

Лабораторна робота 18.

Використання Firebase для збереження даних застосунку.

Частина 3.

Тепер виконаємо у нашому застосунку реалізації аутентифікації користувачів. Запустимо консоль firebase, відкриємо наш проект, перейдемо на вкладку Authentication та натискаємо «Налаштувати засіб входу» (рис. 1).

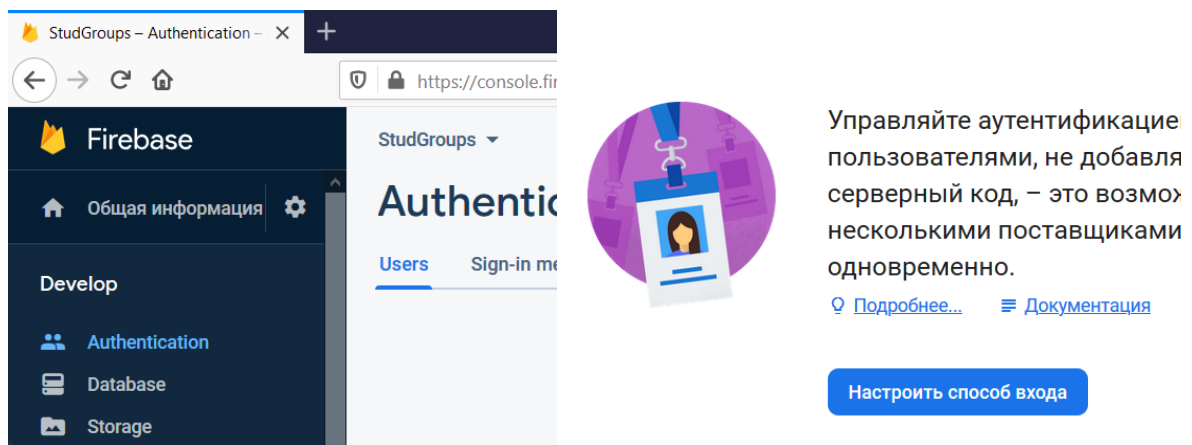
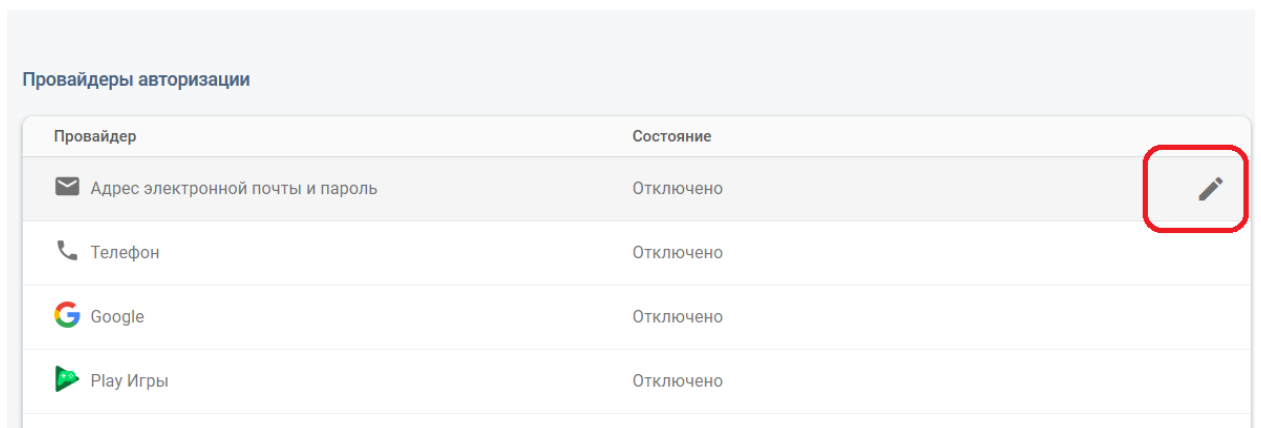


Рис. 1. Налаштування засобу входу у проекті firebase.

