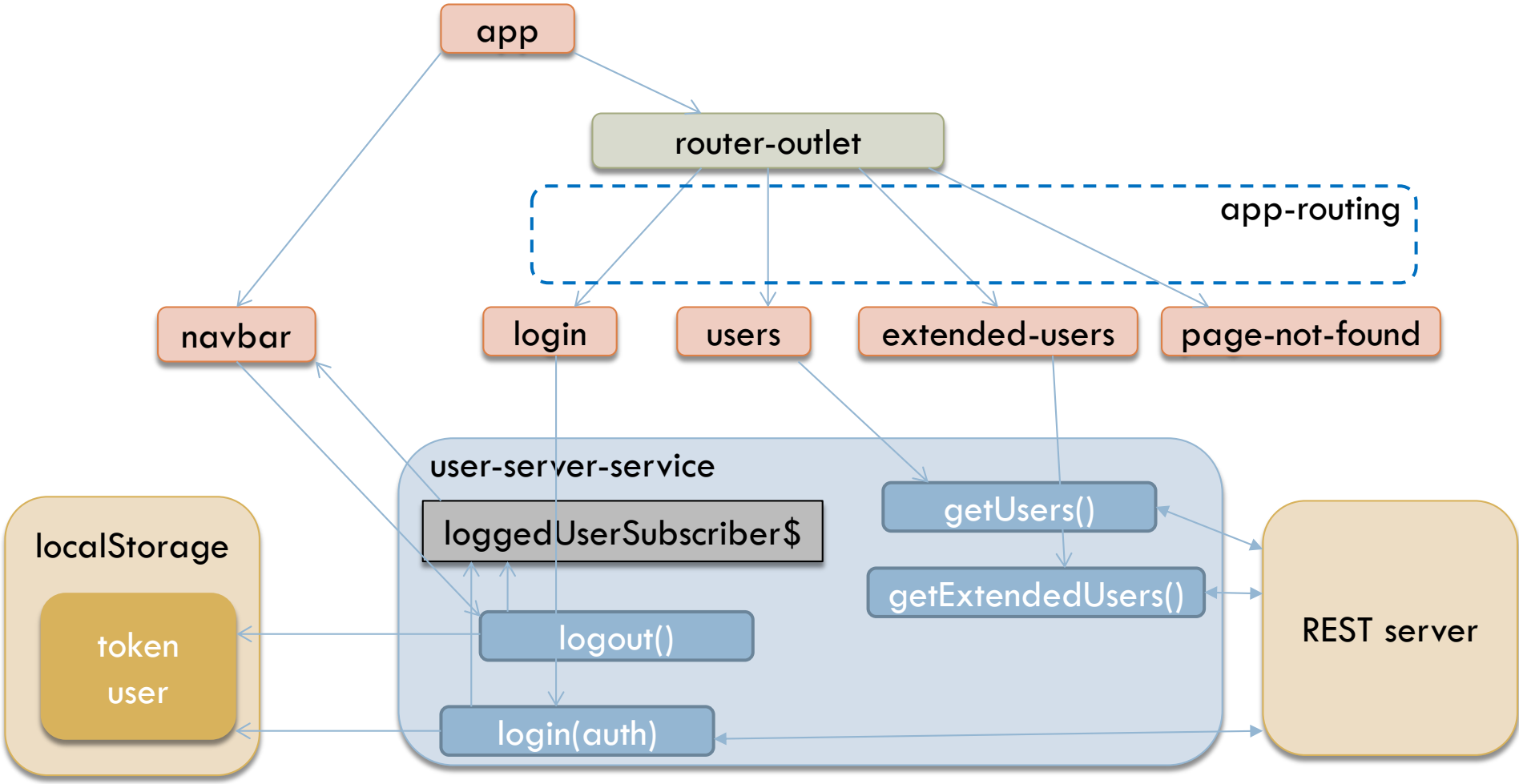
ANGULAR – 5. ČASŤ

Peter Gurský, peter.gursky@upjs.sk

Aktuálny stav



Refaktor

- Chybové/úspechové hlášky by mohol vypisovať message komponent
  - vypisovať chybové, ale aj pozitívne správy
  - všetky možné chyby komunikácie
    - nedostupný server
    - zlý login alebo heslo
    - nedostatočné práva
  - prebaliť chyby do vlastného message objektu
    - poslať ho message komponentu
      - vypísať správu, nech už bude akákoľvek
    - presmerovať sa na hlavnú stránku?

