Grundkonzepte

Ein Einstieg in die Grundkonzepte von FabAccess.

Die Anforderungen an FabAccess sind im Laufe der Jahre rasant gewachsen. Ursprünglich sollte nur eine Drehbank mit Strom versorgt werden, nun können jedoch auch Türen oder Schließfächer verwaltet werden. Um diesen vielfältigen Anforderungen gerecht zu werden, haben wir einige Konzepte erarbeitet, mit denen wir den Prozess zum Freischalten dieser Ressourcen vereinheitlichen und für alle verfügbar machen wollen.

Bitte beachte: noch nicht alle hier festgehaltenen Grundkonzepte sind in FabAccess implementiert. Hier bedarf es Entwicklerearbeit.

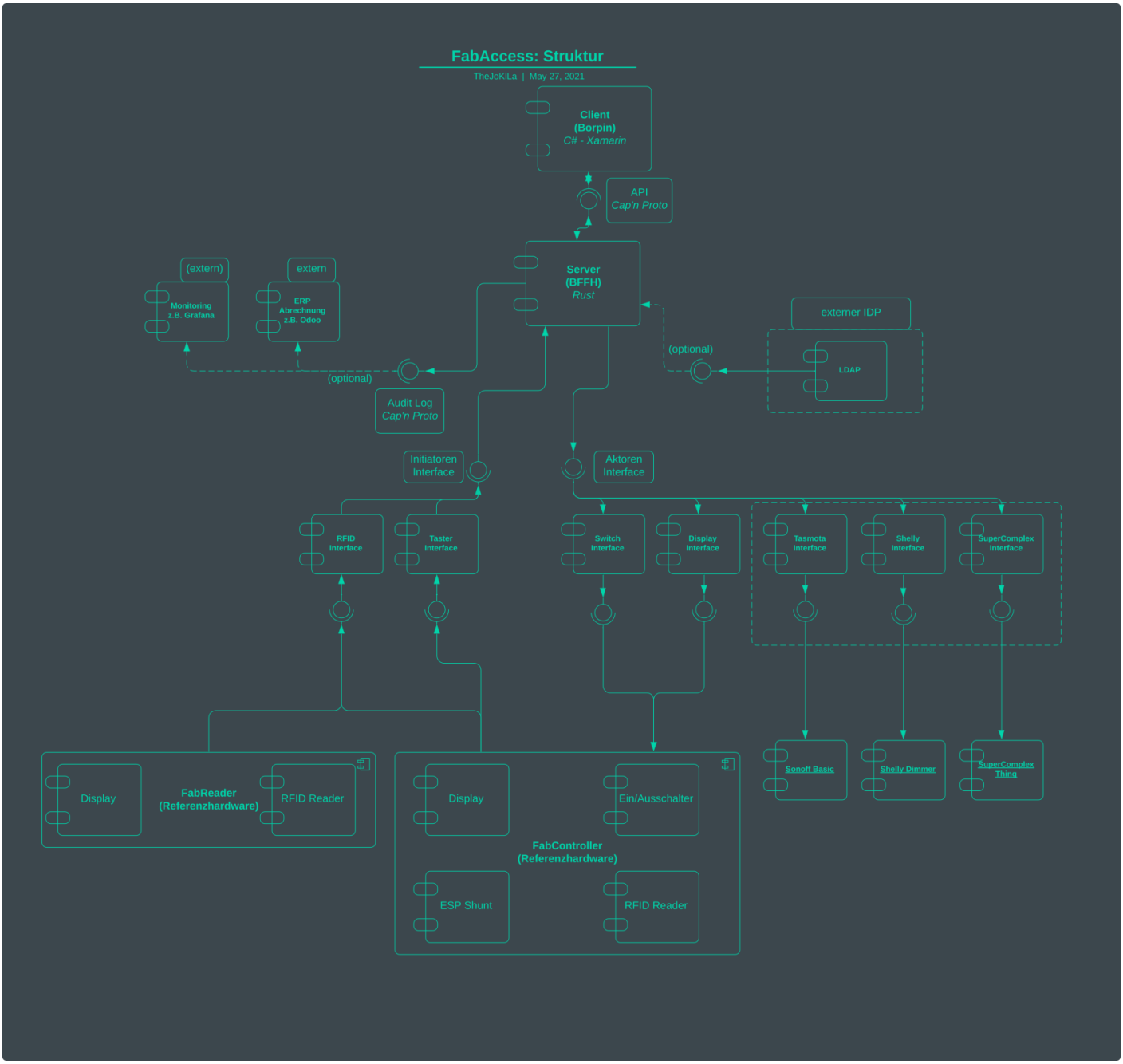
Die Implementierungen von FabAccess resultieren aus den Konzepten. Dabei müssen oft weitere Annahmen getroffen werden, wie genau die Implementierung aussehen soll. Für all diese Fälle werden hier die Entscheidungen festgehalten und möglichst nachvollziehbar erklärt, warum die Entscheidung so getroffen wurde.

Die hier aufgelisteten Konzepte haben viel mit Softwareimplementierung zu tun. Deshalb verwenden wir in den Titeln englische Begriffe, erläutern aber an geeigneten Stellen die deutsche Interpretation.

- RBAC (Benutzerrollen und Berechtigungen)
- Plugins
- Projekte
- Terminals
- Externe Authentifikation
- Nutzerverwaltung
- Traits
- URL und URN

Allgemeine Struktur

Strukturdiagramm von Joseph Langosch, Stand 27.05.2021, es zeigt die grundlegenden Komponenten und deren Interaktionen



Abhängigkeiten von Ressourcen

FabAccess unterstützt die Verwaltung von Ressourcenabhängigkeiten. Dabei werden automatisch Claims auf Ressourcen an die Nutzer vergeben, die eine Ressource beanspruchen, die von einer anderen abhängt.

Die abhängige Ressource kann zusammen mit der anderen Ressource zurückgegeben werden. Auch die Zustände der Traits werden bei abhängigen Ressourcen berücksichtigt. Auf diese Weise kann verhindert werden, dass Ressourcen wie Absaugungen oder Kühlungen ausgeschaltet werden, wenn die Ressource, die diese benötigt, noch aktiv ist.

