

Prüfungsteil A

Prüfling (private Anschrift):

Ausbildungsbetrieb:

Bestätigung über durchgeführte Projektarbeit

diese Bestätigung ist mit der Projektdokumentation einzureichen

Ausbildungsberuf (bitte unbedingt angeben):

Projektbezeichnung:

Projektbeginn: _____ Projektfertigstellung: _____ Zeitaufwand in Std.: _____

Bestätigung der Ausbildungsfirma:

Wir bestätigen, dass der/die Auszubildende das oben bezeichnete Projekt einschließlich der Dokumentation im Zeitraum

vom: _____ bis: _____ selbstständig ausgeführt hat.

Projektverantwortliche(r) in der Firma:

Vorname	Name	Telefon	Unterschrift
---------	------	---------	--------------

Ausbildungsverantwortliche(r) in der Firma:

Vorname	Name	Telefon	Unterschrift
---------	------	---------	--------------

Eidesstattliche Erklärung:

Ich versichere, dass ich das Projekt und die dazugehörige Dokumentation selbstständig erstellt habe.

Ort und Datum: _____ Unterschrift des Prüflings: _____



Abschlussprüfung Sommer 2024

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Digitale Gästebegrüßung und Infotafel

ein Tool zur automatisierten Gästebegrüßung

Abgabetermin: Braunschweig, den 17.04.2024

Prüfungsbewerber:

Jeffrey Aspinall
Feldstraße 29
38640 Goslar



Ausbildungsbetrieb:

CSTx Enterprise Solutions GmbH
Volkmaroder Straße 7
38104 Braunschweig

Inhaltsverzeichnis

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
Listings	V
Abkürzungsverzeichnis	VI
1 Einleitung	1
1.1 Projektumfeld	1
1.2 Projektziel	1
1.3 Projektbegründung	1
1.4 Projektschnittstellen	1
1.5 Projektabgrenzung	2
2 Projektplanung	3
2.1 Projektphasen	3
2.2 Abweichungen vom Projektantrag	3
2.3 Ressourcenplanung	3
2.4 Entwicklungsprozess	4
3 Analysephase	5
3.1 Ist-Analyse	5
3.2 Wirtschaftlichkeitsanalyse	5
3.2.1 „Make or Buy“-Entscheidung	5
3.2.2 Projektkosten	5
3.2.3 Amortisationsdauer	6
3.3 Nicht monetäre Vorteile	7
3.4 Anwendungsfälle	7
3.5 Qualitätsanforderungen	7
3.6 Lastenheft/Fachkonzept	7
4 Vorbereitungsphase	9
4.1 Zielplattform	9
4.2 Architekturdesign	9
4.3 Entwurf der Benutzeroberfläche	9
4.4 Geschäftslogik	10
4.5 Maßnahmen zur Qualitätssicherung	11
5 Implementierungsphase	12
5.1 Implementierung der Benutzeroberfläche	12

Inhaltsverzeichnis

5.2	Implementierung der Geschäftslogik	12
6	Abnahmephase	14
7	Dokumentation	14
8	Fazit	15
8.1	Soll-/Ist-Vergleich	15
8.2	Lessons Learned	15
8.3	Ausblick	16
A	Anhang	i
A.1	Detaillierte Zeitplanung	i
A.2	Lastenheft (Auszug)	ii
A.3	Verwendete Ressourcen	iii
A.4	Use Case-Diagramm	iv
A.5	Oberflächenentwürfe	v
A.6	Produktfotos der Anwendung	vii
A.7	Main-Komponente Logik-Loop	xi
A.8	Main-Komponente IPC	xiii
A.9	Projektstruktur	xv

Abbildungsverzeichnis

Abbildungsverzeichnis

1	Use Case-Diagramm	iv
2	Wireframe der Standby-Ansicht	v
3	Wireframe der Fullscreen-Ansicht	v
4	Wireframe der Ansicht mit einem Termin	vi
5	Wireframe der Ansicht mit mehreren Terminen	vi
6	Ansicht im Standby	vii
7	Vollbildansicht vom Live-Video	viii
8	Ansicht mit einem Termin	ix
9	Ansicht mit mehreren Terminen	x
10	Projektstruktur	xv

Tabellenverzeichnis

1	Zeitplanung	3
2	Kostenaufstellung	6
3	Soll-/Ist-Vergleich	15

Listings

1	Main-Komponente Logik-Loop	xi
2	Main-Komponente IPC	xiii

Abkürzungsverzeichnis

AD	Active Directory
AG	Aktiengesellschaft
API	Application Programming Interface
CSS	Cascading Style Sheets
CSTx ES	CSTx Enterprise Solutions
CSTx SE	CSTx Software Engineering
CP	Container-Presentational
GL	GitLab
GmbH	Gesellschaft mit beschränkter Haftung
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
IHK	Industrie- und Handelskammer
IPC	Inter-Process Communication
IV	Informationsverteilung
JS	JavaScript
JSX	JavaScript XML
MS	Microsoft
MSAL	Microsoft Authentication Library
SV	Source Versioning
TS	TypeScript
TSX	TypeScript XML
UI	User Interface
UML	Unified Modeling Language
VS	Visual Studio

1 Einleitung

1 Einleitung

1.1 Projektumfeld

Seit 2005 gibt es CSTx Software Engineering (CSTx SE) Gesellschaft mit beschränkter Haftung (GmbH) und im Jahr 2008 wurde CSTx SE zum Umsetzungspartner für Volkswagen Aktiengesellschaft (AG). Durch den Großkunden konnte sich die CSTx SE in der Automobilbranche einen festen Platz sichern. Im Jahr 2018 gab es eine Umstrukturierung der Holding und die Gründung der drei Tochterunternehmen kam zustande. Insgesamt ist CSTx SE ein mittelständiger Betrieb mit einer Belegschaft von knapp 100 Mitarbeitern. Die Ausbildung und die Projektarbeit findet in der Tochterfirma CSTx Enterprise Solutions (CSTx ES) GmbH statt. Das Ergebnis des Projekts muss dem Stakeholder präsentiert werden, der das Projekt initiiert hat und Anwender sein wird, der die Anwendung vorrangig nutzt.

1.2 Projektziel

Es soll eine innovative Lösung für die Gästebegrüßung und den Informationsfluss entwickelt werden. Dies umfasst die Darstellung einer auf den Kunden angepasste Info-Karte, die Integration von Live-Video-Streams, einem laufenden Info-Text und einer Uhrzeitangabe. Durch diese Elemente sollen Gäste effektiv und ansprechend begrüßt werden und gleichzeitig wichtige Informationen erhalten. Das Hauptziel besteht darin, ein visuell ansprechendes und informatives Tool zu schaffen, das die Aufmerksamkeit der Besucher auf sich zieht und einen positiven ersten Eindruck vermittelt.

1.3 Projektbegründung

Es besteht eine Notwendigkeit einer zeitgemäßen und effizienten Gästebegrüßung sowie Informationsbereitstellung. Das Unternehmen strebt nach einem Produkt, das Prozesse für den Informationsaustausch vereinfacht und beschleunigt. Zudem soll es äußerst flexibel anpassbar sein und sowohl für Kunden als auch Mitarbeiter einen Mehrwert bieten. Durch die Digitalisierung der Begrüßungs- und Informationsprozesse soll ein innovatives und professionelles Ambiente geschaffen werden, das die Attraktivität der Einrichtung steigert und die Zufriedenheit der Besucher erhöht.

