

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	XII
2	Anhang 2: UML-Klassendiagramm Adapter	XIII
3	Anhang 3: Code	XIV
3.1	Anhang 3.1: Code Adapter	XIV
3.2	Anhang 3.2: Code Smartphone-App	XLIX

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:	
<ul style="list-style-type: none">- 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: <ul style="list-style-type: none">- 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:	
<ul style="list-style-type: none">- 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:	
<ul style="list-style-type: none">- 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:	
<ul style="list-style-type: none">- 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:	
<ul style="list-style-type: none">- 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 Immler	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 Immler	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 Immler	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 Immler	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 designen 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:	
-	

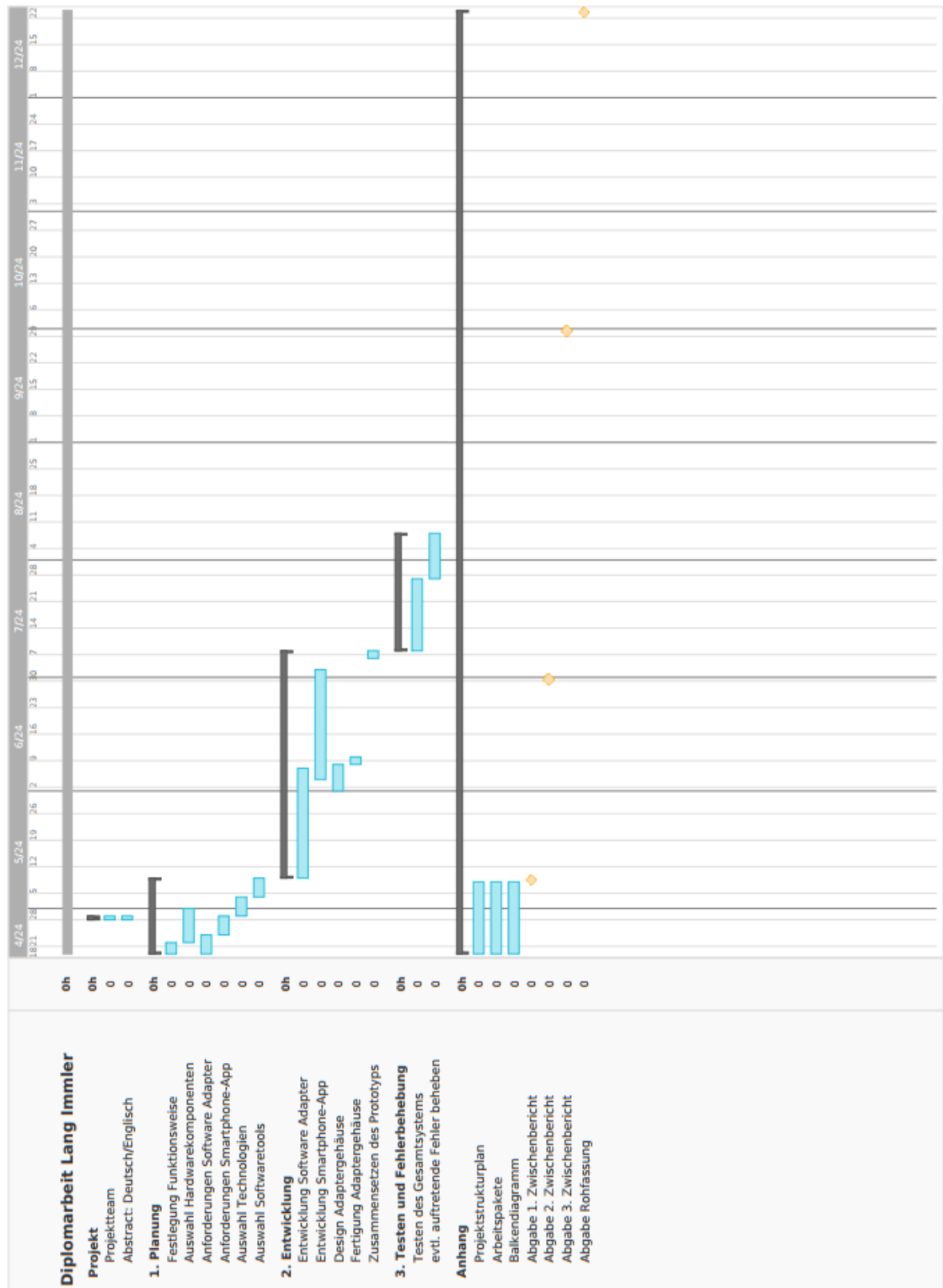
3.1 Testen des Gesamtsystems	
AP3.1: Testen auf Fehler im Gesamtsystem	
Übernommen von:	Zu erledigen bis:
Nico Lang	26.07.2024
Zu erledigen/Durchführung/Ziel/Ergebnis:	
- Test der groben Funktionsweise des Gesamtsystems	