Обираємо провайдера авторизації «Адреса електронної пошти і пароль» (рис. 2).



☒ Включить

Вход в приложение по адресу электронной почты. В SDK доступны примитивы для проверки и смены адреса, а также восстановления пароля. [Подробнее...](#)

☐ Включить

Ссылка по электронной почте (вход без пароля)

Отмена
Сохранить

Рис. 2. Вибір провайдеру авторизації.

Перейдемо на вкладку Users та додаємо нового користувача (рис. 3).

Users
Sign-in method
Templates
Usage

Добавить пользователя

Идентификатор	Поставщики	Время создания	Последний вход	Уникальный идентификатор пользователя ↑
<div style="display: flex; justify-content: space-between;"> <div> <p>Добавьте пользователя</p> <div style="display: flex; justify-content: space-between; margin-bottom: 10px;"> Уведомлять по электронной почте Пароль </div> <div style="display: flex; justify-content: space-between;"> <input style="width: 45%; border: 1px solid #ccc; padding: 5px;" type="text" value="l.m.vogelstarku@gmail.com"/> <input style="width: 45%; border: 1px solid #ccc; padding: 5px;" type="password" value="123456"/> </div> <div style="text-align: right; margin-top: 10px;"> Отмена Добавить пользователя </div> </div> </div>				

Рис. 3. Додавання нового користувача.

Переходимо до сервісу fire-data-getter у нашому клієнтському застосунку та реалізуємо функціонал аутентифікації користувача (рис. 4).

```

5  import {AngularFireAuth} from '@angular/fire/auth';
6
11 export class FireDataGetterService {
12
13     groups: Observable<any[]>;
14     private userName = '';
15
16     constructor(private readonly afs: AngularFirestore,
17                 private afAuth: AngularFireAuth) {

```

```

27
28   checkUser(user) {
29       return this.afAuth.signInWithEmailAndPassword(
30           user.username,
31           user.passwd
32       );
33   }

34
35   getUser() {
36       return this.userName;
37   }
38
39   setUser(name: string) {
40       this.userName = name;
41   }

```

Рис. 4. Зміни у fire-data-getter.service.ts.

Також у app.module.ts додаємо до масиву провайдерів AngularFireAuth (рис. 5).

```

16   import {AngularFireAuth} from '@angular/fire/auth';
17
29   providers: [
30       StatusBar,
31       SplashScreen,
32       { provide: RouteReuseStrategy, useClass: IonicRouteStrategy },
33       AngularFireStore, AngularFireAuth
34   ],

```

Рис. 5. Зміни у app.module.ts.

Використаємо створений метод checkUser на сторінці login.page.ts. Також приберемо код, що відразу перенаправляв нас на сторінку home.page.ts (рис. 6).

```

5   import {FireDataGetterService} from '../services/fire-data-getter.service';
6
15
16   constructor(
17       private router: Router,
18       private dataGetter: DataGetterService,
19       public alertController: AlertController,
20       private fireData: FireDataGetterService) {}
21
22   ngOnInit() {}

23
24   login() {
25       this.fireData.checkUser( user: {
26           username: this.userName,
27           passwd: this.passWord
28       }).then(

```

```

29      onfulfilled: res => {
30        this.fireData.setUser(this.userName);
31        this.router.navigate(commands: ['/home']);
32      },
33      onrejected: err => {
34        this.userNotExistAlert(err.message);
35      }
36    );
37  }

```

Рис. 6. Зміни у login.page.ts.

Змінимо код src/app/guards/auth.guard.ts, щоб отримувати дані про успішне проходження аутентифікації із сервісу fire-data-getter (рис. 7).

```

4  import {FireDataGetterService} from '../services/fire-data-getter.service';
5
10  constructor(private dataGetter: DataGetterService,
11              private router: Router,
12              private fireData: FireDataGetterService) {}
13
14  canActivate(
15    next: ActivatedRouteSnapshot,
16    state: RouterStateSnapshot): boolean {
17
18    const isLoggenIn = this.fireData.getUser() !== '';
19    if (!isLoggenIn) {
20      this.router.navigateByUrl(url: '/login');
21    }
22    return isLoggenIn;
23  }

```

Рис. 7. Зміни guard-y.

