Лабораторна робота 16.

Використання Firebase для збереження даних застосунку. Частина 1.

Backend as a Service (BaaS) — модель, позволяющая разработчикам веб и мобильных приложений связать их приложения с серверным облачным хранилищем и API, а также предоставляющая такие функции, как управление пользователями, извещающие уведомления, интеграция со службами социальных сетей. Поэтому другое название термина — mBaaS, мобильный бэкенд как услуга. Перечисленные выше услуги предоставляются посредством использования интерфейсов прикладного программирования (API).

Реализация backend API – времязатратный и трудоёмкий для разработчиков процесс. Идея BaaS в том, что вместо того, чтобы самим разрабатывать и в дальнейшем поддерживать серверные сервисы, можно воспользоваться готовыми, набор которых вместе формируют необходимый универсальный кросс-платформенный бэкенд для любого проекта. Соответственно, не нужно взаимодействовать с сервером приложения, базой данных, клиент-серверной библиотекой, хостингом, необязательно писать административную панель, продумывать дизайн своего API.

Среди наилучших и наиболее популярных BaaSs можно выделить Firebase от компании Google. Firebase служит базой данных, которая изменяется в реальном времени и хранит данные в JSON. Любые изменения в базе данных тут же синхронизируются между всеми клиентами, или девайсами, которые используют одну и ту же базу данных. Вместе с хранилищем, Firebase также предоставляет пользовательскую аутентификацию, и поэтому все данные передаются через защищенное соединение SSL. Мы можем выбрать любую комбинацию email и пароля для аутентификации, будь то Facebook, Twitter, GitHub, Google, или что-то другое.

Для використання Firebase для збереження даних нашого проекту відкриємо сайт ресурсу https://firebase.google.com, перейдемо у консоль (рис. 1) та ствримо новий проект (рис. 2).

Рис. 1. Офіційний сайт Firebase.

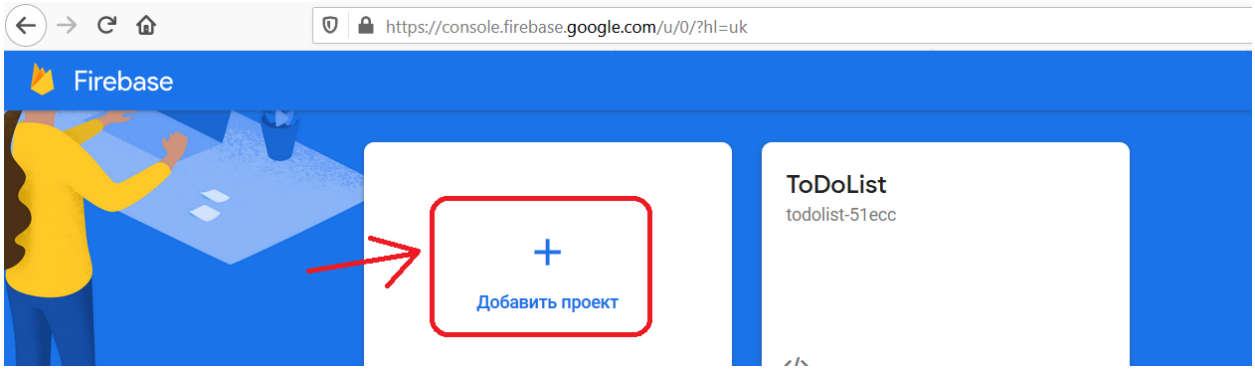

Рис. 2. Створення нового проекту.

Вказуємо назву проекту (рис. 3)

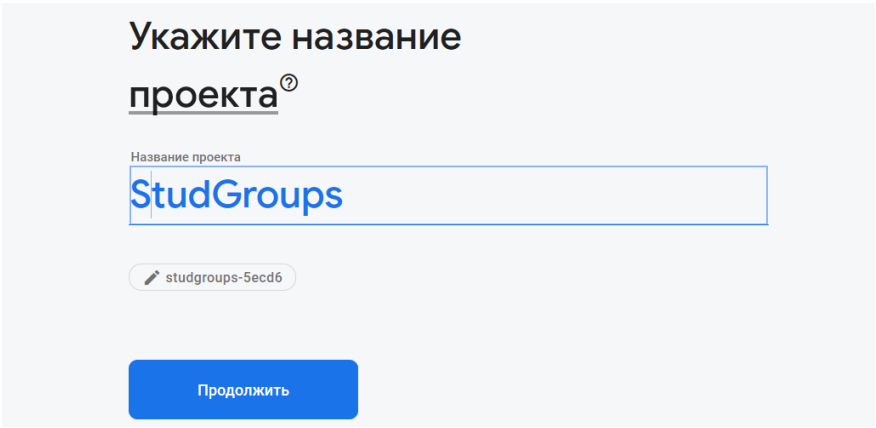

Рис. 3. Уведення назви проекту.

