

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»

**ТЕХНОЛОГІЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.  
МЕТОДОЛОГІЇ ТА ЗАСОБИ РОЗРОБКИ**

**КОМП'ЮТЕРНИЙ ПРАКТИКУМ**

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського  
як навчальний посібник для студентів,  
які навчаються за спеціальністю  
151 «Автоматизація та комп'ютерно-інтегровані технології»,  
спеціалізацією «Технічні та програмні засоби автоматизації»*

Київ  
КПІ ім. Ігоря Сікорського  
2020

Технології розробки програмного забезпечення. Методології та засоби розробки: комп'ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», спеціалізації «Технічні та програмні засоби автоматизації» / КПІ ім. Ігоря Сікорського; уклад.: Д. О. Ковалюк. – Електронні текстові данні (1 файл: 1,2 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 36 с.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 8 від 09.04.2020 р.)*

*за поданням Вченої ради інженерно-хімічного факультету (протокол № 2 від 24.02.2020 р.)*

Електронне мережне навчальне видання

## **ТЕХНОЛОГІЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. МЕТОДОЛОГІЇ ТА ЗАСОБИ РОЗРОБКИ КОМП'ЮТЕРНИЙ ПРАКТИКУМ**

Укладач: *Ковалюк Дмитро Олександрович, канд. техн. наук, доц.*

Відповідальний  
редактор *Жученко Анатолій Іванович, д-р техн. наук, проф.*

Рецензенти: *Гулієнко Сергій Валерійович, канд. техн. наук, доцент*

Посібник призначений для формування у студентів комплексу знань, умінь та досвіду, необхідних для створення програмного забезпечення комп'ютерних систем керування та інформаційних систем. Дозволяє скорочено за формою, але всебічно за змістом викласти студентам технології розробки та функціонування програмного забезпечення розподілених систем.

© КПІ ім. Ігоря Сікорського, 2020

## ЗМІСТ

Вступ.....	4
Робота 1	
Налаштування програмного забезпечення web-програмування .....	5
Робота 2	
PHP: Синтаксис, операції, управляючі конструкції, масиви.....	10
Робота 3	
PHP: Робота з формами .....	13
Робота 4	
SQL: робота з запитамі .....	14
Робота 5	
PHP: робота з базою даних.....	18
Робота 6	
PHP: створення проекту на основі Laravel фреймворку.....	19
Літературні джерела	

## **ВСТУП**

Автоматизація виробничих процесів є важливою задачею, що дозволяє суттєво підвищити продуктивність праці, покращити якість продукції, зберегти енергоресурси, оптимізувати склад обслуговуючого персоналу, підвищити надійність роботи.

Успішне розв'язання усіх вище перерахованих задач досягається впровадженням у виробництво автоматичних систем управління, складовими яких виступає програмне забезпечення. Знання методів і засобів розробки програмного забезпечення відіграє при цьому вирішальну роль.

## **РОБОТА №1**

### **НАЛАШТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ WEB-ПРОГРАМУВАННЯ**

**Мета роботи** – розгорнути програмне забезпечення web-програмування.

#### **Хід роботи**

1. Завантажити і встановити наступне програмне забезпечення
  - Веб-сервер Apache
  - Інтерпретатор мови програмування PHP
  - Засіб для відлагодження PHP-програм Xdebug
  - IDE розробки – на вибір студента (NetBeans, PHPStorm, тощо)
2. Створити два віртуальні хости для запуску сайтів.
3. Перший хост має містити просту html-сторінку з назвою групи
4. Другий хост має містити php-файл з наступним кодом

`<? phpinfo () ; ?>`

#### **Приклад виконання**

1. Встановити пакет програм Open Server, що доступний за посиланням <https://ospanel.io>. Достатньо поставити збірку «BASIC».
2. Ознайомитися з принципом роботи та структурою директорій, представленою на рис. 1.1. Запустити сервер.

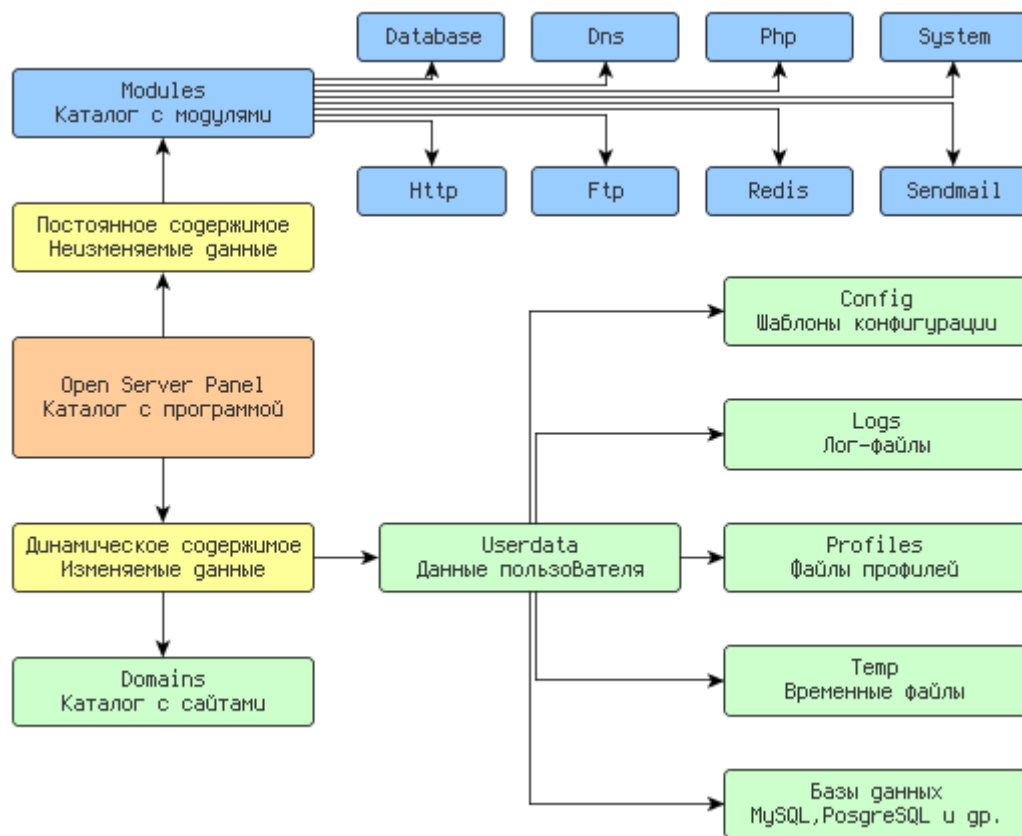


Рис. 1.1. Структура директорій Open Server

3. Перевірити версії встановленого програмного забезпечення (вкладка «Модулі»), і за можливостей операційної системи поставити максимальні версії PHP, MySQL (рис. 1.2.).

4. Створити віртуальні хости на вкладці «Домени» (рис. 1.3.). В кожному віртуальному хості створити файл, як вимагає завдання 3-4.

