RFID Card Management System

Mayrhofer Johannes, Grubauer Patrick, Zöchmann Benedikt

24. Juni 2022
6. Juli 2022
16. August 2022
18. August 2022
17. September 2022

Inhaltsverzeichnis

1 Idee **3**

2 Anforderungen **3**

3 Umsetzung **4**
3.1 Arbeitsaufteilung 5
3.2 Zentrale API 6
3.2.1 Kommunikation zwischen App und API 6
3.2.2 Kommunikation zwischen API und Cardstorage Unit 6
3.2.3 Datenbank 6
3.2.4 Fallbeispiel um das Schließfach von der Client-Anwendung am Smartphone zu öffnen 6
3.2.5 TL;DR 6
3.3 Cardstorage-Controller 7
3.4 Client Anwendung 8
3.5 Login 8
3.5.1 Funktionen 8
3.6 User-Login 8
3.6.1 Generelle Spezifikation 8
3.6.2 Funktionen 8
3.7 Admin-Login 8
3.7.1 Generelle Spezifikation 8
3.7.2 Funktionen 9

1 Idee

Es soll ein Tresor für NFC-Karten entwickelt werden, der NFC-Karten für spezielle Zwecke aufbewahrt. Die NFC-Karten haben das klassische Scheckkarten-Format. Die Gesamtzahl an Schließfächern soll mindestens 10 betragen, kann aber auch größer sein. Jedenfalls soll der gesamte Tresor eine geschlossene Bauform bieten, dessen Bautiefe möglichst gering ist (Maß von Mauer nach vorne). Die folgenden Anforderungen werden erfüllt.

2 Anforderungen

- Quaderförmiges Gehäuse mit wenig Bauhöhe zu der Wand
- Zentrales Display
- Entnahmeregistrierung mit Buchungszeile am Display
- Schließfächer für mind. 10 Karten
- Kartenverwaltung (je Automat)
- Jedes Schließfach soll transparent sein
- Jede Karte soll stehend parallel zur Wand sichtbar sein
- Jedes Schließfach soll elektrisch aufspringen
- Jedes Schließfach soll durch Fingerdruck wieder geschlossen werden
- Rückgabekontrolle der Karte nur am richtigen Fach
- Löschung der zugeordneten Buchungszeile am Display
- Datenverwaltung für Statistik im Hintergrund
- Administrativer Wartungszugang