Після створення проекту необхідно додати до нього застосунок, що буде його використовувати, обравши необхідний тип – iOS, Android або Web. Для гібридного застосунку ionic на базі angular обираємо web-застосунок (рис. 4).

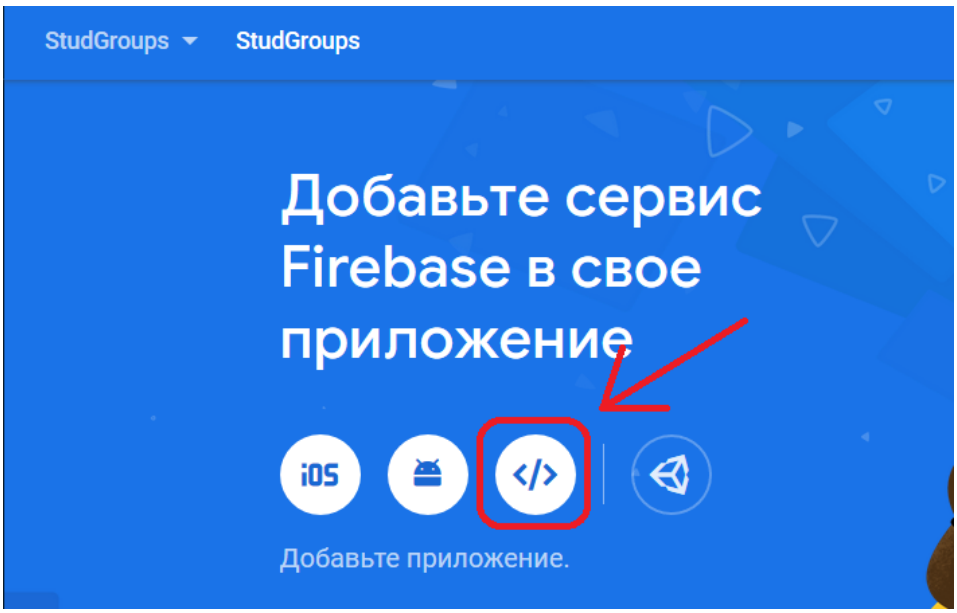

Рис. 4. Додавання застосунку до проекту.

Вводимо псевдонім застосунку та натискаємо «зареєструвати застосунок» (рис. 5).

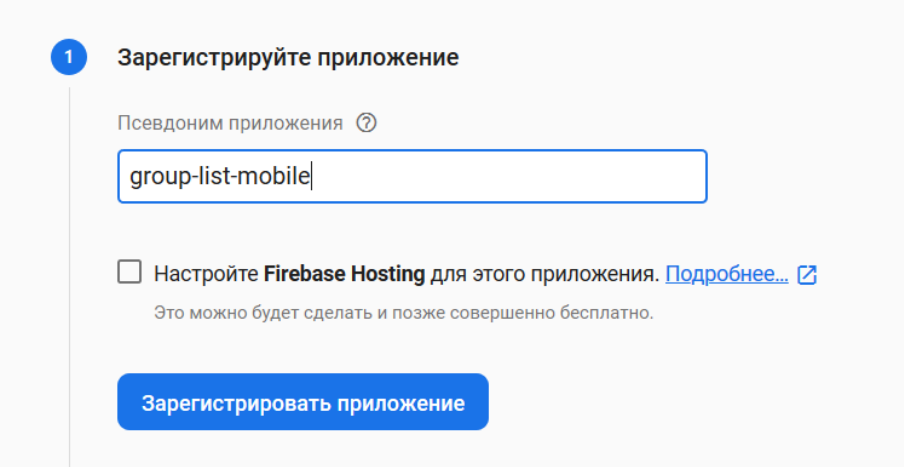

Рис. 5. Реєстрація застосунку.

Із наступного вікна копіюємо поля об’єкту firebaseConfig – вони нам знадобляться у майбутньому при налаштуванні підключення до firebase із застосунку «список студентських груп» (рис. 6).

