НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

на тему: «Програмні засоби реплікації та синхронізації новин інформаційних каналів кафедри (факультету)»

Виконав (-ла):
студент (-ка) IV курсу, групи ТВ-71
Ващенко Аліна Сергіївна _____

Керівник:
Асистент
Діденко Олексій Олександрович _____

Рецензент:

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Ващенко Аліні Сергіївні_____
(прізвище, ім’я, по батькові)

1. Тема роботи _____Програмні засоби реплікації та синхронізації новин інформаційних каналів кафедри (факультету)_____
_____

керівник роботи____Діденко Олексій Олександрович, асистент_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”_24_” травня 2021р. № **_1267с___**

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи ____розроблена програма написана мовою PHP, під платформу Web-браузера_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _з’ясувати, яким чином відбувається публікація постів в такі соціальні мережі, як Telegram та Facebook через сторонні сервіси; сформувати вимоги до системи, що створюється; спроектувати і розробити веб-сервіс для реплікації та синхронізації інформаційних каналів; протестувати продукт._____
_____
_____

5. Перелік ілюстративного матеріалу

Титульна сторінка, Актуальність роботи, Мета та основні завдання роботи, Аналіз існуючих програмних рішень, Ключові моменти, діаграма прецедентів, Засоби розробки, Інтерфейс головної сторінки, Створення поста, Результат роботи програми,Висновки_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____202___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	Захист програмного продукту		
8.	Передзахист		
9.	Захист		

Студент _____ _____Ващенко А. С._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Діденко О. О._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дипломну роботу виконано на 53 аркушах, вона містить 3 додатки. В роботі наведено 19 рисунків та 2 таблиці. Включає перелік посилань на використані джерела з 15 найменувань.

Метою даної дипломної роботи є організація ефективного процесу обробки сторінок кафедри в соціальних мережах шляхом створення програмних засобів реплікації та синхронізації новин інформаційних каналів.

Було досліджено основні методи публікації інформації в соціальні мережі через сторонні сервіси. Головними критеріями для вибору методів стала підтримка їхніх функцій актуальною версією мови програмування PHP.

В роботі було спроектовано систему для реплікації та синхронізації новин інформаційних каналів та розроблено програмний код продукту.

Модуль публікації в соціальні мережі даної роботи можна використовувати, як частину програмного коду сайту кафедри чи факультету.

Ключові слова: веб-сервіс, авторизований доступ, публікація в соціальні мережі, автопостинг, пост.

ABSTRACT

The thesis is presented in 53 pages, it contains 3 appendixes. The research work shows 19 figures and 2 tables. Includes a bibliography of 15 references.

The purpose of this thesis is to organize an effective process of processing the pages of the department in social networks by creating software tools for replication and synchronization of news information channels.

The main methods of publishing information on social networks through third-party services were studied. The main criteria for selecting methods was the support of their functions with the current version of the PHP programming language.

The system for replication and synchronization of news information channels was designed and the program code of the product was developed.

The module of publication in social networks of this work can be used as part of the program code of the department or faculty site.

Keywords: web service, authorized access, publication on social networks, autoposting, post.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД — база даних.

СКБД — система керування базами даних.

MVC — Model-View-Controller.

ОС — операційна система.

ТЗ — технічне завдання.

HTML — HyperText Markup Language.

CSS — Cascading Style Sheets.

PHP — Hypertext Preprocessor.

UML — Unified Modeling Language.

API — Application Programming Interface.

QA — quality assurance.

Пост — це повідомлення на форумі, блозі або у соціальній мережі.

Автопостинг — це автоматична трансляція повідомлень на сайти, в блоги, мікроблоги, соціальні мережі тощо.

ВСТУП

Сьогодні Інтернет є найактуальнішим методом розказати про себе світові. Кожне сучасне підприємство повинно мати сторінку в соціальній мережі чи сайт, щоб потенційні клієнти могли дізнатись про нього, а вже наявні користувачі послуг — новини, акції тощо. Якщо проводити аналогію від підприємства до факультету університету, то новими клієнтами стають абітурієнти, яким треба показати, чому потрібно обрати навчання саме тут, а постійними клієнтами є студенти цієї кафедри чи цього факультету, яким всі новини та зміни в роботі зручніше бачити в мережі, ніж на типовій дошці оголошень.

Проте різні користувачі обирають різні соціальні мережі для щоденного використання: одні Facebook, другі Telegram, треті Instagram. Для того, щоб охопити всю потенційну аудиторію та зацікавити її, потрібно активно вести свої сторінки в основних соцмережах та публікувати контент на кожну з них. А створювати ідентичні пости окремо для кожного профілю — нераціональна трата часу та ресурсів.

Отже, актуальність дипломної роботи зумовлена кількома чинниками:

- відсутністю активної обробки сторінок кафедри в соціальних мережах задля зацікавлення підписників;
- розбіжністю контенту на наявних сторінках;
- необхідністю налагодження та раціоналізації процесу публікації новин.

Мета роботи: організація ефективного процесу обробки сторінок кафедри в соціальних мережах шляхом створення програмних засобів реплікації та синхронізації новин інформаційних каналів.

Завдання:

- з’ясувати, яким чином відбувається публікація постів в такі соціальні мережі, як Telegram та Facebook через сторонні сервіси;
- сформувати вимоги до системи, що створюється;
- спроектувати і розробити веб-сервіс для реплікації та синхронізації інформаційних каналів;
- протестувати продукт.

1 ПОСТАНОВКА ЗАДАЧІ АВТОМАТИЧНОЇ ПУБЛІКАЦІЇ КОНТЕНТУ

Метою роботи є організація ефективного процесу обробки сторінок кафедри в соціальних мережах шляхом створення програмних засобів реплікації та синхронізації інформаційних каналів. Призначенням даного програмного засобу є надання зрозумілого та чіткого інтерфейсу для формування посту-новини кафедри в такі соціальні мережі, як Telegram та Facebook, реалізація збереження наданої інформації в базу даних та подальшої її публікації.

При виконанні роботи потрібно провести проектування системи та моделювання діаграми прецедентів та діаграми розміщення. По вищевказаним матеріалам необхідно розробити загальний вигляд сторінок за допомогою мови розмітки гіпертексту HTML та спеціальної мови стилю сторінок CSS, а також програмно-апаратну частину сервіса, використовуючи PHP-фреймворк Laravel.

Рішення повинно підтримувати наступні функції:

- авторизацію користувача (адміністратора);
- реєстрацію нового користувача вже існуючим адміністратором;
- формування списку існуючих користувачів системи;
- відображення історії опублікованих за допомогою сервісу постів;
- формування статті-новини з можливістю прикріпити зображення;
- постинг допису в обрані соціальні мережі.

2 ОПИС ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ДЛЯ РЕПЛІКАЦІЇ ТА СИНХРОНІЗАЦІЇ ІНФОРМАЦІЇ

Реплікація — це механізм розподілу даних за вузлами, що дозволяє зберігати копії тих самих даних на різних вузлах мережі з метою прискорення пошуку і підвищення стійкості до відмов. Відношення чи фрагмент є реплікованим, якщо його копії зберігаються на двох або більше вузлах (копії ще називають репліками) [1] .

Синхронізація даних — ліквідація відмінностей між двома копіями даних.

Можемо сказати, що синхронізація — це ціль, а реплікація — спосіб її досягнення. Проте в деяких випадках ці два поняття ототожнюються, як, наприклад, в даній дипломній роботі.

Розглянемо суть завдання: потрібно розробити сервіс, за допомогою якого з єдиної сторінки в мережі інформація поширеться на кілька сервісів одночасно. Бачимо прямий приклад реплікації — копіювання даних з одного джерела на інші. В даному випадку синхронізація як повноцінне поняття не відбудеться, адже зміна якоїсь із реплік не планується.

Сервіси, що виконують описану задачу, називаються сервісами автопостингу. Розглянемо деякі з них та визначимо, які моменти необхідно також реалізувати в своєму продукті, а які не використовувати взагалі.

2.1 Сервіс SMMR

Сервіс SMMR — класичний інструмент для автопостингу, який почав працювати ще в 2015 році. Він підтримує п’ять соціальних мереж: ВКонтакте, Одноклассники, Instagram, Facebook и Twitter.

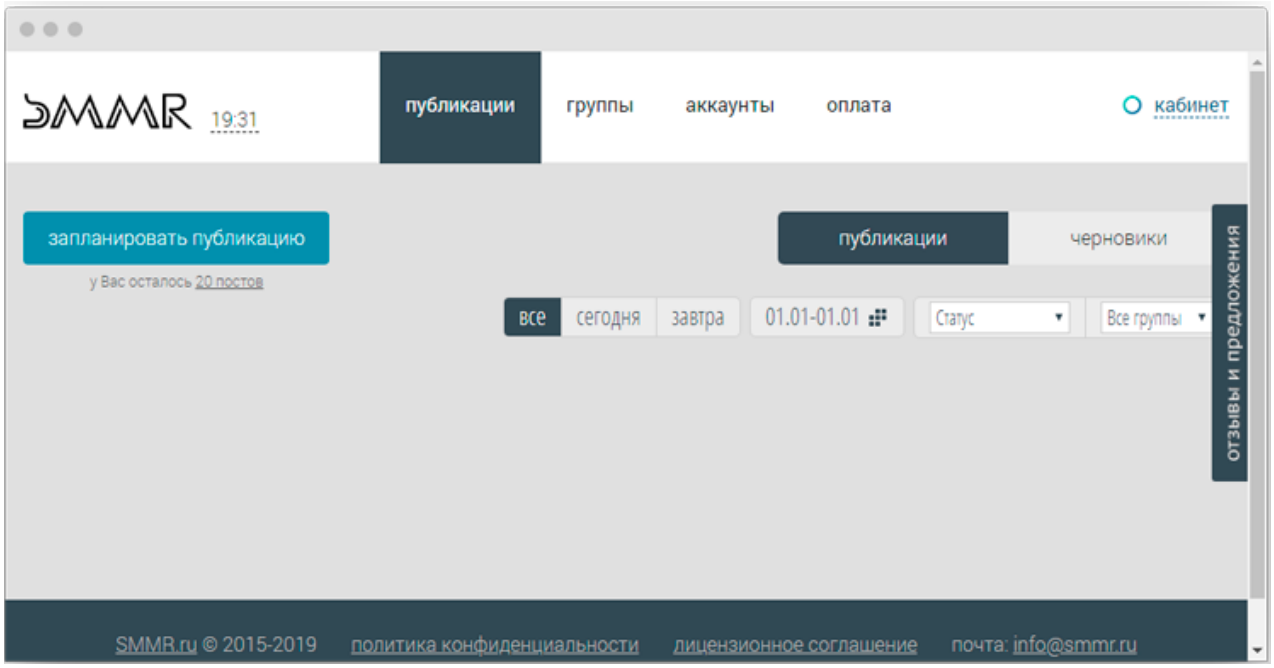

Рисунок 2.1 — Інтерфейс головної сторінки сервісу SMMR

Основні можливості сервісу:

- нічний режим;
- відслідковування статусів публікацій;
- підтримка опитувань та голосувань.

Головними недоліками SMMR є відсутність можливості публікувати в соціальну мережу Telegram та необхідність покупки тарифу для користування функціями.

2.2 Сервіс Roboposting

Сервіс пропонує кілька рішень: API для розміщення публікацій зі свого сайту та автопостинг. Інструмент спеціалізується на ВК, але також підтримує Facebook, Instagram, Twitter.

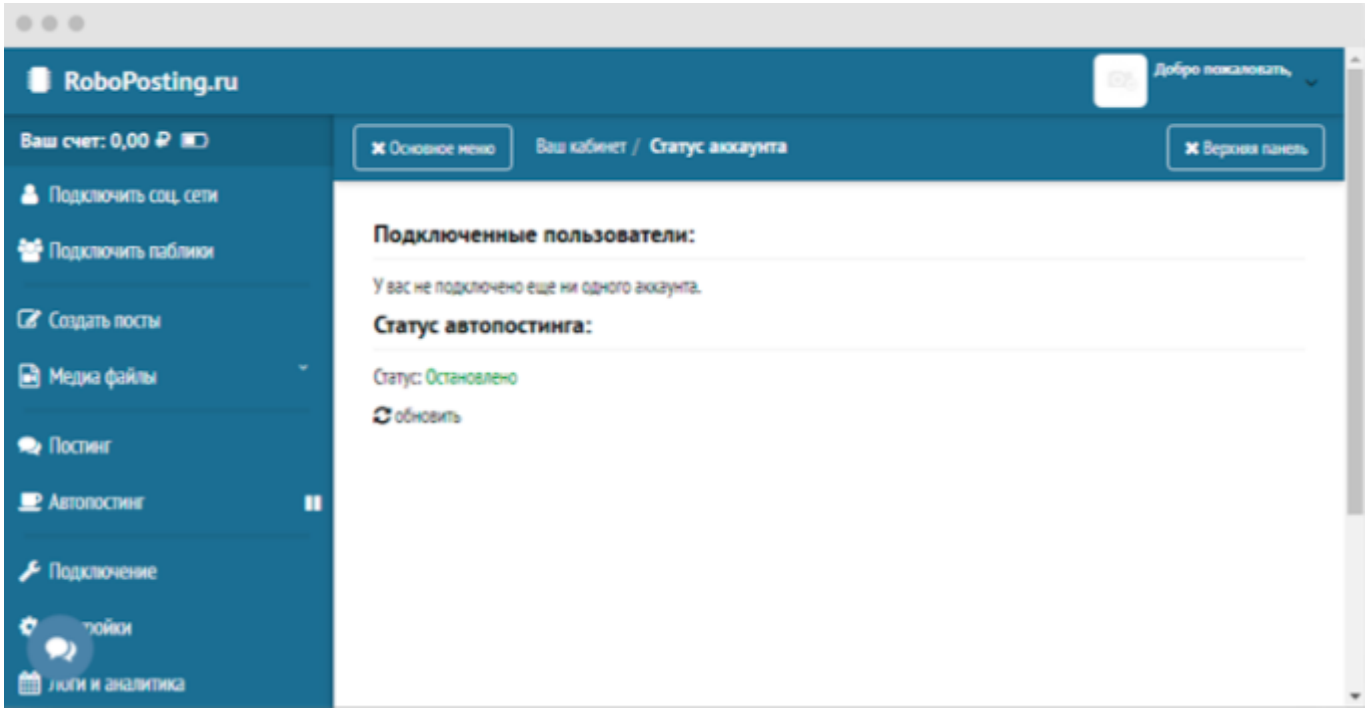

Рисунок 2.2 — Інтерфейс головної сторінки сервісу RoboPosting

Основні можливості сервісу:

- об’єднання постів в папки;
- налаштування часових інтервалів для публікації записів;
- створення шаблонів.

