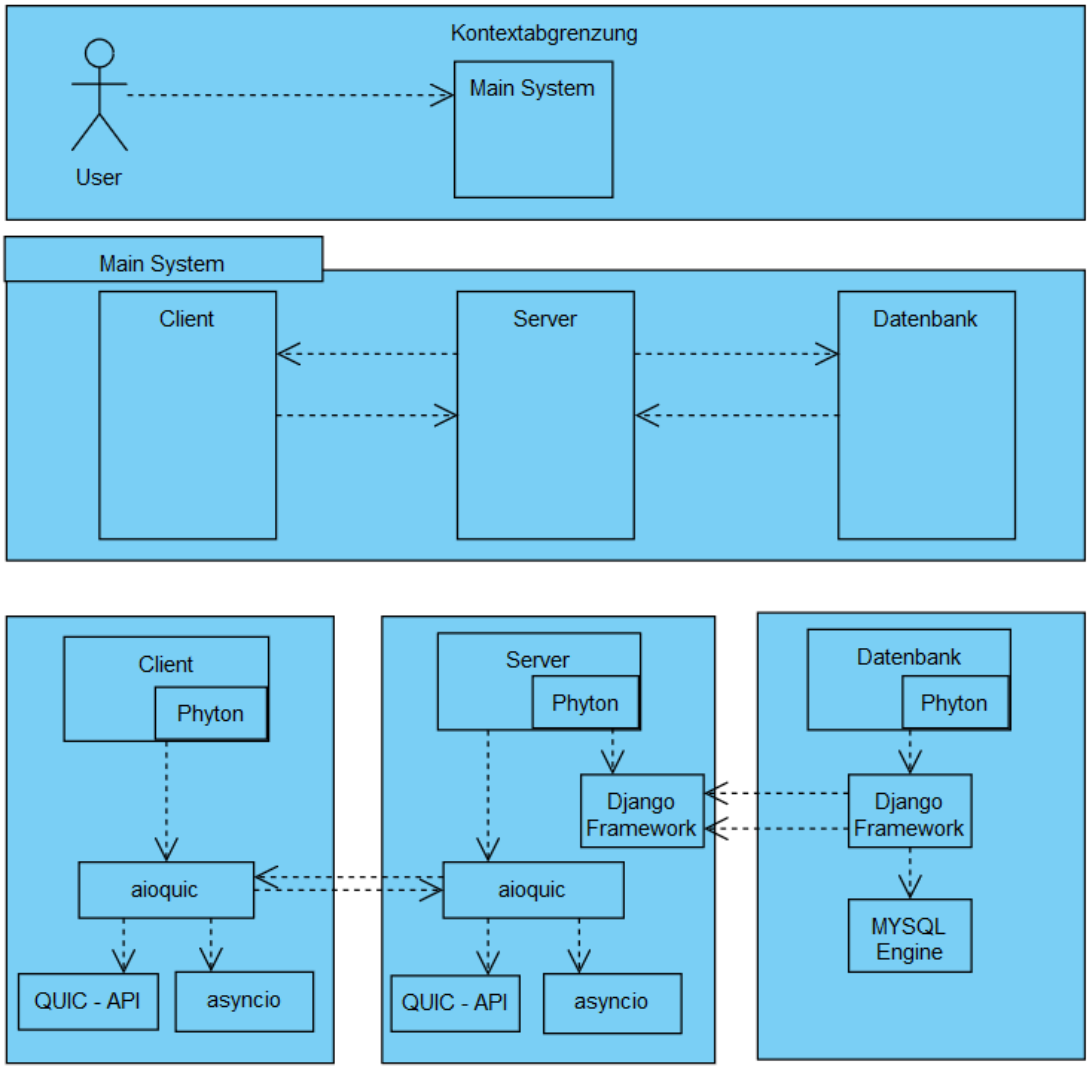
4. Lösungsstrategie

Wesentliche Anforderungen in diesem Projekt ist die Geheimhaltung und Datensicherheit, sodass für die Datenübermittlung das vom Google entwickelte Netzwerktransportprotokoll “QUIC” gewählt wurde. Aufgrund der beabsichtigten Nachhaltigkeit wird die von IETF-Draft vorgegebene Version von QUIC in Betracht gezogen. Das QUIC-Protokoll hat nämlich im Gegensatz zu TCP die gewünschte Eigenschaft, dass sämtlicher Message-Inhalt (inklusive Metadaten) verschlüsselt übermittelt werden kann.