Рис. 6. Дані, необхідні на налаштування підключення до firebase.

Далі у нашому проекті firebase створимо базу даних для збереження списку студентських груп (рис. 7-8).

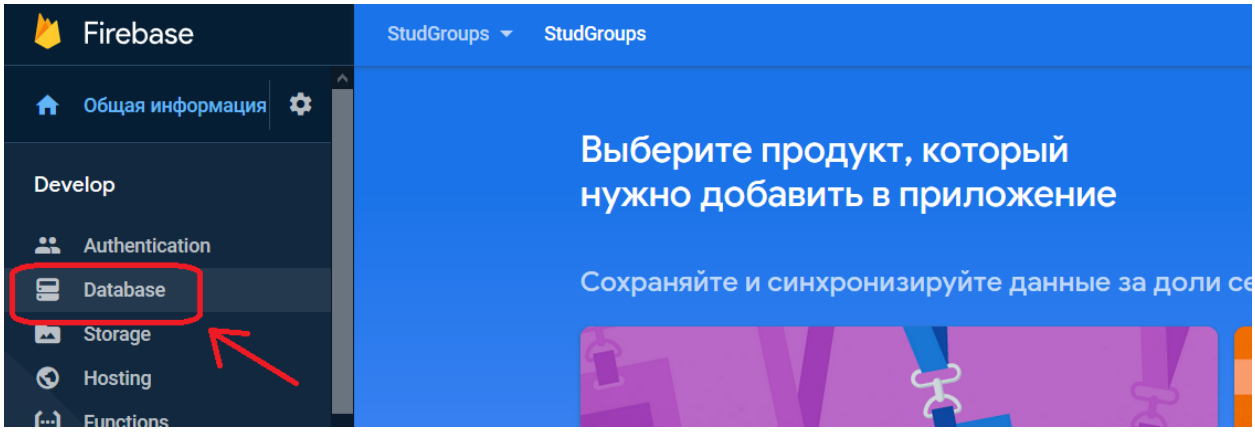

Рис. 7. Додавання бази даних.

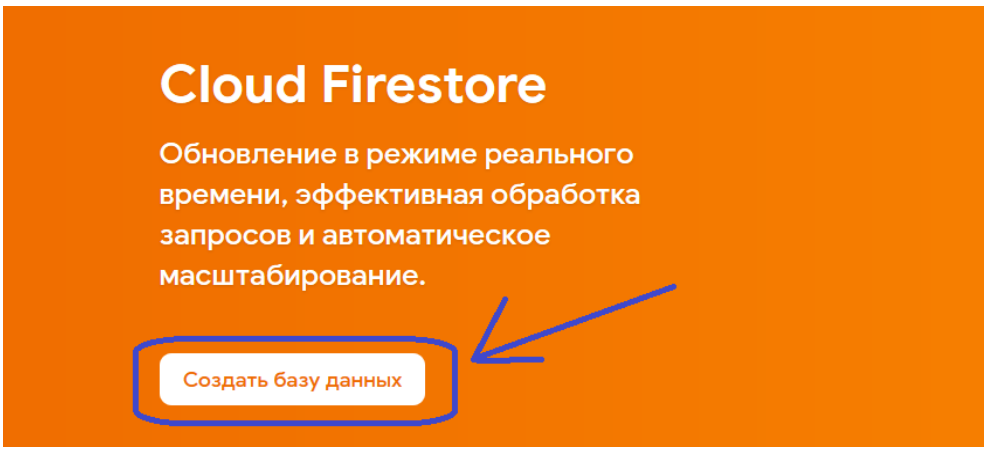

Рис. 8. Створення бази даних.

На першому етапі для можливості роботи із створеною БД без проходження етапу аутентифікації, обираємо «запустити у тестовому режимі» (рис. 9).

