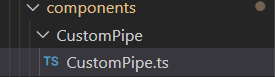
Fejlesztői dokumentáció

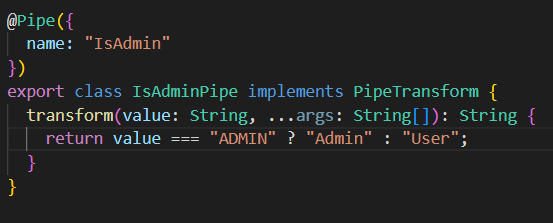
components 🡨 itt található az összes komponens ami az applikációhoz tartozik

components 🡪 CustomPipe 🡪 CustomPipe.ts

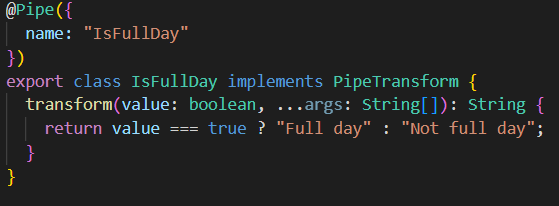


Az összes ellenörzést végző egyedi pipe funkciók itt találhatóak

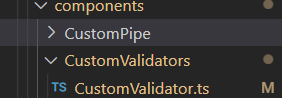
IsAdmin ellenőrzi hogy a felhasználó admin vagy user az adatbázisban tartalmazott szöveg alapján, ha „ADMIN” akkor „Admin” lesz kiiratva más esetben „User”



IsFullDay ellenőrzi boolean érték alapján A „Full Day” értékét ha true akkor „Full Day” lesz kiiratva ha false érték akkor meg „Not full day”

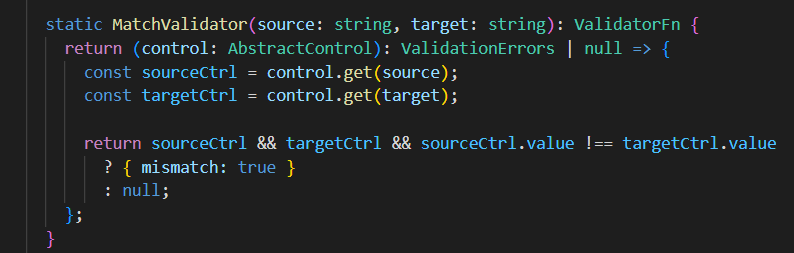


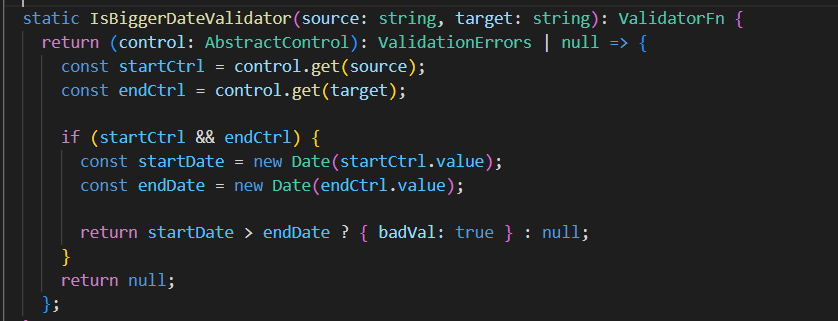
components 🡪 CustomPipe 🡪 CustomPipe.ts



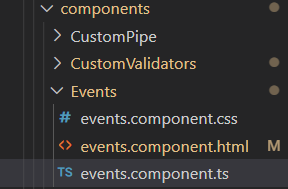
Itt találhatóak az egyedi validációk

MatchValidator ellenőriz két értéket ami ha nem egyezik akkor arra visszadob egy true boolean értéket ha nem egyezik akkor az érték null, ez a jelszó ellenőrzés érdekében lett létrehozva



IsBiggerDateValidátor szintén két értéket ellenőriz, ha a kezdő dátum nagyobb mint a végdátum akkor visszadob egy true értéket más esetben null.Ez a dátum ellenőrzés miatt lett létrehova

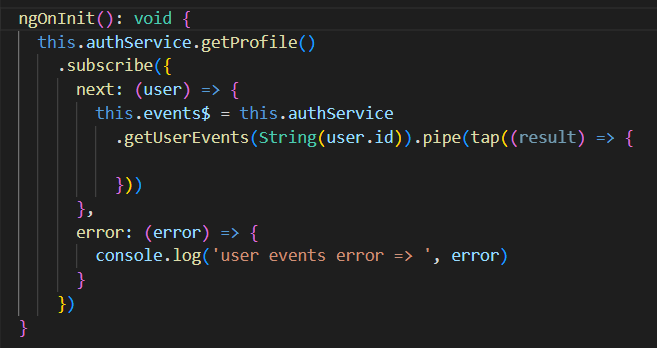
components 🡪 Events



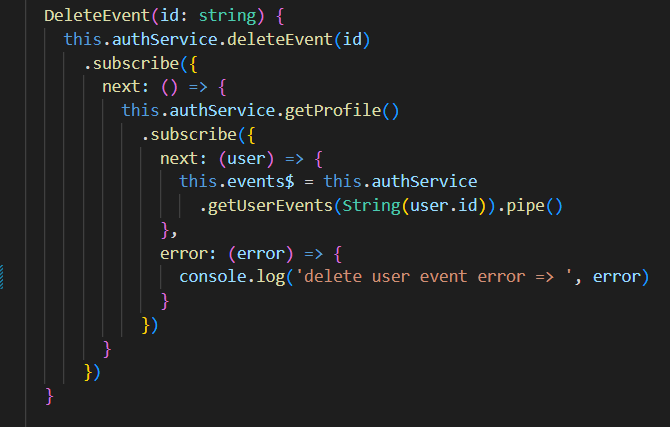
Events felület komponense, a bejelentkezett felhasználó evenetjeinek a kiiratásáról, törléséről gondoskodik

components 🡪 Events 🡪events.component.ts

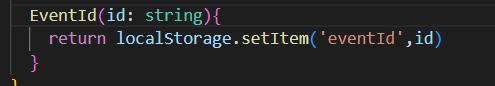
Itt az oldal betöltésekor Mindig lefut egy lekérdezés az end pontoktól ami először lekérdezi a profil majd annak az eventjeit



