НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__”_____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”

на тему: : «Соціальна мережа з адаптивною рекомендацією користувачів»

Виконав: студент 4 курсу, групи ІО-71
(шифр групи)
Костинюк Олександр Анатолійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викладач Алєнін.О.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) д.т.н. проф. Сімоненко В.П. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__”_____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Костинюка Олександра Анатолійовича

1. Тема проєкту *Соціальна мережа з адаптивною рекомендацією користувачів*
керівник проєкту_____ст. викладач Алєнін Олег Ігорович_____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від_____2021 року №_____
2. Термін здачі студентом закінченого проєкту_____*20 травня 2021 р.*
3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) *Опис та аналіз предметної області, аналіз аналогів у даній області, виокремлення основних вимог до ПЗ, проектування архітектури соціальної мережі для пошуку знайомств та тестування.*

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) *структурна схема системи, загальнена схема збереження даних, схема алгоритму користування соціальною мережею.*

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання_____

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2020-15.12.2020*	
2.	*Вивчення та аналіз завдання*	*15.12.2020-15.03.2021*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2021-25.03.2021*	
4.	*Розробка структур окремих підсистем*	*25.03.2021-5.04.2021*	
5.	*Програмна реалізація системи*	*5.04.2021-15.04.2021*	
6.	*Оформлення пояснювальної записки*	*15.04.2021-20.05.2021*	
7.	*Захист програмного продукту*	*25.04.2021*	
8.	*Передзахист*	*23.05.2021*	
9.	*Захист*	*14.06.2021*	

Студент-дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

Анотація

В даній бакалаврській дипломній роботі спроектовано та реалізовано модель соціальної мережі з адаптивним та ручним пошуком користувачів на основі заповнених фільтрів або даних, які система генерує самостійно базуючись на інформації користувача.

Система є готовим закінченим продуктом, який вже може використовуватися.

Програмна частина реалізована мовою програмування JavaScipt із використанням платформи Node.js та бібліотеки React.js.

.

Annotation

In the given bachelor thesis work, a model of a social network with adaptive and manual search of users was designed and implemented. Search is based on filled filters or data that the system generates based on the user information.

The system is a ready-made finished product that can already be used.

The program part is implemented in the JavaScipt programming language using the Node.js platform and the React.js library.

Опис альбому
до дипломного проекту

на тему: «Соціальна мережа з адаптивною рекомендацією користувачів»

Київ – 2021 року

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ ек-земпляра	*Додаток*
			Документація загальна	1		
			Знову розроблена			
	А4	***ІАЛЦ.467100.001 ОА***	Соціальна мережа з	1		
			адаптивною рекомендацією			
			користувачів			
			Опис альбому			
	А4	***ІАЛЦ.467100.002 ТЗ***	Соціальна мережа з	3		
			адаптивною рекомендацією			
			користувачів			
			Технічне завдання			
	А1	***ІАЛЦ.467100.003 ПЗ***	Соціальна мережа з	82		
			адаптивною рекомендацією			
			користувачів			
			Пояснювальна записка			
	А1	***ІАЛЦ.467100.004 Д1***	Соціальна мережа з	1		
			адаптивною рекомендацією			
			користувачів			
			Структурна схема роботи			
	А1	***ІАЛЦ.467100.005 Д2***	Соціальна мережа з	1		
			адаптивною рекомендацією			
			користувачів			
			Функціональна схема даних			
	А1	***ІАЛЦ.467100.006 Д3***	Соціальна мережа з	1		
			адаптивною рекомендацією			
			користувачів			
			Принципова схема роботи			

					ІАЛЦ.467100.001 ОА		
Зм	*Лист*	*№ докум.*	*Підп.*	*Дата*			
Розроб		Костинюк О.А.			*Соціальна мережа з адаптивною рекомендацією користувачів Опис альбому*	Літ.	Арк. 1 / Аркушів 1
Перев		Алєнін О. І.					
Реценз.						*НТУУ "КПІ" ФІОТ ІО-71*	
Н. Контр.		*Сімоненко В.П.*					
Затв.		*Стіренко С. Г.*					

Технічне завдання
до дипломного проекту

на тему: «Соціальна мережа з адаптивною рекомендацією користувачів»

Київ – 2021 року

ЗМІСТ

					ІАЛЦ.467100.002 ТЗ			
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розробив		*Костинюк О.А.*			*Реалізація спеціалізованої нейронної мережі на мобільній платформі* *Технічне завдання*	*Літ.*	*Арк.*	*Аркушів*
Перевірив		*Алєнін О. І.*					1	3
Реценз.						*НТУУ «КПІ ім. І.Сікорського» каф. ОТ, гр. ІО-71*		
Н. Контр.		*Сімоненко В.П.*						
Затв.		*Стіренко С.Г.*						

1.НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Соціальна мережа з адаптивною рекомендацією користувачів». Область застосування: соціальна мережа допомагає в пошуку нових знайомств, дозволяючи вказувати необхідні параметри або автоматично знаходити людей, базуючись на даних користувача.

