DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Lokalisierung von Dokumenten und Tracking des Workflows mit Hilfe von IoT-Geräten

STUDIENARBEIT

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

Max Heidinger
2361793

Pascal Riesinger
7586259

Abgabedatum	18.05.2020
Bearbeitungszeitraum	01.10.2019 - 18.05.2020
Kurs	TINF17B1
Betreuer	Prof. Dr. Marcus Strand

Eidesstattliche Erklärung

Wir versichern hiermit, dass wir unsere Studienarbeit mit dem Thema:

Lokalisierung von Dokumenten und Tracking des Workflows mit Hilfe von IoT-Geräten

gemäß § 5 der „Studien- und Prüfungsordnung DHBW Technik“ vom 29. September 2017 selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, den 26. März 2020

Max Heidinger, Pascal Riesinger

Inhaltsverzeichnis

Abkürzungsverzeichnis

Akku Akkumulator. 19

AMQP Advanced Message Queuing Protocol. 13

AP Access Point. 9

API Application Programming Interface. 18

BSSID Basic Service Set Identifiers. 11

CBOR Concise Binary Object Representation. 13

CoAP Constrained Application Protocoll. 13

DHBW Duale Hochschule Baden-Württemberg. V, 1–3

HTTP Hypertext Transfer Protocol. 13, 14

IoT Internet of Things. 13

JSON JavaScript Object Notation. 13, 14

LED Light Emitting Diode. 19

MQTT Message Queuing Telemetry Transport. 13

REST Representational State Transfer. 14

UART Universal Asynchronous Receiver Transmitter. 19

USB Universal Serial Bus. 18, 19

WLAN Wireless Local Area Network. 9, 18

Abbildungsverzeichnis

Listings

1 Einführung

An der DHBW in Karlsruhe basieren, wie auch noch in vielen anderen Organisationen, viele Prozesse auf Papier. Dies bedeutet, dass beispielsweise für die Genehmigung eines Einkaufes, ein oder mehrere Blätter Papier von unterschiedlichen Personen unterzeichnet werden müssen. Dafür wird das Dokument durch die einzelnen Büros getragen, bis letztendlich der Antragssteller es wieder bekommt.

Auch für simple Prozesse kann dies einige Zeit in Anspruch nehmen. Während dieser ganzen Zeit hat der Antragsteller leider keine Einsicht, bei welcher Person sich die Papiere befinden. Damit kann er auch nicht versuchen den Prozess zu beschleunigen, ohne Schritt für Schritt bei allen Personen des Prozesses nachzufragen. Zusätzlich kann nach Abschluss des Prozesses nicht nachvollzogen werden, wo der Flaschenhals des Prozesses liegt.

Um dieses Problem anzugehen, wird der „Paper-Tracker“ entwickelt. Dieser soll eine Transparenz in alle auf Papier basierten Prozesse bringen und zusätzlich Daten liefern, um die Prozesse selbst zu optimieren.

1.1 Idee des Paper-Tracker

Der Paper-Tracker ist ein kleines Gerät basierend auf einem Mikrocontroller, welcher an den Dokumenten befestigt wird. Dieser soll eine Lokalisierung der Papiere ermöglichen. Auf diese Lokalisierung soll der Antragsteller eines Prozesses über eine App Zugriff bekommen. Weiter soll auch in der App für Mobilgeräte der komplette Prozess oder Workflow selbst verfolgt werden. Somit kann zum Beispiel auch eine Notifizierung zuständiger Personen für einen Schritt in einem Workflow erfolgen.

2 Grundlagen

2.1 Projektumfeld

Das Projekt wird im Umfeld der DHBW Karlsruhe durchgeführt.

2.2 Funktionsweise WLAN

2.2.1 SSID

2.2.2 BSSID

2.2.3 Signalstärke RSSI

2.3 Beispiel eines Workflows

In folgender Abbildung 2.1 ist ein Beispiel für einen Prozess an der DHBW Karlsruhe dargestellt. Es ist das „Beschaffungs-Genehmigungsverfahren“, das benötigt wird, um eine Bestellung für die DHBW durchzuführen.

Der Prozess beginnt mit dem Antragsteller, welcher den Beschaffungsantrag stellt. Der Antrag geht zum Fachvorgesetzten des Antragsteller, welcher diesen fachlich prüft. Ist die Prüfung nicht erfolgreich, wird eine Ablehnung begründet und der Prozess beendet.

Bei erfolgreicher Prüfung muss der Antrag, wenn er über 5.000 liegt, zudem vom Dekan geprüft werden. Dieser kann entweder den Antrag wie oben beschrieben ablehnen oder stimmt diesem zu.