A DeleteEvent az event id-ja alapján törli a kiválasztott eventet

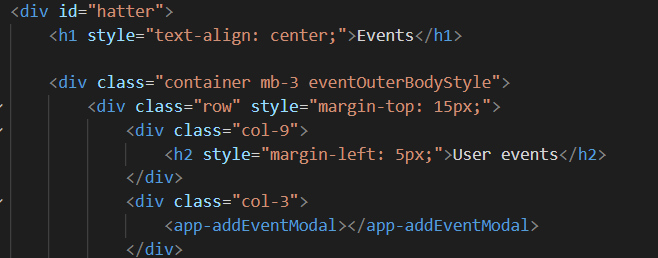


A EventId a kiválasztott event id-át fogja tárolni amit egy másik komponens nyit majd meg

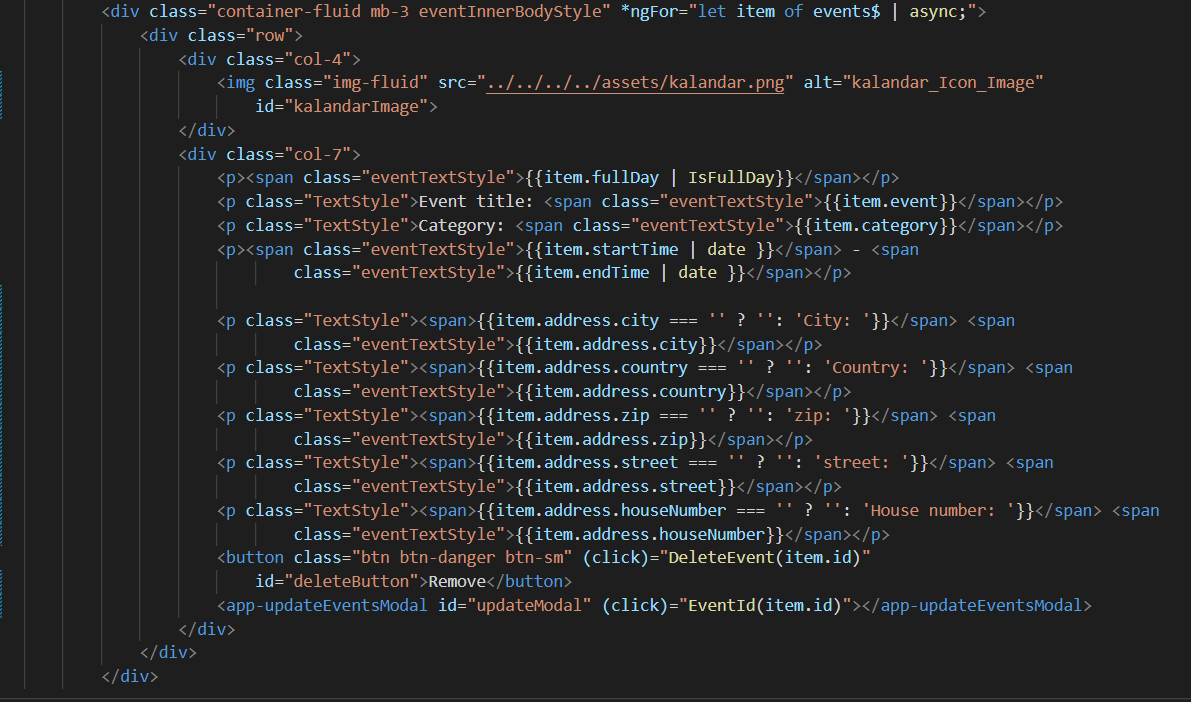


components 🡪 Events 🡪events.component.html

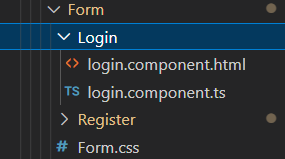
app-EventModal egy másik komponens ami az új eventek felvételeiről gondoskodik



Itt iratódik ki az összes felhasználói event, valamint az event módosítása is egy másik komponensből valósul meg amihez itt kapja meg a kiválasztott event id-ét is.



components 🡪 Form 🡪 Login

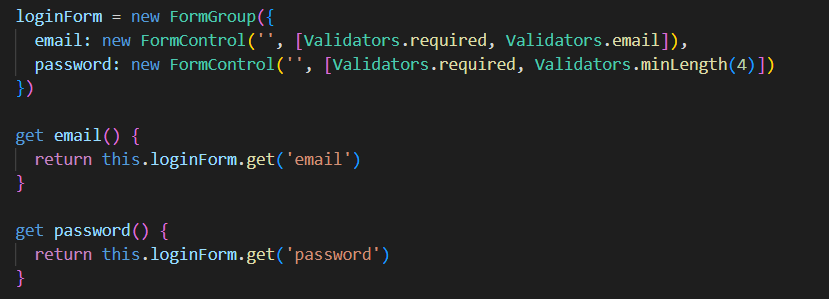


A Login felület komponense. Felhasználói értékeket ellenőrzi és a bejelentkezést valósítja meg.Amit egy közös Form.css file formáz meg.

components 🡪 Form 🡪 Login 🡪 login.component.ts

A loginForm kapja meg a Form értékeket, valamint validálja azokat.