2.ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3.МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення соціальної мережі з адаптивною рекомендацією користувачів.

4.ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, довідники по платформах дистанційного навчання, публікації в мережі Інтернет по даній темі.

5.ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Здатність коректно обробляти дані.
- Висока швидкість роботи, в тому числі, обробки запитів.
- Простий та зрозумілий для використання інтерфейс.

5.2. Вимоги до програмного забезпечення

- Node.js 12 та вище
- React.js 16 та вище

5.3. Вимоги до апаратної частини

- Підключення до Internet
- Оперативна пам’ять не менше 2 Гбайт
- Сервер із підтримкою розгортання додатків на Node.js

6. ЕТАПИ РОЗРОБКИ

	Дата
6.1. Вивчення необхідної літератури	24.02.2021
6.2. Складання та узгодження технічного завдання	08.03.2021
6.3. Написання вступу та огляду готових рішень	22.03.2021
6.4. Проектування архітектури додатку	04.04.2021
6.5. Написання програмної частини	10.04.2021
6.6. Тестування і виправлення помилок	07.05.2021
6.7. Підготовка документації дипломного проекту	19.05.2021

Пояснювальна записка
до дипломного проєкту

на тему: «Соціальна мережа з адаптивною рекомендацією користувачів»

Київ – 2021 року

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

JSON	(JavaScript Object Notation) Об’єкт позначення Джаваскрипт
JSX	(JavaScript XML) Формат Джаваскріпту XML
UI	(UI) Користувацький інтерфейс
SQL	(Structured Query Language) Структурна мови запитів
REST	(Representational State Transfer) Представницький трансфер стану
MVC	(Model-View-Controller) Архітектура «Модель-Вид-Контролер»
СУБД	Система управління базами даних

ВСТУП

Сьогодні, в час новітніх технологій, безліч речей переноситься в цифровий вигляд. Наш світ захопили комп'ютери, смартфони та інші гаджети за допомогою яких можна проводити час в Інтернеті. Пошук товаришів, друзів та просто людей для різних цілей тому не виключення. Все більше й більше людей вдаються до саме такого способу пошуку знайомств. Такий тренд тільки буде ставати ще більш актуальним, враховуючи простоту та доступність цього підходу

Актуальність теми. Через зручність соціальних мереж, а саме, можливість транслювання своїх думок, навичок, вмінь, та взагалі будь-чого, у 2020 році понад 3,6 мільярда людей користувалися соціальними мережами у всьому світі, прогнозується, що їх кількість збільшиться до майже 4,41 мільярда у 2025 році. Якщо порівняти ці дані з 2 мільярдами користувачів ще всього у 2015 році, ми наглядно побачимо, що соціальні мережі зараз – невід'ємна частина життя не лише для підлітків, що прагнуть розваг та дурнощів, сьогодні за допомогою цих платформ ми можемо робити речі, про які раніше мало хто підозрював. Наприклад, пошук роботи, торгова діяльність, та навіть, звичне нам бронювання столиків в улюблених ресторанах – все це зараз можна зробити за допомогою соціальних мереж.

Мета і задачі дослідження. Метою даної роботи є розробка такої моделі, що допоможе людям знаходити знайомства базуючись на їх вподобаннях, а також, в подальшому, продовжувати спілкування в даному ж застосунку.

Для виконання поставленої мети були сформовані наступні задачі:

- Провести аналіз уже готових популярних мереж задля виділення їх найкращих сторін, щоб розглянути можливість їх імплементації в нашій моделі, та недоліків, задля їх уникання;
- Визначитися з технологіями, що найкраще підійшли б для розробки даної моделі:

- Виконати тестове моделювання створеної моделі;
- Зробити висновки по досягненню кінцевої мети враховуючи отримані результати

Практичне значення. Підготовлена модель дозволить успішно знаходити нові знайомства, причому користувачу не доведеться вручну продивлятись безліч профілів, вишукуючи потрібну інформацію. Модель базуючись на наданій інформації користувачем, або на інформації самого користувача підбере найбільш підходящих користувачів, зареєстрованих в системі. Це допоможе зекономити багато сил та часу для шукача.

У розділі 1 проведено огляд найпопулярніших існуючих соціальних мереж, в яких можна заводити знайомства та наведено порівняльну характеристику цих платформ, а також, визначено переваги та недоліки окремо для кожної з них.

У розділі 2 визначено технології та способи їх коректної інтеграції, які були обрані для вирішення раніше поставленої задачі. Розглянуто чому саме вони найкраще підходять для конкретного випадку, їх переваги та недоліки в рамках кінцевої моделі.

У розділі 3 описано та втілено в життя структуру системи з використання методів, які були запропоновані в розділі 2. Проведено оцінку моделі при різних вхідних даних користувачів, а також, наведено можливі способи покращення та подальшого розширення системи.