Ist der Antrag für eine Bestellung unter 5.000 oder er bekommt die Zustimmung des Dekans, erhält als nächstes die Verwaltung den Antrag. Die Verwaltung muss prüfen, ob die Mittel für diese Bestellung vorhanden sind. Sind die Mittel nicht vorhanden, wird der

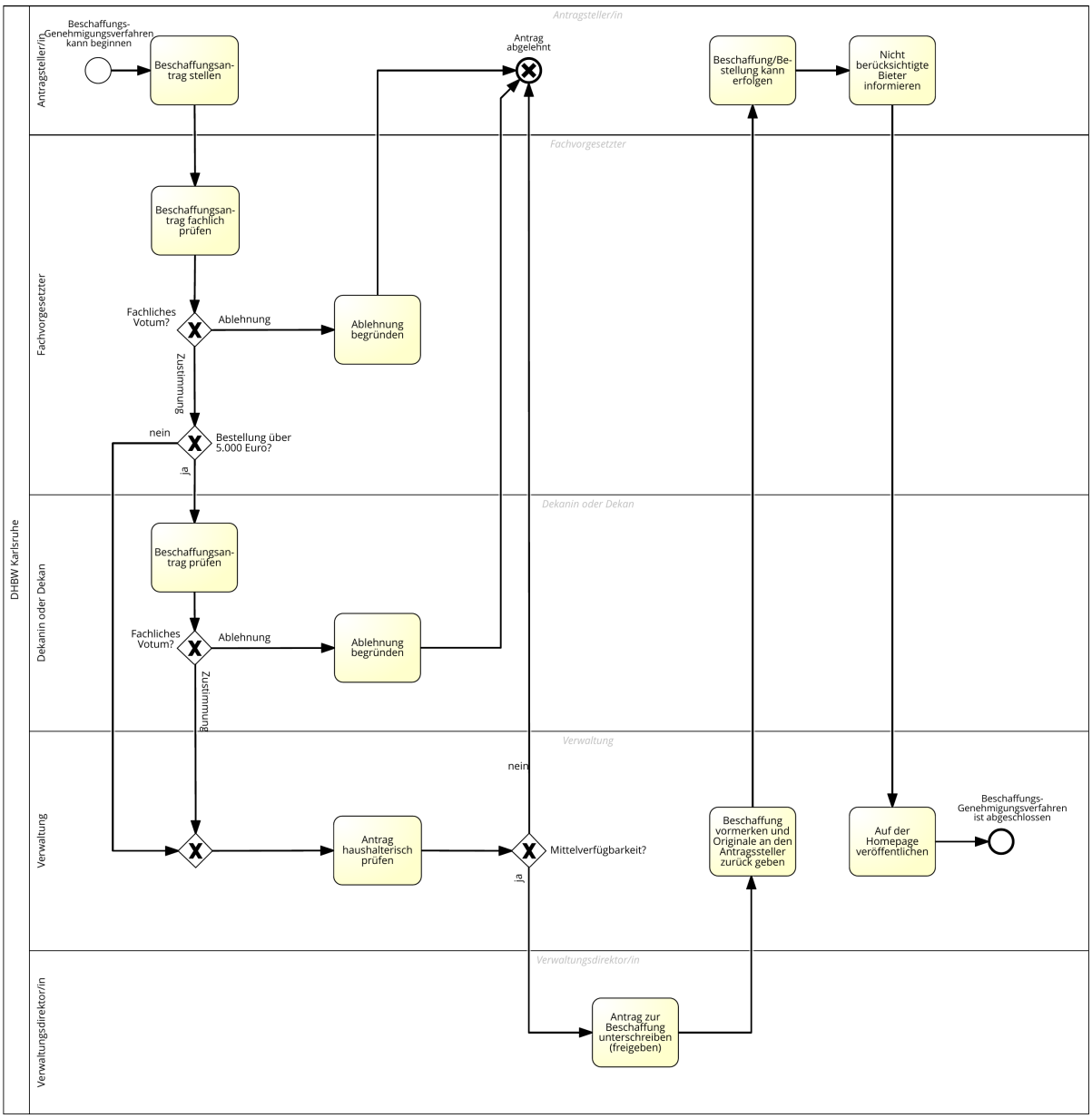

Abbildung 2.1: Beschaffungs-Genehmigungsverfahren der DHBW

Antrag abgelehnt. Bei vorhandenen Mitteln wird der Antag von dem Verwaltungsdirektor unterschrieben und damit freigegeben. Damit wird die Beschaffung in der Verwaltung vorgemerkt und der Antrag geht an den ursprünglichen Antragsteller zurück.

Sobald der Antragsteller den unterzeichneten Antrag erhält, kann die Beschaffung erfolgen. In diesem Schritt müssen mögliche nicht berücksichtigte Bieter im Falle einer Ausschreibung informiert werden.

Im letzten Schritt veröffentlicht die Verwaltung das Ergebnis der Beschaffung auf der Homepage und der Prozess ist erfolgreich abgeschlossen.

3 Analyse

In den folgenden Abschnitten werden der Ist-Zustand, aus welchem sich die Problemstellung ergibt, der Soll-Zustand, sowie die gestellten Anforderungen beschrieben.

3.1 Ist-Analyse

Es besteht das Problem, dass der aktuelle Status eines auf Papier basierten Prozesses nicht nachvollzogen werden kann. Für den Antragsteller eines Prozesses ist es nicht möglich herauszufinden, bei welcher Person oder in welchem Raum sich ein Dokument zu einem bestimmten Zeitpunkt befindet. Zusätzlich kann, sobald der Prozess beendet ist, nicht herausgefunden werden, an welcher Stelle im Prozess zum Beispiel eine besonders hohe Wartezeit stattgefunden hat.

