Лабораторна робота 10.

Ionic Framework. Застосунок «список студентських груп»

Створимо новий ionic blank проект (рис.1)

```
> ionic start ion-stud-group blank
```

Рис.1. Розгортання нового проекту ionic blank.

Відкриємо проект у редакторі коду та додаємо до проекту сервіс, що буде відповідати за отримання даних (рис.2).

```
>>ionic g s service/data-getter
```

Рис. 2. Додавання сервісу data-getter.

Відкриваємо файл service/data-getter.service.ts та реалізуємо у ньому функціонал щодо отримання даних студентських груп (рис. 3). Тут, за допомогою інтерфейсу, визначимо тип даних «студентська група», а у класі DataGetterService – масив студентських груп та методи для отримання, додавання та видалення даних.

```
1  import { Injectable } from '@angular/core';
2  import {Observable, of} from 'rxjs';
3
4  export interface StudGroup {
5    number: number;
6    faculty: string;
7    specialty: string;
8    studentsQuantity: number;
9  }
10
11 @Injectable({
12   providedIn: 'root'
13 })
14 export class DataGetterService {
15   private groups: StudGroup[] = [
16     {
17       number: 201,
18       faculty: `факультет комп'ютерних наук`,
19       specialty: `комп'ютерні науки`,
20       studentsQuantity: 22
```

```
    },
    {
      number: 308,
      faculty: `факультет комп'ютерних наук`,
      specialty: `інженерія програмного забезпечення`,
      studentsQuantity: 24
    },
  ];

  constructor() { }

  getGroups(): Observable<StudGroup[]> {
    return of(this.groups);
  }

  addGroup(group: StudGroup) {
    this.groups.push(group);
  }

  deleteGroup(index) {
    this.groups.splice(index, deleteCount: 1);
  }
}
```

Рис. 3. Сервіс data-getter.

У класі HomePage у конструкторі підключаємо створений сервіс та використовуємо його для отримання масиву студентських груп у локальну змінну. Також декларуємо три метода для додавання, зміни та видалення даних (рис. 4).

```
import { Component } from '@angular/core';
import {DataGetterService, StudGroup} from '../service/data-getter.service';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {

  groups: StudGroup[];

  constructor(private dataGetter: DataGetterService) {
    this.dataGetter.getGroups().subscribe(
        next: (data) => {
          this.groups = data;
        }
    );
  }

```

```
21      add() {}
22
23      edit(group: StudGroup) {}
24
25      delete(group: StudGroup) {}
26  }
```

Рис. 4. Файл home.page.ts.

Тепер створимо представлення для виведення списку студентських груп на сторінці застосунку. Для цього виконаємо відповідні зміни у файлі home.page.html (рис. 5). При реалізації представлення, використаємо такі компоненти ionic-framework, як ion-list, ion-item, ion- item-sliding, ion-button, ion-icon та деякі інші. Із детальним оглядом цих та інших компонент Ionic framework можна ознайомитись на офіційному сайті проекту (https://ionicframework.com/docs/components).

```
1   <ion-header>
2     <ion-toolbar>
3       <ion-title>
4         Студентські групи
5       </ion-title>
6     </ion-toolbar>
7   </ion-header>
8
9   <ion-content>
10    <ion-list>
11      <ion-list-header>
12        <ion-label>Групи</ion-label>
13        <ion-button shape="round" fill="outline"
14                    color="success" (click)="add()">
15          <ion-icon slot="start" name="add"></ion-icon>Додати
16        </ion-button>
17      </ion-list-header>
18
19      <ion-item-sliding *ngFor="let group of groups">
20        <ion-item-options side="start">
21          <ion-item-option color="primary" (click)="edit(group)">
22            <ion-icon name="create"></ion-icon>
23            Змінити
24          </ion-item-option>
25          <ion-item-option color="danger" (click)="delete(group)">
26            <ion-icon name="trash"></ion-icon>
27            Видалити
28          </ion-item-option>
```

```
28          </ion-item-option>
29        </ion-item-options>
30        <ion-item>
31          <ion-icon name="people" slot="start"></ion-icon>
32          <ion-label>{{group.number}}</ion-label>
33          <ion-note slot="end">{{group.specialty}}</ion-note>
34        </ion-item>
35      </ion-item-sliding>
36    </ion-list>
37  </ion-content>
```

Рис. 5. Файл представлення home.page.html.