Серед недоліків знову виділяється відсутність можливості публікації в соціальну мережу Telegram та відсутність командної роботи.

2.3 Сервіс NovaPress Publisher

Інструмент працює з 2010 року. Підтримує: Telegram, LinkedIn, Pinterest, ВК, Facebook, Twitter, Instagram.

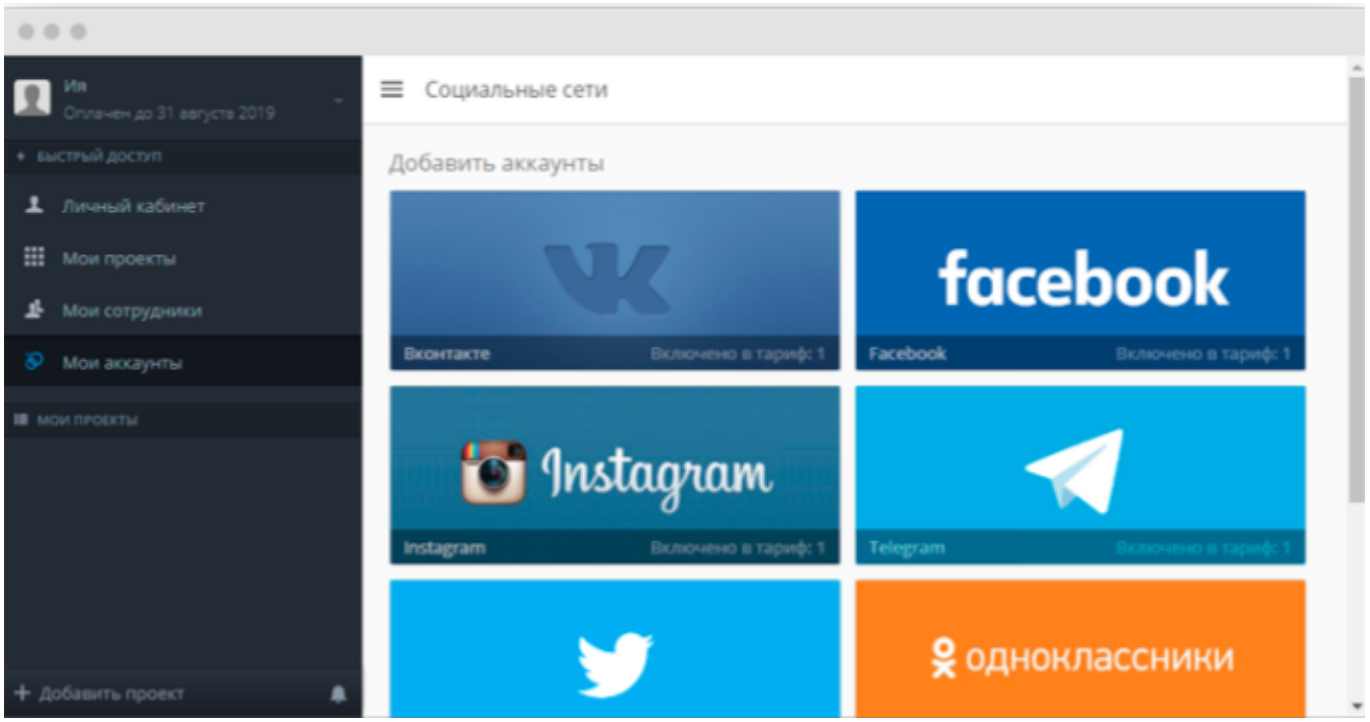

Рисунок 2.3 — Інтерфейс головної сторінки сервісу NovaPress Publisher

Основні можливості:

- автоматичне завантаження постів з сайту компанії;
- командна робота;
- налаштування розкладу публікацій.

З недоліків виділяється необхідність обов’язкової покупки тарифу для користування сервісом.

2.4 Висновки до розділу

Розглянувши основні приклади подібних систем, бачимо, що в деяких немає можливості публікувати в Telegram, але це необхідно реалізувати в своїй системі, адже Telegram зараз є однією з основних мереж в користуванні студентами. А серед плюсів цих сервісів виділяється командна робота, яку також необхідно реалізувати у власній програмі. Також більшість є платними сервісами, що робить роботу з ними проблемною. Майбутній розроблений продукт буде надавати свої послуги безкоштовно.

3 ЗАСОБИ РОЗРОБКИ

Засобами розробки даної роботи були наступні інструменти: HTML, CSS, JavaScript, PHP та фреймворк Laravel.

3.1 Розробка front-end

Розробка front-end — розробка, що зосереджена на елементах, які бачить користувач та з якими взаємодіє. Вона передбачає перетворення коду, що створюється розробниками, у графічний інтерфейс, забезпечуючи представлення даних у зручному та зрозумілому для читання форматі.

Без розробки front-end все, що може побачити звичайний користувач — це нерозбірний код. Але завдяки розробникам веб-інтерфейсів всі можуть легко розуміти та використовувати веб-програми та веб-сайти. Також розробники повинні подбати про правильне відображення сайту на різних пристроях. Всі девайси відрізняються за розміром екрану та операційною системою. Наприклад, користувач може мати мобільний телефон Android, планшет Windows і ноутбук MacBook. Тому робота розробника інтерфейсу полягає в тому, щоб забезпечити і переконатися, що веб-сторінки працюють і коректно відображаються, незалежно від типу пристрою та ОС.

Інша річ, яку повинні врахувати розробники веб-інтерфейсу, полягає в тому, що користувачі можуть віддавати перевагу різним веб-браузерам. Таким чином, необхідно створити веб-сайт, який працює на таких браузерах, як Chrome, Safari, Opera, Internet Explorer, Mozilla Firefox, Microsoft Edge та інші.

Щоб представити контент сайту та організувати взаємодію зі сторінкою в даній дипломній роботі, було використано наступні інструменти: HTML, CSS та JavaScript.

3.1.1 Мова HTML

Мова HTML (HyperText Markup Language — мова розмітки гіпертексту) — це мова тегів, засобами якої здійснюється розмічання веб-сторінок для мережі Інтернет. Браузери отримують HTML-документи з веб-сервера або з локальної пам'яті й передають документи в мультимедійні веб-сторінки. HTML описує структуру веб-сторінки семантично і спочатку включені сигнали для зовнішнього вигляду документа [2].

Текстові документи, що містять розмітку на мові HTML, обробляються спеціальними програмами, які відображають документ в його відформатованому вигляді. Такі програми, що називаються “браузерами”, зазвичай надають користувачу зручний інтерфейс для запиту веб-сторінок, їх перегляду та, при необхідності, відправки введених користувачем даних на сервер.

Документ HTML — це файл з форматом “.html”, в якому текст розмічений HTML-тегами. Засобами HTML задається синтаксис і розміщення тегів, у відповідності з яким браузер відображає вміст веб-документу. Текст самих тегів веб-браузером не відображається [2].

Документ HTML містить текст і вбудовані теги, що є інструкцією про структуру, зовнішній вигляд та функції вмісту. Документ HTML розділяється на дві основні частини: заголовок — head і тіло — body. Заголовок містить такі дані про документ, як його назва і методична інформація, що описує вміст. В тілі знаходиться сам вміст документу, тобто те, що виводиться у вікні браузера.

Приклад структури документу:

```
<html>
    <head>
        <title> HTML struct </title>
    </head>
    <body>
        <h1> Header </h1>
    </body>
</html>
```

3.1.2 Мова CSS

Мова CSS (Cascading Style Sheets, укр. Каскадні таблиці стилів) — це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних [3].

Зазвичай використовується як засіб опису оформлення зовнішнього вигляду веб-сторінок, написаних за допомогою мови розмітки HTML, але може застосовуватися до будь-яких XML-документів. В стандартному HTML при вказанні для будь-якого елементу визначених властивостей (таких, як колір, розмір, розташування на сторінці тощо) необхідно було кожного разу описувати ці властивості, навіть якщо на одній сторінці були розміщені 10 або 110 елементів, що нічим не відрізняються один від одного. CSS діє іншим, більш зручним і економічним способом. При присвоєнні якомусь елементу визначених характеристик необхідно один раз описати цей елемент і визначити його опис, як стиль, а в подальшому просто вказувати, що елемент, який необхідно оформити відповідним чином, повинен прийняти властивості, описані раніше [3]. CSS дозволяє працювати з оформленням сторінок на набагато вищому рівні, ніж стандартний HTML.

3.1.3 Мова JavaScript

При генеруванні веб-сторінок виникає проблема, зв’язана з архітектурою “клієнт-сервер”. Сторінки можна генерувати як на стороні клієнта, так і на стороні сервера. В 1995 році спеціалісти компанії NetScape створили механізм управління сторінками на клієнтській стороні, розробивши при цьому мову програмування JavaScript.

Таким чином, JavaScript — це мова управління сценаріями перегляду гіпертекстових веб-сторінок на стороні клієнта [11]. Якщо бути більш точним, то JavaScript — це не тільки мова програмування на стороні клієнта. LiveWare, що є попередником JavaScript, є засобом підстановок на стороні сервера NetScape. Але найбільшу популярність JavaScript забезпечило програмування на стороні клієнта.

Основна ідея JavaScript полягає в можливості зміни значень атрибутів HTML-контейнерів і властивостей середовища відображення в процесі перегляду

HTML-сторінок користувачем. При цьому перезавантаження сторінки не відбувається.

На практиці це виражається в тому, що можна, наприклад, змінити колір фону сторінки або інтегроване в документ зображення, відкрите нове вікно і вивести якесь попередження.

Для створення механізму керування сторінками на клієнтській стороні запропоновано використовувати об’єктну модель документу. Суть моделі в тому, що кожен HTML-контейнер — це об’єкт, який характеризується наступною трійкою:

- властивості;
- методи;
- події.

Об’єктну модель можна уявити як спосіб зв’язку між сторінками і браузером. Об’єктна модель — це представлення об’єктів, методів, властивостей і подій, які присутні і відбуваються в програмному забезпеченні браузера, в вигляді, зручному для роботи з ними коду HTML і вихідного тексту сценарію на сторінці. Можна з її допомогою повідомляти бажання розробника браузеру і далі — користувачеві сторінки. Браузер виконає команди і відповідно змінить сторінку на екрані.

Об’єкти з однаковим набором властивостей, методів і подій об’єднуються в класи однотипний об’єктів. Самі об’єкти з’являються тільки після завантаження документа браузером або як результат роботи програми. Про це треба постійно пам'ятати, щоб не звертатися до неіснуючих об’єктів.

3.2 Розробка back-end

Розробка back-end — це набір апаратно-програмних засобів, за допомогою яких реалізована логіка роботи сайту [4]. Це те що сховане від очей користувача і відбувається поза його браузером та комп'ютером.

Ця розробка — це та область веб-розробки, яка фокусується на тому, як саме працює веб-сайт або веб-програма.

Основна роль розробника сервера полягає в тому, щоб кінцеві користувачі отримали дані або послуги, які вони вимагали, без збоїв та вчасно. Таким чином, розробка back-end вимагає широкого набору навичок та знань програмування.

Для реалізації back-end частини в даній дипломній роботі було використано мову програмування PHP та фреймворк Laravel.

3.2.1 Мова програмування — PHP

Мова PHP — мова програмування, що використовується на стороні веб-сервера для динамічної генерації HTML-сторінок. Про це каже сама її назва: PHP — Personal HyperText Processor [5].

Мова PHP — одна з небагатьох мов програмування, що була створена спеціально для розробки веб-застосунків. Тому вона включає в себе всі функції, що необхідні саме для роботи на веб-сервері, і при цьому позбавлена надмірності, що властива багатьом її конкурентам [6].

Зручна особливість PHP — те, що його команди можна застосовувати в звичайних HTML-сторінках за допомогою спеціальних тегів, які і змушують PHP-машину виконувати на сервері необхідні дії. Тобто на одній сторінці можна довільно чергувати HTML- та PHP-код.

Мова PHP не залежить від платформи. Вона прекрасно інтегрується у всі популярні веб-сервери: Apache і IIS, Zens і Netscape Enterprise Server, працює під Windows, MacOS і практично всіма UNIX-подібними системами.

Мова PHP підтримує більшість сучасних веб-протоколів: IMAP, FTP, POP, XML, SNMP та інші.

Складно знайти СКБД (систему керування базами даних), підтримка якої, не була б реалізована в PHP. Можна користуватись такими СКБД: MySQL, MS SQL, PostgreSQL, Oracle, Sybase, Interbase та інші [5].

3.2.2 PHP-фреймворк Laravel

Фреймворк Laravel визначають як full-stack-фреймворк, оскільки він може працювати зі всіма можливими частинами застосунку. До таких частин можна віднести: веб-сервіси, керування базами даних, генерація HTML, маршрутизація, кешування, аутентифікація, локалізація, інкапсуляція ресурсів тощо [7]. Таким чином вертикально інтегроване середовище розробки забезпечує кращий досвід для розробника.

Проекти Laravel мають практично однакову файлову структуру — таку, в якій кожен функціональний блок має своє призначене місце.