У розділі 4 описано та продемонстровано роботу системи, а також перевірено коректність при різних початкових умовах та на різних середовищах.

Ключові слова. Соціальна мережа, аналіз, інтереси, масштабованість, система, адаптивність, пошук, обробка данних

РОЗДІЛ 1
ОГЛЯД ВЖЕ ГОТОВИХ РЕАЛІЗАЦІЙ

1.1 Опис завдання

Програмна частина продукту була розділена на 2 частини: клієнтська та серверна. Цей підхід є досить непоганим оскільки дає змогу змінювати клієнтську частину, в той час, не змінюючи серверну, а також серверна частина може перевикористовуватися, оскільки, нею може користуватись одночасно десктоп, мобільний, та інші типи застосунків.

Веб додаток моделі соціальної мережі для пошуку людей за інтересами було спроектовано задля створення платформи, де кожен може знаходити цікавих йому людей по інтересам, місцеположенню, мові якою вони спілкуються.

В систему додано обширний перелік інтересів, які можна вибрати. Вони стосуються різних сфер життя, від маркетингу та волонтерства до спектаклів та баскетболу.

Наявна система авторизації, яка надає додаткові можливості зареєстрованому користувачу. Авторизовані користувачі можуть створювати текстові публікації, редагувати їх після моменту публікації, а також видаляти їх. Вони стають доступними для всіх інших користувачів системи, які можуть їх переглядати, а також ставити вподобання. Основною метою додання функціоналу дописів є можливість оцінки людини не лише за стандартною інформацією та фотографією, а й враховуючи її думки та посили, що передаються в тому, що людина публікує.

Також у додатку присутня можливість зміни своїх особистих даних: пароль чи пошту, а також, фото профілю, рід заняття, інтереси, місцеположення, що дозволяє гнучко змінювати свої налаштування.

Реалізована можливість обміну повідомленнями між авторизованими юзерами у формі діалогів. Це дозволяє краще пізнавати один одного не використовуючи жодних інших засобів.

Оскільки одним з головних принципів соціальних мереж є відслідковування інших людей, то в розробленій платформі можна слідкувати за людьми. Цей неважкий крок нотифікуватиме користувачу коли ті, за ким він слідкує публікують новий матеріал додання конкретних постів на його домашню сторінку, що представляє собою набір постів сортований за датой створення.

1.2 Огляд вже готових систем

Було прийняте рішення переглянути найпопулярніші соціальні мережі та системи для знайомств задля виокремлення найкращих якостей, які могли бути використані в розробці продукту, а також недоліків, які можна виправити.

1.2.1 Facebook

Найбільш популярною світовою мережею сьогодення є Facebook. Марк Цукерберг запустив його зі свого гуртожитку в Гарварді 4 лютого 2004 року. Малоймовірно, що Цукерберг знав, що у віці 19 років він назавжди змінив світ та Інтернет.

Більшість людей використовують його для спілкування з друзями, родиною, шанувальниками, знайомими, по робочим справам, а також, для знаходження друзів. На жаль, для інших платформ Facebook просто простіший у використанні для більшості людей, що відіграло значну роль у його становленні.

Основним функціоналом даного веб-сайту є створення постів, можливість додавати інших користувачів до категорії друзів, можливість реагувати на пости, поширювати їх, завантажувати фотографії та вступати в групи.

Головною сторінкою даного продукту є домашня сторінка, де користувач може побачити стрічку пошуку, свою стрічку новин, створити допис, створити історію, що представляю собою різновид допису, що буде

видимим для інших користувачів не завжди, а лише на деякий наперед визначений період часу. Також, можна побачити список своїх контактів та написати повідомлення. Весь головний функціонал системи доступний з цього місця. Основні елементи показано на Рис. 1.1.