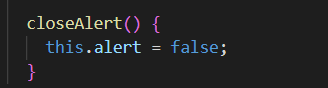
Ezek a a html file-ban megadott „email” , „password” változók által kapja meg



A SubmiLogin a bejelenkezéskor való próbálkozáskor fut le ami megnézi hogy létező személyt adtunk meg .A létező személy esetekor lefut az AuthService komponensből A login metódus amiből kap a felhasználó egyedi tokent, valamint a getProfile metódus amiből letárolódik a localStorageban kijelentkezésig a user id, Role, ezekután átnaviglódik a user a „calendar” oldalra.

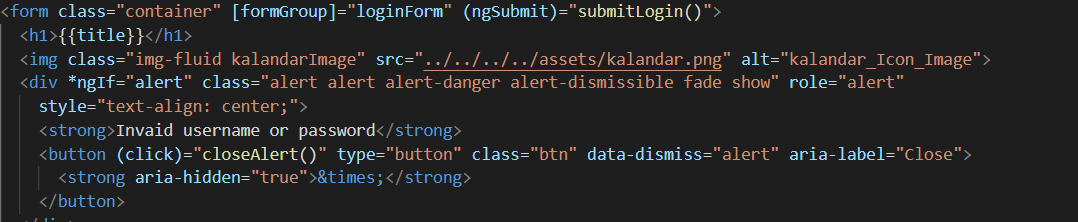


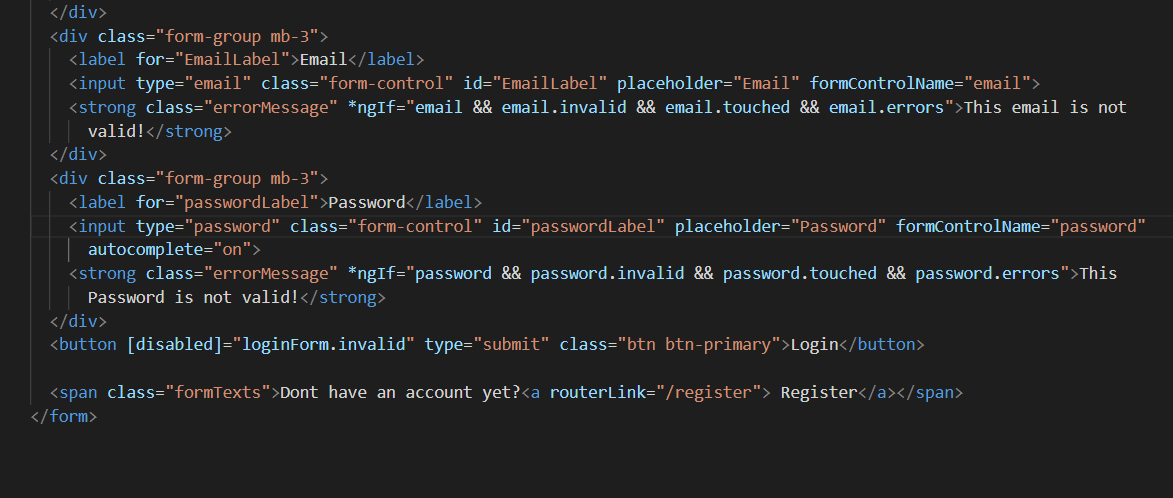
A closeAlert a felugró alert felület becsukásáról gondoskodik rossz felhasználói értékek esetén.



components 🡪 Form 🡪 Login 🡪 login.component.html

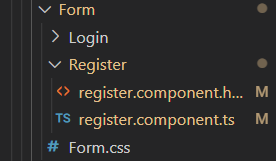
A submitLogin metódus és a LoginForm FormGroup itt hívódik meg.

A closeAlert metódus rossz értékek esetén hívódik meg.

A FormControlName alapján kapja meg a loginFrom a beírt értékeket a Form-ban.

[disabled]=”loginForm.invalid” addig nem teszi lehetővé a gombra kattintást amíg a form nem valid.

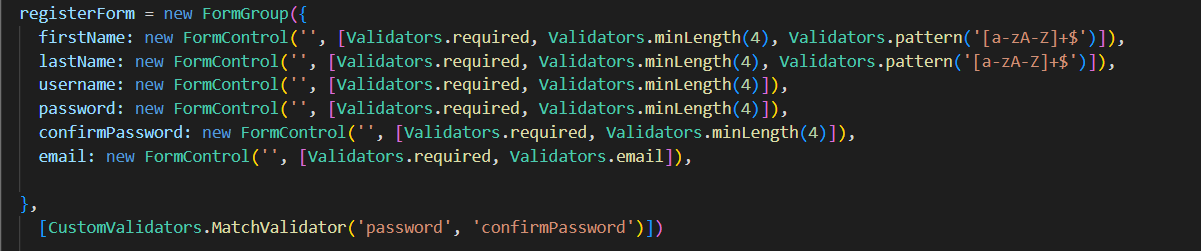
components 🡪 Form 🡪 Register

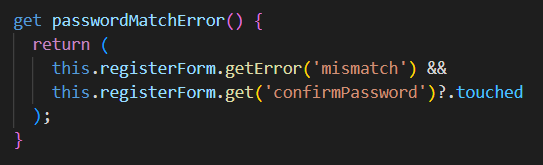


A Register felület komponense. Felhasználói értékeket ellenőrzi és a regisztrációt valósítja meg.Amit egy közös Form.css file formáz meg.

