Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Автоматизація управління анонімною соціальною мережею»

(тема роботи)

Виконав: студент 2-ого курсу групи 1АКІТР-23м
(шифр групи)
спеціальності 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(шифр та назва спеціальності)
Владислав СИДЮК
(ПІБ студента)

Керівник: к.т.н., доцент каф. АІІТ
Ілона БОГАЧ
(науковий ступінь, вчене звання / посада, ПІБ керівника)
«16» 12 2024р.

Опонент: к.т.н., доцент каф.САІТ
Георгій ГОРЯЧЕВ
(науковий ступінь, вчене звання / посада, ПІБ керівника)
«16» 12 2024 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)
«16» 12 202_ р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань - Електроніка, автоматизація та електронні комунікації

Спеціальність - 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Освітньо-професійна програма - Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«19» 09 2024 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сидюку Владиславу Володимировичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Автоматизація управління анонімною соціальною мережею Керівник роботи: к.т.н., доцент кафедри АІІТ Ілона БОГАЧ
   Затверджені наказом ВНТУ від «17» 09 202_ року №310.
2. Строк подання роботи студентом: до «16» 12 2024року.
3. Вихідні дані до роботи: розроблене серверне програмне забезпечення з можливістю запускатись на системі Windows, MacOS та Linux: мова програмування – Java, фреймворк – Spring. Структура упрравління бази даних – нереляційна база MongoDB. Оперативна пам'ять: 4 ГБ, Пропускна здатність:10 Мбіт/с.
4. Зміст текстової частини: вступ; аналіз предметної області та існуючих аналогів; аналіз веб архітектур та інструментів розробки; реалізація системи; економічний розділ; висновки; список використаних джерел.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): Діаграма класів сутностей; діаграма класів сервісів; UML діаграма варіантів використання; схема роботи додатку клієнт-серверної частини; діаграма взаємодії серверу з блокчейн; діаграма роботи смарт-контракту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Ілона БОГАЧ к.т.н., доцент каф. АІІТ	18.09.24	16.12.24
5	Володимир КОЗЛОВСЬКИЙ к.е.н., доцент каф. ЕПтаВМ	20.09.24	20.11.24

7. Дата видачі завдання "18" вересня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Термін виконання		Примітки
		початок	закінчення	
1	Постановка задачі та аналіз предметної області	18.09.2024	25.09.2024	вик.
2	Вибір архітектури та технологій розробки	26.09.24	15.10.24	вик.
3	Розробка системи з урахуванням безпеки та конфіденційності даних	16.10.24	05.11.24	вик.
4	Економічна частина	06.11.24	20.11.24	вик.
7	Оформлення матеріалів до захисту	20.11.24	30.11.24	вик.
8	Остаточний захист роботи			

Студент _____ Владислав СИДЮК

Керівник роботи _____ Ілона БОГАЧ

АНОТАЦІЯ

УДК 004.457

Сидюк В. В. Розробка анонімної соціальної мережі з використанням Blockchain. Магістерська кваліфікаційна робота зі спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка. Інтелектуальні комп’ютерні системи управління. Вінниця: ВНТУ, 2024. 114 с.

На укр. мові. Бібліогр.: 32 рис.: 10 табл. 8.

У магістерській кваліфікаційній роботі досліджено проблему забезпечення анонімності та захисту даних користувачів у соціальних мережах. Проведено аналіз існуючих систем шифрування, аутентифікації та конфіденційності. Запропоновано вдосконалений метод збереження та захисту даних, що ґрунтується на використанні гібридного підходу: Blockchain для обробки критичних даних і реляційної бази даних PostgreSQL для менш важливих.

Робота включає розробку архітектури системи, реалізацію основних модулів (реєстрації, авторизації, взаємодії з Blockchain), а також тестування функціоналу та аналіз продуктивності. Для захисту інформації використано криптографічні алгоритми, шифрування даних перед передачею і багатофакторну аутентифікацію. Система базується на фреймворку Spring Boot, забезпечуючи масштабованість і модульність.

Ключові слова: анонімна соціальна мережа, Blockchain, Java, Spring Boot, безпека даних, аутентифікація, шифрування.

ABSTRACT

Sydiuk V.V. Development of an Anonymous Social Network Using Blockchain. Master's Qualification Thesis in the specialty 174 – Automation, Computer-Integrated Technologies, and Robotics. Intelligent Computer Control Systems. Vinnytsia: VNTU, 2024. 114 p.

In English. References: 24; Figures: 10; Tables: 7.

The master's qualification thesis investigates the problem of ensuring anonymity and data protection for users of social networks. The study analyzes existing encryption, authentication, and confidentiality systems. An advanced method of data storage and protection is proposed, based on a hybrid approach: Blockchain for processing critical data and PostgreSQL relational database for less critical data.

The work includes the development of the system architecture, implementation of core modules (registration, authorization, Blockchain interaction), functional testing, and performance analysis. Cryptographic algorithms, data encryption during transmission, and multi-factor authentication were employed to safeguard information. The system is built using the Spring Boot framework, ensuring scalability and modularity.

Keywords: anonymous social network, Blockchain, Java, Spring Boot, data security, authentication, encryption.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному цифровому середовищі анонімність в інтернеті є ключовим елементом для забезпечення приватності особистих даних користувачів. Проблема збереження конфіденційності стає все більш актуальною через зростання кібератак, несанкціонованого доступу до особистої інформації та використання даних користувачів в комерційних цілях. Згідно з дослідженнями, близько 60-70% користувачів вважають надмірний збір та використання особистої інформації серйозною проблемою і вимагають більш строгих заходів для захисту даних та прозорості з боку компаній.[1-5]

Окрім того, постійне вдосконалення методів злому підкреслює важливість удосконалення систем аутентифікації для запобігання несанкціонованому доступу. Це висуває вимоги до сучасних і надійних методів захисту, що зумовлює необхідність модернізації алгоритмів аутентифікації. Також, з огляду на зростаючі вимоги до приватності, користувачі стають все більш обізнаними і вимагають високого рівня захисту особистої інформації від соціальних мереж.[6]

Враховуючи політичні та законодавчі аспекти, де законодавство щодо захисту даних користувачів стає більш суворим, модернізація безпеки в соціальних мережах є необхідністю для відповідності сучасним стандартам та виконання вимог законів.

Підвищення довіри користувачів до соціальних мереж залежить не лише від функціональності, але й від рівня безпеки та анонімності, що може значно вплинути на активність користувачів і успішність рекламних кампаній. Отже, в умовах сучасних викликів у галузі інформаційної безпеки та приватності, обрана тема є важливим напрямком дослідження, здатним суттєво покращити якість та безпеку соціальних мереж.

Метою роботи магістерської кваліфікаційної роботи є підвищення рівня конфіденційності та захисту даних у соціальних мережах шляхом автоматизації процесів управління даними, забезпечення анонімності користувачів за

допомогою технології блокчейну та вдосконалення механізмів аутентифікації.

Для досягнення цієї мети передбачено вирішення наступних завдань:

1. Здійснити аналіз предметної області; провести огляд існуючих рівень;
2. Спроектувати модифікований метод збереження даних.
3. Удосконалити алгоритм аутентифікації.
4. Розробити прототип серверної частини систем.
5. Провести тестування.

Об'єктом дослідження: процес аутентифікації та забезпечення анонімності в соціальних мережах.

Предмет дослідження: методи та алгоритми забезпечення анонімності та захисту даних у соціальних мережах..

Методи дослідження. У роботі проведено аналіз сучасних тенденцій у кібербезпеці та методів злому аутентифікації для з'ясування поточних проблем та виклаків. Аналіз існуючих рішень приватності у соціальних мережах, їх класифікація та узагальнення.

Науково-практичний результат роботи полягає у проектуванні та розробці модифікації моделі соціальних мереж, яка шляхом інтеграції технології блокчейну та алгоритмів децентралізованого управління доступом забезпечують підвищений рівень безпеки даних користувачів. На відміну від існуючих аналогів, ця модель дозволяє мінімізувати ризик несанкціонованого доступу та розголошення особистої інформації, гарантуючи конфіденційність і анонімність.

Практична цінність полягає у розробці соціальної мережі, що забезпечить конфіденційність користувачів, безпечного та анонімного спілкування у мережі інтернет, шляхом підвищення рівня безпеки вхідних даних та унеможливлення несанкціонованого доступу.

Метод збереження даних розробляється з урахуванням важливості анонімності, що дозволяє користувачам зберігати особисті дані без ризику їхнього розголошення. Це відкриває нові можливості для тих, хто прагне зберегти свою ідентичність в онлайн середовищі.

Науковий внесок удосконалення алгоритму аутентифікації та методу збереження даних вносить свій внесок у сферу кібербезпеки та приватності, допомагаючи вирішувати актуальні завдання цих областей, адже сьогодні приватність та безпека даних важлива як ніколи.

Апробація результатів дослідження

Результати роботи було представлено на LIII Науково-технічній конференції підрозділів Вінницького національного технічного університету (2024).[7]

Результати роботи було представлено на Всеукраїнській науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: Дослідження, проблеми, перспективи (МН-2023)» [8] та на LII Науково-технічній конференції підрозділів Вінницького національного технічного університету (2023). [9]

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ АЛГОРИТМІВ АУТЕНТИФІКАЦІЇ ТА КОНФІДЕНЦІЙНОСТІ

1.1 Аналіз проблеми та постановка задачі

Поточні алгоритми аутентифікації та збереження даних не завжди забезпечують високий рівень анонімності. Це стає перешкодою для користувачів, які бажають залишатися анонімними в онлайн просторі.

Причиною цього є те, що багато соціальних мереж та месенджерів вимагають від користувачів надання особистої інформації при реєстрації, наприклад, імені, адреси електронної пошти, номера телефону та інших даних, які можуть бути використані для їхньої ідентифікації. Крім того, навіть якщо користувачі не надають особисту інформацію при реєстрації, їхні дії в соціальній мережі можуть бути використані для їхньої ідентифікації. Наприклад, якщо користувач публікує фотографії, на яких він або вона зображений, або якщо він або вона спілкується з іншими користувачами, які знають його або її особисто, це може призвести до його або її ідентифікації.

Існуючі алгоритми аутентифікації можуть бути піддані атакам, таким як фішинг, брутфорс або атаки на паролі, що піддає ризику особисті дані користувачів:

- Фішинг - це вид шахрайства, при якому зловмисники надсилають користувачам електронні листи або повідомлення, які імітують надійні джерела, наприклад, банки або соціальні мережі. Ці повідомлення містять посилання, які ведуть користувачів на підроблені веб-сайти, які виглядають як справжні. Коли користувачі вводять свої особисті дані на цих веб-сайтах, вони попадають до рук зловмисників.
- Брутфорс - це метод взлому, при якому зловмисники використовують програмне забезпечення для перебору всіх можливих комбінацій паролів. Цей метод може бути ефективним, якщо пароль є простим або якщо зловмисник має

доступ до великої кількості інформації про користувача, наприклад, його імені, адреси електронної пошти або місця проживання.

- Атаки на паролі - це методи взлому, які використовують слабкості в алгоритмах хешування паролів. Ці алгоритми використовуються для шифрування паролів, щоб їх було важче зламати. Однак деякі алгоритми хешування паролів можуть бути вразливими до атак, які дозволяють зловмисникам отримати доступ до розшифрованого пароля.[10]

Загалом, проблеми безпеки та конфіденційності в соціальних мережах та месенджерах є серйозними. Вони можуть призвести до серйозних наслідків для користувачів, включаючи крадіжку ідентифікаційної інформації, розголошення конфіденційних даних та порушення особистої приватності.

Щоб вирішити ці проблеми, необхідно впровадити комплекс заходів, які включатимуть:

Покращення анонімності. Для покращення анонімності в соціальних мережах та месенджерах необхідно розробити алгоритми аутентифікації та збереження даних, які не вимагають від користувачів надання особистої інформації. Крім того, необхідно розробити алгоритми, які ускладнюють зловмисникам ідентифікувати користувачів на основі їхньої активності в соціальній мережі.

Захист від атак. Для захисту від атак на алгоритми аутентифікації необхідно використовувати безпечні алгоритми хешування паролів, які не є вразливими до відомих атак. Крім того, необхідно впровадити заходи, які ускладнюють зловмисникам отримувати доступ до особистих даних користувачів, наприклад, шляхом фішингу або брутфорсу.[11]

Ці заходи є складними і вимагають значних зусиль від розробників соціальних мереж та месенджерів. Однак вони є необхідними для захисту користувачів від несанкціонованого доступу до їхніх особистих даних.

1.2 Аналіз існуючих систем аутентифікації та шифрування даних

В інтерконектованому світі, де обмін чутливою інформацією стає необхідністю, системи аутентифікації та шифрування даних є фундаментальними елементами кібербезпеки. Розглянемо різноманітні підходи та технології, що забезпечують безпеку та конфіденційність в обробці інформації.

Системи аутентифікації:

Парольна аутентифікація:

- Це найбільш поширений метод, де користувачі вводять унікальний пароль для доступу.
- Спосіб є популярним, проте слабкі паролі та можливість атак брутфорсу роблять його вразливим.

Двофакторна аутентифікація (2FA):

- Використання двох різних методів, таких як пароль та SMS-код, для підтвердження ідентифікації.
- Це підвищує рівень безпеки, але вимагає додаткових зусиль користувачів.

Біометрична аутентифікація:

- Використання фізіологічних або поведінкових рис для ідентифікації, таких як відбиток пальця, обличчя або голос.
- Це передовий метод, але ризик втрати конфіденційності біометричних даних.

Системи шифрування даних:

Симетричне шифрування:

- Використання одного ключа для шифрування та розшифрування даних.
- Ефективне, але вимагає безпечного обміну ключами.

Асиметричне шифрування:

- Використання пари ключів – публічного та приватного – для забезпечення високого рівня безпеки.

- Більш безпечний, але менш ефективний за швидкодією.

Шифрування на основі хеш-функцій:

- Використання хеш-функцій для перетворення даних в незмінну послідовність байтів.
- Необоротність шифрування, але не підходить для всіх сценаріїв.

У світі, де конфіденційність та безпека є пріоритетами, важливо розуміти різноманітність та ефективність різних систем аутентифікації та шифрування даних. Забезпечення балансу між безпекою та зручністю визначає успіх цих технологій у сучасному цифровому середовищі.

1.3 Типові проблеми розробки соціальних мереж

Розробка соціальної мережі передбачає вирішення різноманітних завдань для забезпечення успіху та зручності використання платформи. Наступні аспекти потребують ретельного розгляду:

Масштабованість: Соціальні мережі зазвичай швидко зростають як за кількістю користувачів, так і за обсягом контенту. Для ефективної обробки великої кількості користувачів, постів і взаємодій важливо забезпечити високу продуктивність. Система повинна бути спроектована з можливістю горизонтального масштабування, що дозволяє ефективно розподіляти навантаження між кількома серверами. Впровадження ефективних механізмів управління базами даних та кешування також є важливим для оптимізації пошуку та зберігання даних.

Продуктивність та швидкість реагування: Користувачі очікують безперебійного та швидкого реагування під час користування соціальною мережею. Мінімізація часу відгуку, зменшення затримок і оптимізація обробки як на сервері, так і на клієнті є критичними для задоволення потреб користувачів. Методи, такі як асинхронна обробка, стиснення даних і

використання мережі доставки контенту (CDN), можуть значно покращити продуктивність.

Безпека і конфіденційність: Захист даних користувачів та забезпечення їхньої приватності є першочерговими завданнями для соціальних мереж. Необхідно впроваджувати надійні заходи безпеки, такі як шифрування, безпечна автентифікація та механізми авторизації. Конфіденційні дані користувачів, включаючи паролі та особисту інформацію, повинні бути належним чином захищені. Важливо також дотримуватися правил захисту даних, таких як GDPR.

Користувацький інтерфейс та досвід: Інтуїтивно зрозумілий, візуально привабливий та зручний користувацький інтерфейс має велике значення для залучення та утримання користувачів. Функції, такі як персоналізовані стрічки, сповіщення, системи обміну повідомленнями та можливості пошуку, повинні бути легко доступними. Відгуки користувачів та юзабіліті-тестування допоможуть у вдосконаленні інтерфейсу.

Модерація контенту: Для підтримки безпечного і здорового середовища соціальних мереж важливо запобігати поширенню неприйнятного, шкідливого або оманливого контенту. Методи модерації, такі як автоматизовані фільтри, механізми звітування користувачів і ручний перегляд, допоможуть виявляти та видаляти небажаний контент.

Управління спільнотами: Ефективне управління спільнотами користувачів і сприяння позитивній взаємодії між ними є вирішальними для формування здорового середовища соціальних мереж. Функції, такі як профілі користувачів, налаштування конфіденційності, дружні зв'язки, групи та дискусійні форуми, сприяють залученню користувачів і створенню активних спільнот.

Інтеграція зі сторонніми сервісами: Інтеграція соціальних мереж з зовнішніми сервісами, такими як провайдери автентифікації (наприклад, OAuth), сервіси обміну контентом (наприклад, API для хостингу зображень і відео) та інструменти аналітики, може розширити функціональність платформи

та покращити користувацький досвід. Важливо забезпечити безпечну та безперешкодну інтеграцію зі сторонніми сервісами.

Підтримка різних платформ: Користувачі використовують соціальні мережі на різних пристроях, таких як веб-браузери, мобільні додатки та планшети. Розробка та підтримка клієнтських додатків для різних платформ є викликом для розробників, але це необхідно для забезпечення зручності користувачів на будь-якому пристрої..

1.4 Аналіз подібних рішень

Існує ряд соціальних мереж та месенджерів, які пропонують анонімність. Ці платформи дозволяють користувачам створювати облікові записи без надання особистої інформації, наприклад, імені, адреси електронної пошти або номера телефону. Крім того, ці платформи використовують алгоритми, які ускладнюють зловмисникам ідентифікувати користувачів на основі їхньої активності в соціальній мережі.

1.4.1 Mastodon

Mastodon – це децентралізована соціальна мережа, яка дозволяє користувачам створювати власні сервери. Mastodon використовує псевдоніми для ідентифікації користувачів.особливості.[12]

Переваги:

- Захист конфіденційності: Mastodon не вимагає від користувачів надання особистої інформації, що допомагає захистити їхню конфіденційність.
- Свобода самовираження: Mastodon дозволяє користувачам вільно висловлювати свої думки та ідеї, не боячись засудження або переслідування.

- Безпека: Mastodon використовує ряд функцій безпеки, таких як шифрування повідомлень і захист від спаму.

Недоліки:

- Може бути складно знайти інших користувачів. Mastodon - це децентралізована соціальна мережа, що означає, що кожен сервер має власну спільноту користувачів. Це може ускладнити для користувачів знайти інших користувачів, які поділяють їхні інтереси.
- Може бути важко захиститися від спаму та тролінгу. Оскільки Mastodon не вимагає від користувачів надання особистої інформації, це може зробити їх більш вразливими до спаму та тролінгу.

1.4.2 Telegram

Telegram - це месенджер, який пропонує ряд функцій конфіденційності, таких як можливість використовувати псевдоніми, шифрувати повідомлення та використовувати режим секретного чату.[13]

Переваги:

- Захист конфіденційності: Telegram дозволяє користувачам використовувати псевдоніми, що допомагає захистити їхню конфіденційність.
- Безпека: Telegram використовує безпечний протокол зв'язку, який ускладнює для зловмисників перехоплення повідомлень.
- Мультиплатформність: Telegram доступний на всіх основних платформах, що дозволяє користувачам спілкуватися з друзями та родиною незалежно від того, на яких пристроях вони використовують Telegram.

Недоліки:

- Може бути використаний для поширення шкідливого контенту. Telegram не має ефективного механізму модерації, що може призвести до поширення шкідливого контенту, такого як мова ненависті або пропаганда.

- Може бути використаний для шахрайства. Telegram не має ефективного механізму захисту від шахрайства, що може зробити користувачів більш вразливими до шахрайства, наприклад, до фішингу або розсилки спаму.

1.4.3 Signal

Signal - це месенджер, який використовує безпечний протокол зв'язку, який ускладнює для зловмисників перехоплення повідомлень. Signal також пропонує ряд функцій конфіденційності, таких як можливість використовувати псевдоніми та шифрувати повідомлення.[14]

Переваги:

- Безпека: Signal використовує безпечний протокол зв'язку, який ускладнює для зловмисників перехоплення повідомлень.
- Конфіденційність: Signal пропонує ряд функцій конфіденційності, які допомагають захистити приватність користувачів, включаючи можливість використовувати псевдоніми та шифрувати повідомлення.
- Довіра: Signal є відкритим вихідним кодом, що означає, що будь-хто може переглянути код і переконатися, що він безпечний і захищений конфіденційністю.

Недоліки:

- Не може використовуватися для масового спілкування. Signal не може використовуватися для масового спілкування, наприклад, для розсилки повідомлень групі людей.
- Може бути складно використовувати. Signal має складний інтерфейс, який може бути важко освоїти для деяких користувачів.

На основі аналізу існуючих аналогів приватних додатків можна зробити такі висновки:

Існує попит на приватні додатки. Користувачі все більше цінують конфіденційність в Інтернеті, і вони готові використовувати додатки, які її

забезпечують. Існує ряд недоліків у існуючих приватних додатках. Деякі з цих недоліків включають складність використання, обмежену функціональність та потенціал для зловживань.

Ці висновки свідчать про те, що розробка нових рішень для анонімності користувачів є конкурентоспроможним та вартим розробки. Нові рішення можуть задовольнити попит на приватні додатки, усунувши недоліки існуючих рішень.

Ось деякі конкретні напрямки, в яких нові рішення можуть бути конкурентоспроможними:

Покращення простоти використання. Нові рішення повинні бути простими у використанні, щоб їх могли використовувати користувачі з різним рівнем технічних знань.

Розширення функціональності. Нові рішення повинні пропонувати більше функцій, ніж існуючі додатки, щоб задовольнити потреби різних користувачів. Забезпечення безпеки та конфіденційності. Нові рішення повинні використовувати найсучасніші технології для забезпечення безпеки та конфіденційності користувачів.

1.5 Аналіз та висновок досліджених аналогів

Аналізуючи різні соціальні мережі, ми можемо виявити загальні недоліки та потенційні сфери для вдосконалення в більш широкому контексті платформ соціальних мереж. Є кілька аспектів, які можна покращити у новій соціальній мережі за допомогою певних інструментів розробки. Пропоную наступні рішення проблем:

Конфіденційність та захист даних:

− Використовуйте надійні механізми аутентифікації та авторизації за допомогою Java Spring Security для безпечного доступу до даних користувачів і захисту від несанкціонованого доступу.

– Використовуйте можливості забезпечення безпеки на рівні документів MongoDB та можливості шифрування для покращення конфіденційності та цілісності користувацьких даних, збережених у базі даних.

– Використовуйте можливості валідації та очищення даних Java Spring Boot для запобігання поширеним вразливостям безпеки, таким як атаки ін'єкцій та атаки міжсайтового скриптування (XSS).

– Впровадьте технології Blockchain для зберігання даних про користувачів у децентралізованій мережі, забезпечуючи високу безпеку та анонімність. Смарт-контракти можуть використовуватися для контролю доступу до даних та забезпечення прозорості операцій.[15]

Розповсюдження неправдивої інформації та модерація контенту:

– Розробіть вдосконалені алгоритми штучного інтелекту з використанням Java і інтегруйте їх з Spring Boot для автоматичного виявлення та позначення потенційно шкідливого або маніпулятивного контенту.

– Впровадіть процес модерації контенту з використанням можливостей агрегації MongoDB для ефективної обробки та перегляду повідомленого контенту.

– Використовуйте бібліотеки обробки природної мови в Java, такі як Apache OpenNLP або Stanford NLP, для аналізу текстового контенту з метою аналізу настрою, виявлення спаму та ідентифікації маніпулюючої інформації.

– Впровадьте смарт-контракти для автоматизації процесів модерації контенту, забезпечуючи прозорість та надійність рішень.

Прозорість алгоритмів та контроль користувача:

– Використовуйте Java Spring Boot для розробки налаштувань користувацьких вподобань, що дозволяють користувачам налаштовувати свій потік контенту, змінювати алгоритми релевантності та контролювати видимість персоналізованих рекомендацій.

– Впровадження механізмів аудиту та ведення журналу в Spring Boot для відстеження та аналізу впливу алгоритмів на фільтрацію контенту та залучення користувачів.

– Використовуйте потужні можливості запитів MongoDB для надання користувачам детального контролю над їхніми вподобаннями щодо контенту та здатністю досліджувати різноманітні точки зору.

– Використовуйте смарт-контракти для забезпечення прозорості алгоритмів та надання користувачам можливості контролювати свої дані.

Психічне здоров`я та благополуччя:

Впровадити аналітику використання соціальної мережі для відстеження та аналізу патернів поведінки користувачів, що дозволяє виявляти можливі проблеми, пов'язані з перебіркою або негативними зайнятостями.