1.4 Projektschnittstellen

Die Anwendung interagiert mit Microsoft (MS) Azure Active Directory (AD) für die Authentifizierung und den Zugriff auf Kalendereinträge. Die Genehmigung des Projekts sowie die Bereitstellung von Ressourcen erfolgt durch den Stakeholder und Entscheidungsträger, wie bspw. die Geschäftsführung. Jeder Mitarbeiter, der über einen Azure-Account der Organisation verfügt, kann sich in die Anwendung einloggen und diese nutzen. Die Anwendung zeigt die kommenden Kalendereinträge des

1 Einleitung

Tages an und kann somit auch als Dashboard für den kommenden Arbeitstag einzelner Mitarbeiter verwendet werden.

1.5 Projektabgrenzung

Die Einrichtung der Authentifizierung und jegliche art von Geschäftslogik seitens MS Azure AD sowie das Deployment sind nicht Teil des Projekts. Darüber hinaus wird ausdrücklich gewünscht, dass keine Datenbank verwendet wird, da lediglich das Auslesen der Kalendereinträge erforderlich ist. Die Einbindung weiterer Integrationen von Authentifizierungs- und Kalendersupport von z. B. Google, ist nicht Teil des Projekts.

2 Projektplanung

2.1 Projektphasen

Der Zeitraum für das Projekt startet am 26. Februar 2024 und endet am 12. April 2024, wobei es in Gleitzeit durchgeführt wird. Während dieser Zeit wird es im Büro bearbeitet, um eine effektive Zusammenarbeit mit dem Stakeholder zu gewährleisten.

Tabelle 1 zeigt eine grobe Zeitplanung.

Projektphase	Geplante Zeit
Konzeptionsphase	4 h
Anforderungsanalyse	8 h
Recherche geeigneter Technologien	12 h
Erarbeitung der Architektur	8 h
Umsetzung	32 h
Test und Härtung	8 h
Dokumentationserstellung	4 h
Präsentationserstellung	4 h
Gesamt	80 h

Tabelle 1: Zeitplanung

Eine detailliertere Zeitplanung findet sich im Anhang A.1: Detaillierte Zeitplanung auf Seite i.

2.2 Abweichungen vom Projektantrag

In diesem Projekt wurden keine automatisierten Tests eingesetzt, da die Anwendung eine einfache Funktionalität aufweist, die primär auf das Auslesen von Kalendereinträgen ausgerichtet ist. Die Komplexität der Funktionalität sowie die Anforderungen an die Robustheit und Skalierbarkeit der Anwendung sind nicht hoch genug, um den Einsatz automatisierter Tests zu rechtfertigen. Zudem ist der Aufwand für die Implementierung automatisierter Tests im Vergleich zum Nutzen nicht verhältnismäßig. Stattdessen wird auf manuelle Tests gesetzt, um sicherzustellen, dass die grundlegende Funktionalität der Anwendung ordnungsgemäß funktioniert und den Anforderungen entspricht.

2.3 Ressourcenplanung

Es werden ein Arbeitsgerät, eine stabile Internetverbindung sowie eine geeignete Entwicklungsumgebung benötigt, hier wurde dafür Visual Studio (VS) Code als Integrated Development Environment (IDE) verwendet. Da die Anwendung auf MS Graph Application Programming Interface (API) zugreift, ist der Zugang zu Azure AD unerlässlich. Zusätzlich wird im Büro ein Fernseher installiert,

2 Projektplanung

um die Anwendung zu ermöglichen. Für die Bedienung wird eine Bluetooth-Tastatur, Maus und ein Mini-PC bereitgestellt.

Es werden Git für das Source Versioning (SV) und GitLab (GL) zur effektiven und dokumentierte Verwaltung des Quellcodes genutzt. Außerdem wird Jira in diesem Projekt verwendet und dadurch entsteht eine nahtlose Verbindung zwischen Projektverwaltung und Entwicklung. Der Author kann direkt aus Jira-Tickets heraus an den entsprechenden Codeänderungen arbeiten und den Fortschritt der Aufgaben verfolgen.

2.4 Entwicklungsprozess

Das Projekt wird in einem agilen Arbeitsumfeld durchgeführt, wobei eine enge Zusammenarbeit mit dem Stakeholder gepflegt wird. Tägliche Meetings sind angesetzt, um den aktuellen Entwicklungsstand zu besprechen und sich gegenseitig über neue oder geänderte Anforderungen abzustimmen.

3 Analysephase

3 Analysephase

3.1 Ist-Analyse

Bisher müssen Kunden auf ihren Ansprechpartner im Flur warten, was zu einem Verlust an Zeit führt, die für die Aufklärung über den bevorstehenden Termin verwendet werden könnte. In Anbetracht dessen strebt CSTx ES eine digitale Lösung an, um diesen Prozess zu optimieren. Durch die Einführung eines automatisierten Systems, das Kunden vorab über ihre Termine informiert und sie bei ihrer Ankunft begrüßt, kann die Wartezeit dazu genutzt werden sich über den Termin zu informieren und die Kundenzufriedenheit steigern.

3.2 Wirtschaftlichkeitsanalyse

Dieses Projekt bietet eine Vielzahl von Vorteilen. Durch die Automatisierung der Gästebegrüßung und Informationsverteilung (IV) wird die Effizienz im Eingangsbereich erheblich gesteigert.

Durch die Implementierung einer digitalen Lösung, die Kunden vorab über ihre Termine informiert und sie bei ihrer Ankunft automatisiert begrüßt, können Zeitverluste minimiert und der gesamte Terminablauf optimiert werden. Diese Effizienzsteigerungen tragen nicht nur dazu bei, die interne Betriebsabläufe zu verbessern, sondern ermöglichen es auch den Kunden, ihre Zeit effektiver zu nutzen und ihre Erwartungen an einen reibungslosen und professionellen Service zu erfüllen.

Darüber hinaus bietet das Projekt die Möglichkeit, einen bleibenden positiven Eindruck bei den Gästen zu hinterlassen. Ein reibungsloser Empfangsprozess und eine persönliche Begrüßung vermitteln den Kunden das Gefühl, willkommen und geschätzt zu sein. Dies stärkt nicht nur die Kundenbindung, sondern trägt auch dazu bei, das Unternehmensimage zu verbessern und neue Geschäftsmöglichkeiten zu schaffen.

3.2.1 „Make or Buy“-Entscheidung

CSTx ES entscheidet sich für die interne Entwicklung, da es sicherstellen will, dass das Design genau seinen Vorstellungen entspricht und die Informationen präzise auf seine Bedürfnisse zugeschnitten sind. Durch die interne Entwicklung behält das Unternehmen die volle Kontrolle über den Prozess, was die geforderte Flexibilität ermöglicht, um auf Änderungen zu reagieren.

3.2.2 Projektkosten

Rechnung (verkürzt) 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 900 € Brutto.