І відкоригуємо home.page.ts для виведення імені користувача у верхній частині сторінки (рис. 8).

```

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.fireData.getUser();
31  }

```

Рис. 8. Отримання імені користувача в home.page.ts.

Виконаємо вхід до нашого застосунку та перевіримо його роботу (рис. 9).

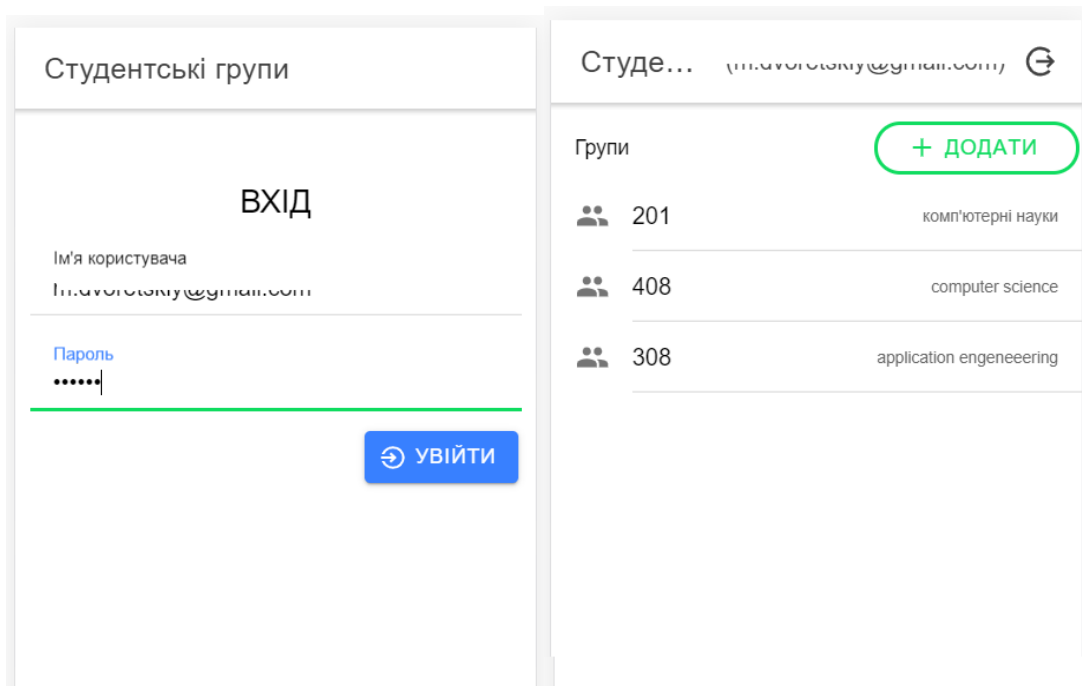


Рис. 9. Перевірка роботи аутентифікації після виконаних змін.

Також виконаємо декілька невдалих спроб авторизації, та переглянемо статуси, що повертає firebase (рис. 10).