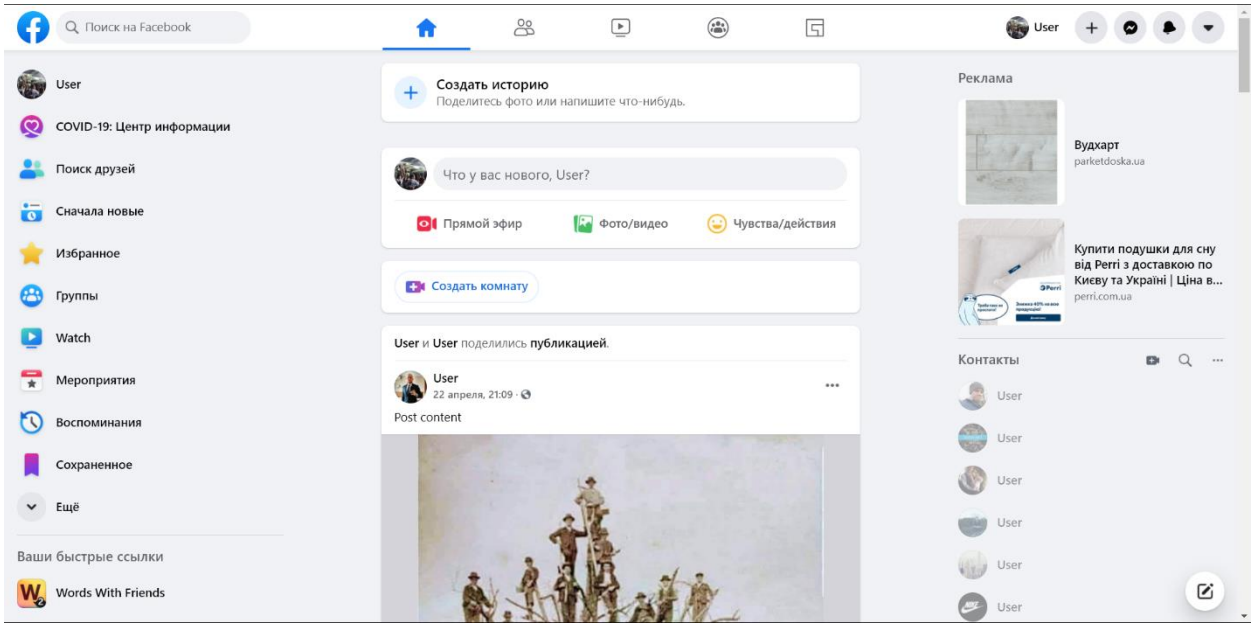


Рисунок 1.1 Домашня сторінка веб-сайту

Головна функція сервісу – поширення публікацій відбувається за допомогою зручного віконця, в якому користувач вводить необхідний текст, а також, має можливість долучити медіа файлами. Натискання кнопки «Опублікувати» зробить публікацію видимою для всіх користувачів мережі.

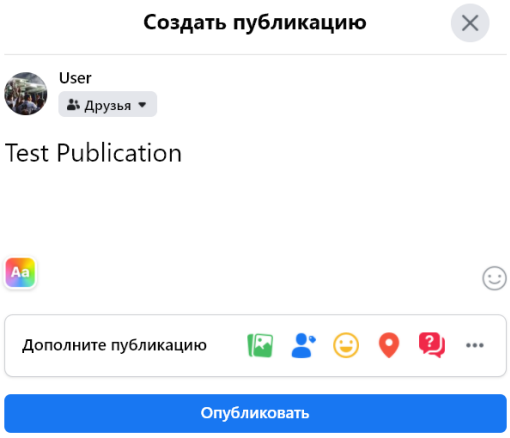


Рисунок 1.2 Вікно для створення публікацій

Задля створення комунікацій між людьми на платформі використовується принцип «Друзі», тобто необхідно надіслати запит на додання в друзі, і якщо він був прийнятий, то два користувачі тепер друзі, і їх публікації будуть з’являтись в стрічках новин один одного. Керування цього відбувається на окремій сторінці, де показуються вхідні запити на додання та профілі, які можуть бути вам цікавими, та людей яких ви можете знати (див рис 1.2.3). На жаль, немає точної інформації згідно з якими даними відбувається підбір даних користувачів, але під час користування можна помітити, що в основному це персони, з якими у вас є спільні друзі.

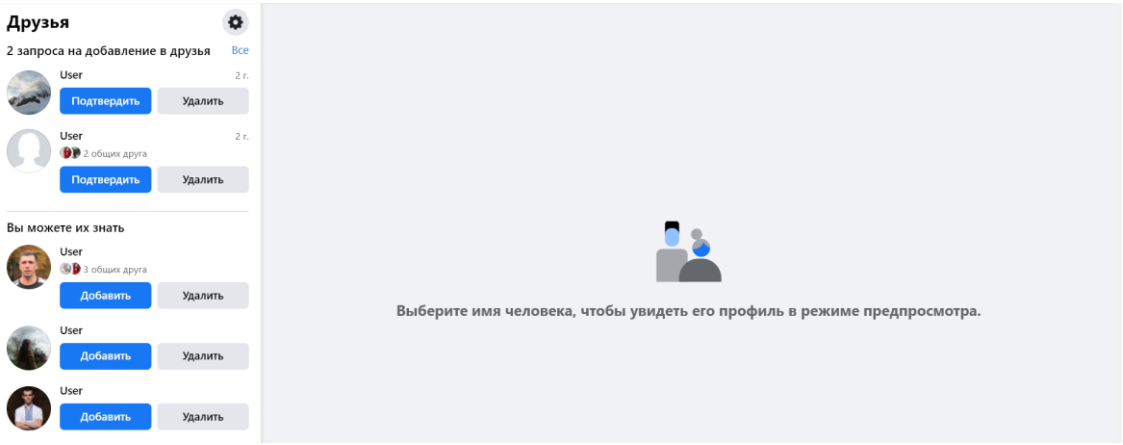


Рисунок 1.3 Сторінка «Друзі»

Серед ключових переваг даного веб-сайту можна виділити наступні пункти:

- Надзвичайно простий та зрозумілий користувацький інтерфейс
- Можливість поширювати свої знання та думки
- Можливість обміну повідомленнями
- Користувачі у всьому світі
- Підтримка аудіо та відео контенту
- Можливість створювати тематичній групи

Недоліками платформи є:

- Дещо застарілий та нудний інтерфейс
- Довге завантаження сторінок

- Не завжди своєчасне адміністрування нових публікацій на предмет порушення правил соціальної мережі
- Незрозуміла та неможлива до налаштування система пошуку людей

1.2.2 Instagram

Соціальною мережею, яка в останній час приваблює все більше й більше користувачів різного віку та статті є Instagram. Спершу це була просто соціальна мережа для поширення своїх фото, а з часом вона розрослася й зараз це повноцінна платформа для спілкування, створення публікацій, ведення бізнесу та відслідковування життя інших людей. Лише трохи більше 9 років з моменту свого створення Instagram став надзвичайно популярним додатком для обміну фотографіями та соціальною мережею. Він конкурує з Facebook за кількістю користувачів, досягаючи мільярдів й це число продовжує постійно зростати.

З понад мільярда зареєстрованих людей, Instagram, який був куплений Facebook в 2012 році, став частиною їх повсякденного життя. Здається, що сьогодні в Instagram є всі - від малого бізнесу до великого, новинних організацій до культурних установ, знаменитостей, фотографів та музикантів до звичайних людей, що прагнуть знайти нові знайомства.

Що робить Instagram таким популярним серед його активних користувачів у всьому світі? Насправді причин багато. Впевнено можна говорити лише про одну – з розвитком мобільних телефонів, їх камери ставали все краще й краще, що дозволяло робити людям не просто знімки для архіву, а дійсно якісні фотографії, якими вони хотіли поділитися, й даний веб-сервіс саме таке платформа.

Головне призначення даного веб-сайту – публікація різних фото та відео матеріалів. Вони не обов’язково мають бути вашими особистими. Багато людей створюють тут свій медіа-ресурс та діляться новинами з якоїсь теми.

До фотографій можна додавати підписи. Підписи - це завжди гарна ідея.

Ви можете використовувати слова, смайлики або хештеги. Ви також можете "згадувати" друзів, розміщуючи "@" перед їх ім'ям користувача й вони будуть усвідомлені першими про вашу нову публікацію. Зручним, також, є те, що ви можете будь-коли редагувати підписи - або видаляти їх, якщо це буде потрібно в майбутньому. Багато людей використовують дану платформу як блог, та підписи виступають найважливішими інструментами в даному випадку.

Instagram дозволяє вам “слідкувати” за іншими людьми. Коли ви стежите за користувачем, його фотографії регулярно з’являються у вашій стрічці. Інші люди бачать тих, за ким слідкуєте ви та тих, хто слідкує за вами.

Ви також можете вподобати фотографії, прокоментувати їх та навіть зберігати в окремі списки створені вами, які є вашою особистісною власністю й ніхто не має до них доступу, й навіть не має гадки про їх існування. На відміну від збережених – вподобані дописи є видимими, тобто будь-яка людина може побачити список людей, які «лайкнули» конкретний пост.

Також є можливість створювати «Історії» - публікації, що будуть видимими лише на протязі доби після їх створення. Далі вони зникають назавжди, хоча ви завжди їх можете переглянути та навіть поділитись в налаштуваннях профілю.

Головна сторінка - ваша основна стрічка, де ви можете прокручувати фотографії та відео, розміщені вашими друзями. На ній з’являються пости та «історії» від людей за якими ви «слідкуєте», а також пропозиції аккаунтів, що можуть бути вам цікавими (див рис 1.4).

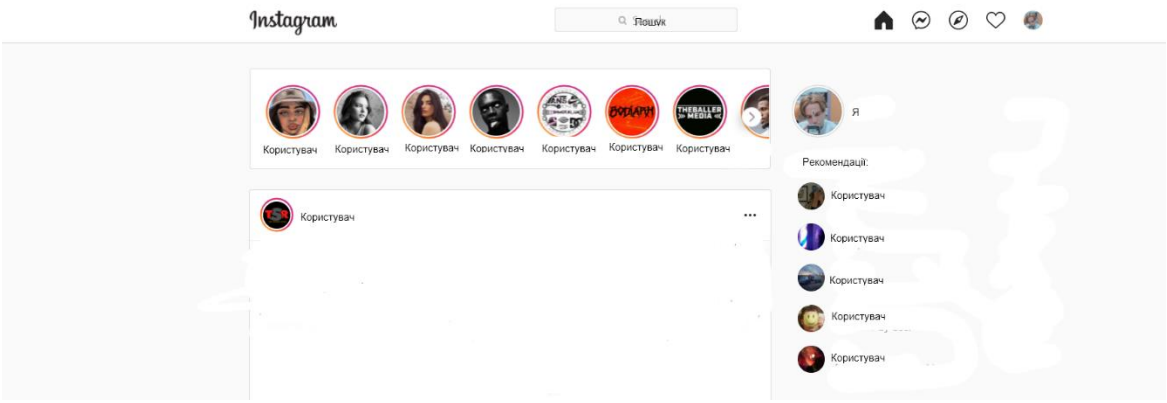


Рис 1.4 Домашня сторінка соціальної мережі Instagram

Оскільки в даній мережі зроблений великий акцент на пропонуванні контенту, який вам був би цікавий, наявна спеціальна сторінка «Досліджування», на якій знаходяться публікації за темами, які вам цікаві. Ця інформація вибирається шляхом аналізу сторінок які ви «відслідковуєте». Ви можете шукати та переглядати вміст з тих облікових записів, за якими ви ще не стежите, але які можуть вас зацікавити.

