



Abschlussprüfung Sommer 2024

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Zimmer Atrium Multimedial

Applikation zur Auswahl und Anzeige von Videos

Abgabetermin: Rheinau, den 12. Juni 2024

Prüfungsbewerber:

Florian Dietrich
Rißbühn 3
77866 Rheinau



Ausbildungsbetrieb:

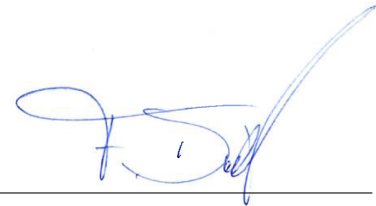
Zimmer GROUP
Am Glockenloch 2
77866 Rheinau

Eigenständigkeitserklärung

Ich versichere hiermit, dass ich meine Projektarbeit mit dem Thema: Zimmer Atrium Multimedial selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Rheinau, 11.06.2024

Ort, Datum



Unterschrift

Gender-Hinweis

Zur besseren Lesbarkeit wird in dieser Dokumentation das generische Maskulinum verwendet. Die in dieser Arbeit verwendeten Personenbezeichnungen beziehen sich – sofern nicht anders kenntlich gemacht – auf alle Geschlechter.

Inhaltsverzeichnis

| | |
|--|-------------|
| Glossar | IV |
| Abkürzungsverzeichnis | V |
| Abbildungsverzeichnis | VI |
| Tabellenverzeichnis | VII |
| Auflistungsverzeichnis | VIII |
| Formelverzeichnis | IX |
| 1 Einleitung | 1 |
| 1.1 Projektumfeld | 1 |
| 1.2 Ist-Analyse | 1 |
| 1.3 Projektziel / Kurzbeschreibung | 2 |
| 1.4 Verwendete Software | 2 |
| 2 Projektplanung | 3 |
| 2.1 Planungsphase | 3 |
| 2.2 Nutzwertanalyse | 4 |
| 2.2.1 "Make or Buy"-Entscheidung | 4 |
| 2.2.2 Projektkosten | 4 |
| 2.2.3 Amortisation | 5 |
| 2.3 Anwendungsfälle | 5 |
| 2.4 User Storys | 5 |
| 2.5 Geplante Vorgehensweise | 6 |
| 2.6 Ressourcenplanung | 7 |
| 3 Umsetzungsphase | 8 |
| 3.1 Zielplattform | 8 |
| 3.2 Einrichten der Clients | 8 |

| | | |
|----------|--|-----------|
| 3.3 | Architekturdesign | 8 |
| 3.4 | Entwurf der Benutzeroberfläche | 9 |
| 3.5 | Programmierung | 10 |
| 3.6 | Maßnahmen zur Qualitätssicherung | 11 |
| 4 | Abschlussphase | 12 |
| 4.1 | Soll-/Ist-Vergleich | 12 |
| 4.2 | Fazit | 12 |
| 4.3 | Reflektion | 13 |
| 4.4 | Ausblick | 13 |
| | Literaturverzeichnis | X |
| | Anhänge | XI |

Glossar

Angular Plattformübergreifendes Framework von Google für Singel-Page Applikationen 2, 9, 10

Materials Angular eigenes UI-Framework 9, 10

NodeJS Serverseitige Laufzeitumgebung für JavaScript 2

Typescript Objektorientiertes Offset für JavaScript 2, 11

Abkürzungsverzeichnis

CSS Cascading Style Sheets 2, 10, 11

GUI Graphical User Interface 9

HDMI High-Definition Multimedia Interface 12

HTML Hypertext Markup Language 2, 11

IDE Integrated Development Environment 2

IoT Internet of Things 3

JSON JavaScript Object Notation 10

NPM Node Package Manager 2

UI User Interface 2

ZAM Zimmer Atrium Multimedial 8

Abbildungsverzeichnis

| | | |
|-----|--|------|
| A.1 | Frontend Controller- und Viewer-App | XI |
| A.2 | Gantt-Diagramm | XII |
| A.3 | Vereinfachte Architektur des Workspace | XIII |
| A.7 | Rückgabe eines Youtube-Objektes in JSON-Format | XVI |

Tabellenverzeichnis

| | | |
|------|------------------------------------|-------|
| 2.1 | Grobe Zeitplanung | 3 |
| 2.3 | Kostenaufstellung | 5 |
| A.8 | Soll-Ist-Vergleich Zeit | XVII |
| A.10 | Detaillierte Zeitplanung | XVIII |
| A.11 | Ressourcen | XIX |

Auflistungsverzeichnis

| | | |
|-----|--|-----|
| 3.1 | HTML des Viewers | 10 |
| A.4 | Abfrage der Playlist | XIV |
| A.5 | YouTube-Service Definition | XV |
| A.6 | Dynamische Generierung der Kacheln | XV |

Formelverzeichnis

| | |
|------------------------------|---|
| 2.2 Kostenrechnung | 4 |
|------------------------------|---|

1. Einleitung

1.1 Projektumfeld

Die Zimmer Group ist ein Fertigungsbetrieb und in verschiedenen Geschäftsfeldern wie Dämpfungssysteme, Kunststofftechnik, Metallverarbeitung und Robotik tätig. Dort ist das Unternehmen Hersteller und Zulieferer für die Automobilindustrie, Automatisierung und Medizintechnik. Die ORG/IT bildet einen kleinen Bereich des Unternehmens und umfasst derzeit 20 Mitarbeiter, bestehend aus Service und Support, Installationen und Softwareverteilung, System- und Netzwerkadministration, Anwendungsentwicklung und Projektmanagement. Ralf Wiedemer, Abteilungsleiter der IT, ist der Auftraggeber für dieses Projekt. Weitere Ansprechpartner in bzw. für dieses Projekt waren:

- Frank Zimmer - Einbinden der Clients in das IoT-Netzwerk und anpassen der Firewall Regeln damit die Client-Geräte den Server erreichen.
- Volker Waldecker - Fragen rund um die Videos, Entscheidungsfindung der Bezugsquelle der Videos und Hilfe rund um Fragen zur Zimmer-Group Corporate Identity.

