1.1 Лабораторна робота 6. Розробка сайту в середовищі WordPress.

Завдання: Розробити локально в середовищі WordPress умовний сайт курсів навчання з наданням послуг Навчання, Консультації, Розробка. В якості теми застосувати тему Sydney (http://athemes.com/theme/sydney/).

Хід роботи.

Передумови
Комп'ютер, з'єднаний з Інтернет.
XAMPP в якості локального сервера.
PHP версії 5.2.4 або більше.
MySQL версії 5.0 або більше.
Пакет WordPress (https://wordpress.org/ або https://uk.wordpress.org/).

1) Установка WordPress:
STEP1 Встановлюємо локальний сервер (XAMPP).
STEP2 Стартуємо сервер.
STEP3 Створюємо директорію xampp/htdocs/myblog і копіюємо в неї вміст архіву пакета WordPress (файл wordpress-4.6.1-uk.zip).
STEP4 Відкриваємо phpMyAdmin і створюємо базу даних my_first_blog (with utf8_general_ci).
STEP5 Відкриваємо WordPress в браузері (http://localhost/myblog) і робимо відповідні на лаштунки (Рис.59)

Рис.59. П'ятий крок установки WordPress

STEP6 Run and Install.
STEP7 Задаємо параметри облікового запису адміністратора (Рис.60) localhost/myblog/wp-login.php).

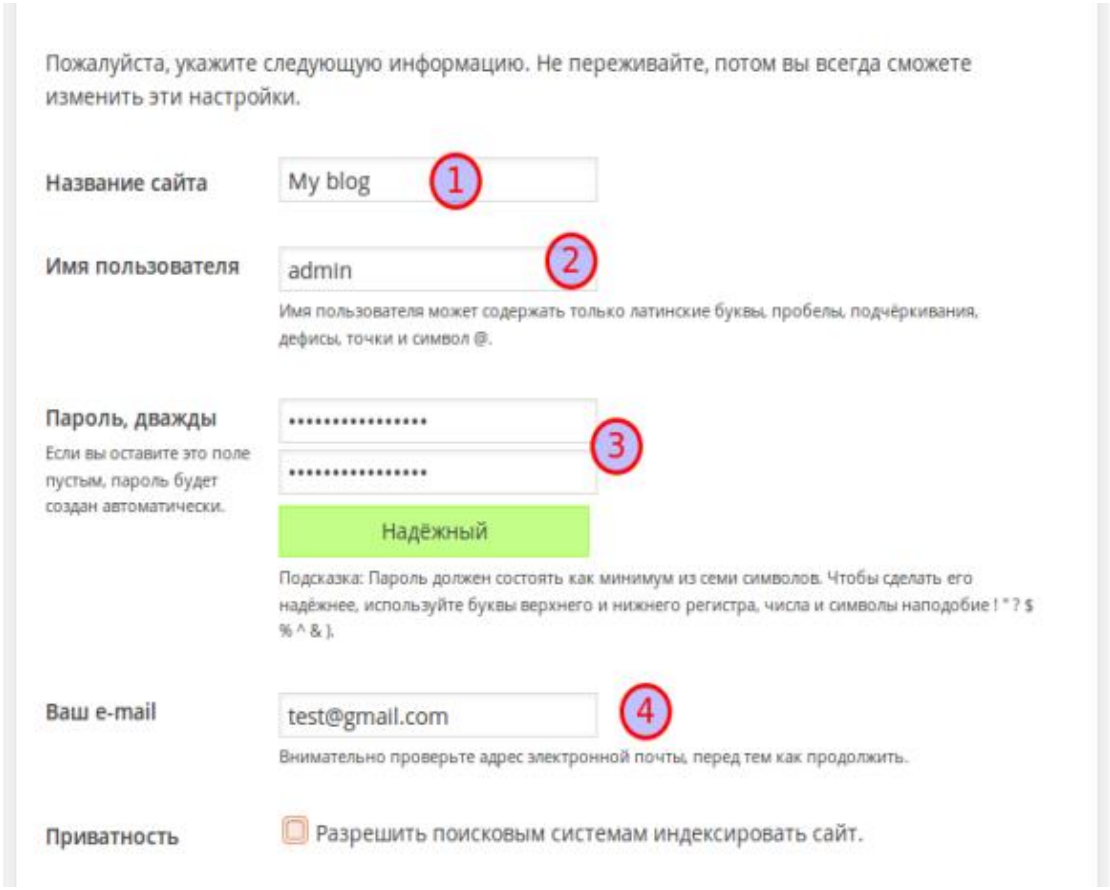

Рис.60. Задання параметрів облікового запису адміністратора

STEP8 Після того, як WP встановлено, для входу в адмін-панель використовуйте адресу http://localhost/myblog/wp-admin (вводячи логін і пароль, задані вами в ході установки WP)

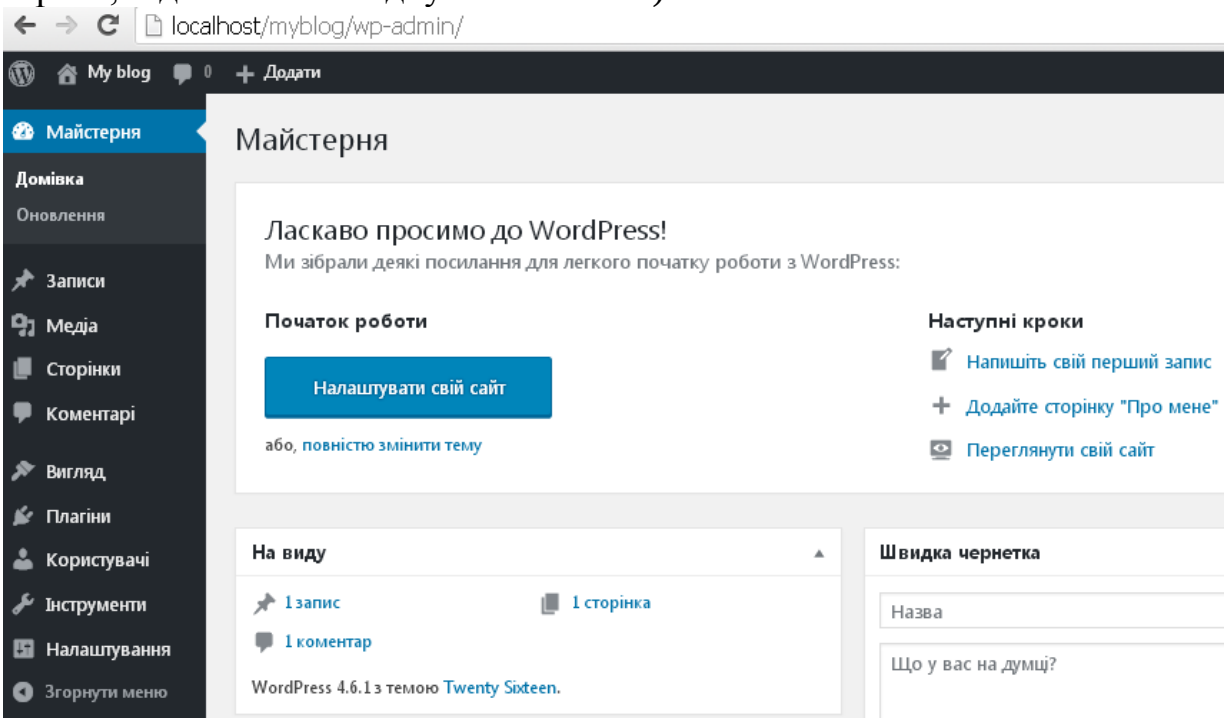

Рис.61. Панель адміністратора

Фронтальна частина буде доступна за адресою http://localhost/myblog/

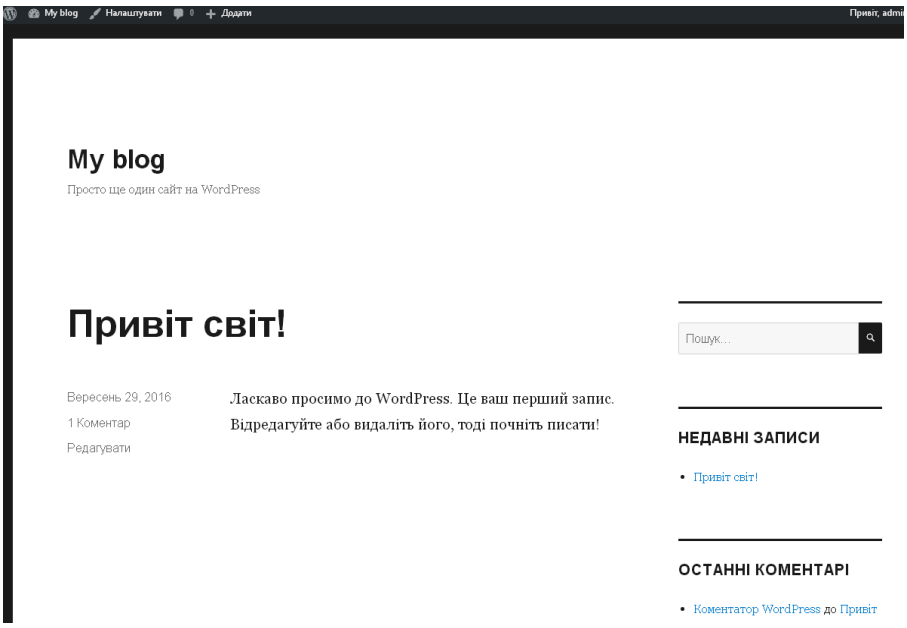

Рис.62. Фронтальна частина сайту

2) Налаштування WordPress після установки.

STEP1 Налагоджуємо людино-зрозумілі посилання (“человеко-понятные урлы - ЧПУ”). Вони використовуються для більш зрозумілого читання URL’ів, як людьми, так і пошуковими машинами. Отже, переходимо в Адмін-панель (*Майстерня)/Налаштування/Постійні посилання* і вмикаємо “*Назва запису*”. Далі встановлюємо пагін **Cyr-To-Lat**. Даний плагін прописує в URL заголовок статті латинськими літерами.

Отже, переходимо в *Адмін-панель/Плагіни/Додати/*, в пошуковому рядку вводимо назву плагіну, завантажуємо і встановлюємо його.

Зауваження. При роботі в корпоративній мережі кафедри ПЗАС ЗДІА треба налаштувати WordPress на роботу з проксі. Для цього необхідно додати в файл wp-config.php два рядки:

define(WP_PROXY_HOST, '192.168.1.1');

define(WP_PROXY_PORT, 3128);

STEP2 Додаємо плагін **Loco Translate**. Це найкращий плагін, призначеним для локалізації (перекладу) тем WordPress, він вбудовується безпосередньо в панель управління системою WordPress, не вимагаючи при цьому використання зовнішніх редакторів, на зразок Poedit.

3) Створення структури сайту. Нехай наш сайт буде складатися з наступних розділів: Головна, Мій блог, Фотогалерея, Про мене, Контакти. Розділи Про мене та Контакти будуть статичними.

STEP1 Створення/редагування статичних сторінок WP. Перейдіть на адмін-панель і виберіть в меню зліва Сторінки/Додати (буде відкрита сторінка редактора). Введіть заголовок сторінки (“Про мене”),

основний текст та, якщо потрібно, зображення. Для збереження змін натисніть Оприлюднити. Далі натисніть Переглянути сторінку.

STEP2 Блог. Категорії та Записи. Блог складається з Постів = статей, які можна згрупувати по Категоріях. В меню зліва на адмін-панелі виберемо Записи/Категорії. Створимо дві категорії: Блог і Фотогалерея. Щоб створити записи Блога перейдіть по посиланню “Записи/Додати” в меню зліва. Для кожного запису треба вибрати Формат (Стандартний, Примітка, Зображення, Відео, Цитата, Посилання, Галерея, Статус, Аудіо, Чат) і Категорію. Додайте декілька записів, вибираючи для Блога формат Стандартний, а для Фотогалереї – формат Зображення.

STEP3 Створення і управління меню. Коли всі необхідні сторінки та категорії створено, ми можемо створити меню – навігаційний список розділів нашого сайту. Для цього перейдемо по посиланню “Вигляд/Меню” в лівій панелі. Для створення нового меню слід дати назву (наприклад, “Головне меню”) і натиснути “Створити меню”. Після чого ви зможете додавати в ваше меню посилання на відповідні Розділи. Список можливих варіантів буде на лівій панелі. Після додавання потрібних пунктів меню, потрібно задати для меню область розташування. В нашому випадку ми повинні розмістити в меню 5 пунктів згідно завданню: Головна, Мій блог, Фотогалерея, Про мене, Контакти.

4) Підключення теми Sydney. Як підключити тему? Через меню Майстерня (Адмін-панель, Консоль)/Вигляд/Теми. За замовчуванням, діючою темою є тема Twenty Sixteen. Якщо стандартні теми вас не влаштовують, ви можете завантажити та встановити сторонню тему, натиснувши Додати нову. Тему треба попередньо завантажити і помістити в каталог …\xampp\htdocs\myblog\wp-content\themes. Після активації теми її треба налаштувати, щоб вміст вашого сайту відображався належним чином. Для цього перевірте ваші меню: можливо потрібно задати нові області призначення. Після зміни теми також потрібно налаштувати віджети.

STEP1 Скачуємо тему Sydney (http://athemes.com/theme/sydney/) і поміщаємо її в каталог myblog\wp-content\themes.

STEP2 Підключаємо тему Sydney. Під час активації теми потрібно додатково встановити плагіни Sydney Toolbox (або Types в старих версіях) та Page Builder.

STEP3 Далі потрібно на основі Sydney створити дочірню тему (Child Theme). Це дасть змогу редагувати тему, пристосувати її до ваших потреб. Для цього потрібно інсталювати плагін child theme (Плагіни/Додати, в рядку пошуку: “child theme”, в списку знайдених плагінів вибрати Child Theme Configurator, встановити і активувати

його). Далі вибираємо Інструменти/Child Themes. В пункті 2 виберемо тему Sydney.

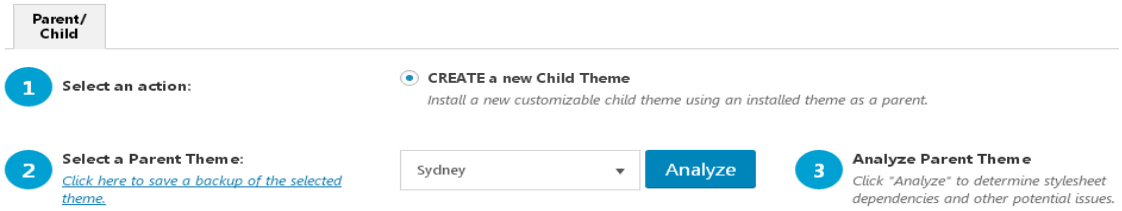

Рис.63. Підключення теми Sydney

Після натискання Analyze з'являються ще пункти 4 – 9. Погоджуємося з налаштуваннями за замовчуванням, в пункті 7 вводимо Theme Website - http://localhost/myblog/theme/sydney. Ставимо "галочку" в пункті 8 і натискаємо Create New Child Theme. Далі переходимо в Вигляд/Теми і бачимо в списку тему Sydney Child. Активуємо тему Sydney Child. Тепер її можна модифікувати.

STEP4 Зміна зображень та тексту в Sydney. Попередньо потрібно підготувати 2 зображення (1920x1275) та одне зображення (1280x549) для заміни титульних зображень в Sydney. Нові зображення повинні відповідати тематиці вашого сайту. Відвідуємо наш сайт (http://localhost/myblog/). Натиснемо "Налаштувати". Натиснемо "Область заголовка", далі "Тип заголовка". Вибираємо Тип заголовка сайта "Без заголовка (тільки меню)". Натиснемо "Зберегти і оприлюднити" та "<". Натиснемо "Слайдер заголовка" потім "Змінити зображення". Натиснемо "Завантаження файлів", вибираємо файл і завантажуємо, далі натиснемо "Обрати зображення". Задамо заголовок слайду "Ласкаво просимо до нашого блогу". Для другого слайду теж змінюємо зображення. Задамо заголовок слайду "Готові почати навчання?". Натиснемо "Зберегти і оприлюднити" та "<". Натиснемо "Зображення в заголовку". Далі можемо змінити поточний заголовок, а можемо сховати зображення, щоб звільнити місце для основного вмісту сторінок. Натиснемо "<" і ще раз "<". Далі можна відредагувати "Назву сайту, ключову фразу, логотип", "Шрифти", "Кольори", "Фонове зображення" і т.д. Далі заходимо в "Параметри блогу", в "Розміщенні блогу" замість "Класично" вибираємо "Сітка". В розділі "Довжина уривка" замість "55" вибираємо "20". Натиснемо "Зберегти і оприлюднити" та "<". Далі заходимо в "Підвал", в Області віджетів підвалу вибираємо "Один" замість "Три". Заходимо в "Кольори", "Фон меню" – "Обрати колір", набираємо "#c9c9c9" (це сірий фон), "Текст Body" – "Обрати колір", вибираємо чорний (#000000). "Колір бічної колонки (sidebar)" – "Обрати колір", чорний. "Зберегти і оприлюднити" та "<".