У файлі home.page.scss визначимо невеликий відступ справа для іконок у розділі ion-item-sliding (рис. 6).

```
1  ion-item-sliding {
2    ion-icon {
3      margin-right: 15px;
4    }
5  }
```

Рис. 6. Файл ion-item-sliding.

Запустимо застосунок та переглянемо отриманий результат (рис. 7).

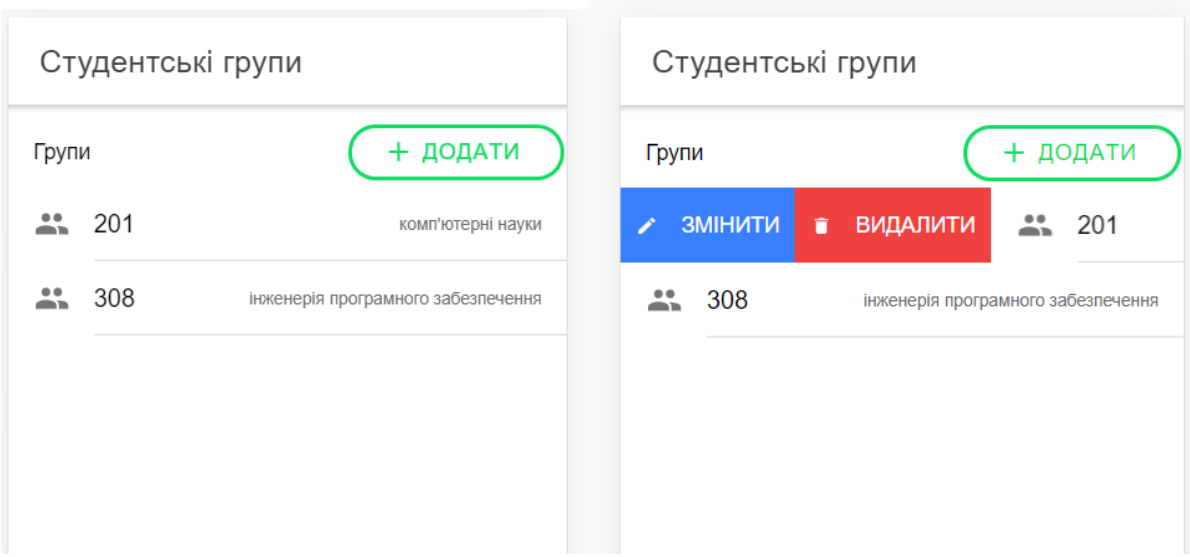

Рис. 7. Сторінка списку студентських груп.

Далі реалізуємо функціонал додавання, зміни та видалення студентських груп. Для цього створимо новий компонент, у що виводитиме детальні дані студентської групи та буде використовуватись для додавання нової або зміни існуючої групи (рис. 8).

```
>ionic g c components/stud-group
```

Рис. 8. Створення компоненти stud-group.

Відкриємо файл класу TypeScript та виконаємо реалізацію логіки компонента. (рис. 9).

```typescript
import {Component, EventEmitter, Input, OnInit, Output} from
 '@angular/core';
import {StudGroup} from '../../service/data-getter.service';

@Component({
  selector: 'app-stud-group',
  templateUrl: './stud-group.component.html',
  styleUrls: ['./stud-group.component.scss'],
})
export class StudGroupComponent implements OnInit {

  @Input() studentGroup: StudGroup;
  @Input() isNew: boolean;
  @Output() addGroup = new EventEmitter();
  @Output() cancelAddingGroup = new EventEmitter();
  title: string;

  constructor() {}

  ngOnInit() {
    if (this.isNew) {
      this.studentGroup = {
        number: null,
        specialty: '',
        studentsQuantity: null,
        faculty: ''
      };
      this.title = 'Нова група';
    }
  }

  addNew() {
    if (this.isNew) {
      this.addGroup.emit(this.studentGroup);
    }
  }

  cancelAdding() {
    if (this.isNew) {
      this.cancelAddingGroup.emit();
    }
  }
}
```

Рис. 9. Файл stud-group.component.ts.