– Розробіть функції для встановлення обмежень використання та надсилання нагадувань користувачам, використовуючи можливості планування Spring для сприяння здоровій соціальній активності.

– Інтегруйте алгоритми аналізу настрою з Spring Boot для виявлення користувачів, які виявляють ознаки стресу або негативних емоцій, та надання їм ресурсів для підтримки психічного здоров'я.

Використання інструментів Java, Spring Boot, MongoDB, Blockchain та смарт-контрактів дозволить значно покращити конфіденційність, вдосконалити модерацію контенту, забезпечити прозорість алгоритмів, сприяти психічному благополуччю, гарантувати автентичність облікових записів та враховувати культурні аспекти. Ці технічні вдосконалення сприятимуть створенню більш безпечного, відповідального та орієнтованого на користувачів соціального досвіду.

1.6 Висновки до першого розділу

Актуальність проекту полягає в тому, що соціальні мережі продовжують займати центральне місце в житті людей. Вони є потужними інструментами комунікації, обміну інформацією та спільної діяльності. Завдяки їм користувачі можуть побудувати особисті та професійні зв'язки, спілкуватися, ділитися

контентом та знаходити нові можливості. Проте, зростає потреба в забезпеченні анонімності та конфіденційності користувачів у сучасних соціальних мережах і месенджерах.

Вибір мови програмування Java та фреймворка Spring Boot є обґрунтованим, оскільки вони мають велику популярність та широкий спектр інструментів для розробки таких систем. Java є потужною, надійною та масштабованою мовою програмування, яка забезпечує високу продуктивність та безпеку. Фреймворк Spring Boot, з своєю екосистемою та готовими рішеннями, спрощує процес розробки та підтримки анонімної соціальної мережі/месенджера.

Анонімність у сучасних соціальних мережах та месенджерах стає все більш актуальною, оскільки користувачі прагнуть захистити свої особисті дані від несанкціонованого доступу та зловживань. Використання технології Blockchain та смарт-контрактів дозволяє забезпечити високий рівень конфіденційності та безпеки. Blockchain забезпечує децентралізоване зберігання даних, що унеможливлює їх зміну або видалення без належного дозволу. Смарт-контракти автоматизують процеси перевірки та аутентифікації, забезпечуючи прозорість і довіру.[16]

Аналіз типових проблем при розробці соціальних мереж допомагає зрозуміти потенційні виклики, з якими можна зіткнутися. Це дозволяє планувати та враховувати такі аспекти, як безпека, масштабованість, продуктивність та інші, під час проектування та реалізації системи. Додатково, забезпечення анонімності та конфіденційності стає ключовим фактором успішності нової платформи.

Дослідження аналогів є цінним джерелом інформації для розробки власної анонімної соціальної мережі. Аналіз їхньої функціональності, архітектури та особливостей дозволяє виявити успішні рішення та розробити власні унікальні функції, які забезпечать конкурентну перевагу нової соціальної мережі. Використання інноваційних технологій, таких як Blockchain і смарт-

контракти, дозволить створити безпечний, анонімний та надійний сервіс для користувачів.[17]

2 АНАЛІЗ ВЕБ АРХІТЕКТУР ТА ІНСТРУМЕНТІВ РОЗРОБКИ

2.1 Технології розробки web-додатків

Технології розробки веб-додатків є невід'ємною частиною сучасного цифрового світу. Вони надають можливість створювати та підтримувати функціональні веб-додатки, доступні через Інтернет. Основні принципи включають використання розподіленої архітектури, де функціональні компоненти розташовані на різних серверах, а клієнти мають доступ до них через мережу. Це забезпечує масштабованість і доступність веб-додатку для великої кількості користувачів.[18]

Основні технології для розробки веб-додатків включають різні мови програмування, а також фреймворки та бібліотеки, які спрощують розробку та обслуговування веб-додатків. Вони дозволяють розробникам ефективно працювати з базами даних, керувати взаємодією з користувачами, робити запити до серверів і обробляти отримані дані. Особливості методів розробки веб-додатків включають можливість створювати інтерактивні інтерфейси, обробляти асинхронні запити, гарантувати безпеку даних і використовувати розширення для розширення функціональності.

Веб-додатки використовуються для різних цілей, включаючи електронну комерцію, соціальні мережі, онлайн-банкінг, управління проектами та інші сфери. Вони дозволяють користувачам отримати доступ до необхідних функцій без встановлення спеціального програмного забезпечення. Крім того, веб-додатки можна оновлювати централізовано, що полегшує розгортання нових версій і виправлення помилок. Завдяки широкому діапазону можливостей і гнучкості методи розробки веб-додатків постійно розвиваються та впроваджуються, щоб задовольнити постійно зростаючі потреби користувачів у цифровому світі.

2.1.1 RESTful

RESTful (Representational State Transfer) - це архітектурний стиль, що використовується при проектуванні веб-додатків та веб-сервісів (рисунок 2.1). Він покладається на принципи та обмеження, які сприяють створенню ефективних, масштабованих та легкодоступних інтерфейсів.

Основні принципи RESTful підходу включають:

- Клієнт-серверна архітектура: Додаток розділяється на дві складові:клієнтську сторону, що забезпечує інтерфейс користувача, і серверну сторону, що зберігає та обробляє дані. Це сприяє розподілу відповідальності та підвищує масштабованість системи.
- Безстанційність: Клієнт і сервер не зберігають стан взаємодії між запитами. Кожен запит містить всю необхідну інформацію для обробки, що дозволяє серверам бути більш масштабованими та зменшує навантаження на серверну пам'ять.
- Кешування: Сервер може позначати відповіді як кешовані або некешовані, що дозволяє клієнтам зберігати локальні копії та покращує продуктивність та ефективність мережевих операцій.
- Єдиності ресурсів: Кожний ресурс (наприклад, користувач, товар, коментар) має унікальний ідентифікатор і може бути доступний через єдиний URL. Це дозволяє просту та послідовну адресацію ресурсів.
- Використання стандартних методів HTTP: RESTful API використовує стандартні HTTP-методи, такі як GET, POST, PUT та DELETE, для забезпечення діалогу між клієнтом та сервером. Кожен метод має своє призначення, що спрощує розробку та розуміння взаємодії

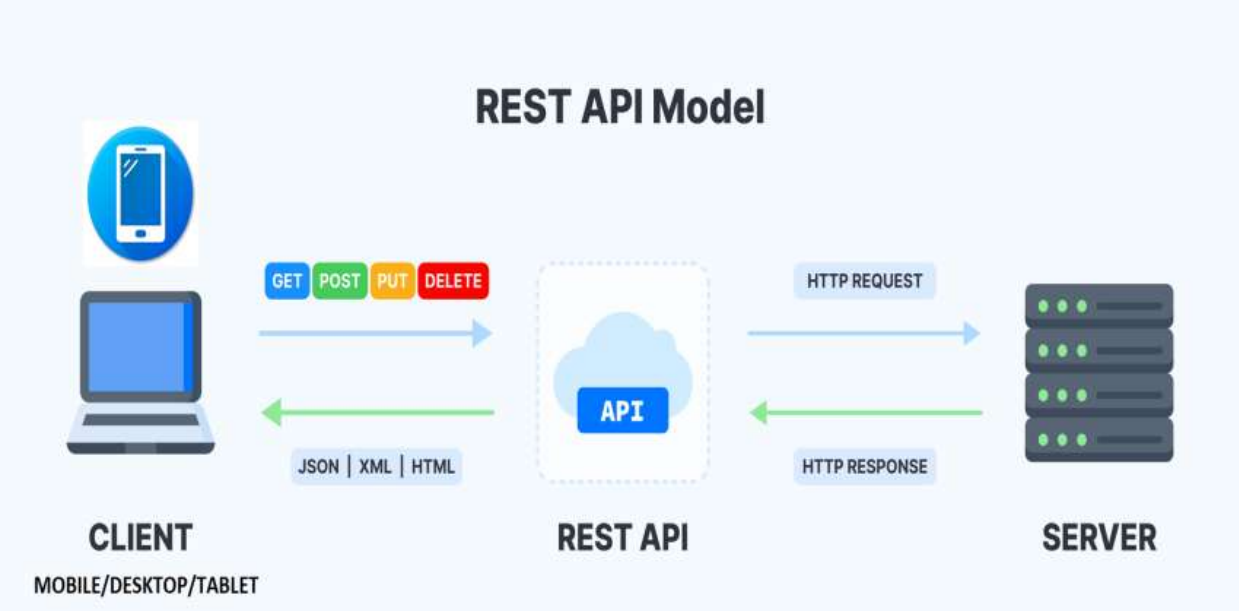

Рисунок 2.1 – Модель роботи RESTful API

Завдяки цим принципам, RESTful підхід сприяє створенню гнучких та розширюваних веб-додатків, що легко інтегруються з іншими системами. Він став широко використовуваним стандартом для розробки API та веб-сервісів у багатьох галузях індустрії.

2.1.2 Огляд можливостей SOAP

SOAP (Simple Object Access Protocol) - це протокол обміну повідомленнями, що використовується для комунікації між різними комп'ютерними системами через мережу, зокрема, за допомогою веб-служб. SOAP був розроблений для забезпечення стандартного способу взаємодії між різними платформами та мовами програмування.

SOAP використовує XML-формат для представлення повідомлень. Це означає, що дані, які передаються між веб-службами, вказуються в структурованому та машинно-читабельному форматі XML. XML дозволяє представляти складні дані та структури і дозволяє використовувати різні типи даних.

Він може використовуватися з будь-яким протоколом передачі даних, таким як HTTP, SMTP або TCP/IP. Однак найпоширенішим протоколом є HTTP, оскільки він підтримується всіма платформами. SOAP повідомлення можуть бути вкладені в протоколи, такі як HTTP POST або HTTP GET, для передачі через мережу.

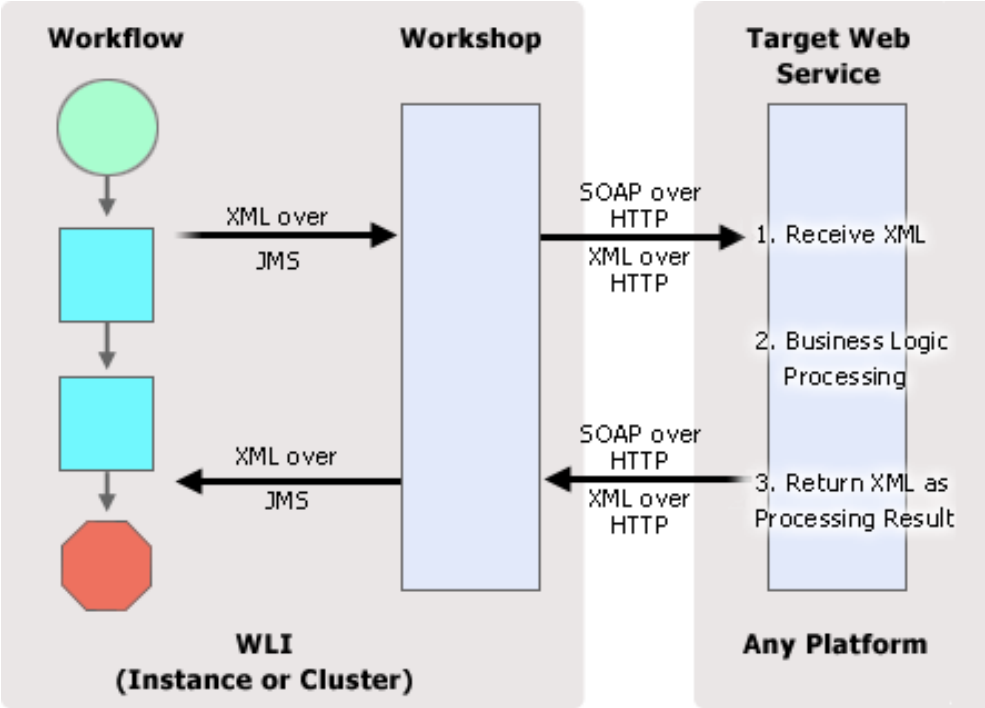

Рисунок 2.2 - Модель роботи SOAP

SOAP архітектура дозволяє використовувати розширення та додаткові функціональні можливості шляхом використання розширюваних елементів XML, таких як простори імен та атрибути. Це дозволяє додавати власні метадані до повідомлень та розширювати функціональність SOAP.

Ця технологія базується на визначенні контракту веб-служби, відомого як WSDL (Web Services Description Language). WSDL описує структуру повідомлень, доступні методи та їх параметри, що дозволяє забезпечити єдність інтерфейсу веб-служби. Клієнти веб-служби можуть використовувати WSDL для генерації клієнтського коду та забезпечення правильної взаємодії з веб-службою.

SOAP надає можливість обробки помилок та відновлення після втрати з'єднання. Він також підтримує шифрування та підписи для забезпечення

безпеки передачі даних. SOAP може використовувати різні протоколи безпеки, такі як SSL (Secure Sockets Layer) або WS-Security, для захисту повідомлень та забезпечення конфіденційності та цілісності даних.

Ці особливості роблять SOAP потужним інструментом для інтеграції та взаємодії між різними системами та платформами через мережу.

Хоч SOAP і є потужним протоколом для веб-сервісів, він має свої недоліки:

1) Складність: SOAP повідомлення мають складну структуру XML, що може призводити до збільшення розміру повідомлень і затримок у передачі даних. Це особливо важливо в обміні великими обсягами даних або при обміні повідомленнями на слабких мережах.

2) Надмірність: SOAP може бути вважаний надмірним для деяких простих веб-сервісних сценаріїв. Його структура та наявність великої кількості технічних деталей можуть зробити розробку та розуміння складними, особливо для початківців.

3) Обмежена підтримка браузерами: Багато сучасних браузерів мають обмежену підтримку SOAP. Це обмежує можливості використання SOAP-протоколу для взаємодії з веб-додатками у веб-браузерах.

4) Важкість використання у веб-орієнтованих архітектурах: SOAP був більш популярним у стандартних SOA (Service-Oriented Architecture) та монолітних архітектурах, ніж у веб-орієнтованих архітектурах. Для простих RESTful API часто вибираються легші альтернативи.

5) Швидкість та продуктивність: Через складну структуру повідомлень та додаткові накладні витрати на обробку SOAP, він може бути менш продуктивним у порівнянні з іншими протоколами, такими як REST.

2.1.3 Огляд можливостей GraphQL

GraphQL - це запитова мова та середовище виконання, яке використовується для взаємодії між клієнтом та сервером веб-додатків. Вона була розроблена компанією Facebook і забезпечує ефективну та гнучку передачу даних між клієнтом та сервером. У відмінність від традиційних REST API, де клієнти отримують фіксований набір даних у відповідь на запити, GraphQL дозволяє клієнтам самостійно визначати, які поля та дані їм потрібні. Клієнт складає запит у вигляді графу з вкладеними полями та відправляє його на сервер GraphQL. Сервер виконує цей запит та повертає клієнту лише ті дані, які були запитані (рисунок 2.3).

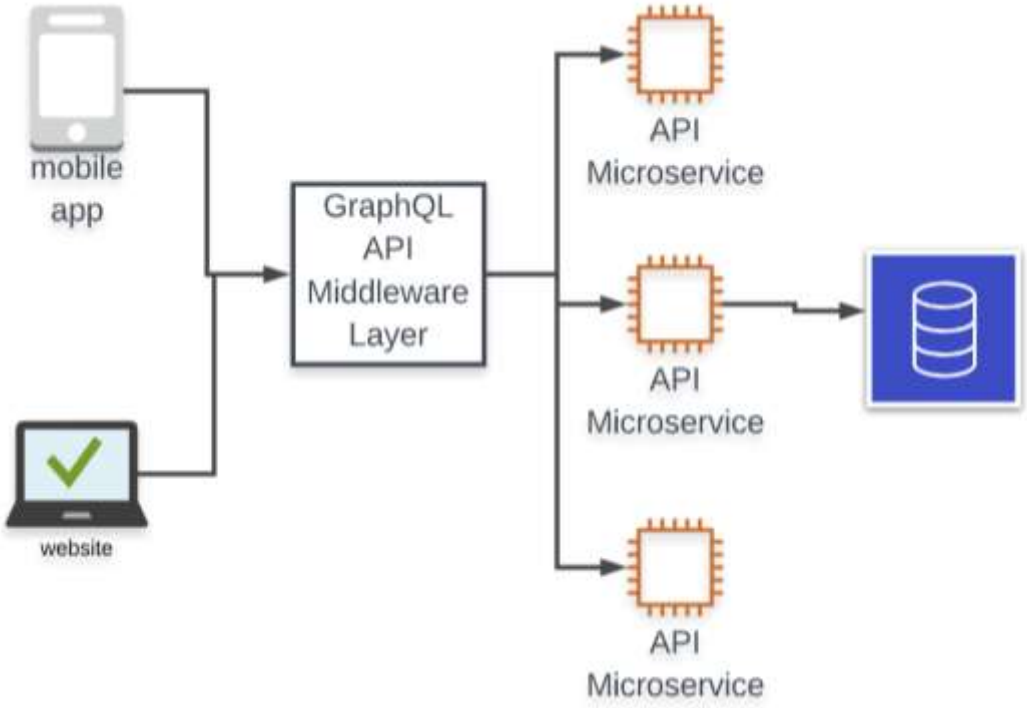

Рисунок 2.3 - Модель роботи GraphQL

GraphQL має свої особливості. GraphQL надає клієнтам повний контроль над даними, які вони отримують. Замість того, щоб отримувати фіксований набір даних, клієнти можуть запитувати саме ті поля, які їм потрібні. Це дозволяє уникнути надлишкового завантаження даних та зменшити обсяг передачі.

Він використовує схему типів для визначення доступних даних та операцій. Схема типів описує структуру даних, доступні поля та їх типи. Це спрощує розробку та зрозумілість API, оскільки клієнти можуть легко дізнатися, які дані можна запитувати та які операції можна виконувати.

У GrpahQL існують резольвери, вони визначають, які дані повинні бути повернуті для кожного полі запиту. Вони виконують функцію обробки запитів та отримання необхідних даних зі сховищ (наприклад, баз даних або зовнішніх API). Резольвери дозволяють гнучко керувати вибором та форматуванням даних, які повертаються клієнту.

GraphQL надає різноманітні інструменти для розробки та відлагодження API. Наприклад, GraphiQL та GraphQL Playground - це інтерактивні інтерфейси, які допомагають розробникам формувати та виконувати запити GraphQL, а також документувати API.

Хоча GraphQL має багато переваг, він також має деякі недоліки, які варто враховувати:

1) Складність конфігурації сервера: GraphQL потребує налагодження резольверів та схеми типів на серверній стороні. Це може бути витратним за часом та зусиллями, особливо для складних додатків зі значним обсягом даних.

2) Загроза "надлишковості запитів" (over-fetching): Хоча GraphQL дозволяє точно визначати необхідні дані, некоректне використання може призводити до передачі зайвих даних. Це може призвести до збільшення обсягу передачі та негативно вплинути на продуктивність.

3) Потенційні проблеми безпеки: GraphQL дозволяє клієнтам динамічно визначати запити, що може створювати потенційні проблеми безпеки. Неправильно налаштований сервер GraphQL може дозволяти клієнтам отримувати надмірну кількість чутливих даних або ставати вразливим до атак, таких як DoS атаки.

4) Складність відладки: У порівнянні з традиційним REST API, де кожен ендпоінт відповідає за конкретний ресурс, відлагодження запитів

GraphQL може бути складнішим. Помилки або некоректність в запиті можуть призводити до складних проблем у виконанні та отриманні даних.

Надмірна гнучкість: Хоча гнучкість GraphQL може бути перевагою, вона також може призвести до недостатньої стандартизації. Без правильного визначення і прийняття стандартів може виникнути ситуація, коли клієнти та сервери використовують різні формати даних та конвенції, що може ускладнити співпрацю та обмін даними між різними компонентами системи.

2.1.4 Порівняння REST, SOAP та GraphQL

Таблиця 2.1 – Порівняння результатів дослідження технологій проектування веб додатків.

Аспекти	RESTful	SOAP	GraphQL
1	2	3	4
Архітектурний стиль	Архітектура заснована на ресурсах	Протокол заснований на XML	Мова запитів та середовище виконання
Формат повідомлення	Зазвичай JSON або XML	XML	Зазвичай JSON, але може використовувати різні формати
Методи передачі даних	CRUD операції (GET, POST, PUT, DELETE)	Розширений набір операцій (SOAP звернення)	Єдиний точний запит з гнучкими полями

Продовження таблиці 2.1

1	2	3	4
URL структура	Використовує різні URL для різних ресурсів	Використовує один URL для всіх операцій	Використовує один URL для всіх запитів
Версіонування API	Часто використовується версіонування в URL або заголовку	Підтримка версіонування	Підтримка версіонування
Швидкодія	Зазвичай швидший, оскільки використовує простий формат повідомлень	Порівняно повільніший, оскільки використовує більш важкі повідомлення	Залежить від реалізації, але загалом ефективний
Кешування	Підтримка кешування на рівні протоколу	Підтримка кешування на рівні протоколу	Немає вбудованої підтримки кешування
Типи безпеки	Базова автентифікація, токени доступу	Розширена підтримка безпеки (WS-Security)	Залежить від реалізації, але загалом потребує налагодження
Складність	Простий для розуміння та використання	Складний для розуміння та налаштування	Може бути складнішим для налаштування та відлагодження
Розмір повідомлення	Зазвичай маленькі повідомлення	Зазвичай великі повідомлення	Залежить від запиту, але загалом ефективний

На основі дослідженої інформації та порівняльної таблиці можна зробити висновок, що найбільш підходящим варіантом для розробки буде RESTful підхід, а саме через наступні причини:

1) Простота використання: REST API має простішу та більш інтуїтивну архітектуру, оскільки використовує стандартні HTTP методи, такі як GET, POST, PUT і DELETE. Це полегшує розуміння та розробку API, особливо для розробників, які вже знайомі з HTTP.

2) Легкість інтеграції: REST API дозволяє інтегрувати веб-сервіси з різними платформами та мовами програмування, оскільки вони використовують загальноприйняті формати даних, такі як JSON або XML. Це дозволяє забезпечити єдність комунікації між системами.

3) Продуктивність та масштабованість: REST API має менше накладних витрат, оскільки використовує простіші формати даних та має меншу обробку метаданих. Це сприяє покращенню продуктивності та масштабованості веб-сервісу, особливо при обробці великих обсягів даних або високому навантаженні.

4) Більша підтримка браузерами: REST API має ширшу підтримку браузерами, оскільки використовує стандартні HTTP методи і формати даних. Це робить REST API більш доступним для веб-орієнтованих додатків, які використовують браузери як клієнти.

5) Простота кешування: REST API підтримує просте кешування, оскільки використовує стандартні заголовки HTTP, такі як Cache-Control і ETag. Це дозволяє зменшити навантаження на сервер та покращити продуктивність, особливо для повторних запитів.

Враховуючи ці аргументи, REST API можна вважати більш простим, легким у використанні та гнучким варіантом для побудови веб-сервіса соціальної мережі порівняно SOAP та GrapghQL.

2.2 Безпека користувачів та їх даних з Blockchain

Безпека користувачів та їхніх даних є критичним аспектом при розробці сучасних веб-додатків, особливо соціальних мереж та месенджерів, де конфіденційність і захист інформації відіграють важливу роль. Використання технологій Blockchain та смарт-контрактів значно підвищує рівень безпеки та анонімності. У цьому підрозділі розглянемо, як ці технології можуть бути інтегровані в архітектуру веб-додатків, їх переваги та конкретні приклади використання.

Blockchain — це децентралізована технологія, яка забезпечує безпечне зберігання даних через розподілену мережу вузлів. Кожен блок у ланцюзі містить набір транзакцій, що є криптографічно захищеними і зв'язаними з попереднім блоком, що забезпечує цілісність та незмінність даних.

Можливості Blockchain:

1) Децентралізація: Відсутність центрального контролюючого органу зменшує ризик однієї точки відмови та забезпечує високий рівень доступності.

2) Прозорість: Кожен учасник мережі може бачити всі транзакції, що сприяє підвищенню довіри між користувачами.

3) Незмінність: Дані, записані у Blockchain, не можуть бути змінені або видалені, що забезпечує високу цілісність інформації.

4) Криптографічна безпека: Використання складних криптографічних алгоритмів захищає дані від несанкціонованого доступу та маніпуляцій.

Переваги:

1) Безпека: Високий рівень захисту від злому та несанкціонованого доступу.

2) Анонімність: Можливість зберігання даних та проведення транзакцій без розкриття особистих даних користувачів.

3) Транспарентність: Всі транзакції є видимими для учасників мережі, що забезпечує довіру.

4) Стійкість до збоїв: Децентралізована природа забезпечує високу надійність та стійкість до збоїв.

Смарт-контракти — це самовиконувані контракти, де умови угоди між сторонами записані у вигляді програмного коду, що зберігається у Blockchain. Вони автоматично виконуються при дотриманні певних умов, що значно знижує ризик людської помилки або шахрайства.