Також є дуже корисна та основоположна можливість обмінюватися повідомленнями, причому як текстовими, так і відео, фото та аудіо. Є можливість налаштувати профіль так, щоб ваші друзі бачили чи знаходитись ви прямо в цей час на просторах даної соціальної мережі чи ні. Це сигналізує зелений кружечок біля нікнейма користувача(тільки якщо він не вимкнув дану можливість в налаштуваннях.

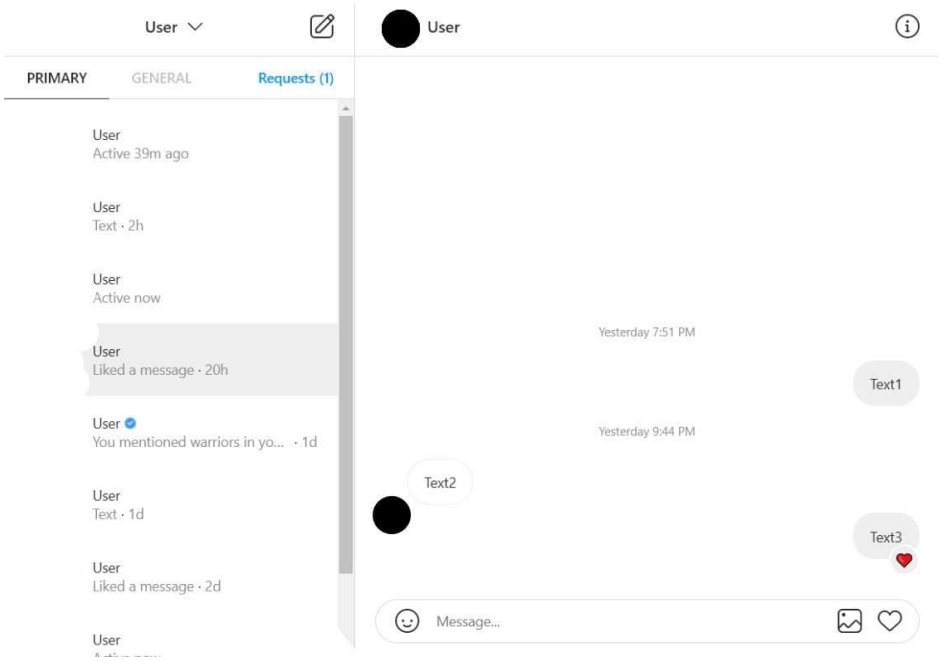


Рис. 1.5 Сторінка особистих повідомлень в Instagram

Однією з найбільш використовуваних елементів інтерфейсу є пошукова стрічка, яка дозволяє виконувати пошук певного користувача серед всім зареєстрованих людей на платформі. Вона не просто знаходить користувача, в якого юзернейм повністю співпадає з введеною послідовністю символів, а й враховуючи алгоритми, показує аккаунти зі схожим унікальним іменем, що часто буває корисним, коли ти точно не запам’ятав коректний юзернейм, але пам’ятаєш якусь його частину.

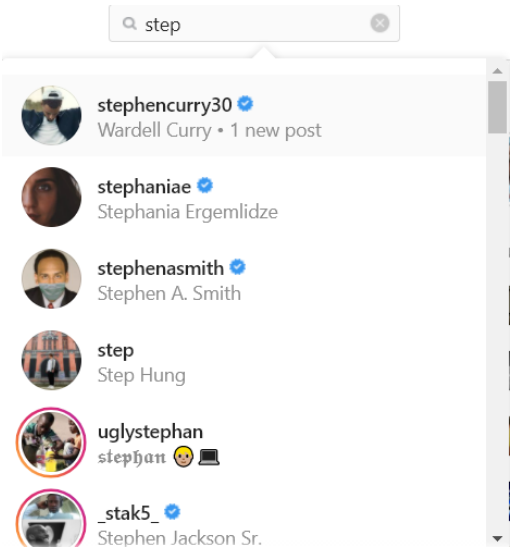


Рис. 1.6 – Пошукова стрічка

До основних переваг даної платформи можна віднести:

- Сучасний та простий в використанні дизайн

- Обмежена кількість функціоналу, що дозволяє недосвідченому користувачу відразу відчувати себе комфортно
- Вбудований обмін повідомленнями
- Аналіз активності користувача задля найточнішої рекомендації контенту
- Мобільний додаток

Головні недоліки:

- Немає можливості пошуку людей по чомусь окрім імені користувача
- Велика кількість реклами

1.2.3 Tinder

Tinder був представлений для світу в 2012 році і є найпопулярнішим у світі веб-сервісом для знайомств з новими людьми. Він був завантажений понад 340 мільйонів разів, доступний у 190 країнах та понад 40 мовами.