STEP5 Створення логотипу сайта. Заходимо на https://logomakr.com/ . Натискаємо "Next" і знову "Next" і далі "DONE". Натискаємо "Browse All Logos". Можемо організувати пошук логотипів за темою (people, book, laptop, . . .). Обравши логотип, задаємо колір, напис, орієнтацію,

розмір, і зберігаємо логотип. Далі переходимо на наш сайт, натискаємо “Налаштувати”, “Назва сайту, ключова фраза, логотип”, в розділі “Завантажити свій логотип” натискаємо “Виберіть зображення”, “Завантаження файлів”, “Обрати файли”, обираємо файл нашого логотипу і натискаємо “Обрати зображення”, а далі “Зберегти і оприлюднити”.

STEP6 Редагування теми Sydney. Далі треба діяти згідно документації (http://athemes.com/documentation/sydney/). Створення статичної головної сторінки. Додати/Сторінку, даємо ім’я, наприклад, “Моя головна сторінка”, Шаблон: Головна сторінка (вибрати із списка), Оприлюднити. Додати/Сторінку, “My blog page”, Оприлюднити. Налаштування сайту. Відвідати сайт (http://localhost/myblog/). Натиснути “Налаштувати”. Головна сторінка відображає: “Статична сторінка”. Статична головна сторінка/ Головна сторінка/ Моя головна сторінка (вибрати із списка), Сторінка записів/ My blog page (вибрати із списка). Натиснути “Збережено”.

STEP7 Редагування теми Sydney. Створення послуг Навчання, Консультації, Розробка. Після встановлення плагіну Sydney Toolbox ви можете створювати Послуги, Співробітники, Клієнти та Відгуки (Services, Employees, Testimonials and Clients – доступні в Майстерні) – Дивись Демо (http://athemes.com/documentation/sydney/). Давайте створимо на головній сторінці щось на кшалт такого:

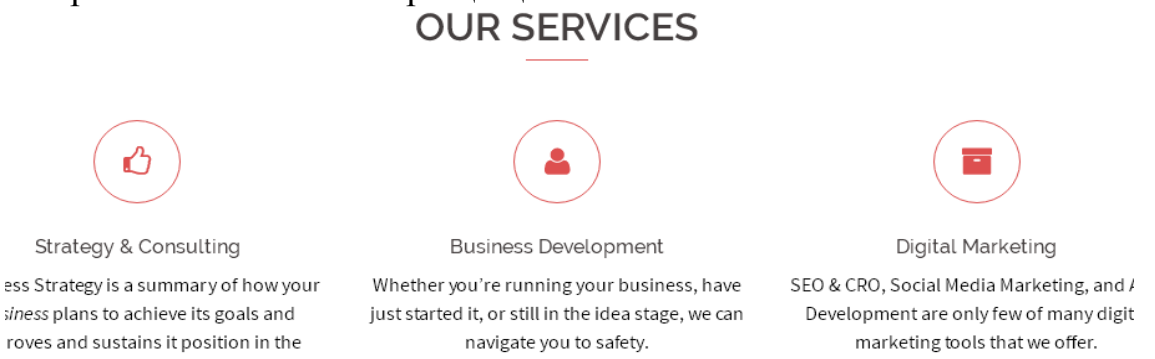

Рис.64. Створення послуг

Додавання послуг. Майстерня/ Services/Add New Service. Даємо назву, наприклад, “Розробка”. В вікні редактора вводимо “Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.”. В розділі “Service Icon” натискаємо “here” і підбираємо, наприклад, “fa-github”. В розділі “Service link” вводимо http://localhost/myblog/development (цю сторінку треба потім додати). Далі можна натиснути “Додати нову категорію” і дати їй ім’я, наприклад, “Розробка”. Далі натиснути “Оприлюднити”. Add New Service. Даємо назву, наприклад, “Навчання”. В вікні редактора вводимо “Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.”. В розділі “Service Icon” – “fa-laptop”. В розділі “Service link” вводимо http://localhost/myblog/education. Приєднуємо до категорії “Розробка”.

Далі натискаємо “Оприлюднити”. Add New Service. Даємо назву, наприклад, “Консультації”. В вікні редактора вводимо “Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.”. В розділі “Service Icon” – “fa-university”. В розділі “Service link” вводимо http://localhost/myblog/consult. Приєднуємо до категорії “Розробка”. Далі натискаємо “Оприлюднити”.

STEP8 Sydney. Редагування головної сторінки. В Майстерні виберемо “Сторінки”, “Всі сторінки”, виберемо для редагування “Моя головна сторінка”. Натиснемо “Page Builder”. Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Тиснемо “Add Widget”, “Віджети теми Sydney”, “Sydney FP: Services Type A”, наводимо курсор миші на область Sydney FP: Services Type A і тиснемо “Edit”. В полі “Заголовок” набираємо “Наші послуги”, в полі “Введіть слаг вашої категорії” наберіть “Розробка” і натисніть кнопку “Done”, а далі натисніть “Оновити” і “Переглянути сторінку” – ми побачимо на головній сторінці секцію “НАШІ ПОСЛУГИ” (Консультації, Навчання, Розробка). Зменшимо верхній і нижній відступ секції “НАШІ ПОСЛУГИ”. Для цього тиснемо “Редагувати сторінку” (повертаємось на Головну сторінку). Перед зоною Sydney FP: Services Type A наводимо курсор миші на зображення гаєчного ключа і тиснемо “Edit Row”, далі тиснемо “Theme”, в полі “Верхній і нижній відступ” задаємо 50. Далі тиснемо “Done” потім “Оновити” і “Переглянути сторінку”.

STEP9 Редагування головної сторінки. Додамо на головну сторінку ще одну секцію, наприклад, “Чим ми займаємося” (What We Do). Для цього тиснемо “Редагувати сторінку” (повертаємось на Головну сторінку). Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Клікаємо на новій секції (щоб зробити активною), тиснемо “Add Widget”, виберемо внизу списку віджет “Текст”. В секції “Текст” тиснемо “Edit”, в полі “Назва”вводимо “Чим ми займаємося” і тиснемо “Done”. В Майстерні наводимо курсор на Записи/Додати, тиснемо праву кнопку миші і виберемо “Open link on the new tab”. Поле заголовку залишаємо пустим, а в зоні вмісту вводимо, наприклад, “Ми даємо можливість отримати професію програміста” та “Приєднуйтесь!”. Виділяємо ці два рядки, на панелі інструментів в випадаючому списку “Параграф” виберемо “Заголовок 3” і Вирівнювання по центру. Далі в редакторі переходимо в режим Тексту, копіюємо в буфер html-код (<h3 style="text-align: center;">Ми даємо можливість отримати професію програміста.</h3><h3 style="text-align: center;">Приєднуйтесь!</h3>). Повертаємось на Головну сторінку, в секції “Текст” тиснемо “Edit”, вставляємо текст з буфера і тиснемо “Done”. Далі будемо редагувати секцію “Текст”. Перед зоною “Текст” наводимо курсор миші на зображення гаєчного ключа і тиснемо “Edit Row”, далі тиснемо “Theme”, в полі “Верхній і нижній відступ” задаємо

60. В полі “Колір” тиснемо “Обрати колір” і виберемо білий колір (@ffffff). В полі “Фонове зображення” тиснемо “Select Image”, “Завантаження файлів”, “Обрати файли” і т.д. В полі “Розміщення розділу” виберемо “На всю ширину” і далі тиснемо “Done” і “Оновити”. Переглянемо сторінку.

STEP10 Додавання на Головну сторінку ще двох секцій типу “Call To Actions” (з кнопками). Переходимо до режиму редагування Головної сторінки. Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Нова секція з’явиться між двома попередніми. Наводимо курсор миші на значок поряд з гаєчним ключем і перетягуємо секцію вниз (після секції “Текст”). Наводимо на гаєчний ключ і тиснемо “Edit Row”, далі “Theme”, в полі “Верхній і нижній відступ” задаємо 50. В полі “Фонове зображення” тиснемо “Select Image”, “Завантаження файлів”, “Обрати файли” і т.д. В полі “Розміщення розділу” виберемо “На всю ширину” і далі тиснемо “Done”. Тиснемо “Add Widget”, виберемо “Layout Builder”. Далі тиснемо “Edit”, потім тиснемо “Add Row”, виберемо 1 (стовбець), тиснемо “Insert”. Далі тиснемо “Add Row”, виберемо 2 (стовбця), тиснемо “Insert”. Робимо активним верхній рядок і тиснемо “Add Widget”, виберемо “Текст”. В рядку “Текст” тиснемо “Edit”, в полі “Назва”вводимо “УЗНАЙТЕ БІЛЬШЕ ПРО НАШІ КУРСИ І ПОСЛУГИ” і тиснемо “Done”. В рядку “Текст” наводимо на гаєчний ключ і тиснемо “Edit Row”, далі “Theme”, в полі “Верхній і нижній відступ” задаємо 0, в полі “Колір” обитаємо червоний (#dd3333) і тиснемо “Done”. В правому стовбці тиснемо “Edit”, в полі “Введіть свій заклик до дії” вводимо, наприклад, “Звертайтесь до нас за консультацією”, в полі “Посилання кнопки” вводимо http://localhost/myblog/contact-us/, в полі “Заголовок кнопки” вводимо “НАШІ КОНТАКТИ”. Далі тиснемо “Attributes”, в полі “CSS Styles” вводимо background-color: rgba(0,0,0,0.3); Тиснемо “Done”. В рядку з двох стовбців наводимо на гаєчний ключ і тиснемо “Edit Row”, далі “Theme”, в полі “Верхній і нижній відступ” задаємо 0, в полі “Колір” обитаємо червоний (#dd3333) і тиснемо “Done”. Тиснемо “Done”, потім тиснемо “Оновити”. Тепер можна переглянути наш оновлений сайт.

STEP11 Створення відео секції на головній сторінці. Переходимо до режиму редагування Головної сторінки. Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Нова секція з’явиться між попередніми. Наводимо курсор миші на значок поряд з гаєчним ключем і перетягуємо секцію вниз (після секції “Layout Builder”). Тиснемо на секцію, щоб зробити її активною. Тиснемо “Add Widget”, виберемо “Віджети теми Sydney”, “Sydney Video”. Наводимо на гаєчний ключ і тиснемо “Edit Row”, далі “Theme”, в полі “Верхній і нижній відступ” задаємо 30, в списку “Розміщення розділу” виберемо “На всю ширину” і тиснемо “Done”. Далі переходимо до редагування нашого віджету.

Тиснемо “Edit”. В полі “Вставте посилання відео” вставимо посилання, наприклад, https://www.youtube.com/watch?v=wej0v-7deoQ. Тиснемо “Layout”, в полі “Padding” вводимо 20. Тиснемо “Done”. Тиснемо “Оновити”. Далі можна переглянути оновлену головну сторінку (з відео).

STEP12 Створення секції для відгуків. Переходимо до режиму редагування Головної сторінки. Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Нова секція з’явиться між попередніми. Наводимо курсор миші на значок поряд з гаєчним ключем і перетягуємо секцію вниз (після секції “Sydney Video”). Тиснемо на секцію, щоб зробити її активною. Тиснемо “Add Widget”, виберемо “Віджети теми Sydney”, “Sydney FP: Call to action”. Перед редагуванням віджету відредагуємо рядок. Тиснемо “Edit Row” далі “Theme”, в полі “Верхній і нижній відступ” задаємо 30, в полі “Фонове зображення” тиснемо “Select Image”, “Завантаження файлів”, “Обрати файли” і т.д. В списку “Розміщення розділу” виберемо “На всю ширину” і тиснемо “Done”. Тепер переходимо до редагування віджету “Sydney FP: Call to action”. Тиснемо “Edit”. В полі “Введіть свій заклик до дії” вводимо, наприклад, “НАШІ ВИПУСКНИКИ ГОТОВІ ПОДІЛИТИСЯ ВРАЖЕННЯМИ”, в полі “Посилання кнопки” вводимо http://localhost/myblog/success-stories, в полі “Заголовок кнопки” вводимо “ВІДГУКИ”. Ставимо галочку в пункті “Відображати кнопку врівень із текстом?”. Тиснемо “Layout”, в полі “Padding” вводимо 0. Тиснемо “Done”. Тиснемо “Оновити”. Далі можна переглянути оновлену головну сторінку.

1.2 Лабораторна робота 7. Розробка сайту за допомогою фреймворку Laravel.

Завдання: Розробити в середовищі Laravel простий сайт, що працює з машинами (клас Car з полями make, model, produced_on). Передбачити перегляд, редагування, додавання та видалення інформації.

Теоретичні відомості, методичні вказівки та хід роботи.

Передумови
Комп’ютер, з’єднаний з Інтернет.
XAMPP 5.6.28 в якості локального сервера.
Laravel 5.2.
Пакетний менеджер Composer.

1) Інсталяція Laravel 5.2 на XAMMP

КРОК1. Встановлення локального сервера (XAMPP 5.6.28)

КРОК2. Встановлення пакетного менеджера Composer (завантажити з сайту https://getcomposer.org/Composer-Setup.exe). В ході інсталяції треба

вказати місцезнаходження файлу php.exe. Якщо інсталювати на Обчислювальних Центрах ЗДІА, то вказати адресу proxy-сервера (http://192.168.0.222:3128).

КРОК3. Відкрийте термінал Windows (cmd.exe) перейдіть в каталог xampp/htdocs і подайте команду: `composer create-project laravel/laravel mysite "5.2.*"`. Після закінчення буде створено каталог `xammp/htdocs/mysite` з наступним вмістом (Рис.65):

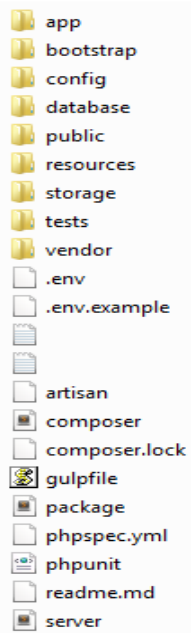

Рис.65. Структура каталогів застосунку

КРОК4. Активізуйте Apache і MySql з панелі управління XAMPP.

КРОК5. Наберіть в браузері: `localhost/mysite/public` (повинен з'явитись екран Laravel 5).

Наступні кроки 6,7,8,9 бажано виконати після налаштування бази даних застосунку (дивись далі).

КРОК6. Для подальшої зручності в роботі бажано сконфігурувати віртуальний хост в XAMMP. Для цього треба відкрити в редакторі (наприклад Notepad++) файл `C:\xampp\apache\conf\extra\httpd-vhosts.conf` (якщо у вас XAMMP на диску C:\) і додати наступні рядки в кінець файлу:

```
# VirtualHost for LARAVEL.DEV
    <VirtualHost  *:80>
            ServerAdmin webmaster@dummy-host2.laravel.dev
        DocumentRoot "C:/xampp/htdocs/mysite/public"
        ServerName  laravel.dev
        ErrorLog  "logs/laravel.dev-error.log"
        CustomLog  "logs/laravel.dev-access.log" common
    </VirtualHost>
```

Після цього наш Apache буде прослуховувати `laravel.dev` з'єднання.

КРОК7. Залишилось внести деякі зміни в системний файл `C:\Windows\System32\drivers\etc\hosts`. Додайте в файл hosts рядок:

```
127.0.0.1  laravel.dev
```

КРОК8. (Тільки на Обчислювальних Центрах ЗДІА). Включіть адресу `laravel.dev` в виключення проксі-сервера в налаштуваннях вашого браузера.

КРОК9. Наберіть в браузері: `laravel.dev` (повинен з'явитись екран Laravel 5).

2) Налаштування фреймворку.