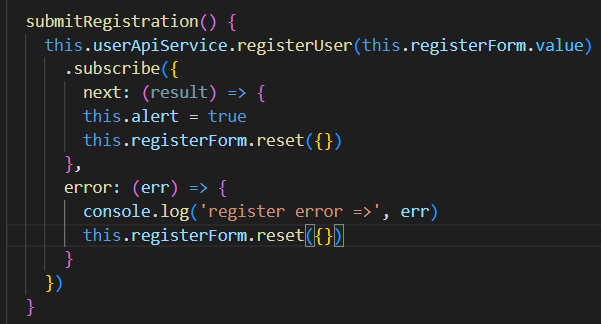
components 🡪 Form 🡪 Register 🡪 register.component.ts

Ugyan az alapján a logika alapján működik mint a login komponens

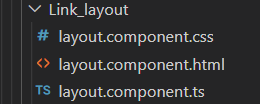
A CustomValidators komponens a MatchValidator metódusa itt hívódik meg, ami ellenőrzi a két jelszó megegyezését.



A submitRegistration a regisztrációkor fut le, ami addig nem elérhető a felhasználó számára amíg nem ad meg minden mezőt a validációnak megfelelően.



components 🡪 Link\_layout

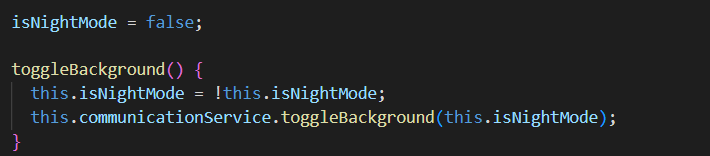


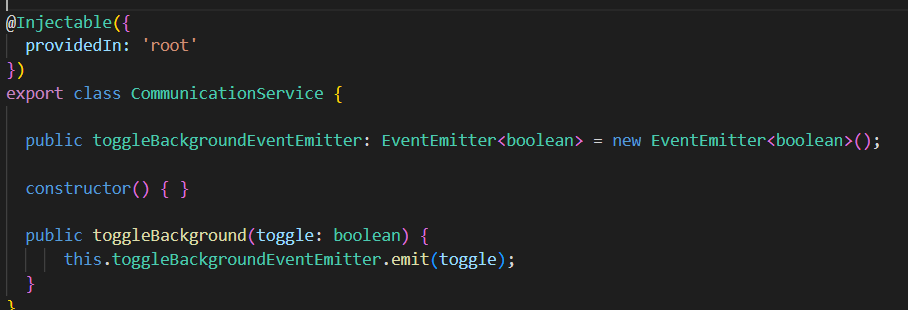
A navigációs felület komponense. Itt állítódik be mi látható user felhasználónak, illetve mi elérhető

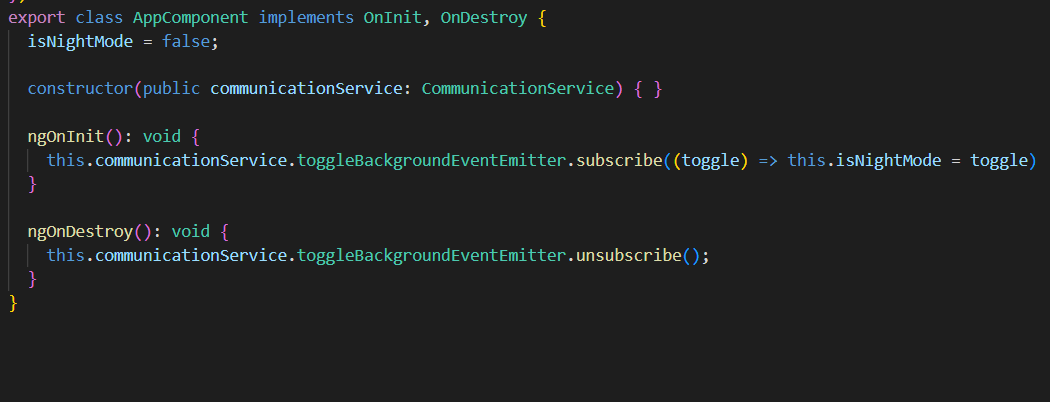
bejelentkezés előtt valamint azt követően

components 🡪 Link\_layout 🡪 layout.component.ts

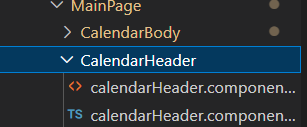
A day és night módot állítja amit a communicationService komponens továbbít, kommunikál az app.component.ts file-nak ami a háttérszínt állítja majd a app.component.css file-ban.



Service 🡪 communications.service.ts 

app.component.ts

component 🡪 MainPage 🡪 CalendarHeader



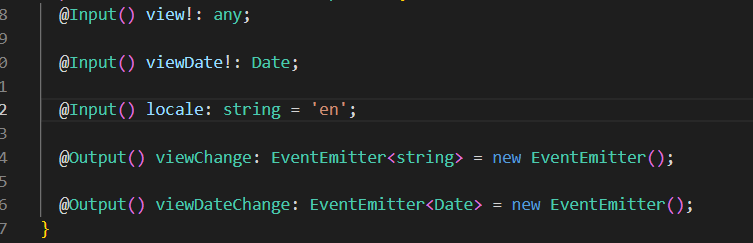
Naptár fejléc része, ez végzi a más napokra lépést és jelenlegi napra ugrást is.

A „@Input()” dekorátorok jelzik, hogy az adott tulajdonságokat(vagy változókat) a komponens szülője átadhatja a komponensnek

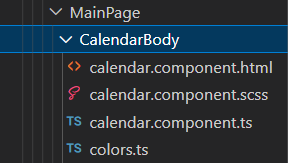
Az alapértelezett nyelvi beállítás az angol a locale nevű bemenetnél

Az „”OutPut()” dekorátorok pedig jelzik, hogy az adott tulajdonságokból(vagy változókból) eseményeket lehet kibocsátani, amelyeket a szülő komponens figyelhet meg.

viewDateChange változtatja a napi nézeteket.