Aktoren (Actors) und Initiatoren (Initiators)

Um FabAccess erweiterbar zu halten, basiert die Steuerung externer Geräte wie Wifi-Schaltsteckdosen oder Türschlössern auf einem Aktoren- und Initiatorenkonzept.

Aktoren (Actors)

Aktoren in FabAccess haben die Aufgabe, die digitalen Zustände von Ressourcen in reale Zustände umzusetzen. Vom Server aus werden die Übergänge der Traits (Eigenschaften) an kleine Skripte oder Prozesse weitergegeben, die entsprechend darauf reagieren. Aktoren erhalten vom Server Mitteilungen über Änderungen an Maschinen, beispielsweise wenn eine Maschine ausgeliehen wird, und passen dann den realen Zustand der Maschine an. Dadurch wird die Maschine für Nutzer freigeschaltet. Darüber hinaus ermöglichen Aktoren die Steuerung zusätzlicher Anzeigen wie Maschinenampeln oder Displays. Aktoren werden an Maschinen gebunden.

Initiatoren (Initiators)

Mit Initiatoren werden reale Zustände auf die digitalen Zustände in FabAccess abgebildet. Initiatoren sind Plugins, die den Status einer Maschine aktiv verändern können. Initiatoren werden an Maschinen gebunden. Dies ermöglicht beispielsweise die automatische Rückgabe von Maschinen, wenn sie nicht verwendet werden, das Festlegen von Zeitschaltungen oder Türkontakte, die den aktuellen Zustand der Tür übertragen können, und FabAccess kann entsprechend darauf reagieren.

Wie wird das genutzt?

Das Einbinden von Aktoren und Initiatoren geschieht in der Hauptkonfiguration `bffh.dhall`.

Eine Sammlung von Initiatoren und Aktoren ist hier zu finden.

Audit Log (Revisionsprotokoll)

Interaktionen und Ergebnisse der Verwendung von Ressourcen werden protokolliert, um sie später auswerten zu können - zum Beispiel im Fehlerfall, in der Schadensaufklärung oder für die Abrechnung der Nutzung (Nutzungsgebühren).

Über das Audit Log File

Das Audit-Log leitet alle Änderungen an Maschinen vom Server aus weiter. Ein Plugin kann diese Informationen nutzen, um Abrechnungen zu erstellen oder Maschinenzeiten grafisch auszuwerten. Diese Funktion bietet auch Möglichkeiten für eine detaillierte Analyse und Optimierung der Maschinennutzung.

Als zusätzliche Option können eigene Prozesse an die API gebunden werden. Diese ermöglichen einen direkten Zugriff auf die Funktionen des Servers über die API und eröffnen die Möglichkeit für Massenänderungen oder die zeitgesteuerte Zuweisung von Rollen.

Beim Erzeugen von Vorratsdaten ist stets dabei darauf zu achten, nur die minimal notwendigen Daten zu speichern und auszuwerten, um die Privatsphäre der Nutzer zu schützen und das Vertrauen nicht zu verletzen. FabAccess ist hier sehr sparsam und schreibt generisch je Ereignis eine Zeile im JSON-Format. Der Zeitstempel ist dabei im Unix-Format:

```
{"timestamp":<UNIX Zeitstempel>,"machine":"Maschinen-ID","state":"<Status> <Benutzer>"}
```

Bffh protokolliert Zustandsänderungen in der Audit-Log-Datei (der Pfad wird über `auditlog_path` in der Konfigurationsdatei (*.dhall) festgelegt). Ausschnitt einer Beispiel Log File:

```
{"timestamp":1726239904,"machine":"Fokoos","state":"inuse local_lab_admin"}
{"timestamp":1726239905,"machine":"Fokoos","state":"free"}
{"timestamp":1726239932,"machine":"Ender","state":"inuse local_lab_admin"}
{"timestamp":1726239933,"machine":"Ender","state":"free"}
{"timestamp":1726240081,"machine":"Ender","state":"inuse local_lab_admin"}
{"timestamp":1726240088,"machine":"Ender","state":"free"}
{"timestamp":1726240112,"machine":"Mjolnir","state":"inuse local_lab_admin"}
{"timestamp":1726240122,"machine":"Mjolnir","state":"disabled"}
{"timestamp":1726240125,"machine":"Mjolnir","state":"disabled"}
```

{"timestamp":1726240126,"machine":"Mjolnir","state":"blocked local_lab_admin"}
{"timestamp":1726240128,"machine":"Mjolnir","state":"free"}
{"timestamp":1726240132,"machine":"Mjolnir","state":"inuse local_lab_admin"}
{"timestamp":1726240134,"machine":"Mjolnir","state":"free"}
{"timestamp":1726240139,"machine":"BashMachine","state":"inuse local_lab_admin"}
{"timestamp":1726240141,"machine":"BashMachine","state":"free"}

Parsing mit JQ

```
sudo apt install jq
```

Allgemeines Parsen

```
jq . /opt/fabinfra/bffh-data/bffh.audit
```


```
{
  "timestamp": 1727110784,
  "machine": "Mjolnir",
  "state": "inuse local_lab_admin"
}
{
  "timestamp": 1729189184,
  "machine": "Mjolnir",
  "state": "free"
}
{
  "timestamp": 1729189186,
  "machine": "Mjolnir",
  "state": "inuse local_lab_admin"
}
```


UTC Timestamps

Parsen mit allgemeingültiger UTC-Zeitstempelformatierung:


```
{
  "timestamp": "2024-09-23T18:59:44 CET",
  "machine": "Mjolnir",
```

```
  "state": "inuse local_lab_admin"
}
{
  "timestamp": "2024-10-17T20:19:44 CET",
  "machine": "Mjolnir",
  "state": "free"
}
{
  "timestamp": "2024-10-17T20:19:46 CET",
  "machine": "Mjolnir",
  "state": "inuse local_lab_admin"
}
```


