Лабораторна робота 14.

Ionic Framework. Власний back-end та post-запити. Частина 1.

У даній роботі створимо власний back-end сервіс, що відповідатиме за збереження даних та обробку команд нашого застосунку. Для реалізації сервісу обрана мова PHP. Сервіс може бути розміщено в мережі інтернет, або, як у нашому випадку, може бути використано локальний веб-сервер. У рамках даної роботи використано пакет openserver.

У папці ospanel\domains\localhost створимо папку api, а в ній розміщуємо файл index.php. Спочатку реалізуємо тестовий функціонал, що отримуватиме дані від фронт-енд та відправлятиме їх назад (рис.1).

```php
<?php
header( string: 'Content-type: application/json');
header( string: 'Access-Control-Allow-Origin: *');
header( string: 'Access-Control-Allow-Methods: GET, PUT, POST, DELETE,
 OPTIONS');
header( string: 'Access-Control-Allow-Headers: Origin, Content-Type,
 X-Auth-Token , Authorization');

$input = json_decode(file_get_contents( filename: 'php://input'),
  assoc: true);

echo json_encode(
    [
        'get-data' => $_GET,
        'input-data' => $input
    ]
);
```

Рис. 1. Створення тестового API.

Реалізуємо у ionic-застосунку get та post запити до тестового API. Після отримання даних з back-end виводитемо їх на консоль (рис. 2).

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

  this.http.get( url: 'http://localhost/api/?a=10&b=30').subscribe(
      next: data => console.log(data)
  );

```