1.2 Ist-Analyse

Im Atrium der Unternehmenszentrale werden aktuell auf zwei Monitoren unabhängig voneinander je ein Video abgespielt. Die verwendeten Thin-Clients an den Monitoren werden mit einer Zahlungspflichtigen Microsoft Windows 10 Lizenz betrieben und spielen das Promotion Video in einem VLC-Player der im Autostart hinterlegt ist, ab. Inhalte können nur umständlich mit einem USB-Stick direkt an den Clients vor Ort mit Hilfe eines IT-Mitarbeiters ausgetauscht werden. Diese Clients wurden nicht ins Firmennetzwerk aufgenommen und können weder Updates erhalten noch gesteuert werden noch spielen die Videos synchron, da jeder Client unabhängig vom anderen arbeitet. Das Auswählen oder Steuern von Videos über Kunden und Besucher ist nicht möglich. Das Promotion Video informiert nicht, da

es nicht auf firmeneigene Produkte der Zimmer-Group Sparten zugeschnitten ist. Das Marketing, welches die Videos erstellt, hat keine Möglichkeit selbst Inhalte auszutauschen.

1.3 Projektziel / Kurzbeschreibung

Ziel dieser Projektarbeit ist es, den Besuchern eine Möglichkeit zu bieten sich durch Produktvideos über die Firmen der Zimmer-Group zu informieren. Es soll möglich sein aus einer Auswahl an Informationsmaterialien, welche dezentral vom Marketing verwaltet werden, wählen zu können und diese über die beiden Monitore synchron abzuspielen. Videos sind bereits im Microsoft Sharepoint und auf dem YouTube-Kanal der Zimmer-Group hinterlegt.

Ist keine aktive Auswahl getroffen, so soll ein Standard Video in Schleife abgespielt werden. Die Lösung soll möglichst energiesparend betrieben werden, ein Wechsel der vorhandenen Thin-Clients soll angestrebt und Lizenzkosten verringert werden. Kunden und Besucher sollen über eine intuitive Bedienung aus einer Auswahl an Informations- und Produktvideos auswählen können. So soll das Atrium der Firmenzentrale um eine aktive Komponente erweitert werden und über Ausstellungsstücke hinaus Einblicke und Informationen zugänglich machen. In Tabelle A.10 sind die Aufgabenbereiche detailliert aufgegliedert, zeitliche Abweichungen vom Projektantrag sind in Tabelle A.8 einzusehen.

1.4 Verwendete Software

Als Entwicklungsumgebung (IDE) wird das kostenlose Visual Studio Code genutzt. Da die Zimmer-Group fast ausschließlich mit Angular arbeitet, wird mit Angular Framework Version 17 (HTML, CSS, Typescript) sowie dem Angular eigenen UI-Framework "Materials" programmiert, wodurch die Wartung und Erweiterung für andere Entwickler erleichtert wird. Zur Versionskontrolle wird der private GitHub-Account verwendet. Abfragen an die YouTube-API, um den Inhalt aus einer YouTube-Playlist abzurufen werden mit dem Tool Postman getestet. Darüber hinaus wird "Node Package Manager" und "NodeJS" als Laufzeitumgebung und "YouTube-Player" verwendet.

2. Projektplanung

Die Projektumsetzung ist entsprechend der Vorgaben der IHK Südlicher Oberrhein für Fachinformatiker mit dem Fachbereich Anwendungsentwicklung auf 80 Stunden begrenzt. Das Projekt wurde vor Beginn in mehrere Phasen aufgeteilt, die den kompletten Entwicklungsprozess abdecken sollen. Die Arbeiten am Projekt wurden innerhalb der täglichen Arbeitszeit zwischen 8:00 und 16:30 Uhr vom 13.05. bis 24.05.2024 ausgeführt.

Tabelle 2.1 zeigt eine grobe Zeitplanung des Projektes welche in A.10 detaillierter aufgegliedert zu finden ist. Ein Gantt-Diagramm mit der zeitlichen Aufteilung findet sich in A.2.

| Projektphase | Geplante Zeit |
|--------------------------|---------------|
| Analysephase | 5 h |
| Umsetzungsphase | 46 h |
| Testphase auf Zielsystem | 6 h |
| Abschlussphase | 19 h |
| Gesamt | 76 h |

Tabelle 2.1: Grobe Zeitplanung

2.1 Planungsphase

Im Meeting mit dem Auftraggeber, Ralf Wiedemer, werden die Anforderungen als User Storys 2.4 aufgenommen. Des Weiteren wird sich mit einem Vertreter der Marketingabteilung abgestimmt, um über die geplante Lösung zu eruieren, wie die Verwaltung umgesetzt werden kann.

Mit einem Netzwerkadministrator wird Netzwerk- und Sicherheitsrelevantes besprochen wie zum Beispiel das Erreichen des Servers für die Endgeräte, Aufnehmen der Geräte ins IoT-Netzwerk und Setzen von Regeln für die Firewall. Abschließend wird mit der Kosten- und Ablaufplanung begonnen.

2.2 Nutzwertanalyse

Das Atrium der Zimmer-Group ist mit einer Ausstellung von Produkten aller Firmensparten ansprechend gestaltet. Diese bieten aber keine Interaktivität mit Besuchern und lassen nur wenig Raum für Informationen. Der interaktive Zugang zu Informationen über Videos wertet die Ausstellung auf und macht über die Ausstellungstücke hinaus Informationsmaterial zugänglich, welches regelmäßig erweitert werden kann. Besucher können sich gezielter informieren und über die Videos Einblicke bekommen die über eine bloße Ausstellung hinaus gehen.

2.2.1 “Make or Buy“-Entscheidung

Die Kosten sollten möglichst geringgehalten werden, weshalb der Auftraggeber Anbieter fertiger Lösungen aufgrund der monatlichen Nutzungskosten ablehnt. Dadurch fällt die Entscheidung auf eine Eigenentwicklung, von einem Angebotsvergleich und einer Kostengegenüberstellung etwaiger Fremdsoftware-Lösungen wird abgesehen.