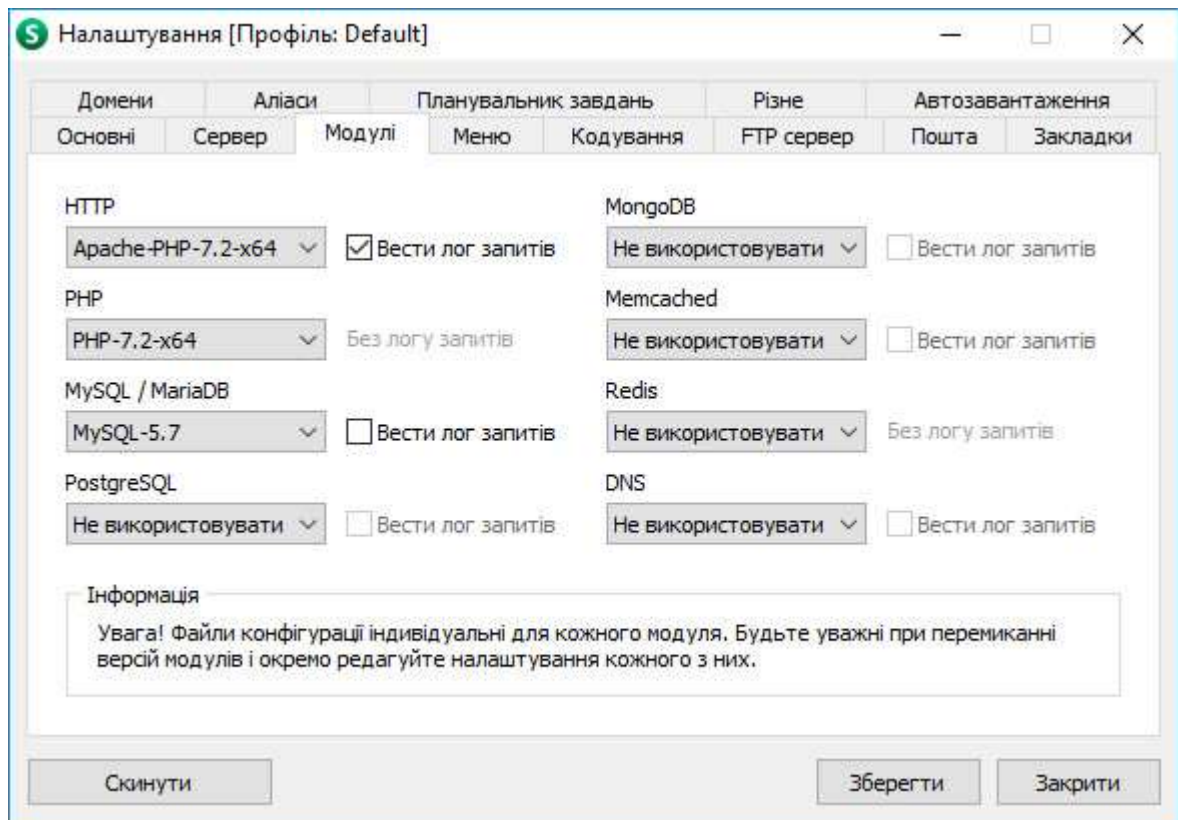


Рис.1.2. Вкладка «Модулі»

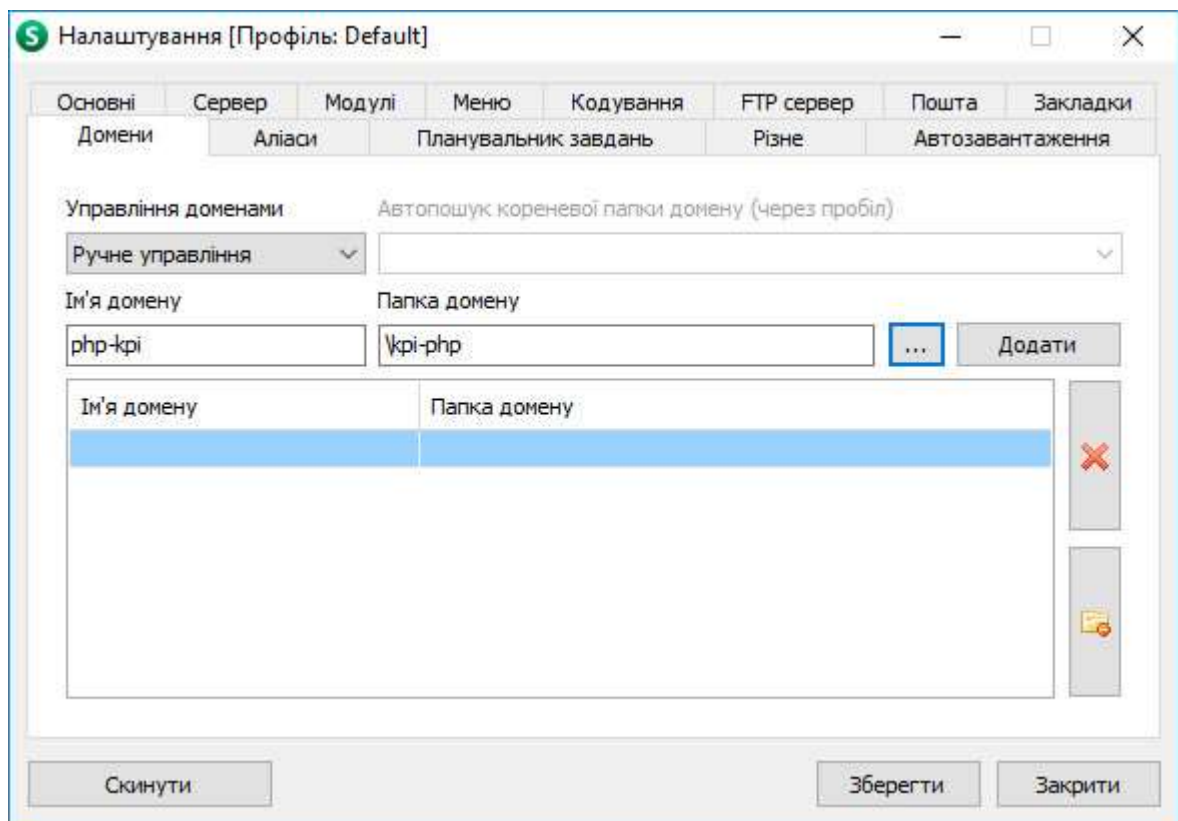


Рис.1.3. Вкладка «Домени»

5. Запустити в браузері сайт на віртуальному хості з php-файлом і перевірити інформацію про PHP (рис. 1.4.).


PHP Version 7.2.0 	
System	Windows NT DESKTOP-3BTJIEQ 10.0 build 17134 (Windows 10) AMD64
Build Date	Nov 28 2017 23:44:10
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x64
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=.\obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\OSPanel\modules\php\PHP-7.2-x64\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,TS,VC15

Рис.1.4. Виведення інформації про PHP

6. Встановлення Xdebug. Даний відлагоджувач за замовчуванням встановлений в Open Server. Для його використання потрібно розкоментувати відповідні строки в файлі налаштування php.ini. <https://xdebug.org/docs/all>

[Xdebug]

```
zend_extension="c:/ospanel/modules/php/PHP-7.2-x64/ext/php_xdebug.dll"
xdebug.remote_enable = 1
xdebug.remote_handler = "dbgp"
xdebug.remote_host = "localhost"
xdebug.remote_port = 9000
```

Примітка: розміщення файлу php.info можна визначити з рис. 1.4. – Loaded Configuration File. Якщо параметри Xdebug не вступають в силу, то потрібно внести аналогічні змінні в папці userdata.