```
27      this.http.post( url: 'http://localhost/api/',
28          body: {
29            a: 10,
30            b: 30,
31            c: 'text data'
32          }).subscribe(
33          next: data => console.log(data)
34      );
35    }
```

Рис. 2. Запити до API з коду http-test.page.ts.

Переглянемо результат, якщо від відповідає рис. 3, то все виконано вірно.

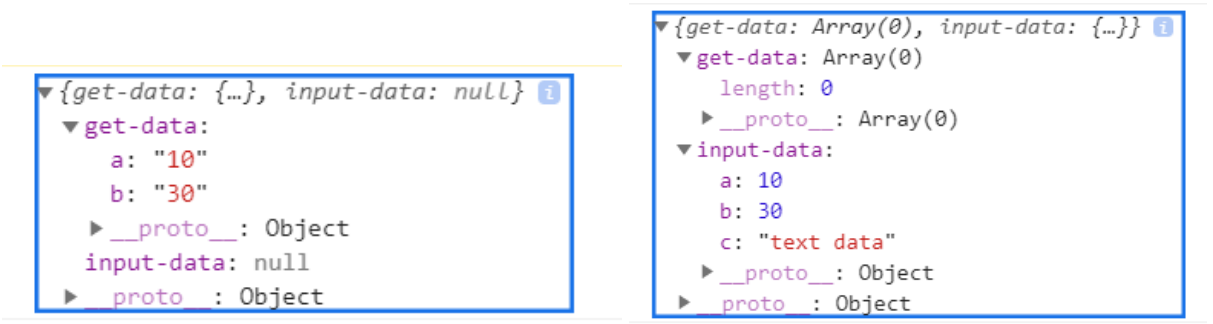


Рис. 3. Виведення даних API на консоль.

Далі переходимо до реалізації роботи із даними студентських груп та студентів. Дані будемо зберігати у БД MySQL. Створимо БД, 3 таблиці та наповнимо їх даними (рис. 4-7).

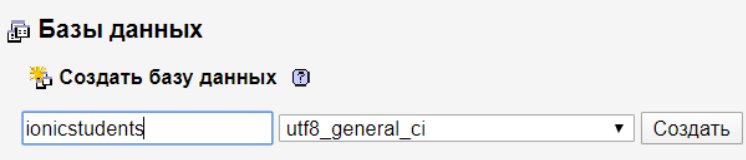
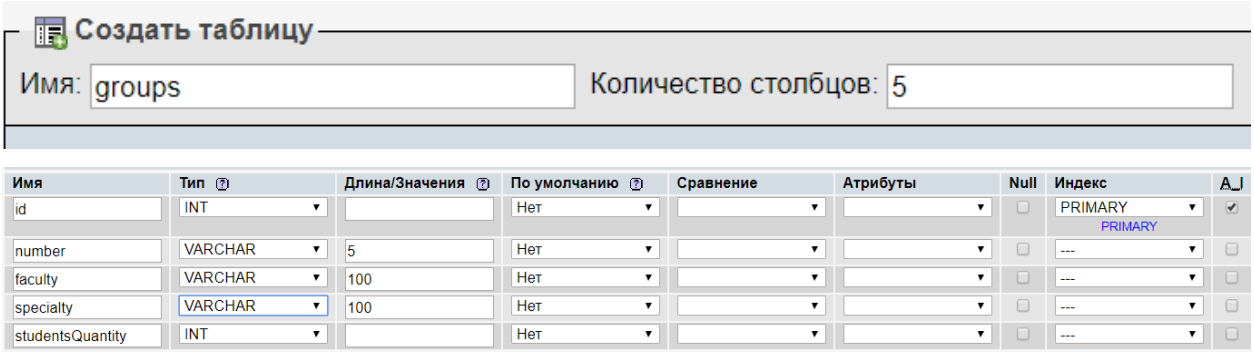


Рис. 4. Створення БД.



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
☐	1	id	int(11)			Нет	*Нем*		AUTO_INCREMENT
☐	2	number	varchar(5)	utf8_general_ci		Нет	*Нем*		
☐	3	faculty	varchar(100)	utf8_general_ci		Нет	*Нем*		
☐	4	specialty	varchar(100)	utf8_general_ci		Нет	*Нем*		
☐	5	studentsQuantity	int(11)			Нет	*Нем*		

id	number	faculty	specialty	studentsQuantity
1	301	факультет комп'ютерних наук	Комп'ютерні науки	22
2	308	факультет комп'ютерних наук	Інженерія програмного забезпечення	24

Рис. 5. Таблиця студентських груп.

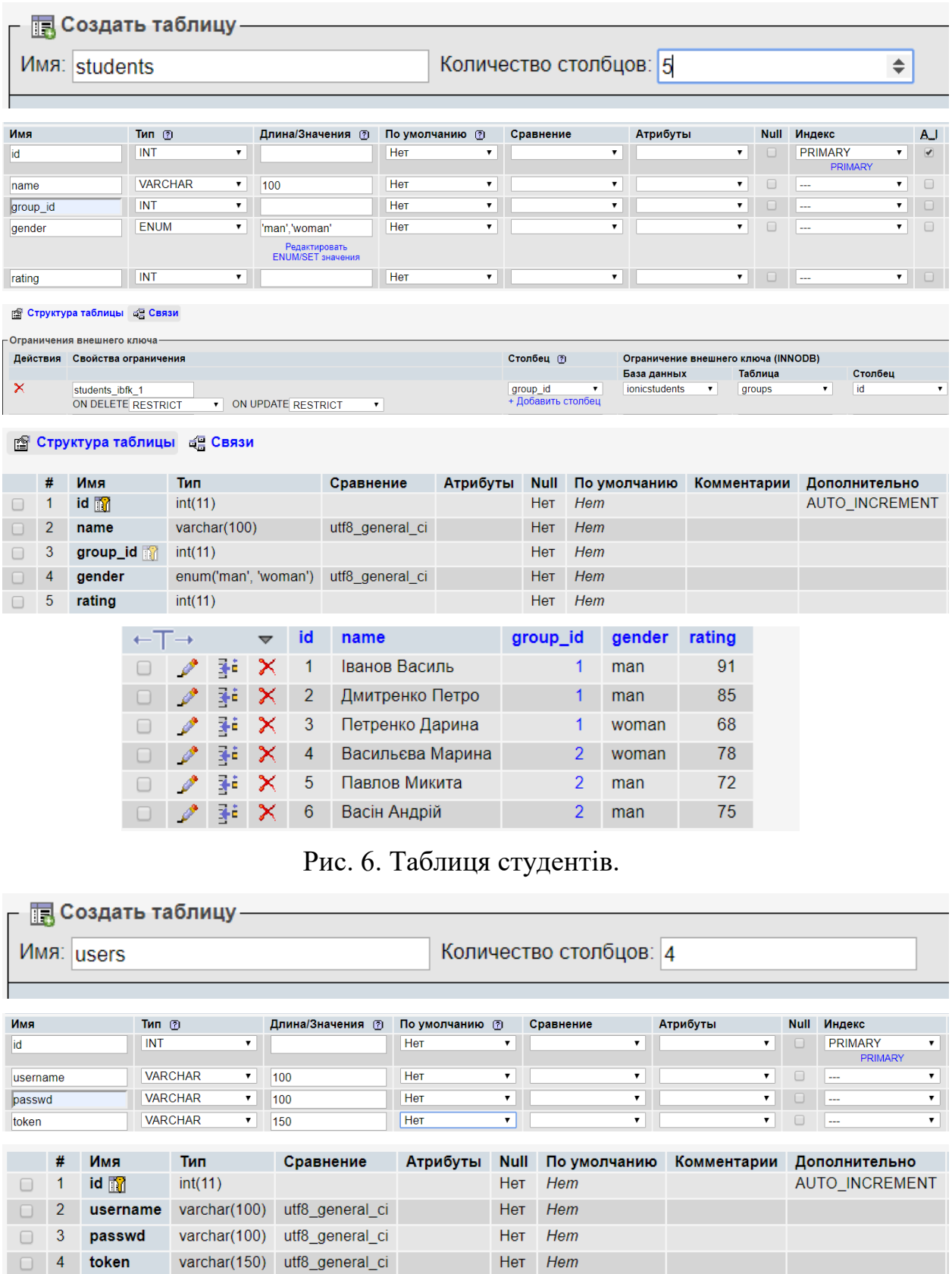


Рис. 6. Таблиця студентів.

Столбец	Тип	Функция	Null	Значение
id	int(11)			
username	varchar(100)			Василь
passwd	varchar(100)	MD5		1
token	varchar(150)			123

Вперёд

id	username	passwd	token
1	Василь	c4ca4238a0b923820dcc509a6f75849b	202cb962ac59075b964b07152d234b70
2	Олена	c81e728d9d4c2f636f067f89cc14862c	caf1a3dfb505ffed0d024130f58c5cfa

Рис. 7. Таблиця користувачів.

Реалізуємо на back-end аутентифікацію та вибірку списку списку студентських груп. У папці api поряд із index.php створимо файли config.php, db.php, model.php, controller.php та auth.php. у файлі config.php зберігаємо налаштування підключення до БД (рис. 8).