Die Implementierung wird mit Hilfe der bereits entwickelten Libraries erfolgen. Da das QUIC-Protokoll noch keine weitgehend etablierte Technologie darstellt, sind die zur Auswahl stehenden Möglichkeiten, die die vollen Funktionalitäten von QUIC bieten können, eingeschränkt. Es bestehen mehrere Risiken (siehe Abschnitt 9), die zur Entwicklung nach dem “Spiral Model” bewegt haben, wobei die Funktionalitäten der Anwendung systematisch in jedem Kreislauf weiterentwickelt werden. Dadurch sollte sichergestellt werden, dass ein außerplanmäßiger Abbruch der Entwicklung ein trotzdem nützliches Produkt zur Folge hat.

5. Bausteinsicht

Die folgende Bausteinsicht unseres Systems wird in 3 Ebenen gebrochen. Diese Ebenen werden nachfolgend genauer durchleuchtet und beschrieben.



Ebene 1 ist die Whitebox-Beschreibung des Gesamtsystems, zusammen mit Blackbox-Beschreibungen der darin enthaltenen Bausteine.

Ebene 2 zoomt in einige Bausteine der Ebene 1 hinein. Sie enthält somit die Whitebox-Beschreibungen ausgewählter Bausteine der Ebene 1, jeweils zusammen mit Blackbox-Beschreibungen darin enthaltener Bausteine.

Ebene 3 zoomt in einige Bausteine der Ebene 2 hinein

5.1. Whitebox Gesamtsystem

Das System wurde auf 3 Ebenen gebrochen damit eine genaue Veranschaulichung in Tiefen des Systems gewährleistet und veranschaulicht wird. Ein tieferer Einblick des Systems wäre aufgrund der zugrunde liegenden Komplexität sinnfrei. Die Veranschaulichung in diesen 3 Ebenen ist somit mehr als ausreichend sowie aussagekräftig genug um die Art der Umsetzung anhand der jeweiligen Technologie zu verstehen.

Enthaltene Bausteine

Die erste Ebene besteht aus einem wesentlichen Baustein nämlich dem Main System. Dieser definiert das grobe Vorhaben im Projekt als Ganzes.

5.2. Ebene 2

Ebene zwei fungiert als Whitebox der ersten Ebene und wird in 3 Bausteine gebrochen.

Whitebox Main System

Das Main System besteht aus den 3 Hauptkomponenten des Systems.

Enthaltene Bausteine

Diese Hauptkomponenten umfassen einen Client einen Server sowie eine Datenbank.

Wichtige Schnittstellen

Essentielle Schnittstellen umfassen die Kommunikation unter den Bausteinen dieser Schicht. Dabei kommuniziert der Client mit dem Server und der Server mit der Datenbank. Die Datenbank ist ein wesentlicher Bestandteil dieser Lösung, welche am Server gehostet wird. Der Client kann sich dann alle nötigen Informationen über eine sichere Serververbindung von dieser Datenbank abrufen. Ein wesentlicher Vorteil davon ist die Dezentralisierung der Datenbank auf dem Server sowie die Sicherheit des QUIC Protokolls in der Datenübertragung. Eine lokal abgespeicherte Datenbank wäre ungünstig da wiederum die Sicherheit der abgespeicherten Daten gefährdet wäre.

Client

Der Client baut eine Verbindung mit dem Server basierend auf dem Netzwerkprotokoll QUIC auf. Somit steuert er die Initialisierung sowie Termination des Systems. Wesentliche Leistungsmerkmale sind in der Laufzeitsicht im Abschnitt 6 wiedergespiegelt in Form von Verhaltensweisen bei nicht Erreichung eines Adressaten.

