kompetenzinventar

Release 0.0.2

Gulliver <gulliver@wtf-eg.de>, Scammo, Weeman

20.06.2021

Contents:

KAPITEL 1

Architecture

1.1 Systembeschreibung

Die Kompentenzinventar-Software dient zur:

- Eintragung und Speicherung von Nutzerprofilen
- Suche nach Nutzerprofilen mit geeigneten Erfahrungen

1.1.1 Kompetenzinventar - Anforderungen

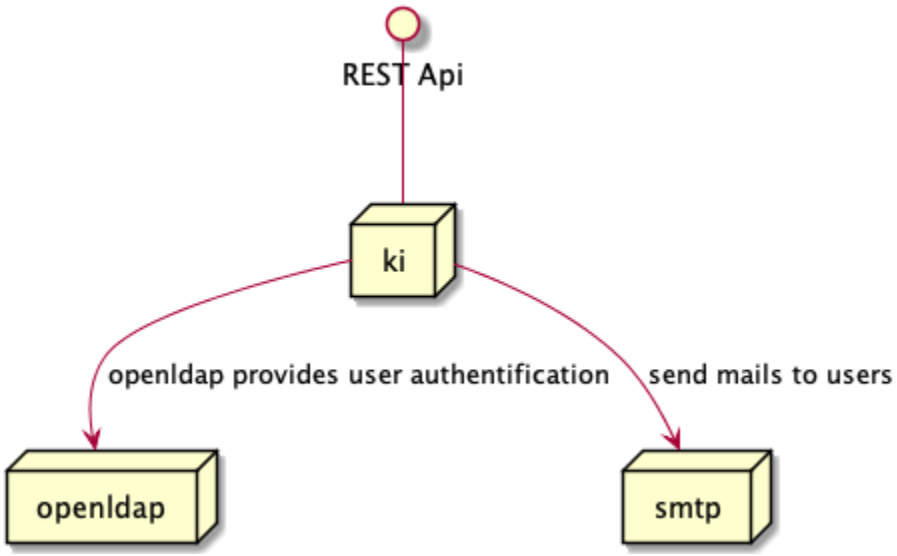
- Kompetenz Verwaltung aller WTF Member (aktuell ca. 150) Tendenz stark wachsen
- Einfache Verwaltung eigener Kompetenzen, Business Wünsche
- Einfache Suche von Fähigkeiten
- Netzwerk
- Nur für Interne Verwendung
- xing aber intern
- Auf der Seite müssen SKills, Erfahrungen und Projektwünsche skizziert werden können
- Z.b. Scammo, Flensburg, Kann: Javascript, Vue, Projekte, PHP, Events Sucht: Projekte als Freelancer 8-40H; Spricht Deutsch und English;
- Muss einfach für die Member erreichbar sein; Primär die Pflege der Daten
- Sollte auch Member abholen, die nicht so viel Zeit bei der WTF verbringen

1.1.1.1 Technisch

- Muss WTF gehostet sein
- Bus Faktor aka Es darf nicht alles von einen Menschen abhängig machen
- Möglichst Resourcen Schonend in Entwicklung und Betrieb