При ввімкненні Xdebug – в файлі, який генерується php.info має з'явитися відповідна секція. Тільки після цього можна налаштовувати процес відлагодження в IDE.



### wddx

WDDX Support	enabled
WDDX Session Serializer	enabled

### xdebug

xdebug support	enabled
Version	2.5.5
IDE Key	Dima

Supported protocols	Revision
DBGp - Common DeBuGger Protocol	\$Revision: 1.145 \$

Directive	Local Value	Master Value
xdebug.auto_trace	Off	Off
xdebug.cli_color	0	0
xdebug.collect_assignments	Off	Off
xdebug.collect_includes	On	On
xdebug.collect_params	0	0
xdebug.collect_return	Off	Off
xdebug.collect_vars	Off	Off
xdebug.coverage_enable	On	On
xdebug.default_enable	On	On
xdebug.dump.COOKIE	<i>no value</i>	<i>no value</i>

Рис.1.5. Виведення інформації про Xdebug

### Контрольні запитання

1. Переваги web-додатків, поняття домену і хостингу
2. Статичний і динамічний контент сайту
3. Як запит з браузера потрапляє на web-сервер
4. Що таке DNS
5. Способи реалізації php-інтерпретатора
6. Основні методи http-протоколу

## РОБОТА №2

### PHP: СИНТАКСИС, ОПЕРАЦІЇ, УПРАВЛЯЮЧІ КОНСТРУКЦІЇ, МАСИВИ

**Мета роботи** – ознайомитися з синтаксисом мови програмування php, основними операціями, управляючими конструкціями, навчитися оголошувати функції і масиви.

#### Хід роботи

1. Написати програму «Калькулятор», яка виконує найпростіші арифметичні операції – додавання, віднімання, множення, порівняння.
2. Кожну арифметичну операцію оформити у вигляді окремого методу
3. Задати константи для кожної з арифметичних операцій.
4. Викликати відповідну функцію, використовуючи інструкцію switch, яка приймає на вхід змінну зі значенням арифметичної операції.
5. Написати програму «Деканат», в якій оголосити асоційований масив «навчальна група», що містить інформацію про студентів: прізвище, рейтинг, стать.
6. Обійти масив циклом і порахувати кількість хлопців і дівчат в групі.
7. Написати функцію, яка шукає в масиві студента за його прізвищем.

#### Приклад виконання

```
const SEX_MALE = 'male';  
const SEX_FEMALE = 'female';  
  
$la7lp = array();  
$la7lp[] = array('fio' => 'Ivanov', 'ratinh' => 4.7, 'sex' => 'male');  
$la7lp[] = array('fio' => 'Petrov', 'ratinh' => 4.3, 'sex' => 'male');  
$la7lp[] = array('fio' => 'Sidorov', 'ratinh' => 4.2, 'sex' => 'male');  
$la7lp[] = array('fio' => 'Sirova', 'ratinh' => 4.1, 'sex' => 'female');
```

```

$men_count = 0;
foreach ($la71p as $student) {
    if($student['sex'] == SEX_MALE){
        $men_count = $men_count + 1;
    }
}
echo 'mens count is ', $men_count;
echo 'girls count is ', count($la71p) - $men_count;

function search_student($group, $rating_bound){
    foreach ($group as $student) {
        if($student['ratinh'] >= $rating_bound){
            $men_count = $men_count + 1;
            print $student['fio'] . ' ' . $student['ratinh'];
        }
    }
}

search_student($la71p, 4.2);

```

```

<?php
function sum($a, $b){
    return $a + $b;
}

function subtraction($a, $b){
    return $a - $b;
}

function compare($a, $b){
    if ($a == $b) return 'a = b';
    return ($a < $b) ? 'a < b' : 'a > b';
}

function calculate($a, $b, $oprator)
{
    switch($oprator)
    {
        case '+':
            $result = sum($a, $b);
            break;

        case '-':
            $result = subtraction($a, $b);
            break;

        case '?':
            $result = compare($a, $b);
            break;

        default:
            $result = "Sorry No command found";
    }
    return $result;
}

?>

```

## **Контрольні запитання**

1. Оголошення змінних і констант
2. Оголошення та ініціалізація масивів
3. Умовні оператори та оператор розгалуження
4. Циклічні конструкції
5. Оголошення та виклик функцій
6. Формальні та фактичні параметри, параметри за замовчуванням
7. Передача аргументів за значенням та за посиланням

## РОБОТА №3

### PHP: РОБОТА З ФОРМАМИ

**Мета роботи** – ознайомитися з html-формами та обробкою їх значень на стороні з використанням PHP.

#### Хід роботи

1. Доопрацювати створені в попередній роботі програми «Калькулятор» і «Деканат».
2. Розробити інтерфейс для даних програм, використовуючи теги html.
3. Програма «Деканат» має відображати:
  - список всіх студентів
  - перегляд студентів певної групи
  - пошук студента
  - сортування студентів
4. Програма «Калькулятор» - повинна містити форму для введення двох аргументів, вибору операції, строку виведення результату.
5. При виконанні арифметичної операції – в елементах форми повинні залишитися значення, які ввів користувач.

#### Контрольні запитання

1. Оголошення форми в html
2. Різниця між методами get і post
3. Суперглобальні змінні, отримання значень форми на сервері
4. Визначення методу http-запиту в коді php-скрипта
5. Відображення значень користувача після спрацювання форми

## РОБОТА №4

### SQL: РОБОТА З ЗАПИТАМИ

**Мета роботи** – ознайомитися з основами реляційних баз даних, встановити систему управління базами даних Mysql та один з графічних клієнтів. Створити схему даних, таблиці, написати запити на вибірку, оновлення, вставку та видалення даних (CRUD).

#### Хід роботи

1. Встановити СУБД Mysql та один з графічних клієнтів (Workbench, Valentine-db, dbForge Studio, phpMyadmin)

<https://dev.mysql.com/downloads/installer/>

<https://dev.mysql.com/downloads/workbench/>

<https://www.valentina-db.com/ru/>

<https://www.devart.com/dbforge/mysql/>

2. Запустити клієнт і створити локальне підключення до сервера СУБД

3. Створити схему КРІ та наступні таблиці:

- навчальна група
- студенти
- рейтинг студентів
- гуртожиток.