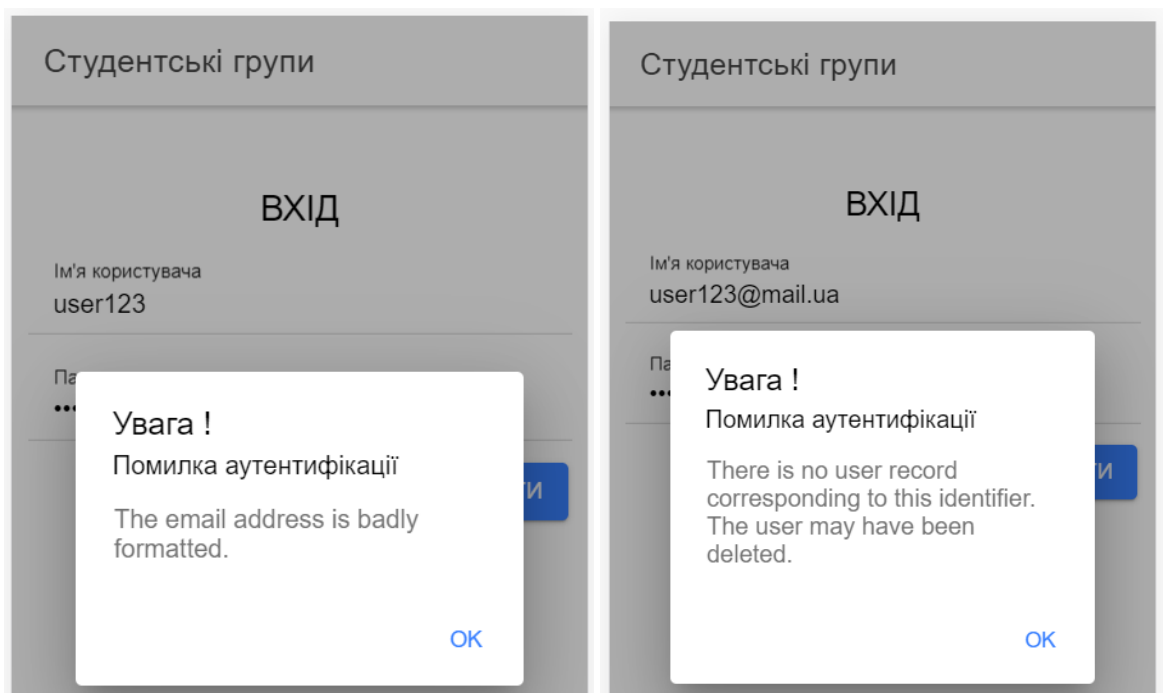


Рис. 10. Спроби невдалої авторизації.

Отже ми реалізували клієнтську аутентифікацію, але станом на зараз залишається можливість доступу до даних напряму неаутентифікованих користувачів. Переконаємось у цьому – додамо у подію onInit сторінки login

операцію отримання списку студентських груп (рис. 11). Звернімо увагу, що дана подія виникає до авторизації.

```
21
22  ngOnInit() {
23      this.firestore.getGroups().subscribe(
24          next: data => console.log(data)
25      );
26  }
```

Рис. 11. Додавання запиту на отримання даних.

Запустимо застосунок та переконаємось, що дані отримано (рис. 12).

```
▼ (3) [{...}, {...}, {...}] ⓘ
  ▼ 0:
    id: "DUiHg2pdk3qNc16Z3MYc"
    number: 201
    specialty: "комп'ютерні науки"
    studentsQuantity: 29
    faculty: "Комп.наук"
    ► __proto__: Object
  ► 1: {id: "RSn4p8gtETdA21DVcIJn", specialty: "computer science", studentsQuantity: 25, fa
  ► 2: {id: "iUbACoUN7BrFnBC8G464", specialty: "application engineeering", studentsQuantity
    length: 3
  ► __proto__: Array(0)
```

Рис. 12. Отримання даних без авторизації.

Для усунення даного недоліку необхідно додати перевірку прав доступу до даних на сервері. Для цього переходимо у консоль firebase на вкладку database rules (правила) та замінимо існуюче правило наступним кодом (рис. 13), що надаватиме право на читання даних тільки аутентифікованим користувачам.