Mit lokaler Zeitzone

Oder zum Beispiel mit der Zeitzone `Europe/Berlin`:

```
TZ=Europe/Berlin jq '.|.timestamp |= strflocaltime("%Y-%m-%dT%H:%M:%S %Z")' /opt/fabinfra/bffh-data/bffh.audit
```


```
{
  "timestamp": "2024-09-23T18:59:44 CET",
  "machine": "Mjolnir",
  "state": "inuse local_lab_admin"
}
{
  "timestamp": "2024-10-17T20:19:44 CET",
  "machine": "Mjolnir",
  "state": "free"
}
{
  "timestamp": "2024-10-17T20:19:46 CET",
  "machine": "Mjolnir",
  "state": "inuse local_lab_admin"
}
```


oder noch lesbarer durch Umstellen des Formats:

```
TZ=Europe/Berlin jq '.|.timestamp |= strflocaltime("%d.%m.%Y %H:%M:%S Uhr")' /opt/fabinfra/bffh-data/bffh.audit
```

```
{
 "timestamp": "08.12.2024 00:57:27 Uhr",
 "machine": "zam-raum1-ecke8-macgyver",
 "state": "free"
}
{
 "timestamp": "08.12.2024 00:57:29 Uhr",
 "machine": "zam-raum1-ecke8-macgyver",
 "state": "inuse Admin"
}
{
 "timestamp": "08.12.2024 00:57:30 Uhr",
 "machine": "zam-raum1-ecke8-macgyver",
 "state": "free"
}
```


Zum Parsen und Patterns testen empfehlen wir das Online-Tool https://jqplay.org

Ausleihen and Transfer

Um die soziale Interaktion im Space zu fördern, ermöglicht FabAccess das Weitergeben von Claims. Benutzer können somit Ressourcen direkt an andere Benutzer weitergeben oder diese zum Zwecke der Ausbildung an andere Benutzer verleihen (englisch: "lending").

Durch diese Funktion kann die soziale Interaktion über FabAccess nachverfolgt und unterstützt werden.

Cache (Zwischenspeicher)

In FabAccess gibt es eine Unterscheidung zwischen statischen und dynamischen Daten, die von Ressourcen besessen werden.

Dynamische Daten umfassen Zustände oder Messwerte, die von Ressourcen erzeugt werden.

Statische Daten hingegen sind Eigenschaften, die sich nicht regelmäßig ändern, sondern nur in größeren zeitlichen Abständen. Diese statischen Daten werden in FabAccess zwischengespeichert, um die Antwortzeiten des Servers zu reduzieren und Ressourcen in Clients schneller anzeigen zu können.

Cap'n Proto API

Die API von FabAccess basiert auf Cap'n Proto. Cap'n Proto wird verwendet, um eine erweiterbare und bidirektionale API bereitzustellen. Dank der Codegenerierungsfunktion von Cap'n Proto kann die API in jede Programmiersprache portiert werden. Jedes Interface wird in einer übersichtlichen Schema-Sprache dargestellt, was die API selbst dokumentierend macht. Die Verfügbarkeit von Interfaces im Client kann durch einfache Null-Checks überprüft werden.

Authentifizierung

Um einen übergeordneten Zugangspunkt zu bieten, ist die API in Systeme unterteilt, die erweitert und ausgetauscht werden können. Der Einstiegspunkt in die API ist das Bootstrap-Interface, das jedem Client angeboten wird. Nach der Authentifizierung werden die für den Client relevanten Interfaces bereitgestellt.

Scripting

Die API soll anderen Clients oder Skripten einen stabilen Zugang zu Ressourcen ermöglichen und die Zusammenarbeit zwischen Systemen fördern. Der Server entscheidet selbst, welche Ressourcen und Interfaces er den Clients zur Verfügung stellt, je nach Berechtigung und Konfiguration. Die Clients müssen daher in der Lage sein, damit umzugehen.

Ressourcenzugang

Durch die API und die damit verbundenen Anforderungen soll der Zugang zu Ressourcen für alle erleichtert und die Zusammenarbeit gefördert werden. Diese Grundlagen sollen dazu beitragen, ein föderiertes System für den Ressourcenverleih aufzubauen.

Details zu Cap'n Proto finden sich auf deren Homepage: https://capnproto.org und

https://github.com/capnproto/capnproto

Warum gibt es keine REST API?

Der einfachste Punkt, weswegen REST eher ungeeignet ist der, dass unsere Verbindung bidirektional ist, also beide Seiten zu jedem Zeitpunkt Daten an die jeweils andere schreiben können. Das würde WebRTC prinzipiell problemlos ermöglichen (Datenaustausch z.B. per JSON), aber eine einfache TCP- oder QUIC-Verbindung ebenso

und mit wesentlich weniger Setupaufwand. Sobald es um WebRTC geht, sind in einem webbasierten Client große Mengen angepasster JavaScript-Code nötig. Ist das der Fall, könnten diese TCP-/QUIC-Verbindungen über Websockets tunneln und die bestehende Cap'n Proto API ansprechen.

Cap'n Proto API ansprechen

Eine Übersicht über verschiedene Sprachen: https://capnproto.org/otherlang.html

Claims, Notify und Interest (das Konzept vom "Anspruch erheben")

Das Konzept von Claims dient als Abstraktion des Verleihens einer Ressource in FabAccess. Ihr Hauptzweck besteht darin, die Möglichkeiten zu verwalten, wie Benutzer Zugriff auf eine Ressource erhalten und diesen Zugriff untereinander teilen können bzw. wie Maschinen voneinander abhängig gemacht werden. Sie vereinfachen den Ausleihprozess für den Space und dessen Nutzer. Claims enthalten sog. "Interests" und "Notifies".

Zum Beispiel kann konfiguriert werden, dass eine Bandsäge nur angeschalten werden kann, wenn auch die Absaugung an ist oder der Laserschneider nur einschaltbar ist, wenn die Kühlung aktiv ist.

Claim (Anspruch erheben)

Ein "Claim" in FabAccess repräsentiert den gewährten Zugriff auf eine Ressource.

Die Flexibilität von Claims ermöglicht es, verschiedene Szenarien des Ressourcenzugriffs effektiv zu unterstützen. Zum Beispiel können mehrere Benutzer gleichzeitig einen Claim für eine Ressource erhalten, was besonders in Umgebungen mit kollaborativem Arbeiten von Vorteil ist. Darüber hinaus bietet die Möglichkeit, sich in eine Warteschlange mit einem Interest einzutragen, eine praktische Lösung für Situationen, in denen die Verfügbarkeit einer Ressource begrenzt ist und Benutzer darauf warten müssen, darauf zugreifen zu können.

in weiterer wichtiger Aspekt von Claims ist ihre Fähigkeit, Ressourcen zwischen Benutzern zu transferieren oder auszuleihen. Diese Funktionen ermöglichen eine flexible Nutzung der Ressourcen und fördern die Zusammenarbeit zwischen den Benutzern. Zum Beispiel kann ein Benutzer, der eine Ressource nicht mehr benötigt, diese einem anderen Benutzer übertragen, der sie gerade benötigt, oder ein Ausbilder kann einem Auszubildenden vorübergehend Zugriff auf eine Ressource gewähren, um bestimmte Fähigkeiten zu erlernen.