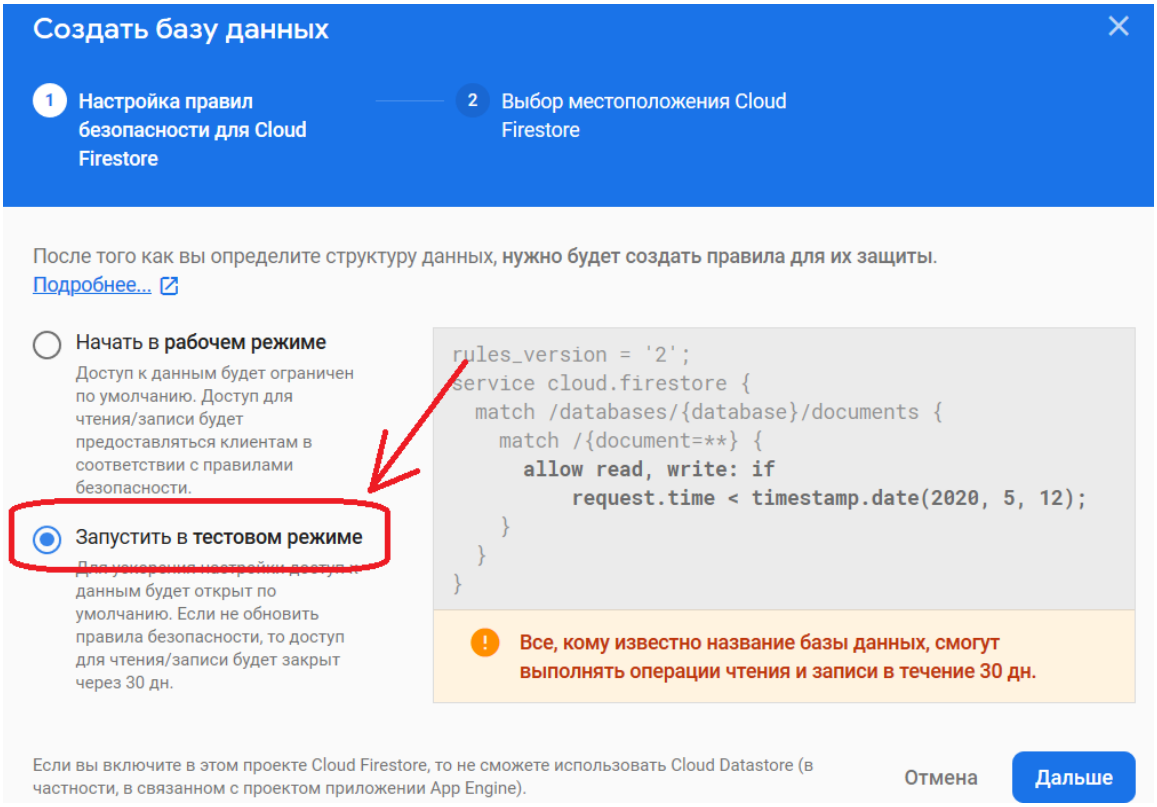

Рис. 9. Запуск у тестовому режимі.

Перейдемо до створеної бази даних та додамо нову колекцію groups, що ідентично таблиці реляційної БД (рис. 10-11).

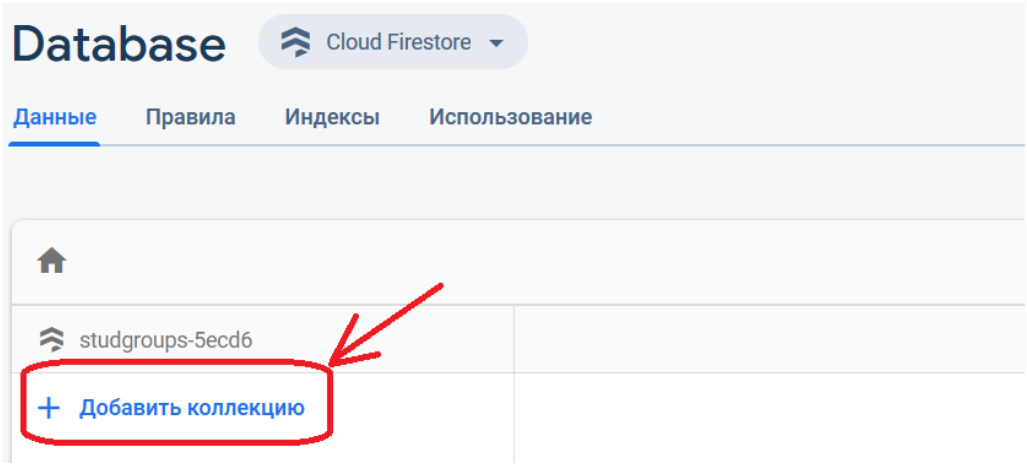

Рис. 10. Додавання нової колекції groups.

