Лабораторна робота 13.

Ionic Framework. Html-запити.

Розглянемо процедуру отримання даних з віддаленого веб-сервісу. Для цього скористаємось механізмом http-сервісів Angular.

Для тестування скористаємось сервісом http://jsonplaceholder.typicode.com. Так, наприклад, за адресою http://jsonplaceholder.typicode.com/posts/10 сервіс повертає публікацію (post) із id = 10 у json-форматі (рис. 1).

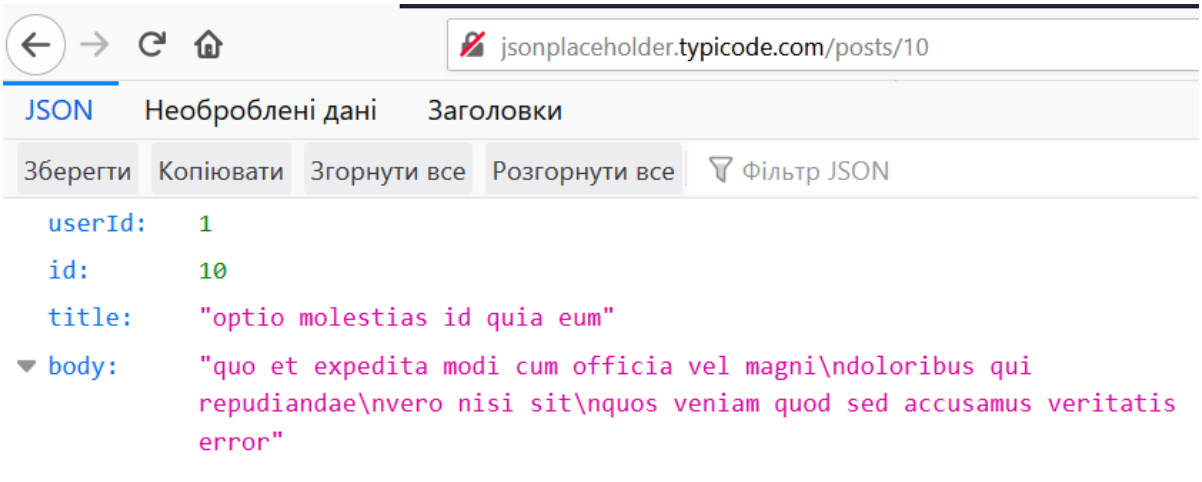

Рис. 1. Повернення публікації у json-форматі.

Тепер отримаємо наведені дані з нашого ionic-застосунку. Для цього створимо нову сторінку (рис. 2) та тимчасово зробимо її стартовою (рис. 3).

```
ionic generate page HttpTest
```

Рис. 2. Створення сторінки http-test.

```
5   const routes: Routes = [
6     { path: '', redirectTo: 'http-test', pathMatch: 'full' },
```

Рис. 3. Зміна стартової сторінки у app-routing.module.ts.

У файлі app.module.ts виконаємо підключення модуля HttpClientModule, що дозволяє використовувати http-сервіс Angular (рис. 4).

```
11  import {HttpClientModule} from '@angular/common/http';
12
13  @NgModule({
14    declarations: [AppComponent],
15    entryComponents: [],
```

```
  imports: [
      BrowserModule,
      IonicModule.forRoot(),
      AppRoutingModule,
      HttpClientModule
  ],
```

Рис. 4. Додавання модуля HttpClientModule до масиву imports.

У представлення нової сторінки додаємо кнопку, по натисканню на яку будемо виконувати http-запит на отримання даних (рис. 5).

```
<ion-header>
  <ion-toolbar>
    <ion-title>HttpTest</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-button color="primary" margin (click)="getData()">
    <ion-icon name="cloud-download" margin-end></ion-icon> Виконати http-get
  </ion-button>
</ion-content>
```

Рис. 5. Додавання кнопки на сторінку в http-test.page.html.

У класі http-test реалізуємо метод getData, що виконуватиме http-запит (рис. 6).

```
import { Component, OnInit } from '@angular/core';
import {HttpClient} from '@angular/common/http';

@Component({
  selector: 'app-http-test',
  templateUrl: './http-test.page.html',
  styleUrls: ['./http-test.page.scss'],
})
export class HttpTestPage implements OnInit {

  constructor(private http: HttpClient) { }

  ngOnInit() {
  }

  getData() {
    this.http.get( url: 'http://jsonplaceholder.typicode.com/posts/10')
        .subscribe(
        next: data => {
          console.log(data);
        }
    );
  }
}
```

Рис. 6. Клас http-test.page.ts.

Запустимо застосунок на виконаємо реалізований запит (рис. 7-8).

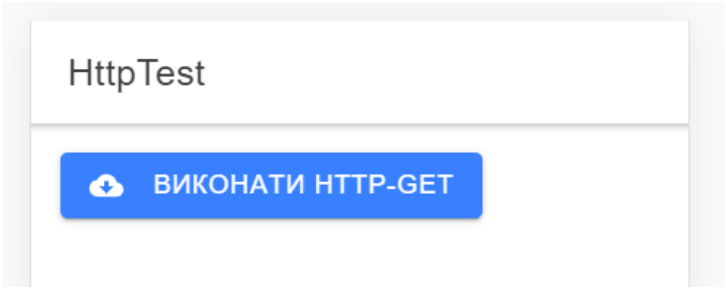

Рис. 7. Зовнішній вигляд сторінки http-test.