Interest (Interesse anmelden)

Ein wichtiger Bestandteil des Claims-Konzepts sind "Interests", die Reservierungen abbilden, die entweder zeitbasiert oder auf einer Warteschlange basieren können. Diese

Interessen bieten den Benutzern die Möglichkeit, einen zukünftigen Zugriff auf eine Ressource zu sichern, entweder basierend auf einer vordefinierten Zeit oder in der Reihenfolge des Eintritts in die Warteschlange.

Mit einem Interest signalisiert der Nutzer dem System sein Interesse an einer bestimmten Ressource. Bei der nächsten Gelegenheit erhält der Nutzer entweder einen Claim auf die Ressource oder hat die Möglichkeit, seinen Interest auf einen Claim zu aktualisieren. Diese Flexibilität ermöglicht es den Benutzern, ihre Bedürfnisse anzupassen und auf Änderungen in der Verfügbarkeit der Ressourcen zu reagieren.

Notify (Benachrichtigen)

Das letzte Element des Claims-Konzepts ist der "Notify". Durch den Notify können Nutzer den Status einer Ressource einsehen und sich über Änderungen benachrichtigen lassen.

Der Notify ermöglicht es Benutzern, den aktuellen Zustand einer Ressource abzurufen und bei Bedarf auf Änderungen zu reagieren. Dies bietet eine wichtige Möglichkeit, den Zustand von Ressourcen zu überwachen und zeitnah auf relevante Ereignisse zu reagieren.

Durch die Möglichkeit, sich für Benachrichtigungen über Zustandsänderungen zu registrieren, können Benutzer effektiv mit den Ressourcen interagieren und sicherstellen, dass sie stets über wichtige Entwicklungen informiert sind.

Federation (Föderation)

FabAccess ist als selbstständiges selbstgehostetes System für jeden Space gedacht, so können die Spaces selber über das komplette System verfügen und es eigenständig bis in kleinste Detail konfigurieren. Um die Zusammenarbeit von Spaces zu ermöglichen und zu fördern, können die einzelnen FabAccess Instanzen (BFFH) mit einander föderieren. Dadurch können Nutzer zwischen diesen Spaces mit verringertem Verwaltungsaufwand hin und her wechseln. Wir wollen so die Spezialisierung von Spaces ermöglichen, ohne dass Nutzer die Vorteile eines vollausgestatten Spaces verlieren. Auch kleineren Spaces wird so der Einstieg vereinfacht.

Föderation ist dabei als aktive Zusammenarbeit erdacht, also müssen sich die Betreiber dieser Spaces dazu entscheiden zu föderieren. Jeder Space kann so den Partner für die Föderation selber wählen.

Der FabAccess Client Borepin ist bereits insofern für Föderation ausgelegt, als dass er mehrere Serveradressen und Benutzer speichern kann. So ist ein Wechsel zwischen verschiedenen Spaces und/oder Nutzermodellen praktisch möglich.

Vorteile und Ziele der Föderation

- beliebiger Wechsel der Nutzer verschiedener Spaces trotz technisch getrennter FabAccess-Instanzen
- Zusammenarbeit von Spaces unterstützen - durch Teilen von Berechtigungen ihrer Nutzerschaften
- FabAccess verbreiten, ähnlich wie das Matrix Chat-Protokoll dies mit Kommunikation macht
- kann langfristig dabei helfen, Maschinendokumentationen und Maschineneinweisungen (Maschinenführerscheinen) zu vereinheitlichen
- Spart NFC Karten (FabCards) ein, denn die Karten können dann in alle föderierten Spaces genutzt werden
- Mit Föderation is theoretisch auch das Bilden von standortübergreifenden Clustern möglich, also eine Art Filialbildung (das ist ein wertungsfreier Fakt)
- ist dabei als aktive Zusammenarbeit erdacht: die Betreiber der Spaces können sich selbst entscheiden, ob sie föderieren wollen. Föderation ist also immer optional und kann zudem jederzeit wieder deaktiviert werden

Modi oder Stufen der Föderation

Die folgenden Stufen bauen aufeinander auf, so können sich Betreiber einfacher an eine Föderation wagen und die Vorteile mit jeder Stufe erfahren.

1. Stufe (Authentifizierung: Teilen von Nutzern)

- Nutzer aus anderen Spaces können ihre FabCard auch in deinem Space verwenden
- Die Nutzer sind auch in deinem Space registriert
- Der föderierte Space übernimmt die Authentifizierung für den anderen Nutzer (wie bei SSO)

Unter dem Teilen von Nutzern verstehen wir, dass Nutzer von Space B bei dir im Space A sich mit der FabCard aus Space B authentifizieren können. Um bei dir im Space A die Maschinen nutzen zu dürfen, müssen die Nutzer aus Space B sich bei dir anmelden und deinen Nutzungsvertrag ausfüllen. Da die Nutzer aus Space B FabAccess schon kennen, werden Sie sich selbstständig registrieren und du musst nur nach einer kurzen Prüfung der Daten diese Nutzer akzeptieren. Dabei erhältst du immer die Kontaktdaten deiner Nutzer und bist nicht auf die Nutzerdaten des anderen Spaces angewiesen, sollte bei dir mal ein Unfall passieren. Die Authentifizierung der Nutzer übernimmt dann die BFFH Instanz des Space B. Um nicht immer mit jedem Space sofort aktiv föderieren zu müssen, ermöglichen wir es dir auch das Teilen von Nutzern für unbekannte Spaces zu erlauben. So kannst du nach einer aktiven Föderation mit dem anderen Space gleich die Nutzerdaten übernehmen, um so nicht noch mal alle Nutzer anzupassen. Bei einer Föderation sind nämlich die IDs der Nutzer in den beiden Spaces gleich und ermöglichen den nächsten Schritt in der Föderation.