3 Analysephase

$$8 \text{ h/Tag} \cdot 220 \text{ Tage/Jahr} = 1.760 \text{ h/Jahr} \quad (1)$$

$$900 \text{ €/Monat} \cdot 12 \text{ Monate/Jahr} = 10.800 \text{ €/Jahr} \quad (2)$$

$$\frac{10.800 \text{ €/Jahr}}{1760 \text{ h/Jahr}} \approx 6,14 \text{ €/h} \quad (3)$$

Es ergibt sich also ein Stundenlohn von 6,14 €. Die Durchführungszeit des Projekts beträgt 80 Stunden. Für die Nutzung von Ressourcen¹ wird ein pauschaler Stundensatz von 14 € angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundenlohn von 25 € angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 2 und sie betragen insgesamt 1.806,20 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	80 h	6,14 € + 14 € = 20,14 €	1.611,20 €
Fachgespräche	3 h	25 € + 14 € = 39 €	117 €
Abnahmetest	1 h	25 € + 14 € = 39 €	39 €
Anwenderschulung	1 h	25 € + 14 € = 39 €	39 €
			1.806,20 €

Tabelle 2: Kostenaufstellung

3.2.3 Amortisationsdauer

Das Produkt führt voraussichtlich zu einer Verkürzung der Vorbereitungsdauer eines Termins um etwa 20 Minuten. Zusätzlich wird für das verteilen von allgemeinen Informationen pauschal 30 Minuten einkalkuliert. Angenommen wird, dass durchschnittlich 10 Kunden zu Besuch kommen und sich 3 Mal pro Woche allgemeine Informationen ändern, während 1 Mitarbeiter dafür verantwortlich ist. Unter Berücksichtigung dieser Annahmen ergibt sich folgende Berechnung.

Rechnung (verkürzt) Bei einem Zeiter sparnis von 20 Minuten pro Termin und 30 Minuten für jede IV, bei einem Benutzer und 44 Arbeitswochen im Jahr, wobei sich 10 Termine und 3 IV pro Woche ereignen, ergibt sich folgendes Gesamtzeiter sparnis.

$$44 \text{ Woche/Jahr} \cdot (20 \text{ min/Termin} \cdot 10 \text{ Termin/Woche}) = 8800 \text{ min/Jahr} \approx 147 \text{ h/Jahr} \quad (4)$$

$$44 \text{ Woche/Jahr} \cdot (30 \text{ min/IV} \cdot 3 \text{ IV/Woche}) = 3960 \text{ min/Jahr} \approx 66 \text{ h/Jahr} \quad (5)$$

Dadurch ergibt sich eine jährliche Einsparung von

$$(147 \text{ h} + 66 \text{ h}) \cdot (25 + 14) \text{ €/h} = 8307 \text{ €} \quad (6)$$

¹Räumlichkeiten, Arbeitsplatzrechner etc.

3 Analysephase

Die Amortisationszeit beträgt also $\frac{\text{Entwicklungskosten €}}{\text{Ersparnis €/Jahr}} = \frac{1.806,20}{8307/\text{Jahr}} \approx 0,22 \text{ Jahre} \approx 11,5 \text{ Wochen.}$

3.3 Nicht monetäre Vorteile

Einer der positiven Aspekte, die die Durchführung dieses Projekts mit sich zieht, ist der gegenüber den Kunden und Anwendern des Moduls vermittelte Eindruck einer zukunfts- und kundenorientierten Entwicklung. Wie bereits in der Projektbegründung erwähnt, können sich die Kunden und Besucher davon überzeugen, dass das Unternehmen sich stets weiterentwickelt. Einen ähnlichen Effekt wird diese Umsetzung auf die Mitarbeiter haben. Diese werden sehen, dass Ressourcen für die Erleichterung Ihres Arbeitsalltags aufgewendet werden.

3.4 Anwendungsfälle

Eine Übersicht über die verschiedenen Anwendungsfälle des Projekts im Anhang A.4: Use Case-Diagramm auf Seite iv.

3.5 Qualitätsanforderungen

Die Anwendung muss stabil und konsistent sein, um Ausfälle oder Betriebsunterbrechungen zu vermeiden. Hierbei ist eine robuste Fehlerbehandlung entscheidend, ebenso wie die Fähigkeit, automatisch die zuletzt eingegebenen Informationen wiederherzustellen. Das Design sollte ansprechend sein und sich nahtlos in das Corporate Design einfügen, um dem Image gerecht zu werden. Darüber hinaus ist es wichtig, die Performance im Blick zu behalten, da die Anwendung über Electron.js lokal auf einem nicht leistungsstarken Mini-PC laufen wird.

3.6 Lastenheft/Fachkonzept

Im Rahmen des agilen Projekts werden die Anforderungen in Form eines Lastenhefts, Funktionen und User Stories erfasst, die die Grundlage für die Entwicklung der digitalen Gästebegrüßungs- und Infotafelanwendung bilden.

Die Funktionen umfassen sowohl wesentliche als auch optionale Aspekte. Zu den unverzichtbaren Funktionen gehören die personalisierte Gästebegrüßung, die Anzeige von aktuellen Veranstaltungen und Terminen, die Integration von Live-Video-Streams sowie die Anzeige der aktuellen Uhrzeit. Optionale Funktionen sind die Anpassung von Inhalten durch Mitarbeiter und das Bereitstellen von Hintergrund Animationen wie z. B. einen Ken Burns Effekt.

Die User Stories repräsentieren die Bedürfnisse und Anwendungsfälle aus Sicht der verschiedenen Benutzer.

3 Analysephase

Ein Beispiel für eine User Story lautet: "Als Guest möchte ich beim Betreten der Einrichtung freundlich begrüßt werden und relevante Informationen angezeigt bekommen."

Eine andere User Story ist: "Als Mitarbeiter möchte ich in der Lage sein, die Inhalte der digitalen Infotafel einfach zu aktualisieren und anzupassen."

Ein Lastenheft findet sich im Anhang A.2: Lastenheft (Auszug) auf Seite ii.

4 Vorbereitungsphase

4 Vorbereitungsphase

4.1 Zielplattform

In Bezug auf die Programmiersprache und das Framework fiel die Wahl auf TypeScript (TS) in Verbindung mit dem React-Framework. TS bietet die Möglichkeit, eine robuste und wartbare Codebasis zu entwickeln, während React eine interaktive Benutzeroberfläche ermöglicht. Diese Kombination erleichtert die Entwicklung einer benutzerfreundlichen Anwendung und bietet die Möglichkeit zur Wiederverwendung von Code.

Die Entscheidung für die Laufzeitumgebung fiel auf Electron, um eine plattformübergreifende Kompatibilität sicherzustellen und die Nutzung von Webtechnologien zu ermöglichen. Durch Electron kann die Anwendung als Desktop-App ausgeführt werden und ist somit auf verschiedenen Betriebssystemen wie Windows, macOS und Linux nutzbar.

4.2 Architekturdesign

Die gewählte Architektur basiert auf dem Konzept des Container-Presentational (CP)-Musters, das eine klare Trennung zwischen der Logik der Anwendung und der Darstellung der Benutzeroberfläche ermöglicht. Dieses Muster wird oft in Kombination mit dem React-Framework verwendet und ermöglicht eine effiziente Entwicklung und Wartung der Anwendung.