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 	

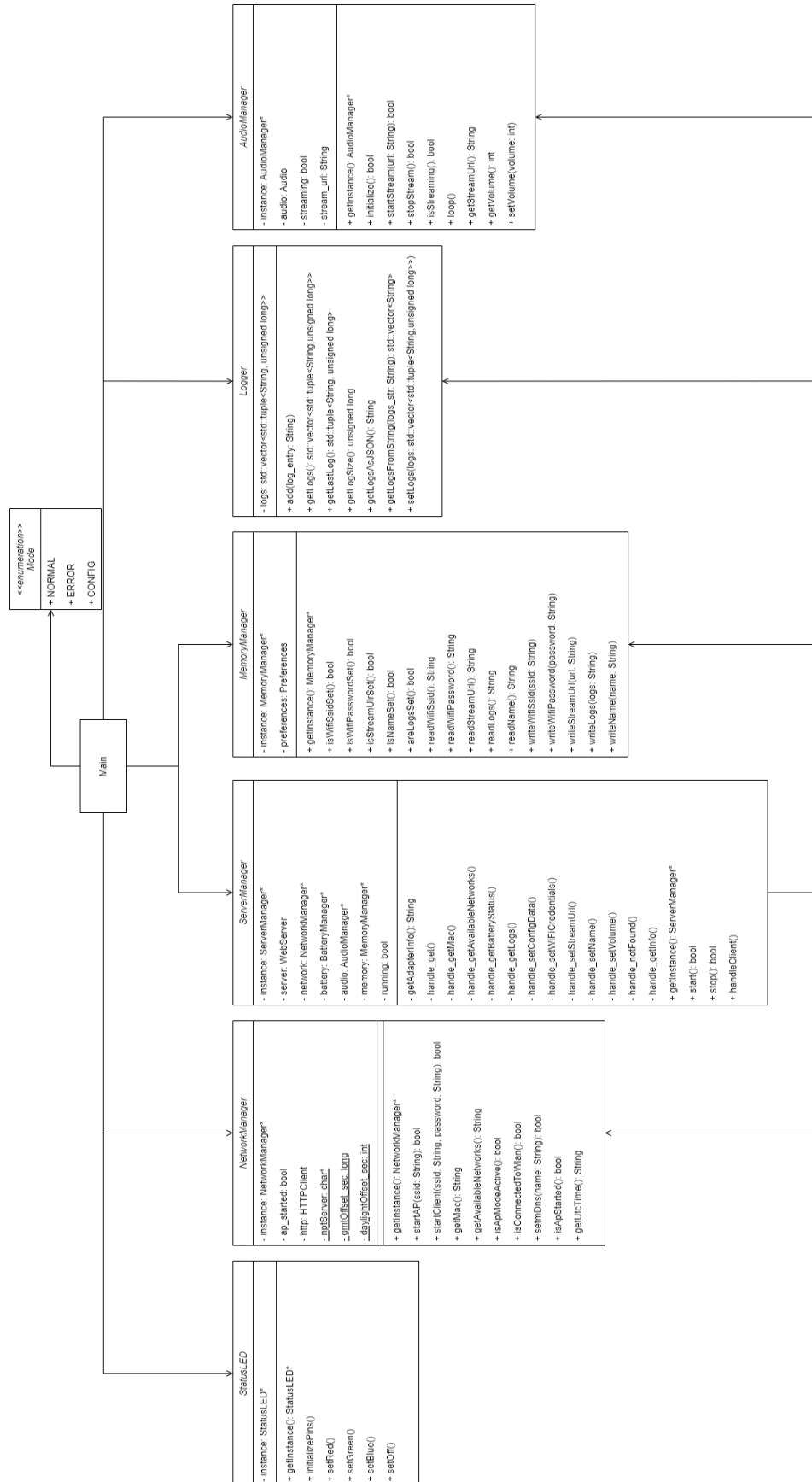
1.2 Anhang 1.2: Projektstrukturplan



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
   ↪ *string) //for debugging
6  {
7      const char *ptr = reinterpret_cast<const char *>(cbData);
8      (void) isUnicode;
9      char s1[32], s2[64];
10     strncpy_P(s1, type, sizeof(s1));
11     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
    ↪ debugging
19 {
20     const char *ptr = reinterpret_cast<const char *>(cbData);
21     // Note that the string may be in PROGMEM, so copy it to RAM for printf
22     char s1[64];
23     strncpy_P(s1, string, sizeof(s1));
24     s1[sizeof(s1)-1]=0;
25     Serial.printf("STATUS(%s) '%d' = '%s'\n", ptr, code, s1);
26     Serial.flush();
27 }
28
29 AudioManager::AudioManager(){
30     stream_url = "";
31     streaming = false;
32     volume = 100;
33     audioLogger = &Serial;
34     src = new AudioFileSourceICYStream();
35     src->RegisterMetadataCB(MDCallback, (void*)"ICY");
36     buff = new AudioFileSourceBuffer(src, AUDIO_BUFFERSIZE);
37     buff->RegisterStatusCB(StatusCallback, (void*)"buffer");
38     out = new AudioOutputI2S();
39     out->SetPinout(I2S_BCLK_PIN, I2S_LRC_PIN, I2S_DOUT_PIN);
40     out->SetBitsPerSample(AUDIO_BITSPERSAMPLE);
41     out->SetChannels(AUDIO_CHANNELS);
42     out->SetRate(AUDIO_SAMPLERATE);
43     gen = new AudioGeneratorMP3();
44     gen->RegisterStatusCB(StatusCallback, (void*)"mp3");
45 }
46
47 AudioManager* AudioManager::getInstance(){
48     if(instance == nullptr){
49         instance = new AudioManager();
50     }
51     return instance;
52 }
53
54 void AudioManager::setStreamUrl(String url){
55     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
31 }
```

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 rssi = 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");