Нижче представлена файлова структура стандартного проекту Laravel та короткий опис функцій кожного з файлів і папок:

Таблиця 3.1. Файлова структура проекту та опис функцій кожного елементу

/app/	Складається з контролерів MVC-моделі, моделі ORM та утиліт консолі.
/bootstrap/	Містить в собі файли, які необхідні для запуску і роботи ядра Laravel.
/config/	Тільки по назві папки можна визначити, що в ній зберігаються файли для налаштування різноманітних модулів Laravel: починаючи з бази даних і кешування, закінчуючи модулем надсилання листів і файловою системою.
/database/	В цій папці зберігаються файли міграцій, сідування і генерації бази даних із моделей.

/node_modules/	Папка, в якій зберігаються JS-модулі, що підключаються, і які завантажені за допомогою утиліти nvm.
/public/	Тільки ця папка на сервері доступна зовнішньому світові. На цей каталог повинен посилатись веб-сервер. Він містить завантажувальний файл index.php, який запускає ядро Laravel. Цей каталог також може бути використаний для зберігання будь-яких загальнодоступних статичних файлів, таких як CSS, JavaScript, зображення та інші.
/resources/	В цій папці знаходяться ресурси для роботи з front-end: вихідні js- та css-файли для завантаження, файли локалізації, і файли представлень.
/routes/	Тут зберігаються файли для конфігурації модуля маршрутизації, а саме web.php.
/storage/	Ця папка призначена для зберігання файлів, що генеруються роботою Laravel-файлів: сесій, кешів і логів.
/tests/	Як видно з назви, в цій папці зберігаються файли модульних тестів.
/vendor/	Це місце для всіх посторонніх частин програми. В типовому застосунку Laravel в нього включається вихідний код, його залежності, а також плагіни, що містять додаткові попередньо упаковані функції.
.env	З цього файлу Laravel бере environment-змінні, наприклад, тут, за допомогою зміни значення змінної DEBUG, можна задати стан Laravel-програми — production чи debug.

Така структура зовсім не обов’язкова для повноцінної структури програми, але пропозиція такого способу зберігання файлів в проекті, гарантує більш-менш уніфіковану архітектуру серед всіх застосунків Laravel, що спрощує роботу з фреймворком [8].

3.3 Архітектура MVC

Фреймворк Laravel — це веб-фреймворк, який базується на архітектурі MVC.

Архітектура MVC (Model View Controller) представляє архітектуру, яку розробники приймають під час створення програми, розглядається структура проекту щодо того, як працює потік даних.

Це програмна архітектура, яка відокремлює логіку програми від інтерфейсу користувача. Це робиться шляхом поділу програми на три частини: модель, вигляд, контролер.

Модель керує фундаментальною поведінкою та даними програми. Вона може реагувати на запити на інформацію, на інструкції щодо зміни стану своєї інформації і навіть сповіщати спостерігачів у системах, керованих подіями, коли інформація змінюється. Це може бути база даних або будь-яка кількість структур даних або систем зберігання. Тобто, це дані та управління даними програми.

Вигляд ефективно забезпечує елемент інтерфейсу користувача програми. Дані з моделі відображатимуться у формі, яка підходить для інтерфейсу користувача.

Контролер отримує введені користувачем дані та здійснює виклики модельних об'єктів та подання для виконання відповідних дій.

Загалом, ці три компоненти працюють разом, щоб створити три основні компоненти MVC.

Структура MVC зображена на рисунку 3.3.1.

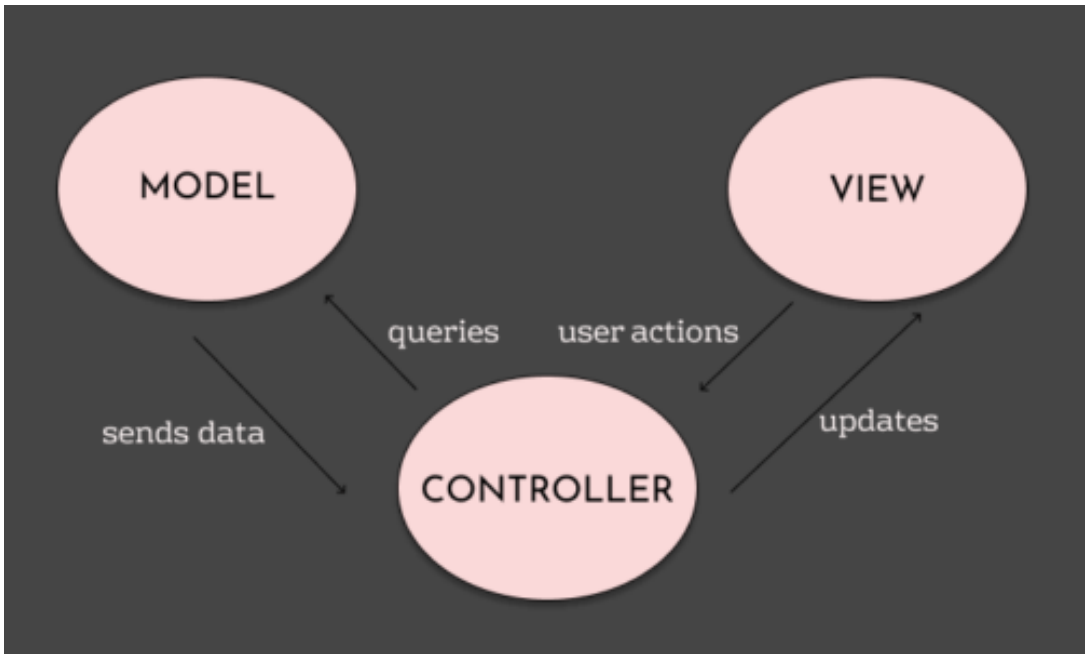

Рисунок 3.3.1 — Структура архітектури MVC

Модель - це представлення реального екземпляра або об’єкта в базі коду. Вид представляє інтерфейс, за допомогою якого користувач взаємодіє з додатком. Коли користувач виконує дію, контролер обробляє дію та при необхідності оновлює Модель.

3.4 Бази даних MySQL

Бази даних MySQL являє собою множину елементів, структуровану з метою одночасного виконання запитів декількох користувачів в конкретний момент часу. Тобто, база даних являє собою сукупність взаємопов’язаних даних і містить інформацію про різноманітні сутності однієї предметної області (існуючих об’єктів або процесів).

Іншими словами, база даних MySQL — це набір таблиць (файлів), в яких зберігаються дані. Якщо база даних складається з ряду зв’язаних між собою таблиць, то вона називається реляційною. В реляційних базах даних використовуються ключі, які визначають запис.

Робота з базою даних здійснювалась у веб-додатку phpMyAdmin. Це безкоштовний програмний засіб, написаний на PHP та призначений для управління

адмініструванням MySQL через Інтернет. Веб-додаток phpMyAdmin підтримує широкий спектр операцій на MySQL та MariaDB. Часто використовувані операції (управління базами даних, таблицями, стовпцями, відносинами, індексами, користувачами, дозволами тощо) можна виконувати через користувальницький інтерфейс, тоді як можливість безпосереднього виконання будь-якого оператора SQL все ще існує [12].

3.5 Середовище розробки

Програмне забезпечення JetBrains PhpStorm являє собою спеціалізований засіб веб-розробки, орієнтований на веб-застосунки та інші види програм, які можна створювати за допомогою мови PHP і з використанням HTML, CSS, JavaScript. Рішення PhpStorm пропонує функції автоматичного завершення мовних конструкцій PHP в коді, інспектування коду, різні алгоритми рефакторингу та швидку навігацію по коду [9].

Реалізований в PhpStorm графічний PHP-налагоджувач підтримує умовні точки зупинки, відслідковування значень і автоматизований вхід в налаштування окремих процедур. Для тестування програм підтримується каркас тестових модулей PhpUnit та графічний інтерфейс для запуску тестів. При редагуванні коду виділяються конструкції синтаксису, здійснюється розширене форматування конфігурації, виявлення помилок в режимі реального часу та завершення коду.

Завдяки застосуванню PhpUnit-тестів можна швидко переглядати результати генерації коду окремих блоків або всієї програми. Якщо тест був проведений неуспішно, продукт дозволяє переглядати окремі кодові рядки, в яких була виявлена помилка [9].

Сотні інспекцій відповідальні за верифікацію коду, аналізуючи проект в цілому і під час розробки. Підтримка PHPDoc, code (re)arranger, форматера коду з конфігурацією стилю коду та інші можливості допомагають розробникам створювати легко-підтримуваний код.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Проект було створено, слідуючи основним етапам розробки програмного забезпечення [10]:

1. Аналіз вимог
2. Проектування ПЗ
3. Розробка ПЗ
4. Тестування ПЗ
5. Впровадження та супровід

В цьому розділі буде детально описано два етапи: проектування та розробку, які є наймасштабнішими та найбільш важливими при роботі над програмою.

4.1 Аналіз вимог

Аналіз вимог — це частина процесу розробки програмного забезпечення, що включає в себе збір вимог до програмного продукту (ПО), їх систематизації, виявленню взаємозв’язків, а також документацію.

Тобто, результатом даного етапу є наявність технічного завдання (ТЗ). Скорочений варіант цього документу наведено в даному розділі.

4.1.1 Глосарій

Таблиця 4.1. Терміни, що є ключовими в технічному завданні та їх опис

Сайт	Інформаційна система, що надає доступ до свого вмісту користувачам Інтернет в вигляді впорядкованого набору взаємопов’язаних HTML-сторінок
HTML-сторінка	Головний носій інформації в мережі Інтернет. Це файл, сформований особливим способом, що може переглядатися

	браузером як єдине ціле
Гіперпосилання	Активний елемент HTML-сторінки, що задається спеціальним HTML-тегом. Це виділений фрагмент тексту чи зображення, що дозволяє завантажити іншу сторінку чи виконати визначену дію
HTML-тег	Керуючі коди, за допомогою яких здійснюється форматування HTML-сторінки
Браузер	Клієнтська програма, що надається третіми особами і дозволяє переглядати вміст HTML-сторінок
HTML-форма	Частина HTML-сторінки, що призначена для взаємодії з користувачем сайту. Являє собою набір елементів (текстових полів, випадаючих списків тощо), за допомогою яких користувач може ввести будь-яку інформацію і відправити її для обробки на сервері.
Поле форми	Структурний елемент, що містить однотипну інформацію, наприклад, текст, дату, числові значення тощо.
Адміністратор	Особа, що здійснює роботу над сайтом та користується його функціоналом.

Інша технічна термінологія розуміється у відповідності до діючих стандартів і рекомендацій міжнародних органів, що є відповідальними за питання стандартизації в мережі Інтернет.

4.1.2 Предмет розробки

Предметом розробки є Інтернет-сайт, що слугує сервісом автопостингу в соціальні мережі для робітників кафедри (факультету).

Призначення сайту:

- надання авторизованого доступу до сервісу;
- надання інформації про існуючих користувачів системи та історії опублікованих постів за допомогою даного сервісу;
- надання можливості формування та подальшої публікації посту в обрані адміністратором соціальні мережі.

4.1.3 Вимоги до графічного дизайну

При розробці сайту повинні бути використані світлі стилі. Дизайн повинен бути витриманий в строгих та м’яких тонах, використовуючи синьо-блакитні відтінки. Дизайн сайту повинен бути виконаний за допомогою мови HTML та CSS.

Сайт повинен коректно відображатись в браузерах Safari 14.0.1, Google Chrome 91.0.4472.77 та Opera 75.0.3969.218.

Необхідно створити структуру сайту, що містить наступні елементи:

- header. В даному блоці необхідно розмістити назву сайту, кнопку “Ввійти” для неавторизованого користувача і кнопку “Вийти” для вже авторизованого;
- блок відображення всіх адміністраторів сайту з виділеним авторизованим користувачем та з посиланням на сторінку з реєстрацією нового адміністратора;
- блок відображення історії раніше опублікованих постів;
- гіперпосилання на сторінку формування поста.

На рисунку 4.1.3 представлена графічна схема шаблону головної сторінки сайту.

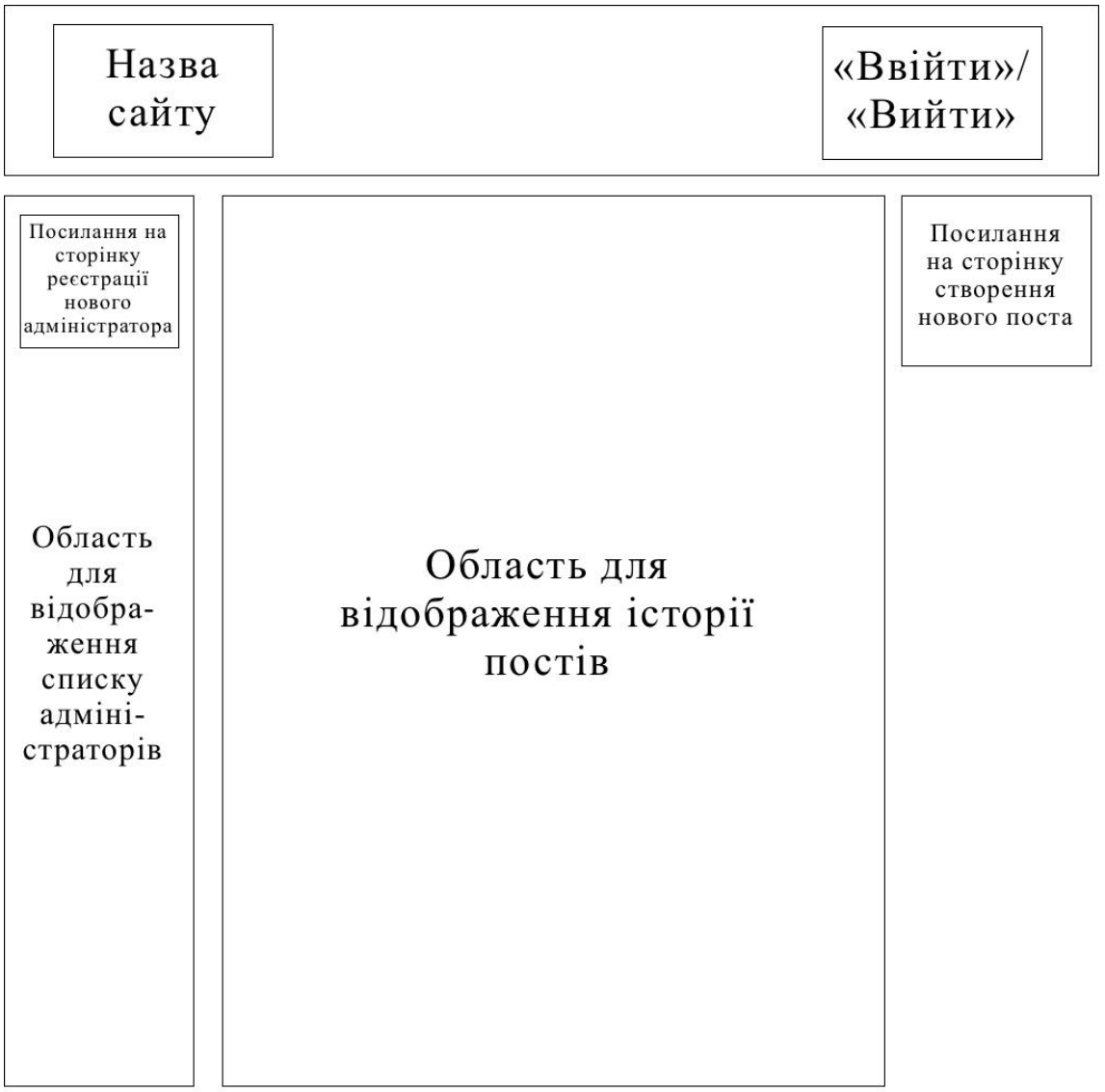

Рисунок 4.1.3 — Графічна схема шаблону головної сторінки сайту

Доступ до сторінок сайту повинен бути наданий тільки авторизованому користувачеві з головної сторінки.

