НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Веб-додаток для створення та управління онлайн-візитівками»

Виконала:
студентка IV курсу, групи КП-51
Пивоварчук Олександра Валеріївна _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Т.М. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
доц. кафедри ММСА інституту прикладного
системного аналізу, доц., к.т.н. Дідковська М.В. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія» (Програмне забезпечення комп'ютерних та інформаційно-пошукових систем)

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Пивоварчук Олександрі Валеріївні

1. Тема проекту «Веб-додаток для створення та управління онлайн-візитівками», керівник проекту Заболотня Тетяна Миколаївна, к.т.н., доцент, затверджені наказом по університету від «22» травня 2019 р. №1331-С

2. Термін подання студентом проекту «16» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - огляд існуючих програмних рішень;
   - обґрунтування вибору засобів реалізації;
   - структурно-алгоритмічна організація;
   - опис реалізації програмних засобів.

5. Перелік обов'язкового графічного матеріалу:
   - структура бази даних (креслення);
   - діаграма прецендентів (креслення);
   - діаграма зв'язків компонентів системи (плакат);
   - дерево проблем (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	20.10.2018	
2.	Розроблення та узгодження технічного завдання	18.11.2018	
3.	Розроблення структури web-додатку	20.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення дизайну сторінок та графічних елементів	10.02.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	22.02.2019	
7.	Програмна реалізація web-додатку	24.03.2019	
8.	Тестування web-ресурсу	29.03.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	05.04.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	12.04.2019	
11.	Підготовка графічної частини дипломного проекту	23.04.2019	
12.	Оформлення документації дипломного проекту	27.05.2019	

Студент О.В. Пивоварчук

Керівник проекту Т.М.Заболотня

АНОТАЦІЯ

Даний дипломний проект присвячено розробленню веб-додатку для створення та управління онлайн-візитівками. Даний веб-застосунок – це рішення у вигляді ресурсу, в якому зберігаються ці односторінкові сайти / візитки / презентації. Загалом це сайт швидше для реклами та розміщення інформації про те чи інше підприємство, яке не має потреби у повноцінному сайті, але зробити якусь мітку про себе в інтернеті хоче.

У даній роботі було проведено аналіз доступних програмних рішень, який показав, що аналогу розроблюваному веб-застосунку, якому присвячена дана робота, не виявлено, що вказує на доцільність створення системи даного типу. На базі цього було сформовано проблему та критерії оцінювання. У результаті виконання аналізу даного завдання було сформовано основні вимоги до технологій розроблення, розглянуто найбільш відповідні, проведено обгрунтування вибору фреймворків для серверної та клієнтської частин, а також СУБД. Також було проведено збір та аналіз вимог до програмного застосунку. Після чого було розроблено структуру веб-додатку та архітектуру БД. Основною складовою ПЗ є модуль конструктора візитівки, який відповідає за збір візитівки із шаблону, плагіну та контенту. Користувачу надається можливість самостійно обрати шаблон, який найбільше задовольняє його потреби, підкорегувати його, обравши необхідні плагіни, та заповнити їх контентом. Також користувачу доступний перегляд статистики відвідувань сторінки візитівки. А при використанні плагіну форми зворотнього зв’язку, всі відгуки відображаються в особистому кабінеті.

Веб-додаток для створення управління онлайн візитівками дозволяє будь-якому користувачу розмістити візитівку та тим самим заявити про себе на просторах Інтернет.

ABSTRACT

This thesis project is devoted to the development of a web application for creating and managing online business cards. This web application is a solution in the form of a resource in which these one-page sites / business cards / presentations are stored. In general, this site is more likely to advertise and post information about a particular enterprise that does not need a full-fledged site, but wants to make some mark about itself on the Internet.

In this paper, an analysis of available software solutions was carried out, which showed that there was no analogue to the developed web application to which this work is devoted, which indicates the feasibility of creating a system of this type. On the basis of this, the problem and evaluation criteria were formed. As a result of the analysis of this task, the basic requirements for development technologies were formulated, the most suitable were considered, a justification was made for the choice of frameworks for the server and client parts, as well as the DBMS. Also collected and analyzed requirements for software applications. After that, the structure of the web application and the database architecture were developed. The main component of the software is the business card designer module, which is responsible for collecting the business card from the template, plug-in and content. The user is given the opportunity to independently choose a template that most satisfies his needs, correct it, select the necessary plugins, and fill them with content. The user can also view the statistics of visits to the page of the business card And when using the plug-in feedback form, all reviews are displayed in your account.

The web application for creating online business card management allows any user to place a business card and thereby declare itself on the Internet.

ДП.045440-01-90 Веб-додаток для створення та управління онлайн-візитівками.
Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Веб-додаток для	5	
	створення та управління		
	онлайн-візитівками.		
	Технічне завдання		
ДП.045440-03-81	Веб-додаток для	73	
	створення та управління		
	онлайн-візитівками.		
	Пояснювальна записка		
ДП.045440-04-51	Веб-додаток для	4	
	створення та управління		
	онлайн-візитівками.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Веб-додаток для	8	
	створення та управління		
	онлайн-візитівками.		
	Керівництво користувача		
ДП.045440-06-99	Веб-додаток для	1	
	створення та управління		
	онлайн-візитівками.		
	Структура бази даних.		
	ERD діаграма		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Веб-додаток для	1	
	створення та управління		
	онлайн-візитівками.		
	Функціональні		
	Можливості користувача.		
	Діаграма прецедентів		
ДП.045440-08-98	Веб-додаток для	1	
	створення та управління		
	онлайн-візитівками.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ВЕБ-ДОДАТОК ДЛЯ СТВОРЕННЯ ТА УПРАВЛІННЯ ОНЛАЙН-ВІЗИТІВКАМИ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Т.М. Заболотня

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.В. Пивоварчук

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-додаток для створення та управління онлайн-візитівками.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання звичайними користувачами, далекими від IT індустрії, в якості системи для створення та розміщення візитівок, презентацій та односторінкових сайтів.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Web-ресурс повинен забезпечувати такі основні функції:

1) робота з візитівками: створення, редагування, збереження, видалення;
2) наявність трьох типів шаблонів онлайн-візитівок;
3) можливість попереднього перегляду візитівки;
4) можливість перегляду візитівок неавторизованим користувачем;

Розробку виконати на платформі Python Django.

Додаткові вимоги:

1) наявність трьох типів шаблонів онлайн-візитівок;

2) наявність інтуїтивно зрозумілого інтерфейсу;

3) розділення прав доступу до функціональності додатку;

4) відображення бокової панелі зі списком доступних плагінів на сторінці редагування візитівки;

5) попередній перегляд візитівки.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

- «Функціональні можливості користувача. Діаграма прецедентів»;
- «Структура бази даних. EDR-діаграма».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи……..20.10.2018

Розроблення та узгодження технічного завдання……....18.11.2018

Розроблення структури web-ресурсу…….....20.12.2018

Розроблення дизайну сторінок та графічних елементів……..10.02.2019

Програмна реалізація web-ресурсу……....24.03.2019

Тестування web-ресурсу…….....29.04.2019

Підготовка матеріалів текстової частини проекту……..….12.04.2019

Підготовка матеріалів графічної частини проекту……...…23.04.2019

Оформлення технічної документації проекту……..…27.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-ДОДАТОК ДЛЯ СТВОРЕННЯ ТА УПРАВЛІННЯ ОНЛАЙН-ВІЗИТІВКАМИ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Т.М. Заболотня

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.В. Пивоварчук

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмний застосунок;

ІТ – інформаційні технології;

MTV – Model Template View;

СУБД – система управління базами даних;

NF – нормальна форма.

ВСТУП

За останне десятиліття Інтернет набув неймовірної популярності. В наш час майже не знайдеться людей, які б не користувалися даною технологією або хоча б не чули про неї. Соціальні мережі, Інтернет магазини, онлайн-курси, новини – це лише маленька частина того, що дає дана мережа. Кожного дня ми користуємося Інтернетом: вмикаємо музику ранком під час сніданку, дорогою на роботу переглядаємо актуальні новини або використовуємо навігатор у автівці, купуємо нові речі у Інтернет-магазині, бронюємо столик у бажаному ресторані на вечір, переглядаємо фільми і так далі. Сучасна людина не може уявити своє життя без Інтернету, навіть якщо її професія далека від IT. Розумні підприємці це чудово розуміють, тому активно розміщують свою рекламу на різних сервісах, соціальних мережах, ну і, звичайно, створюють свої сайти, адже зараз важко знайти поважаючу себе організацію, або малий бізнес, не кажучи вже про компанії-мільйонери, без особистого сайту на просторах Інтернет. Якщо раніше люди дізнавалися про новий продукт, парфюм або відкриття магазину з листівок, реклами на телеканалі або в метро, то зараз – це реклама в мережі Інтернет, яка найчастіше дуже дратує користувачів.

Таким підвищенням попиту на Інтернет-сайти або ресурси починають користуватися і програмісти. Кількість digital компаній з кожний роком все зростає, а ціни на їх послуги, відповідно, не зменшуються. Тому малому бізнесу, або бізнесу на перших етапах дуже скаладно впоратись з високою конкуренцією на ринку, в першу чергу, через брак коштів, адже розробити повноцінний сайт – недешеве задоволення. Також непотрібно забувати про те, що створити сайт – недостатньо, його ще треба розкрутити, підняти у рейтингу у пошукових системах, в першу чергу, за рахунок великого трафіку і так далі. На все це потрібно дуже багато часу, якого найчастіше немає під час створення свого

діла. Крім цього на перших етапах розвитку бізнеса найчастіше взагалі не має такої кількості інформації для створення повноцінного сайту. Але втрачати таку велику кількість потенційних клієнтів, яку може надати Інтернет також не хочеться. Звичайно можна розмістити інформацію про себе на окремих сторінках у соцмережах, але це більше як додаткове рекламне рішення, яке не може виступати у якості основного.

Веб-додаток для створення та управління онлайн-візитівками саме налаштований на клієнтів даного типу, тобто малий бізнес, або людей, які просто хочуть заявити про себе на просторах Інтернету. Будь-який користувач зможе швидко і легко створити строрінку або візитівку з інформацією по себе, свої послуги чи товари. Даною сторінкою у вигляді посилання можна також поділитися у соцмережах. Але плюсами даного сервісу саме для малого бізнесу є те, що на створення даної візитівки або односторінкового сайту непотрібно витрачати купу часу, шукати та наймати веб-розробників, платити за хостинг, оскільки все це дуже б’є по кишені. І ще одним плюсом є те, що у високому трафіку на даному ресурсі, а тобто і на кожній сторінці візитівки, зацікавлений не тільки власник, а інші клієнти, що розміщують візитівки, і також самі розробники даного сервісу.

Отже, веб-додаток для створення та управління онлайн-візитівками – це нове рішення, яке підтримає малий бізнес та допоможе у розміщенні інформації про себе, а тому дана тема є досить актуальною у наш час. Даний дипломний проект присвячений детальному огляду цього питання, опису запропонованого рішення та деталям його реалізації.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Аналіз існуючих аналогів

Веб-додаток для створення та управління онлайн-візитівками знаходиться на перетині двох кардинально різних типів Інтернет систем: CMS-систем та соціальних мереж. Сам процес створення візитівки або односторінкового сайту можна віднести по схожості на CMS системи, а те, що весь ресурс розміщений на одному сайті і по суті користувач просто створює свою окрему сторінку – до соціальних мереж.

В рамках виконання даного дипломного проекту були проаналізовані потенційні аналоги розроблюваного веб-застосунку, але не було знайдено точної копії або ідейно схожого додатку. Наведемо опис кількох найпопулярніших в дечому схожих веб-сервісів.

1.1.1. *CMS системи*

WordPress – є відкритою системою управління контентом, яка завдяки своїй простоті встановлення та використання широко використовується для створення веб-сайтів. Сфера використання – від блогів до складних веб-сайтів. Інтегрована система та доповнення в поєднанні з успішною архітектурою дозволяють створювати практично будь-який веб-проект на основі WordPress. Написана на мові програмування PHP з використанням бази даних MySQL [1].

Joomla! – глобальна система управління контентом відкрита для публікації інформації в Інтернеті. Підходить для великих і малих корпоративних веб-сайтів, онлайн-порталів, Інтернет-магазинів, сайтів спільноти та особистих сторінок. До функцій Joomla входять: гнучкі інструменти керування обліковими записами, інтерфейс для керування мультимедійними файлами, підтримка створення багатомовних сторінок, система управління рекламною кампанією, адресна книга користувача, голосування, інтегрована функція пошуку і кліки, редактор WYSIWYG, система шаблонів, підтримка меню, управління потоками новин,

XML-RPC API для інтеграції з іншими системами, підтримка кешованих сторінок і великий набір готових до використання доповнень. Joomla! написано на PHP за допомогою архітектури MVC. База даних MySQL, PostgreSQL або MS SQL використовується для зберігання інформації [2].

Wix.com – міжнародна хмарна платформа, написана на Scala, для створення і розвитку Інтернет-проектів, яка дозволяє конструювати сайти і їх мобільні версії на HTML5 с допомогою інструментів drag-and-drop. Розширювати функціональність сайтів можна за рахунок додатків, розроблених Wix або сторонніми компаніями [3].

1.1.2. *Соціальні мережі*

Facebook. Користувачі Facebook мають можливість створювати профілі з фотографіями, списками, контактні дані та іншу особисту інформацію. Вони можуть спілкуватися з друзями та користувачами, які використовують приватні чи публічні повідомлення. Користувачі також можуть створювати та приєднуватися до групи інтересів і "сторінок уподобань". Деякі з цих сторінок підтримують організацію як засіб реклами [4].

Instagram – соціальна мережа, що базується на обміні фотографіями, дозволяє користувачам робити фотографії, застосовувати до них фільтри, а також поширювати їх через свій сервіс і низку інших соціальних мереж. Instagram також дуже часто використовується для реклами продукції та послуг. Багато компаній створюють окремі аккаунти саме для виставляння товарів та його опису і у наступному – продажу [5].

1.1.3. *Результати аналізу*

Всі вищерозглянуті системи мають частково схоже функціональне наповнення, але не виступають прямими конкурентами розроблюваного у даній роботі продукту. Проаналізувавши табл. 1.1, можна дійти висновку, що по суті аналог даного додатку знайти не просто, адже в таблиці представлені системи, які кардинально відрізняються один від одного.

Таблиця 1.1

Порівняльна характеристика проаналізованих програмних засобів

№		WordPress	Joomla!	Wix.com	Facebook	Instagram
1	Збереження декількох візитівок на одному аккаунті	–	–	–	+	+/–
2	Створення бажаного дизайну при редагуванні шаблонів	+	+	+	–	–
3	Розміщення системи на одному сайті з можливістю створення окремих сторінок	–	–	–	+	+
4	Наявність трьох видів шаблонів	–	–	–	–	–
5	Наявність трьох типів ролей користувачів	–	–	–	+	+

Тому те, що є в CMS системах, відсутнє в соціальних мережах і навпаки. Так, наприклад, Facebook надає можливість розміщення декілької рекламних сторінок на одному аккаунті, а ось Instagram – не зовсім, адже Instagram має можливість перемикання між акаунтами не виходячи із них, але при цьому один користувач не може мати дві окремі сторінки.

Користувачам Facebook доступне створення декількох публічних сторінок/груп тощо.

Також соціальні мережі взагалі не мають можливості створення бажаного дизайну сторінок. CMS системи ж навпаки – якраз на це й орієнтовані. Головною задачею даних систем є створення сайтів опираючись на дефолтні шаблони з можливістю додавання плагінів та корегування вигляду та наповнення сайту під конкретні потреби.

Під час аналізу за критерієм 3 також схожими виявились соціальні мережі, а от CMS системи не мають можливості створення різних сайтів на одному і тому ж доменному імені.

Отже, проведений аналіз дозволяє зробити висновок про необхідність розроблення веб-додатку для створення та управління онлайн-візитівками.

1.2. Обґрунтування вибору вимог до програмного застосунку

Велика кількість людей не потребує повноцінного сайту, але має необхідність в місці в мережі Інтернет, куди можна помістити інформацію про свої послуги. Веб-додаток для створення та управління онлайн-візитівками – це рішення у вигляді ресурсу, в якому зберігатимуться ці односторінкові сайти / візитівки / презентації. Загалом це сайт швидше для реклами та розміщення інформації про те чи інше підприємство, яке не має потреби у повноцінному сайті, але зробити якусь мітку про себе в Інтернет хоче.

Проведений у попедніх пунктах аналіз існуючих програмних рішень допомагає у формулюванні основних вимог до веб-додатку для створення та управління онлайн-візитівками.

Таким чином, отримуємо такий список вимог:

1. Можливість збереження декількох візитівок у особистому аккаунті.
2. Можливість створення бажаного дизайну при редагуванні шаблонів (вже закладених у системі).

3. Розміщення всієї системи на одному сайті з можливістю створення окремих сторінок для користувачів. Під окремою сторінкою розуміється створення своєї особистої візитівки з можливістю керування розміщенням елементів та взагалі зовнішнім виглядом.
4. Наявність трьох основних видів шаблонів (типів) онлайн-візитівок: візитівка, односторінковий сайт, презентація.
5. Наявність мінімум трьох типів ролей користувачів: адміністратор, зареєстрований користувач (клієнт), неавторизований користувач.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

Одним із важливих кроків при розробленні програмного забезпечення є вибір засобів реалізації. Для даного веб-застосунку необхідно обрати мову програмування та фреймворк для веб-розроблення, також визначитись із СУБД та технологіями клієнтської частини.

2.1. Вибір технологій для розроблення серверної частини

В даний час існує безліч мов програмування зі своїми фреймворками та бібліотеками для веб-розроблення. Розглянемо найпопулярніші з них.

2.1.1. *Python Django*

Python є однією з найбільш використовуваних мов програмування завдяки простоті у вивченні, дизайну і гнучкості, що робить її практично досконалою мовою програмування [7]. Існує ряд причин, за якими Python можна називати такими гучними словами:

- простота у вивченні;
- чистота і читабельність;
- різнобічність;
- швидкість написання;
- цілісність дизайну.

Django є надзвичайно популярним і повнофункціональним серверним веб-фреймворком, написаним на Python. Розглянемо детальніше переваги даного фреймворку.

Гнучкість – даний фреймворк може працювати з будь-якою клієнтською платформою і може доставляти контент практично в будь-якому форматі (включаючи HTML, RSS-канали, JSON, XML і так далі) [6].

Швидкість – Django був розроблений, щоб допомогти розробникам створити додаток, що працює швидше, на скільки це можливо. Це включає в себе формування ідеї, розроблення і випуск проекту, де Django забезпкчує економію часу і ресурсів на кожному з цих етапів. Таким чином, даний фреймворк можна назвати ідеальним рішенням для розробників, для яких питання дедлайна стоїть в пріоритеті.

Повна комплектація – Django працює з десятками додаткових функцій, які заповнюють аутентифікацію користувача, карти сайту, адміністрування вмісту, RSS і багато інших. Дані аспекти допомагають здійснити кожен етап веб-розроблення.

Безпека – Django допомагає розробникам уникати багатьох поширених помилок безпеки, надаючи інфраструктуру, яка була розроблена у якості «оптимального рішення» для автоматичного захисту сайту. Django надає захист від помилок, пов'язаних з безпекою, що становлять загрозу для проекту, такі поширені помилки, як ін'єкції SQL, крос-сайти підробки, clickjacking і крос-сайтовний скриптинг. Для ефективного використання логінів і паролів, система користувальницької аутентифікації є ключем [6].

Масштабованість – Django використовує компонентну "shared-nothing" архітектуру (кожна частина архітектури не залежить від інших, і отже, може бути замінена або змінена при необхідності). Фреймворк Django найкращим чином підходить для роботи з високими трафіками.

Зручність в супроводі – код Django написаний з використанням принципів і шаблонів дизайну, які заохочують створення підтримуваного і багаторазового коду. Зокрема, він використовує принцип Do not Repeat Yourself (DRY), тому немає непотрібного дублювання, що зменшує кількість коду [6].

Звичайно крім переваг існують також і недоліки:

- Django використовує шаблон маршрутизації з вказанням URL;
- Django занадто монолітний;
- все базується на ORM Django;
- компоненти розгортаються спільно;
- необхідне вміння володіти всією системою фреймворку для налаштування та подальшої коректної роботи з веб-застосунком.

Отже, можемо зробити висновок, що Python Django досить добре задовольняє потреби даного веб-застосунку. Можливість масштабування

та гнучкість можна використати в розроблюваному веб-додатку, оскільки він має досить складну архітектуру та у майбутньому перспективу розвитку. Ще однією перевагою, важливою для даного веб-додатку, є читабельність та супровід коду, в яких Django теж непогано себе проявляє. І звичайно, враховуючи той факт, що створюване програмне забезпечення має дуже об'ємну клієнтську частину, особливо конструктор візитівки, важливу роль грає швидкість взаємодії системи з користувачем.

Нижче розглянемо Django у зіставленні з іншими фреймворками та бібліотеками.

2.1.2. *PHP і Python Django*

PHP можна назвати найсильнішим конкурентом Python, обидві ці мови дуже популярні. Однак у Django є декілька переваг, якими не може похвалитися PHP. Нерідко ці переваги є вирішальними для розробників. Вибір між PHP і Python – не найпростіше завдання [8]. Порівняємо Python Django та PHP.

Кращий дизайн – PHP був розроблений спеціально для веб-розроблення, але Django базується на більш надійній мові. Хороший код простіше написати в Python, в порівнянні з PHP.

Python і довгостроковість – PHP дуже хороший в короткостроковій перспективі. Проте, при закінченні першої фази розробки виникають труднощі з більш глибокими можливостями. Тут Python виступає в ролі явного фаворита.

Простий синтаксис – написання коду в Python відбувається набагато простіше і швидше.

Фактор читабельності – PHP слідує класичному підходу, але у Python більш суворі правила ідентифікації.

Інструменти для усунення помилок – пакети Python містять всі необхідні інструменти для виправлення помилок.

Управління пакетами – в Python це реалізовано дуже ефективно: існує можливість писати, читати і ділитися пакетами таким чином, щоб

інші розробники могли використовувати їх в інших додатках.

Отже, по даній порівняльній характеристиці можна зробити висновок, що конкретно для даного проекту більше підходить саме Python Django, адже система має досить складну архітектуру, тому одним із головних критеріїв є саме читабельність коду та його підтримуваність. В цьому Django з Python мають перевагу, адже даний фреймворк закладає необхідний шаблон проектування, що у майбутньому полегшує процес розширення проекту, на відміну від PHP, який не має жорсткої структури проекту. Саме це призводить до повторюваності коду та відсутності модульності, тому проект на PHP врешті-решт стає важко підтримуваним. Також необхідно зазначити, що мова Python вважається однією з найлегших для читання та розуміння, чим не може похизуватися PHP.

2.1.3. *Node.js і Python Django*

Важко порівняти Django і Node.JS, тому що вони є принципово різними. Django – це веб-фреймворк Python, а Node.JS – бібліотека, написана для JavaScript.

Мова програмування – Django використовує Python, який має величезні ресурси з документації та процвітаючу спільноту.