```

```

129         if(!network->setmDns(name)){
130             Logger::add("mDNS couldn't be set");
131         }
132     }
133 }
134 }
135
136 /**
137  * method for setting modes
138  * @param m Mode which should be set
139  */
140 void setMode(Mode m){
141     if(m == NORMAL){
142         Logger::add("setting mode to normal");
143         mode = NORMAL;
144         statusLED->setGreen();
145     } else if(m == ERROR){
146         Logger::add("setting mode to error");
147         mode = ERROR;
148         statusLED->setRed();
149     } else if(m == CONFIG){
150         Logger::add("setting mode to config");
151         mode = CONFIG;
152         statusLED->setBlue();
153     }
154 }
155
156 /**
157  * method for checkin if button is pressed
158  */
159 void handleButton(){
160     int state = digitalRead(BUTTON_PIN);
161     if(state == 1 && last_state == 0){ //button has been pressed
162         press_start = millis();
163     } else if(state == 0 && last_state == 1){ //button has been released
164         press_end = millis();
165     }
166     if(press_start > 0 && press_end > 0){
167         if((press_end - press_start) >= 3000){
168             setMode(CONFIG);
169         } else {
170             activateStandby();
171         }
172         press_start = 0;
173         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
33 bool MemoryManager::isNameSet(){
34     preferences.begin(MEMORY_NAMESPACE.c_str());
35     return preferences.isKey(NAME_KEY.c_str());
36     preferences.end();
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) &&
        ↪ WiFi.softAP(ssid);
57 }
58
59 /**
60  * starts esp32 wlan client which connects to the access point with the
        ↪ given credentials
61  */
62 bool NetworkManager::startClient(String ssid, String password, String
        ↪ hostname){
63     if(WiFi.getMode() == WIFI_AP){ //if wifi is in ap mode, ap mode will
        ↪ be disabled and station mode will be enabled
64         WiFi.softAPdisconnect();
65         WiFi.mode(WIFI_STA);
66     }
67     WiFi.disconnect();
68     int n = WiFi.scanNetworks();
69     for(int i = 0; i < n; i++){
70         if(WiFi.SSID(i) == ssid){
71             String bssid = WiFi.BSSIDstr(i);
72             Logger::add("ap mac: " + bssid);
73             WiFi.setHostname(hostname.c_str());
74             WiFi.begin(WiFi.SSID(i), password, 0, WiFi.BSSID(i));
75             return true;
76         }
77     }
78     return false;
79 }
80
81 void NetworkManager::reconnect(){
82     WiFi.reconnect();
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 };
117 #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::handle_notFound(){
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::handle_notFound, 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 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 public:
89     static ServerManager* getInstance();
90
91     /**
92      * starts the webserver
93      * @return if the start process was successful
94      */
95     bool start();
96
97     /**
98      * stops the webserver
99      * @return if the stop process was successful
100     */
101     bool stop();
102
103     /**
104      * handles the clients
105      */
106     void handleClient();
107
108     /**
109      * returns if the wifi credentials are received from the client
110      * @return if webserver received WiFi-credentials from client
111      */
112     bool wlanCredentialsReceived();
113
114     /**
115      * handles if the stream url is received from the client
116      * @return if webserver received Stream-URL from client
117      */
118     bool urlReceived();
119
120     /**
121      * handles if the name is received from the client
122      * @return if webserver received name of microcontroller from
123      *     ↪ client
124      */
125     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     bool volumeReceived();
133
134     /**
135     * returns the ssid, which was received from the client
136     * @return WLAN-SSID, which the webserver received from the client
137         ↪ , as a String
138     */
139
140     String getReceivedSsid();