Реалізуємо представлення компоненти, що представлятиме собою декілька полів вводу даних. Всі вони за допомогою механізму двостороннього зв’язування пов’язані із відповідними полями об’єкту group. Також для режиму додавання виводимо 2 кнопки (рис. 10).

```
<ion-card>
  <ion-card-header>
    <ion-card-title>{{title}}</ion-card-title>
  </ion-card-header>

  <ion-card-content>
    <ion-item>
      <ion-label position="floating">Номер</ion-label>
      <ion-input type="number" [(ngModel)]="studentGroup.number" name="number"></ion-input>
    </ion-item>

    <ion-item>
      <ion-label position="floating">Факультет</ion-label>
      <ion-input type="text" [(ngModel)]="studentGroup.faculty" name="faculty"></ion-input>
    </ion-item>

    <ion-item>
      <ion-label position="floating">Спеціальність</ion-label>
      <ion-input type="text" [(ngModel)]="studentGroup.specialty" name="specialty"></ion-input>
    </ion-item>

    <ion-item>
      <ion-label position="floating">Студентів</ion-label>
      <ion-input type="number" [(ngModel)]="studentGroup.studentsQuantity" name="studentQuantity"></ion-input>
    </ion-item>
    <ion-button *ngIf="isNew" color="primary" (click)="addNew()">
      <ion-icon slot="start" name="add-circle"></ion-icon> Додати
    </ion-button>

    <ion-button *ngIf="isNew" color="danger"  class="ion-float-right" (click)="cancelAdding()">
      <ion-icon slot="start" name="close"></ion-icon> Відміна
    </ion-button>
  </ion-card-content>
</ion-card>
```

Рис. 10. Файл stud-group.component.html.

Використаємо наш новостворений компонент на сторінці home. Почнімо із змін у представленні home.page.html, де окрім додавання нової компоненти для кожного рядку у циклі, також реалізуємо події, за якими буде виконуватись відображення та приховання компоненти. Далі наводимо повний код представлення сторінки home, після виконання необхідних змін (рис. 11).

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Студентські групи
    </ion-title>
  </ion-toolbar>
</ion-header>

```

```
<ion-content>
  <ion-list>
    <ion-list-header>
      <ion-label>Групи</ion-label>
      <ion-button shape="round" fill="outline"
                  color="success" (click)="add()">
        <ion-icon slot="start" name="add"></ion-icon>Додати
      </ion-button>
    </ion-list-header>

    <app-stud-group *ngIf="showNew" [isNew]="true"
                    (addGroup)="addGroup($event)"
                    (cancelAddingGroup)="showNew=false"></app-stud-group>

    <div *ngFor="let group of groups;let i = index">
      <ion-item-sliding>
        <ion-item-options side="start">
          <ion-item-option color="primary" (click)="showEdit=i">
            <ion-icon name="create"></ion-icon>
            Змінити
          </ion-item-option>
          <ion-item-option color="danger" (click)="delete(i)">
            <ion-icon name="trash"></ion-icon>
            Видалити
          </ion-item-option>
        </ion-item-options>
        <ion-item (click)="showEdit=-1">
          <ion-icon name="people" slot="start"></ion-icon>
          <ion-label>{{group.number}}</ion-label>
          <ion-note slot="end">{{group.specialty}}</ion-note>
        </ion-item>
      </ion-item-sliding>
      <app-stud-group *ngIf="showEdit==i" [isNew]="false"
                      [studentGroup]="group"></app-stud-group>
    </div>
  </ion-list>
</ion-content>
```

Рис. 11. Представлення home.page.html.

У класі HomePage додаємо поля, що відповідатимуть за відображення полів вводу, та реалізуємо методи додавання та видалення даних (рис. 12).