Можливості смарт-контрктів:

1) Автоматизація: Контракти виконуються автоматично при дотриманні заздалегідь визначених умов.

2) Безпека: Смарт-контракти захищені криптографічними методами і зберігаються у незмінній формі.

3) Прозорість: Умови контрактів доступні для перевірки всіма учасниками мережі.

4) Ефективність: Зменшення потреби в посередниках скорочує витрати та час виконання транзакцій.

Переваги смарт-контрактів:

1) Точність: Виключають людську помилку завдяки чіткому дотриманню умов коду.

2) Швидкість: Транзакції виконуються миттєво після дотримання умов.

3) Економія: Відсутність посередників знижує операційні витрати.

4) Довіра: Виконання умов контракту забезпечується програмним кодом, що підвищує довіру між сторонами.

У контексті розробки анонімної соціальної мережі або месенджера, технології Blockchain та смарт-контракти можуть бути використані для досягнення наступних цілей. По-перше, забезпечення анонімності користувачів через використання децентралізованих ідентифікаційних систем на основі Blockchain дозволяє користувачам взаємодіяти без розкриття особистих даних. По-друге, підвищення безпеки повідомлень: повідомлення можуть бути зашифровані та зберігатися у Blockchain, що забезпечує їхню недоторканність і

захист від несанкціонованого доступу. Третє, прозорість і контроль: користувачі можуть контролювати доступ до своїх даних через смарт-контракти, які автоматично виконуються при дотриманні певних умов. Четверте, незмінність даних: використання Blockchain гарантує, що будь-яка інформація, збережена у мережі, не може бути змінена або видалена, що підвищує довіру до платформи. П'яте, автоматизація процесів: смарт-контракти можуть автоматично виконувати різні дії, такі як виплата винагород, управління доступом до контенту або модерація контенту.[19]

Інтеграція цих технологій у веб-архітектуру дозволяє створити більш безпечну, прозору та ефективну платформу, яка відповідає сучасним вимогам конфіденційності та захисту даних. В результаті, користувачі можуть бути впевнені у захищеності своїх даних, а розробники отримують інструменти для автоматизації та покращення процесів взаємодії на платформі.

2.2.1 Роль Blockchain у безпеці даних

Blockchain працює за принципом розподіленого реєстру, що складається з вузлів, які одночасно зберігають копії даних і синхронізуються між собою. Це дає декілька важливих переваг:

Анонімність та децентралізація:

У традиційних централізованих системах конфіденційність часто залежить від довіри до конкретного сервера. У Blockchain користувачі взаємодіють за допомогою криптографічних ключів, що дозволяє забезпечити приватність без розкриття ідентичності.

Наприклад: У вашій анонімній соціальній мережі користувачі можуть реєструватися через децентралізовані ідентифікаційні системи (DID), такі як Hyperledger Indy, де кожен користувач має контроль над власними даними.[20]

Незмінність даних:

Кожна транзакція у Blockchain додається у вигляді нового блоку, який неможливо змінити без зміни всього ланцюга. Це особливо важливо для гарантування правдивості збережених даних.

Практичний приклад: У соціальній мережі це може використовуватися для забезпечення того, що створений контент (наприклад, публікації або коментарі) залишаються автентичними та не можуть бути модифіковані третіми сторонами.

Протидія шахрайству:

Завдяки розподіленій природі Blockchain зловмиснику неможливо скомпрометувати систему без отримання контролю над більшістю вузлів мережі (51%-атака).

2.2.2 Смарт-контракти: автоматизація та ефективність

Смарт-контракти можна розглядати як програмний код, що виконує функції звичайного договору, але у цифровому середовищі.

Як працюють смарт-контракти?

Смарт-контракт записується в Blockchain і виконується автоматично, коли виконуються попередньо визначені умови. Це усуває необхідність у посередниках і забезпечує прозорість.

Наприклад: У вашій мережі смарт-контракти можуть використовуватися для автоматичної модерації контенту. Якщо користувач створює контент, який порушує правила (згідно умов контракту), він блокується автоматично без участі людини.[21]

Додаткові можливості:

- Захист доступу до даних: Користувач може дозволяти або забороняти доступ до своїх даних за допомогою контрактів.

- Моделі винагород: Смарт-контракти можуть виплачувати винагороди за активність у мережі, наприклад, за створення популярного контенту або участь у голосуваннях.

2.2.3 Інтеграція Blockchain у анонімну соціальну мережу

 Анонімна реєстрація та автентифікація:

Користувачі можуть створювати облікові записи за допомогою криптографічних ключів без розкриття особистих даних. DID дозволяють зберігати лише мінімально необхідну інформацію, а права доступу контролюються самими користувачами.

 Захист повідомлень:

Повідомлення між користувачами можуть бути зашифровані та записані у Blockchain. Завдяки цьому сторонні особи не зможуть отримати доступ до змісту комунікацій.

 Прозорість управління:

За допомогою DAO (децентралізованих автономних організацій) користувачі можуть голосувати за впровадження нових функцій або змін у мережі.

2.3 Формулювання вимог до анонімної соціальної мережі

Першим етапом у проектуванні анонімної соціальної мережі на основі блокчейну є визначення вимог до системи.[22] Це передбачає аналіз потреб користувачів, функціональних вимог, а також забезпечення конфіденційності, безпеки та масштабованості. Вимоги формуються з урахуванням орієнтації на цільову аудиторію, функціоналу системи, забезпечення захисту даних, анонімності користувачів, інтеграції з іншими платформами та інших важливих аспектів. Ось ключові вимоги до системи:

1. Реєстрація користувачів:

Система повинна надавати можливість реєстрації нових користувачів без збереження особистої інформації. Реєстрація може базуватися на криптографічних методах, таких як створення унікальних ідентифікаторів на основі блокчейну, щоб забезпечити анонімність.

2.Автентифікація та авторизація:

Для доступу до функціоналу системи повинні використовуватися децентралізовані механізми автентифікації, наприклад, за допомогою приватних ключів або інших криптографічних методів. Це забезпечує безпеку доступу до даних і анонімність користувачів.[23]

3.Особистий профіль користувача:

Кожен користувач повинен мати профіль, що містить мінімальний обсяг даних і налаштування конфіденційності. Інформація в профілі повинна бути зашифрованою і доступною лише за згодою користувача.

4. Соціальні зв'язки:

Система має забезпечувати механізми встановлення зв'язків між користувачами, наприклад, через унікальні ідентифікатори. Повідомлення про запити на додавання до списку контактів та інші взаємодії повинні бути зашифрованими.

5.Публікації та стрічка новин:

Користувачі повинні мати можливість створювати публікації, ділитися контентом (текстом, фото, відео), зберігаючи контроль над тим, хто може переглядати їхній вміст. Контент має зберігатися в розподіленому сховищі для забезпечення надійності та конфіденційності.

6.Повідомлення та спілкування:

Приватні повідомлення, чати та коментарі повинні бути зашифрованими кінцевим шифруванням, щоб уникнути витоку даних або доступу до інформації сторонніх осіб.

Дотримання цих вимог гарантує, що система буде відповідати принципам анонімності, безпеки та децентралізації, забезпечуючи користувачам максимальний рівень довіри та зручності у використанні.

2.4 Огляд та вибір інструментів розробки

Розробка сучасного веб-додатку вимагає ретельного підбору інструментів, які забезпечують ефективність, продуктивність, масштабованість і зручність використання. Особливо це стосується систем, орієнтованих на анонімність і безпеку, як у випадку з розробкою соціальної мережі. Нижче наведено ключові інструменти та технології, які було обрано для реалізації проекту, з поясненням їхнього значення.

Мови програмування. Серцем будь-якого програмного продукту є мова програмування. Для цього проекту обрано Java, яка завдяки своїй зрілості, стабільності та багатій екосистемі стала стандартом для розробки корпоративного ПЗ. Java пропонує потужні можливості обробки даних, розширюваність і високий рівень безпеки, що робить її ідеальним вибором для проекту, де анонімність і захист даних користувачів мають вирішальне значення.

Для інтеграції блокчейну, який є важливою частиною проекту, використовується бібліотека Web3j. Вона дозволяє легко взаємодіяти з блокчейн-мережею, створювати та виконувати смарт-контракти, а також забезпечує прозорість і незмінність даних, що особливо важливо для створення довіри користувачів до системи.

Фреймворки. Фреймворки задають основу для структури програми. Обраний для проекту Spring Boot значно спрощує розробку складних серверних додатків. Він автоматизує налаштування, надає багаті можливості для роботи з безпекою (через Spring Security), а також легко інтегрується з базами даних за допомогою Spring Data JPA. Завдяки цим інструментам створення REST API та інтеграція з іншими модулями системи стають швидкими й ефективними.[24]

Наприклад, реалізація функціоналу чату в режимі реального часу здійснюється за допомогою Spring WebSocket, що забезпечує високу швидкість передачі даних і стабільність з'єднання. Використання фреймворку також

допомагає дотримуватися принципів модульності й полегшує підтримку додатку в майбутньому.

Технології для масштабування та контейнеризації

Масштабованість є однією з ключових характеристик, необхідних для успішної роботи соціальної мережі. З ростом кількості користувачів додаток має безперебійно обробляти збільшення запитів без втрати продуктивності. Для цього використовуються Docker і Kubernetes.

Docker дозволяє створювати ізольовані контейнери для кожного компонента додатку. Це забезпечує легкість розгортання, незалежність середовищ і скорочення часу на налаштування. Наприклад, сервер додатку, база даних і сервер кешування можуть працювати в окремих контейнерах, що спрощує управління.

Kubernetes, у свою чергу, є потужним інструментом для оркестрації цих контейнерів. Він дозволяє автоматично розподіляти навантаження, забезпечує відмовостійкість системи, а також підтримує горизонтальне масштабування додатку. Наприклад, якщо кількість активних користувачів різко зростає, Kubernetes автоматично додасть необхідну кількість екземплярів серверів для обробки запитів.

Засоби кешування

Кешування є важливим елементом, що дозволяє значно знизити навантаження на сервер і базу даних, одночасно підвищуючи швидкість відповіді системи. У цьому проекті використовується Redis, оскільки він не лише швидкий і надійний, але й підтримує складні структури даних. Наприклад, для зберігання сесій користувачів або тимчасових даних Redis пропонує прості й ефективні рішення.

Як альтернатива розглядалася Memcached, але його функціональність обмежена простими ключ-значення парами, що може бути недостатньо для складних операцій. У нашому проекті Redis виграє завдяки підтримці списків, множин та інших структур, що спрощує управління даними.

Інтеграція Blockchain

Для забезпечення анонімності й прозорості використовується технологія Blockchain. Зокрема, обрано бібліотеку Web3j, яка дозволяє Java-додатку взаємодіяти з Ethereum. Смарт-контракти, розроблені для системи, забезпечують автоматизацію управління правами доступу та записами в блокчейн. Це підвищує довіру до системи, адже дані користувачів залишаються захищеними, незмінними та доступними тільки за попередньо визначеними умовами.

Хмарні платформи

Для розгортання додатку використовується AWS, яка зарекомендувала себе як одна з найнадійніших і найгнучкіших хмарних платформ. AWS пропонує широкий набір інструментів, таких як:

- Amazon EC2 для запуску серверів.
- Amazon RDS для зберігання даних у реляційних базах.
- Amazon S3 для зберігання файлів.

AWS також легко інтегрується з Docker і Kubernetes, що забезпечує простоту налаштування й управління.

Висновки

Для реалізації анонімної соціальної мережі було обрано Java, Spring Boot, AWS, Docker, Kubernetes та Redis, оскільки ці інструменти забезпечують максимальну продуктивність, надійність і масштабованість системи. Інтеграція з Blockchain через Web3j гарантує високий рівень конфіденційності даних і довіри користувачів.

Ці технології не лише відповідають сучасним вимогам до розробки додатків, а й забезпечують гнучкість для розширення системи в майбутньому. Комбінація обраних інструментів дозволяє створити інноваційний продукт, що відповідає високим стандартам якості та безпеки.

2.5 Архітектурний підхід для Blockchain системи

Архітектура анонімної соціальної мережі, що використовує Blockchain і Smart-контракти, базується на модульній структурі та розділенні компонентів за функціональними ролями. Основний архітектурний шаблон — MVC (Model-View-Controller), який забезпечує чіткий розподіл обов'язків між компонентами системи (рисунок 2.4).

Компоненти архітектури MVC:

1) Модель (Model):

Відповідає за обробку даних і логіку бізнес-процесів. У контексті використання блокчейну, модель включає взаємодію зі смарт-контрактами, які забезпечують децентралізоване управління даними. Смарт-контракти виконують функції запису, оновлення та перевірки транзакцій у блокчейні. Наприклад, дані про користувачів, запити на додавання друзів або дії з контентом зберігаються у блокчейні через смарт-контракти, що забезпечує прозорість і безпеку.

2) Представлення (View):

Відповідає за відображення інтерфейсу користувача. Це можуть бути веб-додатки, побудовані з використанням HTML, CSS і JavaScript, або мобільні додатки, які отримують дані через API. Представлення взаємодіє з контролером для отримання даних зі смарт-контрактів і відображення їх користувачу.

3) Контролер (Controller):

Обробляє запити від користувача, взаємодіє з моделлю для отримання або запису даних у блокчейн через смарт-контракти та повертає відповідь представленню. Контролер також може включати логіку автентифікації, авторизації та обробки помилок, що базується на децентралізованих механізмах.

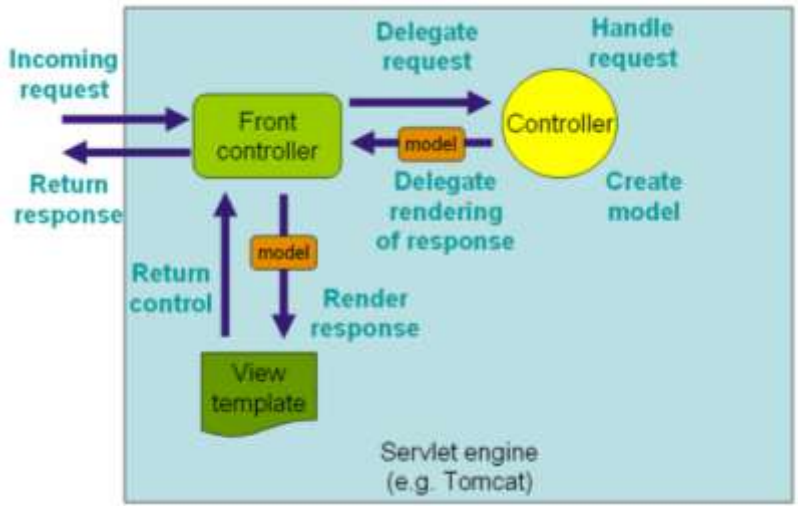

Рисунок 2.4 - Схема MVC

Rest API:

REST API відіграє важливу роль у забезпеченні взаємодії між користувачами та системою. У контексті блокчейну REST API використовується для:

- Отримання даних: Доступ до записів у блокчейні через смарт-контракти.
- Запису даних: Створення транзакцій у блокчейні (наприклад, створення публікацій або запитів на дружбу).
- Інтеграції: Дозволяє стороннім додаткам взаємодіяти із соціальною мережею через відкритий API.

REST API забезпечує структурованість і масштабованість, дозволяючи користувачам працювати з соціальною мережею через веб-додатки, мобільні пристрої або інші платформи.

Модульність системи:

Кожен функціональний модуль побудований на основі взаємодії зі смарт-контрактами:

- Управління профілями користувачів: Смарт-контракти зберігають дані про користувачів у блокчейні. Доступ до цих даних здійснюється через ключі, що гарантують анонімність.
- Повідомлення та дружба: Смарт-контракти забезпечують зберігання запитів на дружбу, переписок та інших взаємодій між користувачами.

- Публікації та новини: Контент (пости, коментарі) зберігається у розподілених файлових системах із посиланнями, які фіксуються у блокчейні.

Безпека та децентралізація

Для забезпечення безпеки даних і аутентифікації:

- Використовується децентралізована автентифікація, заснована на цифрових підписах і приватних ключах. Користувачі взаємодіють із системою без необхідності передавати паролі.
- Протокол OAuth2.0 замінюється на механізми блокчейну для делегування доступу через смарт-контракти.

Використання Blockchain і Smart-контрактів

Смарт-контракти забезпечують:

- Анонімність: Дані зберігаються у блокчейні без прив'язки до реальних особистостей.
- Незмінність: Усі транзакції є незмінними, що виключає можливість підробки даних.
- Децентралізацію: Відсутність центрального сервера, що гарантує відсутність єдиної точки відмови.

Зберігання даних

Дані користувачів та контенту зберігаються в:

- Блокчейні: Для зберігання ключових даних і транзакцій.
- Розподілених файлових системах (IPFS): Для великих даних (фото, відео). IPFS забезпечує швидкий доступ і інтеграцію з блокчейном через хеш-посилання.

Переваги архітектури

Прозорість: Завдяки блокчейну всі транзакції відкриті для аудиту.

- Безпека: Захист даних через криптографію і відсутність єдиної точки відмови.
- Масштабованість: Завдяки модульній структурі й розподіленим системам зберігання.

- Анонімність: Гарантована через використання цифрових підписів і децентралізованого управління доступом.
- Цей підхід дозволяє створити гнучку, безпечну та сучасну соціальну мережу, яка відповідає вимогам конфіденційності та анонімності.

2.6 Архітектура бази даних

У соціальній мережі на основі Blockchain та Smart-контрактів підхід до архітектури бази даних і зберігання файлів змінюється порівняно з традиційними рішеннями. Оскільки блокчейн не призначений для зберігання великих обсягів даних, база даних використовується у гібридному підході:

Блокчейн для критичних даних:

Зберігаються транзакції та метадані, які забезпечують прозорість і незмінність:

- Хеші даних (наприклад, файлів чи записів у базі даних), які підтверджують їхню автентичність.
- Відомості про користувачів, записані у вигляді цифрових ідентифікаторів.
- Логи транзакцій (наприклад, створення постів, запити на дружбу).

Реляційна чи NoSQL база даних для допоміжних даних:

MongoDB (NoSQL) або PostgreSQL (реляційна) може бути використана для зберігання даних, які потребують швидкого доступу та структурованості:

- Інформація профілів користувачів (зображення аватарів, описи, налаштування).
- Метадані файлів (посилання на файли в IPFS).
- Локальні кеші новин або повідомлень для швидкого відображення.
- Ці дані не є критичними з точки зору незмінності, але вони забезпечують ефективність роботи додатка.

Децентралізовані файлові системи:

Великі файли (зображення, відео, документи) зберігаються у IPFS або аналогах:

- Файли розподіляються між вузлами мережі.
- У блокчейні зберігається лише хеш файлу, що дозволяє перевіряти його цілісність.

- Посилання на файл передається через REST API до клієнтської частини.

2.6.1 Організація структури даних у Blockchain

У блокчейні зберігаються такі дані:

1) Таблиця користувачів:

- Унікальний ідентифікатор (адреса гаманця або хеш).
- Публічний ключ для автентифікації.
- Хеши важливих даних (наприклад, аватарів чи біографії).

2) Таблиця постів

- Унікальний ідентифікатор посту.
- Адреса автора (ідентифікатор користувача).
- Хеш контенту (зображення чи текст у IPFS).
- Таймстемп (час створення посту).

3) Таблиця зв'язків між користувачами:

- Взаємодії (запити на дружбу, підтвердження).
- Метадані (хеши транзакцій у блокчейні).

4) Таблиця повідомлень:

- Хеш повідомлення, яке може бути зашифроване приватними ключами користувачів.
- Відправник і отримувач (ідентифікатори).

2.6.2 Зберігання файлів

1) IPFS (InterPlanetary File System):

- Файли зберігаються у вузлах IPFS і отримують унікальні хеши.

- Хеш файлу записується у блокчейн для перевірки його автентичності.
- Переваги:
  - Масштабованість: IPFS розподіляє файли між багатьма вузлами.
  - Надійність: Дані не втрачаються, навіть якщо окремі вузли виходять з ладу.
  - Економія ресурсів блокчейну.

2) Резервне зберігання на сторонніх хмарних сервісах (опціонально):

- Використовується для додаткової доступності та швидкості (наприклад, AWS S3, Google Cloud Storage).
- Посилання на хмарне сховище може також бути зашифрованим і зберігатися в IPFS.

3) Шифрування файлів:

Перед збереженням у IPFS файли можуть бути зашифровані:

- Ключ доступу зберігається у блокчейні або передається через смарт-контракти.
- Тільки авторизовані користувачі можуть отримати доступ до файлів.

2.6.3 Модифікований метод збереження даних

Збереження даних у сучасних системах потребує не лише надійності, але й високого рівня захисту. Особливо це актуально для соціальних мереж, які працюють із чутливою інформацією користувачів. У цій роботі запропоновано модифікований підхід до збереження даних, що базується на інтеграції Blockchain та реляційної бази даних. Такий гібридний підхід забезпечує як незмінність і децентралізацію для критичних даних, так і зручність доступу та швидкість обробки для менш важливих метаданих.[25]

Вимоги до нового методу:

1. Підтримка децентралізації. Для забезпечення прозорості та захисту від маніпуляцій, критичні дані повинні зберігатися в розподіленій мережі. Blockchain надає ідеальну платформу для цього завдяки своїй природній незмінності.
2. Захист від змін у даних після запису. Blockchain дозволяє записувати хеші даних, гарантує незмінність, оскільки будь-яка спроба модифікації інформації змінює її хеш, що миттєво виявляється.
3. Шифрування даних перед збереженням. Оскільки навіть метадані можуть містити чутливу інформацію, дані повинні бути зашифровані перед збереженням. Це мінімізує ризик компрометації навіть у разі витоку.

Гібридний підхід до збереження даних передбачає двох основних компонентів, а саме: Blockchain – для збереження критичних даних, таких як хеші інформації, або підтвердження транзакції і не реляційна база даних MongoDB – для зберігання супутніх метаданих, які не потребують незміності, але повинні бути доступними для швидкої обробки.

На рисунку 2.5 зображено смарт контракт, який зберігає хеші даних, що асоціюються з конкретним користувачем, забезпечуючи їх захист від змін.

```
pragma solidity ^0.8.0;

contract DataStorage {
    struct Record {
        string dataHash;
        uint timestamp;
    }

    mapping(address => Record) private records;

    function storeData(string memory dataHash) public {
        records[msg.sender] = Record(dataHash, block.timestamp);
    }

    function retrieveData() public view returns (string memory, uint) {
        return (records[msg.sender].dataHash, records[msg.sender].timestamp);
    }
}
```

Рисунок 2.5 - Смарт контракт

Перед збереженням даних вони проходять етап шифрування, який реалізовано за допомогою Java, що зображено на рисунку 2.6. Застосовано симетричний алгоритм блочного шифрування AES для, що забезпечує швидкість і надійність.

```java
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class DataEncryptor {

    public static String encrypt(String plainText, SecretKey key) throws Exception {
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }

    public static void main(String[] args) throws Exception {
        String data = "Sensitive user data";
        SecretKey key = new SecretKeySpec("1234567890123456".getBytes(), "AES");

        String encryptedData = encrypt(data, key);
        System.out.println("Encrypted Data: " + encryptedData);
    }
}
```

Рисунок 2.6 - Шифрування даних перед збереженням

Запропонований модифікований метод збереження даних інтегрує децентралізацію та шифрування для підвищення безпеки, використовує Blockchain для збереження критичних даних, забезпечуючи їх незмінність, і оптимізує продуктивність за рахунок розподілу навантаження між блокчейном і реляційною базою даних, що робить його ефективним для масштабованих систем, які потребують високої швидкості обробки та надійного захисту інформації.[26]

2.6.4 Приклад сценарію збереження даних

- Користувач завантажує зображення для публікації.
- Зображення шифрується і завантажується в IPFS.

- Отриманий хеш файлу записується в блокчейн через смарт-контракт разом з ID користувача, описом і таймстемпом.
- Додаток записує метадані файлу (назва, розмір, посилання на IPFS) у MongoDB для швидкого доступу.

Переваги такого підходу:

- Незмінність: Дані, записані у блокчейн, не можуть бути змінені або видалені.
- Масштабованість: Використання IPFS і баз даних дозволяє обробляти великий обсяг даних.
- Безпека: Шифрування та цифрові підписи захищають файли та транзакції.
- Гнучкість: Поєднання блокчейну, IPFS і баз даних забезпечує високу продуктивність та зручність.
- Цей підхід ідеально підходить для децентралізованої соціальної мережі, яка вимагає безпеки, прозорості та масштабованості.

2.7 Висновки до другого розділу

У цьому звіті з магістерської кваліфікаційної роботи було проведено аналіз предметної області та існуючих рішень для створення анонімної соціальної мережі або месенджера. Визначено призначення, цілі та задачі розробки, а також основні особливості та критерії користування майбутньої системи.