Node.JS використовує C, C++ і JavaScript. JavaScript є елегантною і динамічною мовою, що слугує для забезпечення досить легкого процесу входження у розроблення з використанням даної бібліотеки.

Node.JS – це бібліотека з повним стеком, що дозволяє уникнути проблем з FrontEnd і BackEnd частинами розроблення. Може бути невелика команда, але все одно буде витрачатися більше часу на запуск через складність JavaScript [9].

Архітектура – Django використовує шаблон Model View Template (MVT) для організації, що дуже схоже на контролер представлення моделі. Модель – це безпосередньо дані або інформація, а уявлення – спосіб її розміщення. Контролер – це опції, надані користувачу, але MVT організовує всю інформацію в самій структурі. Він не має стану і

легше реагує на нові додатки (хоча і не інтуїтивно зрозумілий).

Node.JS використовує кероване подіями програмування, в якому вся архітектура управляється «подіями» або призначеним для користувача вибором і повідомленнями з інших програм. Основний цикл має функцію зворотного виклику при виявленні події.

Веб-розроблення в основному не дотримувалось об'єктно-орієнтованого програмування, тому що в перші дні веб-розроблення все було орієнтовано на сторінки за допомогою сценаріїв на сторінці. MVT відмінно підходить для розроблення на стороні сервера, але програмування на основі подій є односпрямованим, а не розділяє сторони клієнта і сервера [9].

Безпека – Django виграє у безпеці, але в основному тому, що потрібно багато досвіду і впевненості, щоб отримати таку саму безпеку від Node.JS. JavaScript має всі засоби правильного складання безпеки, але є ризик щось пропустити.

Засноване на Python програмування Django забезпечує кращий, більш простий захист без великого досвіду.

Шаблони і документація – Django використовує власну систему шаблонів, яка допомагає скоротити кількість часу, який витрачається на вибір правильного. Він досить швидко запускається і працює з великою кількістю документації, яка допомагає прийняти ці рішення.

Node.JS залишає розробника трохи завислими, з повністю відкритою системою. Це може бути корисно для розробників, які креативні і досвідчені, тому що допомагає трохи вийти за межі своєї власної голови, але для початківця – втрачається багато часу, намагаючись зрозуміти, як почати [9].

Отже, як бачимо з порівняльної характеристики Node.JS та Django, обидва фрейморки досить хороші, кожен має свої переваги та недоліки, кожні з яких не можна остаточно назвати плюсом чи мінусом, адже все це залежить від конкретного проекту. Якщо брати саме даний веб-додаток, то

свобода архітектури в Node.JS виступає більше недоліком ніж перевагою, адже закладена структура Django полегшує підтримуваність та логіку розуміння проекту. Також великою перевагою Django виступає саме забезпечення безпеки веб-додатку, у Node.JS, як ми бачимо, ця функція не передбачена системою. Тому для веб-додатку для створення та управління онлайн-візитівками більше підходить саме Django.

2.1.4. *Python Flask і Python Django*

Flask – це ще один широко використовуваний веб-фреймворк. На відміну від Django, орієнтованого на поставку товару «з коробки» з готовими рішеннями для кожного завдання, Flask нагадує конструктор LEGO, з яким можна побудувати все, що потрібно, використовуючи величезний набір зовнішніх бібліотек і доповнень. Flask дає можливість приєднання нових модулів, коли приходить час, замість початкового завалювання системи деталями [10].

Архітектура:

- Flask: Немає жорсткої структури. Можна розмістити моделі, контролери та ініціалізацію в одному файлі, можна – в різних. Спочатку проект складається з порожньої папки. Структура формується по ходу розроблення [10].
- Django: Складається з проекту, який ділиться на додатки. Проект – коренева папка і глобальні налаштування. Додатки – це функціональні можливості сайту, розбиті по різних модулях. У кожного проекту вони свої.

Розширення:

- Flask: Встановлюється практично порожнім, але легко розширюється сторонніми бібліотеками. Зазвичай для вирішення однієї задачі є кілька бібліотек: вибір залежить від розміру команди розробників і навантаження на додаток. Так фреймворк стає гнучким.

- Django: Також можна встановити сторонні бібліотеки. Однак в ній замінити щось вбудоване буде вже складно. Django підходить для стандартного набору завдань і, якщо з реальністю це не сходиться, від Django треба йти.

Масштабування:

- Flask: Є blueprints. Вони схожі на додатки Django, але мають кілька відмінностей. Креслення поділяють функціональність програми: кожен має свій набір функцій, шаблонів і моделей. Однак blueprint – не є додатком. Вони скоріше схожі на інтерфейси, за якими відбувається масштабування. Один blueprint може бути основою декількох різних компонентів, тому його можливо багаторазово зареєструвати в одному проекті [10].
- Django: Розширюється додатками, тими які були в пункті про архітектуру. Швидке масштабування – частина філософії фреймворка, тому структура проекту будується на додатках з самого початку. Вони ні до чого не прив'язані і можуть використовуватися в різних проектах [6].

Робота з БД:

- Flask: Немає власної ORM, тому зазвичай підключається бібліотека SQLAlchemy або інші.
- Django: Комплектується вбудованою Django ORM. Також можна використовувати чистий SQL [10].

Отже, виконавши порівняльний аналіз Flask і Django було важко прийти до остаточного рішення про вибір фреймворку, оскільки обидва написані на Python та мають ряд переваг та недоліків. Але все ж таки обраний був саме Django, який трошки більше підходив саме даному веб-застосунку по причині наявності в ньому жорсткої структури, яка вже була зазначена у попередніх порівняннях, та ORM, яка все ж таки полегшує написання коду. Також саме наявність у Django додатків робить роботу з

ним більш зручною та приємною, що в свою чергу грає роль у швидкості розроблення та можливості розширення проекту.

2.1.5. *Висновки*

Проаналізувавши всі найпопулярніші фреймворки та бібліотеки для веб-розроблення серверної частини, прийшли до висновків, що найбільш доцільним для даного веб-застосунку є застосування саме Django на мові програмування Python.

Основними вимогами до фреймворку та мови програмування були читабельність та структурованість коду, швидкодія, можливість використання шаблонів проектування (Django одразу закладає MTV), а також можливість розширення проекту та його гнучкість. Зазначимо, що всі вище представлені фреймворки та бібліотеки мають досить високі показники по кожним з цих пунктів, але вирішальним зіграв той факт, що крім чудових показників за наведеними критеріями, Django має можливість використання такого модулю як CMS, що значно спрощує створення конструктора візитівок – одну із найважливіших та найобширніших можливостей даного проекту.

2.2. Вибір СУБД

Бази даних призначені для структурованого зберігання і швидкого доступу до різних даних. Кожна БД, крім самих даних, повинна мати певну модель роботи, за якою буде виконуватися оброблення інформації. Для управління БД використовуються СУБД.

Реляційні СУБД дозволяють розміщувати дані в таблицях, пов'язуючи рядки з різних таблиць і, таким чином, пов'язуючи різні, об'єднані логічно дані. Перед тим, як використовувати можливість зберігання даних, необхідно створити таблиці певного розміру і вказати тип даних для кожного стовпця. Стовпці представляють поля даних, а самі дані розміщені в рядках [11].

У базі даних NoSQL запис зазвичай зберігається як документ JSON. Для кожного елемента, значення зберігаються в якості атрибутів в єдиному

документі. У такій моделі дані оптимізовані для інтуїтивно зрозумілого розроблення і горизонтальної масштабованості [17].

Розглянемо одні із найпопулярніших СУБД.

2.2.1. *PostgreSQL*

PostgreSQL – це система управління базами даних з відкритим вихідним кодом корпоративного класу. Він підтримує як SQL для реляційних, так і JSON для нереляційних запитів. PostgreSQL підтримується досвідченою спільнотою розробників, які внесли величезний внесок у створення високонадійної системи СУБД [12].

PostgreSQL підтримує розширені типи даних і поліпшену оптимізацію продуктивності, функції, доступні тільки в дорогій комерційній основі даних, такій як Oracle і SQL Server.

PostgreSQL пропонує безліч функцій, які допомагають розробникам створювати додатки, а також допомагають адміністраторам створити відмовостійку середу, захищаючи цілісність даних.

PostgreSQL не просто реляційна, а об'єктно-реляційна СУБД. Це дає їй деякі переваги над іншими SQL базами даних з відкритим вихідним кодом, такими як MySQL, MariaDB і Firebird [12].

Фундаментальна характеристика об'єктно-реляційної бази даних – це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени і індекси. Це робить PostgreSQL неймовірно гнучкою і надійною. Серед іншого, дана СУБД вміє створювати, зберігати та видавати складні структури даних.

Найбільш важливі особливості PostgreSQL:

- сумісність з різними платформами, використовує всі основні мови і проміжне ПЗ;
- наявність найскладнішого механізму блокування;
- підтримка багатоверсійного управління паралелізмом;
- використання зрілих функцій програмування на стороні сервера;
- забезпечення стандарту ANSI SQL;

- підтримка клієнт-серверної мережевої архітектури;
- підтримка SSL на основі журналів і реплікації на основі тригерів;
- забезпечення резервного серверу і високої доступності;
- об'єктна-орієнтованість і сумісність з ANSI-SQL2008;
- підтримка JSON дозволяє зв'язуватися з іншими сховищами даних, такими як NoSQL, які діють як федеративний концентратор для баз даних поліглотів.

Переваги PostgreSQL:

- можливість PostgreSQL запускати динамічні веб-сайти і веб-додатки в якості опції стека LAMP;
- забезпечення запису з випередженням в PostgreSQL робить її базою даних з високою стійкістю до відмов;
- доступність вихідного коду PostgreSQL на умовах ліцензії з відкритим вихідним кодом. Це дає свободу використовувати, змінювати і впроваджувати його відповідно до потреб бізнесу;
- підтримка PostgreSQL географічних об'єктів, тому можливо використовувати дану СУБД для сервісів на основі визначення місця розташування і географічних інформаційних систем;
- легкість використання Postgres, не потрібно багато тренувань, щоб почати роботу;
- адміністрування з мінімальними експлуатаційними витратами як для вбудованого, так і для корпоративного використання.

Недоліки PostgreSQL:

- фокусування PostgreSQL на сумісності, тобто, зміни, внесені для підвищення швидкості, вимагають більше роботи, ніж MySQL;
- нестабільна підтримка PostgreSQL, тобто, багато додатків з відкритим вихідним кодом підтримують MySQL, але можуть не підтримувати PostgreSQL [12].

Отже, можемо зробити висновок, що PostgreSQL досить непогане рішення при виборі СУБД. Якщо розглядати переваги та недоліки в рамках саме даного веб-додатку, то ця система є підходящою, адже, по-перше, вона є реляційною. Так як веб-додаток для створення та управління онлайн-візитівками закладає у собі складну архітектуру, відповідно і структура БД не проста, тому для кращого розуміння та роботи з СУБД краще обрати саме реляційну базу. Також позитивним є те, що PostgreSQL підтримує JSON, що досить полегшує роботу на початку проектування БД. Даний веб-застосунок закладає у собі велике навантаження на БД та серверну частину, адже система повина працювати швидко та якісно, тому висока стійкість до відмов та швидкодія дуже важливі, у PostgreSQL наявні дані переваги.

2.2.2. *PostgreSQL і MySQL*

MySQL – це одна з найпопулярніших і найпоширеніших СУБД в Інтернеті. Вона не призначена для роботи з великими обсягами інформації, але її застосування ідеально для Інтернет сайтів, як невеликих, так і досить об’ємних [13].

MySQL відрізняється високою швидкістю роботи, надійністю та гнучкістю. Робота з нею, як правило, не викликає великих труднощів. Підтримка сервера MySQL автоматично включається в поставку PHP.

Обидві системи управління базами даних, і MySQL, і PostgreSQL належать до реляційних. Далі розглянемо докладніше, чим саме відрізняються обидві СУБД [13].

Зберігання даних:

- MySQL: MySQL – це реляційна БД, для зберігання даних в таблицях використовуються різні DB-Engine (підсистеми зберігання), але робота з ними захована в самій системі. На синтаксис запитів і їх виконання DB-Engine не впливає. Підтримуються такі основні підсистеми зберігання MyISAM,

InnoDB, MEMORY, Berkeley DB. Вони відрізняються між собою способом запису даних на диск, а також методами зчитування [13].

- PostgreSQL: PostgreSQL є об'єктно-реляційною базою даних, яка працює тільки на одному двигуні – storage engine. Всі таблиці представлені у вигляді об'єктів та можуть успадковуватися, а всі дії з ними виконуються за допомогою об'єктно-орієнтованих функцій. Як і в MySQL всі дані зберігаються на диску, в спеціально відсортованих файлах, але структура цих файлів і записів в них дуже сильно відрізняється [12].

Можливість оброблення:

- MySQL: При виконанні запиту MySQL завантажує всю відповідь сервера в пам'ять клієнта, при великих обсягах даних це може бути не зовсім зручно [13].
- PostgreSQL: PostgreSQL підтримує використання курсорів для переміщення по отриманим даним. Отримуємо тільки покажчик, вся відповідь зберігається в пам'яті сервера баз даних. Цей покажчик можна зберігати між сеансами. Тут підтримується побудова індексів відразу для декількох стовпців таблиці. Крім того, індекси можуть бути різних типів, крім hash і b-tree доступні GiST і SP-GiST для роботи з містами, GIN для пошуку по тексту, BRIN і Bloom. PostgreSQL підтримує регулярні вирази в запитах, рекурсивні запити і успадкування таблиць [12].

Продуктивність:

- MySQL: У більшості випадків для організації роботи з базою даних в MySQL використовується таблиця InnoDB, ця таблиця представляє з себе B-дерево з індексами. Індекси дозволяють дуже швидко отримати дані з диска, і для цього буде потрібно менше дискових операцій. Але сканування дерева вимагає знаходження двох індексів, а це вже уповільнює роботу. Все це означає що

MySQL буде швидше PostgreSQL тільки при використанні первинного ключа [13].

- PostgreSQL: Вся заголовна інформація таблиць PostgreSQL знаходиться в оперативній пам'яті. Не можливо створити таблицю, яка буде не в пам'яті. Записи таблиці упорядковано відповідно до індексу, а тому можна їх дуже швидко витягти. Для більшої зручності можна застосовувати кілька індексів до однієї таблиці. В цілому PostgreSQL працює швидше, за виключенням використання первинних ключів.

Типи даних

Типів даних в PostgreSQL більше і вони більш різноманітні, є свої типи полів для певних видів даних, яких немає MySQL.

Підтримка JSON:

- PostgreSQL: Підтримка JSON в PostgreSQL дозволяє перейти до зберігання schema-less даних в SQL базі даних. Це може бути корисно, коли структура даних вимагає певної гнучкості: наприклад, якщо в процесі розроблення структура все ще змінюється або невідомо, які поля буде містити об'єкт даних. Тип даних JSON забезпечують перевірку коректності JSON, який дозволяє використовувати спеціалізовані JSON оператори і функції, вбудовані в PostgreSQL для виконання запитів і маніпулювання даними.
- MySQL: В MySQL 5.7.8 була додана підтримка вбудованих об'єктів JSON. Але, хоча існує безліч функцій і операторів для JSON, які тепер доступні в цих базах даних, вони не індексуються так, як JSON в PostgreSQL [13].

Багатовимірні масиви:

- PostgreSQL: Оскільки PostgreSQL – це об'єктно-реляційна база даних, масиви значень можуть зберігатися для більшості існуючих типів даних. Зробити це можна шляхом додавання квадратних

дужок до специфікації типу даних для стовпця або за допомогою виразу ARRAY. Розмір масиву може бути заданий, але це необов'язково.

- MySQL: MySQL так не вміє. Щоб зберігати такі масиви значень в традиційних реляційних базах даних, доведеться використовувати обхідний шлях і створювати окрему таблицю з рядками для кожного із значень масиву [13].

Отже, проаналізувавши порівняльну характеристику PostgreSQL та MySQL можемо сказати, що PostgreSQL в багатьох випадках випереджає MySQL, але кожний проект має свої особливості. Для даного веб-застосунку все ж таки обрана була PostgreSQL, по-перше, через можливість завантаження великої кількості даних. По-друге, в даному проекті одним іх критеріїв є швидке подання та зміна даних, які знаходяться в БД, тому PostgreSQL, зі своєю високою продуктивністю, повністю підходить для часткового вирішення даного завдання.

2.2.3. *PostgreSQL і MongoDB*

MongoDB – документо-орієнтована СУБД з відкритим вихідним кодом, яка не потребує опису схеми таблиць. Класифікована як NoSQL, використовує JSON-подібні документи і схему бази даних.

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які притаманні об'єктно-реляційних баз даних [14].

На відміну від реляційних баз даних MongoDB пропонує документо-орієнтовану модель даних, завдяки чому MongoDB працює швидше, має кращу масштабованість, її легше використовувати.

Структура даної СУБД полягае в тому, що MongoDB містить в собі безліч колекцій, вони ж містять безліч документів (об'єктів), які в свою чергу містять пари ключ-значення. Документ має динамічну схему, що означає, що документи в одній і тій же колекції не повинні мати однакову

множину пар ключ-значень та те, що типи їх можуть бути різними. Наведемо відмінності PostgreSQL та MongoDB:

Структура бази даних:

- PostgreSQL: PostgreSQL – це система управління об'єктно-реляційними базами даних (ORDBMS) з акцентом на розширюваність і відповідність стандартам. PostgreSQL є ACID-сумісною, транзакційною, має оновлювані і матеріалізовані уявлення, тригери і зовнішні ключі. Дана СУБД також підтримує функції і процедури. PostgreSQL використовує таблиці, обмеження, тригери, ролі, процедури, що зберігаються та подання в якості основних компонентів, з якими виконується робота. Таблиця складається з рядків, і кожен рядок містить один і той же набір стовпців. PostgreSQL використовує первинні ключі для унікальної ідентифікації кожного рядка (або запису) в таблиці, а зовнішні ключі для забезпечення відправної цілісності між двома пов'язаними таблицями. PostgreSQL також підтримує безліч функцій NoSQL [12].
- MongoDB: MongoDB використовує JSON-подібні документи для зберігання даних без схеми. У MongoDB колекції документів не вимагають попередньо визначеної структури, і стовпці можуть відрізнятися для різних документів. MongoDB володіє багатьма функціями реляційної бази даних, включаючи виразну мову запитів і строгу узгодженість. Проте, оскільки MongoDB не має схеми, можна створювати документи без необхідності спочатку створювати структуру для документа [14].

Індексація

Індекси підвищують продуктивність бази даних, оскільки дозволяють серверу баз даних знаходити і витягати певні рядки набагато швидше, ніж без індексу. Але індекси додають певні витрати до системи бази даних в цілому, тому їх слід використовувати розумно.

Без індексу сервер бази даних повинен почати з першого рядка, а потім прочитати всю таблицю, щоб знайти відповідні рядки. Чим більше стовпців, тим дорожче операція.

- PostgreSQL: PostgreSQL включає вбудовану підтримку звичайного B-дерева і хеш-індексів. Індекси в PostgreSQL також підтримують такі функції: індекси виразів, що створюються з індексом результату виразу або функції, а не просто значенням стовпця, та часткові індекси, що досволяють індексувати тільки частину таблиці [12].
- MongoDB: індекси є кращими в MongoDB. Якщо індекс відсутній, кожен документ в колекції повинен бути знайдений, щоб вибрати документи, які були запитані в запиті. Це може сповільнити час читання.

Різниця для бізнесу:

- PostgreSQL: PostgreSQL, схоже, набуває все більшої популярності. Якщо необхідне готове рішення, сумісне із стандартами, транзакціями і ACID, а також широка підтримка функцій NoSQL, то варто спробувати PostgreSQL [12].
- MongoDB: MongoDB може бути відмінним вибором, якщо потрібна масштабованість і кешування для аналітики в реальному часі; однак дана СУБД не призначена для транзакційних даних (системи обліку і т.д.). MongoDB часто використовується для мобільних додатків, управління контентом, аналітики в реальному часі і додатків, пов'язаних з Інтернет-речами. Якщо немає чіткого визначення схеми, MongoDB може бути хорошим вибором [14].

Отже, після проведення порівняння MongoDB та PostgreSQL прийшли до висновку, що для даного проекту краще підходить PostgreSQL, адже проект закладає жорстку архітектуру, для можливості майбутнього розширення, читабельності коду та розуміння логіки модулів. Тому MongoDB як нереляційна база даних не дуже підходить для цього.

Для даного веб-додатку найкращим рішенням є використання саме реляційної БД з чіткою архітектурою таблиць та інших компнентів.

2.2.4. *Висновки*

Проаналізувавши найпопулярніші СУБД можна зробити висновок, що всі вони є досить хорошими рішеннями, але якщо обирати саме для даного веб-застосунку, то найбільш влучним є PostgreSQL, адже вона має кращі показники відмовостійкості та швидкодії, що є одними із основних вимог до даного продукту. Також дана СУБД не має обмежень по кількості даних, що в ній зберігаються. Також PostgreSQL є реляційною базою даних, що надає краще розуміння структури бази даних та облегшує роботу з нею, адже конкретний веб-додаток має непросту організаційну структуру, а тому і архітектуру БД. Також якщо брати найскладніший етап розроблення проекту – початок, то підтримка PostgreSQL функцій NoSQL досить гарне доповнення.

Отже, у якості СУБД для веб-додатку для створення та управління онлайн-візитівками обрана PostgreSQL.

2.3. Вибір технологій для розроблення клієнтської частини

Клієнтська частина сайту грає важливу роль у забезпечені його інтерективності та своренні враження саме на потенційного користувача системи. Для роботи з клієнтською частиною та забезпечення динамічної інтерактивності і була створена мова JavaScript.

У сучасному уявленні JavaScript – мова програмування загального призначення, підтримувана більшістю браузерів. Основною особливістю даної мови, є максимальна інтеграція з HTML і CSS. Початково JavaScript розроблявся для надання більшої динаміки web- сторінці.

JavaScript – одна із найпростіших, універсальних і ефективних мов, які використовуються для розширення функціональності веб-сайтів. JS допомагає з легкістю створювати візуальні ефекти на екрані, а також легко обробляти і обчислювати дані на веб-сторінках [15].

Код JavaScript виконується на процесорі користувача, а не на веб-сервері, що економить пропускну здатність і навантаження на веб-сервер.

Розглянемо детальніше переваги та можливості JS.

JavaScript працює на стороні клієнта. Фрагменти коду JavaScript не вимагають відправки на сервер для обробки. Це економить навантаження на стороні сервера. Коди JavaScript на веб-сайті обробляються з використанням ресурсів системи користувача.

JavaScript – легка мова для вивчення. Мова JavaScript проста у вивченні і має синтаксис, близький до англійського. Вона використовує модель DOM, яка надає безліч визначених функцій для різних об'єктів на сторінках, що дозволяє без труднощів розробити сценарій для вирішення призначених для користувача завдань.

JavaScript порівняно швидкий для кінцевого користувача. Оскільки код виконується на боці клієнта, оброблення завершується практично миттєво в залежності від завдання (завдання в JavaScript на веб-сторінках, зазвичай, прості, щоб не допустити перевантаження пам'яті), оскільки його не потрібно обробляти на веб-сервері і відправляти назад користувачеві, споживаючи як локальну, так і серверну пропускну здатність [15].