4. Заповнити таблиці даними – по 3-10 записів. По 1-2 записи вставити програмно з використанням команди INSERT INTO

## 5. Виконати наступні SQL - запити

- Порахувати кількість усіх дівчат на факультеті
- Вивести всіх хлопців, відсортованих по алфавіту
- Порахувати кількість всіх хлопців і дівчат на факультеті
- Вивести студентів певної групи, які отримують стипендію
- Вивести перших 5 студентів за рейтингом
- Вивести студентів які живуть в гуртожитку
- Вивести всіх студентів, вказавши для кожного можливий номер гуртожитку (LEFT OUTER JOIN)
- Виконати операції вставки, редагування, видалення записів

### Приклад виконання

```
CREATE TABLE `groups` (  
  `group_id` int(11) NOT NULL,  
  `title` varchar(45) NOT NULL,  
  `state` varchar(45) NOT NULL,  
  PRIMARY KEY (`group_id`)
```

```
CREATE TABLE `students` (  
  `student_id` int(11) NOT NULL,  
  `fio` varchar(70) NOT NULL,  
  `group_id` int(11) NOT NULL,  
  `birthday` date DEFAULT NULL,  
  `sex` smallint(6) DEFAULT NULL,  
  PRIMARY KEY (`student_id`)
```

```
CREATE TABLE `students_hostel` (  
  `student_id` int(11) NOT NULL,  
  `hostel_id` tinyint(4) NOT NULL,  
  PRIMARY KEY (`student_id`)  
)
```

```
CREATE TABLE `students_rating` (  
  `student_id` int(11) NOT NULL,  
  `semestr_id` tinyint(4) NOT NULL,  
  `rating` double NOT NULL,  
  `stipend` tinyint(4) NOT NULL DEFAULT '0',  
  PRIMARY KEY (`student_id`, `semestr_id`)  
)
```

```
select count(*) from kpi.students where sex = 2;
```

```
select *  
  from kpi.students  
 where sex = 1  
order by fio;
```

```
select sex, count(student_id)  
  from kpi.students  
group by sex;
```

```
select s.fio, sr.rating, sr.stipend as stipendiya  
  from kpi.students s, kpi.students_rating sr  
 where s.student_id = sr.student_id  
       and sr.stipend = 1  
order by sr.rating desc  
       limit 5;
```

```
select s.fio, sr.rating, sr.stipend as stipendiya  
  from kpi.students s inner join kpi.students_rating sr  
                                on s.student_id = sr.student_id  
 where sr.stipend = 1;
```

```
select s.fio, sr.rating,  
       sr.stipend as stipendiya, g.title as grupa  
  from kpi.students s  
       inner join kpi.students_rating sr on s.student_id =  
       sr.student_id  
       inner join kpi.groups g on g.group_id = s.group_id  
 where sr.stipend = 1  
order by sr.rating desc;
```

```
select s.fio, sr.rating, sr.stipend as stipendiya,  
       g.title as grupa  
  from kpi.students s  
       inner join kpi.students_rating sr  
       on s.student_id = sr.student_id  
       inner join kpi.groups g on g.group_id = s.group_id  
 where sr.stipend = 1  
order by sr.rating desc;
```



```
select s.fio, g.title as grupa, h.hostel_id
  from kpi.students s,
       kpi.groups g,
       kpi.students_hostel h
 where g.group_id = s.group_id
       and s.student_id = h.student_id
order by fio;
```

```
select s.fio, g.title as grupa, h.hostel_id
  from kpi.students s
       inner join kpi.groups g on g.group_id = s.group_id
       left outer join kpi.students_hostel h
       on s.student_id = h.student_id
 where s.sex = 1
order by fio;
```

```
INSERT INTO `kpi`.`groups`(`group_id`, `title`, `state`)
VALUES (4, '1a-71', 'Бюджет');
```

```
UPDATE `kpi`.`students_rating`
  SET `rating` = `rating` + 1
 WHERE `student_id` in (9, 10);
```

```
UPDATE `kpi`.`groups`
  SET `title` = '1А-71'
 WHERE `group_id` = 4;
```

```
DELETE FROM `kpi`.`groups` WHERE `group_id` = 4;
```

## Контрольні запитання

1. Поняття таблиці, первинного та зовнішнього ключів
2. Оператори підтримання цілісності даних
3. Зв'язки між сутностями таблиць (один до одного, один до багатьох)
4. Типи даних MySQL
5. Основні оператори SQL, агрегатні функції
6. Вибір даних за умовою, обмеження на кількість записів
7. Вибір даних з декількох таблиць, псевдоніми
8. Внутрішні і зовнішні об'єднання таблиць

## РОБОТА №5

### PHP: РОБОТА З БАЗОЮ ДАНИХ

**Мета роботи** – ознайомитися з php-розширеннями, які надають доступ до баз даних MySQL. Навчитися динамічно формувати і виконувати SQL-запити на основі параметрів користувача, виводити отриманий результат.

#### Хід роботи

1. Використовувати розроблену в попередніх роботах програму «Деканат»
2. Створити клас підключення до БД – модель «DB»
3. Створити модель «Студент», яка наслідує модель «DB»
4. В моделі «Студент» передбачити наступні методи:
  - Отримання інформації про групи студентів
  - Отримання списку студентів
  - Отримання інформації про конкретного студента
  - CRUD – операції для роботи з сутністю «студент»
5. Створити підключення і вибір даних з використанням Mysqlі або PDO

#### Контрольні запитання

1. Mysqlі та PDO розширення для роботи з базою даних
2. Вбудовані функції для під'єднання до БД
3. Prepare-запити з параметрами
4. Вбудовані функції для виконання SQL-запитів
5. Вивід результату SQL-запиту на екран

## РОБОТА №6

### PHP: СТВОРЕННЯ ПРОЕКТУ НА ОСНОВІ LARAVEL ФРЕЙМВОРКУ

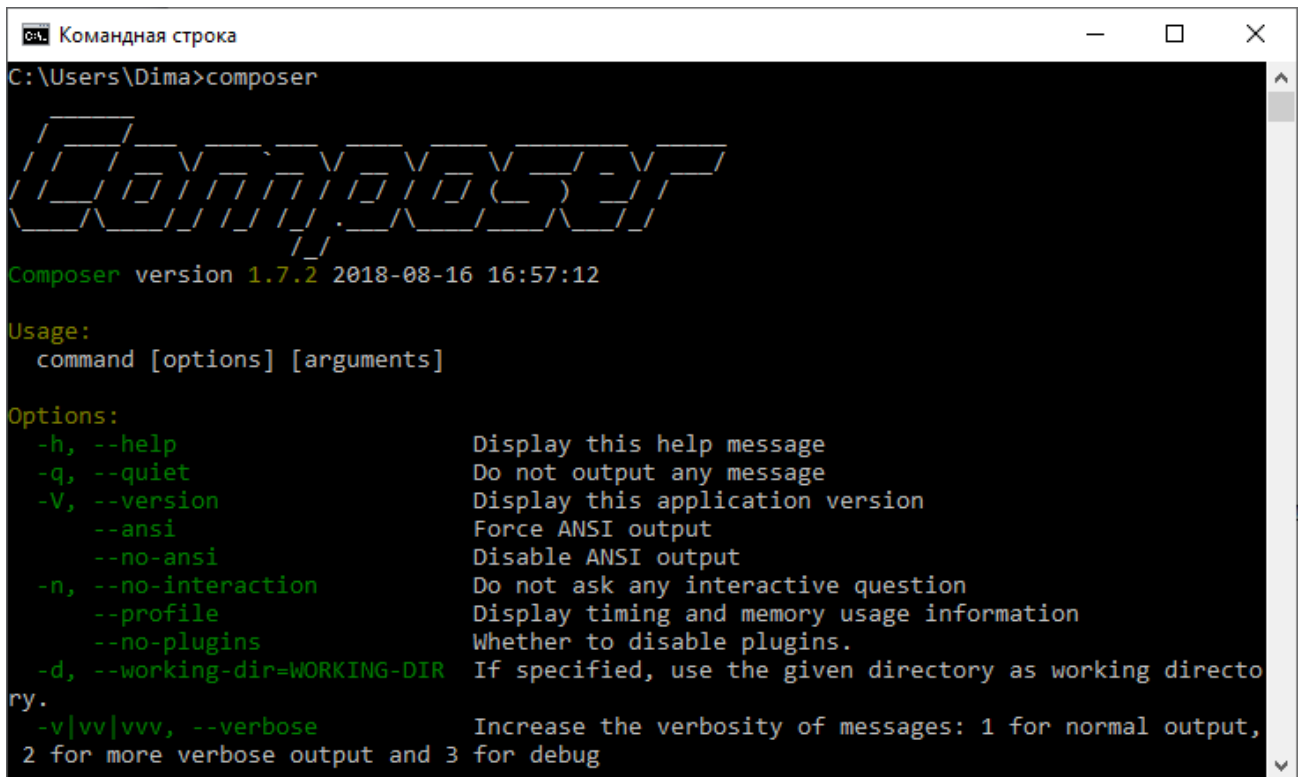
**Мета роботи** – навчитися створювати програмне забезпечення з використанням готових архітектурних рішень, ознайомитися з основними можливостями фреймворку laravel.

