Hypertainment: Aufbau eines flexiblen Multimediasystems

Tobias Braun*
Tobias.braun@student.reutlingen-university.de
Hochschule Reutlingen
MatrikelNr. 767 369

Alexander Bartik*
Alexander.Bartik@student.reutlingen-university.de
Hochschule Reutlingen
MatrikelNr. 812 482

Abstract

[Motivation/Kontext] Konsumenten stehen eine Vielzahl an Geräten und Diensten zur Verfügung, über die digitale Video- und Audioinhalte abgerufen und wiedergegeben werden können. Diese können lokal, in Form von physischen Medien, wie DVDs oder Blueray-Disks, digital auf einem lokalen oder entfernten Server, oder bei einem Streamingdienst vorliegen. **[Problemstellung]** Jeder Hersteller bietet verschiedene Endgeräte und Softwaremöglichkeiten für Entertainmentsysteme an. Darunter zählen verschiedene Wiedergabegeräte, Softwareansätze mit verschiedenen Protokollen und Standards, sowie Hürden bei den Verbindungsmöglichkeiten untereinander. Den Benutzer*innen fehlt dabei die Übersicht über die Verfügbarkeit von Medien. **[Lösungsidee]** In dieser Arbeit wird ein System beschrieben, das es erlaubt, verschiedene Medien aus verschiedenen Quellen zu bündeln und auszugeben. Den Nutzer*innen wird ein einheitlicher Katalog präsentiert, über den sie sich leicht einen Überblick verschaffen können. Dabei wird das Augenmerk auf hohe Flexibilität und einfache Bedienbarkeit gelegt. **[Beitrag]** Diese Arbeit erfasst Anforderungen an ein solches System und erarbeitet, basierend darauf, den Entwurf einer Softwarearchitektur, die diese Anforderungen erfüllt.

CCS Concepts: • **Human-centered computing → Interaction devices**.

Keywords: Entertainment System, Streaming, Architektur, Mobilität

Abkürzungsverzeichnis

In der folgenden Auflistung werden Abkürzungen beschrieben, die in dieser Arbeit verwendet werden:
API - Application Programming Interfaces
CDN - Content Delivery Network
CCDN - Cloud Content Delivery Network
CMCD - Common Media Client Data Standart
DDoS - Distributed Denial of Service
DSGVO - Datenschutzgrundverordnung
HTTPS - Hyper Text Transfer Protocol Secure
TLS - Transport Layer Security
DVI- Digital Visual Interface
HDMI - High Definition Multimedia Interface
HDCP - High-bandwidth Digital Content Protection System
IEEE - Institute of Electrical and Electronics Engineers
ACM - Association for Computing Machinery
IREB - International Requirements Engineering Board
NAS - Network Attached Storage
QoS - Quality of Service

*Both authors contributed equally to this research.

1 Einleitung

Integrierte Multimediasysteme sind im Fernverkehr und vor allem im Flugverkehr bereits weit verbreitet. Diese Systeme geben der Nutzer*in Zugriff auf eine Auswahl an Unterhaltungssoftware. Dies können lokale Quellen, wie Radio, TV [20], aber auch Onlinequellen wie Streamingdienste [18] sein. Studien haben gezeigt, dass umfangreiche Unterhaltungsmöglichkeiten die Zufriedenheit von Fluggästen steigern kann [2]. Diese Systeme erlauben die Auswahl und Wiedergabe der gewählten Medien über integrierte Hardwareschnittstellen wie Displays, Touchscreens, Lautsprecher oder Kopfhörerausgänge. Realisiert werden können diese als zentrales, fahrzeugweites System [20, 457] oder als individuelle Geräte für jeden Nutzer [28]. Studien haben hierbei gezeigt, dass Fluggäste während des Fluges Zugriff auf Onlinequellen, zusätzlich zum angebotenen, lokalen Katalog, wünschen. [12]. Onlinezugriff ist hierbei zum Beispiel in Automobilen, oder Flugzeugen über Mobilfunk [8] oder Satellit möglich.

Multimediasoftware muss in der Lage sein, sowohl mit der Nutzer*in, als auch mit anderen Systemen zu kommunizieren. Die Erstellung eines User-Interfaces (Benutzer- System- Schnittstelle) hat Auswirkung auf den späteren Erfolg des Produkts. So steht eine einfache Bedienung, ohne kompliziertes Nachforschen, im Vordergrund und fördert so die Akzeptanz der Käufer*in [1]. System-System-Schnittstellen sind der Übergang von einem System zu einem Anderen. Dieser Übergang wird zur Kommunikation, Datenaustausch, Ansteuerung und zur Dokumentation verwendet. Eine wichtige Schnittstelle ist jene zum Betriebssystem. Hier bieten die Hard- und Firmware auf der das System ausgeführt werden soll, unterschiedliche Schnittstellen in Form von APIs und Treibern.

[1] Daniel Reinsch. 2018. Software-Schnittstellen: Beschreibung konform IEC 62304. https://www.johner-institut.de/blog/iec-62304-medizinische-software/software-schnittstellen-beschreibung-konform-iec-62304/dokumentation

Das Problem hierbei ist allerdings, dass auf unterschiedlicher Hardware und Software, unterschiedliche APIs und Treiber zur Verfügung stehen. So benötigt die Entwickler*in eine Schnittstelle um Programme schreiben zu können, welche plattformunabhängig sind. Diese kann die Form einer Laufzeitumgebung, eines Adapters oder eines Parsers haben.

1.1 Medienquellen

Auf dem aktuellen Markt existieren eine Vielzahl an konkurierenden Streaminganbietern (Amazon Prime, Spotify, Netflix etc.) [1]. Jeder davon hat Inhalte, die nur bei diesem verfügbar sind, weshalb Nutzer*innen, die diese konsumieren möchten, Zugriff auf jeden individuell benötigen. Diese Video- und Audioinhalte werden bei den meisten Anbietern nicht auf das Endgerät heruntergeladen, sondern von den Servern des Anbieters live gestreamt [3]. Dabei müssen sowohl technische, als auch rechtliche Vorgaben (Datenschutz, Urheberrecht, Jugendschutz etc.) [11] eingehalten werden. Weiter stellen Anbieter keine einheitlichen Schnittstellen zur Verfügung, sondern setzen auf eigene webbasierte Lösungen. Sie bieten eigene properitäre Anwendungen, die zwar Standardprotokolle verwenden, diese jedoch schützen, wodurch sie von Drittanbietersoftware nicht gelesen werden können.

1.2 Audio und Video Streaming

Audio-und Video Streaming begleiten eine Vielzahl an Menschen in der heutigen Zeit. Sei es auf der Heimfahrt von der Arbeit, das Hören von Spotify, ein Film am Abend auf Netflix oder das Livesport-Event am Wochenende. Dabei bietet es den Vorteil, Musik oder Filme nicht zuerst auf das Endgerät herunterladen zu müssen und somit begrenzten Speicherplatz zu sparen. Heutzutage muss dabei nicht auf hochwertige Qualität während dieser Streams verzichtet werden [**?**]. Streaming bedeutet hierbei, dass das gewünschte Medium abgespielt wird, ohne, dass dieses vollständig heruntergeladen wurde. Die Medien können dabei auf einem Computer, einem Medienserver oder einem Network Attached Server gespeichert sein. Ein Netzwerk-Mediaplayer greift daraufhin auf diese Daten zu und kann sie auf dem lokalen Endgerät abspielen ohne diese herunterzuladen. Streaming-Media umfasst dabei wesentlich mehr als Audio und Video. So zählen darunter unter anderem auch Live-Untertitel, Laufschriften und Echtzeittext [19].

