

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

**«Веб-система для завантаження і соціального обміну
світлинами»**

Виконала студентка групи КН-20-1
ЮР'ЄВА Вікторія Вікторівна

Керівник роботи:
КУЗЬМЕНКО Олександр Вікторович

Рецензент:
ПЕТРОСЯН Руслан Валерікович

Житомир – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«25» січня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **ЮР'ЄВА Вікторія Вікторівна**

Керівник роботи: **КУЗЬМЕНКО Олександр Вікторович**

Тема роботи: **«Веб-система для завантаження і соціального обміну світлинами»**,

затверджена наказом закладу вищої освіти від **«25» січня 2024 р., №22/с**

Термін здачі закінченої роботи: **«18» червня 2024 р.**

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

об'єктом дослідження є веб-сайт "Соціальний фотосервіс";

предметом дослідження є побудова архітектури та розробка веб-системи для завантаження і соціального обміну світлинами.

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	Кузьменко О.В.	1 березня 2024	1 березня 2024
2	Кузьменко О.В.	15 квітня 2024	15 квітня 2024
3	Кузьменко О.В.	15 травня 2024	15 травня 2024

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук і огляд програм-аналогів.	19 січня 2024 – 15 лютого 2024	Виконано
2	Формулювання технічного завдання.	16 лютого 2024 – 27 березня 2024	Виконано
3	Проектування структури системи.	28 березня 2024 – 15 квітня 2024	Виконано
4	Написання програмного коду.	16 квітня 2024 – 15 травня 2024	Виконано
5	Тестування системи і виправлення неполадок.	16 травня 2024 – 23 травня 2024	Виконано
6	Оформлення пояснювальної записки.	24 травня 2024 – 2 червня 2024	Виконано
7	Створення презентації.	3 червня 2024 – 9 червня 2024	Виконано

Здобувач вищої освіти **Вікторія ЮР'ЄВА**

Керівник **Олександр КУЗЬМЕНКО**

РЕФЕРАТ

Випускна робота бакалавра складається з створення веб-системи для завантаження і соціального обміну світлинами, а також пояснювальної записки. Пояснювальна записка до дипломної роботи містить 91 сторінку, 39 ілюстрацій, додатки та 31 літературне джерело.

Основною метою даної роботи є планування мікро-сервісної архітектури програмного продукту та безпосередньо реалізації даного сервісу, що буде задовольняти високі стандарти безпеки, зручності та надійності. Основним функціоналом, якого буде можливість поширення фотографій для своїх підписників, пошук своїх друзів та перегляд їхніх профілів. Також до основного функціоналу входить реалізація компонентів редагування зображень прямо з веб-сервісу.

У ході випускної роботи були проведені дослідження з метою покращення ефективності графічних трансформацій на Front-end та Back-end частинах сервісу, за результатами яких було оптимізовано процес обрізання, стискування та збереження зображень використовуючи технологію Amazon Simple Storage Service.

Ключові слова: ФОТОСЕРВІС, PHP, SYMFONY, NGINX, JAVASCRIPT, REACT, DOCKER, AWS, S3, MYSQL, PHPMYADMIN, SWAGGER, COMPOSER, JWT.

					КН.КР.Б – 122 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		В.В. ІОР'ЄВА			Веб-система для завантаження і соціального обміну світлинами Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник		О.В. КУЗЬМЕНКО					3	91
Рецензент		Р.В. ПЕТРОСЯН				Житомирська політехніка, група КН-20-1		
Н. контр.								
Зав. каф.		М.С. ГРАФ						

ABSTRACT

The bachelor's thesis consists of the implementation of the Social Photo Service program, as well as an explanatory note. The explanatory note to the thesis contains 91 pages, 39 illustrations, attachments and 31 literary sources.

The main goal of this work is to plan the microservice architecture of the software product and directly implement this service, which will meet high standards of security, convenience and reliability. The main functionality will be the ability to share photos for your followers, search for your friends and view their profiles. Also, the main functionality includes the implementation of the image editing component directly from the web service.

In the course of the final work, studies were conducted to improve the efficiency of graphic transformations on the Front-end and Back-end parts of the service, because of which the process of cropping, compressing and saving images using Amazon Simple Storage Service technology was optimized.

Keywords: PHOTOSERVICE, PHP, SYMFONY, NGINX, JAVASCRIPT, REACT, DOCKER, AWS, S3, MYSQL, PHPMYADMIN, SWAGGER, COMPOSER, JWT.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	8
1.1 Постановка задачі	8
1.2 Аналіз аналогів програмного продукту	10
1.3 Вибір архітектури	13
1.4 Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення	14
Висновки до розділу 1	29
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-СИСТЕМИ	31
2.1 Проектування веб-системи	31
2.2 Проектування бази даних.....	37
2.3 Розробка логічних модулів	48
Висновки до розділу 2	52
РОЗДІЛ 3. ЗАПУСК ТА НАЛАШТУВАННЯ ЗАСТОСУНКУ	54
3.1 Порядок встановлення та налаштування системи.....	54
3.2 Інтерфейс	57
Висновки до розділу 3	66
ВИСНОВКИ.....	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	70
ДОДАТКИ.....	73

ВСТУП

Актуальність теми. Створення власної соціальної мережі фотосервісу має ряд вагомих переваг. По-перше, це дозволяє користувачам зберігати свої фото та відео у безпечному та приватному середовищі, забезпечуючи контроль над доступом до свого контенту. Крім того, створення власної платформи дає можливість впроваджувати нові функції та сервіси, які відповідають унікальним потребам аудиторії.

Також, соціальні мережі фотосервісів стають не лише місцем спілкування, а й ефективним інструментом для розвитку бізнесу та особистого бренду. Компанії та творчі особистості можуть використовувати цю платформу для просування своїх продуктів, послуг або творчих робіт, залучаючи нових клієнтів та прихильників.

Крім того, власна соціальна мережа фотосервісу сприяє збереженню інтелектуальної власності користувачів, уникненню ризиків, пов'язаних із зловживанням особистою інформацією, а також створює унікальну спільноту за інтересами.

Отже актуальність роботи полягає в відкритті багатьох можливостей для інновацій, розвитку бізнесу та творчого самовираження, роблячи фотосервіс надзвичайно актуальним у сучасному цифровому світі.

Метою цієї кваліфікаційної роботи є розробка застосунку побудованого на мікросервісній архітектурі використовуючи: docker, docker-compose, nginx (Веб-сервер), Symfony фреймворк та React.js. Реалізація такого застосунку дасть можливість користувачу відчувати себе творцем своєї сторінки та мати доступ до перегляду сторінок своїх друзів. Таким чином ця платформа може стати ще одним місцем соціалізації. Розроблений функціонал з редагування зображень повністю задовольнить усі потреби користувача, щодо якості відображення, швидкості редагування та перегляду дописів.

Встановлена мета обумовлює наступні завдання:

- аналіз UI/UX, пов'язані з Front-end частиною сайту;

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

- розробка мікро-сервісної архітектури, проектування структури основних сервісів застосунку;
- обґрунтування засобів реалізації продукту;
- проектування бази даних;
- проектування систем захисту даних користувачів;
- розробка серверної частини;
- реалізація користувацької частини.
- програмна реалізація застосунку.

Об'єктом дослідження є веб-сайт "Соціальний фотосервіс".

Предметом дослідження є побудова архітектури та розробка веб-системи для завантаження і соціального обміну світлинами.

Інструментарієм роботи є сучасні технології розробки веб-застосунків, а саме: мови програмування PHP, Java Script та yaml розмітка для конфігурування нашого застосунку.

Також в дипломній роботі було використано:

- об'єктно-орієнтований підхід програмування;
- систему контейнізації docker;
- систему керування доступом на основі ролей (RBAC);
- компонентний підхід верстки застосунку;
- систему взаємодії застосунку з БД (Doctrine ORM);
- систему надсилання листів на пошту (Symfony-mailer);
- автентифікаційний токен розпізнавання користувача (JWT);
- систему безпеки (Symfony security);
- систему маршрутизації;
- систему cloud збереження даних Amazon S3;

Проект може бути використаний в розважальних, рекламних, маркетингових цілях.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Постановка задачі

Основною задачею роботи є розробка веб-орієнтованої багатокористувацької системи для завантаження і соціального обміну світлинами передбачає створення адміністративної та користувацької частин на фреймворку Symfony та бібліотеці React.js відповідно. Застосунок також потребує бази даних, яка буде спроектована та реалізована на MySQL Server. Для забезпечення масштабованості та легкості управління розгортанням системи, використовується віртуальне середовище на основі Docker контейнерів.

Основною метою розробки є створення функціональності для завантаження, зберігання та обробки фотографій користувачами, а також забезпечення спільного перегляду та обміну цими зображеннями. Користувацька частина, буде забезпечувати зручний інтерфейс для користувачів, де вони зможуть завантажувати свої фотографії, взаємодіяти з іншими учасниками та ділитися своїм творчим контентом.

Реалізація мікросервісної архітектури дозволить долучати нові функції та сервіси без перешкодного впливу на вже існуючі компоненти системи. Це забезпечить гнучкість, масштабованість та легкість управління проектом, дозволяючи швидко реагувати на зміни в потребах користувачів та вимогах ринку.

Загальні функції системи:

- наявність клієнтської (frontend) та серверної (backend) частини;
- каталог фото;
- реєстрація користувачів, робота з клієнтами;
- оповіщення користувачів на пошту, аутентифікація та інше;
- розподілення ролей на гість та авторизований користувач;

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

- кожен автентифікований користувач, незалежно від ролі, повинен мати свою публічну, або приватну сторінку;
- кожен користувач зобов'язаний заповнити всю інформацію про себе, та підтвердити свою особистість;
- інструмент корегування зображень.

Функції клієнтської частини:

- стандартні функції користувачів: реєстрація, вхід;
- кабінет клієнта: редагування профілю, перегляд власних фотографій;
- перегляд фотографій: (сортування, пошук користувачів) та їхні дописів;
- створення дописів;
- можливість підписання на певний профіль.

Стек використовуваних технологій:

- Symfony (backend).
- React.js (frontend).
- Swagger.
- Twig.
- Doctrine ORM.
- JWT auth token + refresh.
- PHP 8.0.
- JS.
- JSX.
- MySQL server.
- PhpMyAdmin.
- Composer.
- Npm.
- Docker, docker-compose.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2 Аналіз аналогів програмного продукту

Сучасний інтернет переповнений соціальними мережами, які спеціалізуються на обміні та публікації фотографій. Ось декілька з найпопулярніших аналогів:

Instagram: Це одна з найвідоміших соціальних мереж, присвячених фотографіям. Користувачі можуть додавати свої знімки, накладати фільтри та ділитися ними зі своїми підписниками.

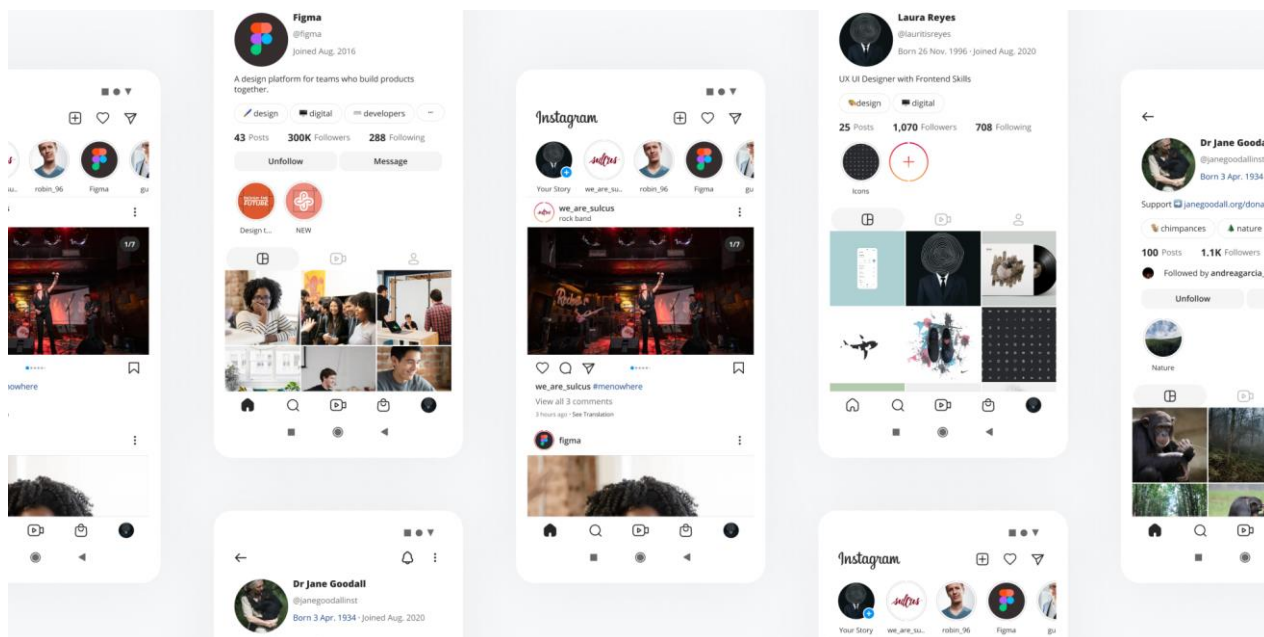


Рисунок 1.1. – Інтерфейс Instagram

Звичайно, Instagram є однією з найпопулярніших соціальних мереж, особливо серед любителів фотографії та візуального контенту. Ось деякі переваги та недоліки цієї платформи:

Переваги:

- Візуальний контент: Instagram є ідеальним місцем для спілкування за допомогою фотографій і відео. Користувачі можуть створювати стильні знімки, застосовувати фільтри та ефекти, щоб робити свій контент привабливішим.
- Залучення аудиторії: Instagram має велику активну аудиторію, що дозволяє користувачам ділитися своїми ідеями, враженнями та творчістю з великою кількістю людей.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

- Соціальна взаємодія: користувачі можуть легко взаємодіяти зі своїми підписниками шляхом вподобань, коментарів та приватних повідомлень, що створює сприятливу атмосферу спілкування.
- Маркетинг та брендинг: для бізнесу Instagram є потужним інструментом для просування товарів та послуг через візуальний контент та взаємодію з аудиторією.
- Можливості для творчості: платформа надає широкі можливості для самовираження та творчості через фотографії, відео, сторіз та інші формати контенту.

Недоліки:

- Алгоритмічне сортування: Instagram використовує алгоритми для сортування контенту, що може призвести до того, що деякі пости можуть бути менше помічені або не показані загалом.
- Спам та фейкові облікові записи: як і в інших соціальних мережах, Instagram також має проблеми зі спамом та фейковими профілями, що може порушувати користувацький досвід.
- Постійні зміни алгоритмів: часті зміни в алгоритмах Instagram можуть змінювати спосіб, яким контент відображається в стрічці, що може вплинути на видимість публікацій.
- Залежність від візуального контенту: для того, щоб бути успішним на Instagram, потрібно регулярно створювати візуальний контент, що може бути вимогливим.
- Проблеми з приватністю даних: як і в більшості соціальних мереж, на Instagram також можуть виникати проблеми з приватністю даних користувачів через збір та використання персональної інформації.

Pinterest: Pinterest - це платформа, де користувачі можуть створювати та ділитися віртуальними дошками зі своїми улюбленими зображеннями та ідеями. Він широко використовується для збору вдома, дизайну та моди.

Переваги:

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

- Креативність: Pinterest є відмінним джерелом для знаходження та реалізації креативних ідей. Користувачі можуть створювати віртуальні дошки зі своїми улюбленими зображеннями, проектами та ідеями.
- Організація: платформа дозволяє легко організовувати та зберігати зображення та ідеї за допомогою віртуальних дошок, які можна розділити за темами або проектами.
- Збір та пошук інформації: Pinterest є чудовим інструментом для збору та пошуку інформації про будь-яку тему, включаючи моду, дизайн, кулінарію, подорожі та багато іншого.
- Спільнота користувачів: платформа об'єднує спільноту користувачів, які обмінюються ідеями, враженнями та ресурсами, що створює стимулююче середовище для творчості та співпраці.
- Можливості для бізнесу: для підприємств Pinterest може бути важливим інструментом для маркетинга та просування продуктів через створення привабливих візуальних контенту та привернення цільової аудиторії.

Недоліки:

- Обмежений контент: незважаючи на широкий спектр тем, доступних на Pinterest, деякі категорії можуть бути менш представленими або обмеженими у порівнянні з іншими соціальними мережами.
- Залежність від візуального контенту: хоча Pinterest ідеально підходить для візуальної креативності, користувачам може бракувати можливості виразити свої думки або ідеї без використання зображень.
- Візуальні стандарти: для того, щоб бути успішним на Pinterest, потрібно відповідати високим візуальним стандартам, що може бути важко реалізувати.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

- Приватність даних: можливі проблеми з приватністю користувачів через збір та обробку їх особистої інформації.

1.3 Вибір архітектури

При виборі архітектури для веб-системи для завантаження і соціального обміну світлинами “Соціальний фотосервіс”, було прийнято рішення побудувати мікросервісну архітектуру використовуючи **docker** та **docker-compose**, як систему контейнеризації мікросервісів, мережевої системи та взаємодії між мікросервісами (**docker network**) та систему автоматичного створення кешування конфігураційних даних та файлів проекту, зокрема і бази даних (**docker volumes**).

Мікросервіси:

- **Nginx** – веб сервер основною перевагою якого є легкість налаштування.
- Fullstack фреймворком **Symfony 5.6**, але в нашому проекті він використовувався лише, як мікросервіс для побудови **RESTful API**.
- Frontend фреймворк React.js.
- **MySQL** – база даних.
- **PhpMyAdmin** – веб-застосунок для зручного адміністрування сервера MySQL.

Таким чином ми отримуємо мікросервісну архітектуру побудовану на системі docker, яку легко масштабувати та хостити на будь-якій операційній системі.

На рівні серверної частини було використано багато компонент для полегшення написання програми, збільшення швидкості роботи та встановлення заходів безпеки даних користувачів.

Перелік **Bundles**:

- JSON Web Token.
- Swagger.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

- AWS S3.
- Doctrine ORM.
- Symfony Mailer.
- Twig.
- Symfony routing.
- Security Bundle.
- Nelmio Cross-Origin Resource Sharing Bundle.
- Imagine.

Перелік пакетів на рівні клієнтської частини:

- MUI.
- React Advanced Cropper.
- JWT-decoder.
- Axios.

1.4 Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення

В ході виконання проекту було використано мікросервісну архітектуру застосунку, це було зумовлено, можливістю розвитку застосунку, більше того на проєкті використовується брокер повідомлень RabbitMq, для спілкування між сервісами, побудови черг повідомлень та розподілу їх за допомогою exchangers.

Кожен мікросервіс представляє собою docker контейнер, як окремий веб-застосунок, поєднаний однією віртуальною мережею. Docker-compose повноцінно групує всі мікро-сервіси в один застосунок.

Docker

Файли docker-compose та docker-compose.local використовуються для налаштування середовищ Docker у проєкті, проте їх використання відрізняється за своєю призначеністю та контекстом:

1) docker-compose.yml (або docker-compose.yaml):

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

- Це основний файл конфігурації для середовища Docker проекту.
- Він містить загальні налаштування для всіх середовищ, такі як контейнери, мережі, томи, змінні середовища тощо.
- Використовується для запуску тестів, розгортання в різних середовищах (наприклад, локально, на стейджі, на продакшні), а також для роботи з командами Docker.

2) **docker-compose.local.yml:**

- Це додатковий файл конфігурації, який використовується для локальної розробки та тестування.
- Використовується для визначення налаштувань, специфічних для локального середовища, таких як розширення контейнерів для підтримки інструментів розробки, налаштування середовища змінних середовища тощо.
- Цей файл містить додаткові налаштування, які перевизначають та доповнюють конфігурацію з основного **docker-compose.yml**.
- Його використання дозволяє мати різні налаштування для локального середовища порівняно з іншими середовищами, що допомагає в уникненні конфліктів та спрощує розробку на локальному комп'ютері.

Docker-compose.yml

Docker-файл визначає конфігурацію для трьох контейнерів у середовищі. Усі приватні ключі, назви токени зберігаються в env файлі.

