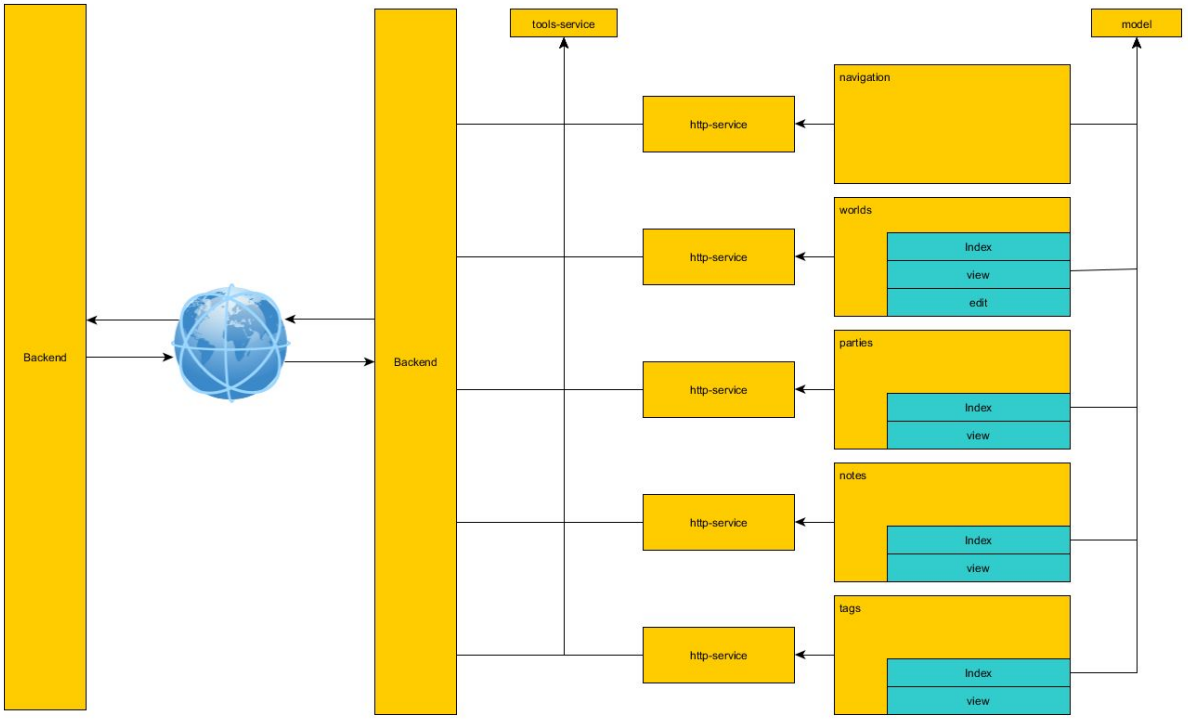
PnPBoard

Eine Webapplication zum kollektiven sammeln, verwalten und nutzen von Daten zu Pen and Paper Rollenspiel Runden.

Modul-/Komponenten-/Serviceübersicht:



Umgesetzt:

- Anlegen und Verwalten von Spielwelten
- Anlegen und Verwalten von Parties (Spielgruppen)
- Anlegen und Verwalten von Notizen
- Anlegen und Verwalten von Tags

Nicht umgesetzt (nicht geschafft):

- Charactersheeteditor
- Anlegen und Verwalten von Charactersheets
- Backendanbindung für den Login + URL-Guard
- Registrierung
- Logout
- Dialogausgaben und Usability Aspekte

Abweichende Benennung und Strategien:

Im laufe des Projekts ist es zur konkretisierung von Ideen gekommen und dem gewinn an Erfahrung und Wissen im umgang mit Angular, was wiederum zu unterschiedlichen Benennungen und Strategien geführt hat. Welche noch nicht auf das ganze System ausgebreitet werden konnte.

Zusätzliche Bibliotheken / Frameworks:

- Bootstrap
- Fontawesome

Klassen:

HttpService - generell

- Es gibt immer einen HttpService pro Modul und bildet die Anbindung an das Internet und somit auch an das Backend.
- Attribute
    - url = die grundlegende URL zum Backend
    - module = das Modul zu dem der Service gehört
    - action = die Aktion die im Backend ausgeführt werden soll
    - data = die zusätzlichen Daten die mitgeschickt werden sollen
- Funktionen / Methoden
    - getIndex() : Array<Model> **und** getIndex(id : number) : Array<Model>
        - holt die Indexliste vom Backend
    - get<Modelname>(id : number)
        - holt das modelentsprechende, spezielle Datenset vom Backend
    - edit<Modelname>(<modelname> : <model>)
        - lässt das Backend den übergebenen Datensatz auf der Datenbank aktualsieren
    - add<Modelname>(<modelname> : <model>)

        - lässt das Backend einen neuen Datensatz anlegen
    - delete<Modelname>(id : number)
        - lasst das Backend das entsprechende Datenset löschen
    - zusätzliche Funktionen
        - get<akt. Model/Modulname><Submodel>List(id : number)
            - holt ein Datenset (welches in relation zum aktuellen Model steht) vom Backend

EditComponent - generell (bei der World + ViewComponent)

- Komponente zum rendern, anzeigen und verarbeiten von Daten
- Methoden:
    - getData() : void
        - holt die Daten (über den HttpService) vom Backend
    - getTemplates() : void
        - holt die Daten (die in Relation zum zugehörigen Model stehen) (über den HttpService) vom Backend
    - onSubmit() : void
        - sendet die Daten nach der Bearbeitung (über den HttpService) an das Backend
    - je nach Modul gibt es noch Methoden die, die Templates manipulieren.
        - z.B: im Party Modul steht der Gamemaster in Relation zu den Mitgliedern, somit aktualsiert sich das Template des Gamemasters anhand der ausgewählten Mitglieder

IndexComponent - generell

- Komponente für die Index / Listenansicht
- Methoden:
    - getData() : void
        - holt die Daten (über den HttpService) vom Backend
    - delete(id : number)
        - übermittelt (über den HttpService) das zu löschende Datenset

NavbarHorizontalComponent

- baut die Navigationsleiste zusammen
- Methoden
    - getData() : void
        - holt die Daten (über den HttpService) vom Backend

ToolsService

- delSingleArrElem(key : number, arr : Array<any>)

    - nicht mehr in Nutzung
- objectToURLStr(obj : Object, basicData : string, deep : number = 0) : string
    - `baut ein Objekt zu einem URL - Array um (ist egal ob es ein einfaches oder komplexes Objekt ist)`
    -