1.3 Streaming Clients

Streaming Dienste nutzen, um mit schwankenden Lasten besser umgehen zu können, Serververbunde, genannt CDN [4]. Dabei existiert nicht ein zentraler Server, mit dem alle Clients kommunizieren, sondern viele Content Delivery Server. Für die Kommunikation zwischen den Clients und dem CDN existiert der CMCD [5]. Dieser beschreibt, wie Clients Medien fragmentiert von einem CDN abrufen können. Dazu wird eine Anfrage über HTTP gestellt, in der spezifiziert wird, welches Fragment benötigt wird. Das CDN, leitet diese an einen zugehörigen Server weiter, der dem Client eine Antwort mit den geforderten Inhalten sendet. Der Client muss diese Fragmente einzeln vom CDN anfordern. Die Entscheidung, welches Fragment, wann benötigt wird, liegt beim Client, der diese nach Erhalt zwischenspeichert und, wenn benötigt, abspielt. Nach dem sie wieder gegeben wurden, können die Inhalte aus dem Speicher gelöscht werden. Es existieren weiter auch Per-To-Per Ansätze für das Streaming von Medieninhalten. Dabei sind die Clients selbst Teil des CDN [17]. Dieser Ansatz wird beispielsweise von der Musikstreaming-Plattform Spotify [17] verwendet. Dies bietet den Vorteil, dass die Kapazität des CDN mit der Anzahl der Nutzer steigt. Beim Per-To-Per Ansatz werden Daten vorzugsweise nicht von zentralen Servern abgerufen, sondern von anderen Netzwerkteilnehmern, die diese Inhalte zur Zeit zwischengespeichert haben. Inhalte werden nur dann von den Servern abgerufen, wenn diese nirgends verfügbar sind. Dies hat zusätzlich den Vorteil, dass Inhalte, die zur Zeit populär sind, höher verfügbar sind. Möchte ein Streamingdienst eine Per-To-Per Architektur verwenden, müssen die Clients in der Lage sein, einander zu finden und Daten direkt auszutauschen. Nach Gabriela Gheorghe et al [13] stellen Per-To-Per Architekturen jedoch auch besondere Anforderungen an die Sicherheit des Systems. Da der Betreiber keinen direkten Zugriff auf die Endgeräte hat, besteht hier ein Risiko. Angreifer könnten die Verbindung zwischen den Clients manipulieren, um illegal an Nutzerdaten, oder Streaminginhalte zu gelangen. Sie könnten versuchen Inhalte zu manipulieren, oder zu zensieren, oder das Netzwerk zu Manipulieren und für DDoS-Angriffe zu missbrauchen. Um Medien von CDNs abrufen zu können, muss der Client in der Lage sein, den Katalog der verfügbaren Medien abzurufen, ein spezielles Medium anzufordern, hierfür existiert der *CMCD* Standard, und dieses Medium wiederzugeben. Für letzteres muss der Client in der Lage sein, die eingehenden Video- oder Audiosegmente zu sortieren, zwischenzuspeichern und zum richtigen Zeitpunkt zu dekodieren. Im Fall von Videoinhalten müssen sowohl Audio-, als auch Videoinhalte übertragen werden. Dabei muss sichergestellt werden, dass diese korrekt synchronisiert werden [26].

1.3.1 Cloud-/Content Distribution Networks.

CDNs werden benötigt um eine große Datennutzung zu ermöglichen und die Inhalte sicher an den Endnutzer weiterzuleiten. Diese bestehen aus einem über die Erde verteilten Netz aus Servern und Datenspeichern, welches Medien an verschiedene Stellvertretersysteme leitet und dort repliziert. CDNs ermöglichen dabei eine hohe Flexibilität und eine Verbesserung der im Internet angebotenen Dienste durch Qualitätserhöhung. Ermöglicht wird dies, in dem CDNs, beispielsweise die Latenzzeit bei der Bereitstellung der Inhalte an Kunden verringern.

Cloud Computing wird definiert als; "ein Modell zur Ermöglichung eines allgegenwärtigen, bequemen und bedarfsgerechten Netzzugangs zu einem gemeinsamen Pool konfigurierbarer Rechenressourcen (z. B. Netzwerke, Server, Speicher, Anwendungen und Dienste), die mit minimalem Verwaltungsaufwand und ohne Interaktion mit dem Dienstanbieter schnell bereitgestellt und freigegeben werden können [21]".

Im Vergleich zu CDNs sind CCDNs dahingehend von Vorteil, da bestehende Infrastrukturen nicht an anderen Standorten repliziert werden müssen, um den Abstand zwischen Inhaltsservern und Endnutzern zu verringern. Die Kombination aus CDN und CCDNs ermöglicht ein flexibles System, welches die Anforderungen des CDN erfüllt und zusätzlich wirtschaftlich rentabel ist. Ein CDN-Aufbau, welcher cloudbasiert ist, kann somit die Kapazität dynamisch nach oben und unten skalieren, versteckt die vorliegende komplexe Infrastruktur und ermöglicht ein Leistungsmanagement welches QoS gesteuert ist [27]. In Abbildung 1 ist der Aufbau eines CDN-Netzwerks beschrieben, während in Abbildung 2 der Aufbau eines CCDN-Netzwerks dargestellt wird.

Aufbau eines Content Distribution Networks:

Ein CDN, besteht aus einem Origin Server, einem POP-Server und dem Request Redirecting mechanism. Dargestellt und erklärt sind diese in Tabelle 1.

Origin Server	Leistungsstarkes Speichersystem, enthält alle Inhalte bzw. Metadaten der Inhalte.
POP-Server	In großer Anzahl in vielen Bereichen verteilt, bietet Inhalt auf Grundlage der Benutzeranfoderung an, falls lokal nicht vorhanden, wird der Inhalt vom Ursprungsserver abgerufen. Dies wird daraufhin gespeichert, da Nutzer in der Region ebenfalls darauf zugreifen könnten.
Request Redirecting mechanism	Ziel: dynamische Umleitung auf optimale Server durch Parameter der Qualitätssicherung (Serverauslastung, Latenz, Netzüberlastung etc.).

Table 1. Content Distibution Networks

Figure 1. Aufbau eines CDN [27]

Vorteile von Cloud Content Distribution Networks:

Für CCDNs sprechen diverse Vorteil im Vergleich zu herkömmlichen CDNs. Diese sind in Tabelle 2 dargestellt und beschreiben:

Pay-as-you-go CCDN model	Der Nutzer kann Inhalte mithilfe eines Pay-as-you-go Modells erwerben. Dies ist wesentlich kostengünstiger als der Aufbau der Infrastruktur eines CDN.
Increased point-of-presence	Einfache Näherbringung an den Endkunden durch die Cloud. Es besteht die Möglichkeit, die Reichweite und Sichtbarkeit des CDN auf Abruf zu erhöhen.
CCDN Interoperability	Ermöglicht den Zugang zu größeren Gruppen ohne lokale Infrastruktur vor Ort. So können neue Märkte und Regionen gewonnen werden.
Support for variety of CCDN application	Dynamische Änderungen können vollzogen werden. Darunter fallen die schnelle Expandierung, schnelleres Wachstum, vorhersehbarer- und unvorhersebahrer Burst, sowie eine Reaktion auf Ressourcenänderungen.

Table 2. Cloud Content Distibution Networks

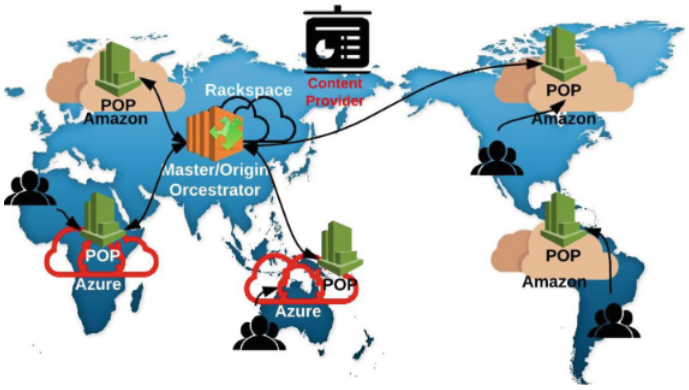

Figure 2. Aufbau eines CCDN [27]

1.3.2 Authentifizierung und Zugangsmanagement bei Online Diensten. Onlinedienste fordern für gewöhnlich, dass der Nutzer ein Profil besitzt und sich für die Benutzung als Inhaber dieses Profils authentifiziert. Üblich ist hier die Eingabe des Nutzernamens und eines Passworts. Da dies jedoch viele Nutzer als unkomfortabel empfinden, wird für gewöhnlich auf die Authentifizierung des aktuellen Geräts über einen Token zurückgegriffen. Dieser Token wird nach der ersten Anmeldung der Nutzers auf dem lokalen Gerät gespeichert und ermöglicht zukünftig die Authentifizierung ohne erneute Eingabe von Nutzerdaten [25].