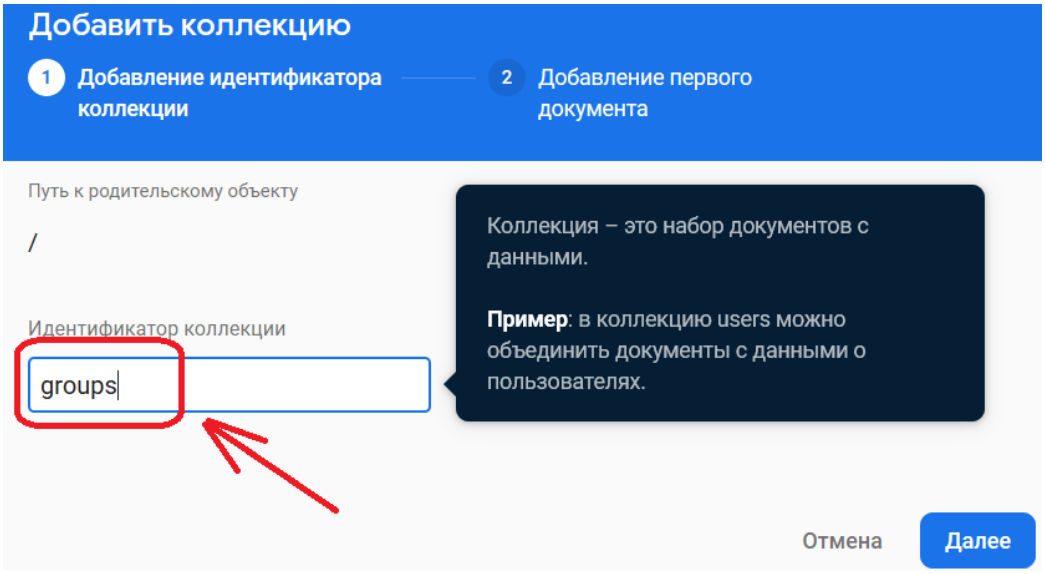

Рис. 11. Введення ідентифікатору колекції.

Далі буде запропоновано ввести новий документ колекції (рядок таблиці). Зробимо це (рис. 12).

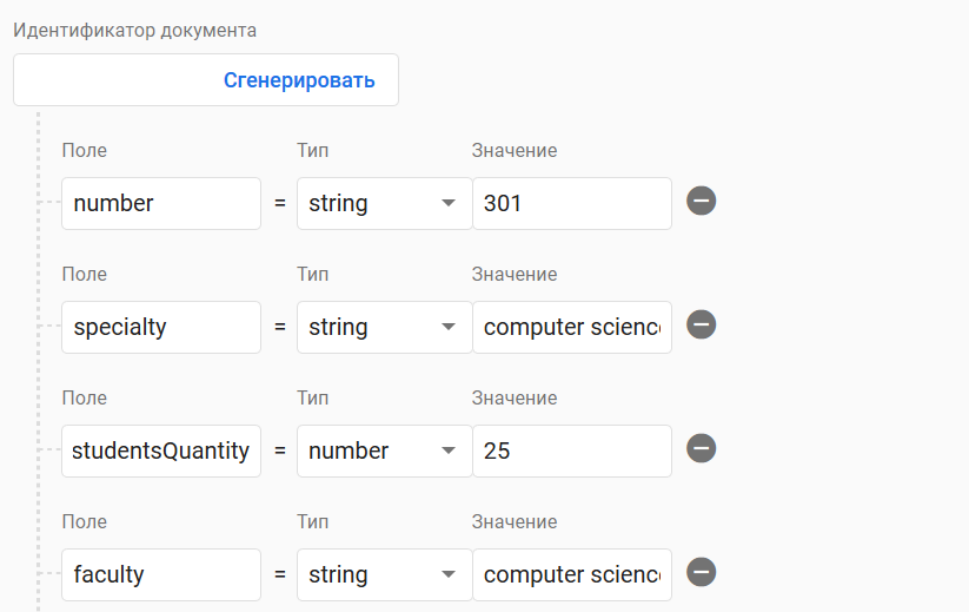

Рис. 12. Додавання нового документу.

Після виконання вищенаведених операцій маємо наступний вигляд БД (рис. 13).