Server

Der Server ist für zweierlei Prozesse verantwortlich und in der Architektur nicht wegzudenken. Zum einen verwaltet er die eingehenden Kommunikationsnachrichten und zum anderen die Abspeicherung dieser in der Datenbank. Somit ist dieser ein wesentlicher Bestandteil des Systems zumal in der aktuellen Architektur keine Redundanz vorhanden ist. Wesentliche Leistungsmerkmale sind in der Laufzeitsicht im Abschnitt 6 wiedergespiegelt in Form von Verhaltensweisen bei nicht Erreichung der Datenbank.

Datenbank

Der Datenbank speichert jegliche vom Client kommende Kommunikation, welche auf dem Server ankommt, ab. Leistungsmerkmale lassen sich von der zu speichernden Datenmenge wie deren Abfragegeschwindigkeit messen. Bei nicht erreichen der Datenbank kann keine Kommunikation stattfinden. Wesentliche Leistungsmerkmale sind in der Laufzeitsicht im Abschnitt 6 wiedergespiegelt in Form von Verhaltensweisen.

5.3. Ebene 3

Da sich in dieser Ebene mehrere Bausteine befinden wurde hier jeweils nur auf die wesentlichen eingegangen und diese genauer betrachtet.

Client Baustein “aioquic”

aioquic ist eine Bibliothek für das QUIC-Netzwerkprotokoll in Python. Es verfügt über mehrere APIs:

- eine QUIC-API nach dem Muster "bring your own I / O", die zum Einbetten in ein beliebiges Framework geeignet ist,
- eine HTTP / 3-API, die auch das Motto "bring your own I / O" verfolgt
- eine QUIC-Convenience-API, die auf asyncio, dem Standard-Asynchronous-I / O-Framework von Python, aufbaut.

Server Baustein Django Framework

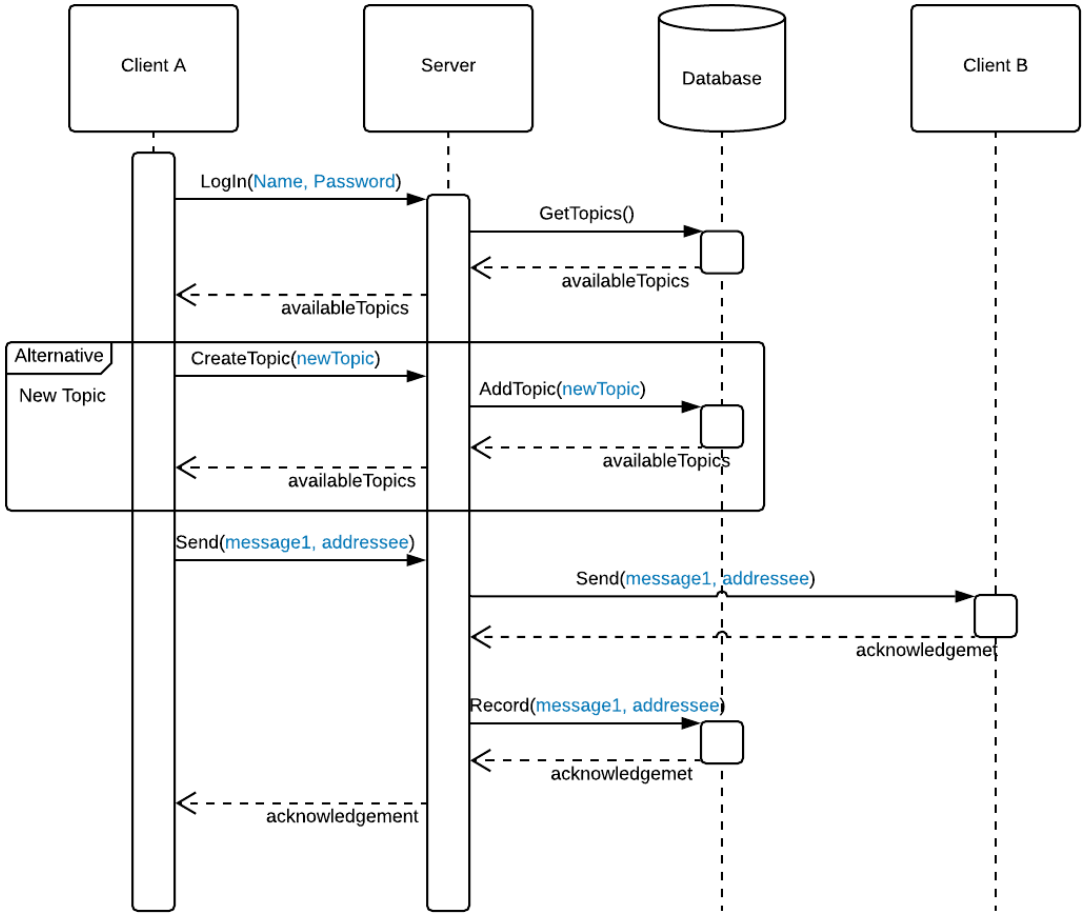
Django ist ein in Python geschriebenes Full Stack Framework, dass die schnelle Entwicklung von Web-Applikationen ermöglicht. Dabei wird Wert auf sauberen Code und die Wiederverwendbarkeit der einzelnen Komponenten gelegt.