141
142     /**
143     * returns the password, which was received from the client
144     * @return WLAN-Password, which the webserver received from the
145         ↪ client, as a String
146     */
147
148     String getReceivedPassword();
149
150     /**
151     * returns the url, which was received from the client
152     * @return Stream-URL, which the webserver received from the
153         ↪ client, as a String
154     */
155
156     String getReceivedUrl();
157
158     /**
159     * returns the name, which was received from the client
160     * @return name of the microcontroller, which the webserver
161         ↪ received from the client, as a String
162     */
163
164     String getReceivedName();
165
166     /**
167     * returns the volume, which was received from the client
168     * @return value of the volume which the webserver received from
169         ↪ the client, as a int
170     */
171
172     int getReceivedVolume();
173
174     /**

```



```

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,
    ↪ initializeAuth, getReactNativePersistence, signOut,
    ↪ sendPasswordResetEmail, confirmPasswordReset } from "firebase/auth";
3  import { getFirestore, setDoc, doc, getDoc, onSnapshot } from "firebase/
    ↪ firestore";
4  import User from "../types/User";
5  import Station from "@types/Station";
6  import AsyncStorage from "@react-native-async-storage/async-storage";
7
8  const firebaseConfig = {

```

```

9   apiKey: "AIzaSyBYW16NMGumkvA271lE6VyTszrAR80UDbo",
10  authDomain: "msa-app-dad57.firebaseio.com",
11  projectId: "msa-app-dad57",
12  storageBucket: "msa-app-dad57.firebaseio.com",
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
27 const storage = getFirestore();
28
29 export const Authentication = {
30   async login(email: string, password: string): Promise<User>{
31     try{
32       const res = await signInWithEmailAndPassword(auth, email,
33         ↪ password);
34       if(res.user.email === null){
35         throw "email is null";
36       }
37       return {uid: res.user.uid, email: res.user.email};
38     } catch(err){
39       throw err;
40     }
41   },
42   async register(email: string, password: string): Promise<void>{
43     try{
44       await createUserWithEmailAndPassword(auth, email, password);
45       return
46     } catch(err) {
47       throw err;
48     }
49   },
50   async logout(){
51     return signOut(auth);
52   },
53   onAuthChange(callback: (user: User | null) => void){
54     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                 ↪ ;
89                 const data = res.data();
90                 if(data === undefined || data.adapterList === undefined){
91                     throw "data is undefined";
92                 }
93                 console.log("adapter data:", data.adapterList);
94                 return data.adapterList;
95             } 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.
          ↪ info/json/countries?order=stationcount&reverse=true&
          ↪ limit=50");
10      const countries = res.data;
11      const countryList: Country[] = [];
12      for(let country of countries){
13        countryList.push({name: country.name, code: country.
          ↪ iso_3166_1});
14      }
15      const sortedCountries = countryList.sort((a, b) => {
16        if(a.name == b.name){
17          return 0;
18        } else if(a.name > b.name) {
19          return 1;
20        } else {
21          return -1;
22        }
23      });
24      return sortedCountries;
25    } catch(err) {
26      throw err;
27    }
28  },
29  async getLanguageNames(): Promise<Language []>{
30    try{
31      const res = await axios.get("https://de1.api.radio-browser.
          ↪ info/json/languages?order=stationcount&reverse=true&
          ↪ limit=20");
32      const languages = res.data;
33      const languageList: Language[] = [];
34      for(let language of languages){
35        let oldName = language.name;
36        let newName = oldName.charAt(0).toUpperCase() + oldName.
          ↪ slice(1);
```

```

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