Платформа незалежна. Будь-який браузер з підтримкою JavaScript може розуміти і інтерпретувати код JavaScript. Будь-який код JavaScript може бути виконаний на різних типах обладнання, для якого написана програма JavaScript.

Подієва мова програмування. Будучи заснованою на подіях мовою, різні сегменти коду виконуються щоразу, коли в JavaScript відбувається певна подія. Простою мовою, сегмент коду виконується, коли користувач натискає кнопку або наводить покажчик миші на об'єкт [15].

Процедурні можливості програмування. Мова JavaScript включає в себе всі можливості процедурної мови. Розгалуження, зациклення, перевірка стану – ось деякі з тих можливостей, які можуть бути виконані на веб-сторінці.

Але звичайно для розроблення повноцінного веб-застосунку необхідно використання бібліотек чи фреймворків клієнтської часини, що полегшують та прискорюють написання коду веб-додатку.

Для розроблення була також використана бібліотека jQuery.

jQuery – бібліотека JavaScript, що фокусується на взаємодії JavaScript і HTML. Бібліотека jQuery допомагає легко отримувати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними. Також з її допомогою можна управляти анімацією, обробляти події, працювати з Ajax запитами. Все це завдяки API і підтримки цієї бібліотеки практично у всіх браузерах [16].

Ajax – це технологія чи підхід обміну інформацією з сервером і оновлення частин веб-сторінки без повного перезавантаження HTML документа. Цей обмін даними відбувається на фоні, і тому назва технології розшифровується як Asynchronous JavaScript and XML [16].

До переваг даної бібліотеки можна віднести:

- jQuery – крос-браузерна (працює однаково добре і сумісно з безліччю популярних браузерів);
- jQuery набагато простіше використовувати, ніж нативний JavaScript;
- бібліотека розширювана;
- jQuery дуже проста і має відмінну підтримку Ajax технології;
- ця бібліотека має відмінну і детальну документацію, яку можна почитати на сайті розробників.

Тобто JQuery сильно спрощує програмування на мові JavaScript.

Отже, для розроблення клієнтської частини була обрана мова програмування JavaScript з використанням бібліотеки jQuery, адже даний веб-додаток вимагає швидкодії та інтерактивності. Також перевагою даної мови та бібліотеки є те, що вони підтримувані багатьма браузерами, що у поєднанні із Python робить додаток кросбраузерним.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ ПЗ

Для логічної та правильної роботи веб-застосунку перед початком самого розроблення необхідно правильно організувати структуру програмного забезпечення. Структрурно-алгоритмічна організація впливає на швидкість розроблення, майбутнє розширення системи, а також на швидкість роботи додатку.

3.1. Вимоги до розроблюваного ПЗ

3.1.1. *Загальний опис системи*

Веб-застосунок для створення та управління онлайн-візитівками – це веб-додаток, який складається з клієнтської та адміністративної частин. Основна задача даного програмного забезпечення полягає в тому, щоб дати звичайному користувачу, далекому від веб-індустрії, можливість створити односторінковий сайт (візитівку) і тим самим заявити про себе на просторах Інтернет. Як було вже зазначено у попередніх розділах, даний застосунок орієнтований на малий та середній бізнес, якому на початку свого розвитку важко інвестувати гроші у розроблення власного повноцінного сайту і також недостатньо інформації, яку б можна було розмістити на ньому.

Даний додаток підтримує три типи користувачів:

- спостерігач – неавторизований (анонімний) користувач, якому доступний тільки перегляд клієнтської частини застосунку, а саме: перегляд головної сторінки, списку шаблонів та їх перегляд, перегляд візитівок користувачів ну і звичайно форма реєстрації для подальшої роботи у якості повноцінного користувача веб-додатком;
- користувач – авторизований користувач даного додатку, який окрім можливостей спостерігача має доступ до користувацької адмін-панелі (особистий кабінет), яка підтримує функції створення

та управління онлайн-візитівками, перегляд статистики та можливість перегляду списку відгуків на контакні форми;

- адміністратор – адмін даного веб-застосунку, який має окрему адмін-панель та можливість управління всім контентом на сайті, створення користувачів, та перегляд інформації про кожного з них, блокування та розблокування користувачів та активацію/деактивацію їх візитівок, а також додавання та видалення шаблонів візиток.

Також дана система підтримує три типи шаблонів:

- односторінковий сайт – шаблон повноцінного односторінкового сайту з можливістю додавання різних видів плагінів (додаткових функцій, таких як : гугл карта, форма зворотнього зв'язку, галерея, текст, контакти і т.д.), які створенні під кожний конкретний шаблон стилістично, та визначені конкретні місця можливості їх розташування на сторінці, що не буде псувати адаптивну версію візитівки;
- візитівка – шаблон короткої інформації, який має менше можливостей порівняно з односторінковим сайтом, дозволяє розміщувати потрібну інформацію у відповідні поля та мінімально змінювати його;
- презентація – шаблон, який дозволяє розмістити тільки відео, слайдер або картинку.

Для того, щоб краще зрозуміти як працює даний веб-додаток, відтворимо кроки стандартного користувача щодо роботи із застосунком у вигляді UML-діаграми на рис. 3.1. Перше, куди потрапляє користувач, – це головна сторінка сайту, на якій розміщена коротка інформація про сам продукт та його можливості. Для повноцінної роботи з даним застосунком потенційному користувачу потрібно пройти короткий шлях реєстрації з підтвердженням email-адреси, далі користувач переходить на сторінку вибору шаблону для створення онлайн візитівки. Після вибору потрібного

шаблону користувачеві потрібно заповнити необхідну інформацію для створення візитівки і далі він переходить до найцікавішої частини роботи з даний веб-додатком – наповнення контентом та вибір потрібних плагінів. Після чого користувач може попередньо переглянути візитівку та опублікувати її. Дана візитівка з'являється у його особистому кабінеті в списку створених ним візитівок (опублікованих та неопублікованих). Слід зазначити, що конкретному користувачеві доступна робота та управління тільки своїми візитівками, які представлені у відповідному пункті особистого кабінету.

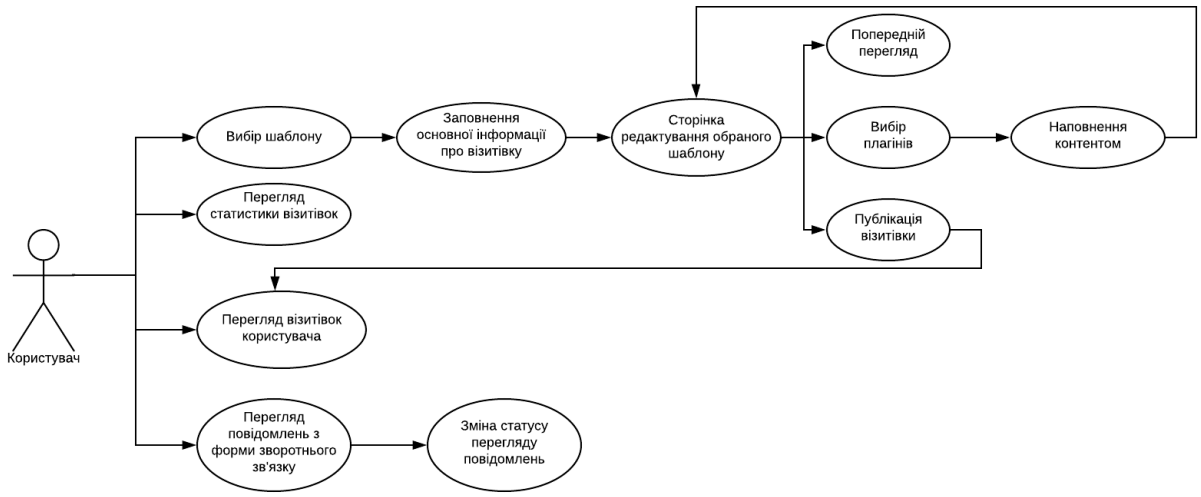


Рис. 3.1. UML-діаграма можливості дій авторизованого користувача

Головний акцент даного веб-застосунку зроблений на інтуїтивно зрозумілий інтерфейс та зручну роботу для будь-якого користувача, тому до програми не додавалося таких функцій, які б могли заплутати людину без достатніх знань в області IT.

Розглянемо функціональні можливості адміністратора:

- керування шаблонами візитівок (створення, редагування, видалення шаблонів);
- наповнення клієнтської частини сайту контентом та його зміна;

- керування даними про користувача (створення, редагування даних, видалення, блокування та розблокування користувача);
- керування візитівками користувачів (видалення, публікація, зняття з публікації);
- керування категоріями та типами шаблонів.

Перейдемо до самого простого типу користувачів – спостерігач. Даному користувачеві недоступний особистий кабінет користувача і будь-які наміри пов'язані зі створенням або управлінням візитівками ведуть його на форму авторизації у системі. Тобто даний тип користувача має обмежений доступ до роботи із системою, але йому все ж таки доступні деякі функції, такі як:

- перегляд списку шаблонів та їх прев’ю;
- перегляд списку візитівок користувачів та самих візитівок;
- авторизація та реєстрація у системі.

Функціональні можливості були представлені на рис. 3.2, для кращого розуміння даної системи.

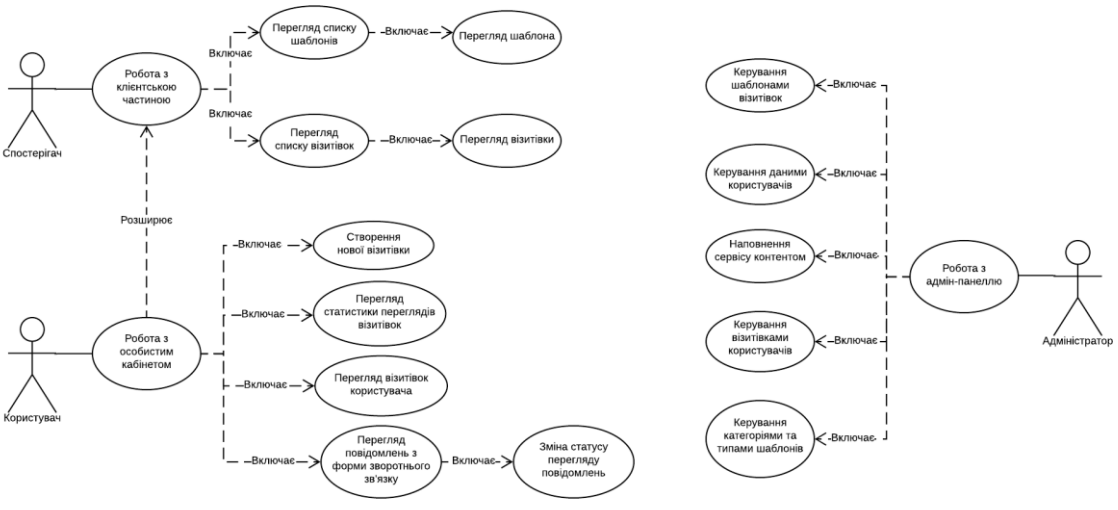


Рис. 3.2. Use-cases діаграма веб-застосунку для створення та управління онлайн-візитівками

3.1.2. *Вимоги до веб-додатку*

Веб-додаток для створення та управління онлайн-візитівками можна уявно поділити на декілька частин: особистий кабінет, адмін-панель та клієнтська частина. Для кращого формулювання вимог будемо розглядати кожну частину окремо.

Вимоги до особистого кабінету користувача

Основні функціональні можливості даної системи закладені саме в клієнтській адмін-панелі (особистому кабінеті). Її завданням є саме вибір шаблону та подальша робота з ним, тобто додавання контенту та плагінів, представлення статистики переглядів та перегляд повідомлень с форм зворотнього зв'язку. Для кращої постановки вимог до цього великого модулю веб-застосунку було виконано їх розділення на дві частини: вимоги до конструктору шаблонів та вимоги до кабінету користувача.

Функціональні та нефункціональні вимоги до кабінету користувача представимо у вигляді реєстру у табл. 3.1.

Таблиця 3.1

Реєстр вимог до кабінету користувача

FEAT	Зміст вимоги	Атрибути		
		Priority	Difficulty	Type
F 1	Реєстрація/авторизація користувача в системі.	HIGH	HIGH	FUNCTIONAL
F 2	Можливість редагування тільки своїх візитівок користувачу.	HIGH	HIGH	FUNCTIONAL
F 3	Представлення шаблонів у вигляді оглядового списку з короткою інформацією про кожен з них.	HIGH	MEDIUM	INTERFACE

Продовження табл. 3.1

F 4	Передача шаблонів для їх оглядового списку.	HIGH	MEDIUM	INTERFACE
F 5	Коректна фільтрація шаблонів по типам та категоріям.	HIGH	MEDIUM	FUNCTIONAL
F 6	Коректний пошук шаблону за назвою в оглядовому каталозі.	HIGH	MEDIUM	FUNCTIONAL
F 7	Коректний збір даних кількості переглядів конкретної візитівки.	HIGH	MEDIUM	FUNCTIONAL
F 8	Представлення статистики у вигляді графіку.	HIGH	MEDIUM	INTERFACE USABILITY
F 9	Передача візитівок конкретного користувача.	HIGH	MEDIUM	FUNCTIONAL
F 10	Коректна фільтрація візитівок за статусом опублікованості.	HIGH	MEDIUM	FUNCTIONAL
F 11	Представлення візитівок конкретного користувача з короткою інформацією про кожну з них.	HIGH	MEDIUM	INTERFACE
F 12	Представлення інформації профіля користувача.	HIGH	MEDIUM	INTERFACE
F 13	Можливість зміни інформації профіля користувача.	HIGH	MEDIUM	FUNCTIONAL
F 14	Коректна передача даних в профіль користувача.	HIGH	MEDIUM	FUNCTIONAL
F 15	Коректна передача даних з форм зворотнього зв’язку із всіх візитівок користувача.	HIGH	MEDIUM	FUNCTIONAL
F 16	Представлення списку відгуків з форми зворотнього зв’язку.	HIGH	MEDIUM	INTERFACE

Продовження табл. 3.1

F 17	Хешування паролю користувача.	HIGH	MEDIUM	FUNCTIONAL
F 18	Можливість зміни статусу відгуку з форми зворотнього зв’язку (переглянуто / непереглянуто).	MEDIUM	MEDIUM	FUNCTIONAL
F 19	Забезпечити доступ до кабінету тільки авторизованим користувачам.	MEDIUM	MEDIUM	FUNCTIONAL

Отже, якщо узагальнити вимоги до кабінету користувача, які вище наведені у табл. 3.1, необхідно реалізувати подання таких сторінок як: сторінка статистики, сторінка перегляду та роботи з власними візитівками, сторінка створення нової візитівки, сторінка редагування особистої інформації та сторінка відгуків із контактних форм. Для даних сторінок необхідно забезпечити коректну роботу та вивід даних. На сторінках у вигляді оглядових списків, таких як сторінка візитівок користувача та сторінка створення нової візитівки, необхідно забезпечити коректне сортування шаблонів за категоріями та типами, а також пошук за назвою.

Функціональні та нефункціональні вимоги до конструктора шаблонів представимо у вигляді реєстру у табл. 3.2.

Таблиця 3.2

Реєстр вимог до конструктора шаблонів

FEAT	Зміст вимоги	Атрибути		
		Priority	Difficulty	Type
F 20	Можливість вибору шаблону.	HIGH	MEDIUM	FUNCTIONAL

Продовження табл. 3.2

F 21	Можливість редагування тільки своїх візитівок користувачу.	HIGH	HIGH	FUNCTIONAL
F 22	Представлення попапу створення шаблону.	HIGH	MEDIUM	INTERFACE
F 23	Надати можливість прев'ю візитівки.	HIGH	MEDIUM	FUNCTIONAL
F 24	Занесення відповідних даних про створений шаблон до БД.	HIGH	HIGH	FUNCTIONAL
F 25	Надати можливість публікації візитівки.	HIGH	MEDIUM	FUNCTIONAL
F 26	Представлення сторінки прев'ю візитівки.	HIGH	MEDIUM	INTERFACE
F 27	Надати можливість вибору плагіну.	HIGH	HIGH	FUNCTIONAL
F 28	Представлення списку плагінів в боковій панелі.	HIGH	HIGH	INTERFACE
F 29	Надати можливість налаштування для кожного плагіна.	HIGH	HIGH	FUNCTIONAL
F 30	Представлення попапу налаштування плагіну.	HIGH	MEDIUM	INTERFACE
F 31	Збереження візитівки зі всіми редагуваннями та налаштуваннями.	HIGH	HIGH	FUNCTIONAL
F 32	Коректне представлення візитівки після збереження (передача зконструйованого шаблону).	HIGH	HIGH	FUNCTIONAL

Продовження табл. 3.2

F 33	Надати можливість перегляду візитівок та прев'ю шаблонів будь-якому користувачу (навіть неавторизованому).	HIGH	MEDIUM	FUNCTIONAL

Отже, якщо узагальнити вимоги до конструктору шаблонів, наведені в табл. 3.2, то необхідно реалізувати можливість вибору шаблону та подальшого створення візитівки на його основі, також редагування та видалення візитівки, налаштування плагінів та коректне збереження всієї структури візитівки в бд з можливість подальшого її подання. Також необхідно забезпечити представлення панелі налаштування та роботи з візитівкою, надати можливість попереднього перегляду візитівки та зміна її статусу (опубліковано/неопубліковано).

Вимоги до адмін-панелі

Вагому частину основного функціонального наповнення закладено також в адмін-панель, якою управляє адміністратор. Функції додавання шаблонів, налаштування та додавання їх типів та категорій, управління візитівками користувачів та взагалі робота з ними, а також наповнення контентом – всі ці обов'язки закладені у роль адміністратора веб-застосунку. Звідси витікають і основні вимоги до створення та зручного наповнення адмін-панелі для керування ПЗ. Всі ці вимоги винесені до реєстру вимог, що знаходиться у табл. 3.3.

Таблиця 3.3

Реєстр вимог до адмін-панелі

FEAT	Зміст вимоги	Атрибути		
		Priority	Difficulty	Type
F 34	Доступність амін-панелі тільки адміністратору.	HIGH	MEDIUM	FUNCTIONAL
F 35	Представлення адмін-панелі відповідно до дизайну.	HIGH	HIGH	INTERFACE
F 36	Можливість додавання, редагування та видалення типів та категорій шаблонів.	HIGH	HIGH	FUNCTIONAL
F 37	Представлення сторінок роботи з типами та категоріями шаблонів.	HIGH	HIGH	INTERFACE
F 38	Можливість управління користувачами (створення, редагування даних, видалення, блокування та розблокування).	HIGH	HIGH	FUNCTIONAL
F 39	Представлення сторінок роботи з користувачами.	HIGH	HIGH	INTERFACE
F 40	Можливість перегляду історії змін здійснених з панелі адміністратора.	HIGH	HIGH	FUNCTIONAL
F 41	Представлення історії змін здійснених з адмін-панелі.	HIGH	HIGH	INTERFACE
F 42	Представлення кнопки переходу з адмін-панелі на користувацьку частину сайту.	HIGH	LIGHT	INTERFACE

Продовження табл. 3.3

F 43	Можливість управління візитівками користувачів.	HIGH	HIGH	FUNCTIONAL
F 44	Представлення сторінки управління візитівками користувачів.	HIGH	MEDIUM	INTERFACE
F 45	Можливість зміни контенту веб-додатку.	HIGH	HIGH	FUNCTIONAL
F 46	Представлення сторінки зміни контенту.	HIGH	MEDIUM	FUNCTIONAL

Отже, узагальнемо вимоги до адмін-панелі, наведені в табл. 3.3. Необхідно реалізувати можливість управління з візитівками користувача (видалення, публікація, зняття з публікації), також керування даними про користувача (створення, редагування, блокування/розблокування), керування шаблонами (створення, редагування, видалення), їх типами та категоріями, а також забезпечити можливість наповнення сторінок контентом через адмін-панель та його редагування. Необхідно реалізувати представлення всіх сторінок з можливістю взаємодії з даним функціональним наповненням.

Вимоги до клієнтської частини

Клієнтська частина орієнтована більше на візуальне сприйняття користувача, який вперше потрапляє на сайт. Головна сторінка – це обличчя веб-додатку, і хоча її функціональне наповнення не найбільше, саме вона грає основну роль у бажані користувача далі працювати з даним сервісом. Також іншими не менш важливими сторінками клієнтської частини є сторінки перегляду шаблонів, візитівок користувачів та контактна форма. Функціональні та нефункціональні вимоги винесені до реєстру вимог у табл. 3.4.

Таблиця 3.4

Реєстр вимог до клієнтської частини

FEAT	Зміст вимоги	Атрибути		
		Priority	Difficulty	Type
F 47	Представлення головної сторінки як в дизайні.	HIGH	HIGH	INTERFACE
F 48	Забезпечити можливість наповнення контентом головної сторінки з адмін-панелі, його збереження в БД.	HIGH	HIGH	FUNCTIONAL
F 49	Представлення візитівок у вигляді оглядового списку з короткою інформацією про кожен з них.	HIGH	HIGH	INTERFACE
F 50	Передача даних візитівок на сторінку перегляду візитівок.	HIGH	HIGH	FUNCTIONAL
F 51	Надати можливість виконувати сортування візитівок по категоріям на сторінці перегляду візитівок.	HIGH	MEDIUM	FUNCTIONAL
F 52	Надати можливість пошуку по заголовку візитівки.	HIGH	MEDIUM	FUNCTIONAL
F 53	Представлення шаблонів у вигляді оглядового списку з короткою інформацією про кожен з них.	HIGH	HIGH	INTERFACE
F 54	Передача даних шаблонів на сторінку перегляду шаблонів.	HIGH	HIGH	FUNCTIONAL

Продовження табл. 3.4

F 55	Надати можливість виконувати сортування шаблонів по категоріям та типам на сторінці перегляду шаблонів.	HIGH	MEDIUM	FUNCTIONAL
F 56	Надати можливість пошуку по заголовку шаблону.	HIGH	MEDIUM	FUNCTIONAL
F 57	Представлення сторінки контактної форми.	LIGHT	MEDIUM	INTERFACE
F 58	Налаштування контактної форми на відправку повідомлень на емейл.	LIGHT	MEDIUM	FUNCTIONAL
F 59	Можливість зміни контенту зі сторінки контактної форми з адмін-панелі.	LIGHT	MEDIUM	FUNCTIONAL

Отже, узагальнемо вимоги до клієнтської частини, наведені в табл. 3.4. Необхідно реалізувати представлення таких сторінок як: головна, сторінка перегляду візитівок та сторінка перегляду шаблонів візитівок. Також забезпечити коректну їх роботу та правильний вивід даних. На сторінках у вигляді оглядових списків необхідно забезпечити коректне сортування за категоріями та типами, а також пошук за назвою.

Висновки

У результаті проведеного аналізу було встановлено вимоги до веб-додатку для створення та управління онлайн-візитівками. Всі вимоги, функціональні та нефункціональні, було занесено до реєстру вимог з відповідними помітками про тип. Також було визначено ролі користувачів та їх можливості, що також представлено у реєстрі вимог. Крім цього, було проаналізовано важливість та складність виконання кожної вимоги і

зазначено у реєстрі у відповідних колонках Priority та Difficulty. Всі вимоги у реєстрі відсортовані за пріоритетом.

