Authenticatie voor de back-end

In de front-end is het authenticeren van een gebruiker zo goed als compleet. In de back-end is dat niet het geval. Dit document zou een beginnende opzet moeten geven van hoe authenticatie in de back-end aangepakt moet worden. Mocht de front-end en/of back-end authenticatie onduidelijk zijn, is er altijd de tutotials/GIT-repositories om te raadplegen.

**Front-end tutorials die gevolgd is:**

<https://www.youtube.com/playlist?list=PL0Zuz27SZ-6PRCpm9clX0WiBEMB70FWwd>

**GIT-repository front-end (react):**

<https://github.com/gitdagray/react_persist_login>

**GIT-repository back-end (Node.js):**

<https://github.com/gitdagray/mongo_async_crud/tree/main/controllers>

**Back-end tutorial (Node.js):**

<https://www.youtube.com/watch?v=f2EqECiTBL8&ab_channel=DaveGray>

**Voor database tabellen:**

Voor de database is het belangrijk dat er in de user tabel een kolom is waar de users refresh token in bewaard wordt, let op deze moet ook leeg kunnen zijn.

Bv. user table:  
id, username, email, password (hashed), refreshtoken

Handig zou zijn om te werken met een role tabel, waar de rollen/namen zoals USER, ADMIN in staan. Hierbij kan er ook gemakkelijk een nieuwe rol worden toegevoegd, zoals SUPER\_ADMIN of MAINTAINER etc. Uiteindelijk zou de gebruikers rol naar de front-end gestuurd moeten worden en hierdoor kan de gebruiker op de front-end toegang krijgen tot bepaalde pagina’s.

Bv. role table:

id, name

Bij het gebruiken van een role tabel, moet er tussen user en role een relatie zitten.

Bv. user\_role table:

user\_id, role\_id

**Toelichting op bepaalde rest-calls:**

**POST  
api/Authentication/register**

* Save de user (name, email, password(hashed)) **+ de role / admin (wordt niet meegegeven bij registreren)** in de database
* Return een OK of response error status.

**POST**  
**api/ Authentication /login**  
In de back-end moeten een paar dingen gecheckt worden voordat iets teruggestuurd kan worden naar de front-end:

* check of de aangekomen body niet leeg is, anders return response error status en bv message: “email en wachtwoord zijn verplicht”
* Check of de user bestaat (valideer email en wachtwoord), anders return response error status en bv message: “gebruiker bestaat niet” || “Verkeerde gegevens ingevuld”
* Maak access token en refresh token. Refresh token moet wel de username, expiration date (bv. 1 dag) bevatten. Access token moet username, role en expiration date (bv. 1 minuut) bevatten.
* Save de aangemaakte refresh token in de database.
* De response moet een JSON access token en HTTP Only cookie refresh token teruggeven naar de front end.

**GET  
api/ Authentication /refresh**  
In de back moeten een paar dingen gecheckt worden voordat iets teruggestuurd kan worden naar de front-end:

* Er moet een cookie aankomen
* (optioneel) check of er iets in de cookie zit, anders return error status
* Vind een volledige user by refreshtoken.
  + Geen user gevonden? return response error status.
  + User gevonden? Check of de username die in de refreshtoken staat (refresh token moet ge-decode worden) == username in database.
    - Maak een nieuwe access token met de username en role en return die via JSON.

**GET  
api/ Authentication /logout**  
In de back moeten een paar dingen gedaan worden voor uitloggen:

* Er moet een cookie aankomen
* (optioneel) check of er iets in de cookie zit, anders return error status
* Check of de cookie (refresh token) in de database bestaat.
  + Nee? clear cookie en return response error status.
  + Ja? empty de refresh token string die bij de user hoort en save dat in de database. Clear de cookie en stuur OK response status.