Відкрийте файл .env і відредагуйте деякі рядки:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=mylaravel
DB_USERNAME=root
DB_PASSWORD=
```

Увійдіть в phpMyadmin (http://localhost/phpmyadmin) і створіть базу даних mylaravel (з utf8_general_ci).

3) Побудова простого застосунку. Для демонстрації можливостей Laravel побудуємо застосунок, що працює з машинами.

Модель Car

Laravel має вбудований в інтерфейс командного рядка ремісник (Artisan), який надає вам багато корисних команд для побудови додатка.

КРОК1. Відкрийте термінал Windows (cmd.exe) перейдіть в каталог застосунку (xampp/htdocs/mysite) і подайте команду для генерації нової моделі Car:

```
php artisan make:model Car --migration
```

Всі моделі зберігаються в каталозі app, так що попередня команда згенерує файл app/Car.php з наступним кодом:

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Car extends Model
{
    //
}
```

Завдяки вбудованій функціональності моделі Laravel, створивши порожній клас моделі, Laravel буде вважати, що ця модель пов'язана з таблицею бази даних з ім'ям cars.

Завдяки прапорцю --migration при створенні моделі, Laravel також генерує міграційний файл бази даних для створення таблиці бази даних (https://laravel.com/docs/5.2/migrations). Файл міграції database/migrations/[мітка часу] _create_cars_table.php містить наступний код:

```
. . . . . .
public function up() {
Schema::create('cars', function (Blueprint $table) {
    $table->increments('id');
    $table->timestamps();
}); }
```

. . . . . .

КРОК2. Додайте три додаткових стовпців, які будуть зберігати характеристики автомобілів (виробника, модель та дату виробництва):

```
$table->string('make');
$table->string('model');
$table->date('produced_on');
```

КРОК3. Подайте Artisan-команду для запуску міграції, що створить таблицю cars:

```
php artisan migrate
```

КРОК4. Додайте вручну декілька записів в таблицю автомобілів (cars) бази даних нашого застосунку (mylaravel). Для цього наберіть в браузері

localhost/phpmyadmin і т.д.

Контролер

В Laravel, тип об'єкта - такий, як Car, в даному випадку - називають в якості ресурсу (resource).

КРОК5. Створіть контролер ресурсів (resource controller) - контролер для обробки всіх запитів, пов'язаних з ресурсом Car - за допомогою іншої простої команди Artisan:

```
php artisan make:controller CarController
```

Це створить файл app/Http/Controllers/CarController.php контролера з відповідним кодом.

Маршрути (Routes)

На попередньому кроці Laravel автоматично згенерував контролер з усіма типовими CRUD операціями. Тепер нам просто потрібно визначити маршрути, щоб зв'язати URL-адреси з усіма цими діями контролера. Знову ж таки, оскільки це такий загальний сценарій, ви можете визначити єдиний маршрут (resource route), який створює маршрути для всіх дій контролера ресурсів.

КРОК6. У файлі конфігурації маршрутів – app/Http/routes.php — додайте наступне визначення маршруту для ресурсу Car:

```
Route::resource('cars', 'CarController');
```

Це єдине визначення маршруту буде визначати всі маршрути, пов'язані з нашим ресурсом Car (Рис.66):

Request Type	Path	Action	Route Name
GET	/cars	index	cars.index
GET	/cars/create	create	cars.create
POST	/cars	store	cars.store
GET	/cars/{car}	show	cars.show
GET	/cars/{car}/edit	edit	cars.edit
PUT/PATCH	/cars/{car}	update	cars.update
DELETE	/cars/{car}	destroy	cars.destroy

Рис.66. Маршрути застосунку

Тепер, наприклад, давайте здійснимо реалізацію сторінки Show Car (машини з конкретним id).

Метод дії show контролера

Згідно з попередньою таблицею, сторінка Show Car буде доступною за адресою: http://app.url/cars/{car} (в нашому випадку app.url – це laravel.dev). В цьому випадку {car} буде, насправді, ідентифікатором id об'єкта автомобіля в базі даних.

Так, наприклад, URL, щоб переглянути автомобіль з ідентифікатором 1 буде http://app.url/cars/1.

З метою реалізації сторінки Show Car, в show дії контролера, нам необхідно:

- Використати модель Car, щоб отримати відповідний об'єкт автомобіля з бази даних.
- Завантажити вид (view) для сторінки Show Car, і передати йому об'єкт Car, витягнутий з бази даних.

КРОК7. По-перше, для того, щоб отримати доступ до моделі Car в контролері, нам потрібно додати новий вираз вище класу контролера (файл app/Http/Controllers/CarController.php):

```
use App\Car;
```

КРОК8. Потім ми можемо завершити метод дії show наступним кодом:

```
public function show($id)
 {
     $car = Car::find($id);
     return view('cars.show', array('car' => $car));
 }
```

Вид (View)

Файли видів Laravel всі зберігаються в папці resources/views. І вони можуть бути організовані в підпапках в цьому каталозі. У попередньому кроці, ми послались на вид з назвою cars.show. Це говорить Laravel шукати файл виду, розташований в підпапці cars (всередині головного каталогу видів resources/views) під назвою show.blade.php.

Файли видів Laravel використовують шаблонний движок Blade, і, отже, називаються *.blade.php

КРОК9. Створіть каталог cars всередині каталогу resources/views

КРОК10. Створіть файл resources/views/cars/show.blade.php з наступним кодом:

```
<!DOCTYPE html>
<html>
<head> <title>Car {{ $car->id }}</title> </head>
<body>
<h1>Car {{ $car->id }}</h1>
     <ul>
         <li>Make: {{ $car->make }}</li>
         <li>Model: {{ $car->model }}</li>
         <li>Produced on: {{ $car->produced_on }}</li>
     </ul>
</body>
</html>
```

Метод дії index контролера

В цьому методі дії необхідно вивести всі автомобілі.

КРОК11. В файл app/Http/Controllers/CarController.php додайте метод дії index з наступним вмістом:

```
public function index()
 {
     $cars = Car::orderBy('id', 'desc')->paginate();
     return view('cars.index', compact('cars'));
 }
```

Вид для перегляду всіх автомобілів.

КРОК12. Створіть файл resources/views/cars/index.blade.php з наступним кодом:

```
@foreach($cars as $car)
<car>
     <h1>Car {{ $car->id }}</h1>
    <ul>
      <li>Make: {{ $car->make }}</li>
      <li>Model: {{ $car->model }}</li>
      <li>Produced on: {{ $car->produced_on }}</li>
    </ul>
</car>
<hr>
@endforeach
```

Виконайте кроки 6 і 7 по конфігуруванню віртуального хоста laravel.dev (слайди 7, 8, 9 даної презентації).

Наберіть в браузері: http://laravel.dev/cars/. Повинен з'явитись список всіх машин, що внесені в базу даних.

Наберіть в браузері: `http://laravel.dev/cars/1`

(повинна з'явитись інформація про першу машину)

КРОК13. Подальше вдосконалення застосунку.

Вивчіть документацію, потрібну для додавання в застосунок можливості додавання нових машин, редагування та видалення інформації.

1.3 Лабораторна робота 8. Розробка сайту за технологією Node+Express.

Завдання: В середовищі фреймворку Express розробити застосунок "PC Repair" з наступним інтерфейсом (Рис.67):

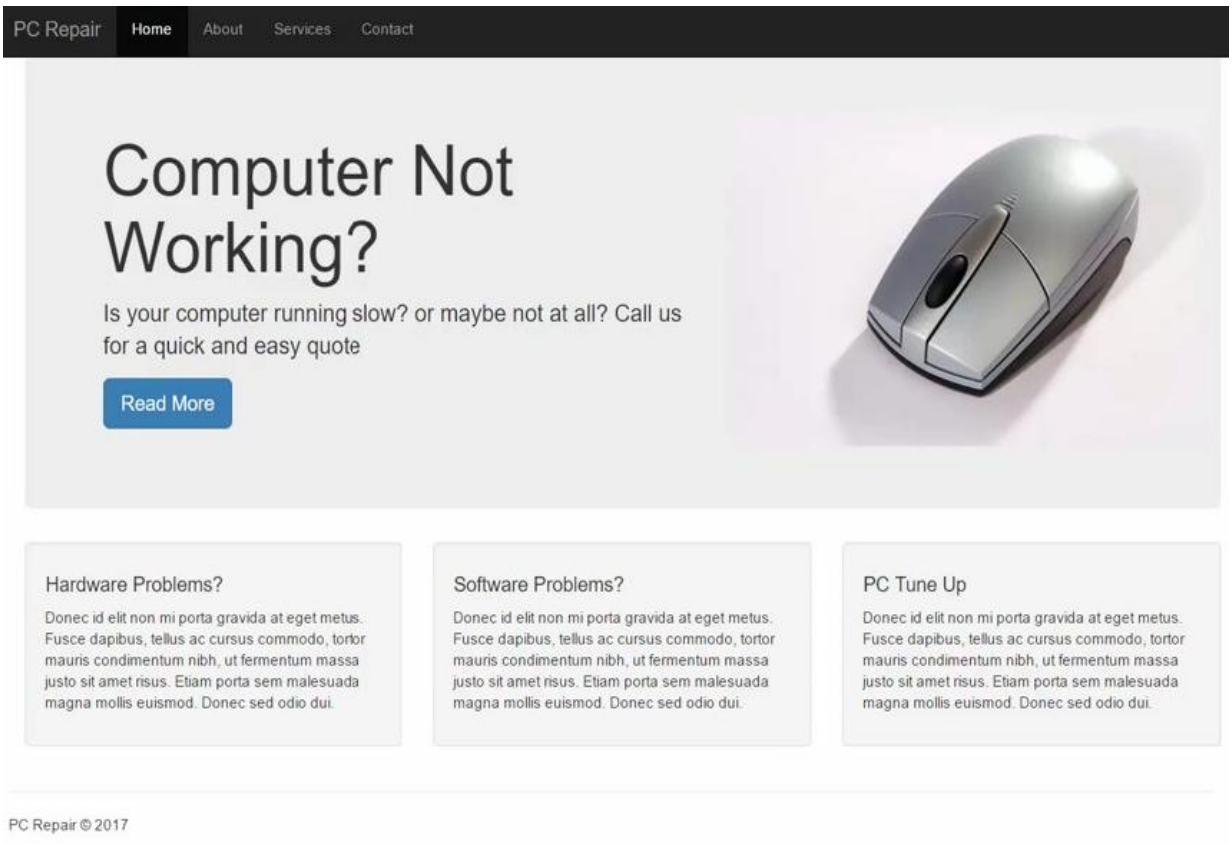

Рис.67. Інтерфейс застосунку

Хід роботи

1. Інсталювати Node.js (https://nodejs.org/)
2. Відкрити термінал (cmd.exe), перейти в каталог проекту.
3. Інсталювати Express
   - npm install express -g
4. Інсталювати express-generator
   - npm install express-generator -g
5. Згенерувати систему директорій проекту
   - express pcrepair --pug
6. Перейти в каталог проекту: cd pcrepair
7. Створити файл package.json
   - npm install
8. Завантажити директорію проекту (pcrepair) в редактор Sublime Text або Visual Studio Code та відкрити файл package.json, звернути увагу на рядок “start”: “node ./bin/www”
9. Запустити проект
   - npm start
   - В вікні браузера набрати localhost:3000
10. В редакторі відкрити файл app.js та уважно його вивчити.
11. Переглянути файли каталогу routes (index.js, users.js). Набрати в браузері localhost:3000/users і проаналізувати відповідь.
12. Переглянути файли каталогу views (error.pug, index.pug, layout.pug).
13. Інсталювати nodemon для автоматичного рестарту:

- npm install -g nodemon

14. Запустити nodemon: nodemon
15. Підготуємо шаблон титульної сторінки за допомогою Bootstrap. Заходимо на http://getbootstrap.com/, тиснемо Download і на наступній сторінці ще раз Download. Видобуваємо файли архіву до допоміжної папки.
16. В папку public/stylesheets нашого проекту копіюємо файл bootstrap.css.
17. В файл views/layouts.pug після 5-го рядка додаємо рядок “link(rel=‘stylesheet’, href=‘/stylesheets/bootstrap.css’)
18. Оновлюємо сторінку в браузері.
19. На сайті Bootstrap вибираємо в випадаючому списку v3.3.7, потім внизу пункт Examples, тиснемо на Starter template, переглядаємо вихідний код сторінки. Копіюємо в секції body тег навігації ‘<nav . . . </nav>’
20. Заходимо на сайт Jade Converter (http://html2jade.org/), вставляємо зліва скопійований фрагмент, в правій панелі отримуємо результат.
21. Копіюємо конвертований фрагмент навігації і вставляємо в секцію body файлу layout.pug, як показано на наступному слайді.

Рис.68. Файл layout.pug

22. Оновлюємо сторінку в браузері щоб побачити навігаційне меню.
23. Відредагуємо файл layout.pug. Змінемо “Project Name” на “PC Repair”, 20-й рядок на “a(href=‘/’) Home”, 22-й рядок на “a(href=‘/about’) About”. Після 22-го рядка вставимо рядок “li” та рядок “a(href=‘/services’) Services”. Підкорегуємо бувший 24-й рядок: “a(href=‘/contact’) Contact”. 19-й рядок представимо у вигляді: “li(class=title==‘Home’ ? ‘active’ : undefined)”.
24. Корегуємо файл layout.pug.

```
ul.nav.navbar-nav
  li(class=title=='Home' ? 'active' : undefined)
    a(href='/') Home
  li(class=title=='About' ? 'active' : undefined)
    a(href='/about') About
  li(class=title=='Services' ? 'active' : undefined)
    a(href='/services') Services
  li(class=title=='Contact' ? 'active' : undefined)
    a(href='/contact') Contact
```

Рис.69. Корегування файлу layout.pug

Перевіримо зміни в браузері.

25. Попрацюємо над головною сторінкою. На сайті http://html2jade.org в лівій колонці наберемо текст, подібний наступному

```
<div class="row">
  <div class="col-md-4">
    <h2>Hardware problems?</h2>
    <p>Donec id elit non mi porta gravida at eget metus </p>
  </div>
  <div class="col-md-4">
    <h2>Software problems?</h2>
    <p>Donec id elit non mi porta gravida at eget metus </p>
  </div>
  <div class="col-md-4">
    <h2>PC Tune Up</h2>
    <p>Donec id elit non mi porta gravida at eget metus </p>
  </div>
</div>
```

Рис. 70. Розмітка для файлу index.pug

26. Копіюємо праву колонку і вставляємо в файл index.pug

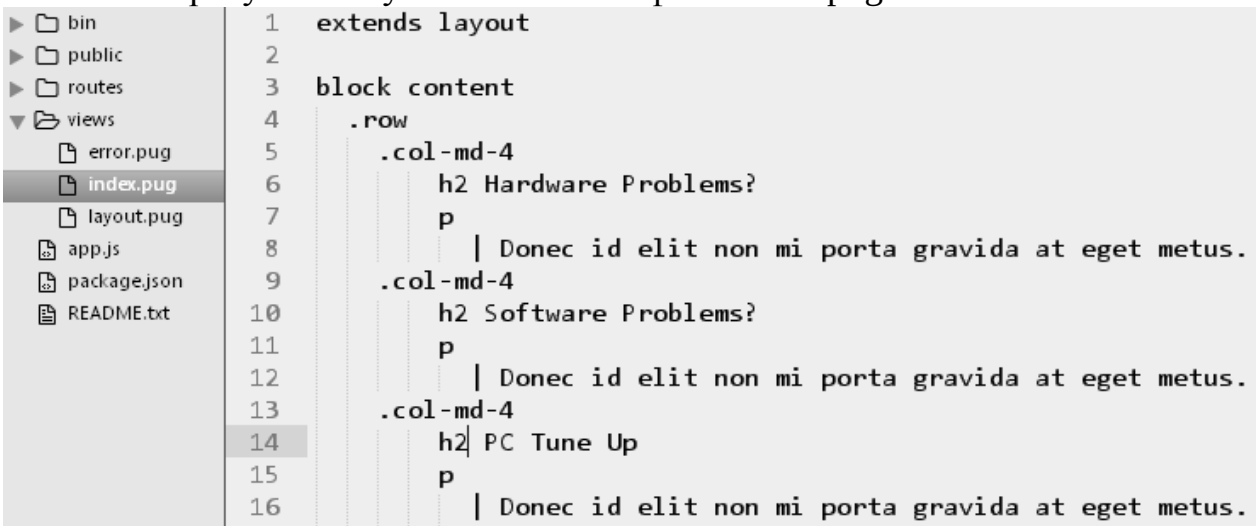

Рис.71. Файл index.pug

27. Оновлюємо браузер і бачимо на головній сторінці три колонки.

28. Трохи підкорегуємо головну сторінку (файл index.pug).