2.2.2 Projektkosten

Die Kosten für die Durchführung des Projekts setzen sich sowohl aus Personal-, als auch aus Ressourcenkosten zusammen. Laut Tarifvertrag verdient ein Auszubildender im dritten Lehrjahr pro Monat 1200€ Brutto.

$$\begin{aligned} 8 \text{ h/Tag} \cdot 220 \text{ Tage/Jahr} &= 1760 \text{ h/Jahr} \\ 1200\text{€/Monat} \cdot 13,3 \text{ Monate/Jahr} &= 15960\text{€/Jahr} \\ \frac{15960\text{€/Jahr}}{1760 \text{ h/Jahr}} &\approx 9,06\text{€/h} \end{aligned} \quad (2.2)$$

Es ergibt sich somit ein Stundenlohn von 9,06€. Die Durchführungszeit des Projekts beträgt 76 Stunden. Über die Buchhaltung sind folgende Stundensätze und Pauschalen bekannt:

Für die Nutzung von Ressourcen (Hardware, Software, Bürobedarf inkl. Energie) wird ein pauschaler Stundensatz von 20€ angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundenlohn von 30€ angenommen. Die Kostenrechnung in Tabelle 2.3 listet die Einzelpositionen mit Pauschal- und Stundenkosten für die in Anspruch genommene Zeit des Kunden Ralf Wiedemer, des Mitarbeiter der Marketingabteilung Volker Waldecker sowie dem Netzwerkadministrator Frank

Zimmer in der Anforderungsermittlung, dem Fachgespräch und der Präsentation.

| Vorgang | Personen | Zeit | Kosten | Summe |
|------------------------|----------|-------|---------|------------------|
| Entwickler | 1 | 76 h | 29.06 € | 2208.56 € |
| Anforderungsermittlung | 2 | 0.5 h | 50 € | 50 € |
| Fachgespräch | 3 | 0.5 h | 50 € | 75 € |
| Raum für Präsentation | 1 | 1 h | 50 € | 50 € |
| Gesamt | | | | 2383.56 € |

Tabelle 2.3: Kostenaufstellung

2.2.3 Amortisation

Der Aufwand, Inhalte zu tauschen wird um ein Vielfaches verringert. Das Bearbeiten einer Youtube-Playlist im unternehmenseigenen Youtube-Account oder eines Verzeichnisses im Microsoft Sharepoint genügt, welches problemlos und ohne vorherige Schulung von der Marketingabteilung durchgeführt werden kann. Es wird kein Mitarbeiter der IT-Abteilung benötigt, um Videos vor Ort am Gerät zu tauschen, was zusätzlich Kosten und Ressourcen spart. Zusätzliche Kosten für Microsoft-Lizenzen fallen bei einer Umsetzung mit Linux nicht an.

2.3 Anwendungsfälle

Die Anwendung soll im Atrium der Unternehmenszentrale zur einfachen Steuerung von Informations- und Unterhaltungsmedien genutzt werden. Spätere Anwendung in Showrooms der Unternehmenssparten Kunststofftechnik, Dämpfungssysteme und Systems ist denkbar. Die Ausstellung im Atrium kann mit gezielten Produkt- und Unternehmensvideos bereichert werden und Kunden oder Besucher können sich in Pausen sowie während des Wartens Interviews, Produktvideos und Anwendungsbeispiele ansehen. Erweiterungen der Applikation um die Anwendungsfälle auszuweiten sind schon in Planung und werden in 4.4 genauer beschrieben.

2.4 User Storys

Folgend werden anhand von User Storys die Wünsche des Kunden aufgelistet, wie zum Beispiel die einfache Bedienung und die grundlegende Funktionalität die gewünscht ist. In regelmäßigen Gesprächen werden über Fortschritte informiert, um auf das Feedback des Auftraggebers reagieren zu können.

“Als Benutzer möchte ich eine einfache Navigation die keine Fragen aufwirft was ich zu tun habe. Keine Menüs, Dropdowns oder Untermenüs“

“Die Applikation soll alle Inhalte übersichtlich und ordentlich darstellen. Meine Auswahl muss mit Informationen über den Inhalt der Videos gestützt werden.“

“Es soll ein festgelegtes Video in Schleife spielen wenn vom Nutzer keine Auswahl getroffen wurde.“

“Die YouTube-Videos sollen über unseren Kanal und die dortige Playlist gesteuert werden können.“

“Der Betrieb soll über stromsparende Alternativen und möglichst ohne Microsoft-Lizenz erfolgen“

“Die Applikation soll Steuerelemente und ähnliches unter Berücksichtigung des Zimmer-Group-Corporate-Design darstellen.“

2.5 Geplante Vorgehensweise

In Tabelle A.10 ist eine detaillierte Zeitplanung aufgelistet. Dort ist abzulesen, wie in der Planungs-, Umsetzungs-, Test- und Implementierungsphase sowie der Abschlussphase vorgegangen wird. Nachdem die Anforderungen abgeklärt sind und die Kostenkalkulation abgeschlossen ist, werden in der Umsetzungsphase die Raspberry Pi 4 konfiguriert. Anschließend wird das Projekt angelegt und Schritt für Schritt alle benötigten Komponenten installiert und in die Projekte Controller und Viewer hinzugefügt, die YouTube-API Abfrage getestet und eingebunden, sowie der Service programmiert, über den beide Projekte miteinander kommunizieren können.

Im Projektantrag wurde nicht definiert, aus welcher Quelle die Daten bezogen werden sollen. Während der Analysephase kam es zum Entschluss, die Anbindung mittels YouTube-API umzusetzen, da vorangegangene Projekte bei dem Versuch scheiterten, die Microsoft Graph-API zu nutzen.

Zu Beginn des Projektes wurde sich für einen Agilen Ansatz entschieden, um bei Bedarf das Feedback des Kunden umsetzen zu können. Augenmerk lag dabei auf der einfachen Bedienbarkeit und Steuerung der Inhalte. Die gewünschten Anforderungen an das Programm wurden als User Storys 2.4 aufgenommen. Änderungen und Features wurden nachträglich nicht hinzugefügt. Nachdem die Anforderungen klar kommuniziert und das Frontend der Bedienoberfläche fertig war, wurde die Beurteilung des Auftraggebers Ralf Wiedemer eingeholt, der diese als sehr gut empfand. In der Abschlussphase wird die Dokumentation erstellt und dem Auftraggeber zusammen mit dem Projekt übergeben. Eine Präsentation des Projektes beim Kunden soll die Ergebnisse und das Erfüllen der Anforderungen zeigen, das Projekt zusammenfassen und die Möglichkeit zu Feedback bieten.

2.6 Ressourcenplanung

Für Fragen und Rücksprachen werden Frank Zimmer (Netzwerk und Firewall) und Volker Waldecker (Inhaltsverwaltung) zu Rate gezogen. Weitere Ressourcen wie Betriebssysteme, Software zur Entwicklung und die genutzte Hardware für die Projektumsetzung sind in der Tabelle Ressourcen A.11 aufgelistet.

