



Höhere Technische Lehranstalt Reutte

DIPLOMARBEIT

Multi-Room Audio Adapter

Erstellt im Schuljahr 2024/2025

Name der Kandidatin/ des Kandidaten	Jahrgang/Klasse	Individuelle Themenstellung
Nico Lang	2024/2025 5H	Entwicklung der Hardware
Philipp Immler	2024/2025 5H	Erstellen der Software

Eingangsvermerk (Datum und Schulstempel):

EINGELANGT

07. Feb. 2025

**HAK HLW HTL / Reutte
erl.**

Unterschrift (Betreuende):

A handwritten signature in blue ink, appearing to read "Peter Steger".

Dipl. Ing. Dr. Peter L. Steger (Haupt-Betreuende/r)

A handwritten signature in blue ink, appearing to read "Johannes Köll".

Johannes Köll, BEd (Co-Betreuende/r)

Projekt

Projektteam



Nico Lang

Wirtschaftsingenieure/Betriebsinformatik
6651 Häselgehr AT
Nico.Lang@hak-reutte.ac.at



Philipp Immler

Wirtschaftsingenieure/Betriebsinformatik
6682 Vils AT
Philipp.Immler@hak-reutte.ac.at

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen Hilfsmittel als die angegebenen benutzt habe. Die Stellen, die anderen Werken (gilt ebenso für Werke aus elektronischen Datenbanken oder aus dem Internet) wörtlich oder sinngemäß entnommen sind, habe ich unter Angabe der Quelle und Einhaltung der Regeln wissenschaftlichen Zitierens kenntlich gemacht. Diese Versicherung umfasst auch in der Arbeit verwendete bildliche Darstellungen, Tabellen, Skizzen und Zeichnungen. Für die Erstellung der Arbeit habe ich auch Hilfsmittel generativer KI-Tools (ChatGPT 3.5) zu folgendem Zweck verwendet: Inspiration und allgemeine Information. Auch Übersetzer (DeepL) wurden zur Hilfe genommen. Die verwendeten Hilfsmittel wurden vollständig und wahrheitsgetreu inkl. Produktversion und Prompt ausgewiesen.

Reutte, 07.02.2025

Ort, Datum

Nico Lang

Nico Lang

Philipp Immler

Philipp Immler

Abstract Deutsch

Die vorliegende Diplomarbeit beschäftigt sich mit der Entwicklung eines Multi-Room Audio Adapters (MAA). Ein einzelner Adapter besitzt dabei die Funktion, einen Audiostream aus dem Internet auf einem Line-Output auszugeben. So lassen sich die Adapter beispielsweise mit aktiven Lautsprechern per Klinkenkabel verbinden, um so Musik in voneinander getrennten Räumen abzuspielen. Welche Musik in welchem Raum spielt, ist frei konfigurierbar. Dabei erfolgt die Konfiguration per Smartphone-App. Das Endergebnis der Diplomarbeit sind somit der Adapter selbst (physischer Prototyp) und die dazugehörige Smartphone-App.

Die Diplomarbeit lässt sich grob in die drei Teile Planung, Entwicklung und Testen/Fehlerbehebung gliedern.

Abstract English

This diploma thesis deals with the development of a multi-room audio adapter. A single adapter has the function of outputting an audio stream from the Internet to a line output. For example, the adapters can be connected to active speakers via a jack cable in order to play music in separate rooms. Which music plays in which room is freely configurable. The configuration is done via smartphone app. The final result of the thesis is the adapter itself (physical prototype) and the corresponding smartphone app.

The thesis can be roughly divided into three parts: planning, development and testing/troubleshooting.

Danksagung

Wir bedanken uns bei den betreuenden Lehrpersonen Dipl. Ing. Dr. Peter L. Steger und BEd Johannes Köll für die kompetente Unterstützung.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Einleitung Hardware	1
1.2 Einleitung Software	2
2 Planung	2
2.1 Festlegung Funktionsweise	2
2.1.1 Was soll das System können?	2
2.1.2 Was muss es nicht können?	4
2.1.3 Wie könnte man es erweitern?	4
2.2 Auswahl Hardwarekomponenten	4
2.2.1 Auswahl externe Hardware	5
2.2.2 Auswahl interne Hardware	5
2.3 Anforderungen Software Adapter	7
2.4 Anforderungen Smartphone-App	9
2.5 Auswahl Technologien	10
2.5.1 Protokolle	10
2.6 Auswahl Softwaretools	12
2.6.1 Softwaretools zum Schreiben der Diplomarbeit	12
2.6.2 Bibliotheken Mikrocontroller	13
2.6.3 Softwaretools Smartphone-App	15
3 Entwicklung	16
3.1 Entwicklung Software Adapter	16
3.1.1 Klassen	16
3.1.2 Programmablauf	19
3.1.3 Erweiterungsmöglichkeiten	23
3.2 Entwicklung Smartphone-App	23
3.2.1 Struktur	23
3.2.2 Design	25
3.2.3 Komponenten	26
3.2.4 Seiten	28
3.2.5 APIs	30
3.2.6 Zustandsverwaltung	31
3.2.7 Benutzerverwaltung	31
3.2.8 Erweiterungsmöglichkeiten	32
3.3 Design Adaptergehäuse	32
3.3.1 Grundsätzlicher Aufbau	32
3.3.2 Wärmeableitung	32

3.3.3	Virtuelles 3D-Design	33
3.4	Fertigung Adaptergehäuse	36
3.4.1	Drucken des Prototyps	36
3.5	Zusammensetzen des Prototyps	37
3.5.1	Schaltplan	37
3.5.2	Verdrahten	38
3.5.3	Löten	38
3.5.4	Kleben	38
4	Testen und Fehlerbehebung	39
4.1	Testen des Gesamtsystems	39
4.1.1	Testen des Prototyps	39
4.2	Auftretende Fehler beheben	39
5	Schluss	41
5.1	Konklusion Hardware	41
5.2	Konklusion Software	41
Literaturverzeichnis		III
Abbildungsverzeichnis		VI
Tabellenverzeichnis		VI
Anhang		VI

1 Einleitung

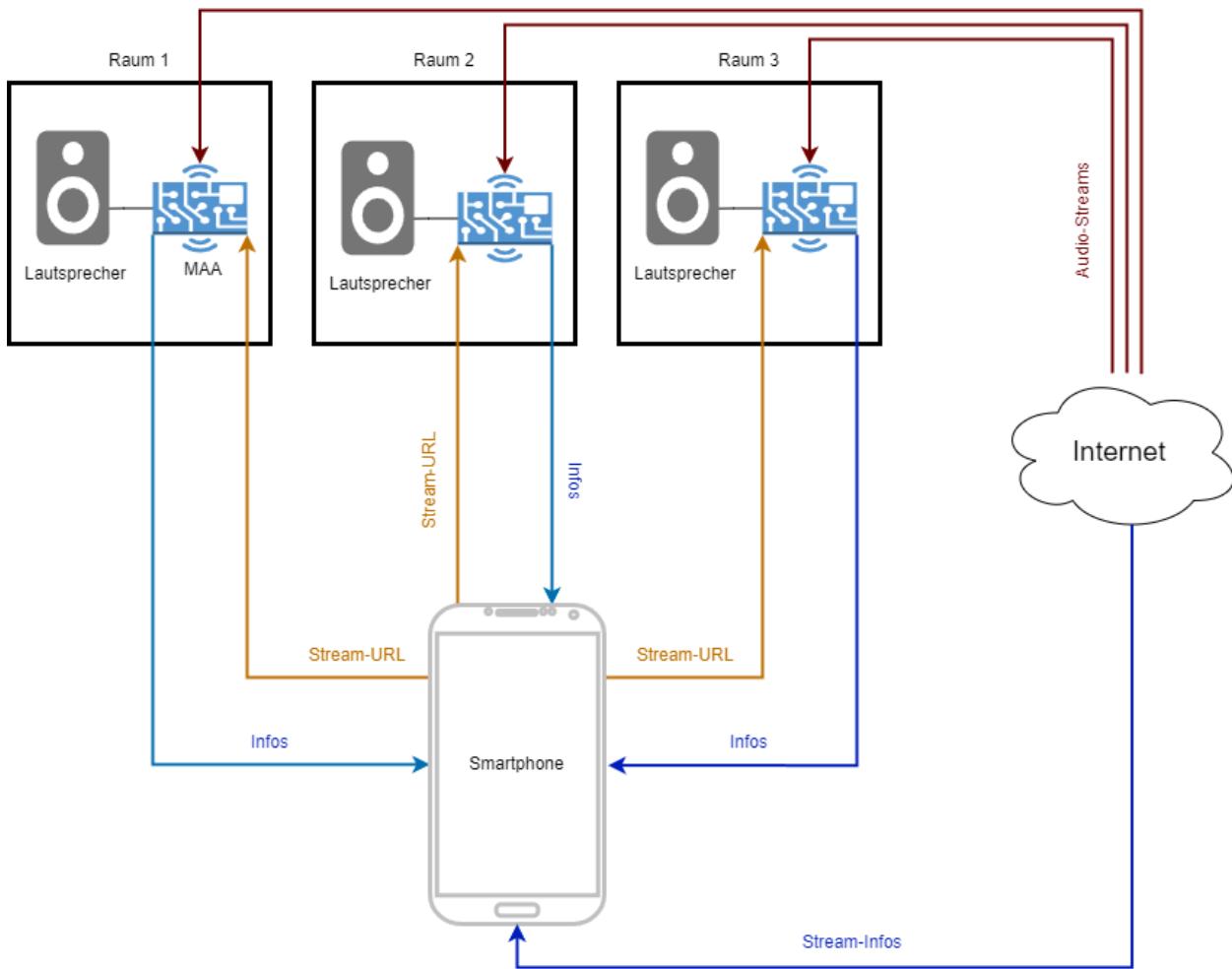


Abbildung 1: MAA Konzept

Hier ist eine vereinfachte Darstellung des groben Konzepts des MAA-Systems abgebildet. In den einzelnen Räumen befindet sich jeweils ein Lautsprecher, welcher per Klinkenkabel mit einem MAA verbunden ist. Der MAA sendet Statusinformationen an das Smartphone, welche dem/der Benutzer/in angezeigt werden. Vom Smartphone aus werden schließlich die URLs des Radio-Streams, welcher der jeweilige MAA empfangen sollen, an diese gesendet. Die Informationen über die Internetradiostationen holt sich die Smartphone-App dabei aus dem Internet. Die Streams selbst empfangen die MAA ebenfalls direkt aus dem Internet.

1.1 Einleitung Hardware

Das Ziel des Hardware-Teils für folgende Diplomarbeit war, einen sinnvollen internen Aufbau des Geräts zu erzielen, die am besten geeigneten Hardware-Komponenten zu finden, das System bzw. die einzelnen Komponenten zusammenzusetzen und zu testen. Dieser Teil der Diplomarbeit wurde

von Nico Lang übernommen. Zudem beschäftigt sich dieser Teil mit dem Gehäuse des Geräts und bestimmt die technischen Anforderungen (Schnittstellen), die der Adapter letztendlich haben soll. Bei der Planung soll zudem darauf geachtet werden, möglichst viele Kosten einzusparen, ohne dabei die Faktoren der Sicherheit und Qualität zu vergessen.

1.2 Einleitung Software

Das Ziel des Software-Teils für folgende Diplomarbeit war, einerseits eine funktionierende Software für den Adapter selbst, andererseits eine funktionierende Smartphone-App zum Steuern des Adapters zu entwickeln. Dieser Teil der Diplomarbeit wurde von Philipp Immler übernommen. Die Software des Adapters wurde mit der Programmiersprache C++ codiert. Für die Entwicklung der Smartphone-App wurde TypeScript verwendet. Bei der Programmierung wurden zahlreiche Bibliotheken und Frameworks verwendet. Dies hat den Vorteil, dass diese schon vorgefertigten Code für bestimmte Probleme bieten und deshalb nicht alles von Grund auf neu entwickelt werden muss.

2 Planung

2.1 Festlegung Funktionsweise

Beim Festlegen einer grundlegenden Funktionsweise des Adapters stellen sich vor allem folgende Fragen:

2.1.1 Was soll das System können?

Das Hauptziel ist, dass das System in verschiedenen, voneinander getrennten Räumlichkeiten bestimmte Audiosignale auf einen Line-Ausgang abspielen kann.

Line-Ausgang

Ein Line-Ausgang (Line-Out) ist eine Ausgangs-Schnittstelle für analoge Audiosignale, deren Ausgangsspannung immer grob dem Line-Pegel entspricht. Dieser „Line-Pegel beträgt etwa 0,5 Volt bis 1 Volt“. (*URLNL01 2025*)

Diese geringe Spannung reicht jedoch nicht, um das Audiosignal direkt an einen Lautsprecher auszugeben. Es muss zuerst noch durch einen Verstärker verstärkt werden. Lautsprecher gibt es mit eingebautem Verstärker (aktive Lautsprecher), es gibt sie jedoch auch ohne internen Verstärker (passive Lautsprecher). (vgl. ebd.)

Aktive vs. Passive Lautsprecher

Ein klassisches Beispiel für passive Lautsprecher sind herkömmliche Hi-Fi Stereoanlagen. Diese bestehen meistens aus einem oder mehreren Playern, einem Verstärker und zwei oder mehreren (Surround Sound, also Raumklang) passiven Lautsprechern. Der Player liest das Signal (beispielsweise einer CD oder einer Schallplatte) und gibt den Line-Pegel über ein Kabel (im Hi-Fi Bereich meist Cinch oder 3,5 mm Klinke) an den Verstärker weiter. Dieser Line-Pegel kann aber auch direkt aus

einem TV-Gerät oder wie in unserem Fall aus einem MAA kommen. Der Verstärker verstärkt das Audiosignal nun von der geringen Spannung des Line-Pegels auf die für die Lautsprecher passende Spannung. Mit dem Lautstärkeregler am Verstärker kann man sich die Spannung (also Lautstärke) letztendlich noch auf persönliche Präferenzen anpassen.

Ein Beispiel für aktive Lautsprecher sind Bluetooth-Lautsprecher, deren Hauptziel es ist, möglichst kompakt und leicht transportierbar zu sein. Solche Bluetooth-Lautsprecher enthalten im Normalfall einen Akku, um auch unterwegs, ohne aktive Stromquelle, Musik hören zu können. Somit enthält das Gehäuse den Verstärker, die Lautsprecher, den Akku und sonstige Elektronik wie unter anderem ein Bluetooth-Modul. Hier fungiert meist ein herkömmliches Smartphone als Signalgeber, ob über Bluetooth oder 3,5 mm Klinke bleibt dem/der Benutzer/-in überlassen.

Lenovo beschreibt Line-Ausgänge zum Beispiel folgendermaßen: „Der Line-Ausgang unterscheidet sich von anderen Audioausgängen wie z. B. Kopfhörerbuchsen, da er ein Signal mit festem Pegel liefert, das nicht von der Lautstärkeregelung Ihres Geräts beeinflusst wird. Er ist für den Anschluss an Geräte gedacht, die das Audiosignal verstärken oder weiterverarbeiten können.“ (*URLNL02* 2025) Man kann also daraus schließen, dass man das Line-Out Signal des MAA vor dem Lautsprecher noch verstärken muss. Wie genau, ist dem/der Endverbraucher/-in überlassen. (vgl. ebd.)

Audioqualität

Zudem ist es wichtig, dass das System den Ton zuverlässig und möglichst flüssig überträgt und ausgibt.

Laut AXIS Communications: „Das menschliche Ohr ist theoretisch in der Lage, Frequenzen von 20 Hz bis 20 kHz zu hören. Die Obergrenze von 20 kHz nimmt mit dem Alter ab.“ (*URLNL03* 2025) Es gibt drei wichtige Grundgrößen, wenn es um Audioqualität geht: Samplerate, Auflösung und Bittiefe.

Jan Baumann hat diese auf seiner Website sehr gut erklärt:

„Die **Samplerate** (Einheit Hz = Hertz) gibt an, wie oft in einer Sekunde der Audio-Pegel erfasst und gespeichert wird. Eine Angabe von 44.100 Hz (44,1 kHz) bedeutet, dass 44.100 Werte für eine Sekunde Musik gespeichert werden. Übliche Sample-Raten sind 44,1 kHz (Musik CD), 48 kHz (Film) und 96 kHz (Tonstudio).“ (*URLNL04* 2025)

„Die **Auflösung** (Einheit Bit) gibt an, wie viel Speicher für so einen Sample-Wert genutzt wird. Zum Beispiel erlauben 16 Bit (2-hoch-16) eine Skala von 65.536 Werten für jeden einzelnen Sample-Wert. Wenn wir viel Speicher für einen Wert haben, können wir das Signal also mit mehr Genauigkeit verarbeiten. Übliche Werte sind 16 Bit (Musik CD) oder 24 Bit bzw. 32 Bit im Studio.“ (ebd.)

„Die **Bitrate** bzw. Datenrate (kBit/s) wird oft mit der Auflösung verwechselt. Sie steht für die “Bandbreite” der Audiodatei, also welche Datenmenge in einer Sekunde verarbeitet wird. Für unkomprimierte Formate wie WAV und AIFF berechnet man die Bitrate ganz einfach, indem man die drei Werte von oben multipliziert.“ (ebd.)

Mehr zu Audioqualität aber im Teil „Auswahl interne Hardware/Digital-/Analogwandler“

Audio-Quellen

Grundsätzlich kann jeder beliebige Audiostream aus dem Internet verwendet werden. Das können beispielsweise Radiosender sein. Ein Beispiel für einen solchen Audiostream wäre der des österreichischen Radiosenders „OE3“:

<https://orf-live.ors-shoutcast.at/oe3-q2a>

Benutzerfreundlichkeit

Es wurde zudem viel Wert auf Benutzerfreundlichkeit gelegt. Das bedeutet, dass sich der Adapter zum einen leicht einrichten lässt, aber auch, dass er sich (mithilfe der Smartphone-App) einfach bedienen lässt.

2.1.2 Was muss es nicht können?

Dieser MAA ist als Hi-Fi-Produkt für den/die klassische/n Durchschnittsbürger/in und/oder Musikliebhaber/innen gedacht. Aufgrund dessen wurde die Bedienung sehr einfach und benutzerfreundlich, jedoch eindeutig nicht so präzise oder vielfältig einstellbar gestaltet, wie es bei professionellem Audio-Equipment der Fall ist. Während der Laie das Produkt einfach anstecken und benutzen möchte, hätte ein Audio-Nerd beispielsweise gerne noch einen eingebauten Acht-Band-Equalizer und vieles mehr. Das war jedoch nicht das Ziel dieser Diplomarbeit. Es ging eher darum, die Hauptfunktion, also Ton kabellos in Räume zu übertragen, und Einstellungsmöglichkeiten per App ohne großes Kopfzerbrechen zu ermöglichen.

Wie schon oben erwähnt liefert der MAA nur einen Audioausgang mit Line-Pegel, und es ist ihm nicht möglich, dieses zu verstärken. Man kann also keine Kopfhörer mit einer Impedanz über 80 Ohm direkt an das Gerät anschließen (man kann theoretisch schon, aber der Ton wird sehr leise sein). Das Audiosignal muss also zuerst mit einem Verstärker verstärkt werden.

2.1.3 Wie könnte man es erweitern?

Unsere Variante des MAA zeichnet sich vor allem durch die beliebige Erweiterbarkeit aus. In der Theorie soll es ein einzelnes Modell, also den Adapter selbst, geben. Mit jedem weiteren Adapter kann dementsprechend ein weiterer Lautsprecher oder ein Raum zugefügt werden. In Zukunft wäre es auch vorstellbar, dass man aus mehreren Adapters Gruppen bilden kann, in denen die Adapter synchronisiert sind und somit der gleiche Audiostream auf mehreren Adapters synchron läuft. Dies ist aber technisch sehr aufwendig, da die Latenz von Wi-Fi ziemlich hoch ist.

2.2 Auswahl Hardwarekomponenten

Zur Auswahl der Hardwarekomponenten des Adapters wurde zuallererst die externe Ausstattung des Adapters überlegt. Das bedeutet praktisch alles, mit dem ein Endverbraucher letztendlich zu tun hat. Dann kann der interne Teil, also die Technik dahinter, individuell auf die Anforderungen des externen Teils designet und entwickelt werden.

2.2.1 Auswahl externe Hardware

Der Adapter sollte ein möglichst kompakt konstruiertes und stabiles Gehäuse bekommen. An diesem ist ein einfacher Taster zur Interaktion angebracht. Mit dem Taster wurden einige Funktionen des Adapters ermöglicht. Beispielsweise per Klick, Doppelklick oder kurzem Halten. Da sich die Aufgaben des Tasters selbst gering halten (Verbindungsvorgang, Ein- und Ausschalten, ...) wurde nur ein einziger Taster verwendet, um die Komplexität des Gesamtsystems zu senken. Die weitaus komplizierteren Funktionen wurden allesamt in der Smartphone-Applikation ermöglicht. Zusätzlich wurde eine RGB-Leuchtdiode zur Statusanzeige verbaut, um beispielsweise den aktuellen Verbindungsstatus zum Mobilgerät und zum Internet anzuzeigen.

Gehäuse

Das Gehäuse soll alle Komponenten auf möglichst kleinem Raum zusammenhalten, schützen und kühlen. Da sich Komponenten und möglicherweise auch das Design selbst laufend änderten, wurde dieses erst gegen Ende des Projekts finalisiert. Mehr dazu im Teil "Design Adaptergehäuse".

Taster

Für den Taster wurde ein herkömmlicher Tactile-Button mit verlängertem Taster in das Gehäuse geplant. Auf ihm klebt eine Art Aufsatz, auf den der/die Endverbraucher/-in letztendlich drückt, um ein gleichbleibendes optisches Design des Gehäuses zu ermöglichen. Mehr dazu im Teil "Design Adaptergehäuse"

LED (Light-Emitting Diode)

Als Statusanzeige wurde in diesem Fall eine herkömmliche RGB-LED verwendet. Mit dieser ist es theoretisch möglich, alle Farben des RGB-Spektrums (16,7 Mio.) mit einer Komponente darzustellen.

Man muss sich jedoch im Klaren darüber sein, für wie viel Spannung die benutzte LED gebaut ist. Dann kann man mit passenden Widerständen arbeiten, um die LED nicht aufgrund zu hoher Spannung zu beschädigen. (vgl. *URLNL05 2025*)

2.2.2 Auswahl interne Hardware

Mikrocontroller (ESP32)

Als Herz des Systems wurde ein ESP32-WROOM-32 Mikrocontroller mit angebauter Platine (DOIT ESP32 DEVKIT V1) verwendet. Der ESP32 ist ein weit verbreiteter Mikrocontroller. Das Konzept des Controllers sind GPIOs (general purpose input/output) also Pins, die man mit Code mit bestimmten Funktionen beschreiben kann. Man kann die Arduino IDE mit C++ als Programmiersprache zum Programmieren verwenden. Zudem verfügt er schon onboard über einen Hybrid Wi-Fi- und Bluetooth-Chip, wodurch externe Module vermieden und somit Platz eingespart werden kann. Espressif selbst beschreibt den ESP32 als optimal für IoT-Anwendungen; auch wegen der hohen Energieeffizienz.

Hier ein paar Grundfakten des ESP32-WROOM-32 (der Controller selbst):

Der von uns benutzte ESP32 hat einen Dual-Core 32-Bit-Prozessor (2x Tensilica-LX6-Kernen). Der Mikrocontroller hat WLAN (802.11b/g/n) und kann ein eigenes WLAN-Netzwerk (Access Point) erstellen (kleiner Webserver). Er unterstützt Bluetooth 4.0 (BLE/Bluetooth Smart) und Bluetooth Classic. Dies ist bei vielen SmartHome und IoT-Anwendungen nützlich. Er hat einen geringen Stromverbrauch von 50-70 mA (kleine Programme ohne Wi-Fi). Der Deep-Sleep Modus macht einen Stromverbrauch von unter 0,1mA möglich. Die Anschaffungskosten fallen mit unter 10 Euro im Vergleich zu anderen Mikrocontrollern auch sehr niedrig aus.

Wie schon erwähnt ist der ESP32 mit der (uns vertrauten) Arduino IDE programmierbar und kann auch in industriellen Umgebungen (-40°C bis +125°C) betrieben werden.

Für den Multi Room Sound-Adapter kommt ein ESP32 Entwicklungsboard (DOIT ESP32 DEVKIT V1) zum Einsatz.

Hier eine Beschreibung von Dev-Kits auf digitalewelt.at: „Um alle Funktionen des ESP32 Moduls einfach nutzen zu können, gibt es sogenannte Entwicklungsboards. Diese sind nicht nur mit zusätzlichen Schaltungen für die Spannungsversorgung ausgestattet, sondern bieten uns auch die gängigen Anschlussmöglichkeiten für unsere externen Komponenten.“ (*URLNL07* 2023)

Erwähnenswert ist auch, dass der ESP32 an manchen GPIOs interne Pullup- /Pulldown-Widerstände hat. „Wenn man einen Pin, an dem z.B. ein Sensor hängt, mit pinMode(5, INPUT) konfiguriert, hat der Pin keinen definierten Pegel. Aus diesem Grund kann er so lange zwischen HIGH und LOW schwanken, bis ihm ein Zustand zugeschrieben wird. Dafür hat der ESP32 interne Pullup- / Pulldown-Widerstände, die zugeschaltet werden können. Diese können mit INPUT_PULLUP als Argument im Befehl pinMode() eingeschaltet werden. Bei INPUT_PULLUP wird der Pin als Eingang deklariert und wenn er nicht beschaltet ist wird dieser auf HIGH gesetzt. Bei INPUT_PULLDOWN hingegen wird der Eingang auf LOW gesetzt.“ (*URLNL08* 2025)

Pullup- /Pulldown-Widerstände sind essenziell für einen sauber konfigurierten Tactile-Button.

(vgl. *URLNL06* 2025)

Digital-/Analogwandler (Audio-Modul)

Dieses Modul ist ausschlaggebend für eine gute Audioqualität. Das digitale Audiosignal vom ESP32 (Radiostream) muss nämlich für die Line-Out Buchse auf analog konvertiert werden. Dafür wird ein Modul mit dem PCM5102A („2VRMS DirectPath™, 112 dB Audio Stereo DAC mit 32-Bit, 384 kHz-PCM-Schnittstelle“) DAC (Digital-Analog-Converter) verwendet. Auf diesem Modul befinden sich alle Komponenten sowie die Klinkenbuchse, die für die Ausgabe des Audiosignals nötig sind. (vgl. *URLNL09* 2025)

Akku (Li-Ion)

Das Endprodukt soll mithilfe eines Akkus auch ohne Strom auskommen, dafür wird ein Lithium-Ionen-Akku mit 3,7 Volt verwendet. Akkus dieser Art zeichnen sich durch ihre hohe Energiedichte und, unter guten Umständen, hohe Lebensdauer aus. Man verwendet Li-Ionen-Akkus meist für (tragbare) Geräte, in denen andere Akkus zu schwer oder zu groß wären. (vgl. *URLNL10* 2025)

Natürlich haben Li-Ionen-Akkus auch gewisse Nachteile und bergen wie jeder andere Akku Gefahren. Beispiele dafür sind elektrische Überlastung, mechanische Beschädigung und thermische Überlastung:

Eine elektrische Überlast kann etliche Gründe haben, darunter:

- Verwendung eines falschen Ladegerätes
- Tiefenentladung
- Falsche Lagerbedingungen (z.B.: zu hohe Temperaturen)

Zitat: „Hier kommt es zur Zersetzung der Elektrolytflüssigkeit und infolgedessen zur Bildung leicht brennbarer Gase. Wird anschließend versucht, die tiefentladenen Lithium-Ionen-Zellen wieder aufzuladen, kann die zugeführte Energie durch das Fehlen von Elektrolytflüssigkeit nicht mehr korrekt umgesetzt werden. Es kann zum Kurzschluss beziehungsweise zum Brand kommen.“ (*URLNL11* 2025)

Eine mechanische Beschädigung jeglicher Art kann zu Kurzschläßen im Inneren der Zelle führen. Da unser Gerät nicht dafür gemacht ist, ständig in Bewegung zu sein, spielt dies keine zu große Rolle, es muss jedoch trotzdem ausreichend Schutz gegeben sein, was durch das Gehäuse sichergestellt wird.

Wie oben schon kurz erwähnt, muss großer Wert auf die richtige Lagerung/Kühlung des Akkus gelegt werden. Wird dieser zu heiß (etwa durch den Mikrocontroller oder sonstige Bauteile) oder durch äußere Einflüsse beschädigt, kann es zum Brand kommen.

Man kann daraus schließen, dass jeder kleinste Fehler beispielsweise zu einem Brand oder sogar einer Explosion des Akkus führen kann. Es ist daher wichtig, den Akku mit absoluter Vorsicht zu handhaben. Ausreichend Tests (Betriebstemperatur, etc.), richtige Konfiguration des Ladereglers und die Auswahl des Akkus sind ausschlaggebend für die Sicherheit des Endverbrauchers und dessen Umfeld. (vgl. ebd.)

Laderegler

Für einen optimalen Ladeprozess und Schutz des Akkus wird ein spezieller Laderegler verwendet. Dieser regelt somit den Ladevorgang des Akkus und hört auf zu laden, sobald dieser voll ist. Der Aufbau dieses Ladereglers ist nicht kompliziert, es gibt jeweils einen Plus- und Minuspol für den Eingang (USB-C Buchse), den Akku (im Englischen auch als Battery bezeichnet) und den Ausgang.

2.3 Anforderungen Software Adapter

Im Folgenden wird beschrieben, welche Anforderungen an die Software des Adapters gestellt wurden:

Access Point

Ein Access Point ist ein Gerät, welches ein WLAN (Wireless Local Area Network) aufbaut. Dabei können sich WLAN-Clients mit diesem verbinden und somit untereinander Daten austauschen. (vgl. *URLPI10* 2024)

Der Mikrocontroller soll im Konfigurationsmodus als Access Point arbeiten. Dabei wird dem/der Benutzer/in ermöglicht, sich mittels Smartphone-App mit dem Netzwerk des Mikrocontrollers zu verbinden und somit die Anmeldedaten des gewünschten Netzwerkes zu senden. Somit kann sich dann der Mikrocontroller in weiterer Folge mit dem gewünschten WLAN verbinden. Die Daten sollen dabei im JSON-Format mittels HTTP von der Smartphone-App an den Mikrocontroller gesendet werden.

WLAN-Client

Ein WLAN-Client ist ein Gerät, welches mit einem Access Point verbunden ist und somit einen Teilnehmer in einem WLAN darstellt. Der WLAN-Client kann dabei Daten mit anderen Clients im Netzwerk austauschen. Wenn der MAA vollständig konfiguriert ist bzw. die Anmeldedaten des zu verbindenden WLANs hat, soll sich dieser als WLAN-Client mit dem gewünschten WLAN verbinden. Um den Datenaustausch zwischen Smartphone und MAA zu ermöglichen, müssen sich beide Geräte im gleichen Netzwerk befinden. Damit der MAA in weiterer Folge Audiostreams von Internetradios empfangen kann, muss dieser zusätzlich mit dem Internet verbunden sein.

REST-API

Der Mikrocontroller soll eine REST-API bereitstellen, mithilfe der es möglich ist, per HTTP Daten mit diesem auszutauschen. Eine REST-API ist eine API, welche den Designprinzipien der REST-Architektur folgt. API steht dabei für „Application Programming Interface“. Eine API bietet im Allgemeinen eine Möglichkeit für den einheitlichen Datenaustausch zwischen Programmen. Eine REST-API (Representational State Transfer Application Programming Interface) ist dabei eine API, welche die von der REST-Architektur vorgegebenen Regeln befolgt. Sie dient zur Kommunikation zwischen Server und Client. Der Client kann mittels HTTP Anfragen an den Server senden. Der Server liefert dann eine Antwort, welche Daten enthalten kann. (vgl. *URLPI04* 2024)

Auf diese Weise kommuniziert die Smartphone-App mit dem Mikrocontroller. Der Client kann dabei vordefinierte Routen aufrufen. Die aufgerufene Route bestimmt die Aktion, welche auf dem Server ausgeführt wird. Es gibt dabei verschiedene Typen von HTTP-Requests. Jeder Typ ist für eine bestimmte Aktion verantwortlich. Für die REST-API Aufrufe des MAA sind GET-, POST- und PUT-Requests möglich. Ein GET-Request wird verwendet, um Daten vom Server abzufragen. Der POST-Request wird verwendet, um neue Daten auf dem Server anzulegen. Mit einem PUT-Request werden Daten auf dem Server aktualisiert. (vgl. *URLPI13* 2024)

Im Folgenden werden die Routen, welche die REST-API des MAA bereitstellen soll, genauer beschrieben:

Route	Anfragen-Typ	Aktion
/getInfo	GET	Der Server (MAA) sendet die Daten: Name, MAC-Adresse, Akkustand, Lautstärke und Stream-URL im JSON-Format an den Client.
/getAvailableNetworks	GET	Der Server sendet eine Liste, welche SSID und RSSI (Stärke) von den Netzwerken, welche sich in der Reichweite von diesem befinden, im JSON-Format an den Client. Diese werden in der Smartphone-App benötigt, um auszuwählen, mit welchem Netzwerk sich der MAA verbinden soll.
/setConfigData	POST	Der Client sendet SSID und Passwort des Netzwerks, mit welchem sich der MAA verbinden soll, im JSON-Format an den Server. Dieser speichert dann die erhaltenen Daten in den EEPROM und startet neu.
/setStreamUrl	PUT	Der Client sendet die URL des Internetradios, von welchem der MAA seinen Audiostream erhalten soll, im JSON-Format an den Server. Dieser startet dann den Empfang dieses Streams und verarbeitet diesen bzw. gibt diesen in weiterer Folge auf dem Line-Out aus.
/setVolume	PUT	Der Client sendet die gewünschte Lautstärke (zwischen 0% und 100%) des Audios an den Server. Dieser setzt dann den Gain (Verstärkung) des Audio-Signals auf einen Wert zwischen 0 und 1.
/pauseStream	POST	Der Server pausiert die Audio-Ausgabe.
/continueStream	POST	Der Server setzt die Audio-Ausgabe fort.

