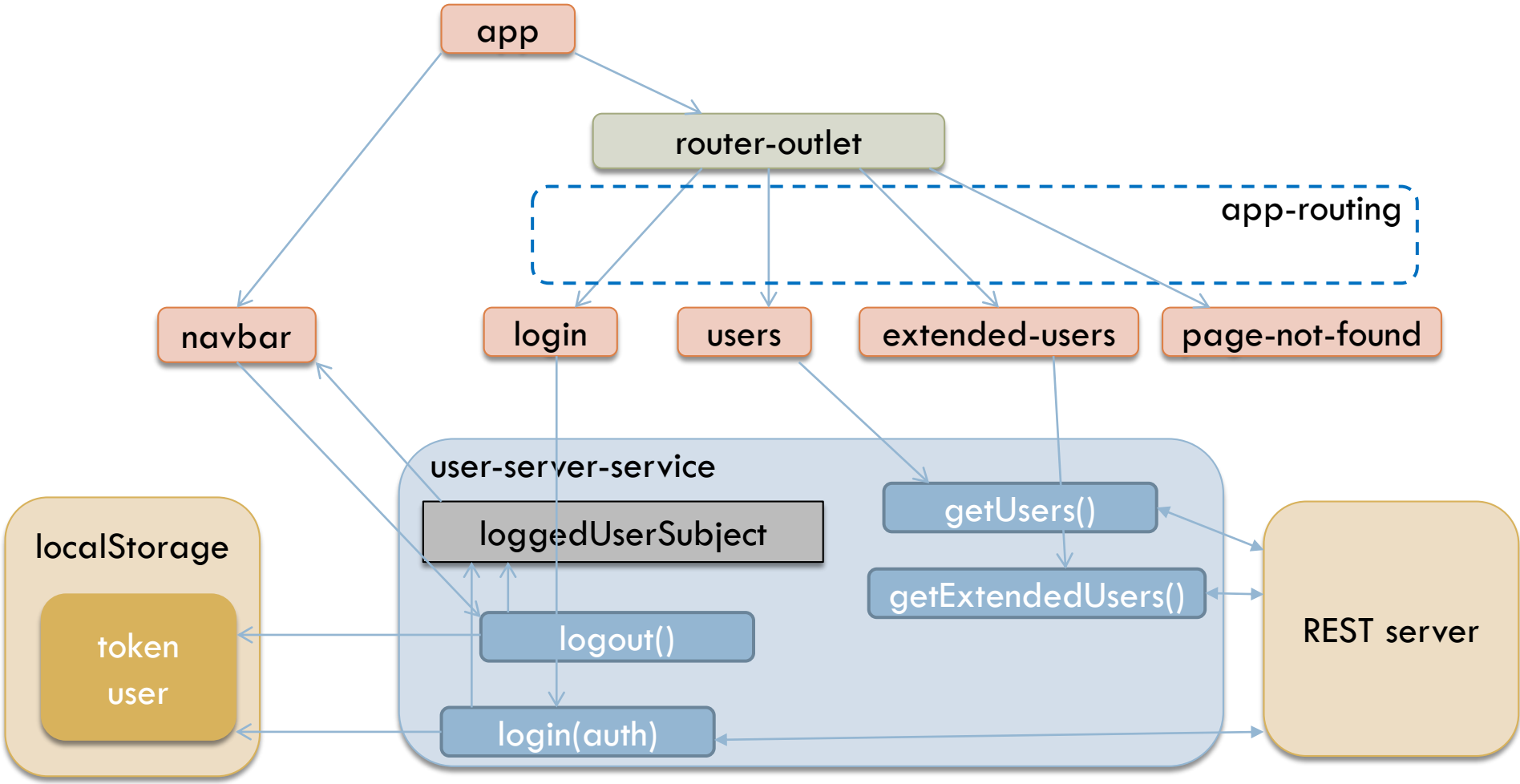
ANGULAR – 5. ČASŤ

Peter Gurský, peter.gursky@upjs.sk

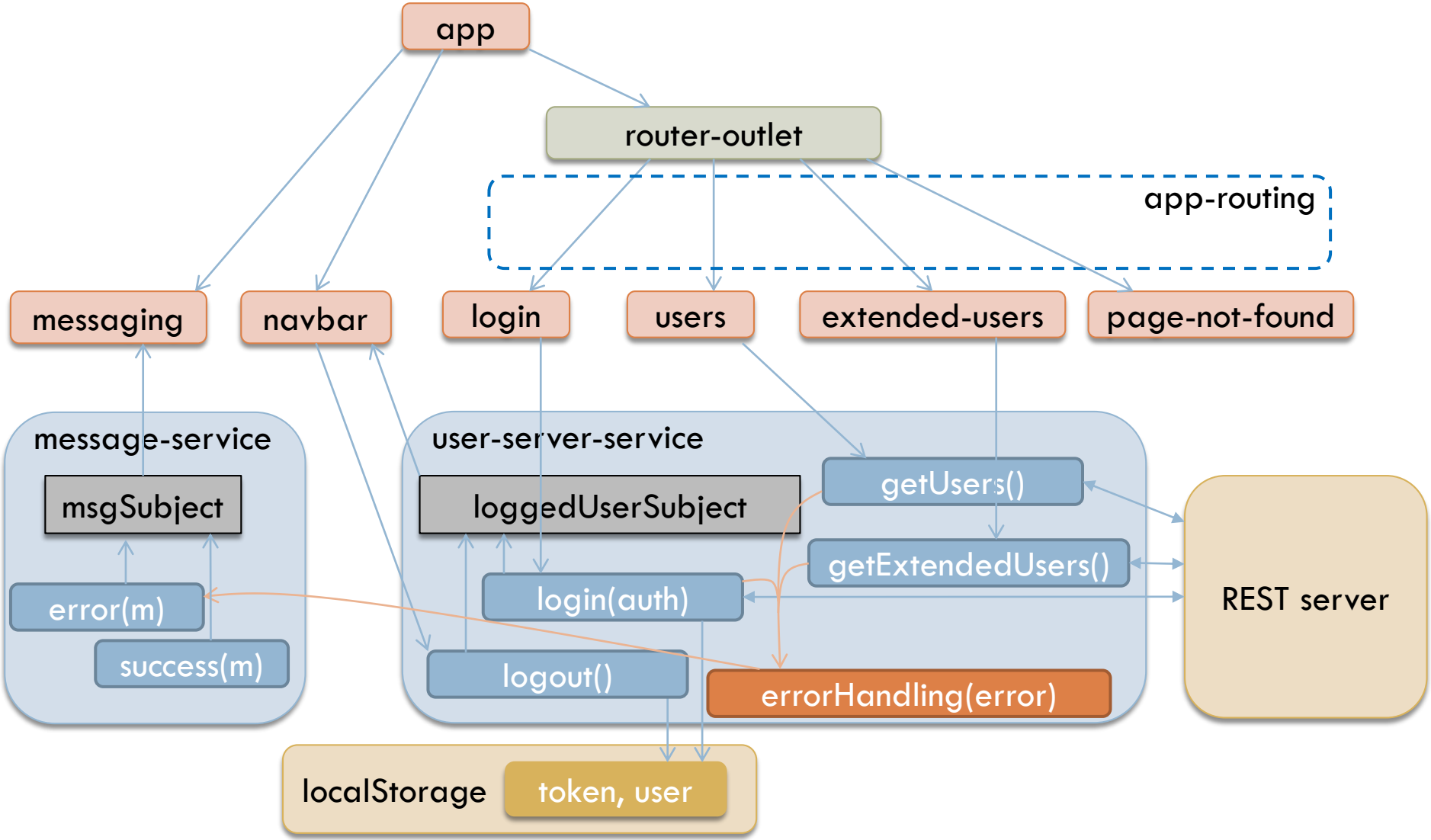
Aktuálny stav



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?

Aktuálny stav



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);
}
```

Nastavovanie skupín

- Používatelia majú v sebe iba im priradené skupiny – noví dokonca žiadne
- V systéme však sú aj ďalšie skupiny
  - Získame si ich zo servera
    - GET: /groups
- Graficky to urobíme tak, že každá skupina bude mať pri sebe checkbox, ktorý má byť zaškrtnutý, ak používateľ má byť členom danej skupiny
- Modelom pre checkbox je boolean bez popisu
- My si vyrobíme rozšírený model, v ktorom budeme vedieť, ktorá skupina patrí ku ktorému checkboxu.

groups: {group: Group, selected: boolean}[]

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

Dodajme aj upozorňujúci text

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

Nová premenná

Sivou sú veci z Bootstrap-u

```
<label for="userName" class="col-sm-3 col-form-label">
  User name :
</label>
<div class="col-sm-9">
  <input type="text" id="userName" class="form-control"
   [(ngModel)]="user.name" name="username" required minlength="3"
   #nameModel="ngModel">
  <div class="invalid-feedback" [style.display]="'block'"
   *ngIf="nameModel.invalid && nameModel.touched">
        Name must have at least 3 characters.
  </div>
</div>
```

Odoslanie obsahu formulára rodičovi

- ...ostatné parametre user-a urobíme analogicky
- odoslanie spravíme cez tlačidlo na uloženie
  - obalíme celú šablónu do elementu <form>
  - znefukčníme tlačidlo, ak je formulár nevalidný
  - v komponente vytvoríme obslužnú metódu onSumbit()

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

Zatvorí modálne okno