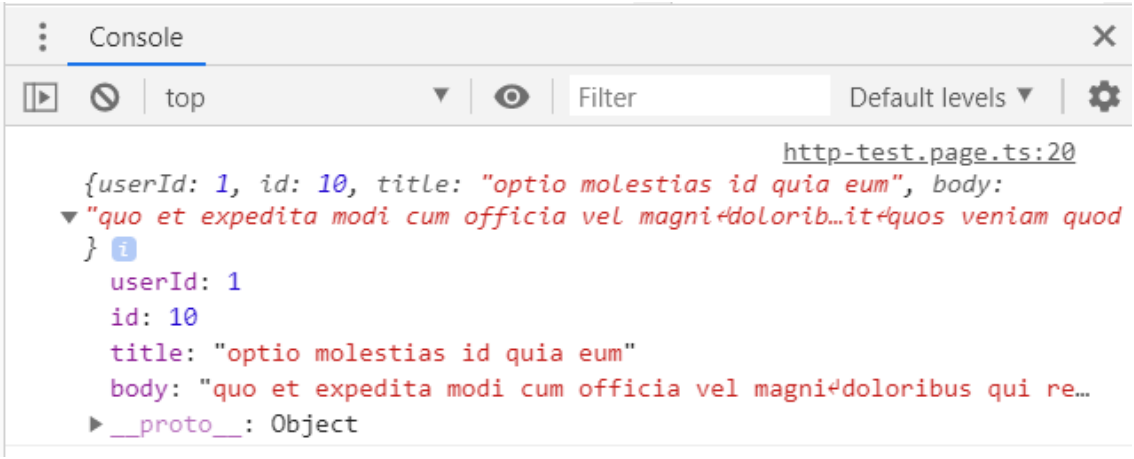

Рис. 8. Результат виведення результату запиту на консоль.

Виконаємо запит для отримання списку публікацій (http://jsonplaceholder.typicode.com/posts/) та розмістимо на сторінці перші 10 у вигляді списку (рис. 9 - 10).

```
1  import { Component, OnInit } from '@angular/core';
2  import {HttpClient} from '@angular/common/http';
3  import { map } from 'rxjs/operators';
4
5  @Component({
6    selector: 'app-http-test',
7    templateUrl: './http-test.page.html',
8    styleUrls: ['./http-test.page.scss'],
9  })
10 export class HttpTestPage implements OnInit {
11
12   posts: any[];
13
14   constructor(private http: HttpClient) { }
15
16   ngOnInit() {
17   }
18
```

```
  getData() {
    this.http.get( url: 'http://jsonplaceholder.typicode.com/posts')
        .pipe(map(
            project: (res: Array<any>) => res.filter(
                callbackfn: row => row.id < 10
            )
        )).subscribe(
            next: data => {
              this.posts = data;
            }
    );
  }
}
```

Рис. 9. Код файлу http-test.page.ts.

```
<ion-content>
  <ion-button color="primary" margin (click)="getData()">
    <ion-icon name="cloud-download" margin-end></ion-icon> Виконати http-get
  </ion-button>
  <ion-list>
    <ion-card *ngFor ="let post of posts">
      <ion-card-header>
        <ion-card-title>{{post.title}}</ion-card-title>
      </ion-card-header>
      <ion-card-content>
        {{post.body}}
      </ion-card-content>
    </ion-card>
  </ion-list>
</ion-content>
```

Рис. 10. Код файлу http-test.page.html.

Переглянемо отриманий результат (рис. 11).

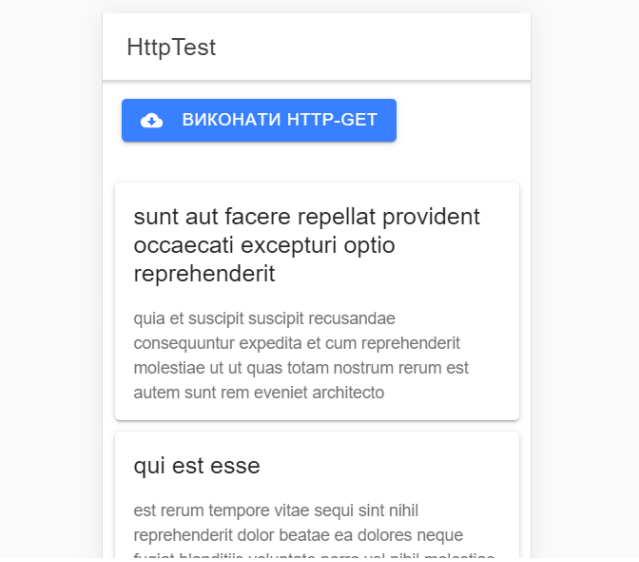

Рис. 11. Результат виведення списку публікацій.