Tabelle 1: REST-API Routen

Audiostreaming

Der Mikrocontroller sollte in der Lage sein, Audiostreams aus dem Internet zu empfangen, diese zu dekodieren und die dekodierten Daten als Signale auf dem Line-Out auszugeben. Dabei beschränken wir uns am Anfang auf Streams im MP3-Format, welches am häufigsten verwendet wird. Bei einem HTTP-Audiostream wird ein Audiofile in mehrere kleine Stücke zerteilt, welche dann per HTTP an den Client gesendet werden. Dieser fügt diese Stücke dann wieder zu einem Audiofile zusammen. Die MP3-kodierten Audiodaten müssen dann in weiterer Folge in ein PCM(Pulse Code Modulation) - Signal dekodiert werden, welches dann von dem Digital-Analog-Wandler in ein analoges Audiosignal umgewandelt wird. Das analoge Audiosignal kann dann auf der Lautsprecherbox ausgegeben werden.

2.4 Anforderungen Smartphone-App

Es wurde festgelegt, dass die Smartphone-App die folgenden Funktionalitäten bereitstellen soll:

Verwaltung von Adapters

Mit der App soll es einerseits möglich sein, neue Adapter zu konfigurieren, andererseits bereits hinzugefügte Adapter zu verwalten. Dabei können von den Adapters Daten, wie MAC-Adresse und Akkustand, angezeigt werden. Außerdem soll regelmäßig überprüft werden, ob die Adapter vom Smartphone aus erreichbar sind.

Verwaltung von Radiostationen

Es sollte zudem möglich sein, nach Audiostreams von Internetradios zu suchen und diese einer persönlichen Favoritenliste hinzuzufügen.

Verwaltung von Verbindungen

In weiterer Folge soll es möglich sein, Verbindungen zwischen Adapters und Internetradio-Streams herzustellen und diese zu verwalten. Dabei soll der Stream gestoppt und fortgesetzt werden können und die Lautstärke der Audio-Ausgabe des MAA einstellbar sein.

Speicherung von Benutzerdaten

Letztendlich soll es noch möglich sein, sich als Benutzer/in anzumelden bzw. zu registrieren und Benutzerdaten, wie Favoritenliste und hinzugefügte Adapter, welche in der Cloud gespeichert sind, abzurufen. Die Speicherung in der Cloud hat dabei den Vorteil, dass der/die Benutzer/in sich auf jedem beliebigen Gerät mit seinen/ihren Anmelddaten anmelden und somit seine/ihre Daten abrufen kann.

2.5 Auswahl Technologien

2.5.1 Protokolle

In diesem Kapitel geht es um die Recherche und Auswahl von Protokollen, die für den Datenaustausch verwendet wurden. Protokolle werden in der Informatik im Allgemeinen verwendet, um eine standardisierte Kommunikation zwischen zwei oder mehreren Geräten zu ermöglichen. Diese standardisierte Kommunikation wird durch das Festlegen von Standards und Normen erreicht. (vgl. *URLPI16* 2025)

HTTP

Das „Hyper Text Transfer Protocol“ ist eines der weitverbreitetsten Protokolle im Web und wird größtenteils für die Kommunikation zwischen Browsern und Webservern eingesetzt. Dabei basiert es grundlegend auf Anfragen (Requests) und Antworten (Responses). Wenn also z.B. ein Client einen HTTP-Request (Anfrage) an einen Server sendet, antwortet dieser im Idealfall mit einer HTTP-Response (Antwort). Das HTTP-Protokoll basiert auf dem TCP/IP-Protokoll, auf welches aber in dieser Diplomarbeit nicht genauer eingegangen wird. In der Diplomarbeit wurde HTTP einerseits für die Kommunikation zwischen Smartphone-App (Client) und Mikrocontroller (Server) mittels REST-API und andererseits zum Streamen der Audio-Daten aus dem Internet verwendet. (vgl. *URLPI01* 2020)

mDNS

Das Multicast Domain Name Service - Protokoll hilft in kleineren, lokalen Netzwerken bei der Namensauflösung. Im Web dominiert das bekanntere DNS (Domain Name Service) - Protokoll. Dieses dient dazu, die IP-Adresse hinter einer Domain zu ermitteln. Dabei gibt es einen DNS-Server, welcher eine Art Liste führt, in der aufgeführt ist, welcher Domainname zu welcher IP-Adresse gehört. Wenn man also im Browser eine URL eingibt, stellt der Client dem DNS-Server eine Anfrage, welche den Domänenamen der angefragten URL beinhaltet. Dieser sendet anschließend die zugehörige IP-Adresse zurück, mit der sich der Client dann verbindet. Dieser Ansatz ist allerdings in kleineren, lokalen Netzwerken eher unpraktisch, da ein eigener Server benötigt wird. Als Alternative dazu dient das sogenannte mDNS -Protokoll, welches mit Multicast arbeitet und daher keinen DNS-Server benötigt. Beim Multicast sendet ein Teilnehmer eines Netzwerks eine Nachricht an mehrere Teilnehmer im Netzwerk. Bei mDNS fragt also das Gerät, welches sich mit einem anderen Gerät verbinden will, andere Geräte im Netzwerk, ob der entsprechende Domänenname zu diesen gehört. Wenn dies der Fall ist, meldet sich das zugehörige Gerät und somit lässt sich eine Verbindung mit diesem herstellen. (vgl. *URLPI19* 2022) In der Diplomarbeit wird mDNS verwendet, um eine Verbindung mit den MAAs aufzubauen. Der Hostname eines MAA setzt sich dabei aus „MAA“ und den letzten drei Byte der MAC-Adresse, getrennt durch einen Unterstrich, zusammen. Dies macht deshalb Sinn, weil somit jeder MAA einen eindeutigen Namen bekommt, da die letzten drei Byte einer MAC-Adresse eines jeden Netzwerkgerätes weltweit eindeutig sind. Die IP-Adresse der MAA kann nicht zur eindeutigen Identifizierung dieser verwendet werden, da in den meisten Netzwerken ein DHCP (Dynamic Host Configuration Protocol) - Server vorhanden ist, der die IP-Adressen automatisch vergibt. Deshalb ändert sich diese ständig. Die MAC-Adresse hingegen bleibt immer gleich. (vgl. *URLPI20* 2025) Mithilfe von mDNS ist jeder Adapter unter seinem Namen gefolgt von „.local“ auffindbar. Ein Adapter mit dem Namen „MAA_3C4D5E“ ist also im lokalen Netzwerk unter „MAA_3C4D5E.local“ auffindbar.

I2S

Das „Inter IC Sound Protocol“ wird verwendet, um Stereo-Audio-Daten zwischen ICs (Integrated Circuits) auszutauschen. Es ist dabei zu unterscheiden von dem I2C (Inter-integrated circuit) - Protokoll, welches rein für den Datenaustausch zwischen ICs verwendet wird und somit nicht für Audiodaten spezialisiert ist. Für die Datenübertragung benötigt das I2S-Protokoll folgende Leitungen:

- Taktleitung
- Wortauswahl
- mindestens eine Datenleitung

Die Datenübertragung erfolgt seriell und synchron. Das heißt, dass die Daten nacheinander durch eine Leitung (die Datenleitung), in einem bestimmten Takt, welcher von der Taktleitung vorgegeben

wird, übertragen werden. Wenn Audiodaten im Stereo-Format (linker und rechter Kanal) übertragen werden, wählt die Wortauswahl-Leitung jeweils den Kanal aus, welcher übertragen werden soll. (vgl. *URLPI12* 2024)

In der Diplomarbeit wurde das I2S-Protokoll verwendet, um die digitalen Stereo-Audio-Daten, welche der Mikrocontroller vom Audiostream erhält, an den Digital-Analog-Wandler zu übertragen. Dieser wandelt das digitale Signal dann in weiterer Folge in ein analoges Signal um. Für die Datenübertragung wurden folgende Parameter verwendet:

- Bittiefe: 16 Bit/Sample
- Audio-Kanäle: 2
- Sample-Rate: 44,1 kHz

2.6 Auswahl Softwaretools

In diesem Kapitel geht es um die Recherche und Auswahl von geeigneten Softwaretools, welche für die App-Entwicklung, als auch für die Entwicklung der Software des Mikrocontrollers verwendet wurden. Des Weiteren werden auch noch die Tools beschrieben, welche für das Schreiben der Diplomarbeit verwendet wurden.

2.6.1 Softwaretools zum Schreiben der Diplomarbeit

Im Folgenden werden die Softwaretools, welche zum Schreiben der Diplomarbeit benutzt wurden, aufgezählt und beschrieben:

LATEX

"LaTeX ist ein hochwertiges Schriftsatzsystem, das Funktionen für die Erstellung technischer und wissenschaftlicher Dokumentationen enthält. LaTeX ist der De-facto-Standard für die Kommunikation und Veröffentlichung von wissenschaftlichen Dokumenten." (*URLPI29* 2025) LaTeX wurde für die Diplomarbeit gewählt, weil es sich sehr gut für das Schreiben von wissenschaftlichen Arbeiten, vor allem mit technischem Bezug, eignet und wir somit schon damit vertraut sind, wenn wir es in Zukunft, z.B. im Studium, benutzen müssen.

draw.io

"draw.io ist eine kostenlose Online-Diagrammsoftware zur Erstellung von Flussdiagrammen, Prozessdiagrammen, Organigrammen, UML, ER und Netzwerkdiagrammen." (*URLPI30* 2025)

Alle Diagramme, welche in der Diplomarbeit zu sehen sind, wurden in draw.io erstellt, weil es einfach zu handhaben ist und eine sehr große Auswahl an Diagrammtypen und Formen bereitstellt.

Visual Studio Code

„Visual Studio Code ist ein leichtgewichtiger, aber leistungsstarker Quellcode-Editor, der auf Ihrem Desktop läuft und für Windows, macOS und Linux verfügbar ist. Er bietet integrierte Unterstützung für JavaScript, TypeScript und Node.js und verfügt über ein umfangreiches Ökosystem von Erweiterungen für andere Sprachen und Laufzeiten (wie C++, C#, Java, Python, PHP, Go, .NET).“ (*URLPI31* 2025)

Visual Studio Code wurde als IDE (Integrated Development Environment) für den Code der Diplomarbeit gewählt, weil durch die unzähligen Erweiterungen viele verschiedene Programmiersprachen und Bibliotheken unterstützt werden und somit der gesamte Code, sowohl für den Mikrocontroller als auch für die Smartphone-App, darin geschrieben werden konnte.

GitHub

Git im Allgemeinen ist das meistverbreitete Versionskontrollsystem. Ein Versionskontrollsystem wird verwendet, um den Versionsverlauf einer Software zu erfassen. Dies hat den Vorteil, dass man bei auftretenden Fehlern jederzeit wieder auf eine ältere Version der Software umsteigen kann. Ein weiterer Vorteil beim Einsatz eines Versionskontrollsysteins ist, dass mehrere Entwickler gleichzeitig an derselben Software arbeiten können. Git gehört zu den DVCS (Distributed Version Control Systems), also verteilten Versionskontrollsysteinen. Das heißt, dass der Versionsverlauf der Software nicht zentral auf einem Server gespeichert ist, sondern auf den Rechnern der Benutzer, in einem sogenannten Repository. Dies bietet höhere Sicherheit, Performance und Flexibilität. (vgl. *URLPI17* 2025)

GitHub ist eine Software, die Git verwendet. Dabei ist Github cloud-basiert, das heißt, es ist online zugänglich und benötigt deshalb keine eigene Software auf dem Rechner. Außerdem hat Github eine sehr benutzerfreundliche Oberfläche, was es für allem für Einsteiger leichter verständlich macht. (vgl. *URLPI18* 2025) In der Diplomarbeit wurde GitHub für die Verwaltung des Codes und der Dokumente verwendet. Der Vorteil dabei ist, dass jedes Projektmitglied auf seinem lokalen PC an den Dokumenten arbeiten kann und die Änderungen dann per GitHub synchronisiert werden können.

DeepL

Bei DeepL handelt es sich um einen Übersetzer, der KI einsetzt und somit einer der genauesten Übersetzer weltweit ist. DeepL wurde aufgrund seiner Genauigkeit und kostenlosen Verfügbarkeit in der Diplomarbeit verwendet, um englische Texte ins Deutsche zu übersetzen.

2.6.2 Bibliotheken Mikrocontroller

Im Folgenden werden die Bibliotheken, welche für die Programmierung der Mikrocontroller-Software verwendet wurden, aufgezählt und beschrieben.

Arduino Core

(<https://github.com/espressif/arduino-esp32>)

Die „Arduino-Core“-Bibliothek wurde verwendet, um den ESP32 ähnlich wie ein Arduino-Board programmieren zu können. Es erleichtert dabei die Programmierung enorm. Vor allem dann, wenn man schon Vorerfahrung mit der Programmierung von Arduino-Boards hat. Ein weiterer Vorteil ist, dass diese Bibliothek bereits weitere nützliche Bibliotheken beinhaltet, welche für die Programmierung benötigt werden.

WiFi

(<https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi>)

Die „WiFi“-Bibliothek ist eine offizielle Bibliothek von Arduino, welche in der „Arduino-Core“-Bibliothek inkludiert ist. Sie wird verwendet, um die Funktionen der eingebauten Wi-Fi-Antenne des ESP32 zu nutzen. Der ESP32 kann dabei entweder als Access Point oder als Client fungieren. Wenn er als Access Point fungiert, stellt er ein eigenes Wi-Fi-Netzwerk bereit, mit dem sich andere Geräte verbinden können und der ESP32 somit einen Host darstellt. Als Client kann er sich mit anderen Wi-Fi-Netzwerken bzw. Access Points verbinden. In der Diplomarbeit fungiert der ESP32 sowohl als Access Point, als auch als Client. (vgl. *URLPI24* 2025)

ArduinoJson

(<https://github.com/bblanchon/ArduinoJson>)

Die „ArduinoJSON“-Bibliothek wird verwendet, um Daten in das JSON-Format zu codieren. JSON (Java Script Object Notation) ist ein Datenformat, welches für den einheitlichen Datenaustausch zwischen Servern und Clients verwendet wird. Dabei verwendet JSON sogenannte Schlüssel-Wert-Paare. Das heißt, einem Wert ist immer ein eindeutiger Schlüssel zugeordnet. In der Diplomarbeit realisiert die „ArduinoJSON“-Bibliothek den einheitlichen Datenaustausch zwischen Webserver (ESP32) und Client (Smartphone). (vgl. *URLPI11* 2024)

WebServer

Die „WebServer“-Bibliothek wird verwendet, um einen Webserver auf dem ESP32 bereitzustellen. Dieser ist in Kombination mit der REST-API wichtig für den Datenaustausch zwischen MAA und Smartphone-App. Auf die REST-API wurde bereits im Kapitel 2.3 genauer eingegangen. Der Webserver kann Anfragen von Clients entgegennehmen und diesen Antworten zurücksenden. Dabei gibt es vordefinierte Routen, welche aufrufbar sind.

ESP8266Audio

(<https://github.com/earlephilhower/ESP8266Audio>)

Die „ESP8266Audio“-Bibliothek wurde ursprünglich für das Vorgängermodell des ESP32, den ESP8266, entwickelt. Sie wurde allerdings mittlerweile von den Entwicklern angepasst, sodass man sie auch für den ESP32 verwenden kann. Die Bibliothek stellt Funktionen zum Empfangen, Dekodieren und Ausgeben von Audio-Daten bereit. Die Bibliothek wurde in der Diplomarbeit verwendet, um MP3-Streams vom Internet zu empfangen, diese zu dekodieren und die dekodierten PCM-Signale über

I2S an den DAC zu senden. Der ESP32 verfügt zwar bereits standardmäßig über Funktionen, mit deren Hilfe man Audiodaten mittels I2S übertragen kann, allerdings sind diese sehr komplex in der Verwendung und Konfiguration. Da die „ESP8266Audio“-Bibliothek bereits die perfekte Lösung für die Anforderungen der Mikrocontroller-Software bietet, wurde diese in der Diplomarbeit verwendet.

Preferences

(<https://github.com/espressif/arduino-esp32/tree/master/libraries/Preferences>)

Die „Preferences“-Bibliothek wird verwendet, um Schreib- und Lesezugriffe auf den eingebauten EEPROM (Electrical Eresable Programmable Read Only Memory) des ESP32 durchzuführen. Dabei werden die im EEPROM gespeicherten Werte mittels eindeutiger Schlüssel identifiziert.

2.6.3 Softwaretools Smartphone-App

TypeScript

Bei TypeScript handelt es sich um eine Erweiterung der Programmiersprache „JavaScript“. Der Vorteil von TypeScript ist, dass dieses eine statische Typisierung ermöglicht. Reines JavaScript hat eine dynamische Typisierung. Das heißt, dass den definierten Variablen erst durch den Interpreter ein konkreter Datentyp zugewiesen wird. Im Gegensatz dazu kann man in TypeScript den Variablen direkt konkrete Datentypen zuweisen. Dadurch wird eine höhere Typensicherheit und eine bessere Wartbarkeit des Codes garantiert. (vgl. *URLPI32 2023*)

Aufgrund dieser Vorteile wurde TypeScript in der Diplomarbeit verwendet.

Node.js

Bei Node.js handelt es sich um eine JavaScript-Laufzeitumgebung, welche es ermöglicht, JavaScript-Code und somit auch TypeScript-Code lokal auf dem Rechner auszuführen. Normalerweise ist JavaScript eine Programmiersprache, welche für die Frontend-Entwicklung eingesetzt wird und nur im Browser läuft. Mit der Verwendung von Node.js ist es allerdings möglich, JavaScript auch im Backend, also lokal auf dem Rechner, zu verwenden. Node.js stellt außerdem einen eigenen Paket-Manager namens „npm“ (Node Package Manager) bereit, der es ermöglicht, zahlreiche Open-Source-Pakete, für unterschiedlichste Anwendungen zu installieren. Somit können mithilfe von Node.js die Vorteile, welche JavaScript bietet, zur Entwicklung zahlreicher Apps verwendet werden. (vgl. *URLPI25 2021*) Node.js wurde in der Diplomarbeit verwendet, um die Entwicklung mit React Native bzw. Expo zu ermöglichen.

React Native

React Native ist ein Framework, welches die plattformübergreifende Entwicklung von Apps ermöglicht. Das heißt, man schreibt einen Code in JavaScript bzw. TypeScript und kann aus diesem dann eine IOS-, Android- oder Web-App bauen. Um eine React Native App auszuführen, wird die Node.js-Laufzeitumgebung benötigt, welche im vorherigen Teil beschrieben wurde. React Native wurde erstmals 2015 von Meta (damals noch Facebook) als Open-Source-Projekt veröffentlicht. Seither wird es weiterhin von Meta gewartet bzw. weiterentwickelt und hat eine riesige Community,

welche ständig neue Bibliotheken für das Framework veröffentlicht. React Native basiert auf React, welches man bereits aus der Webentwicklung kennt. Der Vorteil von React im Gegensatz zur normalen Webentwicklung ist, dass man wiederverwendbare Komponenten bauen kann. Wenn sich der Zustand einer Komponente ändert, wird nur diese neu gerendert (erstellt/berechnet) und nicht die ganze Seite. Dieses Verhalten wird auch in React Native ermöglicht. In React Native stehen dabei einige Standardkomponenten zur Verfügung, welche dann jeweils in native Komponenten, passend für das jeweilige Betriebssystem, gerendert werden. Aus den Standardkomponenten kann man schließlich seine eigenen Komponenten bauen, welche dann ganz einfach im Programm inkludiert werden können. React Native wurde für die Entwicklung der Smartphone-App verwendet, weil es sehr aufwendig gewesen wäre, für jede Plattform eigenen Code zu schreiben und dies außerdem tiefes Vorwissen in der App-Entwicklung vorausgesetzt hätte. (vgl. *URLPI07* 2024)

Expo

Um das Entwickeln der Smartphone-App noch einfacher bzw. effizienter zu gestalten, wurde das Expo-Framework verwendet. Dieses Framework basiert wiederum auf React Native und stellt zusätzlich hilfreiche Bibliotheken und eine Projektstruktur zur Verfügung. Dies hat den Vorteil, dass der grundlegende Aufbau einer App bereits vorgegeben wird und man somit von Beginn an eine solide Basis hat, auf der man weiterentwickeln kann. Expo bietet zudem das sogenannte „File-based routing“, welches die Entwicklung der App-Navigation sehr erleichtert. Ein weiterer Vorteil von Expo ist, dass für das Testen der App die „ExpoGo“-App verwendet werden kann. Diese ermöglicht es, die App bereits bevor diese vollständig fertiggestellt ist, auf dem Smartphone zu testen. Dadurch wird die Entwicklung um ein Vielfaches erleichtert. (vgl. *URLPI08* 2024)

3 Entwicklung

3.1 Entwicklung Software Adapter

In diesem Kapitel wird der Übergang der Planung in die Entwicklung der Software des Mikrocontrollers, welcher im MAA verbaut ist, beschrieben. Zur Entwicklung der Software des Mikrocontrollers wurde die IDE Visual Studio Code, welche bereits im Kapitel 2.6 beschrieben wurde, in Verbindung mit dem Framework „PlatformIO“ verwendet. PlatformIO ist ein Framework, welches für die Entwicklung von Embedded Systems verwendet wird und somit einige hilfreiche Funktionen in Verbindung mit dem ESP32 bietet. (vgl. *URLPI26* 2025) Um die Entwicklung so effizient wie möglich zu gestalten, wurden die Bibliotheken, welche bereits im Kapitel 2.6.2 beschrieben wurden, verwendet.

3.1.1 Klassen

Um die Modularität und somit die Wiederverwendbarkeit und Wartbarkeit des Codes zu gewährleisten, wurde in der Entwicklung der Software für den Adapter auf objektorientierte Programmierung gesetzt. Dies ist möglich, da die Programmiersprache C++, welche für das Schreiben der Software

verwendet wurde, objektorientierte Programmierung unterstützt. Für einen Großteil der Klassen wurde das Singleton-Designpattern verwendet. Dieses gewährleistet, dass immer nur eine Instanz der Klasse vorhanden ist, was Sinn ergibt, wenn von einer Klasse nicht mehrere Instanzen verfügbar sein sollen. (vgl. *URLPI14* 2024).

Das folgende UML-Klassendiagramm veranschaulicht die Beziehung der verschiedenen Klassen zueinander:

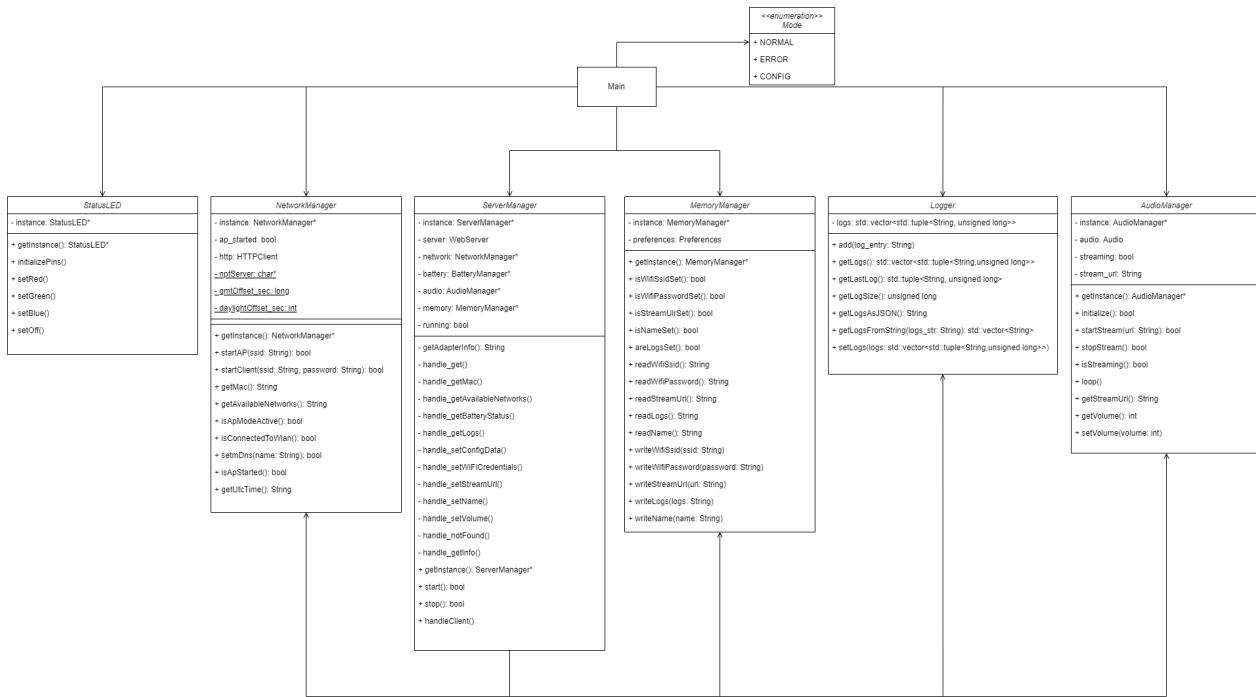


Abbildung 2: UML-Klassendiagramm Mikrocontroller
(Im „Anhang 2“ befindet sich eine vergrößerte Abbildung des Diagramms)

Im Folgenden werden die einzelnen Klassen genauer beschrieben:

Mode

Im ENUM „Mode“ werden die verschiedenen Modi der Software definiert. Ein ENUM ist wie eine Art Datentyp, der Konstanten definiert, welche man verwenden darf. (vgl. *URLPI15* 2024)

In unserem Fall definiert der ENUM die drei Modi „NORMAL“, „CONFIG“ und „ERROR“. Diese geben den Status des MAA an. Der Modus „NORMAL“ signalisiert, dass sich der MAA im normalen Zustand befindet, dieser also voll funktionsfähig ist. Im Modus „CONFIG“ fungiert der MAA als Access Point. Somit können sich Clients mit diesem verbinden und die Anmeldedaten des WLANs, mit dem sich der MAA verbinden soll, senden. Der Modus „ERROR“ wird gesetzt, wenn ein Fehler im Programm auftritt. Dies ist zum Beispiel der Fall, wenn die Verbindung mit dem WLAN unterbrochen wird. Der aktuelle Modus wird mithilfe der Status-LED signalisiert, welche als nächstes genauer beschrieben wird.

StatusLED

Mithilfe der Klasse „StatusLED“ wird die RGB-LED, welche am Mikrocontroller angeschlossen ist, gesteuert. Mit ihr wird der aktuelle Modus des MAA dem/der Benutzer/in angezeigt. Die nachstehende Tabelle beschreibt, welche Farbe für welchen Modus steht:

Modus	Farbe
NORMAL	Grün
CONFIG	Blau
ERROR	Rot

Tabelle 2: Status-Farben

NetworkManager

Die Klasse „NetworkManager“ beinhaltet Funktionen, welche die Wi-Fi-Funktionalität des ESP32 verwenden. Es ist hier möglich, den Access Point - und den Client-Modus zu aktivieren, nach verfügbaren Netzwerken zu suchen, sowie die MAC-Adresse des Adapters zu lesen. Diese Funktionen werden alle mithilfe der im Kapitel 2.6.2 erwähnten „WiFi“-Bibliothek verwirklicht.

AudioManager

Die Klasse „AudioManager“ ist für den Empfang des Internetradio-Streams, für das Dekodieren des empfangenen Streams und für die Ausgabe des dekodierten Audio-Signals zuständig. Dafür wird primär die „ESP8266Audio“-Bibliothek, welche bereits im Kapitel 2.6.2 beschrieben wurde, verwendet.

Logger

Die Klasse „Logger“ ist für die Verwaltung von Logs, welche von anderen Klassen geschrieben werden, zuständig. In ihr befindet sich ein Vektor, welcher Tupels, bestehend aus Log-Text und Log-Zeitpunkt, speichert, definiert. Diese Logs sind essenziell für die Fehlerbehebung, da man sehen kann, welche Aktionen im Programm ausgeführt wurden und wo Fehler aufgetreten sind. Die Klasse bietet Funktionen zum Hinzufügen von Logs und zum Anzeigen bestehender Logs. Die Funktionen sind alle statisch, das heißt, man muss kein Objekt dieser Klasse erzeugen, um die Funktionen auszuführen. Dies ist sinnvoll, weil es möglich sein soll, von jeder Klasse direkt auf die Log-Funktionen zuzugreifen.

ServerManager

Die Klasse „ServerManager“ ist für das Verwalten des Webservers, welcher auf dem MAA läuft, zuständig. In ihr werden die Funktionen, die durch die Aufrufe der Routen der REST-API ausgeführt werden, definiert. Die Kernfunktionen der Klasse werden vor allem durch die im Kapitel 2.6.2 erwähnte „WebServer“-Bibliothek bereitgestellt.

MemoryManager

Die Klasse „MemoryManager“ ist für das Schreiben in und das Lesen aus dem EEPROM des Mikrocontrollers zuständig. Der EEPROM ist ein nichtflüchtiger Speicher, das heißt, er behält die Daten auch nach einem Spannungsverlust des Mikrocontrollers. Dies ist vor allem für Daten sinnvoll, welche nicht jedes Mal neu definiert werden, wie z.B. Anmeldedaten des WLANs. (vgl. *URLPI27* 2018) Um die Funktionen des EEPROMs des ESP32 zu nutzen, wurde die „Preferences“-Bibliothek, welche ebenfalls im Kapitel 2.6.2 beschrieben wurde, verwendet. Diese arbeitet mit Schlüssel-Wert-Paaren. Das heißt, auf die gespeicherten Variablen kann mittels eindeutigem Schlüssel zugegriffen werden.

BatteryManager

Die Klasse „BatteryManager“ ist für das Auslesen des Akkustands des ESP32 zuständig. Dies wird erreicht, indem der Spannungspegel an einem analogen Eingang des ESP32 eingelesen wird und aus diesem dann der Ladestand des Akkus errechnet wird.

3.1.2 Programmablauf

Da die „Arduino-Core“-Bibliothek verwendet wurde, ist der Programmablauf gleich wie bei einem standardmäßigen Arduino-Programm. Ausgeführt wird die Main-Datei, welche aus den Funktionen „setup“ und „loop“ besteht. Die Setup-Funktion wird beim Start des Mikrocontrollers einmalig ausgeführt. In ihr werden also alle Funktionen, die für den weiteren Verlauf des Programms essenziell sind, ausgeführt. Anschließend wird die Loop-Funktion in einer Endlosschleife ausgeführt. Wenn also das Ende der Loop-Funktion erreicht wird, fängt diese wieder von vorne an. Im folgenden Teil wird der Ablauf innerhalb der einzelnen Funktionen genauer beschrieben.

Setup

Im folgenden UML-Ablaufdiagramm wird der grobe Programmablauf der Setup-Funktion dargestellt:

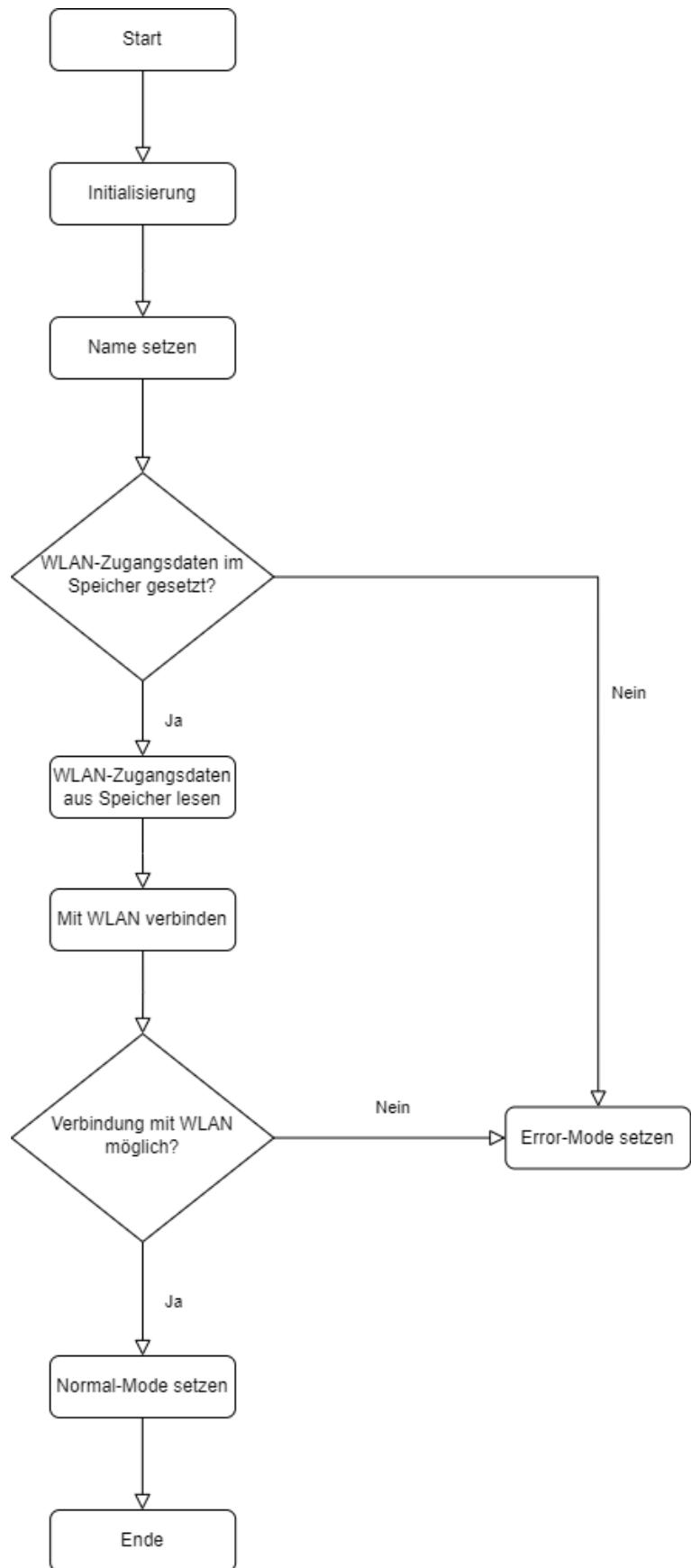


Abbildung 3: UML-Ablaufdiagramm Mikrocontroller Setup

In der Setup-Funktion werden zuerst Konstanten, welche später im Programm benötigt werden, definiert. Auch werden am Anfang die entsprechenden Instanzen der Singleton-Klassen abgerufen. Anschließend wird der Name des Adapters definiert. Dieser setzt sich, wie bereits im Kapitel 2.5.1 erwähnt, aus „MAA“ und den letzten drei Byte der MAC-Adresse zusammen. Er ist somit für jeden Adapter eindeutig. Der Name wird später als SSID des Access Points bzw. als Hostname für den Webserver verwendet. Nachdem der Name gesetzt ist, wird mit der Klasse „MemoryManager“ abgefragt, ob WLAN-Anmeldedaten im EEPROM gespeichert sind. Sind diese vorhanden, so versucht der MAA eine Verbindung mit dem WLAN herzustellen. Ist dies möglich, wird der Modus auf „NORMAL“ gesetzt. Schlägt die Verbindung mit dem WLAN, z.B. durch falsche Anmeldedaten, fehl, wird der Modus auf „ERROR“ gesetzt.