1.3.3 Protokolle. Protokolle sind ein wichtiger Bestandteil, wenn der Entwickler ein flexibles Multimediasystem aufbauen möchte. Vorliegende Protokolle regeln die Art und Weise, wie Daten zwischen verschiedenen Kommunikationssystemen ausgetauscht werden. Wir betrachten Protokolle, welche für das Audio- und Videostreaming benötigt werden [3].

Protokoll	Beschreibung
NDI|HX	Network Device Interface HX
NDI|HX2	Network Device Interface HX2
ST2022	SMPTE-Standards
ST2110	SMPTE-Standards
SRT	Secure Reliable Transmission
WebRTC	Web Real-Time Communication
RTMP	Real Time Messaging Protocol
HLS	HTTP-Livestreaming
FTP	File Transfer Protocol
SMB	Server Message Block
CMCD	Common Media Client Data

Table 3. Protokolle

1.4 Sicherheitsmechanismen

Eine legale Nutzung von Plattformen wie Amazon,Netflix, Disney etc. sind heutzutage ohne Speicherung von nutzerbedingten Daten nicht mehr möglich. Um diese Daten sicher zu speichern und eine risikoarme Datenübertragung zu gewährleisten, sind mehrere Sicherheitsmechanismen notwendig. Innerhalb Deutschlands regelt die DSGVO [9] den Umgang mit personenbezogenen Daten. In Artikel 5 §1a)-f) der DSGVO werden hierbei die sechs Grundsätze der Datenverarbeitung aufgeführt. Neben der Datenverarbeitung ist auch die gesicherte Datenverbindung zwischen Server und Website, bzw. zwischen Server und Server zwingend notwendig. Die relevanten Paragraphen sind in Tabelle 5 gelistet. Darunter fallen zum Besipiel das Hyper Text Transfer Protocol Secure, die Transport Layer Security, das Secure Sockets Layer sowie das High-bandwidth Digital Content Protection System.

1. Hyper Text Transfer Protocol Secure
HTTPS kombiniert das Netzwerkprotokoll HTTP mit dem krypthografischen Protokoll TLS. Hierbei arbeitet der Client und der Server mit einem Schlüssel, welcher nur diesen Beiden bekannt ist. Der Server schickt dabei verschlüsselte Daten an den Client. Diese Verschlüsselung ist asymmetrisch. Dies bedeutet, es können Daten verschlüsselt aber nicht wieder entschlüsselt werden. Der Client erzeugt daraufhin einen symmetrischen Schlüssel und schickt diesen zum Server zurück. Zurück beim Server kann dieser die asymmetrische Verschlüsselung auflösen und erhält den symmetrischen Schlüssel des Client. Die Kommunikation zwischen Server und Client läuft daraufhin über den geschützten symmetrischen Schlüssel [23].

2. Transport Layer Security/ Secure Sockets Layer
Das TLS Protokoll besteht aus zwei Schichten. Diese sind das TLS Record Protokoll und das TLS Handshake Protokoll. Das TLS Record Protokoll besitzt dabei zwei Eigenschaften. Die Verbindung ist privat und die Daten werden durch symmetrische Kryptografie verschlüsselt. Für jede neue Verbindung wird dieser symmetrtische Schlüssel verwendet, welcher auf der Grundlage eines Schlüssel des TLC Handshake Protokoll basiert [24].

3. High-bandwidth Digital Content Protection System
Intel entwickelte 1999 das High-bandwidth Digital Content Protection System, welches vor allem zum Schutz audiovisueller Inhalte via HDMI, DVI und DisplayPort entwickelt wurde. Es handelt sich hierbei um eine kryptografische Erweiterung des DVI. HDCP wurde von der Filmindustie verpflichtend für alle Geräte eingeführt, welche HD fähig sind. HDCP verhindert hiebei das Kopieren und aufzeichnen bei Videoinhalten in hochauflösender Qualität. HDCP 2.2

Artikel 5 DSGVO	**Beschreibung**
§1a): Rechtmäßigkeit, Verarbeitung nach Treu und Glauben, Transparenz	"[Daten müssen] auf rechtmäßige Weise, nach Treu und Glauben und in einer für die betroffene Person nachvolziehbaren Weise verarbeitet werden."
§1b): Zweckbindung	"[Daten müssen] für festgelegte, eindeutige und legitime Zwecke erhoben werden und dürfen nicht in einer mit diesen Zwecken nicht zu vereinbarenden Weise weiterverarbeitet werden."
§1c): Datenminimierung	"[Daten müssen] dem Zweck angemessen und erheblich sowie auf das für die Zwecke der Verarbeitung notwendige Maß beschränkt sein."
§1d): Richtigkeit	"[Daten müssen] sachlich richtig und erforderlichenfalls auf dem neuesten Stand sein [...]."
§1e): Speicherbegrenzung	"[Daten müssen] in einer Form gespeichert werden, die die Identifizierung der betroffenen Personen nur so lange ermöglicht, wie es für die Zwecke, für die sie verarbeitet werden, erforderlich ist."
§1f): Integrität und Vertraulichkeit	"[Daten müssen] in einer Weise verarbeitet werden, die eine angemessene Sicherheit der personenbezogenen Daten gewährleistet, einschließlich Schutz vor unbefugter oder unrechtmäßiger Verarbeitung und vor unbeabsichtigtem Verlust, unbeabsichtigter Zerstörung oder unbeabsichtigter Schädigung durch geeignete technische und organisatorische Maßnahmen."

Table 5. DSGVO [9]

ID	Frage
RQ1	Was sind die Anforderungen, die Nutzer an ein flexibles Multimediasystem stellen?
RQ2	Wie kann eine Architektur für ein Multimedia System aussehen, das der Nutzer*in erlaubt, viele verschiedene Medien aus verschiedenen Quellen abzurufen und wiederzugeben?

Table 6. Forschungsfragen

welches 2015 veröffentlicht wurde, ist für 4K Inhalte optimiert [**?**].

1.5 Forschungsfragen

Im Zuge dieser Arbeit sollen die Anforderungen an ein flexibles Multimediasystem erfasst werden, um die Forschungsfrage *RQ1* zu beantworten. Basierend auf diesen Anforderungen soll für die Beantwortung von *RQ2* eine Softwarearchitektur entworfen werden.

1.6 Methodik

Dieser Abschnitt beschreibt das Vorgehen zum Beantworten der Forschungsfragen. Zu finden sind diese in Tabelle 6. Um die Forschungsfrage 1 zu beantworten wurde eine Literaturrecherche durchgeführt. Dabei wurde in den Archiven von IEEE und ACM gesucht. Dazu wurde ein Suchterm abgeleitet und mit diesem in verschiedenen Quellen gesucht. Weitere Quellen wurden per Snowball-Methode erfasst. Für die Beantwortung von Forschungsfrage 2 wurde ein Requirements Engineering Prozess nach Pohl und Rupp [22] konstruiert und durchgeführt. Da keine Stakeholder, außer den Autoren dieser Arbeit, zur direkter Verfügung standen, konnte für die Erfassung von Anforderungen nur auf artefaktbasierte Techniken und eigene Erfahrungswerte zurückgegriffen werden.

RQ1	Multimedia AND Entertainment AND system AND Requirements

Table 7. Suchterm

2 Spezifikation

In diesem Abschnitt sollen für die Klärung der Forschungsfrage 1, zu finden in Tabelle 6, die Anforderungen an ein flexibles Multimedia System erfasst werden. Dafür wurde der in 1.6 erwähnte Requirements Engineering Prozess durchgeführt.