3.2 Soll-Analyse

Ziel der Entwicklung und damit dieser Studienarbeit ist das Erarbeiten einer Lösung, mit welcher die aktuelle Position eines Dokuments und der aktuelle Fortschritt des Prozesses sichtbar gemacht werden kann. Es soll ein möglichst unauffälliges Gerät sein und eine intuitive Bedienung ermöglichen. Über aktuelle und abgeschlossene Prozesse sollen Daten analysiert werden können, um Möglichkeiten zur Prozessoptimierungen erkennen zu können.

3.3 Anforderungsanalyse

Im Folgenden werden die gestellten Anforderungen in funktionale und nichtfunktionale Anforderungen unterteilt. Dabei beschreiben funktionale Anforderungen konkrete Funktionen, die die entwickelte Lösung bieten muss, nichtfunktionale Anforderungen hingegegen zeigen Rahmenbedingungen, sowie Anforderungen an die technische Umsetzung auf.

Evtl. noch eine Anforderungsanalyse mit anderen Leuten durchführen

Desweiteren werden den Anforderungen Prioritäten zugewiesen, wobei ein Wert von 0 die höchste Priorität, ein Wert von 3 die niedrigste Priorität beschreibt, sodass die Lösung ohne Erfüllung der Anforderungen mit Priorität 0 nicht einsetzbar ist und Priorität 3 optionale Funktionalitäten beschreibt, die nicht zum Betrieb notwendig sind.

F-1 *P0* Der Standort von Dokumenten kann raumgenau verfolgt werden

F-2 *P0* Der aktuelle Status des mit dem Dokument verbundenen Workflows kann nachvollzogen werden

F-3 *P2* Neue Workflows können vom Nutzer definiert werden

F-4 *P1* Neue Tracker können dem System vom Nutzer hinzugefügt werden

F-5 *P2* Der Raumplan kann vom Nutzer eingelernt werden

F-6 *P3* Nutzer werden vom System benachrichtigt, wenn die Akkuladung eines Trackers unter einen gegebenen Prozentsatz fällt oder die Batterie ausgetauscht werden muss

F-7 *P1* Einzelne Schritte von Workflows können optional sein und übersprungen werden

F-8 *P1* Für einen Schritt im Workflow können mehrere Orte hinterlegt werden

NF-1 *P2* Die Laufzeit der Tracker im Batteriebetrieb beläuft sich auf mindestens eine Woche

NF-2 *P1* Die Tracker sollen möglichst unscheinbar und daher klein und leicht sein

NF-3 *P0* Die Daten können in einer Applikation für Mobiltelefone abgefragt werden. Dabei ist besonders die Verwendbarkeit unter dem Betriebssystem Android wichtig, die Unterstützung von iOS ist optional

NF-4 *P1* Die Anwendung muss stabil laufen und sollte sich nicht unerwartet beenden

NF-5 *P2* Die Lösung sollte skalierbar hinsichtlich der Anzahl der Tracker und der Größe des Tracking-Bereiches sein

NF-6 *P2* Zur optimalen Nachvollziehbarkeit von Workflows sollten deren auch nach Abschluss nicht gelöscht werden

NF-7 *P2* Die im System gespeicherten Daten sollen in einem maschinenlesbaren Format exportiert werden können, um Auswertungen über die Effizienz der Workflows durchführen zu können

4 Entwurf

4.1 Systemarchitektur

In Abbildung 4.1 ist eine grobe Übersicht über die verschiedenen Komponenten des Projektes gegeben. Es besteht aus drei Komponenten:

- Hardware-Tracker
- Backend-Anwendung
- App

Der Hardware-Tracker ist die physische Komponente, die direkt an ein zu trackendes Papier angeheftet wird. Er wird regelmäßig Daten erfassen und diese zur Lokalisierung an die Backend-Anwendung senden.

Die Backend-Anwendung dient als zentrales Element der Architektur. Sie bekommt Daten des Trackers zugeschickt, wertet diese aus und speichert sie. Auch das Management der Räume und Workflows wird von der Anwendung übernommen.

Nutzer verwenden zur Kommunikation mit dem Backend eine Smartphone-App. Sie kommuniziert ebenfalls mit der Backend-Anwendung und fragt von dieser Informationen an oder löst Aktionen aus.

Die Entwürfe der drei Komponenten werden in den nachfolgenden Abschnitten im Detail erläutert.