```php
<?php
class Config {
    public static $server = "localhost:3306";
    public static $user = "root";
    public static $pwd = "";
    public static $db = "ionicstudents";
}
```

Рис. 8. config.php.

Файл db.php відповідає за підключення до БД, виконання запиту або команди, та повернення результату (рис. 9).

```php
<?php

class DB {
    private $link;
    public $err;

    public function connect() {
        $this->link = new \mysqli(
            \Config::$server, \Config::$user, \Config::$pwd, \Config::$db
        );
        if (!$this->link) {
            return false;
        }
        $this->runQuery( sql: "SET NAMES 'uft-8'");
        return true;
    }

    public function disconnect() {
        $this->link->close();
        unset($this->link);
    }
```

```
    public function runQuery($sql) {
        if (!$this->link) {
            $this->connect();
        }
        $res = $this->link->query($sql);
        if (!$res) {
            $this->err = $this->link->error;
        }
        return $res;
    }

    public function getArrFromQuery($sql) {
        $res_arr = [];
        $rs = $this->runQuery($sql);
        while($row = $rs->fetch_assoc()) {
            $res_arr[] = $row;
        }
        return $res_arr;
    }
}
```

Рис. 9. Файл db.php.

Файл model.php відповідає за вибірку конкретних даних із БД, або виконання конкретних. На даному етапі ми реалізуємо лише один метод для отримання списку груп, але надалі даний клас буде розширюватися (рис. 10).

```
<?php

class Model {
    public static function getGroupsList() {
        return (new DB())->getArrFromQuery(
            sql: "SELECT id, number, faculty, specialty, studentsQuantity
                FROM groups
                order by number");
    }
}
```

Рис. 10. Файл model.php.

У файлі auth.php реалізуємо процедуру логіну користувача і видачу нового token-a, та процедуру перевірки token-a при виконанні подальших запитів на вибірку та модифікацію даних (рис. 11).