Im Rahmen dieser Architektur werden die verschiedenen Komponenten der Anwendung in Container-Komponenten und Präsentationskomponenten aufgeteilt. Die Container-Komponenten sind für die Logik der Anwendung zuständig, während die Präsentationskomponenten die Darstellung der Benutzeroberfläche übernehmen. Dadurch wird eine klare Trennung von Daten und Darstellung erreicht, was die Wiederverwendbarkeit und Testbarkeit der Komponenten verbessert.

Die Wahl des React-Frameworks unterstützt diese Architektur, da React die Entwicklung von wiederverwendbaren User Interface (UI)-Komponenten erleichtert und eine effiziente Aktualisierung der Benutzeroberfläche ermöglicht. React verwendet eine virtuelle DOM, um Änderungen effizient zu verarbeiten und die Benutzeroberfläche schnell zu aktualisieren.

4.3 Entwurf der Benutzeroberfläche

Die Benutzeroberfläche besteht aus vier Hauptkomponenten, die auf einen Blick alle relevanten Informationen anzeigen: die Uhrzeit und das Datum, den Live-Video-Feed, die Laufschrift und die Terminauflistung. Die Anordnung der Komponenten erfolgt so, dass sie übersichtlich und leicht erkennbar sind.

4 Vorbereitungsphase

Die Uhrzeit und das Datum werden prominent oben rechts oder oben links auf der Anzeige platziert, um sie leicht sichtbar zu machen. Dies ermöglicht den Besuchern, jederzeit die aktuelle Zeit und das Datum im Blick zu haben.

Der Live-Video-Feed nimmt den Bereich oben links oder unten links der Benutzeroberfläche ein. Dadurch können sie wichtige Ereignisse oder Informationen in Echtzeit verfolgen.

Die Laufschrift wird unten am Rand der Anzeige positioniert und zeigt kontinuierlich wichtige Nachrichten oder Mitteilungen von oder über das Unternehmen an. Dies ermöglicht es den Besuchern, relevante Informationen schnell zu erfassen, selbst wenn sie sich nicht aktiv auf die Anzeige konzentrieren.

Die Terminauflistung wird angezeigt, wenn Termine vorhanden sind, und zeigt eine Liste der kommenden Ereignisse oder Besprechungen über die gesamte rechte Seite an. Wenn keine Termine vorhanden sind, wird die Terminauflistung ausgeblendet, um Platz für die anderen Komponenten zu schaffen. Termine innerhalb der Auflistung werden genutzt um letztlich den Besucher zu begrüßen.

Es gibt kein traditionelles Menü oder UI in der Anwendung. Die Anwendung wird ausschließlich von Hotkeys gesteuert, die von den Benutzern verwendet werden können, um zwischen den verschiedenen Ansichten zu navigieren oder Aktionen auszuführen. Darüber hinaus reagiert die Anwendung auf Kalenderupdates, um Änderungen in den Terminen sofort zu reflektieren und die Anzeige entsprechend anzupassen.

Wireframes Entwürfe für die Oberfläche finden sich im Anhang A.5: Oberflächenentwürfe auf Seite v.

4.4 Geschäftslogik

Eine konkrete Geschäftslogik ist nicht Teil des Projekts und wird von MS Azure AD genutzt. Es wird lediglich das benötigte Anfordern und die dazugehörige Darstellung von Kontakt und Termindaten implementiert.

Dieses Projekts ist klar strukturiert und ist unterteilt zwischen einer Renderer- und Main-Komponente. Hier wird das Electron-Framework verwendet, um sowohl eine Chromium Instanz als auch die Kommunikation zwischen den beiden Komponenten mittels des Observer-Patterns bereitzustellen. Der Renderer nutzt React, TS und Tailwind Cascading Style Sheets (CSS), um eine interaktive und ansprechende Oberfläche zu gestalten und integriert zusätzlich einen Key-Handler für eine vereinfachte Benutzerinteraktion. Innerhalb des Renderers sind verschiedene visuelle Komponenten wie Uhrzeit und Datum, Live-Video, Laufschrift, eine Auflistung von Terminen und Login-Funktion integriert.

4 Vorbereitungsphase

Die Main-Komponente ist für die Verwaltung von Fenstern, Stores und Utilities verantwortlich und nutzt die Integration von Helper-Funktionen. Die Authentifizierung erfolgt über Microsoft Authentication Library (MSAL), während eine MS Graph-Schnittstelle über Axios implementiert wird, um auf die Kalendereinträge von dem Benutzer zuzugreifen.

Diagramm Eine vereinfachte Darstellung der Logik ist im Anhang A.9: Projektstruktur auf Seite xv zu finden.

4.5 Maßnahmen zur Qualitätssicherung

Um die Qualität des Projektergebnisses sicherzustellen, werden ausschließlich Anwendertests durchgeführt. Diese Tests werden von dem Entwickler und tatsächlichen Benutzern der Anwendung durchgeführt, um sicherzustellen, dass sie den Anforderungen und Erwartungen des Stakeholders entsprechen.

Die Anwendertests umfassen eine Reihe von Szenarien und Aufgaben, die die Benutzer durchführen sollen. Diese können bspw. das Live-Video Signal durch die Key-Events wechseln oder einen 0 Uhr Kalendereintrag setzen, um die Konfiguration zu testen.

Es liegt der Fokus darauf, die Anwendung in einer realen Umgebung zu testen und Feedback von den Benutzern einzuholen. Dies ermöglicht es, potenzielle Probleme oder Verbesserungsmöglichkeiten frühzeitig zu identifizieren und direkt in die Entwicklung einzubeziehen.

Die Anwendertests werden kontinuierlich während des Entwicklungsprozesses durchgeführt, somit wird die Anforderungen und die Qualität des Projektergebnisses gewährleistet.

5 Implementierungsphase

5.1 Implementierung der Benutzeroberfläche

Die Implementierung der Benutzeroberfläche erfolgt separat von der Implementierung der Geschäftslogik. In diesem Fall wird das React-Framework verwendet, um die verschiedenen UI-Komponenten zu entwickeln und zu gestalten.

Die Benutzeroberfläche wird mithilfe von TypeScript XML (TSX) erstellt, einer Kombination aus TS und JavaScript XML (JSX), welches einer Syntaxerweiterung von JavaScript (JS) ist, die es ermöglicht, Hypertext Markup Language (HTML)-ähnliche Elemente direkt in JS-Code einzubetten. Dadurch können die verschiedenen UI-Komponenten in einer klaren und strukturierten Weise erstellt werden, wobei die Prozesse zur Formatierung und das Markup eng miteinander verbunden sind.

Für das Styling wird Tailwind CSS, ein Utility-First CSS-Framework, das eine Reihe von vorgefertigten Utility-Klassen bereitstellt, genutzt. Dadurch wird die Entwicklung von CSS stark vereinfacht und beschleunigt, da keine separaten Stylesheets erstellt werden müssen.

Produkt in Aktion Bilder vom Produkt im Eingangsbereich mit Dummy-Daten befinden sich im Anhang A.6: Produktfotos der Anwendung auf Seite vii.