Datenbank Baustein MySQL

MySQL ist eines der weltweit verbreiteten relationalen Datenbankverwaltungssysteme. Es ist als Open-Source Software sowie als kommerzielle Enterpriseversion für verschiedene Betriebssysteme verfügbar und bildet eine Grundlage für viele dynamische Webauftritte.

6. Laufzeitsicht

Der Normalverlauf der Kommunikation lässt sich wie folgt darstellen:



Die einzelnen Schritte werden in der Regel synchron ausgeführt. Dies hat folgenden Hintergrund:

Wie im Anforderungsdokument SRS Red Room Discussion Channel vom 4. Dezember 2019, Version 0.2 beschrieben dienen die in der Datenbank gespeicherten Informationen als potentieller Beweis der Geschäftsgrundlage bei einem M&A Deal. Diese ist aber nur dann gegeben, wenn die Information von beiden Parteien erhalten wurde. Die Speicherung der Nachrichten (als Beweismaterial) findet dementsprechend erst nach der Zustellung statt.

6.1. Normalverlauf

Im oben angeführten Beispiel will Client A mit Client B kommunizieren, was in den folgenden Schritten ausgeführt wird:

1. Client A logt sich über die client-seitige Applikation beim Server ein,
2. Server fragt die Liste der vorhandenen Themen,
3. Datenbank schickt in Response an Server die Liste der vorhandenen Themen,
4. Server liefert in Response an Client A die Liste der vorhandenen Themen aus der Datenbank,
5. Alternativ - falls das gewünschte Thema noch nicht vorhanden ist:
    5.1. Ein neues Thema wird vom Client A an Server übermittelt,
    5.2. Der Server schickt das neue Thema der Datenbank zur Speicherung,
    5.3. von der Datenbank gespeichert und Schritte 3-4 werden wiederholt,
6. Client A schickt eine für Client B bestimmte Nachricht an den Server,
7. Der Server sendet die Nachricht an den Client B,
8. Client B schick ein Acknowledgement über den Erhalt der Nachricht an den Server,
9. Sobald der Server ein Acknowledgement vom Client B über den Erhalt der Nachricht bekommt, sendet er diese Nachricht zur Speicherung an die Datenbank,
10. Die Datenbank speichert die Nachricht und sendet darüber ein Acknowledgement an den Server,
11. Der Server schickt ein Acknowledgement über die Zustellung an Client A.

Es wird prinzipiell von einer verlässlichen Verbindung zwischen dem Server und der Datenbank ausgegangen, weil diese vom Provider der SaaS verwaltet wird. In den untenstehenden Szenarien werden daher nur client-seitige Netzwerkprobleme adressiert.