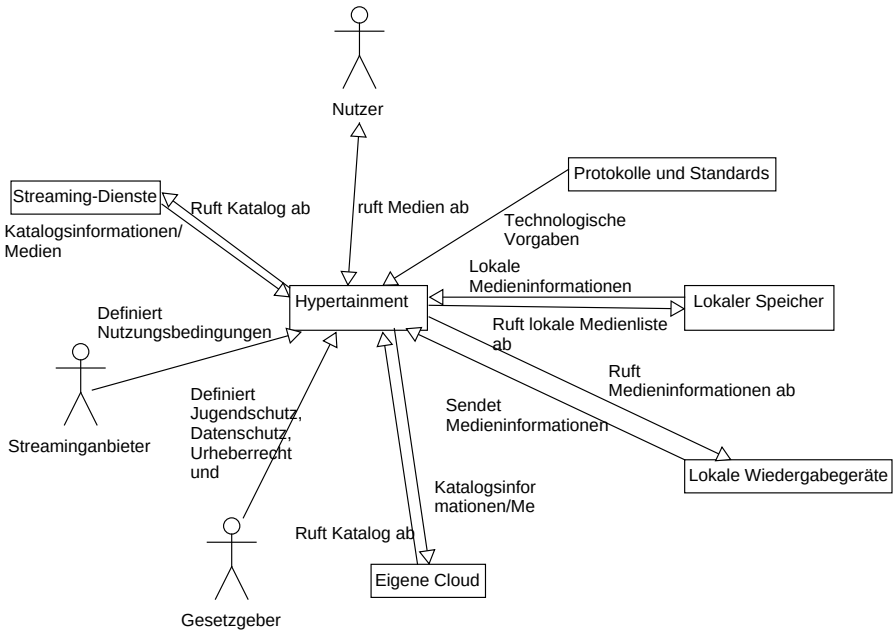

Figure 3. System-Kontextdiagramm

2.1 Anforderungserhebung

Um festzustellen, welche Funktionalitäten das System erfüllen muss und welche technischen und regulatorischen Vorgaben einzuhalten sind, müssen zuerst alle, für das System relevanten, Stakeholder*innen identifiziert werden. Dazu wird eine Kontextanalyse durchgeführt, der Systemkontext modelliert, sowie Use-Cases erfasst.

2.1.1 Kontextanalyse. Um Stakeholder*innen und deren Interaktionen mit dem System zu identifizieren, wurde zuerst der Systemkontext, anhand der Literatur Recherche, sowie Erfahrungswerten der Autoren, analysiert. Dieser wird in Abbildung 3, in Form eines Kontextdiagramms [22, 42] dargestellt.

2.1.2 Stakeholder Identifikation. Im Zuge der Kontextanalyse wurden, die für das System relevanten, Stakeholder*innen identifiziert. Zu sehen sind diese in Abbildung 3. Die identifizierten Stakeholder*innen sind:
Nutzer*innen die, über das System, auf eigene Streamingdienste und Medien zugreifen möchten. **Streaming-Dienste** von denen Medien abgerufen werden. **Streaming-Anbieter** die Vorgaben für den Zugriff und die Wiedergabe von Inhalten ihrer Plattformen aufstellen. **Gesetzgeber** der rechtliche Vorgaben zum Jugendschutz, Datenschutz, Urheberrecht und Vertragsrecht gibt. **Eigene Cloud** auf der, die Nutzer*in, private Medien gespeichert hat, die sie abrufen möchte. **Lokale Wiedergabegeräte** die, die Nutzer*in, mit dem System verbinden möchte um physische Medien, wie BlueRay-Disks oder DVDs abzuspielen. **Lokale Speicher** auf denen die Nutzer*in ebenfalls private Medien gespeichert hat, wie NAS Server oder PCs im Netzwerk. **Protokolle und Standards** für das Streamen von Medien, die das System implementieren oder einhalten muss, um technische und rechtliche Kompatibilität zu anderen Systemen zu gewährleisten.

2.2 Use-Case Identifikation

Im Zuge der Kontextanalyse wurden ebenfalls die relevanten Use-Cases definiert. Diese sind in Abbildung 4 zu sehen:
UC1: Spiele Medium ab Der Nutzer möchte Medieninhalte aus verschiedenen Quellen, wie Streamingdiensten, lokalen Wiedergabegeräten, lokalen Speichern oder der eigenen Cloud wiedergeben. Ein möglicher Teilprozess hierfür ist in Abbildung 6 beschrieben. **UC2: Prüfe Verfügbarkeit eines Mediums** Der Nutzer möchte prüfen, ob ein Medium in einem unterstützten Streaming Dienst, im lokalen Speicher oder der eigenen Cloud verfügbar ist. **UC3: Bereite Medium für spätere Wiedergabe vor** Der Nutzer möchte ein Medium markieren, um dieses zu einem späteren Zeitpunkt abspielen zu können. **UC4: Katalogsüberblick erhalten** Der Nutzer möchte einen Überblick über verfügbare Medien erhalten, um eine informierte Auswahl treffen zu können. **UC5: Filtere Inhalte** Der Nutzer möchte die Auswahl der angezeigten Medien anhand von Filterkriterien, wie Altersfreigabe oder Genre, einschränken können. **UC6: Prüfe Abonnement** Damit Nutzer*innen einen Streamingdienst benutzen können, möchte dieser, dass die Nutzer*innen sich

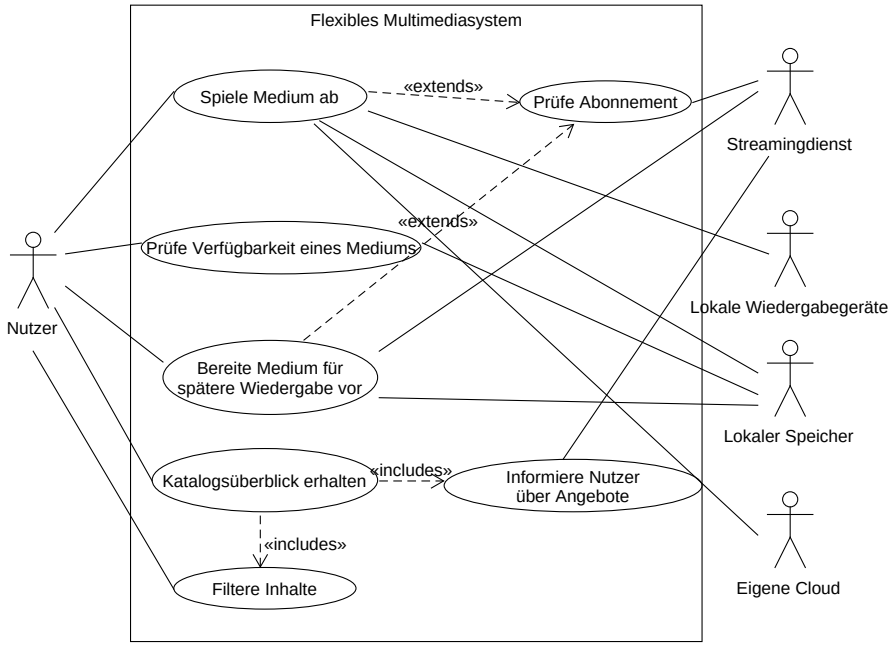

Figure 4. Use-Case-Diagramm

authentifizieren, damit geprüft werden kann, ob und welches Abonnement diese abgeschlossen haben. **UC7: Informiere Nutzer*innen über Angebote** Um der Nutzer*in ein angepasstes Erlebnis zu bieten, möchten Streamingdienste Informationen, wie Neuerscheinungen, Angebote oder Empfehlungen präsentieren.

2.3 Anforderungen

Mit Hilfe der erfassten Literatur wurden Anforderungen der verschiedenen Stakeholder*innen erfasst und nach IREB-Handbuch [14, 10] in Form von User Stories (in Tabelle 8), Qualitätsmerkmalen und Randbedingungen dokumentiert. Da im Zuge dieser Arbeit nicht mit Stakeholder*innen direkt zusammen gearbeitet werden konnte, wurden die Anforderungen ausschließlich aus Erfahrungswerten der Autoren, sowie Artefaktbasiert, aus Literaturquellen erarbeitet [22, 127].

2.3.1 User Stories. Basierend auf den Use Cases wurden Userstories, zu finden in Tabelle 8, erstellt. In diesen werden Aktivitäten, die Stakeholder mit dem System durchführen möchten, beschrieben [14, 47]. Use Cases dienen der Beschreibung gewünschter funktionaler Features.