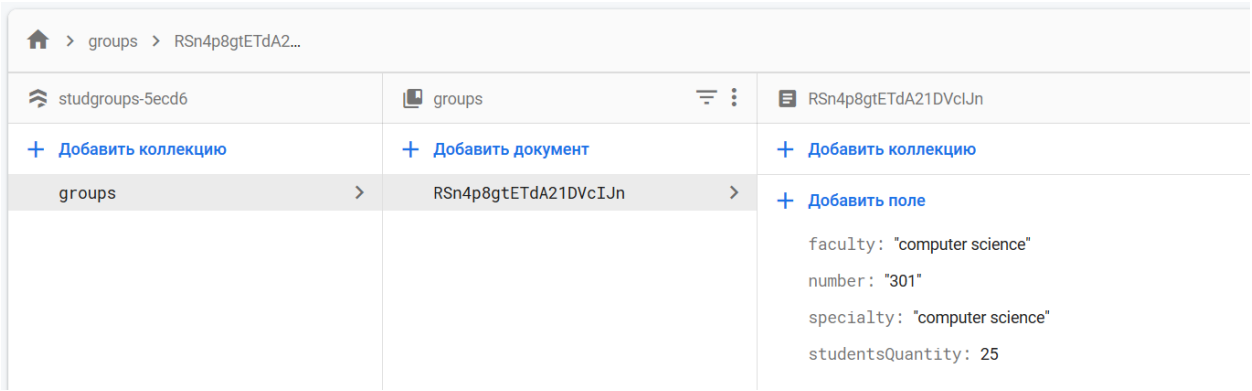

Рис. 13. Стан БД після додавання колекції та документу.

Додаємо до колекції ще хоча б один, а краще декілька документів (рис. 14).

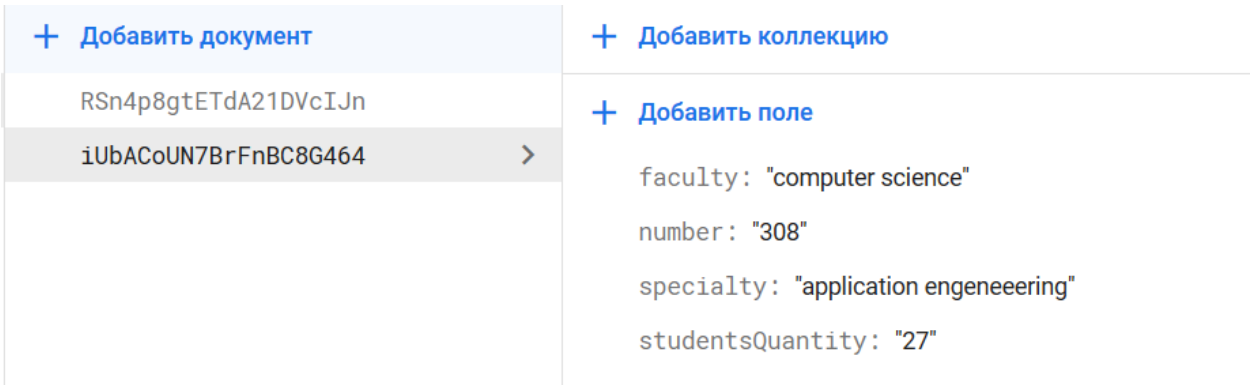

Рис. 14. Додавання документу до колекції.

Далі переходимо до реалізації підключення до бази даних firebase із нашого ionic-застосунку. Починаємо із додавання до проекту ionic бібліотеки angular firebase. Для цього, перебуваючи у директорії проекту, у командному рядку виконуємо «npm install @angular/fire firebase --save» (рис. 15).

```
›npm install @angular/fire firebase --save
```

Рис. 15. Встановлення angular firebase.

Після завершення встановлення, перейдемо до файлу src/environments/environment.ts та вставимо скопійований на рис. 6 код, привевши його до наступного вигляду (рис. 16).

```
export const environment = {
  production: false,
  firebase: {
    apiKey: 'AIzaSyA7gVADz1EkjpZx368NEfGniUNjpqdrOao',
    authDomain: 'studgroups-5ecd6.firebaseapp.com',
    databaseURL: 'https://studgroups-5ecd6.firebaseio.com',
    projectId: 'studgroups-5ecd6',
    storageBucket: 'studgroups-5ecd6.appspot.com',
    messagingSenderId: '12724443661',
    appId: '1:12724443661:web:6393dc1131fee328931344'
  }
};
```

Рис. 16. Налаштування підключення до firebase.