На першій сторінці повинна бути тільки форма авторизації. Можливість реєстрації тут відсутня.

4.1.4 Вимоги до функціональності сайту

Єдиною мовою сайту є українська. Оскільки, сервіс призначений для публікації новин кафедри чи факультету, які будуть формуватись державною мовою, необхідності забезпечення можливості надання інформації користувачеві іншими мовами немає.

Сайт повинен дозволяти користувачам:

- виконувати вхід на сайт як зареєстрований користувач для можливості перегляду інформації та доступу до всього функціоналу сайту;
- здійснювати навігацію по сайту (перехід між сторінками);
- зареєструвати нового адміністратора;
- сформувати пост для подальшої публікації в соціальних мережах
- вивантажувати при необхідності файли-зображення на сервер для їх подальшої публікації;
- вийти з сервісу.

4.1.5 Вимоги до вмісту сайту

Необхідно створити наступні сторінки сайту:

- початкова сторінка сайту з формою авторизації;
- головна сторінка авторизованого користувача;
- сторінка з формою реєстрації нового адміністратора;
- сторінка з формою для створення нового поста.

Початкова сторінка повинна містити тільки форму авторизації по електронній пошті та паролю користувача.

Головна сторінка повинна відображати список адміністраторів, таблицю з попередньо опублікованими постами за допомогою даного сервісу та посилання на сторінку на формування посту. Детальніший опис цієї сторінки наведений в розділі 4.1.3.

Сторінка з формою реєстрації нового адміністратора повинна містити поля для вводу інформації про створюваного користувача, паролю та його підтвердження з обов’язковою перевіркою даних на валідність.

Сторінка з формою для створення нового поста також повинна містити форму для введення такої інформації:

- тіло поста;
- опис зображення;
- посилання на зображення або прикріплення файлу з пристрою користувача;
- відмітки соціальної мережі, в якій необхідно зробити публікацію.

4.2 Проектування програмного забезпечення

Проектування програмного забезпечення — процес визначення інтерфейсів, компонентів, архітектури системи, її інших характеристик та кінцевого результату.

Для цього було використано мову графічного опису для об’єктного моделювання UML.

Мова UML застосовується для проектування програмного забезпечення, моделювання бізнес-процесів, системного проектування і відображення організаційних структур. Ця мова була створена для документування, визначення, візуалізації та проектування програмних систем. Вона не є мовою програмування, але на основі UML-моделей можлива генерація коду.

Для проектування програми по реплікації та синхронізації новин інформаційних каналів було використано діаграму прецедентів та діаграму розміщення.

4.1.1 Діаграма прецедентів

Діаграма прецедентів — діаграма на якій зображено зв’язок між акторами та варіантами використання функцій програми. Основна задача цієї діаграми — надати можливість розробнику та замовнику обговорювати можливості програми на одному рівні.

Для даного проекту діаграма прецедентів була створена за допомогою сервісу draw.io.

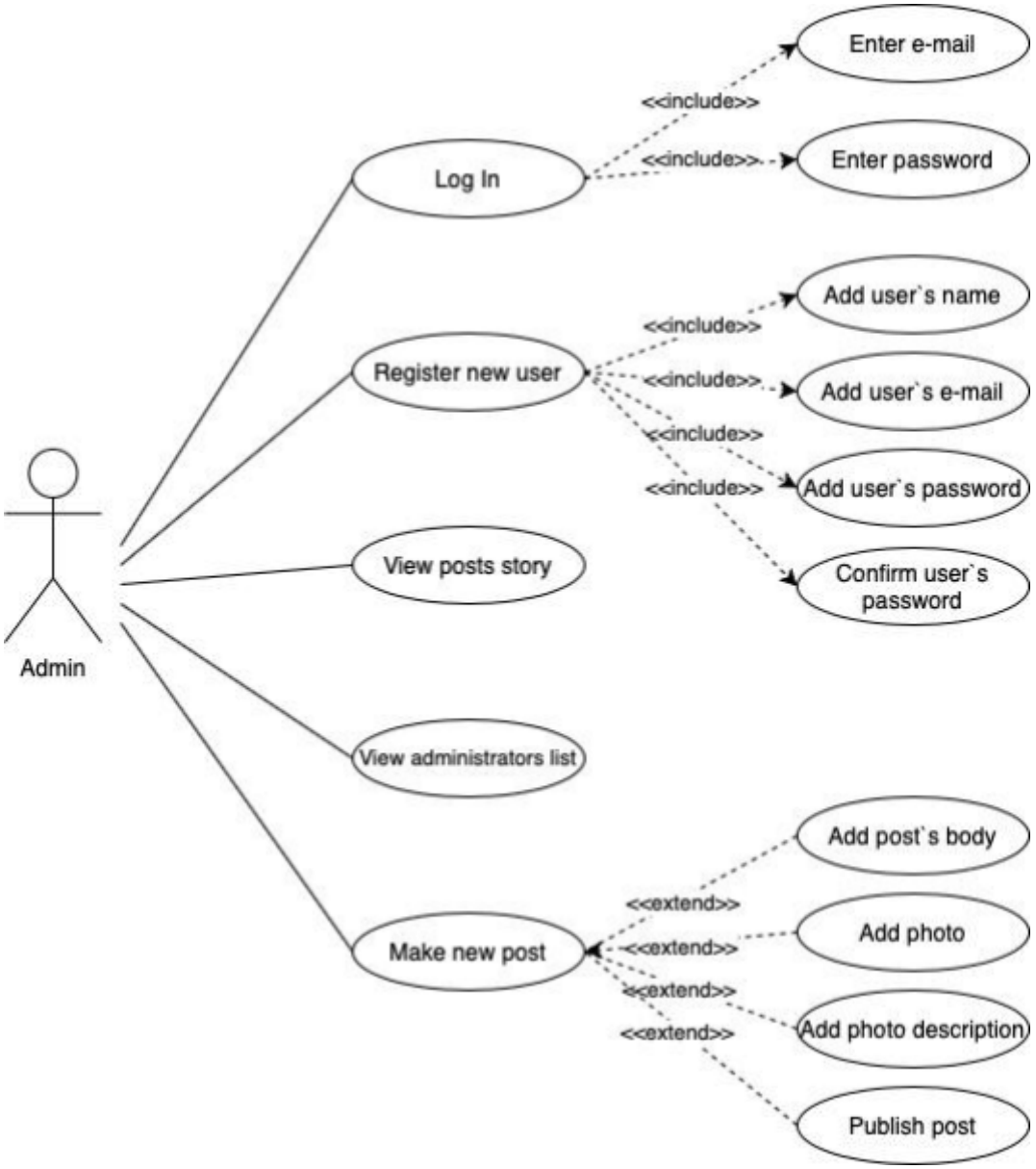

Рисунок 4.1.1 — Діаграма прецедентів

Бачимо такі основні функції користувача: вхід в систему, реєстрація нового адміністратора, перегляд історії постів та списку адміністраторів, створення нового посту.

Для прецедента “Вхід в систему” є два прецеденти “Ввід e-mail” та “Ввід паролю”, які є складовими основного компоненту. Відношення між прецедентами <<include>> означає, що прецедент, що включається, наприклад, “Ввід e-mail” повинен бути обов’язковим для прецеденту, що доповнюється, “Вхід в систему”.

Тобто для того, щоб ввійти в систему необхідно обов’язково ввести електронну та пароль.

За цим же принципом реалізується реєстрація нового користувача. Для того, щоб додати адміністратора, необхідно обов’язково ввести його ім’я, електронну пошту та пароль, підтвердивши його.

Відношення <<extend>> відображає можливе приєднання одного варіанту використання до іншого. Також для приєднуваного прецеденту можна вказати умови його виконання.

Розглянемо діаграму та прецедент “Додати новий пост”. При додаванні поста можна вказати тіло поста, додати фотографію, описати зображення, але ці пункти не є обов’язковими. Адже можна сформувати пост, маючи тільки фотографію без супроводжуючого тексту, або навпаки. А публікація цього посту відбудеться лише за виконання умов: вказані дані та обрана соціальна мережа. Тому тут використовується відношення розширення.

4.1.2 Діаграма розміщення

Діаграма розміщення служить для моделювання працюючих вузлів (апаратних засобів) і артефактів, розгорнутих на них.

Вузли представляються, як прямокутні паралелепіпеди із артефактами, розташованими в них та зображеними у вигляді прямокутників. Вузли можуть мати підвузли, що представляються, як вкладені прямокутні паралелепіпеди. Один вузол діаграми розміщення може представляти безліч фізичних вузлів, таких, як кластер серверів баз даних.

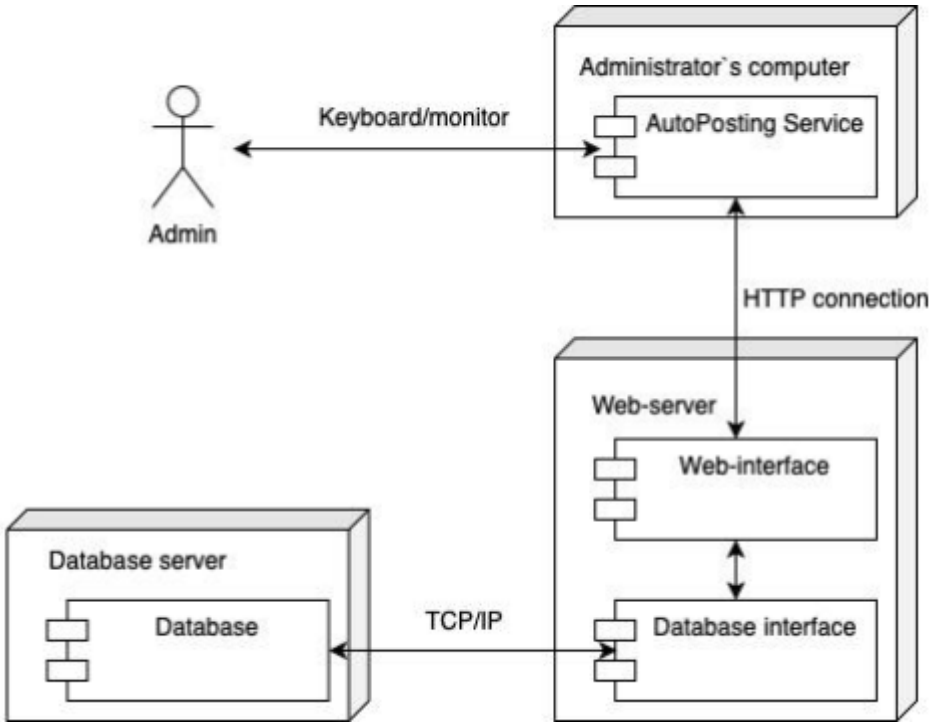

Рисунок 4.1.2 — Діаграма розміщення

Діаграму розміщення системи можна описати наступним чином. Користувач-адміністратор взаємодіє із системою шляхом використання сервісу автопостингу. Програма відправляє запити та отримує відповіді від web-сервера. Сервер надає користувацький інтерфейс, а також – використовує інтерфейс бази даних для доступу до даних, їх зміни, видалення, модифікації.

4.2 Розробка програмного забезпечення

Для створення програмного коду було використано PHP-фреймворк Laravel.

Для того, щоб створити новий проект за допомогою цього фреймворку, необхідно, щоб на комп’ютер вже були встановлені PHP та Composer (менеджер пакетів прикладного рівня). Потім необхідно встановити програму Laravel Installer як глобальну залежність Composer, використовуючи при цьому командний рядок.

Після того, як у обраній папці з’явиться нова директорія з назвою проекту, можна починати роботу над ним.