#### Хід виконання роботи

1. Встановити Composer
2. Встановити Laravel
3. Налаштувати підключення до БД
4. Створити два контролера Students і Groups
5. Контролер Students реалізувати як контролер ресурсів
6. Передбачити наступний функціонал контролеру Students
  - виведення списку всіх груп
  - виведення студентів певної групи за рейтингом
  - виведення інформації про студента
  - створити front-end та back-end сайту на основі авторизації
  - зробити перевірки на основі посередників дій
  - зробити інтерфейс з використанням Bootstrap

## Приклад виконання

1. Встановити менеджер залежностей composer враховуючи відповідну ОС <https://getcomposer.org/download/> та перевірити його роботу



```
Командная строка
C:\Users\Dima>composer

Composer version 1.7.2 2018-08-16 16:57:12

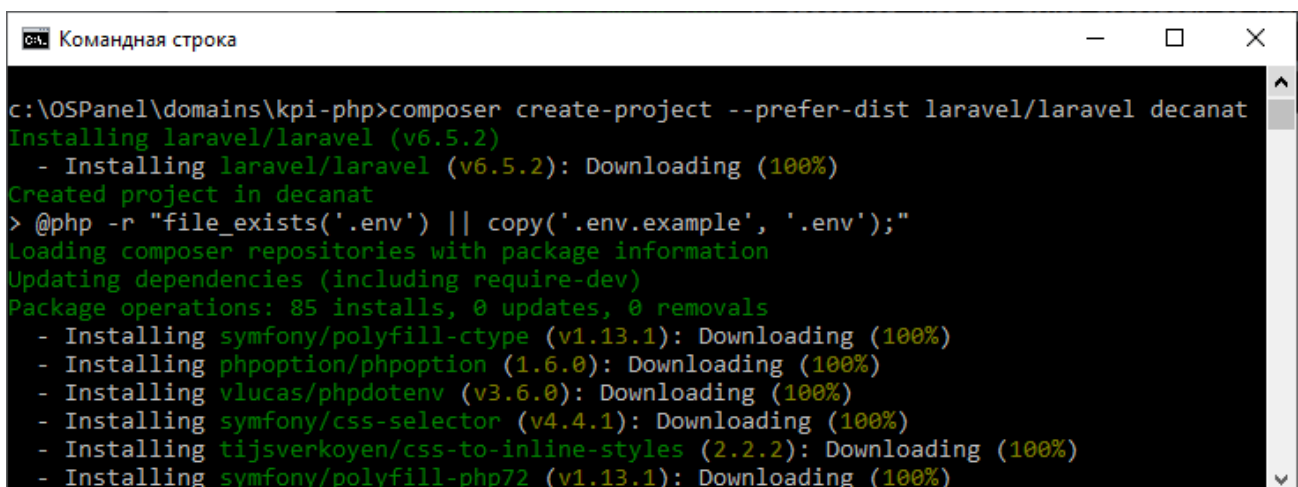
Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi                   Force ANSI output
  --no-ansi                Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
  --profile                Display timing and memory usage information
  --no-plugins              Whether to disable plugins.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  -v|vv|vvv, --verbose      Increase the verbosity of messages: 1 for normal output,
                             2 for more verbose output and 3 for debug
```

Рис. 6.1. Перевірка роботи composer

2. Створити додаток на основі laravel фреймворку

```
composer create-project --prefer-dist laravel/laravel decanat
```



```
Командная строка
c:\OSPanel\domains\kpi-php>composer create-project --prefer-dist laravel/laravel decanat
Installing laravel/laravel (v6.5.2)
 - Installing laravel/laravel (v6.5.2): Downloading (100%)
Created project in decanat
> @php -r "file_exists('.env') || copy('.env.example', '.env');";
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 85 installs, 0 updates, 0 removals
 - Installing symfony/polyfill-ctype (v1.13.1): Downloading (100%)
 - Installing phproption/phpoption (1.6.0): Downloading (100%)
 - Installing vlucas/phpdotenv (v3.6.0): Downloading (100%)
 - Installing symfony/css-selector (v4.4.1): Downloading (100%)
 - Installing tijsverkoyen/css-to-inline-styles (2.2.2): Downloading (100%)
 - Installing symfony/polyfill-php72 (v1.13.1): Downloading (100%)
```

