# Demo

Link auf Präsentation

## Itemtypen

- Erstellen und Feldtypen erklären:
  Text: normaler Text, selbsterklärend (Bsp: Name)
  Nummer: normale Nummer (Integer), selbsterklärend (Bsp: Raumnummer)
  Wahr/Falsch: Wahr/Falsch-Feld, selbsterklärend (Bsp: defekt ja/nein)
  Verlinkung: Verlinkt Feldtyp mit einem anderen vorhandenen Feldtypen (Bsp: Laptop-Verlinkung auf bestimmten Raum)
  Farbe: RGB-Farbbereich, manuell wählbar von `#000000` und `#FFFFFF` (Bsp: schwarzer Laptop)
  Datei: Upload-Feld (Bsp: Rechnung zum Laptop hochladen)
  Datum: Datum kann aus Kalender ausgewählt werden (Bsp: Datum, wie lange der Laptop bereits im Raum ist)
- Benötigt (Markierung Feld ist Pflichtfeld)
- Einzigartig (Nur ein Objekt darf diesen Wert annehmen)
- zu jedem Feldtyp ein Feld erstellen:
  ein Typ mit Benötigt (Bsp: Name)
  ein Typ mit Einzigartig (Bsp: Nummer)
- auch kurz zeigen, die Feldtypen bearbeitet werden können

## Item

- Items zum Feldtypen erstellen
- auch Fehlermeldungen zeigen (Bsp: wenn zwei Laptops mit derselben Nummer erstellt wurden)
- Aufzeigen globales Feld
- Sortierfunktion zeigen
- Filterfunktion zeigen

- Suchfeld demonstrieren
- auch kurz zeigen, wie Items geändert werden können

## Firmen

- neue Firma erstellen
- dort sollen auch Benutzer und Rollen erstellt werden
- Firmenauswahl oben rechts zeigen

## Rollen

- Rolle erstellen
- vorher Firma oben rechts auswählen
- Berechtigungen einstellen

## Benutzer

- Benutzer anlegen
- Benutzer eine Rolle zuteilen
- Passwort zurücksetzen/ändern zeigen
- sich als neuer Benutzer einloggen und zeigen, dass dieser wirklich nur begrenzt Rechte durch Rollenzuweisung hat
- Daten bearbeiten (Stift unten links)

# Technische Umsetzung UI

UI mit Angular geschriebene Aufteilung in Module:

- Types (besitzt Store-Modul):
  Beinhaltet UI für Typenliste, Typendetails und globale Felder
- Items (besitzt Store-Modul):
  Beinhaltet UI für Itemliste, Itemdetails, Spaltenauswahl
- Roles (besitzt Store-Modul):
  Beinhaltet UI für Rollenliste, Details, Anlegen
- Company (besitzt Store-Modul):
  Beinhaltet UI für Unternehmen anlegen, Details
- User (besitzt Store-Modul):
  Beinhaltet UI für Benutzer anlegen, Details
- Permission:
  Das Permisson-Modul beinhaltet Helfer rund um die UI, damit Benutzer wirklich nur das sieht, was er darf
- Shell:
  Beinhaltet UI für die Navigation (links), Login-Fenster und Authorisierung
- Shared:
  Beinhaltet Inhalte, die von mehreren Modulen genutzt werden:
  Confirm Dialog: PopUp-Warnung bei DB-schädlicher Aktion (Bsp: löschen)
  Type Selector: Auswahl-Interface für ein Typ und Feld
  Default Page: Core-UI, auf der alles aufbaut (inkl. Elemente, die jede Seite besitzt, wie die Suchleiste) Store Factory: Alex, was heißt 'gleich' bei dir? xD

# Datenhandling in der UI

## Erklärung Store

Jede kontext-getrennte Ressource (Items/Types/Company/Roles/User) hat einen eigenen sog. Store.
Dieser übernimmt die Backend Kommunikation für das Laden, Editieren und Löschen der Daten sowie das Anzeigen eines Fehlers bei fehlerhaften backend Kommunikation.
(In einem Store gespeicherte Entitäten brauchen mindestens eine ID)

## unsere Stores

Ein Store bietet dem "Benutzer" 6 Methoden:

- `load()`
- `byId(id)`
- `create(enitity)`
- `update(enitity)`
- `delete(id)`
- `store`

Enitäten in einem Store werden so lange nicht neu geladen bis entweder ein forced reload stattfindet (wechseln der Firma) oder die TTL (time to live) abläuft.
Stores funktionieren über rxjs, was bedeutet ein Benutzer des Stores "abboniert" (subscribed) sich auf die gewünschten Daten und bekommt somit Änderungen im Store mit.
Stores werden über das StoreFactory Modul erzeugt. Dies übernimmt das Einstellen von Standart Konfigurationen und das bereitstellen von anderen Services an den Store (bsp. TranslateService für Übersetzungen)

## Lessons learned

- Einstieg in Angular ziemlich schwer
- TypeScript nicht so fehlertollerant wie JavaScript
- Besser recherchieren vor und während des Projekts
- Reviewgespräche besser vorbereiten
- Zeit besser einteilen
-
-
-
-
-