ANGULAR 4

Peter Gurský, peter.gursky@upjs.sk

ExtendedUsersComponent

- Zapýtame si rozšírených používateľov
  - GET: http://localhost:4200/users/{token}
- Rozšírime triedu User o ďalšie parametre
  - Aj pole objektov typu Group
- Zaevidujeme si komponent v app.routes.ts
- Vypíšeme tabuľku s rozšírenými používateľmi

Životnosť servisu

- Service je singleton
  - Každý komponent, ktorý si ho nechá injektnúť vidí rovnakú inštanciu
- Pri zmene URL cez <a href=""> však dochádza k reštartu celej stránky
  - Service sa vytvára nanovo
  - Token z predchádzajúcej URL už nebude uložený
- Vieme využiť session úložisko v prehliadači
  - Funkcie v globálnom JS objekte **sessionStorage** alebo **localStorage**:
    - setItem(kľúč, hodnota)
    - getItem(kľúč)
    - removeItem(kľúč)
    - clear()
  - max 10MB pre origin (doménu) a iba stringové hodnoty

Zmena bez reštartu: Router

- Nechceme komunikovať so serverom, keď už aj tak máme natiahnuté všetky komponenty v prehliadači
- namiesto <a href="/users"> použijeme:
  - <a routerLink="/users">
    - To isté, ako keď v kóde zavoláme router.navigateByUrl("/users");
- importujeme do komponentu `RouterLink`
- router Angularu v tomto prípade iba vymení komponenty, ale nepýta server o novú stránku

Nastavovanie class link elementom

- zvýraznenie kliknutého elementu – aktívna linka v menu

```
<a routerLink="/users" routerLinkActive="active">Users</a>
```

  - Nastavíme tým class="active"
- importujeme
  - `RouterLinkActive`

navbar.component.css

```
.active {
  background: rgba(0,0,0,.2);
}
```

Login / Logout v hlavičke stránky

- Chceme zobraziť meno prihláseného používateľa
- Navbar sa má nejako dozvedieť, keď sa stav zmení
  - NavbarComponent nevie **kedy** to príde a **koľko krát** to príde – je to závislé od toho, čo urobia iné komponenty, alebo čo sa stane v servise pri komunikácii so serverom
  - Použijeme signálnu premennú, ktorú spravuje UsersService
  - Navbar ju bude sledovať a prekreslí sa, keď sa zmení

