Angular SimpleBlog v9.x

## **Admin dasboard**

Tegyük fel, hpgy a a SimpleBlog blog alkalmazást egy személy látja el tartalommal, a szerző, aki egyben az alkalmazás adminisztrátor jellegű használója.

A SimpleBlog jelen állapotában az adatok egy db.json fájlban csücsülnek, ha a szerző új cikket akar felvinni, akkor ezt a fájl kell szerkesztenie, ami nem tól felhasználó barát, főleg, ha az illető nem jártas informatikában.

Minden valamire való alkalmazásban, ahol tartalmat töltenek fel csak bizonyos jogkörű felhasználó(k) tehetik, meg egy erre a célra külön kialakított felületen az alkalmazásban vagy azon kívül (admin dashboard).

## A cél

A SimpleBlog alaklmazásban új “feature”-ként létrehozunk egy csak az admin felhasználó számára elérhető felületet, ahol a blogon megjelenő tartalamakat tudja szerkeszteni → ez lesz angolosan elnevezve az “admin dashboard”, röviden “dashboard”.

Egy látványterv a dashboard-ról: <https://projects.invisionapp.com/freehand/document/oXE4WlOTS>

## Menetrend

A dashboard-ot több iteráción keresztül kerül lefejlesztésre

* 9.1 Előskészöletek, alap elemek létrehozása, mint:
  + dashboard feature module
  + guardok
  + komponensek
* 9.2 AuthService

# 

# 

# 9.1 Admin dashboard alap

## 

## Feladatok

1. 1DashboardModule
   1. Hozz létre egy új feature modult routing-gal!
      1. A modult az src/features mappába helyzd!
   2. ezt a modult lazyloading technikával töltsd be a fő routing modulban a ‘admin’ útvonalon!
   3. A modul saját routing moduljában az üres “” ütvonalon (ami egyenlő a “admin” útvonallal) töltse be a DashboardComponent-et, ami egy üres komponens
   4. tesztelés: az alkalmazás a “/admin” útvonalon betölti a még üres DashboardComponent-et
2. AuthModule - Ez a modul kizárólag a felhasználó autentikációra szolgál, nem tartalmaz komponenseket, csak provider tipusó elemeket, mint service és guard  
   1. Hozd létre az AuthModule-t a src/app mappa alá!
   2. a modulban hozz létre egy AuthGuard-ot, ami majd a korlátozza az útvonal elérést, attól függöen, hogy a felhasználó be van jelentkezve vagy sem
      1. implementálja a **canLoad** és **canActivate** metódusokat, ami boolean típussal térnek vissza.   
         Jelenleg még nincs service, ami eldöntené, hogy a felhasználó valóban jogosult a belépésre. A fejlesztés ezen korai szakaszában még beérjük enélkül és mindig engedjük a belépést azzal, hogy a metódusaik egy boolean true-val térnek vissza!   
          **Ezt a részt késóbb tovább fejlesztjük, hogy csak akkor adjon vissza true-t a guard, ha a felhasnáló helyes email/jelszó párost adott meg!**
      2. **Használjuk és kössük be a AuthGuard-ot!**
         1. importáljuk az AuthModule-t az AppModulban
         2. a fő routing modulban a modul importnál használjuk a canLoad propertyvel az AuthGuard-ot
            1. tesztelés: AuthGuard canLoad metódus vissztérési értékét változtatva nézd meg (true/false), hogy a modul betöltődik-e
         3. Védjük le a Dashboard component útvonalát CanActivate guarddal (add meg a dashboard routing modul “” path-re)  
            Teszteljük a guard-ot a canActivate metódus visszatérési értéket változtatva!

9.2 Autentikáció JWT token

## Előkészítés

Allítsd be a json-server-auth csomagot a NG SimpleBlog projekten!

* users tömb a db.json-ben
* indító script módosítás

## Feladatok

1. AuthService - Hozz létre egy új service-t az AuthModule-ban
   * 1. jwtSubject - A service osztály egy RxJS BehaviorSubject observable-ben tárolja e bejelentkezés állapotát!
     2. login metódus bejelentkeztet egy felhasználót és visszad egy AccesToken interfászű objectet, amiben accesToken kulcs alatt szerepel a JWT token, ha sikeres a bejeletkezés a JWT token elmenti a localStorage-ben a” jwt” kulcs alatt
     3. isLoggedIn - egy metódus true/false-t ad visza annak apalján, hogy a felhasznál be van-e jelentkezve!
     4. logout metódus - kiléteti a felhasználót, a jwtSubject observable-t állapotát null-ra állítja (next metódusát null-ra állítja)
2. auth.interface.ts
   1. hozz létre interface-eket az alábbi adatokatnak
      1. login metódus visszatérési értékének
      2. LoginCredentials - emal/pass object
3. AuthGuard
   1. injektálja az AuthService-t DI segítségével
   2. a canLoad,canActivate metódusai most már a AuthService isLoggedIn metódusával döntik el, hogy a route navigálást engedélyézzék vagy sem
      1. ha isLoggedIn true → a guard true-t ad vissza → navigálás működik
      2. ha isLoggedIn false→ a guard false-t ad vissza → navigálás nem működik
4. Regisztrálj egy egy user-t POSTMAN.nel
5. Login komponens
   1. készítsd el a Login komponenst
      1. ha van shared module-od, akkor az egy jó modul erre, de lehet a AuthModule-ban
      2. A komponeseben használj hibakezelést!
      3. exportáld a login komponenst!
      4. készíts egy /login útvonalat a login komponesre a fő routing modulban
      5. ha a felhasználó a /dashboard-ra akar navigálni és nincs bejelentkezve, akkor navigálja át a /login útvonalra
      6. ha sikeres a bejelentkezés, akkor automatikusan vigye át a /dashboard útvonalra
   2. Regiter komponens opcionális
6. Teszteld az autentikációt a
   1. bejelentkezés nélkül prübálj a /dashboard oldalra navigálni (nem szabad sikerülnie)
   2. a regelt felhasználóval jelentkezz be, és próbálj a /dashboard path-re menni (működnie kellne!)