5.2 Implementierung der Geschäftslogik

Die Geschäftslogik umfasst mehrere Schlüsselfunktionen wie das Bereitstellen von Kalenderdaten wofür die Kommunikation mit MS Azure AD benötigt wird. Zusätzlich wird die Verarbeitung von Benutzerinteraktionen und die Verwaltung von Zuständen Bestandteil der Geschäftslogik.

Für die Bereitstellung von Kalenderdaten ist es wichtig, Daten aus MS Azure AD abzurufen, sie zu parsen und entsprechend vorzubereiten, um kommende Termine und Veranstaltungen im Renderer anzuzeigen. Diese Daten müssen dynamisch in die Benutzeroberfläche integriert werden, um den Benutzern einen aktuellen Überblick zu bieten, dies wird erreicht, indem eine Inter-Process Communication (IPC) basierend auf dem Observer-Pattern erstellt wird.

Die Verwaltung von Benutzerinteraktionen ist ein weiterer zentraler Aspekt der Geschäftslogik. Da die Anwendung von einer aktiven Authentifikation abhängig ist, müssen Funktionalitäten definiert und entsprechende Aktionen zugewiesen werden, die den Authorization Code Flow von MS Azure AD nutzen. Dies erfordert das Implementieren von MSAL und einer korrekten Handhabung von Access-Tokens.

Die Verwaltung von Zuständen ist ein entscheidender Aspekt, um sicherzustellen, dass die Benutzeroberfläche der Anwendung korrekt aktualisiert wird und auf Änderungen reagiert. Dies umfasst, das

5 Implementierungsphase

Definieren und Aktualisieren von Konfigurationen und der zu nutzenden Session, sowie das Aktualisieren vom aktuellen Zustand im Renderer.

Teile der Implementierung Der Logik-Loop für das Abrufen der Kalendereinträge in der **MainKomponente** findet sich im Anhang A.7: Main-Komponente Logik-Loop auf Seite xi.

6 Abnahmephase

6 Abnahmephase

Für die Abnahme werden ähnlich wie in der Qualitätskontrolle, Anwendertests von tatsächlichen Benutzern, der Anwendung im vollen Umfang durchgeführt. Die Benutzer sollen wieder typische Interaktionen der Anwendung durchführen, dazu gehören das Nutzen von Key-Events und das prüfen von vorgenommenen Änderungen aus der Qualitätskontrolle.

Insgesamt verliefen die Anwendertests erfolgreich, und die Anwendung erwies sich als stabil, zuverlässig und benutzerfreundlich. Es wurden nur wenige kleinere Probleme gefunden, die schnell behoben werden konnten.

Basierend auf den Ergebnissen der Anwendertests und dem positiven Feedback der Benutzer wurde die Anwendung offiziell abgenommen und für den Produktivbetrieb freigegeben. Sie wurde erfolgreich bereitgestellt und steht nun den Benutzern zur Verfügung.

7 Dokumentation

Die vorliegende Projektdokumentation gibt einen Überblick über die Entstehung und Umsetzung des Projekts. Sie ist in LaTeX10 verfasst worden, orientiert sich an vorhandenen Projektdokumentationen sowie an den Vorgaben der Prüfungsordnung und den offiziellen Richtlinien der Industrie- und Handelskammer (IHK) zur Projektdokumentation.

8 Fazit

8.1 Soll-/Ist-Vergleich

Das Projektziel, eine innovative Lösung für die Gästebegrüßung und den Informationsfluss zu entwickeln, wurde erreicht. Die entwickelte Lösung umfasst eine Auflistung von Termin-Info-Karten, Live-Video-Streams, einen laufenden Info-Text und einer Uhrzeit- und Datumsangabe. Diese Elemente tragen effektiv zur Gästebegrüßung und Informationsbereitstellung bei und vermitteln einen positiven ersten Eindruck. Jedoch gab es einige Herausforderungen während der Implementierung, die zu kleinen Abweichungen führten.

Der Auftraggeber ist zufrieden mit dem Projektergebnis, da die entwickelte Lösung die wesentlichen Anforderungen erfüllt. Dennoch gibt es Punkte, die in Zukunft noch verbessert werden könnten, wie etwa die Integration zusätzlicher Funktionen und Dienste.

Änderungen der Zeitplanung Wie in Tabelle 3 zu erkennen ist, wurde die Zeitplanung durch unerwartete Hindernisse geändert. Als Resultat wurde an anderen Stellen Zeit eingespart.

Phase	Geplant	Tatsächlich	Differenz
Konzeptionsphase	4 h	4 h	
Anforderungsanalyse	8 h	8 h	
Recherche geeigneter Technologien	12 h	18 h	+6 h
Erarbeitung der Architektur	8 h	4 h	-4 h
Umsetzung	32 h	34 h	+2 h
Test und Härtung	8 h	3 h	-5 h
Dokumentationserstellung	4 h	6 h	+2 h
Präsentationserstellung	4 h	2 h	-2 h
Projekt Abgabe	0 h	1 h	+1 h
Gesamt	80 h	80 h	

Tabelle 3: Soll-/Ist-Vergleich

8.2 Lessons Learned

Durch die sehr selbstständige Projektdurchführung wurde der Autor mit allen Prozessbereichen der Softwareentwicklung in Kontakt gebracht.

Während der Durchführung des Projekts hat der Autor eine Reihe wichtiger Lektionen gelernt. Er erkannte die entscheidende Bedeutung einer realistischen Zeitplanung und lernte, wie wichtig es ist, Pufferzeiten für unvorhergesehene Verzögerungen oder technische Herausforderungen einzuplanen. Die Erfahrung verdeutlichte, dass die Einhaltung des Zeitplans oft schwieriger ist als zunächst angenommen und dass Flexibilität und Anpassungsfähigkeit entscheidend sind.

8 Fazit

Des Weiteren erwies sich die Verwendung von TS in Verbindung mit dem React-Framework als äußerst vorteilhaft. Die Typsicherheit von TS half dabei, Fehler frühzeitig zu erkennen und die Codequalität zu verbessern, während React eine effiziente Entwicklung von benutzerfreundlichen UI-Komponenten ermöglichte. Diese Erfahrung bestätigte die Vorteile moderner Frameworks für die Entwicklung hochwertiger Anwendungen.

8.3 Ausblick

Für die zukünftige Entwicklung des Projekts sind verschiedene Erweiterungen und Verbesserungen geplant. Dazu gehören die Erweiterung einer Eingabeoberfläche, der Integration von weiteren Providern und die Anpassung an spezifische Anforderungen.