Ця платформ - місце, побудоване на світі можливостей. Можливість комунікацій, що може призвести до більшого. Його можна використовувати, щоб познайомитися з новими людьми, розширити своє коло спілкування, познайомитися з місцевими жителями, коли ви подорожуєте, або просто живете в одному місті, але й не знаєте про існування одне одного.

Tinder надзвичайно простий - щоб дати комусь знати, що ця людина вам сподобалась, необхідно просто використати функцію «Свайп вправо», і якщо ви сподобається цій людині також, то це співпадіння і у вас є можливість продовжувати ваше спілкування прямо у цій соціальній мережі.

Щоб використовувати Tinder, потрібно створити профіль із зазначенням вашого поточного місцезнаходження, статі, віку та гендерних уподобань, та деяких інших не обов’язкових параметрів. Наприклад опис – текстове поле, де ви можете написати інформацію про себе або просто щось, що приверне увагу незнайомців до вас, а також перелік ваших інтересів.

Список всіх можливих інтересів є досить численним, що дозволяє навіть людям з незвичними хобі відобразити своє життя в них. Наявна спроможність вибрати не більше п’яти й не менше трьох різних захоплень.

Є можливість завантажити до 9 фотографій, оскільки зазвичай користувачі більш активні до тих аккаунтів, де є фото користувача.

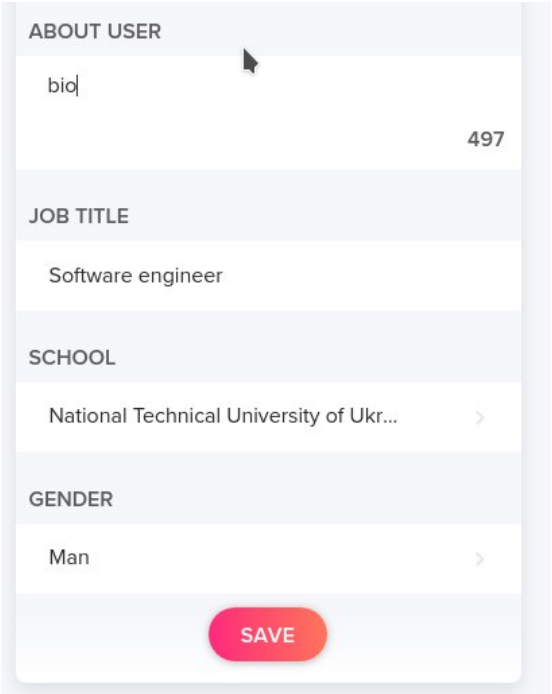


Рис. 1.7 Форма для заповнення детальної інформації користувача

Домашньою сторінкою даної платформи є сторінка рекомендацій, де вам рекомендуються люди поблизу вас, для того щоб зробити пошук більш точним та гнучким можна фільтрувати користувачів по наступним параметрам:

- Стать
- Вік
- Місце проживання

Запропонований профіль містить фото, вік, опис та інтереси цього користувача. Якщо у вас є спільні інтереси, то вони будуть виділені окремим кольором, щоб швидше помітити їх. Цікавим та корисним також є те, що якщо особа була активна в даній соціальній мережі хоча б раз за останню добу, то це буде відображено на його анкеті зеленим кружечком з написом «Нещодавно активний».

Якщо натиснути на іконку інформації, то висвітлиться вся публічна інформація, яку користувач заповнив при реєстрації акаунту. Наприклад: місце проживання, навчальний заклад, професія і тд.

Коли перед вам з’явилася анкета, то під ним з’являється меню взаємодії, що дозволяє виконати наступні дії:

- Повернутися до попереднього користувача. Ця функція є досить корисною коли ви випадково відхилили анкету й хочете змінити своє рішення
- Відкинути профіль
- «Супер Лайк» - надає більше шансів на те, що вибрана людина помітить ваш запит
- Вподобати людину
- «Пришвидшувач» - спеціальна функція, що більше користувачів бачать ваш аккаунт на певний проміжок часу