Контейнери:

- 1) **api:** контейнер будується з використанням Dockerfile, розташованого у директорії **./docker/api**. Він має назву **\${PROJECT_NAME}-api**. До цього контейнера підключається

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

том **./api**, який мапується до **\${API_DIRECTORY}**. Цей контейнер приєднується до мережі **\${PROJECT_NAME}**.

2) **frontend**: цей контейнер будується з використанням **Dockerfile**, розташованого у директорії **./frontend**. Він має назву **\${PROJECT_NAME}-frontend**. Цей контейнер також приєднується до мережі **\${PROJECT_NAME}** з вказаною IPv4-адресою **\${FRONTEND_IPV4}**.

3) **nginx**: цей контейнер будується з використанням **Dockerfile**, розташованого у директорії **./docker/nginx**. Він має назву **\${PROJECT_NAME}-nginx**. До цього контейнера приєднуються два тома: **./api**, який мапується до **\${API_DIRECTORY}**, та **./frontend**, який мапується до **\${FRONTEND_DIRECTORY}**. Цей контейнер також залежить від контейнера **api**. Йому також передаються змінні середовища **FRONTEND_DIRECTORY** та **API_DIRECTORY**. Він також приєднується до мережі **\${PROJECT_NAME}** з вказаною IPv4-адресою **\${NGINX_IPV4}**.

Крім того, у файлі визначені три тома - **api**, **nginx** та **data**. Також створюється мережа **\${PROJECT_NAME}** зі згаданою підмережею.

Docker-файл встановлює зв'язок між трьома основними компонентами застосунку - **API**, клієнтським інтерфейсом (Frontend) та веб-сервером (Nginx), та забезпечує належне налаштування їх спільної роботи у Docker-середовищі.

Docker-compose.local.yml

Цей файл **docker-compose.local** визначає конфігурацію для кількох контейнерів у середовищі розробки.

Основні складові цього файлу:

1) **api**: контейнер будується за допомогою **Dockerfile.local**, який знаходиться в директорії **./docker/api**. Він має налаштування для рестарту, виключаючи випадок, коли контейнер зупинений. Контейнер використовує том, щоб мапувати конфігураційний

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

файл xdebug.ini для налаштування Xdebug. Також встановлено середовище для Xdebug, вказуючи хост, режим ім'я сервера та інші параметри.

- 2) **frontend**: контейнер будується за допомогою Dockerfile.local, який знаходиться в **../docker/frontend**. Використовується команда `prpm start` для запуску контейнера, а також включаються деякі параметри, які забезпечують плавний процес розробки з допомогою hot-reloading.
- 3) **nginx**: цей контейнер також будується за допомогою Dockerfile.local. Використовуються теми для мапування сертифікатів SSL та параметри для встановлення середовища, такі як хост і порт.
- 4) **mysql**: контейнер використовує офіційний образ MySQL версії 8.0.20. Йому призначається ім'я контейнера, встановлюються порти для доступу до бази даних, налаштовуються середовищні змінні, такі як ім'я бази даних та пароль кореневого користувача.
- 5) **phpmyadmin**: контейнер використовує образ phpMyAdmin для взаємодії з MySQL. Встановлюються порти для доступу до phpMyAdmin, налаштовуються змінні середовища для з'єднання з MySQL.
- 6) **rabbitmq**: контейнер використовує образ RabbitMQ з включеним менеджером, який дає можливість керувати чергами. Встановлюються порти для доступу до менеджера та до самої системи черг, налаштовуються середовищні змінні, такі як користувач, пароль та інше.

Крім того, визначений том для MySQL, який мапується до **/var/lib/mysql**, щоб зберігати дані бази даних після перезапуску контейнера.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

```
=> => exporting manifest sha256:dcfb190936c34d38e0d679584435bbf51861576d1435df8414e0c4b7d8aa1b1e
=> => exporting config sha256:802c47e07449e87bde53e158312aba3401d3e9d2184323e88de02f3477b19bce
=> => sending tarball
=> [nginx nginx] importing to docker
[+] Running 6/6
✓ Container courselab-mysql      Started
✓ Container courselab-frontend  Started
✓ Container courselab-api        Started
✓ Container courselab-rabbitmq   Started
✓ Container courselab-phpmyadmin Started
✓ Container courselab-nginx      Started
```

Рисунок 1.2. – Старт роботи контейнерів

Symfony

Цей проект - це сучасний веб-застосунок, побудований на Symfony 5.4. Symfony 5.4 - це потужний PHP-фреймворк, який відомий своєю надійністю, гнучкістю та високою продуктивністю. Його розширені можливості дозволяють розробникам швидко створювати різноманітні веб-застосунки, включаючи підприємницькі рішення, забезпечуючи при цьому високу стабільність і безпеку.

Однією з головних переваг Symfony є його широкий спектр функціональності та готових компонентів, які дозволяють швидко розробляти рішення для різних потреб бізнесу. Фреймворк надає зручні інструменти для роботи з базами даних, формами, маршрутизацією, аутентифікацією та іншими аспектами веб-розробки. Крім того, Symfony має велику та активну спільноту розробників, що постійно вносять удосконалення та виправляють помилки, що забезпечує постійний розвиток та підтримку фреймворку.

Ще однією важливою перевагою Symfony є його висока безпека. Фреймворк надає інструменти для захисту від таких загроз, як SQL-ін'єкції, XSS атаки та інші види атак на безпеку, що робить його ідеальним вибором для розробки підприємницьких застосунків, які потребують надійного захисту конфіденційної інформації.

Він використовує ряд важливих бібліотек та фреймворків, таких як Doctrine для роботи з базою даних, Symfony Security Bundle для забезпечення безпеки, а також Symfony Framework Bundle для базового функціоналу.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

Для роботи з зовнішніми сервісами та інфраструктурою використовуються пакети, такі як AWS SDK для роботи з Amazon Web Services, та PHPAmqpLib для роботи з RabbitMQ.

React

Фронтенд частина застосунку реалізується сучасною веб-бібліотекою - React з використанням різноманітних бібліотек та компонентів для забезпечення високої продуктивності та зручності в розробці.

Використовуються такі ключові бібліотеки та інструменти:

- **React та ReactDOM:** основа для створення користувацького інтерфейсу, яка дозволяє реалізовувати компоненти та логіку веб-застосунку.
- **Tailwind CSS:** модульний CSS-фреймворк, який дозволяє швидко створювати стильні та респонсивні інтерфейси з використанням набору змінних класів.
- **Styled-components та Emotion:** бібліотеки для роботи з CSS в React, які дозволяють вбудовувати CSS безпосередньо в компоненти.
- **Material-UI:** колекція готових компонентів та інструментів для розробки інтерфейсу, яка забезпечує швидку і зручну реалізацію стандартних та власних елементів.
- **React Router:** бібліотека для навігації між сторінками в React-застосунку, що дозволяє створювати односторінкові застосунки та додавати маршрутизацію.
- **axios:** бібліотека для здійснення HTTP-запитів з використанням Promise, що дозволяє зручно спілкуватися з сервером.
- **JWT-decode:** утиліта для декодування JSON Web Tokens (JWT), яка використовується для розшифрування та отримання даних з токенів автентифікації.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

- **React Helmet Async:** компонент React для динамічного управління метатегами сторінки, що дозволяє налаштовувати заголовки, мета-теги та інші важливі елементи SEO.

Ці бібліотеки разом забезпечують стабільність, продуктивність та зручність в розробці веб- застосунку.

AWS S3

AWS SDK - це набір інструментів та бібліотек для роботи з хмарними послугами Amazon Web Services. Однією з ключових послуг, яку надає AWS SDK, є Amazon S3 (Simple Storage Service) - потужна та надійна послуга для зберігання об'єктів у хмарі. Amazon S3 дозволяє розробникам створювати віртуальну файлову систему, яка надає доступ до збережених об'єктів через HTTP або HTTPS протоколи. Коли користувач реєструється у застосунку, застосунок автоматично створює папку на Amazon S3 для збереження користувацьких файлів, фотографій, постів та аватарок. Це робиться шляхом створення унікального ідентифікатора користувача, який буде використовуватися для створення унікального шляху до папки в сховищі Amazon S3.

Наприклад, папка для зберігання файлів фотографій користувача має наступний шлях:

s3://bucket-name/users/user_id/photos/

Аналогічно, для збереження файлів постів та аватарок користувача, застосунок створює окремі директрії:

s3://your-bucket-name/users/user_id/posts/

s3://your-bucket-name/users/user_id/avatars/

У застосунку реалізовано генерацію підписних посилань для файлів, які зберігаються на Amazon S3. Ці підписні посилання дозволяють обмежувати доступ до файлів лише авторизованим користувачам та контролювати термін дії посилання.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

Коли користувач потребує доступу до конкретного файлу, ваш застосунок автоматично генерує підписне посилання для цього файлу, використовуючи AWS SDK. Це посилання містить унікальний підпис, який перевіряється Amazon S3 для авторизації користувача, а також обмеження щодо терміну дії посилання.

Після створення підписного посилання, застосунок надісилає його користувачеві, який запросив доступ до файлу. Користувач може використовувати це посилання для отримання доступу до файлу, і Amazon S3 автоматично перевіряє його дійсність перед наданням доступу до файлу.

Цей механізм дозволяє застосунку забезпечувати безпеку та контроль доступу до файлів, що зберігаються на Amazon S3, та надає зручний спосіб обміну файлами з користувачами.

У застосунку реалізовано оптимізацію запитів, щоб зменшити вагу та збільшити швидкість передачі даних. Замість передачі файлів напряму через тіло відповіді, ваш надає лише посилання на ці файли. Це дозволяє значно зменшити розмір та обсяг кожного запиту, оскільки в тілі відповіді передається лише текстова інформація - URL до файла на Amazon S3.

При запиті файлу, клієнт отримує лише URL, а не сам файл. Це дозволяє значно заощадити пропускну здатність мережі та ресурси сервера, оскільки передача великих файлів через мережу може призвести до затримок та перевантаження сервера.

Завдяки цій оптимізації, забезпечується більш швидка та ефективна робота з файлами, зменшуючи витрати на мережевий трафік та збільшуючи продуктивність. Крім того, цей підхід дозволяє збільшити масштабованість додатка, оскільки менше ресурсів потрібно для обробки кожного запиту.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		21

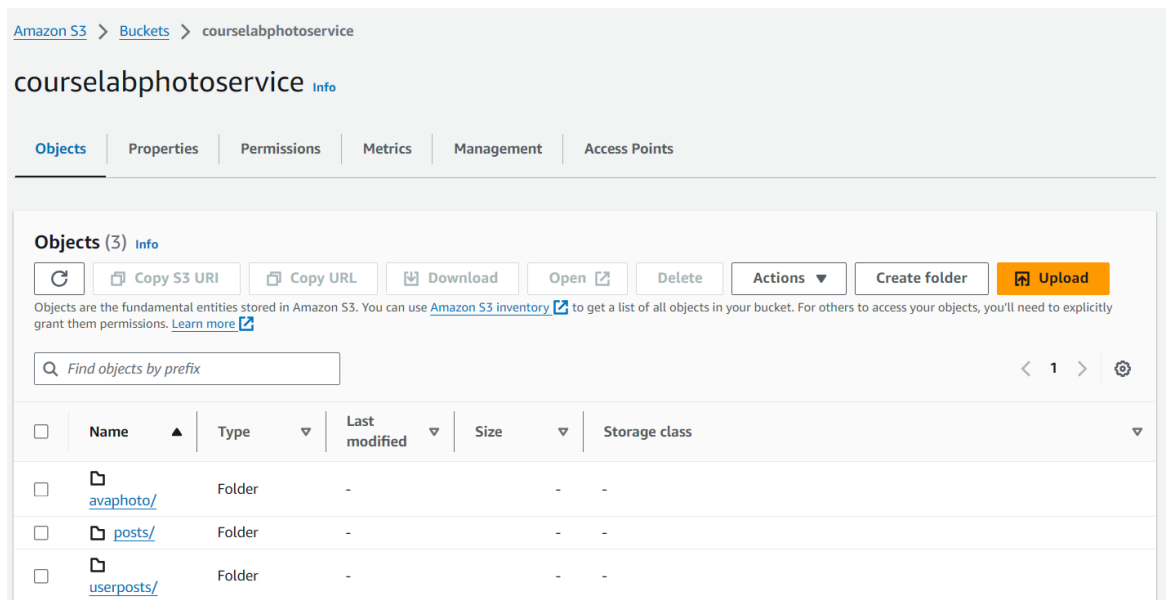


Рисунок 1.3. – Старт роботи контейнерів

RabbitMQ, AMQP

RHPAmqplib - це бібліотека для роботи з RabbitMQ, яка дозволяє РНР-розробникам здійснювати асинхронну комунікацію між додатками за допомогою протоколу AMQP (Advanced Message Queuing Protocol). Вона надає зручний інтерфейс для взаємодії з RabbitMQ, включаючи надсилання та отримання повідомлень, створення черг, керування обмінниками та інші операції. За допомогою RHPAmqplib, розробники можуть легко і надійно інтегрувати асинхронну комунікацію у свої РНР-застосунки та забезпечити ефективну обробку повідомлень.

У проекті використовується API Platform для розробки та управління API. API Platform - це потужний інструмент, який дозволяє розробникам швидко створювати та управляти API для взаємодії з вашим застосунком.

API Platform надає зручний інтерфейс для опису API за допомогою специфікацій, таких як OpenAPI (Swagger). За допомогою цих специфікацій, API автоматично документується та візуалізується в форматі Swagger UI, що дозволяє розробникам та іншим зацікавленим сторонам легко ознайомитися з його функціональністю та можливостями.

Swagger UI - це інтерактивна документація, яка надає зручний інтерфейс для тестування та взаємодії з різними ендпоінтами.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

З використанням API Platform та Swagger я забезпечую зручний та ефективний спосіб розробки, документування та управління моїм API, що дозволяє швидко реагувати на зміни вимог та впроваджувати нові функції у застосунок.

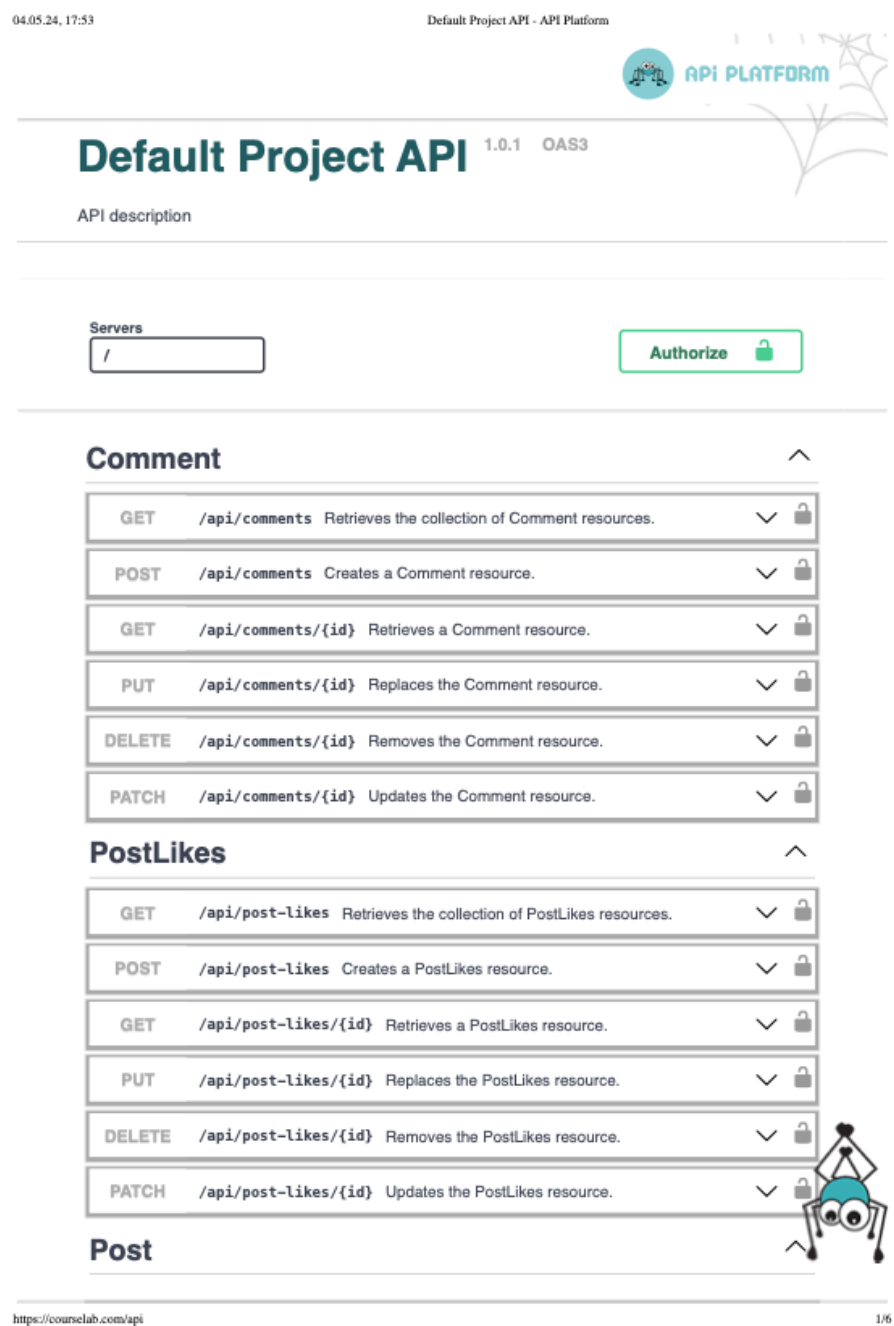


Рисунок 1.4. – Документація Swagger



GET	/api/posts/{id}	Retrieves a Post resource.	✓	🔒
PUT	/api/posts/{id}	Replaces the Post resource.	✓	🔒
DELETE	/api/posts/{id}	Removes the Post resource.	✓	🔒
PATCH	/api/posts/{id}	Updates the Post resource.	✓	🔒

ProfileVisit

GET	/api/profile-visits	Retrieves the collection of ProfileVisit resources.	✓	🔒
POST	/api/profile-visits	Creates a ProfileVisit resource.	✓	🔒
GET	/api/profile-visits/{id}	Retrieves a ProfileVisit resource.	✓	🔒
PUT	/api/profile-visits/{id}	Replaces the ProfileVisit resource.	✓	🔒
DELETE	/api/profile-visits/{id}	Removes the ProfileVisit resource.	✓	🔒
PATCH	/api/profile-visits/{id}	Updates the ProfileVisit resource.	✓	🔒

SavedPosts

GET	/api/saved-posts	Retrieves the collection of SavedPosts resources.	✓	🔒
POST	/api/saved-posts	Creates a SavedPosts resource.	✓	🔒
GET	/api/saved-posts/{id}	Retrieves a SavedPosts resource.	✓	🔒
PUT	/api/saved-posts/{id}	Replaces the SavedPosts resource.	✓	🔒
DELETE	/api/saved-posts/{id}	Removes the SavedPosts resource.	✓	🔒
PATCH	/api/saved-posts/{id}	Updates the SavedPosts resource.	✓	🔒

SubscriptionData

GET	/api/subscription-datas	Retrieves the collection of SubscriptionData resources.	✓	🔒
-----	-------------------------	---	---	---

Рисунок 1.5. – Документація Swagger

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		



/{subscription}				
PUT	/api/subscription-datas/{subscription}	Replaces the SubscriptionData resource.	▼	🔒
DELETE	/api/subscription-datas/{subscription}	Removes the SubscriptionData resource.	▼	🔒
PATCH	/api/subscription-datas/{subscription}	Updates the SubscriptionData resource.	▼	🔒
Subscription				^
GET	/api/subscriptions	Retrieves the collection of Subscription resources.	▼	🔒
POST	/api/subscriptions	Creates a Subscription resource.	▼	🔒
GET	/api/subscriptions/{id}	Retrieves a Subscription resource.	▼	🔒
PUT	/api/subscriptions/{id}	Replaces the Subscription resource.	▼	🔒
DELETE	/api/subscriptions/{id}	Removes the Subscription resource.	▼	🔒
PATCH	/api/subscriptions/{id}	Updates the Subscription resource.	▼	🔒
User				^
GET	/api/users	Retrieves the collection of User resources.	▼	🔒
POST	/api/users	Creates a User resource.	▼	🔒
GET	/api/users/{id}	Retrieves a User resource.	▼	🔒
PUT	/api/users/{id}	Replaces the User resource.	▼	🔒
Schemas				
Comment				
Comment-empty				

<https://courselab.com/api>

3/6



Рисунок 1.6. – Документація Swagger

Детальніший опис роботи дотаку можна знайти за цим посиланням <https://courselab.com/api> , після запуску проекту на локальній машині.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

MYSQL

MySQL та **phpMyAdmin** використовуються разом для керування та адміністрування бази даних у проекті.