2.3.2 Qualitätsmerkmale. Für die Identifikation und Klassifikation von notwendigen Qulitätsmerkmalen wurde auf die ISO Norm 25010 [16] zurückgegriffen. Dokumentiert wurden sie in Tabelle 9 in Form von Qualitätsszenarien [14, 40]. Dabei werden Beispiele für Situationen und das gewünschte Verhalten in diesen, oder Vergleiche mit anderen Systemen genutzt, um qualitative Kriterien zu beschreiben. Wichtig ist hierbei, dass Qualitätsmerkmale überprüfbar sein müssen. Entweder durch eine Messung oder einen Versuch.

2.3.3 Randbedingungen. Geplante Anwendungsszenarien und rechtliche, vertragliche Vorgaben geben eine Reihe an Randbedinungen [14, 10] vor. Dies sind unveränderliche Vorgaben, auf die bei der Entwicklung keinen Einfluss genommen werden kann. Diese geben jedoch Einschränkungen für mögliche Lösungen vor. Gelistet sind sie in Tabelle 10.

ID	Beschreibung	Use Case
US1	Als Nutzer*in möchte ich Videoinhalte abspielen können, sodass ich diese konsumieren kann.	UC1
US2	Als Nutzer*in möchte ich Audioinhalte abspielen können sodass ich diese konsumieren kann.	UC1
US3	Als Nutzer*in möchte ich feststellen können, ob ein Medien Inhalt bei einem meiner Streamingdienste, oder in einer lokalen Quelle verfügbar ist.	UC2
US4	Als Nutzer*in möchte ich mich informieren können, welche Quellen Medieninhalte anbieten, auf die ich zur Zeit keinen Zugriff habe.	UC2
US5	Als Nutzer*in möchte ich feststellen können, ob ich ein Medium physisch besitze, um dieses finden und abspielen zu können.	UC2
US6	Als Nutzer*in möchte ich feststellen können, ob ein Medium auf einem lokalen Speicher verfügbar ist, um dieses von dort abspielen zu können.	UC2
US7	Als Nutzer*in möchte ich ausgewählte Inhalte herunterladen, um diese auch ohne Netzwerkverbindung abspielen zu können.	UC3
US8	Als Nutzer*in möchte ich Inhalte in einer Favoriten Liste speichern können, um diese später einfach wiederzufinden.	UC3
US9	Als Nutzer*in möchte ich Medieninhalte von verschiedenen Streaming Diensten abrufen können, um diese zu konsumieren.	UC4
US10	Als Nutzer*in möchte ich eine Übersicht über alle verfügbaren Inhalte erhalten, um eine Auswahl treffen zu können.	UC4
US11	Als Nutzer*in möchte ich Jugendschutzeinstellungen vornehmen können, um einzuschränken, welche Inhalte wiedergegeben werden können.	UC5
US12	Als Nutzer*in möchte ich Medieninhalte nach Genres filtern können, um die Auswahl zu vereinfachen.	UC5
US13	Als Nutzer*in möchte ich Medieninhalte nach Art filtern können, um die Auswahl zu vereinfachen.	UC5
US14	Als Nutzer*in möchte ich Medieninhalte nach Altersfreigabe filtern können, um die Auswahl zu vereinfachen.	UC5
US15	Als Streamingdienst möchte ich das Abonnement der Nutzer*in verifizieren, um festzustellen, auf welche Inhalte dieser zugreifen kann.	UC6
US16	Als Steamingdienst möchte ich der Nutzer*in ein Angebot machen, falls dieser noch nicht Kunde ist, um möglicherweise einen neuen Kunden anzuwerben.	UC7
US17	Als Streamingdienst möchte ich die Nutzer*in über aktuelle Angebote informieren, um den Nutzer an den Streamingdienst zu binden.	UC7
US18	Als Streamingdienst möchte ich die Nutzer*in über neu Erscheinungen informieren, um den Nutzer an den Streamingdienst zu binden.	UC7

Table 8. User Stories

ID	Beschreibung	Quelle
QS1	Das System muss auf Eingaben des Nutzers innerhalb von 100 ms reagieren.	[7]
QS2	Das System muss auf einem handelsüblichen Smart-TVs (Samsung, Sony, Phillips etc.) fehlerfrei funktionieren.	
QS3	Der Nutzer benötigt keine externen Hilfsmittel, um sich im System zurechtzufinden.	
QS4	Bei der Bedienung können keine Szenarien auftreten, in denen eine Nutzereingabe zum Ausfall des Systems führt	
QS5	Das System muss Updates und ein Wiederherstellen der Software ermöglichen	
QS6	Das System muss mit anderen Systemen zusammenarbeiten und koexistieren können	

Table 9. Qualitätsszenarien

ID	Beschreibung
RB1	Das System muss auf gängigen Smart-TVs (Samsung, Sony, Phillips etc.) und Streaminggeräten lauffähig sein.
RB2	Das System muss die Nutzungsbedinungen der unterstützten Streamingdienste einhalten.
RB3	Das System muss rechtliche Vorgaben zum Datenschutz beachten [10].
RB4	Das System muss rechtliche Vorgaben zum Jugendschutz beachten [6].

Table 10. Randbedingungen

3 Entwurf

In diesem Abschnitt wird für die Klärung der Forschungsfrage 2, zu finden in Tabelle 6, eine Softwarearchitektur erarbeitet, die die Anforderungen aus Abschnitt 2.3 erfüllt.

3.1 Architekturentscheidungen

Im Zuge der Entwicklung einer Softwarearchitektur, basierend auf den erhobenen Anforderungen müssen Architekturfragen gestellt und in Form von Architekturentscheidungen beantwortet werden [29, 64]. In diesen werden wichtige Entscheidungen, Einflussfaktoren und die gewählte Lösung festgehalten.

AE1.Wie sieht die grundlegende Struktur des Systems aus? Es existieren verschiedene Grundlegende Strukturparadigmen für den Aufbau eines Softwaresystems. Relevante Strukturen hier sind:

- Monolithisch: Das System besteht aus einem einzigen Programm, das alle Abläufe steuert
- Einfacher Client-Server: Das System besteht aus einer Client-Applikation, die die Benutzeroberfläche bereitstellt und einem Server, der Medien und Zugriffe verwaltet.
- Micro Services: Das System besteht aus einer Ansammlung von Diensten. Jeder Dienst erfüllt einen fest definierten Funktionsumfang.

Es wurde sich für den Aufbau des Systems als Client-Server-Architektur entschieden. Dadurch können Server und Client auf dem selben, oder auf unterschiedlichen Geräten ausgeführt werden. Dies erhöht die Kompatibilität mit Geräten, da der Server, wenn nötig auch extern, zum Beispiel in der Cloud ausgeführt werden kann. Dies erlaubt auch eine einfache lokale Installation und Ausführung.

AE2.Wie kann die Liste der verfügbaren Medien in der Software erzeugt und verwaltet werden? Die Software kann dem Nutzer einen Katalog aller verfügbarer Medien präsentieren. Dieser kann lokale Medien, Medien von verfügbaren Servern, aus den Cloud Konten des Nutzer und aus den verfügbaren Streaming Diensten enthalten. Mögliche Lösungen für das Erzeugen dieses Katalogs sind:

- Direkter Zugriff auf APIs: Der Katalog wird aus verschiedenen Datenquellen aufgebaut. Die Kataloge der einzelnen Streamingdienstleister werden direkt abgerufen. Die Liste der lokalen Medien muss dazu abrufbar gespeichert sein.
- Lokale Datenbank: Der Katalog wird aus den verschiedenen Quellen, lokal und online, abgerufen und lokal gespeichert.
- Zentrale Datenbank: Der Katalog wird aus den verschiedenen Quellen, lokal und online, abgerufen und auf einem zentralen Server gespeichert.
- Hybride Datenbank: Der Katalog wird aus lokalen Quelle abgerufen. Lokale Medien werden in einer lokalen Datenbank gespeichert. Der Katalog von Streamingdiensten wird von einem zentralen Server abgerufen.

Um die schnellen Reaktionszeiten zu gewährleisten, wird der Hybride-Datenbankansatz gewählt. Informationen über Streaminginhalte werden von einem zentralen Server abgerufen und müssen nicht von jedem Client selbst erfasst werden. Informationen über lokale Inhalte werden lokal gespeichert und sind durch den Client direkt abrufbar, auch ohne, oder bei schlechter Netzwerkverbindung.