6.2. Eine Nachricht zwischen den Sender (Client A) und dem Server geht verloren

Der Sender (Client A) erhält kein Acknowledgement über die Zustellung vom Server. Er darf daher dementsprechend nicht davon ausgehen, dass diese Information (seine Nachricht) zum Vertragsinhalt geworden ist – diese ist nämlich bei dem Empfänger nie angekommen. Gleichzeitig wird diese Nachricht von der Datenbank nicht aufgezeichnet, was dem richtigen Stand entspricht (die Datenbank enthält nur zugestellte Willenserklärungen und Informationen).

6.3. Eine Nachricht zwischen dem Server und dem Empfänger (Client B) geht verloren

Der Server erhält kein Acknowledgement vom Empfänger (Client B), und die Nachricht wird von der Datenbank nicht aufgenommen, was mangels Zustellung wieder dem richtigen Stand entspricht.

6.3. Empfänger noch nicht vorhanden

Ist der beabsichtigte Empfänger im System noch nicht vorhanden, muss dieser zuerst als User von einem dazu berechtigten Benutzer (siehe Rollen im Anforderungsdokument) angelegt werden.

6.4. Datenbank nicht erreichbar

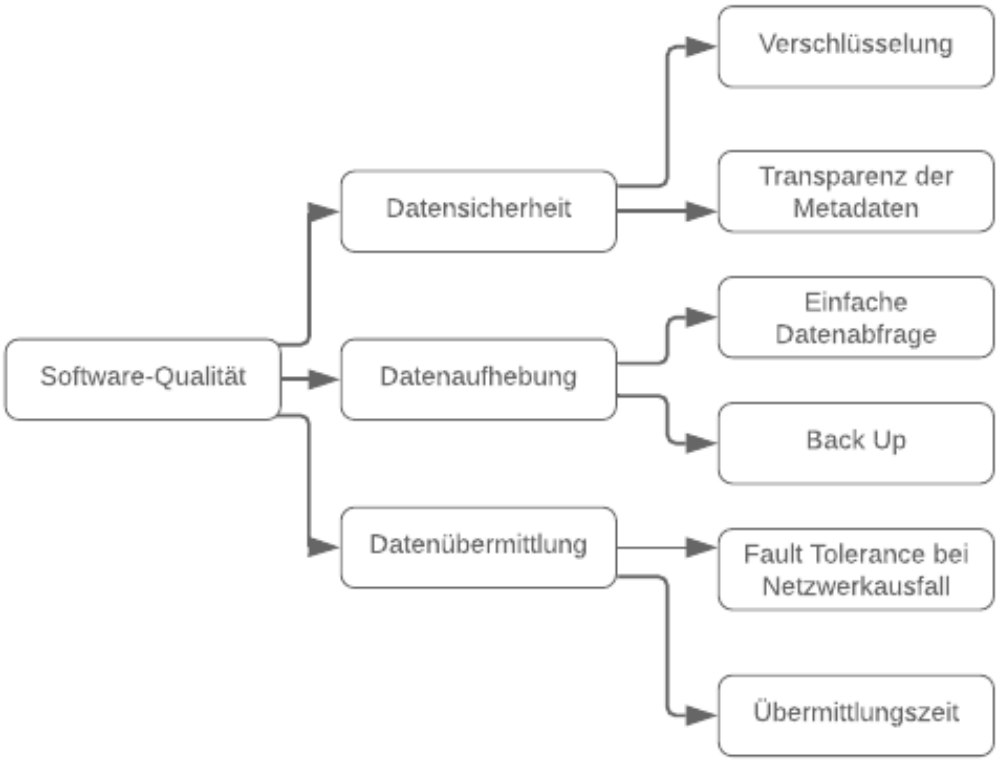
Da die Datenbank in der Architektur eine wesentliche Rolle spielt, ist es so umgesetzt, dass es für einen User des Systems nicht möglich ist bei nicht erreichen der Datenbank Nachrichten zu empfangen wie auch zu senden.