3.2. Структурна організація програмного забезпечення

Даний веб додаток написаний на мові програмування Python з використанням фреймворку Django, що автоматично закладає певну структуру проекту з можливістю її розширення. У Django використовується шаблон проектування MTV. Основним принципом даної архітектури виступає те, що основний проект розбивається на додатки, кожен з яких, в свою чергу, використовує вище зазначений шаблон проектування, тобто складається із моделей, шаблонів та контролера (View). Також кожний додаток має маршрутизатор urls.py, який відповідає за зв'язок маршруту з функцією у контролері, у якому в свою чергу визначається використовуваний шаблон.

Основний проект Simpoo складається із додатків simpoo, controller, client та admin:

- simpoo – основний (кореневий) модуль роботи ПЗ;
- controller – модуль роботи з конструктором візитівки;
- client – модуль роботи для клієнтської частини веб-додатку;
- admin – модуль для адмін-панелі адміністратора.

Всі ці модулі тісно взаємодіють між собою, розглянемо основні моменти взаємодії та організації їх роботи. Для кращого розуміння взаємодії даних модулів була наведена структурна організація програмного забезпечення у виляді діаграми на рис. 3.3.

Першим та основним модулем виступає модуль Simpoo. Це модуль, у якому описані всі стилі проекту, js, а також знаходяться шаблони html, які використовують всі інші модулі проекту. Саме в даному модулі знаходиться початковий файл index.html. Також в даному модулі знаходяться всі налаштування: підключення бази даних, модулів та необхідних бібліотек, налаштування мов (за архітектурою Django) та інше.

Як вище було зазначено даний додаток виступає у ролі кореневої папки проекту.

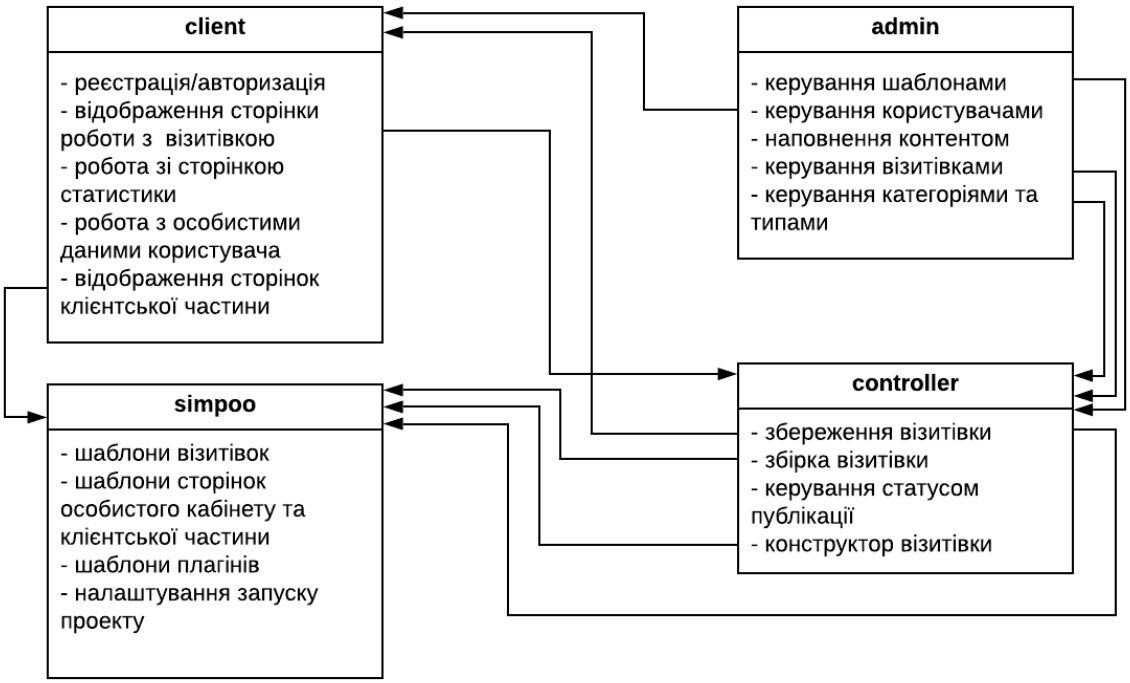


Рис. 3.3 Діаграма зв’язків модулів системи

Наступний модуль client – це модуль для роботи з клієнтською частиною додатку. Даний модуль відповідає за представлення та роботу з такими сторінками як: головна сторінка, сторінка перегляду шаблонів, сторінка перегляду візитівок, а також за особистий кабінет користувача. Таким чином, основні функції даного модуля можна абстрактно поділити на дві умовні частини: функції кабінету користувача та функції клієнтської частини.

Основними функціями клієнтської частини є:

- представлення головної сторінки з відповідним контентом, який зберігається у БД;
- представлення та робота зі сторінкою перегляду шаблонів, тобто: вивідення шаблонів (передача відповідних даних із БД), фільтрація за категоріями та типами, пошук за назвою шаблону;

- представлення та робота зі сторінкою перегляду візитівок, тобто: вивід візитівок (передача відповідних даних із БД), фільтрація за категоріями та пошук за назвою візитівки.

Основними функціями кабінету користувача є:

- реєстрація, авторизація та вихід з системи користувача;
- перегляд та редагування особистих даних (передача відповідних даних з та в БД);
- представлення та робота зі сторінкою вибору шаблону (використовується функція клієнтської частини зі сторінки перегляду шаблонів);
- представлення та робота зі сторінкою перегляду статистики (передача відповідних даних з БД);
- представлення та робота зі сторінкою візитівок користувача, тобто: передача відповідних даних візитівок тільки конкретного користувача, фільтрація за статусом публікації та пошук за назвою візитівки;
- можливість перегляду візитівки іншими користувачами.

Даний модуль взаємодіє з кореневим модулем Simpoo, беручи у ньому шаблони html для представлення сторінок клієнтської частини, а також стилі та js файли. Client служить для представлення основних сторінок, які потім під час рендеру доповнюються за допомогою інших модулів. Даний модуль тісно взаємодіє з модулем controller, адже служить обгорткою для представлення сконструйованої візитівки. Таким чином, на сторінці перегляду візитівки обидва модулі взаємодіють разом, client забезпечує маршрутизацію та передає до модуля controller інформацію про те, що конкретно потрібно представити користувачу, в свою чергу модуль controller збирає повноцінну візитівку та повертає її клієнту. Разом із цим, модуль client перевіряє дані користувача та визначає, в якому саме форматі передати готову візитівку (з можливістю редагування чи ні).

Тобто модуль client, якщо узагальнити, відповідає за роботу

особистого кабінету користувача, клієнтську частину сайту, а також є обгорткою для перегляду візитівок, зібраних конструктором.

Controller – це модуль конструктор візитівок. Його основною задачею є збирання візитівки із основного шаблону, плагінів та контенту. Для цього використовується спеціально розмічений шаблон (особливості розмітки знаходяться у розділі реалізації програмного застосунку), який зберігається у кореневому модулі simpoo. Програмно для кожного шаблону виділяються можливі місця розміщення плагінів. Самі стилі та js файли плагінів також знаходяться у модулі simpoo. Для кожного місця в шаблоні виділяється окремий список плагінів. Користувачу надається можливість розміщення свого контенту та вибору потрібних плагінів, після чого всі ці дані зберігаються в БД для конкретної візитівки. При наступному перегляді controller зчитує дані з бд та формує саму візитівку, при цьому взаємодіючи з модулем simpoo беручи необхідні шаблони та плагіни.

Ще однією функцією є створення самої візитівки та занесення початкових даних про неї в БД. Коли користувач переглядає шаблон та хоче його обрати, на екрані з’являється попап створення візитівки, при цьому в нього передається інформація про уподобаний шаблон і після заповнення всієї необхідної початкової інформації про майбутню візитівку дані зберігаються в БД. І після редагування зв’язуються із цим записом. Також під час створення візитівки даний модуль взаємодіє з модулем client та отримує від нього необхідну інформацію про користувача для того , щоб прив’язати конкретну візитівку саме під конкретного юзера.

При перегляді візитівки користувачем модуль simpoo збирає візитівку та передає її до модуля client, який і відповідає за представлення всієї сторінки перегляду візитівки.

Також зазначимо, що саме даний модуль дає можливість клієнту користуватися конструктором, тобто він надає інтерфейс конструктора на сторінці створення та редагування візитівки.

Також даний модуль реалізує функцію прев'ю (попередній перегляд візитівки) та публікації.

Останнім модулем є admin – модуль, який відповідає за представлення та роботу адмін-панелі адміністратора. Він містить в собі такі функції як:

- Керування шаблонами візитівок. Адміністратор може заносити шаблон до списку доступних користувачам шаблонів, редагувати інформацію про них і видаляти. Адміністратор визначає категорію та тип, до яких буде відноситися даний шаблон. При цьому відбувається взаємодія модуля admin з модулем controller.
- Керування даними про користувачів. Адміністратору доступні функції створення нових користувачів, редагування їх особистих даних, також видалення юзерів. Також доступні функції блокування та розблокування користувачів. При цьому відбувається взаємодія з модулем client.
- Наповнення клієнтської частини сайту контентом, а також його редагування. Таким чином всі подані дані на сторінках модуля client, який у свою чергу бере їх з кореневого модуля simpoo, заносяться у БД саме за допомогою модуля admin.
- Керування візитівками користувачів. Адмін має можливість і управління публікаціями, зміна їх статусу (опубліковано/ неопубліковано), зміна дати та часу початку та кінця публікації візитівки, а також видалення візитівки взагалі.
- Керування категоріями та типами шаблонів. Адміністратор може створювати, редагувати та видаляти типи та категорії шаблонів, які представлені користувачу.

Всі чотири модулі тісно взаємодіють між собою і мають чітко визначені області впливу в проекті. Детальна реалізація та опис їх взаємодії наведені у розділі 4.

3.3. Структура СУБД

Вдала архітектура БД має таку ж важливу роль, як і закладення коректної структури серверної та клієнтської частини. Даний веб-додаток написаний на Python Django, що автоматично закладає не тільки структуру серверної частини, а й СУБД. Даний фреймворк автоматично створює необхідні для його коректної роботи таблиці та зв’язки між ними. Таким чином, створюються таблиці: django_migrations, django_session, auth_user, auth_permission, auth_user_user_permissions та django_admin_log. Ці таблиці відповідають за налаштування системи, сесії та роботу з користувачами.

Взагалі структуру СУБД можно абстрактно поділити на два модулі : модуль взаємодії з користувачем та конструктор візитівки. Основна робота даного веб-застосунку знаходиться саме навколо них і тому для реалізації всього раніше згаданого функціонального наповнення, потрібно забезпечити тісну взаємодію між ними.

Для кращого розуміння структура БД знаходиться на рис. 3.4.

Абстрактний модуль користувача включає в себе такі таблиці як: auth_user, user_app_callback, user_app_pagebyuser, django_admin_log, auth_permission, auth_user_user_permissions. Основною таблицею є auth_user, яку закладає Django фреймворк. Вона відповідає за основні поля з моделі користувача. На дану таблицю посилаються всі інші вище зазначені таблиці. user_app_callback – таблиця для збереження даних плагіна зворотньої форми, яка знаходиться у списку плагінів, і яка взаємодіє з модулем конструктора. Всі відгуки зберігаються в даній таблиці та в подальшому виводяться на сторінці повідомлень зі зворотньої форми у кабінеті користувача. Таблиця user_app_pagebyuser також пов'язана з модулем конструктора і служить зв'язкою візитівки з потрібним користувачем. auth_user_user_permissions відповідає за визначення прав доступу користувача.

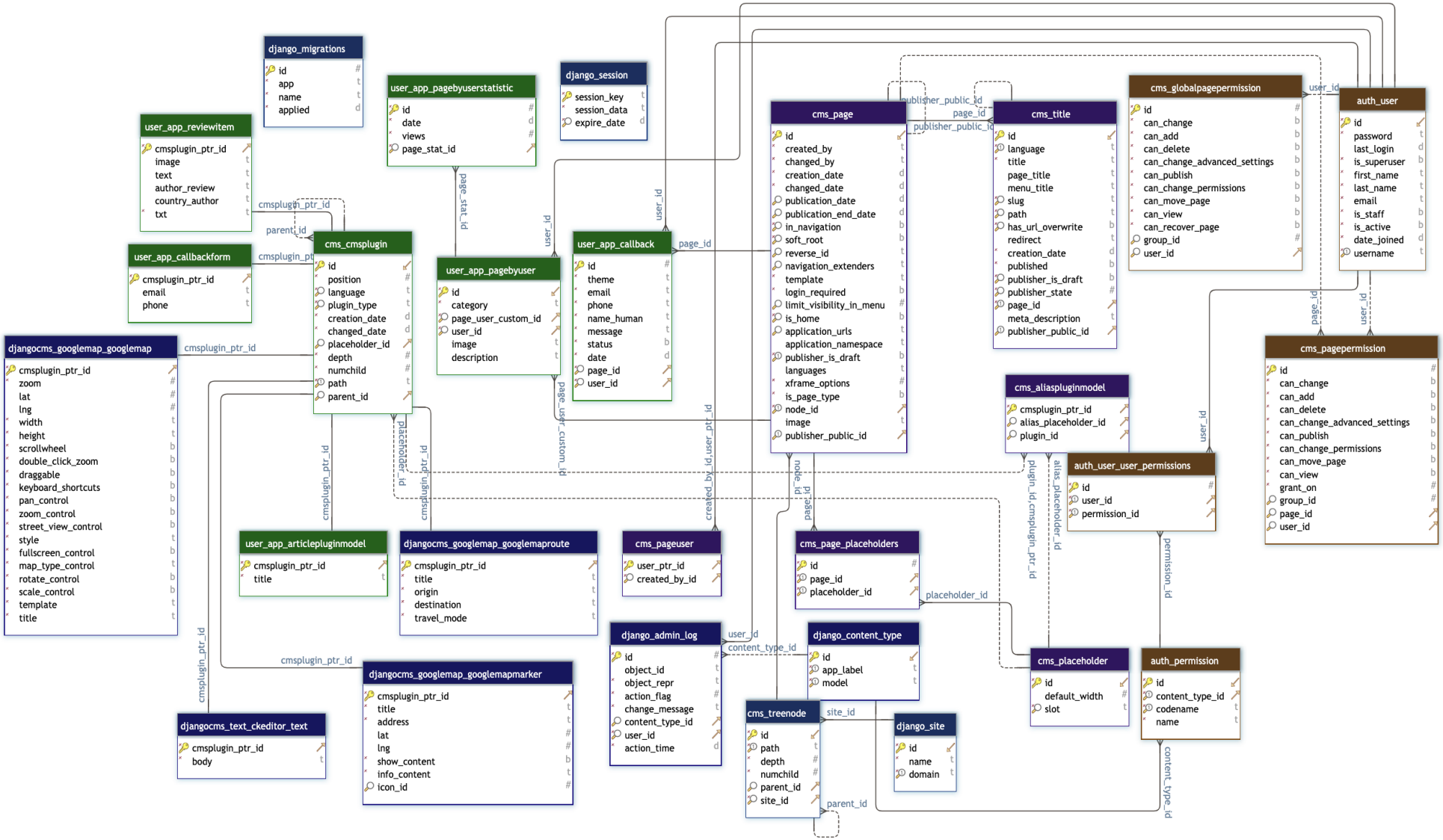


Рис. 3.4. Діаграма структури СУБД

Наступний абстрактний модуль – це конструктор. До нього відносяться таблиці, що відповідають за представлення самої візитівки (cms_page), основну таблицю плагінів (cms_cmsplugins) та таблиці для кожного плагіна (наприклад: user_app_reviewitem, user_app_callbackform і т.д.), а також таблиці роботи зі placeholder (cms_ placeholder). Таблиця cms_page містить у собі інформацію про створену візитівку, обраний для неї шаблон, дату публікації і так далі. Як було зазначено вище основна таблиця cms_page пов’язана з таблицею auth_user через проміжну (об'єднуючу) таблицю user_app_pagebyuser, яка посилається на обидві таблиці. Таблиця cms_page пов’язана з cms_ placeholder також через проміжну таблицю cms_ page_placeholder, яка зсилається на обидві таблиці. Placeholder (Django тег) у проекті виступає у якості списку доступних плагінів для кожного розміченого місця в шаблоні. Дані списки задаються програмно. Відповідно дані плагіна із вказанням плейсхолдера заносяться в таблицю cms_placeholderreference, яка в свою чергу посилається на cms_ placeholder. На дану таблицю посилається таблиця cms_cmsplugins, яка відповідає за збереження даних самих плагінів, на як в свою чергу посилаються окремі таблиці кожного плагіну, наприклад user_app_callbackform (таблиця плагіну контактної форми).

Одним із правил гарного тону при проектуванні БД є приведення структури зберігання даних до нормальних форм. Загальне призначення процесу нормалізації полягає у виключенні деяких типів надмірності, приведенні до інтуїтивної зрозумілості та можливості подальшого розширення, забезпечення цілісності та точності результатів вибірки. Адже низька якість зв'язку в реляційній моделі даних, потенційно призводить до логічно помилкових результатів вибірки або зміни даних. У результаті база даних була приведена до 3-ї нормальної форми, що передбачає атомарність, відсутність залежності неключових полів від ключових, виключає залежність неключових полів від інших неключових полів.

4. ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНИХ ЗАСОБІВ

Веб-додаток для створення та управління онлайн-візитівками складається із серверної та клієнтської частин. Структурна організація серверної частини була описана у попередньому розділі, розглянемо детальніше її реалізацію.

4.1. Особливості реалізації об’єктів системи

Даний веб-додаток написаний на мові програмування Python з використанням фреймворку Django, що автоматично закладає певну структуру проекту з можливістю її розширення. Був використаний шаблон проектування MVC (у Django використовується його видозміна MTV – Model Template View), тому структура проекту напряму залежить від нього. Також даний фреймворк закладає і певну структуру директорій.

4.1.1. *Структура директорій програмного застосунку*

Структура директорій грає важливу роль в розроблені програмного засобу, особливо, якщо мова йде про використання фреймворків. Також початкове закладення правильної структури зменшує час роботи над проектом, особливо, якщо проект є досить великим, або орієнтований на майбутнє розширення.

Simpoo – це основний (загальний) проект, який складається із трьох додатків, які організаційно розбиті по відповідним директоріям: simpoo, cabinet_controller та client. Кожний додаток відповідає за свою частину проекту, при цьому вони тісно взаємодіють між собою. Також на одному рівні з додатками знаходиться директорія media, у яку завантажуються зображення при роботі з веб-застосунком. На одному рівні з додатками знаходяться файли налаштування та запуску проекту. Загальна структура пректу знахдиться на рис. 4.1.

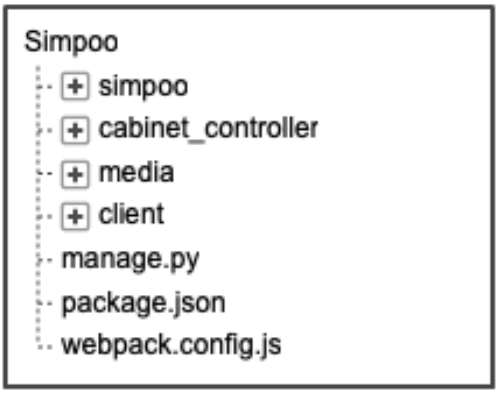


Рис. 4.1. Структурна організація основних директорій проекту

Розберемо кожний додаток окремо. Додаток Simpoo являє собою кореневу папку проекту. Він містить у собі директорію front-end, у якій знаходиться вихідний код стилей та js-файли, які у подальшому збираються за допомогою webpack для мінімізації. Файли js збираються за допомогою babel. Babel – призначений для транспірації коду. Транспірація, це переклад вихідного коду з однієї мови на іншу. Тобто переводиться код написаний на ES6 в код ES5. Після чого, код може використовуватись в будь-якому сучасному браузері. Мінімізовані файли стилів та js збираються у папку static. Все це зроблено для оптимізації роботи клієнтської частини. Також даний додаток містить директорію templates, у якій знаходяться файли html, якими користуються всі додатки системи. Так можемо бачити такі папки як plugins (розмітка плагінів), client (розмітка необхідна додатку client) та temlates_build (самі шаблони для майбутніх візитівок). На одному рівні з основними директоріями даного додатку знаходяться файли settings.py з налаштуваннями додатку та urls.py – налаштування маршрутизації додатку. Для кращого розуміння, вся архітектура даної директорії наведена на рис. 4.2.

Наступний додаток cabinet_conroller, який відповідає за кабінет та можливості авторизованого користувача. Директорія admin – відповідає за адмін-панель адміністратора веб-додатку. Forms – директорія з налаштуваннями форм розміщених на сайті (авторизація, валідація і т.д.).

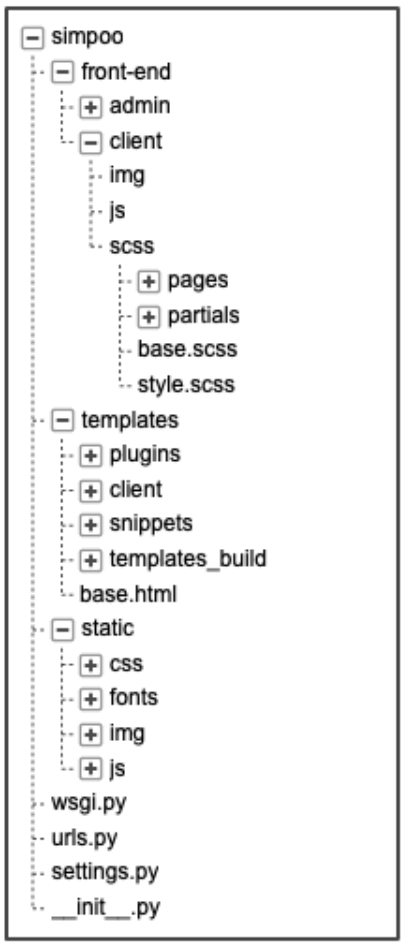


Рис. 4.2. Структурна організація додатку simpoo

Management – складова управління адмін-панелі адміністратора. Models – директорія, яка містить у собі моделі додатку. Templatetags – складова додатку, яка відповідає за розбиття шаблонів майбутніх візитівок для їх використання у конструкторах і можливості додавання плагінів. Wizards – складова додатку, у якій знаходяться налаштування панелі управління на сторінці конструювання візитівки. На одному рівні з директоріями знаходяться файли, що відповідають за роботу з плагінами (plugin_base.py, plugin_pool.py та cab_plugins.py), маршрутизацію (urls.py) та контролер (views.py). Детальніше додаток cabinet_conroller був розглянутий у якості модуля у попередньому розділі. Організація директорій додатку cabinet_conroller розташована на рис. 4.3.