Loop

Im folgenden UML-Ablaufdiagramm wird der grobe Programmablauf der Loop-Funktion dargestellt:

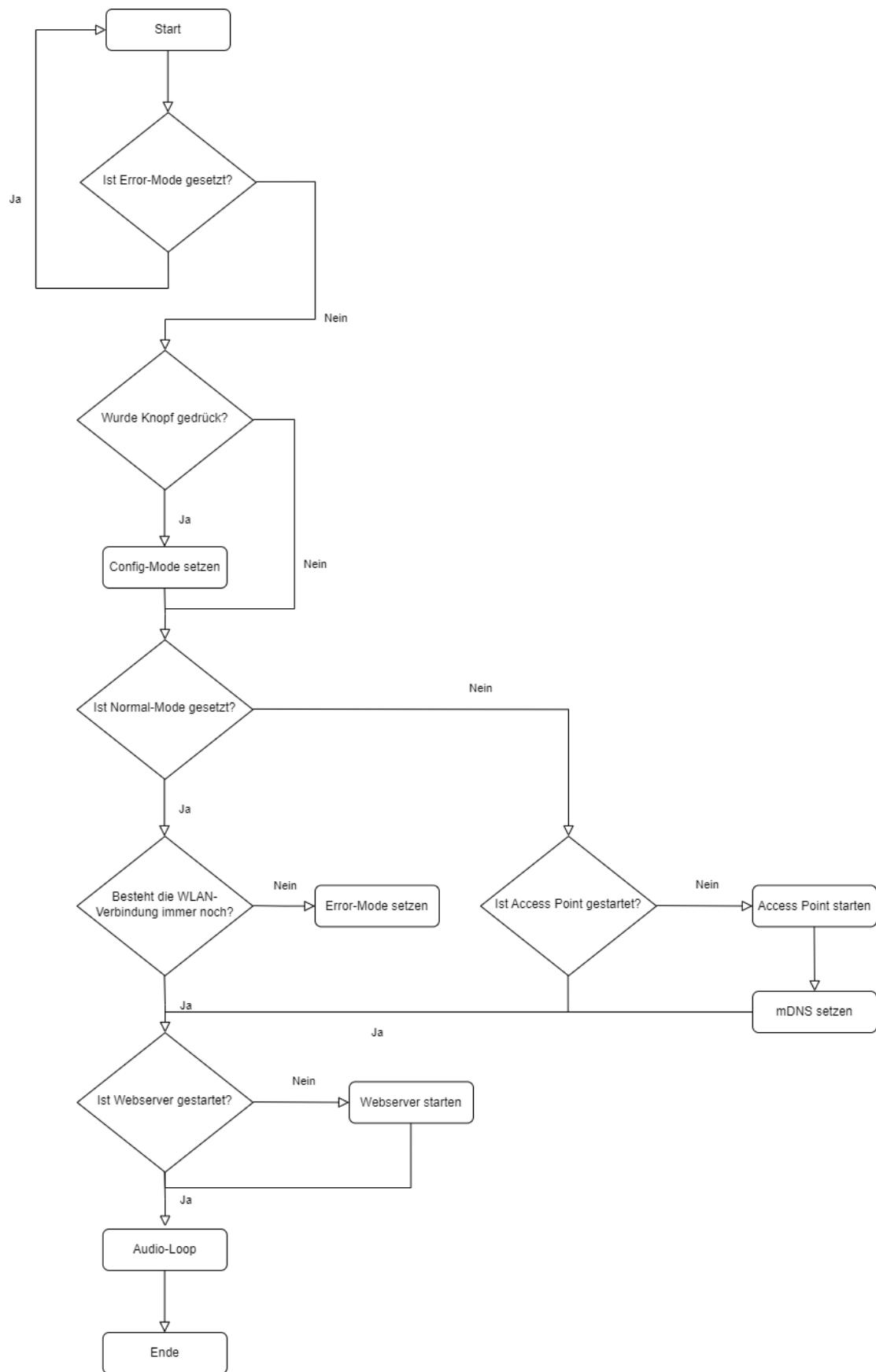


Abbildung 4: UML-Ablaufdiagramm Mikrocontroller Loop

Am Anfang der Loop-Funktion wird abgefragt, ob der Modus auf „ERROR“ gesetzt ist. Ist dies der Fall, wird nichts mehr ausgeführt, das heißt, die Funktion springt wieder an den Startpunkt. Ist der Modus allerdings nicht auf „ERROR“ gesetzt, wird überprüft, ob der Taster gedrückt ist. Ist dieser gedrückt, wird der Modus auf „CONFIG“ gesetzt. Anschließend wird überprüft, ob der Modus auf „NORMAL“ gesetzt ist. Ist dies der Fall, wird überprüft, ob immer noch eine Verbindung mit dem WLAN besteht. Dies geschieht mithilfe der Klasse „NetworkManager“. Wenn eine WLAN-Verbindung besteht, wird mithilfe der Klasse „ServerManager“ überprüft, ob der Webserver läuft. Wenn nicht, wird dieser gestartet. Ist der Modus allerdings nicht auf „NORMAL“ gesetzt, also auf „CONFIG“, so wird zuerst überprüft, ob der Access Point - Modus gestartet ist, da dieser für die Verbindung zwischen Client (Smartphone) und Server (MAA) benötigt wird. Wenn nicht, wird dieser schließlich gestartet und das mDNS-Protokoll, welches bereits im Kapitel 2.5.1 beschrieben wurde, wird konfiguriert. Diese Prozesse werden wieder mithilfe der „NetworkManager“-Klasse durchgeführt. Anschließend wird ebenfalls der Zustand des Webservers überprüft. Am Schluss der Loop-Funktion wird noch die Loop-Funktion der Klasse „AudioManager“ ausgeführt. Diese regelt das Audiostreaming.

3.1.3 Erweiterungsmöglichkeiten

In Zukunft sind noch weitere Funktionalitäten, welche außerhalb der Diplomarbeit implementiert werden, für die Software geplant. Im Hinblick auf die Sicherheit wäre es durchaus sinnvoll, den Webserver des Mikrocontrollers auf HTTPS umzustellen. Somit wäre eine verschlüsselte Kommunikation mit diesem möglich. Zusätzlich wäre es auch denkbar eine Funktion zu implementieren, die es ermöglicht mehrere Adapter zu Gruppen zu verbinden. In diesen Gruppen soll dann jeweils der gleiche Audiostream synchron empfangen und ausgegeben werden. Die Synchronisierung der Adapter bringt aber eine hohe Komplexität mit sich, da man per WLAN-Verbindung mit sehr großen Latenzen zu rechnen hat. Ein Ansatz wäre zum Beispiel, dass jeder Mikrocontroller in einem bestimmten Intervall die Zeit von einem NTP (Network Time Protocol) - Server abfragt und somit auf allen Mikrocontrollern die Zeit annähernd gleich ist. Dann könnte man die Buffer, welche vom Audiostream erhalten werden, mit einem Zeitstempel versehen und sie beim Erreichen dieser Zeit abspielen. Das Hauptproblem dieser Lösung wäre, dass die Zeitdifferenz zwischen den Mikrocontrollern vermutlich zu groß wäre und somit eine hörbare Latenz entsteht.

3.2 Entwicklung Smartphone-App

In diesem Kapitel wird der Übergang der Planung in die Entwicklung der Smartphone-App beschrieben.

3.2.1 Struktur

In diesem Kapitel wird der Aufbau bzw. die Struktur der Smartphone-App genauer beschrieben. Die folgende Grafik stellt dabei die grobe Struktur der App dar:

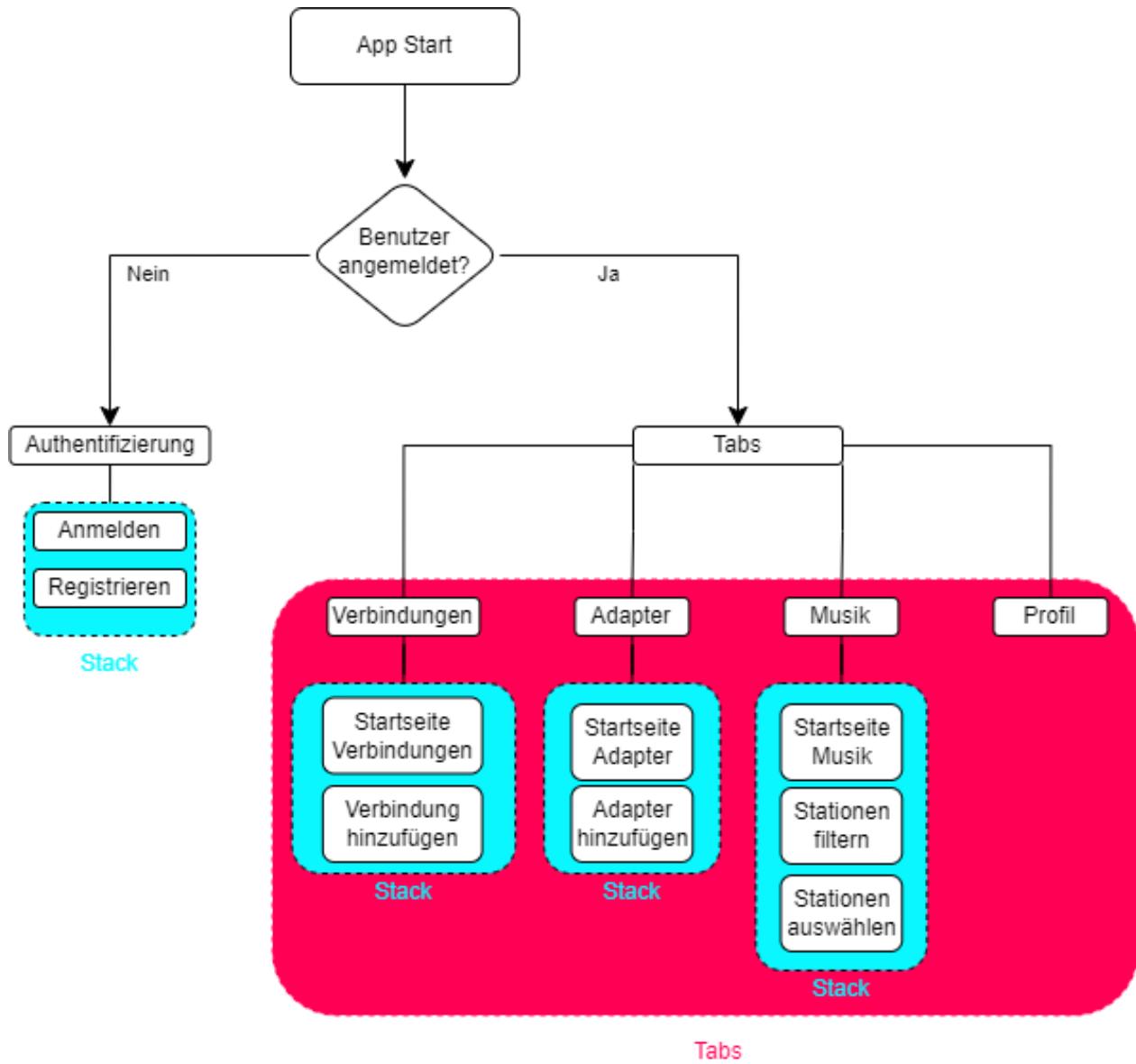


Abbildung 5: Struktur Smartphone-App

Navigation

Für die Navigation zwischen den Seiten der App wurde einerseits die Tab-Navigation und andererseits die Stack-Navigation von „Expo Router“ verwendet. Dabei wird grob zwischen dem Ordner „auth“ und dem Ordner „tabs“ unterschieden. Der Ordner „auth“ enthält die Seiten, welche für die Benutzerauthentifizierung, also Login und Registrierung, notwendig sind. Der Ordner „tabs“ hingegen enthält alle Seiten, welche für angemeldete Benutzer sichtbar sind und somit die eigentliche App darstellen. Die App ist grundlegend in die vier Tabs „Verbindungen“, „Adapter“, „Musik“ und „Profil“ gegliedert. Zwischen diesen Tabs kann mithilfe der Tab-Bar, welche unten in der App sichtbar ist, gewechselt werden. In den einzelnen Tabs befinden sich jeweils mehrere Seiten. Zwischen diesen Seiten kann mittels Stack-Navigator gewechselt werden. Der Stack-Navigator arbeitet dabei nach dem sogenannten „Last In - First Out“- Prinzip (LIFO). Das heißt, dass die Seite, welche als letztes zum Stack (Stapel) hinzugefügt wurde, als Erstes wieder von diesem entfernt wird. Somit wird die Navigations-Historie festgehalten und der/die Benutzer/in kann jederzeit wieder auf die vorherigen Seiten zurückgehen. (vgl. *URLPI28* 2025)

3.2.2 Design

Beim Style des User-Interface der App wurde auf Übersichtlichkeit und wenig Komplexität gesetzt. Der/Die Benutzer/in soll sich in der App schnellstmöglich zurechtfinden. Durch die gute Benutzbarkeit werden allerdings gleichzeitig die Konfigurationsmöglichkeiten eingeschränkt, was die Flexibilität der App senkt. Bei der Auswahl der Farben, welche in der App sichtbar sind, wurde darauf geachtet, dass diese die App modern und schlicht erscheinen lassen. Für die App wurde ein Darkmode implementiert, weil dieser in den meisten modernen Applikationen verwendet wird und somit die App moderner aussehen lässt.

Farben

Um den erwähnten Darkmode umzusetzen, wurden in der App primär dunkle Farben verwendet. Im Kontrast zu diesen dunklen Farben wurden knallige, satte Farben im Turquoise-Ton für die Steuerlemente, wie z.B. Buttons, verwendet. Im Folgenden ist eine Tabelle mit den verwendeten Farben und deren Farbcodes im Hex-Format abgebildet:

Bezeichnung	Farbcodes (hex)	Farbe
grey	#2B2C28	
lightTurquoise	#7DE2D1	
white	#FFFAFB	
red	#D90B0B	
lightGrey	#3E403A	

Tabelle 3: App-Farben

3.2.3 Komponenten

Wie bereits im Kapitel 2.6.3 erwähnt, bietet React Native die Möglichkeit, eigene Komponenten zu erstellen. Als Grundlage dafür dienen die Standardkomponenten von React Native. Der Vorteil bei der Verwendung von Komponenten ist, dass diese wiederverwendbar sind und ein eigenes Verhalten aufweisen. Man muss also den Code für eine Komponente nur einmal schreiben und kann diese Komponente dann in der gesamten App verwenden. Im Folgenden werden die wichtigsten Komponenten, welche selbst erstellt wurden, aufgezählt und kurz beschrieben:

- **AdapterItem**

Zeigt die Daten eines Adapters an. Wenn dieser nicht verbunden ist, wird dies mit einem grauen Hintergrund und einem Symbol einer durchgestrichenen Wolke signalisiert.



Abbildung 6: AdapterItem-Komponente

- **StationItem**

Zeigt die Daten einer Internetradiostation an.

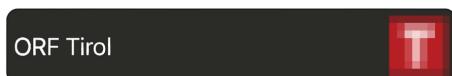


Abbildung 7: StationItem-Komponente

- **ConnectionItem**

Stellt eine Verbindung zwischen Radiostation und Adapter dar. Dabei werden die StationItem-Komponente und die AdapterItem-Komponente verwendet. Die Komponente ermöglicht es auch, den aktuellen Stream des Adapters zu stoppen bzw. die Lautstärke des Streams zu regeln.

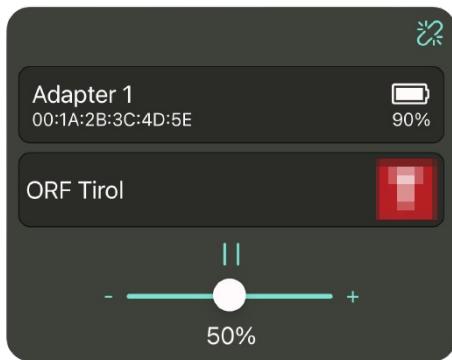


Abbildung 8: ConnectionItem-Komponente

- **AdapterList**

Erstellt eine Liste aus mehreren AdapterItem-Komponenten. Realisiert außerdem die Möglichkeit, einen Adapter zu löschen.

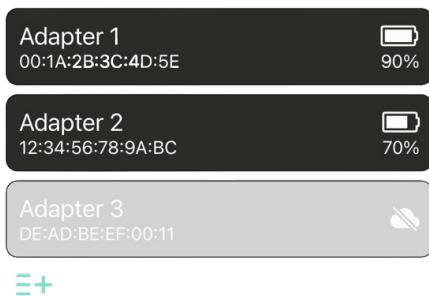


Abbildung 9: AdapterList-Komponente

- **FavouriteStationList**

Stellt eine Liste, gefüllt mit allen Stationen, welche sich in der Favoritenliste des Benutzers befinden, mithilfe von StationItem-Komponenten dar.

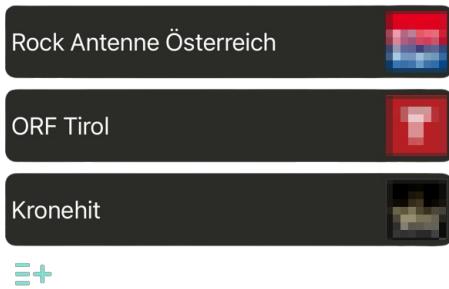


Abbildung 10: FavouriteStationList-Komponente

- **ConnectionList**

Erstellt eine Liste, in der sich alle aktuellen Verbindungen zwischen Adapters und Radiostationen befinden, dar. Die einzelnen Verbindungen werden mithilfe von ConnectionItem-Komponenten dargestellt.

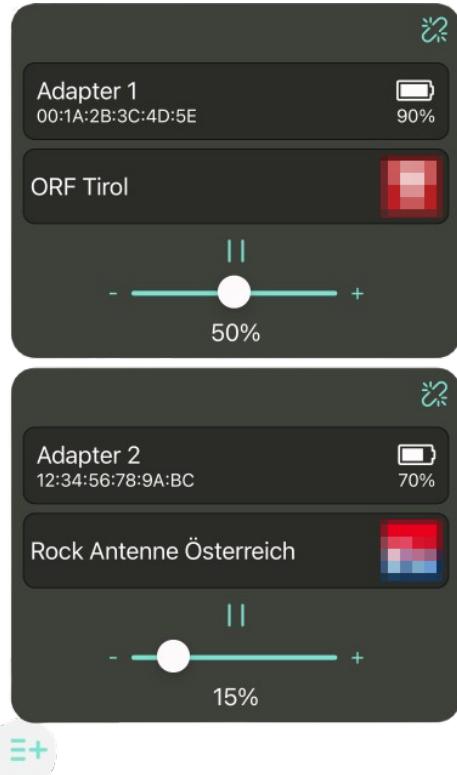


Abbildung 11: ConnectionList-Komponente

- **ErrorScreen**

Stellt eine Ansicht bereit, die signalisiert, dass irgendwo ein Fehler aufgetreten ist. Als Parameter wird dieser Komponente der Fehlertext und die Aktion, welche der/die Benutzer/in als Reaktion auf den Fehler ausführen kann, mitgegeben.

- **LoadingScreen**

Stellt eine Lade-Animation dar und einen Text, der als Parameter übergeben wird. Diese Ansicht soll signalisieren, dass Daten in der App geladen werden und der/die Benutzer/in deshalb warten muss.

3.2.4 Seiten

Wie bereits im Kapitel Struktur erwähnt, besteht die App aus mehreren Seiten, zwischen diesen mit verschiedenen Navigationsarten gewechselt werden kann. Eine Seite setzt sich dabei aus den Grundkomponenten von React Native und den selbst erstellten Komponenten, welche im Kapitel 3.2.3 genauer beschrieben wurden, zusammen. Dabei ist die Grundstruktur jeder Seite gleich. Wie in Abbildung 12, am Beispiel der Seite „Stationen filtern“ zu sehen ist, befindet sich oben ein Header, der den Namen der aktuellen Seite anzeigt und unten die Tab-Bar, welche anzeigt, in welchem Tab

man sich aktuell befindet. Dazwischen befindet sich der Inhalt der Seite, welcher sich, wie bereits erwähnt, aus verschiedenen Komponenten zusammensetzt.

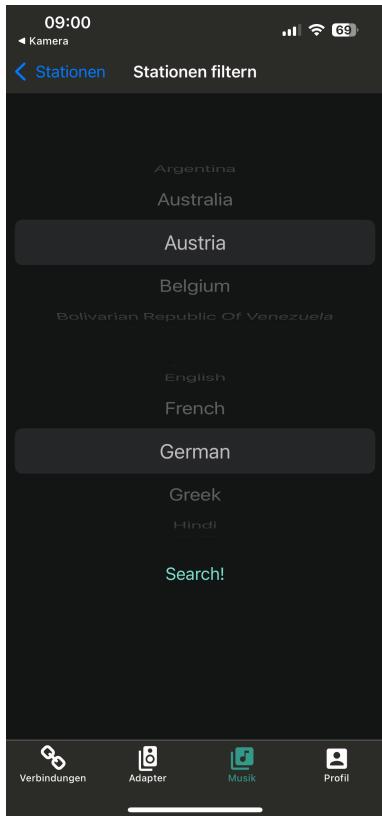


Abbildung 12: „Stationen filtern“ - Seite

Im Folgenden werden die einzelnen Seiten der App aufgezählt und kurz beschrieben.

- **Anmelden**

Ermöglicht es einem Benutzer / einer Benutzerin, sich in der App anzumelden.

- **Registrieren**

Ermöglicht es einem Benutzer / einer Benutzerin, sich neu zu registrieren.

- **Startseite Verbindungen**

Zeigt die aktuellen Verbindungen mithilfe der ConnectionList-Komponente an. Auf dieser Seite ist es möglich, bestehende Verbindungen zu verwalten und zu trennen.

- **Verbindung hinzufügen**

Ermöglicht es, eine neue Verbindung zwischen einem Adapter und einer Internetradiostation anzulegen.

- **Startseite Adapter**

Zeigt alle gespeicherten Adapter des Benutzers / der Benutzerin an. Hier ist es außerdem möglich, Adapter zu löschen.

- **Adapter hinzufügen**

Ermöglicht es, einen neuen Adapter hinzuzufügen bzw. diesen zu konfigurieren.

- **Stationen filtern**

Ermöglicht es, das Land bzw. die Sprache auszuwählen, nach denen die Stationen, welche in weiterer Folge auf der Seite „Stationen auswählen“ angezeigt werden, gefiltert werden.

- **Stationen auswählen**

Ermöglicht es, mehrere Stationen auszuwählen, welche dann in die Favoritenliste des Benutzers / der Benutzerin übernommen werden. Es werden nur Stationen angezeigt, die den Filteroptionen, welche auf der Seite „Stationen filtern“ festgelegt wurden, entsprechen.

- **Profil**

Hier werden Informationen zum Profil des Benutzers / der Benutzerin angezeigt. Auf dieser Seite wird dem/der Benutzer/in außerdem ermöglicht, sich abzumelden bzw. sein/ihr Passwort zurückzusetzen.

3.2.5 APIs

In diesem Kapitel werden die APIs (Application Programming Interfaces) beschrieben, mit welchen die Smartphone-App Daten austauscht. Eine API ist dabei, wie bereits kurz im Kapitel 2.3 erläutert, eine einheitliche Schnittstelle zum Austausch von Daten. Folgende APIs wurden in Verbindung mit der Smartphone-App verwendet:

RadioBrowser-API

Die RadioBrowser-API ist eine Open-Source-API welche eine sehr umfangreiche Sammlung von Internet-Radios bereitstellt. Dabei ist es einerseits möglich, mithilfe dieser API neue Radiostationen zu der Sammlung hinzuzufügen und andererseits, bestehenden Radiostationen nach bestimmten Suchkriterien zu suchen und die Daten derer abzufragen. (vgl. *URLPI21* 2025)

In der Smartphone-App wird die RadioBrowser-API verwendet, um nach Radiosendern zu suchen und in weiterer Folge die Stream-URLs dieser an den MAA zu senden. Der MAA ruft in weiterer Folge den jeweiligen Audiostream auf und empfängt diesen.

Firebase

Firebase ist eine Entwicklungsplattform von Google, welche verschiedene Dienste, die hilfreich für die Entwicklung von Apps bzw. Web-Apps sind, bereitstellt. Firebase stellt dabei ein Backend für die Apps dar. Damit werden Prozesse, wie z.B. Benutzerauthentifizierung, Speicherung von Benutzern und Speicherung von Daten nicht lokal auf dem Gerät, sondern in der Cloud ausgeführt (vgl.

URLPI22 2025)

In der Smartphone-App wurde Firebase zur Benutzerverwaltung verwendet. Dabei wurde einerseits der „Firebase Authentication“-Dienst für die Benutzerauthentifizierung und andererseits der „Cloud Firestore“-Dienst für die Speicherung von Benutzerdaten verwendet. Firebase wurde für die Entwicklung der Smartphone-App gewählt, da dadurch der Ressourcenverbrauch der App minimiert wird und der/die Benutzer/in von jedem Gerät aus Zugriff auf seine/ihre Daten hat und somit nicht auf ein bestimmtes Gerät angewiesen ist. Außerdem müssen Dienste, wie z.B. die Benutzerauthentifizierung, nicht von Grund auf entwickelt werden.

3.2.6 Zustandsverwaltung

Um Zustände, welche in mehreren Teilen der App benötigt werden, global bereitzustellen, wurde die „Context“-Funktionalität von React Native verwendet. Context ermöglicht es, Zustände global in der App bereitzustellen und auf diese in den einzelnen Komponenten der App zuzugreifen. Dabei werden bei der Änderung eines Zustands die betroffenen Komponenten automatisch aktualisiert. Um den Context bereitzustellen, wird ein sogenannter Context-Provider verwendet. Alle Komponenten, welche in den Context-Provider eingebettet sind, können auf den Zustand von diesem zugreifen. (vgl. *URLPI23 2025*)

In der Smartphone-App wurden folgende ContextProvider definiert:

- **UserContext**

Stellt das Benutzerobjekt bereit. Dieses beinhaltet wichtige Benutzerdaten, welche an mehreren Stellen der App verwendet werden. Das Benutzerobjekt wird von Firebase Authentication abgefragt.

- **AdapterContext**

Stellt eine Liste von hinzugefügten Adapters bereit. Diese Liste wird von Firebase Firestore abgefragt. Falls sich die in Firestore gespeicherte Liste ändert, wird die Liste, welche der AdapterContext bereitstellt, automatisch aktualisiert.

- **StationContext**

Stellt eine Liste von hinzugefügten Radiostationen bereit. Diese Liste wird ebenfalls von Firebase Firestore abgefragt. Die hier gespeicherte Liste wird ebenfalls automatisch aktualisiert, falls sich die Liste in Firestore ändert.

3.2.7 Benutzerverwaltung

Die Verwaltung von Benutzerdaten wurde mithilfe der im Kapitel 3.2.5 beschriebenen Firebase-API in Kombination mit dem UserContext, welcher im Kapitel 3.6.2 beschrieben wurde, realisiert. Durch das im UserContext gespeicherte Benutzerobjekt sind die Benutzerdaten in der gesamten App global verfügbar. Das Objekt wird beim Start der App, falls vorhanden, aus dem lokalen Gerätespeicher ausgelesen oder mittels Login von Firebase abgefragt. Der Status des Benutzerobjektes wird mithilfe

eines Event-Handlers verfolgt. Das heißtt, wenn sich der Status des Benutzerobjektes, z.B. durch einen Logout, ändert, wird dies direkt registriert.

3.2.8 Erweiterungsmöglichkeiten

In Zukunft wäre es noch denkbar, außerhalb der Diplomarbeit weitere Funktionen in die Smartphone-App zu implementieren. Dabei wäre es durchaus sinnvoll, mehrere Audio-Streaming-Quellen, wie zum Beispiel Musikstreamingdienste (Spotify, Amazon Music etc.) einzubinden. Außerdem könnte man, wie bereits in vorherigen Kapiteln erwähnt, das Erstellen von Adapter-Gruppen realisieren, so dass mehrere Adapter zu einer Gruppe zusammengefügt werden können und diese dann den gleichen Stream, möglichst synchron ausgeben. Um dies umzusetzen, müsste allerdings auch die Software des Adapters weiterentwickelt werden.

3.3 Design Adaptergehäuse

Das Adaptergehäuse trägt einen wesentlichen Teil zur Sicherheit des Endverbrauchers sowie zur optimalen Funktionalität der Komponenten bei. Zudem soll es möglichst kompakt sein.

3.3.1 Grundsätzlicher Aufbau

Die Grundlage für den Prototyp bildet ein Kasten mit Deckel.

Das Gehäuse wurde mit frei schwebenden, jedoch an den Wänden befestigten Stützen ausgestattet, um den Mikrocontroller fest montieren zu können. Der Prototyp wurde zudem mit kleinen Zylindern auf den Stützen ausgestattet, um den Mikrocontroller mit seinen bereits vorhandenen Aussparungen darauf platzieren zu können. Der Digital-/Analogwandler und der Akku-Laderegler finden auch auf solchen Stützen ihren Platz. Der Gedanke dahinter war, den Akku unter den Bauteilen zu platzieren. Mehr dazu im Teil "Wärmeableitung". Zudem wurden in einer Wand des Gehäuses Aussparungen für die Buchsen platziert. Die Aussparungen für die RGB-LED und den Taster wurden im Deckel platziert. Eine Art Abdeckung für den Taster selbst wird auf diesen geklebt, um ein gleichbleibendes Erscheinungsbild des Gehäuses zu behalten. Der Deckel, der von oben auf das Gehäuse gedrückt wird, schließt dieses. Im Deckel ist zusätzlich ein Belüftungsgitter eingelassen.

3.3.2 Wärmeableitung

Wärmeableitung ist wichtig, da Mikrocontroller, wie alle anderen Prozessoren auch, bei intensivem Betrieb Hitze entwickeln. Laut einer eigens durchgeföhrten Messung wird der in diesem Fall eingesetzte ESP32 meistens nur rund 38°C warm. Bei hoher Rechenleistung sind jedoch auch höhere Temperaturen möglich. Der ESP32 hat laut Hersteller eine mögliche Betriebstemperatur (Umgebungstemperatur) von -40°C bis +125°C. Damit die entstandene Wärme gut ableiten kann und keine der Komponenten stark beeinflusst (vor allem den Akku, da dieser bei hohen Temperaturen eine Explosionsgefahr darstellt), wird der Mikrocontroller im Gehäuse oben, also auf Stützen, angebracht. Die aufsteigende Wärme kann somit nach oben durch das dafür vorgesehene Gitter

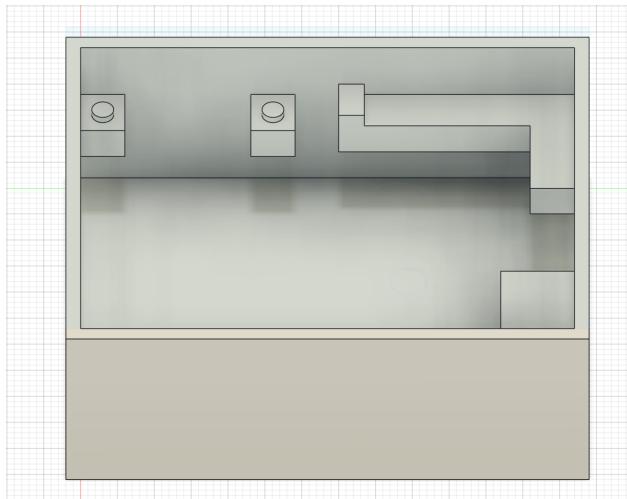


Abbildung 13: Stützen für die Komponenten

entweichen und staut sich somit nicht im Inneren des Gehäuses. Der Akku liegt dementsprechend unter dem Mikrocontroller und allen anderen Komponenten und wird durch die abstrahlende Hitze dementsprechend nur etwas wärmer als Raumtemperatur. (vgl. *URLNL06* 2025)

3.3.3 Virtuelles 3D-Design

Um ein 3D-Modell des Prototyps zu zeichnen, wurde AUTODESK Fusion verwendet. Der Anbieter beschreibt seine Software folgendermaßen: „Autodesk Fusion verbindet Ihren gesamten Fertigungsprozess durch die Integration von CAD, CAM, CAE und PCB in einer einzigen Lösung, mit der Sie Ihre Ideen verwirklichen und praktisch alles fertigen können.“ (*URLNL12* 2025)

Wenn man schon früh beachtet, dass die Prototypen mit einem 3D-Drucker gefertigt werden, kann man schon das 3D-Design für eine gute Druckbarkeit auf 3D-Drucker anpassen. Damit ist hauptsächlich gemeint, überhängende Drucke, komplizierte Stützstrukturen oder ähnliches zu vermeiden.