Робота над сайтом починається з маршрутизації. Маршрутизація або роутинг — процес, який відповідає за визначення обробника для конкретної запитуваної сторінки. Кожна url-адреса, наприклад, http://researchwork.online/newpost, являє собою конкретний маршрут. В Laravel проектах їхня обробка проходить у файлі routes/web.php. Приклад даного файлу для сервісу автопостингу наведено на рисунку 4.2.1.

```
Route::get( uri: '/', function () {
    return view( view: 'auth/login');
});
Route::get( uri: '/privacy', function () {
    return view( view: 'privacy');
});

Route::get( uri: '/newuser', function () {
    return view( view: 'auth/register');
});

Route::get( uri: '/newpost', function () {
    return view( view: 'newpost');
});

Route::post( uri: '/storenewuser', [App\Http\Controllers\NewUserController::class, 'registerUser']);

Route::post( uri: '/storenewpost', [App\Http\Controllers\NewPostController::class, 'makeNewPost']);


Auth::routes();

Route::get( uri: '/home', [App\Http\Controllers\HomeController::class, 'index'])->name( name: 'home');
```

Рисунок 4.2.1 — Скриншот файлу web.php

Бачимо перелік можливих маршрутів — це ті дані, що знаходяться після домену. Наприклад, researchwork.online/ — домен, а newpost — маршрут.

Після визначення маршруту системою відбувається або виклик контролера, або вивід відповідної сторінки на екран користувача. При умові, що потрібна попередня обробка даних, викликається контролер.

Наприклад, при отриманні такої адреси researchwork.online/home викличеться наступна функція контролера app/Http/Controllers/HomeController.php index().

```php
public function index()
{
    $posts = Post::all();

    $users = User::all();

    $currentUser = auth()->user();

    return view( view: 'home')->with(['posts' => $posts])->with(['users' => $users])->with(['currentUser' => $currentUser]);
}
```

Рисунок 4.2.2 — Скриншот функції index() файлу HomeController.php

Тут виконуються наступні дії: отримуємо список всіх постів з бази даних, список всіх користувачів з бази даних, дані авторизованого користувача та передаємо цю інформацію у файл з HTML-кодом, де вона виведеться у зрозумілому для користувача вигляді на відповідну сторінку.

Вся інформація з бази даних обробляється за допомогою моделей, які є звичайним прикладом класу. Властивості цього класу повинні співпадати зі стовпчиками відповідної таблиці бази даних. Зв’язок моделі з таблицею відбувається за допомогою міграцій.

4.2.1 Огляд алгоритму реалізації публікації в соціальну мережу Telegram

Для публікації контенту в соціальну мережу Telegram використовується поняття “канал”. Канал в Telegram — це одновекторний спосіб передачі інформації підписникам. Канал нагадує спільноту в соціальній мережі, в якій право на публікацію має тільки адміністратор. Якщо в кафедри чи факультету ще не існує даного каналу зв’язку, його необхідно створити.

Для тестування продукту було створено власний канал в Telegram, аналогією якому в майбутньому може стати канал кафедри.

Наступним обов’язковим кроком є створення боту Telegram. Бот — це спеціальна програма, що виконує різноманітні функції. Бот імітує дії живого користувача. Для його створення було використано бот BotFather. Це найбільш простий спосіб для реєстрації, налаштування та управління іншими telegram-ботами. Робота з ним проста і не вимагає специфічних навиків. Єдиною умовою реєстрації нового бота є його унікальний username.

Взаємодія з BotFather здійснюється за допомогою простих команд. Для того, щоб створити нового бота, необхідно відправити в чат команду /newbot і слідувати наступним інструкціям:

1. Придумати ім’я бота, яке буде відображатись в чатах і контактах;
2. Придумати username. Це ім’я повинно бути унікальним і мати закінчення “bot”. Можна використовувати букви латинського алфавіту, цифри та символ підкреслювання.
3. Потім у відповідь користувач повинен отримати повідомлення з токеном. Даний токен необхідний для роботи з Bot API за допомогою http-протоколу.

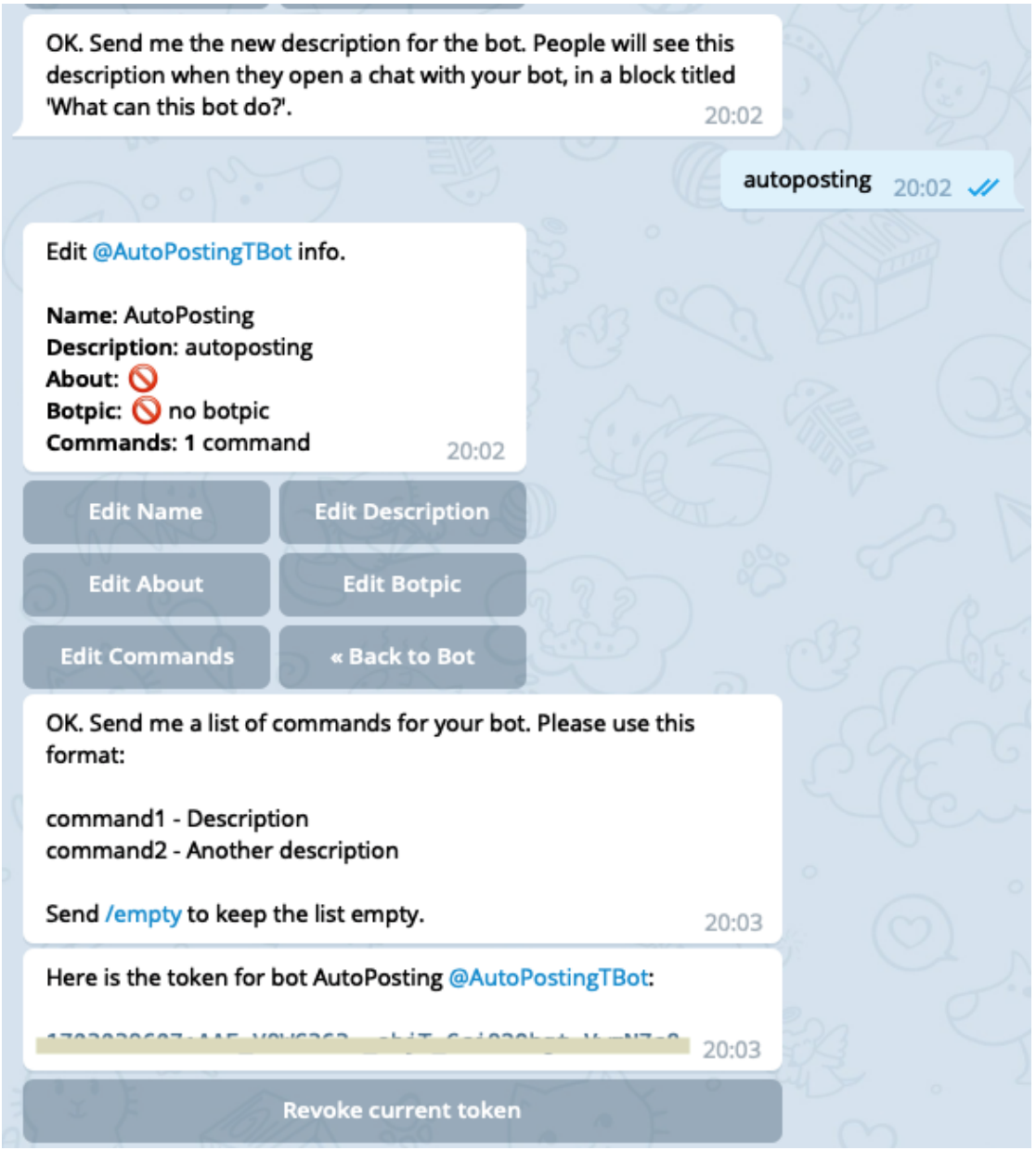

Рисунок 4.2.1 — Процес створення нового бота за допомогою BotFather

Оскільки, публікувати в канал може тільки користувач з правами адміністратора, новоствореного бота необхідно додати в список адміністраторів тестового каналу.

Для роботи з даним ботом в коді наступним кроком необхідно встановити бібліотеку Telegram API.

На цьому підготовка до написання самого коду закінчується.

```
28      if(!empty($_POST['telegram'])) {
29          require_once("vendor/autoload.php");
30          $isTelegram = true;
31          if (true) {
32              error_reporting( level: E_ALL & ~(E_NOTICE | E_USER_NOTICE | E_DEPRECATED));
33              ini_set( varname: 'display_errors',  newvalue: 1);
34          }
35
36          $token = "1703039607:AAF_V9WC363-_objT_Cci92Obgt-VwmN7s8";
37          $grupa = '@autopostingt';
38          $bot = new \TelegramBot\Api\Client($token,  trackerToken: null);
39
40          if($text){
41              $bot->sendMessage($grupa, $text);
42          }
43
44
45          if ($pic) {
46              $bot->sendPhoto($grupa, $pic, $textPhoto);
47          }
48          $network='Telegram';
49
50      }
```

Рисунок 4.2.2 — Фрагмент коду, що опрацьовує публікацію в Telegram

На рисунку 4.2.2 зображено фрагмент коду, основні моменти якого це:

1. Рядки 36, 38 — створення змінної бота;
2. Рядок 36 — фіксування того токена, що був отриманий раніше за допомогою BotFather;
3. Рядок 37 — фіксування id каналу, в який необхідно зробити публікацію;
4. Рядки 40-42 — якщо користувачем було вказано основний текст поста, тобто його тіло, то відбувається надсилання вмісту цієї змінної у визначений канал;
5. Рядки 45-47 — якщо користувачем було додано фотографію та супроводжуючий текст, то відбувається надсилання зображення з його описом у визначений канал.

4.2.2 Огляд алгоритму реалізації публікації в соціальну мережу Facebook

Налаштування публікації в Facebook можна поділити на три етапи:

1. Створення додатку Facebook;

2. Отримання токена;

3. Налаштування публікацій на сторінку.

Проходження даного процесу можливе тільки зареєстрованим у Facebook користувачам. Тому якщо облікового запису Facebook ще немає, його необхідно створити.

Створення додатку відбувається за допомогою Facebook for Developers. Це інструмент, що наданий для того, щоб допомогти розробникам інтегрувати служби Facebook на сторонніх платформах.

При створенні додатку необхідно вказати його назву, якою у випадку дипломної роботи стала “AutoPosting”. Після всіх налаштувань, стане доступним App ID та App Secret, що є найголовнішим. Ці значення являють собою ідентифікатори додатку в Facebook.

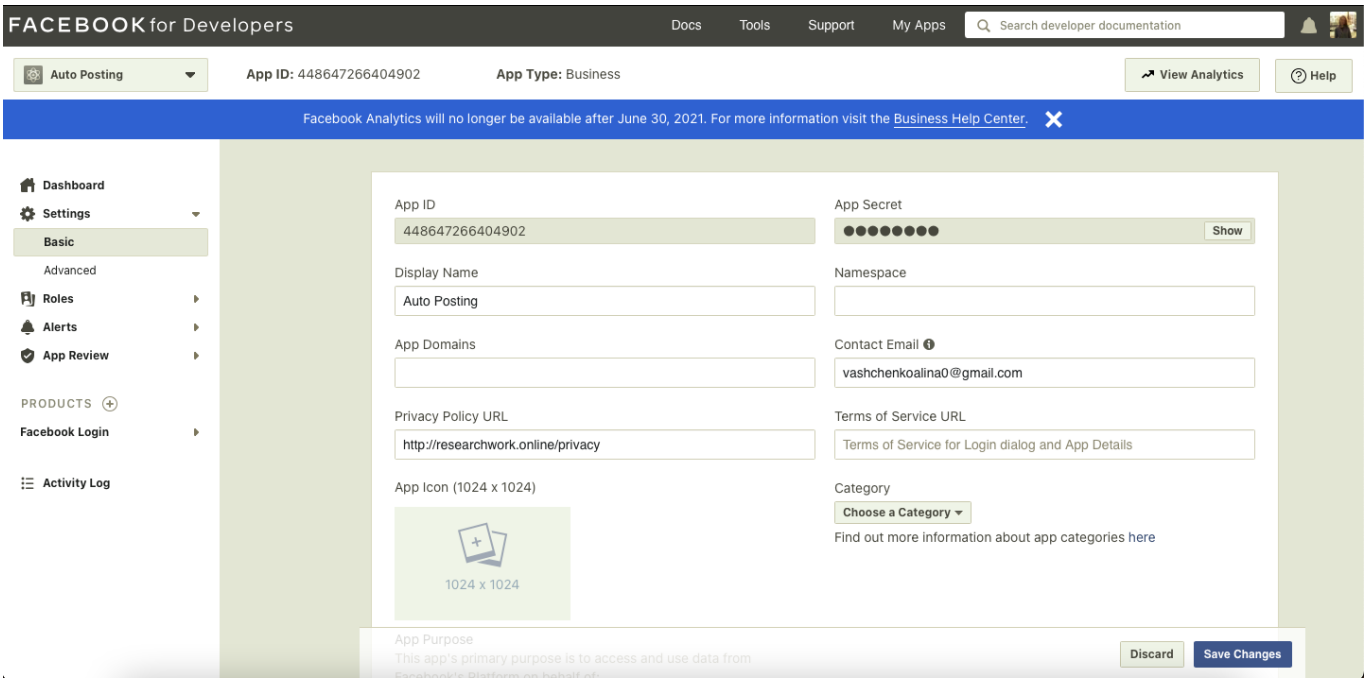

Рисунок 4.2.3 — Інтерфейс платформи Facebook for Developers

Наступним кроком є створення сторінки, на яку буде відбуватись публікація. Не потрібно плутати поняття особистого профілю і сторінки. Сторінки дають

брендам, компаніям, організаціям та публічним особам можливість ділитися новинами і взаємодіяти з користувачами. Так само як і профілі, сторінки можна заповнити подіями, новинами тощо. Люди, які вподобали сторінку або стежать за нею, можуть переглядати оновлення про неї в стрічці новин.

Далі необхідним є отримати токен для того, щоб здійснити роботу з Facebook API в коді. Його можна отримати за допомогою Graph API Explorer. Інтерфейс даної платформи зображено на рисунку 4.2.4.