7. Qualitätsanforderungen

Qualitätsziele in der Architektur sind aufgrund einer simplen Client-Server Kommunikation gering gehalten mit dem Fokus auf dem Netzwerkprotokoll, das für die Datensicherheit sorgen soll. Des Weiteren ist die verlässliche Datenaufhebung und -übermittlung von großer Bedeutung. Diese werden untenstehend näher erläutert.

7.1. Qualitätsbaum

Die primären Qualitätsziele lassen sich wie folgt weiter gliedern:



7.2. Qualitätsszenarien

Darstellung der Unterziele und deren Prioritäten		
Priorität	**Beschreibung**	**Erläuterung / Szenario**
1	Verschlüsselung	Die Nachrichten können von unbefugten nicht gelesen werden – sie sind verschlüsselt.
2	Back up	Die übermittelten Nachrichten werden mit einer sicheren Technologie aufgehoben
3	Übermittlungszeit	Die Übermittlung soll nicht länger als 2 Sekunden dauern
4	Transparenz der Metadaten	Die Metadaten sollen nicht frei lesbar sein, sondern verschlüsselt.
5	Fault Tolerance bei Netzwerkausfall	Bei nicht zugestellten Nachrichten gibt es einen erneuten Zustellversuch.
6	Einfache Datenabfrage	Die alten Daten können einfach über die grafische Interface von einem dazu befugten User abgefragt werden.

8. Risiken und technische Schulden

Diese Applikation beinhaltet eine auf QUIC basierende Client-Server Kommunikation. Die Herausforderung dabei liegt ganz klar in der bis dato noch unreifen Technologie, welche noch keine ausgereifte Programmiercommunity bietet.

Es bestehe folgende konkrete Risiken:

Rang	Risiko	Risikobeschreibung	Risikovermeidung
1	Inkompatibilität	Gewählte Technologien lassen sich nicht verbinden (zB aioquic-Library mit Django-Framework)	Vor der Entwicklung der einzelnen Komponente wird der Aufbau der Schnittstellen/Interfaces untersucht; bei Inkompatibilität wird die Architektur angepasst.
2	Knappes Budget	Das berechnete Budget (300 Entwicklungsstunden) wird für vollständige Entwicklung nicht reichen.	Die Entwicklung verfolgt “Spiral Development Model” wobei Funktionalitäten schrittweise aufgebaut und erweitert werden. Sollte das Budget noch vor der Vollbeendigung aufgebraucht werden, wird es somit dennoch ein nützliches Produkt geben.
3	Technologiewechsel	Das QUIC-Protokoll wird von IETF verworfen oder umgekehrt, die	Die Weiterentwicklungen werden laufend beobachtet, die Ziele des

		Implementierungen werden so schnell weiterentwickelt, dass die Bemühungen dieses Projekts gegenstandlos werden (zB aufgrund ausgereiften Libraries für Java oder C#/.NET)	bzw. Vorgaben des Projekts werden bei Bedarf angepasst.

9. Glossar

Wesentlichen fachliche und technische Begriffe:

HTTP HyperText Transfer Protocol. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands

IETF Internet Engineering Task Force is the body that defines standard Internet operating protocols such as TCP/IP

M&A Mergers & Acquisition: Kauf und Konsolidierung von Betrieben im Rahmen von sowohl Asset Deals (Kauf einzelner Wirtschaftsgüter) als Share Deals (Kauf von Beteilungen

PIE Public Interest Entity: gem Art 2 der Richtlinie 2013/34/EU sind Unternehmen von öffentlichem Interesse va börsennotierte Unternehmen, Kreditinstitute,

QUIC Ein experimentelles Transportprotokoll, das von Google entwickelt wurde. Andere Bezeichnungen: qQUIC, iQUIC, HTTP-over-QUIC oder HTTP/QUIC

SaaS Software as a service: In diesem Kontext wird ein Geschäftsmodell gemeint, in dem eine Applikation dem Kunden online für bestimmte Zeit zur Verfügung gestellt wird.