```

```

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 } catch(err) {
90     throw err;
91 }
92 },
93 async getStationInfo(streamUrl: string){
94     const url = "http://de1.api.radio-browser.info/json/stations/byurl
95         ↪ ?url=" + streamUrl;
96     try{
97         const apiRes = await axios.get(url);
98         return apiRes.data[0];
99     } catch(err) {
100         throw err;
101     }
102 }
103 }
104 }

```

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/>
49        </View>
50        <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}>Haben Sie noch kein Konto?</
60     <Text style={GlobalStyle.textMedium}>Haben Sie noch kein Konto?</
61     <Text style={GlobalStyle.textMedium}>Haben Sie noch kein Konto?</
62     <Text style={GlobalStyle.textMedium}>Haben Sie noch kein Konto?</
63     <Text style={GlobalStyle.textMedium}>Haben Sie noch kein Konto?</
64     <Text style={GlobalStyle.textMedium}>Haben Sie noch kein Konto?</
65     <Text style={GlobalStyle.textMedium}>Haben Sie noch kein Konto?</
66     <Text style={GlobalStyle.textMedium}>Haben Sie noch kein Konto?</

```

app/(auth)/register.tsx

```

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

```

```

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/
   ↪ 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        ↳ hinzufügen" onPress={() => router.push("/(tabs)/adapter
15        ↳ /addNewAdapter")} />
16      <Button color={Colors.lightTurquoise} title="Bestehenden
17        ↳ Adapter hinzufügen" onPress={() => router.push("/(tabs)
18        ↳ /adapter/addExistingAdapter")} />
19    </SafeAreaView>
20  )
21 }
```

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         }} />
20     </SafeAreaView>
21   )
22 }
```

```

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

```

app/(tabs)/adapter/addNewAdapter.tsx

```

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

```

```

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={({
68                         ↪ text) => {setName(text)}}/>
69                 </View>
70                 <Text style={GlobalStyle.textBig}>{"Mac: " + adapter.
71                     ↪ mac}</Text>
72                 <Text style={GlobalStyle.textBig}>Mit WLAN verbinden

```

```

70         ↪ :</Text>
71         <View style={style.listContainer}>
72             <NetworkList networks={networkList} onSelect
73                 ↪ ={(item: Network) => setSelectedSsid(item.
74                 ↪ ssid)}/>
75         </View>
76         <Button title="Adapter hinzufuegen" color={Colors.
77             ↪ lightTurquoise}/>
78         {selectedSsid.length > 0 &&
79             <TextInputWindow text={"Passwort fuer " +
80                 ↪ selectedSsid + " eingeben:"} isPassword={
81                 ↪ true} onEnter={(password: string) => {alert(
82                 ↪ password)}} onCancel={() => {setSelectedSsid
83                 ↪ ("")}}/>
84             }
85         </SafeAreaView>
86     )
87 } else {
88     return(
89         <SafeAreaView style={GlobalStyle.page}>
90             <ErrorScreen errorText="Adapter nicht erreichbar.
91                 ↪ Versichere dich, dass du mit dem WLAN des
92                 ↪ Adapters verbunden bist!" buttonText="Nochmal
93                 ↪ Versuchen" onPress={() => {
94                 ↪ console.error("function not available!");
95                 ↪ }}/>
96             </SafeAreaView>
97         )
98     }
99 }

