Laboratorium 7: Comments API

Tym razem celem laboratoriów będzie wykorzystanie API do zapisu komentarzy do galerii. Wykorzystamy do tego rozwiązania, które opracowaliśmy wcześniej w laboratorium o RESTful APi.

Do dyspozycji mamy tym razem:

POST 'http://project.usagi.pl/comment' – tworzy komentarz

GET 'http://project.usagi.pl/comment/byGallery/{galleryId}' – pobiera wszystkie komentarze w galerii

POST 'http://project.usagi.pl/comment/delete/{commentId}' - usuwa komentarz

Przejdźmy do komponentu z komentarzami (laboratorium Tworzenie komentarzy, nr 5)

Ćwiczenie 1. Przygotowanie do pracy z HTTP

1. Żeby nie mieć problemów z typami danych sprawdźmy na początek plik IComment.ts, powinien wyglądać tak:

```
export interface IComment {
    galleryId: number;
    commentId: number;
    dateCreated: Date;
    firstName: string;
    lastName: string;
    email: string;
    message: string;
}
```

2. Z kolei nasz pusty komentarz w pliku z formularzem komentarzy powinien wyglądać tak:

```
private setEmptyComment() {
    return {
        galleryId: parseInt(this.galleryId),
        commentId: null,
        firstName: '',
        lastName: '',
        email: '',
        message: '',
        dateCreated: new Date();
    };
}
```

Ponieważ *galleryId* pobieramy jako string z adresu strony, zaś w komentarzu występuje jako *number* – używamy *parseInt()* do konwersji.

CommentId zostanie przyznane przez bazę danych, więc możemy usunąć własne generowanie id.

3. **Zadanie samodzielne**

   W funkcji onSubmit() zakomentuj linię odpowiadającą za zapis do localStorage i dodaj funkcję do zapisu komentarza do bazy danych – analogicznie jak to było zrobione w galleries.component.ts z galeriami.

   Pamiętaj o dodaniu klienta http do konstruktora oraz dodaniu części z nagłówkami.

   Podpowiedź: Nie potrzebujesz pętli, bo mamy tylko jeden komentarz, wystarczy przesłać this.comment.

4. **Zadanie samodzielne**

   Podobnie jak wcześniej, pobierz wszystkie komentarze z bazy danych i wyświetl na stronie. Pamiętaj o dodaniu *galleryId* do ścieżki (w galeriach robiliśmy coś takiego przy usuwaniu galerii)

5. Usuwanie poszczególnych komentarzy – obok każdego z komentarzy dodaj przycisk/link do usunięcia – działający na onClick.

   Po kliknięciu powinien on usunąć konkretny komentarz z bazy i odświeżyć listę. Zrobimy to tak.

```
onDelete(commentId) {
    const index = this.comments.findIndex(item => item.commentId ===
commentId);

    this.http.post('http://project.usagi.pl/comment/delete/' +
commentId, {}, this.httpOptions).toPromise().then(() => {
        this.comments.splice(index, 1);
    }, (errResponse) => {
        console.log('error', errResponse);
    });
}
```

Uwaga: zwróć uwagę na zapis *this.comments.findIndex* – odpowiada on za przeszukanie tablicy z komentarzami i sprawdzenie, który element tablicy ma takie samo id co przesłane do funkcji. Jeśli je znajdzie, zwraca indeks elementu w tablicy.

Dzięki temu później możemy wykorzystać metodę *this.comments.splice()*, która usuwa z tablicy element o podanym indeksie.