3 Umsetzung

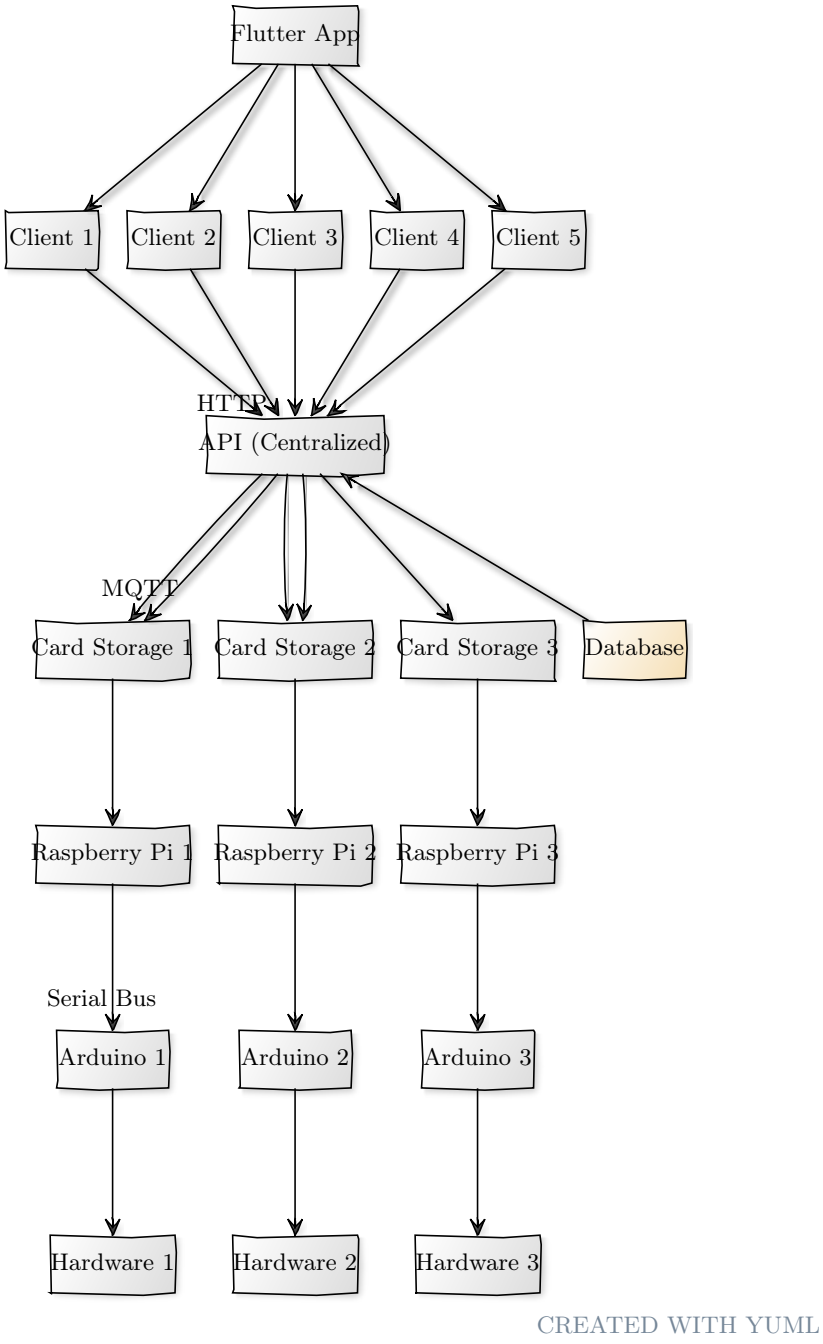

Abbildung 1: Architektur

3.1 Arbeitsaufteilung

- Mayrhofer Johannes: Zentrale API, Datenbank, API-Raspberry-μC Kommunikation
- Grubauer Patrick: Flutter App $\rightarrow$ Client-Login, Login-Page, Hardware (RFID), Arduino - Raspberry Kommunikation
- Zöchmann Benedikt: Flutter App $\rightarrow$ Admin-Login, Hardware (Display), Arduino - Raspberry Kommunikation

3.2 Zentrale API

Die API ist der zentrale Knotenpunkt zum Austausch von Daten zwischen Client Anwendung (Flutter App) und den Cardstorages (siehe Abbildung 1). Die Kommunikation zwischen der API und den Clients wird mithilfe der REST (Representaional State Transfer) Architektur realisiert. Die API kommuniziert mit den Cardstorages mithilfe MQTT welche jeweils ein Topic zugewiesen bekommen und so Daten senden als auch empfangen können. Die API speichert etwaige Daten in einer Datenbank. Diese Daten können ausschließlich von der API verwaltet werden. Diese wird mithilfe der Programmiersprache Go implementiert. Daten werden im JSON Format ausgetauscht da dies der de-facto Standard zum Austausch von Daten im Internet ist und es sehr gut von Go unterstützt wird.

3.2.1 Kommunikation zwischen App und API

Die Client bzw. Admin Anwendung auf dem Endgerät des Benutzers als auch auf dem Display des Schließfaches kommunizieren mittels der oben beschriebenen REST API miteinander.

3.2.2 Kommunikation zwischen API und Cardstorage Unit

Die Cardstorage Einheiten kommunizieren mit der API mithilfe dem MQTT Protokoll. Das Controller-Programm welches auf dem Raspberry (bzw. ein anderer Mikroprozessor) der Cardstorage läuft, kommuniziert mit der API, um die Kommandos welche die API von der Client bzw. Admin Anwendung erhält, auszuführen.

3.2.3 Datenbank

Wie bereits erwähnt werden alle aufzubewahrenden Daten (Schließfächer, Benutzer, Karten, Rechte) in der Datenbank gespeichert. Zum Einsatz kommt eine relationale Datenbank (für Testzwecke PostgreSQL).

3.2.4 Fallbeispiel um das Schließfach von der Client-Anwendung am Smartphone zu öffnen

Request (POST) vom Smartphone an API mit Schließfach ID; API sendet Kommando per MQTT an Schließfach-Topic um dieses zu öffnen; API speichert möglicherweise dieses Kommando in der DB um später Auswertungen anfertigen zu können; Schließfach-Controller (bsp. Raspberry) erhält Kommando und gibt dies an den Mikrocontroller der Schließfach-Schaltung weiter welches dieses schließlich öffnet.

3.2.5 TL;DR

Die API ist das Herzstück der Anwendung. Sowohl die Client- und Admin-Anwendung als auch die Cardstorage Einheiten kommunizieren damit. Die API ist, wenn man so will, der Übersetzer zwischen den beiden. Sie übersetzt einerseits die Kommandos der Client- und Admin-Anwendung in MQTT-Befehle welche dann von dem Controller-Programm am z.B. Raspberry des Cardstorage ausgewertet und an die Schaltung weitergegeben werden. Andererseits verwaltet die API auch den Zugriff zur Datenbank, in welcher alle Cardstorage Units (Schließfächer), Benutzer, Karten und Rechte pro Benutzer hinterlegt sind.