```

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} onSelect={() => {}}
26             ↪ onDeleteAdapter={(adapter: AdapterData) => {
27             ↪ deleteAdapter(adapter)}}/>
28     </SafeAreaView>
29 )
30 }

```

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: '
16                 ↪ Bestehenden Adapter hinzufuegen'}}/>
17         </Stack>
18     );
19 }

```

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 |
14         ↪ null>(null);
15     const [selectedStation, setSelectedStation] = useState<Station | null>(
16         ↪ null);
17     const [buttonDisabled, setButtonDisabled] = useState(true);
18     const { adapterList } = useContext(AdapterContext);
19
20     useEffect(() => {
21         if(selectedAdapter === null || selectedStation === null){
22             setButtonDisabled(true);
23         } else {
24             setButtonDisabled(false);
25         }
26     }, [selectedAdapter, selectedStation]);
27
28     return(
29         <SafeAreaView style={GlobalStyle.page}>
30             <Text style={GlobalStyle.textBig}>Adapter auswaehlen:</Text>
31             <AdapterList adapterList={adapterList} editable={false}
32                 ↪ showOnlyAvailable onItemClick={({item: AdapterData}) => {
33                 ↪ setSelectedAdapter(item)}} onDeleteAdapter={()=>{}}/>
34             <Text style={GlobalStyle.textBig}>Station auswaehlen:</Text>
35             <StationList editable={false} onItemClick={({item: Station}) =>
36                 ↪ {setSelectedStation(item)}}/>
37             <Button title="Bestaetigen" disabled={buttonDisabled} color={
38                 ↪ Colors.lightTurquoise} onPress={() => {
39                 if((selectedAdapter !== null) && (selectedStation !== null
40                 ↪ )){
41                     AdapterAPI.sendStreamUrl(selectedAdapter.name,
42                         ↪ selectedStation.url).then(() => {
43                             router.back();
44                         })
45                 }
46             }}/>
47         </SafeAreaView>
48     )
49 }

```

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: '
14       ↪ Verbindung hinzufuegen'}}/>
15   </Stack>
16 );
17 }

```

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) => {

```

```

29         selectedNames.push(val.name);
30     })
31     console.log(selectedNames);
32     setSelectedStations([... newSelectedStations]);
33 }
34
35 function isSelected(station: Station){
36     let selectedUids: string[] = [];
37     selectedStations.forEach((val) => {
38         selectedUids.push(val.uuid);
39     })
40     let selected = selectedUids.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         </Pressable>
74     </>
75     <Pressable onPress={() => {
76         let newStationList;
77         if(stationList != null){
78             newStationList = stationList.concat(selectedStations);
79         } else {
80             newStationList = selectedStations;
81         }
82         console.log("new stationlist:", newStationList);
83         try{
84             CloudStorage.setStationList(newStationList).then(() =>
              ↪ {
85                 router.replace("/music");
86             })
87         }catch(err){
88             console.error(err);
89         }
90     }}>
91         <AntDesign style={style.icon} name="check" size={50} color
              ↪ ={Colors.lightTurquoise}/>
92     </Pressable>
93 </SafeAreaView>
94 )
95 }

```

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 onItemSelect={() => {}}/>
9          </SafeAreaView>
10      )
11  }

```

app/(tabs)/music/radiosearch.tsx

```

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

```

```

3 import {Picker} from '@react-native-picker/picker';
4 import { Colors, GlobalStyle } from "@/constants/Style";
5 import { router } from "expo-router";
6 import { SafeAreaView } from "react-native-safe-area-context";
7 import { RadioBrowserAPI } from "@/api/RadioBrowserAPI";
8 import Language from "@/types/Language";
9 import Country from "@/types/Country";
10 import { SystemService } from "@/services/SystemService";
11
12 const Item = Picker.Item;
13
14 export default function RadioSearch(){
15     const [selectedCountryName, setSelectedCountryName] = useState("");
16     const [selectedLanguageName, setSelectedLanguageName] = useState("");
17     const [countryDataset, setCountryDataset] = useState<Country[] | null
18         ↪ >(null);
19     const [languageDataset, setLanguageDataset] = useState<Language[] |
20         ↪ null>(null);
21     const [isDataFetched, setDataFetched] = useState(false);
22
23     useEffect(()=>{
24         RadioBrowserAPI.getCountryNames().then(res => {
25             if(res !== null){
26                 setCountryDataset(res);
27             }
28         }).catch(err => {
29             console.error(err);
30         });
31
32         RadioBrowserAPI.getLanguageNames().then(res => {
33             if(res !== null){
34                 setLanguageDataset(res);
35             }
36         }).catch(err => {
37             console.error(err);
38         });
39     },[]);
40
41     useEffect(() => {
42         const systemCountryCode = SystemService.getRegionCode();
43         const systemLanguageCode = SystemService.getLanguageCode();
44         if(countryDataset !== null){
45             let systemCountry = countryDataset.find(country => country.
46                 ↪ code == systemCountryCode);
47             if(systemCountry !== undefined){