Рис. 4.3. Структурна організація додатку cabinet_conroller

Наступний додаток client, який відповідає за частину веб-застосунку, яка всім користувачам, як авторизованим так і неавторизованим. Він містить у собі директорію міграцій (migrations), меню сторінки конструювання або вибору шаблону (templatestags), файли моделей (models.py), маршрутизації (url.py), контролер (views.py) та додаткові файли-контролери. Організація директорій додатку client подана на рис. 4.4.

Рис. 4.4. Структурна організація додатку client

4.1.2. *Особливості реалізації*

Оскільки Django фреймворк закладає шаблон проектування MTV, на його основі були розроблені відповідні модулі admin, simpoo, controller та client. Кожен з них відповідає за свої окремі задачі, але при цьому між ними відбувається тісна взаємодія. Детальніше розглянемо кожен модуль.

Модуль controller є одним із найосновніших та найобширніших модулів даного веб-застосунку, адже він відповідає саме за конструктор візитівки. Даний модуль складається із моделей, які представлені класами:

- Page – проста ієрархічна модель сторінки;
- PageType – категорії візитівок;
- Title – інформація про шаблон/візитівку;
- PlaceholderField – базовий змінний контент для плагінів;
- PageManager – менеджер з підтримкою обробки візитівки;
- AbstractPagePermission – абстрактний клас прав доступу;
- PagePermission – модель відповідаюча за права доступу користувачів;
- Plugin – базовий клас для моделі плагіна.

Кожен плагін є окремим класом, що успадковується від моделі Plugin. Наприклад: CallBackForm та ReviewItem.

Кожен клас в свою чергу має ряд методів для роботи з ними, розберемо найосновніші з них:

- get_root_page – метод класу PageType, який повертає тип візитівки (односторінкова, візитівка або презентація);
- get_categori – метод класу PageType, який повертає категорію візитівки (односторінкова, візитівка або презентація);
- get_permissions – метод класу AbstractPagePermission для отримання прав доступу користувача;
- pre_save – метод класу PlaceholderField відповідає за збереження даних для попереднього перегляду візитівки;

- save_form_data – метод класу PlaceholderField відповідає за збереження плагіну в даному плейсхолдері;
- get_placeholders – метод класу PlaceholderField, який повертає плейсхолдери шаблону;
- get_title – метод класу Title для отримання даних про шаблон/візитівку;
- set_or_create – метод класу Title для зміни даних шаблону/візитівки або, якщо такої не існує, її створення;
- public – метод класу Title для публікації візитівки;
- unpublish – метод класу Title для зміни статусу візитівки на неопубліковано;
- get_template – метод класу Page для отримання шаблону;
- get_plugins_list – метод класу PlaceholderField для повернення списку доступних для цього плейсхолдера плагінів;
- get_plugin – метод класу PlaceholderField для отримання плагіна обраного користувачем для даного плейсхолдера;
- get_plugin_info – метод класу Plugin для отримання інформації про плагін;
- get_permission – метод класу PagePermission для отримання прав доступу до візитівки;
- get_render_template – метод класу Plugin для збирання плагінів з контентом;
- render_page – метод класу Page для збирання візитівки;
- create_page – метод класу Page для створення екземпляру даного класу
- create_title – метод класу Title для створення екземпляру даного класу;
- add_plugin – метод класу Plugin для додавання плагіну;

- can_change_page – метод класу PagePermission для перевірки прав користувача по редагуванню сторінки.

Наступний модуль Client відповідає за роботу з даними користувача, прив’язку візитівки до нього, права доступу та статистику. Даний модуль складається із моделей, що представлені класами:

- User – клас інформації про користувача та роботи з ним;
- PageByUser – клас, який відповідає за прив’язку візитівки до користувача;
- PageByUserStatistic – клас статистики перегляду візитівки;
- UserPermission – клас прав доступу користувача.

Класи даного модуля в свою чергу реалізують методи, основними з яких є:

- user_has_permissions – метод класу UserPermission для повернення прав доступу користувача;
- create_user – метод класу User для створення нового користувача;
- user_has_perm – метод класу UserPermission для повернення прав користувача;
- get_count_views – метод класу PageByUserStatistic для повернення статистики переглядів візитівки;
- set_page_to_user – метод класу PageByUser для прив’язки візитівки до користувача.

Наступний модуль Admin відповідає за роботу адміністратора та його панель управління. Даний модуль складається із моделей, які представлені класами:

- PageAdmin – клас керування сторінки адміністратора;
- AdminPermission – клас можливостей адміністратора.

Класи даного модуля реалізують методи, основними з яких є:

- change_permission – метод класу AdminPermission для зміни прав користувачів, який взаємодіє з класом UserPermission;

- get_users – метод класу PageAdmin для перегляду списку користувачів, який взаємодіє з класом User;
- change_template – метод класу AdminPermission для редагування шаблонів, який взаємодіє з класом Page;
- publish_page – метод класу PageAdmin для публікації візитівок, взаємодіє з класом Title;
- unpublish – метод класу PageAdmin для відміни публікації візитівок, взаємодіє з класом Title;
- get_pages – метод класу PageAdmin для перегляду списку візитівок, взаємодіє з класом Title.

Кожному класу, із наведених у даному підпункті, доступні методи create, update, delete, get та set. Методи get та set доступні для кожного атрибуту класу. Всі моделі та їх атрибути відповідають наведеним на схемі бази даних на рис. 3.4.

4.2. Опис інтерфейсу користувача

Даний веб-застосунок орієнтований на звичайного користувача далекого від IT індустрії, тому потрібно, щоб інтерфейс був у мінімалістичному стилі зрозумілий кожному. Для розроблення веб-інтерфейсу використовувались технології HTML, CSS та JS.

Для виконання дизайну було використано сітку кольорів #BC2F3C (відтінок червоного), #777777 (відтінок сірого) та #FFFFFF (фоновий білий), всі інші кольори розміщені на сайті є отриманими внаслідок зміни прозорості основних кольорів. Як бачимо, було використано всього 3 кольори – це вважається хорошою практикою при виконані дизайну. Також можна помітити, що не було використано чорного (#000000), адже він має негативний вплив на сприйняття користувача, тому даний колір був замінений темно сірим.

Кожна сторінка застосунку містить хедер, у якому розміщені логотип, меню та кнопка переходу до особистого кабінету. Меню складається із пунктів Advertisement (перегляд всіх опублікованих

візитівок), Templates (перегляд шаблонів для створення візитівок) та Contact us (сторінка зв’язку із службою підтримки). Також у хедері представлена коротка інформація про статус авторизованості користувача в системі, тобто, коли користувач неавторизований, у хедері розміщується кнопка Log in (авторизація), коли авторизований – Log out, логін користувача та кнопка переходу до особистого кабінету. Приклад представлення хедера наведений на рис. 4.5.

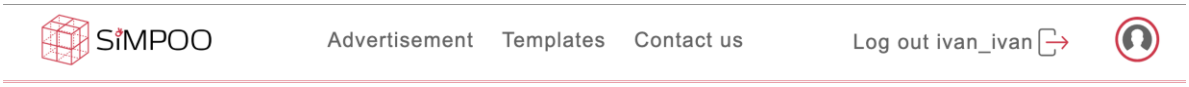


Рис. 4.5. Хедер авторизованого користувача

Головна сторінка виступає в якості інформаційної, тому на ній знаходиться покрокова інструкція взаємодії користувача із системою, приклади деяких можливих для використання шаблонів, список категорій шаблонів та презентаційний блок з інформацією про призначення даного веб-додатку.

Для реєстрації та авторизації були використані спливаючі вікна. Вікно авторизації має поля логіна та паролю, а реєстрації – логін, емейл, пароль та підтвердження паролю. Також для обох вікон були додані чербокси показати пароль та кнопки переходу між даними вікнами. Приклад роботи даного вікна у режимі реєстрації нового користувача знаходиться на рис. 4.6.

Сторінка Advertisement відповідає за перегляд візитівок, які знаходяться в опублікованому статусі. На даній сторінці у вигляді оглядового списку розташовані візитівки з короткою інформацією про них: назва, опис, категорія та логін автора.

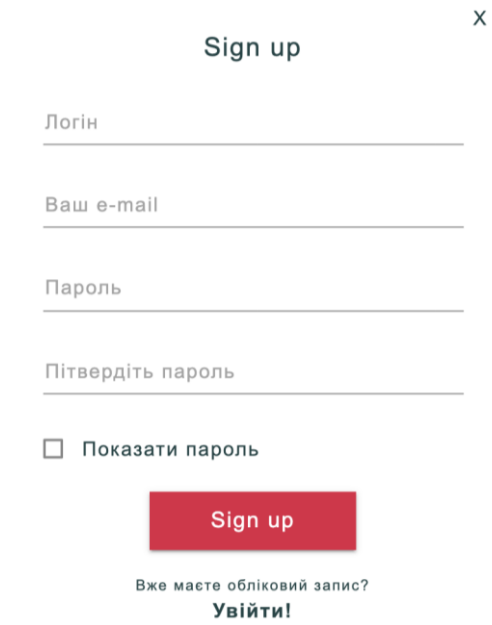


Рис. 4.6. Вспиваюче вікно реєстрації

Також на даній сторінці доступна фільтрація по категоріям, що виводяться з лівого боку, та пошук за назвою візитівки. Приклад роботи даної сторінки знаходиться на рис. 4.7.

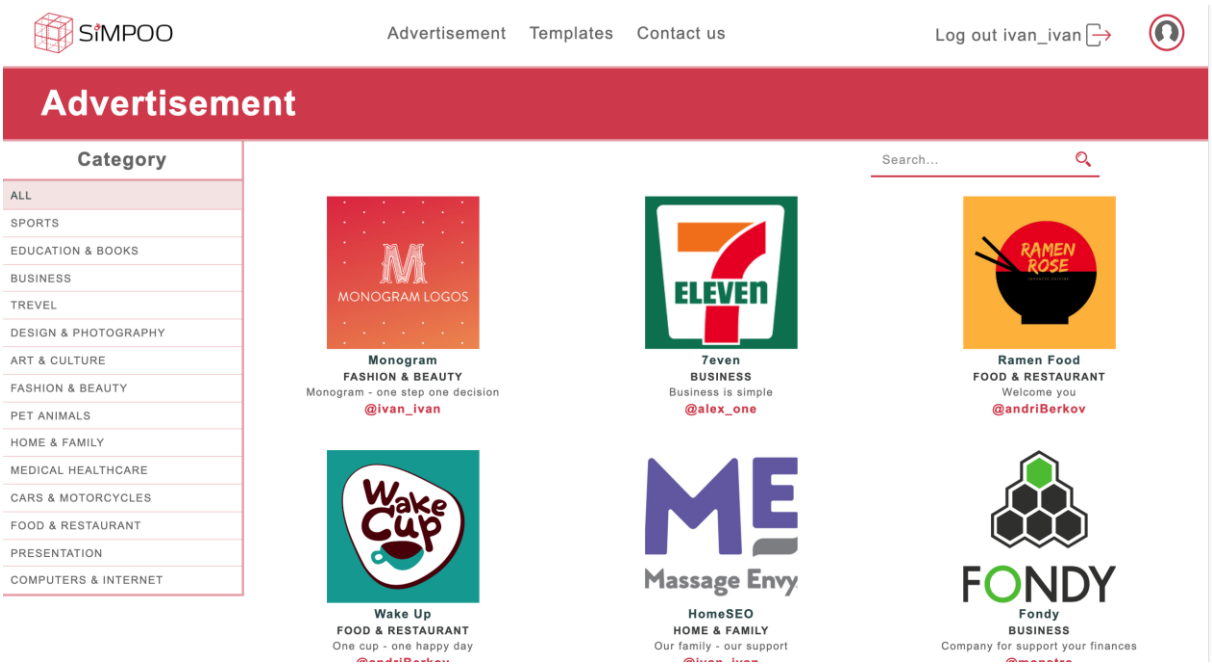


Рис. 4.7. Сторінка перегляду візитівок

Наступна сторінка Templates відповідає за перегляд шаблонів для майбутніх візитівок. Сторінка зовнішнім виглядом дуже схожа на

попередню, так само у вигляді оглядового списку виводяться шаблони з короткою інформацією: назва, категорія, коротка інформація та тип шаблону. На даній сторінці також доступна фільтрація за категоріями та типами, а також пошук візитівки за назвою. Приклад роботи сторінки перегляду шаблонів візитівок знаходиться на рис. 4.8.

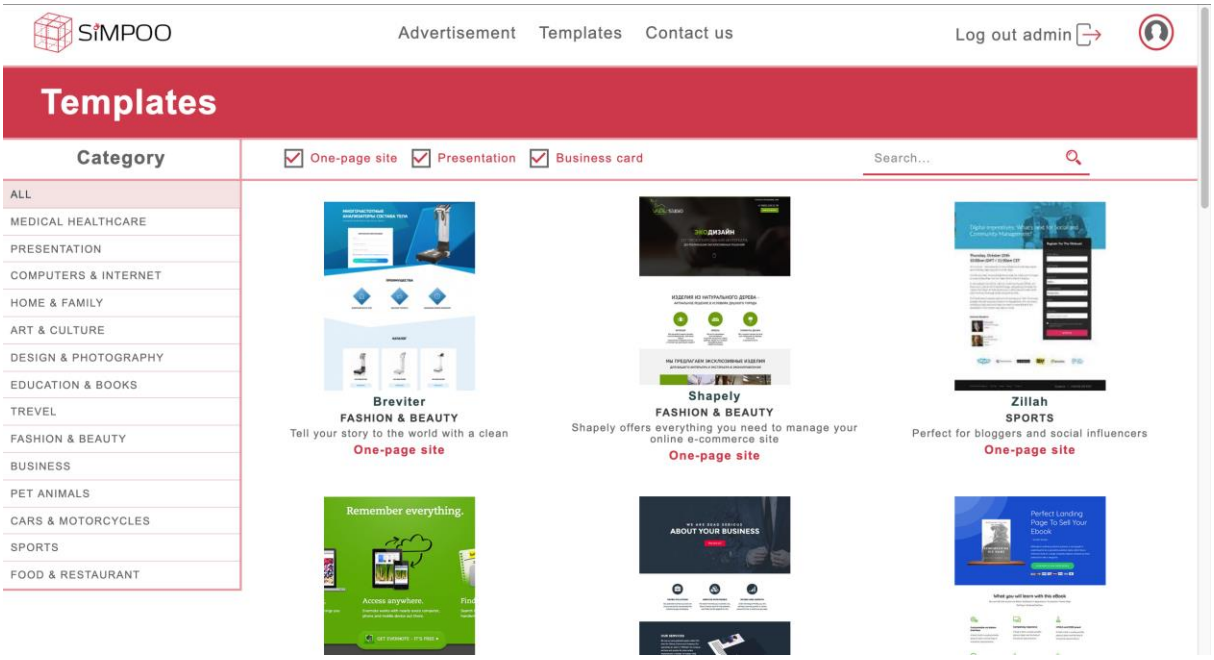


Рис. 4.8. Сторінка перегляду шаблонів візитівок

Перейдемо до особистого кабінету, який містить у собі чотири основні сторінки: Profile, My Templates, Create New Templates, Statistics та Contact Form. Profile – сторінка з особистими даними користувача, на якій виводяться: фото профіля користувача, повне ім’я, логін, емейл, коротка інформація про юзера та пароль. Всі ці поля можна редагувати, тому було додано ще два поля новий пароль та підтвердження нового паролю. Всі поля знаходяться в активному стані, тому при бажані змінити будь-яку інформацію, користувач просто вводить нові дані та натискає на кнопку збереження. Приклад роботи сторінки профілю користувача знаходиться на рис. 4.9.

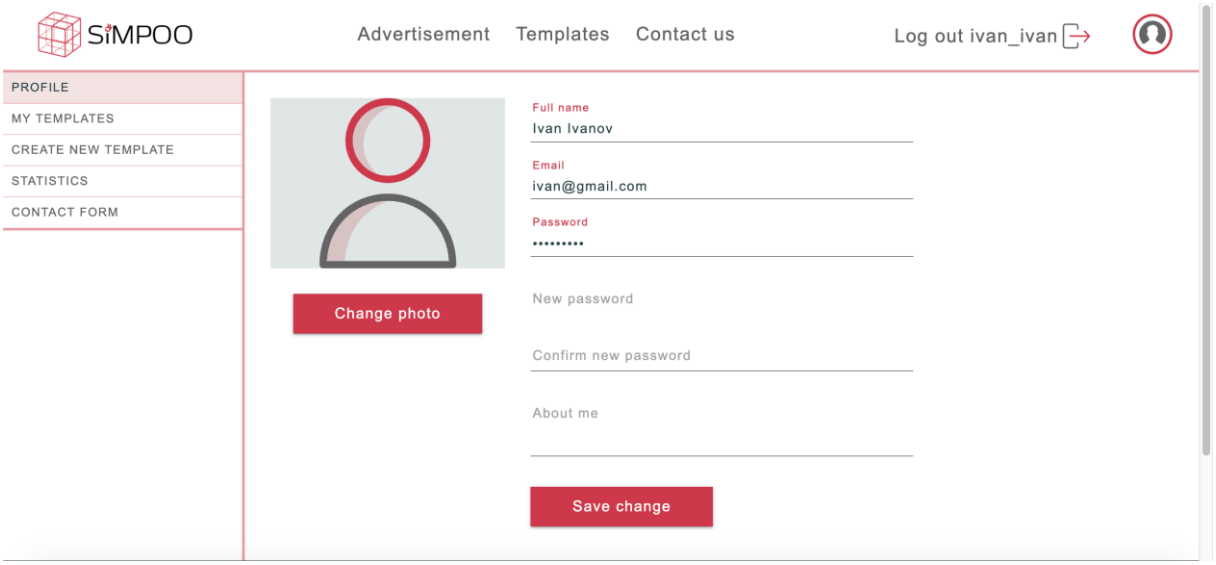


Рис. 4.9. Сторінка профілю користувача

Наступна сторінка My Templates має майже схожий вигляд як і сторінка Advertisement, відмінність полягає лише в тому, що на даній сторінці виводяться тільки візитівки конкретного користувача у будь-якому статусі публікації. Для зручного користування даною сторінкою була додана фільтрація за статусом опублікованності (publish, unpublish та all). Також на даній сторінці виводиться кількість переглядів.

Сторінка Create New Templates має також подібний вигляд до сторінки Templates, різниця полягає лише у зовнішньому вигляді та розташуванні фільтра за категоріями.

Сторінка Statistics подає статистику переглядів кожної візитівки користувача у вигляді графіка, який був створений за допомогою бібліотеки Highcharts на базі JQuery. За допомогою можливостей даної бібліотеки для графіків реалізована можливість скачування у форматах JPEG, PDF, PNG, SVG, CSV, XLS, друк та перегляд у повноекранному режимі. Користувач може переглядати статистику тільки свої візитівок, для забезпечення конфіденційності інформації. Приклад роботи сторінки перегляду статистики візитівок користувача знаходиться на рис. 4.10.

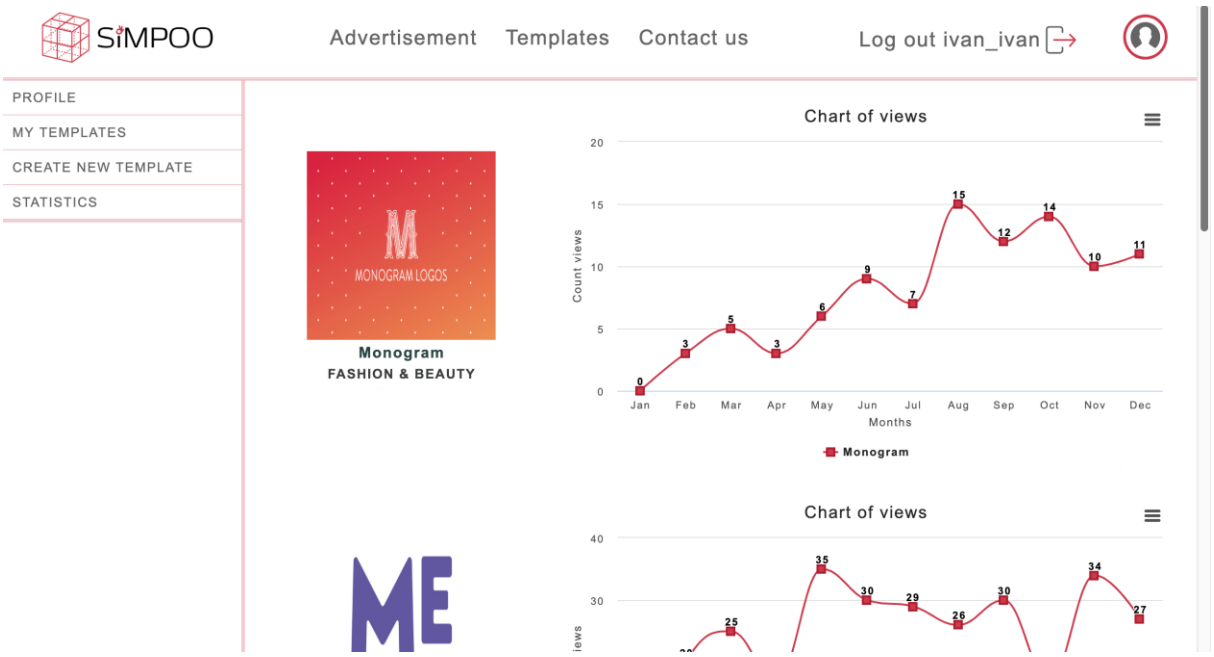


Рис. 4.10. Сторінка перегляду статистики візитівок користувача

Наступна сторінка Contact Form, на якій представляються повідомлення відправлені із форм зворотнього зв'язку розташованих у візитівках. Всі повідомлення дублюються на пошту автора візитівки, яку він задав при налаштуванні плагіну контактної форми і все спілкування відбувається за допомогою пошти або по телефону, проте на даній сторінці можна переглянути всі повідомлення із всіх візитівок користувача та відмітити як переглянуте. При натискані на поле таблиці з'являється спливаюче вікно з повним текстом листа та всією іншою інформацією, такою як email, телефон, ім'я відправника, темою та назвою візитівки, з якої повідомлення було відправлене. Також в даному вікні доступна функція відмітки листа як прочитаного. Приклад роботи сторінки зворотнього зв'язку знаходиться на рис. 4.11.