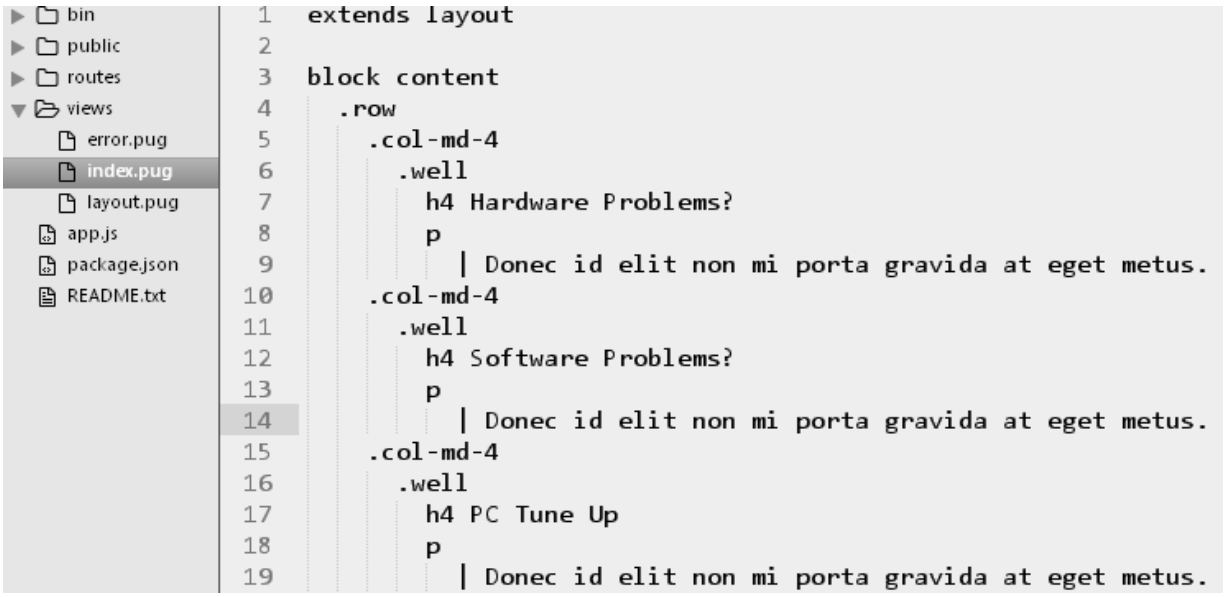

Рис.72. Корегування файлу index.pug

29. Оновлюємо браузер.
30. Для покращення вигляду вставляємо в файл layout.pug перед останнім рядком рядок ".container", оновлюємо браузер щоб побачити результат.
31. Вставимо на головну (index.pug) сторінку джамботрон.

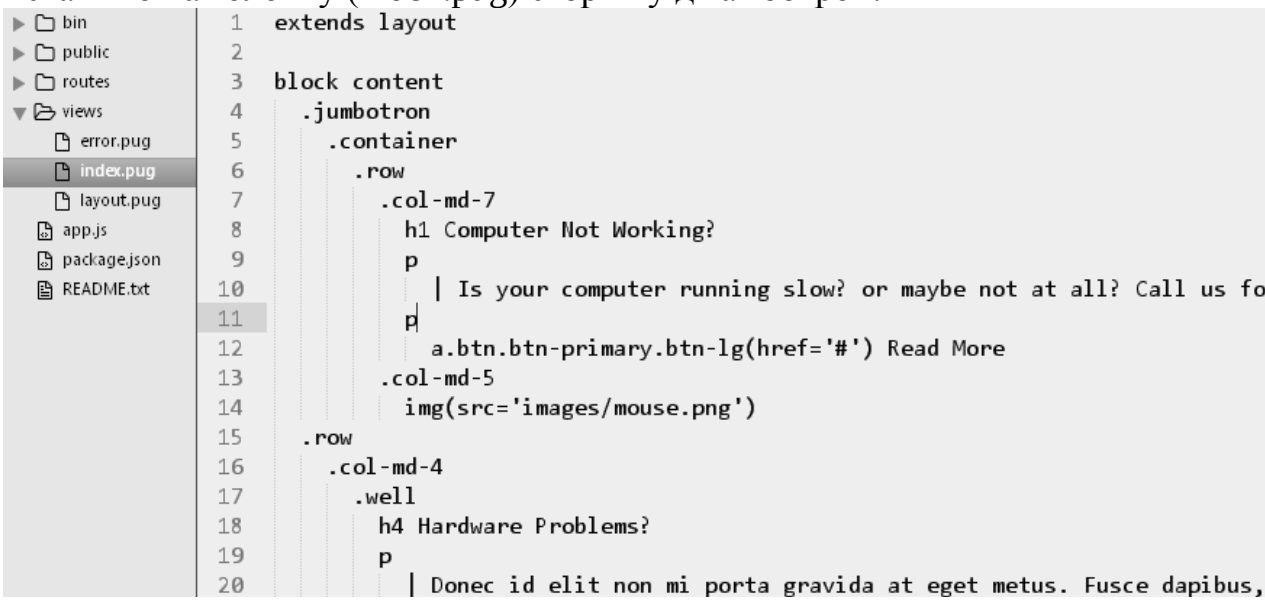

Рис.73. Вставка джамботрону

32. Всталяємо mouse.png в каталог public/images, оновлюємо браузер щоб побачити результат.
33. Додамо до шаблону сайту (layout.pug) footer і оновимо браузер.

```
12              span.sr-only Toggle navigation
13              span.icon-bar
14              span.icon-bar
15              span.icon-bar
16            a.navbar-brand(href='#') PC Repair
17          #navbar.collapse.navbar-collapse
18            ul.nav.navbar-nav
19              li(class=title=='Home' ? 'active' : undefined)
20                a(href='/') Home
21              li(class=title=='About' ? 'active' : undefined)
22                a(href='/about') About
23              li(class=title=='Services' ? 'active' : undefined)
24                a(href='/services') Services
25              li(class=title=='Contact' ? 'active' : undefined)
26                a(href='/contact') Contact
27      .container
28        block content
29
30      hr
31      footer
32        p PC Repair &copy; 2017
```

Рис.74. Додавання footer

34. Попрацюємо над пунктом About. В директорії routes створюємо файл about.js.

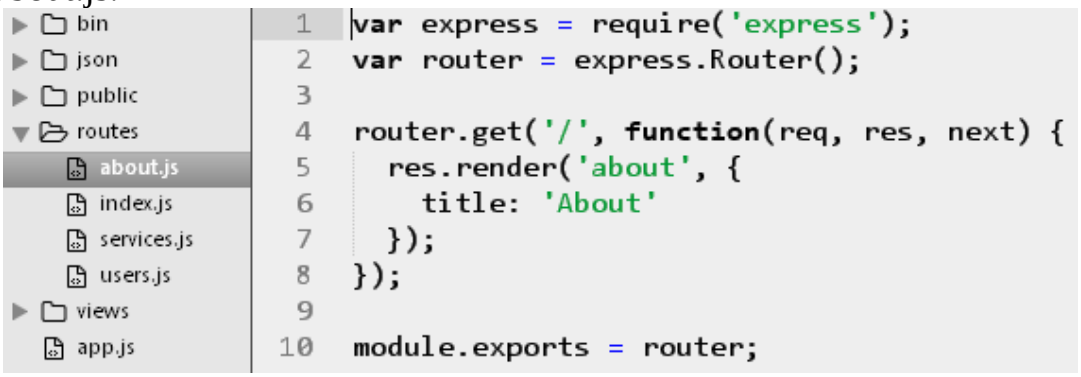

Рис.75. Файл about.js

35. Редагуємо файл app.js:
    - 9-й рядок: var about = require(‘./routes/about’);
    - 26-й рядок: app.use(‘/about’, about);
36. В директорії views створюємо файл about.pug і оновлюємо браузер. Пункт About працює.

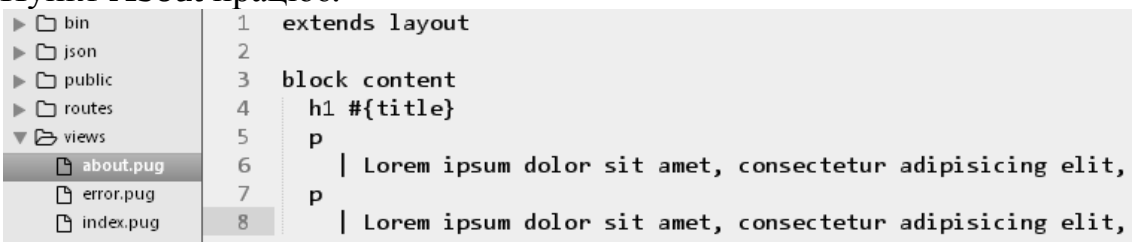

Рис.76. Файл about.pug

37. Попрацюємо над сторінкою Services. В директорії routes створюємо файл services.js і копіюємо в нього вміст файлу about.js.
38. В кореневій директорії нашого проекту створюємо каталог json і в ньому створюємо файл services.json. Вміст файлу services.json:

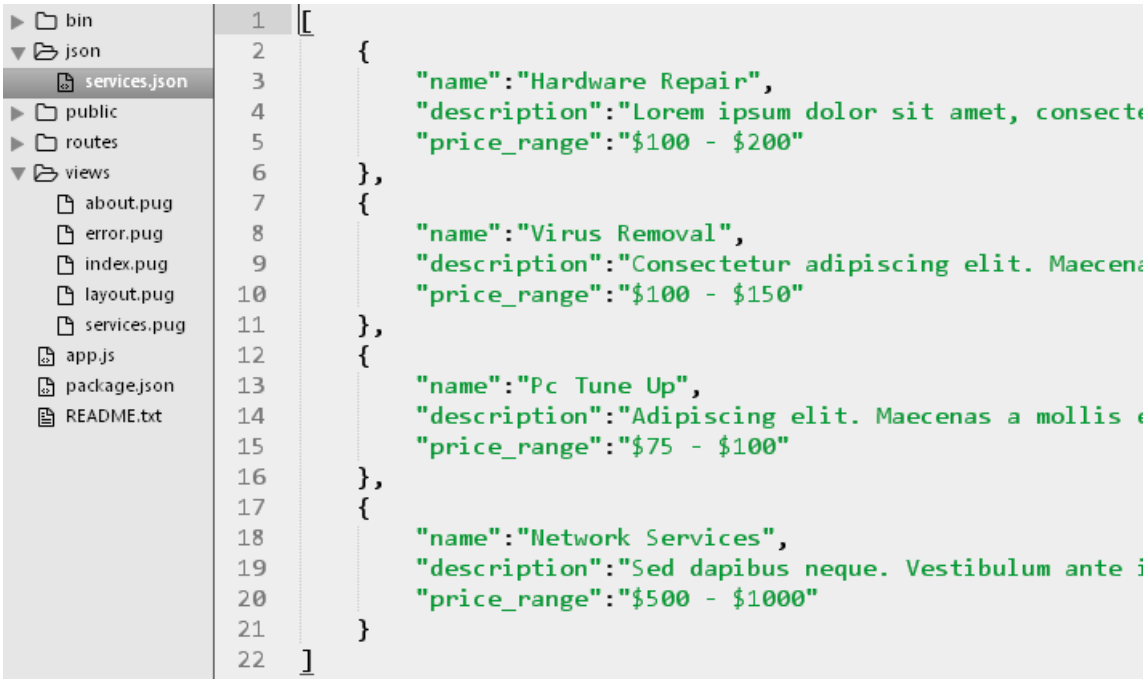

Рис.77. Файл services.json

39. Корегуємо файл services.js.

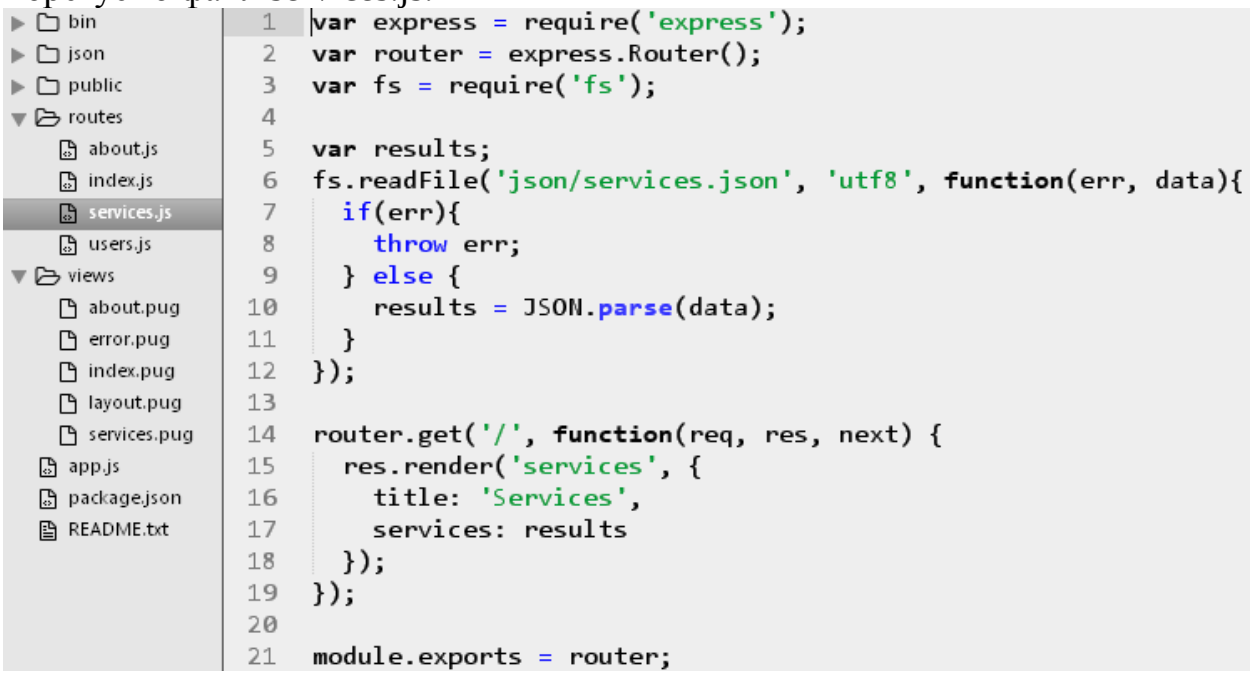

Рис.78. Файл services.js

40. Повертаємось до файлу app.js.
    - Після 9-го рядка вставимо рядок var services = require(‘./routes/services’);
    - Після 27-го рядка вставимо рядок app.use(‘/services’, services);
41. В директорії views створюємо файл services.pug. В браузері вибираємо пункт меню Services щоб побачити результат.