3. Umsetzungsphase

3.1 Zielplattform

Zielplattform zur Ausführung der Applikation ist ein Microsoft Windows Server 2019 Datacenter auf dem alle betriebseigenen Programme als Webapplikation bereitgestellt werden. Die Raspberry Pi 4 (Viewer-Komponente) werden mit Raspberry OS auf Basis von Debian 12 betrieben

3.2 Einrichten der Clients

In der Umsetzungsphase werden die Raspberry Pi 4 installiert und die Cronjobs eingerichtet, die den Browser im Autostart mit der richtigen Serveradresse und dem Port der Viewer Applikation öffnet. Das System wird mittels Flags im Cronjob so konfiguriert, dass Fehler- oder Updatemeldungen, sowie Systemhinweise im Hintergrund bleiben und der Browser im Kiosk-Modus angezeigt wird und eine störungsfreie Anzeige gewährleistet werden kann.

```
@chromium-browser --noerrdialogs --disable-infobars --kiosk [URL]
```

Die Tablets für den Controller werden mit Android 14, der derzeit aktuellen Version betrieben. Da das Android-Betriebssystem keine Möglichkeit bietet, Applikationen automatisch starten zu lassen wird in der Applikation "MacroDroid" der Browser und die Zieladresse hinterlegt und beim Start des Gerätes aufgerufen. In "MacroDroid" lässt sich ebenfalls der Kioskmodus aktivieren und Meldungen im Hintergrund halten.

3.3 Architekturdesign

In A.3 wird eine vereinfachte Architektur der ZAM-Applikation dargestellt. Es handelt sich dabei um eine verteilte Multiprojektarchitektur mit gemeinsam genutzten Diensten. Im Angular Workspace befinden sich zwei Projekte.

1. Ein Controller-Projekt das die Interaktion des Benutzers verarbeitet (Auswählen eines Videos).
2. Ein Viewer-Projekt, das für die Anzeige des gewählten Videos verantwortlich ist.

Als gemeinsam genutzte Dienste kommen der Node-Server, sowie ein WebSocket-Service zur Echtzeitkommunikation zwischen beiden Projekten zum Einsatz. Jedes Projekt verfügt über einen eigenen Service um WebSocket-Interaktionen zu verarbeiten. Es existiert ein Dienst außerhalb der Projekte im Workspace, der die WebSocket-Kommunikation ermöglicht, sowie einen Node-Server, der als zentraler Hub für die WebSocket-Kommunikation zwischen dem Controller-Projekt und dem Viewer-Projekt fungiert. Der Vorteil des WebSockets ist, dass beliebig viele Clients den Kanal abonnieren können. Sendet der Controller eine Änderung durch Auswählen eines neuen Videos, können alle Clients die diesen Kanal abonniert haben darauf reagieren. So lässt sich das gleichzeitige Abspielen auf verschiedenen Geräten umsetzen und die Anzahl der angebotenen Clients beliebig skalieren.

3.4 Entwurf der Benutzeroberfläche

Beim Gespräch mit dem Kunden ist der Wunsch nach einer einfachen Bedienung geäußert worden. Mit Blick auf die Zielgruppe wurde schnell klar, dass unnötige Bedienelemente oder Buttons zur Navigation die intuitive Benutzung beeinträchtigen. Eine weitere Anforderung an die GUI ist, neben den dynamischen Laden der Inhalte die Einfachheit, sich zurecht zu finden.

Der Controller zur Interaktion des Benutzers wird durch das Angular Material `<mat-card>`-Element als klickbare Kacheln, die eine Aussicht auf das Thema des Videos geben und leicht mit einer Touch Geste aufzurufen sind, umgesetzt. In der HTML-Datei des Controllers (siehe A.6) wird über eine `for`-Schleife über ein Array mit Objekten aus der YouTube-Playlist iteriert und für jedes Element je eine Kachel erzeugt. Die über die YouTube-API abgefragten Inhalte liefern Thumbnails mit, die den Kacheln als Icons dienen. In Abbildung A.1a sowie A.1b sind Screenshots beider Komponenten zu sehen. In der Controller-Komponente wird jede Kachel mit einem blauen Rahmen umrandet, die Gruppe aller geladenen Inhalte hat ebenfalls einen dünnen blauen Rahmen um den Fokus auf die Elemente zu lenken. Dieser Rahmen erweitert sich dynamisch nach unten, je nach Menge der geladenen Videos und Größe des zur Verfügung stehende Anzeigemediums.

Durch `filter: brightness(2) blur(15px);` im CSS wurde das Zimmer-Group Logo im Hintergrund in leichter Unschärfe integriert. Der Hintergrund des Controllers wurde dunkel gestaltet, um bei schlechten Lichtverhältnissen die Augen zu schonen und ihm eine edlere Ästhetik zu verleihen. Die Viewer-Komponente ist frei von Bedienelementen oder sonstige Hinweisen, um Videos in Vollbild auf dem gesamten Monitor abspielen zu können.

3.5 Programmierung

Nach dem Einrichten und Strukturieren von Entwicklungsumgebung und Workspace wird die Controller- Komponente programmiert. Hier wird sich Angular Materials [2], der Angular eigenen Bibliothek für Gestaltungselemente bedient. In der Controller-Komponente ist wichtig für jedes Element in der Playlist dynamisch ein Element (Kachel) im Frontend zu generieren. Ein YouTube-Service in der Controller-Komponente der den YouTube API-Schlüssel und die Playlist-ID des Firmeneigenen Kanals enthält, ruft die YouTube-Playlist ab und speichert die Objekte in einem Array. Ein solches Objekt ist in A.7 als JSON abgebildet, dort enthalten sind alle Informationen wie Position und Video-ID. Dieses Array wird anschließend in einer Schleife im Frontend durchlaufen um eine Kachel pro Element zu generieren. Die Methode `sendToView(url: string, position: any)` leitet die Video-ID und die Positionsnummer des Videos in der Playlist dann über Websocket in den Kanal `youtube`, der von der Viewer-Komponente abonniert wird. Der Aufbau der Kacheln, des YouTube-Service und die Abfrage der Playlist sind A.6, A.5 und A.4 in zu sehen

Das Styling der Controller-Komponente wird nach gelungener Abfrage der YouTube-Playlist vorgenommen, um Thumbnails, ID und Position aus den Objekten auslesen zu können. Da die Generierung der Kacheln dynamisch ist, muss vor allem darauf geachtet werden, dass unabhängig der Menge der geladenen Elemente die Anzeige ansprechend, sauber und überschaubar bleibt. Dies wurde mittels CSS-Styling unter Verwendung von `flexbox` realisiert. Daraufhin wird die Viewer-Komponente programmiert und gestaltet. Ziel war es, die Wiedergabe des Videos bei Nutzung der gesamten Bildschirmdiagonale zu gewährleisten.