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase	12 h
1. Analyse des Ist-Zustands	4 h
1.1. Fachgespräch mit dem Stakeholder	2 h
1.2. Brainstorming und Ideengenerierung	2 h
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse	1 h
3. Erstellen des Lastenhefts mit dem Stakeholder	7 h
Vorbereitungsphase	22 h
1. Recherche Benötigter Ressourcen	18 h
1.1 React	5 h
1.2 Electron.js	2 h
1.3 Video.js	2 h
1.4 MSAL	3 h
1.5 Zeitzonen	1 h
1.6 MS Graph und MS Azure AD	3 h
1.7 Authorization Code Flow	2 h
2. Entwurf vom Renderer- und Main-Komponente	2 h
3. Erstellen relevanter Diagramme	2 h
Implementierungsphase	34 h
1. Projekt Boilerplate anlegen	8 h
1.1 Electron.js und React initialisieren	6 h
1.2 Einbinden von TS	1 h
1.3 Einbinden von Tailwind CSS	1 h
2. Benötigte Packages installieren und Konfigurieren	2 h
3. Authentifizierung	11 h
3.1 Integrieren von MSAL	6 h
3.2 Anfragen und Verarbeitung von MS Graph API TS	5 h
4. Komponenten erstellen	13 h
4.1 Uhrzeit und Datum	3 h
4.2 Live-Video Anzeige	4 h
4.3 Laufschrift	1 h
4.4 Termin Auflistung	1 h
4.5 Termin	2 h
4.6 Login	2 h
Testen und Projektübergabe	12 h
1. Anwendertests durchführen	3 h
2. Dokumentation erstellen	4 h
3. Präsentation erstellen	4 h
4. Abgabe Projekt	1 h
Gesamt	80 h

A.2 Lastenheft (Auszug)

Es folgt ein Auszug aus dem Lastenheft mit Fokus auf die Anforderungen:

Die Anwendung muss folgende Anforderungen erfüllen:

1. Authentifizierung
 - 1.1. Die Anwendung muss in der Lage sein, mit MS Azure AD zu interagieren, um die Authentifizierung von Benutzern zu ermöglichen.
 - 1.2. Die Anwendung soll den MS Azure AD Authentication Code Flow verwenden, um die Authentifizierung von Benutzern zu ermöglichen. Dies beinhaltet den Austausch von Autorisierungscodes gegen Access-Token und Refresh-Token.
 - 1.3. Die Anwendung braucht die nötigen Berechtigungen, um auf den vom Benutzer eingeloggten Kalender über MS Azure AD und dem zugehörigen Tenant zuzugreifen.
2. Darstellung der Daten
 - 2.1. Die Anwendung muss eine Liste aller noch folgenden Termine für den aktuellen Tag anzeigen.
 - 2.2. Vergangene Termine müssen 30 Minuten nach Beendigung weiterhin angezeigt werden.
 - 2.3. Jeder Besucher soll über die Oberfläche mit folgenden Informationen begrüßt werden.:
 - 2.3.1 Dem vollständigen Namen.
 - 2.3.2 Der Firmenzugehörigkeit des Besuchers.
 - 2.3.3 Der Kontaktperson oder Organisator des Termins.
 - 2.3.4 Ggf. zusätzlicher oder relevanter Informationen zum Termin.
 - 2.4. Die Anwendung muss eine Laufschrift am unteren Bildschirmrand für allgemeine Informationen bereitstellen.
 - 2.5. Die Anwendung muss einen für die wartenden Besucher bereitgestellten Live-Stream von z. B. den öffentlichen Fernsehprogrammen darstellen.
 - 2.6. Die aktuelle Uhrzeit und Datum muss angezeigt werden.
3. Sonstige Anforderungen
 - 3.1. Die Anwendung soll als gebaute Executable ausführbar sein.
 - 3.2. Die Anwendung soll über den aktuellsten 0 Uhr Kalender-Eintrag konfiguriert werden können. Dazu zählen folgende Konfigurationsmöglichkeiten.:
 - 3.2.1 Das setzen des Hintergrundbildes der Anwendung.
 - 3.2.2 Das setzen des Texts für die Laufschrift.

A.3 Verwendete Ressourcen

Hardware

- MacBook Pro 13 Zoll
- Docking Station
- 2 Monitore
- Bluetooth Headset
- Bluetooth Maus und Tastatur

Software

- Betriebssystem: macOS Sonoma 14.1.2
- Entwicklungsumgebung: VS Code mit Erweiterungen
- Programmiersprache TS 4.5.5
- Auszeichnungssprache: HTML und TSX
- Paketmanager: npm
- Diagrammframework: PlantUML² und Draw.io
- Stylesheetssprache: CSS und Tailwind CSS
- Versionierung: git/GitLab
- Zielplattform: Webbrowser (Chromium)
- Meetings: Microsoft Teams und Büro Räumlichkeiten
- Unternehmensplattform: Microsoft 365
- Mockups: Excalidraw
- Dokumentation: LaTeX
- Dokumentationseditor: VS Code mit Erweiterungen

²Unified Modeling Language (UML)

A Anhang

Personal

- Stakeholder für Betreuung, Codereview und Qualitätssicherung, sowohl für Festlegung der Anforderungen an das Projekt und Projektabnahme.
- Anwender des Produkts.