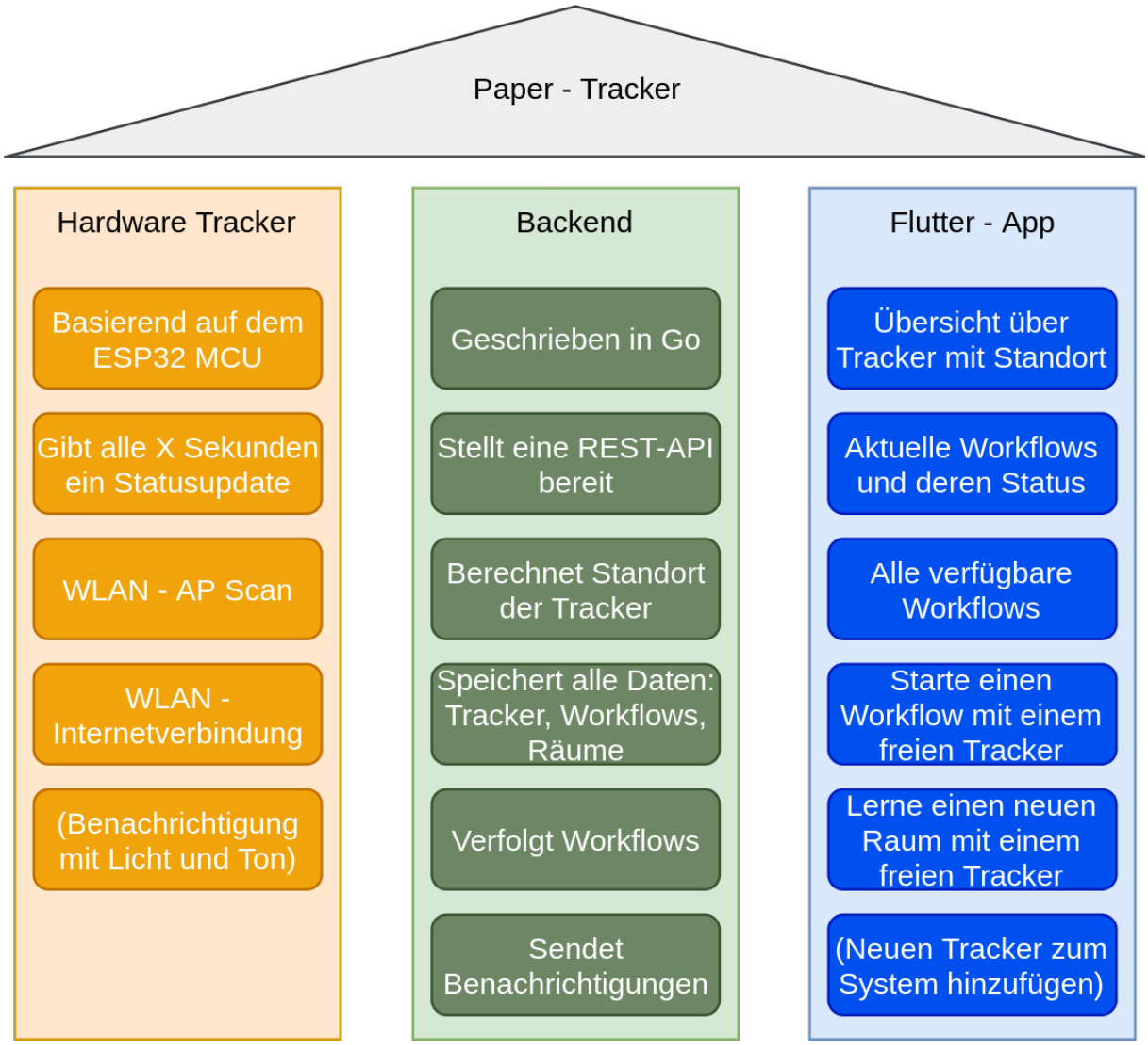

Abbildung 4.1: Architektur des Paper-Tracker

4.2 Komponentenarchitektur

In den folgenden Abschnitten wird detaillierter auf die einzelnen Komponenten des in Abschnitt 4.1 beschriebenen Systems eingegangen. Dabei werden die Komponenten in derselben Reihenfolge beschrieben, in welcher die Daten im Gesamtsystem fließen.

4.2.1 Tracking mit IoT-Hardware

Zur Erfassung der Standortdaten der Dokumente werden, wie in Abschnitt 3.2 beschrieben, Hardware-Tracker eingesetzt. Diese verwenden jedoch keine absolute Positionierungstechnologie wie beispielsweise GPS, da das hierfür benötigte Satellitensignal in Gebäuden zu schwach, und somit die Positionierung fehlschlagen kann.

Es wäre schön, Glossar-Referenzen zu kennzeichnen (oft mit Glos, oder Gls)

Für die Lokalisierung von Geräten innerhalb eines Gebäudes gibt es daher mehrere Ansätze. Eine Möglichkeit ist es, ein Mesh-Netzwerk aus den Tracker-Geräten aufzubauen. Ist die Position einiger weniger Knoten des Meshes bekannt, kann durch die Signalstärke zu umliegenden Knoten oder über die sogenannte Time-of-Arrival bestimmt werden, wo sich ein Knoten relativ zu anderen Knoten und damit auch zu den Knoten mit bekannter Position befindet. Mit dieser Methode kann eine Genauigkeit von bis zu einem Meter bei der Lokalisierung erzielt werden. (vgl. [1])

Quelle für die Genauigkeit von GPS in Gebäuden

Ist das hier eher Analyse?