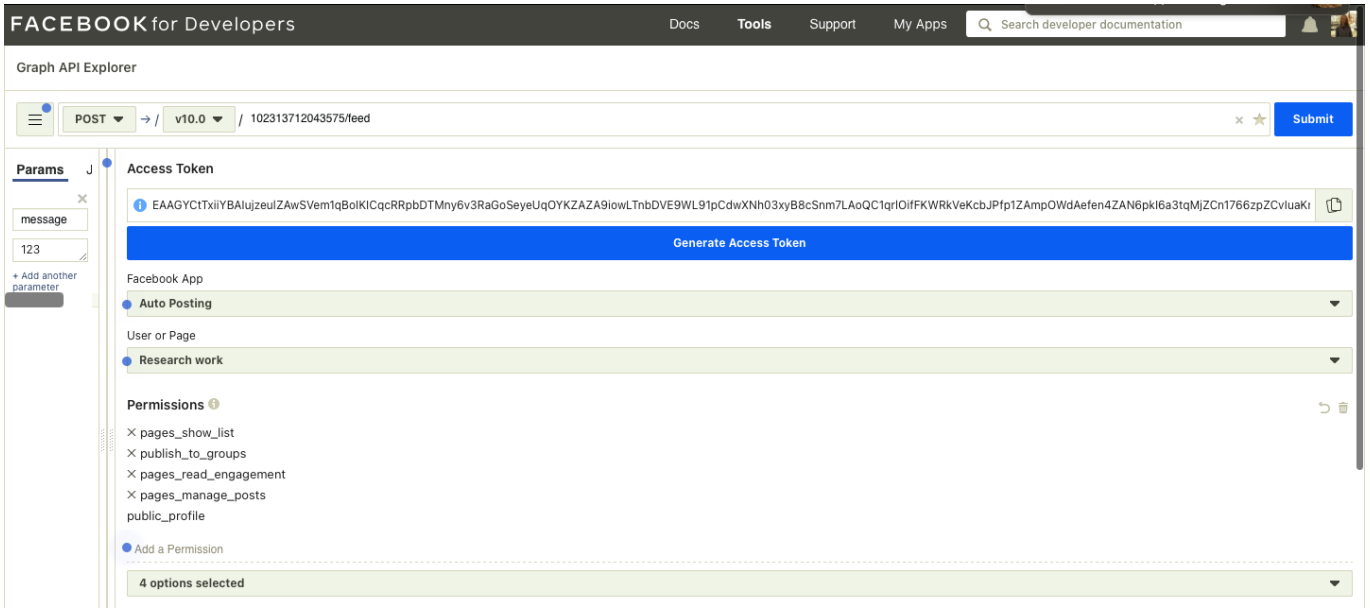

Рисунок 4.2.4 — Інтерфейс Graph API Explorer

Бачимо сформований токен для відповідного додатку та відповідної сторінки. Також в розділі Permissions можна побачити ті дії, які дозволено робити розробнику. Для того, щоб отримати ширший список даних дозволів необхідно пройти перевірку програми працівниками Facebook. Оскільки для реалізації завдання даних дозволів було достатньо, перевірка не проводилась.

Тепер можна переходити до налаштувань самої публікації в коді проекту. На рисунку 4.2.5 наведено фрагмент коду, що відповідає за це завдання.

```
55      $app_id = "448647266404902";
56      $app_secret = '[redacted]';
57      $token = "EAAGYCtTxiiYBACTziZA1k0rjUK9GxUFDsA5aN5kQiuAUTGxjfTSmH4Wjs0PJK2CyNMOYZBcdCJDMTRqUGOnTUNh1MsRuVpwMBnoHwaIaE0
58      $fb = new \Facebook\Facebook([
59          'app_id' => $app_id,
60          'app_secret' => $app_secret,
61          'default_graph_version' => 'v10.0',
62      ]);
63
64      if($pic){
65          $data = [
66              'message' => $text,
67              //'source' => $fb->fileToUpload($pic),
68              'link' => $pic
69          ];
70      }
71      else{
72          $data = array('message' => $text);
73      }
74
75      try {
76
77          $response = $fb->post("/102313712043575/feed", $data, $token);
78
79      } catch(\Facebook\Exceptions\FacebookResponseException $e) {
80          echo 'Graph returned an error: ' . $e->getMessage();
81          exit;
82      } catch(\Facebook\Exceptions\FacebookSDKException $e) {
83          echo 'Facebook SDK returned an error: ' . $e->getMessage();
84          exit;
85      }
86
87      $graphNode = $response->getGraphNode();
```

Рисунок 4.2.5 — Фрагмент коду, що опрацьовує публікацію в Facebook

Основні моменти даного фрагменту коду це:

1. Рядки 55-56, 58-62 — створення об’єкту, що містить дані про додаток Facebook;
2. Рядок 57 — фіксування того токена, що був отриманий раніше за допомогою Graph API Explorer;
3. Рядок 77 — публікація посту зі сформованими змінними у стрічку відповідної сторінки.

4.2.3 База даних проекту

Робота з базою даних відбувається за допомогою веб-додатка phpMyAdmin. Сформовано дві таблиці:

- таблиця користувачів;
- таблиця опублікованих за допомогою сервісу постів.

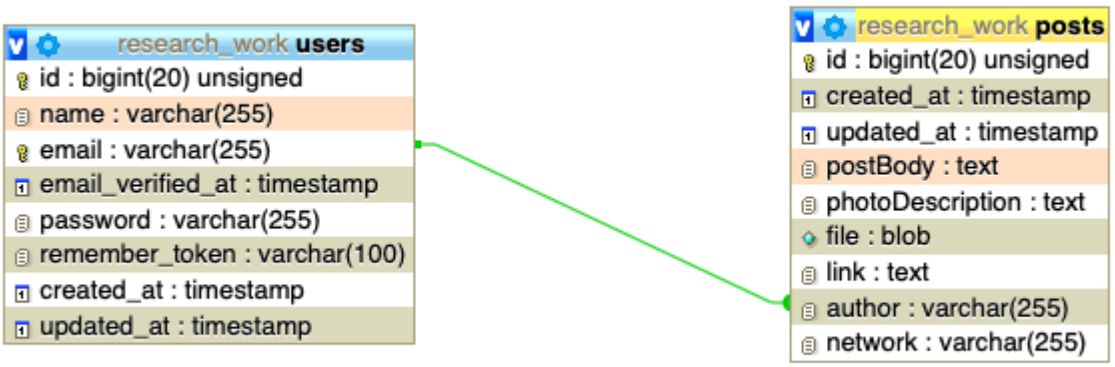

Рисунок 4.2.6 — Схема бази даних проекту

Бачимо таблицю Користувачі, де є його ідентифікатор, ім’я, електронна пошта, пароль тощо. Пароль юзера в таблиці не зберігається в тому форматі, яким був заданий при реєстрації. Дана змінна шифрується за допомогою Laravel Hash, що забезпечує безпечне хешування Bcrypt та Argon2 для зберігання паролів користувачів.Bcrypt за замовчуванням буде використовуватися для реєстрації та автентифікації.

Bcrypt — чудовий вибір для хешування паролів, оскільки його "робочий коефіцієнт" можна регулювати, а це означає, що час, необхідний для генерації хешу, може бути збільшений із збільшенням потужності обладнання. При хешуванні паролів повільний процес — це добре. Чим довше алгоритм займається хешуванням пароля, тим більше часу знадобиться для злому облікового запису.

Таблиця Пости містить ідентифікатор, час створення та оновлення запису, тіло поста, опис зображення, файл-зображення, посилання на нього з мережі, автора поста та соціальну мережу, де він був опублікований.

Також бачимо зв’язок між таблицями типу “Один до багатьох”, адже один користувач може формувати безліч постів.

4.2.4 Домен та хостинг

Для налаштування додатку Facebook необхідно вказувати посилання з мережі на даний проект. Тому було вирішено придбати домен та хостинг для даної дипломної роботи.

Доменні імена та послуги веб-хостингу є важливими елементами веб-сайту. По суті, доменне ім’я служить адресою сайту, тоді як послуга хостингу надає простір та ресурси для запуску та роботи веб-сайту.

Оскільки обидва елементи працюють рука об руку, перш ніж створювати веб-сайт, важливо зрозуміти відмінності між ними.

Доменне ім’я відноситься до URL-адреси, яку люди вводять в адресному рядку веб-браузера для доступу до вашого веб-сайту. Іншими словами, доменні імена пропонують зручний спосіб доступу людей до веб-сайтів.

Без цього їм довелося б використовувати цифрову мітку, присвоєну кожному веб-сайту та серверу в Інтернеті, також відому як IP-адреса [13]

Щоб отримати доменне ім’я для свого веб-сайту, спочатку його потрібно зареєструвати. Реєстрація домену — це процес резервування імені в Інтернеті на певний період.

Для даного проекту було обрано наступну адресу: researchwork.online.

Послуги веб-хостингу дозволяють публікувати веб-сайт в Інтернеті. Іншими словами, провайдери хостингу орендують частину свого веб-сервера для зберігання файлів та даних вашого веб-сайту.

Щоразу, коли хтось вводить доменне ім’я веб-сайту, постачальник веб-хостингу нестиме відповідальність за надсилання його вмісту відвідувачеві [14].

Веб-хости також надають зручні для початківців інструменти для всіх користувачів, так що можна керувати веб-сайтом практично без технічних навичок. Крім того, хостинг-провайдер зазвичай пропонує підтримку клієнтів, обслуговування серверів та розробників веб-сайтів, щоб допомогти користувачам створити та підтримувати свій сайт.

Окрім зберігання файлів веб-сайтів, веб-хост захищає сервер та файли вашого веб-сайту від зловмисних атак.

Домен і хостинг працюють разом — можна уявити, що і IP-адреса, і доменне ім’я виконують роль адреси веб-сайту, а веб-хостинг - це власне дім. Без веб-хостингу сайт не може існувати в Інтернеті.

Отже, потрібно синхронізувати і направити доменне ім’я на постачальника веб-хостингу після його придбання.

Для даної дипломної роботи було обрано хостинг з можливістю вміщати 25 ГБ об’єму даних, що проходять через сайт.

В процесі роботи проблем з даними питаннями не виникало, а розробка продукту відбувалась в середовищі розробки PhpStorm за допомогою Remote Host.

4.3 Тестування програмного забезпечення

Тестування даного продукту відбувалось методом QA тестування.

Тестування забезпечення якості (QA) - це процес забезпечення того, щоб продукт був найвищої якості для клієнтів. Забезпечення якості - це методи, які використовуються для запобігання проблемам із програмним продуктом або послугою та забезпечення чудового досвіду роботи для клієнтів.

Раніше тестування на забезпечення якості, як правило, впроваджувалось відносно пізно у циклі розробки. Наприклад, у Waterfall програмне забезпечення виробляється та передається команді QA, яка пропонує відгуки, а потім відправляє програмне забезпечення назад у розробку, щоб вони могли вирішити проблеми.

В середовищі Agile всі, включаючи команду з забезпечення якості, працюють спільно, щоб постійно вдосконалювати. Проте, оскільки розробники, оператори та тестувальники несуть спільну відповідальність за те, щоб забезпечити якісний кінцевий продукт, QA часто намагається знайти своє місце [15].

Тестування забезпечення якості - це забезпечення якості (QA) або процес тестування якості, який гарантує, що організація надає найкращі продукти чи послуги.

Контроль якості спрямований на отримання послідовних результатів за допомогою набору стандартизованих процедур, що означає, що організації також повинні переконатися, що їх процеси для досягнення бажаних результатів самі досягають певних показників якості.

Коротше кажучи, можна сказати, що перевірка якості включає всю діяльність, яка зосереджується на впровадженні стандартів та процедур, пов’язаних із забезпеченням відповідності програмного забезпечення певному набору вимог, перш ніж воно буде оприлюднене для широкого загалу.

Єдиний спосіб мінімізувати ризик дефектів, одночасно максимізуючи досвід роботи кінцевого користувача, - це включити тестування програмного забезпечення та забезпечення якості протягом усього процесу розробки.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА

Для використання сервісу необхідно мати доступ в Інтернет та завантажений браузер для відображення сайту.

Після переходу за посиланням http://researchwork.online користувач потрапляє на сторінку авторизації, де необхідно ввести електронну пошту та пароль, що попередньо були додані в базу даних. Для того, щоб уникнути випадкового потрапляння на сервіс та доступу до його функцій, можливість реєстрації нового користувача на початковій сторінці відсутня. Перший адміністратор системи додається особисто розробником, всі подальші потенційні користувачі будуть додаватись вже наявним адміністратором. При умові, що електронна пошта чи пароль не співпадають з записами в базі даних, виведеться повідомлення про помилку.

Якщо користувач успішно авторизувався він потрапляє на головну сторінку сервісу.