3.3 Cardstorage-Controller

Das Programm, welches am Mikroprozessor, in der obigen Grafik (siehe Abbildung 1) ein Raspberry Pi, läuft und einerseits die Kommandos, beispielsweise die Schließfachtür per App öffnen, interpretiert und ausführt aber auch andererseits Daten an die API, wenn diese danach fragt, beispielsweise die aktuelle Anzahl der vorhandenen Schlüsselkarten im Schließfach, zurücksendet. Das Cardstorage-Controller-Programm kommuniziert mittels MQTT mit der API. Dieses wird mithilfe der Programmiersprache Go implementiert. Jeder Cardstorage-Controller bekommt ein MQTT-Topic zugewiesen, womit dieser mit dem Broker (Zentrale API) kommunizieren kann. Um, wie oben erwähnt, Funktionen bereitstellen zu können die Hardware beinhaltet, welche von den Mechatronikern enwtickelt wird, beispielsweise die aktuelle Anzahl der vorhandenen Schlüsselkarten im Schließfach, muss der Cardstorage-Controller auch mit dem Mikrocontroller kommunizieren, welcher eben diese Hardware steuert. **Sobald ein Konzept seitens der Mechatroniker zur Verfügung steht werden Einzelheiten weiter erläutert.**

3.4 Client Anwendung

Als Grundlage zur adminstrativen als auch benutzerspezifizischen Verwaltung, kommt eine Flutter App zur Anwendung. Wir haben uns für dieses Framework entschieden, da es auf verschiedenen Plattformen bzw. Betriebssystem mit einer Version der Software läuft (Native Framework). Um zu dem Admin-Login zu gelangen, wird ein eigener Login bei der Login-Seite benötigt. D.h Admin und User Anwendung teilen sich ab den Login auf.

3.5 Login

Der Login dient dazu, um zu der Admin bzw. User Anmeldung hinzugelangen.

3.5.1 Funktionen

- Neuanmeldung eines Lehrers möglich
- "Remember me"→ Funktion welche Passwörter sicher speichern (Encrypted-Shared-Preferences)
- Möglichkeit sein Passwort zurückzusetzen
- Quick Login/Registrierung Möglichkeit am Terminal mittels Rfid Karte der Lehrer

3.6 User-Login

3.6.1 Generelle Spezifikation

Der User Login wird von den Lehrenden verwendet, um Karten zu beantragen, reservieren, anzufordern [...]. Weiters soll der Login am Display als auch im Web zur Verfügung gestellt, damit wirklich jeder Zugang zum Automaten hat.

3.6.2 Funktionen

- Als Homepage wird eine Seite mit verschiedenen Grafiken bspw. über die aktuellen verwendeten, reservieren, benutzen Karten angezeigt. Weiters sollen Informationen, wie die durschnittliche Verwendungsdauer, meist verwendeten Karten visualisiert werden, damit der User besser einschätzen kann, ob er die Karte reservieren soll.
- Es soll eine Seite mit allen Karten angezeigt werden, wobei man die angezeigten Karten, anhand der Automaten filtern kann.
- Eine eigene Seite zum Reservierungsystem
- Quick Card Get Möglichkeit mittels Pop-Up am Display.
- Möglichkeit Nachrichten an den Admin zu senden (Feedback, Bug-report, Verbesserungen [...])
- Seite zum ändern der Benutzerdaten

3.7 Admin-Login

3.7.1 Generelle Spezifikation

Der Admin Login soll dazu verwendet werden können, direkt am Display am Automaten, administrative Aufgaben erledigen zu können z.B: Karten hinzufügen, Benutzer anlegen, Karten Tresore hinzufügen [...] Diese Anwendung soll auch als Webseite und APP zur Verfügung stehen, um solche Einstellungen nicht direkt Vorort machen zu müssen. Die Admin App wird ebenfalls mit Flutter erstellt. Um die Admin App benutzen können, wird es bestimmte Benutzer geben die sich als Admin anmelden können.

- In der App soll der User sich in einer Grafik ansehen können, welche Karten gerade verfügbar sind und welche nicht. Diese kann pro Automaten oder mit allen Karten gemachte werden.
- Weiters soll man in der App einen neuen Automaten hinzufügen können, dazu muss man einige Parameter angeben wie Name, IP-Adresse, Anzahl Karten, Ort. Danach werden diese in einer Datenbank im Hintergrund eingepflegt.
- In der App sollen auch neue Karten hinzugefügt werden können oder Karten getauscht werden, dazu wird, wenn das Fenster geöffnet wird am Automaten angezeigt, dass die Karten zum Lesegerät gehalten werden soll, damit die Karte im System gespeichert werden kann.
- Eine weitere Option ist, dass man auch neue Benutzer anlegen können soll, oder auch verwalten kann.
- Als letzten soll sich der Admin auch Statistiken erstellen können, damit die Auslastung im Auge behalten werden kann.
- Weiters soll sich der Admin Logs zu der Datenbank und API anzeigen lassen können.

3.7.2 Funktionen

- Ansicht, welche Karten Verfügbar sind
- Neuen Automaten anlegen / löschen / bearbeiten
- Neue Karten hinzufügen / löschen / bearbeiten
- Benutzer anlegen / löschen / bearbeiten (Daten ändern)
- Statistik / Auswertungen erstellen
- Logs anzeigen