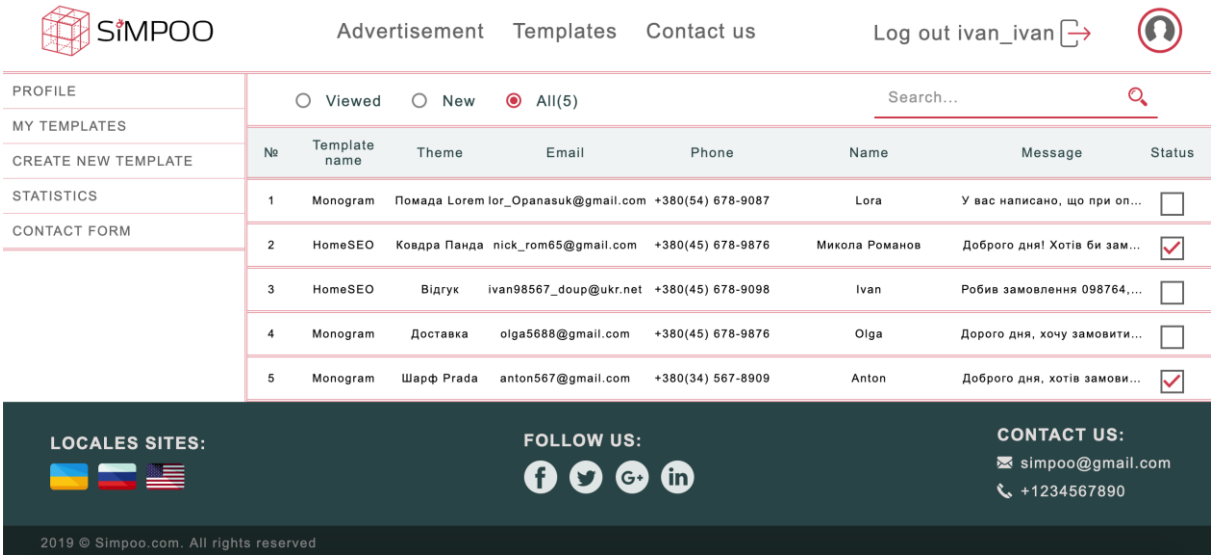


Рис. 4.11. Сторінка зворотнього зв’язку

Одним із найважливіших компонентів даного веб-застосунку є конструктор візитівки. Для даного модуля потрібно забезпечити не тільки чітку архітектуру та гнучкість при роботі з ним, а також і інтуїтивно зрозумілий інтерфейс, що надасть можливість працювати з даним веб-додатком будь-якому користувачу. Принцип роботи зі сторінкою створення та редагування візитівки полягає в наступному: спочатку користувач переходить на сторінку перегляду шаблону, після чого натискає на кнопку вибору шаблону, далі з’являється спливаюче вікно для вводу початкової інформації про майбутню візитівку, тобто завантаження зображення (логотип), яке буде представлене на сторінці перегляду візитівок, назву, короткий опис та обирає категорію зі списку запропонованих. Вибір категорії не залежить від категорії, до якої початково належав шаблон, для зручності використання додатку користувачем. Після заповнення форми спливаючого вікна візитівка потрапляє до списку у особистому кабінеті користувача у статусі неопубліковано. Далі, при переході на дану візитівку, користувач обирає запропоновані плагіни для кожної ділянки шаблону та заповнює його конентом у спливаючих вікнах. Для зручності користування даним конструктором була створена можливість обновлення сторінки або перехід

на іншу без втрати даних змін. При цьому зміни не показуються на опублікованій візитівці, доки користувач не натисне опублікувати зміни. Таким чином, користувач може не хвилюватися про втрату доданих змін при збоях у мережі або комп'ютері. Також для зручності редагування елементів шаблону була додана можливість подвійного кліку на місце у шаблоні, що потребує редагування, при цьому з'являється спливаюче вікно з налаштуваннями плагіну розташованого у цьому місці. Також дана сторінка надає можливість редагування попередньо введених даних при створені візитівки: назву, опис та зображення. Також доступна зміна статусу (опубліковано/неопубліковано). Приклад роботи сторінки редагування візитівки знаходиться на рис. 4.12.

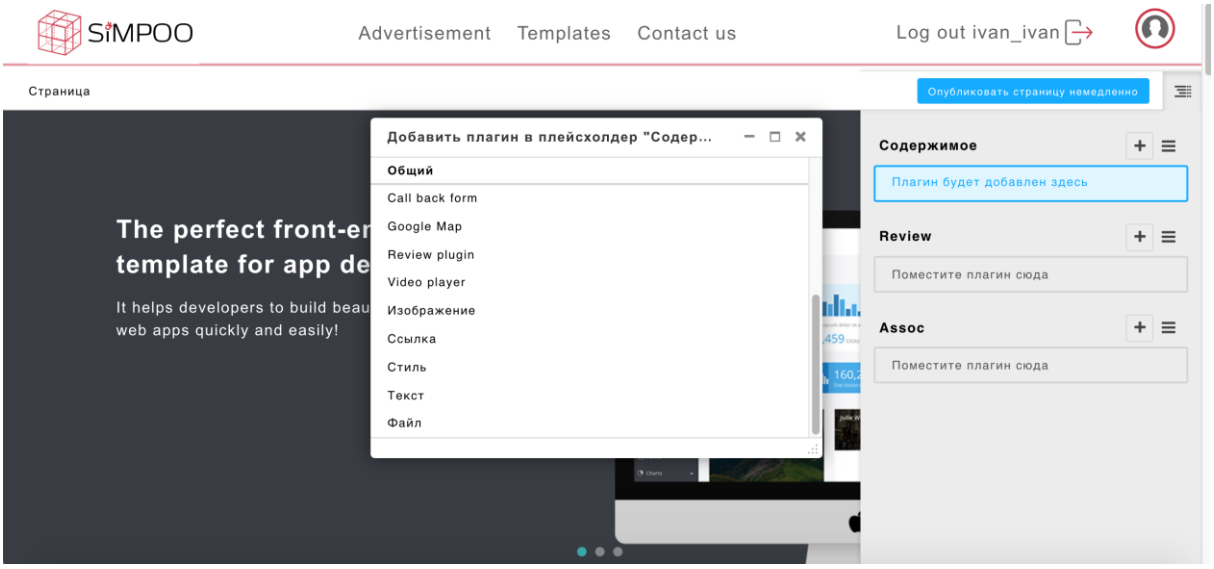


Рис. 4.12. Сторінка редагування візитівки

Отже, під час розроблення інтерфейсу веб-додатку для створення та управління онлайн-візитівками основна мета була забезпечити інтуїтивну зрозумілість та простоту використання. Для цього клієнтська частина не була перевантажена безліччю кольорами, великою кількістю анімованих елементів та важким для розуміння звичайному користувачу контентом. Інтерфейс був виконаний лаконічним, але при цьому з наданням всіх можливостей системи.

4.3. Тестування веб-додатку

Для забезпечення якості веб-додатку необхідно було провести тестування. Для перевірки правильності роботи застосунку було обране димове тестування, яке полягало у тест-кейсах наведених у табл. 4.1.

Таблиця 4.1

Тест-кейси димового тестування

№ п/п	ID вимоги	Назва тест-кейсу	Опис	Очікувані результати
1	F1	Авторизація користувача в системі	1. Натиснути кнопку авторизації. 2. Ввести логін та пароль. 3. Натиснути кнопку Log in.	1. Система подає спливаюче вікно авторизації. 2. Пароль представлений у вигляді крапок. 3. Користувач знаходиться на тій же сторінці, в хедері є логін та кнопка переходу до кабінету.
2	F20, F22	Вибір шаблону	1. Перейти на сторінку Templates. 2. Обрати шаблон. 3. Натиснути на Choose this templates. 4. Ввести дані у спливаюче вікно та підтвердити. 5. Перейти на сторінку даної візитівки.	1. Система представляє у вигляді оглядового списку шаблони згідно дизайну. 2. Перехід на сторінку перегляду даного шаблону. 3. Поява спливаючого вікна створення візитівки. 4. Перехід на сторінку візитівок в особистому кабінеті користувача. Дана візитівка у статусі unpublish. 5. Перехід на сторінку візитівки, на якій представлено вірний шаблон та доступне редагування.

Продовження табл. 4.1

3	F 21	Доступність редагування візитівки	1. Перейти на візитівку іншого користувача. 2. Перейти на сторінку своєї візитівки.	1. Панель редагування візитівки не представлена на екрані. 2. Представлення панелі редагування візитівки.
4	F15, F16, F18	Коректна робота форм зворотнього зв’язку	1. Перейти на сторінку редагування візитівки. 2. Додати плагін форми зворотнього зв’язку. 3. Налаштувати плагін (ввести емейл). 4. Зберегти зміни. 5. Заповнити форму зворотнього зв’язку. 6. Перейти на сторінку Contact Form. 7. Натиснути на повідомлення. 8. Відмітити повідомлення як прочитане.	1. Коректне представлення сторінки. 2. Плагін додано, відкрилося спливаюче вікно налаштування. 3. Дані введені. 4. Форма представлена на візитівці. 5. Валідація даних форми коректно працює. 6. Коректне представлення сторінки Contact Form, подане залишене повідомлення у статусі непрочитане. 7. З’являється всплваюче вікно з поданням коректних попередньо введених даних. 8. Повідомлення переходить у статус переглянутого.
5	F9, F10, F11	Представлення візитівок в особистому кабінеті	1. Перейти на сторінку My Templates. 2. Відфільтрувати для перегляду тільки неопубліковані.	1. Представлення тільки візитівок даного користувача. 2. Представлення неопублікованих візитівок тільки даного користувача.

Продовження табл. 4.1

6	F27, F28, F29, F30, F31, F32	Редагування візитівки	1. Перейти на сторінку редагування візитівки. 2. Натиснути кнопку редагування. 3. Додати плагін. 4. Налаштувати плагін. 5. Зберегти зміни візитівки.	1. Правильне представлення шаблону та наповнення візитівки, доступна кнопка переходу в режим редагування. 2. Подання панелі редагування. 3. Представлення допустимих плагінів для цього місця. Подання вікна налаштування плагіну. 4. Коректне представлення плагіну з урахуванням всіх налаштувань. 5. Зміни представлені на сторінці перегляду візитівки.

В табл. 4.1 наведені тільки основні тест-кейси, які є найнеобхіднішими для повноцінної роботи веб-додатку для створення та управління онлайн візитівками, в цьому і полягає принцип димового тестування. В результаті перевірки та оцінки якості за допомогою даних тест-кейсів відхилень від вимог не виявлено.

4.4. Рекомендації щодо подальшого вдосконалення

Наступним кроком для вдосконалення та розширення даної системи є додання нового типу візитівки – html шаблон, який буде полягати у тому, що користувачу надається можливість самостійно додавати html, css та js код. Під час розроблення даного типу візитівки особливу увагу слід звернути на безпеку сайту, адже можливість додавання файлів даного типу підвищує ризик здійснення веб-атак, таких як наприклад ін’єкції різних типів.

Також цікавим розширенням може бути додання онлайн-чатів, що забезпечить спілкування між клієнтом на користувачем, що надає послуги.

Ще одним варіантом для забезпечення монетизації даного веб-застосунку є додання платіжних систем та доступності безкоштовного режиму тільки на обмежений період.

Досить цікавим є також варіант щодо додання мов (локацій додатку) та надання можливості користувачам обирати зручну для себе мову користування веб-додатком, а також перекладу самих візитівок.

ВИСНОВКИ

Метою даного дипломного проекту є розроблення веб-додатку для створення та управління онлайн візитівками.

Під час виконання проекту було проведено аналіз доступних програмних рішень, який показав, що аналогу розроблюваному веб-застосунку немає, що вказує на доцільність створення системи даного типу. На базі результатів аналізу існуючих рішень було сформовано основні вимоги до технологій розроблення, розглянуто найбільш відповідні, проведено обгрунтування вибору фреймворків для серверної та клієнтської частин, а також СУБД, та обрано обгрунтовані рішення у вигляді Python Django, JavaScript та PostgreSQL. Також було проведено збір та аналіз вимог до програмного застосунку. Після чого було розроблено структуру веб-додатку та бази даних.

В останньому розділі даної пояснювальної записки розглянуто основні сутності та методи взаємодії між ними, також наведено приклади із реалізації інтерфейсу користувача, на який було зроблено великий акцент при розроблені. У результаті, даний веб-додаток було створено у повному обсязі та на основі попередньо сформованих вимог було проведено димове тестування, у результаті якого відхилень не виявлено. Також було надано рекомендації щодо подальшого вдосконалення та розширення розроблюваного веб-застосунку.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

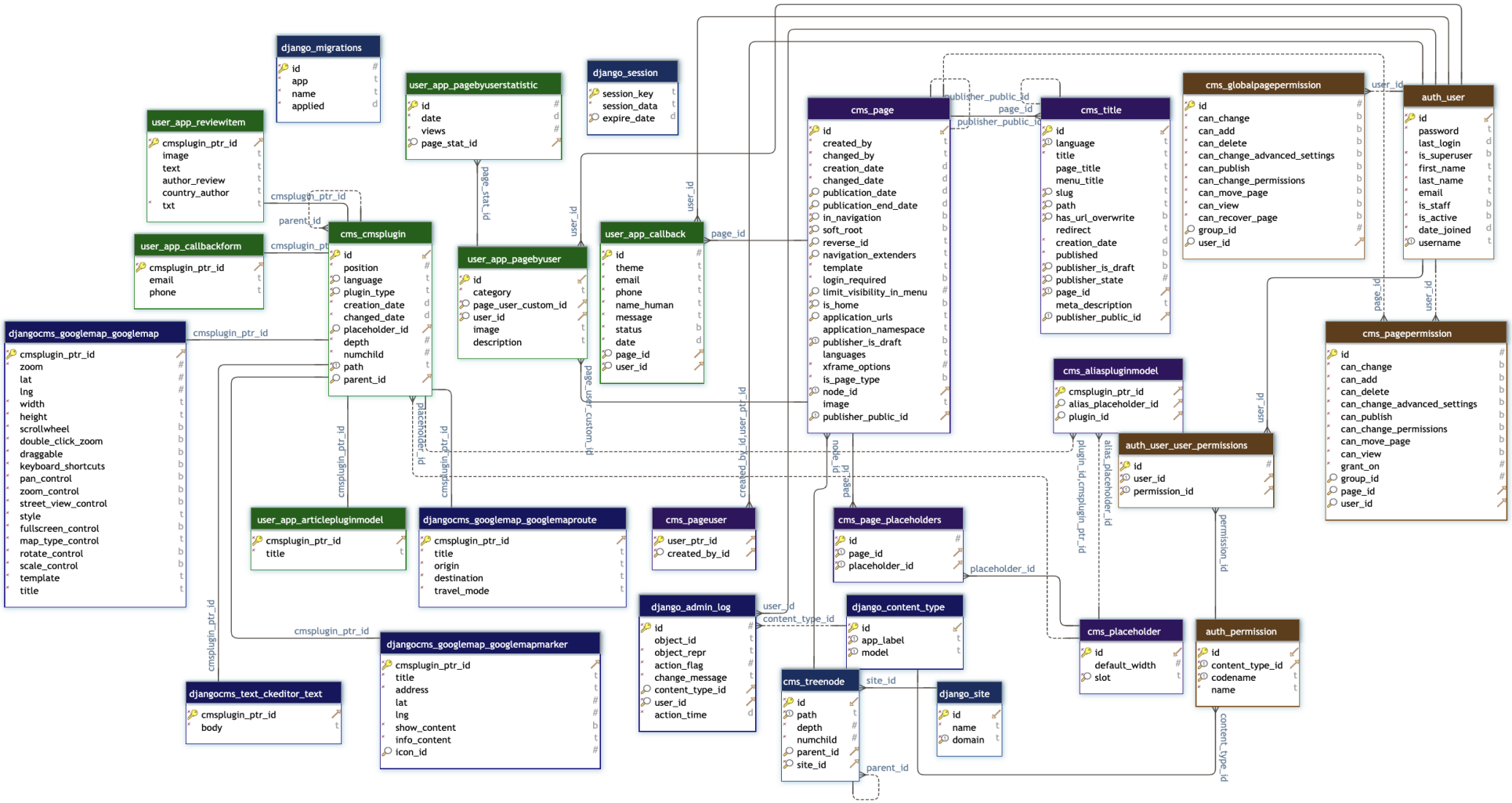
1. Что такое WordPress: [Електронний ресурс]. – Режим доступу до ресурсу: https://hostenko.com/wpcafe/tutorials/chto-takoe-wordpress/. – (14.12.2018).
2. Joomla: [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Joomla!. – (14.12.2018).
3. Wix.com: [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Wix.com. – (14.12.2018).
4. Что такое Facebook: [Електронний ресурс]. – Режим доступу до ресурсу: https://semantica.in/blog/chto-takoe-facebook.html. – (14.12.2018).
5. Инстаграм – что это такое и как им пользоваться: [Електронний ресурс]. – Режим доступу до ресурсу: https://ktonanovenkogo.ru/web-obzory/instagram-chto-eto-takoe-kak-polzovatsya-registracii-v-instagram-s-kompyutera-onlajn.html. – (14.12.2018).
6. Django documentation: [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.djangoproject.com/en/2.2/. – (03.02.2019).
7. Advantages and Disadvantages of Python Programming Language: [Електронний ресурс]. – Режим доступу до ресурсу: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121. – (03.02.2019).
8. Django vs PHP: [Електронний ресурс]. – Режим доступу до ресурсу: https://www.educba.com/django-vs-php/. – (03.02.2019).
9. Node.js vs. Django: Is JavaScript Better Than Python?: [Електронний ресурс]. – Режим доступу до ресурсу: https://dzone.com/articles/nodejs-vs-djangois-javascript-better-than-python. – (03.02.2019).
10. Flask vs Django: [Електронний ресурс]. – Режим доступу до ресурсу: http://www.mindfiresolutions.com/blog/2018/05/flask-vs-django/. – (03.02.2019).

11. Системы управления базами данных: [Електронний ресурс]. – Режим доступу до ресурсу: https://lecturesdb.readthedocs.io/databases/dbms.html. – (05.03.2019).
12. PostgreSQL: [Електронний ресурс]. – Режим доступу до ресурсу: http://www.sai.msu.su/~megera/postgres/talks/what_is_postgresql.html. – (05.03.2019).
13. PostgreSQL Vs. MySQL: [Електронний ресурс]. – Режим доступу до ресурсу: https://blog.panoply.io/postgresql-vs.-mysql. – (05.03.2019).
14. PostgreSQL Vs. MongoDB: [Електронний ресурс]. – Режим доступу до ресурсу: https://blog.panoply.io/postgresql-vs-mongodb. – (05.03.2019).
15. TOP 10 Advantages Of JavaScript In 2019: [Електронний ресурс]. – Режим доступу до ресурсу: https://www.webcreate.me/top-advantages-javascript/. – (20.03.2018).
16. Зачем нужна библиотека jquery?: [Електронний ресурс]. – Режим доступу до ресурсу: https://www.jsexpert.net/zachem-nuzhna-biblioteka-jquery/. – (20.03.2019).
17. Что такое базы данных NoSQL?: [Електронний ресурс]. – Режим доступу до ресурсу: https://aws.amazon.com/ru/nosql/. – (22.04.2019).

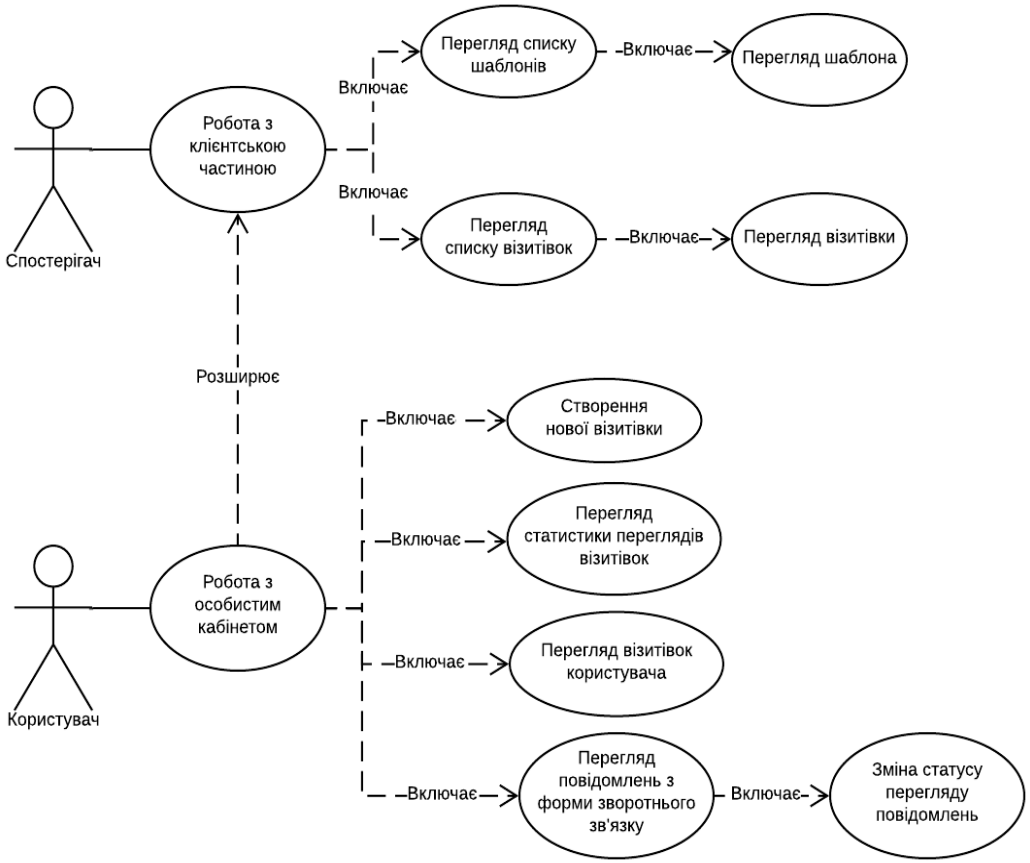
ДОДАТКИ

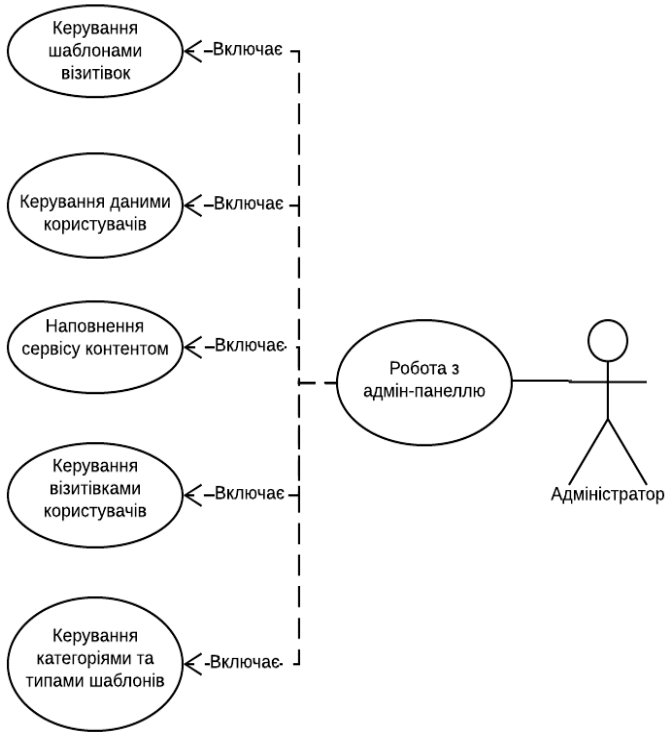
Додаток 1

Копії графічних матеріалів



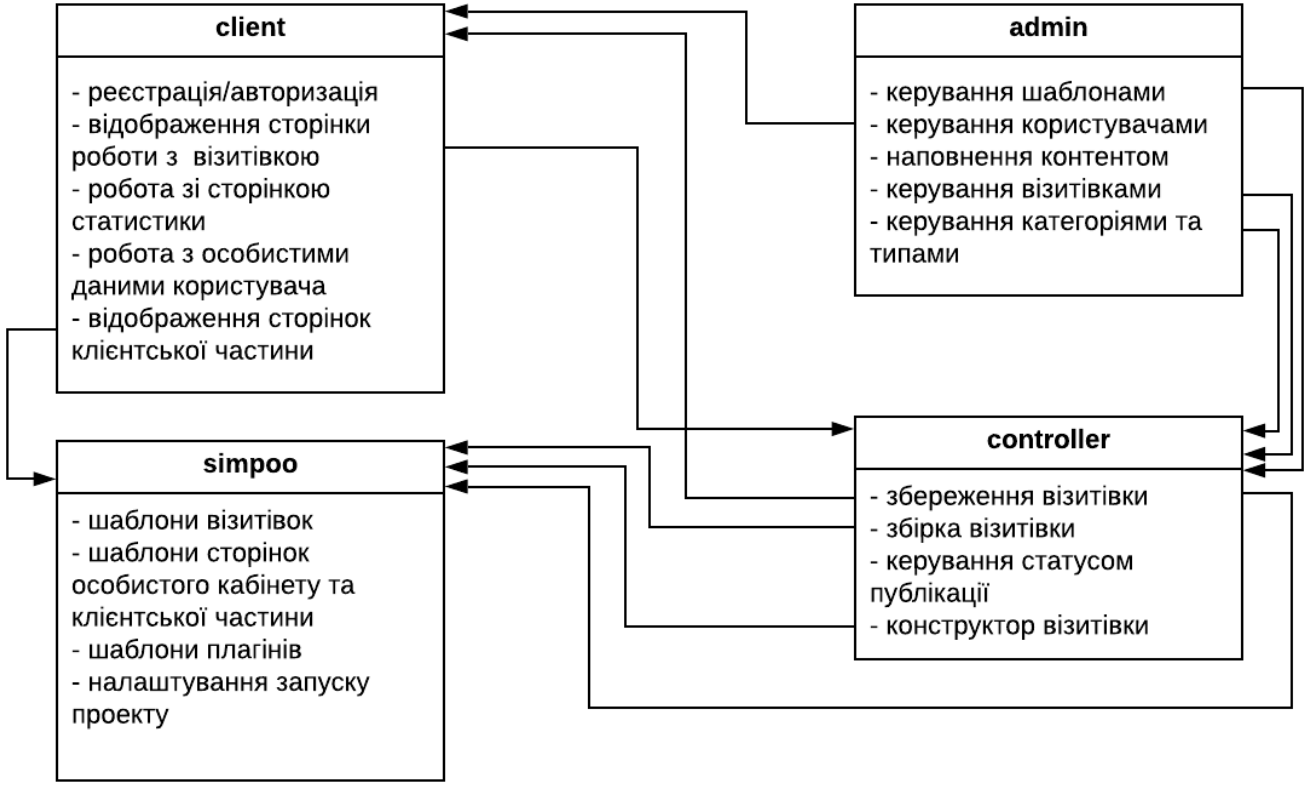
ДП.045440-06-99
Веб-додаток для створення та управління онлайн-візитівками.
Структура бази даних. ERD діаграма