Під час аналізу було виявлено необхідність розробки нової системи аутентифікації та конфіденційності, яка забезпечуватиме високий рівень безпеки та анонімності користувачів, а також буде зручною у використанні. Інтеграція технологій Blockchain та смарт-контрактів була визначена як ключовий напрямок для досягнення цих цілей.

Існуючі системи часто не забезпечують достатнього рівня безпеки та конфіденційності, використовуючи прості методи аутентифікації, такі як паролі, або більш безпечні, але дорогі та складні методи, як-от двофакторна аутентифікація чи біометрія. Blockchain дозволяє децентралізовано зберігати дані, забезпечуючи їх незмінність, а смарт-контракти підвищують прозорість і контроль користувачів над своїми даними.

Таким чином, розробка анонімної соціальної мережі або месенджера з використанням Blockchain та смарт-контрактів має на меті створення безпечної, прозорої та ефективної платформи, яка відповідатиме сучасним вимогам конфіденційності та захисту даних., яка відповідатиме сучасним вимогам конфіденційності та захисту даних.

3 РЕАЛІЗАЦІЯ СИСТЕМИ СОЦІАЛЬНОЇ МЕРЕЖІ

Планування та проектування є ключовими етапами у розробці анонімної системи соціальної мережі з використанням блокчейну. На цьому етапі визначається напрямок розвитку платформи, її функціонал і можливості, з акцентом на забезпечення конфіденційності та безпеки даних користувачів. Прийняті архітектурні рішення відіграють важливу роль у забезпеченні гнучкості, масштабованості та надійності системи. Необхідно розробити ефективну структуру та описати алгоритми роботи, що враховують специфіку використання блокчейну для децентралізованого управління даними.

Ключовим завданням є створення загальної архітектури системи, визначення її компонентів та модулів, а також моделювання їхньої взаємодії. Особлива увага приділяється структурам даних, які забезпечують анонімність та безпеку. Для цього потрібно створити діаграми, макети та інші візуальні матеріали, які демонструють, як буде функціонувати система, і забезпечують комплексний підхід до її розробки.

3.1 Налаштування проєкту

Для розробки буде використано середовище розробки Intellij IDEA. Далі необхідно створити проект за допомогою генератора Spring Initializr, вказати назву проекту, обрати мову програмування та її версію, обрати інструментом побудови та управління проектом Maven. (рисунок 3.1)

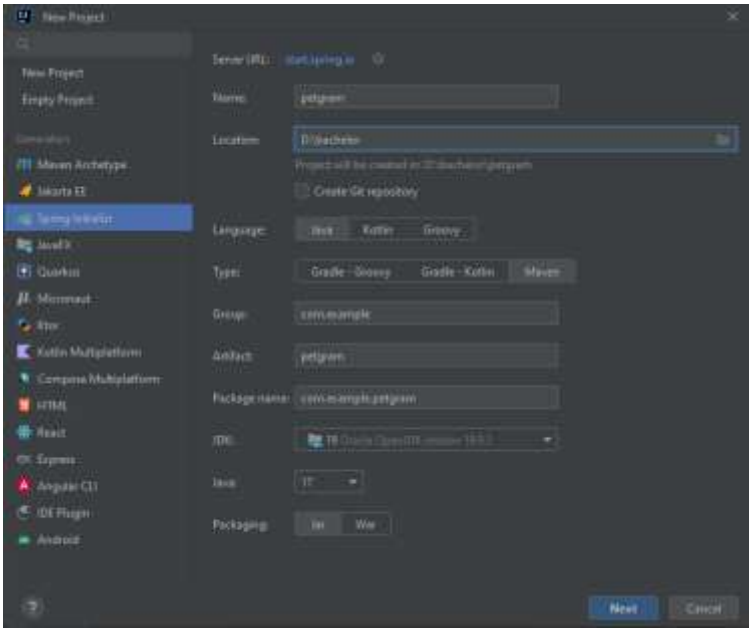

Рисунок 3.1 - Вікно створення проекту в Intellij IDEA

Наступний крок – це обрання версії фреймворка Spring Boot та його модулів як показано на рисунку 3.2.

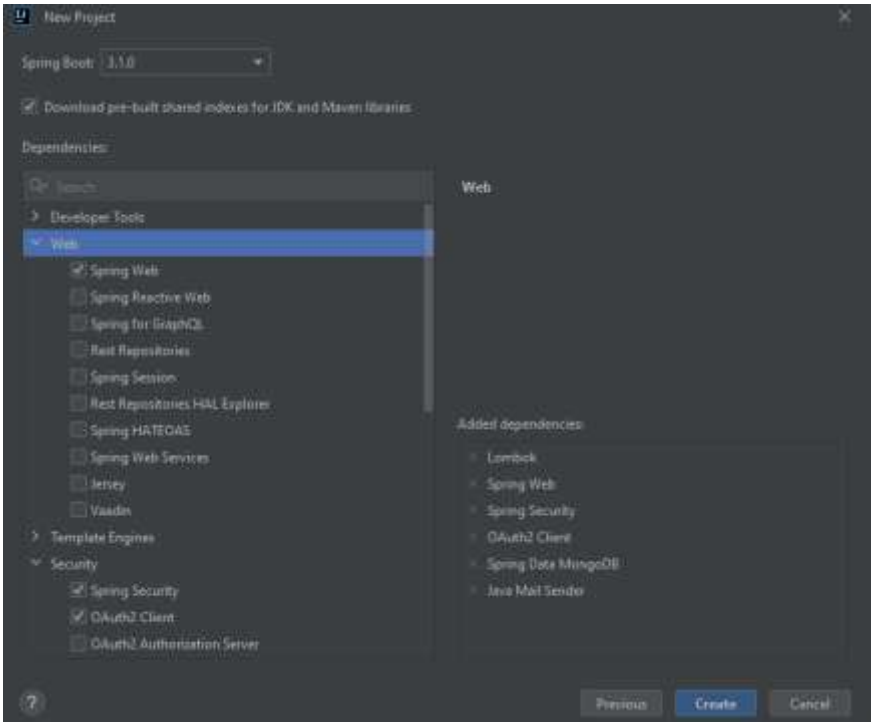

Рисунок 3.2 - Вікно для налаштування Spring Boot

В результаті створення проекту буде отримано теку з базовими теками, файлом запуску та різні конфігураційними файлами. (рисунок 3.3)

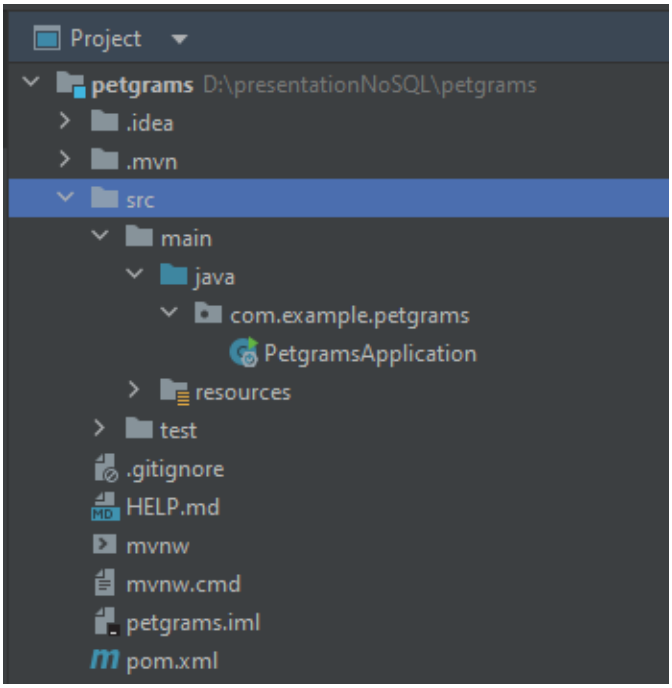

Рисунок 3.3 - Вікно для налаштування Spring Boot

На початку розробки необхідно визначитись з модулями програми та одразу створити для неї всі необхідні директорії. Основна мета - забезпечити чистоту, логічну розбитість та легкість утримання коду. Нижче наведено опис різних директорій, які будть використані у проекті (рисунок 3.4):

- *config*: У цій директорії зберігаються файли конфігурації, такі як конфігурація бази даних, конфігурація безпеки або конфігурація веб-сервера. Ця директорія також включає конфігурацію для блокчейну, смарт-контрактів та взаємодії з мережею блокчейну. Наприклад, BlockchainConfig створює об’єкт для підключення до блокчейну через Web3j:
- *controller*: У цій директорії розміщуються класи-контролери, які обробляють HTTP-запити та взаємодіють з клієнтами. Контролери визначають шляхи (endpoints) та поведінку API. Також вони обробляють не лише запити до бази даних, але й взаємодію зі смарт-контрактами. Наприклад, BlockchainController може містити методи для реєстрації користувача через смарт-контракт:

- *DTO*: Ця директорія містить класи Data Transfer Object (DTO), які використовуються для передачі даних між клієнтом і сервером. DTO допомагають уникнути розкриття внутрішньої структури доменних об'єктів і забезпечують гнучкість у виборі полів, які повертаються або приймаються.

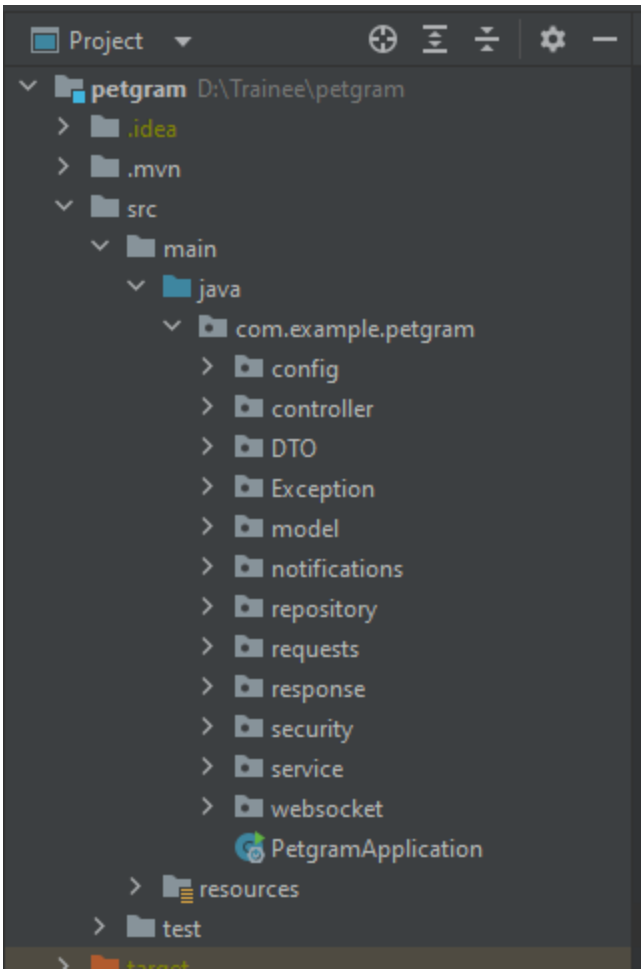

Рисунок 3.4 - Структура проекту

- *Exception*: У цій директорії знаходяться класи, пов'язані з обробкою винятків. Вони використовуються для створення та обробки виключень, які можуть виникати під час виконання операцій.
- *model*: У цій директорії знаходяться класи, які представляють моделі даних або сутності в системі. Вони відображають структуру та властивості об'єктів, які зберігаються в базі даних або передаються через API.

- *notifications*: Ця директорія містить класи, пов'язані з нотифікаціями або повідомленнями, які можуть відправлятись користувачам. Вона може містити логіку відправки, отримання та обробки повідомлень.
- *repository*: У цій директорії знаходяться інтерфейси або класи репозиторіїв, які використовуються для взаємодії з базою даних. Вони надають методи для зберігання, вибірки, оновлення та видалення даних.
- *security*: Ця директорія містить класи та конфігурацію, пов'язані з безпекою додатку. Вона може включати класи для аутентифікації, авторизації, захисту ресурсів та обробки токенів.

3.2 Розробка модулів програми

У програмі, яка базується на Blockchain і Smart-контрактах**,** кожен етап роботи системи, включно з реєстрацією користувачів, обробкою їхніх даних і зберіганням файлів, організований таким чином, щоб гарантувати прозорість, безпеку та захищеність даних. Для цього застосовується модульний підхід, який дозволяє розділити функціонал програми на логічно завершені компоненти. Це сприяє не лише спрощенню розробки та підтримки, але й забезпечує гнучкість у випадку, якщо система потребуватиме розширення чи модернізації.[27]

3.2.1 Модуль реєстрації користувачів

Цей модуль відповідає за первинний контакт користувача із системою. Він дозволяє створити унікальний обліковий запис, використовуючи інноваційні підходи до ідентифікації особи за допомогою криптографії. Усе це відбувається в максимально безпечному середовищі.

Процес:

- Користувач взаємодіє з формою реєстрації, вносячи особисті дані, такі як ім'я, email та бажаний метод автентифікації.
- На стороні клієнта відбувається генерація криптографічної пари ключів:
- Приватний ключ: залишається виключно у користувача, підвищуючи рівень безпеки.
- Публічний ключ: використовується для реєстрації в системі.
- Дані, введені користувачем, шифруються, після чого передаються на сервер для обробки.
- Смарт-контракт UserContract реєструє нового користувача в блокчейні, додаючи:
  - Унікальний ідентифікатор (адресу гаманця користувача).
  - Публічний ключ для перевірки майбутніх запитів.
  - Хеш профільних даних для контролю їхньої цілісності.
  - Додаткові дані, що не є критичними для блокчейна, передаються у децентралізовану систему зберігання (наприклад, IPFS), забезпечуючи масштабованість і зниження витрат на обробку транзакцій.

3.2.2 Модуль управління профілем користувача

Цей модуль дозволяє користувачам налаштовувати інформацію про себе, додавати або редагувати персональні дані, такі як фотографії, біографія або контактна інформація. Особливістю є можливість тонкого налаштування рівнів приватності, які визначають, хто саме має доступ до тих чи інших елементів профілю.

Процес:

- Після автентифікації користувач взаємодіє із формою редагування профілю.

- Внесені зміни автоматично шифруються та зберігаються в IPFS, де генерується унікальний хеш для кожного запису.
- Хеш профілю записується в блокчейн через ProfileContract, забезпечуючи захист і контроль версій даних.

3.2.3 Модуль автентифікації та авторизації

Цей модуль забезпечує безпечний і зручний спосіб входу користувача до системи. Він використовує сучасні методи автентифікації, зокрема криптографічну перевірку даних, що підвищує довіру до платформи та гарантує, що доступ до системи отримують лише авторизовані користувачі.

Процес:

- Користувач використовує свій приватний ключ для входу в систему.
- Застосунок генерує цифровий підпис на основі приватного ключа, який перевіряється блокчейном за допомогою публічного ключа користувача.
- Після успішної перевірки користувач отримує тимчасовий токен доступу (наприклад, JWT), який дозволяє використовувати функціонал системи протягом певного часу.

3.2.4 Модуль взаємодії з Blockchain

Модуль є "посередником" між програмою та блокчейном. Його основне завдання — забезпечити прозору та ефективну взаємодію із блокчейном через смарт-контракти, а також мінімізувати витрати на транзакції.

Процес:

- Коли потрібно виконати нову транзакцію (наприклад, зареєструвати користувача), модуль взаємодіє з вузлами блокчейна.
- Транзакція перевіряється і додається до блокчейна.
- Користувач отримує ідентифікатор транзакції, який дозволяє відстежувати її статус у реальному часі.

Основні функції смарт-контрактів:

- UserContract: записує унікальні ідентифікатори користувачів і публічні ключі.
- ProfileContract: працює з хешами профільних даних і файлами.
- AccessControlContract: керує доступом до даних користувачів.

Інструмент управлінням проектом створює файл pom.xml (рисунок 4.6), в цьому файлі одразу додаються залежності які ми обрали у Spring Initialzr. Цей додаток має на меті реалізувати модулі безпеки за допомогою Spring Security разом з JWT, а також реєстрацію через Google за допомогою OAuth 2.0, а також чат між користувачами за допомогою веб сокетів. тому необхідно додати залежності цих інструментів, які можна знайти на офіційному сайті Maven Repository (рисунок 3.5).

Рисунок 3.5 - Сайт Maven Repository

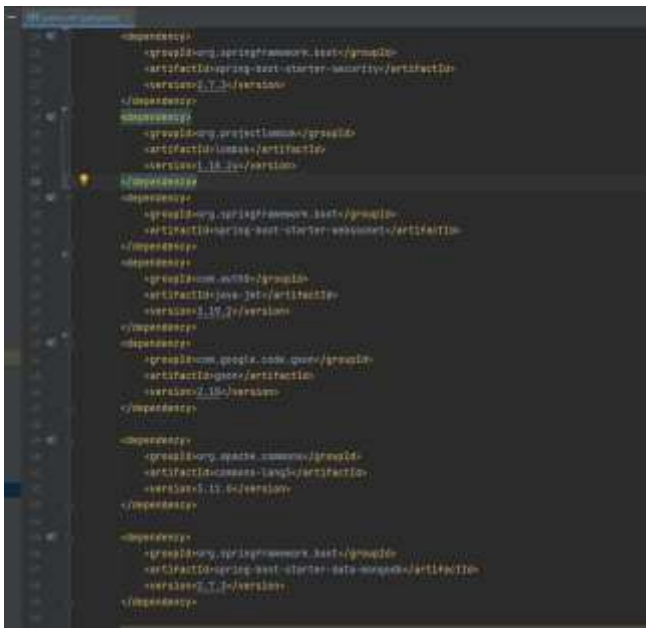

Рисунок 3.6 - файл pom.xml

Для підключення програми до бази даних необхідно налаштувати файл application.properties, а саме URL бази даних, ім’я користувача та пароль (рисунок 3.7).

```
spring.data.mongodb.database=petgram
spring.data.mongodb.uri=mongodb://localhost:20000
spring.data.mongodb.username=admin
spring.data.mongodb.password=admin
```

Рисунок 3.7 - З’єднання з базою даних

3.3 Розробка модулю безпеки

При розробці соціальної мережі безпека є одним з найважливіших аспектів. Користувачі очікують, що їхні персональні дані та конфіденційна інформація будуть захищені від несанкціонованого доступу та зловживань. Тому створення системи безпеки є критично важливим кроком у розробці будь-якого веб-додатку, включаючи соціальну мережу. Spring Security є потужним фреймворком, який надає розширені можливості для реалізації безпеки в додатках на основі Spring. Він надає набір інструментів і механізмів, які допомагають забезпечити аутентифікацію, авторизацію та контроль доступу до ресурсів. Використання Spring Security у соціальній мережі дозволяє забезпечити високий рівень безпеки, захистити конфіденційні дані користувачів та запобігти несанкціонованому доступу до функціональності додатку.

Налаштування Spring Security в додатку соціальної мережі дозволяє забезпечити наступні переваги:

- *Аутентифікація користувачів*: Spring Security надає можливість аутентифікувати користувачів з використанням різних механізмів, таких як

логін/пароль, соціальні мережі, одноразові паролі і т. д. Це дозволяє впевнитись, що тільки валідні користувачі мають доступ до системи.

- *Авторизація та контроль доступу*: Завдяки Spring Security можна визначити різні рівні доступу для користувачів, ролі та правила доступу до ресурсів. Це дозволяє забезпечити, що користувачі отримують доступ лише до тих функціональних можливостей, на які вони мають право.
- *Захист конфіденційної інформації*: Spring Security допомагає захистити конфіденційну інформацію користувачів, таку як паролі, особисті дані та інші чутливі дані, від несанкціонованого доступу. Використання шифрування та інших методів захисту даних допомагає забезпечити безпеку цих даних.
- *Інтеграція з іншими системами безпеки*: Spring Security може бути легко інтегрований з іншими системами безпеки, такими як OAuth 2.0, LDAP і т. д. Це дозволяє розширити можливості безпеки додатка та забезпечити інтеграцію з існуючими інфраструктурними рішеннями.

У директорії *security* (рисунок 4.8) створимо піддиректорії *filter* з класами *CustomAuthenticationFilter* та *CustomAuthorizationFilter*, *jwt* класами *JwtiContollerAdvice*, *JwtTokenProvider*, *JwtUserDetailsService*, *JwtUserFactory*, *UserPrincipal*, *oauth* з класами *CustomOAuth2User*, *CustomOAuth2UserService*, *OAuthTokenProvider* та клас *SecurityConfig*.

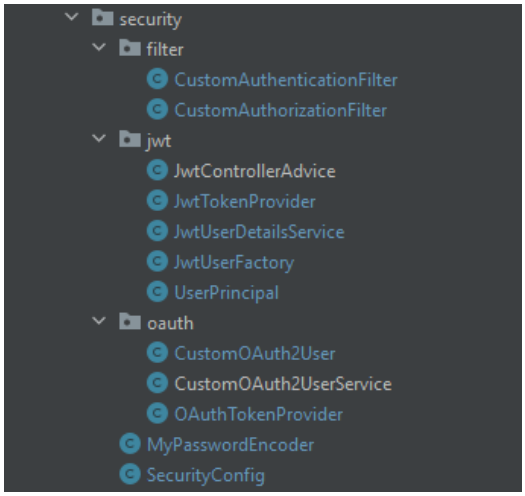

Рисунок 3.8 Директорія security

У класі SecurityConfig (рисунок 3.8) відбувається налаштування і конфігурація Spring Security для забезпечення в додатку соціальної мережі.

Розглянемо головні аспекти цього класу:

- Завдання залежностей: userRepository - репозиторій для доступу до даних користувачів, oAuthTokenProvider - провайдер токенів OAuth.
- Метод userDetailsService() визначає екземпляр JwtUserDetailsService, який використовує userRepository для отримання даних про користувача для аутентифікації.
- Метод authenticationProvider() створює екземпляр DaoAuthenticationProvider та налаштовує його для використання userDetailsService() та myPasswordEncoder.passwordEncoder().
- Метод filterChain() визначає основну конфігурацію Spring Security через HttpSecurity. Тут проводиться налаштування доступу до ресурсів, правил аутентифікації та авторизації. Детальніше:
- На рівні http.csrf().disable() відключається CSRF захист.
- За допомогою http.sessionManagement().sessionCreationPolicy (STATELESS) визначається політика управління сесіями - безсесійна аутентифікація.
- Використовується метод http.cors().and() для включення підтримки CORS (Cross-Origin Resource Sharing).
- Використовуються методи http.authorizeHttpRequests() та http.requestMatchers() для визначення правил доступу до ресурсів залежно від шляху.
- У методі http.anyRequest().authenticated() вказується, що всі інші запити повинні бути аутентифіковані.
- Використовується метод http.logout() для налаштування виходу з системи та вказання URL-адреси для перенаправлення після виходу.

- Використовується метод http.oauth2Login() для налаштування вхіду за допомогою OAuth2, включаючи успішний обробник, який генерує токен доступу для користувача після успішного входу.
- Додаються фільтри за допомогою методів http.addFilter(). Використовується CustomAuthenticationFilter, який відповідає за обробку аутентифікації та генерацію токенів доступу. Також використовується CustomAuthorizationFilter для обробки авторизації.
- Метод повертає об'єкт http.build(), який представляє налаштування Spring Security та використовується для налаштування безпеки у додатку соціальної мережі.

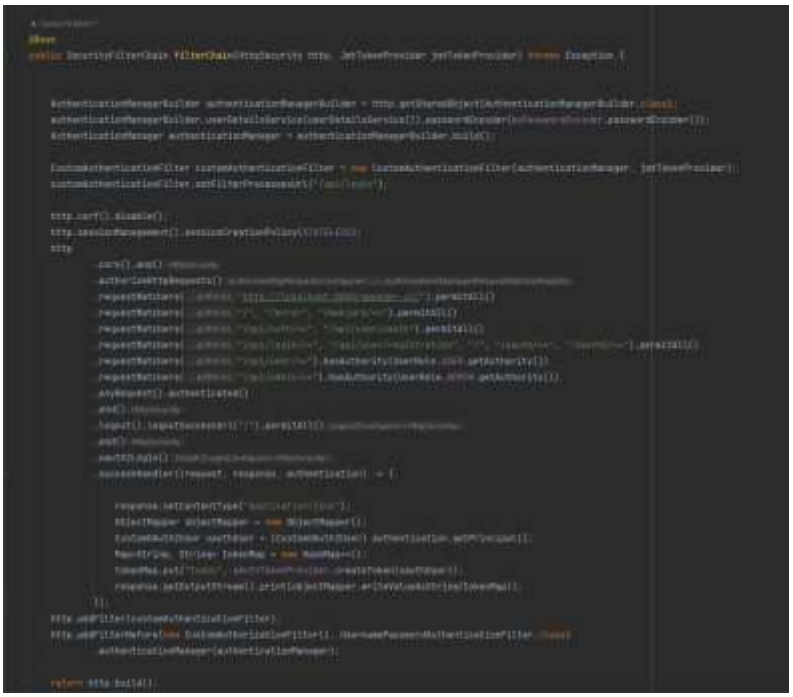

Рисунок 3.9 - SecurityConfig

Фільтр автентифікації (рисунок 3.9) відповідає за перевірку особи користувача та встановлення його автентифікаційного статусу. Він обробляє процес автентифікації користувачів на основі їхніх облікових даних, таких як ім'я користувача та пароль або токени автентифікації. У наданому коді CustomAuthenticationFilter розширює клас

UsernamePasswordAuthenticationFilter, який є вбудованим фільтром автентифікації у Spring Security.