Рис. 6.2. Створення власного проекту на основі Laravel

### 3. Створити віртуальний хост і направити його на папку **public**

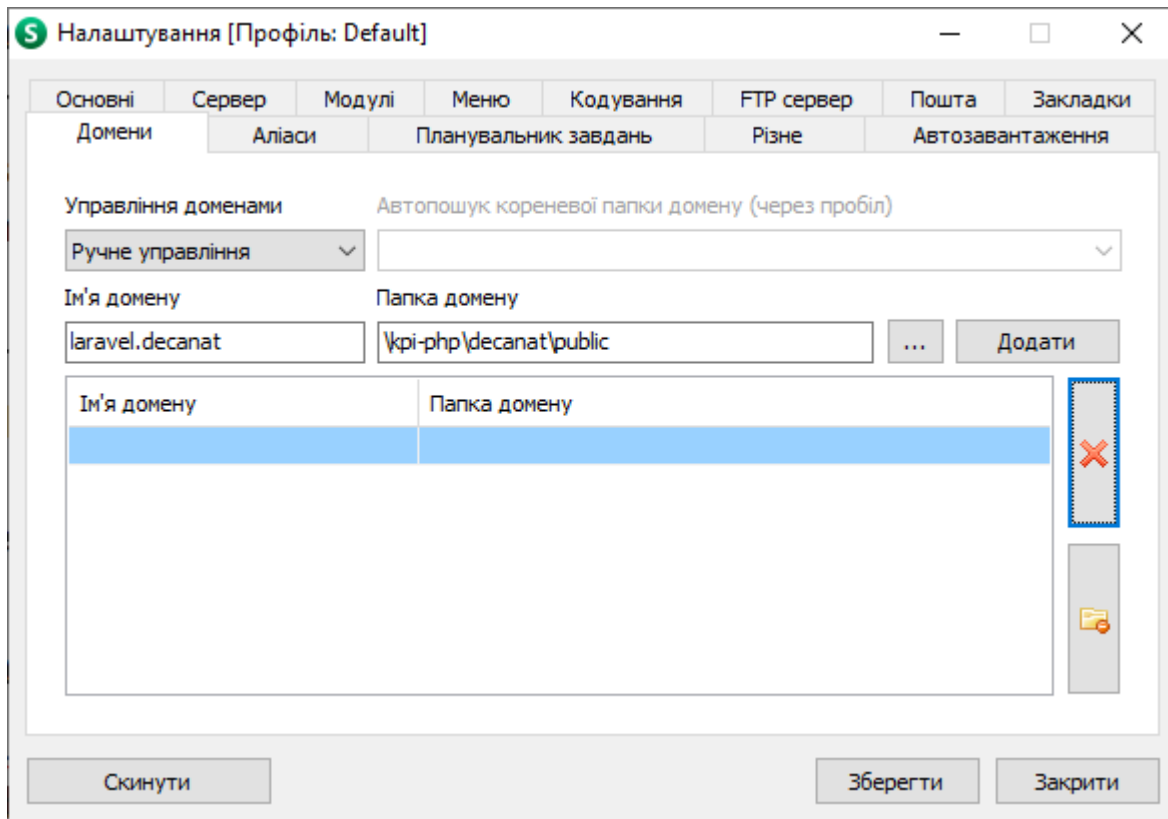


Рис. 6.3. Створення домену

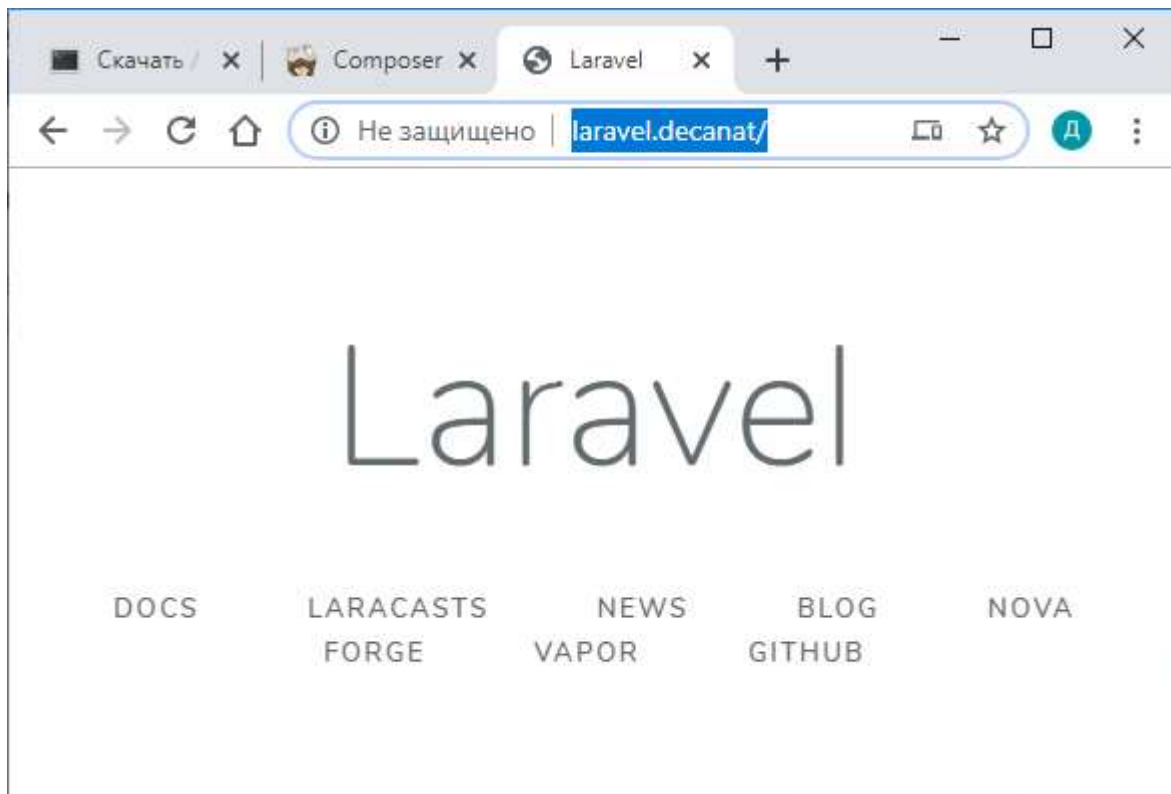


Рис. 6.4. Перевірка роботи Laravel

#### 4. Створити роутери

URL-адреси запитів

/groups

/groups/{id}, де параметр id – шифр групи.

Направити запити на контролер Groups

Request	Controller	Action
/groups	GroupsController	index
/groups/{id}	GroupsController	group

В файлі routes/web.php додати наступні строки

```
Route::get('groups', 'GroupsController@groups');  
Route::get('groups/{group}', 'GroupsController@group');
```

#### 5. Створити контролер з використанням Artisan

```
php artisan make:controller GroupsController
```

#### 6. Створити в контролері відповідні методи (actions)

```
public function index(Request $request) {...}
```

```
public function group($id) {...}
```

7. Створити шаблон для frontend – частини, який повинен містити наступні складові:

- секцію для заголовка сторінки
- секцію для виведення контенту
- підключати зовнішній файл футера сайту.

8. Створити скрипти виду для методів у GroupsController, які наслідують frontend-шаблон

- list – відображає список навчальних груп, передбачає селектори форми навчання та курсу
- group – відображає інформацію про конкретну навчальну групу

## Файл шаблону сайту

```
<html>
  <head>
    <title>Інформаційна система "Деканат"</title>
  </head>

  <body>

    @yield('page_title')

    <br/><br/>

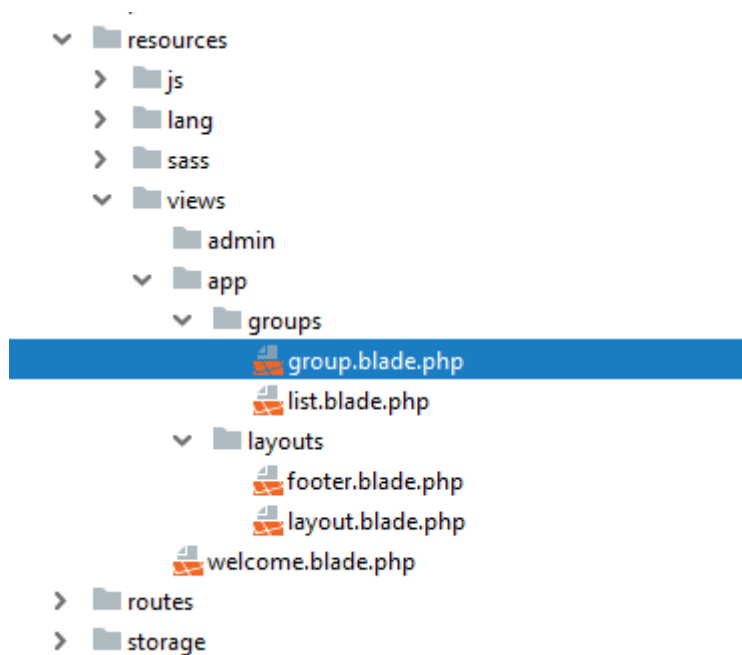
    <div class="container">
      Основна інформація
      @yield('content')
    </div>

    <br/>
    <br/>
  </body>

  @include('layouts.footer')
</html>
```