Basis

Die Basis des Adapters bildet ein 71x58x27mm großer Kasten mit 2 mm Wanddicke. Aus Erfahrung kann man sagen, dass diese Dicke bei 3D-Drucken stabil ist, während sie jedoch nicht zu klobig wirkt.

Die Stützen des Mikrocontrollers beginnen auf 11 mm Höhe und sind auf einer Längenseite 6x6x5mm und auf der anderen 6,9x6x5mm groß. Sie sind jeweils mit einer oder zwei Seiten an der Wand der Basis und somit überhängend. Dies wird im Teil "Drucken des Prototyps/Stützstruktur" noch wichtig. Auf den Stützen befinden sich jeweils Zylinder, die genau auf die Aussparungen des Mikrocontrollers vermessen wurden. Die Zylinder haben einen Durchmesser von exakt 3 mm. Der Abstand der Mittelpunkte dieser Zylinder war bei unserem Modell 47,10 mm in der Länge und 23,10 mm in der Breite. Auf einer Seite befindet sich zwischen Wand und Zylinder etwas mehr Platz, da der USB-C-Port des Mikrocontrollers etwas über diesen herausragt. Deswegen auch der zuletzt erwähn-

te Versatz der Stütze von 0,9 mm.

Die Breite der Basis ist also genau auf die Länge des von uns benutzten Mikrocontroller zugeschnitten.

In den gegenüberliegenden Ecken der Basis befinden sich der Digital-/Analogwandler und der Laderegler. Die Maße des Digital-/Analogwandlers sind 31,8x17,2 mm. Die Maße des Laderegler sind 28x17,45 mm. Beide liegen, wie der Mikrocontroller auch, auf überhängenden, 5 mm hohen Stützen auf. Aufgrund der USB-C Buchse wird der Halt des Laderegler noch von einer 3,5 mm breiten 2 mm-Erhöhung am Ende der Stütze verstärkt. Die USB-C Buchse wurde passend zum Laderegler und der Norm entsprechend (8,34x2,56 mm Größe) eingelassen. Der Digital-/Analogwandler wird aufgrund der 3,5 mm Klinkenbuchse von einer herab stehenden Wand gestützt, die im Teil "Deckel" genauer beschrieben wird. Die Aussparung der AUX-Buchse wurde auch passend für den Digital-/Analogwandler platziert und hat einen Durchmesser von 6 mm.

(vgl. *URLNL17* 2025)

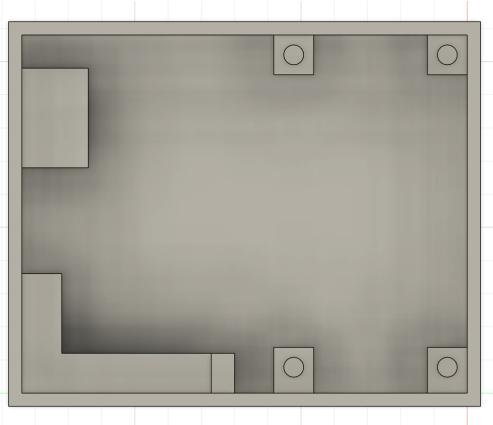


Abbildung 14: Draufsicht der Basis



Abbildung 15: Basis mit den Komponenten (während des Zusammensetzens)

Deckel

Der Deckel des Adapters ist grundsätzlich, wie die Wände der Basis, 71x58x2 mm groß. Dieser hat jedoch eine zentrierte 67x54x1 mm große Stufe. Mit dieser Stufe lässt sich der Deckel kleberlos auf den Adapter setzen und hält aufgrund der Eigenschaften des 3D-Drucks auch, zumindest für den Prototypen, fest genug. Wie schon erwähnt, wird der DAC durch eine herabstehende Wand zusätzlich gestützt. Die Maße dieser Wand sind 32x2x10 mm. Es würde keinen Sinn machen, die Wand wie beim Laderegler von der Außenwand aus überhängend zu machen, da der DAC länger als der Laderegler ist und die Buchse sich eher mittig in der entsprechenden Außenwand der Basis befindet.

An der freien Seite der Stützwand (nicht die des DAC) befindet sich ein runder Schacht für die RGB-LED mit 5 mm Durchmesser und eine Aussparung für den Aufsatz des Tasters mit 10,10 mm Durchmesser. Der Taster mit den Grundmaßen 6x6 mm wird durch eine Art U-Form aus Wänden im Gehäuse gehalten. Eine Wand davon bildet die gerade eben beschriebene Stützwand. Die untere Wand, auf der der Taster aufliegt, hat zudem zwei auf den Taster angepasste, 1 mm große Aussparungen für die zwei Pole des Tasters.

Wenn man in der Draufsicht auf den Deckel schaut, ist dort, wo sich der Prozessor selbst befindet, ein Gitter in den Deckel eingelassen. Dieses Gitter hat die Maße 19,9x21,6 mm, was etwas größer als der Prozessor des ESP32 ist. Das Gitter besteht aus diagonalen Streben, die jeweils 1,5 mm breit sowie 1,5 mm weit voneinander entfernt sind. So entsteht eine einfache und stabile Möglichkeit, Wärme durch den Deckel abzuleiten.

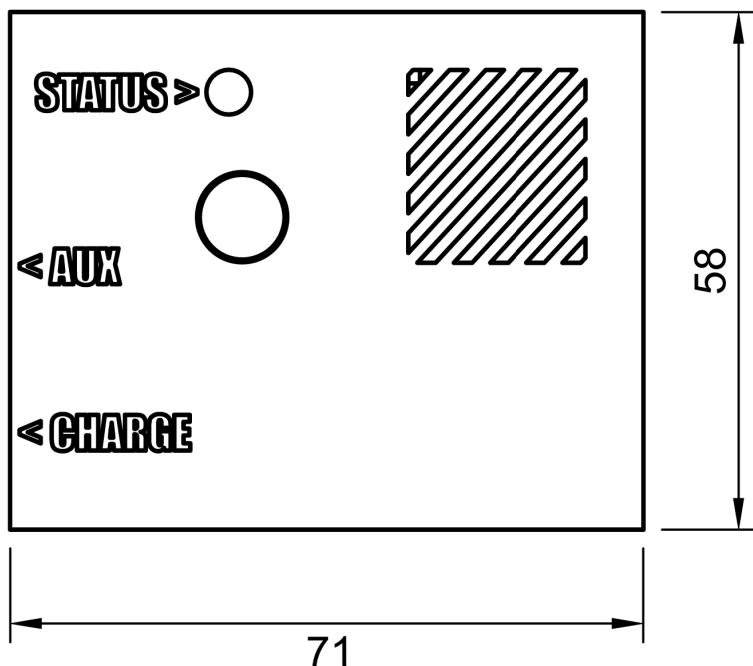


Abbildung 16: Draufsicht des Adaptergehäuses

Tasteraufsatz

Den Tasteraufsatz bildet ein Zylinder mit 9,5 mm Durchmesser, auf dem sich ein weiterer Zylinder mit 5,5 mm Durchmesser und zentrierter 3,5 mm Aussparung für den Taster befindet.

3.4 Fertigung Adaptergehäuse

Das Material unseres Gehäuses wurde auf Kunststoff begrenzt. Für die Fertigung von Kunststoffgehäusen gibt es hauptsächlich diese Möglichkeiten (in diesem Fall kommt mangels Alternativen nur der 3D-Druck in Frage):

Spritzgießen

„Beim Spritzgießen wird der Kunststoff aus einem Plastifiziergerät (erwärmt den Kunststoff auf Schmelztemperatur) in einen Hohlraum (Formwerkzeug) gespritzt, in welchem er erst verdichtet wird und dann erkaltet.“ (*URLNL14* 2025)

Ein Vorteil für dieses Verfahren ist, dass auch komplizierte Formteile voll automatisiert sehr schnell in hohen Stückzahlen produziert werden können. Der große Nachteil sind jedoch die hohen Stückkosten für die Formwerkzeuge. (vgl. ebd.)

3D-Druck

Die zwei gängigsten 3D-Druck-Methoden sind Filament und Resin (Harz). Aufgrund des hohen Aufwands, den ein Resin-Drucker mit sich bringt, wurde für dieses Projekt die Methode mit Filament gewählt.

Beim 3D-Drucken durch Fused Deposition Modeling (Schichtschmelzverfahren) wird Kunststoff in Drahtform (Filament) (die häufigsten Dicken sind 2,85mm (allgemein als 3mm bezeichnet) und 1,75mm wobei die 1,75mm Version weltweit am häufigsten verbreitet ist) (vgl. *URLNL16* 2025) durch beheizte Düsen geleitet und somit geschmolzen. Das nun weiche Filament wird in Schichten auf die Druckplatte aufgetragen und erhärtet kurz darauf. Durch dieses Schichten lassen sich präzise Körper aus Kunststoff bauen. (vgl. *URLNL15* 2025)

3.4.1 Drucken des Prototyps

Die Gehäuse-Prototypen wurden mit einem „PRUSA MK4S“ 3D-Drucker gefertigt. Alle FFF-Drucker von Prusa sind grundsätzlich für 1,75-Filament konfiguriert.

Druckeinstellungen

Die Temperatur der Build Plate lag bei uns standardmäßig auf 60°C. Die Drucktemperatur, also die der Nozzle (Düse) lag bei etwa 185°C.

Das erste Layer wurde mit 0,15mm Dicke gedruckt, die restlichen mit 0,2mm Dicke.

Als Infill-Pattern wurde „Grid“ mit 20% Dichte und 4mm Line-Distance gewählt.

Das Drucken verlief mit den von uns gewählten Einstellungen reibungslos, jedoch an manchen Stellen etwas unsauber. Beispielsweise war das Ergebnis der Aussparung für den Taster im Deckel des

Adapters so ungenau, dass der Durchmesser des Aufsatzes für den Taster um 0,5mm verkleinert werden musste. Sonstige Ungenauigkeiten stellten, zumindest abgesehen von der Optik, kein Problem dar.

Man darf auch nicht vergessen, dass gewisse Drucke (Gehäusedeckel und -basis) Stützstrukturen erfordern. Stützstrukturen sind Konstruktionen, die der Drucker zur Unterstützung stark überhängender oder freischwebender Strukturen druckt. Jedoch sind diese nur temporär. Da das Stützmaterial nicht so fest an der Druckfigur haftet, wie die Teile der Figur selbst, kann es nach dem Druck mit einer herkömmlichen Zange entfernt werden. Diese Stützstrukturen lassen sich in der Slicer-Software genau einstellen und konfigurieren. So kann man unter anderem sicherstellen, dass das Entfernen einfach möglich ist und die Stützstruktur die Figur selbst nicht allzu stark beeinflusst. (vgl. *URLNL16* 2025)

3.5 Zusammensetzen des Prototyps

Die Komponenten werden jeweils einzeln direkt mit dem Mikrocontroller verbunden. Bei diesem Prototyp lässt sich der Mikrocontroller nämlich vorerst als Platine betrachten, welcher zusätzliche Komponenten zugefügt werden.

3.5.1 Schaltplan

Die Verdrahtung zwischen den Komponenten und dem Mikrocontroller ist folgendermaßen gelöst:

Digital-/Analogwandler

- BCK (Wandler) an D26 (Mikrocontroller)
- LCK (Wandler) an D25 (Mikrocontroller)
- DIN (Wandler) an D22 (Mikrocontroller)

Taster

- 1 (Taster) an 3V3 (Mikrocontroller)
- 2 (Taster) an D12 (Mikrocontroller)

RGB-LED

- R (LED) an D15 (Mikrocontroller)
- G (LED) an D2 (Mikrocontroller)
- B (LED) an D4 (Mikrocontroller)
- GND (LED) an GND (Mikrocontroller)

3.5.2 Verdrahten

Bei unserem Prototypen wurden alle Komponenten, der Pinbelegung entsprechend, mit dem Mikrocontroller verbunden. Dabei kamen Drahtkabel, Litzenkabel und Steckkabel (herkömmliche JumperKabel) zur Verwendung. Dies hatte hauptsächlich den Grund, dass zum Beispiel nur mit Drähten nicht alle Verbindungen optimal möglich gewesen wären. Das ist hauptsächlich der Löthaftung an einigen Komponenten geschuldet. Um keine Wackelkontakte oder gar unterbrochene Verbindungen zu riskieren, wurde für jede Verbindung einzeln entschieden, welches der genannten Verfahren sich am besten eignet. So entstanden zum Beispiel Steckverbindungen, gelötete Drahtverbindungen oder Mischungen aus gelötzten Litzen- und Steckverbindungen (jeweils am anderen Ende des Kabels).

3.5.3 Löten

„Das Löten ist das Verbinden von Metallteilen durch eine Metalllegierung (das Lot) unter Einfluss von Wärme/Hitze.“ (*URLNL13* 2025)

Man unterscheidet grundsätzlich zwischen Weich- und Hartlöten. Ausschlaggebend dabei ist die Schmelztemperatur des Lots. So haben Weichlote eine Schmelztemperatur unter 450°C während Hartlote erst ab 450°C bis etwa 1100°C schmelzen. „Das Weichlot wird verwendet, wenn die Verbindung zweier Metalle dicht und leitfähig sein soll und um die mechanische Belastbarkeit keine hohe Anforderung gestellt wird.“ (ebd.)

Da die in diesem Projekt benutzten Bauteile jedoch keine höheren Temperaturen vertragen, war Weichlöten für dieses Projekt die einzige Wahl.

Um einen Lötorgang aus eigener Erfahrung möglichst kurz zu beschreiben:

Man hat beispielsweise zwei Kabel, die man verbinden möchte, und die Verbindung sollte möglichst fest halten und gut leiten. Zuerst müssen die Enden der Kabel abisoliert werden. Es kann helfen, wenn man die Enden schon vor der eigentlichen Verbindung sozusagen etwas „verzinnt“. Nun richtet man die Kabel zueinander so aus, wie sie später fixiert sein sollen. Man fährt mit dem Ende des Lots (umgangssprachlich Lötzinn) an die heiße Spitze des Lötkolbens und schmilzt so etwas Lot auf die Verbindungsstelle (nicht zu viel, sonst erhält man dicke Tropfen; jedoch auch nicht zu wenig, da die Verbindung dann möglicherweise nicht ausreichend hält). Nach dem Abkühlen macht es Sinn, die Lötverbindung durch leichtes Rütteln oder ziehen zu überprüfen. Löten ist letztendlich aber ein Handwerk, das für saubere Ergebnisse Geduld und Übung voraussetzt. (vgl. ebd.)

3.5.4 Kleben

Damit alle Komponenten und Teile sicher im Gehäuse sitzen und nichts klappert oder sich gar bewegt, worunter die Lötverbindung leiden würde, müssen gewisse Komponenten angeklebt werden. Vor allem bei den Buchsen ist eine feste Verbindung wichtig, da diese bei jedem Ein- und Ausstecken großer Belastung ausgesetzt sind. Somit wurden der Laderegler und der Digital-/Analogwandler an den dafür gedruckten Stützen angeklebt. Zudem wurde der gedruckte Aufsatz in Gehäuseoptik für den Taster auf diesem befestigt. Für alle Klebverbindungen im Adapter wurden entweder herkömmliches Cyanacrylat (Superkleber) oder Schmelzklebstoff (Heißkleber) verwendet.

4 Testen und Fehlerbehebung

4.1 Testen des Gesamtsystems

4.1.1 Testen des Prototyps

Es gibt unzählbar viele Möglichkeiten, einen Prototyp auf Herz und Nieren zu testen. Diese Diplomarbeit beschränkt sich jedoch auf einige wesentliche Aspekte wie Verarbeitung, Funktion und Usability.

Verarbeitung

Positives:

Die äußere Verarbeitung des MAA ist für einen Prototyp sehr gut ausgefallen. Fertig zusammengesetzt wirkt das Gerät stabil und wertig. Es hat etwas Gewicht und nichts klappert, wenn man es schüttelt. Die Buchsen halten der für Buchsen gemäßen Belastung stand. Beide der Buchsen lassen sich problemlos benutzen, beim Ein- und Ausstecken gibt es keine Probleme. Der Taster des MAA ist, zumindest unseren Erwartungen nach, besonders gut ausgefallen, er hat ein angenehmes Klickfeeling und lässt sich reibungslos betätigen. Die Statusanzeige funktioniert so, wie sie soll.

Negatives:

Man sieht dem Gehäuse bei guter Beleuchtung eindeutig an, dass es 3D-gedruckt wurde. Manche Oberflächen wirken eher geriffelt als flach. Diese Oberflächeneigenschaften sind völlig normal für 3D-Drucker und beeinflussen die Funktionsweise des Gesamtsystems kaum. Bei einem tatsächlichen Vertrieb des Produkts müsste man sich jedoch noch einmal genau über Herstellungsmöglichkeiten für kleinere Gehäuse informieren oder das Gehäuse, wie auch bei der Auswahl des Fertigungsverfahrens schon kurz diskutiert, einfach Spritzgießen lassen. Ein Spritzguss hat bessere Oberflächeneigenschaften als ein 3D-Druck. (vgl. *URLNL18* 2020)

Funktion

Der MAA funktioniert und erfüllt seine Funktionen ohne Fehler oder Komplikationen.

Usability

Die Anwendung des Adapter ist grundsätzlich angenehm. Es gilt jedoch zu bedenken, dass uns sowohl Software als auch Hardware schon bekannt sind. Wie sich die Usability verändert, wenn der Adapter von jemandem benutzt wird, der ihn noch nie gesehen hat, kann man nur schlecht sagen. Dafür wären mehrere Produkttests nötig. Falls ein Vertrieb des Produkts in Betracht gezogen werden würde, wäre dies auf jeden Fall ein wichtiger Punkt.

4.2 Auftretende Fehler beheben

Es handelt sich hierbei nicht direkt um einen Fehler, jedoch ist uns wichtig, dass auch die Optik des Gehäuses so gut wie möglich ausfällt. Deshalb wurde viel mit verschiedenen Tricks und Methoden

gearbeitet, die 3D-Drucke wertiger ausfallen zu lassen.

So war anfangs beispielsweise die Druckgenauigkeit in Millimeter nicht auf dem kleinst-möglichen Wert, was für Testdrucks aufgrund der wesentlich kürzeren Druckzeit jedoch völlig in Ordnung ist. Wenn alles passt, macht es aber definitiv Sinn, eine lange Druckdauer in Kauf zu nehmen, um das für den vorliegenden 3D-Drucker genaueste Ergebnis zu erhalten.

Eine Herausforderung war die Beschriftung des Deckels. Diese war beim ersten Testdruck teilweise unlesbar ungenau. Dass der „Boden“ der Schrift aufgrund des druckabhängigen Überhanges (die größte Fläche, also die Oberseite des Deckels muss praktisch auf der Druckplatte sein) nicht glatt wird, war uns klar. Jedoch hatte dieses Erscheinungsbild augenscheinlich nichts mit dem Überhang zu tun. Bei genauerem Betrachten fiel auf, dass der erste Layer fast wie gepatzt aussah. Ein weiterer Versuch mit simpler Schriftart, und bei dem lediglich die Druckgenauigkeit des ersten Layers von 0.15 auf 0.1, die der restlichen Layer von 0.2 auf 0.15 kalibriert wurde, zeigte jedoch schon für einen 3D-Druck schöne Ergebnisse.

Eine weitere Möglichkeit zum Glätten der schon zuvor beschriebenen rifflartigen Oberflächen ist sogenanntes Ironing. „Ironing“ beschreibt die einstellbare Funktion in einiger Slicer-Software. Ironing bietet dir die Möglichkeit, die letzte 3D-Druckschicht mit der Nozzle zu „bügeln“. Die Nozzle deines 3D-Druckers bewegt sich ohne dabei zu extrudieren nochmals über die zuletzt gefertigte Schicht. Durch diesen Vorgang glättet sie die Objektoberfläche. Du bekommst somit eine verbesserte und glatte letzte Schicht..„ (*URLNL19 2025*)

Beim PRUSA MK4S war diese Funktion sehr zufriedenstellend. Oberflächen werden, ohne den gesamten Druck zu beeinflussen, glatter.

5 Schluss

5.1 Konklusion Hardware

Ziel des Hardware-Teils in dieser Diplomarbeit war es, passende Komponenten zu finden und diese in einem selbst designten Gehäuse zu einem Prototyp des MAA zusammenzusetzen. Dies wurde, wie geplant, umgesetzt. Dabei kamen grundlegende Methoden wie 3D-Design, 3D-Druck, Löten und weitere zum Einsatz. Grundsätzlich lief alles wie geplant, es ergaben sich jedoch öfters größere Herausforderungen dort, wo man eigentlich keine erwartete. Für diese fand sich aber immer eine passende Lösung.

5.2 Konklusion Software

Abschließend kann man sagen, dass das Ziel, eine funktionierende Software für den Adapter sowie eine funktionierende Smartphone-App im Zuge der Diplomarbeit zu entwickeln, erfüllt wurde. Das heißt aber nicht, dass der Code dieser beiden Applikationen fertiggestellt ist. Wie bereits beschrieben, können in Zukunft noch zahlreiche neue Features für die Applikationen entwickelt werden, welche ansonsten den Rahmen der Diplomarbeit gesprengt hätten.

Literaturverzeichnis

- URLNL01 (2025). de. URL: <https://www.monacor.de/magazin/audio-pege> (besucht am 02.02.2025).
- URLNL02 (2025). de. URL: <https://www.lenovo.com/at/de/glossary/line-out/> (besucht am 02.02.2025).
- URLNL03 (2025). URL: <https://www.axis.com/dam/public/ad/35/af/einf%C3%BChrung-in-das-thema-audio-akustik--lautsprecher-und-audio-terminologie-de-DE-191125.pdf> (besucht am 02.02.2025).
- URLNL04 (2025). de-DE. URL: <https://www.baumannmusic.com/de/2012/sampleratehz-und-khz-aufloesung-bit-und-bitrate-kbits/> (besucht am 02.02.2025).
- URLNL05 (2025). URL: https://www.elektronik-kompendium.de/sites/praxis/bauteil_rgbl-ed.htm (besucht am 02.02.2025).
- URLNL06 (2025). URL: <https://www.espressif.com/en/products/socs/esp32> (besucht am 02.02.2025).
- URLNL07 (Apr. 2023). de. Section: ESP32. URL: <https://digitalewelt.at/esp32-grundlagen/> (besucht am 02.02.2025).
- URLNL08 (2025). URL: <https://www.xplore-dna.net/mod/page/view.php?id=2741&forceview=1> (besucht am 02.02.2025).
- URLNL09 (2025). URL: <https://www.ti.com/product/de-de/PCM5102A> (besucht am 02.02.2025).
- URLNL10 (2025). URL: <https://www.chemie.de/lexikon/Lithium-Ionen-Akkumulator.html> (besucht am 02.02.2025).
- URLNL11 (2025). de-DE. URL: <https://www.denios.de/beratung-planung/denios-magazin/gefahren-im-umgang-mit-lithium-ionen-akkus> (besucht am 02.02.2025).
- URLNL12 (2025). de-DE. URL: <https://www.autodesk.com/de/products/fusion-360/overview> (besucht am 02.02.2025).
- URLNL13 (2025). URL: <https://www.elektronik-kompendium.de/sites/grd/0705261.htm> (besucht am 02.02.2025).
- URLNL14 (2025). URL: <https://www.chemie.de/lexikon/Kunststoffverarbeitung.html> (besucht am 02.02.2025).
- URLNL15 (2025). URL: <https://www.printer-care.de/de/drucker-ratgeber/wie-funktioniert-ein-3d-drucker> (besucht am 02.02.2025).
- URLNL16 (2025). de. URL: https://help.prusa3d.com/glossary/175-mm_134816 (besucht am 02.02.2025).
- URLNL17 (2025). de-DE. URL: <https://www.elektronik-kompendium.de/sites/com/2009021.htm> (besucht am 02.02.2025).
- URLNL18 (Dez. 2020). de-DE. URL: <https://2d-spritzguss.de/3d-druck-spritzguss> (besucht am 02.02.2025).
- URLNL19 (2025). de. URL: <https://www.3dprima.com/de/3dprima/tipps-tricks-begriffe-3d-druck> (besucht am 02.02.2025).

- URLPI01* (Mai 2020). de. URL: <https://www.ionos.de/digitalguide/hosting/hosting-technik/http-request-erklaert/> (besucht am 23.09.2024).
- URLPI04* (2024). de. URL: <https://www.redhat.com/de/topics/api/what-is-a-rest-api> (besucht am 23.09.2024).
- URLPI07* (2024). en. URL: <https://www.netguru.com/glossary/react-native> (besucht am 30.10.2024).
- URLPI08* (2024). en. URL: <https://docs.expo.dev/get-started/introduction/> (besucht am 30.10.2024).
- URLPI10* (2024). de. Publication Title: techbold. URL: <https://www.techbold.at/lexikon-eintrag/access-point> (besucht am 16.12.2024).
- URLPI11* (2024). de. URL: <https://www.oracle.com/at/database/what-is-json/> (besucht am 16.12.2024).
- URLPI12* (2024). URL: <https://www.mikrocontroller.net/articles/I2S> (besucht am 16.12.2024).
- URLPI13* (2024). URL: <https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/HTTP-methods> (besucht am 17.12.2024).
- URLPI14* (2024). de. Section: Design Pattern. URL: <https://www.geeksforgeeks.org/singleton-design-pattern/> (besucht am 17.12.2024).
- URLPI15* (2024). URL: <https://en.cppreference.com/w/cpp/language/enum> (besucht am 17.12.2024).
- URLPI16* (2025). de. URL: <https://www.informatik-verstehen.de/lexikon/protokoll/> (besucht am 05.01.2025).
- URLPI17* (2025). de. URL: <https://www.atlassian.com/de/git/tutorials/what-is-git> (besucht am 02.02.2025).
- URLPI18* (2025). de. URL: <https://kinsta.com/de/wissensdatenbank/was-ist-github/> (besucht am 06.01.2025).
- URLPI19* (Feb. 2022). de. URL: <https://www.ionos.at/digitalguide/server/knowhow/multicast-dns/> (besucht am 06.01.2025).
- URLPI20* (2025). de. URL: <https://zid.univie.ac.at/dhcp/> (besucht am 06.01.2025).
- URLPI21* (2025). URL: <https://www.radio-browser.info/> (besucht am 28.01.2025).
- URLPI22* (2025). de. URL: <https://www.it-intouch.de/glossar.firebaseio/> (besucht am 28.01.2025).
- URLPI23* (2025). en. URL: <https://de.legacy.reactjs.org/docs/context.html> (besucht am 28.01.2025).
- URLPI24* (2025). URL: <https://docs.espressif.com/projects/arduino-esp32/en/latest/api/wifi.html> (besucht am 29.01.2025).
- URLPI25* (Sep. 2021). de. URL: <https://www.mittwald.de/blog/hosting/was-ist-eigentlich-node-js> (besucht am 29.01.2025).
- URLPI26* (2025). URL: <https://docs.platformio.org/en/latest/what-is-platformio.html> (besucht am 29.01.2025).

URLPI27 (Nov. 2018). en-US. URL: <https://randomnerdtutorials.com/esp32-flash-memory/> (besucht am 29.01.2025).

URLPI28 (2025). en. URL: <https://docs.expo.dev/router/introduction/> (besucht am 29.01.2025).

URLPI29 (2025). URL: <https://www.latex-project.org/> (besucht am 30.01.2025).

URLPI30 (2025). URL: <https://app.diagrams.net/> (besucht am 30.01.2025).

URLPI31 (2025). en. URL: <https://code.visualstudio.com/docs> (besucht am 30.01.2025).

URLPI32 (Dez. 2023). de. URL: <https://www.ionos.at/digitalguide/websites/web-entwicklung/typescript/> (besucht am 02.02.2025).

Abbildungsverzeichnis

1	MAA Konzept	1
2	UML-Klassendiagramm Mikrocontroller	17
3	UML-Ablaufdiagramm Mikrocontroller Setup	20
4	UML-Ablaufdiagramm Mikrocontroller Loop	22
5	Struktur Smartphone-App	24
6	AdapterItem-Komponente	26
7	StationItem-Komponente	26
8	ConnectionItem-Komponente	26
9	AdapterList-Komponente	27
10	FavouriteStationList-Komponente	27
11	ConnectionList-Komponente	28
12	„Stationen filtern“ - Seite	29
13	Stützen für die Komponenten	33
14	Draufsicht der Basis	34
15	Basis mit den Komponenten (während des Zusammensetzens)	34
16	Draufsicht des Adaptergehäuses	35

Tabellenverzeichnis

1	REST-API Routen	9
2	Status-Farben	18
3	App-Farben	25

Anhang

Anhangsverzeichnis

1 Anhang 1: Projektmanagement-Tools	VIII
1.1 Anhang 1.1: Definition Arbeitspakete	VIII
1.2 Anhang 1.2: Projektstrukturplan	XI
1.3 Anhang 1.3: Gantt-Diagramm	XIII
2 Anhang 2: UML-Klassendiagramm Adapter	XV
3 Anhang 3: Code	XVII
3.1 Anhang 3.1: Code Adapter	XVII
3.2 Anhang 3.2: Code Smartphone-App	LII
4 Anhang 4: Zeichnung Gehäuse Adapter	CX

1 Anhang 1: Projektmanagement-Tools

1.1 Anhang 1.1: Definition Arbeitspakete

1. Planung	
AP1: Planen des Gesamtsystems	
Übernommen von:	Zu erledigen bis:
Nico Lang, Philipp Immler	08.05.202
Zu erledigen/Durchführung/Ziel/Ergebnis:	
- Auswahl von Technologien, Hardware und Softwaretools	
- Festlegen der Funktionsweise	
- Festlegen der Anforderungen an die Software	

1.1 Festlegung Funktionsweise	
AP1.1: Festlegung der Funktionsweise des Gesamtsystems	
Übernommen von:	Zu erledigen bis:
Nico Lang	21.04.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
Ermittlung der groben Funktionsweise des Gesamtsystems:	
- was soll das System können?	
- was soll/muss es nicht können?	
- wie könnte man es erweitern?	

1.2 Auswahl Hardwarekomponenten	
AP1.2: Auswahl der Hardware des Adapters (Elektronik)	
Übernommen von:	Zu erledigen bis:
Nico Lang	30.04.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
- Wie sollte der Adapter ausgestattet sein?	
- Welche technischen Anforderungen sollte dieser erfüllen?	
- Welche elektronischen Bauteile eignen sich/welche nicht?	

1.3 Anforderungen Software Adapter	
AP1.3: Anforderungen an die Software des Adapter	
Übernommen von:	Zu erledigen bis:
Philipp Immler	23.04.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
- Welche Funktionalitäten sollte die Software des Adapters bereitstellen	

1.4 Anforderungen Smartphone-App	
AP1.4: Anforderungen an die Smartphone-App	
Übernommen von:	Zu erledigen bis:
Philipp Immler	28.04.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
- Welche Funktionalitäten soll die Smartphone-App bereitstellen	

1.5 Auswahl Technologien	
AP1.3: Auswahl der Technologien des Adapters	
Übernommen von:	Zu erledigen bis:
Nico Lang	03.05.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
- Welche Technologie sollte der Adapter zum Streamen verwenden?	
- Welche Schnittstellen sollte der Adapter haben?	
- Wie sollen die Adapter untereinander kommunizieren?	

1.6 Auswahl Softwaretools	
AP1.4: Auswahl der Tools für die Softwareentwicklung	
Übernommen von:	Zu erledigen bis:
Philipp Immmer	08.05.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - Welche Bibliotheken/Frameworks/Programmiersprachen werden für die Software des Adapters und für die Smartphoneapp verwendet? - Welche Tools eignen sich/eignen sich nicht? - Mit welchen Tools kann man die Performance steigern? 	

2. Entwicklung	
AP2: Entwicklung/Fertigung der Soft- und Hardware	
Übernommen von:	Zu erledigen bis:
Nico Lang, Philipp Immmer	07.07.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - Herstellung des Adapters (Gehäuse, Zusammensetzen) - Entwicklung der Software des Adapters - Entwicklung der Smartphoneapp 	

2.1 Entwicklung Software Adapter	
AP2.3: Entwicklung der Software des Adapters	
Übernommen von:	Zu erledigen bis:
Philipp Immmer	06.06.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - Entwicklung der Software des Adapters 	

2.2 Entwicklung Smartphone-App	
AP2.4: Entwicklung/Programmierung der Smartphoneapp	
Übernommen von:	Zu erledigen bis:
Philipp Immmer	02.07.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - Entwicklung der Smartphoneapp 	

2.3 Design Adaptergehäuse	
AP2.5: Entwicklung/Design des Adaptergehäuses	
Übernommen von:	Zu erledigen bis:
Nico Lang	07.06.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - Design des Modells für das Adaptergehäuse in einem CAD - Wie soll das Gehäuse grob aussehen/worauf sollte Wert gelegt werden? (schlicht, modern, einfach) - Wie kann man das Gehäuse möglichst praktisch und kompakt designen? - Wie kann man das Gehäuse sicher/robust designen? - Wie löst man die Wärmeableitung? 	

2.4 Fertigung Adaptergehäuse	
AP2.6: Fertigung/Herstellung des Adaptergehäuses	
Übernommen von:	Zu erledigen bis:
Nico Lang	09.06.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - Fertigung des zuvor designten Gehäuses für den Adapter - Welche Fertigungsverfahren kommen in Frage? - Welches Fertigungsverfahren wird verwendet? - Wie viel kostet die Herstellung eines Gehäuses? 	