```

```

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                 </Button>
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>
16 );
17 }

```

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         Authentication.logout().then(()=> router.replace("/"));
25     }>
26 </SafeAreaView>
27 )
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   return (
48     <View style={[style.container1, {backgroundColor: backgroundColor}]}>
49       <View>
50         <Text style={GlobalStyle.textBig}>{adapter.name}</Text>
51         <Text style={GlobalStyle.textMedium}>{adapter.mac}</Text>
52       </View>
53       {reachable
54         ?
55         <BatteryIndicator batteryPercentage={adapter.battery}/>
56         : <Icons name="cloud-offline" size={24} color={Colors.white}/>
57       }
58     </View>
59   );
60 }

```

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   onSelect: Function;
14   onDeleteAdapter: Function;
15   editable: boolean;
16   showOnlyAvailable: boolean;
17 };
18
19 export default function AdapterList({
20   adapterList,
21   onSelect,
22   onDeleteAdapter,
23   editable,
24   showOnlyAvailable,
25 }: Props) {
26   const [selectedAdapter, setSelectedAdapter] = useState<AdapterData |
27     ↪ null>(null);
28
29   function handleItemPress(item: AdapterData) {
30     if (selectedAdapter !== null && selectedAdapter.mac == item.mac) {
31       setSelectedAdapter(null);
32       onSelect(null);
33     } else {
34       setSelectedAdapter(item);
35       onSelect(item);
36     }
37   }
38
39   function handleDeletePress() {
40     if (selectedAdapter !== null) {
41       Alert.alert(
42         "Adapter loeschen",
43         "Wollen Sie den Adapter '" +
44           selectedAdapter.name +
45           "' wirklich loeschen?",

```

```

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
63 function isSelected(item: AdapterData) {
64     if (selectedAdapter !== null && selectedAdapter.mac == item.mac) {
65         if ((showOnlyAvailable && item.connected) || !showOnlyAvailable) {
66             return true;
67         }
68     }
69     return false;
70 }
71
72 const style = StyleSheet.create({
73     container: {
74         width: "95%",
75         alignSelf: "center",
76     },
77     icon: {
78         alignSelf: "flex-start",
79     },
80     iconContainer: {
81         flexDirection: "row",
82         width: "95%",
83         justifyContent: "space-between",
84         alignSelf: "center",
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     />
104     {editable && (
105       <View style={style.iconContainer}>
106         <AddToListButton
107           onPress={() => router.push("/(tabs)/adapter/addAdapter")}
108         />
109         {selectedAdapter !== null && (
110           <DeleteButton
111             onPress={() => {
112               handleDeletePress();
113             }}
114           />
115         )}
116       </View>
117     )}
118   );
119 } else {
120   if(showOnlyAvailable) {
121     return (
122       <ErrorScreen
123         errorText="Kein Adapter verfuegbar!"
124         buttonText="Neuen Adapter hinzufuegen"
125         onPressed={() => 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         onPressed={() => 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 }
```

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){
20     iconName = "battery-half";
21   } else {
22     iconName = "battery-empty";
23   }
24 }
```

```

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             <Icons 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 }:
    ↪ Props) {
16     const style = StyleSheet.create({
17         container: {
18             width: "95%",
19             alignSelf: "center",
20         },
21         icon: {
22             alignSelf: "flex-start",
23         },
24     });
25
26     if (connectionList.length > 0) {
27         return (
28             <View style={style.container}>
29                 <FlatList
30                     data={connectionList}
31                     renderItem={({ item }) => (
32                         <Pressable onPress={() => onItemPress(item)}>
33                             <ConnectionItem
34                                 connection={item}
35                             />
36                         </Pressable>
37                     )}
38                 />
39                 <AddToListButton
40                     onPress={() => router.push("/(tabs)/connection/addConnection")}
41                 />
42             </View>
43         );
44     } else {
45         return (
46             <SafeAreaView style={GlobalStyle.page}>
47                 <ErrorScreen
48                     errorText="Es sind zurzeit keine Verbindungen vorhanden!"
49                     buttonText="Verbindung erstellen"

```