```
<form (ngSubmit)="onSubmit()" #userForm="ngForm">
  ...
  <button type="submit" class="btn btn-primary" data-bs-dismiss="modal"
     [disabled]="userForm.form.invalid">Ulož</button>
  ...
</form>
```

Odoslanie nového user-a rodičovi

user-edit-component.ts

```
import {EventEmitter, Output } from '@angular/core';

export class UserEditComponent ...{
  @Output('change') eventPipe = new EventEmitter<User>();
   …
   onSubmit() {
       ...
       this.eventPipe.emit(this.user);
   }
}
```

Odoslanie nového user-a rodičovi

user-edit-component.ts

```
import {EventEmitter, Output } from '@angular/core';

export class UserEditComponent ...{
  @Output('change') eventPipe = new EventEmitter<User>();
   …
   onSubmit() {
       ...
       this.eventPipe.emit(this.user);
   }
}
```

extended-users-component.html

```
<app-user-info (change)="onUserChanged($event)"></app-user-info>
```

Odoslanie nového user-a rodičovi

user-edit-component.ts

```
import {EventEmitter, Output } from '@angular/core';

export class UserEditComponent ...{
  @Output('change') eventPipe = new EventEmitter<User>();
   …
   onSubmit() {
       ...
       this.eventPipe.emit(this.user);
   }
}
```

extended-users-component.ts

```
onUserChanged(user:User) {
    this.users.push(user);
}
```

extended-users-component.html

```
<app-user-info (change)="onUserChanged($event)"></app-user-info>
```

Pridávanie/úprava používateľa

app/users/users-service.ts

```
import { HttpClient } from '@angular/common/http';
...
private restServerUrl: string = "http://localhost:8080/";

public saveUser(user: User):Observable<User> {
   return this.httpClient.post<User>
            (this.restServerUrl + "users/"+ this.token, user)
             .pipe(catchError(error => this.processError(error)));
}
```

edit-user pre pridávanie aj editáciu

- Potrebujeme, aby rodičovský komponent informoval o tom, koho editujeme
- V user-edit.component.ts si zadefinujeme, ktoré inštančné premenné chce mať na vstupe
  - Rodičovský komponent nevolá konštruktor, to robí framework
  - premenná user (nový alebo editovaný),
  - premenná actionWithUser (text, čo sa deje)
- V users.component.html zadefinujeme hodnoty pre tieto premenné

Odoslanie vstupu od rodiča

user-edit-component.ts

```
import {Input, OnChanges} from '@angular/core';

export class UserEditComponent implements OnChanges {
  @Input() public user: User;
  @Input() public actionWithUser: string;
   …
   ngOnChanges() { }
}
```

extended-users-component.html

```
<app-user-info [user]="selectedUser" [actionWithUser]="actionWithUser"
(change)="onUserChanged($event)"></app-user-info>
```

- Ešte doplníme obslužné metódy pre udalosti, keď používateľ chce pridať alebo editovať používateľa, ktoré nastavia actionWithUser a selectedUser

REST server - metódy

Ďalší preddefinovaní používatelia a ich heslá :
Lucia : lucia
John : john
Andrej : andre

- GET: /users
- POST: /login
  - V tele pošleme {"name": "Peter", "password": "elct"}
  - príde vygenerovaný token
- GET: /users/{token}
  - prídú používatelia aj s neverejnými atribútmi
- GET: /user/{id}/{token}
- GET: /bg-user/{id}/{token}
  - vnútorná reprezentácia používateľa (obsahuje aj heslo)
- POST: /users/{token}
  - Cez POST pošleme JSON používateľa, ktorého chceme uložiť
- DELETE: /user/{id}/{token}

REST server - metódy

- GET: /groups
- GET: /group/{id}
- POST: /groups/{token}
  - Cez POST pošleme JSON skupiny, ktorú chceme uložiť
- DELETE: /group/{id}/{token}

Nastavenie @Input() parametrov

extended-users-component.html (časť)

```
<tr *ngFor="let user of users" (dblclick)="editUserClick(user)">
```

extended-users-component.ts

```
addUserButtonClick() {
      this.selectedUser = new User("","");
      this.actionWithUser = "add";
}

editUserClick(user: User) {
      this.selectedUser = user;
      this.actionWithUser = "edit";
      $('#myModal').modal('toggle');  // zobrazí modálne okno
}
```

Vymazanie používateľa

app/users/users-service.ts

```
import { HttpClient } from '@angular/common/http';
...
private restServerUrl: string = "http://localhost:8080/";

public deleteUser(user: User):Observable<boolean> {
    return this.httpClient.delete<boolean>
            (this.restServerUrl + "user/" + user.id+ "/" + this.token)
            .pipe(map(_ => true),
                    catchError(error => this.processError(error)));;
}
```