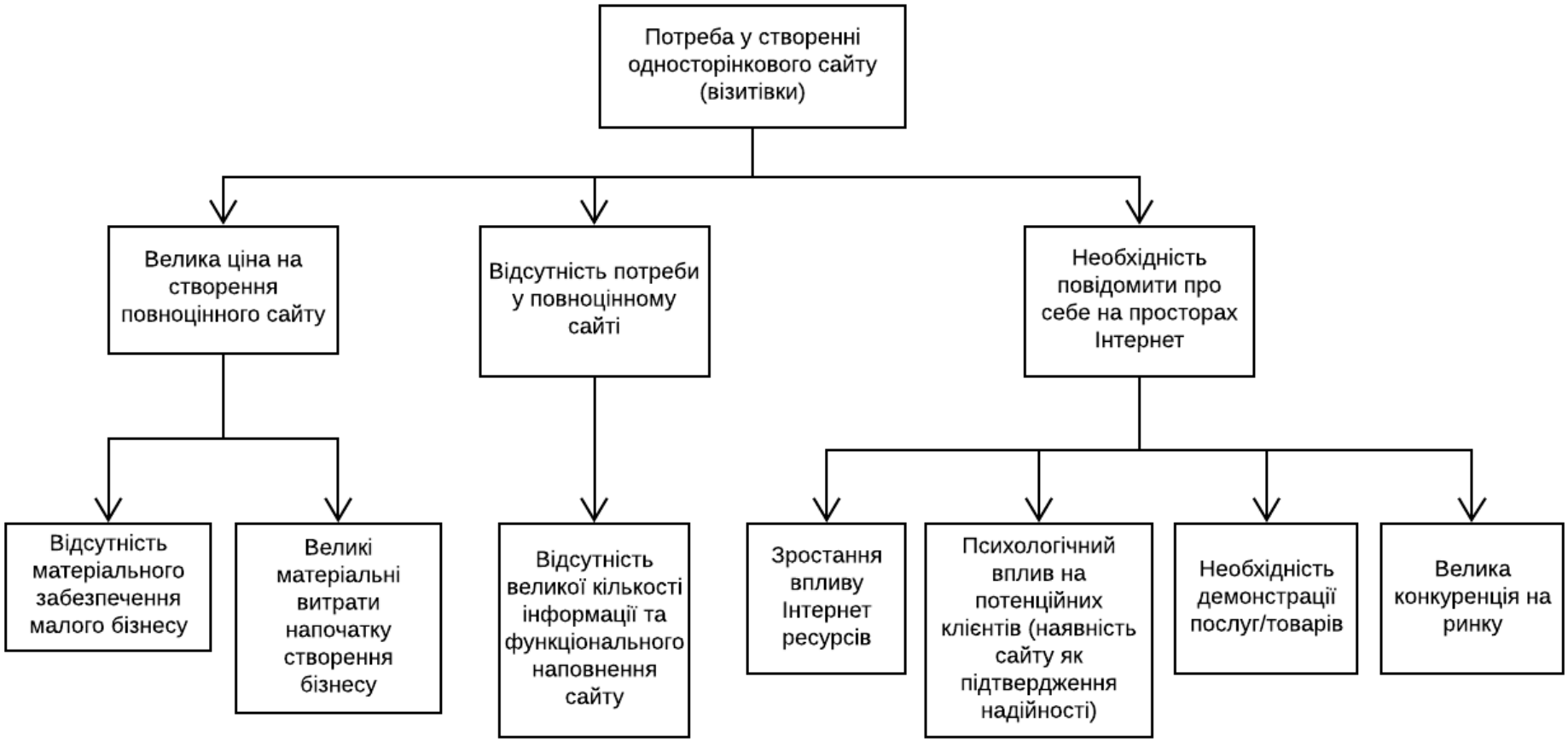


ДП.045440-07-99
Веб-додаток для створення та управління онлайн-візитівками. Функціональні можливості користувача. Діаграма прецедентів

Діаграма зв’язків модулів системи



Пивоварчук О.В., група КП-51



Дерево проблем
Пивоварчук О.В., група КП-51

Додаток 2

Копія презентації

Веб-додаток для створення та управління онлайн-візитівками

Пивоварчук Олександра, КП-51
Науковий керівник: к.т.н., доцент каф. ПЗКС ФПМ,
Т.М.Заболотня

Ідея

Створити веб-додаток, на якому можна легко розміщувати візитівки, а не створювати повноцінний сайт.

Актуальність

Веб-додаток для створення та управління онлайн-візитівками – це нове рішення, яке дає можливість для підтримки малого бізнесу за допомогою розміщення інформації про себе на просторах Інтернет.

Проблеми, що вирішує проект

- Невиправдана необхідність створення повноцінного сайту
- Велика кількість часу для створення сайту
- Висока ціна розробки сайту
- Велика кількість часу для розкрутки

Дерево проблем

Існуючі рішення

CMS

Соціальні мережі

Постановка задачі

Мета: забезпечити можливість розміщення презентаційно-рекламної інформації у мережі Інтернет.

Завдання:

1. Проаналізувати існуючі програмні рішення
2. Розробити програмний застосунок відповідно до вимог
3. Протестувати роботу веб-додатку

Вимоги до продукту

- Забезпечення зручності та швидкого розуміння роботи з програмою
- Створення та редагування презентаційно-рекламних сторінок
- Наявність окремого доступу користувачу для створення та управління онлайн-візитівкою
- Наявність трьох основних видів шаблонів (типів) онлайн-візитівок:
  1. Візитівка
  2. Односторінковий сайт
  3. Презентація

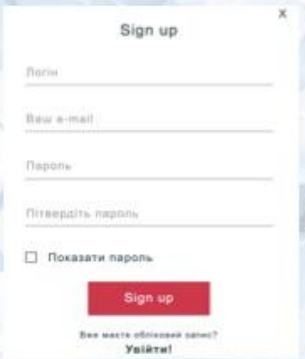
Вибір технологій

Діаграма прецедентів

Діаграма зв’язків модулів системи

Діаграма структури СУБД

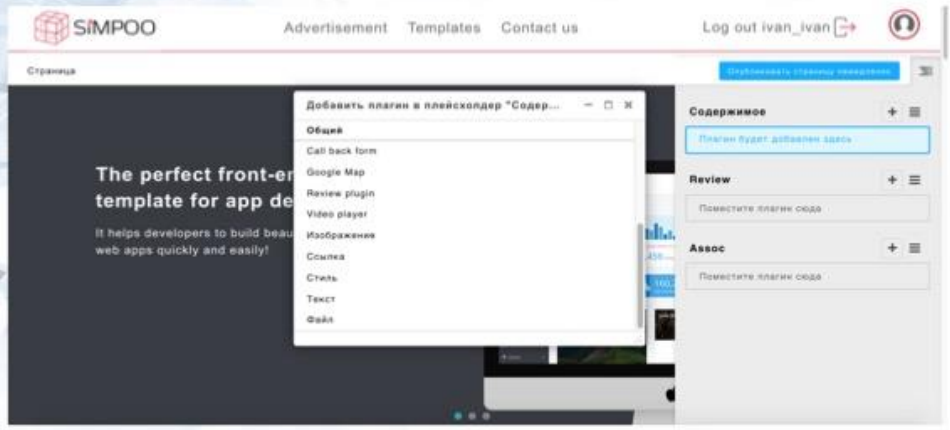
Приклади роботи



Вспиваюче вікно реєстрації

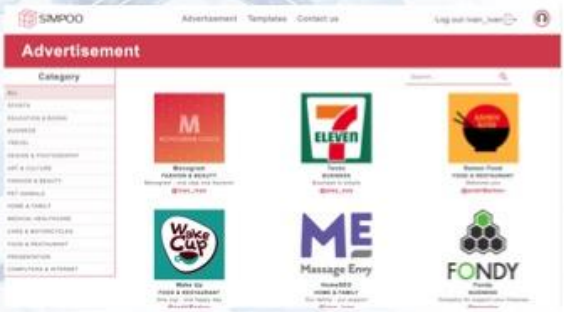
Сторінка перегляду шаблонів візитівок

Приклади роботи



Сторінка редагування візитівки

Приклади роботи



Сторінка перегляду візитівок

Сторінка перегляду статистики візитівок користувача

Приклади роботи

Сторінка зворотнього зв'язку

Приклади роботи

Готова візитівка

Конференції

IV Міжнародна науково-технічна конференція «Поліграфічні, мультимедійні та web-технології» 2019 року. Результати опубліковані у вигляді тез доповіді.

Висновки

- проведено аналіз доступних програмних рішень;
- розроблено структуру веб-додатку та архітектуру БД;
- реалізовано серверну та клієнтську частини;
- реалізовано вимоги до ПЗ;
- розроблено особистий кабінет з конструктором візитівки;
- протестовано роботу веб-додатку.

Дякую за увагу!

Додаток 3

Лістинг програми

Модуль контролерів. Функції-контролери клієнтської частини веб-додатку

```
from django.shortcuts import render, redirect
from django.http import HttpResponse, JsonResponse, HttpResponseRedirect
from django import template
from cms.models.pagemodel import Page
from cms.models.permissionmodels import PageUser
from cms.api import create_page
from cms.forms.wizards import CreateCMSPageForm
from cms.admin.forms import AddPageForm
from cms.models import Page, PageType, Title
from cms.admin.forms import AddPageForm
from .forms import SignUpForm
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from .models import PageByUserStatistic
base_context = {
    'register_form': SignUpForm,
}
from cms.admin import pageadmin
from cms.admin.forms import AddPageTypeForm
from cms.api import publish_page, get_page_draft

def index(request, **kwargs):
      ctx = {}
            ctx.update(base_context)
            return render(request, 'user_app/pages/home.html', ctx)

def show_template(request, id):
    current_page = Page.objects.get(id=id)
    template_for_page = current_page.get_template()
    title = current_page.get_title()
    print(current_page.get_slug())
    print(type(template_for_page))
    ctx = {
        'id_template': id,
        'title': title,
        'page': current_page,
        'choice_template_btn_flag': True,
        'disabled': True
    }
    ctx.update(base_context)
```

```
    return render(request, template_for_page, ctx)

from cms.api import create_page
from .models import PageByUser

def set_template(request, id_template=''):
    if request.method == 'GET':
        print(CreateCMSPageForm)
        current_page = Page.objects.get(id=id_template)
        print(current_page.get_title())
        type_pages = Page.objects.filter(
            is_page_type=True,
            publisher_is_draft=True,
        )
ctx = {'form': AddPageForm,
               'type_pages': type_pages,
               'level_3': PageType.objects.filter(is_page_type=True,
node__depth=3),
               'current_page': current_page,
               'id_template': id_template}
        ctx.update(base_context)
        return render(request, 'user_app/add_form.html', ctx)
    elif request.method == 'POST':
        from django.template.defaultfilters import slugify
        from unidecode import unidecode
        title = request.POST.get('title')
        description = request.POST.get('description')
        print()
        # page_title = request.POST.get('page_title')
        type_page_id = request.POST.get('type_page')
        print("slugify ", slugify(unidecode(title)))
        print(Page.objects.get(pk=int(type_page_id)).get_template())
        page = create_page(title=title,
                           meta_description= description,
                           language='ru',

template=Page.objects.get(pk=int(type_page_id)).get_template(),
                           created_by=request.user,
                           slug=slugify(unidecode(title)))
        page_by_user = PageByUser.objects.create(page_user_custom=page,
                                                 user=request.user,

category=request.POST.get('parent_page'),
```

```
image=request.FILES['image'])
        PageByUserStatistic.objects.create(page_stat=page_by_user)
        print('page_by_user')
        print(page_by_user)
        username = request.user.username
        password = request.user.password
        return HttpResponseRedirect('/cabinet/')

def _login(request):
    if request.method == 'GET':
        ctx = {}
        ctx.update(base_context)
        return render(request, 'user_app/login.html', ctx)
    elif request.method == 'POST':
        print('asdasdasdsa')
        username = request.POST.get('login')
        password = request.POST.get('password')
        print(username, password)
        user = authenticate(username=username, password=password)
        if user:
            login(request, user)
            return redirect('cabinet')

from django.template.loader import get_template
from django.template import Context
from django.template.loader import render_to_string
from django.core.mail import EmailMultiAlternatives
from .models import CallBack


def contact_form(request):
    author_user_page =
User.objects.get(pk=int(request.POST.get('create_by')))
    user_mail = request.POST.get('to_email')
    theme = request.POST.get('theme')
    phone = request.POST.get('phone')
    name_user = request.POST.get('name_user')
    user_msg = request.POST.get('msg')
    email = request.POST.get('email')
    context_email = {
        'phone': phone,
        'name_user': name_user,
```

```
            'msg': user_msg,
            'email': email,
            'user_page': request.POST.get('user_page'),
        }
        subject, from_email = theme, 'kpi.study1@gmail.com'
        message = render_to_string('snippets/email_template.html',
    context_email)
        msg = EmailMultiAlternatives(subject, 'test!', from_email, [user_mail])
        msg.attach_alternative(message, "text/html")
        msg.send()
        CallBack.objects.create(
            page=Page.objects.get(id=int(request.POST.get('user_page_id'))),
            user=author_user_page,
            theme=theme,
            email=email,
            phone=phone,
            message=user_msg,
            name_human=name_user
        )
        return JsonResponse({
            'send': True
        })

    def start(request):
        ctx = {}
        level_2 = PageType.objects.filter(is_page_type=True, node__depth=2)
        level_3 = PageType.objects.filter(is_page_type=True, node__depth=3)
        ctx['level_2'] = level_2
        ctx['level_3'] = level_3
        ctx.update(base_context)
        return render(request, 'user_app/start.html', ctx)

    def signup(request):
        print('signup')
        if request.method == 'POST':
            form = UserCreationForm(request.POST)
            if form.is_valid():
                username = form.cleaned_data.get('username')
                raw_password = form.cleaned_data.get('password1')
                u = User(username=username, password=raw_password)
                u.is_active = True
                u.is_superuser = True
                u.save()
```

```
            print('is super user', u.is_superuser)
            user = authenticate(username=username, password=raw_password)
            login(request, user)
            return redirect('cabinet')
    else:
        form = UserCreationForm()
    return render(request, 'user_app/signup.html', {'form': form})

def _logout(request):
    logout(request)
    return redirect('/')

def cabinet(request):
    ctx = {}
    if request.user.is_authenticated:
         try:
         user_pages =
    PageByUser.objects.filter(pagebyuserstatistic__page_stat__user=request.
    user).distinct()
            ctx['user_pages'] = user_pages
        except Exception as e:
            print(type(e))
        ctx['current_page'] = 'my_template'
        ctx.update(base_context)
        return render(request, 'user_app/cabinet.html', ctx)

def statistics(request):
    ctx = {}
    if request.user.is_authenticated:
        try:
            user_pages =
PageByUser.objects.filter(pagebyuserstatistic__page_stat__user=request.user
).distinct()
            ctx['user_pages'] = user_pages
        except Exception as e:
            print(type(e))
        ctx['current_page'] = 'my_template'
        ctx.update(base_context)
        return render(request, 'user_app/partials/statistic.html', ctx)

def about(request):
    ctx = {}
    ctx.update(base_context)
```

```
    return render(request, 'user_app/pages/about.html', ctx)

def advertisement(request):
    ctx = {}
    level_2 = PageType.objects.filter(is_page_type=True, node__depth=2)

print(PageByUser.objects.filter(page_user_custom__title_set__published=True
).distinct())
ctx['all_pages'] =
PageByUser.objects.filter(page_user_custom__title_set__published=True).dist
inct()
    ctx['level_2'] = level_2
    for page in PageByUser.objects.all():
        print(page.page_user_custom.is_published('ru'))
    print(PageByUser.objects.all())

    title_arr = []
    for elem in level_2:
        if elem.get_child_pages():
            for child_page in elem.get_child_pages():
                title_arr.append(child_page.get_title())
    title_arr = list(set(title_arr))
    ctx['title_arr'] = title_arr
    ctx.update(base_context)
    return render(request, 'user_app/pages/advertisement.html', ctx)

def templates(request):
    level_2 = PageType.objects.filter(is_page_type=True, node__depth=2)
    level_3 = PageType.objects.filter(is_page_type=True, node__depth=3)
    title_arr = []
    for elem in level_2:
        if elem.get_child_pages():
            for child_page in elem.get_child_pages():
                title_arr.append(child_page.get_title())

    title_arr = list(set(title_arr))
   ctx = {
        'level_2': level_2,
        'level_3': level_3,
        'title_arr': title_arr
    }
    ctx.update(base_context)
    print(ctx)
```

```
    return render(request, 'user_app/pages/templates.html', ctx)

def user_profile(request):
    ctx = {}
    ctx.update(base_context)
    return render(request, 'user_app/pages/user_profile.html', ctx)

def my_profile(request):
    ctx = {}
    ctx['current_page'] = 'profile'
    ctx.update(base_context)
    return render(request, 'user_app/pages/profile.html', ctx)

def contact_form_page(request):
    ctx = {}
    ctx['current_page'] = 'contact_form'
    if request.user.is_authenticated:
        all_query = CallBack.objects.filter(user=request.user).order_by('-
date')
        ctx['all_query'] = all_query
    ctx.update(base_context)
    return render(request, 'user_app/pages/contact_form.html', ctx)

def create_new(request):
    level_2 = PageType.objects.filter(is_page_type=True, node__depth=2)
    level_3 = PageType.objects.filter(is_page_type=True, node__depth=3)
    title_arr = []
    for elem in level_2:
        if elem.get_child_pages():
            for child_page in elem.get_child_pages():
                title_arr.append(child_page.get_title())
    title_arr = list(set(title_arr))
    ctx = {
        'level_2': level_2,
        'level_3': level_3,
    }
    ctx['title_arr'] = title_arr
    ctx['current_page'] = 'create_new'
    ctx.update(base_context)
    return render(request, 'user_app/pages/create_new.html', ctx)
```