```
import { Component } from '@angular/core';
import {DataGetterService, StudGroup} from '../service/data-getter.service';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
```

```
export class HomePage {

  groups: StudGroup[];

  showNew = false;
  showEdit = -1;

  constructor(private dataGetter: DataGetterService) {
    this.dataGetter.getGroups().subscribe(
        next: (data) => {
          this.groups = data;
        }
    );
  }

  add() {
    this.showNew = true;
  }

  delete(index: number) {
      this.dataGetter.deleteGroup(index)
  }

  addGroup(group) {
    this.dataGetter.addGroup(group);
    this.showNew = false;
  }
}
```

Рис. 12. Клас HomePage.

Перевіряємо роботу додавання нової групу, змінюємо дані, та видаляємо (рис. 13).

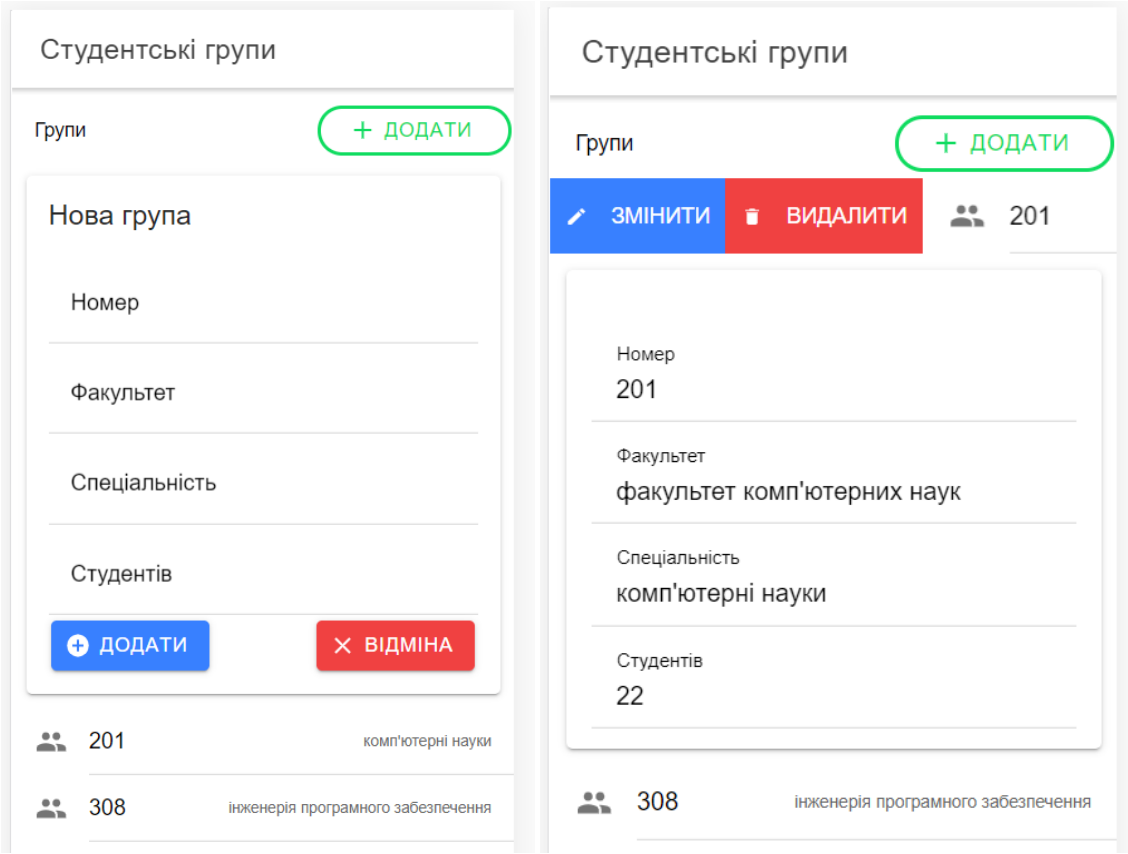

Рис. 13. Додавання та зміна елементу.

Завдання для самостійного виконання.

На базі ionic framework реалізувати застосунок, що виконує виведення та подальше редагування списку сутностей згідно варіанту індивідуального завдання.