Laboratorium 8: Formularz dodawania pojedynczej galerii

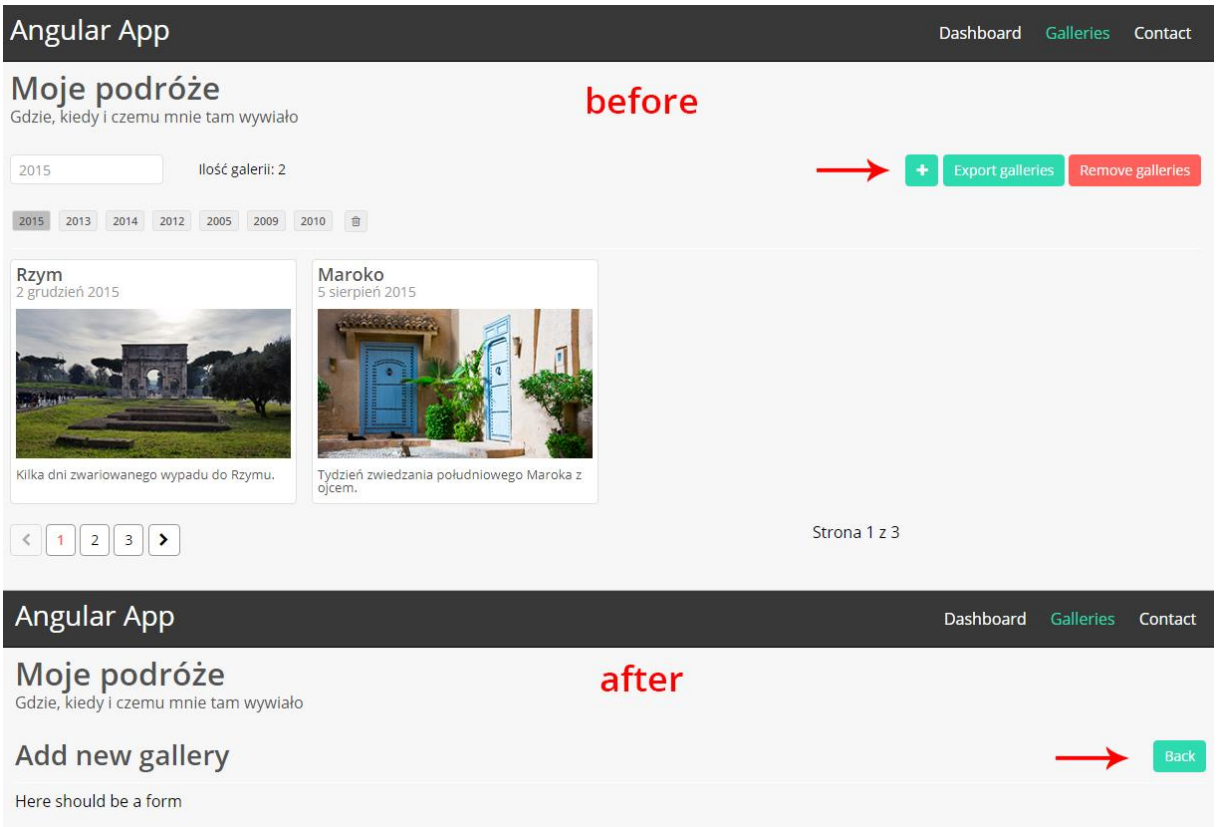
Na tych zajęciach zajmiemy się dodawaniem nowych galerii ze strony głównej. Jak do tej pory można było jedynie eksportować je z pliku. Teraz dodamy też nową galerię.

Formularz będzie schowany na liście z galeriami i będzie pokazywał się, po kliknięciu przycisku do dodania nowej galerii.