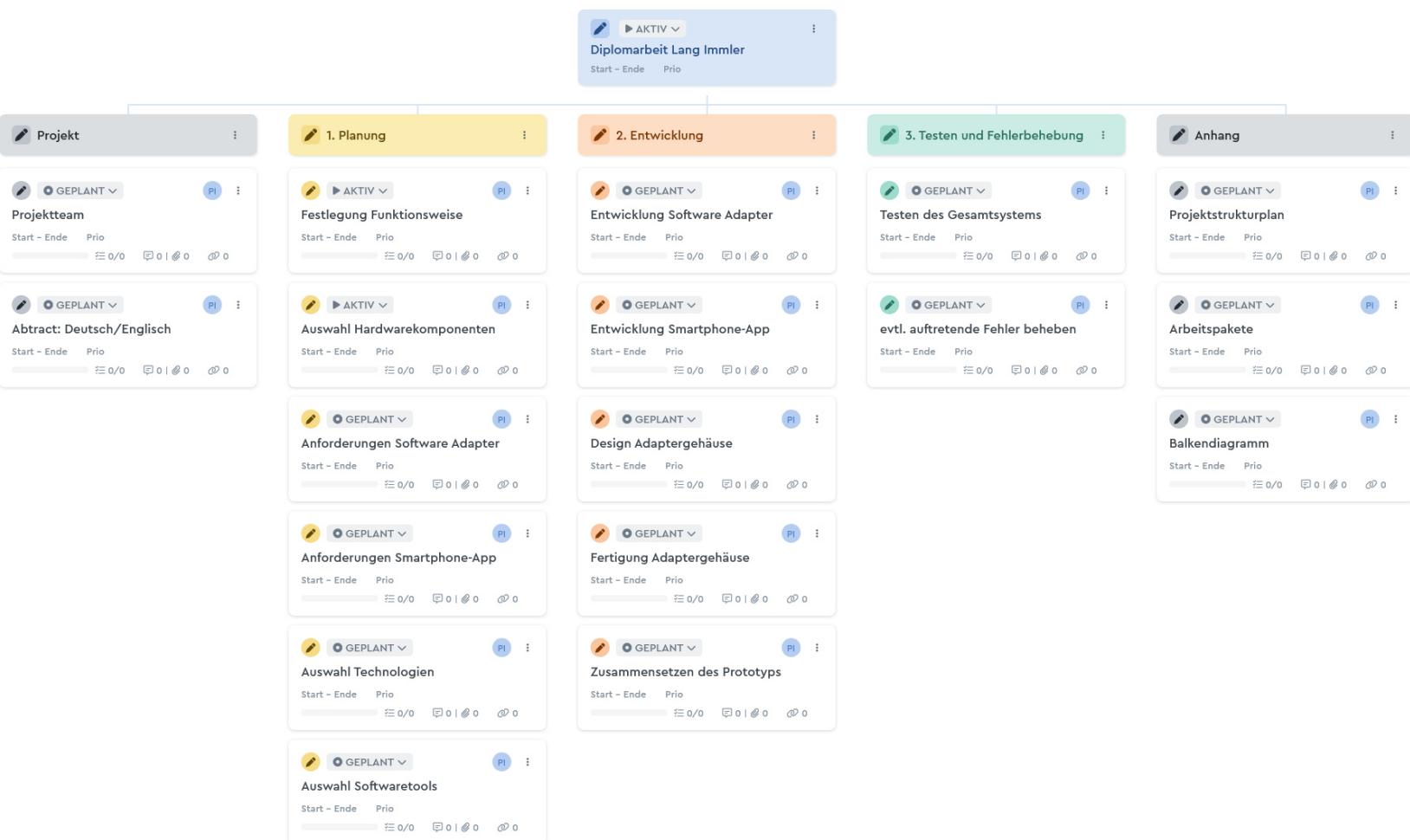
2.5 Zusammensetzen des Prototyps	
AP2.2: Zusammensetzen des Prototyps	
Übernommen von:	Zu erledigen bis:
Nico Lang	07.07.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - Schaltplan - Verdrahten - Kleben 	

3. Testen und Fehlerbehebung	
AP3: Überprüfung des Gesamtsystems auf Fehler und Behebung dieser	
Übernommen von:	Zu erledigen bis:
Nico Lang, Philipp Immler	07.08.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - 	

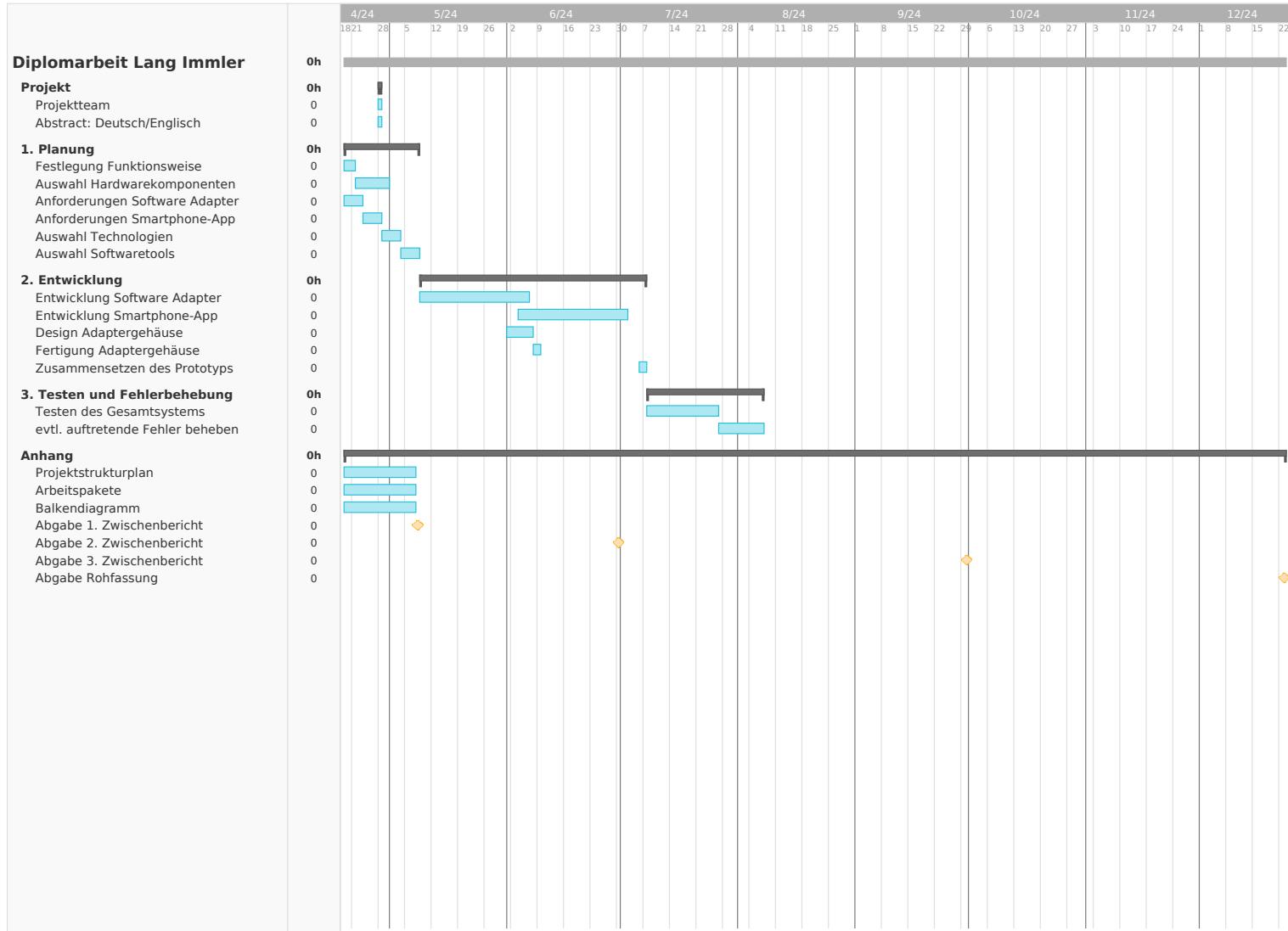
3.1 Testen des Gesamtsystems	
AP3.1: Testen auf Fehler im Gesamtsystem	
Übernommen von:	Zu erledigen bis:
Nico Lang	26.07.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - Test der groben Funktionsweise des Gesamtsystems 	

3.2 evtl. auftretende Fehler beheben	
AP3.2: falls Fehler im Gesamtsystem auftreten, diese beheben	
Übernommen von:	Zu erledigen bis:
Nico Lang, Philipp Immler	07.08.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
<ul style="list-style-type: none"> - falls Fehler im Gesamtsystem auftreten, diese beheben - je nach Fehler, Komponenten austauschen/Funktionsweisen ändern 	

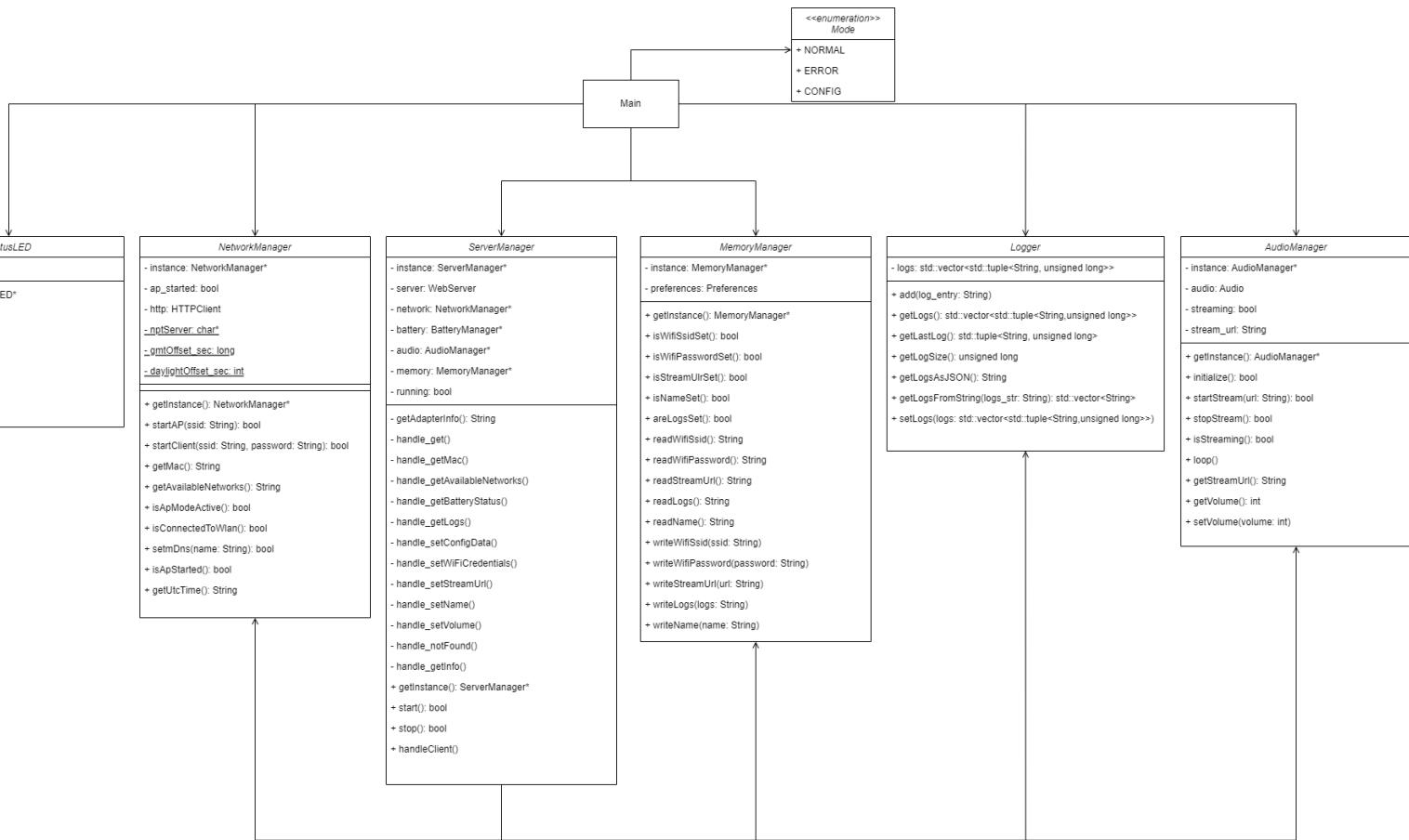
1.2 Anhang 1.2: Projektstrukturplan



1.3 Anhang 1.3: Gantt-Diagramm



2 Anhang 2: UML-Klassendiagramm Adapter



3 Anhang 3: Code

3.1 Anhang 3.1: Code Adapter

AudioManager.h

```
1 #ifndef AUDIOMANAGER_H
2 #define AUDIOMANAGER_H
3
4 #include <Arduino.h>
5 #include "constants.h"
6 #include "AudioFileSourceICYStream.h"
7 #include "AudioFileSourceBuffer.h"
8 #include "AudioGeneratorMP3.h"
9 #include "AudioOutputI2S.h"
10 #include "Logger.h"
11
12 class AudioManager{
13     private:
14         static AudioManager* instance;
15         AudioGeneratorMP3 *gen;
16         AudioFileSourceICYStream *src;
17         AudioFileSourceBuffer *buff;
18         AudioOutputI2S *out;
19         bool streaming;
20         String stream_url;
21         int volume;
22
23     AudioManager();
24     ~AudioManager();
25
26     public:
27         static AudioManager* getInstance();
28
29         /**
30          * sets the url, from which the audio stream should be received
31          */
32         void setStreamUrl(String url);
33
34         /**
35          * starts to receive the audio stream from the given url
36          * @param url URL of the audio stream, which should be received
37          */
38         void startStream();
39
40         /**
```

```

41     * stops the current audio stream
42     */
43     void stopStream();
44
45     /**
46     * returns the stream url
47     */
48     String getStreamUrl();
49
50     /**
51     * returns if the audio stream paused
52     */
53     bool isPaused();
54
55     /**
56     * handles the audio process
57     */
58     void loop();
59
60     /**
61     * sets the volume of the output
62     * @param volume the desired volume, in the range between 0 an 100
63     */
64     void setVolume(int volume);
65
66     /**
67     * returns the volume, which is currently set
68     */
69     int getVolume();
70 };
71 #endif

```

AudioManger.cpp

```

1 #include "AudioManager.h"
2
3 AudioManager* AudioManager::instance = nullptr;
4
5 void MDCallback(void *cbData, const char *type, bool isUnicode, const char
6     ↪ *string) //for debugging
7 {
8     const char *ptr = reinterpret_cast<const char *>(cbData);
9     (void) isUnicode;
10    char s1[32], s2[64];
11    strncpy_P(s1, type, sizeof(s1));
12    s1[sizeof(s1)-1]=0;

```

```

12    strncpy_P(s2, string, sizeof(s2));
13    s2[sizeof(s2)-1]=0;
14    Serial.printf("METADATA(%s) '%s' = '%s'\n", ptr, s1, s2);
15    Serial.flush();
16 }
17
18 void StatusCallback(void *cbData, int code, const char *string) //for
19     ↪ debugging
20 {
21     const char *ptr = reinterpret_cast<const char *>(cbData);
22     // Note that the string may be in PROGMEM, so copy it to RAM for printf
23     char s1[64];
24     strncpy_P(s1, string, sizeof(s1));
25     s1[sizeof(s1)-1]=0;
26     Serial.printf("STATUS(%s) '%d' = '%s'\n", ptr, code, s1);
27     Serial.flush();
28 }
29
30 AudioManager::AudioManager(){
31     stream_url = "";
32     streaming = false;
33     volume = 100;
34     audioLogger = &Serial;
35     src = new AudioFileSourceICYStream();
36     src->RegisterMetadataCB(MDCallback, (void*)"ICY");
37     buff = new AudioFileSourceBuffer(src, AUDIO_BUFFERSIZE);
38     buff->RegisterStatusCB(StatusCallback, (void*)"buffer");
39     out = new AudioOutputI2S();
40     out->SetPinout(I2S_BCLK_PIN, I2S_LRC_PIN, I2S_DOUT_PIN);
41     out->SetBitsPerSample(AUDIO_BITSPERSAMPLE);
42     out->SetChannels(AUDIO_CHANNELS);
43     out->SetRate(AUDIO_SAMPLERATE);
44     gen = new AudioGeneratorMP3();
45     gen->RegisterStatusCB(StatusCallback, (void*)"mp3");
46 }
47
48 AudioManager* AudioManager::getInstance(){
49     if(instance == nullptr){
50         instance = new AudioManager();
51     }
52     return instance;
53 }
54
55 void AudioManager::setStreamUrl(String url){
      this->stream_url = url;

```

```

56 }
57
58 void AudioManager::stopStream(){
59     Logger::add("stopping audio stream");
60     streaming = false;
61     if(gen->isRunning()){
62         gen->stop();
63     }
64     if(src->isOpen()){
65         src->close();
66     }
67 }
68
69 void AudioManager::startStream(){
70     Logger::add("start streaming audio from " + stream_url);
71     stopStream();
72     src->open(stream_url.c_str());
73     gen->begin(buff, out);
74     streaming = true;
75 }
76
77 String AudioManager::getStreamUrl(){
78     return stream_url;
79 }
80
81 bool AudioManager::isPaused(){
82     return !streaming;
83 }
84
85 void AudioManager::loop(){
86     gen->loop();
87 }
88
89 void AudioManager::setVolume(int volume){
90     if(volume >= 0 && volume <= 100){
91         this->volume = volume;
92         float gain = (float)volume/(float)100;
93         Logger::add("setting gain to " + String(gain));
94         out->SetGain(gain);
95     }
96 }
97
98 int AudioManager::getVolume(){
99     return volume;
100 }
```

BatteryManager.cpp

```
1 #include "BatteryManager.h"
2
3 BatteryManager* BatteryManager::instance = nullptr;
4
5 BatteryManager::BatteryManager(){}
6
7 BatteryManager::~BatteryManager(){}
8
9 BatteryManager* BatteryManager::getInstance(){
10     if(instance == nullptr){
11         instance = new BatteryManager();
12     }
13     return instance;
14 }
15
16 /**
17 * initializes the needed pins
18 */
19 void BatteryManager::initializePins(){
20     // ...
21 }
22
23 /**
24 * returns the charging status of the battery
25 *
26 * @return charging status of the battery, in percent (0 - 100), as a
27     → String
28 */
29 int BatteryManager::getBatteryStatus(){
30     return 100; //default
}
```

BatteryManager.h

```
1 #ifndef BATTERYMANAGER_H
2 #define BATTERYMANAGER_H
3
4 #include "Arduino.h"
5 #include "constants.h"
6
7 /**
8 * manages the loading and status of the battery
9 */
```

```

10 class BatteryManager{
11     private:
12         static BatteryManager *instance;
13
14     BatteryManager();
15     ~BatteryManager();
16
17     public:
18         static BatteryManager* getInstance();
19         void initializePins();
20         int getBatteryStatus();
21     };
22 #endif

```

constants.h

```

1 /**
2  * file with constants, which are needed in the code
3 */
4
5 #ifndef CONSTANTS_H
6 #define CONSTANTS_H
7
8 #include "Arduino.h"
9
10 //pins
11 const int I2S_BCLK_PIN = 27;
12 const int I2S_LRC_PIN = 26;
13 const int I2S_DOUT_PIN = 25;
14 const int BUTTON_PIN = 12;
15 const int LED_RED = 15;
16 const int LED_GREEN = 2;
17 const int LED_BLUE = 4;
18
19 //network
20 const IPAddress AP_LOCAL_IP(192,168,0,1);
21 const IPAddress AP_GATEWAY_IP(192,168,0,1);
22 const IPAddress AP_SUBNET_IP(255,255,255,0);
23 //const String AP_SSID = "Microcontroller";
24 const int MAX_RECONNECTION_TRIES = 2;
25 const unsigned long MAX_CONNECTION_TIME = 5000;
26
27 //memory
28 const String MEMORY_NAMESPACE = "variables";
29
30 const String SSID_KEY = "ssid";

```

```

31 const String PASSWORD_KEY = "password";
32 const String URL_KEY = "wifi";
33 const String LOGS_KEY = "logs";
34 const String NAME_KEY = "name";
35 const String IP_KEY = "ip";
36
37 //audio
38 const int AUDIO_BUFFERSIZE = 32768;
39 const int AUDIO_BITSPERSAMPLE = 16;
40 const int AUDIO_SAMPLERATE = 44100;
41 const int AUDIO_CHANNELS = 2;
42
43 //button
44 const int BUTTON_CONFIG_DURATION = 3000; //time for which the button has
    ↪ to be pressed, that config mode is activated
45
46 //other constants
47 const unsigned long SERIAL_BAUDRATE = 9600;
48 const int BUTTON_PRESS_SLEEP_TIME = 2000;
49 const unsigned long WLAN_REQUEST_PERIOD = 10000;
50 const int AUDIO_VOLUME = 10; //0-21
51 const int SERVER_PORT = 8080;
52 //const String DEFAULT_NAME = "MSA";
53 const String TIME_URL = "http://worldtimeapi.org/api/ip";
54 const int DEFAULT_VOLUME = 10;
55 #endif

```

Logger.cpp

```

1 #include "Logger.h"
2
3 std::vector<std::tuple<String, unsigned long>> Logger::logs;
4
5 void Logger::add(String log_entry){
6     Serial.println(log_entry); //for debug purposes
7     int time = 0;
8     Logger::logs.push_back(std::make_tuple(log_entry, time));
9 }
10
11 std::vector<std::tuple<String, unsigned long>> Logger::getLogs(){
12     return logs;
13 }
14
15 String Logger::getLogsAsJSON(){
16     JsonDocument doc;
17     for(int i = 0; i < logs.size(); i++){

```

```

18     doc[i]["log_entry"] = std::get<0>(logs.at(i));
19     doc[i]["time"] = std::get<1>(logs.at(i));
20 }
21 String logs;
22 serializeJson(doc, logs);
23 return logs;
24 }
25
26 std::tuple<String, unsigned long> Logger::getLastLog(){
27     int log_size = getLogSize();
28     return logs.at(log_size);
29 }
30
31 unsigned long Logger::getLogSize(){
32     return logs.size();
33 }
34
35 std::vector<String> Logger::getLogsFromString(String logs_str){
36     //empty
37 }
38
39 void Logger::setLogs(std::vector<std::tuple<String, unsigned long>> logs){
40     Logger::logs = logs;
41 }
42
43 void Logger::clearLogs(){
44     Logger::logs.clear();
45 }
```

Logger.h

```

1 #ifndef LOGGER_H
2 #define LOGGER_H
3 #include <Arduino.h>
4 #include <vector>
5 #include <ArduinoJson.h>
6
7 class Logger{
8     private:
9         /**
10          * vector, in which the logs are written as a String
11          */
12     static std::vector<std::tuple<String, unsigned long>> logs;
13
14     public:
15         /**
```

```

16     * adds a log entry to the logs vector
17     */
18     static void add(String log_entry);
19
20 /**
21  * returns the vector of all logs
22  */
23     static std::vector<std::tuple<String,unsigned long>> getLogs();
24
25 /**
26  * returns the last log of the logs vector
27  */
28     static std::tuple<String, unsigned long> getLastLog();
29
30 /**
31  * returns the size of the logs vecotor, as an unsigned long
32  */
33     static unsigned long getLogSize();
34
35 /**
36  * returns the logs as a serialized json
37  */
38     static String getLogsAsJSON();
39
40 /**
41  * reconverts a string with logs, seperated with commas to a log
42  *      ↢ vector
43  */
44     static std::vector<String> getLogsFromString(String logs_str);
45
46 /**
47  * sets log vector to the given log vector
48  */
49     static void setLogs(std::vector<std::tuple<String,unsigned long>>
50                         ↢ logs);
51
52 /**
53  * clears the vector
54  */
55     static void clearLogs();
56 };
57 #endif

```

main.cpp

```

1 //including libraries

```

```

2 #include "Arduino.h"
3 #include "constants.h"
4 #include "NetworkManager.h"
5 #include "StatusLED.h"
6 #include "MemoryManager.h"
7 #include "Logger.h"
8 #include "ServerManager.h"
9 #include "AudioManager.h"
10 #include "Mode.h"

11
12 Mode mode = NORMAL;
13 unsigned long actual_time = 0;
14 unsigned long last_wlan_request_time = 0;
15 int wlan_reconnect_tries = 0;
16 unsigned long last_log_size = 0;
17 String last_log = "";
18 String name;

19
20 unsigned long wlan_connection_start = 0;
21 int wlan_reconnection_tries = 0;

22
23 //for button:
24 unsigned long press_start = 0;
25 unsigned long press_end = 0;
26 bool last_state = 0;

27
28 NetworkManager* network;
29 StatusLED* statusLED;
30 MemoryManager* memory;
31 ServerManager* server;
32 AudioManager* audio;
33 BatteryManager* battery;

34
35 void setMode(Mode m);
36 void handleButton();
37 void activateStandby();

38
39 void setup(){
40     //set serial baudrate
41     Serial.begin(SERIAL_BAUDRATE);

42
43     //initialize button pin and attach interrupt to button
44     pinMode(BUTTON_PIN, INPUT_PULLDOWN);
45     esp_sleep_enable_ext0_wakeup(GPIO_NUM_12, 1); //wakes the esp32 up
        ↳ from deep sleep, when gpio 12 (button pin) is HIGH

```

```

46
47 //getting instances of singleton classes
48 network = NetworkManager::getInstance();
49 statusLED = StatusLED::getInstance();
50 memory = MemoryManager::getInstance();
51 server = ServerManager::getInstance();
52 battery = BatteryManager::getInstance();
53 audio = AudioManager::getInstance();

54
55 //setting name
56 //name = "MAA_" + network->getMac()

57
58 //turn status led off at the beginning
59 statusLED->setOff();

60
61 //if WLAN-credentials are set, read them and try to connect to WLAN
62 if(memory->isWlanSsidSet() && memory->isWlanPasswordSet()){
63     Logger::add("wlan credentials set in memory");
64     String wlan_ssid = memory->readWlanSsid();
65     String wlan_password = memory->readWlanPassword();
66     Logger::add("SSID: " + wlan_ssid);
67     Logger::add("password: " + wlan_password);
68     Logger::add("starting wlan client");
69     network->startClient(wlan_ssid, wlan_password, name);
70     wlan_connection_start = millis();
71     while(!network->isConnectedToWlan() && mode != ERROR){
72         Serial.print(".");
73         delay(100);
74         if((millis() - wlan_connection_start) >= MAX_CONNECTION_TIME){
75             ↪ //if the max connection time for the wifi is exceeded,
76             ↪ activate error mode
77             Logger::add("max wlan connection time exceeded");
78             setMode(ERROR);
79         }
80     }
81     if(network->isConnectedToWlan()){//if connected to wlan, set mode
82         ↪ to normal
83         Logger::add("connected to wlan");
84         setMode(NORMAL);
85     }
86 } else { //if wlan credentials are not set, set mode to error
87     Logger::add("wlan credentials not set in memory");
88     setMode(ERROR);
89 }

```

```

88
89 void loop(){
90     handleButton(); //check if button is pressed
91     actual_time = millis(); //time since start in ms
92
93     if(mode != ERROR){
94         if(mode == NORMAL){
95             //check if still connected to Wlan
96             if((actual_time - last_wlan_request_time) >= WLAN_REQUEST_PERIOD){
97                 Serial.println("free heap: " + String(esp_get_free_heap_size()
98                                         ↪));
99                 if(!network->isConnectedToWlan()){ //if not connected to wlan,
100                     ↪ try to reconnect
101                     if(wlan_reconnection_tries <= MAX_RECONNECTION_TRIES){
102                         network->reconnect();
103                         wlan_reconnect_tries++;
104                         Logger::add("reconnecting to wlan");
105                     } else {
106                         Logger::add("not connected to wlan");
107                         setMode(ERROR);
108                     }
109                 } else {
110                     int rssи = network->getRssi();
111                     wlan_reconnect_tries = 0;
112                 }
113                 last_wlan_request_time = actual_time;
114             }
115             if(!audio->isPaused()){ //if audio routine is running, execute
116                 ↪ audio loop
117                 audio->loop();
118             }
119             } else { //mode is config
120                 if(!network->isApStarted()){ //if ap is not running, start ap
121                     Logger::add("starting ap");
122                     network->startAP(name);
123                 }
124             }
125
126             if(server->isRunning()){ //if server is running, it should handle
127                 ↪ clients
128                 server->handleClient();
129             } else {
130                 Logger::add("starting web server");
131                 server->start();
132                 Logger::add("setting mDNS");
133             }
134         }
135     }
136 }
```

```

129         if(!network->setmDns(name)){
130             Logger::add("mDNS couldn't be set");
131         }
132     }
133 }
134
135 /**
136 * method for setting modes
137 * @param m Mode which should be set
138 */
139 void setMode(Mode m){
140     if(m == NORMAL){
141         Logger::add("setting mode to normal");
142         mode = NORMAL;
143         statusLED->setGreen();
144     } else if(m == ERROR){
145         Logger::add("setting mode to error");
146         mode = ERROR;
147         statusLED->setRed();
148     } else if(m == CONFIG){
149         Logger::add("setting mode to config");
150         mode = CONFIG;
151         statusLED->setBlue();
152     }
153 }
154
155 /**
156 * method for checkin if button is pressed
157 */
158 void handleButton(){
159     int state = digitalRead(BUTTON_PIN);
160     if(state == 1 && last_state == 0){ //button has been pressed
161         press_start = millis();
162     } else if(state == 0 && last_state == 1){ //button has been released
163         press_end = millis();
164     }
165     if(press_start > 0 && press_end > 0){
166         if((press_end - press_start) >= 3000){
167             setMode(CONFIG);
168         } else {
169             activateStandby();
170         }
171         press_start = 0;
172         press_end = 0;

```

```

174     }
175     last_state = state;
176 }
177
178 /**
179 * method for activating standby mode (deep sleep)
180 */
181 void activateStandby(){
182     Logger::add("enabling standby");
183     statusLED->setOff();
184     esp_deep_sleep_start();
185 }
```

MemoryManager.cpp

```

1 #include "MemoryManager.h"
2
3 MemoryManager* MemoryManager::instance = nullptr;
4
5 MemoryManager::MemoryManager(){}
6 MemoryManager::~MemoryManager(){}
7
8 MemoryManager* MemoryManager::getInstance(){
9     if (!instance) {
10         instance = new MemoryManager();
11     }
12     return instance;
13 }
14
15 bool MemoryManager::isWlanSsidSet(){
16     preferences.begin(MEMORY_NAMESPACE.c_str());
17     return preferences.isKey(SSID_KEY.c_str());
18     preferences.end();
19 }
20
21 bool MemoryManager::isWlanPasswordSet(){
22     preferences.begin(MEMORY_NAMESPACE.c_str());
23     return preferences.isKey(PASSWORD_KEY.c_str());
24     preferences.end();
25 }
26
27 bool MemoryManager::isStreamUrlSet(){
28     preferences.begin(MEMORY_NAMESPACE.c_str());
29     return preferences.isKey(URL_KEY.c_str());
30     preferences.end();
31 }
```

```

32     bool MemoryManager::isNameSet(){
33         preferences.begin(MEMORY_NAMESPACE.c_str());
34         return preferences.isKey(NAME_KEY.c_str());
35         preferences.end();
36     }
37
38
39     bool MemoryManager::areLogsSet(){
40         preferences.begin(MEMORY_NAMESPACE.c_str());
41         return preferences.isKey(LOGS_KEY.c_str());
42         preferences.end();
43     }
44
45     String MemoryManager::readWlanSsid(){
46         Logger::add("reading wlan ssid from memory");
47         preferences.begin(MEMORY_NAMESPACE.c_str());
48         String ssid = preferences.getString(SSID_KEY.c_str());
49         preferences.end();
50         return ssid;
51     }
52
53     String MemoryManager::readWlanPassword(){
54         Logger::add("reading wlan password from memory");
55         preferences.begin(MEMORY_NAMESPACE.c_str());
56         String ssid = preferences.getString(PASSWORD_KEY.c_str());
57         preferences.end();
58         return ssid;
59     }
60
61
62     String MemoryManager::readStreamUrl(){
63         Logger::add("reading stream url from memory");
64         preferences.begin(MEMORY_NAMESPACE.c_str());
65         String url = preferences.getString(URL_KEY.c_str());
66         preferences.end();
67         return url;
68     }
69
70     String MemoryManager::readLogs(){
71         Logger::add("reading logs from memory");
72         preferences.begin(MEMORY_NAMESPACE.c_str());
73         String logs = preferences.getString(LOGS_KEY.c_str());
74         preferences.end();
75         return logs;
76     }

```

```

77
78 String MemoryManager::readName(){
79     //Logger::add("reading name from memory");
80     preferences.begin(MEMORY_NAMESPACE.c_str());
81     String name = preferences.getString(NAME_KEY.c_str());
82     preferences.end();
83     return name;
84 }
85
86 String MemoryManager::readIp(){
87     Logger::add("reading ip from memory");
88     preferences.begin(MEMORY_NAMESPACE.c_str());
89     String ip = preferences.getString(IP_KEY.c_str());
90     preferences.end();
91     return ip;
92 }
93
94 void MemoryManager::writeWlanSsid(String ssid){
95     Logger::add("writing wlan ssid in memory");
96     preferences.begin(MEMORY_NAMESPACE.c_str());
97     preferences.putString(SSID_KEY.c_str(), ssid);
98     preferences.end();
99 }
100
101 void MemoryManager::writeWlanPassword(String password){
102     Logger::add("writing wlan password in memory");
103     preferences.begin(MEMORY_NAMESPACE.c_str());
104     preferences.putString(PASSWORD_KEY.c_str(), password);
105     preferences.end();
106 }
107
108 void MemoryManager::writeStreamUrl(String url){
109     Logger::add("writing stream url in memory");
110     preferences.begin(MEMORY_NAMESPACE.c_str());
111     preferences.putString(URL_KEY.c_str(), url);
112     preferences.end();
113 }
114
115 void MemoryManager::writeLogs(String logs){
116     Logger::add("writing logs in memory");
117     preferences.begin(MEMORY_NAMESPACE.c_str());
118     preferences.putString(LOGS_KEY.c_str(), logs);
119     preferences.end();
120 }
121