Diese lässt sich mit `window.screen.width` und `window.screen.height` auslesen und dem YouTube eigenen Player-Tag mitgeben. Zusätzlich werden Tags genutzt, die das Autoplay des geladenen Videos triggern und so ein sofortiges abspielen erfolgt, sowie die Kontrollelemente wie Pause, Play und den Wiedergabefortschritt ausblenden.

```
1 <youtube-player
2   [videoId]="videoID"
3   [width]="screenWidth"
4   [height]="screenHeight"
5   [playerVars]="{'controls': 0, 'autoplay': 1}">
6 </youtube-player>
```

Auflistung 3.1: HTML des Viewers

3.6 Maßnahmen zur Qualitätssicherung

In Angular kann während der Entwicklung der einzelnen Projekte je ein eigener Entwicklungsserver gestartet werden. Dieser baut und hostet die Anwendung, überwacht den Quellcode auf Änderungen und führt bei jeder Speicherung einen automatischen Neu-Build durch, sodass die Änderungen sofort im Browser sichtbar werden. Auf diese Weise wird bei jeder Änderung die Funktionalität überprüft und Fehler mittels Konsolenausgaben im Browser lokalisiert und behoben. In diesem Projekt wurden sowohl HTML, CSS als auch Typescript Code auf diese Weise entwickelt. Darüber hinaus werden einzelne Komponente wie dem YouTube-Player, die YouTube-API Abfrage und das Übermitteln der Videodaten getestet, um schrittweise Fehler ausschließen zu können.

4. Abschlussphase

4.1 Soll-/Ist-Vergleich

Wie in Tabelle A.8 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden. Die Abweichungen in der Umsetzungsphase sind mit der Zeitersparnis bei der Implementierung der YouTube-API einerseits und dem Mehraufwand beim Integrieren des WebSockets andererseits zu erklären. Da die YouTube-API gut dokumentiert ist und dadurch die Abfrage der Playlist schnell umgesetzt werden konnte, wurden von der veranschlagten Zeit 4 Stunden eingespart. Bei der Programmierung des WebSockets sind wiederum 7 Stunden mehr Zeit aufgewendet worden. Grund dafür war die fehlende Erfahrung mit WebSockets und ihrer Funktion sowie Probleme bei der Implementierung, die sich daraus ergaben. Die Videoinformationen von der Controller-Komponente zur Viewer-Komponente zu übermitteln war der zeitaufwendigste Posten der Umsetzungsphase. Am Ende mussten insgesamt 79 Stunden und nicht wie geplant 76 Stunden für das Projekt aufgewendet werden. Die Forderung, ein Standardvideo in Schleife zu spielen konnte aus zeitlichen Gründen nicht umgesetzt werden. Über die Integration eines zweiten WebSockets sollte das Video mit der Positionsnummer 0 ausgelesen und mit einer noch nicht implementierten Logik immer dann abgespielt werden, wenn gerade kein anderes Video spielt.

4.2 Fazit

In diesem Projekt wurden bis auf das Abspielen eines Standardvideos alle Anforderungen und Wünsche des Auftraggebers umgesetzt. Das Managen der Inhalte ist dezentral vom Marketing über eine YouTube-Playlist möglich. Die günstigen Einplatinencomputer lassen sich ohne Microsoft Lizenz betreiben und sind Stromsparend, sowie an jeden Monitor mit HDMI-Anschluss anzuschließen. Die Bedienung ist durch die Lösung mit klickbaren Kacheln einfach und intuitiv und wirft keine Fragen auf, da sie gänzlich ohne zusätzliche Buttons oder Dropdown-Menüs auskommt und der Benutzer dadurch stets die Übersicht behält. Die Ka-

cheln geben auf einen Blick Aufschluss über den Inhalt des jeweiligen Videos und der Klick auf eine Kachel genügt, um die Videos synchron an beiden Bildschirmen abzuspielen.

4.3 Reflektion

Das Arbeiten mit unbekannten Technologien wie WebSockets haben während der Umsetzung für Verzug gesorgt. Bei der Einschätzung des zeitlichen Aufwandes muss künftig mehr Zeit für die Umsetzung eingeplant werden, um den Projektrahmen nicht zu sprengen. Puffer für das Gesamtprojekt oder einzelner Pakete kann diese Zeit abfangen und die Einhaltung von Terminen erleichtern.

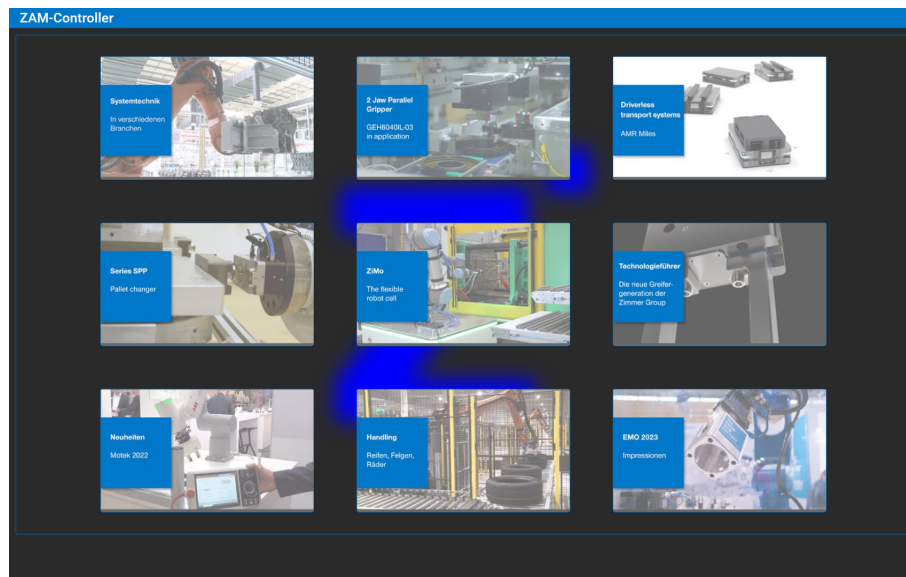
4.4 Ausblick

Die Applikation lässt sich durch die Organisation als Workspace modular erweitern und bietet dadurch viele Möglichkeiten. Beispielsweise könnten Außendienstmitarbeiter die Applikation nutzen, um beim Kunden nicht öffentliche Präsentationsvideos aufzurufen oder sogar offline zu speichern. Durch eine Erweiterung des Programms könnte ein Login für Benutzer erstellt werden, indem jeder Außendienstmitarbeiter seine Favoriten im eigenen Account abspeichert, um sie beim Präsentieren schnell zu finden. Auch das Nutzen der Applikation auf Messen um Videowände zu steuern wäre ein denkbarer Anwendungsfall, ebenso wie eine Erweiterung, um Slideshows und Präsentationen darüber abzuspielen. Auch eine Anbindung an den Sharepoint, um an Inhalte zu gelangen, wäre denkbar. Der Workspace könnte letzten Endes als Grundlage für andere Applikationen dienen, die die Dienste nutzen möchten oder selbst welche hinzufügen.