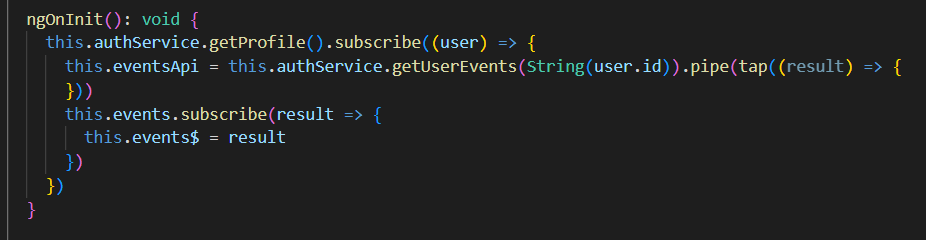


component 🡪 MainPage 🡪 CalendarBody



Naptár Fő része itt jelenik meg a felvett nap kalendár formátumban hónap, hét, nap formátumban.

components 🡪 MainPage 🡪 calendar.component.ts

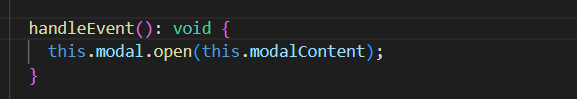
Az oldal betöltésekor mindig lefut egy getProfile metódus ami lekéri a profilnak az összes eventjét valamint a naptár listában tárolt eventeket

Az events listátba betöltődik az összes user event amit változó nevek alapján felvesz

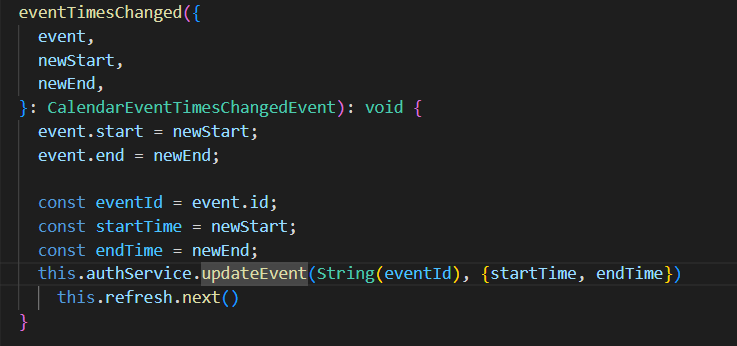
húzható az event is a „dragabble:true” segítségével

Az event szinei a kiválasztott kategóriáktól függnek. Primary és secondary színt lehetséges megadni ami az event színét majd a kijelölt event hátterét szinezi be

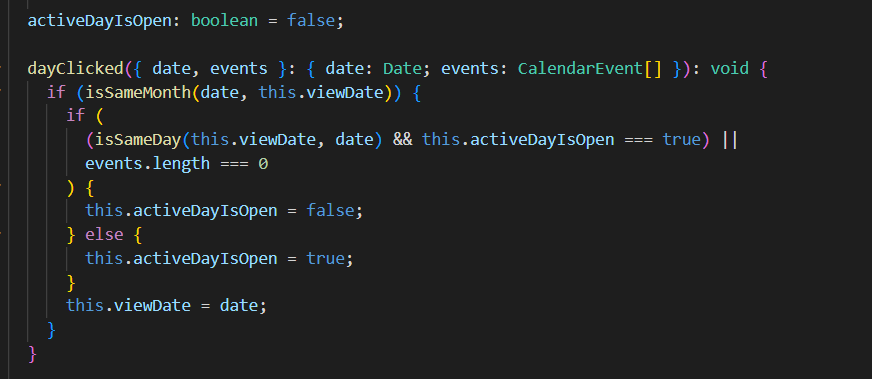


A handleEvent akkor hívodik meg amikor kiválasztunk, rákattintunk egy eventre ilyenkor megnyílik egy modal az eventeket tartalmazva amit az api-ból tölt be event modált.

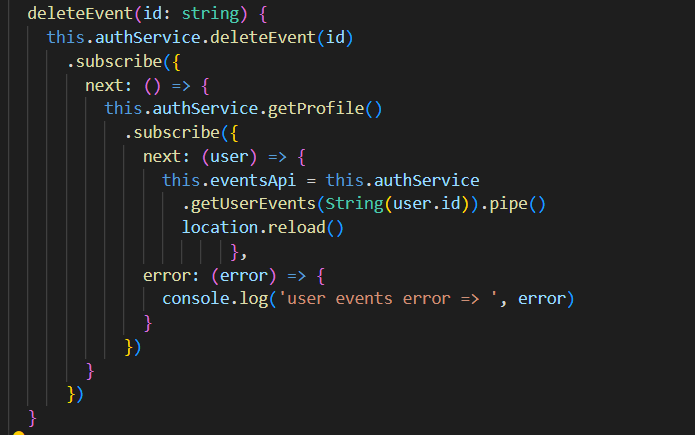
Event változtatásakor az eventTimesChanged hívódik meg