2. Stufe (1. Stufe ergänzt um Teilen von Berechtigungen)

- Nutzer aus anderen Spaces können ihre FabCard und ihre Gruppen auch in deinem Space verwenden
- Die Nutzer sind auch in deinem Space registriert
- die Gruppen des anderen Space können in deinem Space verwendet werden, wenn zum Beispiel die gleichen Maschinen im Besitz sind. Das spart Zeit bei der Einweisung und der Dokumentation.

Nach dem das Teilen von Nutzern erfolgt ist und du sich mit dem Betreiber des Space B gut verstehst, stellt ihr beide fest, dass ihr einige gleiche Maschinen in euren Spaces besitzt. Da du dem Betreiber von Space B auch bei den Einweisungen von den Maschinen vertraust, könnt ihr in der 2. Stufe auch Berechtigungen teilen. Somit müssen Nutzer von Space B, die bei dir die gleiche Maschine wie im Space B verwenden wollen, von dir nicht noch mal eingewiesen werden. Das spart Zeit bei dir und bei den Nutzern von Space B. Du

behältst dabei die Kontrolle, welche Gruppen von Space B bei dir im Space A welche Maschinen nutzen können, somit kannst du dir sicher sein, dass die Nutzer wirklich eingewiesen sind.

3. Stufe (2. Stufe ergänzt um Teilen von Abrechnungen)

- Nutzer aus anderen Spaces können ihre Karte und ihre Gruppen auch in deinem Space verwenden
- Die Abrechung für diese Nutzter übernimmt dabei der andere Space für deinen Space. Über beispielsweise monatlich gesammelte Differenzrechnungen kann gegen den andere Spacecs abgerechnet werden. Das spart Zeit bei der Buchführung und die Nutzer können auch Ihre Rechnungen besser im Blick behalten.

Das Teilen von Abrechnungen stellt die letzte Stufe der Föderierbarkeit da. Da wir uns bei dem Thema noch nicht vollständig sicher sind, inwieweit das umgesetzt werden kann, steht es noch in Klammern. Die Idee zu der Föderation ist, dass Nutzer von Space B, die bei dir eine kostenpflichtige Maschine nutzen, ihre Abrechnung von Space B erhalten und nicht von dir. Das Geld der Nutzer von Space B kannst du dann mit dem Betreiber von Space B zum Beispiel monatlich verrechnen. Der Vorteil für den Nutzer liegt dabei darin, dass sie nicht bei jedem Space ihre Abrechnung machen müssen, sondern gesammelt bezahlen können.

Wie sicher ist Föderation?

Föderation benötigt Vertrauen, jedoch möchten wir nicht, dass dieses Vertrauen einfach missbraucht werden kann. Daher ist der oben beschriebene Ansatz, dass du jeden deiner Nutzer auch kennst.

Sicherheit des Servers

Für den Austausch von Benutzern und Berechtigungen ist ein sicherer Datenaustausch über das Internet notwendig. Das bedeutet auch, dass ein Dienst Ziel eines Angriffs von außen sein kann. Sobald der Dienst auf einem geöffneten Port lauscht, muss sichergestellt sein, dass nur authorisierte Server bzw. Personen darauf Zugriff haben und keinen Missbrauch begehen. Daraus ergeben sich verschiedene Fragen und Anforderungen, wie zum Beispiel

- erfolgt der Informationsaustausch zwischen den föderierenden Servern auf dem gleichen Port oder gibt es hierfür z.B. eine https-verfügbare API (z.B. REST, Json, XML RPC, ...)?

- wie wird das Verhalten gemonitored und Schadverhalten abgewehrt (z.B. Greylisting, Blacklisting, DDoS Vermeidung, Bruteforce-Login Attacken, Fail2Ban Jails, Firewall-Regeln...)
- welche APIs sind unterinander kompatibel? Nicht jeder Space wird schnell genug seinen Server auf die aktuelle Version updaten können
- Wie isolieren wir den FabAccess-Server ausreichend gegen Angriffe von außerhalb? Immerhin hat bffh Zugriff auf physische Geräte durch das Schalten von Strom an Aktoren

Sicherheit von FabFire Karten

Um die Authentifizierung sicher zu gestalten, haben wir uns für die etwas teureren NXP MIFARE DESFire EV2 Karten entschieden (ca. 1-5 € pro Karte bzw. Tag). Diese haben den Vorteil, dass sie ein Ende-zu-Ende (E2EE) Verfahren unterstützen und bisher als ungehacked gelten. Somit musst Du nicht die Schlüssel deiner Karten an jeden Reader in einem anderen Space verteilen, sondern deine BFFH Instanz kommuniziert verschlüsselt direkt mit der Karte. Auf den FabCards befinden sich auch keine Informationen der Nutzer, sondern nur der DNS-Eintrag deines Space sowie eine pseudonymisierte Nutzer ID. Somit wollen wir die technische Seite der Föderation absichern und eine Zusammenarbeit möglichst einfach gestalten.

Die Anforderungen an ein föderierbar einsetzbares SmartCard-System sind hoch, da so die jeweils andere Partei keine einfache Validierung der Karte durchführen kann. Auch das Authentifizieren mit einem Schlüssel gestaltet sich über Föderationen hinweg schwierig, da der Reader dem einen Space gehört und die Karte dem anderen. Ein Reader hat daher nicht immer die Schlüssel für die gelesenen Karten. Die DESFire-Karte besitzt ein Feature, mit dem eine Karte via OTA-Verfahren (Over-the-Air) direkt mit einem entfernten Server kommunizieren kann, ohne dass der Reader Schlüssel für die Karte benötigt. Der Reader übernimmt dabei nur die Rolle eines Proxys (Brücke/Vermittler). Mit diesem Ansatz stellen wir ein sicher föderierbares Kartensystem, was wir so bisher noch nicht im OpenSource-Bereich finden konnten.