```

```

122 void MemoryManager::writeName(String name){
123     Logger::add("writing name in memory");
124     preferences.begin(MEMORY_NAMESPACE.c_str());
125     preferences.putString(NAME_KEY.c_str(), name);
126     preferences.end();
127 }
128
129 void MemoryManager::writeIp(String ip){
130     Logger::add("writing ip in memory");
131     preferences.begin(MEMORY_NAMESPACE.c_str());
132     preferences.putString(IP_KEY.c_str(), ip);
133     preferences.end();
134 }
135
136 void MemoryManager::clear(){
137     preferences.begin(MEMORY_NAMESPACE.c_str());
138     preferences.clear();
139     preferences.end();
140 }
```

MemoryManager.h

```

1 #ifndef MEMORYMANAGER_H
2 #define MEMORYMANAGER_H
3
4 #include <Arduino.h>
5 #include <Preferences.h>
6 #include <constants.h>
7 #include "Logger.h"
8
9 class MemoryManager{
10     private:
11         static MemoryManager* instance;
12         MemoryManager();
13         ~MemoryManager();
14         MemoryManager(const MemoryManager&) = delete;
15         MemoryManager& operator = (const MemoryManager&) = delete;
16         Preferences preferences;
17
18     public:
19         static MemoryManager* getInstance();
20
21         /**
22          * returns, if the wlan ssid is set to the memory
23          *
24          * @return if WLAN-SSID is set to the memory

```

```

25     */
26     bool isWlanSsidSet();
27
28 /**
29 * returns, if the wlan password is set to the memory
30 *
31 * @return if WLAN-Password is set to the memory
32 */
33     bool isWlanPasswordSet();
34
35 /**
36 * returns, if the stream url is set to the memory
37 *
38 * @return if Stream-URL is set to the memory
39 */
40     bool isStreamUrlSet();
41
42 /**
43 * returns, if the name is set to the memory
44 *
45 * @return if name of the microcontroller is set to the memory
46 */
47     bool isNameSet();
48
49 /**
50 * returns, if the last logs are set to the memory
51 *
52 * @return if Last logs are set to the memory
53 */
54     bool areLogsSet();
55
56 /**
57 * returns, if the ip-address is set to the memory
58 *
59 * @return if ip address is set to the memory
60 */
61     bool isIpSet();
62
63 /**
64 * reads the wlan ssid from the memory
65 * @return WLAN-SSID, as a String
66 */
67     String readWlanSsid();
68
69 /**

```

```

70     * reads the wlan password from the memory
71     * @return WLAN-Password, as a String
72     */
73     String readWlanPassword();
74
75     /**
76     * reads the last stream url from the memory
77     * @return last Stream-URL, as a String
78     */
79     String readStreamUrl();
80
81     /**
82     * reads the last logs from the memory
83     * @return last Logs, as a String
84     */
85     String readLogs();
86
87     /**
88     * reads the name from the memory
89     *
90     * @return name of the microcontroller, as a String
91     */
92     String readName();
93
94     /**
95     * reads the ip address from the memory
96     *
97     * @return ip address of the microcontroller, as a String
98     */
99     String readIp();
100
101    /**
102     * writes the given ssid to the memory
103     *
104     * @param ssid WLAN-SSID which should be written to the memory
105     */
106     void writeWlanSsid(String ssid);
107
108    /**
109     * writes the given password to the memory
110     *
111     * @param password WLAN-Password which should be written to the
112     *                 ↢ memory
113     */
114     void writeWlanPassword(String password);

```

```

114
115     /**
116      * writes the given url to the memory
117      *
118      * @param url Stream-URL which should be written to the memory
119      */
120     void writeStreamUrl(String url);

121
122     /**
123      * writes the given logs to the memory
124      *
125      * @param logs Logs which should be written to the memory
126      */
127     void writeLogs(String logs);

128
129     /**
130      * writes the given name to the memory
131      *
132      * @param name Name of the microcontroller, as a String
133      */
134     void writeName(String name);

135
136     /**
137      * writes the given ip address to the memory
138      *
139      * @param ip IP Address of the microcontroller, as a String
140      */
141     void writeIp(String ip);

142
143     /**
144      * clears the memory
145      */
146     void clear();
147 };
148 #endif

```

Mode.h

```

1 enum Mode{
2     NORMAL ,
3     ERROR ,
4     CONFIG
5 };

```

NetworkManager.cpp

```

1 //including header file
2 #include "NetworkManager.h"
3
4 NetworkManager* NetworkManager::instance = nullptr;
5
6
7 /**
8 * constructor
9 * declares the needed variables
10 */
11 NetworkManager::NetworkManager(){
12     ap_started = false;
13     //Log::add("network manager class created");
14 }
15
16 NetworkManager* NetworkManager::getInstance(){
17     if(instance == nullptr){
18         instance = new NetworkManager();
19     }
20     return instance;
21 }
22
23 /**
24 * returns the mac address of the esp32
25 */
26 String NetworkManager::getMac(){
27     return WiFi.macAddress();
28 }
29
30 /**
31 * scans for available networks and returns the ssid and rssi (strength)
32 * of the found networks as a json
33 */
34 String NetworkManager::getAvailableNetworks(){
35     JsonDocument networks;
36     if(WiFi.getMode() == WIFI_AP){
37         int available_networks = WiFi.scanNetworks(false);
38         for(int i = 0; i < available_networks; i++){
39             networks[i]["ssid"] = WiFi.SSID(i);
40             networks[i]["rssi"] = WiFi.RSSI(i);
41         }
42     }
43     String result;
44     serializeJson(networks, result);
45     return result;

```

```

45 }
46
47 /**
48 * starts an access point
49 */
50 bool NetworkManager::startAP(String ssid){
51     //Log::add("starting ap");
52     if(WiFi.getMode() != WIFI_AP){
53         WiFi.mode(WIFI_AP);
54     }
55     ap_started = true;
56     return WiFi.softAPConfig(AP_LOCAL_IP, AP_GATEWAY_IP, AP_SUBNET_IP) &&
57         WiFi.softAP(ssid);
58 }
59 /**
60 * starts esp32 wlan client which connects to the access point with the
61 * given credentials
62 */
63 bool NetworkManager::startClient(String ssid, String password, String
64     hostname){
65     if(WiFi.getMode() == WIFI_AP){ //if wifi is in ap mode, ap mode will
66         // be disabled and station mode will be enabled
67         WiFi.softAPdisconnect();
68         WiFi.mode(WIFI_STA);
69     }
70     WiFi.disconnect();
71     int n = WiFi.scanNetworks();
72     for(int i = 0; i < n; i++){
73         if(WiFi.SSID(i) == ssid){
74             String bssid = WiFi.BSSIDstr(i);
75             Logger::add("ap mac: " + bssid);
76             WiFi.setHostname(hostname.c_str());
77             WiFi.begin(WiFi.SSID(i), password, 0, WiFi.BSSID(i));
78             return true;
79         }
80     }
81     return false;
82 }
83
84
85 bool NetworkManager::isApModeActive(){

```

```

86     return WiFi.getMode() == WIFI_AP;
87 }
88
89 bool NetworkManager::isConnectedToWlan(){
90     if(!isApModeActive()){
91         return WiFi.status() == WL_CONNECTED;
92     }
93     return false;
94 }
95
96 bool NetworkManager::setmDns(String name){
97     return MDNS.begin(name) && MDNS.addService("http", "tcp", 80);
98 }
99
100 bool NetworkManager::isApStarted(){
101     return ap_started;
102 }
103
104 String NetworkManager::getUtcTime(){
105     struct tm timeinfo;
106     configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
107     getLocalTime(&timeinfo);
108     return "example";
109 }
110
111 int NetworkManager::getRssi(){
112     if(this->isConnectedToWlan()){
113         return WiFi.RSSI();
114     }
115     return 0;
116 }
```

NetworkManager.h

```

1 #ifndef NETWORKMANAGER_H
2 #define NETWORKMANAGER_H
3
4 //including needed libraries
5 #include "Arduino.h"
6 #include "WiFi.h"
7 #include "constants.h"
8 #include "ArduinoJson.h"
9 #include "MemoryManager.h"
10 #include "Logger.h"
11 #include "ESPmDNS.h"
12 #include "HTTPClient.h"
```

```

13 #include "time.h"
14
15 //using namespace std for String an vectors
16 using namespace std;
17
18 /**
19 * responsible for network tasks, like:
20 * providing an access point
21 * acting as a WiFi client
22 */
23 class NetworkManager{
24 private:
25     static NetworkManager* instance;
26     NetworkManager();
27     ~NetworkManager();
28     NetworkManager(const NetworkManager*) = delete;
29     NetworkManager& operator = (const NetworkManager&) = delete;
30     bool ap_started;
31     HTTPClient http;
32     const char* ntpServer = "pool.ntp.org";
33     const long gmtOffset_sec = 0;
34     const int daylightOffset_sec = 3600;
35
36 public:
37     static NetworkManager* getInstance();
38
39 /**
40 * starts the access point
41 *
42 * @param ssid SSID of the access point, as a String
43 * @return starting process successful, as a boolean
44 */
45     bool startAP(String ssid);
46
47 /**
48 * starts a wifi client
49 *
50 * @param ssid WLAN-SSID, as a String
51 * @param password WLAN-Password, as a String
52 * @return connection successful, as a bool
53 */
54     bool startClient(String ssid, String password, String hostname);
55
56 /**
57 * reconnects to the ap

```

```

58     */
59     void reconnect();
60
61     /**
62      * returns the MAC-Address of the ESP32, as a String
63      *
64      * @return Mac-Address of the ESP32, as a String
65      */
66     String getMac();
67
68     /**
69      * scans for available networks and returns the ssid and rssi (
70      * ↗ strength)
71      * of the found networks as a JSON converted to a String
72      *
73      * @return all available networks, as a serialized json
74      */
75     String getAvailableNetworks();
76
77     /**
78      * returns if wifi module is in access point mode
79      *
80      * @return Acces Point Mode active, as a bool
81      */
82     bool isApModeActive();
83
84     /**
85      * returns if wifi client is connected to WLAN
86      *
87      * @return connected to WLAN, as a bool
88      */
89     bool isConnectedToWlan();
90
91     /**
92      * sets mDNS
93      *
94      * @param name Name of the domain
95      */
96     bool setmDns(String name);
97
98     /**
99      * returns if ap is started
100     *
101     * @return ap started, as a bool
102     */

```

```

102     bool isApStarted();
103
104     /**
105      * returns the current utc time, requested from a time server, as
106      * ↪ a String
107      *
108      * @return utc time, as a string
109      */
110     String getUtcTime();
111
112     /**
113      * returns the RSSI of the network currently connected
114      */
115     int getRssi();
116 };
#endif

```

ServerManager.cpp

```

1 #include "ServerManager.h"
2
3 ServerManager* ServerManager::instance = nullptr;
4
5 ServerManager::ServerManager(){
6     network = NetworkManager::getInstance();
7     battery = BatteryManager::getInstance();
8     audio = AudioManager::getInstance();
9     memory = MemoryManager::getInstance();
10    running = false;
11 }
12
13 ServerManager::~ServerManager(){}
14
15 ServerManager* ServerManager::getInstance(){
16     if (!instance) {
17         instance = new ServerManager();
18     }
19     return instance;
20 }
21
22 String ServerManager:: getInfo(){
23     String name = memory->readName();
24     String mac = network->getMac();
25     int volume = audio->getVolume();
26     int battery_status = battery->getBatteryStatus();
27     String station_url = audio->getStreamUrl();

```

```

28     JsonDocument doc;
29     doc["name"] = name;
30     doc["mac"] = mac;
31     doc["volume"] = volume;
32     doc["battery"] = battery_status;
33     doc["stationUrl"] = station_url;
34     String info;
35     serializeJson(doc, info);
36     return info;
37 }
38
39 void ServerManager::handle_get(){
40     Logger::add("get request on route / received");
41     server.send(200, "text/plain", "get request received");
42 }
43
44 void ServerManager::handle_getInfo(){
45     //Logger::add("get request on route /getInfo received");
46     String adapterInfo = getInfo();
47     server.send(200, "application/json", adapterInfo);
48 }
49
50 void ServerManager::handle_getAvailableNetworks(){
51     Logger::add("get request on route /getAvailableNetworks received");
52     String availableNetworks = network->getAvailableNetworks();
53     server.send(200, "application/json", availableNetworks);
54 }
55
56 void ServerManager::handle_getLogs(){
57     Logger::add("get request on route /getLogs received");
58     String logs = Logger::getLogsAsJSON();
59     server.send(200, "application/json", logs);
60 }
61
62 void ServerManager::handle_setWifiCredentials(){
63     Logger::add("post request on route /setWifiCredentials received");
64     if(server.hasArg("ssid") && server.hasArg("password")){
65         String ssid = server.arg("ssid");
66         String password = server.arg("password");
67         Logger::add("writing ssid: " + ssid + " to memory");
68         memory->writeWlanSsid(ssid);
69         Logger::add("writing password: " + password + " to memory");
70         memory->writeWlanPassword(password);
71         server.send(201);
72         Logger::add("restarting esp");

```

```

73     ESP.restart();
74 } else {
75     server.send(400);
76 }
77 }

78

79 void ServerManager::handle_setStreamUrl(){
80     Logger::add("put request on route /setStreamUrl received");
81     if(server.hasArg("url")){
82         String url = server.arg("url");
83         audio->setStreamUrl(url);
84         audio->startStream();
85         server.send(200);
86     } else {
87         server.send(400);
88     }
89 }

90

91 void ServerManager::handle_setName(){
92     Logger::add("put request on route /setName received");
93     if(server.hasArg("name")){
94         String name = server.arg("name");
95         Logger::add("setting new name: " + name + " to memory");
96         memory->writeName(name);
97         server.send(200);
98         Logger::add("restarting esp");
99         ESP.restart();
100    } else {
101        server.send(400);
102    }
103 }

104

105 void ServerManager::handle_setVolume(){
106     Logger::add("put request on route /setVolume received");
107     if(server.hasArg("volume")){
108         int volume = server.arg("volume").toInt();
109         audio->setVolume(volume);
110         server.send(200);
111     } else {
112         server.send(400);
113     }
114 }

115

116 void ServerManager::handle_pauseStream(){
117     Logger::add("put request on route /pauseStream received");

```

```

118     audio->stopStream();
119     server.send(200);
120 }
121
122 void ServerManager::handle_continueStream(){
123     Logger::add("put request on route /continueStream received");
124     audio->startStream();
125     server.send(200);
126 }
127
128 void ServerManager::handleNotFound(){
129     server.send(404, "not found!");
130 }
131
132 bool ServerManager::start(){
133     server.begin(SERVER_PORT);
134     server.on("/", HTTP_GET, bind(&ServerManager::handle_get, this));
135     server.on("/getAvailableNetworks", HTTP_GET, bind(&ServerManager::
136         ↪ handle_getAvailableNetworks, this));
137     server.on("/getLogs", HTTP_GET, bind(&ServerManager::handle_getLogs,
138         ↪ this));
139     server.on("/ getInfo", HTTP_GET, bind(&ServerManager::handle_getInfo,
140         ↪ this));
141     server.on("/ setName", HTTP_PUT, bind(&ServerManager::handle_setName,
142         ↪ this));
143     server.on("/setStreamUrl", HTTP_PUT, bind(&ServerManager::
144         ↪ handle_setStreamUrl, this));
145     server.on("/setVolume", HTTP_PUT, bind(&ServerManager::
146         ↪ handle_setVolume, this));
147     server.on("/setWifiCredentials", HTTP_POST, bind(&ServerManager::
148         ↪ handle_setWifiCredentials, this));
149     server.on("/pauseStream", HTTP_POST, bind(&ServerManager::
150         ↪ handle_pauseStream, this));
151     server.on("/continueStream", HTTP_POST, bind(&ServerManager::
152         ↪ handle_continueStream, this));
153     server.onNotFound(bind(&ServerManager::handleNotFound, this));
154     running = true;
155     return true;
156 }
157
158 bool ServerManager::stop(){
159     server.stop();
160     running = false;
161     return true;
162 }

```

```

154
155 void ServerManager::handleClient(){
156     server.handleClient();
157 }
158
159 bool ServerManager::isRunning(){
160     return running;
161 }
```

ServerManager.h

```

1 #ifndef ServerManager_H
2 #define ServerManager_H
3
4 #include "Arduino.h"
5 #include "WebServer.h"
6 #include "constants.h"
7 #include "NetworkManager.h"
8 #include "BatteryManager.h"
9 #include "ArduinoJson.h"
10 #include "AudioManager.h"
11 #include "MemoryManager.h"
12
13 class ServerManager{
14     private:
15         static ServerManager* instance;
16         ServerManager();
17         ~ServerManager();
18         ServerManager(const ServerManager*) = delete;
19         ServerManager& operator = (const ServerManager&) = delete;
20         WebServer server;
21         NetworkManager* network;
22         BatteryManager* battery;
23         AudioManager* audio;
24         MemoryManager* memory;
25         bool running;
26
27     /**
28      * creates a json, filed with info about the adapter
29      *
30      * @return info info about the adapter as a serialized json
31      */
32     String getInfo();
33
34     /**
35      * handles a get request to the standard / route

```

```
36     */
37     void handle_get();
38
39     /**
40      * handles a put request to the /getInfo route
41      */
42     void handle_getInfo();
43
44     /**
45      * handles a get request to the /getAvailableNetworks route
46      */
47     void handle_getAvailableNetworks();
48
49     /**
50      * handles a get request to the /getLogs route
51      */
52     void handle_getLogs();
53
54     /**
55      * handles a post request to the /setWifiCredentials route
56      */
57     void handle_setWifiCredentials();
58
59     /**
60      * handles a post request to the /setStreamUrl route
61      */
62     void handle_setStreamUrl();
63
64     /**
65      * handles a post request to the /setName route
66      */
67     void handle_setName();
68
69     /**
70      * handles a post request to the /setVolume route
71      */
72     void handle_setVolume();
73
74     /**
75      * handles a post request to the /pauseStream route
76      */
77     void handle_pauseStream();
78
79     /**
80      * handles a post request to the /continueStream route
```

```

81     */
82     void handle_continueStream();
83
84     /**
85      * handles a request to a undefined route
86      */
87     void handle_notFound();
88
89     public:
90         static ServerManager* getInstance();
91
92         /**
93          * starts the webserver
94          * @return if the start process was successful
95          */
96         bool start();
97
98         /**
99          * stops the webserver
100         * @return if the stop process was successful
101         */
102        bool stop();
103
104        /**
105          * handles the clients
106          */
107        void handleClient();
108
109        /**
110          * returns if the wifi credentials are received from the client
111          * @return if webserver received WiFi-credentials from client
112          */
113        bool wlanCredentialsReceived();
114
115        /**
116          * handles if the stream url is received from the client
117          * @return if webserver received Stream-URL from client
118          */
119        bool urlReceived();
120
121        /**
122          * handles if the name is received from the client
123          * @return if webserver received name of microcontroller from
124              ↢ client
125          */
126        bool nameReceived();

```

```

125
126     /**
127      * handles if the volume is received from the client
128      * @return if webserver received volume for audio output from
129      *         ↪ client
130      */
131
132     /**
133      * returns the ssid, which was received from the client
134      * @return WLAN-SSID, which the webserver received from the client
135      *         ↪ , as a String
136      */
137
138     /**
139      * returns the password, which was received from the client
140      * @return WLAN-Password, which the webserver received from the
141      *         ↪ client, as a String
142      */
143
144     /**
145      * returns the url, which was received from the client
146      * @return Stream-URL, which the webserver received from the
147      *         ↪ client, as a String
148      */
149
150     /**
151      * returns the name, which was received from the client
152      * @return name of the microcontroller, which the webserver
153      *         ↪ received from the client, as a String
154      */
155
156     /**
157      * returns the volume, which was received from the client
158      * @return value of the volume which the webserver received from
159      *         ↪ the client, as a int
160      */
161
162
163     /**

```

```
164     * returns if the webserver is running
165     * @return if webserver is running
166     */
167     bool isRunning();
168 };
169 #endif
```

StatusLED.cpp

```
1 #include "StatusLED.h"
2
3 StatusLED* StatusLED::instance = nullptr;
4
5 StatusLED::StatusLED(){
6     //initializing led pins
7     pinMode(LED_RED, OUTPUT);
8     pinMode(LED_GREEN, OUTPUT);
9     pinMode(LED_BLUE, OUTPUT);
10 }
11
12 StatusLED::~StatusLED(){
13     //empty
14 }
15
16 StatusLED* StatusLED::getInstance(){
17     if(instance == nullptr){
18         instance = new StatusLED();
19     }
20     return instance;
21 }
22
23 /**
24 * sets the color of the led to red
25 */
26 void StatusLED::setRed(){
27     digitalWrite(LED_RED, HIGH);
28     digitalWrite(LED_GREEN, LOW);
29     digitalWrite(LED_BLUE, LOW);
30 }
31
32 /**
33 * sets the color of the led to green
34 */
35 void StatusLED::setGreen(){
36     digitalWrite(LED_RED, LOW);
37     digitalWrite(LED_GREEN, HIGH);
```

```

38     digitalWrite(LED_BLUE, LOW);
39 }
40
41 /**
42 * sets the color of the led to blue
43 */
44 void StatusLED::setBlue(){
45     digitalWrite(LED_RED, LOW);
46     digitalWrite(LED_GREEN, LOW);
47     digitalWrite(LED_BLUE, HIGH);
48 }
49
50 /**
51 * sets the led off (no light)
52 */
53 void StatusLED::setOff(){
54     digitalWrite(LED_RED, LOW);
55     digitalWrite(LED_GREEN, LOW);
56     digitalWrite(LED_BLUE, LOW);
57 }
```

StatusLED.h

```

1 #ifndef STATUSLED_H
2 #define STATUSLED_H
3
4 #include <Arduino.h>
5 #include <constants.h>
6
7 /**
8 * manages the state of the connected RGB led
9 */
10 class StatusLED{
11     private:
12         static StatusLED *instance;
13
14     StatusLED();
15     ~StatusLED();
16
17     public:
18         static StatusLED* getInstance();
19
20         /**
21         * sets the color of the led to red
22         */
23         void setRed();
```

```

24
25     /**
26      * sets the color of the led to green
27      */
28     void setGreen();
29
30     /**
31      * sets the color of the led to blue
32      */
33     void setBlue();
34
35     /**
36      * sets the led off (no light)
37      */
38     void setOff();
39 };
40 #endif

```

3.2 Anhang 3.2: Code Smartphone-App

api/AdapterAPI.tsx

```

1 import axios from "axios";
2 import Network from "../types/Network";
3 import AdapterData from "@/types/AdapterData";
4
5 export const AdapterAPI = {
6
7     getUrlFromMac(mac: string): string{
8         let withoutSeperator = mac.replace(":", " ");
9         let uniquePart = withoutSeperator.substring(6, withoutSeperator.
10             ↪ length-1);
11         let url = "http://msa_" + uniquePart + ".local:8080";
12         return url;
13     },
14     /**
15      * get information of adapter via http get-request
16      * @param {string} mac - mac of adapter
17      * @returns {Promise<AdapterData>} - Promise, with Data as AdapterData
18      *   ↪ type (name: string, mac: string, volume: number, battery:
19      *   ↪ number, stationUrl: string)
20     */
21     async getInfo(mac: string): Promise<AdapterData>{
22         const url = this.getUrlFromMac(mac) + "/getInfo";
23         try{

```

```

21         const res = await axios.get(url, {timeout: 2500});
22         return {name: res.data.name, mac: mac, volume: res.data.volume
23             ↪ , battery: res.data.battery, streamUrl: res.data.
24             ↪ stationUrl, connected: true};
25     } catch(err) {
26         throw err;
27     }
28 },
29 async getInfoFromHost(hostName: string): Promise<AdapterData>{
30     const url = hostName + "/getInfo";
31     try{
32         const res = await axios.get(url, {timeout: 2500});
33         return JSON.parse(res.data);
34     } catch(err) {
35         throw err;
36     }
37 },
38 async getAvailableNetworks(mac: string): Promise<Network []>{
39     const url = this.getUrlFromMac(mac) + "/getAvailableNetworks";
40     try{
41         const res = await axios.get(url);
42         return JSON.parse(res.data);
43     } catch(err) {
44         throw err;
45     }
46 },
47 async getPaused(mac: string): Promise<boolean>{
48     const url = this.getUrlFromMac(mac) + "/getPaused";
49     try{
50         const res = await axios.get(url);
51         return JSON.parse(res.data).paused;
52     } catch(err) {
53         throw err;
54     }
55 },
56 async sendConfigData(mac: string, wifiSsid: string, wifiPassword:
57     ↪ string, newAdapterName: string){
58     const url = this.getUrlFromMac(mac) + "/setConfigData";
59     const data = "ssid=" + wifiSsid + "&password=" + wifiPassword + "&
60         ↪ name=" + newAdapterName;
61     return axios.post(url, data);
62 },
63 async sendVolume(mac: string, volume: number){
64     const url = this.getUrlFromMac(mac) + "/setVolume";
65     const data = "volume=" + volume;

```

```

62     try{
63         return axios.put(url, data);
64     } catch(err){
65         throw err;
66     }
67 },
68 async sendStreamUrl(mac: string, streamUrl: string){
69     const url = this.getUrlFromMac(mac) + "/setStreamUrl";
70     const data = "url=" + streamUrl;
71     try{
72         return axios.put(url, data);
73     } catch(err){
74         throw err;
75     }
76 },
77 async sendPauseStream(mac: string){
78     const url = this.getUrlFromMac(mac) + "/pauseStream";
79     try{
80         return axios.post(url);
81     } catch(err){
82         throw err;
83     }
84 },
85 async sendContinueStream(mac: string){
86     const url = this.getUrlFromMac(mac) + "/continueStream";
87     try{
88         return axios.post(url);
89     } catch(err){
90         throw err;
91     }
92 }
93 }
```

api/FirebaseAPI.tsx

```

1 import { initializeApp } from "firebase/app";
2 import { createUserWithEmailAndPassword, signInWithEmailAndPassword ,
3     initializeAuth, getReactNativePersistence, signOut,
4     sendPasswordResetEmail, confirmPasswordReset } from "firebase/auth";
5 import { getFirestore, setDoc, doc, getDoc, onSnapshot } from "firebase/
6     firestore";
7 import User from "../types/User";
8 import Station from "@types/Station";
9 import AsyncStorage from "@react-native-async-storage/async-storage";
10
11 const firebaseConfig = {
```

```

9  apiKey: "AIzaSyBYW16NMGumkvA271lE6VyTszrAR80UDbo",
10 authDomain: "msa-app-dad57.firebaseio.com",
11 projectId: "msa-app-dad57",
12 storageBucket: "msa-app-dad57.firebaseiostorage.app",
13 messagingSenderId: "278556649604",
14 appId: "1:278556649604:web:6eb08d9dc209d160ccbad1",
15 measurementId: "G-WMPLDFTYY2"
16 };
17
18 type Adapter = {
19   name: string,
20   mac: string
21 }
22
23 const app = initializeApp(firebaseConfig);
24 const auth = initializeAuth(app, {persistence: getReactNativePersistence(
25   ↪ AsyncStorage)}));
26 const storage = getFirestore();
27
28 export const Authentication = {
29   async logIn(email: string, password: string): Promise<User>{
30     try{
31       const res = await signInWithEmailAndPassword(auth, email,
32         ↪ password);
33       if(res.user.email === null){
34         throw "email is null";
35       }
36       return {uid: res.user.uid, email: res.user.email};
37     } catch(err){
38       throw err;
39     }
40   },
41   async register(email: string, password: string): Promise<void>{
42     try{
43       await createUserWithEmailAndPassword(auth, email, password);
44     } catch(err) {
45       throw err;
46     }
47   },
48   async logOut(){
49     return signOut(auth);
50   },
51   onAuthChange(callback: (user: User | null) => void){
52     auth.onAuthStateChanged((user) => {

```

```

52     let newUser;
53     if(user !== null && user.uid !== null && user.email !== null){
54         newUser = {uid: user.uid, email: user.email};
55     } else {
56         newUser = null;
57     }
58     callback(newUser);
59 });
60 },
61 onAuthReady(callback: () => void){
62     auth.authStateReady().then(() => callback())
63     .catch(err => {
64         console.error(err);
65     });
66 },
67 getUser(): User | null{
68     const user = auth.currentUser;
69     if(user !== null && user.email !== null){
70         return {uid: user.uid, email: user.email};
71     }
72     return null;
73 },
74 async sendPwResetEmail(email: string){
75     return sendPasswordResetEmail(auth, email);
76 },
77 async confirmPwReset(code: string, newPw: string){
78     return confirmPasswordReset(auth, code, newPw);
79 }
80 }

81
82 export const CloudStorage = {
83     async getAdapterList(): Promise<Adapter []>{
84         if(auth.currentUser !== null){
85             let uid = auth.currentUser.uid;
86             try{
87                 const docName = "user_" + uid;
88                 const res = await getDoc(doc(storage, "adapter", docName))
89                 ↛ ;
90                 const data = res.data();
91                 if(data === undefined || data.adapterList === undefined){
92                     throw "data is undefined";
93                 }
94                 console.log("adapter data:", data.adapterList);
95                 return data.adapterList;
96             } catch(err) {

```

```

96         throw err;
97     }
98 } else {
99     throw "user is null";
100 }
101 },
102 async getStationList(): Promise<Station[]>{
103     if(auth.currentUser !== null){
104         let uid = auth.currentUser.uid;
105         try{
106             const docName = "user_" + uid;
107             const res = await getDoc(doc(storage, "station", docName))
108             ↗ ;
109             const data = res.data();
110             if(data === undefined || data.stationList === undefined){
111                 throw "data is undefined";
112             }
113             return data.stationList;
114         } catch(err) {
115             throw err;
116         }
117     } else {
118         throw "user is null";
119     }
120 },
121 async setAdapterList(newAdapterList: Adapter[]): Promise<void>{
122     if(auth.currentUser !== null){
123         let uid = auth.currentUser.uid;
124         try{
125             const docName = "user_" + uid;
126             const data = {adapterList: newAdapterList};
127             await setDoc(doc(storage, "adapter", docName), data);
128             return
129         } catch(err){
130             throw err;
131         }
132     } else {
133         throw "user is null";
134     }
135 },
136 async setStationList(newStationList: Station[]): Promise<void>{
137     if(auth.currentUser !== null){
138         let uid = auth.currentUser.uid;
139         try{
140             const docName = "user_" + uid;

```

```

140         const data = {stationList: newStationList};
141         await setDoc(doc(storage, "station", docName), data);
142         return
143     } catch(err){
144         throw err;
145     }
146 } else {
147     throw "user is null";
148 }
149 },
150 onAdapterChange(callback: (newAdapterList: Adapter[]) => void){
151     if(auth.currentUser !== null){
152         let uid = auth.currentUser.uid;
153         const docName = "user_" + uid;
154         const document = doc(storage, "adapter", docName);
155         onSnapshot(document, (newDoc) => {
156             const data = newDoc.data();
157             let adapterList = [];
158             if(data !== undefined){
159                 adapterList = data.adapterList;
160             }
161             callback(adapterList);
162         })
163     } else {
164         throw "user is null";
165     }
166 },
167 onStationChange(callback: (newStationList: Station[]) => void){
168     if(auth.currentUser !== null){
169         let uid = auth.currentUser.uid;
170         const docName = "user_" + uid;
171         const document = doc(storage, "station", docName);
172         onSnapshot(document, (newDoc) => {
173             const data = newDoc.data();
174             let stationList = [];
175             if(data !== undefined){
176                 stationList = data.stationList;
177             }
178             callback(stationList);
179         })
180     } else {
181         throw "user is null";
182     }
183 }
184 }
```

api/RadioBrowserAPI.tsx

```
1 import axios from "axios";
2 import Station from "@types/Station";
3 import Country from "../types/Country";
4 import Language from "../types/Language";
5
6 export const RadioBrowserAPI = {
7     async getCountryNames(): Promise<Country []>{
8         try{
9             const res = await axios.get("https://de1.api.radio-browser.
10                 ↪ info/json/countries?order=stationcount&reverse=true&
11                 ↪ limit=50");
12             const countries = res.data;
13             const countryList: Country[] = [];
14             for(let country of countries){
15                 countryList.push({name: country.name, code: country.
16                     ↪ iso_3166_1});
17             }
18             const sortedCountries = countryList.sort((a, b) => {
19                 if(a.name == b.name){
20                     return 0;
21                 } else if(a.name > b.name) {
22                     return 1;
23                 } else {
24                     return -1;
25                 }
26             });
27             return sortedCountries;
28         } catch(err) {
29             throw err;
30         }
31     },
32     async getLanguageNames(): Promise<Language []>{
33         try{
34             const res = await axios.get("https://de1.api.radio-browser.
35                 ↪ info/json/languages?order=stationcount&reverse=true&
36                 ↪ limit=20");
37             const languages = res.data;
38             const languageList: Language[] = [];
39             for(let language of languages){
40                 let oldName = language.name;
41                 let newName = oldName.charAt(0).toUpperCase() + oldName.
42                     ↪ slice(1);
43             }
44         }
45     }
46 }
```

```

37         if(oldName.includes(' ')){
38             let spaceIdx = newName.indexOf(' ');
39             newName = newName.slice(0, spaceIdx) + ' ' + newName.
40             ↪ charAt(spaceIdx+1).toUpperCase() + newName.slice
41             ↪ (spaceIdx+2);
42         }
43         languageList.push({name: newName, code: language.iso_639})
44             ↪ ;
45     }
46     const sortedLanguages = languageList.sort((a, b) => {
47         if(a.name == b.name){
48             return 0;
49         } else if(a.name > b.name) {
50             return 1;
51         } else {
52             return -1;
53         }
54     });
55     return sortedLanguages;
56 } catch(err) {
57     throw err;
58 }
59 },
60 async getStations(countryName: string, languageName: string,
61     ↪ maxStations: number, dontShow: Station[] | null): Promise<
62     ↪ Station[]>{
63     let url = "http://de1.api.radio-browser.info/json/stations/search?
64         ↪ order=clickcount&reverse=true&hidebroken=true&codec=mp3&
65         ↪ limit=" + maxStations;
66     if(languageName !== null && languageName !== "-"){
67         url += "&language=" + languageName.toLowerCase();
68     }
69     if(countryName !== null && languageName !== "-"){
70         url += "&country=" + countryName;
71     }
72     console.log(url);
73     try{
74         const stations = await axios.get(url);
75         const result: Station[] = [];
76         stations.data.forEach((val: any) => {
77             if(dontShow !== null){
78                 let containsUuid = false;
79                 for(let favStation of dontShow){ //check if station is
80                     ↪ already in favourite stations
81                     if(favStation.uuid == val.stationuuid){
82