```

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

```

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     onSelect: Function
9 }
10
11 export default function NetworkList({networks, onSelect}: 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                     onSelect(item);
28                 }}>
29                 <NetworkItem ssid={item.ssid} rssi={item.rssi}
30                     ↪ selected= {(selectedNetwork !== null) && (item.
31                     ↪ ssid == selectedNetwork.ssid)} />
32                 </Pressable>
33             </FlatList>
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>
32     </View>
33   )
34 }

```

```

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   onItemSelect: Function;
17   editable: boolean;
18 };
19
20 export default function StationList({ onItemSelect, editable }: Props) {
21   const { stationList } = useContext(StationContext);
22   const [selectedStation, setSelectedStation] = useState<Station | null>(<
    ↪ null);
23
24   function handleItemPress(item: Station) {
25     if (selectedStation !== null && selectedStation.uuid == item.uuid) {
26       setSelectedStation(null);
27       onItemSelect(null);
28     } else {
29       setSelectedStation(item);
30       onItemSelect(item);
31     }
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         onPress={() => 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     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 }

```

```

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     </View>
55   )
56 }

```

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   );
38 }

```

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]} {...

```

```
    ↪ rest} />;
9 }
```

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
9   expect(tree).toMatchSnapshot();
10 });
```

components/__tests__/__snapshots__/ThemedText-test.tsx.snap

```
1 // Jest Snapshot v1, https://goo.gl/fbAQLP
2
3 exports[`renders correctly 1`] = `
4 <Text
5   style={
6     [
7       {
8         "color": "#11181C",
9       },
10      {
11        "fontSize": 16,
12        "lineHeight": 24,
13      },
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 = {

```

```

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 });
57 for(let adapter of adapterList) {
58     let containsMac = false;
59     for(let newAdapter of newAdapterList){
60         if(newAdapter.mac == adapter.mac){
61             containsMac = true;
62             break;
63         }
64     }
65     if(!containsMac){
66         let newAdapter = {
67             name: adapter.name,
68             mac: adapter.mac,
69             battery: 0,
70             volume: 0,
71             streamUrl: "",
72             connected: false,
73         };
74         newAdapterList.push(newAdapter);
75     }
76 }
77 setAdapterList(newAdapterList);
78 }
79 }
80
81 useEffect(() => {
82     let intervalId = 0;
83     if (user !== null) {
84         CloudStorage.onAdapterChange((newAdapterList) => {
85             for(let newAdapter of newAdapterList){
86                 let containsMac = false;
87                 for(let adapter of adapterList){
88                     if(adapter.mac == newAdapter.mac){
89                         containsMac = true;
90                         break;

```

```

91     }
92   }
93   if(!containsMac){
94     let newAdapters = [... adapterList];
95     let adapter: AdapterData = {name: newAdapter.name, mac:
      ↪ newAdapter.mac, volume: 0, battery: 0, streamUrl: "",
      ↪ connected: false};
96     newAdapters.push(adapter);
97     setAdapterList(newAdapters);
98   }
99   requestAdapters();
100 }
101 intervalId = setInterval(() => requestAdapters(), 5000);
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
  ↪ "react";
2 import { CloudStorage } from "../api/FirebaseAPI";
3 import Station from "@types/Station";
4 import { UserContext } from "../UserContext";
5
6 type Props = {
7   children: ReactNode;
8 };
9
10 type StationContextType = {
11   stationList: Station[]
12 };
13
14 const defaultContext = {
15   stationList: []
16 };

```



```

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

```

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 defaultContext = {
13     connectedToInternet: false
14 };
15
16 export const SystemDataContext = createContext<SystemDataContextType>(
17     ↪ defaultContext);
18
19 export const SystemDataProvider = ({children}: Props) => {
20     const [connectedToInternet, setConnectedToInternet] = useState(false);

```

```

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 defaultContext = {
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         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;
```