Glossar: Mesh-Netzwerk

Eine weitere Möglichkeit ist, auf bestehende Infrastruktur zurückzugreifen. Wireless Local Area Network (WLAN) ist heutzutage in praktisch jedem Gebäude verfügbar. In Gebäuden, in welchen es mehrere Access Points (APs) gibt, kann die Position dadurch bestimmt werden, dass der Tracker auswertet, von welchen APs er ein Signal empfängt und wie stark der Empfang zum jeweiligen AP ist. Da die Positionsdaten ohnehin zur Auswertung an das Backend übertragen werden müssen, wofür eine WLAN-Verbindung notwendig ist, wird diese Methode eingesetzt. Die gesammelten Daten werden dann im Anschluss an das Backend übermittelt.

Time-of-Arrival erläutern

Lifecycle des Paper-Trackers erläutern

4.2.2 Backend-Anwendung

UML Daten Modellierung

Da die Backend-Anwendung die zentrale Komponente des Systems ist und auch die Daten verwaltet, werden für sie die Daten-Klassen modelliert. Das resultierende UML-Diagramm ist in Abbildung 4.2 abgebildet.

Die zentrale Klasse des Diagramms bildet die „Tracker“-Klasse. Sie modelliert einen Hardware-Tracker. Neben einigen Attributen, wie dem Label, der Batteriestatus und einigen gespeicherten Zeitstempeln, besitzt sie eine Referenz auf einen Status. Dieser Status gibt an, zu was der Tracker zur Zeit verwendet wird und in welchen anderen Zustand gewechselt werden kann.

Es gibt vier verschiedene Zustände, die in Integern kodiert sind:

Idle = 1
: Tracker ist frei verfügbar und kann in „Learning“ oder „Tracking“ übergehen.

Learning = 2
: Tracker wird für das Lernen eines Raumes verwendet und kann bei erfolgreichem Abschluss in „LearningFinished“ oder bei Abbruch in „Idle“ übergehen.

LearningFinished = 3
: Das Lernen eines Raumes mit diesem Tracker wurde beendet. Die Ergebnisse können berechnet und einem Raum zugewiesen werden. In Folge darauf, geht der Tracker in den Zustand „Idle“ über.

Tracking = 4
: Der Tracker besitzt aktuell einen Workflow, den er trackt. Nach Abschluss dieses Workflows oder nach Abbruch, geht der Tracker in den Status „Idle“ über.

Weiter besitzt der Tracker eine Referenz auf einen aktiven Workflow und den aktuellen Raum. Beide Referenzen können nichtexistent sein, falls der Tracker keinen aktiven Workflow besitzt oder der Raum nicht bestimmbar ist.

Ein „Room“ ist ein physikalischer Raum, der von einem Tracker erkannt werden soll. Neben einem Identifizierer und einem „Label“ (Name) für den Raum werden zusätzlich mehrere „BSSIDTrackingData“ gespeichert. Diese beinhalten für einen spezifischen

„Access Point“/Basic Service Set Identifiers (BSSID) einige Messdaten zu der gemessenen Signalstärke. Über diese kann über eine Antwort des Trackers mit „ScanResults“ die Position des Tracker bestimmt werden.

Damit der Tracker eine Antwort sendet, wird diese mit Hilfe eines Kommandos angefragt. Diese Kommandos werden jeweils in einer Instanz der „Command“-Klasse abgebildet. Die Kommandos besitzen einen spezifischen Typ der Enumeration „CommandType“, der die Aktion des Kommandos bestimmt. Auch wird pro Kommando eine „SleepTimeSec“ festgelegt, die bestimmt wie lange der Tracker sich nach der Aktion abschalten darf.

Spezielle Antworten der Tracker auf Kommandos und auch des Servers auf Anfragen der App sind in dem Unterpaket „communication“ modelliert. Dies ist zum Beispiel eine Fehler-Antwort und eine Basisklasse für Antworten des Trackers. Diese „TrackerResponse“-Klasse besitzt auch ein Feld für den aktuellen Batteriestand des Trackers, um diesen überwachen zu können.

Unter den Klassen „WorkflowTemplate“ und „WorkflowExec“ sind die Workflows modelliert. „WorkflowTemplate“ stellt eine Blaupause dar, wie ein Workflow aussieht. In diesem werden die einzelnen Schritte („Steps“) des Workflows festgelegt. Für einen „Step“ werden ein Label und eine Referenz auf den dazugehörigen Raum gespeichert. Über die Klasse „NextStep“ werden Referenzen auf nachfolgende Schritte modelliert. Die „NextStep“ Klasse besitzt ein Attribut „DecisionLabel“. Ist diese ein leerer String, so ist dieser verbundene Schritt ein linearer. Mit einem nicht-leeren String, gibt es mit diesem Schritt verschiedene Auswahlmöglichkeiten. Durch das transitive Attribut „StepEditingLocked“ der „WorkflowTemplate“-Klasse, wird die Bearbeitung blockiert. Dieses Attribut ist gesetzt, sobald es eine Ausführung des Templates gab.

Die „WorkflowExec“-Klasse stellt einen konkreten Workflow in der Durchführung dar. Als Attribute besitzt diese Klasse ein Label, einen Status und Start-/Endzeit. Der Status ist entweder „Running“, „Completed“ oder „Canceled“. Weiter besitzt die Klasse eine Referenz auf das Template, das ausgeführt wird, eine Menge an „ExecStepInfo“ und an den aktuellen Schritt. Die Klasse „ExecStepInfo“ besitzt Informationen über einzelne Schritte des Workflows. Dies sind zum einen die Start-/Endzeit des Schrittes sowie, ob der Schritt übersprungen wurde, und zum anderen die „Decision“-Information. Der „Decision“ String wird im Falle von mehreren Auswahlmöglichkeiten in einem Workflow dazu genutzt, in der Ausführung den richtigen Weg zu wählen.