У файлі \src\app\app.module.ts виконаємо необхідні підключення для подальшої роботи із angular firebase (рис. 17).

```
import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';
import {HttpClientModule} from '@angular/common/http';
import {AngularFireModule} from '@angular/fire';
import {environment} from '../environments/environment';
import {AngularFireStorageModule} from '@angular/fire/storage';
import {AngularFirestore} from '@angular/fire/firestore';

@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [
      BrowserModule,
      IonicModule.forRoot(),
      AppRoutingModule,
      HttpClientModule,
      AngularFireModule.initializeApp(environment.firebase),
      AngularFireStorageModule
  ],
```

```
  providers: [
    StatusBar,
    SplashScreen,
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy },
    AngularFirestore
  ],
```

Рис. 17. Зміни у app.module.ts.

Також виконаємо невеликі доповнення до файлу tsconfig.json у розділі compilerOptions (рис. 18).

```
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "sourceMap": true,
    "declaration": false,
    "module": "esnext",
    "moduleResolution": "node",
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "importHelpers": true,
    "target": "es2015",
    "typeRoots": [
      "node_modules/@types"
    ],
    "lib": [
      "es2018",
      "dom"
    ],
    "skipLibCheck": true
  },
```

Рис. 18. Зміни у файлі tsconfig.json.

Створимо сервіс fire-data-getter для роботи із даними FireStore (рис. 18), що повертатиме список студентських груп через метод getGroups (рис. 19).

```
>>ionic g service services/fire-data-getter
```

Рис. 18. Створення сервісу fire-data-getter.

```
import { Injectable } from '@angular/core';
import {AngularFirestore} from '@angular/fire/firestore';
import {Observable} from 'rxjs';
import {map} from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})

```

```
export class FireDataGetterService {

  groups: Observable<any[]>;

  constructor(private readonly afs: AngularFirestore) {
    const groupsCollection = afs.collection( path: 'groups');
    this.groups = groupsCollection.snapshotChanges().pipe(
        map( project: actions => actions.map( callbackfn: a => {
```

```
export class FireDataGetterService {

  groups: Observable<any[]>;

  constructor(private readonly afs: AngularFirestore) {
    const groupsCollection = afs.collection( path: 'groups');
    this.groups = groupsCollection.snapshotChanges().pipe(
        map( project: actions => actions.map( callbackfn: a => {
          const data = a.payload.doc.data();
          const id = a.payload.doc.id;
          return { id, ...data };
        }))
    );
  }

  getGroups() {
    return this.groups;
  }
}
```

Рис. 19. Код сервісу fire-data-getter.

Перевіримо роботу сервісу, підключивши його до сторінки логіну та вивівши на консоль список студентських груп (рис. 20).

```
  constructor(
      private router: Router,
      private dataGetter: DataGetterService,
      public alertController: AlertController,
      private fireDataGetter: FireDataGetterService) {}

  ngOnInit() {
    this.fireDataGetter.getGroups().subscribe(
        next: data => console.log(data)
    );
  }
```

Рис. 20. Зміни у \src\app\login\login.page.ts.

Запустимо наш застосунок та перевіримо вивід даних (рис. 21).