A.4 Use Case-Diagramm

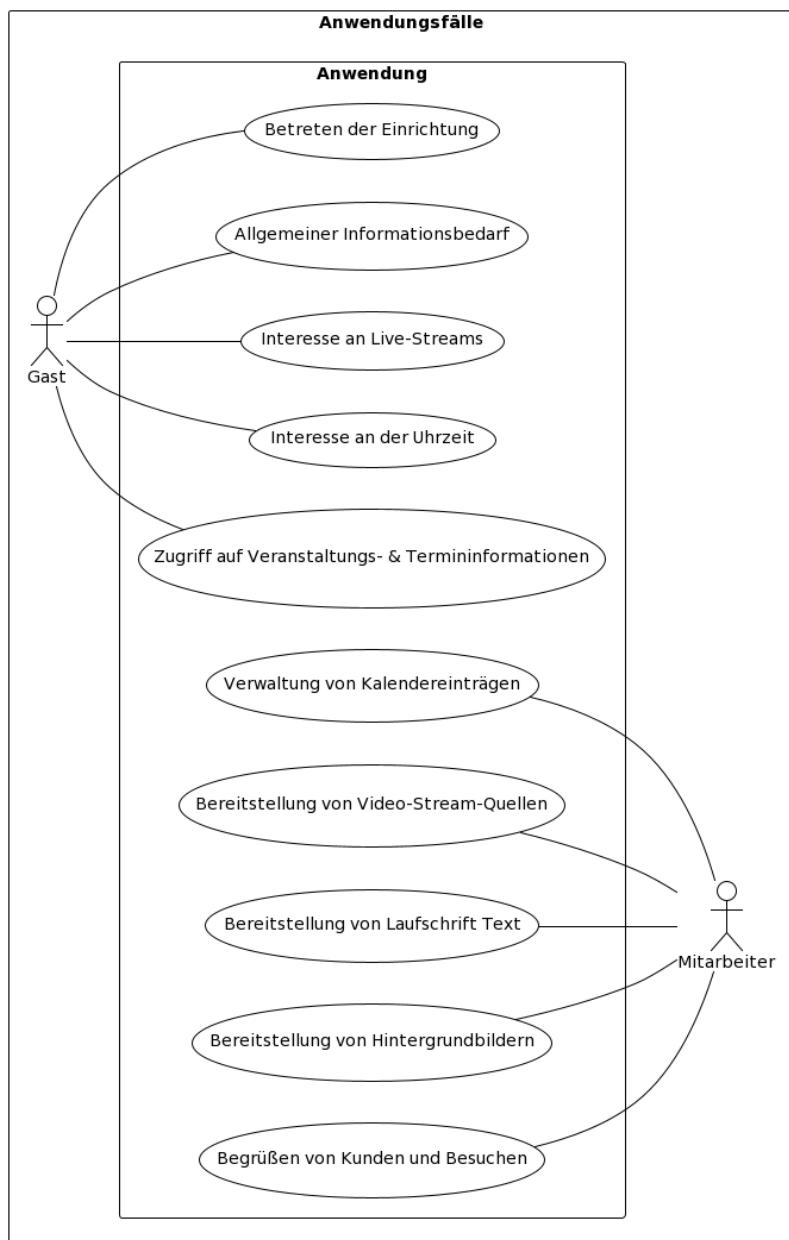


Abbildung 1: Use Case-Diagramm

A Anhang

A.5 Oberflächenentwürfe



Abbildung 2: Wireframe der Standby-Ansicht



Abbildung 3: Wireframe der Fullscreen-Ansicht

A Anhang



Abbildung 4: Wireframe der Ansicht mit einem Termin

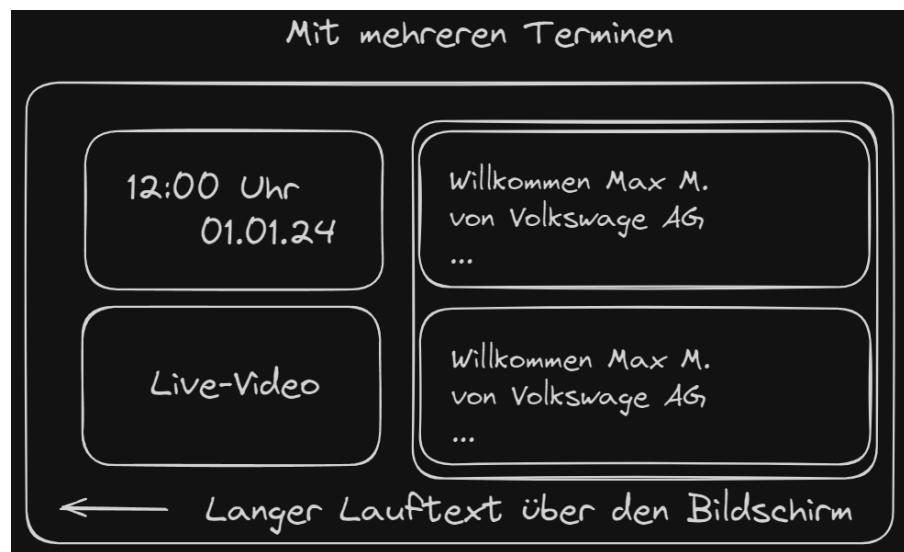


Abbildung 5: Wireframe der Ansicht mit mehreren Terminen

A Anhang

A.6 Produktfotos der Anwendung



Abbildung 6: Ansicht im Standby

A Anhang



Abbildung 7: Vollbildansicht vom Live-Video

A Anhang



Abbildung 8: Ansicht mit einem Termin

A Anhang



Abbildung 9: Ansicht mit mehreren Terminen

A.7 Main-Komponente Logik-Loop

Main Logic-Loop für das Pulling der Kalendereinträge

```
1 //Es wird lediglich der Logik-Loop der Main-Komponente dargestellt!
2
3 setInterval(async (): Promise<void> => {
4     if (!this.isLoggedIn) return;
5
6     const ianaTimeZones = this.findIana(this.options.windowsTimeZone);
7     const startDate = this.moment.tz(ianaTimeZones![0].valueOf()).utc();
8     const events = await this.graph.getUserDayCalendar(this.authResult, ianaTimeZones![0].valueOf(), startDate);
9     const currentDate = new Date();
10    const filteredEvents = [];
11    let configEvent: any = null;
12
13    for(const event of events) {
14        const eventStartDate = new Date(event.start.dateTime);
15        const eventEndDate = new Date(event.end.dateTime);
16        if (eventStartDate.getDay() === currentDate.getDay()) {
17            const preTime = this.moment(eventStartDate).subtract(30, 'm').toDate();
18            const pastTime = this.moment(eventEndDate).add(30, 'm').toDate();
19            const body = event.body.content.toString().replace(/(&nbsp;)/ig, ' ')
20            if ((currentDate > preTime) && (currentDate < pastTime)) {
21                //SETTING NAME
22                const nameFilter = /(?:\$NAME(\s*)=(\s*)(.*?)(?=\n|\$)/gs;
23                const name = nameFilter.exec(body);
24                const nameOutput: string = name ? name[3] : '';
25                //SETTING COMPANY
26                const companyFilter = /(?:\$COMPANY(\s*)=(\s*)(.*?)(?=\n|\$)/gs;
27                const company = companyFilter.exec(body);
28                const companyOutput: string = company ? company[3] : '';
29                //SETTING RESPONSIBLE
30                const contactFilter = /(?:\$CONTACT(\s*)=(\s*)(.*?)(?=\n|\$)/gs;
31                const contact = contactFilter.exec(body);
32                let contactOutput: string = contact ? contact[3] : '';
33                //SETTING INFO-TEXT
34                const infoFilter = /(?:\$INFO(\s*)=(\s*)(.*?)(?=\n|\$)/gs;
35                const info = infoFilter.exec(body);
36                let infoOutput: string = info ? info[3] : '';
37                //SETTING EFFECT
38                const animatedFilter = /\$ANIMATED/;
39                const animated = animatedFilter.exec(body);
40                let animatedOutput: boolean = animated !== null;
41                //SETTING LOCATION
42                const roomOutput = event?.location.displayName || '';
43                //SETTING SUBJECT
44                const subjectOutput = event?.subject || '';
45                //SETTING ATTENDEES
46                const attendeesOutput = event?.attendees || '';
47                //SETTING BACKGROUND IMAGE
```

A Anhang

```
48     const bgFilter = /(?:\$\\$BACKGROUND(\\s)*=(\\s*)(.*?)(?=\\n|\\$\\$)/gs;
49     const bg = bgFilter.exec(body);
50     let bgOutput: string = bg ? url('${bg[3]}') : url('\\public/images/skyline-ez.PNG';
51     filteredEvents.push({
52       subject: subjectOutput,
53       guestName: nameOutput,
54       guestCompany: companyOutput,
55       attendees: attendeesOutput,
56       guestResponsible: contactOutput,
57       location: roomOutput,
58       background: bgOutput,
59       info: infoOutput,
60       animated: animatedOutput,
61       start: event.start.dateTime,
62       end: event.end.dateTime
63     });
64   }
65   else if (event.startDate.getHours() === 0) {
66     //SETTING BACKGROUND IMAGE
67     const bgFilter = /(?:\$\\$BACKGROUND(\\s)*=(\\s*)(.*?)(?=\\n|\\$\\$)/gs;
68     const bg = bgFilter.exec(body);
69     let bgOutput: string = bg ? url('${bg[3]}') : url('\\public/images/skyline-ez.PNG';
70     //SETTING MOTTO OF THE DAY
71     const tickerFilter = /(?:\$\\$TICKER(\\s)*=(\\s*)(.*?)(?=\\n|\\$\\$)/gs;
72     const ticker = tickerFilter.exec(body);
73     let tickerOutput = ticker ? ticker [3] : undefined;
74     tickerOutput += '
75       '; // for additional whitespace after the string
76     configEvent = {
77       background: bgOutput,
78       ticker: tickerOutput,
79     };
80     this.storeHelper.setConfig(configEvent);
81     this.mainWindow.webContents.send('CONFIG_UPDATE', configEvent);
82   }
83 }
84 this.mainWindow.webContents.send('INTERNAL_UPDATE', filteredEvents);
85 }, this.options.intervals.updateEvents);
86
87 this.app.on('window-all-closed', () => {
88   this.storeHelper.setSession(this.authResult);
89   this.app.quit();
90});
```