1.2 Systemkontext

1.2.1 Systemkontext Backend

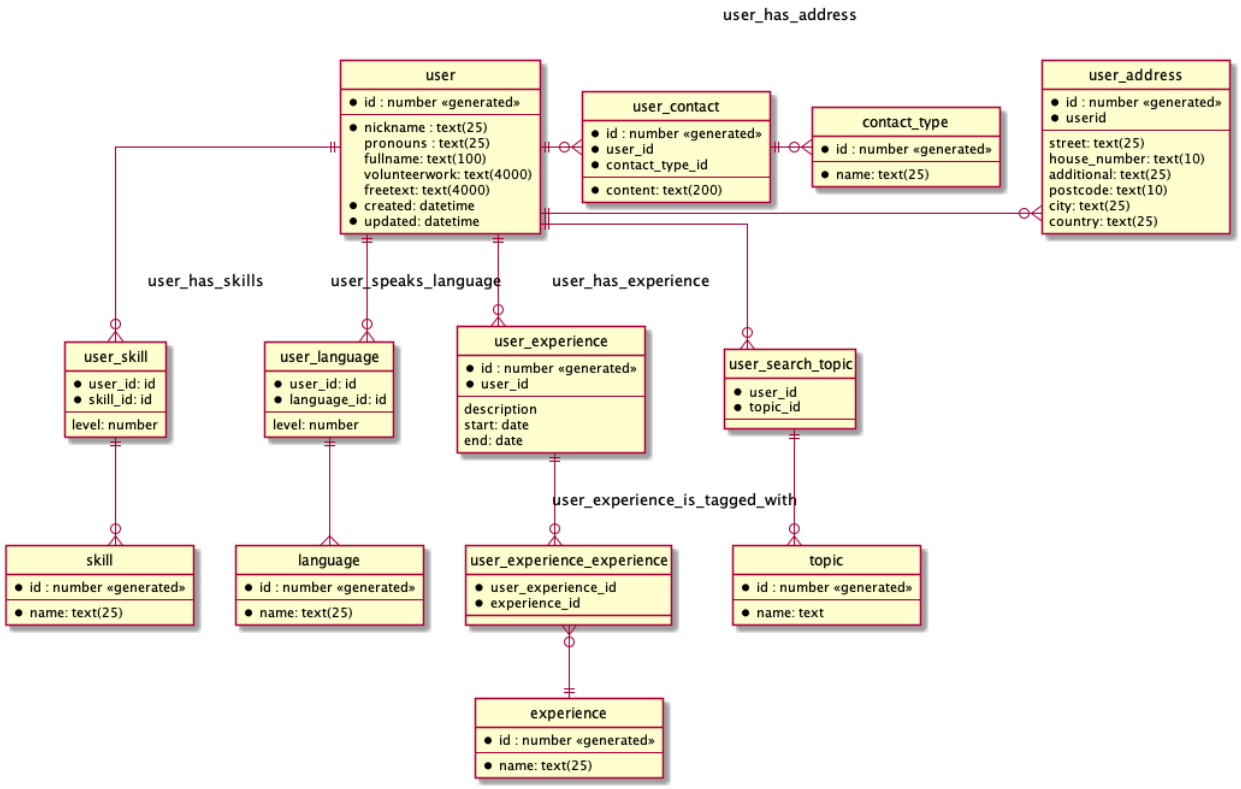


1.3 Use cases

1.3.1 Nutzerprofilverwaltung

1.3.2 Suche nach Mitgliedern

1.4 Datenmodell



1.5 Überblick Rest API

1.5.1 Authentifizierung

HTTP Bearer Token. Wird vom Backend verwaltet. Es gibt eine konfigurierbare, maximale Gültigkeit für einen Token.

Allgemeine Antworten:

- 401: Anfrage ohne Token
- 403: Token abgelaufen oder ungültig

1.5.2 Authorisierung

- Zugriff nur mit Login
- Jeder eingloggte Benutzer darf alles sehen
- Ein Benutzer darf nur sein eigenes Profil bearbeiten

1.5.3 Endpunkte

1.5.3.1 POST */users/login*

Hiermit kann sich ein Benutzer anmelden. Der Endpunkt authentifiziert gegen LDAP.

Anfrage:

```
{
   "username": "peter",
    "password": "asdasd"
}
```

Antwort: **Status 200**

Login hat geklappt

```
{
   "token": "aSDASDADASDASD"
}
```

Status 403

Login hat nicht geklappt. Grund ist Benutzer existiert nicht oder Passwort war falsch. Kein weiterer Text über die API wegen Sicherheit.

1.5.3.2 DELETE */users/login*

Markiert den zur Request Authentifizierung verwendeten Token als nicht mehr gültig.

1.5.3.3 POST */users/profile*

Speichert ein Profil ab. Übertragene Daten entsprechen #3 im JSON Format.

Anfrage:

```
{
   "profile": {}
}
```

Antwort:

Status 200

Gespeichert, ok

Status 400

Validierung fehlgeschlagen

```
{
   "messages": {
      "nickname": "Bitte ausfüllen"
   }
}
```

1.5.3.4 GET */users/{id}/profile*

Endpunkt um ein Profil gezielt nach ID abzurufen.

Antwort:

Status 200

Profil gefunden

```
{
  "profile": {}
}
```

1.5.3.5 GET */users/profiles*

Suche nach Profilen mit Query-Parametern:

- *skill[]=PHP*
- *page=1* für die Paginierung
- *page_size=20* Einträge pro Seite

(Liste wird erweitert).

Antwort:

```
{
  "total": 23,
  "profiles": []
}
```

- *total* Gesamtanzahl der passenden Profile
- *profils* Liste mit Profilen

1.5.3.6 GET */skills*

Hier können die auswählbaren Fähigkeiten inkl. Autovervollständigung abgerufen werden sowie „Ich Suche“.

- *search=an*
- Es werden 10 Einträge zurückgegeben

Antwort:

```
{
  "skills": [
     {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
            "id": 23,
            "name": "Angular"
        },
        {
            "id": 42,
            "name": "Anforderungs-Management"
        }
 ]
}
```


1.5.3.7 GET */skills/{id}/icon*

Icon einer Fähigkeit (weils gut ausschaut).

1.5.3.8 GET */languages*

Abruf der Sprachen.

- *search=fra*
- Es werden 10 Einträge zurückgegeben

Antwort:


```
{
  "languages": [
     {
         "id": 23,
         "name": "Französisch"
     }
  ]
}
```


1.5.3.9 GET */languages/{id}/icon*

Icon einer Sprache (weils gut ausschaut).