```

```

74             containsUuid = true;
75         }
76     }
77     if(!containsUuid){
78         const station = {uuid: val.stationuuid, name: val.
79                         ↪ name, iconUrl: val.favicon, url: val.url};
80         result.push(station);
81     }
82     } else {
83         const station = {uuid: val.stationuuid, name: val.name
84                         ↪ , iconUrl: val.favicon, url: val.url};
85         result.push(station);
86     }
87 }
88 return result;
89 }
90 },
91 async getStationInfo(streamUrl: string){
92     const url = "http://de1.api.radio-browser.info/json/stations/byurl
93         ↪ ?url=" + streamUrl;
94     try{
95         const apiRes = await axios.get(url);
96         return apiRes.data[0];
97     } catch(err) {
98         throw err;
99     }
100 }
```

app/index.tsx

```

1 import { useContext } from "react";
2 import { UserContext } from "../context/UserContext";
3 import { Redirect } from "expo-router";
4 import { StyleSheet, Text } from "react-native";
5 import LoadingScreen from "@/components>LoadingScreen";
6 import { SafeAreaView } from "react-native-safe-area-context";
7 import { GlobalStyle } from "@/constants/Style";
8
9 export default function Index(){
10     const { user, available } = useContext(UserContext);
11
12     const style = StyleSheet.create({
13         container: {
```

```

14         alignItems: 'center'
15     }
16 )
17
18 if(available){
19     return( user !== null
20         ? <Redirect href={"/(tabs)/connection"} />
21         : <Redirect href={"/(auth)/login"} />
22     )
23 } else {
24     return(
25         <SafeAreaView style={[GlobalStyle.page, style.container]}>
26             <Text style={GlobalStyle.textBig}>Willkommen in der MSA
27                 ↪ App!</Text>
28             <LoadingScreen text="Lade Daten ..." />
29         </SafeAreaView>
30     )
31 }

```

app/_layout.tsx

```

1 import { Stack } from 'expo-router';
2 import { UserProvider } from '../context/UserContext';
3
4 export default function Layout() {
5     return(
6         <UserProvider>
7             <Stack screenOptions={{
8                 headerShown: false
9             }}>
10                 <Stack.Screen name='index' />
11             </Stack>
12         </UserProvider>
13     )
14 }

```

app/(auth)/login.tsx

```

1 import { useState } from "react";
2 import { Text, TextInput, Button, SafeAreaView, View, StyleSheet } from "
3     ↪ react-native";
4 import { GlobalStyle, Colors } from "@constants/Style";
5 import { router } from "expo-router";
6 import { Authentication } from "../../api/FirebaseAPI";
7 import { MemoryService } from "../../services/MemoryService";

```

```

8  export default function LoginScreen(){
9    const [email, setEmail] = useState("");
10   const [password, setPassword] = useState("");
11   const [errorText, setErrorText] = useState("");
12
13   const style = StyleSheet.create({
14     inputContainer: {
15       alignItems: 'center'
16     }, error: {
17       color: Colors.red
18     },
19     container: {
20       backgroundColor: Colors.grey,
21       width: '80%',
22       alignSelf: 'center',
23       marginTop: 70,
24       padding: 10,
25       borderRadius: 20,
26       alignItems: 'center'
27     },
28     input: {
29       fontSize: 18,
30       borderColor: Colors.lightGrey,
31       borderRadius: 5,
32       borderWidth: 2,
33       width: 200,
34       marginBottom: 20,
35       marginTop: 5,
36       color: Colors.white,
37       textAlign: 'center',
38     }
39   })
40
41   return(
42     <SafeAreaView style={GlobalStyle.page}>
43       <View style={style.container}>
44         <View style={style.inputContainer}>
45           <Text style={GlobalStyle.textBig}>E-Mail:</Text>
46           <TextInput style={style.input} onChangeText={(text) => {setEmail
47             ↪ (text)}}/>
48           <Text style={GlobalStyle.textBig}>Passwort:</Text>
49           <TextInput style={style.input} onChangeText={(text) => {
50             ↪ setPassword(text)}} secureTextEntry/>
51         </View>
52         <Button color={Colors.lightTurquoise} title="Anmelden" onPress={()}

```

```

51     => {
52       Authentication.logIn(email, password).then(res => {
53         MemoryService.setUser({uid: res.uid, email: res.email});
54         router.replace("/(tabs)/connection");
55       }).catch(err => {
56         setErrorText(err.message);
57       })
58     })}/>
59     <Text style={[GlobalStyle.textMedium, style.error]}>{errorText}</
60       <Text style={GlobalStyle.textMedium}>Haben Sie noch kein Konto?</
61         <Text>
62         <Button color={Colors.lightTurquoise} title="Registrieren" onPress
63           =>={() => {
64             router.replace("/register");
65           }}/>
66         </View>
67       </SafeAreaView>
68     )
69   }

```

app/(auth)/register.tsx

```

1 import { useState } from "react";
2 import { Text, TextInput, Button, SafeAreaView, View, StyleSheet } from "
3   react-native";
4 import { GlobalStyle, Colors } from "@/constants/Style";
5 import { router } from "expo-router";
6 import { Authentication } from "../../api/FirebaseAPI";
7
8 export default function RegisterScreen(){
9   const [email, setEmail] = useState("");
10  const [password, setPassword] = useState("");
11  const [errorText, setErrorText] = useState("");
12
13  const style = StyleSheet.create({
14    inputContainer: {
15      alignItems: 'center'
16    },
17    error: {
18      color: Colors.red
19    },
20    container: {
21      backgroundColor: Colors.grey,
22      width: '80%',
23      alignSelf: 'center',
24      marginTop: 70,
25    }
26  }
27}

```

```

23     padding: 10,
24     borderRadius: 20,
25     alignItems: 'center'
26   },
27   input: {
28     fontSize: 18,
29     borderColor: Colors.lightGrey,
30     borderRadius: 5,
31     borderWidth: 2,
32     width: 200,
33     marginBottom: 20,
34     marginTop: 5,
35     color: Colors.white,
36     textAlign: 'center',
37   }
38 })
39
40 return(
41   <SafeAreaView style={GlobalStyle.page}>
42     <View style={style.container}>
43       <View style={style.inputContainer}>
44         <Text style={GlobalStyle.textBig}>E-Mail:</Text>
45         <TextInput style={style.input} onChangeText={(text) => {setEmail
46           ↪ (text)}}/>
47         <Text style={GlobalStyle.textBig}>Passwort:</Text>
48         <TextInput style={style.input} onChangeText={(text) => {
49           ↪ setPassword(text)}} secureTextEntry/>
50       </View>
51       <Button color={Colors.lightTurquoise} title="Registrieren" onPress
52         ↪ ={() => {
53           Authentication.register(email, password).then(() => {
54             console.log("redirecting to login");
55             router.replace("/login");
56           }).catch(err => {
57             setErrorText(err.message);
58           })
59         }}/>
60       <Text style={[GlobalStyle.textMedium, style.error]}>{errorText}</
61         ↪ Text>
62       <Text style={GlobalStyle.textMedium}>Haben Sie bereits ein Konto
63         ↪ ?</Text>
64       <Button color={Colors.lightTurquoise} title="Anmelden" onPress={() =
65         ↪ => {
66           router.replace("/login");
67         }}/>

```

```
62     </View>
63   </SafeAreaView>
64 )
65 }
```

app/(auth)/_layout.tsx

```
1 import { Stack } from 'expo-router';
2
3 export default function Layout() {
4   return(
5     <Stack>
6       <Stack.Screen name='index' />
7       <Stack.Screen name='login' />
8       <Stack.Screen name='register' />
9     </Stack>
10   )
11 }
```

app/(tabs)/_layout.tsx

```
1 import FontAwesome from "@expo/vector-icons/FontAwesome";
2 import MaterialIcons from "@expo/vector-icons/MaterialIcons";
3 import { Tabs } from "expo-router";
4 import { Colors } from "@/constants/Style";
5 import MaterialCommunityIcons from "@expo/vector-icons/
6   ↪ MaterialCommunityIcons";
6 import { StationProvider } from "@/context/StationContext";
7 import { AdapterProvider } from "@/context/AdapterContext";
8 import { UserProvider } from "@/context/UserContext";
9
10 export default function TabLayout() {
11   return (
12     <UserProvider>
13       <AdapterProvider>
14         <StationProvider>
15           <Tabs
16             screenOptions={{
17               tabBarActiveTintColor: Colors.darkTurquoise,
18               tabBarStyle: { backgroundColor: Colors.grey },
19               tabBarInactiveTintColor: Colors.white,
20               headerShown: false,
21             }}
22           >
23             <Tabs.Screen
24               name="connection"
25               options={{


```

```

26         title: "Verbindungen",
27         tabBarIcon: ({ color }) => (
28             <FontAwesome name="chain" size={28} color={color} />
29         ),
30     )}
31   />
32   <Tabs.Screen
33     name="adapter"
34     options={{
35       title: "Adapter",
36       tabBarIcon: ({ color }) => (
37           <MaterialIcons size={28} name="speaker-group" color={
38             ↗ color} />
39         ),
40     )}
41   />
42   <Tabs.Screen
43     name="music"
44     options={{
45       title: "Musik",
46       tabBarIcon: ({ color }) => (
47           <MaterialIcons size={28} name="library-music" color={
48             ↗ color} />
49         ),
50     )}
51   />
52   <Tabs.Screen
53     name="profile"
54     options={{
55       title: "Profil",
56       tabBarIcon: ({ color }) => (
57           <MaterialCommunityIcons
58             name="account-box"
59             size={28}
60             color={color}
61           />
62         ),
63     )}
64   />
65   </Tabs>
66   </StationProvider>
67   </AdapterProvider>
68   </UserProvider>
69 );
70 }

```

app/(tabs)/adapter/addAdapter.tsx

```
1 import { SafeAreaView, Button, StyleSheet } from "react-native";
2 import { GlobalStyle, Colors } from "@constants/Style";
3 import { router } from "expo-router";
4
5 export default function AddAdapter(){
6     const style = StyleSheet.create({
7         container: {
8             justifyContent: 'center'
9         }
10    })
11    return(
12        <SafeAreaView style={[GlobalStyle.page, style.container]}>
13            <Button color={Colors.lightTurquoise} title="Neuen Adapter
14                ↪ hinzufuegen" onPress={() => router.push("/(tabs)/adapter
15                ↪ /addNewAdapter")}/>
16            <Button color={Colors.lightTurquoise} title="Bestehenden
17                ↪ Adapter hinzufuegen" onPress={() => router.push("/(tabs)
18                ↪ /adapter/addExistingAdapter")}/>
19        </SafeAreaView>
20    )
21}
22
```

app/(tabs)/adapter/addExistingAdapter.tsx

```
1 import { SafeAreaView, Text, TextInput, Button } from "react-native";
2 import { GlobalStyle } from "@constants/Style";
3 import { useContext, useState } from "react";
4 import { AdapterAPI } from "@api/AdapterAPI";
5 import { AdapterContext } from "@context/AdapterContext";
6 import AdapterData from "@types/AdapterData";
7 import { CloudStorage } from "@api/FirebaseAPI";
8
9 export default function AddExistingAdapter(){
10     const [mac, setMac] = useState("");
11     const { adapterList } = useContext(AdapterContext);
12     return(
13         <SafeAreaView style={GlobalStyle.page}>
14             <Text>Mac:</Text>
15             <TextInput value={mac} onChangeText={(text) => setMac(text)}/>
16             <Button title="Suche!" onPress={() => {
17                 AdapterAPI.getInfo(mac).then(res => {
18                     let newAdapterList = [... adapterList],
```

```

19         let adapter: AdapterData = {name: res.name, mac,
20             ↪ volume: res.volume, battery: res.battery,
21             ↪ streamUrl: res.streamUrl, connected: true}
22     newAdapterList.push(adapter);
23     CloudStorage.setAdapterList(newAdapterList);
24   }).catch(err => {
25     alert("Adapter kann nicht gefunden werden! Versuche es
26           ↪ erneut!")
27   })
28 }

```

app/(tabs)/adapter/addNewAdapter.tsx

```

1 import { useEffect, useState } from "react";
2 import { Text, View, Button, SafeAreaView, TextInput } from "react-native"
3   ↪ ";
4 import { StyleSheet } from "react-native";
5 import ErrorScreen from "@/components/ErrorScreen";
6 import LoadingScreen from "@/components>LoadingScreen";
7 import TextInputWindow from "@/components/TextInputWindow";
8 import { AdapterAPI } from "@api/AdapterAPI";
9 import { GlobalStyle, Colors } from "@constants/Style";
10 import Network from "@types/Network";
11 import NetworkList from "@components/NetworkList";
12 import AdapterData from "@types/AdapterData";
13
14 export default function AddNewAdapter(){
15   const [isReachable, setReachable] = useState(false);
16   const [loading, setLoading] = useState(true);
17   const [adapter, setAdapter] = useState<AdapterData|null>(null);
18   const [networkList, setNetworkList] = useState<Network[]|null>(null);
19   const [selectedSsid, setSelectedSsid] = useState("");
20   const [name, setName] = useState("");
21
22   const host = "http://192.168.0.1:8080";
23
24   useEffect(() => {
25     setLoading(true);
26     AdapterAPI.getInfoFromHost(host).then((res) => {
27       setAdapter({name: res.name, mac: res.mac, battery: res.battery
28           ↪ , volume: res.volume, connected: false, streamUrl: res.
29             ↪ streamUrl});
30     setLoading(false);
31   })
32 }

```

```

28         setReachable(true);
29         AdapterAPI.getAvailableNetworks(host).then(res => {
30             setNetworkList(res);
31         })
32     }).catch(err => {
33         setLoading(false);
34         console.error(err);
35         setReachable(false);
36     })
37 }, []);
38
39 const style = StyleSheet.create({
40     container: {
41         alignSelf: 'center'
42     },
43     container2: {
44         flexDirection: 'row'
45     },
46     icon: {
47         marginLeft: 10
48     },
49     listContainer: {
50         height: '30%',
51     }
52 })
53
54 if(loading){
55     return(
56         <SafeAreaView style={GlobalStyle.page}>
57             <LoadingScreen text="Versuche Adapter zu erreichen..."/>
58         </SafeAreaView>
59     )
60 } else {
61     if(isReachable && (adapter !== null) && (networkList !== null)){
62         return(
63             <SafeAreaView style={GlobalStyle.page}>
64                 <View style={style.container2}>
65                     <Text style={GlobalStyle.textBig}>{"Name: " +
66                         ↪ adapter.name}</Text>
67                     <TextInput value={adapter.name} onChangeText={(text) => {setName(text)}}/>
68                 </View>
69                 <Text style={GlobalStyle.textBig}>{"Mac: " + adapter.
70                         ↪ mac}</Text>
71                 <Text style={GlobalStyle.textBig}>Mit WLAN verbinden
72             </SafeAreaView>
73     )
74 }
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169

```

```

    ↵ : </Text>
  70   <View style={style.listContainer}>
  71     <NetworkList networks={networkList} onItemSelected
  72       ↵ ={(item: Network) => setSelectedSsid(item.
  73         ↵ ssid)}>
  74   </View>
  75   <Button title="Adapter hinzufuegen" color={Colors.
  76     ↵ lightTurquoise}>
  77     {selectedSsid.length > 0 &&
  78       <TextInputWindow text={"Passwort fuer " +
  79         ↵ selectedSsid + " eingeben:"} isPassword={true}
  80         onEnter={(password: string) => {alert(password)}}
  81         onCancel={() => {setSelectedSsid("")}}>
  82       }
  83     </SafeAreaView>
  84   )
  85 } else {
  86   return(
  87     <SafeAreaView style={GlobalStyle.page}>
  88       <ErrorScreen errorText="Adapter nicht erreichbar.
  89         ↵ Versichere dich, dass du mit dem WLAN des
  90         ↵ Adapters verbunden bist!" buttonText="Nochmal
  91         ↵ Versuchen" onButtonPress={() => {
  92           console.error("function not available!");
  93         }}>
  94     </SafeAreaView>
  95   )
  96 }
  97 }
  98 }
```

app/(tabs)/adapter/index.tsx

```

1 import { SafeAreaView } from "react-native";
2 import AdapterList from "@/components/AdapterList";
3 import { GlobalStyle } from "@/constants/Style";
4 import { useContext } from "react";
5 import { AdapterContext } from "@/context/AdapterContext";
6 import AdapterData from "@/types/AdapterData";
7 import { CloudStorage } from "@/api/FirebaseAPI";
8
9 export default function AdapterScreen(){
10   const { adapterList } = useContext(AdapterContext);
11
12   function deleteAdapter(selectedAdapter: AdapterData){
```

```

13     let newAdapterList = [];
14     for(let adapter of adapterList){
15         if(!(selectedAdapter.mac == adapter.mac)){
16             newAdapterList.push(adapter);
17         }
18     }
19     CloudStorage.setAdapterList(newAdapterList);
20 }
21
22 return(
23     <SafeAreaView style={GlobalStyle.page}>
24         <AdapterList adapterList={adapterList} editable
25             ↪ showOnlyAvailable={false} onItemSelected={() => {}}
26             ↪ onDeleteAdapter={(adapter: AdapterData) => {
27                 ↪ deleteAdapter(adapter)}}/>
28     </SafeAreaView>
29 )
30
31 }

```

app/(tabs)/adapter/_layout.tsx

```

1 import { Stack } from 'expo-router';
2 import { Colors } from '@constants/Style';
3
4 export default function Layout() {
5     return (
6         <Stack screenOptions={{
7             headerStyle: {backgroundColor: Colors.grey},
8             headerTitleStyle: {color: Colors.white}
9         }>
10         <Stack.Screen name='index' options={{headerTitle: 'Adapter'}}/>
11         <Stack.Screen name='addAdapter' options={{headerTitle: 'Adapter
12             ↪ hinzufuegen'}}/>
13         <Stack.Screen name='addNewAdapter' options={{headerTitle: 'Neuen
14             ↪ Adapter hinzufuegen'}}/>
15         <Stack.Screen name='addExistingAdapter' options={{headerTitle: 'Bestehenden Adapter hinzufuegen'}}/>
16     </Stack>
17 );
18 }

```

app/(tabs)/connection/addConnection.tsx

```

1 import { Button, SafeAreaView, Text } from "react-native";
2 import { GlobalStyle, Colors } from "@constants/Style";
3 import { useState, useEffect, useContext } from "react";
4 import AdapterList from "@components/AdapterList";

```

```

5 import Station from "@/types/Station";
6 import { router } from "expo-router";
7 import StationList from "@/components/StationList";
8 import { AdapterAPI } from "@/api/AdapterAPI";
9 import AdapterData from "@/types/AdapterData";
10 import { AdapterContext } from "@/context/AdapterContext";
11
12 export default function AddConnection(){
13     const [selectedAdapter, setSelectedAdapter] = useState<AdapterData | null>(null);
14     const [selectedStation, setSelectedStation] = useState<Station | null>(null);
15     const [buttonDisabled, setButtonDisabled] = useState(true);
16     const { adapterList } = useContext(AdapterContext);
17
18     useEffect(() => {
19         if(selectedAdapter === null || selectedStation === null){
20             setButtonDisabled(true);
21         } else {
22             setButtonDisabled(false);
23         }
24     }, [selectedAdapter, selectedStation]);
25
26     return(
27         <SafeAreaView style={GlobalStyle.page}>
28             <Text style={GlobalStyle.textBig}>Adapter auswaehlen:</Text>
29             <AdapterList adapterList={adapterList} editable={false}
30                 &gt; showOnlyAvailable onItemSelect={(item: AdapterData) => {
31                     setSelectedAdapter(item)}}
32                     &gt; onDeleteAdapter={()=>{}}>
33             <Text style={GlobalStyle.textBig}>Station auswaehlen:</Text>
34             <StationList editable={false} onItemSelect={(item: Station) =>
35                 &gt; setSelectedStation(item)}>
36             <Button title="Bestaetigen" disabled={buttonDisabled} color={
37                 &gt; Colors.lightTurquoise} onPress={() => {
38                     if((selectedAdapter !== null) && (selectedStation !== null)
39                         &gt; )) {
40                         AdapterAPI.sendStreamUrl(selectedAdapter.name,
41                             &gt; selectedStation.url).then(() => {
42                             router.back();
43                         })
44                     }
45                 }>
46             </SafeAreaView>
47         )
48     }

```

app/(tabs)/connection/index.tsx

```
1 import { SafeAreaView } from "react-native"
2 import ConnectionList from "@/components/ConnectionList";
3 import { GlobalStyle } from "@/constants/Style";
4 import { useContext } from "react";
5 import { AdapterContext } from "@/context/AdapterContext";
6 import { useState, useEffect } from "react";
7 import { RadioBrowserAPI } from "@/api/RadioBrowserAPI";
8 import Connection from "@/types/Connection";
9
10 export default function ConnectionScreen(){
11     const [connectionList, setConnectionList] = useState<Connection[]>([])
12     ↪ ;
13     const { adapterList } = useContext(AdapterContext);
14
15     useEffect(() => {
16         for(let adapter of adapterList){
17             if(adapter.connected && adapter.streamUrl.length > 0){
18                 let newConnectionList: Connection[] = [];
19                 let stationInfo = RadioBrowserAPI.getStationInfo(adapter.
20                     ↪ streamUrl);
21                 let connection: Connection = {adapter: adapter, station: {
22                     ↪ name: stationInfo.name, uuid: stationInfo.uuid, url:
23                     ↪ stationInfo.url, iconUrl: stationInfo.favicon},
24                     ↪ paused: true};
25                 newConnectionList.push(connection);
26                 setConnectionList(newConnectionList);
27             }
28         }
29     }, [adapterList]);
30
31     return (
32         <SafeAreaView style={GlobalStyle.page}>
33             <ConnectionList connectionList={connectionList} onItemPress
34                 ↪ ={()=>{}}/>
35         </SafeAreaView>
36     );
37 }
```

app/(tabs)/connection/_layout.tsx

```
1 import { Stack } from 'expo-router';
2 import { Colors } from '@/constants/Style';
3
```

```

4 export default function Layout() {
5   return (
6     <Stack screenOptions={{{
7       headerStyle: {backgroundColor: Colors.grey},
8       headerTitleStyle: {color: Colors.white},
9       headerBackTitle: "Zurueck"
10      }}}>
11      <Stack.Screen name='index' options={{headerTitle: 'Verbindungen
12        ↪ '}}/>
13      <Stack.Screen name='addConnection' options={{headerTitle: 'Verbindung hinzufuegen'}}/>
14    </Stack>
15  );
}

```

app/(tabs)/music/favouriteStationSelect.tsx

```

1 import { useEffect, useState, useContext } from "react";
2 import { FlatList, StyleSheet, Pressable, SafeAreaView } from "react-
3   ↪ native";
4 import { Colors, GlobalStyle } from "@constants/Style";
5 import Station from "@types/Station";
6 import { router, useLocalSearchParams } from "expo-router";
7 import AntDesign from '@expo/vector-icons/AntDesign';
8 import StationItem from "@components/StationItem";
9 import { RadioBrowserAPI } from "@api/RadioBrowserAPI";
10 import { StationContext } from "@context/StationContext";
11 import { CloudStorage } from "@api/FirebaseAPI";
12
13 export default function Radios(){
14   const [stations, setStations] = useState(Array());
15   const [selectedStations, setSelectedStations] = useState(Array());
16   const maxStations = 50;
17   const {countryName, languageName} = useLocalSearchParams();
18   const { stationList } = useContext(StationContext);
19
20   function handleStationPress(station: Station){
21     let newSelectedStations = [... selectedStations];
22     if(newSelectedStations.includes(station)){
23       const idx = newSelectedStations.indexOf(station);
24       newSelectedStations.splice(idx, 1);
25     } else {
26       newSelectedStations.push(station);
27     }
28     let selectedNames: string[] = [];
29     newSelectedStations.map((val) => {
30       selectedNames.push(val.name);
31     });
32     setSelectedStations(newSelectedStations);
33   }
34
35   return (
36     <SafeAreaView style={GlobalStyle.safeArea}>
37       <FlatList
38         data={stationList}
39         keyExtractor={(item) => item.id}
40         renderItem={({item}) =>
41           <StationItem
42             station={item}
43             handleStationPress={handleStationPress}
44           />
45         }
46       </FlatList>
47     </SafeAreaView>
48   );
}

```

```

29         selectedNames.push(val.name);
30     })
31     console.log(selectedNames);
32     setSelectedStations([... newSelectedStations]);
33   }
34
35   function isSelected(station: Station){
36     let selectedUuids: string[] = [];
37     selectedStations.forEach((val) => {
38       selectedUuids.push(val.uuid);
39     })
40     let selected = selectedUuids.includes(station.uuid);
41     return selected;
42   }
43
44   const style = StyleSheet.create({
45     list: {
46       height: '90%'
47     },
48     icon: {
49       marginTop: 10,
50       marginRight: 20,
51       alignSelf: 'flex-end'
52     }
53   })
54
55   useEffect(()=>{
56     if(typeof countryName === "string" && typeof languageName === "
57       ↪ string"){
58       RadioBrowserAPI.getStations(countryName, languageName,
59         ↪ maxStations, stationList).then(res =>{
60           console.log(res);
61           if(res != null){
62             setStations(res);
63           }
64         }).catch(err => {
65           console.error(err);
66         })
67     }
68   }, []);
69
70   return (
71     <SafeAreaView style={GlobalStyle.page}>
72       <FlatList style={style.list} data={stations} renderItem={({
73         ↪ item}) =>

```

```

71             <Pressable onPress={() => handleStationPress(item)}>
72                 <StationItem station={item} selected={isSelected(item)}
73                     ↪ />
74             </Pressable>
75         }/>
76         <Pressable onPress={() => {
77             let newStationList;
78             if(stationList != null){
79                 newStationList = stationList.concat(selectedStations);
80             } else {
81                 newStationList = selectedStations;
82             }
83             console.log("new stationlist:", newStationList);
84             try{
85                 CloudStorage.setStationList(newStationList).then(() =>
86                     ↪ {
87                         router.replace("/music");
88                     }
89                 )catch(err){
90                     console.error(err);
91                 }
92             }})
93         <AntDesign style={style.icon} name="check" size={50} color
94             ↪ ={Colors.lightTurquoise}/>
95         </Pressable>
      </SafeAreaView>
    )
}

```

app/(tabs)/music/index.tsx

```

1 import { SafeAreaView } from "react-native";
2 import { GlobalStyle } from "@constants/Style";
3 import StationList from "@components/StationList";
4
5 export default function MusicScreen(){
6     return(
7         <SafeAreaView style={GlobalStyle.page}>
8             <StationList editable onItemSelected={() => {}}/>
9         </SafeAreaView>
10    )
11}

```

app/(tabs)/music/radiosearch.tsx

```

1 import { useEffect, useState } from "react";
2 import { ScrollView, Button } from "react-native";

```



```

45         setSelectedCountryName(systemCountry.name);
46     }
47   }
48   if(languageDataset !== null){
49     let systemLanguage = languageDataset.find(language => language
50       ↪ .code == systemLanguageCode);
51     if(systemLanguage !== undefined){
52       setSelectedLanguageName(systemLanguage.name);
53     }
54   }, [countryDataset, languageDataset]);
55
56   if(countryDataset !== null && languageDataset !== null){
57     return(
58       <SafeAreaView style={GlobalStyle.page}>
59         <ScrollView>
60           <Picker onValueChange={(countryName: string) => {
61             ↪ setSelectedCountryName(countryName)}}
62             ↪ selectedValue={selectedCountryName}>
63             <Item key={"-"} value={"-"} label="-" color={
64               ↪ Colors.white}/>
65             {countryDataset.map((country, idx) =>
66               <Item key={idx} value={country.name} label={
67                 ↪ country.name} color={Colors.white}/>
68             ))}
69           </Picker>
70           <Picker onValueChange={(languageName: string) => {
71             ↪ setSelectedLanguageName(languageName)}}
72             ↪ selectedValue={selectedLanguageName}>
73             <Item key={"-"} value={"-"} label="-" color={
74               ↪ Colors.white}/>
75             {languageDataset.map((language, idx) =>
76               <Item key={idx} value={language.name} label={
77                 ↪ language.name} color={Colors.white}/>
78             ))}
79           </Picker>
80           <Button color={Colors.lightTurquoise} title="Search !"
81             ↪ onPress={() => {
82               router.push({pathname: "/(tabs)/music/
83                 ↪ favouriteStationSelect", params: {
84                   ↪ countryName: selectedCountryName,
85                   ↪ languageName: selectedLanguageName}})}
86             }}/>
87         </ScrollView>
88       </SafeAreaView>

```

```
77     )
78   }
79 }
```

app/(tabs)/music/_layout.tsx

```
1 import { Stack } from 'expo-router';
2 import { Colors } from '@constants/Style';
3
4 export default function Layout() {
5   return (
6     <Stack screenOptions={{
7       headerStyle: {backgroundColor: Colors.grey},
8       headerTitleStyle: {color: Colors.white}
9     }}>
10    <Stack.Screen name='index' options={{headerTitle: 'Stationen'}}/>
11    <Stack.Screen name='favouriteStationSelect' options={{headerTitle:
12      ↪ 'Stationen auswaehlen'}}/>
13    <Stack.Screen name='radiosearch' options={{headerTitle: 'Stationen
14      ↪ filtern'}}/>
15  </Stack>
)
```

app/(tabs)/profile/index.tsx

```
1 import { useContext } from "react";
2 import { Text, Button, SafeAreaView, StyleSheet } from "react-native";
3 import { GlobalStyle, Colors } from "@constants/Style";
4 import { UserContext } from "@context/UserContext";
5 import { Authentication } from "@api/FirebaseAPI";
6 import { router } from "expo-router";
7
8 export default function ProfileScreen(){
9   const { user } = useContext(UserContext);
10
11   const style = StyleSheet.create({
12     inputContainer: {
13       alignItems: 'center'
14     }, error: {
15       color: Colors.red
16     }
17   })
18
19   if(user !== null){
20     return(
21       <SafeAreaView style={[GlobalStyle.page]}>
```

```

22         <Text>{"Email: " + user.email}</Text>
23         <Button title="Abmelden" color={Colors.lightTurquoise} onPress={()}
24             ↪     => {
25                 Authentication.logOut().then(()=> router.replace("/"));
26             }}/>
27         </SafeAreaView>
28     )
29 }
```

app/(tabs)/profile/_layout.tsx

```

1 import { Stack } from 'expo-router';
2 import { Colors } from '@constants/Style';
3
4 export default function Layout() {
5     return (
6         <Stack screenOptions={{{
7             headerStyle: {backgroundColor: Colors.grey},
8             headerTitleStyle: {color: Colors.white}
9         }}}>
10            <Stack.Screen name='index' options={{headerTitle: 'Profil'}}/>
11        </Stack>
12    );
13 }
```

components/AdapterItem.tsx

```

1 import { Text, View } from "react-native";
2 import Ionicons from '@expo/vector-icons/Ionicons';
3 import { StyleSheet } from "react-native";
4 import {Colors, GlobalStyle} from "@constants/Style";
5 import Adapter from "../types/AdapterData";
6 import BatteryIndicator from "./BatteryIndicator";
7
8 type Props = {
9     adapter: Adapter,
10    selected: boolean,
11    reachable: boolean
12};
13
14 const style = StyleSheet.create({
15     icon: {
16         width: 50,
17         height: 50,
18     },
19     container1: {
```

```

20     flexDirection: 'row',
21     justifyContent: 'space-between',
22     alignItems: 'center',
23     backgroundColor: Colors.white,
24     borderColor: Colors.black,
25     borderWidth: 1,
26     borderRadius: 10,
27     padding: 10,
28     marginBottom: 7
29   },
30   container2: {
31     flexDirection: 'row',
32     justifyContent: 'space-between',
33     alignContent: 'space-between',
34     width: '20%'
35   }
36 )
37 export default function AdapterItem({adapter, selected, reachable}: Props)
38   {
39     let backgroundColor = "lightgrey";
40     if(selected){
41       backgroundColor = Colors.lightTurquoise;
42     } else if(reachable){
43       backgroundColor = Colors.grey;
44     } else {
45       backgroundColor = "lightgrey";
46     }
47
48     return (
49       <View style={[style.container1, {backgroundColor: backgroundColor}]}>
50         <View>
51           <Text style={GlobalStyle.textBig}>{adapter.name}</Text>
52           <Text style={GlobalStyle.textMedium}>{adapter.mac}</Text>
53         </View>
54         {reachable
55           ?
56             <BatteryIndicator batteryPercentage={adapter.battery}/>
57             : <Ionicons name="cloud-offline" size={24} color={Colors.white}/>
58           }
59       </View>
60     );
61   }

```

components/AdapterList.tsx

```

1 import { useState } from "react";