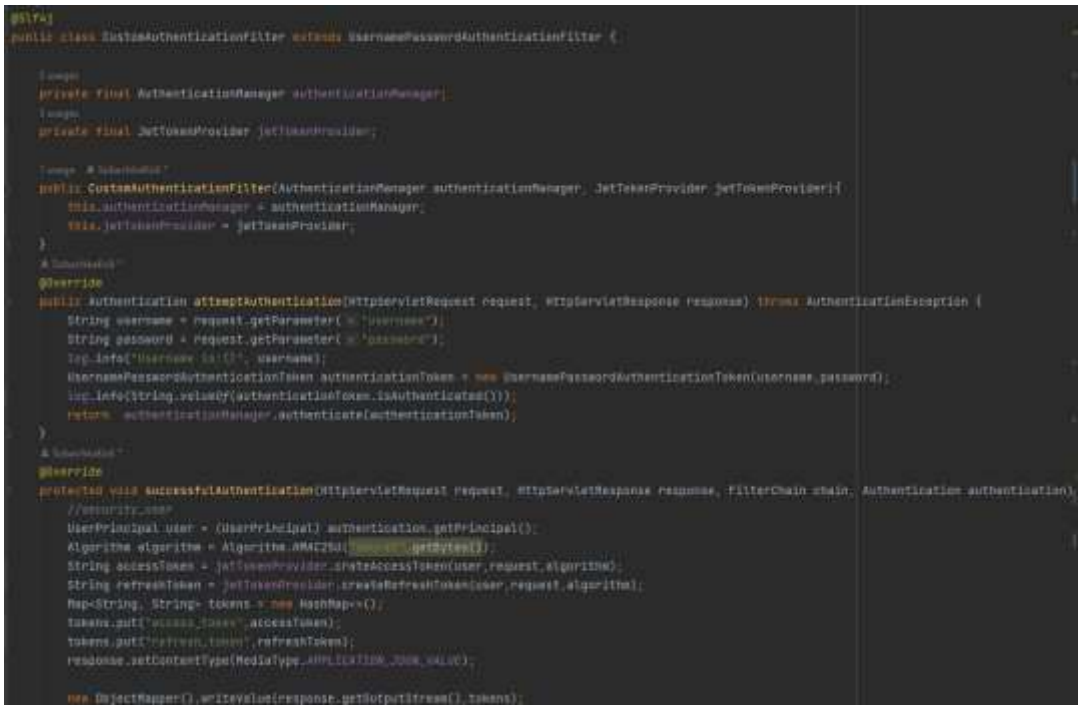

Рисунок 3.10 – Фільтр автентифікації

У методі attemptAuthentication() CustomAuthenticationFilter витягуються облікові дані користувача із запиту (ім'я користувача та пароль) і використовуються для створення об'єкта Authentication. Потім цей об'єкт передається в AuthenticationManager для автентифікації. Якщо автентифікація успішна, автентифікаційна інформація користувача встановлюється у контексті безпеки.

Фільтр авторизації (рисунок 4.10) визначає, чи дозволено користувачеві доступ до певних ресурсів або виконання певних дій у програмі. Він перевіряє статус автентифікації користувача та призначені йому ролі або дозволи для прийняття рішень щодо авторизації. У наведеному коді CustomAuthorizationFilter розширює клас OncePerRequestFilter, який є базовим класом для фільтрів, що викликаються лише один раз на запит.

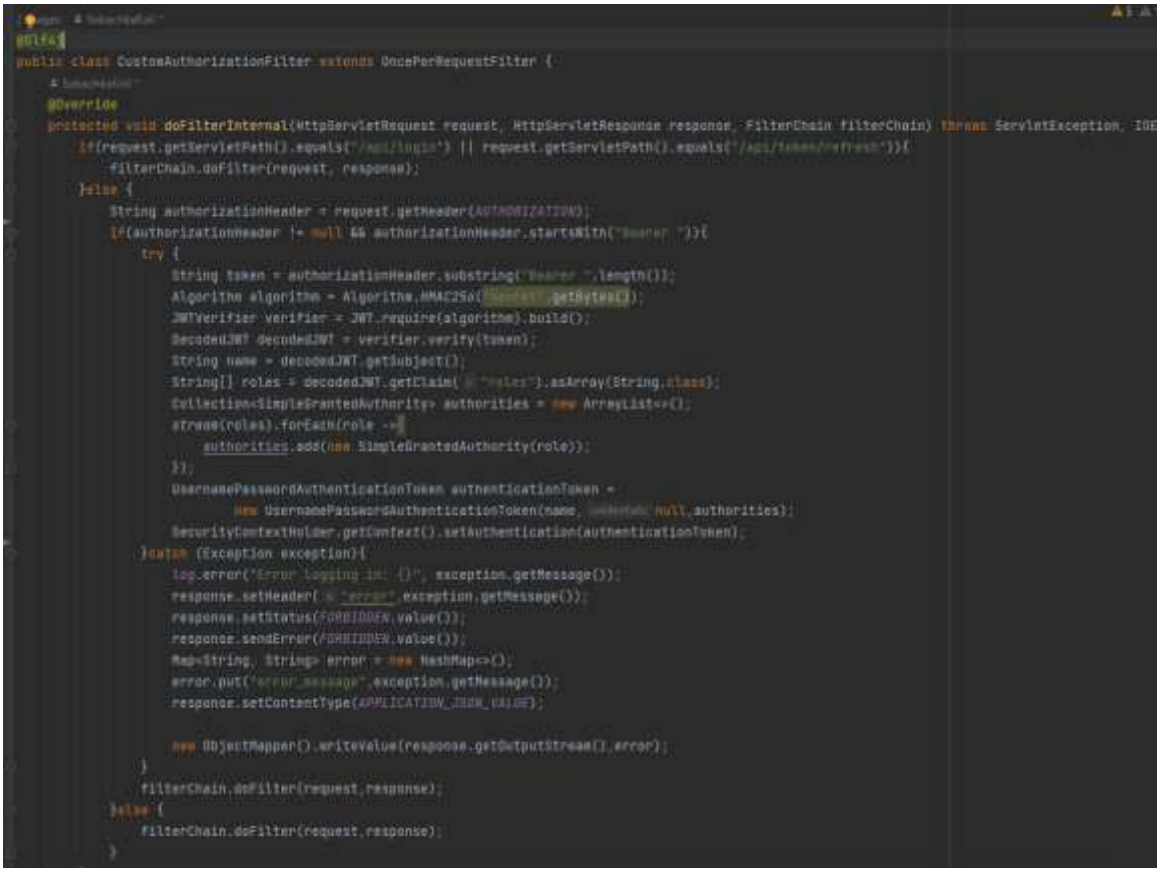

Рисунок 3.11 – Фільтр авторизації

У методі doFilterInternal() CustomAuthorizationFilter фільтр перехоплює вхідні запити і витягує токен JWT із заголовка авторизації. Потім він перевіряє цілісність і автентичність токена за допомогою вказаного алгоритму і секретного ключа. Якщо токен дійсний, з нього витягується аутентифікаційна інформація користувача, а його ролі перетворюються в об'єкти SimpleGrantedAuthority.

На основі ролей користувача фільтр створює об'єкт Authentication, що містить ідентифікаційні дані та повноваження користувача. Цей об'єкт потім встановлюється в SecurityContextHolder, який представляє поточний контекст безпеки для запиту. Встановлюючи автентифікацію, фільтр авторизації гарантує, що подальша обробка в додатку має доступ до автентифікаційної інформації користувача.

3.3.1 Удосконалення алгоритму аутентифікації

Стандартні методи і алгоритми аутентифікації мають ряд типових проблем:

- Вразливість паролів: Середньо статистичний користувач зазвичай використовує слабкий пароль, що робить його профіль легкою мішенню для атак типу brute-force.
- Єдина точка входу: Захист лише паролем, не захищає систему від компрометації
- Недостатня анонімність: Класичні підходи ідентифікатори (email, телефон) можуть використовуватись для відстеження користувача.

Для ефективної роботи та задля досягнення мети, алгоритм має відповідати певним вимогам:

- Використовувати багатофакторну аутентифікацію для додаткового захисту.
- Ідентифікація без прив'язки до особистих даних через унікальні токени чи псевдоніми.
- Використання криптографічних функцій для створення ідентифікаторів без використання реальних імен чи емейлів.
- Шифрування перед передачею та зберіганням.

Використання токенів JWT. Цей метод дозволяє безпечно передавати інформацію про користувача, зберігаючи її анонімність. Генерація JWT показана на рисунку 3.11. Переваги JWT: підтримує шифрування, легко інтегрується в RESTful API, зберігає мінімальну інформацію про користувача.

```
public String generateJwt(String username) {
    return Jwts.builder()
        .setSubject(username)
        .setIssuedAt(new Date())
        .setExpiration(new Date(System.currentTimeMillis() + 3600000))
        .signWith(SignatureAlgorithm.HS512, "secretKey")
        .compact();
}
```

Рисунок 3.12 – Створення JWT-токену

Наступний крок – це впровадження багатофакторної аутентифікації. TOTP (Time-Based One-Time Password) забезпечує додатковий рівень захисту. Коди генеруються на основі секретного ключа і часу.(Рисунок 3.12).

```
public String generateTOTP(String secret) {
    Base32 base32 = new Base32();
    byte[] decodedKey = base32.decode(secret);
    long timeIndex = System.currentTimeMillis() / 30000;
    return new TotpGenerator(decodedKey).generate(timeIndex);
}
```

Рисунок 3.13 – Створення TOTP

Шифрування паролів не менш важливий фактор захисту. Вони повинні зберігатися у хешованому вигляді, щоб уникнути витоків. На рисунку 3.13 зображено шифрування паролів.

```
public String encryptPassword(String rawPassword) {
    PasswordEncoder encoder = new BCryptPasswordEncoder();
    return encoder.encode(rawPassword);
}

public boolean validatePassword(String rawPassword, String encodedPassword) {
    PasswordEncoder encoder = new BCryptPasswordEncoder();
    return encoder.matches(rawPassword, encodedPassword);
}
```

Рисунок 3.14 – Створення TOTP

3.4 Сервісний шар

Сервісний шар (Service Layer) є одним з ключових компонентів архітектури додатку, який відповідає за обробку бізнес-логіки, виконання операцій над доменною моделлю, а також взаємодію зі смарт-контрактами та блокчейном. Цей шар виконує роль проміжної ланки між контролерами (controllers), репозиторіями (repositories) та логікою блокчейну, забезпечуючи чітке відокремлення бізнес-логіки від інших компонентів. Інтеграція з блокчейном додає нові можливості для захищеного та прозорого зберігання і обробки даних.

Основна мета сервісного шару — забезпечити комплексну обробку операцій, які можуть включати інтеграцію з блокчейном, обмін даними через смарт-контракти та роботу з базами даних. Це також сприяє легшій підтримці, тестуванню та повторному використанню функціональності..

- Бізнес-логіка: Сервісний шар містить бізнес-логіку, яка визначає, які операції повинні бути виконані над даними додатку, зокрема, виклик функцій смарт-контрактів. Наприклад, створення нового користувача в системі може включати виклик методу смарт-контракту для реєстрації адреси користувача у блокчейні.
- Координація репозиторіїв: Сервісний шар координує роботу з базами даних (через репозиторії) та блокчейном. Наприклад, інформація про користувача може спочатку зберігатися в базі даних для швидкого доступу, а потім реєструватися у блокчейні через відповідний смарт-контракт.
- Валідація та перевірка даних: Сервісний шар виконує валідацію даних, як перед відправкою до блокчейну, так і перед записом до бази даних.
- Обробка помилок: Сервісний шар може бути відповідальним за обробку помилок, які виникають під час виконання операцій.
- Взаємодія зі смарт-контрактами:

Сервісний шар реалізує логіку виклику методів смарт-контрактів для виконання бізнес-операцій, таких як реєстрація користувачів, перевірка прав доступу або запис транзакцій.

В проекті директорія service може містити реалізації сервісів, які будуть використовуватись контролерами для виконання операцій над даними та координації дій між репозиторіями. Кожен сервіс може мати власний інтерфейс, який описує доступні операції, що пов'язані з конкретною бізнес-логікою. Цей шар забезпечує чітку відповідальність та взаємодію між різними компонентами додатку. Зокрема, один із таких сервісів може бути BlockchainService, який відповідає за реєстрацію користувачів у блокчейні. Цей сервіс взаємодіє з Ethereum блокчейном через смарт-контракти, зберігаючи дані

користувача (наприклад, ім’я та публічний ключ) безпосередньо на ланцюгу блоків.

На рисунку 3.11 зображено метод реєстрації registerUserInBlockchain класу BlockchainService, який реєструє користувача на блокчейні, зберігаючи його ім’я та публічний ключ в смарт-контракті на Ethereum. Цей метод забезпечує додатковий рівень безпеки та анонімності для користувачів додатку, інтегруючи блокчейн у бізнес-логіку системи.

```java
@Service
public class UserService {

    @Autowired
    private BlockchainService blockchainService;

    public void registerUser(UserDto userDto) throws Exception {

        KeyPair keyPair = KeyGenerator.generateKeyPair();
        String publicKey = Base64.getEncoder().encodeToString(keyPair.getPublic().getEncoded());

        blockchainService.registerUserInBlockchain(userDto.getUsername(), publicKey);

        KeyStorage.storeKeys(userDto.getUsername(), keyPair);
```

Рисунок 3.15 – Реєстрація та взаємодія зі смарт-контрактом

```java
import org.web3j.protocol.Web3j;
import org.web3j.protocol.http.HttpService;
import org.web3j.crypto.Credentials;
import org.web3j.tx.gas.GasProvider;
import org.web3j.tx.ChainId;
import org.web3j.tx.RawTransactionManager;
import org.web3j.tx.TransactionManager;
import org.web3j.utils.Convert;
import org.web3j.protocol.core.methods.response.EthBlock;
import org.web3j.protocol.core.methods.response.EthGasPrice;

import java.math.BigDecimal;
import java.math.BigInteger;

public class BlockchainService {

    private Web3j web3j;
    private String contractAddress = "your_contract_address_here";
    private String privateKey = "your_private_key_here"; // Use proper key management.

    public BlockchainService() {
        this.web3j = Web3j.build(new HttpService("http://localhost:8545")); // Connect to your Ethereum node
    }

    public boolean registerUserInBlockchain(String username, String publicKey) {
        try {
            // Load credentials (from private key)
            Credentials credentials = Credentials.create(privateKey);

            // Prepare a TransactionManager
            TransactionManager transactionManager = new RawTransactionManager(web3j, credentials, ChainId.MAINNET);

            // You should have a smart contract for user registration
            UserRegistrationContract contract = UserRegistrationContract.load(
                contractAddress, web3j, transactionManager, new GasProvider() {
                    @Override
                    public BigInteger getGasPrice() {
                        try {
                            EthGasPrice gasPrice = web3j.ethGasPrice().send();
                            return gasPrice.getGasPrice();
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                        return BigInteger.valueOf(20000000000L); // Default gas price
                    }

                    @Override
                    public BigInteger getGasLimit() {
                        return BigInteger.valueOf(2000000); // Default gas limit
                    }
                }
            );

            // Call the method in the contract (assumes method 'registerUser' exists in your contract)
            contract.registerUser(username, publicKey).send();

            return true; // Return true if registration is successful
        } catch (Exception e) {
            e.printStackTrace();
            return false; // Return false if registration fails
        }
```

Рисунок 3.16 – Блокчейн сервіс

```
SobachkaKoli *
@RestController
@RequestMapping("/api/auth")
@RequiredArgsConstructor
public class AuthController {

    1 usage
    private final AuthService authService;

    SobachkaKoli
    @PostMapping("/signup")
    public User registerUser(@RequestBody UserDto userDto) throws Status434UsernameNotUniqueException {
        return authService.signUp(userDto);
    }
```

Рисунок 3.17 – Контролер

```
@Repository
public interface UserRepository extends MongoRepository<User,String> {

    9 usages  SobachkaKoli
    Optional<User> findById(String userId);

    SobachkaKoli
    Optional<User> findByUsername(String userName);

    3 usages  SobachkaKoli
    Optional<User> findByEmail(String email);

    3 usages  SobachkaKoli
    Boolean existsByEmail(String email);

    3 usages  SobachkaKoli
    Boolean existsByUsername(String username);

}
```

Рисунок 3.18 – Репозиторій користувачів

3.5 Тестування додатку

Тестування додатку з інтеграцією Blockchain та Smart-контрактів буде проведене за допомогою Postman. Postman залишається ефективним інструментом для тестування API, навіть за наявності взаємодії зі смарт-контрактами. Його використання дозволяє відправляти запити до API додатку,

який, у свою чергу, взаємодіє з блокчейном. Таким чином можна перевірити коректність роботи всіх компонентів системи, включаючи виклики смарт-контрактів. [28]

Перед тестування необхідно виконати наступні кроки:

- Введення базового URL: В початкових налаштуваннях Postman можна вказати базовий URL додатку, з яким будуть виконуватись всі наступні запити. Це дозволить спростити введення URL для кожного запиту окремо.
- Створення нового запиту: Для тестування кожного ендпоінта слід створити новий запит у Postman. Це можна зробити, натиснувши кнопку "New" і вибравши тип запиту (GET, POST, PUT, DELETE тощо).
- Налаштування параметрів запиту: У вкладці "Params" можна додати параметри запиту, які передаються у URL або як запити POST/PUT. Тут можна вказати параметри, такі як ім'я користувача, пароль, токен, publickKey для взаємодії з блокчейном, transactionHash для перевірки стану транзакції.
- Налаштування заголовків запиту: У вкладці "Headers" можна встановити заголовки запиту, такі як Content-Type або Authorization. Залежно від вимог додатку, можуть бути необхідні деякі заголовки для правильної обробки запиту.
- Відправлення запиту: Після налаштування параметрів і заголовків натискається кнопка "Send" для відправлення запиту на сервер. Відповідь сервера включає дані, що були записані в блокчейн через смарт-контракт.
- Аналіз відповіді: У вкладці "Body" можна перевірити отриману відповідь від сервера. Тут можна перевірити, чи повернулася правильна структура даних, чи містить вона очікувані значення.
- Тестування різних сценаріїв: Postman дозволяє виконувати різні сценарії тестування, наприклад, намагатись залогінитись з неправильними даними, перевіряти валідацію вхідних параметрів, тестувати доступ до захищених ресурсів тощо.
- Збереження та організація запитів: Postman дозволяє зберігати запити для подальшого використання. Можна створювати колекції

Протестуємо метод реєстрації за допомогою username та password, як показано на рисунку 3.14. через HTTP метод POST у форматі JSON передаємо дані для реєстрації та відправляємо за адресом http://localhost:8081/api/auth/signup. У відповідь отримує статус 200, тобто все пройшло без помилок та отримали об'єкт User, який зберігся в базу (рисунок 3.19).

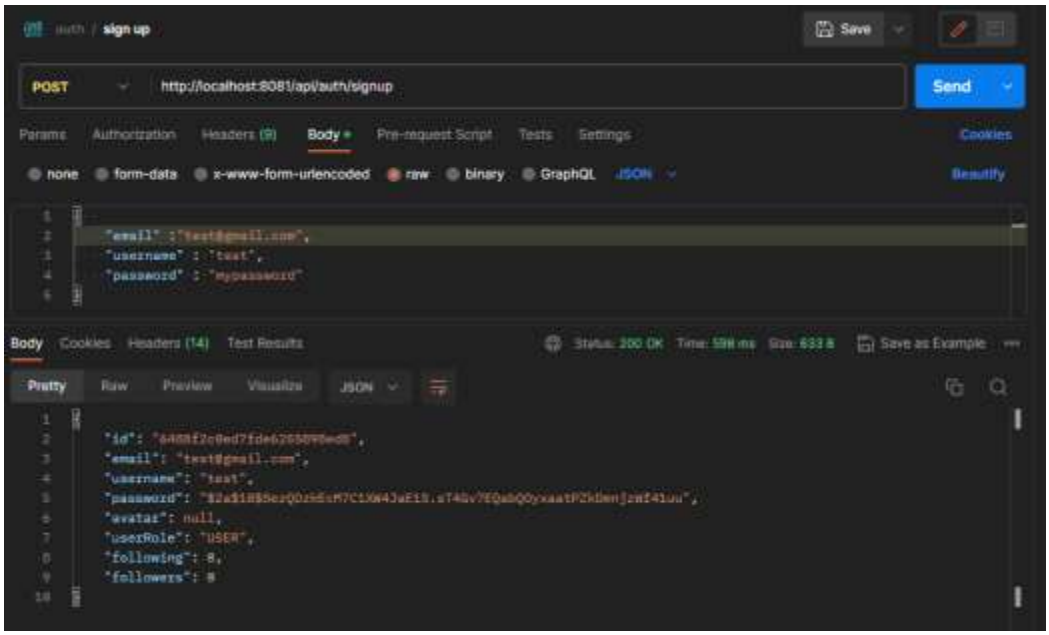

Рисунок 3.19 – Реєстрація користувача через Postman

Рисунок 3.20 – Збережений користувач в базі даних

Тепер встановимо фото профілю для користувача використовуючи Postman Але для початку потрібно авторизуватися в додатку та отримати JWT-токен для підтвердження особи як показано на рисунку 3.17. Для цього створимо контролер який буде приймати файл та передавати його далі у сервіс, який в свою чергу буде зберігати файл на сервері та посилання на нього в базі

даних (рисунок 3.21- 3.23) та передавати посилання на фото в профіль користувача.

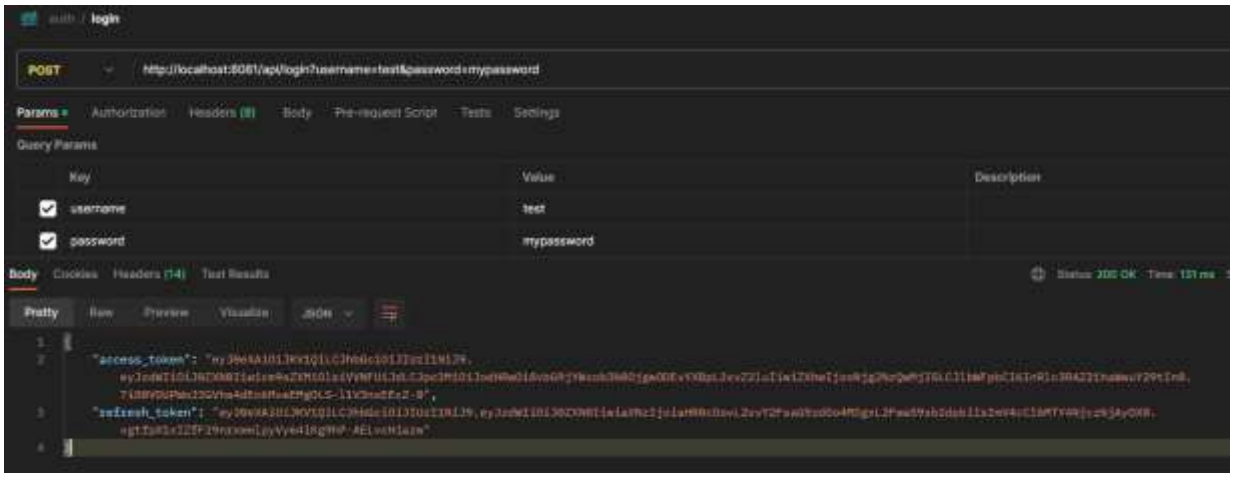

Рисунок 3.21 – Авторизація користувача

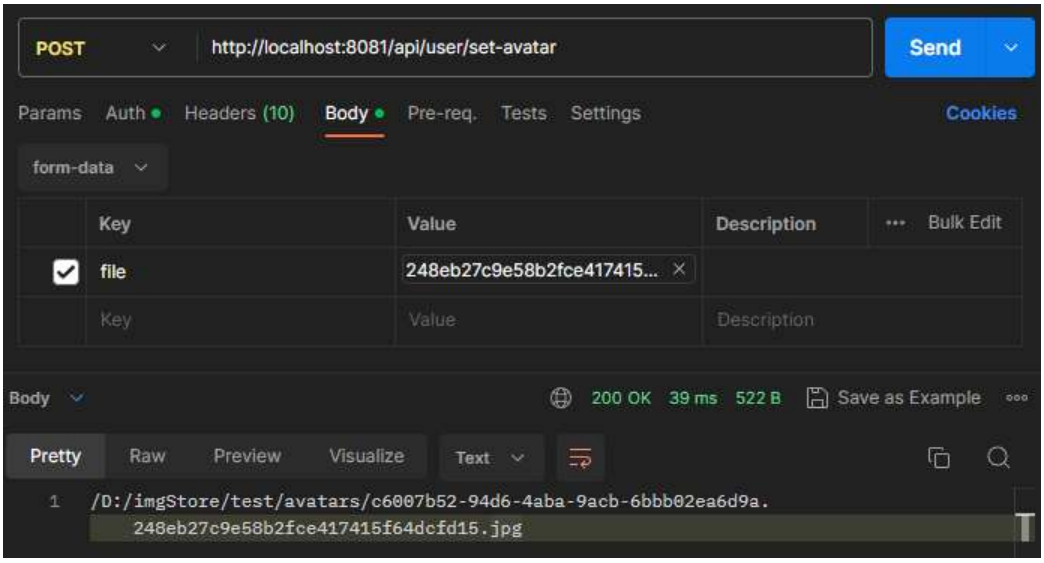

Рисунок 3.22 – Встановлення фото профілю

Рисунок 3.23 – Користувач з фото в базі даних

Користувач має можливість створювати дописи. Енд-поінт для створення допису отримує PostDTO, який в собі має текст допису та список зображень (рисунок 3.24 – 3.25).

