**Documentatie Front End - Calendar**

**Proiect colectiv**

Pentru partea de client a proiectului, am decis sa realizam o aplicatie web. Ca si web framework, am ales sa ne utilizam de **React**.

Inainte de toate, ca si limbaj de programare, in loc de javascript, am ales sa folosim **typescript**, deoarece pe masura ce proiectul se complica, *static typing-ul* ne-a ajutat foarte mult sa dezolvam aplicatia in continuare.

Librarii folosite:

* **Material UI**, deoarece compoentele in stilul material arata chiar foarte bine, am ales sa scutim timp in a ne stiliza prea mult aplicatia facand toate componentele (butoane, labels, input-uri) de la 0.
* **Axios,** este libraria pe care am folosit-o pentru a face request-uri la Back end, ne-a ajutat mai mult decat functiile cu care venea javascript default, astfel am putut avea o structura mai buna a codului.
* **React-big-calendar**, este componenta principala de calendar, pe care ne afisam toate evenimentele, pe alocuri, am suprascris stilul unor bucati din calendar, pentru a-l face sa arate pe placul nostru.
* **React query**, libraria pe care, alaturi de Axios, o folosim sa facem query-uri pe back-end, si la fiecare modificare a calendarului, acesta sa ramana mereu up-to-date.

Proiectul este structurat astfel:

* **Api**: aici avem toate componentele care ne ajuta in legatura cu back end-ul, avem partea de
  + **mutations** (avem functii de create/update/delete pe event si tag-uri, le folosim cu react-query, deoarece dupa fiecare mutatie, se mai face un fetch pe server pentru a vedea real-time modificarile).
  + **queries** aici avem toate functiile pentru **GET** request.
  + **services** aici avem clasele care apeleaza functiile de mai sus, si pe care le folosim in componentele in care si avem ui-ul.
* **Components**, in acest packet tinem toate componentele de react create de noi, pe care le afisam in pagini
* **Pages**, toate paginile folosite in aplicatie.
* **Models**, clasele modelelor folosite in aplicatie.
* **Helpers,** clase cu functii de care avem nevoie in aplicatie, de exemplu, aici tinem fisierul care ne ajuta cu afisarea culorilor pentru tag-urile de pe event-uri.

De asemenea, ne folosim de **JWT Token** pentru a tine sesiunea user-ului, la login (sau register), primim de la server un jwt aferent user-ului (daca a fost logat cu succes), pe care il tinem salvat in **localstorage**, si il trimitem mai departe ca header pentru fiecare request (ce nu presupune autentificarea).

Sunt prezente urmatoarele cazuri de utilizare:

* Creare cont:
  + User-ul introduce username si parola pentru a crea un cont nou
* Autentificare:
  + User-ul isi introduce username-ul si parola pentru a accesa calendarul lor
* Gestionare event-uri:
  + User-ul vede pe pagina de calendar toate evenimentele sale
  + User-ul poate schimba modul de vizionare al evenimentelor, de la luna, la saptaman, si zi
  + User-ul poate naviga de la o luna/saptaman/zi la alta prin intermediul unor butoane de previous/next, sau poate sari direct apeland la butonul de jump
  + User-ul poate creea sau accesa un eveniment deja existent, fiindu-I deschis un form in care poate introduce/modifica titlul, data de start, data de finish, descrierea, tag-urile si poate selecta daca evenimentul este de tip All Day sau nu
  + Din acelasi form, user-ul poate sterge event-ul respectiv apasand pe un buton de delete
* Gestionare tag-uri:
  + User-ul poate accesa o lista master detail cu tag-urile sale
  + User-ul poate creea sau accesa un tag deja existent,fiindu-I deschis un form in care poate introduce/modifica numele si culoarea tag-ului
  + Din acelasi form, user-ul poate sterge tag-ul respectiv apasand pe un buton de delete
* Filtrari evenimente:
  + Pe pagina principala exista un view de tip agenda ce permite filtrarea evenimentelor dupa o perioada calendaristica, intre o data de start si o data de finish
  + De asemeni, user-ul poate selecta o grupare de tag-uri, pe calendar fiind ulterior afisate doar evenimentele ce contin cel putin gruparea respectiva