1) MySQL:

- **MySQL** - це система управління базами даних (СУБД), яка використовується для зберігання, організації та управління структурованою інформацією.
- Запускається контейнер MySQL, який містить базу даних для застосунку. Встановлюються дані конфігурації, такі як ім'я бази даних, користувачі та паролі через змінні середовища.
- MySQL працює на вказаному порті (у цьому випадку 13306), щоб забезпечити з'єднання з застосунком та іншими сервісами.

2) phpMyAdmin:

- **phpMyAdmin** - це веб-інтерфейс для керування MySQL базами даних. Він дозволяє виконувати різноманітні операції з базою даних, такі як створення, видалення, редагування таблиць, запити SQL, імпорт та експорт даних тощо.
- В середовищі запускається окремий контейнер з phpMyAdmin, який надає доступ до веб-інтерфейсу через веб-браузер.
- Налаштовуються параметри підключення phpMyAdmin до MySQL, такі як хост, порт, ім'я користувача та пароль через змінні середовища.

При використанні MySQL з phpMyAdmin зручно керувати та адмініструвати базу даних через веб-інтерфейс, виконуючи різноманітні завдання безпосередньо у браузері. Така інтеграція полегшує роботу з базою даних та забезпечує зручний спосіб управління вашими даними.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

- 5) **Збереження токена:** клієнт зберігає отриманий JWT токен, наприклад, у локальному сховищі (наприклад, у localStorage або cookies).
- 6) **Використання токена для авторизації:** під час кожного запиту до захищених ресурсів, клієнт додає JWT токен до заголовків запиту. Сервер перевіряє цей токен та надає доступ до ресурсу, якщо токен валідний і містить необхідні дозволи.
- 7) **Оновлення токена:** якщо токен має обмежений термін дії (наприклад, декілька годин), клієнт може оновити його, надславши свій старий токен разом з запитом на оновлення. Сервер перевіряє та оновлює токен, якщо він ще не закінчився.

Завдяки використанню JWT, авторизація стає децентралізованою та масштабованою, оскільки всі необхідні дані про користувача містяться в самому токені, і серверу не потрібно зберігати стан авторизації користувача.

Nelmio Cors Bundle

Nelmio Cors Bundle - це інструмент для Symfony, який дозволяє керувати політикою Cross-Origin Resource Sharing (CORS) у застосунку. CORS - це механізм безпеки веб-браузера, який обмежує запити до ресурсів з інших доменів.

Робота з Nelmio Cors Bundle включає наступні кроки:

- 1) Встановлення та налаштування: спочатку встановлюється Nelmio Cors Bundle у свій Symfony-проект за допомогою Composer, а потім налаштовуєте його конфігурацію в файлі **config/packages/nelmio_cors.yaml**. Тут ви можете визначити дозволені домени, методи запитів, заголовки та інші параметри CORS.
- 2) Перевірка запитів: кожен запит до Symfony-сервісу буде перевіряти Nelmio Cors Bundle, щоб визначити, чи потрібно дозволити запит на основі встановлених правил CORS. Якщо

запит відповідає встановленим умовам, сервер надсилає відповідь зі статусом 200 та необхідними заголовками CORS.

3) Налаштування CORS-заголовків: при потребі Nelmio Cors Bundle може додавати або змінювати CORS-заголовки у відповідях сервера. Це дозволяє налаштувати правила CORS на рівні HTTP-відповіді та контролювати, як браузері взаємодіють з застосунком через JavaScript (React).

4) Управління CORS для конкретних маршрутів: ви також можете налаштувати CORS для конкретних маршрутів у вашому Symfony-застосунку. Це дозволяє гнучко керувати політикою CORS для різних частин застосунку, наприклад, API та статичних ресурсів.

За допомогою Nelmio Cors Bundle застосунок ефективно керує політикою CORS у Symfony-сервісі, забезпечуючи безпечну та надійну взаємодію з іншими доменами та додатками у веб-середовищі.

Конфігурація Composer демонструє вибір правильних версій пакетів для забезпечення стабільності проекту. Також використовуються автоматизовані скрипти для оптимізації роботи додатка, а конфлікти з пакетами Symfony уникнуті завдяки налаштуванню конфліктів.

Усі ці компоненти разом утворюють потужну архітектуру, яка забезпечує ефективну розробку, масштабованість та стабільність веб-додатка.

Висновки до розділу 1

Планування архітектури застосунку враховує сучасні підходи та технології для забезпечення гнучкості, розширюваності та надійності системи. Використання мікросервісної архітектури дозволяє розділити застосунок на окремі компоненти, що спрощує розробку, тестування та масштабування.

Використання Docker контейнерів для кожного мікросервісу забезпечує ізольоване середовище для кожного компонента, що полегшує розгортання та управління. Крім того, Docker-compose дозволяє групувати та керувати цими

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

контейнерами як єдиним застосунком, що спрощує розробку та тестування в локальному середовищі.

Використання брокера повідомлень RabbitMQ для спілкування між сервісами дозволяє забезпечити надійну та ефективну комунікацію, а побудова черг повідомлень з розподілом за допомогою exchangers додає гнучкості та масштабованості до системи.

Узагальнюючи, планування архітектури застосунку відображає ретельне обґрунтування вибору інструментів та технологій для створення сучасного, гнучкого та надійного рішення, яке може ефективно працювати та розвиватися в майбутньому.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-СИСТЕМИ

2.1 Проектування веб-системи

При проектуванні веб-системи було створено діаграму User Way, яка демонструє можливі шляхи користувача в веб-застосунку, починаючи з головної сторінки та описуючи можливі дії та переходи між різними сторінками та функціями. Саме ця діаграма відображає основні дії та переходи між різними сторінками застосунку, що дозволяє користувачам зручно навігувати та використовувати всі функції.

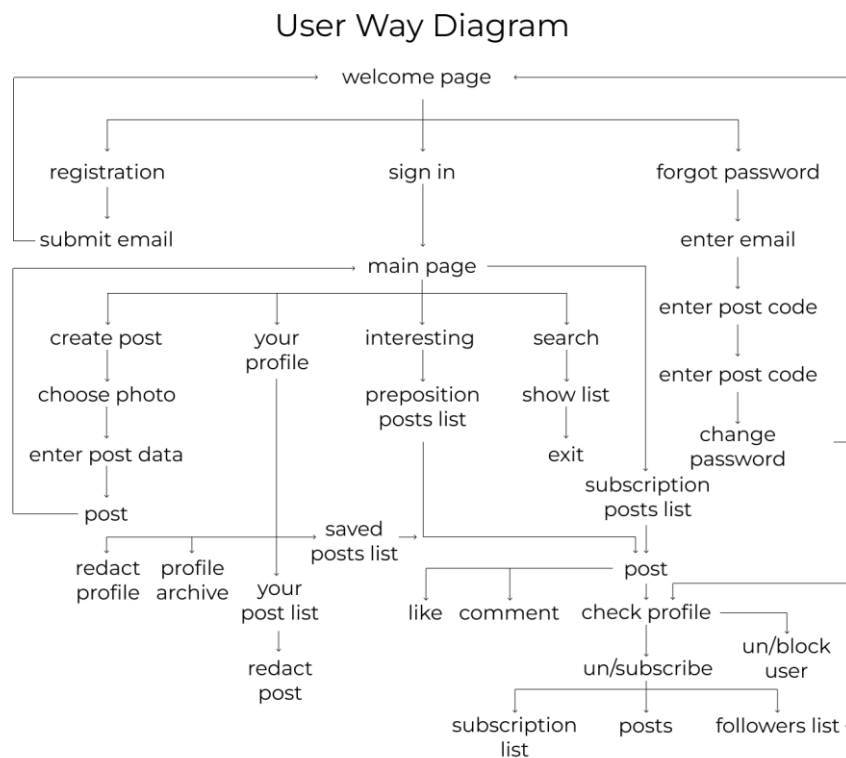


Рисунок 2.1. – User Way діаграма

Ось детальний опис кожного етапу та дій користувача:

1. Welcome Page (Головна сторінка):

- Registration (Реєстрація):

- Заповнення форми реєстрації та подання електронної пошти.
- Sign In (Вхід):
- Вхід у систему.
- Forgot Password (Забули пароль):
- Введення електронної пошти для відновлення пароля.
- Введення коду з пошти.
- Зміна пароля.

2. Main Page (Головна сторінка після входу):

- Create Post (Створення посту):
- Вибір фото.
- Введення даних посту.
- Публікація посту.
- Your Profile (Ваш профіль):
- Перегляд профілю.
- Перегляд архіву профілю.
- Перегляд списку ваших постів.
- Редагування посту.
- Редагування профілю.
- Interesting (Цікаве):
- Перегляд списку запропонованих постів.
- Search (Пошук):
- Введення запиту для пошуку.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

- Перегляд результатів пошуку.
- Вихід із пошуку.
- Saved Posts List (Список збережених постів):
- Перегляд збережених постів.
- Subscription Posts List (Список постів підписок):
- Перегляд постів, на які користувач підписаний.

3. Post Actions (Дії з постами):

- Like (Лайк):
- Лайкання посту.
- Comment (Коментар):
- Додавання коментаря до посту.
- Check Profile (Перевірка профілю):
- Перегляд профілю автора посту.
- Підписка або відписка від користувача.
- Блокування або розблокування користувача.

4. Subscription and Followers Lists (Списки підписок та підписників):

- Subscription List (Список підписок):
- Перегляд списку підписок.
- Followers List (Список підписників):
- Перегляд списку підписників.
- Posts (Пости):- Перегляд постів від підписок.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

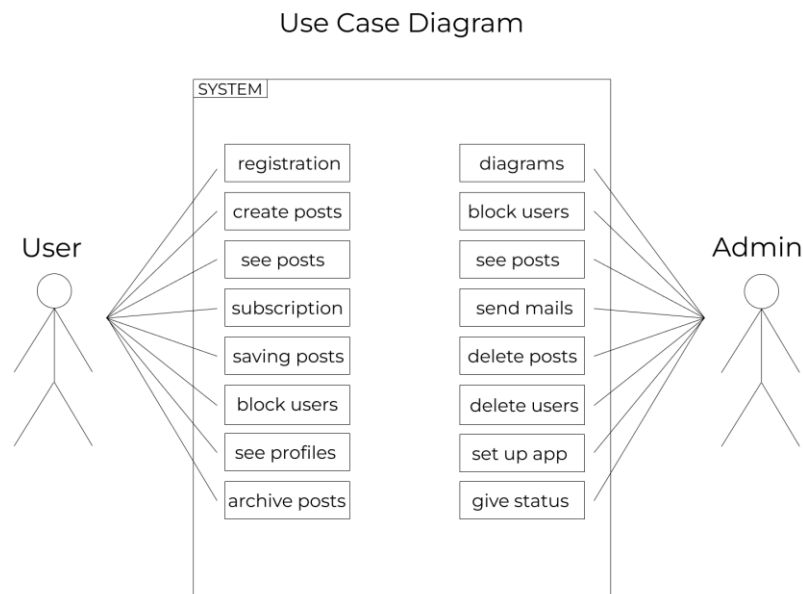


Рисунок 2.2. – Use Case діаграма

Діаграма випадків використання (use case diagram) ілюструє різні функціональні можливості системи та взаємодії двох основних ролей користувачів: **User** (користувач) та **Admin** (адміністратор). Ось детальний опис функцій, доступних кожній ролі:

User (Користувач):

- 1. Registration (Реєстрація):** користувач може зареєструватися в системі.
- 2. Create Posts (Створення постів):** користувач може створювати нові пости.
- 3. See Posts (Перегляд постів):** користувач може переглядати пости.
- 4. Subscription (Підписка):** користувач може підписуватися на інші профілі.

5. Saving Posts (Збереження постів): користувач може зберігати пости для подальшого перегляду.

6. Block Users (Блокування користувачів): користувач може блокувати інших користувачів.

7. See Profiles (Перегляд профілів): користувач може переглядати профілі інших користувачів.

8. Archive Posts (Архівування постів): користувач може архівувати свої пости.

Admin (Адміністратор):

1. Diagrams (Діаграми): адміністратор має доступ до діаграм для аналізу системи.

2. Block Users (Блокування користувачів): адміністратор може блокувати користувачів.

3. See Posts (Перегляд постів): адміністратор може переглядати всі пости.

4. Send Mails (Відправка листів): адміністратор може надсилати електронні листи користувачам.

5. Delete Posts (Видалення постів): адміністратор може видаляти пости.

6. Delete Users (Видалення користувачів): адміністратор може видаляти користувачів.

7. Set Up App (Налаштування застосунку): адміністратор може налаштовувати систему.

8. Give Status (Надання статусу): адміністратор може надавати різні статуси користувачам.

Система

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

Система надає платформу, де реалізовані всі вищезазначені функції, і служить центральним елементом, через який користувачі та адміністратори взаємодіють з відповідними функціональними можливостями.

Взаємодія

- **User** має можливість виконувати ряд дій, спрямованих на взаємодію з контентом та іншими користувачами, такі як реєстрація, створення постів, перегляд постів, підписка, збереження постів, блокування користувачів, перегляд профілів, архівування постів.

- **Admin** має більше прав та можливостей для управління системою та користувачами, включаючи перегляд постів, блокування та видалення користувачів, відправлення листів, налаштування системи, доступ до діаграм та надання статусів.

Діаграма Use Case допомагає зрозуміти основні функціональні можливості системи та розмежувати права та обов'язки звичайних користувачів та адміністраторів.

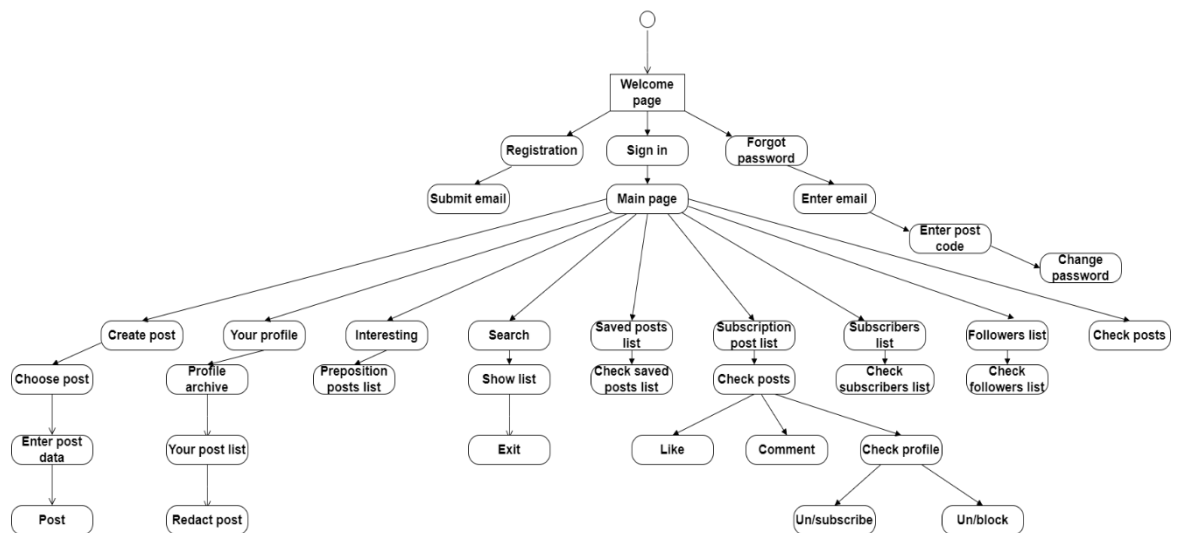


Рисунок 2.3. – Діаграма активності

Діаграма активності відображає процес взаємодії користувача з застосунком, починаючи з головної сторінки та закінчуючи різними діями на

головній сторінці після входу в систему. На головній сторінці користувач має можливість зареєструватися, увійти до системи або відновити пароль, якщо він забутий. Після входу в систему користувач потрапляє на головну сторінку, де доступні різні функції, такі як створення посту, перегляд профілю, цікаві пропозиції, пошук, збережені пости та список підписок.

Кожна дія детально відображена на діаграмі. Наприклад, створення посту включає вибір фото, введення даних посту та публікацію. Перегляд профілю надає можливість переглядати архів профілю, список постів користувача та редагувати профіль. Користувач також може переглядати пости, здійснювати пошук, переглядати збережені пости.

Діаграма також відображає дії з постами, такі як лайк, коментар, перегляд профілю автора посту, підписка або відписка від користувача та блокування або розблокування користувача. Списки підписок та підписників надають можливість переглядати відповідні списки та пости від підписок.

Загалом, діаграма демонструє повний цикл взаємодії користувача з застосунком, починаючи від реєстрації та входу до різних дій на головній сторінці, що забезпечує зручність та ефективність користування застосунком.

2.2 Проектування бази даних

Проектування бази даних – це найважливіший етап розробки застосунку. Під час цього етапу створюються усі зв'язки між даними та проектується основна логіка роботи з даними. Разом з базою даних також проектуються і моделі на основі яких і формується БД з усіма зв'язками за допомогою `php/doctrine/schemas`.

Моделі:

Comments:

- 1) **id:** це унікальний ідентифікатор коментаря. Використовується анотація `#[ORM\Id]`, щоб позначити його як первинний ключ бази

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

даних. `#[ORM\GeneratedValue]` вказує на автоматичне згенерування значення для цього поля, а `#[ORM\Column]` вказує, що воно є стовпцем в базі даних.

- 2) **text:** це текстове поле коментаря. Використовується анотація `#[ORM\Column(type: Types::TEXT)]` для визначення його типу в базі даних як TEXT.
- 3) **parent:** це відношення багато-до-багатьох, яке вказує на батьківські коментарі цього коментаря. Використовується анотація `#[ORM\ManyToOne]`, оскільки коментар може мати багато батьківських коментарів.
- 4) **post:** це відношення багато-до-одного, яке вказує на пов'язаний пост. Використовується анотація `#[ORM\ManyToOne]`, оскільки кожен коментар належить лише одному посту, але один пост може мати багато коментарів.
- 5) **commentator:** це відношення багато-до-одного, яке вказує на користувача, який залишив коментар. Використовується така ж анотація `#[ORM\ManyToOne]`, оскільки кожен коментар належить лише одному користувачеві, але один користувач може залишити багато коментарів.
- 6) **date:** це поле, яке вказує на дату та час створення коментаря. Використовується анотація `#[ORM\Column(type: Types::DATE_MUTABLE)]` для визначення типу даних як DATE_MUTABLE.
- 7) **Конструктор:** у конструкторі створюються об'єкти `ArrayCollection` для колекцій, щоб вони були готові до використання при створенні нового коментаря. Гетери та сетери: Для кожної властивості класу `Comment` визначені методи `get` і `set` для отримання та встановлення значень відповідно.

- 8) **ApiResponse** **анотація:** вказує на те, що цей клас є ресурсом API, а також визначає доступні операції (GET та POST) та вимоги безпеки для цих операцій.
- 9) **Groups** **анотації:** групи серіалізації використовуються для визначення того, які властивості коментаря включати або виключати при серіалізації у відповіді API.

Post:

- 1) **id:** це унікальний ідентифікатор поста. Використовується анотація `#[ORM\Id]`, `#[ORM\GeneratedValue]`, `#[ORM\Column]`.
- 2) **user:** це поле вказує на користувача, який опублікував пост. Використовується анотація `#[ORM\ManyToOne]`, `#[ORM\JoinColumn(nullable: false)]`.
- 3) **imageUrl:** посилання на зображення, яке прикріплене до поста. Використовується анотація `#[ORM\Column(length: 255)]`.
- 4) **caption:** текстовий опис поста. Використовується анотація `#[ORM\Column(type: Types::TEXT, nullable: true)]`.
- 5) **created_at:** дата та час створення поста. Використовується анотація `#[ORM\Column(type: Types::DATE_MUTABLE)]`.
- 6) **postLikes:** колекція лайків поста. Використовується анотація `#[ORM\OneToMany]`.
- 7) **comments:** колекція коментарів до поста. Використовується анотація `#[ORM\OneToMany]`.
- 8) **location:** місцезнаходження, пов'язане з постом. Використовується анотація `#[ORM\Column(length: 255, nullable: true)]`.
- 9) **savedPosts:** колекція збережених постів. Використовується анотація `#[ORM\OneToMany]`.
- 10) **isHided:** поле, що вказує, чи прихований пост. Використовується анотація `#[ORM\Column]`.