AE3.Wie sollen die verschiedenen Streaming Dienste angesprochen werden? Die Software soll in der Lage sein, den Katalog von verschiedenen Streamingdiensten abzurufen. Mögliche Lösungen hierfür sind:

- Apps: Das System stellt für jede Quelle eine eigene App für den Zugriff zur Verfügung. Solche Apps existieren für die meisten Streamingdienste bereits.
- Webschnittstelle: Alle populären Streamingdienste erlauben den Zugriff auf ihren Katalog über eine Website. Dabei stellen jedoch Kopierschutz und Nutzungsbedingungen ein Problem beim Abrufen der notwendigen Informationen dar.
- Webscraper: Das System kann Informationen über verfügbare Inhalte über den Zugang des Nutzers im Hintergrund abrufen, sammeln und der Nutzer*in in angepasster Form präsentieren.
- Zentrale Datenbank: Das System kann Informationen über die Verfügbarkeit von Inhalten aus einer Datenbank abrufen. Dienstleister wie die IMDb.com Inc. [15] stellen hier Datenbanken, wie die International Movie Database bereit, die Informationen über Medien und ihre Verfügbarkeit enthalten.
- Individuelle Schnittstellen: Es wird eine eigene Schnittstelle für die direkte Kommunikation mit der API der

Streamingdienste implementiert. Dafür wird der Zugriff auf diese vorausgesetzt.

Es wurde sich für den Zugriff auf Informationen über eine zentrale Datenbank entschieden. Diese kann selbst aufgebaut, oder von einem Dienstleister bezogen werden. Dadurch muss nicht jeder Client individuell die Kataloge der verschiedenen Streamingdienste erfassen.

AE4.Wie sollen Medien Inhalte von verschiedenen Streamingdiensten abgerufen werden? Unsere Software muss in der Lage sein, Medieninhalte von entfernten Streamingdiensten abzurufen und darzustellen. Dazu muss ein Audio- und möglicherweise ein Videostream von den Streamingdiensten empfangen und dekodiert werden. Dabei muss jedoch beachtet werden, dass diese Verbindungen verschlüsselt sind. Mögliche Lösungen sind:

- Proxy über Server: Der Server öffnet auf Anfrage des Clients die Verbindung zum Streamingdienst. Der Medienstream wird vom Anbieter zum Server übermittelt, der diesen dekodiert und für den Client recodiert. Dies würde die Übertragung und Steuerung erleichtern, da der Server alle Medien, egal aus welcher Quelle, gleich kodiert und behandelt.
- Direkte Kommunikation: Der Server informiert den Client über die Quelle des Streams und der Client baut diese Verbindung selber auf. Dadurch hat der Server jedoch keine Kontrolle über den Medienstream.

Es wurde sich für den Einsatz des Servers als Proxy entschieden, da dadurch die Kontrolle über den Stream beim Server liegt. Für den Client macht es keinen Unterschied, aus welcher Quelle ein Medium stammt. Es wird jedoch die Kommunikation zwischen Client und Server, sowie der Client selbst vereinfacht.

AE5.Sollen Medienstreams von externen Quellen durch den Server geleitet werden? Wenn Medien von externen Quellen, wie Streamingdiensten, bezogen werden, können diese direkt vom Client abgespielt werden, oder erst über den Server geleitet werden. Dies kann unterschiedliche gelöst werden:

- Direktzugriff Client: Beim Abruf eines Mediums von einer externen Quelle, erhält der Client vom Server nur die Adresse des Mediums und ruft diese selbstständig ab. Dazu muss der Client Zugriff auf alle externe Quellen haben.
- Server als Proxy: Der Client erhält alle Medien über dieselbe Verbindung. Sowohl lokale, als auch externe Medien werden vom Server abgerufen und an den Client weitergeleitet.

Es wurde entschieden den Server als Proxy zu verwenden wie ebenfalls in AE4 entschieden.

AE6.Wie sollen Nutzerdaten für verschiedene Quellen verwaltet werden? Für den Zugriff auf Streamingdienste und andere Medienquellen werden Zugangsdaten benötigt. Diese können unterschiedlich behandelt werden:

- Kein Speicher: Die Zugangsdaten werden nicht permanent gespeichert. Nach jedem Neustart des Systems muss sich die Nutzer*in bei den benötigten Diensten neu authentifizieren. Dies wäre für die Nutzer*in umständlich und zeitraubend. Außerdem müsste sie alle benötigten Passwörter zur Hand haben.
- Zentraler Zugangsdatenspeicher: Die Zugangsdaten werden in einer Datenbank auf dem Gerät gespeichert und das System authentifiziert sich automatisch nach dem Start.
- Zentraler, geschützter Speicher: Die Zugangsdaten werden in einer Datenbank auf dem Gerät gespeichert. Diese Datenbank ist durch ein Passwort geschützt, das die Nutzer*in nach dem Start eingeben muss.
- Zentraler Tokenspeicher: Nach der ersten Eingabe der Zugangsdaten für die einzelnen Streamingdienste, werden Authentifizierungstokens von diesen angefordert. Diese werden in einem zentralen Speicher abgelegt.
- Zentraler, geschützter Tokenspeicher: Nach der ersten Eingabe der Zugangsdaten für die einzelnen Streamingdienste, werden Authentifizierungstokens von diesen angefordert. Diese werden in einem zentralen, geschützten Speicher abgelegt.

Da Streamingdienste eine Authentifizierung über einen Token ermöglichen, sollen diese zentral in einem geschützten Speicher abgelegt werden. Dieser Speicher ist verschlüsselt und wird durch die Nutzer*in bei der Anmeldung entsperrt. Lokale Quellen, wie NAS-Server, unterstützen jedoch nicht immer die Authentifizierung per Token. Für diese müssen Nutzername und Passwort gespeichert werden. Diese werden im selben, gesicherten Speicher wie die Zugangstokens gespeichert.

3.2 Bausteine

Basierend auf den in Abschnitt 3.1 getroffenen Architekturentscheidungen wurde eine Softwarearchitektur entworfen, die die Anforderungen aus Abschnitt 2.3 erfüllen kann. Die Bausteine dieser Architektur sind in Abbildung 5 zu sehen. Die Applikation ist in einen Server und einen Client unterteilt. Der Server verwaltet alle Medien und Streams und stellt eine Api für die Kommunikation mit dem Client bereit. Der Server kann auf dem selben System wie der Client laufen, oder auf einem Server gehostet werden. Der Client läuft auf dem Endgerät der Nutzer*innen. Dies kann Beispielsweise ein Smart-TV, ein Tablett, oder ein PC sein.

Im Zentrum des Servers steht ein zentraler Medienkatalog, der die individuellen Kataloge der verfügbaren Medien aller Quellen bündelt. Dieser bezieht seine Informationen über Lokale- und Streamingmedien von verschiedenen Quellen. Der *Konntektor für Streamingkatalog*, ruft die Liste der bei