```
<?php

class Auth {
    public static function getUserToken($user) {
        $res = (new DB())->getArrFromQuery(
            sql: "select id from users where username='".$user['username']."'
            and passwd=md5('".$user['passwd']."')"
        );
```

```
        if (count($res) > 0) {
            $id = $res[0]['id'];
            $token = bin2hex(random_bytes( length: 64));
            $res2 = (new DB())->runQuery(
                 sql: "update users set token='$token' where id=$id"
            );
            if ($res2) {
                return ['token' => $token];
            }
        }
        return ['error' => 'Username or password is incorrect'];
    }

    public static function checkToken($token) {
        $res = (new DB())->getArrFromQuery(
             sql: "select id from users where token='$token'"
        );
        if (count($res) > 0) {
            return true;
        }
        return false;
    }
}
```

Рис. 11. Файл auth.php.

У файлі controller.php визначаємо, яку саме дію хоче виконати користувач, делегуємо виконання відповідному методу класу model та повертаємо результат користувачу у вигляді json (рис. 12).

```
<?php
class Controller {
    private $data;
    private $action;
    private $protectedActions = ['get-groups'];

    function __construct() {
        $this->action = $_GET['action'];
        $this->data = json_decode(file_get_contents( filename: 'php://input'),  assoc: true);
    }

    function run() {
        $res = [];
        if (in_array($this->action, $this->protectedActions)
            && !Auth::checkToken($_GET['token'])) {
            return ['error' => 'authentication failed !'];
        }
        switch ($this->action) {
            case 'login':
                $res = Auth::getUserToken($this->data);
                break;
            case 'get-groups':
                $res = Model::getGroupsList();
                break;
            default:
                $res = ['error' => 'this route is incorrect !'];
                break;
        }
        echo json_encode($res);
    }
}
```

Рис. 12. Файл controller.php.

І, нарешті, розглянемо index.php, що є своєрідною точкою входу до back-end api (рис. 13).

```php
<?php
require 'config.php';
require 'db.php';
require 'model.php';
require 'auth.php';
require 'controller.php';

header( string: 'Content-type: application/json');
header( string: 'Access-Control-Allow-Origin: *');
header( string: 'Access-Control-Allow-Methods: GET, PUT, POST, DELETE,
 OPTIONS');
header( string: 'Access-Control-Allow-Headers: Origin, Content-Type,
 X-Auth-Token , Authorization');

$input = json_decode(file_get_contents( filename: 'php://input'), assoc: true);

(new Controller())->run();
```

Рис. 13. Файл index.php.

Наступний крок – зміна ionic-застосунку для взаємодії із back-end api. Почнімо із сервісу DataGetterService, що відповідає за отримання даних та передачу їх до необхідної сторінки. Реалізуємо механізми взаємодії із api аутентифікацією та отриманням списку студентських груп (рис. 14).

```typescript
export class DataGetterService {
  baseUrl = 'http://localhost/api/';
  groups = [];
  students = [];
  users = [];

  constructor(private http: HttpClient) {
  }

  private userName = '';
  private token = '';

  checkUser(user) {
    return this.http.post<any>( url: this.baseUrl + '?action=login',
user);
  }

```

```
31    getUser() {
32      return this.userName;
33    }
34
35    setUser(name: string) {
36      this.userName = name;
37    }
38
39    setToken(token: string) {
40      this.token = token;
41    }
42
43    getGroups() {
44      return this.http.get<any>( url: this.baseUrl +
      '?action=get-groups&token=' + this.token);
45    }
46
47    getStudents() {}
48  }
```

Рис. 14. Файл service/data-getter.service.ts.

У файлі app-routing.module.ts повертаємо сторінку login, як стартову (рис. 15).

```
5   const routes: Routes = [
6     { path: '', redirectTo: 'login', pathMatch: 'full' },
7     {
```

Рис. 15. Редірект на сторінку login.

У представлення сторінки login додаємо поле вводу для паролю користувача (рис. 16).