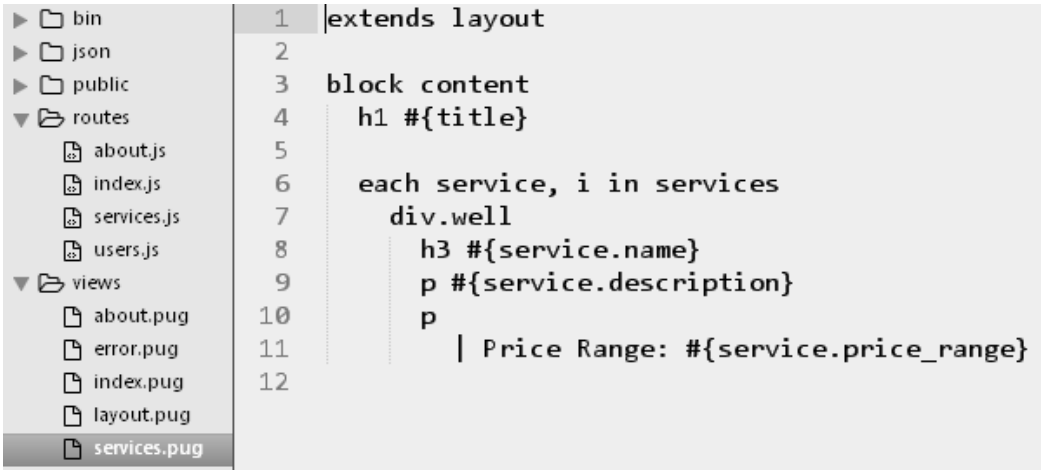

Рис.79. Файл services.pug

42. В директорії routes створюємо файл contact.js. Копіюємо вміст файлу index.js в contact.js. В 6-ому рядку змінюємо слово index на contact, а в 7-ому рядку слово Home на Contact.
43. Заходимо в файл app.js. Після 10-го рядка додаємо рядок var contact= require('./routes/contact'); Після 29-го рядка вставимо рядок app.use('/contact', contact);
44. В директорії views створюємо файл contact.pug.:

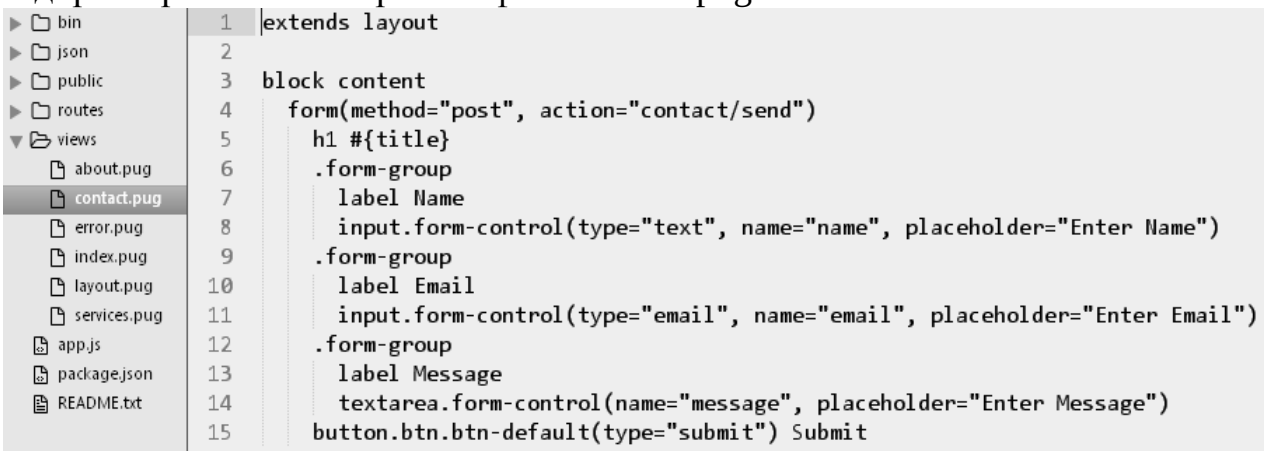

Рис.80. Вміст файлу contact.pug

45. В браузері вибираємо пункт меню Contact щоб побачити результат.
46. Для відправки повідомлень будемо використовувати Nodemailer (https://nodemailer.com/). Подаємо в консолі команду npm install nodemailer –save . Після цього подаємо команду nodemon
47. Повертаємось до файлу contact.js. Вивчемо зразок цього файлу, наведений нище, та документацію до Nodemailer. Внесемо корективи до цього файлу, налаштувавши вашу власну електронну адресу. Перевіримо в браузері роботу пункту меню Contact.

```
// File contact.js
var express = require('express');
var router = express.Router();
var nodemailer = require('nodemailer');
router.get('/', function(req, res, next) {
  res.render('contact', {
    title: 'Contact'
  });
```

```
});
// Send Email
// Send Email
router.post('/send', function(req, res, next){
    var transporter = nodemailer.createTransport({
      service: 'Gmail',
      auth: {
        user: 'YOUREMAIL',
        pass: 'YOURPASS'
      }
    });
    var mailOptions = {
      from: '"YOURNAME" <YOUREMAIL>',
      to: 'TOEMAIL',
      subject: 'Hello from PCRepair',
      text: 'You have a submission from... Name:
'+req.body.name+' Email: '+req.body.email+' Message:
'+req.body.message,
      html: '<p>You have a submission from...</p>
<ul><li>Name: '+req.body.name+'</li><li> Email:
'+req.body.email+'</li><li> Message:
'+req.body.message+'</li></ul>'
    }
    transporter.sendMail(mailOptions, function(error,
info){
      if(error){
        return console.log(error);
      }
      console.log('Message Sent: '+ info.response);
      res.redirect('/');
    });
});
module.exports = router;
```

1.4 Лабораторна робота 9. Розробка сайту за технологією Python+Django.

Завдання: За допомогою фреймворку Django розробити сайт простого блогу з наступними можливостями:

- Користувач-адміністратор може створювати, додавати, видаляти, редагувати записи (повідомлення) блогу.
- Звичайним користувач може переглядати список повідомлень блогу та детальну інформацію по кожному повідомленню.
- Передбачити розбиття на сторінки списку повідомлень блогу.
- На сторінці списку повідомлень розмістити посилання на RSS-канал, що відображає 5 останніх повідомлень блогу.

Теоретичні відомості, методичні вказівки та хід роботи.

Установка Django в Windows

1) Інсталяція Python (https://www.python.org/). На березень 2018 маємо реліз Python 3.6.4. Зверніть увагу на другу сторінку "Customize" майстера установки, вам потрібно пролистати вниз і вибрати опцію "Add python.exe to the Path" (додати python.exe в системну змінну Path).

2) Створення віртуального середовища. Для роботи з Django необхідно спочатку створити віртуальне середовище для роботи. Віртуальне середовище являє собою підрозділ системи, в якому ви можете встановлювати пакети в ізоляції від решти пакетів Python. Відділення бібліотек одного проекту від інших проектів принесе користь при розгортанні вашого застосунку на сервері. Створіть для проекту новий каталог, наприклад з ім'ям D:\DjangoSites, перейдіть в цей каталог в термінальному режимі і створіть віртуальне середовище. Якщо ви працюєте в Python 3, то зможете створити віртуальне середовище наступною командою:
   `python -m venv djangosites_env`
   Команда запускає модуль venv і використовує його для створення віртуального середовища з ім'ям djangosites_env.

3) Активізація віртуального середовища. Після того, як віртуальне середовище буде створене, його необхідно активізувати наступною командою (для Windows):
   `djangosites_env\Scripts\activate`
   Команда запускає сценарій activate з каталогу djangosites_env\Scripts. Коли середовище активізується, його ім'я виводиться в круглих дужках. Тепер ви можете встановлювати пакети в середовищі і використовувати ті пакети, що були встановлені раніше. Пакети, встановлені в djangosites_env, будуть доступні тільки в той час, поки середовище залишається активним. Щоб завершити використання віртуального середовища, введіть команду deactivate.

4) Установка Django. Після того як ви створили своє віртуальне середовище і активізували його, встановіть Django:
   `pip install django==1.11.5`
   Так як ви працюєте в віртуальному середовищі, ця команда виглядає однаково в усіх системах. Версію Django можна не вказувати, тоді буде встановлена остання версія. (Якщо ви отримуєте помилку коли викликаєте pip на Windows, перевірте чи шлях до вашого проекту не містить пробілів, наголосів чи спеціальних символів).

5) Створення проекту mysite в Django. Не виходячи з активного віртуального середовища, введіть наступні команди для створення нового проекту (не забудьте про крапку . в кінці):
   - **`django-admin.py startproject mysite .`**
   - **`dir`** щоб побачити вміст каталогу D:\DjangoSites (там будуть каталоги djangosites_env і mysite та файл manage.py)

- **`dir mysite`** щоб побачити вміст каталогу D:\DjangoSites\mysite (там будуть файли __init__.py, settings.py, urls.py, wsgi.py).

Файл manage.py – це коротка утиліта командного рядка, яка отримує команди і передає їх відповідній частині Django для виконання. Ми використовуємо ці команди для управління такими завданнями, як робота з базами даних і запуск серверів.

В каталозі mysite знаходяться чотири файли, найважливішими з яких є файли settings.py, urls.py і wsgi.py. Файл settings.py визначає те, як Django взаємодіє з вашою системою і управляє вашим проектом. Ми змінимо деякі з існуючих налаштувань і додамо кілька нових налаштувань в ході розробки проекту. Файл urls.py повідомляє Django, які сторінки слід будувати у відповідь на запити браузера. Файл wsgi.py допомагає Django надавати створені файли (ім'я файлу є скороченням від «Web Server Gateway Interface»).

Оскільки ми використовуємо Python 3.5, який постачається з вбудованою базою SQLite, ми можемо використовувати цю базу даних у нашому проекті для розробки. Якщо ви плануєте розгортати вашу програму у виробничому середовищі, ви повинні використовувати розширену базу даних, таку як MySQL, Oracle або PostgreSQL. Згенерований файл settings.py містить базову конфігурацію для використання бази даних SQLite та список програм Django, які за замовчуванням працюють у вашому проекті. Нам потрібно створити таблиці в базі даних для початкового застосування.

6) Створення бази даних. Подамо команду
   `cd mysite`
   шоб перейти в каталог mysite. Щоб створити базу даних для проекту mysite, введіть наступну команду
   `python manage.py migrate`
   В терміналі ви побачите процес створення бази даних. Кожна зміна бази даних називається міграцією. Перше виконання команди migrate наказує Django перевірити, що база даних відповідає поточному стану проекту. Під час першого виконання цієї команди в новому проекті з використанням SQLite Django створює нову базу даних за нас. Django повідомляє про створення таблиць бази даних, необхідних для зберігання інформації, використовуваної в проекті, а потім перевіряє, що структура бази даних відповідає поточному коду. Django створює файл з ім'ям db.sqlite3.

7) Запуск Django Development Server.
   Переконаємося в тому, що проект був створений правильно. Введіть команду runserver:
   `python manage.py runserver`
   Django запускає сервер, щоб ви могли переглянути проект в своїй системі і перевірити, як він працює. Коли ви запитуєте сторінку,

вводячи URL в браузері, сервер Django відповідає на запит; для цього він будує відповідну сторінку і відправляє сторінку браузеру.

Тепер відкрийте браузер і введіть URL

`http://localhost:8000/`

- або http://127.0.0.1:8000/, якщо перша адреса не працює. Ви побачите щось схоже на Рис.81 - сторінку, яку створює Django, щоб повідомити вам, що все поки працює правильно.

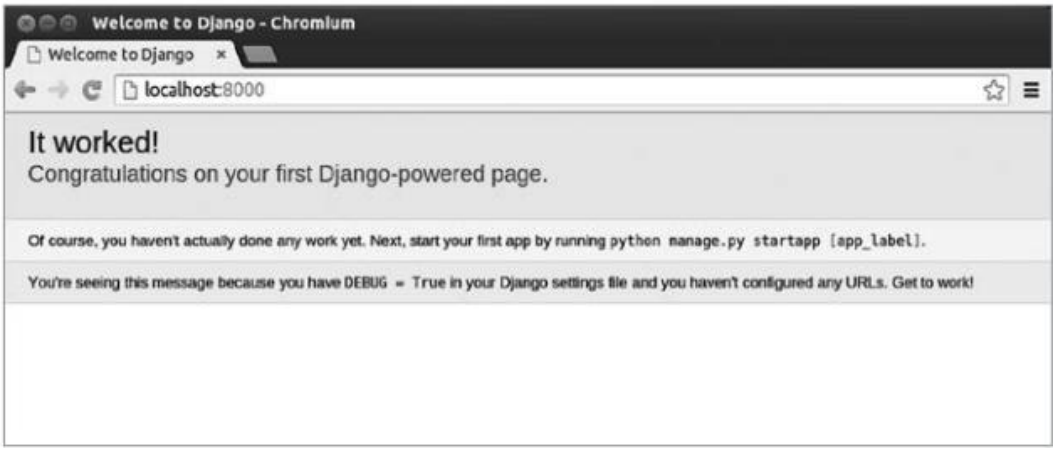

Рис.81. Перша сторінка Django

Ще не завершуйте роботу сервера (але, коли ви захочете перервати його, це можна зробити натисканням клавіш Ctrl + C).

8) Початок роботи над додатком blog. Створення моделей.

Проект Django є групою окремих додатків, спільна робота яких забезпечує роботу проекту в цілому. Поки ми створимо один додаток blog, який буде виконувати більшу частину роботи в нашому проекті.

До цього моменту команда runserver повинна продовжувати роботу в термінальному вікні, яке ви відкрили раніше. Відкрийте нове термінальне вікно (або вкладку) і перейдіть в каталог, що містить manage.py. Активізуйте віртуальне середовище і виконайте команду startapp:

`python manage.py startapp blog`

Команда startapp ім'я_додатку наказує Django створити інфраструктуру, необхідну для побудови додатку. Заглянувши зараз в каталог проекту, ви знайдете в ньому новий підкаталог з ім'ям blog. Відкрийте цей каталог, щоб побачити, які файли були створені Django. Найважливіші файли в цьому каталозі - **models.py**, **admin.py** і **views.py**.

Ми скористаємося файлом models.py для визначення даних, якими потрібно управляти в нашому додатку.

admin.py - Тут ви реєструєте моделі, щоб включити їх в сайт адміністрування Django.

views.py - логіка вашого додатку. Кожне представлення отримує http-запит, обробляє його та повертає відповідь.

Зараз ми створимо схему даних для нашого блогу. Відкриємо каталог проекту в редакторі Sublime Text і відкриємо файл models.py. Модель - це клас Python (підклас django.db.models.Models), в якому кожен атрибут представляє поле бази даних. Django створить таблицю для кожної моделі, яка визначена у файлі models.py. Коли ви створюєте модель, Django пропонує вам практичний API для легкого здійснення запитів до вашої бази даних.

Запишіть у файл blog/**models.py** наступний код:

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User

# Our Post Model

class Post(models.Model):
    STATUS_CHOICES = (
        ('draft', 'Draft'),
        ('published', 'Published'),
    )
    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250,
unique_for_date='publish')
    author = models.ForeignKey(User, related_name='blog_posts')
    body = models.TextField()
    publish = models.DateTimeField(default=timezone.now)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    status = models.CharField(max_length=10,
choices=STATUS_CHOICES, default='draft')

    class Meta:
        ordering = ('-publish',)

    def __str__(self):
        return self.title
```

Це наша основна модель для нашого блогу, давайте розглянемо поля, які ми щойно визначили для цієї моделі:

title - це поле для заголовка повідомлення. Це поле CharField, яке перетворюється на varchar у базі даних SQL.

slug - це поле, призначене для використання в URL-адресах. Slug має коротке позначення, містять лише літери, цифри, підкреслення чи дефіс. Ми будемо використовувати це поле slug для створення красивих, дружніх до SEO сторінок для наших публікацій в блозі. Ми додали параметри unique_for_date для цього поля, щоб ми могли створювати URL-адреси для публікації, використовуючи дату та slug публікації. Django дозволить запобігти декілька постів, що мають однакові slug та дату.

author - автор, це поле є зовнішнім ключем. Це поле визначає відносини багато до одного, ми повідомляємо Django, що кожен пост написаний

користувачем, і користувач може написати кілька повідомлень. У цьому випадку ми покладаємось на модель користувача системи автентифікації Django.

body – це тіло повідомлення.

publish – це поле date time покаже, коли повідомлення було опубліковане. Ми використовуємо метод Django timezone.now як значення за замовчуванням.

created - це поле date time вказує, коли було створено повідомлення. Оскільки ми використовуємо auto_now_add, дата буде автоматично збережена при створенні об'єкта.

updated - цей поле date time показує, коли останній раз був оновлений певний пост. Оскільки ми тут використовуємо auto_now, дата буде автоматично оновлена при збереженні об'єкта.

status - це поле, яке показує статус публікації. Ми використовуємо параметри вибору (STATUS_CHOICES), тому значення цього поля можна встановити лише на один із зазначених варіантів.

Як ви бачите, Django поставляється з різними типами полів, які можуть бути використані для визначення ваших моделей. Ви можете знайти все про різні поля Django в https://docs.djangoproject.com/en/1.11/ref/models/fields/ .

Клас Meta містить метадані. Ми доручаємо Django сортувати результати за полями публікації в порядку зменшення за замовчуванням, коли ми запитуємо дані. Ми вказуємо спадний порядок, використовуючи знак мінус ('-publish',).

Django викликає метод __str __ () для виведення простого представлення моделі. Ми написали реалізацію __str __ (), яка повертає рядок, що зберігається в атрибуті title.

9) Оскільки ми маємо справу з датою, нам потрібно встановити модуль pytz. Цей модуль забезпечить визначення часового поясу для Python, і він потрібен, щоб SQLite працювала з датами. Отже заходимо в термінал і подаємо команду

```
pip install pytz
```

10) Активізація моделей. Щоб використовувати моделі, необхідно наказати Django включити додаток blog в спільний проект. Відкриємо файл settings.py, знайдемо секцію INSTALLED_APPS = () і додамо додаток blog. Ця секція буде виглядати так:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
]
```

Тепер Django знає, що наш додаток blog активний для цього проекту і зможе самостійно переглядати його моделі. Тепер ми будемо

створювати і застосовувати міграції. Давайте створимо в базі даних таблицю для нашої моделі Post.

11) Впевнившись, що в консолі активною є коренева директорія нашого проекту (тобто доступний файл manage.py), подаємо команду

```
python manage.py makemigrations blog
```

Django створив файл 0001_initial.py всередині каталогу migrations нашого додатку blog. Ви можете відкрити цей файл, щоб побачити, як виглядає міграція. Виконайте наступну команду, щоб перевірити код:

```
python manage.py sqlmigrate blog 0001
```

Для синхронізації нашої бази даних з моделлю Post виконайте таку команду:

```
python manage.py migrate
```

База даних тепер відображає поточний стан нашої моделі блогу.