## Файл футеру

```
<footer>
  ТРПЗ - 2019, Лабораторна робота №6
</footer>
```



## 9. Створити модель Groups, яка має повертати інформацію про групи

```
php artisan make:model Groups
```

В моделі передбачити масив-заглушку, з якого будуть вибиратися дані. В подальшому ця інформація буде вибрана з бази даних

```
private $groups = array(
    array('group_id' => 3, 'type' => 1, 'kurs' => 2, 'title' => 'ЛА-82',
    'starosta' => 'Меркушина'),
    array('group_id' => 1, 'type' => 1, 'kurs' => 2, 'title' => 'ЛА-81',
    'starosta' => 'Джима'),
    array('group_id' => 6, 'type' => 2, 'kurs' => 1, 'title' => 'ЛА-92',
    'starosta' => 'Симиренко'),
    array('group_id' => 5, 'type' => 2, 'kurs' => 1, 'title' => 'ЛАЗ-92',
    'starosta' => 'Підгрушна'),
);

public static $groups_types = array(
    '1' => 'стаціонар',
    '2' => 'заочна'
);
```

Додати та реалізувати в моделі 2 методи

Повертає список груп для відповідної форми навчання та курсу:

```
public function getGroups($type, $kurs){

    $groups = array();
    foreach ($this->groups as $group){
        $match_type = ($type == 0 || $group['type'] == $type);
        $match_kurs = ($kurs == 0 || $group['kurs'] == $kurs);

        if($match_type && $match_kurs){
            $groups[] = $group;
        }
    }
    return $groups;
}
```

Повертає інформацію про групу з відповідним ідентифікатором

```
public function getGroupByID($id){
    if(!$id) return null;

    foreach ($this->groups as $group){
        if($group['group_id'] == $id){
            return $group;
        }
    }
    return null;
}
```



## 10. Реалізувати методи контролера, використовуючи модель.

```
public function index(Request $request) {
    $type = $request->input('type', null);
    $kurs = $request->input('kurs', null);

    $model_groups = new Groups();

    $groups = $model_groups->getGroups($type, $kurs);

    return view('app.groups.list', [
        'groups' => $groups,
        'group_types' => Groups::$groups_types,
        'type_selected' => $type,
        'kurs_selected' => $kurs
    ]);
}

public function group($id) {

    $model_groups = new Groups();
    $group = $model_groups->getGroupByID($id);

    return view('app.groups.group')->with('group', $group);
}
```

## 11. Реалізувати скрипти виду

resources/views/app/groups/group.blade.php

```
@extends('app.layouts.layout')

@section('page_title')
    <b>Інформація про групу {{ $group['title'] }}</b>
@endsection

@section('content')
    <p>Шифр - {{ $group['title'] }}</p>
    <p>Курс - {{ $group['kurs'] }}</p>
    <p>Староста - {{ $group['starosta'] }}</p>

    <a href="/groups">Дивитися всі групи</a>
@endsection
```

resources/views/app/groups/list.blade.php

```
@extends('app.layouts.layout')

@section('page_title')
    <b>Список груп</b>
@endsection

@section('content')
    <form method="get" action="/groups">
        <select name="type">
            <option value="0">Всі категорії</option>

            @foreach($group_types as $type_id => $type_title)
                <option value="{{ $type_id }}"
                    {{ ( $type_id == $type_selected ) ? 'selected' : '' }}>
                    {{ $type_title }}
                </option>
            @endforeach
        </select>

        <select name="kurs">
            <option value="0">Всі курси</option>

            @foreach(array(1, 2, 3, 4, 5, 6) as $kurs_id)
                <option value="{{ $kurs_id }}"
                    {{ ( $kurs_id == $kurs_selected ) ? 'selected' : '' }}>
                    {{ $kurs_id }}
                </option>
            @endforeach
        </select>

        <input type="submit" value="Знайти" />
    </form>

    <table>
        <tr>
            <th>Шифр групи</th>
            <th>Курс</th>
            <th>Форма навчання</th>
            <th>Староста</th>
        </tr>
        @foreach ($groups as $group)
            <tr>
                <td>
                    <a href="/group/{{ $group['group_id'] }}">
                        {{ $group['title'] }}
                    </a>
                </td>
                <td>{{ $group['kurs'] }}</td>
                <td>{{ $group_types[$group['type']] }}</td>
                <td>{{ $group['starosta'] }}</td>
            </tr>
        @endforeach
    </table>
@endsection
```

## Робота з БД – конструктор запитів

### 1. Змінити методи моделі Groups.

```
use Illuminate\Support\Facades\DB;

public function getGroups($type, $kurs){
    $query = DB::table('groups');

    $query->select('group_id', 'title', 'state', 'kurs')
        ->orderBy('title');

    if($type){
        $query->where('type', '=', $type);
    }

    if($kurs){
        $query->where('kurs', '=', $kurs);
    }

    $groups = $query->get();
    return $groups;
}

public function getGroupByID($id){
    if(!$id) return null;

    $group = DB::table('groups')
        ->select('*')
        ->where('group_id', $id)
        ->get()->first();

    return $group;
}
```

### 2. Враховуючи, що записи БД повертаються як об'єкти, то змінюємо скрипти виду.

resources/views/app/groups/group.blade.php

```
@extends('app.layouts.layout')

@section('page_title')
    <b>Інформація про групу {{ $group->title }}</b>
@endsection

@section('content')
    <p>Шифр - {{ $group->title }}</p>
    <p>Курс - {{ $group->kurs }}</p>
    <p>Форма навчання - {{ $group->state }}</p>

    <a href="/groups">Дивитися всі групи</a>
@endsection
```

resources/views/app/groups/list.blade.php

```
@extends('app.layouts.layout')
```

```

@section('page_title')
    <b>Список груп</b>
@endsection

@section('content')
    <form method="get" action="/groups">
        <select name="type">
            <option value="0">всі категорії</option>

            @foreach($group_types as $type_id => $type_title)
                <option value="{{ $type_id }}"
                    {{ ( $type_id == $type_selected ) ? 'selected' : '' }}>
                    {{ $type_title }}
                </option>
            @endforeach
        </select>

        <select name="kurs">
            <option value="0">всі курси</option>

            @foreach(array(1, 2, 3, 4, 5, 6) as $kurs_id)
                <option value="{{ $kurs_id }}"
                    {{ ( $kurs_id == $kurs_selected ) ? 'selected' : '' }}>
                    {{ $kurs_id }}
                </option>
            @endforeach
        </select>

        <input type="submit" value="Знайти" />
    </form>

    <table>
        <th>Шифр групи</th>
        <th>Курс</th>
        <th>Форма навчання</th>
        @foreach ($groups as $group)
            <tr>
                <td>
                    <a href="/group/{{ $group->group_id }}">{{ $group->title }}</a>
                </td>
                <td>{{ $group->kurs }}</td>
                <td>{{ $group->state }}</td>
            </tr>
        @endforeach
    </table>
@endsection