Literaturverzeichnis

- [1] *Angular Dokumentation*. <https://angular.dev/overview>.
- [2] *Angular Materials Dokumentation*. <https://material.angular.io/components/categories>.
- [3] David Flanagan. *Javascript: The Definitive Guide: Master the World's Most-Used Programming Language*. O'Reilly, 7. Auflage.
- [4] *Google Youtube-API Dokumentation*. <https://developers.google.com/youtube/v3/getting-started?hl=de>.
- [5] *How to integrate YouTube API with Angular von Kheronn K. Machado*. <https://kheronn-machado.medium.com/youtube-angular-en-2ed98f07e0f9>.
- [6] *Real-Time WebSocket von Amartya Deshmukh (nitorinfotech)*. <https://www.nitorinfotech.com/blog/angular-and-custom-websocket-integration-build-a-real-time-to-do-list-app/>.
- [7] Manfred Steyer. *Angular, Das Praxisbuch zu Grundlagen und Best Practices*. O'Reilly, 3. Auflage.
- [8] *YouTube Iframe-API Dokumentation*. https://developers.google.com/youtube/iframe_api_reference?hl=de.

Anhänge



(a) Bedienoberfläche der Controller-App



(b) Vollbildanzeige Viewer-App

Abbildung A.1: Frontend Controller- und Viewer-App

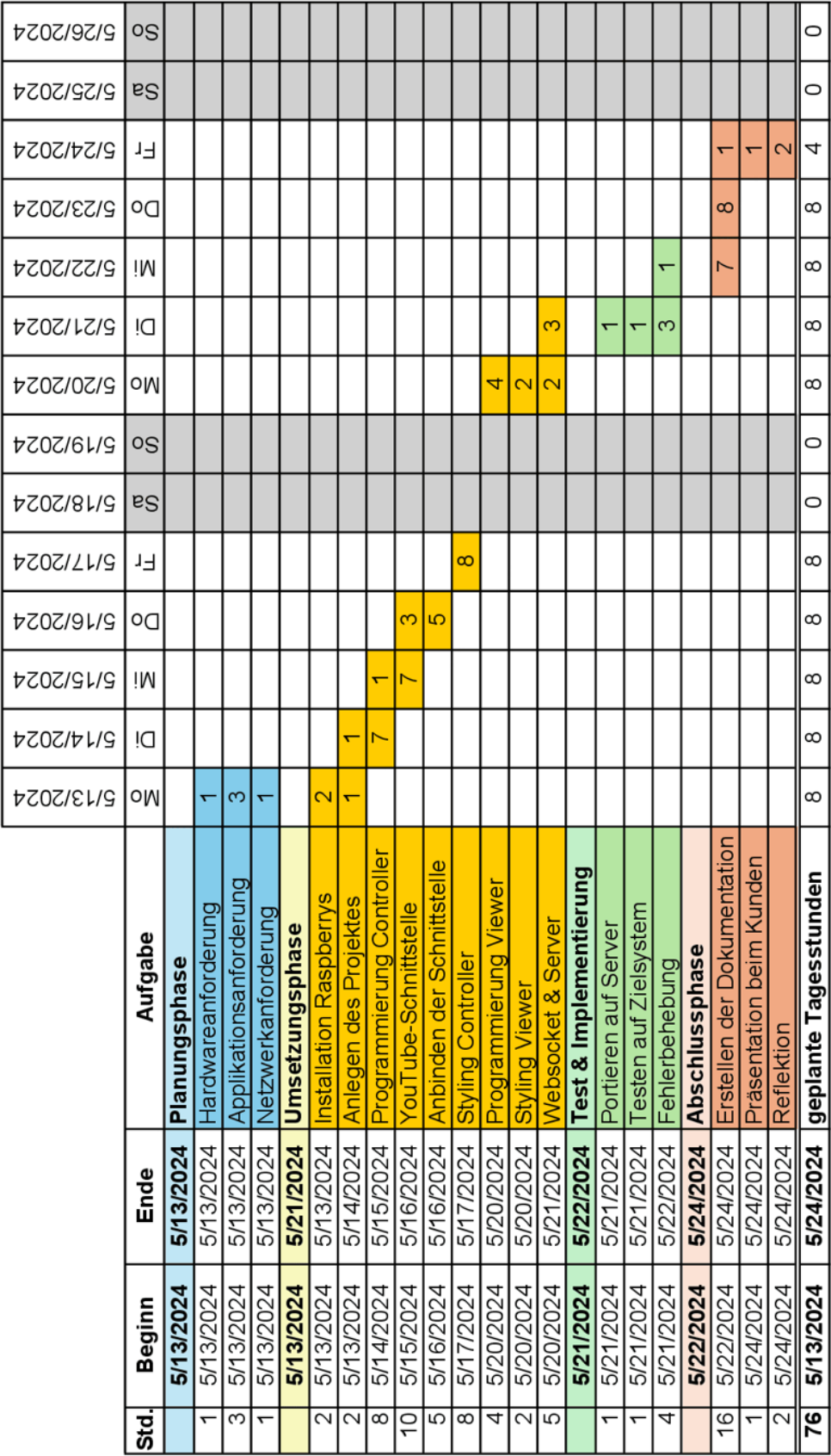


Abbildung A.2: Gantt-Diagramm

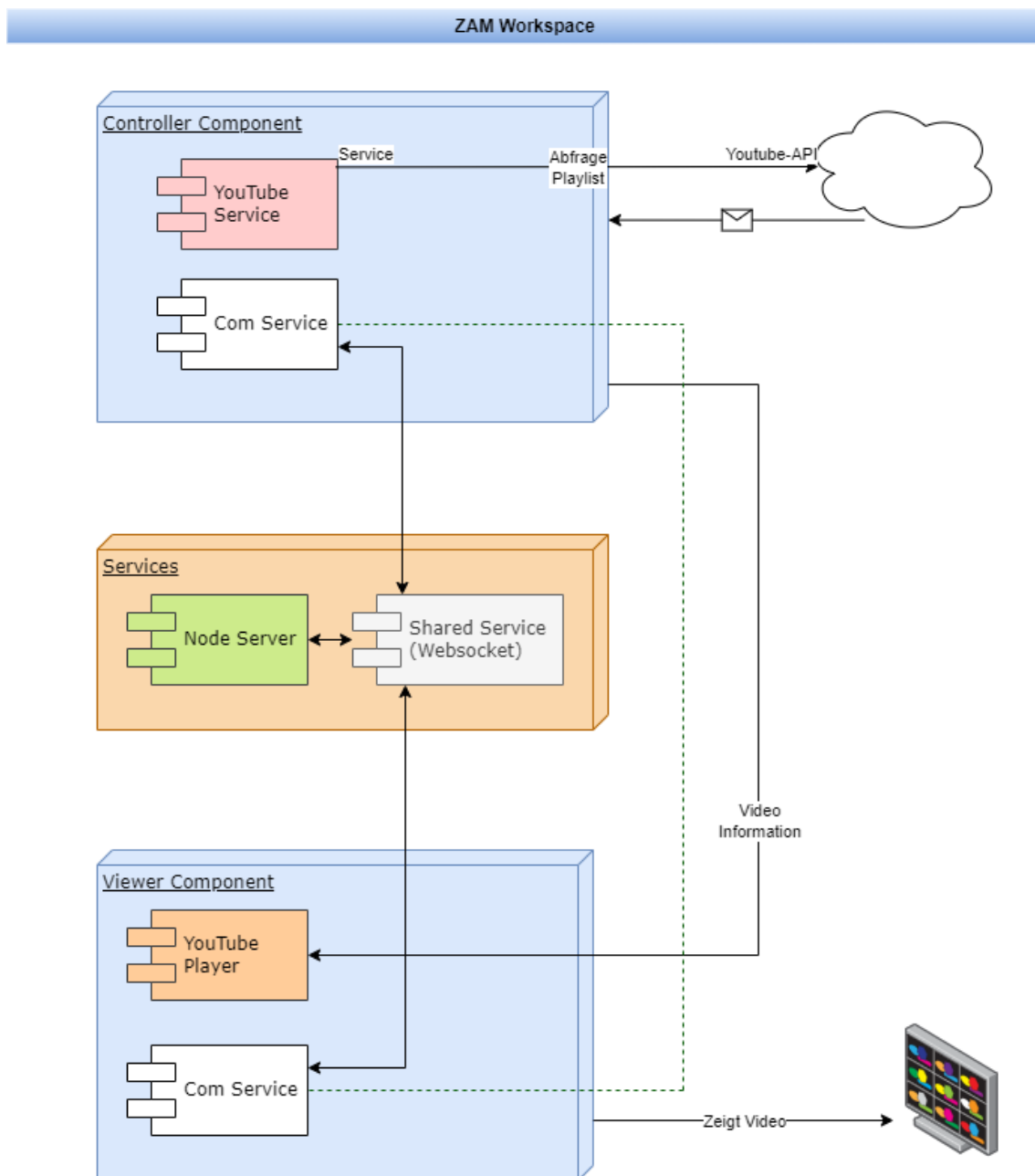


Abbildung A.3: Vereinfachte Architektur des Workspace

```
1 ngOnInit() {
2     this.spinner.show()
3     setTimeout(()=>
4     {
5         this.spinner.hide()
6     },3000)
7     this.youtubeService
8     .getVideosForPlaylist()
9     .pipe(takeUntil(this.unsubscribe$))
10    .forEach((list: any)=> {
11        for (let element of list["items"]) {
12            this.videos.push(element)
13
14            if(element.snippet.position == 0){
15                this.sharedService.setData(element.snippet.resourceId.
16                    videoId)
17                this.comService.send("youtube", {
18                    "url" : element.snippet.resourceId.videoId
19                })