Obejmie to kilka kroków:

1. Dodamy przyciski do pokazywania/chowania formularza galerii
2. Stworzymy prosty formularz z dodawaniem 1 zdjęcia
3. Zapiszemy formularz
4. Stworzymy mechanizm do dodawania większej ilości zdjęć

Ćwiczenie 1. Przyciski do formularza galerii



1. Do komponentu *galleries.component.ts* dodajemy nową zmienną showGalleryForm i w ngOnInit() ustawiamy ją na false

```
showGalleryForm: boolean;

httpOptions = {
    headers: new HttpHeaders({
        'Content-Type': 'application/json',
        'Authorization': '3'
    })
};

constructor(private http: HttpClient) {

    this.title = 'Moje podróże';
    this.description = 'Gdzie, kiedy i czemu mnie tam wywiało';
}

ngOnInit() {
    this.showGalleryForm = false;
```

2. Następnie w *galleries.component.html* dodajemy przyciski do dodawania galerii, które mają działać następująco:

   - jeśli *!showGalleryForm*, widać przycisk do dodania galerii i listę galerii

```
<div class="grid-8 text-right" *ngIf="!showGalleryForm">
    <button class="button button-success" (click)="showGalleryForm=true"><i
class="fa fa-plus"></i></button>
</div>
```

   - jeśli *showGalleryForm*, widać formularz i przycisk do powrotu do listy z galeriami

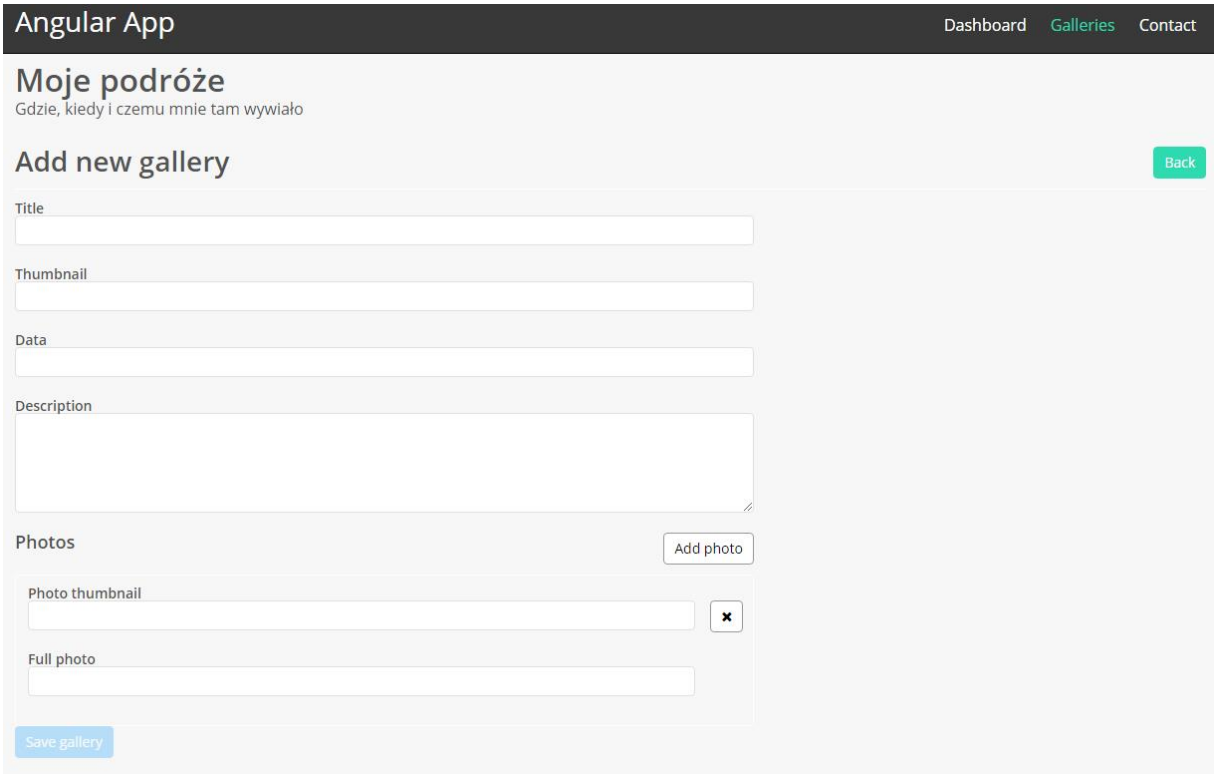
```
<div class="row" *ngIf="showGalleryForm">
    <div class="grid-8">
        <h2>Add new gallery</h2>
    </div>
    <div class="grid-8 text-right">
        <button class="button button-success"
(click)="showGalleryForm=false">Back</button>
    </div>
</div>
```

Dokończ!

