18.02.2020 // JitSi Call 18.02.

Teilnehmer:

- Marcus
- Kevin
- Fabi
- Dequbed
- Tasso (listening)

Call

Jira wird eingerichtet (am Donnerstag)
Ziel ist, ADRs “herausfallen” zu lassen.

Sprache: TypeScript und Rust sind noch drin -->
Wenn komplett modular: anders.
Diskussionsbedarf für “anders als NodeJS und Rust”: …
Frage: Was braucht die Zielgruppe??
Antwort: User werden eher nicht an der Infrastruktur-Software bauen. --> Prio, dass Nutzer Anpassungen vornehmen können ist eher klein. Admins wollen nur installieren, haben keine Zeit zum bauen.

Das deutet irgendwie auf einen “Kern” hin, in dem viel fertig (konfigurierbar) vorhanden ist… gerne modular erweiterbar.

Ziel: Waagen sollen z.B. abfragbar sein. --> Modular.
Türen aufmachen usw. --> Monolithisch.

Waagen sind “Verbrauch von Werten” --> kann im “Kern” gebaut werden.

Was sind die ersten 2/3/4/5 UseCases?
<dequbed/Marcus>: Maschinen verwenden - Berechtigungssystem
2 Arten: reservierbar / one-time-use / Berechtigungen
Abrechnung … wurde x min. verwendet - kostet y.

Ansatz für heute:
Was haben wir bisher gemacht --> Wie kommen wir daraus sinnvoll zu einer Architektur?

Sprache klären oder nochmal aufschieben? --> Aufschieben.
Mit welchen Beweggründen haben wir was gemacht? … aufgeteilt in Frontend / Backend.

Fabian: Screensharing.
Backend: TypeScript / Node
Anforderung war: Generell erst mal anfangen, Zugangskontrolle zu automatisieren.
Bisher: viele Eigenbaulösungen mit ESP / Reader / Türschlössern / Relays / … FabLab ist aber versichert, daher ist selbstgebaute Hardware nicht gern gesehen. Es gibt aber viel HomeAutomation-Hardware von der Stange, die das Problem an der Stelle lösen kann.
–> Schaltbare Steckdose finden und die anbinden.
Auf ein Protokoll einschränke ist doof --> wie können wir viele Protokolle mit wenig Aufwand einbinden? --> OpenHab2 mit RestAPI, um Aktoren anzusprechen.
Andere FabLabs nutzen odoo als UserBasis und zur Abrechnung. User / Booking / Rechnung / …

–> github.com/faaaaabi/fablab-api-gateway

github.com/faaaaabi/fablab-access-app
App: ReactNative (FaceBook-Frontend-Framework)
Nicht nur WebApp, sondern native.
App fragt Karte, kann Maschinen einschalten die er darf --> gibt Backend bescheid --> das schaltet über OpenHAB.
Räumlichkeiten sind wichtig, weil verschiedene Tablets verschiedene Maschinen schalten können.
Angedacht: 2FA

fabaccess.bian-meyer.de/
React-App. Kann Berechtigungen definieren, odoo hat nur Checkbox für “Hat Sicherungseinweisung”, odoo soll aber nur Stammdaten pflegen.
Berechtigungen sollen ins Backend.
Dependencies: Frameworks, xpress? analog zu Django bei Python. Odoo / OpenHAB / … aber nicht als Dependency, sondern mit abstraktem Interface implementiert.
Kommunikation zwischen Backend / OpenHab: REST; Backend / odoo: XML

RoseGuarden:
vor ca. 3 Jahren angefangen mit V1
Angular-App als Frontend. Backend aus Python, monolithisch.
Primäre Anwendung: Türen. Geräte nicht so wichtig. Viele Vereine, nicht zwingend FabLabs.
Hochschulen / größere MakerSpaces: Geräte … kleinere (haben keine Geräte) daher eher Türen.

Basis: RaspberryPi, liefert FrontEnd. Knoten synchronisieren sich über RestAPI.
Knoten die ausfallen bringen das System nicht durcheinander.

RoseGuarden 2:
ServerAnsatz wäre besser gewesen. Sollte Fernwartung zulassen. Ist gar nicht so viel anders als RG1.
Backend: modular, in Workspaces organisiert. mit 2FA und Pin lässt sich die Tür öffnen.
Neben Türen gab es den Ansatz, dass sich User verwalten lassen müssen.
Viel an der Hardware geschraubt “polling über REST”
FrontEnd: VueJS (React / Angular / Vue)
PythonCode - definiert Actions und Views einfach und modular. Kann evtl. “krachen”, wenn das Interface nicht ausreichend generisch ist.