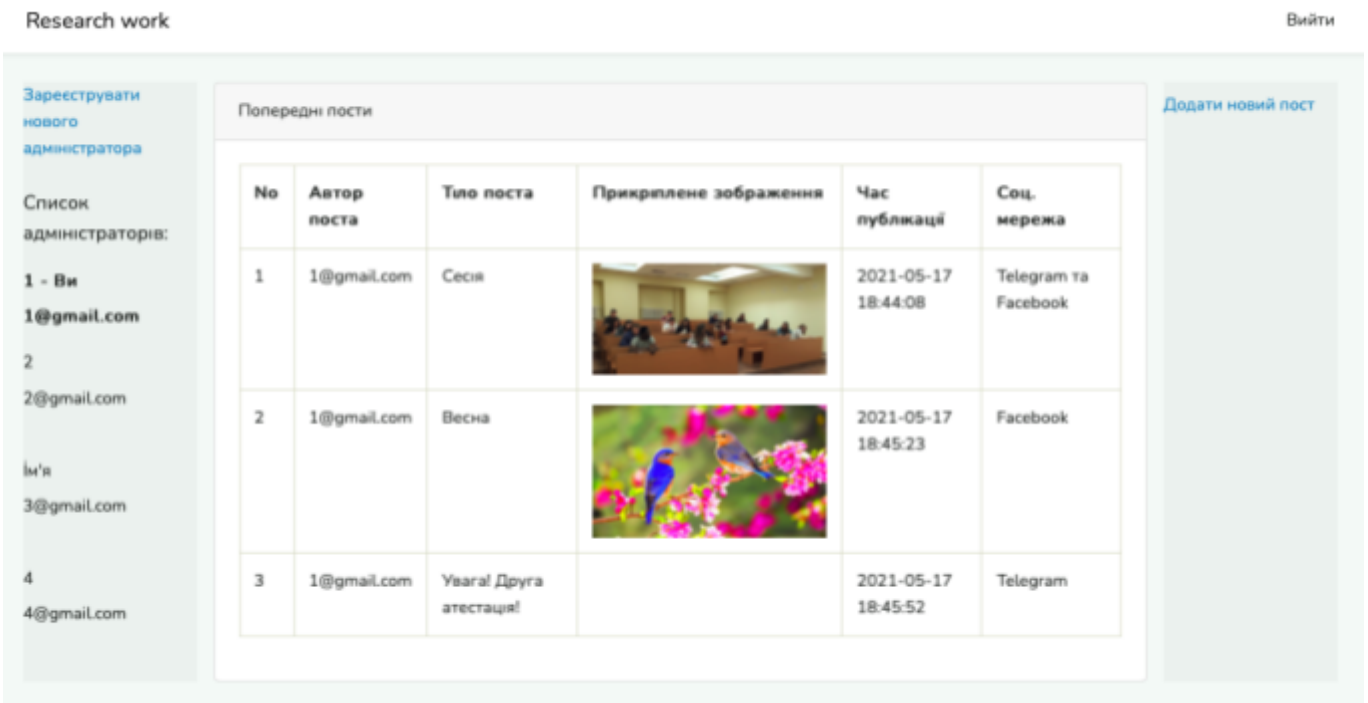

Рисунок 5.1 — Головна сторінка сервісу автопостингу

На рисунку 5.1 бачимо лівий сайдбар, де є посилання на сторінку для реєстрації нового адміністратора, а під ним список всіх адміністраторів, де авторизований користувач виділений жирним шрифтом.

В центральній частині відображається історія опублікованих постів, де вказаний автор посту, його зміст, час створення та соціальна мережа, де він був опублікований.

За допомогою клавіші “Вийти” у верхній частині сторінки можна вийти з системи, потрапивши після цього на форму авторизації.

Для того, щоб сформувати та опублікувати новий пост необхідно натиснути на кнопку “Створити новий пост” у правому сайдбарі.

Після цього потрапляємо на сторінку, зображену на рисунку 5.2.

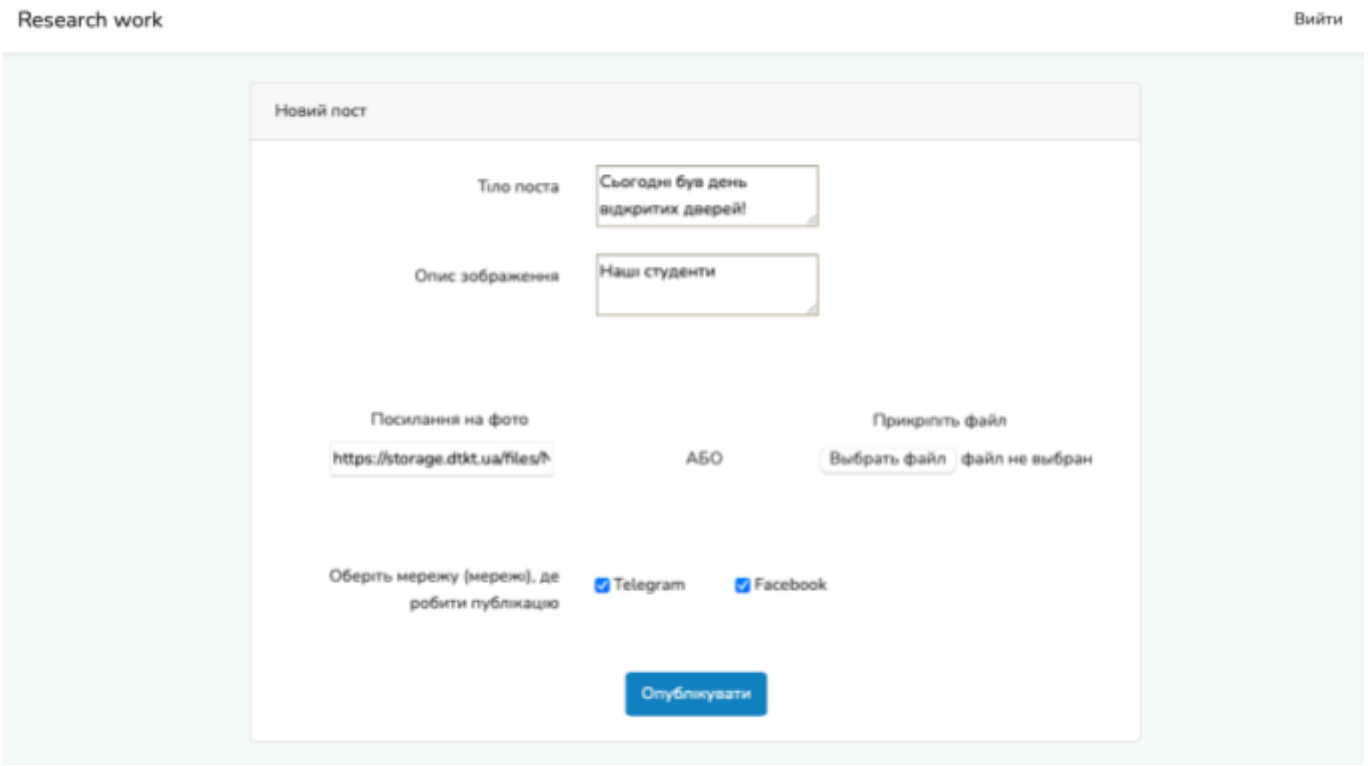

Рисунок 5.2 — Сторінка додавання нового поста

Бачимо такі поля для заповнення у формі:

- тіло поста;
- опис зображення;
- посилання на фото;
- прикріплення файлу;

- вибір соціальної мережі.

Є два варіанти поста, що може бути опублікованим: сам текст та текст, що супроводжується зображенням. Основний текст вказується в полі “Тіло поста”. “Опис зображення” супроводжує зображення тільки при публікації в телеграм. Фотографію можна додати з власних файлів користувача на його комп’ютері або вставити посилання на нього. Після натиснення на кнопку “Опублікувати” відбувається публікація поста в тих соціальних мережах, які обрав користувач.

Публікація постів в даній дипломній роботі відбувається в попередньо створені канали зв’язку. Це:

- канал в Telegram “Test AutoPosting”;
- сторінка в Facebook “Research Work”.

Очікуємо, що з’явиться публікація з наступним вмістом:

- тіло поста — “Сьогодні був день відкритих дверей”;
- опис зображення “Наші студенти”;
- зображення по посиланню в мережі.

Результат зображено на рисунку 5.3 та на рисунку 5.4.

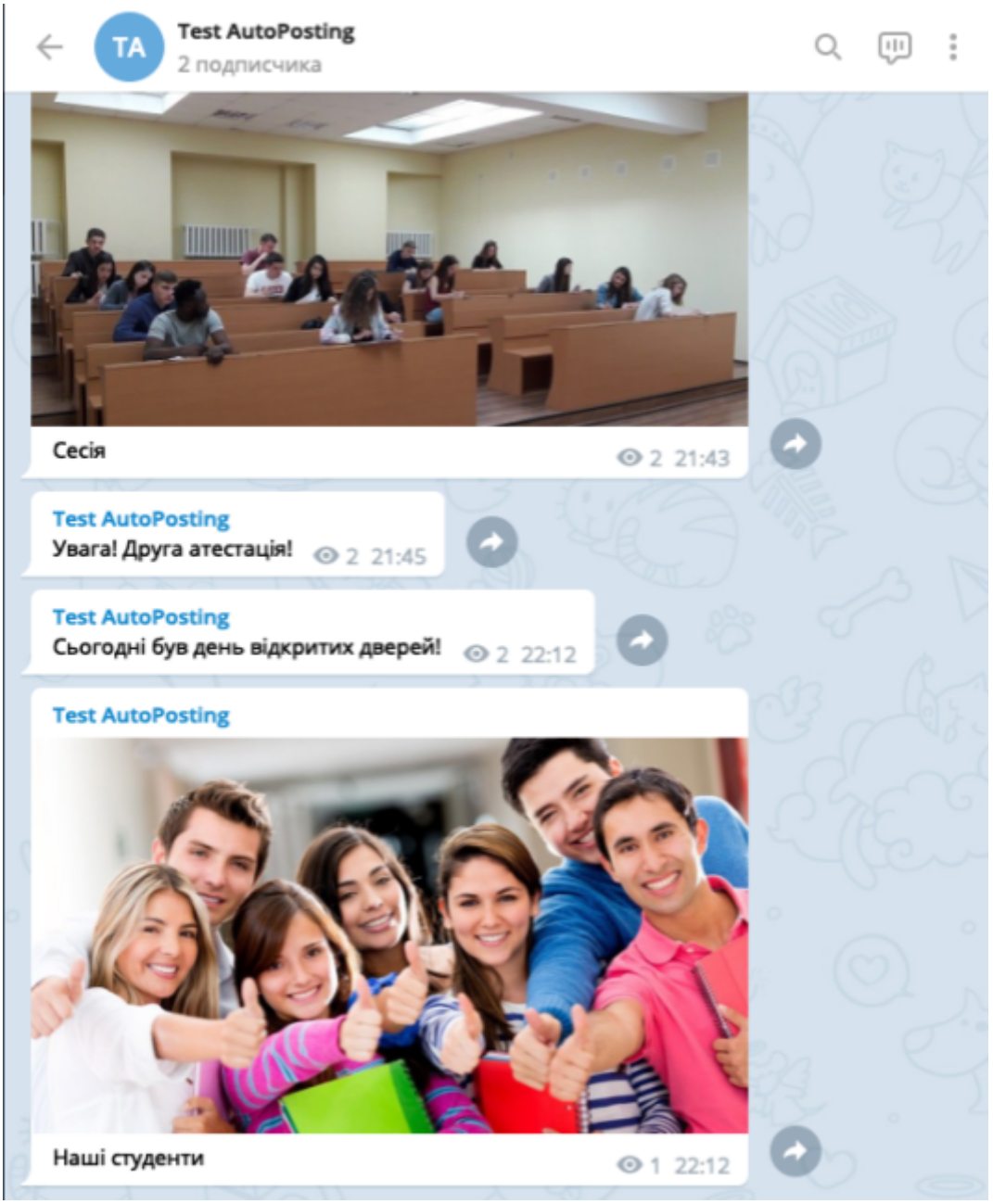

Рисунок 5.3 — Результат публікації посту на каналі в Telegram

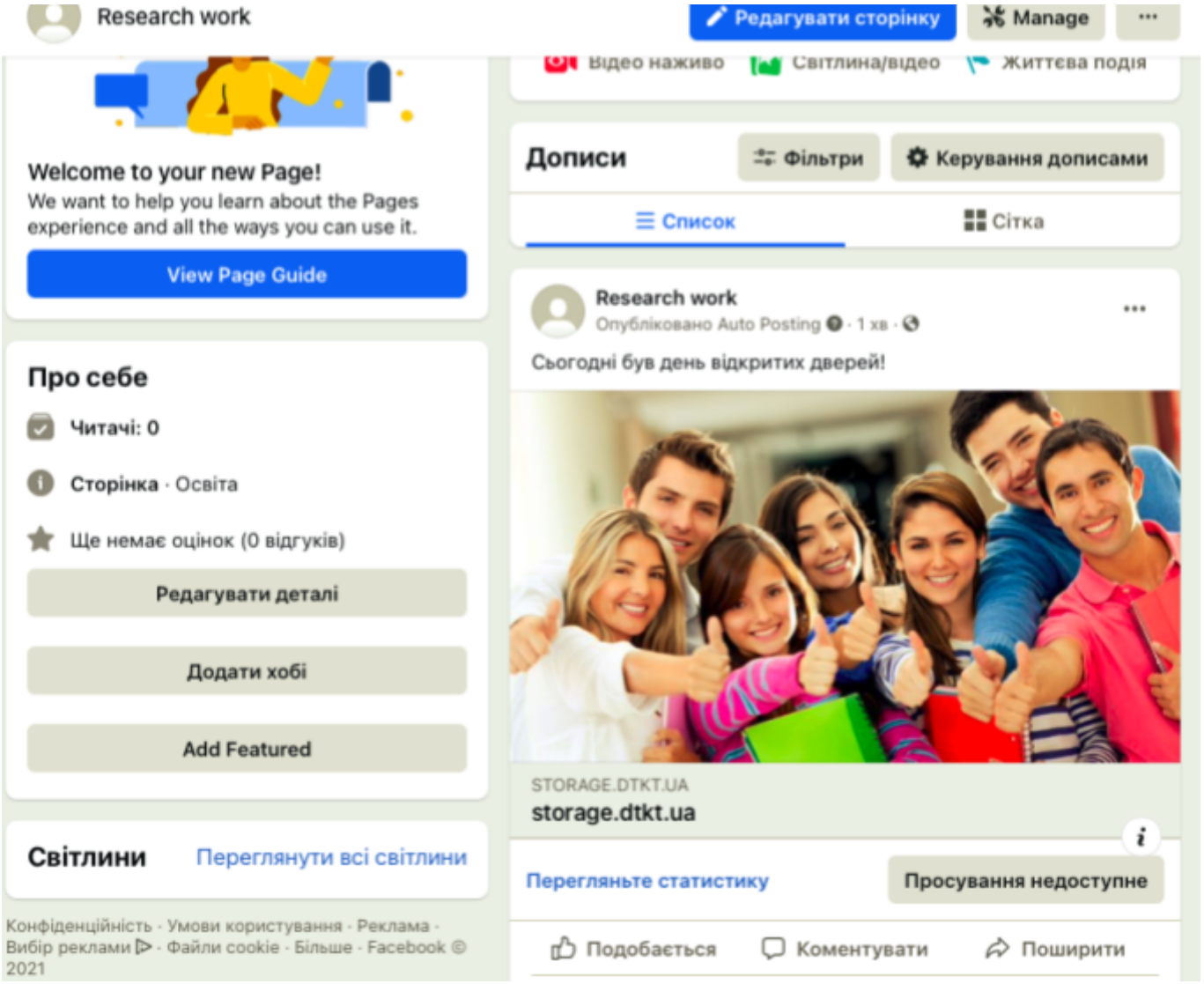

Рисунок 5.4 — Результат публікації посту на сторінці в Facebook

Як бачимо, в заданих соціальних мережах з’явився новий пост саме з тим наповненням, яке було задане. Також з’явився новий запис в таблиці бази даних, який відображається на головній сторінці сервісу в історії постів.