- 11) **file:** зображення, яке прикріплене до поста. Використовується анотація `#[ApiProperty]`, `#[Assert\File]`.
- 12) **Конструктор:** ініціалізує всі колекції та стандартні значення, такі як час створення та статус прихованості.
- 13) **Гетери та сетери:** для кожної властивості класу `Post` визначені методи `get` та `set` для отримання та встановлення значень відповідно.

PostLikes:

- 1) **id:** це унікальний ідентифікатор лайку. Використовується анотація `#[ORM\Id]`, `#[ORM\GeneratedValue]`, `#[ORM\Column]`.
- 2) **post:** зв'язок з постом, до якого прикріплений цей лайк. Використовується анотація `#[ORM\ManyToOne]`.
- 3) **user:** зв'язок з користувачем, який поставив лайк. Використовується анотація `#[ORM\ManyToOne]`.
- 4) **date:** дата та час, коли був поставлений лайк. Використовується анотація `#[ORM\Column(type: Types::DATE_MUTABLE)]`.
- 5) **Гетери та сетери:** для кожної властивості класу `PostLikes` визначені методи `get` та `set` для отримання та встановлення значень відповідно.

ProfileVisit:

- 1) **id:** це унікальний ідентифікатор відвідування профілю. Використовується анотація `#[ORM\Id]`, `#[ORM\GeneratedValue]`, `#[ORM\Column]`.
- 2) **visitor:** зв'язок з користувачем, який відвідав профіль. Використовується анотація `#[ORM\ManyToOne]`.
- 3) **currentUser:** зв'язок з користувачем, чий профіль був відвіданий. Використовується анотація `#[ORM\ManyToOne]`.
- 4) **visitDate:** дата та час відвідування профілю. Використовується анотація `#[ORM\Column(type: Types::DATE_MUTABLE)]`.
- 5) **isViewed:** показує, чи був переглянутий цей запис профілю. Використовується анотація `#[ORM\Column]`.

6) **count:** кількість відвідувань профілю. Використовується анотація `#[ORM\Column]`.

7) **Гетери та сетери:** для кожної властивості класу `ProfileVisit` визначені методи `get` та `set` для отримання та встановлення значень відповідно.

SavedPosts:

1) **id:** тип: `int|null`. Використовується для ідентифікації збережених постів.

2) **users:** тип: `Collection|ArrayCollection`. Відношення один до багатьох із моделлю `User`. Представляє користувачів, які зберегли цей пост.

3) **posts:** тип: `Collection|ArrayCollection`. Відношення один до багатьох із моделлю `Post`. Представляє пости, які збережені цим користувачем.

4) **user:** тип: `User|null`. Відношення багато до одного із моделлю `User`. Представляє користувача, який зберіг пост.

5) **post:** тип: `Post|null`. Відношення багато до одного із моделлю `Post`. Представляє пост, який був збережений.

Методи:

1) `__construct()`: ініціалізує властивості `users` та `posts` як нові колекції `ArrayCollection`.

2) `getId()`: повертає ідентифікатор збереженого поста.

3) `getUsers()`: повертає колекцію користувачів, які зберегли цей пост.

4) `addUser(User $user)`: додає користувача до колекції збережених користувачів.

5) `removeUser(User $user)`: видаляє користувача з колекції збережених користувачів.

6) `getPosts()`: повертає колекцію постів, які збережені цим користувачем.

7) `addPost(Post $post)`: додає пост до колекції збережених постів. R

8) `removePost(Post $post)`: видаляє пост з колекції збережених постів.

9) `getUser()`: повертає користувача, який зберіг пост.

10) `setUser(?User $user)`: встановлює користувача, який зберіг пост.

- 11) **getPost()**: повертає пост, який був збережений.
- 12) **setPost(?Post \$post)**: встановлює пост, який був збережений.

Subscription:

- 1) **id**: ідентифікатор підписки.
- 2) **user**: представляє користувача, який підписався на пост.
- 3) **post**: представляє пост, на який підписалися.

Методи:

- 1) **getId()**: повертає ID підписки.
- 2) **getUser()**: повертає користувача, який підписався.
- 3) **setUser()**: встановлює користувача, який підписався.
- 4) **getPost()**: повертає пост, на який підписалися.
- 5) **setPost()**: встановлює пост, на який підписалися.

User:

Ця модель User представляє користувача системи. Вона містить інформацію про користувача, таку як електронну пошту, пароль, ролі, номер телефону, статус підтвердження, нікнейм, біографію, URL профілю, URL веб-сайту, дату народження, дату реєстрації, останній вхід, країну, місто, IP-адресу клієнта, повне ім'я тощо.

Основні особливості моделі:

Поля:

- 1) **id**: унікальний ідентифікатор користувача, представлений за допомогою класу Uuid.
- 2) **email**: електронна пошта користувача, унікальна. **roles**: Ролі користувача у системі.
- 3) **password**: захешований пароль користувача.
- 4) **phoneNumber**: номер телефону користувача, унікальний.
- 5) **isVerified**: статус підтвердження облікового запису. **verification**: Зв'язок з підтвердженням облікового запису.

- 6) **nickname**: нікнейм користувача, унікальний. **bio**: Біографія користувача.
- 7) **profilePictureUrl**: URL зображення профілю користувача.
- 8) **websiteUrl**: URL веб-сайту користувача.
- 9) **birthDate**: дата народження користувача.
- 10) **registrationDate**: дата реєстрації користувача.
- 11) **lastLogin**: дата останнього входу користувача.
- 12) **country**: країна проживання користувача.
- 13) **city**: місто проживання користувача.
- 14) **subscriptions**: колекція підписок користувача на інших користувачів.
- 15) **isSubscribers**: колекція підписників користувача, тобто тих, хто підписався на цього користувача.
- 16) **isPrivate**: статус приватності облікового запису.
- 17) **profileVisits**: колекція відвідувань профілю користувачем.
- 18) **posts**: колекція постів, які створив користувач.
- 19) **postLikes**: колекція лайків, які користувач поставив постам.
- 20) **clientIP**: IP-адреса клієнта користувача.
- 21) **fullName**: повне ім'я користувача.
- 22) **savedPosts**: колекція збережених постів користувача.

Методи:

- Методи для додавання та видалення підписок і підписників.
- Методи для додавання та видалення відвідувань профілю, постів та лайків.
- Методи для управління іншими характеристиками користувача, такими як електронна пошта, пароль, нікнейм тощо.

Ця модель представляє базовий функціонал користувача системи і може бути розширена або модифікована згідно з конкретними потребами застосунку.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

Структура бази даних

Ця база даних має такі таблиці:

1) comment_comment:

- comment_source (int, NOT NULL): посилання на коментар, що породив інший коментар.
- comment_target (int, NOT NULL): посилання на коментар, який був відповіддю.

2) post:

- id (int, NOT NULL): ідентифікатор поста.
- user_id (binary(16), NOT NULL): ідентифікатор користувача, який опублікував пост.
- image_url (varchar(255), NOT NULL): URL зображення, що додається до поста.
- caption (longtext): підпис до поста.
- location (varchar(255)): місце, де був створений пост.
- is_hided (tinyint(1), NOT NULL): прапорець, що вказує, чи прихований пост.
- created_at (date, NOT NULL): дата створення поста.

3) post_likes:

- id (int, NOT NULL): ідентифікатор лайку.
- post_id (int): посилання на пост, який був вподобаний.
- user_id (binary(16)): ідентифікатор користувача, який вподобав пост.
- date (date, NOT NULL): дата, коли був доданий лайк.

4) profile_visit:

- id (int, NOT NULL): ідентифікатор відвідування профілю.
- visitor_id (binary(16)): ідентифікатор користувача-відвідувача.
- visit_date (date, NOT NULL): дата відвідування.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

- `is_viewed` (tinyint(1), NOT NULL): прапорець, що показує, чи було відвідування переглянуте.
- `count` (int, NOT NULL): лічильник кількості відвідувань.

5) `saved_posts`:

- `id` (int, NOT NULL): ідентифікатор збереженого поста.
- `user_id` (binary(16)): ідентифікатор користувача, який зберіг пост.
- `post_id` (int): посилання на збережений пост.

6) `subscription`:

- `id` (int, NOT NULL): ідентифікатор підписки.
- `follower_id` (binary(16)): ідентифікатор користувача-підписника.
- `subscription_status` (tinyint(1), NOT NULL): статус підписки (активна/неактивна).
- `leader_id` (binary(16)): ідентифікатор користувача, на якого підписуються.

7) `subscription_data`:

- `subscription_id` (int, NOT NULL): посилання на ідентифікатор підписки.
- `is_subscribed` (tinyint(1), NOT NULL): прапорець, що показує, чи активна підписка.
- `subscription_date` (date, NOT NULL): дата початку підписки.
- `dissubscription_date` (date): дата відписки від підписки (якщо така є).

8) `user`:

- `id` (binary(16), NOT NULL): ідентифікатор користувача.
- `email` (varchar(180), NOT NULL): email користувача.
- `roles` (json, NOT NULL): ролі користувача.
- `password` (varchar(255), NOT NULL): пароль користувача.

- phone_number (varchar(255), NOT NULL): номер телефону користувача.
- is_verified (tinyint(1), NOT NULL): прапорець, що вказує, чи підтверджений аккаунт.
- nickname (varchar(255), NOT NULL): нікнейм користувача.
- bio (longtext): біографія користувача.
- profile_picture_url (varchar(255)): URL профільного зображення користувача.
- website_url (varchar(255)): URL веб-сайту користувача.
- birth_date (date, NOT NULL): дата народження користувача.
- registration_date (date, NOT NULL): дата реєстрації користувача.
- last_login (datetime): дата останнього входу в систему.
- country (varchar(255)): країна користувача.
- city (varchar(255)): місто користувача.
- is_private (tinyint(1)): прапорець, що вказує, чи є профіль приватним.
- client_ip (varchar(255)): IP-адреса клієнта.
- full_name (varchar(255), NOT NULL): повне ім'я користувача.

9) verification:

- id (int, NOT NULL): ідентифікатор підтвердження.
- user_id (binary(16), NOT NULL): ідентифікатор користувача, якому потрібно підтвердити обліковий запис.
- code (int, NOT NULL): код підтвердження.

Ця структура бази даних представляє соціальну мережу і включає таблиці для збереження даних про користувачів, їхні пости, коментарі, лайки, підписки, відвідування профілів та інші зв'язки між ними. Кожна таблиця має відповідні поля для зберігання інформації про користувачів, пости, їхні

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		46

взаємодії та інші дії в мережі. Відносини між таблицями встановлені через зовнішні ключі, що дозволяє забезпечити консистентність даних та зв'язок між різними елементами системи.

Адміністрування бази даних

На рівні бази даних є багато запобіжних засобів безпеки, головний з яких – це хешування паролів. Всі паролі хешуються на рівні бекенду, а потім зберігаються в БД. Використання UUID також є один з засобів безпеки збереження даних. Також на рівні БД встановлено декілька користувачів з різними рівнями доступу до даних, аби легко роздавати права різним сервісам, на перегляд, видалення чи редагування даних.

Огляд облікових записів користувача

☐	Ім'я користувача	Ім'я хосту	Пароль	Глобальні права	Надати	Дія
☐	courselab-user	%	Так	USAGE	Hi	Редагувати привілеї Експорт Заблокувати
☐	mysql.infoschema	localhost	Так	SELECT	Hi	Редагувати привілеї Експорт Розблокувати
☐	mysql.session	localhost	Так	SHUTDOWN, SUPER	Hi	Редагувати привілеї Експорт Розблокувати
☐	mysql.sys	localhost	Так	USAGE	Hi	Редагувати привілеї Експорт Розблокувати
☐	root	%	Так	ALL PRIVILEGES	Так	Редагувати привілеї Експорт Заблокувати
☐	root	localhost	Так	ALL PRIVILEGES	Так	Редагувати привілеї Експорт Заблокувати

☐ Позначити все Вибрані: Експорт

Рисунок 2.4. – Облікові записи БД

id (DC2Type:uuid)	email	roles	password	phone_number	is_verified
0x1eec03e2625d6860bd5efb386aaecc6d	user@gmail.com	["ROLE_ADMIN"]	\$2y\$13\$006YN7hOm2rcKTNVlxwZTOAXbYo/qdm7Y0i6iQuy6rG...	380674110697	1
0x1eccc4c1e42a6ddea9c66d04bf1649f	lymph@gmail.com	["ROLE_USER"]	\$2y\$13\$03lSnRq/pLP/5.9hgn9JveCJ0b/zZZp.qZXzwgTg54K...	380674110691	1
0x1eed81f5524a64549e6c779840806609	efwef@gmail.com	["ROLE_USER"]	\$2y\$13\$G0714r2mapa15FNDpQdk.f1QlvMKNPyodYPlr.GwA9...	213e	0
0x1eeda0b3466664c9202779364985532	dsfdfs@gmail.com	["ROLE_USER"]	\$2y\$13\$RNKPC.QVcV3iuJPiGr4ZluPGFu/xRvvywvSrSc2iAr2...	4345342342	0
0x1eedb2a5ea0b699ca4f461bd0372a17e	google@gmail.com	["ROLE_USER"]	\$2y\$13\$uxVEwT7iA6xCKtUmHWIUZemp7bsbYt2w/PnTj4Fr8AS...	3453453453	1
0x1eee614bb0065c69b66630b134e6741	forcas@gmail.com	["ROLE_USER"]	\$2y\$13\$9mPZLFUnFD4NpkFziefY5OVc8phztPCDEFktrvP1Erp...	89347747483	1
0x1eee7b5697816daaa4163938cfb5d8a2	andrew2@gmail.com	["ROLE_USER"]	\$2y\$13\$YDN2pUWBk4ewFJSGWshdKu8ViSyBa12NF4DkiVldi6...	5435784654	1
0x1eee845641d16b22bb8174e96454671	sdfjdsfjh@gmail.com	["ROLE_GUEST"]	\$2y\$13\$IzH5sDxj/prksr/GODkTxusurK1019Jj6esUwYM1A.e...	4637643763467	0
0x1eeea00b3466664c9202779364985532	myAccount@gmail.com	["ROLE_USER"]	\$2y\$13\$3mVGA8iaAfm8ZLM8/qOcyekPlyJC4b6OP37lacRLlNW...	43847384	1
0x1eef75c64de46092a259198578c8f56d	yurchenko.ale@gmail.com	["ROLE_USER"]	\$2y\$13\$5G04RqdGQ99wSaJCnyyjk.Dq.aWJOgi11tRZ1WKsZl...	380976395252	1
0x1eefa2baf296914a05f1f62f451e582	helloworld@gmail.com	["ROLE_USER"]	\$2y\$13\$svy7GJ6FdGZQW8R3cRePr.rApjP64KjXPKthzt3HPJV...	54545524343	1
0x1eefb4089346476aa7439e171fab4e0	romashuk@gmail.com	["ROLE_USER"]	\$2y\$13\$itUN84q5.lvBOzK4SaJJJ.lmLnPWiQSRte1n7hFV5F7...	484389389439	1
0x1ef00080204b664aa3116f1072d1ead	anton@gmail.com	["ROLE_USER"]	\$2y\$13\$5xs5JyyLUepHs.QvwN.EbO4hP5J8KJvwBmSURUgnubN...	43844897347	1
0x1ef000b5ab376b48b4474986c4aad462	you@gmail.com	["ROLE_USER"]	\$2y\$13\$H7P38oFSPLC244RuEV4YXeusy3sba/sbdEEfwpkzDw...	309384898444	1
0x1ef0a5f4a7a46460b1a313e2fab728f9	root@gmail.com	["ROLE_USER"]	\$2y\$13\$Q1TfqcisMxnhfSRDCSWyOgdWV.aQX5DjFxp9Wml.R2...	3456789032	1
0x1ef0aace4a276f4cb3e6e1512c3249b4	newUser@gmail.com	["ROLE_GUEST"]	\$2y\$13\$duOQY.twt8ZZJhskUUVWE0giSjMT.TB1StuyH63vh1...	380674238438	0

Рисунок 2.5. – Представлення збережених даних

2.3 Розробка логічних модулів

Згалом в проекті реалізовано 2 логічних модуля, а саме:

- 1) Модуль стиснення зображень.
- 2) Модуль збереження файлів та зображень.

Модуль стиснення зображень

Почнемо з верхніх шарів застосунку. В першу чергу користувач завантажує зображення через візуальний скорпер на клієнській частині.

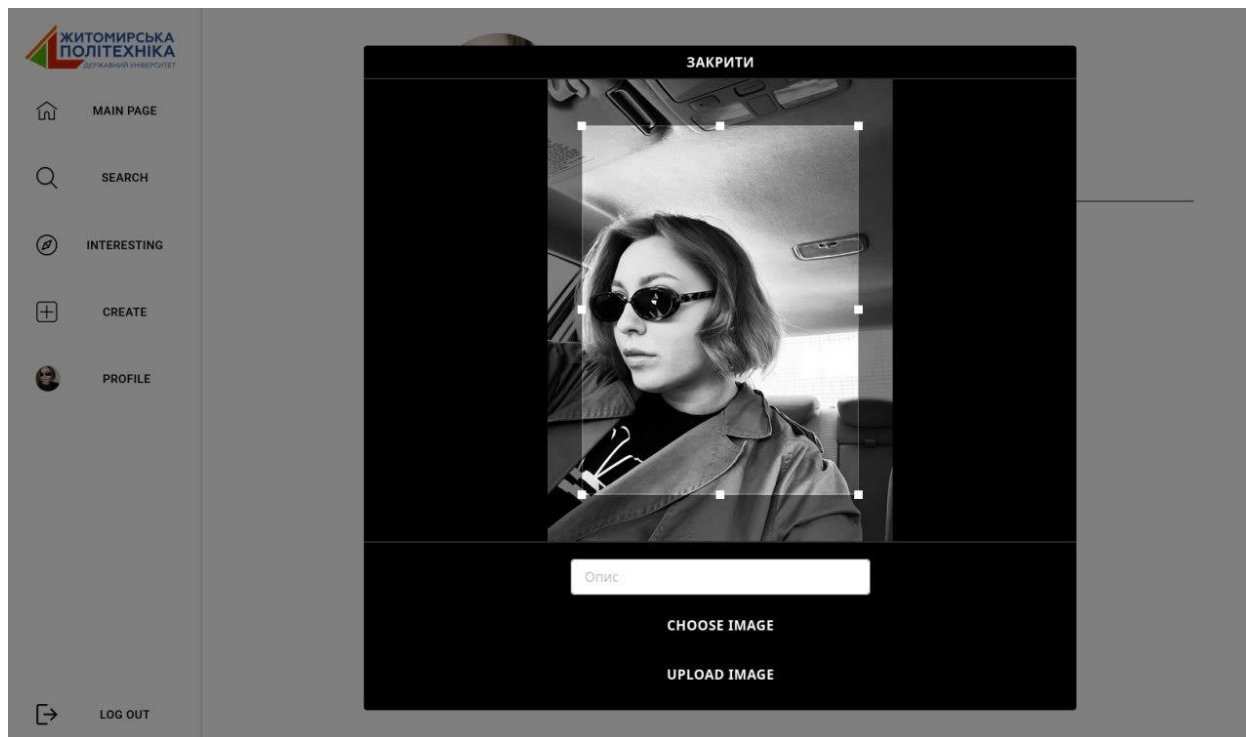


Рисунок 2.6. – Cropper

Наступним кроком користувач обирає розмір, розміщення, та приближення картинки.