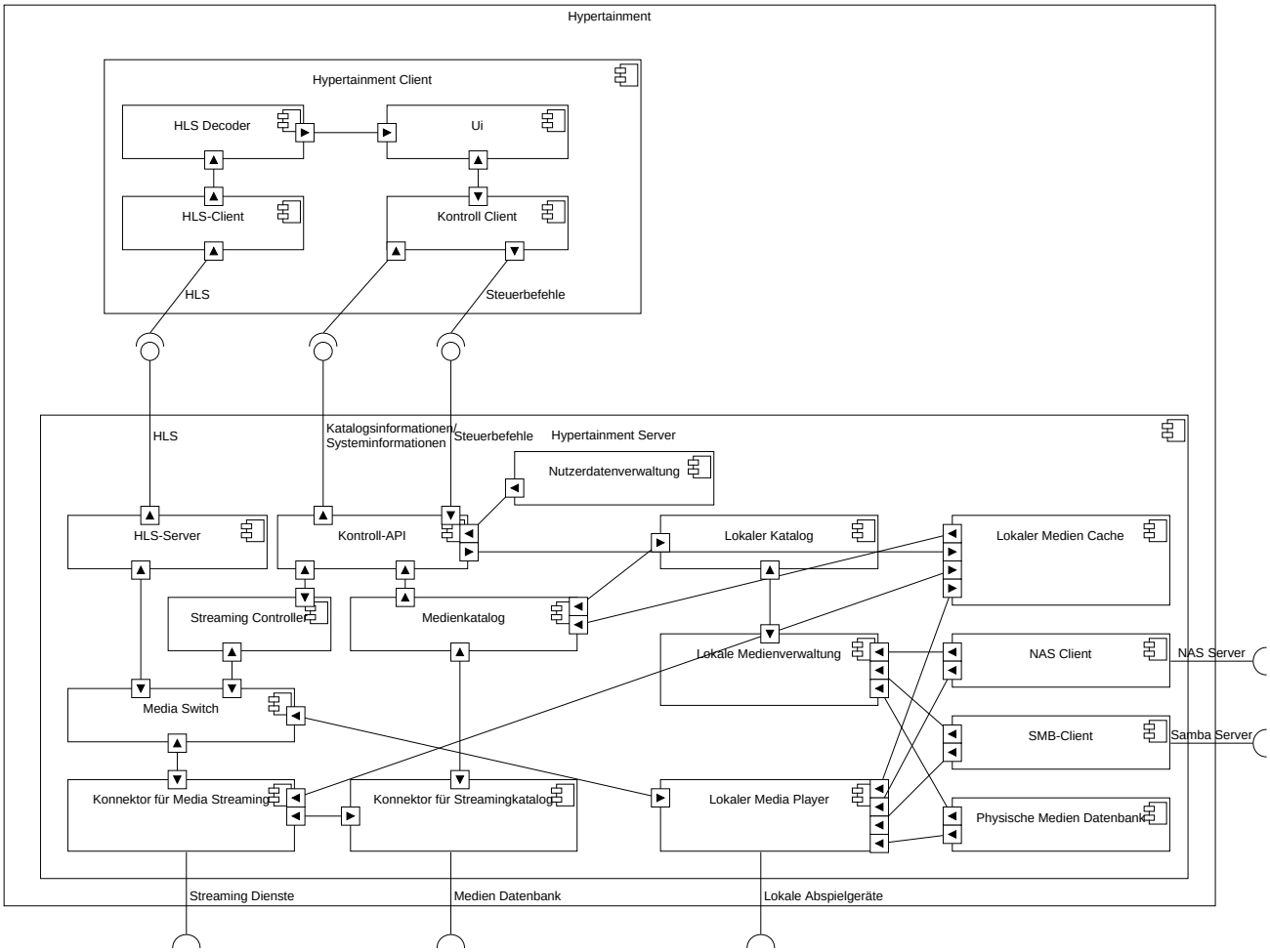

Figure 5. Komponentendiagramm

Streamingdiensten verfügbaren Medien von einer Zentralen Datenbank ab. Der *Lokale Katalog* sammelt Informationen über Medien, die in der Cloud, oder auf privaten Geräten und Servern verfügbar sind. Außerdem hat er Zugriff auf eine Datenbank, in der die Nutzer*in Medien, die diese auf physischen Datenträgern besitzt, eintragen kann.

3.3 Laufzeitsichten

Möchte die Nutzer*in ein Medium wiedergeben, so wählt sie dieses im Medienkatalog aus. Dieser prüft dann, in welcher Quelle das Medium verfügbar ist. Liegt das Medium lokal vor, wird der *Lokale Media Player* aufgefordert, dieses wiederzugeben. Der *Lokale Media Player* ruft das Medium von der Quelle, beispielsweise einem *NAS-Server* ab. Dafür nutzt er die Komponente *NAS-Client*, die ein Medium von einem *NAS-Server* abruft und den Stream an den Lokalen Media Player weiterleitet. Dieser spielt das Medium ab und kodiert den Medienstream für die Wiedergabe durch den Client. Der Medienstream wird über den *Media Switch* an den *HLS-Server* weitergeleitet, der den Medienstream dem Client bereit stellt. Liegt das Medium nicht lokal vor, sondern muss von einem Streamingdienst bezogen werden, so meldet der *Medienkatalog* dies dem *Konnektor für Streamingkatalog*. Dieser prüft bei welchem Streamingdienstleister das Medium verfügbar ist. Und gibt die, für die Wiedergabe, benötigten Informationen, falls verfügbar, an den *Konnektor für Mediastreaming* weiter. Dieser ruft den Stream ab und leitet diesen an den *Media Switch* weiter, der diesen wie zuvor beschrieben an den Client weiter leitet.

In Aktivätsdiagramm 6 ist der Prozess ein Medium abzurufen aus der Sicht der Nutzer*in dargestellt. Diese Sicht ein Medium. Dazu zeigt das System den Medienkatalog. Aus diesem trifft die Nutzer*in eine Auswahl. Wird keine

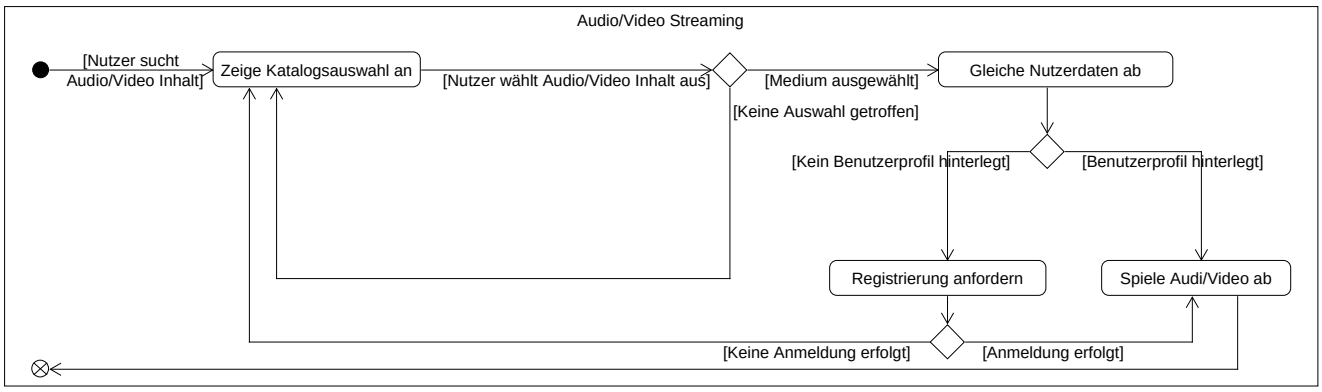

Figure 6. Aktivitätsdiagramm: Videoinhalte von Streamingdienst abrufen

Auswahl getroffen, wird weiter der Katalog gezeigt. Wurde eine Auswahl getroffen, prüft das System auf welchem Streamingdienst das Medium verfügbar ist und ob für diesen Dienst gültige Nutzerdaten hinterlegt sind. Ist dies nicht der Fall wird der Nutzer aufgefordert sich zu registrieren, um das Medium abspielen zu können. War der Nutzer bereits angemeldet, oder hat sich nach der Aufforderung angemeldet, wird das gewählte Medium abgespielt.

4 Fazit

Diese Arbeit enthält die theoretische Vorarbeit für die Entwicklung eines Multimediasystems, das Medien aus unterschiedlichen Quellen abruft und der Nutzer*in gebündelt präsentiert. Dafür wurden die Forschungsfragen aus Tabelle 6 geklärt. Für die Klärung von Frage Forschungsfrage 1, wurde eine Übersicht über benötigte Technologien, sowie die Anforderungen die durch Nutzer, aber auch andere relevante Parteien, wie Streaming Anbieter, aber auch durch den Gesetzgeber an ein solches System, erstellt. Dabei haben die Nutzer und die Streaminganbieter jeweils eigene Use-Cases, aus denen sich die User Stories, zu finden in Tabelle 8, ableiten. Qualitätsmerkmale, zu finden in Tabelle 9 beschreiben Szenarien, die das System lösen können muss. Gesetzte und technische Standards geben Randbedingungen vor, zu finden in Tabelle 10, die bei der Entwicklung eines Systems eingehalten werden müssen. Für die Klärung von Forschungsfrage 2, ebenfalls zu finden in Tabelle 6 wurde basierend auf den erfassten Anforderungen eine Architektur für ein Softwaresystem entwickelt, das diese erfüllen kann. Für den Entwurf dieser wurden in Abschnitt 3.1 Architekturfragen gestellt, beantwortet und dokumentiert. Auf Grundlage dieser Entscheidungen wurde eine Softwarearchitektur entworfen, bei der ein Server die Medien von verschiedenen Quellen bezieht, recodiert und an den Client weiter leitet. Der Client selbst enthält nur die nötige Logik für das Wiedergeben des Medienstreams. Der Server selbst verwaltet eine Liste aller, lokal und online, verfügbaren Medien und präsentiert diese, über den Client, dem Nutzer. Wird ein Medium gewählt, sucht der Server unter welcher Quelle dieses verfügbar ist und ruft es ab. Der Server verwaltet dazu auch die Zugangsdaten zu Streamingdiensten, Clouddiensten, NAS-Servern und anderen Quellen. Diese Zugangsdaten werden in einer verschlüsselten Datenbank, innerhalb des Servers abgelegt. Dies ermöglicht es einem Multimediasystem, dem Nutzer ein möglichst nahtloses, einheitliches Nutzungserlebnis zu präsentieren, unabhänig aus welcher Quelle die wiedergegeben Medien stammen.