Téma: komunikácia komponentov

- **Cieľ:** Vytvoríme si komponent s formulárom, v ktorom budeme môcť pridať nového používateľa
- V src/app spustíme
  - ng g component user-edit
- Ak ho chceme vidieť v komponente extended-users, vložíme do jeho šablóny tag, ktorý sa volá rovnako, ako selector v novom komponente

```
<app-user-edit></app-user-edit>
```

Vytvoríme šablónu obsahujúcu formulár

- Spravíme si modálne okno, ktoré bude predstavovať náš nový komponent na editáciu
- Tlačidlo na jeho zobrazenie ponecháme v rodičovskom komponente extended-users.component
- Pozrime sa, ako sa robia modálne okná v Bootstrape a okopírujme si HTML zdrojáky

Model editačného komponentu

- Základný model komponentu, v ktorom budeme editovať nového používateľa je objekt typu User
- Môžeme si ho v komponente vyrobiť prázdneho, aby sa prvky šablóny mali s čím previazať

```
...
export class UserEditComponent implements OnInit {
   private user: User = new User("","");
   ...
}
```

Pomocný text s obsahom modelu/user-a

app/user-edit/user-edit-component.html (časť)

```
<div class="modal-body">
        <p>aktuálny user: {{vypisUsera}}</p>
</div>
```

app/user-edit/user-edit-component.ts

```
get vypisUsera():string {
    return JSON.stringify(this.user);
}
```

Editácia skupín

- Používatelia majú v sebe iba nim priradené skupiny – noví dokonca žiadne
- V systéme však sú aj ďalšie skupiny
  - Získame si ich zo servera
    - GET: /groups
- V komponente bude modelom pre vybraté skupiny mapa zo skupín na boolean členstva user-a v nich

```
private groups: Map<Group,boolean> = new Map<Group,boolean>();
```

*ngFor pre mapy (od Angularu 6)
pozn: [(ngModel)] nefunguje

```
<div *ngFor="let item of groups | keyvalue">
        <label>
                <input type="checkbox" name="grchb"
                  [checked]="item.value"
                  (change)="toggleGroup($event,item.key)">
            {{item.key.name}}
        </label>
</div>
```

```
toggleGroup(event, group) {
      this.groups.set(group, event.target.checked);
 }
```

*ngFor pre mapy (od Angularu 6)
pozn: [(ngModel)] nefunguje

```
<div *ngFor="let item of groups | keyvalue">
```

Pre Angular < 6 je potrebné v komponente prerobiť mapu na pole objektov dvojíc a iterovať toto pole:

```
private mapArray = [];

value.forEach((entryVal, entryKey) => {
    mapArray.push({key: entryKey, val: entryVal });
});
```

```
        this.groups.set(group, event.target.checked);
    }
```

Validácia vstupu

- Nechceme, aby meno ostalo prázdne a ak áno, tak o tom informovať používateľa
- ngModel nám znova pomôže – nastavuje sám od seba pre každý formulárový element jednu z tried
  - ng-touched / ng-untouched – element bol navštívený / nebol
  - ng-dirty / **ng-pristine** – hodnota je zmenená / nie je
  - **ng-valid** / **ng-invalid** - hodnota je správna / nie je
    - ak element má atribút required, hodnota musí byť vyplnená
    - okrem required máme aj: minlength, maxlength, pattern
    - …zložitejšie kontroly viď. Validator v ngModel-i
- Pozrime si cez inšpektora

Naštýlujme si elementy podľa týchto tried

app/user-edit/user-edit-component.css

```
.ng-valid[required] {
  border-left: 5px solid limegreen;
}

.ng-invalid  {
  border-left: 5px solid darkred;
}
```

- selektor . označuje elementy s danou triedou
- [required] znamená, že daný element musí mať aj atribút required