Project concept suggestions

1. Architecture - 2 servers:
    1. API server to serve **JSON** data on **api.mysite.com**
    2. View server to serve the **HTML** on **mysite.com**
2. Session
    1. backend API tracks user session via session cookie
    2. session cookie should be available in all subdomains, therefore the cookie domain should be: **.mysite.com**
3. API
    1. Should return JSON data;
    2. Should set proper status codes for the response (404 - not found, 401- unauthorized..)
    3. Client preferences - available in request headers as follows:
        1. Language - `X-Language: en`, fallback on header `Accept-Language: da, en-gb;q=0.8, en;q=0.7;` We can try to merge them 🤔
        2. Currency - `X-Currency: BGN`
    4. REST URLs - common URLs for each resource in the DB. Here is an example for **users** (`=>` means response) :

```
GET     /users?q=search&page=1&limit=10&sort=-name,age  => User list + Pagination
POST    /users      # add new user => The User
GET     /users/:id  # get user data => The User
POST    /users/:id  # update user data => The updated user
DELETE  /users/:id  # delete user => null

ADDITIONAL OPTIONAL REQUESTS

PUT     /users/:id  # update user data => The updated user
PATCH   /users/:id  # update user data fields => The updated user
DELETE  /users?id[] # Delete multiple users => null
POST    /users  # Create/update a user if payload has existing id => The User
```

    5. Response standart
        - Common models scheme - these fields are required for each resource scheme:

```
{
    id: Number,
    name: String
}

//example
user = { id: 1, name: 'Luc' }
article = { id: 1, name: 'The old republic' }
```

        - No repetitive prefixes of the fields:

```
// invalid
user = {
    user_id: Number,
    user_name: String,
}

// valid
user = {
    id: Number,
    name: String
}
```

- Lists should be `0`-based Arrays

```
// invalid
users = {
    2: User,
    10: User
}

// valid
users = [ User, User ]

// use array_values()
```

- Discuss pagination response!