Ennek a segítségével rakódik át az event a naptár oldalon

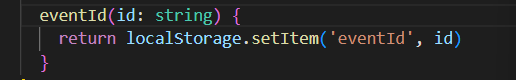
A dayClicked funkció ellenőrzi hogy a kiválasztott napot, amire kattintunk, hogy voltak-e eventek, ha igen akkor lenyil egy menü ami megjeleníti az azon a napon lévő eventeket



deleteEvent itt is arról gondoskodik hogy ki legyen törölve a kiválaszott event

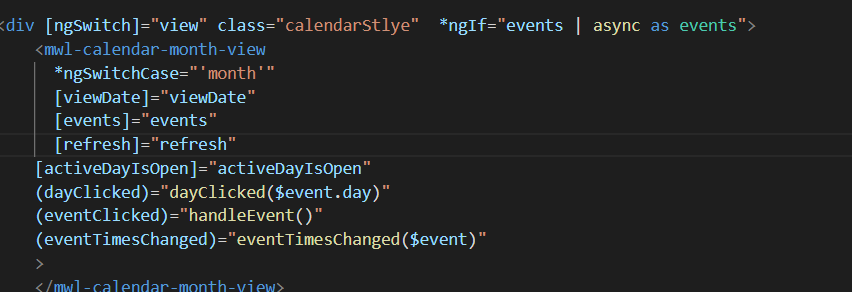


A megnyitott eventModál itt is megnyit a háttérben egy event id-it ami alapján tudja milyen eventet változtasson meg

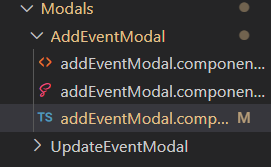


components 🡪 MainPage 🡪 calendar.components.html

A \*ngSwitchCase=’”month”’ a dátum nézetről gondoskodik



components 🡪 Modals 🡪 AddEventModal



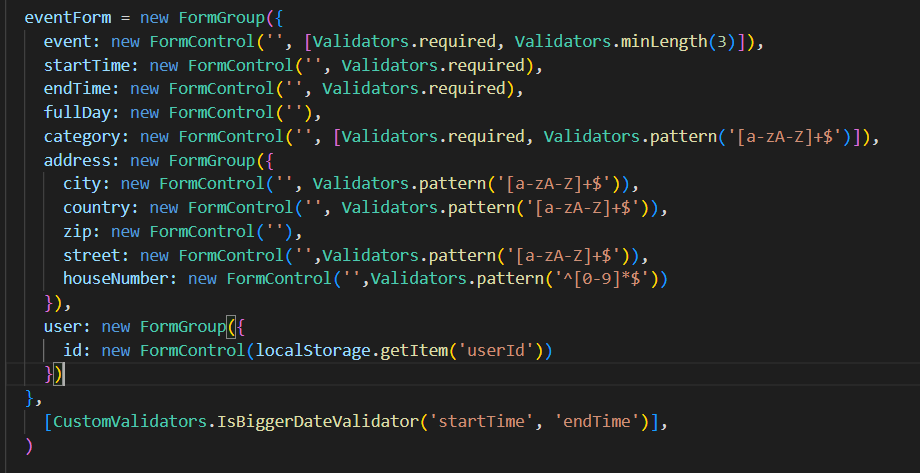
Itt kezelődik le az event felvétel modal formátumban, ami a naptárnézetben és az event oldalon is meghívódik, mint külön komponens.Ez egy külön ablak formátumban ugrik fel az új event kiválasztása során.

components 🡪 Modals 🡪 addEventModal.component.ts

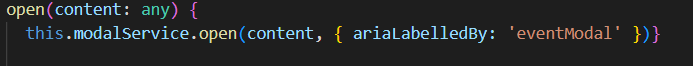
A felvett eventek az eventFrom objektumban veszódnek fel amin belül még egy adress objektum található, ahol a cím értékei vevődnek fel.

Valamint itt hívódik meg a dátum validációja is.

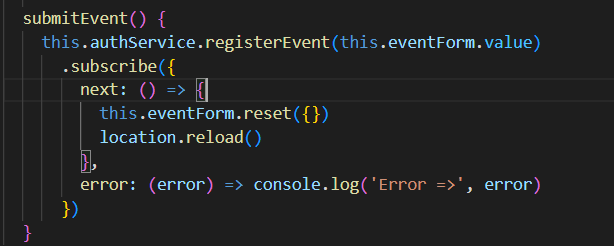
A user id-ja alapján tudja mely eventeket kell majd megjelenítsen, amit a localstorage-ból kap meg.W



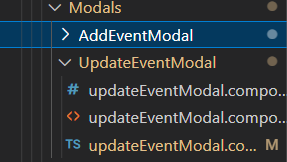
Open funkció akkor hívódik meg amikor rákattintunk egy új event létrehozására ez megnyitja a modal tartalát



SubmitEvent az event felvételekor fut le, ami meghívja az AuthService-ből a registerEvent metódust és felveszei az eventForm tartalmát.



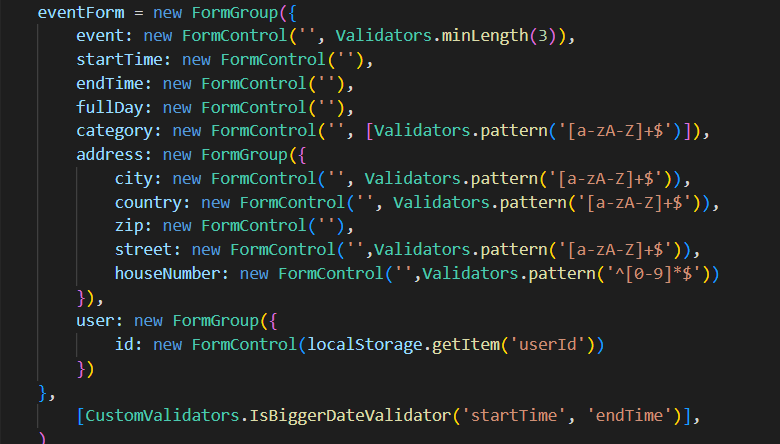
components 🡪 Modals 🡪 UpdateEventModal



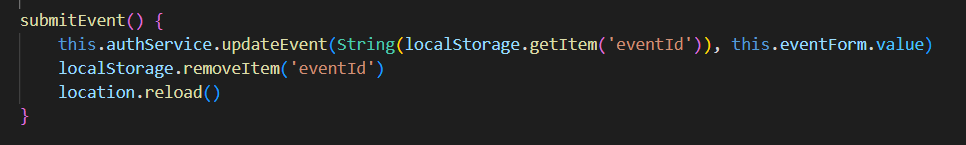
Itt az eventek megváltoztatása kezelődik le szintén modal segítségével.