ВИСНОВКИ

В результаті проведеної роботи були виконані наступні завдання:

- сформовані вимоги до системи, що створюється;
- проаналізовано існуючі сервіси автопостингу;
- виділено їхні переваги та недоліки;
- прийнято рішення про створення нової системи для реплікації та синхронізації новин інформаційних каналів кафедри (факультету);
- зроблено вибір інструментів для реалізації системи;
- розглянуто методи публікації контенту в такі соціальні мережі, як Telegram та Facebook;
- спроектовано і розроблено веб-сервіс для реплікації та синхронізації інформаційних каналів;
- протестовано продукт.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gerardus Blokdyk Database Replication A Complete Guide - 2020 Edition, 2020. — 31с.
2. Дакетт Джон. HTML и CSS. Разработка и создание веб-сайтов, 2017. — 23-25с.
3. Мельник Р.А. Програмування веб-застосувань (фронт-енд та бек-енд). — Видавництво Львівської політехніки. 2018. — 34-37с.
4. Джоэл Х. Спольски, Джоэл о программировании, 2006. — 79с.
5. Кузнецов Максим, Симдянов Игорь. Объектно-ориентированное программирование на PHP. СПб.: «БХВ-Петербург», 2007. — 27-30с.
6. Мэтт Зандстра, PHP. Объекты, шаблоны и методики программирования, 2009. — 103с.
7. Christopher John Pecoraro. Mastering Laravel Packt Publishing, 2015. — 56-57с.
8. Matula, T. Laravel Application Development Cookbook. Packt Publishing, 2013. 10с.
9. Mukund Chaudhary, Ankur Kumar. PhpStorm Cookbook, 2014. — 17с.
10. Ирина Бородкина, Георгий Бородкин, Інженерія програмного забезпечення. Посібник для студентів вищих навчальних закладів. — Центр навчальної літератури. 2018. — 38-41с.
11. David Flanagan, and Gregor M. Novak Java-Script: The Definitive Guide, Second Edition / American Institute of Physics — 1998 — с.2-3 — Електронне видання: https://doi.org/10.1063/1.168647
12. Marc Delisle Mastering Phpmyadmin 3.1 for Effective Mysql Management — 2009 — 12с. Електронне видання. ISBN: 978-1-847197-86-3
13. Denis Shestakov Databases on the web: national web domain survey. — 2011 — 156с. Електронне видання:
14. Swaminathan Sivasubramanian, Michal Szymaniak, Guillaume Pierre, Maarten van Steen. Replication for web hosting systems — 2004 — 291с. Електронне видання: https://doi.org/10.1145/1035570.1035573

15. David Woodhouse. Quality and Quality Assurance — 1999 — 29с. Електронне видання: https://www.oecd-ilibrary.org/docserver/9789264173361-en.pdf?expires=1622926984&id=id&accname=guest&checksum=5BA644CBA8B5323F40998E7C4552C343#page=30

ДОДАТОК А

Програмні засоби реплікації та синхронізації новин інформаційних каналів кафедри (факультету)

Специфікація

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ ТВ7166_21А 1

Аркушів 1

Київ – 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ7166 _21Б 2	Записка Ващенко А.С ТВ-71.pdf	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ7166 _21Б 2-1	Post.php	Компонент — модель, що описує сутність поста
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ7166 _21Б 2-2	User.php	Компонент — модель, що описує сутність користувача
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ7166 _21Б 2-3	HomeController.php	Компонент, що відповідає за відображення головної сторінки
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ7166 _21Б 2-4	NewPostController.php	Компонент, що відповідає за створення та публікації поста
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ7166 _21Б 2-5	NewUserController.php	Компонент, що відповідає за реєстрацію нового користувача
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ7166 _21Б 2-6	web.php	Компонент роботи з маршрутами

ДОДАТОК Б

Програмні засоби реплікації та синхронізації новин інформаційних каналів кафедри (факультету)

Лістинг програми

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ ТВ7166_21Б 2

Аркушів 9

Київ – 2021

Post.php

```
<?php

namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
  protected $fillable = ['id', 'postBody', 'file', 'link', 'photoDescription', 'author', 'network'];

  public function render(){
      $posts = Post::all()->orderBy('created_at', 'DESC')->get();
      echo view($this->container)->with(['posts' => $posts]);
  }
}
```

User.php

```
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
  use HasFactory, Notifiable;
```

```
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    public function __get($key){
```

```
        return $this->attributes[$key];

    }
}
```

HomeController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Post;
use App\Models\User;

class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
```

```
  {
      $posts = Post::all();

      $users = User::all();

      $currentUser = auth()->user();




      return view('home')->with(['posts' => $posts])->with(['users' => $users])->with(['currentUser'
=> $currentUser]);
  }
}
```

NewPostController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Post;
use TelegramBot\Api\BotApi;

class NewPostController extends Controller
{
  public function makeNewPost(){



      $text = $this->checkingFields()[0];
      $textPhoto = $this->checkingFields()[1];
      $pic = $this->checkingFields()[2];
```

```
    $network ='';
    $isTelegram = false;



    if(empty($_POST['telegram']) && empty($_POST['facebook'])){
        return redirect('newpost')->with('error', 'Новий пост не доданий! Ви не вказали соціальну мережу');
    }


    if(!empty($_POST['telegram'])) {
        require_once("vendor/autoload.php");
        $isTelegram = true;
        if (true) {
            error_reporting(E_ALL & ~(E_NOTICE | E_USER_NOTICE | E_DEPRECATED));
            ini_set('display_errors', 1);
        }


        //$token = "1703039607:AAF_V9WC363-_objT_Cci92Obgt-VwmN7s8";
        $token = "1703039607:AAF_V9WC363-_objT_Cci92Obgt-VwmN7s8";
        $grupa = '@autopostingt';
        $bot = new \TelegramBot\Api\Client($token, null);

        if($text){
            $bot->sendMessage($grupa, $text);
        }


        if ($pic) {
```

```
            $bot->sendPhoto($grupa, $pic, $textPhoto);
        }
        $network='Telegram';


        //$bot->run();


    }
    if(!empty($_POST['facebook'])){

        require_once 'vendor/facebook/graph-sdk/src/Facebook/autoload.php';

        $app_id = "448647266404902";
        $app_secret = "1991b7a6f631abcd2dbc49f796fe6077";
        $token =
"EAAGYCtTxiiYBACTziZA1k0rjUK9GxUFDsA5aN5kQiuAUTGxjfTSmH4Wjs0PJK2CyNM
OYZBcdCJDMTRqUGOnTUNh1MsRuVpwMBnoHwaIaE0AIGt3HUUBzGiqWgKMS5fClkKo
fBh6DdxhvwOUzFpC4ZC8NyW1gOjCjnfDFzKGSV71DGmaeZCN1GzheluJNZAQJC7k0skp
WknQZDZD";

        $fb = new \Facebook\Facebook([
            'app_id' => $app_id,
            'app_secret' => $app_secret,
            'default_graph_version' => 'v10.0',
        ]);


        if($pic){
            $data = [
                'message' => $text,
                //'source' => $fb->fileToUpload($pic),
```

```
            'link' => $pic
        ];
    }
    else{
        $data = array('message' => $text);
    }


    try {

        $response = $fb->post("/102313712043575/feed", $data, $token);

    } catch(\Facebook\Exceptions\FacebookResponseException $e) {
        echo 'Graph returned an error: ' . $e->getMessage();
        exit;
    } catch(\Facebook\Exceptions\FacebookSDKException $e) {
        echo 'Facebook SDK returned an error: ' . $e->getMessage();
        exit;
    }

    $graphNode = $response->getGraphNode();

    if($isTelegram){
        $network = 'Telegram та Facebook';
    }else{
        $network='Facebook';
    }



}
```

```
    $email = auth()->user()->email;

    if(!$text && !$textPhoto && !$pic){
        return redirect('newpost')->with('error', "Новий пост не доданий! Ви не вказали дані!");
    }else{
        $post = new Post(array('postBody' => $text, 'photoDescription' => $textPhoto, 'link' =>
$pic, 'author' => $email, 'network' => $network));
        $post->save();
        return redirect()->route('home')->with('success', 'Новий пост доданий!');
    }
  }

  public function checkingFields(){
    if($_POST['postBody']){
        $text = $_POST['postBody'];
    }
    else{
        $text='';
    }
    if ($_POST['link']) {
        $pic = $_POST['link'];
        $textPhoto = $_POST['photoDescription'];
    } else if($_FILES && $_FILES["filename"]["error"]== UPLOAD_ERR_OK){
            $name = "files/" . $_FILES["filename"]["name"];
            move_uploaded_file($_FILES["filename"]["tmp_name"], $name);
            $pic = "http://researchwork.online/" . $name;
            $textPhoto = $_POST['photoDescription'];

    }else{
        $textPhoto='';
        $pic='';
    }
```

```
        return array($text, $textPhoto, $pic);
    }
}
```

NewUserController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;

class NewUserController extends Controller
{
    public function registerUser(){
        //echo "hi!";
        $name = $_POST['name'];
        //echo $name;
        $email = $_POST['email'];
        //echo $email;
        $password = $_POST['password'];
        //echo $password;

        $user = new User(array('name' => $name, 'email' => $email, 'password' =>
Hash::make($password)));
        $user->save();
        return redirect()->route('home')->with('success', 'Новий адміністратор доданий!')}}
```

web.php

```
<?php
use Illuminate\Support\Facades\Route;
Route::get('/', function () {
  return view('auth/login');
});
Route::get('/privacy', function () {
  return view('privacy');
});

Route::get('/newuser', function () {
  return view('auth/register');
});

Route::get('/newpost', function () {
  return view('newpost');
});
Route::post('/storenewuser', [App\Http\Controllers\NewUserController::class, 'registerUser']);
Route::post('/storenewpost', [App\Http\Controllers\NewPostController::class, 'makeNewPost']);
Auth::routes();
Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');
```

ДОДАТОК В

Програмні засоби реплікації та синхронізації новин інформаційних каналів кафедри (факультету)

Опис програмного коду

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТВ7166_21В 3

Аркушів 10

Київ – 2021

АНОТАЦІЯ

Додаток надає інформацію про застосунок, що є програмним засобом для реплікації та синхронізації новин інформаційних каналів кафедри чи факультету.

Додаток надає можливість користувачам сервісу пройти реєстрацію та авторизацію, оформити статтю новин за допомогою форми та здійснити її публікацію в обрані соціальні мережі. Програма підтримує такі мережі, як Telegram та Facebook. Також адміністратор має можливість зареєструвати нового користувача системи, який також матиме повний доступ до функціоналу сервісу, переглянути повний список користувачів та історію раніше опублікованих постів.

Програма розроблена з використанням фреймворку Laravel та мови програмування PHP в середовищі розробки PHPStorm.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Відповідно до назви дипломної роботи, проект має такий опис :“Програмні засоби реплікації та синхронізації новин інформаційних каналів кафедри (факультету)”.

Для користування розробленим додатком необхідно мати встановлений Інтернет-браузер та доступ до мережі. Операційна система ролі не грає.

Програма розроблена з використанням фреймворку Laravel та мови програмування PHP в середовищі розробки PHPStorm.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Додаток призначений для вирішення проблеми несистематизованого процесу обробки сторінок кафедри чи факультету в соціальних мережах. Програма має наступний функціонал:

1. Можливість авторизованого доступу в систему;
2. Створення та публікація постів в соціальні мережі;
3. Перегляд історії постів;
4. Можливість реєстрації нового користувача вже існуючим адміністратором;
5. Перегляд списку адміністраторів системи.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Загальна логічна структура проекту така:

1. Користувач потрапляє на сервіс та починає процедуру авторизації.
2. Система отримує логін та пароль, введені користувачем, та шукає відповідний запис в таблиці бази даних. Якщо авторизація пройшла успішно, проходить перенаправлення на головну сторінку, в іншому випадку — виводиться повідомлення про помилку та очікується повторний ввід даних.
3. З бази даних вивантажується таблиця з раніше опублікованими постами на даному сервісі та виводиться на екран користувача.
4. Якщо авторизованому адміністратору необхідно додати нового користувача системи, то необхідно заповнити форму реєстрації на відповідній сторінці валідними даними.
5. При переході на сторінку формування нового поста, користувач повинен заповнити форму валідними даними. Якщо все коректно, система виконує публікацію введеної інформації в соціальні мережі та зберігає їх як новий запис в таблиці бази даних.

ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для користування даним сервісом необхідно мати пристрій зі встановленим браузером та доступом до мережі Інтернет.

Операційна система пристрою не важлива.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для користування веб-сервісом необхідно перейти за посиланням http://researchwork.online та пройти процес авторизації. Дані першого зареєстрованого користувача знаходяться в базі даних, доступ до якої має тільки розробник.

ВХІДНІ ДАНІ

Вхідними даними системи є дані, які вводить користувач при авторизації, та дані з форми створення поста, які в подальшому формують публікацію в соціальні мережі.

ВИХІДНІ ДАНІ

Вихідними даними розробленої системи є повідомлення у відповідних каналах інформаційного зв’язку, тобто опублікований пост в таких соціальних мережах як Telegram та Facebook.