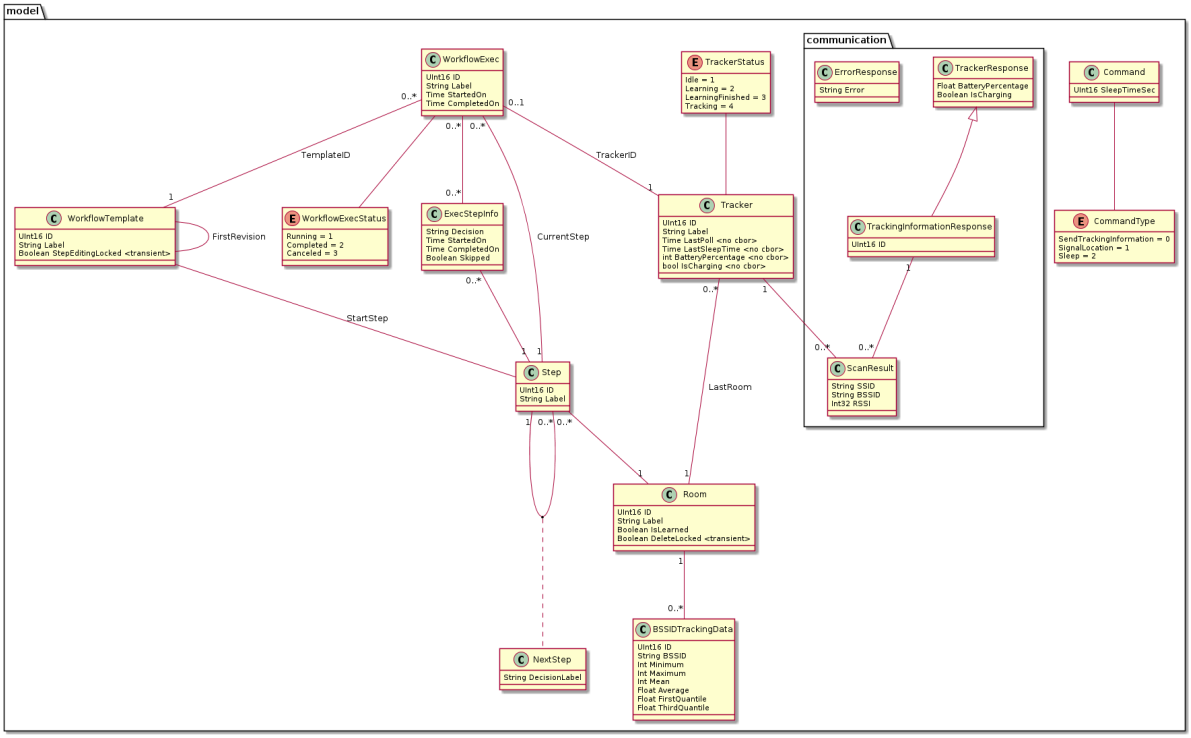

Abbildung 4.2: UML-Diagramm für die Backend-Anwendung

Schnittstellen

Für die Backend-Anwendung sind zwei verschiedene Schnittstellen vorgesehen. Eine Schnittstelle ist für den Tracker und eine für die App vorgesehen. Die unterschiedlichen Schnittstellen sollen die aus der Analyse entwickelten Ansprüche entsprechend erfüllen.

Tracker-Schnittstelle Für die Schnittstelle des Trackers bedeutet dies, dass sie möglichst simpel und effizient sein muss. Dies wird über ein entsprechendes Protokoll und einem Datenformat erreicht. Das verwendete Protokoll ist das Constrained Application Protocoll (CoAP) Protokoll und das Datenformat ist Concise Binary Object Representation (CBOR).

Bei Constrained Application Protocoll (CoAP) handelt es sich um ein spezialisiertes Protokoll für die Verwendung in Internet of Things (IoT)-Hardware, welches dafür optimiert wurde möglichst wenig Overhead zu haben.

Im IoT-Bereich haben sich einige Protokolle durchsetzen können. Dazu zählen vor allem Hypertext Transfer Protocol (HTTP), Message Queuing Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP) und CoAP. Da die Tracker batteriebetrieben sind, ist für diesen Anwendungsfall besonders die Performance des eingesetzten Protokolles wichtig. Je weniger Overhead das Protokoll aufweist, desto weniger Instruktionen müssen vom Prozessor des Trackers durchgeführt werden, was zu einer längeren Laufzeit führt. Werden die Protokolle anhand dieses Aspektes verglichen, erweist sich CoAP als effizientestes Protokoll. (vgl. [2], [3])

Quellen verwenden, die in diesem Paper zitiert werden

Concise Binary Object Representation (CBOR) ist ein Datenformat, das auf kleine Code- und Nachrichtengrößen optimiert ist. Demnach ist es auch für der IoT-Bereich sehr gut geeignet. Vom Aufbau des Formats, ist es sehr gut mit JavaScript Object Notation (JSON) zu vergleichen, da es auch Schlüssel-Wert-Paare verwendet. Der große Unterschied ist jedoch, dass CBOR binär kodiert wird und JSON in Text kodiert wird. Dadurch ist CBOR nicht menschenlesbar aber sehr gut maschinenlesbar. Bei einem Vergleich zwischen der CBOR Bibliothek „libCBOR“ und der JSON Bibliothek „ArduinoJson“ konnte zudem festgestellt werden, dass bei gleicher Nachricht das CBOR Format ca. 26.5% kleinere Nachrichten in 83.3% der Zeit schnellerer Zeit serialisiert hat.