12) Адміністративний сайт Django.

Django дозволяє легко працювати з моделями, визначеними для додатка, через адміністративний сайт. Цей сайт використовується адміністраторами сайту, а не рядовими користувачами. У цьому розділі ми створимо адміністративний сайт і використаємо його для додавання деяких тем через модель Post.

Спочатку ми створимо суперкористувача, а потім додамо модель Post на сайт адміністрації, далі ми налаштуємо те, як моделі відображаються. Щоб створити суперкористувача, запустіть таку команду:

```
python manage.py createsuperuser
```

Ваш термінал попросить вас ввести ім'я користувача, адресу електронної пошти та пароль. Ви можете ввести: admin, admin@example.com і залишити поля для пароля пустими.

Тепер ми розглянемо сайт адміністрування Django. Запустіть сервер розробника, запустивши таку команду:

```
python manage.py runserver
```

У своєму браузері перейдіть за цим посиланням:

http://localhost:8000/admin/

Ви повинні побачити екран входу адміністратора, як показано нижче (Рис.82):

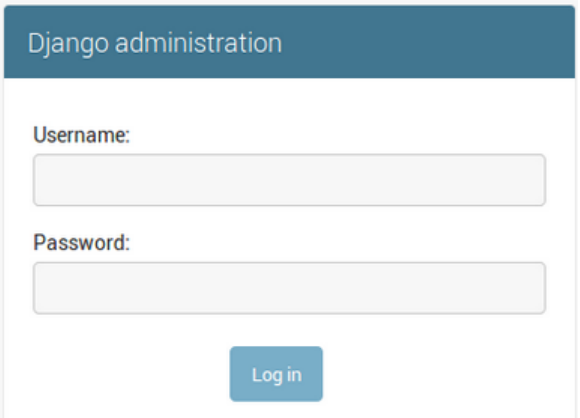

Рис.82. Форма входу адміністратора

Після введення даних ви побачите сторінку адміністратора, як показано тут:

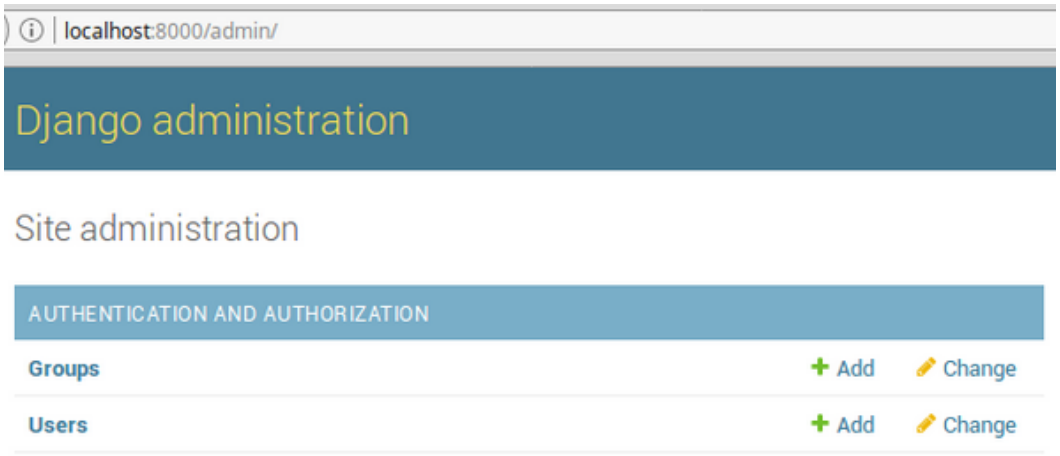

Рис. 83. Сторінка адміністратора

13) Реєстрація моделі на адміністративному сайті.

Django додає деякі моделі (наприклад, User і Group) на адміністративний сайт автоматично, але моделі, які ми створили, доведеться реєструвати вручну.

Щоб додати нашу модель Post на сайт адміністратора, давайте відкриємо файл admin.py і напишемо наступний код:

```
from django.contrib import admin
from .models import Post

admin.site.register(Post)
```

Тепер перезавантажте адміністративний сайт у своєму браузері, і ви побачите такий екранний знімок:

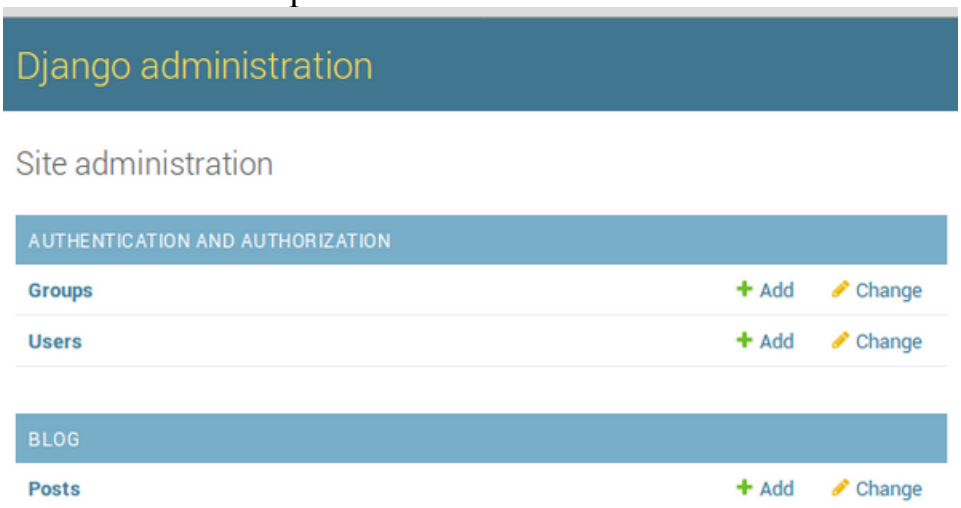

Рис.84. Сторінка адміністратора з моделлю Post

Тепер модель блогу доступна на нашому адміністративному сайті. Це лише невелика частка того, що може зробити адміністратор сайту Django. Коли ви реєструєте свою модель на сайті адміністратора Django, ви отримуєте зручний інтерфейс користувача для ваших моделей, що дозволяє вам просто переглядати, редагувати, створювати та видаляти об'єкти.

Натисніть посилання Add праворуч від Posts, щоб додати нову публікацію, ви побачите форму створення, яку Django згенерував динамічно на основі моделі Post. Ось екранний знімок:

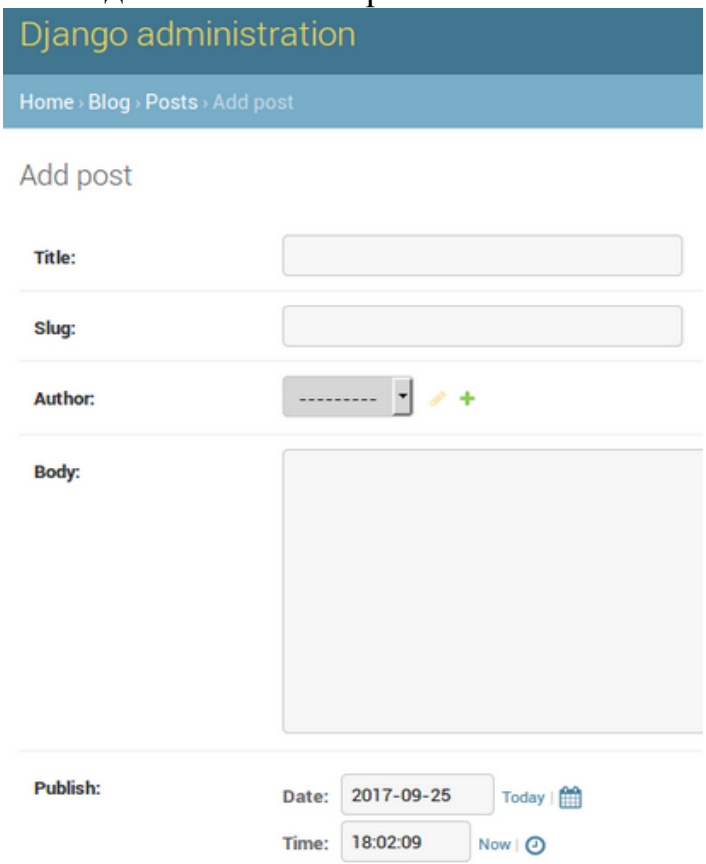

Рис.85. Форма введення данних, згенерована Django

Заповніть форму та натисніть кнопку збереження. Ви будете перенаправлені на сторінку Post з успішним повідомленням про створений вами пост. Django використовує різні формати віджетів для кожного типу полів, навіть складні поля, такі як поля часу дати, які відображаються за допомогою простого інтерфейсу, наприклад, вибору дати JavaScript.

Давайте налаштуємо адміністратора сайту для цього. Додайте наступний код у файл admin.py програми вашого блогу.

```
from django.contrib import admin
from .models import Post

class PostAdmin(admin.ModelAdmin):
    list_display                                    =
('title','slug','author','status','created')
    list_filter = ('status','created','publish','author')
    search_fields = ('title','author')
    prepopulated_fields = {'slug': ('title',)}
    raw_id_fields = ('author',)
    date_hierarchy = 'publish'
    ordering = ['status', 'publish']

admin.site.register(Post, PostAdmin)
```

У клас PostAdmin ми можемо включити інформацію про те, як відображати моделі та взаємодіяти з ними на сайті адміністратора. Атрибут

list_display дозволяє вам встановити поля вашої моделі, які ви хочете відобразити на сторінці веб-сайту адміністратора. Щоб дізнатись більше про те, як налаштувати свій сайт адміністратора, відвідайте https://docs.djangoproject.com/en/1.11/ref/contrib/admin/ .

Тепер поверніться до свого браузера та перезавантажте сторінку списку публікацій. Тепер він буде виглядати як екранний знімок нижче:

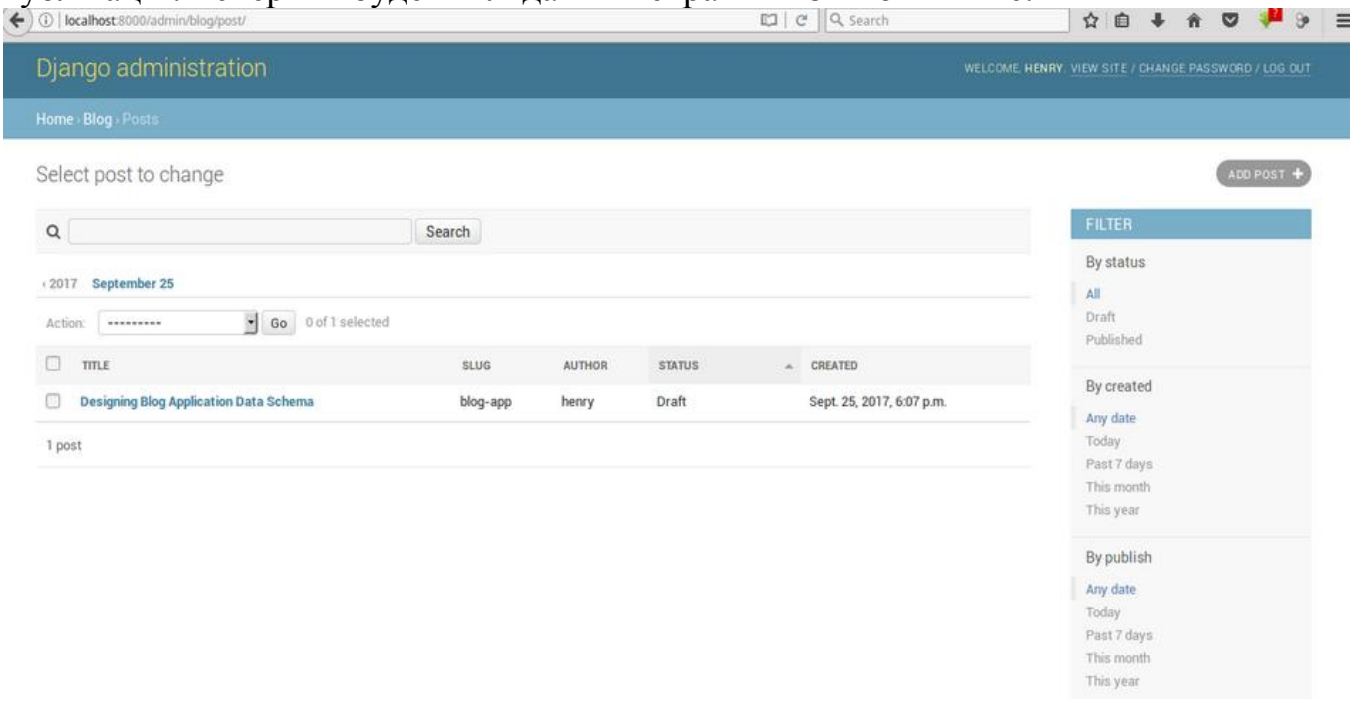

Рис.86. Сторінка списку повідомлень

Сторінка списку тепер включає праву бічну панель, яка дозволяє фільтрувати публікації на основі полів, включених до атрибута list_filter у рядку 6. Панель пошуку з'явилася на сторінці через атрибут search_fields на рядку 7. Трохи нижче поля пошуку присутня панель для навігації через ієрархію дат, це було визначено атрибутом date_hierarchy в рядку 10.

За допомогою декількох рядків коду ми налаштували спосіб відображення моделі нашого повідомлення на сайті адміністратора. Є багато способів налаштування та розширення сайту адміністрування Django. Красиво, ми зробили чудову роботу.

14) Робота з QuerySet & Managers для нашого блогу. Інтерактивна оболонка Django.

Введені дані можна проаналізувати на програмному рівні в інтерактивному термінальному сеансі. Це інтерактивне середовище, зване оболонкою (shell) Django, прекрасно підходить для тестування і діагностики проекту.

У цьому розділі ми спочатку вивчимо, як створювати, оновлювати, завантажувати та видаляти об'єкти в базі даних. Потім ми побачимо, як ми можемо створити модельних менеджерів. Тепер, коли у вас є повністю функціональний сайт адміністрування для керування вмістом вашого блогу, пора навчитися отримувати інформацію з бази даних та взаємодіяти з нею.

Django поставляється з потужною API абстракції баз даних, які дозволяють легко створювати, отримувати, оновлювати та видаляти об'єкти.

Mapper Relationship Object Django (ORM) сумісний з такими базами даних, як MySql, SQLite, PostgreSQL та Oracle. Пам'ятайте, що ви можете визначити базу даних свого проекту, редагуючи налаштування бази даних у файлі settings.py вашого проекту. Django може працювати з декількома базами одночасно, і навіть ви можете запрограмувати маршрутизатори баз даних, які обробляють дані будь-яким способом. Створивши свої моделі даних, Django надає вам безкоштовну API для взаємодії з ними. Тепер давайте створювати об'єкти.

Щоб створити свій перший об'єкт, активуйте віртуальне середовище та перейдіть до базової директорії проекту, де знаходиться файл manage.py на вашому терміналі. Запустіть таку команду, щоб відкрити оболонку python:

```
python manage.py shell
```

Три знаки greater than (‘>>>’) показують, що наша інтерактивна консоль python готова до введення. По-перше, ми повинні імпортувати бібліотеку, введіть таку команду:

```
from blog.models import Post
```

і натисніть Enter.

Тепер щоб отримати користувача з бази даних за допомогою інтерактивної консолі python, введіть таку команду:

```
user = User.objects.get(username='admin')
```

і натисніть Enter.

Щоб створити публікацію, введіть таку команду:

```
post = Post.objects.create(title='Django Blog App',slug='django-blog-app', body='learning how to create blog',author=user)
```

Створений нами пост об'єкт знаходиться в пам'яті комп'ютера і не зберігається в базі даних. Щоб зберегти об'єкт повідомлення, нам потрібно скористатись методом save, отже запустіть таку команду:

```
post.save()
```

Отримання об'єкта на нашій інтерактивній консолі.

Mapper для зв'язку об'єктів (ORM) Django заснований на QuerySet. QuerySet - це сукупність об'єктів у вашій базі даних, які можуть мати кілька фільтрів для обмеження результатів. Ви вже знаєте, як отримати один об'єкт з бази даних за допомогою методу get. Кожна модель Django має щонайменше одного менеджера, а менеджер за замовчуванням називається **objects**.

Щоб отримати весь наш блог з бази даних, ми можемо використовувати all() метод, отже запустіть таку команду:

```
post_query = Post.objects.all()
```

Зауважте, що post_query = Post.objects.all() ще не виконано, Django QuerySet «ліниві», і вони оцінюються тільки тоді, коли вони змушені це зробити. Ця поведінка робить Django QuerySet дуже ефективним, отже щоб виконати наш запит запустіть таку команду:

```
Post.objects.all()
```

Тепер ми побачимо список всіх наших постів.