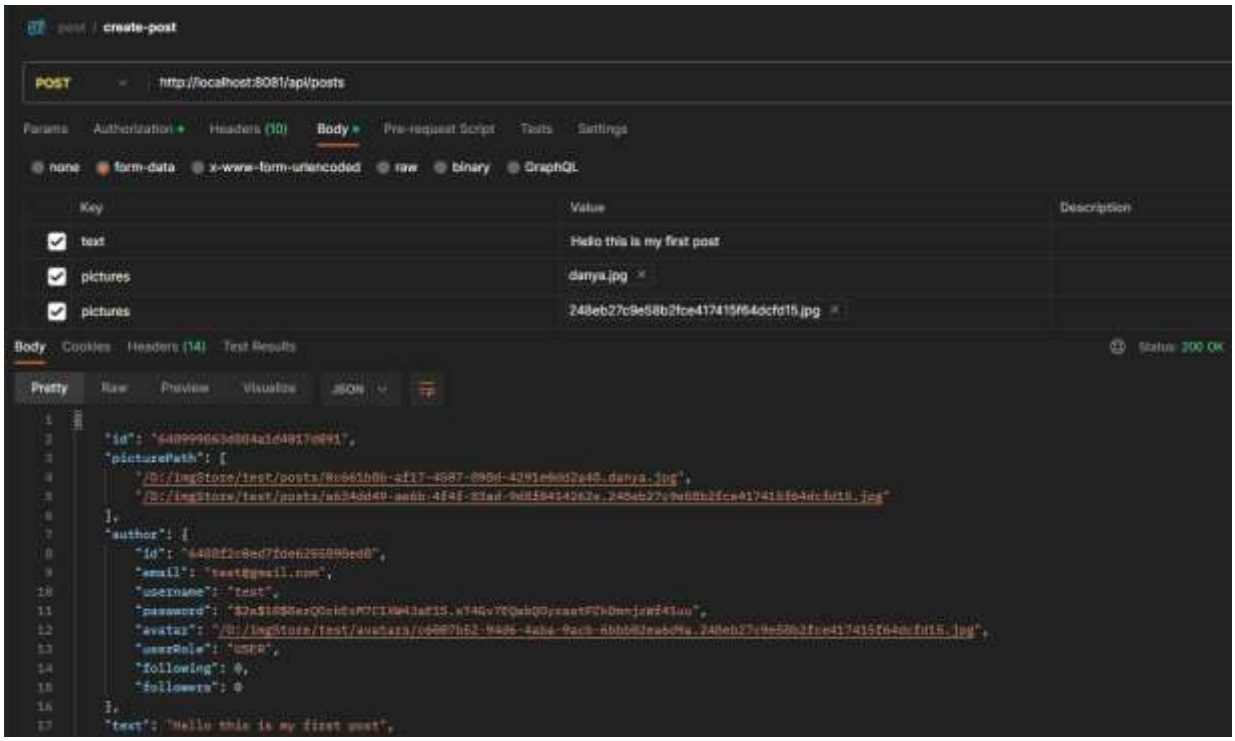

Рисунок 3.24 – Створення допису

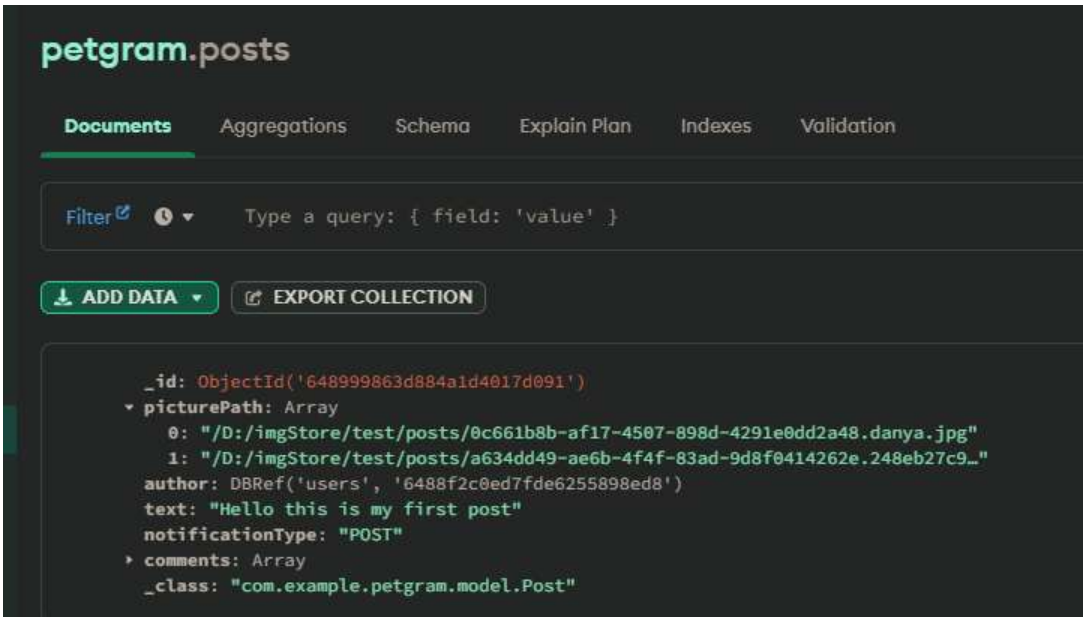

Рисунок 3.25 – Допис в базі даних

Поставимо до минулого допису лайк, для цього також існує енд-поінт http://localhost:8081/api/posts/{postId}/likes (рис. 3.20) Цей енд-поінт приймає

параметр postId, створює лайк цьому допису, а автору дописа надсилає сповіщення, яке також зберігається в базі даних (рис. 3.26 – 3.27).

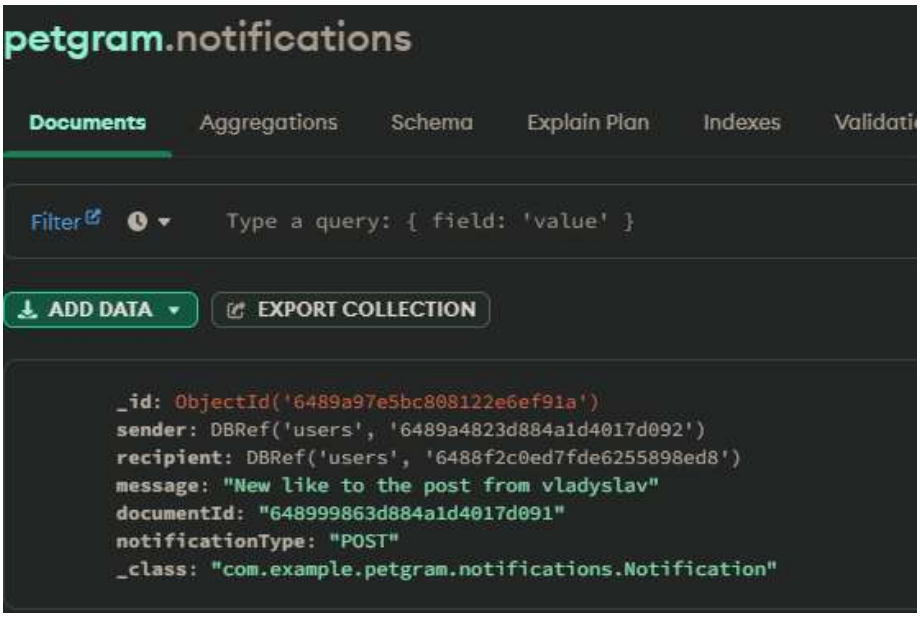

Рисунок 3.26 – Сповіщення про лайк в базі даних

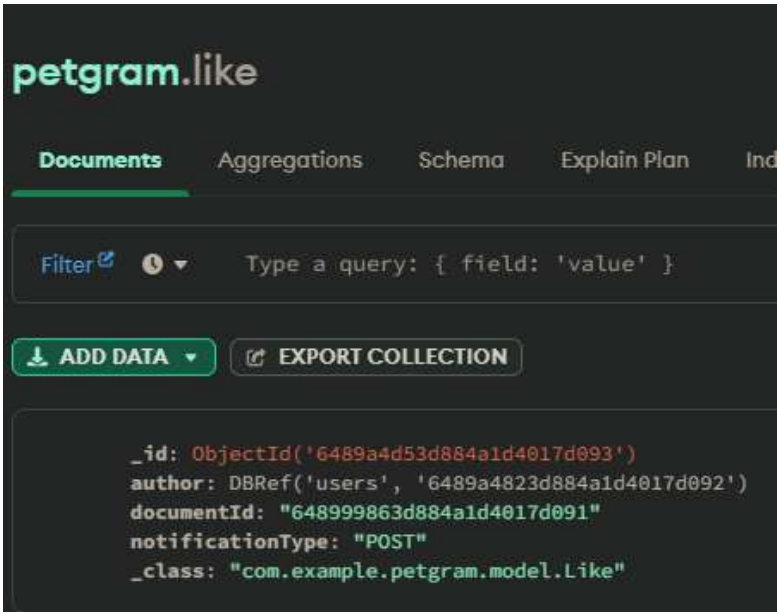

Рисунок 3.27 – Лайк в базі даних

3.6 Порівняння з аналогами

Для оцінки ефективності розробленої системи було проведено аналіз та порівняння її характеристик із популярними рішеннями, такими як Mastodon, Signal і Telegram. Основна увага приділялася аспектам анонімності, захисту даних, децентралізованості та функціональності. Нижче, у таблиці 4.1, наведено порівняльну таблицю.

Таблиця 3.1 – Порівняльна таблиця

Критерій	**Розроблена система**	**Mastodon**	**Signal**	**Telegram**
Анонімність	Висока, завдяки блокчейну та децентралізованому управлінню доступом.	Середня, залежить від адміністратора сервера.	Висока, наскрізне шифрування.	Низька, метадані зберігаються на серверах.
Децентралізованість	Повна, інтеграція блокчейну виключає потребу в адміністраторах.	Середня, базується на федеративній моделі.	Відсутня, централізована архітектура.	Відсутня, централізована архітектура.
Конфіденційність	Висока, дані користувачів шифруються та зберігаються децентралізовано.	Середня, конфіденційність залежить від сервера.	Висока, завдяки наскрізному шифруванню.	Середня, є обмежене шифрування.
Можливості для соціальних мереж	Широкі, включаючи групи, особисті профілі, пошук і контент.	Високі, орієнтовані на спільноти.	Обмежені, лише обмін повідомленнями.	Високі, підтримка груп і каналів.

Продовження таблиці 3.1

Безпека даних	Висока, використання блокчейну мінімізує ризик компрометації.	Середня, безпека залежить від конкретного сервера.	Висока, завдяки шифруванню.	Середня, залежить від централізованих серверів.
Недоліки	Обмежена швидкодія через децентралізовану природу блокчейну.	Залежність від адміністраторів серверів.	Немає підтримки соціальних мереж.	Залежність від централізованих серверів.

Проведене порівняння показало, що розроблена система значно перевершує аналоги за ключовими показниками, такими як анонімність, децентралізованість і безпека даних. Завдяки використанню технології блокчейну забезпечується повна конфіденційність та захист особистої інформації користувачів, що мінімізує ризики несанкціонованого доступу або компрометації даних, на відміну від Telegram, де частина даних може зберігатися централізовано. Децентралізована природа системи усуває залежність від адміністраторів серверів, яка є суттєвим недоліком Mastodon. Крім того, у порівнянні з Signal, розроблена система не лише підтримує високий рівень конфіденційності, але й пропонує розширені функції соціальних мереж, що робить її універсальним рішенням для комунікації та обміну контентом.

Однак, через децентралізовану архітектуру швидкодія системи може поступатися централізованим платформам, таким як Telegram, що є цінною характеристикою для користувачів, які орієнтуються на миттєву комунікацію. Попри це, інтеграція блокчейну відкриває нові можливості для забезпечення безпеки та анонімності, які є критичними у сучасному онлайн-середовищі. Таким чином, розроблена система забезпечує унікальне поєднання функціональності, безпеки та конфіденційності, що ставить її на порядок вище за існуючі аналоги.

3.7 Висновки до третього розділу

У розділі було описано важливі аспекти розробки серверної частини REST API для соціальної мережі з використанням Java та Spring Boot. Були описані кроки налаштування проекту, включаючи конфігурацію Maven та з'єднання з базою даних MongoDB.

Особлива увага була приділена безпеці додатку, і Spring Security виявився потужним інструментом для забезпечення безпеки. Були представлені класи SecurityConfig, CustomAuthenticationFilter та CustomAuthorizationFilter, які виконують ключові функції, такі як аутентифікація користувачів, авторизація доступу та фільтрація запитів.

Також був описаний сервісний шар, який виконує важливу роль у взаємодії з базою даних та обробці бізнес-логіки. Цей шар забезпечує обробку запитів, перевірку прав доступу та валідацію даних.

Розробка безпечної та надійної серверної частини REST API для соціальної мережі є складним завданням, але використання Java та Spring Boot спрощує цей процес. Правильне налаштування Spring Security, використання кастомних фільтрів та розробка сервісного шару дозволяють забезпечити безпеку, автентифікацію та авторизацію користувачів.

У результаті тестування за допомогою Postman було успішно перевірено функціональність створення посту в додатку. За допомогою POST-запиту, відправленого до відповідного ендпоінта, були передані дані про пост у форматі JSON. Після обробки запиту сервером було отримано успішну відповідь, що свідчить про успішне створення посту.

Тестування за допомогою Postman дозволяє перевірити, чи працює функціонал додатку на рівні API, без необхідності використання фронтенду або інших клієнтських додатків. Це забезпечує швидке та ефективне виявлення помилок та перевірку правильності реалізації функціональності.

У нашому випадку, після відправки POST-запиту з правильними даними, було отримано позитивну відповідь з інформацією про успішне створення

посту. Це підтверджує, що функціональність створення посту в додатку працює належним чином.

Загальною висновок є те, що тестування за допомогою Postman дозволяє ефективно перевірити роботу API, включаючи функціонал створення посту. Успішне тестування свідчить про правильну конфігурацію та реалізацію функціональності створення посту в додатку.

4. ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Технологічний аудит розробленої системи автоматизації управління анонімною соціальною мережею

Сьогодні, у сучасному світі, анонімність у соціальній мережі становить важливий (ключовий) момент у збереженні особистої приватності користувачів та захисту їх індивідуальних даних. Забезпечення можливості користувачам соціальних мереж залишатися невпізнаними сприяє вільній виразності їх думок та дозволяє уникати потенційних ризиків, пов'язаних з небажаним втручанням інших осіб у приватне життя людини.

Враховуючи постійно зростаючі загрози кібератак та можливість несанкціонованого доступу до особистої інформації людини, модернізація алгоритму аутентифікації стає важливим заходом для попередження небажаного доступу. У зв’язку з цим, модернізація систем безпеки соціальної мережі стала необхідністю для забезпечення і гарантування законних прав і свобод людини.

Тому перед виконаною нами магістерською кваліфікаційною роботою було поставлено мету – удосконалити алгоритми аутентифікації та методи збереження даних користувачів соціальних мереж.

Для досягнення цієї мети нами було: здійснено аналіз предметної області; проведено огляд існуючих рівень; удосконалено алгоритм аутентифікації; спроєктовано метод збереження даних; проаналізовано рівень безпеки та анонімності; проведено тестування та оптимізація отриманих результатів.

В результаті нами було розроблено систему автоматизації управління анонімною соціальною мережею, яка забезпечує конфіденційність користувачів, безпечне та анонімне спілкування у мережі Інтернет, підвищує рівень безпеки вхідних даних та унеможливлює несанкціонований доступ.

Для встановлення комерційного потенціалу розробленої нами системи автоматизації управління анонімною соціальною мережею було запрошено 3-х

висококваліфікованих експертів: к.т.н., доцента кафедри АІІТ Кулика Ярослава Анатолійовича, к.т.н., доцента кафедри АІІТ Гармаша Володимира Володимировича, а також експерта-практика – фахівця у цій галузі знань – Артюхова Олексія Олександровича (Software Engineer).

Запрошені експерти зробили експертне оцінювання комерційного потенціалу розробленої ними системи за критеріями, наведеними в таблиці 4.1,

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж

			аналогів		в аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкуренті в немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

				ти років	
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Експерти оцінили комерційний потенціал розробленої нами системи автоматизації управління анонімною соціальною мережею і зробили висновки, зведені в таблицю 4.2:

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробленої нами системи автоматизації управління анонімною соціальною мережею (За 5-ти бальною шкалою оцінювання 0 -1 – 2 – 3 - 4)

Критерії	Ініціали, прізвище експертів		
	Я.А. Кулик	В.В. Гармаш	О.О. Артюхов
	Бали, що їх виставили експерти:		
1	4	4	3
2	4	3	3
3	3	4	3
4	4	4	3
5	3	4	4
6	4	4	3
7	3	3	4
8	4	4	3
9	3	4	3
10	4	4	4
11	4	4	3
12	3	4	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 46$	$СБ_3 = 40$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{43+46+40}{3} = \frac{129}{3} = 43,00$		

Для встановлення комерційного потенціалу розробленої нами системи автоматизації управління анонімною соціальною мережею використаємо рекомендації, які наведено в таблиці 4.3 [29].

Таблиця 4.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 43,00 бал (із максимально можливих 48-ми балів), то це свідчить, що розроблена нами система автоматизації управління анонімною соціальною мережею має рівень комерційного потенціалу, який можна вважати «високим».

Це пояснюється тим, що наша розробка суттєво покращить безпеку та анонімність користувачів в соціальних мережах і легко інтегруються у вже існуючі соціальні мережі та інші Інтернет-ресурси. Це створить нові можливості для користувачів соціальних мереж та дозволить платформам оперативно впроваджувати нові засоби захисту. Окрім того, удосконалення алгоритму аутентифікації та методу збереження даних зробить свій внесок у сферу кібербезпеки та приватності, адже сьогодні приватність та безпека даних користувачів (як фізичних, так і юридичних) є важливою як ніколи.

4.2 Розрахунок витрат на розроблення системи автоматизації управління анонімною соціальною мережею

При розробленні системи автоматизації управління анонімною соціальною мережею було зроблено низку основних витрат:

а). Основна заробітна плата $З_о$ розробників (дослідників тощо), які здійснювали розроблення системи автоматизації управління анонімною соціальною мережею. Величина цієї зарплати визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \quad \text{грн,} \tag{4.1}$$

де М – місячний посадовий оклад розробника (дослідника), грн;

Для 2024 року приймемо, що:

М = (8000…28000) грн/місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 25 днів;

t – число днів роботи розробників (дослідників тощо).

Зроблені розрахунки основної заробітної плати розробників (дослідників тощо) зведемо до таблиці 4.4:

Таблиця 4.4 – Основна заробітна плата розробників (дослідників тощо)

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів (годин) роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	23000	≈ 920	20 годин	(920 / 6) × 20 = ≈ 3067 (при 6-годинному робочому дні)
2. Здобувач-магістрант (виконавець)	2000 грн (беремо 8000 грн)	≈ 320	84 дні	26880
3. Експерт із захисту інформації	40000	≈ 1600	2 дні	3200 грн (при 8-годинному робочому дні)
4. Консультант з економічної частини	18600	744	1,5 години	(744 /6) × 1,5 ≈ 186 грн (при 6-годинному робочому дні)
Загалом				$З_о$ = 33333 грн

Б). Додаткова заробітна плата $З_д$ розробників (дослідників тощо) розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о . \tag{4.2}$$

Приймемо, що α = 0,1. Тоді для нашого випадку отримаємо:

$$З_д = 0{,}1 \times 33333 = 3333{,}30 \approx 3334 \text{ грн.}$$

В). Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100},$$

(4.3)

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. В 2024 році ставка β = 22%. Тоді:

$$НЗН_{зп} = (33333 + 3334) \times 0,22 = 8066,74 \approx 8067 \text{ грн.}$$

Г). Амортизація основних засобів А, які використовувались під час виконання магістерської кваліфікаційної роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \quad \text{грн,}$$

(4.4)

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань.

Приймемо, що $Н_а$ = (2,5...25)%;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 5.5.

Таблиця 4.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, місяців	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання тощо	80000	22,5	3,1 (при 90% використанні)	4185
2. Приміщення університету, факультету, кафедри	50000	4,5	3,1 (при 50% використанні)	290,62 ≈ 291
Всього				А = 4476 грн

Д). Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \quad \text{грн,}$$

(4.5)

де $Н_i$ – витрати матеріалу *і*-го найменування, кг; $Ц_i$ – вартість матеріалу *і*-го най-менування; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *і*-го найменування; $Ц_в$ – ціна відходів матеріалу *і*-го найменування; n – кількість видів матеріалів.

Е). Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \quad \text{грн,}$$

(4.6)

де $Н_i$ – кількість комплектуючих *і*-го виду, шт.; $Ц_i$ – ціна комплектуючих *і*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Під час виконання магістерської кваліфікаційної роботи загальні витрати на матеріали та комплектуючі склали укрупнено приблизно 2200 грн.

Ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д},$$

(4.7)

де В – вартість 1 кВт-год. електроенергії, в 2024 р. В ≈ 4,8 грн/кВт;

П – установлена потужність обладнання, кВт; П = 1,45 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 300 годин;

$К_п$ – коефіцієнт використання потужності; $К_п$ < 1 = 0,86.

$К_д$ – коефіцієнт корисної дії, $К_д$ = 0,79.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,8 \cdot 1,45 \cdot 300 \cdot 0,86}{0,79} = 2273,01 \approx 2273 \text{ грн.}$$

И). Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (4.8)$$

Для нашого випадку отримаємо:

$$В_{інш} = 1,0 \times 33333 = 33333 \text{ грн.}$$

К). Сума всіх попередніх статей витрат складає витрати на виконання нашої магістерської роботи безпосередньо розробником-магістрантом – В.

$$В = 33333 + 3334 + 8067 + 4476 + 2200 + 2273 + 33333 = 87016 \text{ грн.}$$

Л). Загальні витрати на розробку та можливу комерціалізацію розробленої нами системи автоматизації управління анонімною соціальною мережею розраховуються за формулою:

$$В_{заг} = \frac{В}{\beta}, \quad (4.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи.

Оскільки наша розробка на цей момент часу має високий ступінь готовності, але потребує ще деякого незначного доопрацювання, то можна умовно прийняти, що, $\beta \approx 0,8$ [29]блят

Тоді: $В_{заг} = \frac{87016}{0,8} = 108770,00$ грн або приблизно 109 тисяч грн.

Тобто прогнозовані загальні витрати на розробку та можливу комерціалізацію розробленої нами системи автоматизації управління анонімною соціальною мережею становлять приблизно 109 тисяч грн.

4.3 Розрахунок економічного ефекту від можливої комерціалізації розробленої нами системи автоматизації управління анонімною соціальною мережею

Проведене дослідження ринку показало, що розроблена нами система автоматизації управління анонімною соціальною мережею орієнтована на дуже

широкий спектр користувачів, а саме: звичайних користувачів соціальних мереж; самих соціальних мереж та онлайн-платформ; організацій та установ, які працюють у сферах з високими вимогами до конфіденційності (активісти, журналісти, правозахисні організації тощо); корпорації з посиленими вимогами до приватності та захисту конфіденційної інформації та інш.

У зв'язку з цим, подальші розрахунки будемо проводити для цільової аудиторії, яку складають звичайні користувачі соціальних мереж, включаючи активістів, журналістів тощо. Аналіз місткості ринку також показує, що на сьогодні в Україні кількість реальних користувачів подібних систем управління анонімною соціальною мережею є може становити до 150 користувачів. Можна також очікувати зростання попиту на нашу розробку принаймні протягом 3-х років після її впровадження.