20
21                this.comService.subscribe("config", (msg: any) => {
22                    if(msg.cmd == "get_config") {
23                        this.comService.send("config", {
24                            url: element.snippet.resourceId.videoId
25                        });
26                    }
27                })
28            }
29        });
30    }
```

Auflistung A.4: Abfrage der Playlist

```

1 import { Injectable, EventEmitter } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { map, takeUntil } from 'rxjs/operators';
4 import { Observable, Subject } from 'rxjs';
5 import { NgxSpinnerService } from 'ngx-spinner';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class YoutubeService {
11
12   apiKey : string = 'AIzaSyDvpVFzT6jJulm-w6aw7riMEK8CDy5D4p4';
13   playlistId: string = 'PLNf7WrW3VV-yW71-xs-QVc0bvZh32_qVC';
14   maxResults: number = 50
15
16   constructor(public http: HttpClient) { }
17
18   getVideosForPlaylist(): Observable<Object> {
19     let url = 'https://www.googleapis.com/youtube/v3/playlistItems?
20               key='
21               + this.apiKey
22               + '&playlistId='
23               + this.playlistId
24               + '&order=date&part=snippet &type=video,id&maxResults='
25               + this.maxResults
26     return this.http.get(url)
27       .pipe(map((res) => {
28         return res;
29       })))
30 }

```

Auflistung A.5: YouTube-Service Definition

```

1 <div class="grid-container">
2   <div class="background-image"></div>
3   <div class="card-container" *ngFor="let video of videos" style=
4     "margin:2em">
5     <mat-card matRipple class="clickable" (click)="sendURL(
6       video.snippet.resourceId.videoId,video.snippet.position)
7     ">
8     <a>
9       <img [src]="video.snippet.thumbnails.medium.url" class="
10        card-img-top">
11     </a>
12   </mat-card>
13 </div>
14 </div>