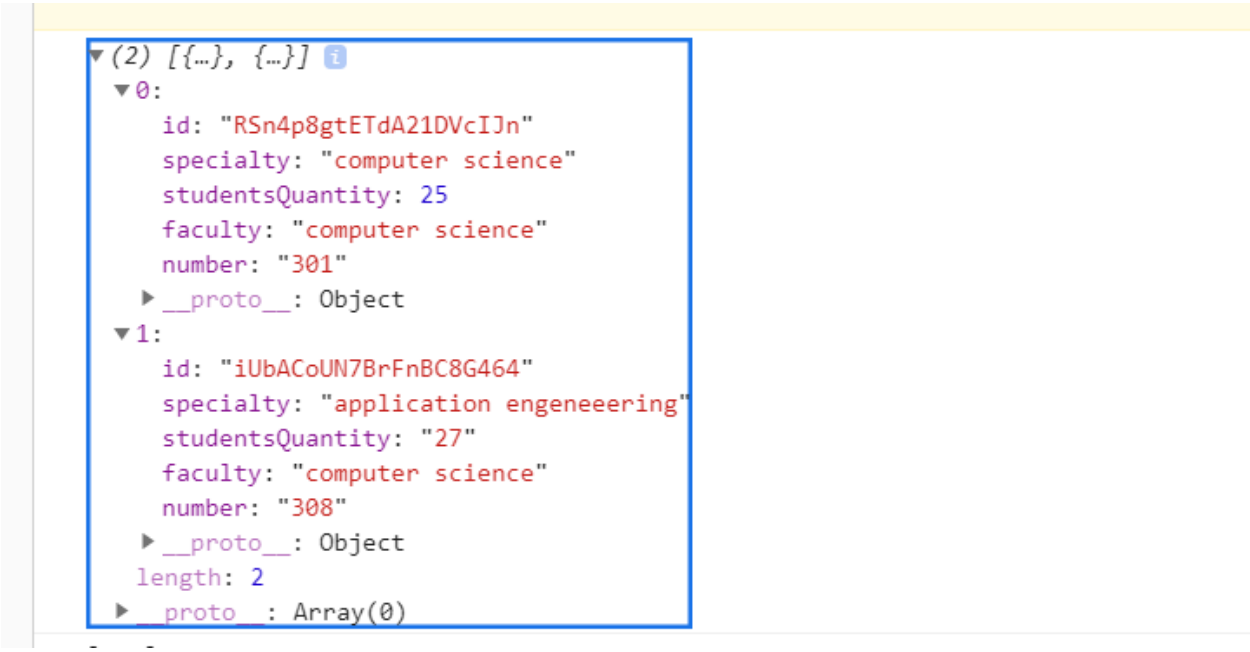

Рис. 21. Перевірка виводу списку груп на консоль.

Виконаємо виведення отриманий із FireStore список студентських груп на сторінці home нашого застосунку. Збереження та роботу із користувачами з боку firebase ми поки не налаштовували, тому відключимо також тимчасово аутентифікацію і з боку клієнтського застосунку. Для цього змінимо метод onInit на сторінці login (рис. 22).

```
16    constructor(
17        private router: Router,
18        private dataGetter: DataGetterService,
19        public alertController: AlertController) {}
20
21    ngOnInit() {
22      this.dataGetter.setUser('FakeUser');
23      this.router.navigate( commands: ['/home']);
24    }
```

Рис. 22. Зміни у login.page.ts.

Тепер виконаємо підключення до сервісу та отримання даних у home.page.ts (рис. 23).

```
5  import {FireDataGetterService} from '../services/fire-data-getter.service';
6
```

```
23    constructor(private dataGetter: DataGetterService,
24                private router: Router,
25                private dataExchanger: DataExchangerService,
26                private fireData: FireDataGetterService) {
27      this.fireData.getGroups().subscribe(
28          next: data => this.groups = data
29      );
30      this.userName = this.dataGetter.getUser();
31    }
```

Рис. 23. Зміни у home.page.ts.

Запускаємо застосунок та перевіряємо відображення списку студентських груп на сторінці (рис. 24).

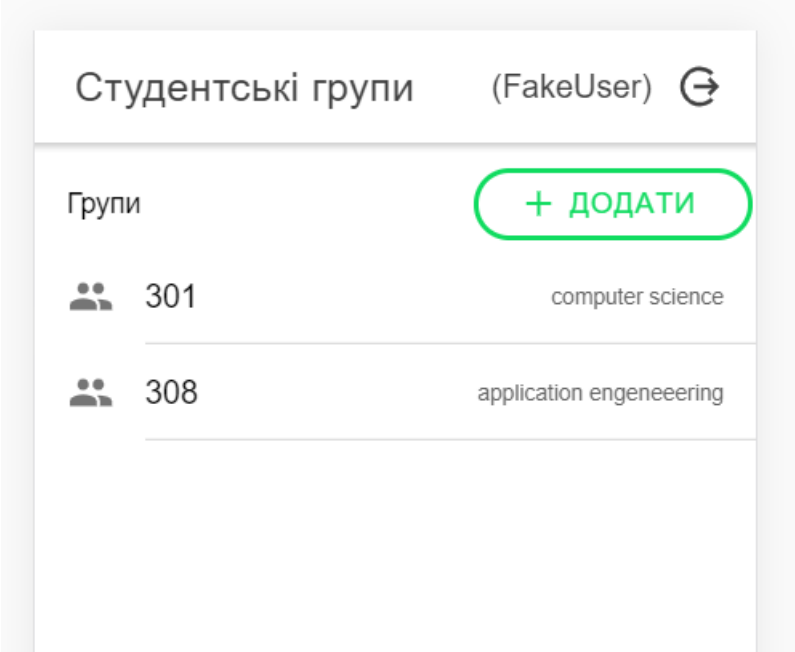

Рис. 24. Виведення списку студентських груп.