HTML розмітка. Шаблон візитівки

```
{% extends "base.html" %}
{% load cms_tags %}
{% block title %}{% page_attribute "page_title" %}{% endblock title %}
{% load static %}
{% block header %} {% endblock %}
{% block content %}
<div class="">
<link rel="stylesheet" href="{{STATIC_URL}}css/templates/a.css">
<header id="home">
    <nav>
        <div class="container-fluid">
            <div class="row">
                <div class="col-md-8 col-md-offset-2 col-sm-8 col-sm-
offset-2 col-xs-8 col-xs-offset-2">
                    <nav class="pull">
                        <ul class="top-nav">
                            <li><a href="#intro">Introduction <span
class="indicator"><i class="fa fa-angle-right"></i></span></a>
                            </li>
                            <li><a href="#features">Features <span
class="indicator"><i
                                    class="fa fa-angle-
right"></i></span></a></li>
                            <li><a href="#responsive">Responsive Design
<span class="indicator"><i
                                    class="fa fa-angle-
right"></i></span></a></li>
                            <li><a href="#portfolio">Portfolio <span
class="indicator"><i class="fa fa-angle-right"></i></span></a>
                            </li>
                            <li><a href="#team">Team <span
class="indicator"><i
                                    class="fa fa-angle-
right"></i></span></a></li>
                            <li><a href="#contact">Get in Touch <span
class="indicator"><i
                                    class="fa fa-angle-
right"></i></span></a></li>
                        </ul>
                    </nav>
                </div>
```

```
            </div>
        </div>
    </nav>
    {% placeholder 'cover_img_txt' or %}
    {% endplaceholder %}
</header>
<section class="intro text-center section-padding" id="intro">
    <div class="container">
        {% placeholder 'title_text'  or %}
        <div class="row">
            <div class="col-md-8 col-md-offset-2">
                <h1 class="arrow">A Creative Portfolio Template</h1>
                <p>Sed a lorem quis neque interdum <a href="#">consequat ut
sed sem</a>. Duis quis tempor nunc. Interdum
                    et malesuada fames ac ante ipsum primis in faucibus.
Praesent id tempor ipsum. Fusce at massa ac
                    nunc porta fringilla sed eget neque. Quisque quis
pretium nulla. Fusce eget bibendum neque, vel
                    volutpat augue. Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Interdum et malesuada fames
                    ac ante ipsum primis in faucibus.</p>
            </div>
        </div>
        {% endplaceholder %}
    </div>
</section>
<section class="features text-center section-padding" id="features">
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <h1 class="arrow">Love what you do, and you'll do it
well</h1>
                <div class="features-wrapper">
                    {% placeholder 'col' or %}
                    <div class="col-md-4 ">
                        <div class="icon">
                            <i class="fa fa-laptop shadow"></i>
                        </div>
                        <h2>Digital Design</h2>
                        <p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Sed a lorem quis neque interdum
                            consequat ut sed sem. Duis quis tempor nunc.
Interdum et malesuada fames ac ante ipsum
```

```
                        primis in faucibus.</p>
                    </div>
                    <div class="col-md-4  delay-05s">
                        <div class="icon">
                            <i class="fa fa-code shadow"></i>
                        </div>
                        <h2>Web Development</h2>
                        <p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Sed a lorem quis neque interdum
                            consequat ut sed sem. Duis quis tempor nunc.
Interdum et malesuada fames ac ante ipsum
                            primis in faucibus.</p>
                    </div>
                    <div class="col-md-4  delay-1s">
                        <div class="icon">
                            <i class="fa fa-heart shadow"></i>
                        </div>
                        <h2>Creative Direction</h2>
                        <p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Sed a lorem quis neque interdum
                            consequat ut sed sem. Duis quis tempor nunc.
Interdum et malesuada fames ac ante ipsum
                            primis in faucibus.</p>
                    </div>
                    <div class="clearfix"></div>
                    {% endplaceholder %}
                </div>
            </div>
        </div>
    </div>
</section>
<section class="text-center" id="responsive">
    <div class="container-fluid nopadding responsive-services">
        <div class="wrapper">
            <div class="iphone">
                {% placeholder 'col_50' %}
            </div>
            <div class="fluid-white"></div>
        </div>
        <div class="container designs">
            <div class="row">
                <div class="col-md-5 col-md-offset-7">
                    <div id="servicesSlider">
```

```
                        <ul class="slides">
                            {% placeholder 'text_title_50' or %}
                            <li>
                                <h1 class="arrow">Responsive Design
Specialists</h1>
                                <p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Nunc ultricies nulla non
                                    metus pulvinar imperdiet. Praesent non
adipiscing libero. </p>
                                <p>
                                    Mauris ultrices odio vitae nulla
ultrices iaculis. Nulla rhoncus odio eu lectus
                                    faucibus facilisis. Maecenas ornare
augue vitae sollicitudin accumsan. </p>
                                <p>Etiam eget libero et erat eleifend
consectetur a nec lectus. Sed id tellus lorem.
                                    Suspendisse sed venenatis odio, quis
lobortis eros.</p>
                            </li>
                            {% endplaceholder %}
                        </ul>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
<section class="subscribe text-center">
    <div class="container">
        <h1><i class="fa fa-paper-plane"></i><span>Subscribe to stay in the
loop</span></h1>
        <div class="green__callback__form">
        {% placeholder 'form' %}
            <br>
            <br>    <br>
            </div>
    </div>
</section>
<section class="dark-bg text-center section-padding contact-wrap"
id="contact">
    <a href="#top" class="up-btn"><i class="fa fa-chevron-up"></i></a>
    <div class="container">
        <div class="row">
```

```
            <div class="col-md-12">
                <h1 class="arrow">Drop us a line</h1>
            </div>
        </div>
        <div class="row contact-details">
            <div class="col-md-4">
                <div class="light-box box-hover">
                    <h2><i class="fa fa-map-
marker"></i><span>Address</span></h2>
                    <p>Level 6, 23 Pitt St, Sydney</p>
                </div>
            </div>
            <div class="col-md-4">
                <div class="light-box box-hover">
                    <h2><i class="fa fa-mobile"></i><span>Phone</span></h2>
                    <p>+61 9 211 3747</p>
                </div>
            </div>
            <div class="col-md-4">
                <div class="light-box box-hover">
                    <h2><i class="fa fa-paper-
plane"></i><span>Email</span></h2>
                    <p><a href="#">hey@halcyondays.com</a></p>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="col-md-12">
                <ul class="social-buttons">
                    <li><a href="#" class="social-btn"><i class="fa fa-
dribbble"></i></a></li>
                    <li><a href="#" class="social-btn"><i class="fa fa-
twitter"></i></a></li>
                    <li><a href="#" class="social-btn"><i class="fa fa-
envelope"></i></a></li>
                </ul>
            </div>
        </div>
    </div>
</section>
<footer>
    <div class="container">
        <div class="row">
```

```
            <div class="col-md-6">
                <ul class="legals">
                    <li><a href="#">Terms &amp; Conditions</a></li>
                    <li><a href="#">Legals</a></li>
                </ul>
            </div>
            <div class="col-md-6 credit">
                <p>Designed &amp; Developed by <a
href="http://www.peterfinlan.com/">Peter Finlan</a> exclusively for <a

href="http://tympanus.net/codrops/"><em>Codrops</em></a></p>
            </div>
        </div>
    </div>
</footer>
</div>
{% endblock %}
```

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ВЕБ-ДОДАТОК ДЛЯ СТВОРЕННЯ ТА УПРАВЛІННЯ ОНЛАЙН-ВІЗИТІВКАМИ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Т.М. Заболотня

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.В. Пивоварчук

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-додаток для створення та управління онлайн-візитівками, який являє собою веб-застосунок, створений на платформі Python Django.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність елементів сторінок web-ресурсу;
2) наявність доступу до бази даних візитівок;
3) взаємодію сервера з клієнтською частиною конструктора візитівки;
4) забезпечення коректної обробки запитів від користувача;
5) забезпечення належного рівня безпеки даних;
6) зручність роботи з веб-додатком;
7) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність web-ресурсу перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування web-ресурсу в різних web-браузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-ДОДАТОК ДЛЯ СТВОРЕННЯ ТА УПРАВЛІННЯ ОНЛАЙН-ВІЗВТІВКАМИ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Т.М. Заболотня

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.В. Пивоварчук

2019

ЗМІСТ

1. Опис структури web-ресурсу

Веб-додаток для створення та управління онлайн-візитівками складається із статичних web-сторінок та web-сторінок, вміст яких формується динамічно. Web-ресурс є одномовним з англійською мовою.

До статичних належать наступні web-сторінки:

- головна сторінка;
- сторінка статистики візитівок;

Динамічна частина web-ресурсу включає наступні web-сторінки:

- сторінка перегляду шаблонів онлайн-візитівок;
- сторінка перегляду візитівок;
- сторінка перегляду візитівок конкретного користувача;

сторінка створення візитівки;

- сторінка персональної інформації користувача;
- сторінка перегляду шаблону;
- сторінка редагування візитівки;
- сторінка форми зворотнього зв'язку.

Кожна web-сторінка містить інформацію про користувача, що ввійшов в систему, його логін та кнопку виходу із системи, а також логотип системи, яке містить посилання на головну сторінку, та меню.

Web-ресурс також має адміністративну панель, доступну лише адміністратору ресурсу.

2. Опис вмісту статичних web-сторінок

Головна сторінка виступає в якості інформаційної, тому на ній знаходиться покрокова інструкція взаємодії користувача із системою, приклади деяких можливих для використання шаблонів, список категорій шаблонів та презентаційний блок з інформацією про призначення даного веб-додатку.

Сторінка Statistics (рис. 1) відображає статистику переглядів кожної візитівки користувача у вигляді графіка. Для графіків реалізована можливість скачування у форматах JPEG, PDF, PNG, SVG, CSV, XLS, друк та перегляд у повноекранному режимі. Користувач може переглядати статистику тільки свої візитівок, для забезпечення конфіденційності інформації.

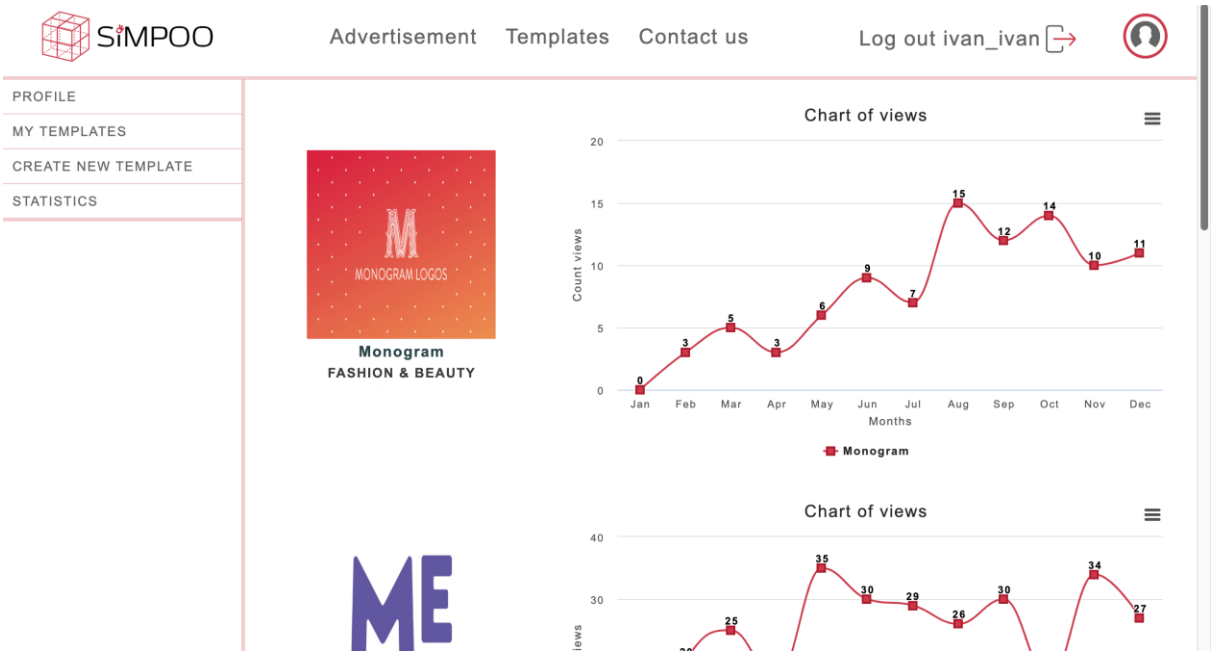


Рис. 1. Сторінка перегляду статистики візитівок користувача

3. Процедура авторизації користувача

Для реєстрації та авторизації були використані спливаючі вікна (рис. 2). Посилання на виклик даного вікна знаходиться у хедері на всіх сторінках системи. Вікно авторизації має поля логіна та паролю, а реєстрації – логін, емейл, пароль та підтвердження паролю. Також для

обох вікон були додані чербокси показати пароль та кнопки переходу між даними вікнами.

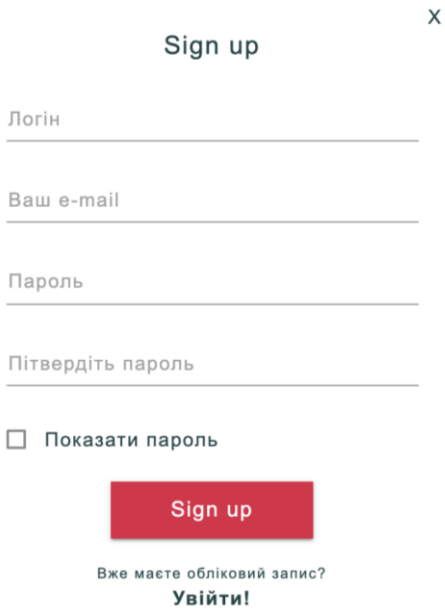


Рис. 2. Вспиваюче вікно реєстрації

4. Створення візитівки та робота з нею

Принцип роботи зі сторінкою створення та редагування візитівки (рис. 3) полягає в наступному: спочатку користувач переходить на сторінку перегляду шаблону, після чого натискає на кнопку вибору шаблону, далі з’являється спливаюче вікно для вводу початкової інформації про майбутню візитівку, тобто завантаження зображення (логотип), яке буде відображатися на сторінці перегляду візитівок, назву, короткий опис та обирає категорію зі списку запропонованих. Вибір категорії не залежить від категорії, до якої початково належав шаблон, для зручності використання додатку користувачем. Після заповнення форми спливаючого вікна візитівка потрапляє до списку у особистому кабінеті користувача у статусі неопубліковано. Далі, при переході на дану візитівку, користувач обирає запропоновані плагіни для кожної ділянки шаблону та заповнює його конентом у спливаючих вікнах. Для зручності користування даним конструктором була створена можливість обновлення сторінки або перехід на іншу без втрати даних змін. При цьому зміни не

відображаються на опублікованій візитівці, доки користувач не натисне опублікувати зміни. Таким чином користувач може не хвилюватися про втрату доданих змін при збоях у мережі або комп'ютері. Також для зручності редагування елементів шаблону була додана можливість подвійного кліку на місце у шаблоні, що потребує редагування, при цьому з'являється спливаюче вікно з налаштуваннями плагіну розташованого у цьому місці. Також дана сторінка надає можливість редагування попередньо введених даних при створені візитівки: назву, опис та зображення. Також доступна зміна статусу (опубліковано/неопубліковано).

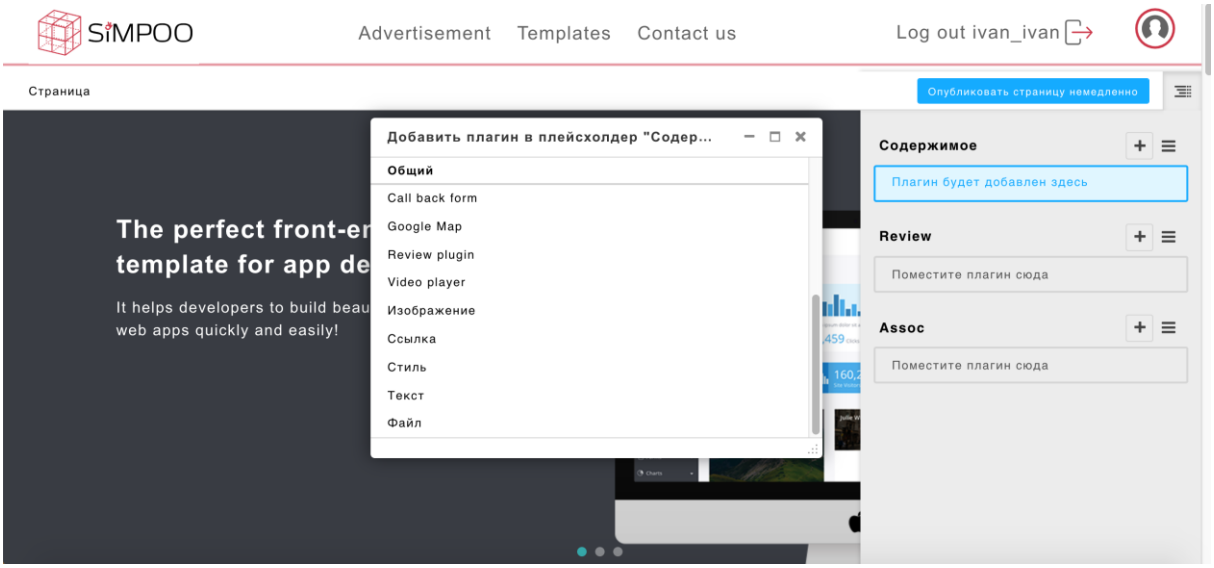


Рис. 3. Сторінка редагування візитівки

5. Сторінки перегляду візитівок та шаблонів

Сторінка Advertisement (рис. 4) відповідає за перегляд візитівок, які знаходяться в опублікованому статусі. На даній сторінці у вигляді оглядового списку розташовані візитівки з короткою інформацією про них: назва, опис, категорія та логін автора. Також на даній сторінці доступна фільтрація по категоріям, що виводяться з лівого боку, та пошук за назвою візитівки.

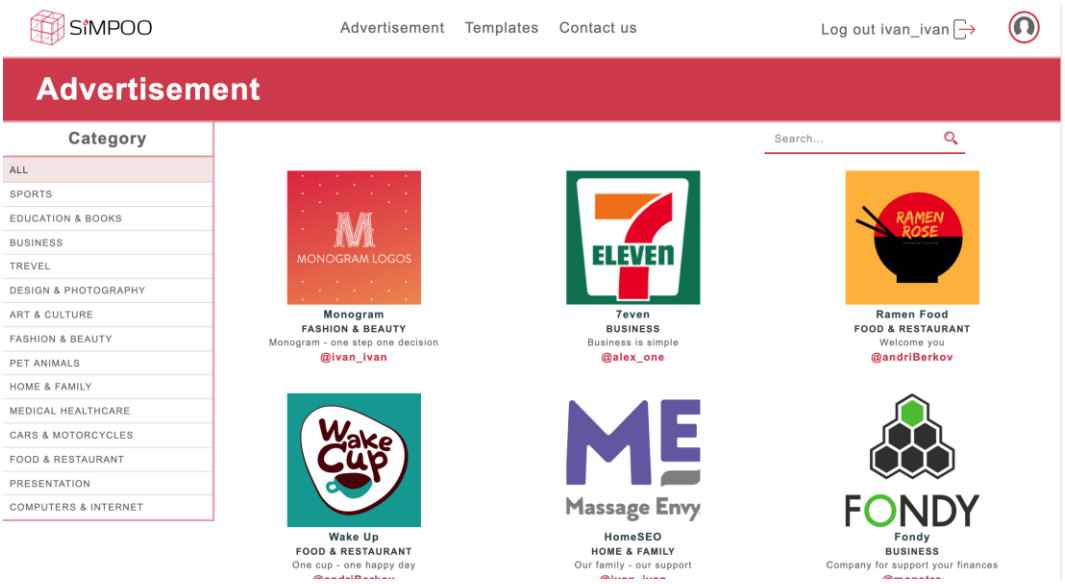


Рис. 4. Сторінка перегляду візитівок

Сторінка Templates (рис. 5) відповідає за перегляд шаблонів для майбутніх візитівок. Сторінка зовнішнім виглядом дуже схожа на попередню, так само у вигляді оглядового списку виводяться шаблони з короткою інформацією: назва, категорія, коротка інформація та тип шаблону. На даній сторінці також доступна фільтрація за категоріями та типами, а також пошук візитівки за назвою.

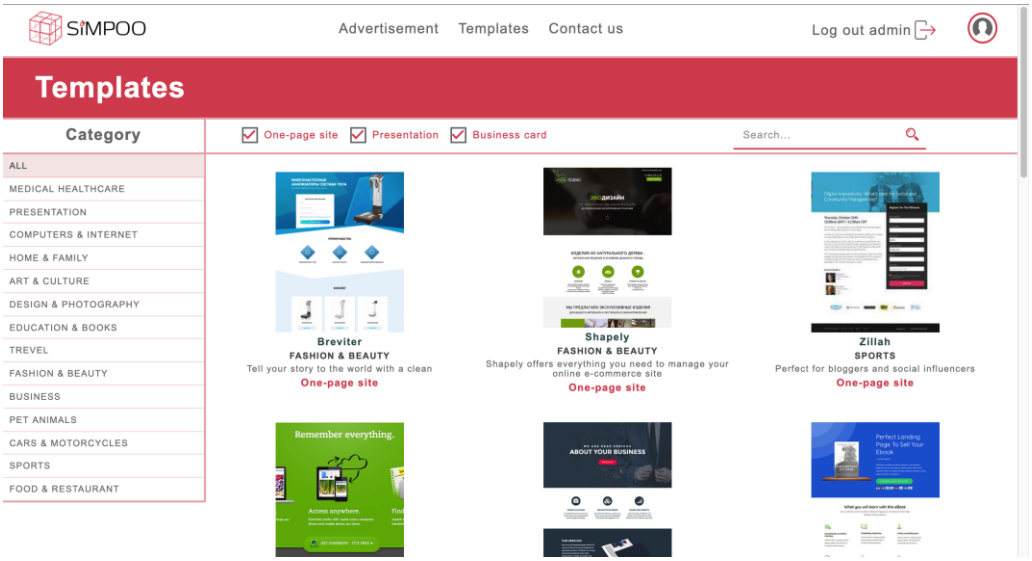


Рис. 5. Сторінка перегляду шаблонів візитівок

6. Перегляд відгуків з форми зворотнього зв'язку

Сторінка Contact Form (рис. 6), на якій відображаються повідомлення відправлені із форм зворотнього зв’язку розташованих у візитівках. Всі повідомлення дублюються на пошту автора візитівки, яку він задав при налаштуванні плагіну контактної форми і все спілкування відбувається за допомогою пошти або по телефону, проте на даній сторінці можна переглянути всі повідомлення із всіх візитівок користувача та відмітити як переглянуте. При натискані на поле таблиці з’являється спливаюче вікно з повним текстом листа та всією іншою інформацією, такою як email, телефон, ім’я відправника, темою та назвою візитівки, з якої повідомлення було відправлене. Також в даному вікні доступна функція відмітки листа як прочитаного.

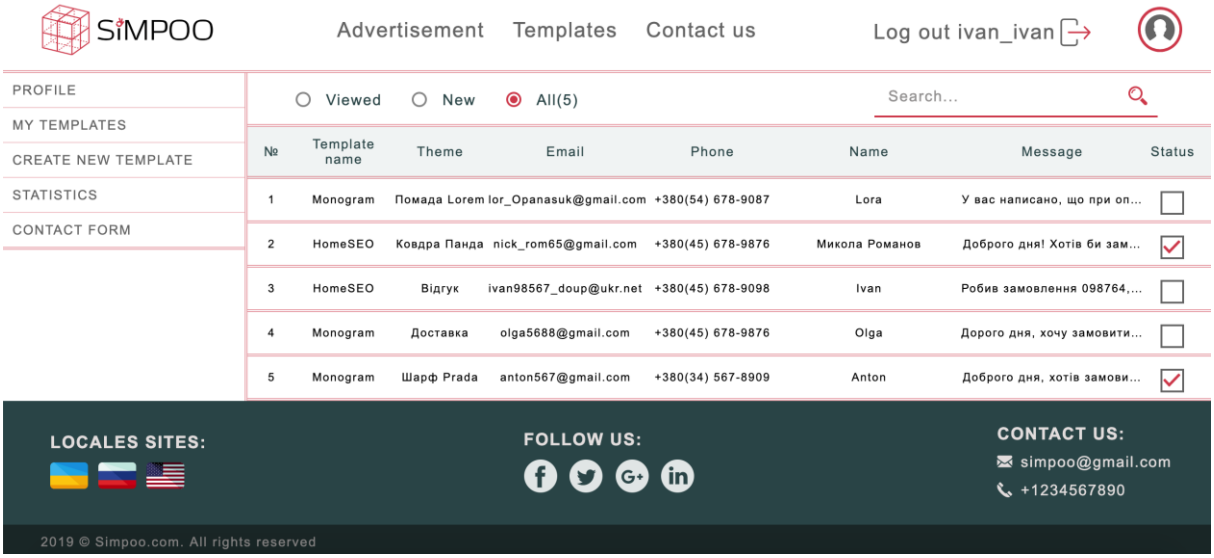


Рис. 6. Сторінка зворотнього зв’язку