```
14        <ion-item>
15          <ion-label position="floating">Ім'я користувача</ion-label>
16          <ion-input type="text" [(ngModel)]="userName"></ion-input>
17        </ion-item>
18        <ion-item>
19          <ion-label position="floating">Пароль</ion-label>
20          <ion-input type="password" [(ngModel)]="passWord"></ion-input>
21        </ion-item>
```

Рис. 16. Зміни у представленні login.page.html.

Далі доробимо клас LoginPage для підтримки пароля та взаємодії із сервісом back-end (рис. 17).

```
11  export class LoginPage implements OnInit {
12    userName: string;
13    passWord: string;

23    login() {
24      this.dataGetter.checkUser( user: {
25        username: this.userName,
26        passwd: this.passWord
27      }).subscribe(
```

```
28          next: result => {
29            if (result.hasOwnProperty( v: 'error')) {
30              this.userNotExistAlert(result.error);
31            } else {
32              if (result.hasOwnProperty( v: 'token')) {
33                this.dataGetter.setUser(this.userName);
34                this.dataGetter.setToken(result.token);
35                this.router.navigate( commands: ['/home']);
36              } else {
36              } else {
37                this.userNotExistAlert( massage: 'Unexpected error');
38              }
39            }
40          }
41      );
42    }
43
44    async userNotExistAlert(massage) {
45      const alert = await this.alertController.create({
46        header: 'Увага !',
47        subHeader: 'Помилка аутентифікації',
48        message: massage,
49        buttons: ['OK']
50      });
51
52      await alert.present();
53    }
```

Рис. 17. Зміни у login.page.ts.

Перевіримо роботу застосунку, виконаємо вхід та перегляд списку студентських груп (рис. 18). Звернімо увагу, що редагування даних поки не працює.

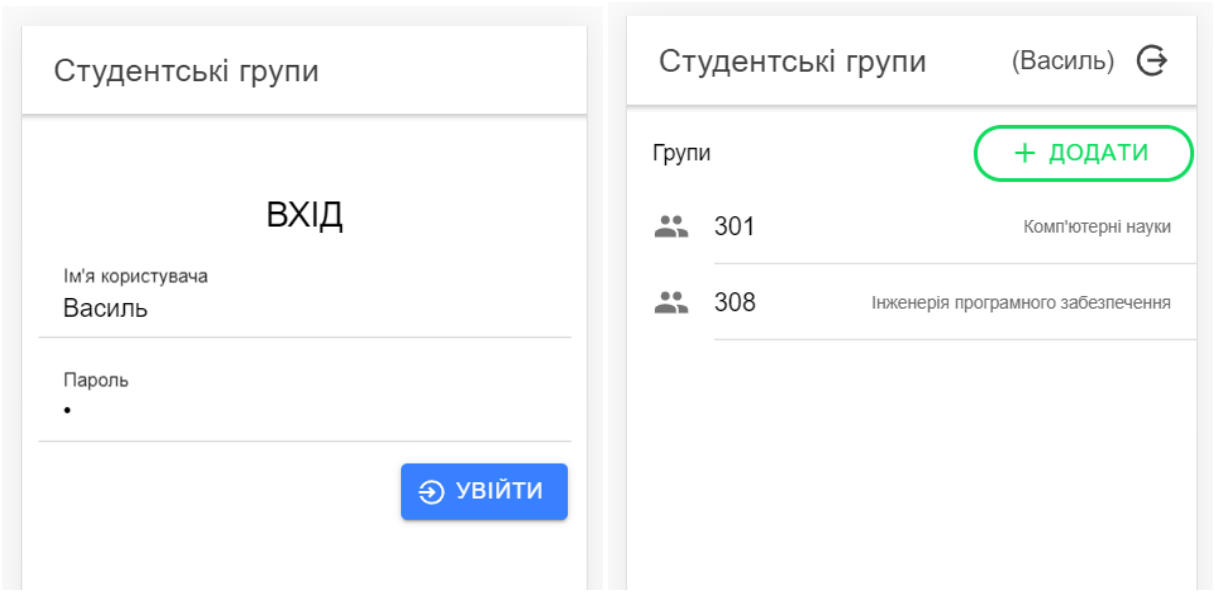


Рис. 18. Перевірка роботи застосунку.