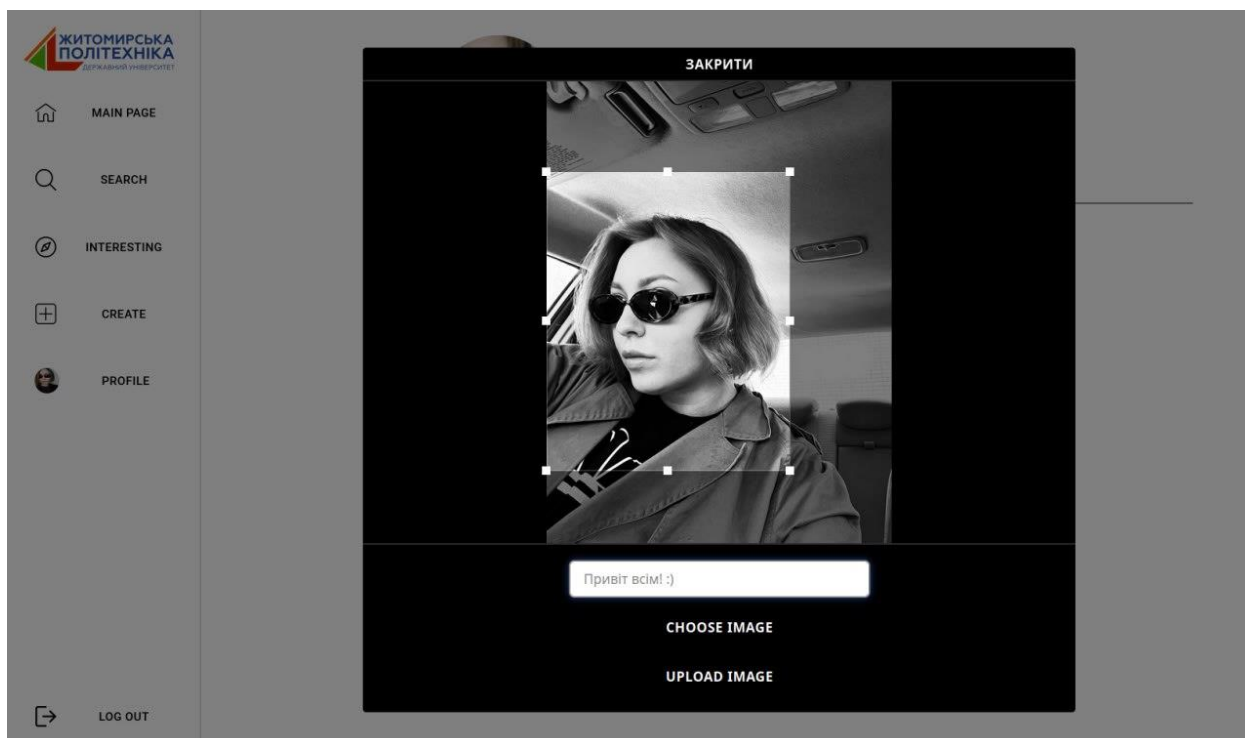


Рисунок 2.7. – Cropper

Після чого фотографія передається на серверну частину, де потрапляє в post контроллер на цей роут:

```
#[Route(path: "/userPost", name: "postUserPost", methods:
["POST"])]
#[IsGranted("ROLE_USER")]
public function postUserPost(Request $request): JsonResponse

    $photo = $this->image->open($uploadedFile);
    $photo->crop(new Point($x, $y), new Box($width, $height));
    $photo->resize(new Box($myWidth, $myHeight));
    $myPhoto = $photo->get('jpg');

    $fileData = $this->AWSS3Service->uploadUserPost($uploadedFile,
    $myPhoto, "userposts", $user->getUsername());

    return new JsonResponse([$fileData["Title"]],
    Response::HTTP_OK);
```

В середині якого картинка обрізається за допомогою компонентів Imagine та ImageService написаним вручну. ImageService на всіх приймає данні

про картинку. Розмір, позицію на екрані, та віддає потрібні параметри для обрізання картинки.

Також на рівні Imagine картинка стискається, завдяки чому сайт завантажується дуже швидко. Після всіх операцій над картинкою, я передаю картинку на AWS service, де вказую, що це за картинка, якому користувачеві вона належить та в яку директорію його покласти. S3 сервіс, який також написаний своїми руками автоматично, створює шлях картинки до її майбутнього місця положення та прив’язки до користувача в S3 корзині.

Після всіх цих дій користувача направляє на головну сто рінку, звідки React відправляє запит на серверну частину для отримання підписного посилання на новостворений допис. Наш запит попадає знову в AWS service, де створюється підписне посилання на потрібний нам файл. Після чого на клієнтську частину приходить масив з даними про найновіші пости.

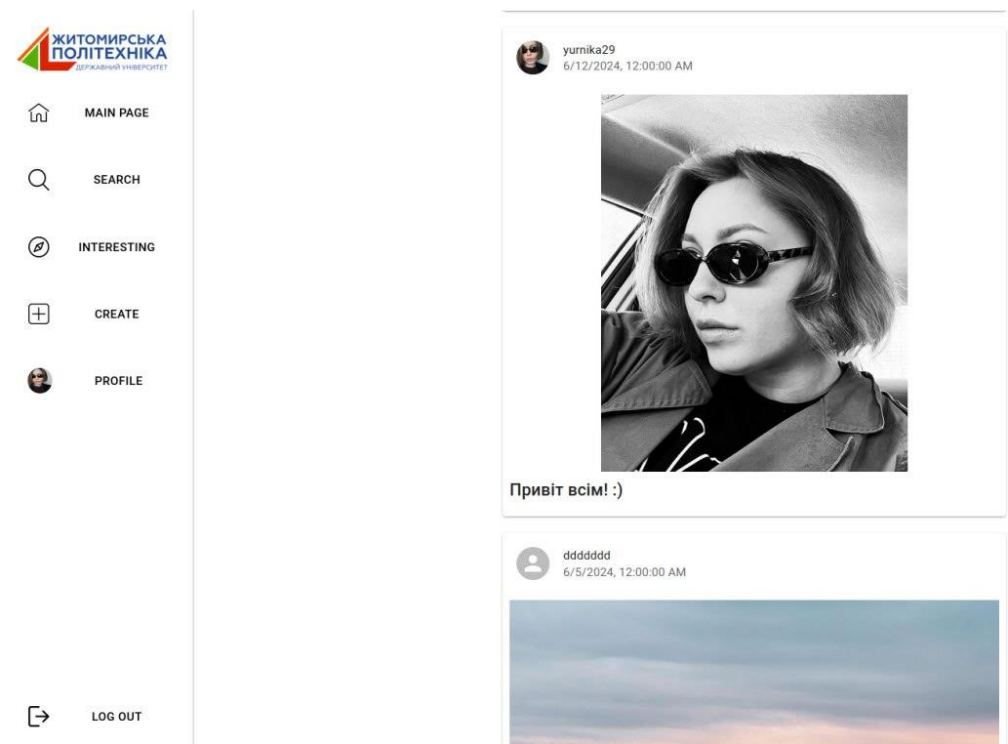


Рисунок 2.8. – Новий пост

Зараз можемо побачити розмір завантажувано зображення. Розмір необробленого зображення сягає 184.4 кілобайт у форматі jpeg.



Рисунок 2.9. – Розмір зображення

Після обробки нашим стискачем, розмір зображення зменшується до 100-600 кбайт. Назва зображення змінюється на захешовану назву, та зберігається на S3. Після чого в локальній базі даних застосунок зберігає захешовану назву в таблиці posts.

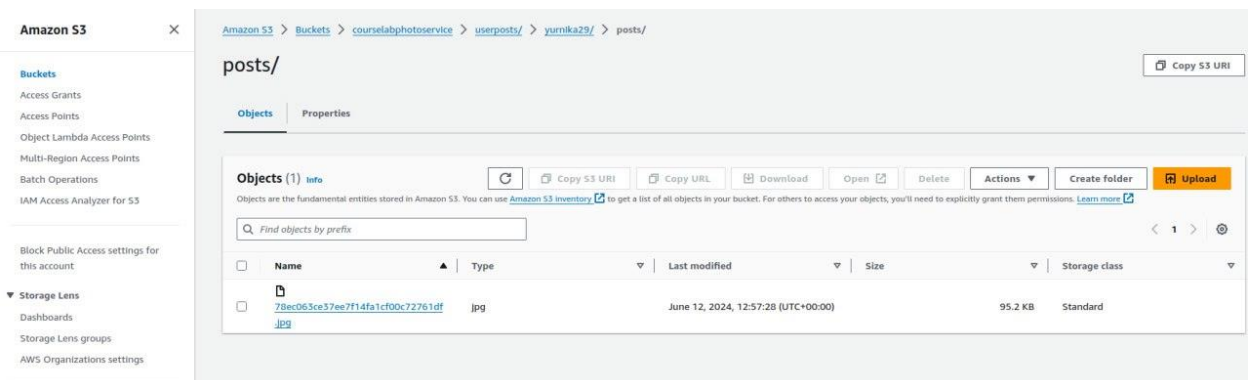


Рисунок 2.10. – Розміри оброблених зображень

Модуль збереження файлів та зображень

При розробці такого проекту, дуже важко було прийняти рішення, щодо збереження даних в безпечному вигляді та, можливістю роздавати доступ до файлів на певний проміжок часу, певним користувачам. Було проведено величезну роботу, щодо написання цього модуля, і в решті було прийнято рішення використовувати AWS S3. За допомогою Simple Storage Service, дуже легко видавати права на файли на певним проміжком часу. При створенні

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

підписного посилання вказується шлях до файлу в бакеті нашого продукту, час на який буде видано це посилання та формується масив інформації про цей об'єкт. На рівні AWS створюється умовний користувач, якому надається доступ на видачу підписних посилань, також налаштовується автоматична робота цього користувача, який працює за певними правилами. Для авторизації на серверах AWS використовуються ключі secret та public keys. Також необхідно передати версію AWS та регіон в якому знаходяться наші файли. Таким чином описується з'єднання з AWS сервером:

```
aws:
  version: latest
  region: eu-north-1
  credentials:
    key: "%env(AWS_KEY)%"
    secret: "%env(AWS_SECRET)%"
```

Висновки до розділу 2

З викладеної інформації про модулі реалізації бази даних та сервісів застосунку фотосервісу можна зробити декілька висновків.

- 1) Засоби безпеки бази даних: використання хешування паролів на рівні бекенду є ефективним засобом захисту конфіденційності користувачів. Встановлення різних рівнів доступу до даних для різних користувачів також допомагає у забезпеченні безпеки.
- 2) Модуль стиснення зображень: цей модуль дозволяє ефективно оптимізувати розмір зображень, що покращує швидкодію застосунку та економить простір на сервері. Використання власних компонент та AWS S3 для збереження зображень підвищує надійність та масштабованість системи.
- 3) Модуль збереження файлів та зображень: використання AWS S3 для збереження файлів дозволяє легко керувати доступом до них та забезпечує їх безпеку. Генерація підписних посилань для обмеження доступу є ефективним рішенням з точки зору безпеки.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

Загалом, реалізація бази даних та сервісів застосунку фотосервісу побудована на високому рівні безпеки та ефективності, що сприяє якості та зручності користування застосунком.

Також було розроблено діаграми Use Case та User Way, які ілюструють можливі шляхи користувача у веб-застосунку та різні функціональні можливості системи. Діаграма Use Case розмежовує права та обов'язки користувачів та адміністраторів, що допомагає зрозуміти основні функціональні можливості системи та взаємодію між різними ролями.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ЗАПУСК ТА НАЛАШТУВАННЯ ЗАСТОСУНКУ

3.1 Порядок встановлення та налаштування системи

Для запуску застосунку знадобиться будь-яка операційна система, docker, docker-compose та git. Рекомендовано використовувати Linux задля ефективності роботи застосунку. У першу чергу встановлюємо усі вище названі програми та клонуємо гіт репозиторій. Після чого в директорії з'явиться нова директорія diploma-social-photo-service. В ній будуть знаходитися такі дані:

```
admin@MacBook-Pro-admin-2 diploma-social-photo-service % ls -la
total 64
drwxr-xr-x 17 admin staff 544 May  4 16:59 .
drwxr-xr-x  4 admin staff 128 Jan 31 14:51 ..
-rwxr-xr-x@ 1 admin staff 768 Apr 21 16:54 .env
drwxr-xr-x@ 12 admin staff 384 May  5 12:25 .git
-rwxr-xr-x@ 1 admin staff 125 Mar  3 21:41 .gitignore
drwxr-xr-x 12 admin staff 384 May  2 00:42 .idea
-rwxr-xr-x  1 admin staff 1683 Jan 31 14:51 README.md
drwxr-xr-x 16 admin staff 512 Apr 21 20:50 api
drwxr-xr-x  2 admin staff  64 Jan 31 14:56 autocluster-0.4.1.ez
-rwxr-xr-x  1 admin staff 344 Jan 31 14:51 clear_cache.sh
drwxr-xr-x  7 admin staff 224 Jan 31 14:51 docker
-rwxr-xr-x@ 1 admin staff 2718 May  4 16:59 docker-compose.local.yaml
-rwxr-xr-x  1 admin staff 880 Jan 31 14:51 docker-compose.yaml
drwxr-xr-x  4 admin staff 128 Jan 31 14:51 environment_data
drwxr-xr-x 12 admin staff 384 May  4 17:00 frontend
-rwxr-xr-x  1 admin staff 611 Jan 31 14:51 local_recreate_docker.sh
-rwxr-xr-x  1 admin staff 1544 Jan 31 14:51 prepare_os.sh
admin@MacBook-Pro-admin-2 diploma-social-photo-service %
```

Рисунок 3.1. – Список перелічених файлів та директорій.

Головними файлами для запуску проекту є: README.md, prepare_os.sh та local_recreate_docker.sh. Якщо у вас встановлено будь-який linux дистрибутив ви можете просто виконати команду ./prepare_os.sh , після якої ваш сервер буде готовий до розгортання проекту.

Після підготовки локальної машини необхідно запустити проєкт. З цим нам допоможе ./ local_recreate_docker.sh . Він зупинить всі контейнери, якщо якісь були в процесі виконання та почне виконання команди:

```
docker-compose -f docker-compose.yaml -f docker-  
compose.local.yaml up --build -d
```

```
admin@MacBook-Pro-admin-2 diploma-social-photo-service % ./local_recreate_docker.sh  
-e =====  
-e =====STARTING REBUILD PROCESS=====  
-e =====  
-e  
1a2a930a63ff  
7e519e642a8f  
9cab3bb1523c  
e21cfc87b2e6  
0aa6e3ae8789  
c79625a12001  
5b167d631fad  
1a2a930a63ff  
7e519e642a8f  
9cab3bb1523c  
e21cfc87b2e6  
0aa6e3ae8789
```

Рисунок 3.2. – Початок запуску проекту

```
[+] Building 130.4s (35/38)                                docker-container:recurring_archimedes  
=> [nginx] exporting to docker image format                0.2s  
=> => exporting layers                                     0.0s  
=> => exporting manifest sha256:dcfb190936c34d38e0d679584435bbf51861576d1435df8414e0c4b7d8aa1b1e 0.0s  
=> => exporting config sha256:802c47e07449e87bde53e158312aba3401d3e9d2184323e88de02f3477b19bce 0.0s  
=> => sending tarball                                       0.2s  
=> [nginx nginx] importing to docker                       0.0s  
=> [frontend 5/9] RUN npm cache clean -f                   2.5s  
=> [frontend 6/9] RUN npm cache verify                     0.4s  
=> [frontend 7/9] RUN npm install -g react-scripts         79.8s  
=> => # longer maintained. Please use @babel/plugin-transform-nullish-coalescing-operator instead.  
=> => # npm WARN deprecated @babel/plugin-proposal-class-properties@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer mai  
=> => # ntained. Please use @babel/plugin-transform-class-properties instead.  
=> => # npm WARN deprecated abab@2.0.6: Use your platform's native atob() and btoa() methods instead  
=> => # npm WARN deprecated domexception@2.0.1: Use your platform's native DOMException instead  
=> => # npm WARN deprecated w3c-hr-time@1.0.2: Use your platform's native performance.now() and performance.timeOrigin.
```

Рисунок 3.3. – Процес завантаження встановлених компонентів

```
✓ Network diploma-social-photo-service_courselab Created  
✓ Container courselab-mysql Started  
✓ Container courselab-api Started  
✓ Container courselab-frontend Started  
✓ Container courselab-rabbitmq Started  
✓ Container courselab-phpmyadmin Started  
✓ Container courselab-nginx Started  
admin@MacBook-Pro-admin-2 diploma-social-photo-service %
```

Рисунок 3.4. – Успішний запуск проекту

Команди docker автоматично запускають застосунок, завантажують всі необхідні скрипти та відвантажують усі бекапи та конфігурації застосунку.

Також на рівні docker відбувається абсолютно все логування процесів застосунку, та відображення можливий проблем.

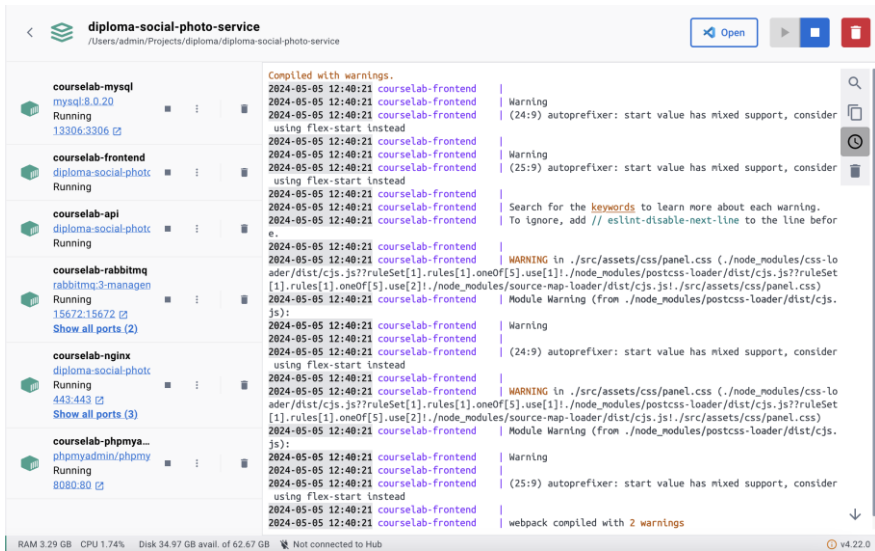


Рисунок 3.5. – Логування процесів застосунку

На панелі можемо побачити перелічені контейнери, та можемо побачити системні записи по кожному з них. Також можемо подивитися перелік збережених статичних даних та конфігурацій та їх версії.

mysql	VOLUME	55 minutes ago	drwxr-xr-x
ib_16384_0.dbiwr	VOLUME	192 kB	55 minutes ago
ib_16384_1.dbiwr	VOLUME	8.2 MB	2 months ago
innodb_temp	VOLUME	55 minutes ago	drwxr-xr-x
auto.cnf	VOLUME	56 Bytes	3 months ago
binlog.000013	VOLUME	33.9 kB	28 days ago
binlog.000014	VOLUME	209.1 kB	21 days ago
binlog.000015	VOLUME	369.5 kB	20 days ago
binlog.000016	VOLUME	284.5 kB	19 days ago
binlog.000017	VOLUME	427.2 kB	17 days ago
binlog.000018	VOLUME	728.5 kB	14 days ago
binlog.000019	VOLUME	54.1 kB	6 days ago
binlog.000020	VOLUME	18.8 kB	2 days ago
binlog.000021	VOLUME	179 Bytes	21 hours ago
binlog.000022	VOLUME	179 Bytes	21 hours ago
binlog.000023	VOLUME	74.8 kB	1 hour ago

Рисунок 3.6. – Конфігурації MySQL

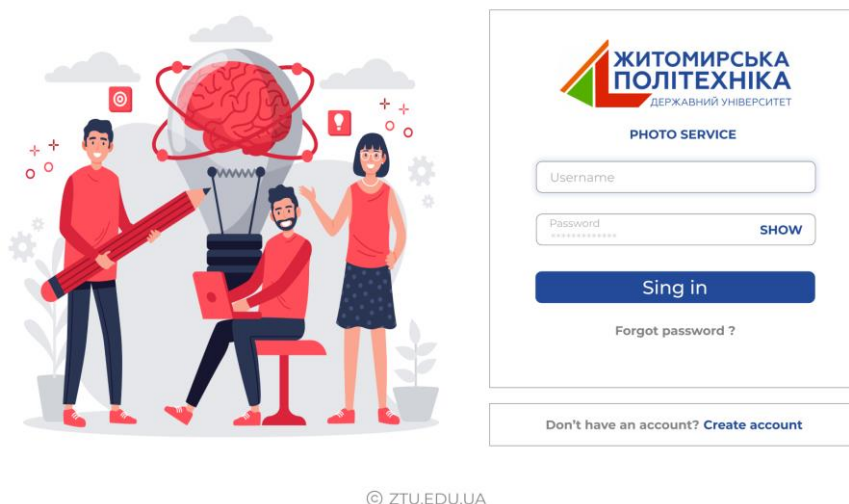
Документацію стосовно прямого запуску проекту можна знайти в README.md файлі на репозиторії застосунку.

https://gitlab.com/kn201_mv/diplomaphotoservice

3.2 Інтерфейс

Перед початком створення програми було створено дизайн макет певних сторінок сайту. В макеті було використано основні кольори Державного університету “Житомирська політехніка”. Також було дотримано усіх вимог до розміщення логотипів, кольорів, шрифтів та інше. Розробка дизайну відбувалася за допомогою програми Figma.