Приберемо кнопку «Виконати http-get» та будемо завантажувати дані разом із завантаженням сторінки. Також реалізуємо механізм оновлення даних за

подією pull down (потягнути сторінку вниз). Для цього до представлення додаємо елемент управління refresher (рис.12) та виконуємо деякі зміни у коді класу HttpTest (рис. 13).

```
<ion-content>
  <ion-refresher slot="fixed" (ionRefresh)="getData($event)">
    <ion-refresher-content
            pullingIcon="arrow-dropdown"
            pullingText="Pull to refresh"
            refreshingSpinner="circles"
            refreshingText="Refreshing...">
    </ion-refresher-content>
  </ion-refresher>
  <ion-list>
    <ion-card *ngFor ="let post of posts">
```

Рис. 12. http-test.page.html.

```
ngOnInit() {
  this.getData( refresher: false);
}

getData(refresher) {
  this.http.get( url: 'http://jsonplaceholder.typicode.com/posts')
      .pipe(map(
          project: (res: Array<any>) => res.filter(
              callbackfn: row => row.id < 10
          )
      )).subscribe(
          next: data => {
            this.posts = data;
            if (refresher) {
              refresher.target.complete();
            }
          }
  );
}
}
```

Рис. 13. http-test.page.ts.

Переглянемо отриманий результат (рис. 14).

Рис. 14. Використання refresher для оновлення даних.

Виконаємо довантаження нових даних на сторінку по мірі їх необхідності. Тобто, під час прокручення вмісту сторінки вниз при досяганні кінця даних будемо завантажувати на сторінку нову порцію даних (10 наступних публікацій). Додаємо елемент управління InfiniteScroll (рис. 15).

```
<ion-content>
  <ion-refresher slot="fixed" (ionRefresh)="refreshData($event)">
    <ion-refresher-content
            pullingIcon="arrow-dropdown"
    </ion-card>
  </ion-list>
  <ion-infinite-scroll threshold="100px" (ionInfinite)="addData($event)">
    <ion-infinite-scroll-content
            loadingSpinner="bubbles"
            loadingText="Loading more data...">
    </ion-infinite-scroll-content>
  </ion-infinite-scroll>
</ion-content>
```

Рис. 15. Додавання елементу InfiniteScroll.

У класі HttpTest додаємо декілька змінних для фіксації кількості поточних даних. Також розділяємо окремо оновлення та додавання нових даних (рис. 16).

```
export class HttpTestPage implements OnInit {

  private posts: any[];

  private postCount = 0;
  private postStep = 10;

  constructor(private http: HttpClient) {
  }

  ngOnInit() {
    this.refreshData( refresher: false);
  }

  refreshData(refresher) {
    this.posts = [];
    this.postCount = 0;
    this.addData(refresher);
  }

```

```
30    addData(refresher) {
31      this.http.get( url: 'http://jsonplaceholder.typicode.com/posts')
32          .pipe(map(
33              project: (res: Array<any>) => res.filter(
34                  callbackfn: row => row.id > this.postCount &&
35                      row.id < this.postCount + this.postStep
36              )
37          )).subscribe(
38              next: data => {
39                this.posts = this.posts.concat(data);
40                if (refresher) {
41                  refresher.target.complete();
42                }
43                this.postCount += this.postStep;
44              }
45      );
46    }
47  }
```

Рис. 16. Зміни у коді http-test.page.ts.

Переглядаємо результат, завантажуємо застосунок та переходимо у кінець списку (рис. 17).

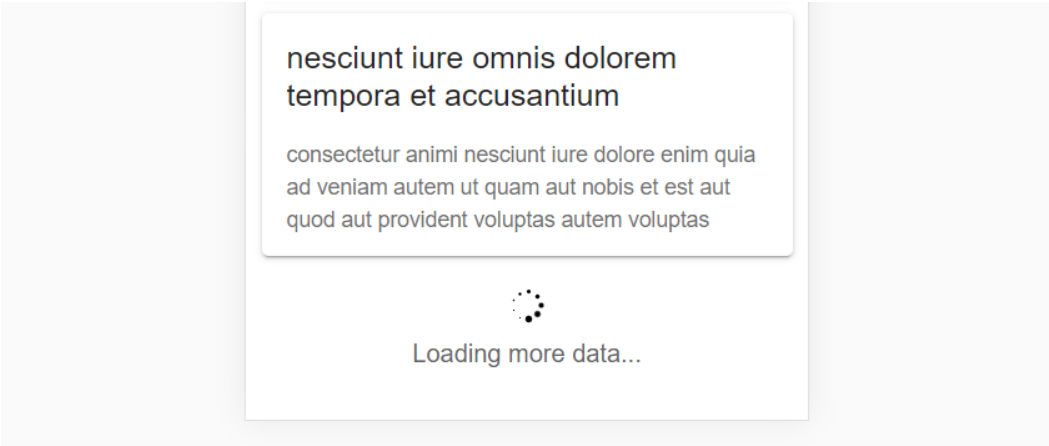

Рис. 17. Робота елементу InfiniteScroll.