Te same warunki trzeba wykorzystać, żeby schować listę galerii, paginację i wyszukiwarki jeśli *showGalleryForm*, i zamienić je na formularz do dodawania galerii.

Poklikaj w przyciski, jeśli raz widać listę galerii, raz miejsce na formularz, idziemy dalej.

Ćwiczenie 2. Formularz do galerii



1. Tworzymy nowy komponent **add-gallery-form**

   `ng generate component components/add-gallery-form`

   I dołączamy go na stronie z listą w miejscu, gdzie ma się pojawić po kliknięciu przycisku dodawania galerii.

2. W *add-gallery-form.component.html* budujemy formularz dodawania galerii (tak jak to było w komentarzach). Kod formularza może wyglądać tak jak poniżej– sprawdź, czy masz takie same nazwy pól w galerii!

```
<form #galleryForm="ngForm" (ngSubmit)="onSubmit()" novalidate>
    <div class="row">
        <div class="grid-16 input-group">
            <label>Title</label>
            <input type="text" class="input-control" name="title" #title="ngModel"
[(ngModel)]="gallery.title" required>
            <p class="text-danger" [hidden]="title.valid || title.pristine">Title
required!</p>
        </div>
        <div class="grid-16 input-group">
            <label>Thumbnail</label>
            <input type="text" class="input-control" name="thumbUrl"
#thumbUrl="ngModel" [(ngModel)]="gallery.thumbUrl" required>
            <p class="text-danger" [hidden]="thumbUrl.valid ||
thumbUrl.pristine">Thumbnail required!</p>
        </div>
        <div class="grid-16 input-group">
            <label>Data</label>
            <input type="text" class="input-control" name="dateCreated"
#dateCreated="ngModel" [(ngModel)]="gallery.dateCreated" required>
            <p class="text-danger" [hidden]="dateCreated.valid ||
dateCreated.pristine">Date required!</p>
        </div>
        <div class="grid-16 input-group">
            <label>Description</label>
            <textarea class="input-control" name="description"
#description="ngModel" [(ngModel)]="gallery.description" required></textarea>
            <p class="text-danger" [hidden]="description.valid ||
description.pristine">Description required!</p>
        </div>
    </div>
    <div class="row">
        <div class="grid-8"><h4>Photos </h4></div>
        <div class="grid-8 text-right"><button type="button" (click)="addPhoto()"
class="button button-default">Add photo</button></div>
    </div>

    <fieldset *ngFor="let photo of gallery.photos; let i = index">
    <div class="row">
        <div class="grid-15 input-group">
            <label>Photo thumbnail</label>
            <input type="text" class="input-control" name="thumbUrl{{i}}"
#thumbUrl="ngModel" [(ngModel)]="photo.thumbUrl" required>
            <p class="text-danger" [hidden]="thumbUrl.valid ||
thumbUrl.pristine">Thumbnail required!</p>
        </div>
        <div class="grid-1"><br>
            <button type="button" (click)="removePhoto(i)" class="button button-
default"><i class="fa fa-remove"></i></button>
        </div>
        <div class="grid-15 input-group">
            <label>Full photo</label>
            <input type="text" class="input-control" name="imgUrl{{i}}"
#imgUrl="ngModel" [(ngModel)]="photo.imgUrl" required>
            <p class="text-danger" [hidden]="imgUrl.valid || imgUrl.pristine">Photo
required!</p>
        </div>
    </div>
    </fieldset>
    <div class="row">
        <div class="grid-16">
            <button class="button button-primary"
[disabled]="!galleryForm.form.valid">Save gallery</button>
        </div>
    </div>
</form>
```

3. W *add-gallery-form.component.ts* musimy teraz zdefiniować pustą galerię.

```
import { Component, OnInit } from '@angular/core';
import { IGallery } from '../../../interfaces/igallery';
import * as uuid from 'uuid/v4';

@Component({
  selector: 'add-gallery-form',
  templateUrl: './add-gallery-form.component.html',
  styleUrls: ['./add-gallery-form.component.scss']
})
export class AddGalleryFormComponent implements OnInit {

    gallery: IGallery;

    constructor() { }

    ngOnInit() {
        this.gallery = this.setEmptyGallery();
    }

    setEmptyGallery() {
        return {
            title: '',
            thumbUrl: '',
            dateCreated: '',
            description: '',
            tags: [],
            photos: [{
                this.setEmptyPhoto()
            }]
        };
    }

    setEmptyPhoto() {
        return {
            photoId: uuid(),
            thumbUrl: '',
            imgUrl: ''
        };
    }
}
```