```

Auflistung A.6: Dynamische Generierung der Kacheln

```
▼ Object ⓘ
  etag: "cx5yh2upnCKOIoy_8Rk2yE7nH9k"
  id: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
  kind: "youtube#playlistItem"
  ▼ snippet:
    channelId: "XXXXXXXXXXXXXXXXXXXX"
    channelTitle: "Zimmer Group"
    description: []
    playlistId: "PLNf7WrW3VV-yW71-xs-QVc0bvZh32_qVC"
    position: 0
    publishedAt: "2024-02-01T13:16:10Z"
  ▼ resourceId:
    kind: "youtube#video"
    videoId: "ZxF0b2JlYQ0"
    ▶ [[Prototype]]: Object
  ▼ thumbnails:
    ▶ default: {url: 'https://i.ytimg.com/vi/ZxF0b2JlYQ0/default.jpg', width: 120, height: 90}
    ▶ high: {url: 'https://i.ytimg.com/vi/ZxF0b2JlYQ0/hqdefault.jpg', width: 480, height: 360}
    ▶ medium: {url: 'https://i.ytimg.com/vi/ZxF0b2JlYQ0/mqdefault.jpg', width: 320, height: 180}
    ▶ standard: {url: 'https://i.ytimg.com/vi/ZxF0b2JlYQ0/sddefault.jpg', width: 640, height: 480}
    ▶ [[Prototype]]: Object
    title: "Systemtechnik - Anwendung in verschiedenen Branchen"
    videoOwnerChannelId: "XXXXXXXXXXXXXXXXXXXX"
    videoOwnerChannelTitle: "Zimmer Group"
    ▶ [[Prototype]]: Object
  ▶ [[Prototype]]: Object
```

Abbildung A.7: Rückgabe eines Youtube-Objektes in JSON-Format

| Phase | Geplant | Tatsächlich | Differenz |
|--|---------|-------------|-----------|
| Planungsphase: | 5 h | 5 h | - |
| Analyse des Ist-/Soll-Zustands | 1 h | 1 h | - |
| Hardwareanforderungen | 1 h | 1 h | - |
| Applikationsanforderungen | 2 h | 2 h | - |
| Netzwerkanforderungen | 1 h | 1 h | - |
| Umsetzungsphase: | 46 h | 49 h | +3 h |
| Installation und Einrichtung der Raspber- rys | 2 h | 2 h | - |
| Programmierung: | | | |
| Anlegen des Angular Workspace | 1 h | 1 h | - |
| Installation aller Komponenten | 1 h | 1 h | - |
| Implementierung der Youtube-API | 10 h | 6 h | -4 h |
| Programmieren des Controllers | 8 h | 8 h | - |
| Einbinden der Youtube-API in den Con- troller | 5 h | 5 h | - |
| Styling des Controllers unter Verwen- dung des Corporate Design | 8 h | 8 h | - |
| Programmieren des Viewers | 4 h | 4 h | - |
| Styling des Viewers | 2 h | 2 h | - |
| Programmierung und Integration des Websocket | 5 h | 12 h | +7 h |
| Testphase auf Zielsystem: | 6 h | 6 h | - |
| Portieren der Applikation in die Server- umgebung | 1 h | 1 h | - |
| Testen der Applikation auf den Tablets und Bildschirmen | 1 h | 1 h | - |
| Fehlerbehebung und Abschluss | 4 h | 4 h | - |
| Abschlussphase: | 19 h | 19 h | - |
| Erstellen der Projektdokumentation | 16 h | 16 h | - |
| Präsentation des Projektes beim Auf- traggeber | 1 h | 1 h | - |
| Reflektion | 2 h | 2 h | - |
| Gesamt | 76 h | 79 h | +3 h |

Tabelle A.8: Soll-Ist-Vergleich Zeit

| | |
|--|-------------|
| Planungsphase | 5 h |
| 1. Analyse des Ist/Soll-Zustands | 1 h |
| 2. Hardwareanforderungen Ist-Analyse | 1 h |
| 3. Applikationsanforderungen | 2 h |
| 3.1 Ermitteln der gewünschten Anforderungen | 1 h |
| 3.2 Kostenkalkulation | 1 h |
| 4. Netzwerkanforderungen | 1 h |
| Umsetzungsphase | 46 h |
| 1. Installation und Konfiguration der Raspberrys | 2 h |
| 1.1 Image aufspielen, Clientname, Freischaltung für Netzwerk | 1 h |
| 1.2 Erstellen des Cronjob, Einbauen in das Gehäuse | 1 h |
| 2. Programmierung | 44 h |
| 2.1 Anlegen des Projektworkspace (zwei Projekte, View-er und Controller) | 1 h |
| 2.2 Installation benötigter Komponenten | 1 h |
| 2.3 Implementieren der Youtube-Schnittstelle | 10 h |
| 2.4 Programmieren des Controller | 8 h |
| 2.5 Anbindung des Controller an die Youtube-Schnittstelle | 5 h |
| 2.6 Styling des Controller | 8 h |
| 2.7 Programmieren des Viewer zur Wiedergabe der Videos | 4 h |
| 2.8 Styling des Viewer | 2 h |
| 2.9 Programmieren des Service zur Kommunikation zwischen beiden Apps | 5 h |
| Test- und Implementierungsphase | 6 h |
| 1. Portieren der entwickelten App auf den Server | 1 h |
| 2. Testen der App auf den Zielsystemen | 1 h |
| 3. Fehlerbehebung seitens Netzwerk, Hardware und App | 4 h |
| Abschlussphase | 19 h |
| 1. Erstellen der Projektdokumentation | 16 h |
| 2. Präsentation des Projektes beim Kunden | 1 h |
| 3. Reflektion | 2 h |
| Gesamt | 76 h |

Tabelle A.10: Detaillierte Zeitplanung

| Ressourcen | Anzahl | Beschreibung |
|-------------------------|--------|-----------------------------|
| Entwickler | | |
| Fujitsu Lifebook | 1 | Arbeitsplatz |
| Iiyama 24 Zoll Monitor | 2 | Arbeitsplatz |
| Lenovo Dockingstation | 1 | Arbeitsplatz |
| Cherry Tastatur | 1 | Arbeitsplatz |
| Logitech Maus | 1 | Arbeitsplatz |
| Hardware | | |
| Raspberry Pi 5 Kit | 2 | Monitor |
| Samsung Tablets Tab 9 | 2 | Bedienung |
| Samsung Monitor | 2 | Anzeige |
| Software | | |
| Raspberry OS (Bookworm) | 2 | Betriebssystem Raspberry Pi |
| Windows 11 | 1 | Betriebssystem Entwickler |
| Visual Studio Code | 1 | Entwicklungsumgebung |
| Postman | 1 | Testen von API Abfragen |
| Rufus 4.3 | 1 | Imaging Tool |

Tabelle A.11: Ressourcen