components 🡪 Modal 🡪 UpdateEventModal 🡪updateEventModal.components.ts

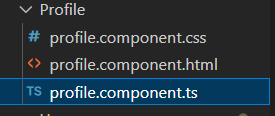
Itt nincs kötelezően megadandó mező a form felvételekor.



updateEvent az event megváltoztatásakor fut le, ami meghívja az AuthService-ből a updateEvent metódust és felveszei az eventForm tartalmát.

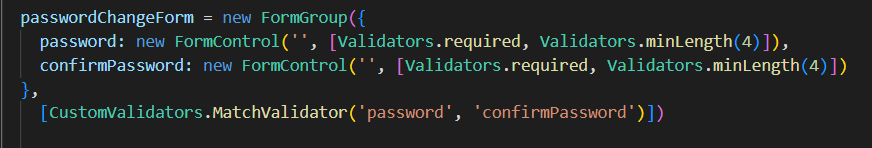


components 🡪 Profile

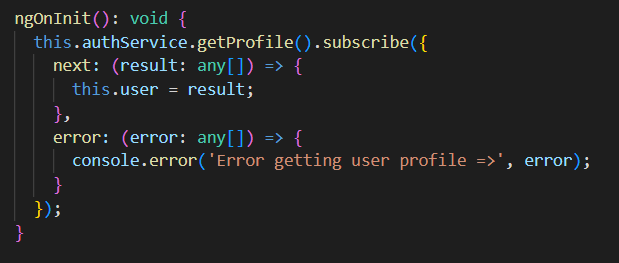


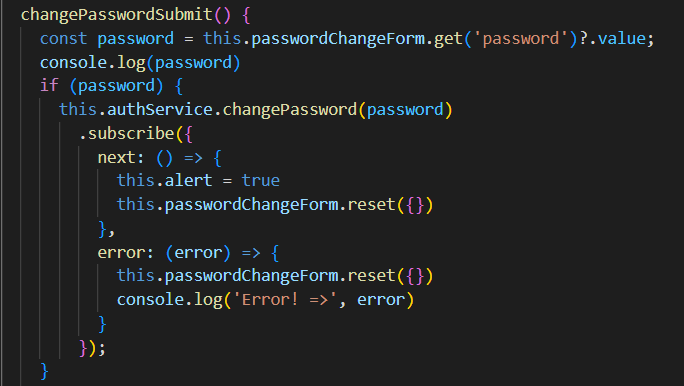
A profil oldalt a profil komponens fedi le, ami a jelszó megváltoztatását is lehetővé teszi

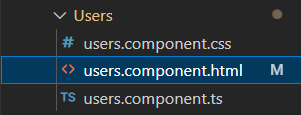
components 🡪 Profile 🡪 profile.component.ts

A profilon a jelszó megegyezés szintén a matchValidátorral ellenörződik le, egy passwordChangeForm-ban

Az oldal lefutásakor lekérődik a profil , ami tartalmazza a username, email, first name, last name-et, ez a html oldalon ki is iródik egyben.



Jelszó módosításakor lefutódik a changePasswordSubmit funkció ami kicseréli a jelenlegi felhasználó jelszavát az újonnan megadottra

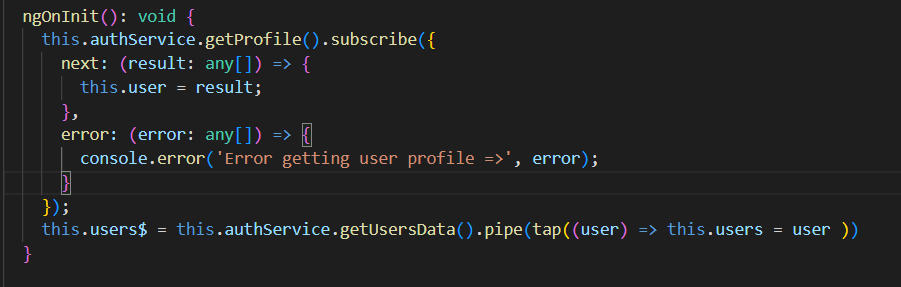
components 🡪 Users

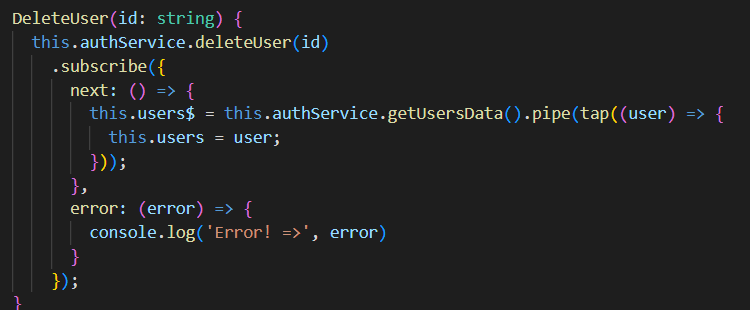
A Users komponens a Users admin felületet fedi le, amit csak admin felhasználók érhetnek el.

Lehetőségünk van itt a felhasználók role-át állítani valamit user felhasználókat törölni.