4. Ok, to formularz już jest. Teraz dodajmy funkcje do dodawania i usuwania zdjęć.

```
addPhoto() {
    this.gallery.photos.push(this.setEmptyPhoto());
}

removePhoto(index) {
    if (this.gallery.photos.length > 0) {
        this.gallery.photos.splice(index, 1);
    }
}
```

Usuwamy zdjęcia tylko do momentu, aż zostanie jedno.

Ćwiczenie 3. Zapisywanie galerii

Ostatni etap jest już łatwy. Jeśli klikniemy na **Save gallery**, galeria powinna się zapisać, a formularz zniknąć. W tym przypadku skorzystamy z event emitera, żeby powiadomić listę galerii, że ma dodać galerię do listy galerii i zamknąć formularz.

1. Dopisujemy event emiter w *add-gallery-form.component.ts*:

```
@Output()
saveGallery: EventEmitter<string> = new EventEmitter<string>();
```

2. Wywołujemy go w onSubmit() i przekazujemy mu galerię:

```
onSubmit() {
    this.saveGallery.emit(this.gallery);
}
```

3. W komponencie z listą galerii *galleries.component.ts* dopisujemy dodawanie galerii do bazy:

```
saveGallery(event) {
    this.http.post('http://project.usagi.pl/gallery', event,
this.httpOptions).toPromise().then((response) => {
        this.galleries.push(response);
        this.numberOfPages = Array(Math.ceil(this.galleries.length /
this.limit)).fill(1);
        this.showGalleryForm = false;
    });
}
```

4. A w pliku *galleries.component.html* dodajemy event do wywołania komponentu z formularzem.

```
<div *ngIf="showGalleryForm">
    <add-gallery-form (saveGallery)="saveGallery($event)"></add-gallery-
form>
</div>
```

Ćwiczenie 4. Edycja istniejącej galerii

Mamy formularz i możemy dodać nową galerię, ale… jeszcze fajniej byłoby móc zmienić dane w galerii istniejącej. Nic prostszego, w końcu wszystko już mamy przygotowane.

1. Na początek przechodzimy do pliku z pojedynczą galerią gallery-details.component.html i dodajemy w nim przycisk, który po kliknięciu pozwoli nam wyświetlić formularz do edycji galerii – dokładnie jak to było na liście galerii. Oczywiście dodajemy też komponent z formularzem.
2. W pliku gallery-details.component.html dodajemy te same zmienne i funkcje, co wcześniej przy dodawaniu nowej galerii.

CAŁOŚĆ POWINNA DZIAŁAĆ IDENTYCZNIE JAK NA LIŚCIE GALERII!

3. Jedyna różnica polega na tym, że teraz chcielibyśmy zmienić galerię istniejącą, więc musimy ją do formularza wczytać przez dodanie [(gallery)]=”gallery” do komponentu formularza.

```
<add-gallery-form [(gallery)]="gallery"
(saveGallery)="saveGallery($event)"></add-gallery-form>
```

Oraz w pliku add-gallery-form.component.ts dodajemy @Input() oraz w ngOnInit przypisujemy przekazaną galerię, jeśli istnieje, a jeśli nie – pustą galerię.

```
@Input()
gallery?: IGallery;

constructor() { }

ngOnInit() {
    this.gallery = (this.gallery && this.gallery.galleryId) ? this.gallery
: this.setEmptyGallery();
}
```

4. Na koniec trzeba jeszcze informacje o galerii nadpisać, zamiast dodawać nową. Czyli w pliku gallery-details.component.ts dodajemy zmieniamy funkcję saveGallery.

```
saveGallery(event) {
    this.http.post('http://project.usagi.pl/gallery/' + this.galleryId,
event, this.httpOptions).toPromise().then((response: IGallery) => {
        this.gallery = response;
    });
}
```

Praca samodzielna

1. Podobnie jak zdjęcia dodaj możliwość dodawania wielu tagów.