Fragen und Ansprüche an Föderation und potentielle Probleme

- muss stabil sein
- es muss sicher sein (siehe oben)

- Benutzer und Berechtigungen anderer Spaces können sich ändern. Nutzer könnten dem Space nicht mehr angehören. Wie halten wir den Informationsstamm ausreichend aktuell?
- es müssen verschiedene Server-/API-Versionen interoperabel funktionieren
- es braucht eine Whitelist bzw. ggf. sogar Blacklist für alle Spaces, die für FabAccess nutzbar sein sollen
- Vertrags-, Datenschutz- und Haftungsfragen: ist die Übertragung der Nutzerdaten an andere Spaces geregelt, und wenn ja, wie? Welche vertraglichen Regeln müssen für eine Föderation mit FabAccess festgehalten werden?
- Föderierte Spaces arbeiten ggf. sehr unterschiedlich, was Maschineneinweisungen und Co. angeht - wie kann sichergestellt werden, dass dazu übergreifende Standards geteilt werden?

Stand der Föderation

> Das Feature Föderation ist aktuell (Stand Dezember 2024) noch nicht in FabAccess implementiert.

Wer nutzt heute schon FabAccess?

Potentielle Nutzer für FabAccess-Föderation findest du unter Mitmachen / Unterstützen / Join the Community

Measurements (Messwerte)

Durch "Measurements" in FabAccess werden Daten von Ressourcen gesammelt, um deren Leistung und Nutzung zu erfassen. Das Ziel ist es, dass FabAccess diese Messwerte sammelt und an die Nutzer weitergibt.

Die Funktion Measurements ermöglicht es FabAccess, wichtige Daten über die Nutzung von Ressourcen zu erfassen und den Nutzern zur Verfügung zu stellen. Dies umfasst Informationen wie Betriebsstunden, Auslastung und andere Leistungsindikatoren, die es den Nutzern ermöglichen, die Effizienz und Produktivität ihrer Arbeitsprozesse zu optimieren.

Hinweis: Das Konzept zu Measurements exitiert zwar, jedoch gibt es noch keine Spezifikation!

RBAC (Benutzerrollen und Berechtigungen)

FabAccess verwendet eine Role-Based Access Control (RBAC)-Struktur zur Verwaltung von Berechtigungen. Dabei werden Berechtigungen Maschinenrollen zugewiesen, und diese Rollen werden dann den Benutzern zugewiesen. Auf diese Weise lässt sich ein komplexes Berechtigungssystem einfach, ufassend und flexibel abbilden.

Das Berechtigungssystem ermöglicht es, die Zugriffe auf Maschinen nur für bestimmt Nutzer zu erlauben. Somit kann darüber die Einweisungen für gefährliche Maschinen abgebildet werden. Auch die Abbildungen von einfachen Maschinen, die keine Einweisung benötigen, ist möglich.

Berechtigungsübersicht

Die Resourcen haben folgende Berechtigungen, die zugewiesen werden und dem Nutzer folgendes ermöglichen:

disclose (offenlegen)
die Resource über die API zu entdecken, also in einer Liste zu erhalten. Mit dieser Berechtigung können Nutzer die Maschine in der Maschinenliste der App sehen. Ohne diese Berechtigung wird die Maschine in der Liste nicht angezeigt.

read (lesen)
statische Informationen aus der Resource erhalten**.** Diese Berechtigung ermöglicht es Nutzern, Informationen zur Maschine abzurufen. Dazu gehört das Auslesen des aktuellen Zustands der Maschine und das Lesen weiterer Informationen wie Wiki-Links. Es ist wichtig zu beachten, dass ein Nutzer ohne disclose, aber mit read, die Maschine dennoch auslesen kann. Diese Funktion kann z.B. durch QR-Codes auf der Maschine als Präsenzfunktion genutzt werden.

write (schreiben)
Die write-Berechtigung ermöglicht es einem Nutzer, die Maschine zu verwenden. Auch das Ändern von Zuständen der Maschine wird dadurch möglich.

manage (verwalten)
einen claim zu überschreiben. Mit dieser Berechtigung kann die Maschine verwaltet werden. Ein Nutzer mit dieser Berechtigung kann eine verwendete Maschine freigeben oder als defekt markieren.

Nicht implementierte Berechtigungen

Folgende Berechtigungen sind konzipiert, aber aktuell nicht implementiert:

notify (benachrichtigen)
Notify aufrufen, um statische Informationen zu erhalten

claim (Anspruch erheben)
Die Resource leihen und Traits auszuführen

transfer (transferieren)
Die Resource an einen anderen Nutzer weitergeben

lend (ausleihen)
Die Resource an andere verleihen, die nicht die Berechtigung haben die Resource zu verwenden

Plugins

Die modulare Struktur von FabAccess benötigt Plugins, um das System an die Bedürfnisse des Spaces anzupassen. Plugins ermöglichen die Erweiterung der Grundfunktionen von FabAccess und die Integration zusätzlicher Features, ohne dass die Kompatibilität des Kernsystems beeinträchtigt wird.

Plugins können an folgenden Hauptstellen eingesetzt werden:

- Aktoren und Initiatoren
- Audit Log (Revisionsprotokoll)

Projekte

"Projects" in FabAccess sollen die Zusammenarbeit zwischen den Nutzern fördern und gleichzeitig die Abrechnung von Maschinenzeiten verbessern. Ein wichtiger Aspekt der Projektstruktur ist die Möglichkeit für Nutzer, Claims innerhalb desselben Projekts miteinander zu teilen, um gleichzeitig auf Ressourcen zugreifen zu können.

Durch die Zuweisung von Nutzern zu Projekten können Teams effizient zusammenarbeiten und ihre Ressourcen optimal nutzen. Das Teilen von Claims innerhalb eines Projekts ermöglicht es den Teammitgliedern, nahtlos auf benötigte Ressourcen zuzugreifen und gemeinsam an Projekten zu arbeiten. Darüber hinaus erleichtert diese Funktionalität die Abrechnung von Maschinenzeiten, da die Nutzung der Ressourcen innerhalb eines Projekts besser nachverfolgt und zugeordnet werden kann.

Hinweis: Das Konzept zu Projekten exitiert zwar, jedoch gibt es noch keine Spezifikation!

Terminals

Terminals in FabAccess bieten einen eingeschränkten Zugang zum Server. Diese Terminals können nur auf die ihnen zugewiesenen Maschinen zugreifen und haben die Möglichkeit, Maschinen an andere Benutzer auszuleihen.