Також в системі авторизації задіяно декілька під сервісів. Lexik JSON web token authentication bundle – це основний компонент при взаємодії з ролями користувача. В даній системі використовується RBAC (Role Based Access Control) підхід розподілення прав доступу до різних end-points.



© ZTU.EDU.UA

Рисунок 3.7. – Початкова сторінка неавторизованого користувача

Якщо користувач не має акаунту в застосунку, йому необхідно створити профіль. Реєстрація складається з двох етапів, перший етап заповнення обов’язкових полів. Другим етапом користувач заповнює інформацію про себе, для зручності користування застосунку.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		



PHOTO SERVICE

 [Show](#)

Рисунок 3.8. – Початкова сторінка реєстрації

Якщо користувач ввів пароль, або не валідну пошту, то користувачеві висвічується toast, який оповістить користувача про проблему та не дасть перейти до наступного етапу реєстрації.



PHOTO SERVICE

© ZTU.EDU.UA

Рисунок 3.9. – Другий етап реєстрації

На цьому етапі користувач заповнює інформацію про себе (дата народження, місто, повне ім'я та країну).

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

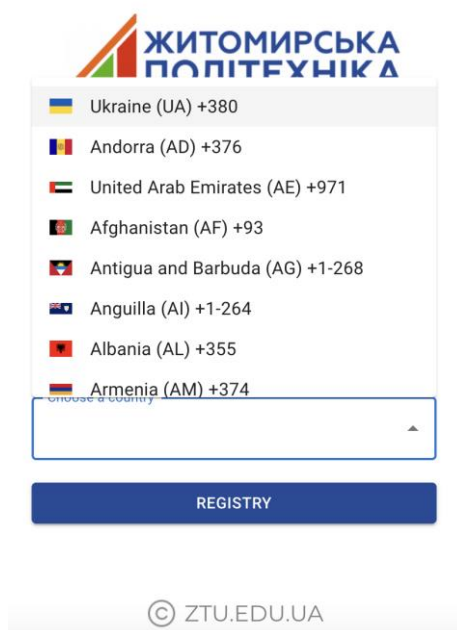


Рисунок 3.10. – Список перелічених країн

Користувач має доступ до вибору будь якої країни. Україна встановлена за замовчуванням.

Рисунок 3.11. – Фінальний етап реєстрації

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		59

При реєстрації видається JWT, що містить в своїй структурі такі дані: роль користувача, його логін та час його існування. Після натискання на клавішу реєстрації створюється користувач з роллю гість.

Система підписок

Зараз реалізовано свою систему підписок на користувачів. Коли користувач підписується на іншого користувача, він матиме доступ до перегляду дописів користувача в стрічці. На перспективу розвитку також реалізовано систему перегляду користувацьких дописів , якщо акаунт є приватним.

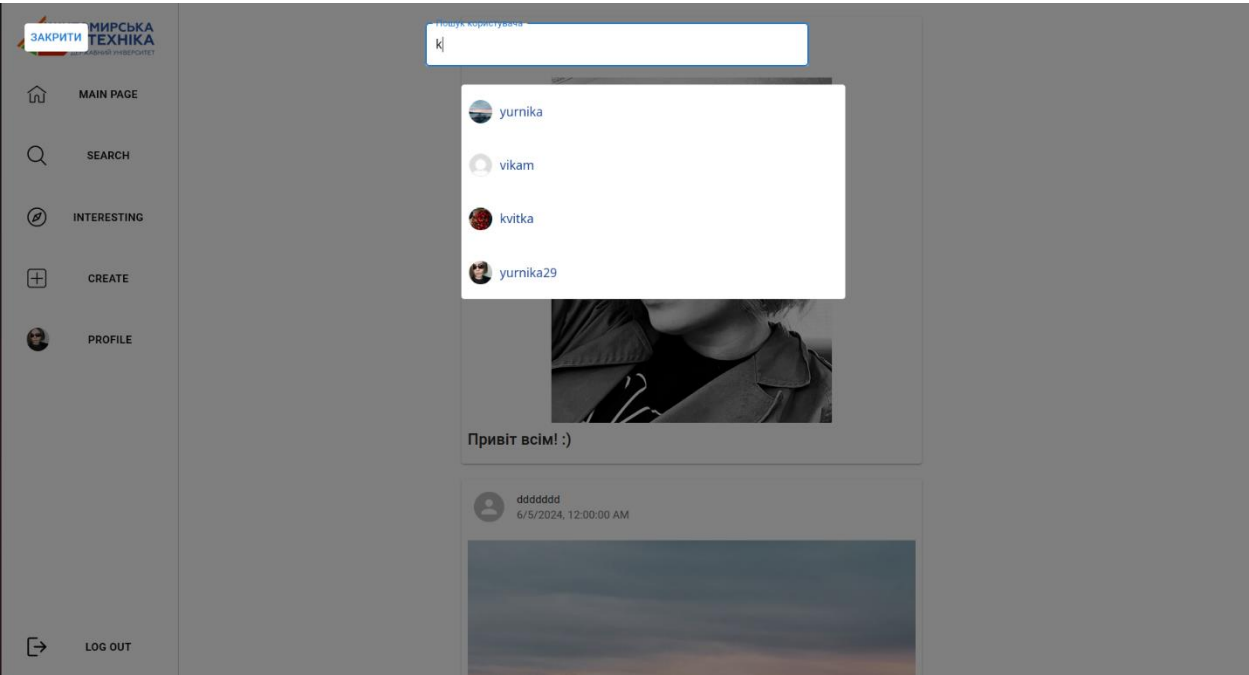


Рисунок 3.12. – Пошук користувачів

На цьому етапі користувач динамічним чином шукає своїх знайомих та може перейти на будь-який профіль знайдений ним в пошуковій стрічці.



kvitka

Subscribe

0 posts

Subscriptions: 0

Followers: 0

Marta Arni

Дописи

Рисунок 3.13. – Профіль користувача

Якщо ви не підписані на користувача у вас є можливість це зробити. На цій сторінці відображається певна статистика про користувача, його кількість постів, підписників та підписок. Натискаючи кнопку subscribe ми підписуємося на користувача.



kvitka

Unsubscribe

0 posts

Subscriptions: 0

Followers: 1

Marta Arni

Дописи

Рисунок 3.14. – Профіль користувача

Система надсилання повідомлень на пошту

При реєстрації користувача необхідно верифікувати особу та підтверджувати чи дійсно це реальна людина, тому було реалізовано систему, яка надсилає на пошту повідомлення з кодом допуску до системи.

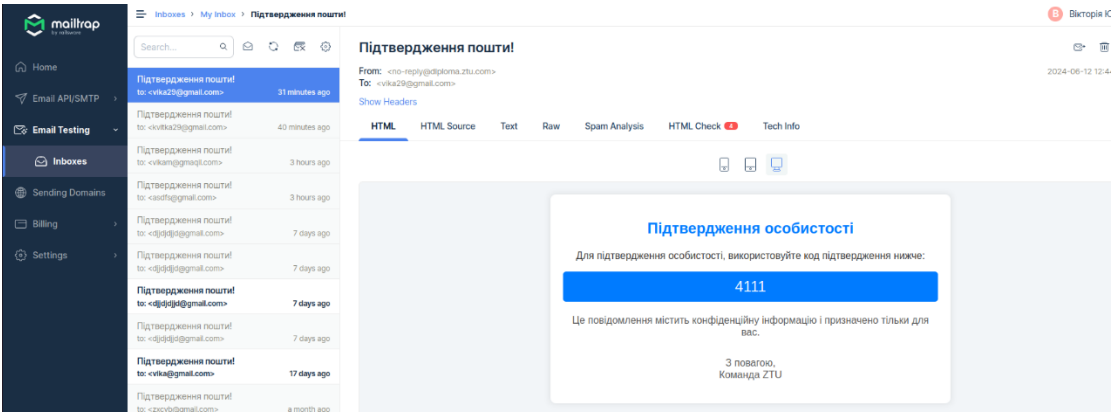


Рисунок 3.15. – Mailtrap

На цей сервіс приходять усі повідомлення, які йдуть на пошту. В повідомленні приходиться чотирьох значний номер який необхідно ввести в поле для підтвердження своєї особи при реєстрації.

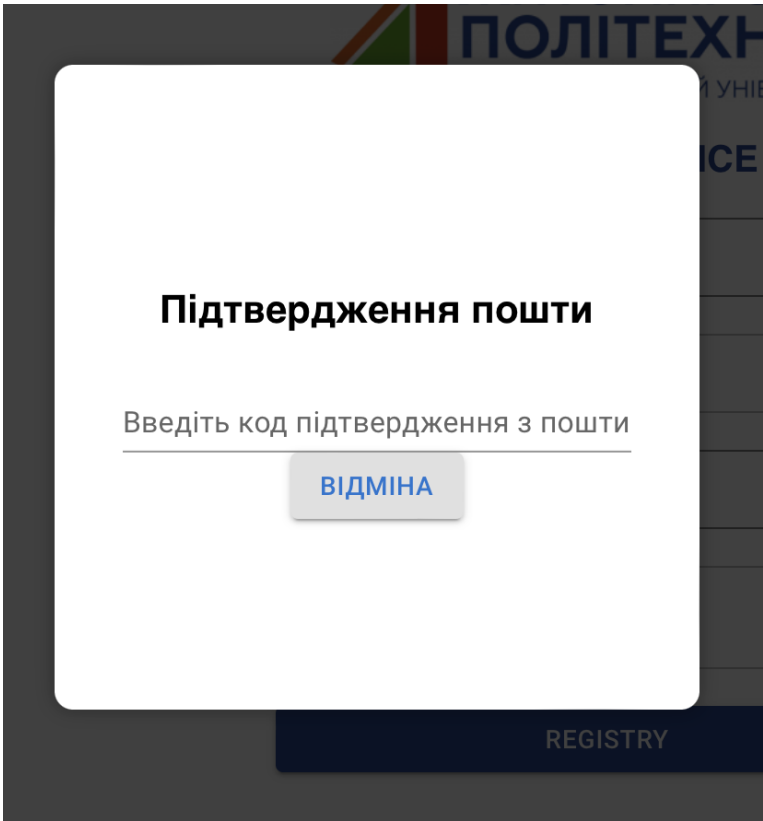


Рисунок 3.16. – Підтвердження пошти

Реєструючись у користувача є дві варіанти підтвердження своєї особи, одразу перейшовши до своєї пошти та переглянути отримане повідомлення. Також користувач може відкласти цю подію на пізніше, але тоді він отримає роль не ROLE_USER, а ROLE_GUEST.

Рисунок 3.17. – Підтвердження пошти

Ми можемо повернутися до цього у будь-який момент. Тож користувач не захотів одразу переходити до пошти і тепер йому необхідно згенерувати новий код. Він генерується автоматично, але якщо все ж таки користувачеві не прийшло повідомлення він може його регенерувати. Користувачеві відображається останні 3 символи пошти, на яку прийде повідомлення, оскільки авторизація на сайті відбувається за username.

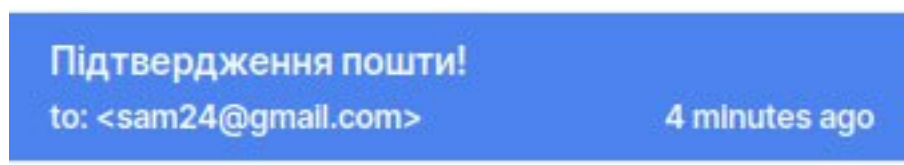


Рисунок 3.18. – Підтвердження пошти

Після генерації коду код прийшов на пошту і ми можемо підвищити свій рівень доступу до рівня користувача.

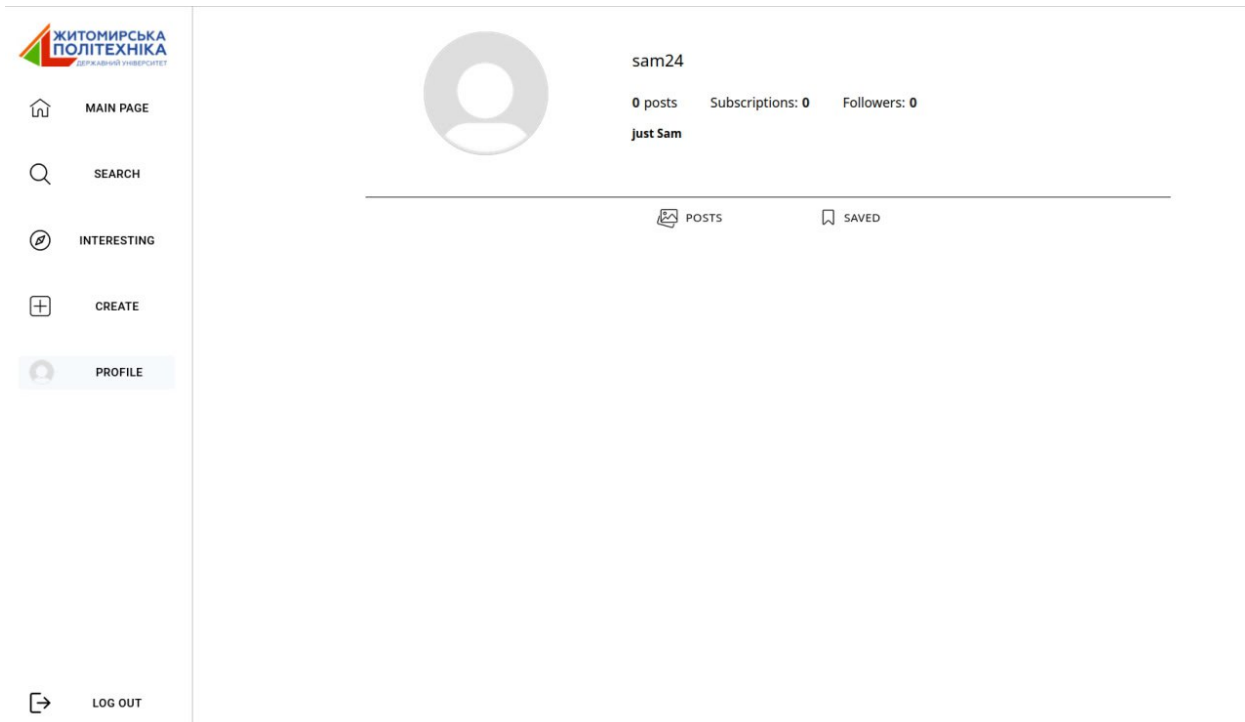


Рисунок 3.19. – Пуста сторінка

Ось ми і отримали нашу нову сторінку, для розміщення нових дописів.

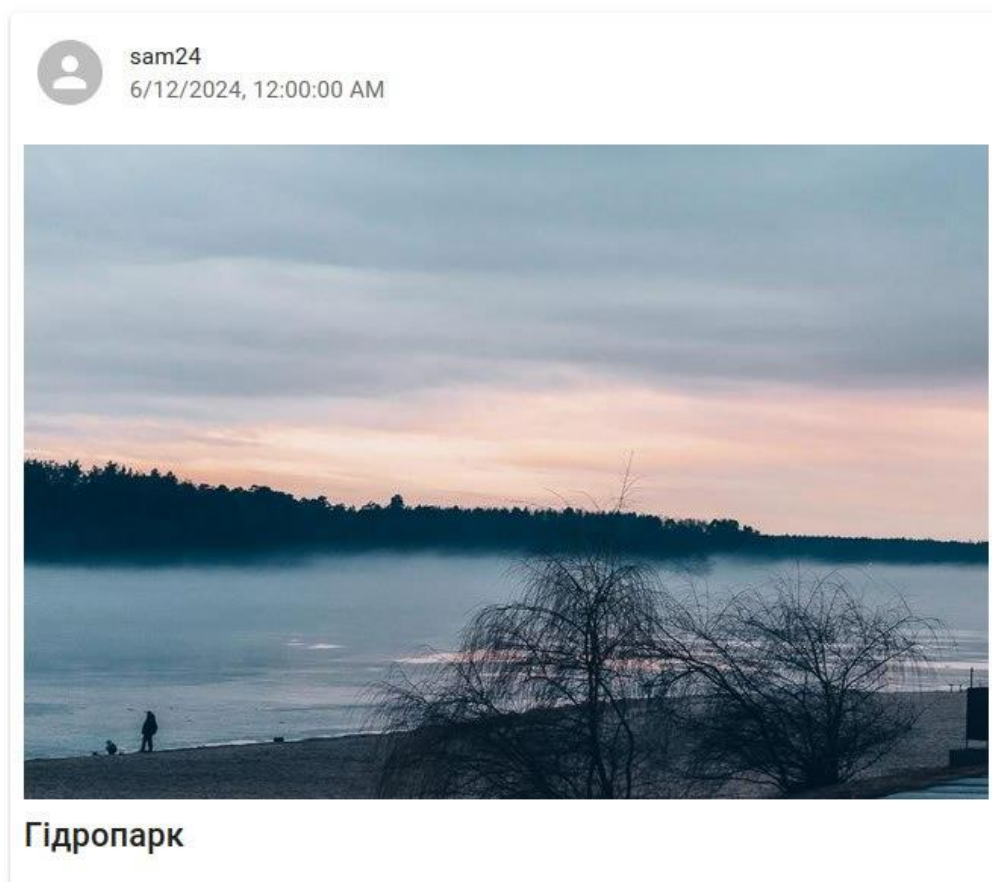


Рисунок 3.20. – Новий пост нового користувача

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

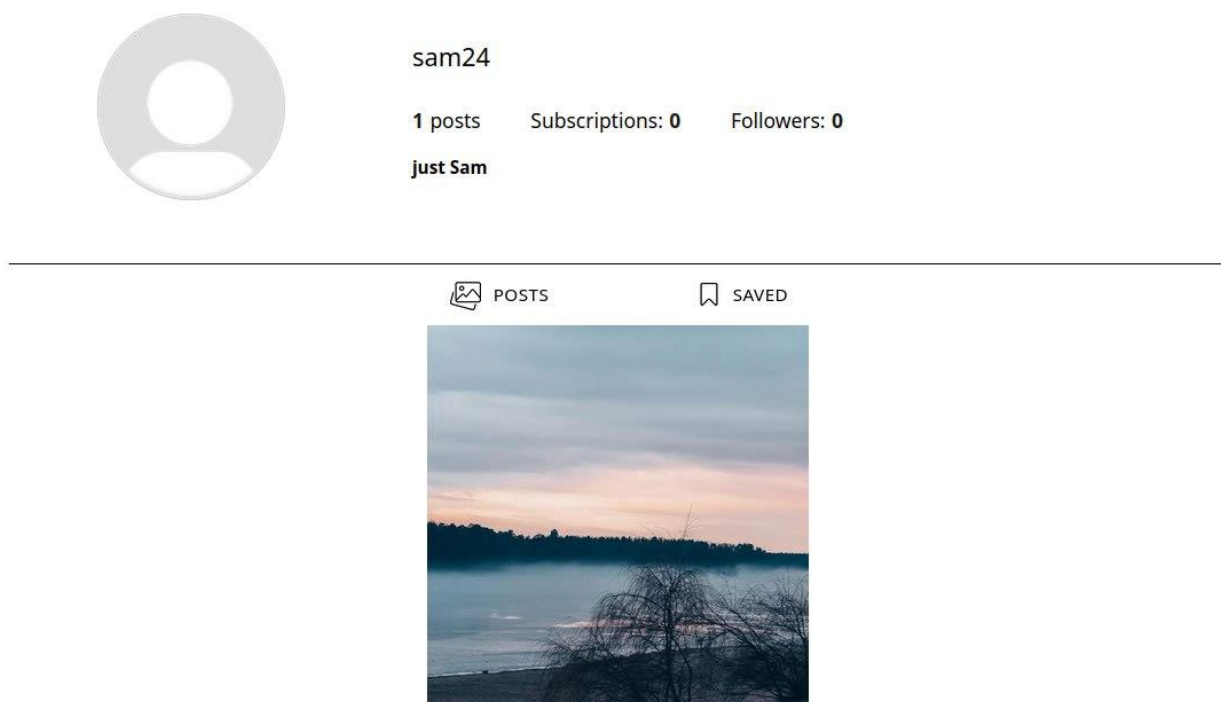


Рисунок 3.21. – Профіль нового користувача

Перегляд зображень доступний в декількох форматах. Один з яких приближений формат. В середині якого, користувач зможе видалити свій допис.