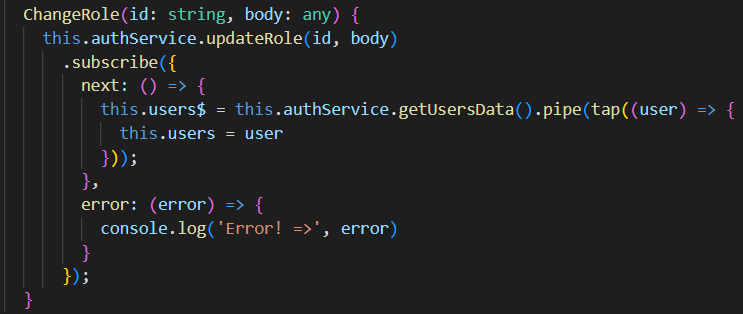
components 🡪 Users 🡪 users.component.ts

Az oldal lefutásakor megkapja a user változó a felhasználó id-át, ami a törléshez lesz felhasználva, valamint lekérődik az a getUsersDate metódussal összes felhasználó



A user felhasználók törlését a DeleteUser funkció végzi

A felhasználók szerepének a változtatására a ChangeRole funkció hivódik meg, ami megújítja a felhasználó szerepét egy id és egy role alapján amit a body kap majd meg.

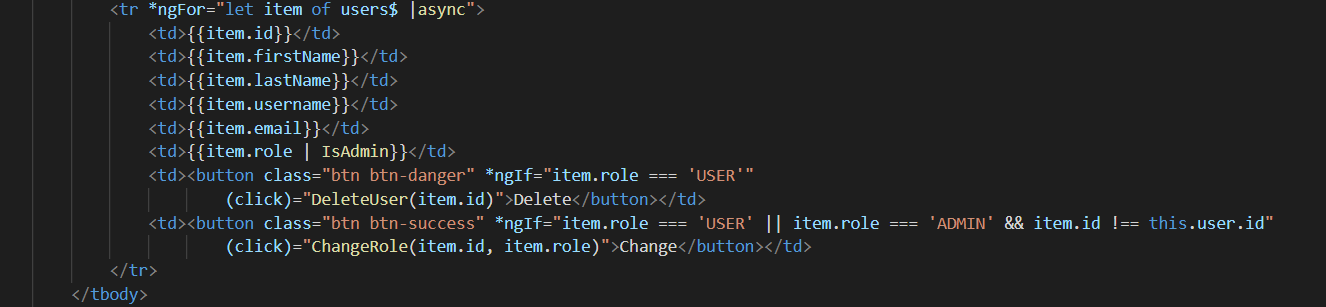


components 🡪 Users 🡪 users.component.html

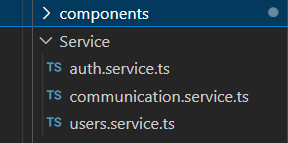
egy for ciklussal kiiródik az összes felhasználó

A felhasznló törlésnél egy If ellenőrzi hogy csak a „USER” felhasználóknál jelenjen meg a lehetőség.

A ChangeRole szintén egy If alapján kapja meg a felhasználó role-át.A role button csak akkor jelenik meg ha nem egyezik meg a jelenlegi felhasználó id-val tehát admin nem tudja a saját role-át vissza rakni USER-re.



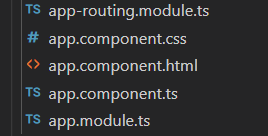
Service



Az auth.service az autentikációt kérő endpoint kéréseket kezeli, valamint a navigációs felületen azt hogy mi jelenjen meg.

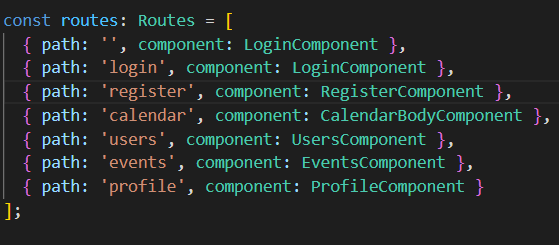
A communication.service a light és dark módért felelős a komponensek között

A users.service a nem autentikációt igénylő kérésekért felelős

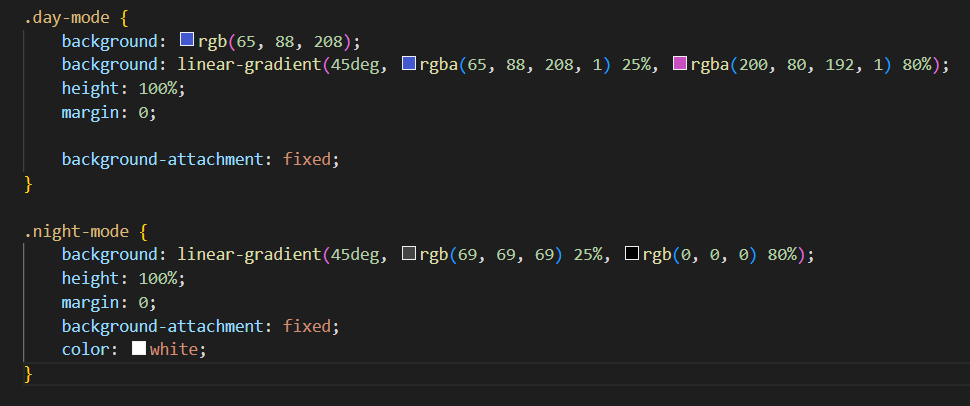


app-routing.module.ts

Itt kötődnek a Link\_layout path-ai a megfelelő komponensekkel



app.component.css a night és day mód hátterét állítja be



app.module.ts

Az összes komponens illetve modulok behívása itt történik meg.