Beadandó feladat

Haladó web alapú programfejlesztés

Beadandó feladat

A következő feladatpontokhoz készítsen Angular alapú alkalmazást felhasználói felületekkel. A megvalósítás során alkalmazza a kurzuson bemutatott elveket, technikákat, eszközöket. A szerver műveleteket valamilyen mock/fake megoldással legyenek megvalósítva (InMemoryWebApi, Http interceptor-okból adott válaszok, egyéb létező mock/fake megoldás). Az értékelés során figyelembe lesz véve a modulok/komponensek, tervezési minták, szoftverfejlesztési alapelvek megfelelő használata, illetve az RxJS Store használata az adatok/állapot menedzselésére. A feladatokban nem rögzített részek tetszőleges módon megoldhatóak. A felület design nem képezi részét a feladatnak, lehet alap HTML-t használni vagy bármilyen elérhető UI csomagot (Bootstrap, Material, Tailwind, Ionic, stb.).

1. Készítsen alkalmazást, amely képes kezelni, hogy melyik oktató melyik félévben melyik tantárgyat oktatta, és melyik tantárgyat melyik hallgatók vették fel. Az **oktatóknak** van Neptun kódja, neve, email címe, beosztása. A **beosztásoknak** fix értékkészlete van: docens, adjunktus, mesteroktató, ügyvivő szakértő, tanársegéd, egyéb. A **tantárgyaknak** van neve, kódja, kreditértéke (egész szám), felelős tanszéke (pl: VIRT, RSZT, Matematika, stb.). A **féléveknek** van neve (pl: 2022/23/1), kezdő dátuma és végdátuma. A **hallgatóknak** van Neptun kódja, neve, email címe és szakja. A **szakoknak** fix értékkészlete van: Mérnökinformatikus Msc, Programtervező informatikus Msc, Mérnökinformatikus Bsc, Programtervező informatikus Bsc, Gazdaságinformatikus Bsc. Egy tantárgyat oktathat több oktató egy félévben; egy félévben lehet több tantárgy is; egy hallgató felvehet több tantárgyat is egy félévben; egy tantárgyon több hallgató is lehet.
2. Az alkalmazásban lehessen listázni az összes tantárgyat, oktatót, hallgatót, félévet külön-külön komponensekben.
3. Lehessen lekérni egy oktató egy félévben oktatott összes tantárgyát.
4. Lehessen lekérni egy hallgató egy félévben felvett összes tantárgyát.
5. Lehessen új oktatót, tantárgyat, félévet és hallgatót felvinni a rendszerbe, ezeket összerendelni és a meglévőket lehessen módosítani.
6. Az oktató, tantárgy, félév, hallgató entitásokat törölni nem kell, de a köztük lévő összerendeléseket lehessen törölni.
7. Minden űrlapon, ahol felvinni vagy módosítani lehet a rekordokat legyenek a mezőkhöz definiálva megfelelő validációs szabályok (az egyes paraméterek nincsenek meghatározva, értelmes értékek legyenek választva pl. bevihető karakterek maximális számához, stb.).
8. A Neptun kódhoz legyen egyedi validátor: a Neptun kód pontosan 6 karakterből áll, csak számokat vagy betűket tartalmazhat, és nem kezdődhet számmal.
9. A táblázatos felületeken lehessen szűrni a rekordokat. A szűrés a szűrő mezőben történő gépelés hatására történjen, két szűrés között min. 300ms legyen és csak akkor ha módosult a szűrő kifejezés.
10. A táblázatos felületeken lehessen rendezni a rekordokat bármelyik mező szerint.
11. Egészítse ki az alkalmazást, hogy csak autentikált felhasználók tudják elérni a funkciókat. A felhasználónak legyen születési dátuma, neve, Neptun kódja és oktató esetén tanszéke (ez a

mező nem oktató esetén „ismeretlen” értékű legyen). Alakítsa át az entitásokat úgy, hogy minden oktató és hallgató felhasználó legyen. A felhasználóknak egy vagy több hozzárendelt szerepköre lehet. A szerepkörök legyenek Admin, Teacher és Student.

12. A felhasználók egy bejelentkező felületen tudjanak bejelentkezni. A bejelentkezés után a felhasználó kapjon egy JWT tokent, ami tartalmazza az azonosítóját, a felhasználónevét, az email címét és a hozzárendelt szerepköröket. A szerver oldali funkciókat (belépés ellenőrzése, token előállítása, stb.) tetszőleges mock/fake megoldással lehet megvalósítani (pl. manuálisan előre generált JWT tokeneket lehet használni: http://jwtbuilder.jamiekurtz.com/). A tokent tárolja az alkalmazás memóriában vagy valamilyen storage megoldással.
13. A teszteléshez készítsen legalább 3 felhasználót a rendszerben, akik közül az egyik Teacher a másik Admin a harmadik Student szerepkörrel rendelkezzen. A felhasználó adatokhoz nem kell se lekérdező se karbantartó komponenseket készíteni.
14. A komponenseket a bejelentkező oldalon kívül csak autentikált felhasználók érhetik el. Ennek ellenőrzését RouteGuard-ok segítségével oldjuk meg. Autentikált a felhasználó ha van tárolt JWT tokenje az alkalmazásban.
15. Készítsen egy HttpInterceptor-t, ami minden kéréssel elküldi a bejelentkezett felhasználó tokenjét az Authorization fejlécben.
16. Legyen kijelentkezés funkció, ami törli a tokent és visszairányítja a felhasználót a belépő felületre.

17. Az oktató, hallgató, tantárgy funkciók egy külön feature modulba legyen kiszervezve (lazy loading + navigáció).
18. Az alkalmazás adatait kezelje RxJs Store állapotmenedzsment megoldásokkal, elég egy globális állapotot használni.
19. Valósítson meg kliens oldali cache-elést a GET kérések esetén HttpInterceptor vagy RxJs Subject + operátorok segítségével. HTTP kérés csak akkor menjen a szerver felé ha a cache-ben nincs a kéréshez tárolt adat. A létrehozó, módosító, törlő funkciók után legyenek kiürítve a megfelelő cache-ek.
20. Sikeres bejelentkezés után dekódolja a JWT token tartalmát és a felhasználó adatait (email, név, azonosító, szerepkörök) tegye elérhetővé a többi szolgáltatás/komponens számára egy megfelelő RxJs Subject formájában, úgy hogy azok mindig a legfrissebb állapotot kapják meg.
21. Az adatfelvitel, módosítás, törlés, összerendelés funkciókat csak Admin szerepkörrel lehessen elérni.

2-es szint: az első 16 pont maradéktalan megvalósítása, vagy a kimaradó feladatpontok helyett a többi feladatpontból azokkal ekvivalens pontok megoldása.