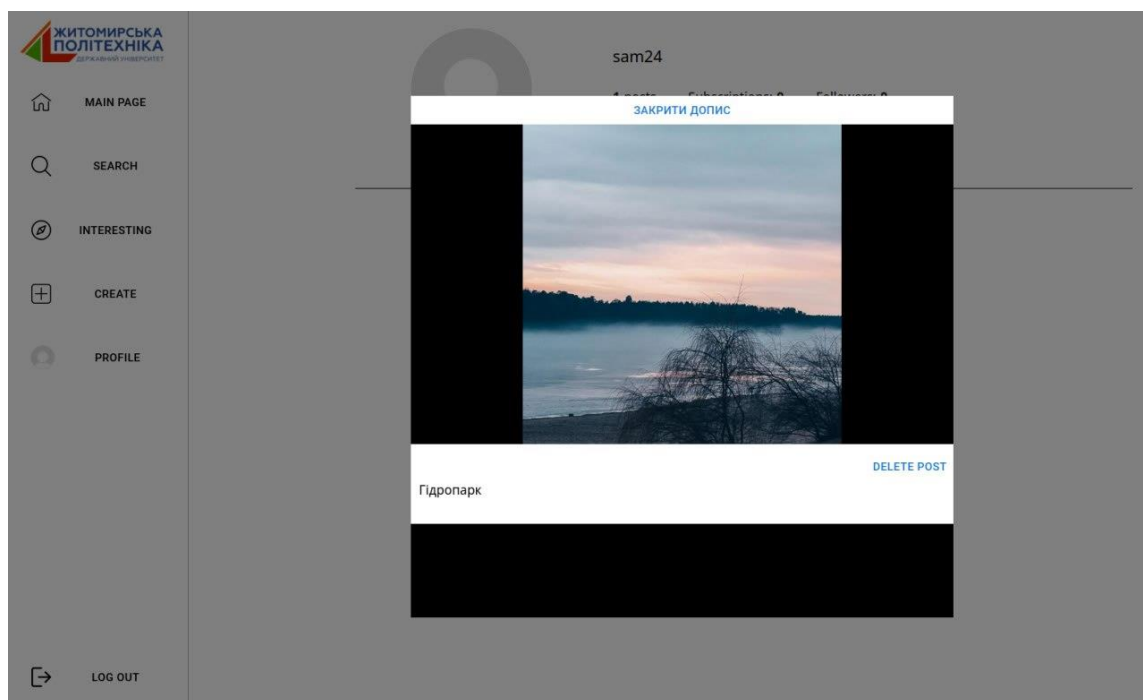


Рисунок 3.22. – Режим перегляду картинки

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		65

Функція видалення доступна лише на свої дописи, як на рівні серверної частини так і на рівні користувацької.

Висновки до розділу 3

У даному розділі детально розглянуто процес встановлення та налаштування системи для запуску застосунку, а також описано інтерфейс користувача.

По-перше, для запуску застосунку необхідно мати встановлені Docker, Docker Compose та Git. Найбільш рекомендованою операційною системою для розгортання проекту є Linux через його високу ефективність. Після встановлення всіх необхідних програм та клонування репозиторію, у директорії diploma-social-photo-service будуть доступні головні файли для запуску проекту: README.md, prepare_os.sh та local_recreate_docker.sh. Виконання скрипту prepare_os.sh підготує сервер до розгортання проекту, а скрипт local_recreate_docker.sh забезпечить запуск застосунку за допомогою Docker.

Далі розглянуто інтерфейс користувача. Спочатку було створено дизайн макети сторінок сайту з використанням Figma, дотримуючись вимог до стилю, кольорів та логотипів Державного університету “Житомирська політехніка”. Інтерфейс охоплює основні сторінки, такі як головна сторінка, сторінка реєстрації, профіль користувача та інші функціональні елементи.

Також було розглянуто систему авторизації та автентифікація, а саме те, що використання JWT та RBAC підходу дозволяє ефективно керувати правами доступу користувачів та забезпечує безпеку авторизації. Такий підхід дозволяє гнучко налаштовувати права доступу до різних ресурсів застосунку.

Реалізація системи підписок дозволяє користувачам стежити за активністю інших користувачів. Це збільшує взаємодію та залучення користувачів до застосунку, а використання системи верифікації пошти для

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		66

підтвердження особистості користувачів покращує безпеку та впевненість у реєстрації реальних користувачів.

У результаті, розділ 3 надає вичерпну інформацію про встановлення, налаштування та запуск застосунку, а також детально описує інтерфейс та функціональні можливості системи. Це забезпечує необхідні знання для ефективного використання та адміністрування застосунку, сприяючи кращому розумінню його структури та функціонування.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Розробка власної соціальної мережі фотосервісу є актуальною та корисною задачею в сучасному цифровому світі. Проєкт, що був виконаний, має на меті створення платформи, яка забезпечує користувачам безпечне, приватне середовище для збереження та обміну фото та відео, надаючи їм повний контроль над своїм контентом.

Реалізація даного проєкту дозволяє користувачам не тільки зберігати свої медіа-файли, але й використовувати платформу для розвитку бізнесу та особистого бренду. Завдяки можливості впровадження нових функцій, платформа задовольняє унікальні потреби своєї аудиторії, сприяє збереженню інтелектуальної власності та уникає ризиків, пов'язаних із зловживанням особистою інформацією.

Основною метою роботи була розробка застосунку, побудованого на мікросервісній архітектурі з використанням сучасних технологій, таких як Docker, Docker-Compose, Nginx, Symfony та React.js. В результаті виконання проєкту були досягнуті наступні завдання:

- Проведено аналіз UI/UX для покращення користувацького досвіду.
- Розроблено мікросервісну архітектуру та спроектовано структуру основних сервісів застосунку.
- Обґрунтовано вибір засобів реалізації продукту.
- Спроектовано базу даних та системи захисту даних користувачів.
- Розроблено серверну частину та реалізовано користувацьку частину.
- Виконано програмну реалізацію застосунку.

Проектування та реалізація застосунку включали використання таких технологій та підходів, як об'єктно-орієнтоване програмування, контейнеризація з Docker, система керування доступом на основі ролей

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

(RBAC), компонентний підхід до верстки, взаємодія з базою даних за допомогою Doctrine ORM, система надсилання листів Symfony Mailer, автентифікаційний токен (JWT), система безпеки Symfony Security, система роутингу та cloud-збереження даних Amazon S3.

Успішно виконаний проєкт може бути використаний для розважальних, рекламних та маркетингових цілей, забезпечуючи нові можливості для соціалізації, розвитку бізнесу та творчого самовираження. Розроблений функціонал дозволяє користувачам редагувати зображення, що повністю задовольняє їхні потреби щодо якості відображення, швидкості редагування та перегляду дописів.

Таким чином, даний проєкт є вагомим внеском у розвиток сучасних веб-технологій та демонструє широкі можливості для інновацій та створення нових цифрових продуктів.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація фреймворку Symfony [Електронний ресурс] - Режим доступу до ресурсу: <https://symfony.com/doc/current/index.html>
2. Документація фреймворку React.js [Електронний ресурс] - Режим доступу до ресурсу: <https://uk.legacy.reactjs.org/>
3. Документація AWS S3 [Електронний ресурс] - Режим доступу до ресурсу: <https://docs.aws.amazon.com/s3/>
4. Образи всіх мікросервісів додатку DockerHub [Електронний ресурс] - Режим доступу до ресурсу: <https://hub.docker.com/>
5. Material UI [Електронний ресурс] - Режим доступу до ресурсу: <https://mui.com/material-ui/all-components/>
6. Swagger (бекенд компонент) [Електронний ресурс] - Режим доступу до ресурсу: <https://symfonycasts.com/screencast/api-platform/swagger>
7. Nginx (веб-сервер) [Електронний ресурс] - Режим доступу до ресурсу: <https://nginx.org/en/docs/>
8. Документація системи контейнеризації Docker [Електронний ресурс] - Режим доступу до ресурсу: <https://docs.docker.com/>
9. PHP & MySQL: Novice to Ninja: навчання PHP та MySQL для початківців. – Amazon, 2017. – 352 с.
10. Modern PHP: New Features and Good Practices: сучасні можливості та найкращі практики PHP. – O'Reilly Media, 2015. – 270 с.
11. PHP Objects, Patterns, and Practice: об'єктно-орієнтоване програмування, патерни та практики в PHP. – Apress, 2018. – 528 с.
12. PHP Cookbook: рецепти та рішення для повсякденних задач в PHP. – O'Reilly Media, 2020. – 818 с.
13. Docker: Up & Running: використання Docker для створення контейнеризованих додатків. – O'Reilly Media, 2020. – 344 с.
14. Learning React: основи та передові техніки роботи з React. – O'Reilly Media, 2017. – 350 с.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

15. REST API Design Rulebook: найкращі практики та правила для проектування REST API. – O'Reilly Media, 2011. – 144 с.
16. Design Patterns: Elements of Reusable Object-Oriented Software: класичні патерни проектування для об'єктно-орієнтованого програмування. – Addison-Wesley Professional, 1994. – 395 с.
17. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability: основи веб-юзабіліті. – Steve Krug, 2013. – 216 с.
18. Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines: психологія користувачів та принципи UI-дизайну. – Jeff Johnson, 2020. – 312 с.
19. A Practical Guide to Web Accessibility: основи доступності веб-додатків. – Jon Duckett, 2021. – 320 с.
20. Web Design Playground: HTML & CSS the Interactive Way: візуальний підхід до навчання веб-дизайну. – Paul McFedries, 2019. – 376 с.
21. The Principles of Beautiful Web Design: основи красивого веб-дизайну. – Jason Beaird, 2020. – 288 с.
22. Content Strategy for the Web: розробка стратегій контенту для веб-сайтів. – Kristina Halvorson, Melissa Rach, 2012. – 256 с.
23. Site Reliability Engineering: How Google Runs Production Systems: підходи до забезпечення надійності сайтів. – Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Richard Murphy, 2016. – 552 с.
24. Information Architecture: For the Web and Beyond: структуризація інформації для веб-сайтів. – Louis Rosenfeld, Peter Morville, Jorge Arango, 2015. – 486 с.
25. Responsive Web Design with HTML5 and CSS: створення адаптивних веб-дизайнів. – Ben Frain, 2020. – 480 с.
26. Lean UX: Designing Great Products with Agile Teams: принципи Lean UX у веб-розробці. – Jeff Gothelf, Josh Seiden, 2021. – 304 с.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

27. The Elements of User Experience: User-Centered Design for the Web: основи користувацького досвіду для веб-дизайну. – Jesse James Garrett, 2010. – 272 с.
28. JavaScript and JQuery: Interactive Front-End Web Development: інтерактивна розробка веб-інтерфейсів. – Jon Duckett, 2014. – 640 с.
29. Strategic Writing for UX: Drive Engagement, Conversion, and Retention with Every Word: стратегічне написання текстів для UX. – Torrey Podmajersky, 2019. – 194 с.
30. User Stories Applied: For Agile Software Development: використання юзер-сторі в Agile веб-розробці. – Mike Cohn, 2004. – 288 с.
31. Соціальний фотосервіс [Електронний ресурс] - Режим доступу до ресурсу: https://gitlab.com/kn201_mv/diplomaphotoservice

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

AWSS3Service.php – клас AWSS3Service відповідає за роботу з Amazon

S3.

```

<?php

namespace App\Services;

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

class AWSS3Service
{
    /**
     * @param S3Client $s3Client
     */
    public function construct(
        private S3Client $s3Client
    )
    {
    }

    public function uploadFile($uploadedFile, string $bucketName): array
    {
        if ($uploadedFile && $uploadedFile->isValid() !== null && $uploadedFile->isValid()) {
            $originalName = $uploadedFile->getClientOriginalName();
            $fileName = md5(uniqid()) . '.' . $uploadedFile->getClientOriginalExtension();

            $file = fopen($uploadedFile->getPathname(), 'r');

            if ($file) {
                $this->s3Client->putObject([
                    'Bucket' => $bucketName,
                    'Key' => $fileName,
                    'Body' => $file,
                ]);

                fclose($file);

                return ["Title" => "File received and uploaded to S3: $originalName",
                    "IsTrue" => true,
                    "Name" => "$fileName"
                ];
            } else {
                return ["Title" => "Error opening file: $originalName",
                    "IsTrue" => false
                ];
            }
        }

        return ["Title" => "File do not exist !",
            "IsTrue" => false
        ];
    }

    public function uploadAva($uploadedFile, string $folderName, string $userId): array
    {
        if ($uploadedFile && $uploadedFile->isValid() !== null && $uploadedFile->isValid()) {
            $originalName = $uploadedFile->getClientOriginalName();
            $fileName = md5(uniqid()) . '.' . $uploadedFile->getClientOriginalExtension();

            $file = fopen($uploadedFile->getPathname(), 'r');

            if ($file) {
                $this->s3Client->putObject([
                    'Bucket' => 'myztuphotoservice',
                    'Key' => $folderName . "/" . $fileName,
                    'Body' => $file,
                ]);

                fclose($file);
            }
        }
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return ["Title" => "File received and uploaded to S3: $originalName",
                "IsTrue" => true,
                "Name"   => "$fileName"
               ];
    } else {
        return ["Title" => "Error opening file: $originalName",
                "IsTrue" => false
               ];
    }
}

return ["Title" => "File do not exist !",
        "IsTrue" => false
       ];
}

public function uploadUserPost($uploadedFile, $cropPhoto , string $folderName, string
$userId): array
{
    if ($uploadedFile && $uploadedFile->isValid() !== null && $uploadedFile->isValid()) {
        $originalName = $uploadedFile->getClientOriginalName();
        $fileName = md5(uniqid()) . '.' . $uploadedFile->getClientOriginalExtension();

        $userDirectoryKey = $folderName . "/" . $userId;
        if (!$this->checkDirectoryExists($userDirectoryKey)) {
            $this->createDirectory($userDirectoryKey);
            $this->createDirectory('posts');
        }

        $file = fopen($uploadedFile->getPathname(), 'r');

        if ($file) {
            $this->s3Client->putObject([
                'Bucket' => 'myztuphotoservice',
                'Key'     => $userDirectoryKey . "/" . 'posts/' . $fileName,
                'Body'    => $cropPhoto,
            ]);

            fclose($file);

            return ["Title" => "File received and uploaded to S3: $originalName",
                    "IsTrue" => true,
                    "Name"   => "$fileName"
                   ];
        } else {
            return ["Title" => "Error opening file: $originalName",
                    "IsTrue" => false
                   ];
        }
    }

    return ["Title" => "File do not exist !",
            "IsTrue" => false
           ];
}

public function uploadUserAva($uploadedFile, $cropPhoto , string $folderName, string
$userId): array
{
    if ($uploadedFile && $uploadedFile->isValid() !== null && $uploadedFile->isValid()) {
        $originalName = $uploadedFile->getClientOriginalName();
        $fileName = md5(uniqid()) . '.' . $uploadedFile->getClientOriginalExtension();

        $userDirectoryKey = $folderName . "/" . $userId;
        if (!$this->checkDirectoryExists($userDirectoryKey)) {
            $this->createDirectory($userDirectoryKey);
        }

        $file = fopen($uploadedFile->getPathname(), 'r');

        if ($file) {
            $this->s3Client->putObject([
                'Bucket' => 'myztuphotoservice',
                'Key'     => $userDirectoryKey . "/" . $fileName,
                'Body'    => $cropPhoto,
            ]);
        }
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        fclose($file);

        return ["Title" => "File received and uploaded to S3: $originalName",
                "IsTrue" => true,
                "Name"   => "$fileName"
        ];
    } else {
        return ["Title" => "Error opening file: $originalName",
                "IsTrue" => false
        ];
    }
}

return ["Title" => "File do not exist !",
        "IsTrue" => false
];
}

private function checkDirectoryExists(string $directoryKey): bool
{
    try {
        $result = $this->s3Client->headObject(['Bucket' => 'myztuphotoservice', 'Key' =>
$directoryKey]);
        return true;
    } catch (S3Exception $e) {
        return false;
    }
}

private function createDirectory(string $directoryKey): void
{
    $this->s3Client->putObject([
        'Bucket' => 'myztuphotoservice',
        'Key'    => $directoryKey . ".keep", // Add a placeholder file to keep the directory
structure
        'Body'   => '',
    ]);
}

public function updateFile(string $fileName, string $bucketName, string $newContent): array
{
    try {
        $this->s3Client->putObject([
            'Bucket' => $bucketName,
            'Key'    => $fileName,
            'Body'   => $newContent,
        ]);

        return [
            'Title' => "File $fileName successfully updated on S3",
            'IsTrue' => true,
        ];
    } catch (S3Exception $e) {
        return [
            'Title' => "Error updating file $fileName on S3: " . $e->getMessage(),
            'IsTrue' => false,
        ];
    }
}

public function getFile(string $fileName, string $bucketName): array
{
    try {
        $result = $this->s3Client->getObject([
            'Bucket' => $bucketName,
            'Key'    => $fileName,
        ]);

        $fileContents = $result['Body'];

        return [
            'Title'   => "File $fileName successfully retrieved from S3",
            'IsTrue'  => true,
            'FileName' => $fileName,
            'Content'  => $fileContents,
        ];
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } catch (S3Exception $e) {
        return [
            'Title' => "Error retrieving file $fileName from S3: " . $e->getMessage(),
            'IsTrue' => false,
        ];
    }
}

public function getFilesList(array $fileNames, string $username): array
{
    $filesData = [];

    foreach ($fileNames as $fileName) {
        try {
            $cmd = $this->s3Client->getCommand('GetObject', [
                'Bucket' => 'myztuphotoservice',
                'Key' => "userposts/" . "$username/posts/" . $fileName,
            ]);
            $request = $this->s3Client->createPresignedRequest($cmd, '+20 minutes');
            $presignedUrl = (string)$request->getUri();
            $fileInfo = $this->s3Client->headObject([
                'Bucket' => 'myztuphotoservice',
                'Key' => "userposts/" . "$username/posts/" . $fileName,
            ]);

            $filesData[] = [
                'Title' => "File $fileName successfully retrieved from S3",
                'FileName' => $fileName,
                'IsTrue' => true,
                'Content' => $presignedUrl,
                'CreatedAt' => $fileInfo->get('LastModified'),
            ];
        } catch (S3Exception $e) {
            $filesData[] = [
                'Title' => "Error retrieving file $fileName from S3: " . $e->getMessage(),
                'IsTrue' => false,
                'CreatedAt' => null,
            ];
        }
    }

    return $filesData;
}

public function getUserAva(array $fileName, string $folderName, string $userId): array
{
    $filesData = [];

    try {
        $cmd = $this->s3Client->getCommand('GetObject', [
            'Bucket' => 'myztuphotoservice',
            'Key' => $folderName . "/" . $userId . "/" . $fileName[0],
        ]);
        $request = $this->s3Client->createPresignedRequest($cmd, '+20 minutes');
        $presignedUrl = (string)$request->getUri();
        $isTrue = $request->getUri()->getPath() === "/avaphoto/" ? false : true;

        $filesData[] = [
            'Title' => "File $fileName[0] successfully retrieved from S3",
            'FileName' => $fileName[0],
            'IsTrue' => $isTrue,
            'Content' => $presignedUrl,
        ];
    } catch (S3Exception $e) {
        $filesData[] = [
            'Title' => "Error retrieving file $fileName[0] from S3: " . $e->getMessage(),
            'IsTrue' => false,
        ];
    }

    return $filesData;
}

public function getUserPosts(string $folderName, string $userId): array
{
    $filesData = [];

    try {

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

$paginator = $this->s3Client->getPaginator('ListObjectsV2', [
    'Bucket' => 'myztuphotoservice',
    'Prefix' => $folderName . "/" . $userId . "/" . "/posts",
]);

foreach ($paginator as $result) {
    foreach ($result['Contents'] as $object) {
        $cmd = $this->s3Client->getCommand('GetObject', [
            'Bucket' => 'myztuphotoservice',
            'Key' => $object['Key'],
        ]);
        $request = $this->s3Client->createPresignedRequest($cmd, '+20 minutes');
        $presignedUrl = (string)$request->getUri();
        $isTrue = $request->getUri()->getPath() === "/avaphoto/" ? false : true;

        $filesData[] = [
            'Title' => "File {$object['Key']} successfully retrieved from S3",
            'FileName' => $object['Key'],
            'IsTrue' => $isTrue,
            'Content' => $presignedUrl,
        ];
    }
} catch (S3Exception $e) {
    $filesData[] = [
        'Title' => "Error retrieving files from S3: " . $e->getMessage(),
        'IsTrue' => false,
    ];
}

return $filesData;
}

public function deleteUserAva(array $fileName, string $folderName, string $userId): array
{
    $filesData = [];

    try {
        $this->s3Client->deleteObject([
            'Bucket' => 'myztuphotoservice',
            'Key' => $folderName . "/" . $userId . "/" . $fileName[0],
        ]);

        $filesData[] = [
            'Title' => "File $fileName[0] successfully deleted from S3",
            'FileName' => $fileName[0],
            'IsTrue' => true,
        ];
    } catch (S3Exception $e) {
        $filesData[] = [
            'Title' => "Error deleting file $fileName[0] from S3: " . $e->getMessage(),
            'IsTrue' => false,
        ];
    }

    return $filesData;
}
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

User.php - клас User у Symfony представляє собою модель користувача з різноманітними властивостями.