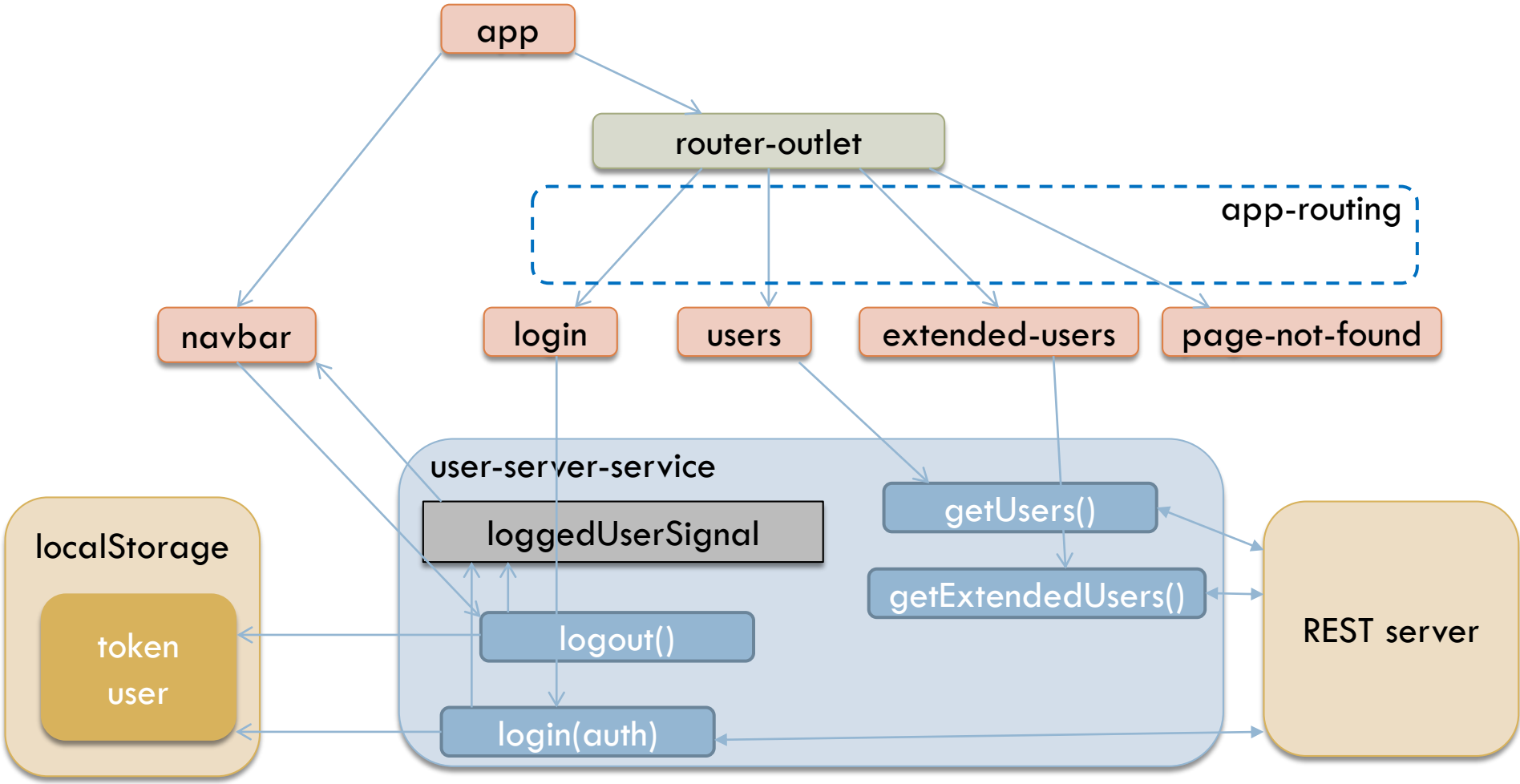
Signály

- Obal, ktorý umožňuje sledovať zmeny premennej
  - Vytvorenie
    - `const count = signal(0);`
  - Čítanie
    - `console.log('The current count is: ' + count());`
  - Nastavovanie
    - `count.set(2);`
    - `count.update(value => value + 1);`
  - Reagovanie na zmenu
    - `effect(() => console.log('Count = ' + count()));`
  - Závislý signál
    - `const doubleCount = computed(() => count() * 2);`

NavBar

- Na informovanie o tom, kto je prihlásený, použijeme signál
  - V usersService:
    - `loggedUser = signal(this.username);`
  - V navbar.component.ts:
    - `usersService = inject(UsersServerService);`
    - `userName = this.usersService.loggedUser;`
  - V navbar.component.html:
    - `{{userName()}}`

Aktuálny stav



Material table - zložitejšie stĺpce

ng-container sa nestane elementom v DOM

```
<ng-container matColumnDef="lastLogin">
  <th mat-header-cell *matHeaderCellDef>Last login</th>
  <td mat-cell *matCellDef="let user">
    {{ user.lastLogin | date: 'd.M.y H:mm:ss' }}
  </td>
</ng-container>
```

Vlastná pipe

- Spravíme si vlastnú pipe pre výpis názvov skupín a práv z nich vyplývajúcich
  - transform(values: Group[], property?: string): string { }
- ng g pipe groups-to-string
  - Importujeme triedu GroupsToStringPipe do komponentu
- použijeme ju v šablóne

```
<ng-container matColumnDef="permissions">
  <th mat-header-cell *matHeaderCellDef>Permissions</th>
  <td mat-cell *matCellDef="let user">
    {{ user.groups | groupsToString: 'permissions' }}
  </td>
</ng-container>
```