KAPITEL 2

Journal

2.1 2021-05-31 AG Dev - Kompentenzinventar

backend repository aufgesetzt https://git.wtf-eg.de/kompetenzinventar/ki-backend.git

Fürs Backend soll flask eingesetzt werden.

Für Datenmodell siehe #3.

2.1.1 Authentifizierung

HTTP Bearer Token. Wird vom Backend verwaltet. Es gibt eine konfigurierbare, maximale Gültigkeit für einen Token.

Allgemeine Antworten:

- 401: Anfrage ohne Token
- 403: Token abgelaufen oder ungültig

2.1.2 Authorisierung

- Zugriff nur mit Login
- Jeder eingloggte Benutzer darf alles sehen
- Ein Benutzer darf nur sein eigenes Profil bearbeiten

2.1.3 Endpunkte

2.1.3.1 POST */users/login*

Hiermit kann sich ein Benutzer anmelden. Der Endpunkt authentifiziert gegen LDAP.

Anfrage:

```
{
   "username": "peter",
    "password": "asdasd"
}
```

Antwort: **Status 200**

Login hat geklappt

```
{
   "token": "aSDASDADASDASD"
}
```

Status 403

Login hat nicht geklappt. Grund ist Benutzer existiert nicht oder Passwort war falsch. Kein weiterer Text über die API wegen Sicherheit.

2.1.3.2 DELETE */users/login*

Markiert den zur Request Authentifizierung verwendeten Token als nicht mehr gültig.

2.1.3.3 POST */users/profile*

Speichert ein Profil ab. Übertragene Daten entsprechen #3 im JSON Format.

Anfrage:

```
{
   "profile": {}
}
```

Antwort:

Status 200

Gespeichert, ok

Status 400

Validierung fehlgeschlagen

```
{
   "messages": {
      "nickname": "Bitte ausfüllen"
   }
}
```

2.1.3.4 GET */users/{id}/profile*

Endpunkt um ein Profil gezielt nach ID abzurufen.

Antwort:

Status 200

Profil gefunden

```
{
  "profile": {}
}
```

2.1.3.5 GET */users/profiles*

Suche nach Profilen mit Query-Parametern:

- *skill[]=PHP*
- *page=1* für die Paginierung
- *page_size=20* Einträge pro Seite

(Liste wird erweitert).

Antwort:

```
{
  "total": 23,
  "profiles": []
}
```

- *total* Gesamtanzahl der passenden Profile
- *profils* Liste mit Profilen

2.1.3.6 GET */skills*

Hier können die auswählbaren Fähigkeiten inkl. Autovervollständigung abgerufen werden sowie „Ich Suche“.

- *search=an*
- Es werden 10 Einträge zurückgegeben

Antwort:

```
{
  "skills": [
     {
         "id": 23,
         "name": "Angular"
     },
     {
         "id": 42,
         "name": "Anforderungs-Management"
     }
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
 ]
}
```

2.1.3.7 GET */skills/{id}/icon*

Icon einer Fähigkeit (weils gut ausschaut).

2.1.3.8 GET */languages*

Abruf der Sprachen.

- *search=fra*
- Es werden 10 Einträge zurückgegeben

Antwort:

```
{
  "languages": [
     {
         "id": 23,
         "name": "Französisch"
     }
  ]
}
```

2.1.3.9 GET */languages/{id}/icon*

Icon einer Sprache (weils gut ausschaut).

2.2 2021-05-28 KI Onboarding Planung

scammos wireframes vom Frontend

`pdf`

weeman brachte den Link ein: https://it.notchdelta.com/fragen#/was-suchst-du

2.2.1 zu erfassende Daten

- Nickname
- Pronomen
- Wohnort
- Ehrenämter (Freitext)
- Email für Gravatar Bild
- Freitext
- Kenntnisse inkl. 1 - 3 Sternen, z.B. PHP ***

- Qualifikationen, z.B. Studium, Ausbildung oder Zertifikate
- “Ich suche”, z.B. “Software Entwicklung”, “Projekt-Management”, Vollzeit, Teilzeit, stundenweise
- Einsatzort (vor Ort und oder Remote)
- Sprache
- Auflistung bisheriger Projekte
- Titel
- Link
- Freitext
- Kontaktmöglichleiten
  - E-Mail
  - Matrix
  - GSM
  - Freitext

2.2.2 Techstack

- Frontend vue.js
- Python für Restfull/json $framework
- Datenbank: Production: Postgres; Development: SQLlite
- separate Repositories für Front- und Backend
- ki-doku repository ist für Dokumentation und Verwaltung der Issues

KAPITEL 3

Indices and tables

- genindex
- modindex
- search