Тепер давайте дізнаємося, як ми можемо використовувати метод фільтрації. Розглянемо приклад, де ми фільтруємо публікації, створені автором admin. Щоб виконати це, треба запускати таку команду:

```
Post.objects.filter (author__username = 'admin')
```

І, нарешті, розглянемо приклад, як фільтрувати за допомогою декількох полів. Наприклад, ми хочемо отримати всі пости, написаі admin в 2017 році. Це команда, яку ми буемо використовувати:

```
Post.objects.filter (publish__year = 2017, author__username
= 'admin')
```

Ми створювали запити за допомогою методу пошуку полів із використанням двох підкреслень, наприклад author__username. Наступне, що нам потрібно навчитися, - метод виключення, ви можете виключити певний результат із вашого запиту, використовуючи метод exclude менеджера. Наприклад, ми можемо отримати весь пост, опублікований в 2017 році, і титул якого не починається словом Django. Ми можемо використовувати цю команду:

```
Post.objects.filter(publish__year=2017)\
.exclude(title__startswith='Django')
```

Тепер розглянемо метод order_by. Наприклад, ви можете отримати всі об'єкти, упорядковані за їхнім заголовком. Для цього виконайте таку команду:

```
Post.objects.order_by('title')
```

Видалення об'єктів

Якщо ви хочете видалити об'єкт, ви можете зробити це з екземпляра об'єкта, передаючи ідентифікатор, використовуючи цю команду:

```
post_del = Post.objects.get(id=1)
```

а потім викликати метод видалення за допомогою такої команди:

```
post_del.delete()
```

Менеджери моделей

Як ми вже згадували раніше, objects є менеджером за умовчанням кожної моделі, який використовується для доступу до всіх об'єктів у базі даних, але ми також можемо визначити спеціальні менеджери для наших моделей. Ми збираємося створити спеціальний менеджер, щоб отримати всі публікації з опублікованим статусом. Існує два способи додавання менеджерів до ваших моделей, додавання додаткових менеджерських методів або зміна початкового менеджера QuerySet.

Відредагуйте файл models.py нашої програми для блогу та переконайтеся, що він містить такий код.

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User

# Custom Manager
class PublishedManager(models.Manager):
    def get_queryset(self):
        return
super(PublishedManager,self).get_queryset().filter(status='published')
```

```
# Our Post Model
class Post(models.Model):
    STATUS_CHOICES = (
        ('draft', 'Draft'),
        ('published', 'Published'),
    )
    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250, unique_for_date='publish')
    author = models.ForeignKey(User, related_name='blog_posts')
    body = models.TextField()
    publish = models.DateTimeField(default=timezone.now)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    status = models.CharField(max_length=10, choices=STATUS_CHOICES,
default='draft')

    # The default manager
    objects = models.Manager()

    # Custom made manager
    published = PublishedManager()

    class Meta:
        ordering = ('-publish',)

    def __str__(self):
        return self.title
```

В рядках 6 - 8 ми визначили нашого індивідуального менеджера моделі нашої публікації. На рядку 7 ми визначили метод get_queryset, який повертає queryset для виконання. Після визначення нашого індивідуального менеджера ми повинні використовувати його в нашій моделі Post для виконання запитів, для цього ми називаємо цього менеджера в рядку 29. Щоб дізнатися більше про менеджерів відвідайте https://docs.djangoproject.com/en/1.11/topics/db/managers/ .

15) Створення списку публікацій та детальних переглядів для нашого блогу.

Перегляд (view) Django - це лише функція Python, яка отримує веб-запит і повертає веб-відповідь. Всередині перегляду йде вся логіка, необхідна для повернення бажаної відповіді. Спочатку ми створимо представлення для нашого блогу, потім ми визначимо шаблон URL для кожного представлення даних і, нарешті, ми створимо шаблон HTML, щоб відтворити данні, згенеровані views.

Кожне представлення (view) рендерить шаблон, передавши в нього змінну, і повертає HTTP-відповідь з рендерованим виходом. Почнемо зі створення функції post_list_view, яка буде містити список публікацій, та функції post_detail_view для детального представлення. Відредагуйте файл views.py програми блогу та переконайтеся, що у вас є такий код:

```
from django.shortcuts import render, get_object_or_404
from .models import Post

def post_list_view(request):
```

```
    posts = Post.published.all()
    return render(request, 'blog/post/list.html', {'posts':
posts})

def post_detail_view(request, year, month, day, post):
    post = get_object_or_404(Post, slug=post,
status='published', publish__year=year, publish__month=month,
publish__day=day)
    return render(request, 'blog/post/detail.html', {'post':
post})
```

Ви тільки що створили свої перші Django views. На рядку 4, post_list_view - цей вигляд приймає об'єкт запиту як єдиний параметр, завжди пам'ятайте, що цей параметр потрібен усім переглядам. На рядку 5, posts = Post.published.all () - ми отримуємо всі публікації, використовуючи менеджер публікацій, який ми створили раніше. У рядку 6 ми використовуємо функцію render, надану Django, щоб відобразити список публікацій у даний шаблон Ця функція приймає об'єкт запиту, шлях шаблону та змінні для рендеринга.

На рядку 8, post_detail_view - це наша друга функція, це представлення, яке покаже окреме повідомлення. Цей перегляд приймає запит, рік, місяць, день та post параметри, щоб отримати опублікований допис з даним значенням slug та датою. Пам'ятайте, коли ми створили модель Post, ми додали унікальний параметр дати в поле slug, таким чином, ми гарантуємо, що буде лише одна публікація з унікальним slug за певну дату.

Для кожного перегляду в Django потрібен шаблон url для його відображення в браузері. Шаблон url у Django складається з регулярних виразів Python, представлення та назви, які дозволяють назвати його будь-де у вашому проекті. Django проходить через кожний шаблон URL-адреси і зупиняється на першому, який відповідає запитуваній URL-адресі. Тоді Django імпортує вигляд, що відповідає шаблону url, і виконує його, передаючи екземпляр класу HTTP-запита і аргументів.

У каталозі додатка blog створіть файл **urls.py** та переконайтеся, що він містить такі рядки коду:

```
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.post_list_view, name='post_list_view'),

url(r'^(?P<year>\d{4})/(?P<month>\d{2})/(?P<day>\d{2})/(?P<post>
[-\w]+)/$', views.post_detail_view, name='post_detail_view'),
]
```

На рядку 5 - це URL-адреса, яка не приймає жодних аргументів і відображається до post_list_view. У рядку 6 - це шаблон url, який приймає чотири аргументи, а саме рік - вимагає чотирьох цифр, місяць - вимагає двох цифр, день - вимагає двох цифр і post - який може складатися з слів і дефісів. Цей URL посилається на post_detail_view.

Створення файлу urls.py для кожного додатку в Django - це найкращий спосіб повторно використовувати ваш додаток іншими проектами Django. Тепер ми маємо включити шаблони url нашого блогу в основні шаблони URL-адрес проектів. Відредагуйте файл urls.py, який знаходиться в директорії mysite, і переконайтеся, що він містить такий код:

```
"""mysite URL Configuration

The `urlpatterns` list routes URLs to views. For more information
please see:
    https://docs.djangoproject.com/en/1.11/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(),
name='home')
Including another URLconf
    1. Import the include() function: from django.conf.urls import
url, include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'blog/', include('blog.urls', namespace='blog',
app_name='blog')),
]
```

У рядку 21 ми повідомляємо Django, що він включає в себе шаблони URL-адрес, визначені у файлі urls.py блогу, ми даємо цим URL-адресам простір імен blog, що полегшує посилання на цю групу URL-адрес.

Канонічні URL-адреси для моделей

Ми можемо використовувати post_detail_view, який ми визначили раніше, для створення канонічної URL-адреси для об'єкта post. Згідно конвенції Django слід додати до моделі метод get_absolute_url, який повертає канонічну URL-адресу об'єктів. Відкрийте файл models.py нашого блогу та переконайтеся, що він містить такий код:

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User
from django.core.urlresolvers import reverse

# Custom Manager

class PublishedManager(models.Manager):
    def get_queryset(self):
        return
super(PublishedManager,self).get_queryset().filter(status='published')
```

```
# Our Post Model

class Post(models.Model):
    STATUS_CHOICES = (
        ('draft', 'Draft'),
        ('published', 'Published'),
    )
    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250, unique_for_date='publish')
    author = models.ForeignKey(User, related_name='blog_posts')
    body = models.TextField()
    publish = models.DateTimeField(default=timezone.now)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    status = models.CharField(max_length=10, choices=STATUS_CHOICES,
default='draft')

    # The default manager
    objects = models.Manager()

    # Custom made manager
    published = PublishedManager()

    class Meta:
        ordering = ('-publish',)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('blog:post_detail_view',
args=[self.publish.year,
self.publish.strftime('%m'),self.publish.strftime('%d'),self.slug])
```

У рядку 4 ми імпортуємо бібліотеку для нашого канонічного методу reverse url. На лініях 41 та 42 ми використовуємо метод reverse, який дозволяє створювати URL-адреси за їхніми іменами та опційними параметрами. Зауважте, що ми використовуємо функцію strftime у self.publish.strftime ('% m'), self.publish.strftime ('% d'), щоб побудувати URL-адресу, використовуючи місяць та день з передуючими нулями. Ми будемо використовувати метод get_absolute_url у наших шаблонах.

16) Створення шаблонів (templates) для нашого блогу.

У цьому розділі ми будемо додавати шаблони, щоб відображати наші публікації блогу зручним способом. Ми вже створили перегляди (views) та шаблони URL-адрес (url patterns) для нашого додатку blog, тепер пора додавати шаблони (templates). У нашому каталозі blog ми збираємося створити дві папки, названі **templates** та **static**. Всередині каталогу templates створіть іншу папку та назвіть її **blog**. Всередині каталогу blog створіть інший каталог і назвіть його **post**. Ще в каталозі blog створіть файл HTML і назовіть його **master.html**. Всередині папки post створіть два HTML-файли, **detail.html** та **list.html** Всередині каталогу static створіть іншу папку з назвою

css. Завантажте bootstrap і розмістіть файл **bootstrap.min.css** в папці css, яку ви щойно створили.

Тепер ваш каталог має виглядати так:

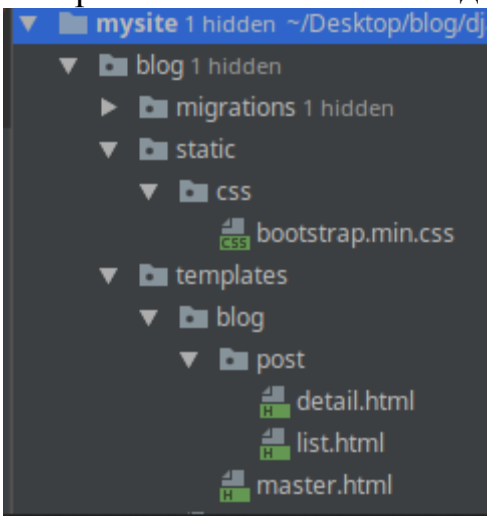

Рис.87. Каталог проекту

Файл master.html включатиме основну HTML-структуру веб-сайту, ми розділимо файл master.html на дві основні області, основну область вмісту та бічну панель. Файли list.html і detail.html успадкують від master.html, щоб відобразити список публікацій блогу та окреме повідомлення відповідно. Django має потужну мову шаблонів, яка дозволяє вказати, яким чином відображаються дані, вона базується на тегах шаблонів (template tags), які виглядають як змінні шаблону тегів (tag template variable) та фільтри (filters). Щоб дізнатись більше про Django template language відвідайте https://docs.djangoproject.com/en/1.11/topics/templates/ .

Відредагуйте файл **master.html** та переконатися, що він має такий код:

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
    <link rel="stylesheet" href="{% static "css/bootstrap.min.css"
%}">
</head>
<body>
  <div class="container">
      <div class="row">
          <div class="col-md-8">
              {% block content %}

              {% endblock %}
          </div>
          <div class="col-md-4">
              <h3>Muvalab Blog</h3>
              <p>This is our sidebar</p>
          </div>
      </div>
  </div>
</body>
```

```
</html>
```

Тепер давайте поглянемо на код. На рядку 1, де ми маємо {% load staticfiles%}, ми повідомляємо Django про завантаження статичних файлів тегів шаблонів файлів, які надаються програмою django.contrib.staticfiles. За допомогою цього тегу шаблонів ви можете включити статичні файли. В рядку 7 ми завантажуємо bootstrap.min.css, використовуючи тег статичного фільтра. У рядку 5 ми маємо теги блоків {% block title%} (% endblock%}, блочні теги повідомляють Django, що ми хочемо визначити блок у цій області. Шаблони, які успадковуються з шаблону master.html, можуть заповнювати цю область блоку вмістом. У цьому шаблоні ми визначили два блоки під назвою title та content.

Відредагуйте файл **list.html** та переконатися, що він має такий код:

```
{% extends "blog/master.html" %}

{% block title %}Our Blog{% endblock %}

{% block content %}
   <h2 class="page-header">Recents Posts</h2>
    {% for post in posts %}
       <h3><a href="{{ post.get_absolute_url }}">{{ post.title
}}</a></h3>
        <p class="small">Published {{ post.publish }} by {{
post.author }}</p>
        <p>{{ post.body|truncatewords:25|linebreaks }}</p>
    {% endfor %}

{% endblock %}
```

У рядку 1 ми використовуємо тег extends, щоб успадкувати master.html. У рядку 3 ми заповнюємо блок title нашим заголовком. З рядків від 5 до 13 ми визначили нашу область вмісту (block content), в рядку від 7 до 11 ми проводимо ітерацію по всім повідомленням, і ми показуємо назву повідомлення, дату, тіло повідомлення та автора. У рядку 8 ми включаємо посилання на публікацію, використовуючи get_absolute_url (канонічний URL).

Давайте подивимось, що ми маємо на цей момент. Запустіть свій сервер розробки і завжди пам'ятайте, що треба активувати віртуальне середовище. Щоб запустити сервер, виконайте таку команду:

```
python manage.py runserver
```

Відкрийте наступне посилання у своєму веб-переглядачі:

```
http://localhost:8000/blog/
```

і ви повинні побачити такий екран:

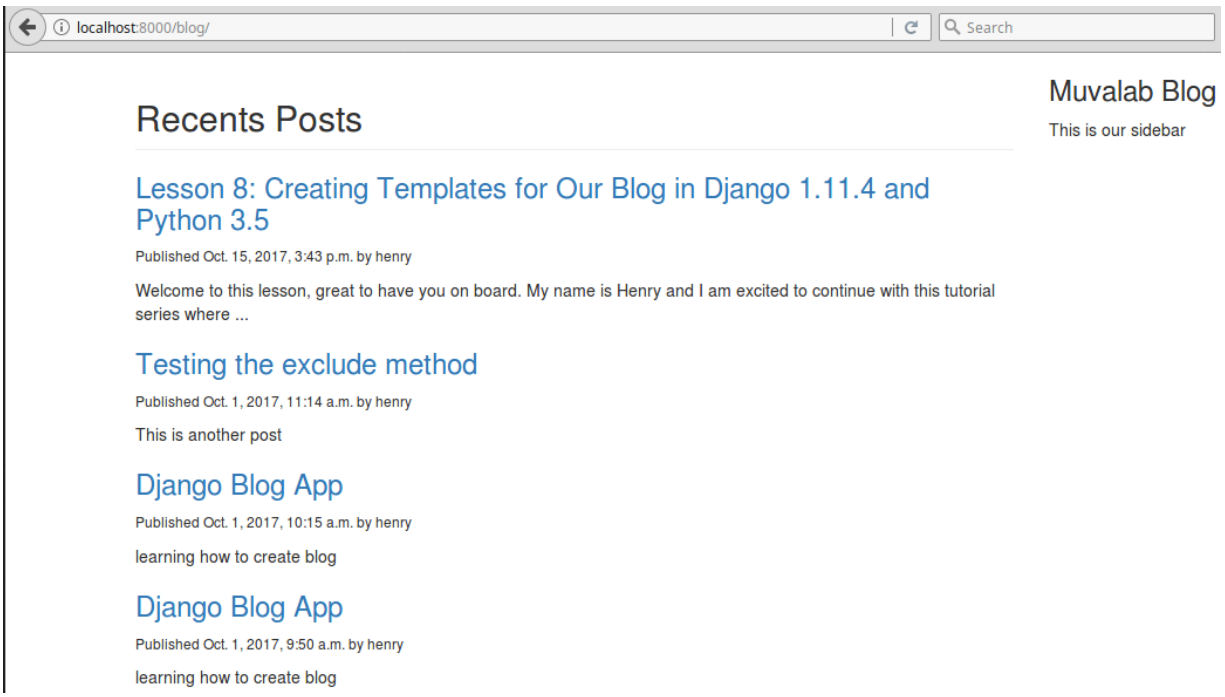

Рис.88. Список повідомлень блогу

Примітка. Вам потрібно мати публікацію, статус якої є published, щоб ви могли переглянути список публікацій.

Тепер відредагуйте файл **detail.html** та переконайтеся, що він має такий код:

```
{% extends "blog/master.html" %}

{% block title %}{{ post.title }}{% endblock %}

{% block content %}
   <h3>{{ post.title }}</h3>
    <p class="small">Published {{ post.publish }} by {{ post.author
}}</p>
    <p>{{ post.body|linebreaks }}</p>
{% endblock %}
```

Тепер поверніться до свого веб-переглядача та натисніть на одне з заголовків повідомлень, щоб побачити повний допис. Ви повинні побачити щось на зразок цього:

Рис.89. Сторінка окремого повідомлення

Тепер подивіться на URL-адресу свого браузера, ми створили дружню URL-адресу для нашого блогу.

17) Створення розбивки на сторінки (Pagination) для нашого блогу.

Коли кількість публікацій у блозі буде зростати, ви скоро зрозумієте, що вам потрібно розділити список публікацій на кілька сторінок. Django має вбудований клас pagination, який дозволяє розробникам легко керувати розбивкою на сторінки. Тепер відредагуйте файл views.py і переконайтеся, що він має такий код:

```
from django.shortcuts import render, get_object_or_404
from django.core.paginator import Paginator, EmptyPage,
PageNotAnInteger
from .models import Post

def post_list_view(request):
    list_objects = Post.published.all()
    paginator = Paginator(list_objects, 3)
    page = request.GET.get('page')
    try:
        posts = paginator.page(page)
    except PageNotAnInteger:
        posts = paginator.page(1)
    except EmptyPage:
        posts = paginator.page(paginator.num_pages)
    return render(request, 'blog/post/list.html', {'posts':
posts})

def post_detail_view(request, year, month, day, post):
    post = get_object_or_404(Post, slug=post,
status='published', publish__year=year, publish__month=month,
publish__day=day)
    return render(request, 'blog/post/detail.html', {'post':
post})
```

На рядку 2 ми імпортуємо Django paginated classes. Ми змінили post_list_view, розглянемо як працює pagination у Django:

Рядок 7 – створюється об’єкт класу Paginator з заданням кількості об'єктів, які ви хочете відобразити на кожній сторінці.

Рядок 8 - Ми отримуємо параметр GET сторінки, який вказує поточний номер сторінки.

Рядок 10 - Ми отримуємо об'єкт для потрібної сторінки page, викликаючи метод page об'єкта paginator.

Рядок 12 - Якщо параметр сторінки не є цілим числом, ми отримуємо першу сторінку результатів, якщо цей параметр є більшим, ніж остання сторінка результатів, ми отримуємо останню сторінку.

Рядок 14 - ми передаємо в paginator кількість сторінок та отримуємо об'єкти до шаблону.

Тепер нам потрібно створити шаблон, щоб відобразити paginator таким чином, щоб він міг бути включений у будь-який шаблон, який використовує

pagination. У папці **template** додатку blog створіть новий файл та назвіть його **pagination.html**. Переконайтеся, що pagination.html має такий код:

```
<div class="container">
    <div class="row">
    <span class="pagination">
        {% if page.has_previous %}
            <a href="?page={{ page.previous_page_number
}}">previous</a>
        {% endif %}

      <span class="active">
          page {{ page.number }} of {{ page.paginator.num_pages
}}
      </span>
    {% if page.has_next %}
        <a href="?page={{ page.next_page_number }}">next</a>
    {% endif %}
    </span>
</div>
</div>
```

Шаблон сторінок (pagination template) очікує об'єкта page, щоб відобразити попереднє посилання, наступне посилання, поточну сторінку та загальну сторінку. Тепер давайте відкриємо наш файл шаблону **list.html**, розташованого в blog/post/ директорії, і переконайтеся, що він має такий код:

```
{% extends "blog/master.html" %}

{% block title %}Our Blog{% endblock %}

{% block content %}
   <h2 class="page-header">Recents Posts</h2>
    {% for post in posts %}
       <h3><a href="{{ post.get_absolute_url }}">{{ post.title
}}</a></h3>
        <p class="small" style="color: #777777">Published {{
post.publish }} by {{ post.author }}</p>
        <p>{{ post.body|truncatewords:25|linebreaks }}</p>
    {% endfor %}

    {% include "pagination.html" with page=posts %}
{% endblock %}
```

У рядку 13 ми просто включили нашу сторінку pagination.html у нижню частину блоку вмісту. Оскільки об'єкт page ми передаємо до шаблону list.html, ми у рядку 13 задаємо page = posts. Подібний підхід ви можете використовувати для повторного використання шаблону сторінок у ваших проектах Django, а також для розбивки на сторінки різних моделей.

Переконайтеся, що ви додали більше 5 повідомлень для тестування своїх сторінок. Після завершення додавання повідомлень розгорніть сервер розробки та відкрийте це посилання у своєму веб-переглядачі:

http://localhost:8000/blog/

Ось можливий екранний знімок браузера:

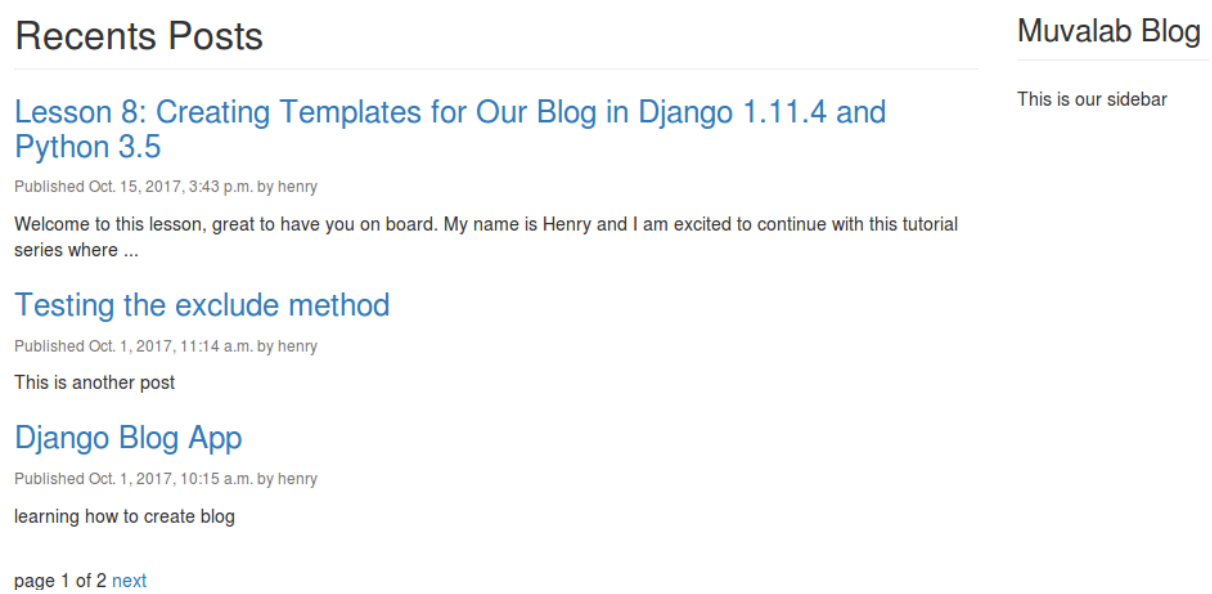

Рис.90. Список повідомлень блогу з розбиттям на сторінки

Ви побачите розбивку на сторінки (pagination) в нижній частині сторінки списку публікацій, і ви матєте змогу переходити по сторінках.

18) Створення RSS-каналів для нашого блогу.

Веб-фреймворк Django має вбудований syndication feed framework, який ви можете використовувати для динамічного створення каналів RSS або atom feeds. У нашому каталозі blog створіть новий файл з назвою **feeds.py** і переконайтеся, що в ньому є такий код:

```
from django.contrib.syndication.views import Feed
from django.template.defaultfilters import truncatewords
from .models import Post

class PostsFeed(Feed):
    title = 'Henry Blog Feeds'
    link = '/blog/'
    description = 'Our latest Posts!'

    def items(self):
        return Post.published.all()[:5]

    def item_title(self, item):
        return item.title

    def item_description(self, item):
        return truncatewords(item.body, 20)
```

Давайте зрозуміємо вищезазначений код. У рядку 1 ми імпортуємо бібліотеку синдикації, яку нам надає Django. У рядку 2 ми імпортуємо фільтр шаблонів, який допоможе нам підсумувати тіло нашого блогу. У рядку 3 ми імпортуємо нашу модель Post з файлу models.py.

У рядках 5 - 8 ми створюємо клас під назвою PostsFeed, і ми передаємо потік синдикації в цей клас, таким чином, у нас є PostFeed (Feed), який стає

підкласом класу Feed базової синдикації Django. Атрибути title, link та description відповідають назві, посиланням та опису RSS відповідно.

У рядках 10 - 11 ми створили метод items, який витягує об'єкти з нашої моделі Post, які будуть включені в канал, в цьому випадку ми вказуємо 5 останніх публікацій. У рядках 13 - 14 ми створили метод item_title, який повертає назву отриманого об'єкта. У рядках 16 - 17 ми створили метод item_description, який повертає тіло повідомлення, і ми вирізаємо це тіло до перших 20 слів.

Тепер нам потрібно змінити файл **urls.py** нашої блогової програми, переконайтеся, що файл має такий код:

```
from django.conf.urls import url
from . import views
from .feeds import PostsFeed

urlpatterns = [
    url(r'^$', views.post_list_view, name='post_list_view'),

url(r'^(?P<year>\d{4})/(?P<month>\d{2})/(?P<day>\d{2})/(?P<post>
[-\w]+)/$', views.post_detail_view, name='post_detail_view'),
    url(r'^feed/$', PostsFeed(), name='post_feed')
]
```

У рядку 3 ми імпортуємо клас PostsFeed, який ми щойно створили в нашому файлі feeds.py. У рядку 8 ми інсталюємо PostsFeed в новому шаблоні URL-адреси. Давайте перевіримо наш код, повертаємо ваш сервер розробки та переходимо до цього URL:

http://localhost:8000/blog/feed/

і ви побачите наступний вивід у своєму веб-переглядачі.

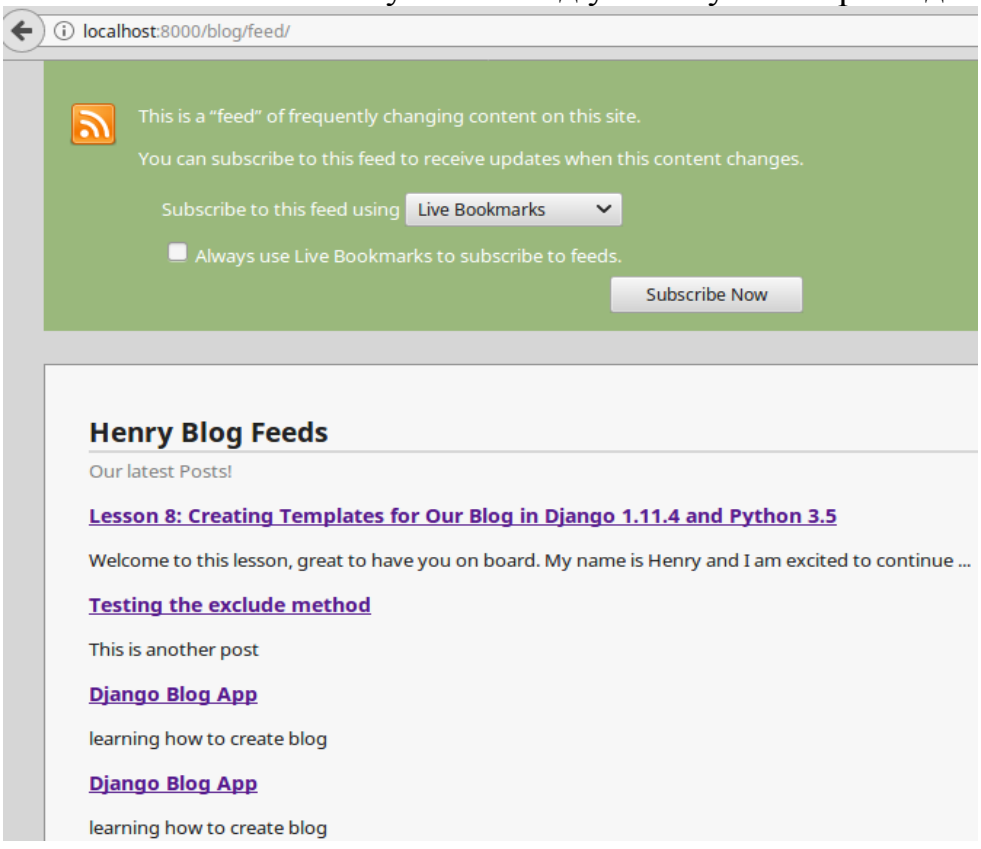

Рис.91. Сторінка RSS-каналів для блогу

Останній крок - створити посилання на передплату RSS на бічній панелі блогу. У нашому шаблоні відкрийте **master.html** і переконайтеся, що він має такий код:

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
    <link rel="stylesheet" href="{% static
"css/bootstrap.min.css" %}">
</head>
<body>
  <div class="container">
      <div class="row">
          <div class="col-md-8">
              {% block content %}

              {% endblock %}
          </div>
          <div class="col-md-4">
              <h3 class="page-header">Muvalab Blog</h3>
              <p><a href="{% url "blog:post_feed" %}"
target="_blank">Subscribe to RSS Feed</a></p>
          </div>
  </div>
</body>
</html>
```

У рядку 19 ми додали цей код: ** Subscribe to RSS Feed **, які додають посилання для підписки на канал RSS на бічній панелі блогу. Тепер відкрийте наступне посилання у своєму веб-переглядачі:

http://localhost:8000/blog/

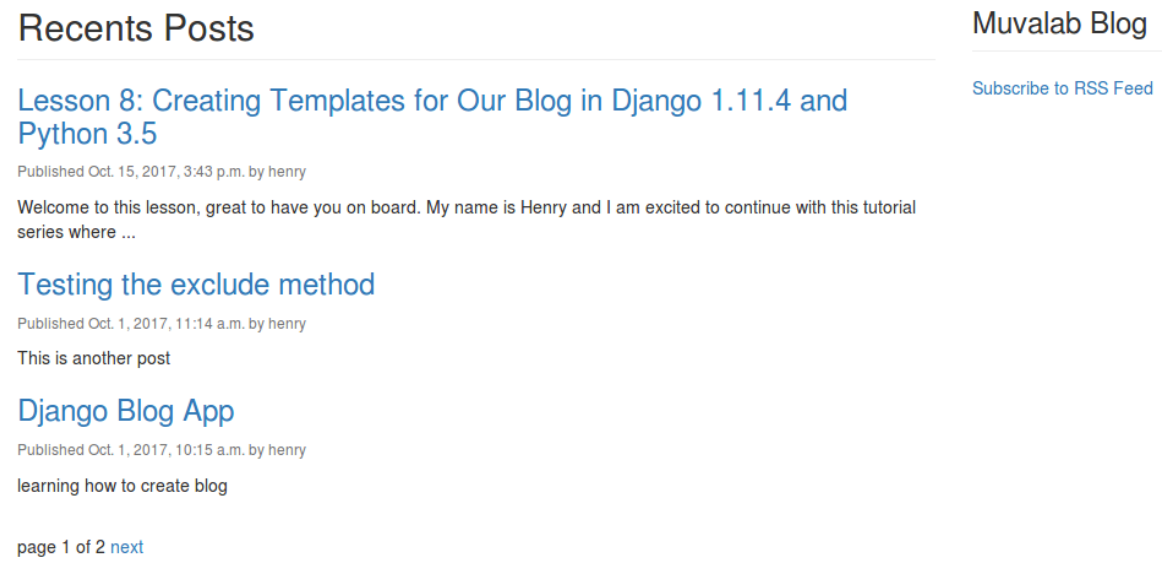

Рис.92. Сторінка з посиланням підписки на RSS-канали

Наведене вище зображення показує, що наше посиланя підписки було успішно додане.