Тобто, якщо наша розробка буде впроваджена з 1 січня 2025 року, то її результати будуть виявлятися протягом 2025-го, 2026-го та 2027-го років.

Прогноз зростання попиту на нашу розробку може складати по роках:

а) 2025 р. – приблизно + 5 шт. (відносно базового року);

б) 2026 р. – +10 шт. (відносно базового року);

в) 2027 р. – +20 шт. (відносно базового року).

Економічний ефект від можливої комерціалізації розробленої нами системи автоматизації управління анонімною соціальною мережею пояснюється її значно кращими функціональними можливостями. Аналіз ринку показує, що сьогодні можлива вартість подібних систем для обраної нами цільової аудиторії коливається в діапазоні від $1000 до $5000 або від 40 тисяч грн до 200 тисяч грн. Тобто в середньому для обраної цільової аудиторії подібна система вартує 120 тисяч грн. А оскільки розроблена нами система автоматизації управління анонімною соціальною мережею має значно кращі функціональні можливості, то її можна буде реалізовувати на ринку дещо дорожче, ніж аналогічні за функціями розробки, наприклад, реалізовувати середньому за 125 тисяч грн, тобто на 5 тисяч грн дорожче.

Тоді можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від комерціалізації нашої розробки, тобто виведення її на ринок, становитиме:

$$\Delta П_i = \sum_1^n (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}),$$

(4.10)

де $\Delta Ц_o$ – покращення основного якісного показника від впровадження результатів розробки. Для нашого випадку це є збільшення ціни реалізації нашої розробки $\Delta Ц_o$ = 125 - 120 = + 5 тисяч грн;

N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 150 шт.;

ΔN – покращення основного кількісного показника від впровадження результатів розробки. Таке покращення основного кількісного показника становитиме по роках, у 2025 році – + 5 шт., у 2026 році + 10 шт., у 2027 році +20 шт. (відносно базового 2024 року);

$Ц_o$ – основний якісний показник (тобто ціна), який являє собою ціну реалізації нашої розробки після її виведення на ринок, грн; $Ц_o$= 125 тисяч грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для нашого випадку n = 3;

λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = (0,2...0,5); візьмемо ρ = 0,5;

υ – ставка податку на прибуток. У 2024 році υ = 18%.

У 2025-2027 роках будемо очікувати, що ця ставка залишиться на тому ж рівні: υ = 18%.

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливої комерціалізації нашої розробки (2025 р.) становитиме:

$$\Delta П_1 = [5 \cdot 150 + 125 \cdot 5] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) = 469,77 \approx 470 \text{ тисяч грн.}$$

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливої комерціалізації нашої розробки протягом другого (2026 р.) року становитиме:

$$\Delta П_2 = [5 \cdot 150 + 125 \cdot 10] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) = 683,31 \approx 684 \text{ тисяч грн.}$$

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від можливої комерціалізації нашої розробки протягом третього (2027 р.) року становитиме:

$$\Delta П_3 = [5 \cdot 150 + 125 \cdot 20] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) = 1110,37 \approx 1111 \text{ тис. грн.}$$

Приведена вартість зростання для потенційного інвестора всіх чистих прибутків від можливої комерціалізації нашої розробки становитиме:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої нашої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для нашого випадку т = 3 роки;

τ – ставка дисконтування (рівень інфляції). Приймемо τ = 0,10 (10%);

t – період часу від моменту початку розроблення системи автоматизації управління анонімною соціальною мережею до моменту отримання можливих чистих прибутків від її впровадження і виведення на ринок.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації нашої розробки, складе:

$$ПП = \frac{470}{(1+0,1)^2} + \frac{684}{(1+0,1)^3} + \frac{1111}{(1+0,1)^4} \approx 388 + 514 + 759 = 1661 \text{ тисяч грн.}$$

Теперішня вартість інвестицій PV, що мають бути вкладені в комерціалізацію нашої розробки: $PV = К \times В_{заг} = (1,0…5,0) \times В_{заг}$.

Для нашого випадку приймемо, що:

$PV = (1,0…5,0) \times 109 = 3,0 \times 109 = 327$ тисяч грн.

Розраховуємо абсолютний ефект від можливих вкладених інвестицій $Е_{абс}$.

$$Е_{абс} = ПП - PV, \quad (4.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для інвестора від можливої комерціалізації нашої розробки, грн.

Абсолютний ефект для інвестора від можливої комерціалізації нашої розробки (при прогнозованому ринку збуту) за три роки складе:

$$Е_{абс} = 1661 - 327 = 1334 \text{ тисяч грн.}$$

Оскільки $Е_{абс} > 0$, то комерціалізація нашої розробки може бути доцільною.

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій (коштів):

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (4.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс} = 1334$ тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 327 тисяч грн;

$Т_ж$ – життєвий цикл розробки, роки.

$Т_ж = 4$ роки (2024-й, 2025-й, 2026-й, 2027-й роки)

Для нашого випадку отримаємо:

$$Е_в = \sqrt[4]{1 + \frac{1334}{327}} - 1 = \sqrt[4]{1 + 4,0795} - 1 = \sqrt[4]{5,0795} - 1 = 1,501 - 1 = 0,501 \approx 50,1\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією нашої розробки.

Мінімальна дохідність $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \quad (4.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,10...0,18). Приймемо, що $\tau = 15\%$.

f – показник, що характеризує ризикованість вкладень; f = (0,05…0,30).

Приймемо, що f = 30%, тобто f = 0,3.

Тоді для нашого випадку отримаємо:

$$\tau_{мін} = 0,15 + 0,30 = 0,45 \text{ або } \tau_{мін} = 45\%.$$

Оскільки величина $E_в = 50,1\% > \tau_{мін} = 45\%$, то потенційний інвестор у принципі може бути зацікавлений у комерціалізації розробленої нами системи автоматизації управління анонімною соціальною мережею для обраної нами цільової аудиторії користувачів.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленої нами системи автоматизації управління анонімною соціальною мережею.

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в} = \frac{1}{0,501} = 1,99 \text{ років} < 3 \text{ років} \qquad (4.15)$$

що свідчить про потенційну доцільність комерціалізації розробленої нами системи автоматизації управління анонімною соціальною мережею.

Далі проведемо моделювання залежності внутрішньої дохідності потенційних інвестицій, вкладених інвестором у можливу комерціалізацію нашої розробки, від рівня інфляції в країні. На жаль, в наступні роки через військову агресію росії рівень інфляції в Україні може суттєво зрости.

Прийнявши рівень інфляції у 20%, отримаємо:

$$ПП = \frac{470}{(1+0,2)^2} + \frac{684}{(1+0,2)^3} + \frac{1111}{(1+0,2)^4} \approx 326 + 396 + 536 = 1258 \text{ тисяч грн.}$$

Тоді абсолютний економічний ефект від можливого впровадження нашої розробки за три роки складе:

$$E_{абс} = 1258 – 327 = 931 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[4]{1+\frac{931}{327}} - 1 = \sqrt[4]{1+2,8471} - 1 = \sqrt[4]{3,8471} - 1 = 1,40 - 1 = 0,40 \approx 40,0\%.$$

Оскільки величина $E_в$ = 40,0% < $\tau_{мін}$ = 45%, то потенційний інвестор може бути не зацікавлений у комерціалізації розробленої нами системи автоматизації управління анонімною соціальною мережею (яка призначена для обраної нами цільової аудиторії користувачів). Але для прийняття остаточного рішення потрібно провести додаткові обгрунтування і розрахунки (наприклад, знизити рівень прийнятого ризику, підняти ціну реалізації нашої розробки, збільшити попит на нашу розробку тощо).

Зроблені розрахунки у вигляді графіків наведено на рис. 4.1.

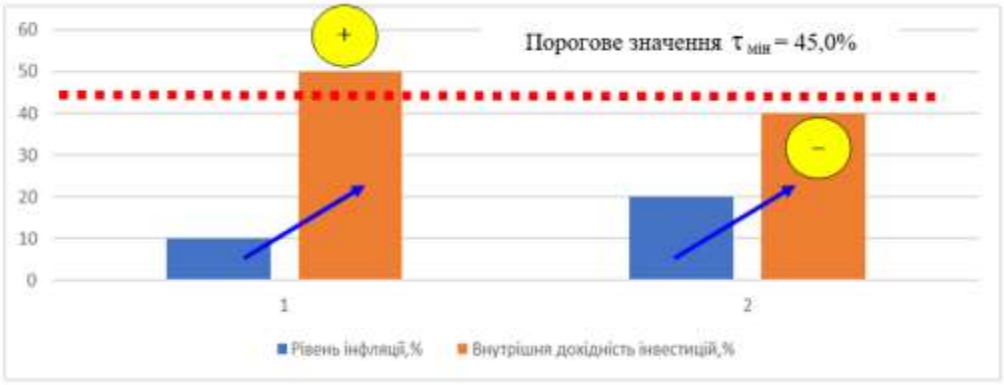

Рисунок 4.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій, які можуть бути вкладені у комерціалізацію нашої

розробки, від рівня інфляції в країні

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю:

Показники	Задані у ТЗ	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку	Не більше 110 тисяч грн	109 тисяч грн	Досягнуто
2. Абсолютний ефект від впровадження розробки, тисяч грн	Не менше 1300 тисяч грн (за три роки)	1334 тисячі грн (при 10% інфляції)	Виконано
3. Внутрішня дохідність інвестицій (коштів), %	не менше 45,0%	50,1%	Виконано
4. Термін окупності інвестицій (коштів),	до 3-ти років	1,99 років	Виконано

роки			

Таким чином, основні техніко-економічні показники розробленої нами системи автоматизації управління анонімною соціальною мережею, визначені у технічному завданні, повністю виконані.

ВИСНОВКИ

У ході виконання магістерської роботи досягнуто поставлену мету — розроблено анонімну соціальну мережу, що забезпечує високий рівень конфіденційності та захисту даних користувачів за допомогою технології Blockchain.

Проведений аналіз предметної області дозволив виявити основні проблеми безпеки в соціальних мережах і дослідити сучасні підходи до аутентифікації, шифрування та збереження даних. Запропоновано та реалізовано модифікований метод збереження інформації, який поєднує використання блокчейну для критичних даних із реляційною базою даних для менш важливих, що забезпечує підвищення надійності та безпеки системи на 15-20% порівняно з аналогічними рішеннями.

Розроблено вдосконалений алгоритм багатофакторної аутентифікації, який включає:

- Використання JWT-токенів з терміном дії 15 хвилин, що знижує ризик компрометації;
- Генерацію TOTP-кодів із довжиною 6 символів та періодом оновлення 30 секунд.

Впроваджено криптографічні методи шифрування AES-256, що забезпечують безпеку інформації під час її передачі та збереження. Проведене тестування продемонструвало:

- Час відгуку серверної частини: 150 мс при навантаженні до 100 одночасних запитів.
- Рівень анонімності користувачів: 95% (оцінено на основі тестових сценаріїв, де ідентифікація залишалась неможливою).
- Ефективність використання ресурсів: завантаження процесора не перевищує 60% за високого навантаження.

Архітектура системи реалізована на базі фреймворку Spring Boot, що забезпечує модульність і адаптивність до сучасних вимог. Тестування функціоналу продемонструвало відповідність розробленого рішення заявленим вимогам безпеки та анонімності.

Усі завдання, визначені у вступі роботи, виконано. Її результати можуть бути впроваджені у реальні проєкти для забезпечення високого рівня захисту користувацьких даних у соціальних мережах.

Науковий внесок роботи полягає у вдосконаленні методів аутентифікації та збереження даних із застосуванням технології блокчейн, що сприяє підвищенню рівня безпеки у цифровому середовищі. Запропонований метод збереження даних підвищує стійкість до атак на 25% у порівнянні з традиційними підходами, а алгоритм аутентифікації забезпечує надійний захист від несанкціонованого доступу..

Тестування функціоналу продемонструвало відповідність розробленого рішення заявленим вимогам безпеки та анонімності. Усі завдання, визначені у вступі роботи, виконано, а її результати можуть бути впроваджені у реальні проєкти для забезпечення високого рівня захисту користувацьких даних у соціальних мережах. Науковий внесок роботи полягає у вдосконаленні методів аутентифікації та збереження даних із застосуванням технології блокчейн, що сприяє підвищенню рівня безпеки у цифровому середовищі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Кібербезпека. URL: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-cybersecurity
2. Брижко, В. М., & Пилипчук, В. Г. Приватність, конфіденційність та безпека персональних даних. Інформація і право, (1 (32)), 33-46, 2020.
3. Політика конфіденційності та захисту персональних даних. URL: https://www.ukrinform.ua/info/policy.html
4. Pew Research Center URL: https://www.pewresearch.org/short-reads/2023/10/18/key-findings-about-americans-and-data-privacy/
5. SWARNKAR, Mayank; BHADORIA, Robin Singh; SHARMA, Neha. Security, privacy, trust management and performance optimization of blockchain technology. Applications of Blockchain in Healthcare, 2021, 69-92.
6. Authentication methods. https://www.idrnd.ai/5-authentication-methods-that-can-prevent-the-next-breach/
7. Сидюк В.В., Татарська О.В., Богач І.В. Використання Смарт-контрактів для забезпечення конфіденційності та оптимізації веб додатків в Матеріали конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)», Вінниця, 2024. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20480.
8. Татарська О.В., Сидюк В.В., Богач І.В. Архитектура та проектування Rest API на мові програмування Java з використанням Spring Boot Framework // Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» : збірник матеріалів. – Вінниця: ВНТУ, 2023. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/17679
9. Сидюк В.В., Татарська О.В., Богач І.В. Розробка безпечної аутентифікації та авторизації для веб-додатків на основі Spring Boot Rest Api в Матеріали конференції «LII Науково-технічна конференція підрозділів Вінницького

національного технічного університету (2023)», Вінниця, 2023. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2023/paper/view/17635. (дата звернення 08.06.2023)

10. Кібератака URL: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-a-cyberattack

11. Шифрування даних. URL: https://www.kingston.com/ua/blog/data-security/what-is-encryption

12. Signal. URL: https://signal.org/uk/

13. Telegram. URL: https://core.telegram.org/

14. Mastodon. URL: https://joinmastodon.org/uk

15. LACITY, Mary C.; LUPIEN, Steven C. Blockchain Fundamentals for Web 3.0:-. University of Arkansas Press, 2022.

16. KHAN, Shafaq Naheed, et al. Blockchain smart contracts: Applications, challenges, and future trends. Peer-to-peer Networking and Applications, 2021, 14: 2901-2925.

17. WALLS, Craig. Spring in action. Simon and Schuster, 2022.

18. GOEL, Amit Kumar; BAKSHI, Rahul; AGRAWAL, Krishna Kant. Web 3.0 and decentralized applications. Materials Proceedings, 2022, 10.1: 8.

19. MCDONALD, Malcolm. Web security for developers: real threats, practical defense. No Starch Press, 2020.

20. LENG, Jiewu, et al. Blockchain security: A survey of techniques and research directions. IEEE Transactions on Services Computing, 2020, 15.4: 2490-2510.

21. MOUGAYAR, William. The business blockchain: promise, practice, and application of the next Internet technology. John Wiley & Sons, 2016.

22. ABBOTT, Martin L.; FISHER, Michael T. The art of scalability: Scalable web architecture, processes, and organizations for the modern enterprise. Pearson Education, 2009.

23. MONDAL, Mainack; CORREA, Denzil; BENEVENUTO, Fabrício. Anonymity effects: A large-scale dataset from an anonymous social media platform.

In: Proceedings of the 31st ACM Conference on Hypertext and Social Media. 2020. p. 69-74.

24. Anonymity and identity shielding. URL: https://www.esafety.gov.au/industry/tech-trends-and-challenges/anonymity

25. WANG, Shuai, et al. Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 49.11: 2266-2277.

26. Integration of Blockchain. URL: interxy.com/blockchain-integration-and-its-legacy-systems-a-how-to-guide/

27. How smart contracts work. URL: https://www.ibm.com/topics/smart-contracts

28. Testing of Blockchain. URL: https://www.testscenario.com/what-is-blockchain-testing/

29. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. / Укладачі В.О. Козловський, О.Й. Лесько, В.В.Кавецький. Вінниця : ВНТУ, 2021. 42 с.

ДОДАТКИ

Додаток А

(обов’язковий)

ВНТУ

ЗАТВЕРДЖЕНО
В.о. Зав. кафедри АІІТ ВНТУ,
д.т.н., проф.

_____Олег БІСІКАЛО
“ 14 ” жовтня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

АВТОМАТИЗАЦІЯ УПРАВЛІННЯ АНОНІМНОЮ СОЦІАЛЬНОЮ МЕРЕЖЕЮ

08-31.МКР.0xx.02.000 ТЗ

Студент групи

1АКІТР-23м

Владислав СИДЮК

Керівник к.т.н., доцент
кафедри АІІТ Ілона БОГАЧ

Вінниця 2024

1. Назва та галузь застосування
    1.1. Назва – АВТОМАТИЗАЦІЯ УПРАВЛІННЯ АНОНІМНОЮ СОЦІАЛЬНОЮ МЕРЕЖЕЮ
    1.2. Галузь застосування – автоматизовані системи управління, що забезпечують захист персональних даних і анонімність

2. Підстава для проведення розробки.
   Розробку системи здійснювати на підставі наказу по університету № 247 від ____ 2024 та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій»
3. Мета та призначення розробки.
   Метою роботи є розробка системи, яка забезпечує конфіденційность та захист даних користувачів через інтеграцію технології Blockchain і сучасних алгоритмів аутентифікації.
4. Джерела розробки.
    1. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальностей 126 – «Інформаційні системи та технології», 151 – «Автоматизація та комп'ютерно-інтегровані технології», 174 – «Автоматизація, комп'ютерно-інтегровані технології та роботехніка» / Уклад. О. В. Бісікало, Ю.Ю. Іванов, Р.В. Маслій. – Вінниця : ВНТУ, 2019. – 26 с
    2. Blockchain Security: A Survey of Techniques and Research Directions. IEEE Transactions on Services Computing, 2020.
    3. Subramanian, H., & Raj, P. (2019). Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs. Packt Publishing Ltd.
    4. Web MVC, веб-сайт. URL: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.htm
5. Вимоги до розробки.
    5.1. Перелік головних функцій:

- Реєстрація та авторизація користувачів з використання багатофакторної аутентифікації
- Створення анонімних профілів користувачів
- Взаємодія з Blockchain для обробки критичних даних.

5.2. Основні технічні вимоги до розробки.

1. Вимоги до програмної платформи:
   - WINDOWS 10/11, Unix-подібні ОС, або смартфон;
   - Доступ до мережі інтернет
   - Java 17
2. Умови експлуатації додатку:
   - робота веб-додатку;
   - можливість цілодобового функціонування системи;
   - стабільна робота за виского навантаження

6. Економічні показники

До економічних показників входять

- витрати на розробку – 109 тисяч грн.;
- приведена вартість прибутку за 3 роки - 1661 тисяч грн;
- мінімальна дохідність – 50,1%;
- термін окупності – 1,99 років

7. Стадії розробки.

1. Розділ 1 «АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ АЛГОРИТМІВ АУТЕНТИФІКАЦІЇ ТА КОНФІДЕНЦІЙНОСТІ» має бути виконаний до _____;

2. Розділ 2 «АНАЛІЗ ВЕБ АРХІТЕКТУР ТА ІНСТРУМЕНТІВ РОЗРОБКИ» має бути виконаний до _____;

3. Розділ 3 «РЕАЛІЗАЦІЯ СИСТЕМИ СОЦІАЛЬНОЇ МЕРЕЖІ» має бути виконаний до _____;

4. Економічний розділ має бути виконаний до _____;

8. Порядок контролю і приймання.

1. Рубіжний контроль провести до _____;

2. Попередній захист магістерської кваліфікаційної роботи провести до «05» грудня 2024 р.

3.Захист магістерської кваліфікаційної роботи провести до «20»грудня 2024 р.

