/api/v1/

● = Logged In

Bei 401 = muss einloggen

→ /users
- → /me GET User
- → auth
  - Post User → JwtToken ← Bei success → JWT in samesite strict HttpOnly Cookie

→ courses
- GET → Array<Course>
  - ?start=<Date>&end=<Date> → Array<Course>
- → /:id
  - isLecturer true → Post Course
  - GET → Course
  - → /dates
    - UID ist lecturer → PUT Course
    - Delete → Course
  - → /categories
    - GET → Array<String>
  - GET → Array<CourseDate>
  - → /:id
    - GET → CourseDate
    - → /bookings
      - PUT → CourseDate
      - Delete
      - UID ist lecturer → Post Course Dele
      - GET → Array<Booking>
      - Post → Booking
      - → /:id Get Post Put Dd

Nur wenn lecturer/Eigentümer

Objekte

Course

```
{ "title"       : string,
  "description" : string,
  "lecturer"    : number, ← userid
  "price"       : number,
  "dates"       : Array<SchulungDate>,
  "id"          : number
  "category"    : string ←
  "organiser"   : string
```

Konferenz
Sprachkurs
Meeting
...

}

CourseDate

```
{
  "date"      : ISO8601/string,
  "available" : number,
  "total"     : number,
  "place"     : string,
  "id"        : number
}
```

Booking

```
{ "id"     : number,
  "user"   : number, ← userid
  "places" : number
}
```

User

```
{ "id"       : number,
  "username" : string,
  "isLecturer" : boolean
}
```

Für verändernde Aufrufe und Daten, die bestimmte Nutzerinformationen enthalten Login, alle anderen frei verfügbar.

Auth entweder über same site strict cookie oder bearer auth header (Webapp braucht dank cookie keine extra Logik, direkte REST Calls können api aber mitbenutzen)

Schulungen können mit Queryparametern nach Datum gefiltert werden.

Schulungsobjekte geben auch eine Liste an Terminen und Status der Termine mit (gebucht/frei). Damit diese direkt im Frontend ohne weiteren call angezeigt werden können.

Anhand einer Schulungen können die Einzelinformationen abgefragt und geändert werden.

Lecturer sehen im Buchungsendpunkt nicht ihre eigenen, sondern alle Buchungen ihrer Schulung. Buchen können Sie aber normal.