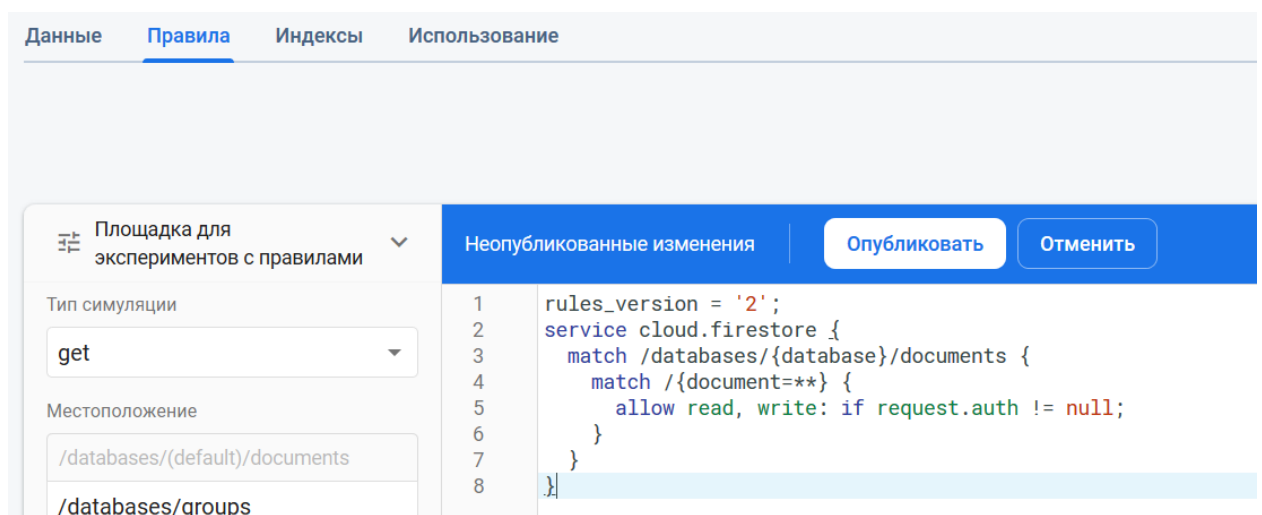


Рис. 13. Створення правила для контролю доступу до даних.

Перед застосуванням змін запусимо симуляцію для варіанту без аутентифікації (рис. 14) та з аутентифікацією (рис. 15).

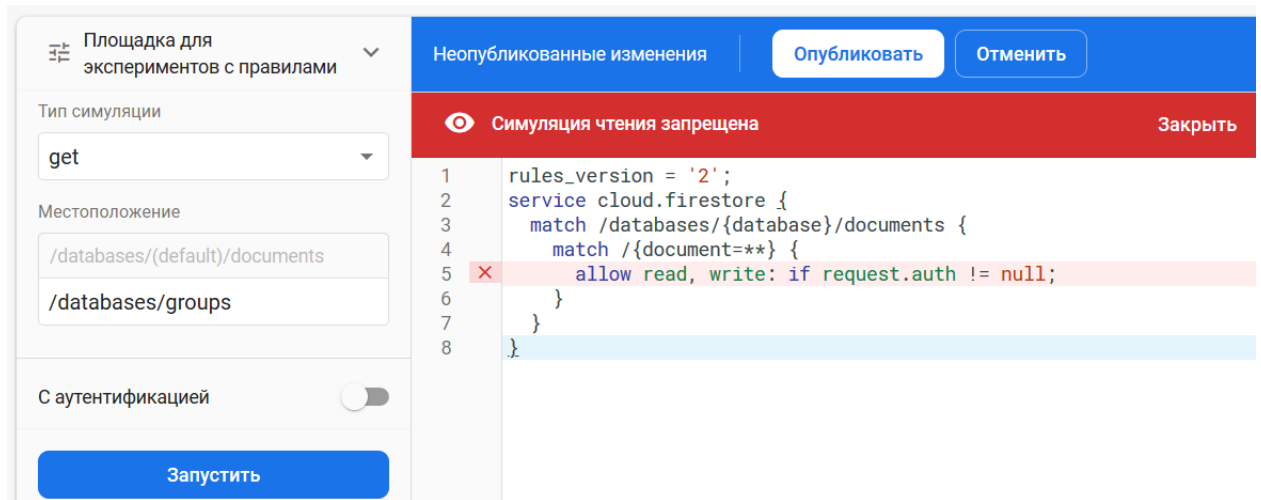


Рис. 14. Симуляція доступу до даних без аутентифікації.