```

## Робота з БД – ORM

Створюємо модель Group

```
php artisan make:model Group
```

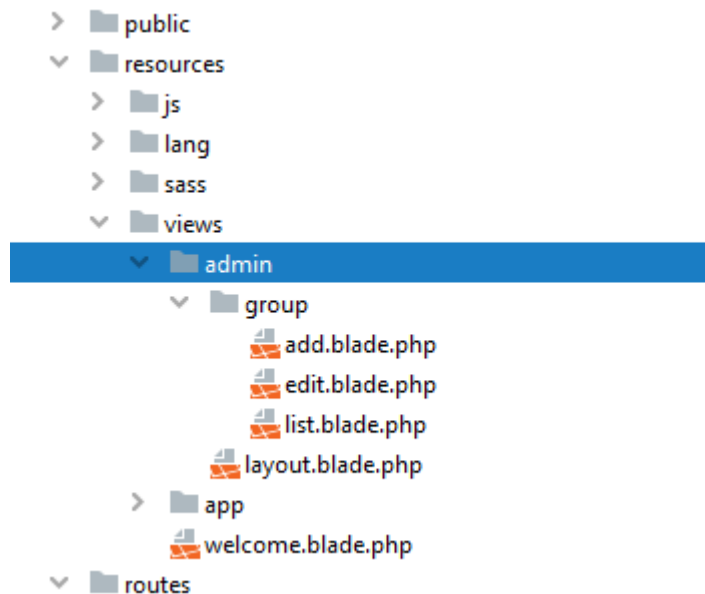
Створюємо контролер ресурсів AdminGroupController

```
php artisan make:controller AdminGroupController --resource
```

Створюємо роутер для контролеру ресурсів AdminGroupController

```
Route::resource('/admin/groups', 'AdminGroupController');
```

Створюємо окрему папку для шаблону і скриптів виду Адміністратора



Реалізовуємо методи AdminGroupController, використовуючи модель Group

```
use App\Group;
use App\Groups;
use Illuminate\Support\Facades\Redirect;

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function index()
{
    $groups = Group::get();
    return view('admin.group.list', ['groups' => $groups]);
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    return view('admin.group.add', ['groups_types' => Groups::$groups_types]);
}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $title = $request->input('title');
    $kurs = $request->input('kurs');
    $type = $request->input('type');

    $group = new Group();
    $group->title = $title;
    $group->state = Groups::$groups_types[$type];
    $group->kurs = $kurs;
    $group->type = $type;

    $group->save();

    return Redirect::to('/admin/groups');
}
```

```

/**
 * Show the form for editing the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    $group = Group::where('group_id', $id)->first();

    return view('admin.group.edit', [
        'group' => $group,
        'groups_types' => Groups::$groups_types]);
}

/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    $group = Group::where('group_id', $id)->first();

    $group->title = $request->input('title');
    $group->kurs = $request->input('kurs');
    $group->type = $request->input('type');

    $group->state = $group->state = Groups::$groups_types[$group->type] ;

    $group->save();
    return Redirect::to('/admin/groups');
}

/**
 * Remove the specified resource from storage.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    Group::destroy($id);
    return Redirect::to('/admin/groups');
}

```

## Створюємо шаблон панелі адміністратора

resources/views/admin/layout.blade.php

```
<h1>Адмінка</h1>

<div class="leftnav" style="float: left; width: 150px; height: 100%; margin-right:
30px;">
  <b>Групи</b>
  <ul>
    <li><a href="/admin/groups">список</a></li>
    <li><a href="/admin/groups/create">додати</a></li>
  </ul>
  <br/>

  <b>Кабінет</b>

  <ul>
    <li><a href="/logout">Logout</a></li>
  </ul>

  <ul>
    <li><a href="/groups">Frontend</a></li>
  </ul>
</div>

<div>
  @yield('content')
</div>
```



resources/views/admin/group/add.blade.php

```
@extends('admin.layout')

<style type="text/css">
    label {
        min-width: 150px;
        display: inline-block;
    }
</style>

@section('content')
    <h2>Додати групу</h2>

    <form action="/admin/groups" method="POST">
        {{ csrf_field() }}

        <label>Шифр групи </label>
        <input type="text" name="title">
        <br/><br/>

        <label>Курс </label>
        <input type="text" name="kurs">
        <br/><br/>

        <label>Форма навчання</label>
        <select name="type">
            @foreach($groups_types as $type_id => $type_title)
                <option value="{{ $type_id }}">
                    {{ $type_title }}
                </option>
            @endforeach
        </select>
        <br/><br/>

        <input type="submit" value="Зберегти">
    </form>

@endsection
```

resources/views/admin/group/edit.blade.php

```
@extends('admin.layout')

<style type="text/css">
    label {
        min-width: 150px;
        display: inline-block;
    }
</style>

@section('content')
    <h2>Редагування групи</h2>

    <form action="/admin/groups/{{ $group->group_id }}" method="POST">
        {{ method_field('PUT') }}
        {{ csrf_field() }}

        <label>Шифр групи</label>
        <input type="text" name="title" value="{{ $group->title }}">
        <br/><br/>

        <label>Курс</label>
        <input type="text" name="kurs" value="{{ $group->kurs }}">
        <br/><br/>

        <label>Форма навчання</label>
        <select name="type">
            @foreach($groups_types as $type_id => $type_title)
                <option value="{{ $type_id }}"
                    {{ ( $type_id == $group->type ) ? 'selected' : '' }}>
                    {{ $type_title }}
                </option>
            @endforeach
        </select>
        <br/>
        <br/>
        <input type="submit" value="Зберегти">
    </form>

@endsection
```

resources/views/admin/group/list.blade.php

```
@extends('admin.layout')

@section('content')
    <h2>Список груп</h2>

    <table border="1" style="text-align: center;">
        <th>Шифр групи</th>
        <th>Курс</th>
        <th>Форма навчання</th>
        <th>Дія</th>
        @foreach ($groups as $group)
            <tr>
                <td>
                    <a href="/group/{{ $group->group_id }}">{{ $group->title }}</a>
                </td>
                <td>{{ $group->kurs }}</td>
                <td>{{ $group->state }}</td>

                <td>
                    <a href="/admin/groups/{{ $group->group_id }}/edit">edit</a>

                    <form style="float:right; padding: 0 15px;"
                        action="/admin/groups/{{ $group->group_id }}"method="POST">
                        {{ method_field('DELETE') }}
                        {{ csrf_field() }}
                        <button>Delete</button>
                    </form>
                </td>
            </tr>
        @endforeach
    </table>

@endsection
```

### Контрольні запитання

1. Принцип роботи програм з MVC-архітектурою
2. Алгоритм роботи менеджера залежностей Composer
3. Структура директорій Laravel фреймворку
4. Маршрутизація web-сторінок в Laravel-проекті
5. Створення шаблонів і скриптів, шаблонізатор Blade
6. Отримання даних користувача в контролері
7. Передача даних з контролера в скрипт виду
8. Конструктор SQL-запитів та технологія ORM

## **Літературні джерела**

1. Офіційна документація PHP – Режим доступу <http://php.net/manual/ru/>
2. Офіційна документація MySQL – Режим доступу <https://dev.mysql.com/doc>
3. Офіційна документація Laravel – Режим доступу <https://laravel.com>
4. PHP Tutorial – Режим доступу <https://www.w3schools.com/php/default.asp>
5. MySQL Tutorial – Режим доступу <https://www.w3schools.com/sql/default.asp>