Hier das Polling-Prinzip oder unter Hardware?!

Die Tracker-Schnittstelle soll drei Endpunkte zur Verfügung stellen. Diese sind in Tabel-

Verb	Pfad	Aktion	Rückgabe-Wert
POST	/tracker/new	Neuen Tracker erstellen	Tracker ID
GET	/tracker/poll	Kommando für Tracker abfragen	Tracker Kommando
POST	/tracker/tracking	Ergebnisse des Tracking speichern	-

Tabelle 4.1: Verfügbare Endpunkte der Tracker-Schnittstelle

le 4.1 aufgelistet. Bis auf den Endpunkt zum erstellen eines neuen Trackers, erhalten alle Endpunkte als Parameter die Tracker ID des Trackers, der die Anfrage stellt. Der Endpunkt zum Speichern der Tracking Ergebnisse hat zudem als Parameter die Ergebnisste selbst.

App-Schnittstelle Die Schnittstelle für die App hat keine besonderen Anforderungen und kann daher frei gewählt werden. Aus diesem Grund werden Protokolle und Datenformate verwendet, die momentan dem aktuellen Stand der Technik entsprechen. Dies ergibt den Vorteil, dass sie mit allen anderen gewählten Technologien sehr gut kombinierbar sind und entsprechende Bibliotheken vorhanden sind. Das verwendete Protokoll ist das HTTP-Protokoll und das Datenformat ist JSON. Die Schnittstelle basiert auf dem Representational State Transfer (REST) Paradigma. Alle verfügbaren Endpunkte der App-Schnittstelle sind in Tabelle 4.2 dargestellt.

Quelle für aktuelle Protokolle

Soll REST erläutert werden?

Zwischen"überschriften"hinzufügen für Room, Tracker, etc.

Erläuterung der einen Abkürzung (wf)?

Hinzufügen einer caption führt zu Fehlern

Verb	**Pfad**	**Aktion**	**Parameter**	**Rückgabe-Wert**
GET	/room	Auflistung der Räume	-	Liste aller Räume
POST	/room	Erstellen eines neuen Raums	Raum	Erstellter Raum
PUT	/room/:id	Raum aktualisieren	Raum	Aktualisierter Raum
DELETE	/room/:id	Raum löschen	-	-
GET	/tracker	Auflistung der Tracker	-	List aller Tracker
PUT	/tracker/:id	Tracker aktualisieren	Tracker	Aktualisierter Tracker
DELETE	/tracker/:id	Tracker löschen	-	-
POST	/tracker/:id/learn	Lernen mit Tracker starten	-	Lernzeit
GET	/tracker/:id/learn	Lern Status abfragen	-	Abgeschlossen Gefundene SSIDs
DELETE	/tracker/:id/learn	Lernen abbrechen	-	-
POST	/tracker/:id/learn/finish	Lernen abschließen	Raum ID SSIDs	
GET	/workflow/template	Auflistung der Templates	-	Liste aller Templates
POST	/workflow/template	Template erstellen	Template	Erstelltes Template
PUT	/workflow/template/:id	Template aktualisieren	Tempalte	Aktualisiertes Template

Verb	**Pfad**	**Aktion**	**Parameter**	**Rückgabe-Wert**
DELETE	/workflow/template/:id	Template löschen	-	-
POST	/workflow/template/:id/start	Start Schritt erstellen	Step	Erstellter Step
POST	/workflow/template/:id/step	Schritt hinzufügen	Step Vorgänger ID	Erstellter Step
GET	/workflow/template/:id/step/:id	Schritt abfragen	-	Step
PUT	/workflow/template/:id/step/:id	Schritt aktualisieren	Step	Aktualisierter Step
DELETE	/workflow/template/:id/step/:id	Schritt löschen	-	-
POST	/wf/template/:id/step/:id/move	Schritt verschieben	Richtung	-
POST	/workflow/template/revision	Template Revision erstellen	Label	Erstellte Revision
GET	/workflow/exec	Auflistung der Ausführungen	-	Execution
POST	/workflow/exec	Ausführung erstellen	Execution	Erstelle Execution
POST	/workflow/exec/:id/progress/:id	Ausführung zu Schritt progressieren	-	-
POST	/workflow/exec/:id/cancel	Ausführung abbrechen	-	-

4.2.3 App für Mobilgeräte

5 Implementierung und Validierung

5.1 Backend-Server

5.1.1 Verwendete Technologien

5.1.2 Architektur

5.1.3 Analyse-Algorithmus

5.1.4 REST-Schnittstelle

5.1.5 COAP-Schnittstelle

5.2 Hardware und Firmware

Im Folgenden wird erläutert, wie der Tracker hinsichtlich der verwendeten Hardware und der entsprechenden Firmware implementiert wurde.