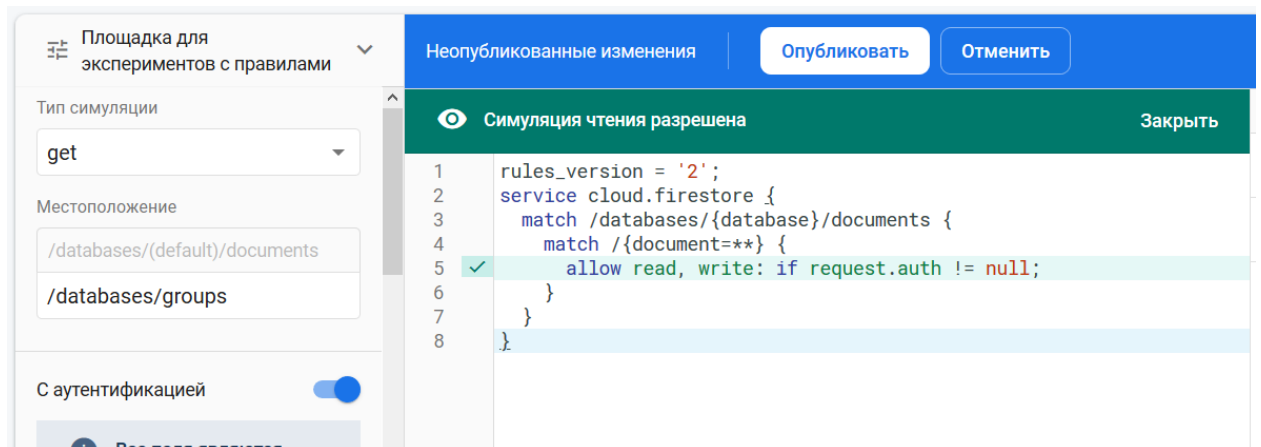


Рис. 15. Симуляція доступу до даних з аутентифікацією.

Переходимо до нашого застосунку та переконуємося у тому, що до проходження аутентифікації спроба доступу до даних веде до помилки (рис. 16), а операція читання даних після ауторизації цілком успішна (рис. 17).

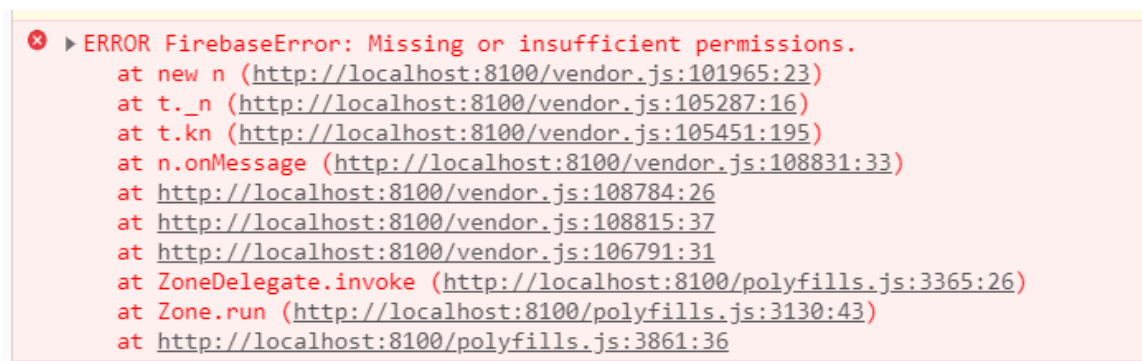


Рис. 17. Спроба звернення до даних без аутентифікації.




Групи		+ ДОДАТИ
	201	комп'ютерні науки
	408	computer science
	308	application engeneering

Рис. 18. Читання даних після виконання входу.