Aufgrund ihrer eingeschränkten Zugriffsrechte sind Terminals ideal für die Authentifizierung in FabFire. Durch die Verwendung von Terminals als Authentifizierungsmethode können Benutzer sicherstellen, dass nur autorisierte Personen auf die Ressourcen zugreifen und diese nutzen.

Die Verwendung von Terminals zur Authentifizierung bietet eine zusätzliche Sicherheitsebene und hilft, die Integrität des Systems zu wahren. Darüber hinaus ermöglicht es eine effiziente Verwaltung der Ressourcennutzung, indem der Zugriff nur autorisierten Benutzern gewährt wird.

Externe Authentifikation

Das Authentifizieren in FabAccess basiert vollständig auf SASL und unterstützt daher verschiedene Mechanismen.

Die Rollen, die aus den Gruppen bei LDAP und OAuth abgeleitet werden, werden additiv zu denen betrachtet, die intern in FabAccess vergeben werden.

Authentifikation per OAuth

> In Entwicklung seit 10/2024 - Ansprechpartner: Jonathan Krebs

Authentifikation per LDAP

> In Entwicklung seit 10/2024 - Ansprechpartner: Jonathan Krebs

Es besteht keine native LDAP Integration. Es gibt jedoch Tools zum Importieren von LDAP-Benutzern in die bffh-Datenbank. Siehe LDAP Anbindung.

Authentifikation per FabFire

FabFire ist eine Entwicklung, die auf NXP Mifare DESFire basiert und die Anwendung von Karten nutzt, um Benutzer über das OTA (Over the Air)-Verfahren zu authentifizieren.

Nutzerverwaltung

Die Nutzerverwaltung ermöglicht es, den Überblick über die Nutzer im Space nicht zu verlieren. FabAccess ist als SmartCard System gedacht, somit erhält jeder Nutzer eine NXP Mifare DESFire (FabCard) Karte, um sich an der Maschinen zu authentifizieren.

Um den administrativen Aufwand zu verringern, können sich Nutzer selbstständig anmelden und können dann nach der Prüfung der Daten durch den Betreiber freigeschaltet werden. Die Schließung eines Nutzungsvertrages wird dabei abgebildet.

Mit der FabCard können Nutzer sich sowohl an den Maschinen authentifizieren als auch an unserem Client Borepin.

Traits

Traits bieten die Möglichkeit, den Zustand von Ressourcen zu ändern. Ressourcen können mehrere Traits besitzen und diese kombiniert nutzen. Mit Traits erhalten Nutzer Zugriff auf die Ressource, nachdem sie einen [Claim](https://docs.fab-access.org/books/fabaccess-setup/page/claims-notify-und-interest-das-konzept-vom-anspruch-erheben "Claims, Notify und Interest (das Konzept vom "Anspruch erheben")") erhalten haben. Dabei können Traits verwendet werden, um Ressourcen aus bestehenden Traits zusammenzusetzen oder spezifische Traits zu implementieren. Um eine optimale Anzeige der Traits für Nutzer in Clients zu ermöglichen, kann einer Ressource ein "Hint" hinzugefügt werden. Dieser ermöglicht es einem Client, eine verbesserte Ansicht der Ressource für Nutzer zu generieren. Traits werden anhand einer OID (Object Identifier) bereitgestellt. In FabAccess gibt es bereits vordefinierte Traits für grundlegende Funktionen, mit denen viele Zustände von Ressourcen abgebildet werden können.

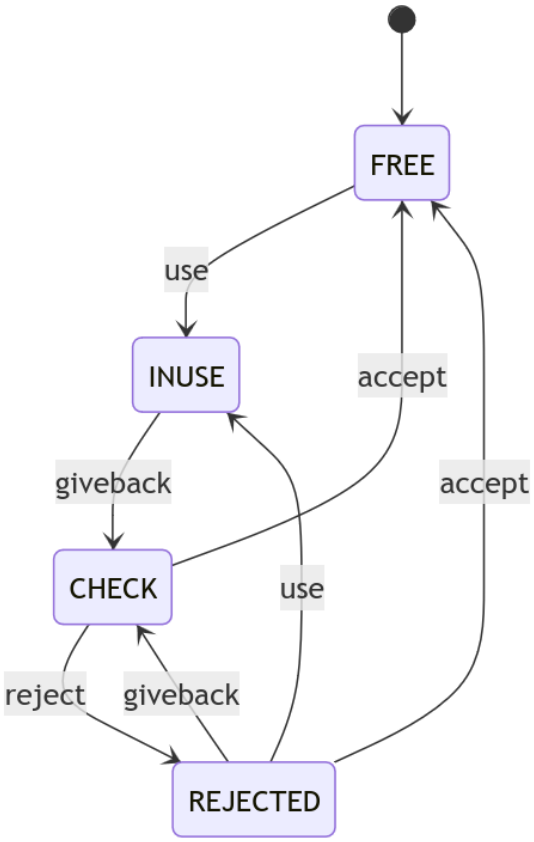
Übersicht über die vorhandenen Traits

Checkable

Der komplexere Trait "Checkable" ermöglicht die Abbildung von Ressourcen, die nach der Benutzung überprüft werden müssen. Bei einer Überprüfung durch einen berechtigten Nutzer kann die Ressource entweder für alle wieder freigegeben oder zurückgewiesen werden. Falls die Ressource zurückgewiesen wurde, kann der ursprüngliche Nutzer die Ressource weiterhin verwenden oder erneut zur Überprüfung einreichen, nachdem die Fehler behoben wurden.

OID `1.3.6.1.4.1.61783.612.1.3`

States



als Mermaid

```
stateDiagram
    [*] --> FREE
    FREE --> INUSE: use
    INUSE --> CHECK: giveback
    CHECK --> FREE: accept
    CHECK --> REJECTED: reject
    REJECTED --> INUSE: use
    REJECTED --> CHECK: giveback
    REJECTED --> FREE: accept
```

Claimable

Der Trait "Claimable" stellt einen Sonderfall dar, da er dazu dient, dass sich ein Nutzer über den Claim-Zustand einer Ressource informieren kann.

Nutzer können auf diesem Trait keine Aktionen ausführen, sondern lediglich den Zustand abfragen.