ESP: RFID, Display&Touch, kann Relay schalten. Focus war darauf.
Frage: Wie löst das Modulsystem Abhängigkeiten zwischen Modulen auf?
Antwort: Bisher nicht notwendig, daher nicht implementiert. “Ich möchte aus View a Action b aufrufen” ist möglich.
Kommunikation über ActionManager ist möglich, bisher keine Kontrolle / FailSafe von Abhängigkeiten.

Wenn Maschinen eingeschaltet werden, soll angezeigt werden: “Kuck, ob der Wiederanlauf-Schutz aktiv ist!” usw. um Verletzungen und Schaden zu vermeiden.
Ergänzung aus dem VoW-Treffen:
Fast jeder hat ein SmartPhone, Tablets sind teuer --> QR-Code an Maschinen. Aktionen über SmartApps.

dequbed: tmate.io
DFBH
cap’n proto als Stichwort …
Rest zugeschaut - Fazit: durchdacht, generisch, ziemlich cool. Kann man nicht so gut verkaufen, ist aber gut gemacht.
Cap’n Proto stellt sicher, dass Schnittstellen zwischen Sprachen funktionieren. Es werden Schemata definiert und zur Verfügung gestellt, die Nachrichten sind sprachübergreifend sichergestellt definiert.
Das Tool generiert sprachspezifischen Code, der die Schnittstellen einbindet.
Vergleichbar mit Protocol-Buffer, aber mächtigere RPC. Einfachere bidirektionale Interfaces.

Marcus: Muss RG1 fixen --> schaltet auf ListenOnly.

Wie bekommen wir hin, dass Fabian & Marcus bei dem coolen Shit, den dequbed “hingeworfen” hat mitkommen? – Busfaktor sollte 3 sein, nicht 1.

Zu klären: Kommunikation Front-/Backend
Varianten: Websockets / Server dazwischen.

Fabi: Für APIs, die wir nach außen zur Verfügung stellen, sollten wir auf REST setzen. Geringe Einstiegshürde.
0815-User kann mit Cap'n Proto jeweils Schnittstellen erstellen, die sie als Start nutzen können.
a) Dokumentation für User, die mit CP dann ihre Schnittstellen erstellen können.
b) Wir maintainen eine Python-Lib, die die Schnittstelle abbildet.

Modul-Implementation:
Bei V1.0: FabLab braucht Hue-Lampen.
MQTT ist im Kern eingebaut. Wir können MQTT sprechen. Du (externer User) schreibst in Python eine Implenentation, die nur Encoding macht … erstellst also eine Message, die von uns über MQTT versendet wird.
Wie können wir mit anderen Sprachen interagieren? - Ruby,C,Python,JS, … Kommunikation mit dem Kern soll möglichst einfach werden.

Aufwand ist "gigantisch" Code soll am Ende nicht mehr von uns geschrieben werden, sondern von Usern. Wir kümmern uns um alles, was im Kern stattfindet. Bieten EINE Schnittstelle an, aber nicht mehr.
Alles was kaputt gehen kann, muss in Rust geschrieben werden, damit ich wenigstens mitbekomme, wenn es kaputt gegangen ist.

Supergute Dokumentation ist *extrem* wichtig. Evtl. weniger Aufwand, Dokumentation für eine Schnittstelle gut zu machen als 10 Schnittstellen zu maintainen.

dequbed wünscht sich Backend mit 1 Schnittstelle zum FrontEnd.
Gedanken über die API machen.

Testweise, exemplarisch schauen, wie das mit der Anbindung Cap'n Proto funktioniert.
dequbed kann den (rudimentären) Server zum testen ins Netz werfen.

Lizenzen…
MIT - Copyleft … erhöht die Innovativität …
OSI-Lizenz ist Voraussetzung.
Ist (…teilweise …evtl.) von der öffentlichen Hand finanziert.
Gregor: APIs dürfen nicht kommerzialisiert werden. Alles andere … kann und soll diskutiert werden.

Wir machen eine Umfrage - stoßen eine Diskussion an … evtl. MultipleChoice-Umfrage in Telegram.
AGPL-3.0 GPL3/2 EUPL-1.1 LGPL-2.1 MPL-2.0 Apache2.0 BSD3-Clause MIT CC0

Version #2
Erstellt: 13 Oktober 2024 00:45:36 von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 25 Februar 2025 21:22:13 von Mario Voigt (Stadtfabrikanten e.V.)