5.2.1 Verwendete Technologien

Als Mikrokontrollerplattform wurde die sogenannte „TinyPICO“-Plattform gewählt. Nach eigener Aussage handelt es sich dabei um die kleinste vorgefertigte Entwicklungsplattform basierend auf Mikrocontrollern vom Typ ESP32. Dieser Aspekt ist für die Erfüllung von **NF-2** wichtig. Dieser Mikrocontroller wurde gewählt, da er alle benötigten Funktionen wie beispielsweise WLAN bereits über ein Application Programming Interface (API) bereitstellt.

Die TinyPICO-Entwicklungsplattform stellt neben dem Mikrocontroller weitere Hardware bereit, so sind unter anderem die Stromversorgung über Universal Serial Bus (USB) sowie

Lithium-Polymer-Akkumulator (Akku), ein Übersetzer von USB zu Universal Asynchronous Receiver Transmitter (UART) zur Programmierung des Mikrocontroller und eine Vollspektrum-Light Emitting Diode (LED) verbaut. Ein weiterer Vorteil ist das bereits integrierte Ladesystem, welches einen angeschlossen Lithium-Polymer-Akku automatisch auflädt, sobald der Tracker über USB mit einer Stromquelle verbunden wird. Außerdem können über das Ladesystem der aktuelle Ladezustand und die Spannung des Akku abgefragt werden, was für die Erfüllung von **F-6** unabdingbar ist.

Die Firmware wurde in der Programmiersprache C++ auf Basis des Arduino-Frameworks erstellt. Dieses Framework wurde gewählt, da es auf vielen Plattformen, wie auch dem ESP32 lauffähig ist, weit verbreitet ist und somit viele Bibliotheken für das Framework existieren.

5.3 App

Im diesem Kapitel wird die technische Umsetzung der App für Mobilgeräte beschrieben.

5.3.1 Verwendete Technologien

Zur Programmierung der App wurden die Programmiersprache Dart und das Framework Flutter verwendet. Ein großer Vorteil dieser Kombination ist, dass Anwendungen sowohl für Android, als auch für iOS kompiliert werden können, ohne den Quellcode anpassen zu müssen. Auch eine Laufzeitumgebung für Desktopcomputer unter Windows, macOS und GNU/Linux befindet sich aktuell im Teststadium, sodass die Paper-Tracker App gegebenenfalles auch auf Arbeitsplatzrechnern verwendet werden kann.

5.4 Bekannte Probleme

5.4.1 Ungenauigkeit des Tracking

5.4.2 UI-Details in der App

6 Validierung

6.1 Genauigkeitsmessung des Tracking

6.1.1 Beschreibung

6.1.2 Durchführung

6.1.3 Ergebnisse

6.2 Nutzerumfrage zur UX der App

6.2.1 Beschreibung

6.2.2 Durchführung

6.2.3 Ergebnisse

6.2.4 Gesamtergebniss der Validierung

7 Ausblick und Weiterentwicklungen

7.1 Audiovisuelle Signale

7.2 Verbessertes Gehäuse

7.3 Tracking von Smartphones

Es ist wünschenswert, dass eine Person, die einen Raum betritt, in welchem ein Dokument auf deren Bearbeitung wartet, darüber informiert wird. Zu diesem Zweck können Smartphones in das Tracking-System mit aufgenommen werden.

8 Zusammenfassung

Glossar

GPS Das Global Positioning System (GPS) ist ein von den Vereinigten Staaten von Amerika entwickeltes globales Navigationssystem, welches im zivilen, kommerziellen, wissenschaftlichen und militärischen Bereich Anwendung findet, um die Position von GPS-Empfängern zu bestimmen. (vgl. [4]). 9

Overhead Also Overhead werden Metadaten bezeichnet, welche zusätzlich zu den eigentlichen Nutzdaten anfallen. Beispiele für Overhead sind Prüfsummen, Netzwerkadressen, oder Codierungszeichen. . 13

Quelle

Literatur

[1] Patwari, N. u. a. „Relative location estimation in wireless sensor networks“. In: *IEEE Transactions on Signal Processing* 51.8 (08/2003), S. 2137–2148 (siehe S. 9).

[2] Dizdarević, J. u. a. „A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration“. In: *ACM Computing Surveys* 51.6 (01/2019), S. 1–29 (siehe S. 13).

[3] Naik, N. „Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP“. In: *2017 IEEE International Systems Engineering Symposium (ISSE)*. IEEE, 10/2017 (siehe S. 13).

[4] Department Of Transportation, F. A. A. *Global Positioning System Wide Area Augmentation System (WAAS) Performance Standard*. 10/2008. URL: https://web.archive.org/web/20170427033332/http://www.gps.gov/technical/ps/2008-WAAS-performance-standard.pdf (besucht am 30. 10. 2019) (siehe S. 23).

Todo list