```
<?php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiProperty;
use ApiPlatform\Core\Annotation\ApiResource;
use App\Action\CreateUserAction;
use App\Repository\UserRepository;
use DateTime;
use DateTimeInterface;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Serializer\Annotation\Groups;
use Symfony\Component\Uid\Uuid;
use Symfony\Component\Validator\Constraints>Email;
use Symfony\Component\Validator\Constraints\Length;
use Symfony\Component\Validator\Constraints\NotBlank;
use Symfony\Component\Validator\Constraints\NotNull;
use Symfony\Component\Validator\Constraints\Regex;

#[ApiResource(
    collectionOperations: [
        "get" => [
            "method"                => "GET",
            "security"              => "is_granted('" . User::ROLE_ADMIN . "')",
            "normalization_context" => ["groups" => ["get:collection:user"]]
        ],
        "post" => [
            "method"                => "POST",
            "security"              => "is_granted('IS_AUTHENTICATED_ANONYMOUSLY')",
            "denormalization_context" => ["groups" => ["post:collection:user"]],
            "normalization_context"  => ["groups" => ["empty"]],
            "controller"            => CreateUserAction::class
        ]
    ],
    itemOperations: [
        "get" => [
            "method"                => "GET",
            "security"              => "(is_granted('" . User::ROLE_ADMIN . "') or
(is_granted('" . User::ROLE_USER . "') and object == user)",
            "normalization_context" => ["groups" => ["get:item:user"]]
        ],
        "put" => [
            "method"                => "PUT",
            "security"              => "object == user",
            "denormalization_context" => ["groups" => ["put:item:user"]],
            "normalization_context"  => ["groups" => ["get:item:user"]]
        ]
    ]
)]
#[ORM\Entity(repositoryClass: UserRepository::class)]
#[UniqueEntity(fields: ["email"], message: "Email is already in use")]
#[UniqueEntity(fields: ["phoneNumber"], message: "Phone number is already in use")]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    public const ROLE_USER      = "ROLE_USER";
    public const ROLE_ADMIN    = "ROLE_ADMIN";
    public const ROLE_GUEST    = "ROLE_GUEST";

    /**
     * @var Uuid
     */
    #[ORM\Id]
    #[ORM\Column(type: "uuid", unique: true)]
```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#[ORM\GeneratedValue(strategy: "CUSTOM")]
#[ORM\CustomIdGenerator(class: "doctrine.uuid_generator")]
#[Groups([
    "get:collection:user"
])]
private Uuid $id;

/**
 * @var string|null
 */
#[ORM\Column(length: 180, unique: true)]
#[Email]
#[NotBlank]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "post:collection:user",
    "put:item:user"
])]
private ?string $email = null;

/**
 * @var string[]
 */
#[ORM\Column]
private array $roles = [];

/**
 * @var string|null The hashed password
 */
#[ORM\Column]
#[Length(min: 8, minMessage: "Password must be at least {{ limit }} characters long")]
#[Groups([
    "post:collection:user"
])]
private ?string $password = null;

/**
 * @var string|null
 */
#[Regex(
    pattern:
"/\+\\s*(9[976]\\d|8[987530]\\d|6[987]\\d|5[90]\\d|42\\d|3[875]\\d|2[98654321]\\d|9[8543210]|8[6421]|6[6543210]|5[87654321]|4[987654310]|3[9643210]|2[70]|7|1)\\s*\\d{1,14}$/",
    message: 'Your name cannot contain a number',
    match: false,
)]
#[ORM\Column(length: 255, unique: true)]
#[NotBlank]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
    "post:collection:user"
])]
private ?string $phoneNumber = null;

/**
 * @var bool
 */
#[ORM\Column]
private bool $isVerified = false;

/**
 * @var Verification|null
 */
#[ORM\OneToOne(mappedBy: 'user', cascade: [
    'persist',
    'remove'
])]
private ?Verification $verification = null;

/**
 * @var string|null
 */
#[ORM\Column(length: 255, unique: true)]
#[Groups([

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "get:item:user",
        "get:collection:user",
        "put:item:user",
        "post:collection:user",
    ]])
private ?string $nickname = null;

/**
 * @var string|null
 */
#[ORM\Column(type: Types::TEXT, nullable: true)]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
])]
private ?string $bio = null;

/**
 * @var string|null
 */
#[ORM\Column(length: 255, nullable: true)]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
])]
private ?string $profilePictureUrl = null;

/**
 * @var string|null
 */
#[ORM\Column(length: 255, nullable: true)]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
])]
private ?string $websiteUrl = null;

/**
 * @var DateTimeInterface|null
 */
#[ORM\Column(type: Types::DATE_MUTABLE)]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
    "post:collection:user"
])]
private ?DateTimeInterface $birthDate = null;

/**
 * @var DateTimeInterface|null
 */
#[ORM\Column(type: Types::DATE_MUTABLE)]
private ?DateTimeInterface $registrationDate = null;

/**
 * @var DateTimeInterface|null
 */
#[ORM\Column(type: Types::DATETIME_MUTABLE, nullable: true)]
private ?DateTimeInterface $lastLogin = null;

/**
 * @var string|null
 */
#[ORM\Column(length: 255, nullable: true)]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
    "post:collection:user"
])]
private ?string $country = null;

/**

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    * @var string|null
    */
#[ORM\Column(length: 255, nullable: true)]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
    "post:collection:user"
])]
private ?string $city = null;

/**
 * @var Collection
 */
#[ORM\OneToMany(mappedBy: 'follower', targetEntity: Subscription::class)]
#[Groups([
    "get:item:user",
    "get:item:user:subscriptions", // Add the same group here
])]
private Collection $subscriptions;

/**
 * @var Collection
 */
#[ORM\OneToMany(mappedBy: 'leader', targetEntity: Subscription::class)]
#[Groups([
    "get:item:user",
    "get:item:user:subscriptions", // Add the same group here
])]
private Collection $isSubscribers;

/**
 * @var bool|null
 */
#[ORM\Column(nullable: true)]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
])]
private bool $isPrivate = false;

/**
 * @var Collection
 */
#[ORM\OneToMany(mappedBy: 'visitor', targetEntity: ProfileVisit::class)]
private Collection $profileVisits;

/**
 * @var Collection
 */
#[ORM\OneToMany(mappedBy: 'user', targetEntity: Post::class)]
private Collection $posts;

/**
 * @var Collection
 */
#[ORM\OneToMany(mappedBy: 'user', targetEntity: PostLikes::class)]
private Collection $postLikes;

/**
 * @var string|null
 */
#[ORM\Column(length: 255, nullable: true)]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
])]
private ?string $clientIP = null;

/**
 * @var string|null
 */

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#[ORM\Column(length: 255)]
#[Groups([
    "get:item:user",
    "get:collection:user",
    "put:item:user",
    "post:collection:user"
])]
private ?string $fullName = null;

/**
 * @var Collection
 */
#[ORM\OneToMany(mappedBy: 'user', targetEntity: SavedPosts::class)]
private Collection $savedPosts;

/**
 * User constructor
 */
public function __construct()
{
    $this->roles = [self::ROLE_USER];
    $this->subscriptions = new ArrayCollection();
    $this->isSubscribers = new ArrayCollection();
    $this->profileVisits = new ArrayCollection();
    $this->posts = new ArrayCollection();
    $this->postLikes = new ArrayCollection();
    $this->setRegistrationDate(new DateTime());
    $this->savedPosts = new ArrayCollection();
    $this->isPrivate = false;
}

/**
 * @return Uuid|null
 */
public function getId(): ?Uuid
{
    return $this->id;
}

/**
 * @return string|null
 */
public function getEmail(): ?string
{
    return $this->email;
}

/**
 * @param string $email
 * @return $this
 */
public function setEmail(string $email): self
{
    $this->email = $email;

    return $this;
}

/**
 * A visual identifier that represents this user.
 *
 * @see UserInterface
 */
public function getUserIdentifier(): string
{
    return (string)$this->email;
}

/**
 * @deprecated since Symfony 5.3, use getUserIdentifier instead
 */
public function getUsername(): string
{
    return (string)$this->nickname;
}

/**

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    * @see UserInterface
    */
    public function getRoles(): array
    {
        return $this->roles;
    }

    /**
     * @param array $roles
     * @return $this
     */
    public function setRoles(array $roles): self
    {
        $this->roles = $roles;

        return $this;
    }

    /**
     * @see PasswordAuthenticatedUserInterface
     */
    public function getPassword(): string
    {
        return $this->password;
    }

    /**
     * @param string $password
     * @return $this
     */
    public function setPassword(string $password): self
    {
        $this->password = $password;

        return $this;
    }

    /**
     * Returning a salt is only needed, if you are not using a modern
     * hashing algorithm (e.g. bcrypt or sodium) in your security.yaml.
     *
     * @see UserInterface
     */
    public function getSalt(): ?string
    {
        return null;
    }

    /**
     * @see UserInterface
     */
    public function eraseCredentials(): void
    {
        // If you store any temporary, sensitive data on the user, clear it here
        // $this->plainPassword = null;
    }

    /**
     * @return string|null
     */
    public function getPhoneNumber(): ?string
    {
        return $this->phoneNumber;
    }

    /**
     * @param string $phoneNumber
     * @return $this
     */
    public function setPhoneNumber(string $phoneNumber): self
    {
        $this->phoneNumber = $phoneNumber;

        return $this;
    }

    /**
     * @return bool|null

```

```

    */
    public function isVerified(): ?bool
    {
        return $this->isVerified;
    }

    /**
     * @param bool $isVerified
     * @return $this
     */
    public function setIsVerified(bool $isVerified): self
    {
        $this->isVerified = $isVerified;

        return $this;
    }

    /**
     * @return string|null
     */
    public function getNickname(): ?string
    {
        return $this->nickname;
    }

    /**
     * @param string $nickname
     * @return $this
     */
    public function setNickname(string $nickname): static
    {
        $this->nickname = $nickname;

        return $this;
    }

    /**
     * @return string|null
     */
    public function getBio(): ?string
    {
        return $this->bio;
    }

    /**
     * @param string|null $bio
     * @return $this
     */
    public function setBio(?string $bio): static
    {
        $this->bio = $bio;

        return $this;
    }

    /**
     * @return string|null
     */
    public function getProfilePictureUrl(): ?string
    {
        return $this->profilePictureUrl;
    }

    /**
     * @param string $profilePictureUrl
     * @return $this
     */
    public function setProfilePictureUrl(string $profilePictureUrl): static
    {
        $this->profilePictureUrl = $profilePictureUrl;

        return $this;
    }

    /**
     * @return string|null
     */
    public function getWebsiteUrl(): ?string
    {

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return $this->websiteUrl;
    }

    /**
     * @param string|null $websiteUrl
     * @return $this
     */
    public function setWebsiteUrl(?string $websiteUrl): static
    {
        $this->websiteUrl = $websiteUrl;

        return $this;
    }

    /**
     * @return DateTimeInterface|null
     */
    public function getBirthDate(): ?DateTimeInterface
    {
        return $this->birthDate;
    }

    /**
     * @param DateTimeInterface $birthDate
     * @return $this
     */
    public function setBirthDate(DateTimeInterface $birthDate): static
    {
        $this->birthDate = $birthDate;

        return $this;
    }

    /**
     * @return DateTimeInterface|null
     */
    public function getRegistrationDate(): ?DateTimeInterface
    {
        return $this->registrationDate;
    }

    /**
     * @param DateTimeInterface $registrationDate
     * @return $this
     */
    public function setRegistrationDate(DateTimeInterface $registrationDate): static
    {
        $this->registrationDate = $registrationDate;

        return $this;
    }

    /**
     * @return string|null
     */
    public function getCountry(): ?string
    {
        return $this->country;
    }

    /**
     * @param string|null $country
     * @return $this
     */
    public function setCountry(?string $country): static
    {
        $this->country = $country;

        return $this;
    }

    /**
     * @return string|null
     */
    public function getCity(): ?string
    {
        return $this->city;
    }

```

```

/**
 * @param string|null $city
 * @return $this
 */
public function setCity(?string $city): static
{
    $this->city = $city;

    return $this;
}

/**
 * @return Collection<int, Subscription>
 */
public function getSubscriptions(): Collection
{
    return $this->subscriptions;
}

/**
 * @param Subscription $subscription
 * @return $this
 */
public function addSubscription(Subscription $subscription): static
{
    if (!$this->subscriptions->contains($subscription)) {
        $this->subscriptions->add($subscription);
        $subscription->setFollower($this);
    }

    return $this;
}

/**
 * @param Subscription $subscription
 * @return $this
 */
public function removeSubscription(Subscription $subscription): static
{
    if ($this->subscriptions->removeElement($subscription)) {
        // set the owning side to null (unless already changed)
        if ($subscription->getFollower() === $this) {
            $subscription->setFollower(null);
        }
    }

    return $this;
}

/**
 * @return Collection<int, Subscription>
 */
public function getisSubscribers(): Collection
{
    return $this->isSubscribers;
}

/**
 * @param Subscription $subscriber
 * @return $this
 */
public function addSubscriber(Subscription $subscriber): static
{
    if (!$this->isSubscribers->contains($subscriber)) {
        $this->isSubscribers->add($subscriber);
        $subscriber->setLeader($this);
    }

    return $this;
}

/**
 * @param Subscription $subscriber
 * @return $this
 */
public function removeSubscriber(Subscription $subscriber): static
{

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if ($this->isSubscribers->removeElement($subscriber)) {
            // set the owning side to null (unless already changed)
            if ($subscriber->getLeader() === $this) {
                $subscriber->setLeader(null);
            }
        }

        return $this;
    }

    /**
     * @return bool|null
     */
    public function isIsPrivate(): bool
    {
        return $this->isPrivate;
    }

    /**
     * @param bool $isPrivate
     * @return $this
     */
    public function setIsPrivate(bool $isPrivate): static
    {
        $this->isPrivate = $isPrivate;

        return $this;
    }

    /**
     * @return Collection<int, ProfileVisit>
     */
    public function getProfileVisits(): Collection
    {
        return $this->profileVisits;
    }

    /**
     * @param ProfileVisit $profileVisit
     * @return $this
     */
    public function addProfileVisit(ProfileVisit $profileVisit): static
    {
        if (!$this->profileVisits->contains($profileVisit)) {
            $this->profileVisits->add($profileVisit);
            $profileVisit->setVisitor($this);
        }

        return $this;
    }

    /**
     * @param ProfileVisit $profileVisit
     * @return $this
     */
    public function removeProfileVisit(ProfileVisit $profileVisit): static
    {
        if ($this->profileVisits->removeElement($profileVisit)) {
            // set the owning side to null (unless already changed)
            if ($profileVisit->getVisitor() === $this) {
                $profileVisit->setVisitor(null);
            }
        }

        return $this;
    }

    /**
     * @return Collection<int, Post>
     */
    public function getPosts(): Collection
    {
        return $this->posts;
    }

    /**
     * @param Post $post
     * @return $this
     */

```

```

    */
    public function addPost(Post $post): static
    {
        if (!$this->posts->contains($post)) {
            $this->posts->add($post);
            $post->setUser($this);
        }

        return $this;
    }

    /**
     * @param Post $post
     * @return $this
     */
    public function removePost(Post $post): static
    {
        if ($this->posts->removeElement($post)) {
            // set the owning side to null (unless already changed)
            if ($post->getUser() === $this) {
                $post->setUser(null);
            }
        }

        return $this;
    }

    /**
     * @return Collection<int, PostLikes>
     */
    public function getPostLikes(): Collection
    {
        return $this->postLikes;
    }

    /**
     * @param PostLikes $postLike
     * @return $this
     */
    public function addPostLike(PostLikes $postLike): static
    {
        if (!$this->postLikes->contains($postLike)) {
            $this->postLikes->add($postLike);
            $postLike->setUser($this);
        }

        return $this;
    }

    /**
     * @param PostLikes $postLike
     * @return $this
     */
    public function removePostLike(PostLikes $postLike): static
    {
        if ($this->postLikes->removeElement($postLike)) {
            // set the owning side to null (unless already changed)
            if ($postLike->getUser() === $this) {
                $postLike->setUser(null);
            }
        }

        return $this;
    }

    /**
     * @return string|null
     */
    public function getClientIP(): ?string
    {
        return $this->clientIP;
    }

    /**
     * @param string|null $clientIP
     * @return $this
     */
    public function setClientIP(?string $clientIP): static

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
    $this->clientIP = $clientIP;

    return $this;
}

/**
 * @return string|null
 */
public function getFullName(): ?string
{
    return $this->fullName;
}

/**
 * @param string $fullName
 * @return $this
 */
public function setFullName(string $fullName): static
{
    $this->fullName = $fullName;

    return $this;
}

/**
 * @return SavedPosts|null
 */
public function getSavedPost(): ?SavedPosts
{
    return $this->savedPost;
}

/**
 * @param SavedPosts|null $savedPost
 * @return $this
 */
public function setSavedPost(?SavedPosts $savedPost): static
{
    $this->savedPost = $savedPost;

    return $this;
}

/**
 * @return Collection<int, SavedPosts>
 */
public function getSavedPosts(): Collection
{
    return $this->savedPosts;
}

/**
 * @param SavedPosts $savedPost
 * @return $this
 */
public function addSavedPost(SavedPosts $savedPost): static
{
    if (!$this->savedPosts->contains($savedPost)) {
        $this->savedPosts->add($savedPost);
        $savedPost->setUser($this);
    }

    return $this;
}

/**
 * @param SavedPosts $savedPost
 * @return $this
 */
public function removeSavedPost(SavedPosts $savedPost): static
{
    if ($this->savedPosts->removeElement($savedPost)) {
        // set the owning side to null (unless already changed)
        if ($savedPost->getUser() === $this) {
            $savedPost->setUser(null);
        }
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return $this;
    }

    /**
     * @return DateTimeInterface|null
     */
    public function getLastLogin(): ?DateTimeInterface
    {
        return $this->lastLogin;
    }

    /**
     * @param DateTimeInterface|null $lastLogin
     */
    public function setLastLogin(?DateTimeInterface $lastLogin): void
    {
        $this->lastLogin = $lastLogin;
    }
}

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		