```

```

2 import { View, FlatList, StyleSheet, Pressable } from "react-native";
3 import { router } from "expo-router";
4 import ErrorScreen from "@/components/ErrorScreen";
5 import DeleteButton from "./DeleteButton";
6 import AddToListButton from "./AddToListButton";
7 import AdapterItem from "./AdapterItem";
8 import { Alert } from "react-native";
9 import AdapterData from "@/types/AdapterData";
10
11 type Props = {
12   adapterList: AdapterData[];
13   onItemSelected: Function;
14   onDeleteAdapter: Function;
15   editable: boolean;
16   showOnlyAvailable: boolean;
17 };
18
19 export default function AdapterList({
20   adapterList,
21   onItemSelected,
22   onDeleteAdapter,
23   editable,
24   showOnlyAvailable,
25 }: Props) {
26   const [selectedAdapter, setSelectedAdapter] = useState<AdapterData | null>(null);
27
28   function handleItemPress(item: AdapterData) {
29     if (selectedAdapter !== null && selectedAdapter.mac == item.mac) {
30       setSelectedAdapter(null);
31       onItemSelected(null);
32     } else {
33       setSelectedAdapter(item);
34       onItemSelected(item);
35     }
36   }
37
38   function handleDeletePress() {
39     if (selectedAdapter !== null) {
40       Alert.alert(
41         "Adapter loeschen",
42         "Wollen Sie den Adapter '" +
43           selectedAdapter.name +
44           "' wirklich loeschen?",
45         [

```

```

46     {
47         text: "Nein",
48         onPress: () => {
49             setSelectedAdapter(null);
50         },
51     },
52     {
53         text: "Ja",
54         onPress: () => {
55             onDeleteAdapter(selectedAdapter);
56         },
57     },
58 ]
59 );
60 }
61 }

62 function isSelected(item: AdapterData) {
63     if (selectedAdapter !== null && selectedAdapter.mac == item.mac) {
64         if ((showOnlyAvailable && item.connected) || !showOnlyAvailable) {
65             return true;
66         }
67     }
68 }
69 return false;
70 }

71 const style = StyleSheet.create({
72     container: {
73         width: "95%",
74         alignSelf: "center",
75     },
76     icon: {
77         alignSelf: "flex-start",
78     },
79     iconContainer: {
80         flexDirection: "row",
81         width: "95%",
82         justifyContent: "space-between",
83         alignSelf: "center",
84     },
85 },
86 });
87

88 if (adapterList.length > 0) {
89     return (
90         <View style={style.container}>

```

```

91      <FlatList
92        data={adapterList}
93        renderItem={({ item }) => (
94          <Pressable
95            onPress={() => {
96              handleItemPress(item);
97            }}
98          >
99            <AdapterItem adapter={item} selected={isSelected(item)}
100             ↪ reachable={item.connected}/>
101          </Pressable>
102        )}
103      {editable && (
104        <View style={style.iconContainer}>
105          <AddToListButton
106            onPress={() => router.push("/:tabs)/adapter/addAdapter")}
107          />
108          {selectedAdapter !== null && (
109            <DeleteButton
110              onPress={() => {
111                handleDeletePress();
112              }}
113            />
114          )}
115        </View>
116      )})
117    </View>
118  );
119 } else {
120   if(showOnlyAvailable) {
121     return (
122       <ErrorScreen
123         errorText="Kein Adapter verfuegbar!"
124         buttonText="Neuen Adapter hinzufuegen"
125         onButtonPress={() => router.push("/:tabs)/adapter/addAdapter")}
126         />
127     );
128   } else {
129     return (
130       <ErrorScreen
131         errorText="Du hast noch keine Adapter hinzugefuegt!"
132         buttonText="Adapter hinzufuegen"
133         onButtonPress={() => router.push("/:tabs)/adapter/addAdapter")}
134       />

```

```
135     );
136   }
137 }
138 }
```

components/AddToListButton.tsx

```
1 import { Pressable } from "react-native";
2 import Entypo from "@expo/vector-icons/Entypo";
3 import { Colors } from "@/constants/Style";
4
5 type Props = {
6   onPress: Function
7 }
8
9 export default function AddToListButton({onPress}: Props){
10   return (
11     <Pressable style={{alignSelf: 'flex-start'}} onPress={() =>
12       ↪ onPress()}
13       <Entypo name="add-to-list" size={30} color={Colors.
14         ↪ lightTurquoise} />
15     </Pressable>
16   )
17 }
18 }
```

components/BatteryIndicator.tsx

```
1 import { Text, View } from "react-native";
2 import FontAwesome from '@expo/vector-icons/FontAwesome';
3 import Ionicons from '@expo/vector-icons/Ionicons';
4 import { GlobalStyle, Colors } from "@/constants/Style";
5
6 type Props = {
7   batteryPercentage: number
8 };
9
10 type IconNameType = "battery-empty" | "battery-full" | "battery-three-
11   ↪ quarters" | "battery-half" | "battery-quarter";
12
13 export default function BatteryIndicator({batteryPercentage}: Props){
14   let iconName: IconNameType;
15   if(batteryPercentage > 75){
16     iconName = "battery-full";
17   } else if(batteryPercentage > 50){
18     iconName = "battery-three-quarters";
19   } else if(batteryPercentage > 25){
     iconName = "battery-half";
```

```

20     } else if(batteryPercentage > 0){
21         iconName = "battery-quarter";
22     } else {
23         iconName = "battery-empty";
24     }
25
26     if(batteryPercentage > 0){
27         return(
28             <View>
29                 <FontAwesome name={iconName} size={24} color={Colors.white
30                     ↪ }/>
31                 <Text style={GlobalStyle.textMedium}>{batteryPercentage +
32                     ↪ "%"}</Text>
33             </View>
34         )
35     } else {
36         return(
37             <Ionicons name="battery-charging" size={24} color={Colors.
38                 ↪ white} />
39         )
40     }
41 }

```

components/ConnectionItem.tsx

```

1 import { View, Pressable } from "react-native";
2 import AntDesign from '@expo/vector-icons/AntDesign';
3 import { StyleSheet } from "react-native";
4 import { Colors } from "@constants/Style";
5 import AdapterItem from "./AdapterItem";
6 import StationItem from "./StationItem";
7 import Connection from "../types/Connection";
8 import PlayPauseButton from "./PlayPauseButton";
9 import VolumeSelector from "./VolumeSelector";
10 import { AdapterAPI } from "@api/AdapterAPI";
11
12 type Props = {
13     connection: Connection
14 };
15
16 const style = StyleSheet.create({
17     container: {
18         flexDirection: 'column',
19         justifyContent: 'space-between',
20         width: '100%',
21         backgroundColor: Colors.lightGrey,

```

```

22     padding: 10,
23     borderRadius: 20,
24     marginBottom: 7
25   },
26   controlElementContainer: {
27     alignItems: 'center'
28   },
29   xButton: {
30     alignSelf: 'flex-end',
31     paddingBottom: 15
32   },
33 }
34
35 export default function ConnectionItem({connection}: Props) {
36   function endConnection(){
37     AdapterAPI.sendPauseStream(connection.adapter.mac).then(() => {
38       AdapterAPI.sendStreamUrl(connection.adapter.mac, "");
39     })
40   }
41
42   return (
43     <View style={style.container}>
44       <Pressable style={style.xButton} onPress={() => endConnection()}>
45         <AntDesign name="disconnect" size={24} color={Colors.
46           ↪ lightTurquoise} />
47       </Pressable>
48       <AdapterItem adapter={connection.adapter} selected={false} reachable
49         ↪ ={true}/>
50       <StationItem station={connection.station} selected={false}/>
51       <View style={style.controlElementContainer}>
52         <PlayPauseButton paused={connection.paused} onPress={() => {}}/>
53         <VolumeSelector initVolumePercentage={connection.adapter.volume}
54           ↪ onValueChange={(val: number) => {AdapterAPI.sendVolume(
55             ↪ connection.adapter.name, val)}}/>
56       </View>
57     </View>
58   );
59 }

```

components/ConnectionList.tsx

```

1 import { View, FlatList, StyleSheet, Pressable } from "react-native";
2 import { GlobalStyle } from "@/constants/Style";
3 import { router } from "expo-router";
4 import ErrorScreen from "@/components/ErrorScreen";
5 import { SafeAreaView } from "react-native-safe-area-context";

```

```

6 import AddToListButton from "./AddToListButton";
7 import ConnectionItem from "./ConnectionItem";
8 import Connection from "@types/Connection";
9
10 type Props = {
11   connectionList: Connection[];
12   onItemPress: Function;
13 };
14
15 export default function ConnectionList({ connectionList, onItemPress }: 
16   ↪ Props) {
17   const style = StyleSheet.create({
18     container: {
19       width: "95%",
20       alignSelf: "center",
21     },
22     icon: {
23       alignSelf: "flex-start",
24     },
25   });
26
27   if (connectionList.length > 0) {
28     return (
29       <View style={style.container}>
30         <FlatList
31           data={connectionList}
32           renderItem={({ item }) => (
33             <Pressable onPress={() => onItemPress(item)}>
34               <ConnectionItem
35                 connection={item}
36               />
37               </Pressable>
38             )}
39           </FlatList>
40           <AddToListButton
41             onPress={() => router.push("/(tabs)/connection/addConnection")}
42             />
43         </View>
44     );
45   } else {
46     return (
47       <SafeAreaView style={GlobalStyle.page}>
48         <ErrorScreen
49           errorText="Es sind zurzeit keine Verbindungen vorhanden!"
50           buttonText="Verbindung erstellen"
51         </ErrorScreen>
52     );
53   }
54 }

```

```

50         onButtonPress={() => router.push("/(tabs)/connection/
      ↪ addConnection")}
51     />
52   </SafeAreaView>
53 );
54 }
55 }
```

components/DeleteButton.tsx

```

1 import { Pressable, StyleSheet } from "react-native"
2 import FontAwesome from "@expo/vector-icons/FontAwesome"
3 import { Colors } from "@constants/Style"
4
5 type Props = {
6   onPress: Function
7 }
8
9 export default function DeleteButton({onPress}: Props){
10   return (
11     <Pressable style={{alignSelf: 'flex-end'}} onPress={() => {onPress
      ↪ ()}}>
12       <FontAwesome name="trash-o" size={30} color={Colors.red}/>
13     </Pressable>
14   )
15 }
```

components/ErrorScreen.tsx

```

1 import { Colors, GlobalStyle } from "@constants/Style";
2 import { View, Text, Button, StyleSheet } from "react-native";
3
4 type Props = {
5   errorText: string,
6   buttonText: string,
7   onButtonPress: Function
8 }
9
10 const style = StyleSheet.create({
11   container: {
12     height: '70%',
13     justifyContent: 'center',
14     alignItems: 'center',
15   },
16   text: {
17     textAlign: 'center',
18   }
19 }
```

```

19 })
20
21 export default function ErrorScreen({errorText, buttonText, onButtonPress
22   ↪ }: Props){
23   return(
24     <View style={style.container}>
25       <Text style={[GlobalStyle.textBig, style.text]}>{errorText}</
26         ↪ Text>
27       <Button color={Colors.lightTurquoise} title={buttonText}
28         ↪ onPress={() => onButtonPress()}>
29     </View>
30   )
31 }

```

components/LoadingScreen.tsx

```

1 import { GlobalStyle } from "@/constants/Style";
2 import { ActivityIndicator, Text, View, StyleSheet } from "react-native";
3 import { Colors } from "react-native/Libraries/NewAppScreen";
4
5 type Props = {
6   text: string
7 }
8
9 const style = StyleSheet.create({
10   container: {
11     flex: 1,
12     justifyContent: 'center',
13     alignItems: 'center'
14   }
15 })
16
17 export default function LoadingScreen({text}: Props){
18   return (
19     <View style={style.container}>
20       <ActivityIndicator size="large" color={Colors.white}/>
21       <Text style={GlobalStyle.textMedium}>{text}</Text>
22     </View>
23   )
24 }

```

components/NetworkItem.tsx

```

1 import { Text, StyleSheet, View } from "react-native";
2 import { Colors, GlobalStyle } from '@/constants/Style';
3 import MaterialIcons from '@expo/vector-icons/MaterialIcons';
4

```

```

5 type Props = {
6   ssid: string,
7   rssi: number,
8   selected: boolean
9 };
10
11 const style = StyleSheet.create({
12   container: {
13     flexDirection: 'row',
14     justifyContent: 'space-between',
15     alignItems: 'center',
16     borderWidth: 1,
17     borderRadius: 10,
18     padding: 10,
19     marginBottom: 7,
20   }
21 })
22
23 function getWifiItem(rssi :number){
24   if(rssi > -50){
25     return "network-wifi";
26   } else if(rssi > -60){
27     return "network-wifi-3-bar";
28   } else if(rssi > -70){
29     return "network-wifi-2-bar";
30   } else {
31     return "network-wifi-1-bar";
32   }
33 }
34
35 export default function NetworkItem({ssid, rssi, selected}: Props) {
36   return (
37     <View style={[style.container, {backgroundColor: selected ? Colors.
38       ↪ lightTurquoise : Colors.grey}]}>
39       <Text style={GlobalStyle.textMedium}>{ssid}</Text>
40       <MaterialIcons name={getWifiItem(rssi)} size={24} color={Colors.
41         ↪ white}/>
42     </View>
43   );
44 }

```

components/NetworkList.tsx

```

1 import { useState } from "react";
2 import { View, FlatList, StyleSheet, Pressable } from "react-native";
3 import Network from "@types/Network";

```

```

4 import NetworkItem from "./NetworkItem";
5
6 type Props = {
7   networks: Network[],
8   onItemSelected: Function
9 }
10
11 export default function NetworkList({networks, onItemSelected}: Props){
12   const [selectedNetwork, setSelectedNetwork] = useState<Network|null>(
13     null);
14
15   const style = StyleSheet.create({
16     container: {
17       width: '95%',
18       alignSelf: 'center'
19     }
20 )
21
22   return(
23     <View style={style.container}>
24       <FlatList data={networks} renderItem={({item}) =>
25         <Pressable onPress={() => {
26           setSelectedNetwork(item);
27           onItemSelected(item);
28         }}>
29           <NetworkItem ssid={item.ssid} rssi={item.rssi}
30             selected={(selectedNetwork !== null) && (item.
31               ssid == selectedNetwork.ssid)}/>
32         </Pressable>
33       }/>
34     </View>
35   )
36 }

```

components/PlayPauseButton.tsx

```

1 import { Pressable } from "react-native";
2 import AntDesign from '@expo/vector-icons/AntDesign';
3 import { Colors } from "@constants/Style";
4 import Entypo from '@expo/vector-icons/Entypo';
5
6 type Props = {
7   paused: boolean,
8   onPress: Function
9 }
10

```

```

11 export default function PlayPauseButton({paused, onPress}: Props) {
12   return (
13     <Pressable onPress={() => onPress()}>
14       { paused
15         ? <Entypo name="controller-play" size={30} color={Colors.
16           ↪ lightTurquoise}>/>
17         : <AntDesign name="pause" size={30} color={Colors.
18           ↪ lightTurquoise}>/>
19       }
20     </Pressable>
21   )
22 }

```

components/StationItem.tsx

```

1 import { Text, View, Image, StyleSheet, Pressable } from "react-native";
2 import { Colors, GlobalStyle } from '@constants/Style';
3 import Station from "../types/Station";
4
5 type Props = {
6   station: Station,
7   selected: boolean
8 };
9
10 const style = StyleSheet.create({
11   icon: {
12     width: 50,
13     height: 50,
14   },
15   container: {
16     flexDirection: 'row',
17     justifyContent: 'space-between',
18     alignItems: 'center',
19     borderColor: Colors.black,
20     borderWidth: 1,
21     borderRadius: 10,
22     padding: 5,
23     marginBottom: 7
24   }
25 })
26
27 export default function StationItem({station, selected}: Props) {
28   return (
29     <View style={[style.container, {backgroundColor: selected ? Colors.
30       ↪ lightTurquoise : Colors.grey}]}>
31       <Text style={GlobalStyle.textBig}>{station.name}</Text>

```

```

31         <Image source={{uri: station.iconUrl}} style={style.icon}/>
32     </View>
33   );
34 }

```

components/StationList.tsx

```

1 import { useContext, useState } from "react";
2 import { View, FlatList, StyleSheet, Pressable } from "react-native";
3 import { GlobalStyle } from "@constants/Style";
4 import StationItem from "@components/StationItem";
5 import Station from "@types/Station";
6 import { router } from "expo-router";
7 import ErrorScreen from "@components/ErrorScreen";
8 import { SafeAreaView } from "react-native-safe-area-context";
9 import DeleteButton from "./DeleteButton";
10 import AddToListButton from "./AddToListButton";
11 import { Alert } from "react-native";
12 import { StationContext } from "@context/StationContext";
13 import { CloudStorage } from "@api/FirebaseAPI";
14
15 type Props = {
16   onItemSelected: Function;
17   editable: boolean;
18 };
19
20 export default function StationList({ onItemSelected, editable }: Props) {
21   const { stationList } = useContext(StationContext);
22   const [selectedStation, setSelectedStation] = useState<Station | null>(
23     null);
24
25   function handleItemPress(item: Station) {
26     if (selectedStation !== null && selectedStation.uuid == item.uuid) {
27       setSelectedStation(null);
28       onItemSelected(null);
29     } else {
30       setSelectedStation(item);
31       onItemSelected(item);
32     }
33
34   function deleteItem() {
35     if(selectedStation !== null && stationList.length > 0) {
36       let newStationList = [... stationList];
37       for(let i = 0; i < newStationList.length; i++){
38         if(newStationList[i].uuid == selectedStation.uuid){

```

```

39         newStationList.splice(i, 1);
40         break;
41     }
42 }
43 if(newStationList.length !== 0){
44     CloudStorage.setStationList(newStationList);
45 } else{
46     CloudStorage.setStationList([]);
47 }
48 }
49 }

50
51 function handleDeletePress() {
52     if (selectedStation !== null) {
53         Alert.alert(
54             "Station loeschen",
55             "Wollen Sie die Station '" +
56             selectedStation.name +
57             "' wirklich loeschen?",
58             [
59                 {
60                     text: "Nein",
61                     onPress: () => {
62                         setSelectedStation(null);
63                     },
64                 },
65                 {
66                     text: "Ja",
67                     onPress: () => {
68                         deleteItem();
69                     },
70                 },
71             ]
72         );
73     }
74 }

75
76 const style = StyleSheet.create({
77     container: {
78         width: "95%",
79         alignSelf: "center",
80         marginTop: 20
81     },
82     icon: {
83         alignSelf: "flex-start",

```

```

84 },
85 iconContainer: {
86   flexDirection: "row",
87   width: "95%",
88   justifyContent: "space-between",
89   alignSelf: "center",
90 },
91 );
92
93 if(stationList.length > 0) {
94   return (
95     <View style={style.container}>
96       <FlatList
97         data={stationList}
98         renderItem={({ item }) => (
99           <Pressable
100             onPress={() => {
101               handleItemPress(item);
102             }}
103           >
104             <StationItem
105               station={item}
106               selected={
107                 selectedStation !== null && selectedStation.uuid == item
108                   .→ .uuid
109               }
110             />
111             </Pressable>
112           )}
113         />
114         {editable && (
115           <View style={style.iconContainer}>
116             <AddToListButton
117               onPress={() =>
118                 router.push("/(tabs)/music/radiosearch", {
119                   relativeToDirectory: true,
120                 })
121               }
122             />
123             {selectedStation !== null && (
124               <DeleteButton
125                 onPress={() => {
126                   handleDeletePress();
127                 }}
128               />

```

```

128         )}
129     </View>
130   )}
131 </View>
132 );
133 } else {
134   return (
135     <SafeAreaView style={GlobalStyle.page}>
136       <ErrorScreen
137         errorText="Du hast noch keine Stationen hinzugefuegt!"
138         buttonText="Station hinzufuegen"
139         onButtonPress={() => router.push("/(tabs)/music/radiosearch")}
140       />
141     </SafeAreaView>
142   );
143 }
144 }
```

components/TextInputWindow.tsx

```

1 import { useState } from "react";
2 import { Text, Button, View, TextInput, StyleSheet } from "react-native";
3 import { Colors, GlobalStyle } from "@constants/Style";
4
5 type Props = {
6   text: string,
7   isPassword: boolean,
8   onEnter: Function,
9   onCancel: Function
10 }
11
12 const style = StyleSheet.create({
13   container: {
14     backgroundColor: Colors.grey,
15     position: 'absolute',
16     zIndex: 2,
17     padding: 20,
18     alignSelf: 'center',
19     borderRadius: 10,
20     marginTop: 50
21   },
22   input: {
23     borderColor: Colors.white,
24     borderWidth: 0.2,
25     marginTop: 20,
26     color: Colors.white
27   }
28 }
```

```

27 },
28 container2: {
29   flexDirection: 'row',
30   marginTop: 20
31 }
32 )
33
34 export default function TextInputWindow({text, isPassword, onEnter,
35   ↪ onCancel}: Props){
36   const [password, setPassword] = useState("");
37   return(
38     <View style={style.container}>
39       <Text style={GlobalStyle.textMedium}>{text}</Text>
40       <TextInput style={style.input} value={password} onChangeText
41         ↪ ={(text) => setPassword(text)} secureTextEntry={
42           ↪ isPassword}/>
43       <View style={style.container2}>
44         <Button color={Colors.lightTurquoise} title="Abbrechen"
45           ↪ onPress={() => {onCancel()}}/>
46         <Button color={Colors.lightTurquoise} title="Bestaetigen"
47           ↪ onPress={() => {onEnter(password)}}/>
48       </View>
49     </View>
50   )
51 }

```

components/VolumeSelector.tsx

```

1 import { Text, View, Button } from "react-native";
2 import { GlobalStyle, Colors } from "@constants/Style";
3 import Slider from "@react-native-community/slider";
4 import { useState } from "react";
5 import { StyleSheet } from "react-native";
6
7 type Props = {
8   initVolumePercentage: number,
9   onValueChange: Function
10 };
11
12 const style = StyleSheet.create({
13   container: {
14     alignItems: 'center'
15   },
16   innerContainer: {
17     flexDirection: 'row'
18 }

```

```

19 })
20
21 export default function VolumeSelector({initVolumePercentage ,
22   ↪ onValueChange}: Props){
23   const [volume, setVolume] = useState(initVolumePercentage);
24   return(
25     <View style={style.container}>
26       <View style={style.innerContainer}>
27         <Button title="-" color={Colors.lightTurquoise}
28           onPress={() => {if(volume > 0) {
29             setVolume(volume-1)
30             onValueChange(volume)
31           }}}
32         />
33         <Slider
34           minimumValue={0}
35           maximumValue={100}
36           step={1}
37           value={volume}
38           onSlidingComplete={(val) => onValueChange(volume)}
39           onValueChange={(val) => {setVolume(val)}}
40           vertical={true}
41           thumbTintColor={Colors.white}
42           style={{width: '50%'}}
43           minimumTrackTintColor={Colors.lightTurquoise}
44           maximumTrackTintColor={Colors.lightTurquoise}
45         />
46         <Button title"+" color={Colors.lightTurquoise}
47           onPress={() => {if(volume < 100) {
48             setVolume(volume+1)
49             onValueChange(volume)
50           }}}
51         />
52       </View>
53       <Text style={GlobalStyle.textBig}>{volume + "%"}</Text>
54     )
55   }

```

components/WifiItem.tsx

```

1 import { Text, View, StyleSheet } from "react-native";
2 import {Colors} from "@constants/Style";
3
4 type Props = {
5   ssid: string,

```

```

6   rssi: number,
7   selected: boolean
8 };
9
10 const style = StyleSheet.create(
11   {
12     icon: {
13       width: 50,
14       height: 50,
15     },
16     container: {
17       flexDirection: 'row',
18       justifyContent: 'space-between',
19       alignItems: 'center',
20       backgroundColor: Colors.white,
21       borderColor: Colors.black,
22       borderWidth: 1,
23       borderRadius: 10,
24       padding: 10,
25       marginBottom: 7
26     }
27   }
28 );
29
30 export default function WifiItem({ssid, rssi, selected}: Props) {
31   return (
32     <View style={[style.container, {backgroundColor: selected ? Colors.
33       ↪ lightTurquoise : Colors.white}]}>
34       <Text>{ssid}</Text>
35       <Text>{rssi}</Text>
36     </View>
37   );
}

```

components/navigation/TabBarIcon.tsx

```

1 // You can explore the built-in icon families and icons on the web at
2   ↪ https://icons.expo.fyi/
3
4 import Ionicons from '@expo/vector-icons/Ionicons';
5 import { type IconProps } from '@expo/vector-icons/build/createIconSet';
6 import { type ComponentProps } from 'react';
7
8 export function TabBarIcon({ style, ...rest }: IconProps<ComponentProps<
9   ↪ typeof Ionicons>['name']>) {
10   return <Ionicons size={28} style={[{ marginBottom: -3 }, style]} {...
```

```
9     ↪ rest} />;
 }
```

components/_tests_/ThemedText-test.tsx

```
1 import * as React from 'react';
2 import renderer from 'react-test-renderer';
3
4 import { ThemedText } from '../ThemedText';
5
6 it(`renders correctly`, () => {
7   const tree = renderer.create(<ThemedText>Snapshot test!</ThemedText>).
8     ↪ toJSON();
9
10  expect(tree).toMatchSnapshot();
11});
```

components/_tests_/_snapshots_/ThemedText-test.tsx.snap

```
// Jest Snapshot v1, https://goo.gl/fbAQLP
1
2
3 exports[`renders correctly 1`] = `

<Text
4   style={
5     [
6       {
7         "color": "#11181C",
8       },
9       {
10         "fontSize": 16,
11         "lineHeight": 24,
12       },
13       undefined,
14       undefined,
15       undefined,
16       undefined,
17       undefined,
18       undefined,
19     ]
20   }
21 >
22   Snapshot test!
23 </Text>
24 `;
```

context/AdapterContext.tsx

```
1 import {
```

```

2  createContext,
3  useContext,
4  useState,
5  useEffect,
6  ReactNode,
7 } from "react";
8 import { CloudStorage } from "../api/FirebaseAPI";
9 import { AdapterAPI } from "@/api/AdapterAPI";
10 import { UserContext } from "./UserContext";
11 import AdapterData from "@/types/AdapterData";
12
13 type Props = {
14   children: ReactNode;
15 };
16
17 type AdapterContextType = {
18   adapterList: AdapterData[];
19 };
20
21 const defaultContext: AdapterContextType = {
22   adapterList: [],
23 };
24
25 export const AdapterContext = createContext<AdapterContextType>(
26   ↗ defaultContext);
27
28 export const AdapterProvider = ({ children }: Props) => {
29   const { user } = useContext(UserContext);
30   const [adapterList, setAdapterList] = useState<AdapterData[]>(
31     defaultContext.adapterList
32   );
33
34   function requestAdapters() {
35     if (adapterList !== null) {
36       let newAdapterList: AdapterData[] = [];
37       let promiseList = [];
38       for(let adapter of adapterList) {
39         let promise = AdapterAPI.getInfo(adapter.mac);
40         promiseList.push(promise);
41       }
42       Promise.allSettled(promiseList).then((results) => {
43         for (let result of results) {
44           if (result.status == "fulfilled") {
45             let val = result.value;
46             let newAdapter = {
47               mac: adapter.mac,
48               name: adapter.name,
49               type: adapter.type,
50               status: result.status,
51               value: val
52             }
53             newAdapterList.push(newAdapter);
54           }
55         }
56       }
57     }
58   }
59
60   return (
61     <AdapterContext.Provider value={{ user, adapterList, setAdapterList }}>
62       {children}
63     </AdapterContext.Provider>
64   );
65 }

```

```

46         name: val.name,
47         mac: val.mac,
48         battery: val.battery,
49         volume: val.volume,
50         streamUrl: val.streamUrl,
51         connected: true,
52     };
53     newAdapterList.push(newAdapter);
54 }
55 );
56 for(let adapter of adapterList) {
57     let containsMac = false;
58     for(let newAdapter of newAdapterList){
59         if(newAdapter.mac == adapter.mac){
60             containsMac = true;
61             break;
62         }
63     }
64     if(!containsMac){
65         let newAdapter = {
66             name: adapter.name,
67             mac: adapter.mac,
68             battery: 0,
69             volume: 0,
70             streamUrl: "",
71             connected: false,
72         };
73         newAdapterList.push(newAdapter);
74     }
75 }
76 setAdapterList(newAdapterList);
77 }
78 }
79 }

80 useEffect(() => {
81     let intervalId = 0;
82     if (user !== null) {
83         CloudStorage.onAdapterChange((newAdapterList) => {
84             for(let newAdapter of newAdapterList){
85                 let containsMac = false;
86                 for(let adapter of adapterList){
87                     if(adapter.mac == newAdapter.mac){
88                         containsMac = true;
89                         break;
90                     }
91                 }
92             }
93         })
94     }
95 }

```

```

91         }
92     }
93     if(!containsMac){
94         let newAdapters = [... adapterList];
95         let adapter: AdapterData = {name: newAdapter.name, mac:
96             ↪ newAdapter.mac, volume: 0, battery: 0, streamUrl: "", 
97             ↪ connected: false};
98         newAdapters.push(adapter);
99         setAdapterList(newAdapters);
100    }
101    requestAdapters();
102  }
103 } else {
104   console.log("user is null");
105 }
106 return () => clearInterval(intervalId);
107 }, [user]);
108
109 return (
110   <AdapterContext.Provider value={{ adapterList }}>
111     {children}
112   </AdapterContext.Provider>
113 );
114 };

```

context/StationContext.tsx

```

1 import { createContext, useContext, useState, useEffect, ReactNode } from
2   ↪ "react";
3 import { CloudStorage } from "../api/FirebaseAPI";
4 import Station from "@types/Station";
5 import { UserContext } from "./UserContext";
6
7 type Props = {
8   children: ReactNode;
9 };
10
11 type StationContextType = {
12   stationList: Station[]
13 };
14
15 const defaultContext = {
16   stationList: []
17 };

```

```

17
18 export const StationContext = createContext<StationContextType>(
19   ↪ defaultContext);
20
21
22 export const StationProvider = ({children}: Props) => {
23   const { user } = useContext(UserContext);
24   const [stationList, setStationList] = useState<Station[]>(
25     ↪ defaultContext.stationList);
26
27   useEffect(() => {
28     if(user !== null){
29       CloudStorage.onStationChange((newStationList: Station[]) => {
30         setStationList(newStationList);
31       })
32     }
33   }, [user]);
34
35   return(
36     <StationContext.Provider value={{stationList}}>
37       {children}
38     </StationContext.Provider>
39   )
40 }

```

context/SystemDataContext.tsx

```

1 import { createContext, useState, useEffect, ReactNode } from "react";
2 import { SystemService } from "../services/SystemService";
3
4 type Props = {
5   children: ReactNode;
6 };
7
8 type SystemDataContextType = {
9   connectedToInternet: boolean
10 };
11
12 const defaultCenter = {
13   connectedToInternet: false
14 };
15
16 export const SystemDataContext = createContext<SystemDataContextType>(
17   ↪ defaultCenter);
18
19 export const SystemDataProvider = ({children}: Props) => {
20   const [connectedToInternet, setConnectedToInternet] = useState(false);
21
22   useEffect(() => {
23     CloudStorage.onConnectedToInternetChange((isConnected) => {
24       setConnectedToInternet(isConnected);
25     })
26   }, []);
27
28   return(
29     <SystemDataContext.Provider value={{connectedToInternet}}>
30       {children}
31     </SystemDataContext.Provider>
32   )
33 }

```

```

20
21     useEffect(() => {
22         SystemService.isConnectedToInternet().then(res => {
23             setConnectedToInternet(res);
24         }).catch(err => {
25             console.error(err);
26         })
27     }, []);
28
29     return (
30         <SystemDataContext.Provider value={{connectedToInternet}}>
31             {children}
32         </SystemDataContext.Provider>
33     )
34 }
```

context/UserContext.tsx

```

1 import { createContext, useState, useEffect, ReactNode } from "react";
2 import { Authentication } from "../api/FirebaseAPI";
3 import User from "../types/User";
4
5 type Props = {
6     children: ReactNode;
7 };
8
9 type UserContextType = {
10     user: User | null,
11     available: boolean
12 };
13
14 const defaultCenter = {
15     user: null,
16     available: false
17 };
18
19 export const UserContext = createContext<UserContextType>(defaultContext);
20
21 export const UserProvider = ({children}: Props) => {
22     const [user, setUser] = useState<User | null>(defaultContext.user);
23     const [available, setAvailable] = useState(defaultContext.available);
24
25     useEffect(() => {
26         Authentication.onAuthChange((newUser) => {
27             console.log("auth changed");
28             console.log("user:", newUser);
29         })
30     });
31 }
32
33 
```

```

29         setUser(newUser);
30         setAvailable(true);
31     })
32 }, []);
33
34 return (
35     <UserContext.Provider value={{user, available}}>
36         {children}
37     </UserContext.Provider>
38 )
39 }

```

types/AdapterData.ts

```

1 type AdapterData = {
2     name: string,
3     mac: string
4     volume: number;
5     battery: number;
6     streamUrl: string;
7     connected: boolean;
8 }
9
10 export default AdapterData;

```

types/Connection.ts

```

1 import Station from "./Station";
2 import AdapterData from "./AdapterData";
3
4 type Connection = {
5     adapter: AdapterData;
6     station: Station;
7     paused: boolean;
8 }
9
10 export default Connection;

```

types/Country.ts

```

1 type Country = {
2     name: string,
3     code: string
4 }
5
6 export default Country;

```

types/Language.ts

```
1 type Language = {
2     name: string,
3     code: string
4 }
5
6 export default Language;
```

types/Network.ts

```
1 type Network = {
2     ssid: string,
3     rssi: number
4 }
5
6 export default Network;
```

types/Station.ts

```
1 type Station = {
2     uuid: string;
3     name: string;
4     iconUrl: string;
5     url: string;
6 }
7
8 export default Station;
```

types/User.ts

```
1 type User = {
2     uid: string,
3     email: string
4 }
5
6 export default User;
```

types/UserData.ts

```
1 import Station from "./Station"
2 import AdapterData from "./AdapterData";
3
4 type UserData = {
5     adapterList: AdapterData[] | null,
6     stationList: Station[] | null
7 }
8
9 export default UserData;
```


4 Anhang 4: Zeichnung Gehäuse Adapter