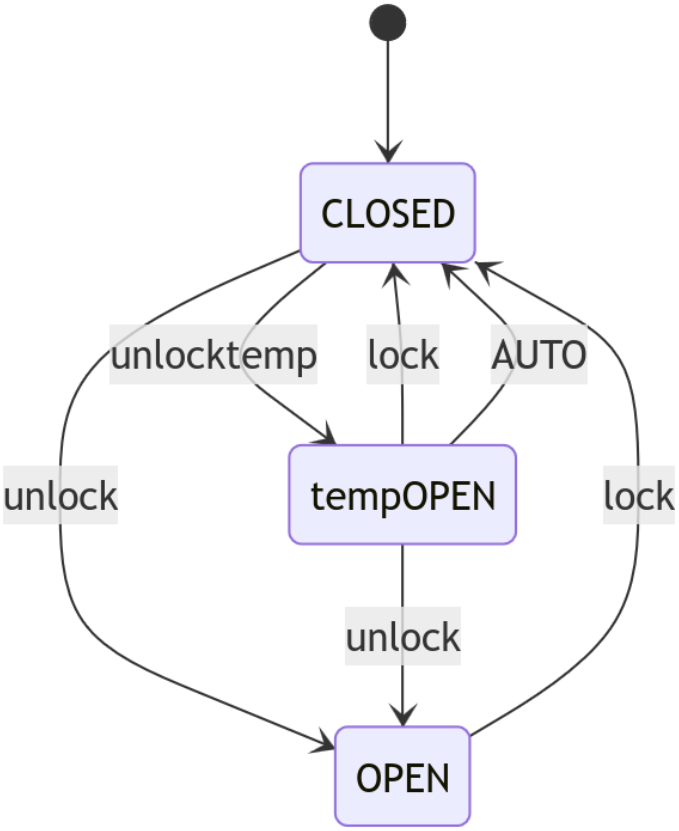
OID 1.3.6.1.4.1.61783.612.1.0

Doorable

Der Trait "Doorable" ermöglicht die Abbildung von Türen oder anderen Schließsystemen. Dabei besteht die Möglichkeit, kurzzeitige Öffnungen zu realisieren. Die genaue Zeitdauer, für die die Ressource geöffnet wird, wird dabei vom Server bestimmt.

OID 1.3.6.1.4.1.61783.612.1.2

States



als Mermaid

```
stateDiagram
    [*] --> CLOSED
    CLOSED --> OPEN: unlock
    OPEN --> CLOSED: lock
    CLOSED --> tempOPEN: unlocktemp
```

```
tempOPEN --> OPEN: unlock
tempOPEN --> CLOSED: lock
tempOPEN --> CLOSED: AUTO
```

Locatable

Der Trait "Locatable" ermöglicht die Identifizierung von Ressourcen, wie beispielsweise Schließfächer oder 3D-Drucker in einer Druckerfarm. Dabei kann entweder eine kurzfristige Identifikation abgegeben werden oder die Identifizierung dauerhaft gesetzt werden.

OID `1.3.6.1.4.1.61783.612.1.5`

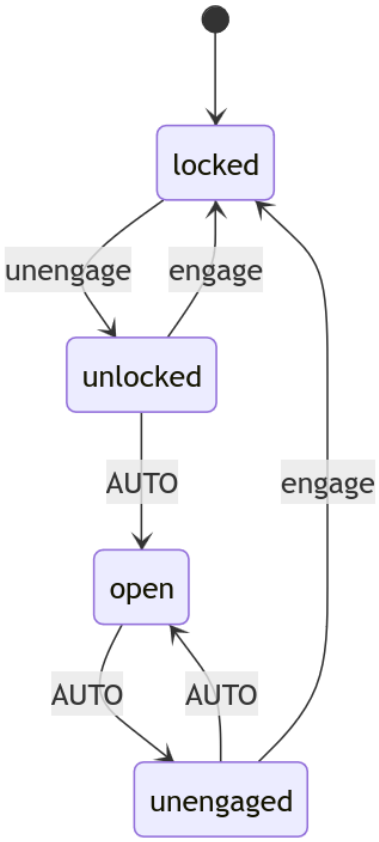
Lockers

"Lockers" ist einer der komplexeren Traits, mit dem Schlösser von Ressourcen genauer abgebildet werden können. Der Nutzer kann die Ressource nicht in jedem Zustand zurückgeben; erst wenn alles korrekt zurückgegeben wurde, kann die Ressource auch wieder gesperrt werden.

Die Zustandsänderungen zwischen den vom Nutzer verwendbaren Übergängen können von einem Initiator herbeigeführt werden.

OID `1.3.6.1.4.1.61783.612.1.4`

States



als Mermaid

```
stateDiagram
    [*] --> locked
    locked --> unlocked: unengage
    unlocked --> locked: engage
    unlocked --> open: AUTO
    open --> unengaged: AUTO
    unengaged --> locked: engage
    unengaged --> open: AUTO
```

Powerable

"Powerable" ist der grundlegendste Trait, den FabAccess unterstützt. Er dient dazu abzubilden, ob eine Ressource eingeschaltet bzw. mit Strom versorgt ist.

OID `1.3.6.1.4.1.61783.612.1.1`

Hinweis: Die Mermaid-Diagramme sind mit https://mermaid.live gerendert und als PNG exportiert und hier importiert worden, da BookStack keinen integrierten Mermaid Renderer besitzt.

URL und URN

Ressourcen in FabAccess werden durch einen Namen dargestellt. Um jedoch die Suche nach Ressourcen zu verbessern, können diese Referenzen als URN (Uniform Resource Name) oder URL (Uniform Resource Locator) geschrieben werden.

Die URN ist ein Identifikator ohne Bezug zum Space, während die URL auch den Space referenzieren kann, um einen Austausch von Ressourcen zu ermöglichen.

Genutzte Schemata in FabAccess

Folgende URN Schemata werden in FabAccess verwendet:

- Space Name: `urn:fabaccess:lab:{spacename}`
- Space Info: `urn:fabaccess:lab:{spacename}\x00{instanceurl}`
  - dieser Wert wird aktuell nicht verwendet, muss jedoch ausgefüllt werden, damit die Konfiguration `bffh.dhall` valide ist!
- Maschinen: `urn:fabaccess:resource:{machine id}`