Um die Funktionalität und Effektivität der Architektur zu Testen würde im nächsten Schritt ein Prototyp implementiert werden. Anhand diesem kann anschließend geprüft werden, ob das System die geforderten Anforderungen erfüllt und von Nutzer*innen angenommen wird.

References

[1] 2023. Consumer Insights Global Umfrage 2023. https://de.statista.com/prognosen/999834/deutschland-beliebteste-video-on-demand-anbieter

[2] Fariba Alamdari. 1999. Airline in-flight entertainment: the passengers' perspective. *Journal of Air Transport Management* 5, 4 (1999), 203–209. https://doi.org/10.1016/S0969-6997(99)00014-9

[3] Ali Begen, Tankut Akgul, and Mark Baugher. 2011. Watching Video over the Web: Part 1: Streaming Protocols. *IEEE Internet Computing* 15, 2 (2011), 54–63. https://doi.org/10.1109/MIC.2010.155

[4] Ali C. Begen, Abdelhak Bentaleb, Daniel Silhavy, Stefan Pham, Roger Zimmermann, and Will Law. 2021. Road to Salvation: Streaming Clients and Content Delivery Networks Working Together. *IEEE Communications Magazine* 59, 11 (2021), 123–128. https://doi.org/10.1109/MCOM.121.2100137

[5] Abdelhak Bentaleb, May Lim, Mehmet N. Akcay, Ali C. Begen, and Roger Zimmermann. 2021. Common media client data (CMCD). In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (ACM Digital Library)*, Andra Lutu (Ed.). Association for Computing Machinery, New York,NY,United States, 25–33. https://doi.org/10.1145/3458306.3461444

[6] Bundesrepublik Deutschland. 2021. Jugendschutzgesetz: JuSchG.

[7] Stuart K. Card, George G. Robertson, and Jock D. Mackinlay. 1991. The information visualizer, an information workspace. In *Proceedings of the SIGCHI conference on Human factors in computing systems Reaching through technology*, Scott P. Robertson (Ed.). ACM, New York, NY, 181–186. https://doi.org/10.1145/108844.108874

[8] Riccardo Coppola and Maurizio Morisio. 2016. Connected Car: Technologies, Issues, Future Trends. *ACM Comput. Surv.* 49, 3 (2016). https://doi.org/10.1145/2971482

[9] dsgvo-gesetz.de/. 15.01.2024. Datenschutz-Grundverordnung. https://dsgvo-gesetz.de/art-5-dsgvo/

[10] Europäische Union. 2018. Datenschutz-Grundverordnung: DSGVO.

[11] Kaja J. Fietkiewicz. 2020. The Law of Live Streaming: A Systematic Literature Review and Analysis of German Legal Framework. In *Social Computing and Social Media*, Gabriele H. Meiselwitz (Ed.). LNCS Sublibrary: SL3 - Information systems and applications, incl. Internet/web, and HCI, Vol. 12194. Springer, Cham, Switzerland, 227–242. https://doi.org/10.1007/978-3-030-49570-1{_}16

[12] Christopher Frank and Mathilde Deveraux. 2015. A consumer-centric methodology for selecting architectures against multiple objectives and its application to the in-flight catering and entertainment system.

[13] Gabriela Gheorghe, Renato Lo Cigno, and Alberto Montresor. 2011. Security and privacy issues in P2P streaming systems: A survey. *Peer-to-Peer Networking and Applications* 4, 2 (2011), 75–91. https://doi.org/10.1007/s12083-010-0070-6

[14] Martin Glinz, Hans van Loenhound, Stefan Staal, and Stan Bühne. 2024. Handbuch für das CPRE Foundation Level nach dem IREB-Standard: Aus- und Weiterbildung zum Certified Professional for Requirements Engineering (CPRE) Foundation Level. https://www.ireb.org/content/downloads/3-cpre-foundation-level-handbook/cpre_foundationlevel_handbook_de_v1.1.2.pdf

[15] IMDb.com Inc. 04.01.2024. International Movie Database. https://www.imdb.com

[16] ISO Internationale Organisation für Normung. 01.03.2011. Software-Engineering - Qualitätskriterien und Bewertung von Softwareprodukte (SQuaRE) - Qualitätsmodell und Leitlinien. https://www.iso.org/standard/35733.html

[17] G. Kreitz and F. Niemela. 2010. Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*. IEEE, 1–10. https://doi.org/10.1109/P2P.2010.5569963

[18] Hao Liu. 20–. In-Flight Entertainment System: State of the Art and Research Directions. In *Second International Workshop on Semantic Media Adaptation and Personalization (SMAP 2007),London, United Kingdom, 17.12-18.12.2007*. IEEE, [S.l.]n[s.n.], 241–244. https://doi.org/10.1109/SMAP.2007.37

[19] Sober Lourembam, Venkatata Rajesh M, and Meina Singh. 2014. Multimedia Streaming and Related Issues. *International Journal of Computer Science and Information Technology Research* 2 (2014), 185–188. https://www.researchpublish.com/papers/multimedia-streaming-and-related-issues

[20] Marcello Chiaberge. 2011. *New Trends and Developments in Automotive System Engineering*. IntechOpen, [sine loco]. https://directory.doabooks.org/handle/20.500.12854/54622

[21] Peter Mell, Tim Grance, et al. 2011. The NIST definition of cloud computing. (2011).

[22] Klaus Pohl and Chris Rupp. 2021. *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level* (5. auflage ed.). dpunkt.verlag, Heidelberg. https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2913478

[23] Eric Rescorla. 2000. HTTP Over TLS. RFC 2818. https://doi.org/10.17487/RFC2818

[24] Eric Rescorla and Tim Dierks. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. https://doi.org/10.17487/RFC5246

[25] Syed W. Shah and Salil S. Kanhere. 2019. Recent Trends in User Authentication – A Survey. *IEEE Access* 7 (2019), 112505–112519. https://doi.org/10.1109/access.2019.2932400

[26] Shailendra Mishra, Ranjeeta Yadav, and Sachin Yadav. 2010. Article: Comparison between Centralized & Decentralized Overlay Networks for Media Streaming. *International Journal of Computer Applications* 5, 5 (2010), 33–36.

[27] Meisong Wang, Prem Prakash Jayaraman, R. Ranjan, Karan Mitra, Miranda Zhang, Zheng (Eddie) Li, Samee Khan, Mukkaddim Pathan, and Dimitrios Georgeakopoulos. 2015. *An Overview of Cloud Based Content Delivery Networks: Research Dimensions and State-of-the-Art*. Vol. 9070. 131–158. https://doi.org/10.1007/978-3-662-46703-9_6

[28] David Wilfinger, Alexander Meschtscherjakov, Martin Murer, Sebastian Osswald, and Manfred Tscheligi. 2011. Are We There Yet? A Probing Study to Inform Design for the Rear Seat of Family Cars. In *Human-Computer Interaction - INTERACT 2011*, Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler (Eds.). Lecture Notes in Computer Science, Vol. 6947. Springer Nature, Berlin, Heidelberg, 657–674. https://doi.org/10.1007/978-3-642-23771-3{_}48

[29] Stefan Zörner. 2012. *Softwarearchitekturen dokumentieren und kommunizieren: Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten*. Hanser, München. https://doi.org/10.3139/9783446431287