Listing 1: Main-Komponente Logik-Loop

A.8 Main-Komponente IPC

Main IPC für Kommunikation zwischen Renderer und Main

```

1 //Es wird der Inter-Process Communication teil vom Main-Komponent dargestellt!
2
3 ipcMain.on('LOGIN', async (): Promise<void> => {
4     const authWindow = this.createWindow('auth', this.authWindowOptions);
5     this.authResult = await thisAuthProvider.login(authWindow);
6     authWindow.close();
7
8     if (this.authResult.access_token) {
9         this.storeHelper.setSession(this.authResult);
10        const username = await this.graph.getUserName(this.authResult.access_token);
11
12        this.mainWindow.webContents.send('SUCCESS_LOGIN', username);
13        this.isLoggedIn = true;
14
15        this.continuousTokenUpdate(this.authResult.refresh_token);
16    }
17    else console.error('Couldn't find AccessToken after login!');
18 });
19
20 ipcMain.on('LOGOUT', async (): Promise<void> => {
21     this.isLoggedIn = false;
22     if (this.authResult.access_token) {
23         clearInterval(this.options.refreshInterval);
24         this.storeHelper.clearSession();
25         this.storeHelper.clearConfig();
26         setTimeout(() => {
27             this.mainWindow.webContents.send('INTERNAL_UPDATE', []);
28         }, 300);
29         this.mainWindow.webContents.send('SUCCESS_LOGOUT');
30     }
31 });
32
33 ipcMain.on('LOADED', async (): Promise<void> => {
34     const session = this.storeHelper.getSession();
35     if (session) {
36         this.authResult = session;
37         if (this.authResult.refresh_token) {
38             let username;
39             try {
40                 this.authResult = await thisAuthProvider.getNewToken(this.authResult.refresh_token);
41                 this.continuousTokenUpdate(this.authResult.refresh_token);
42                 username = await this.graph.getUserName(this.authResult.access_token);
43                 this.mainWindow.webContents.send('SUCCESS_LOGIN', username);
44                 this.isLoggedIn = true;
45             } catch (e) {
46                 console.warn('Sign in again!', e);
47             }
48         }
49     }
50 });

```

A Anhang

```
48     }
49     else console.error('Couldn't find RefreshToken after loading the application !');
50   }
51
52
53   this.config = this.storeHelper.getConfig();
54
55   if (this.config) {
56     this.mainWindow.webContents.send('CONFIG_UPDATE', this.config);
57   }
58 });
59
60 ipcMain.on('CHANGE_SCREEN', (event, screenState): void => {
61   this.mainWindow.webContents.send('UPDATE_SCREEN', screenState);
62 });
63
64
65 ipcMain.on('EXIT', (event): void => {
66   this.storeHelper.setSession(this.authResult);
67   this.app.quit();
68 });
```

Listing 2: Main-Komponente IPC

A.9 Projektstruktur

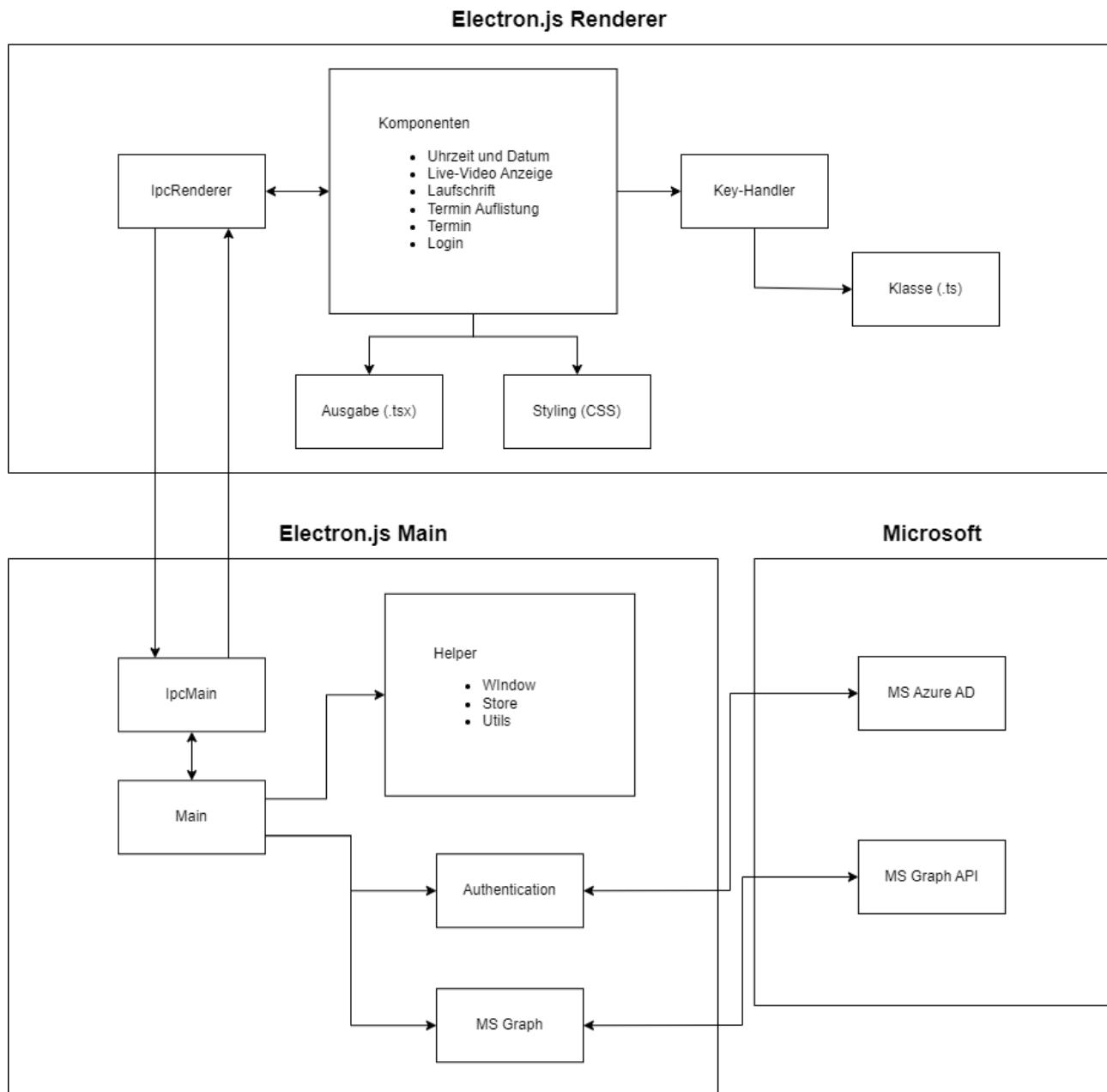


Abbildung 10: Projektstruktur