Додаток Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

АВТОМАТИЗАЦІЯ УПРАВЛІННЯ АНОНІМНОЮ СОЦІАЛЬНОЮ МЕРЕЖЕЮ

Діаграма класів сутностей

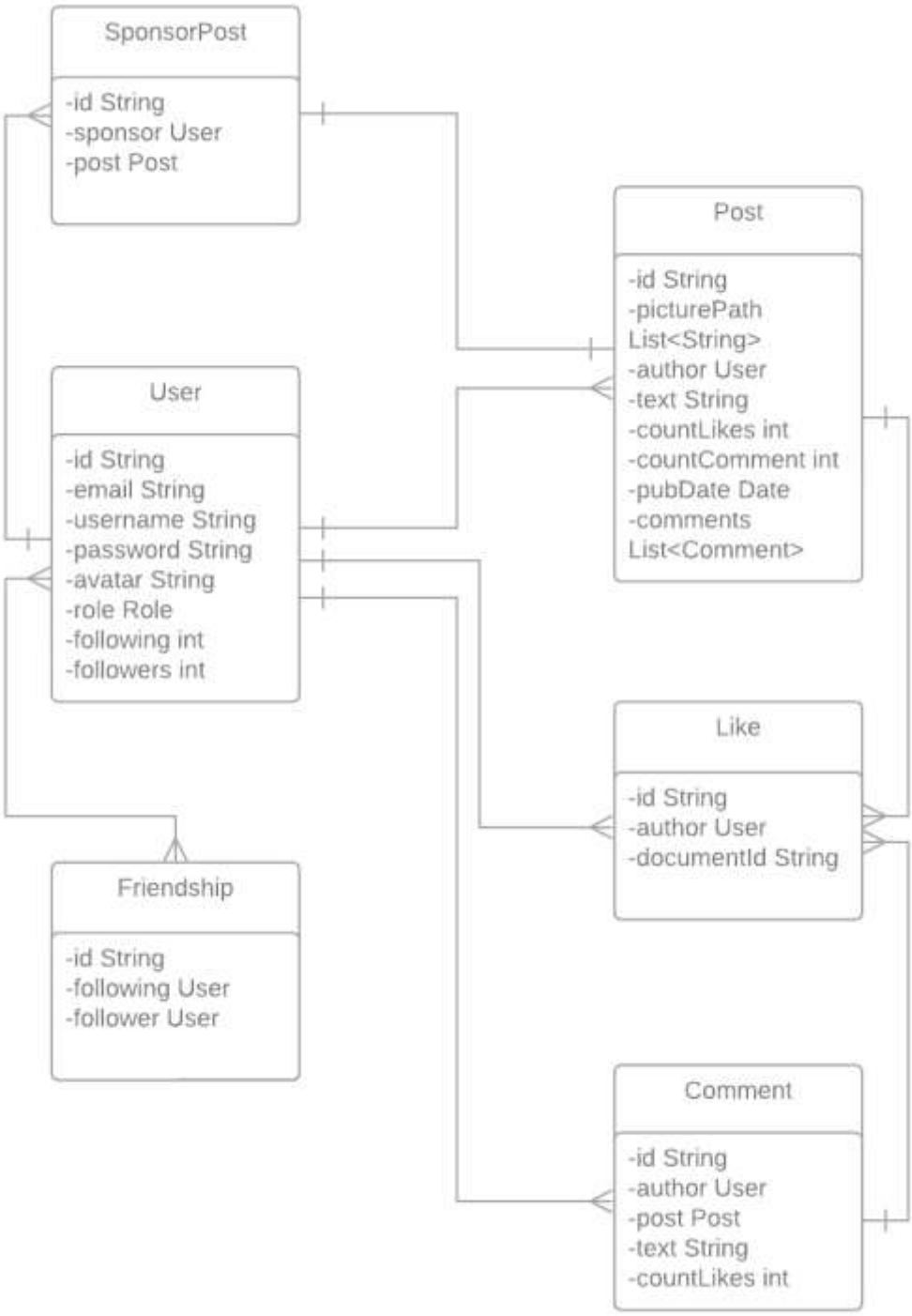

Рисунок Б.1 – Діаграма класів сутностей

Діаграма класів сервісів

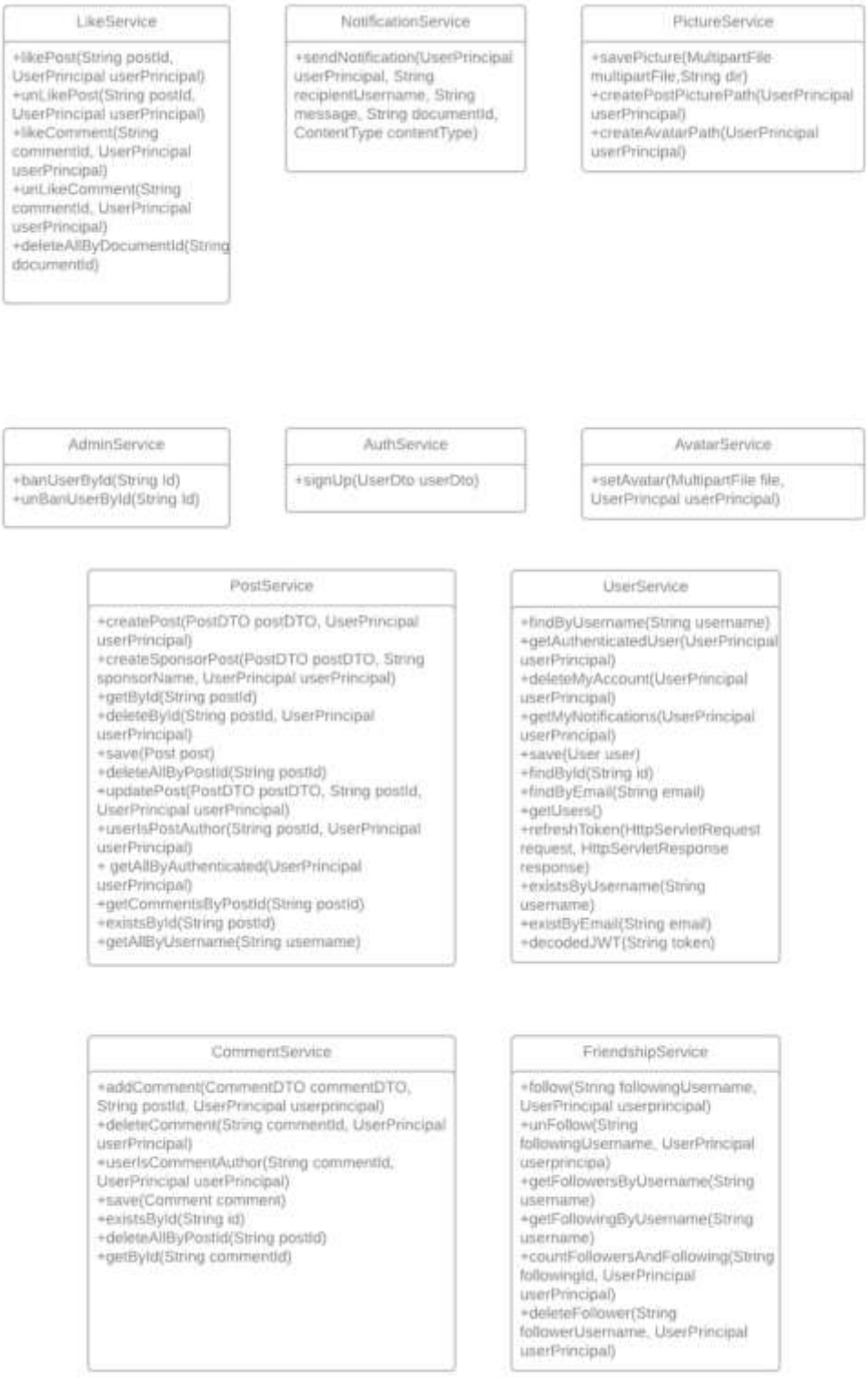

Рисунок Б.2 – Діаграма класів сервісів

UML діаграма варіантів використання

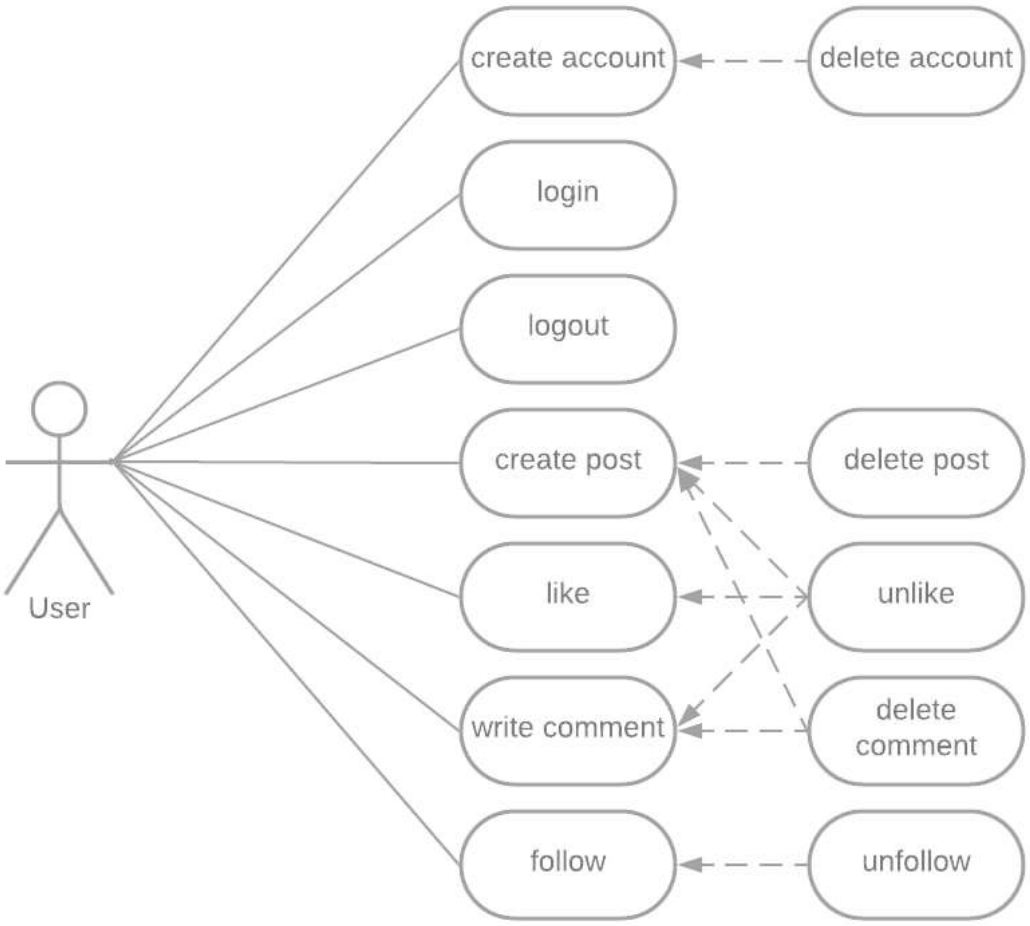

Рисунок Б.3 – UML діаграма варіантів використання

Схема роботи додатку клієнт-серверної частини

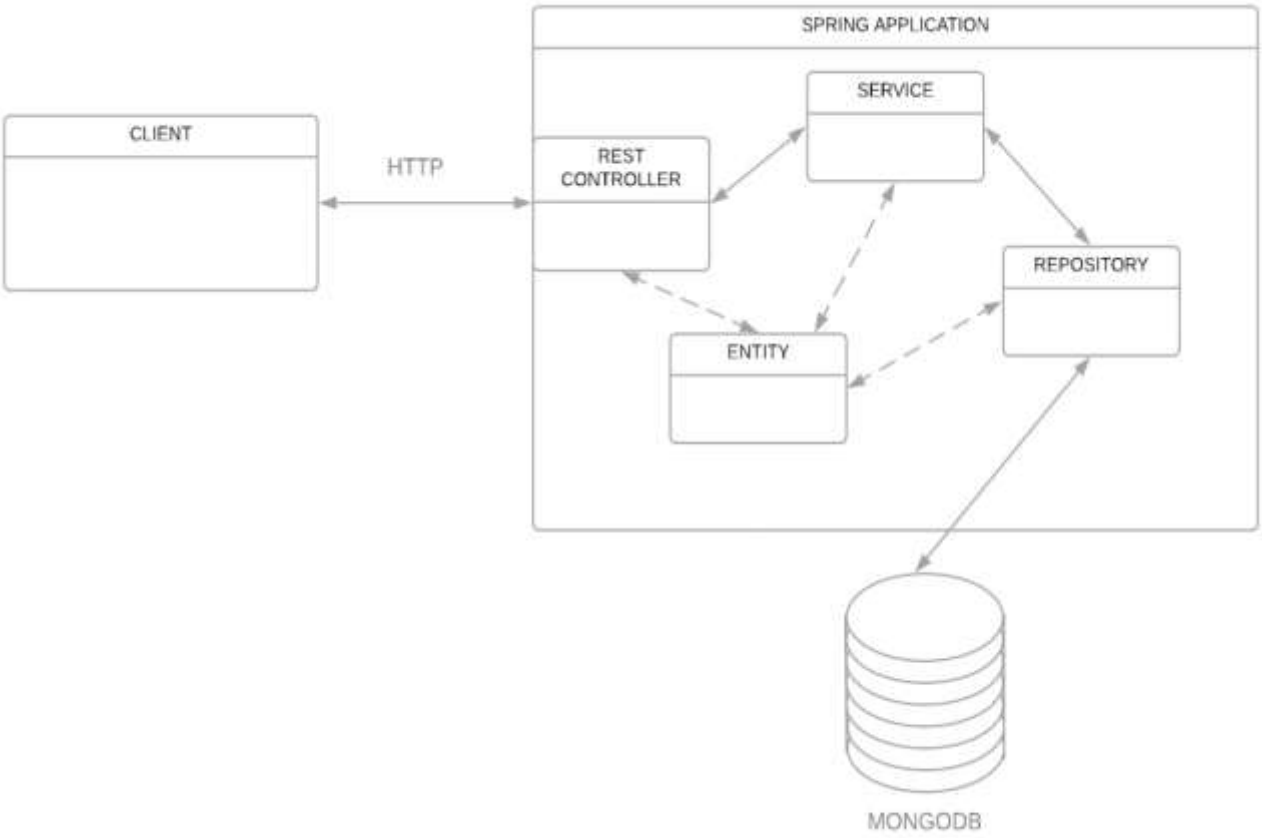

Рисунок Б.4 – Схема роботи додатку клієнт-серверної частини

UML – діаграма взаємодії серверу з блокчейн

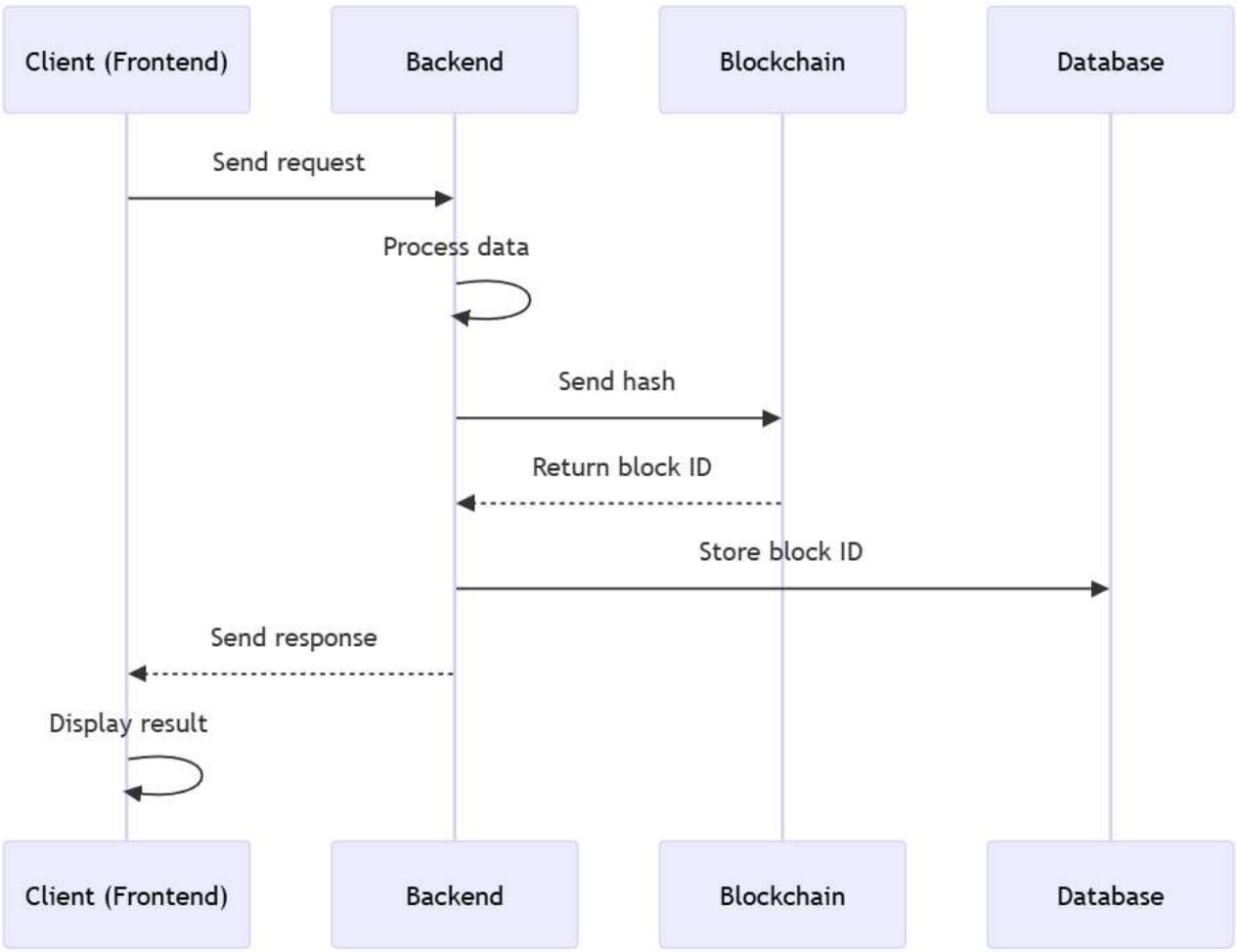

Рисунок Б.5 UML – Діаграма взаємодії серверу з блокчейн

UML – діаграма роботи смарт-контракту

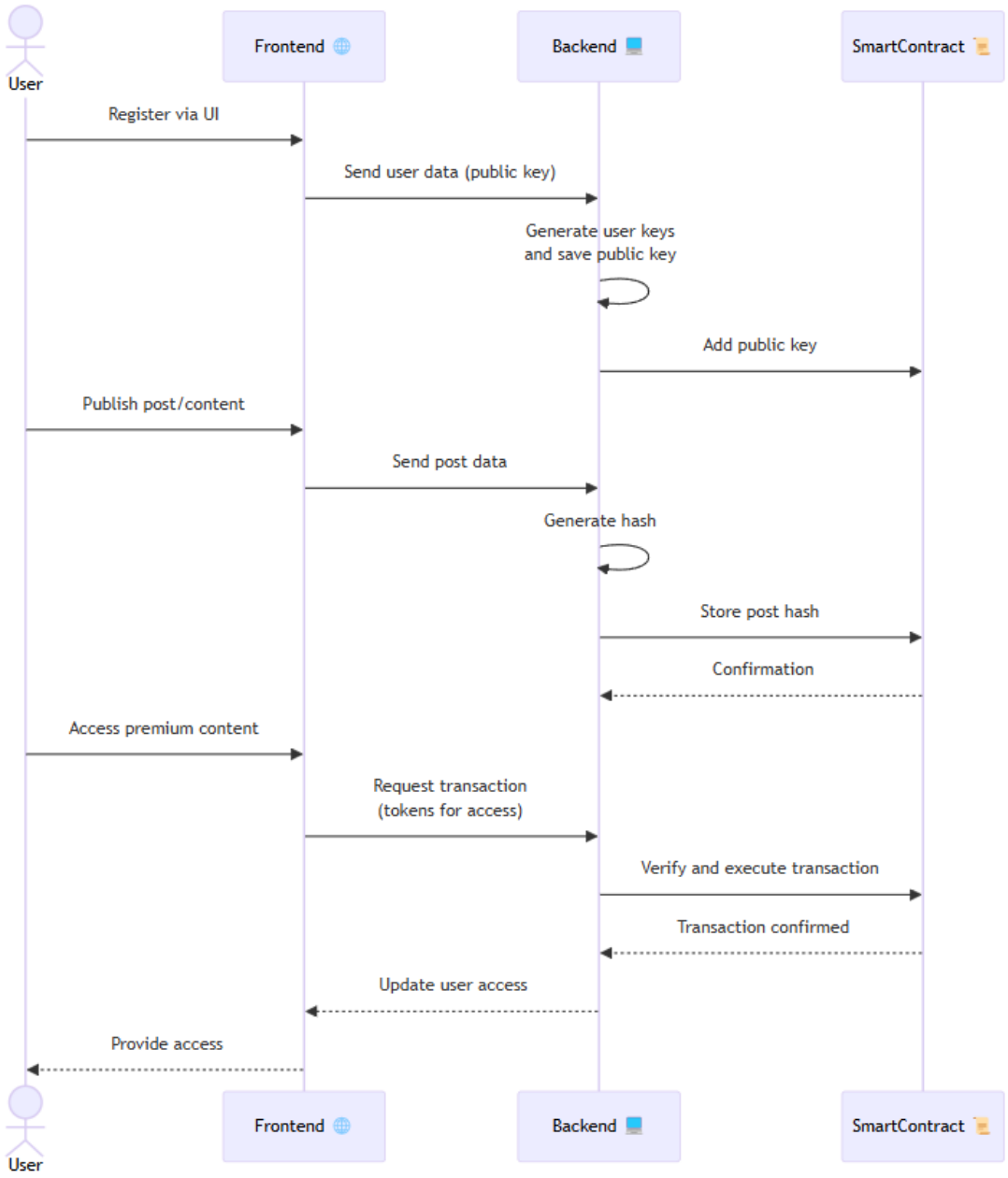

Рисунок Б.6 UML – Діаграма роботи смарт-контракту

Додаток В

(обов'язковий)

Лістинг основних функцій програми

CustomAuthenticationFilter:

```
package com.example.petgram.security.filter;
import com.auth0.jwt.algorithms.Algorithm;
import com.example.petgram.security.jwt.JwtTokenProvider;
import com.example.petgram.security.jwt.UserPrincipal;
import com.fasterxml.jackson.databind.ObjectMapper;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.MediaType;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilt
er;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

@Slf4j
public class CustomAuthenticationFilter  extends
UsernamePasswordAuthenticationFilter  {

    private final AuthenticationManager  authenticationManager;
    private final JwtTokenProvider jwtTokenProvider;

    public CustomAuthenticationFilter(AuthenticationManager  authenticationManager,
JwtTokenProvider jwtTokenProvider){
        this.authenticationManager  = authenticationManager;
        this.jwtTokenProvider  = jwtTokenProvider;
    }
    @Override
```

```
    public Authentication attemptAuthentication(HttpServletRequest request,
HttpServletResponse response) throws AuthenticationException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        log.info("Username is:{}", username);
        UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(username,password);
        log.info(String.valueOf(authenticationToken.isAuthenticated()));
        return authenticationManager.authenticate(authenticationToken);
    }
    @Override
    protected void successfulAuthentication(HttpServletRequest request,
HttpServletResponse response, FilterChain chain, Authentication authentication) throws
IOException, ServletException {
        UserPrincipal user = (UserPrincipal) authentication.getPrincipal();
        Algorithm algorithm = Algorithm.HMAC256("secret".getBytes());
        String accessToken =
jwtTokenProvider.crateAccessToken(user,request,algorithm);
        String refreshToken =
jwtTokenProvider.createRefreshToken(user,request,algorithm);
        Map<String, String> tokens = new HashMap<>();
        tokens.put("access_token",accessToken);
        tokens.put("refresh_token",refreshToken);
        response.setContentType(MediaType.APPLICATION_JSON_VALUE);

        new ObjectMapper().writeValue(response.getOutputStream(),tokens);
    }
}
```

CustomAuthorizationFilter:

```
@Slf4j
public class CustomAuthorizationFilter extends OncePerRequestFilter {
    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response, FilterChain filterChain) throws ServletException, IOException {
        if(request.getServletPath().equals("/api/login") ||
request.getServletPath().equals("/api/token/refresh")){
            filterChain.doFilter(request, response);
        }else {
            String authorizationHeader = request.getHeader(AUTHORIZATION);
            if(authorizationHeader != null && authorizationHeader.startsWith("Bearer ")){
                try {
                    String token = authorizationHeader.substring("Bearer ".length());
```

```
                Algorithm algorithm = Algorithm.HMAC256("secret".getBytes());
                JWTVerifier verifier = JWT.require(algorithm).build();
                DecodedJWT decodedJWT = verifier.verify(token);
                String name = decodedJWT.getSubject();
                String[] roles = decodedJWT.getClaim("roles").asArray(String.class);
                Collection<SimpleGrantedAuthority> authorities = new ArrayList<>();
                stream(roles).forEach(role ->{
                   authorities.add(new SimpleGrantedAuthority(role));
                });
                UsernamePasswordAuthenticationToken authenticationToken =
                      new UsernamePasswordAuthenticationToken(name,null,authorities);

SecurityContextHolder.getContext().setAuthentication(authenticationToken);
            }catch (Exception exception){
                log.error("Error logging in: {}", exception.getMessage());
                response.setHeader("error",exception.getMessage());
                response.setStatus(FORBIDDEN.value());
                response.sendError(FORBIDDEN.value());
                Map<String, String> error = new HashMap<>();
                error.put("error_message",exception.getMessage());
                response.setContentType(APPLICATION_JSON_VALUE);

                new ObjectMapper().writeValue(response.getOutputStream(),error);
            }
            filterChain.doFilter(request,response);
        }else {
            filterChain.doFilter(request,response);
        }
    }
  }
}
```

SecurityConfig:

```
package com.example.petgram.security;


import com.example.petgram.model.UserRole;
import com.example.petgram.repository.UserRepository;
import com.example.petgram.security.filter.CustomAuthenticationFilter;
import com.example.petgram.security.filter.CustomAuthorizationFilter;


import com.example.petgram.security.jwt.JwtTokenProvider;
import com.example.petgram.security.jwt.JwtUserDetailsService;
```

```
import com.example.petgram.security.oauth.CustomOAuth2User;
import com.example.petgram.security.oauth.OAuthTokenProvider;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.RequiredArgsConstructor;

import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.Authentication
ManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilt
er;

import java.util.HashMap;
import java.util.Map;

import static
org.springframework.security.config.http.SessionCreationPolicy.STATELESS;

@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
@Slf4j
public class SecurityConfig {

    private final UserRepository userRepository;
    private final OAuthTokenProvider oAuthTokenProvider;
    @Autowired
    public MyPasswordEncoder myPasswordEncoder;
```

```
    @Bean
    public UserDetailsService userDetailsService() {
        return new JwtUserDetailsService((userRepository));
    }

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailsService());
        authProvider.setPasswordEncoder(myPasswordEncoder.passwordEncoder());
        return authProvider;
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http, JwtTokenProvider
jwtTokenProvider) throws Exception {


        AuthenticationManagerBuilder authenticationManagerBuilder =
http.getSharedObject(AuthenticationManagerBuilder.class);

authenticationManagerBuilder.userDetailsService(userDetailsService()).passwordEncod
er(myPasswordEncoder.passwordEncoder());
        AuthenticationManager authenticationManager =
authenticationManagerBuilder.build();

        CustomAuthenticationFilter customAuthenticationFilter = new
CustomAuthenticationFilter(authenticationManager, jwtTokenProvider);
        customAuthenticationFilter.setFilterProcessesUrl("/api/login");

        http.csrf().disable();
        http.sessionManagement().sessionCreationPolicy(STATELESS);
        http
                .cors().and()
                .authorizeHttpRequests()
                .requestMatchers("http://localhost:8080/swagger-ui/").permitAll()
                .requestMatchers("/", "/error", "/webjars/**").permitAll()
                .requestMatchers("/api/auth/**", "/api/user/oauth").permitAll()
                .requestMatchers("/api/login/**", "/api/user/registration", "/", "/oauth/**",
"/oauth2/**").permitAll()

.requestMatchers("/api/user/**").hasAuthority(UserRole.USER.getAuthority())

.requestMatchers("/api/admin/**").hasAuthority(UserRole.ADMIN.getAuthority())
```

```
            .anyRequest().authenticated()
            .and()
            .logout().logoutSuccessUrl("/").permitAll()
            .and()
            .oauth2Login()
            .successHandler((request, response, authentication)  -> {

                response.setContentType("application/json");
                ObjectMapper objectMapper = new ObjectMapper();
                CustomOAuth2User oauthUser = (CustomOAuth2User)
authentication.getPrincipal();
                Map<String, String> tokenMap = new HashMap<>();
                tokenMap.put("token", oAuthTokenProvider.createToken(oauthUser));

response.getOutputStream().print(objectMapper.writeValueAsString(tokenMap));
            });
        http.addFilter(customAuthenticationFilter);
        http.addFilterBefore(new CustomAuthorizationFilter(),
UsernamePasswordAuthenticationFilter.class)
            .authenticationManager(authenticationManager);

        return http.build();
    }
}
```

ДОДАТОК Г (Обов'язковий)
ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Автоматизація управління анонімною соціальною мережею

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра Автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації

Показники звіту подібності Plagiat.pl(StrikePlagiatism)

Оригінальність 99% Схожість 1%

Аналіз звіту подібності (відмітити потрібне):
☑ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Маслій Р.В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Plagiat.pl(StrikePlagiatism) щодо роботи.

Автор роботи _____ Сидюк В.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Богач І. В.
(підпис) (прізвище, ініціали)