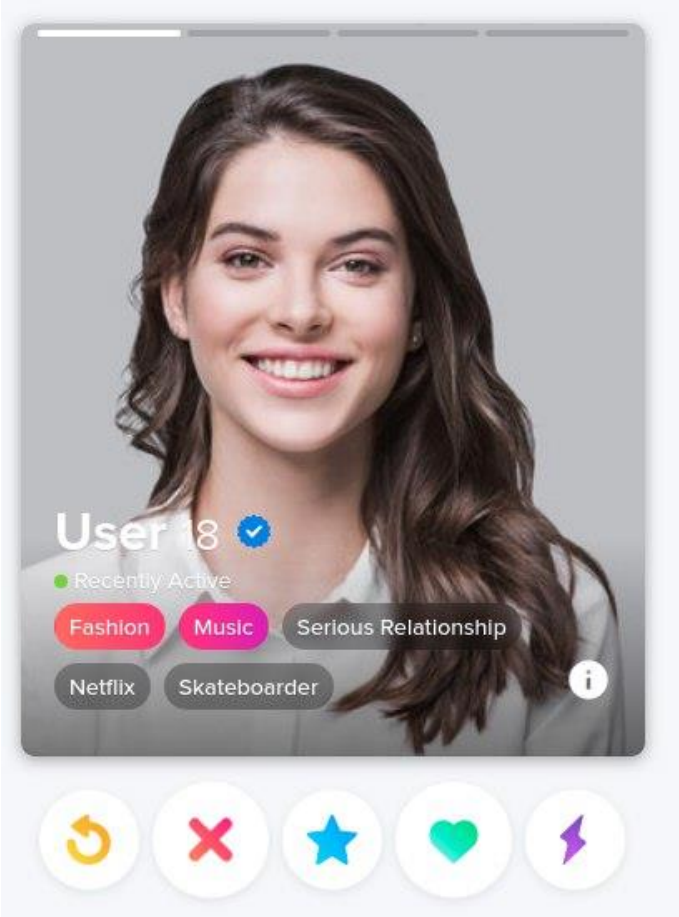


Рис 1.8 – приклад анкети на Tinder

До основних переваг даної платформи можна віднести:

- Сучасний та мінімалістичний дизайн
- Відсутність надлишкових та не потрібних налаштувань – через декілька хвилин після реєстрації можна повністю використовувати веб-сайт
- Можливість спілкування прямо на платформі за допомогою вбудованого обміну повідомленнями
- Можливість закінчити спілкування в будь-який момент видаливши всю історію повідомлень як для себе, так і для протилежної сторони
- Мобільний додаток

Головні недоліки:

- Немає можливості пошуку людей
- Вибрані інтереси не сильно впливають на рекомендації профілей
- Відсутня можливість ведення хоч якогось блогу або просто заміток, щоб люди могли познайомитися з вами не лише фактами про вас, а й вашими думками та способом мислення, що можна було б передати за допомогою дописів
- Більшість функції доступна лише в платному режимі
- Обмеження вибору кількості інтересів лише до п’яти, що не завжди дозволяє користувачу в повній мірі поділитися своїми захопленнями

ВИСНОВОК ДО РОЗДІЛУ 1

В першому розділі були розглянуті найпопулярніші на даний момент веб-сервіси для знаходження знайомств в мережі Інтернет такі, як Facebook, Instagram та Tinder. Кожен з них є особливим, надає унікальний перелік можливостей для користувача при пошуку людей. Проаналізовано сильні та слабкі сторони кожної реалізації.

В результаті підготовки було виділено набір вимог та характеристик, що мають бути притаманні успішному та вдалому програмному забезпеченню:

- Швидкий та простий в користуванні користувацький інтерфейс
- Естетично привабливий зовнішній вигляд веб-ресурсу
- Можливість налаштування явних фільтрів для пошуку людей
- Можливість ведення блогу та створення дописів
- Підтримка вбудованих повідомлень
- Широкий вибір інтересів
- Підтримка адаптивних рекомендації – фільтри формуються на основі даних про користувача та його активності в створюваній соціальній мережі
- Можливість «слідкування» за цікавими профілями

Дотримавшись всіх наданих вище вимог та характеристик можна реалізувати програмне забезпечення, що буде виконувати функції ресурсу для пошуку знайомств та в той же час бути простим та зручним в користуванні для тих, хто ним буде користуватися

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ

2.1 Підготовка до вибору технологій

Для успішного створення програмного продукту передусім необхідно визначитися з засобами, які будуть використовуватися для його імлепентації. Найчастіше для веб-проектів використовується клієнт-серверна архітектура. Це такий різновид архітектури комп'ютерної мережі, коли багато клієнтів відправляють запит та отримують відповідь від централізованого сервера, причому сервер не обов'язково має бути один. Кожен з клієнтів забезпечує інтерфейс, що дозволяє кінцевому користувачу запитувати та відображати інформацію, що знаходиться на серверній стороні.

Зазвичай сервери не можуть самостійно ініціювати відправлення результату клієнту, їм необхідно отримати запит, обробити його й лише після цих кроків, надіслати відповідь, тому початковий стан серверу – очікування на вхідні повідомлення.

Клієнти майже завжди розташовані на персональних комп'ютерах або інших не відносно не дуже продуктивних девайсах. В той час, серверам необхідно використовувати більш потужні машини для виконання всіх обробок.

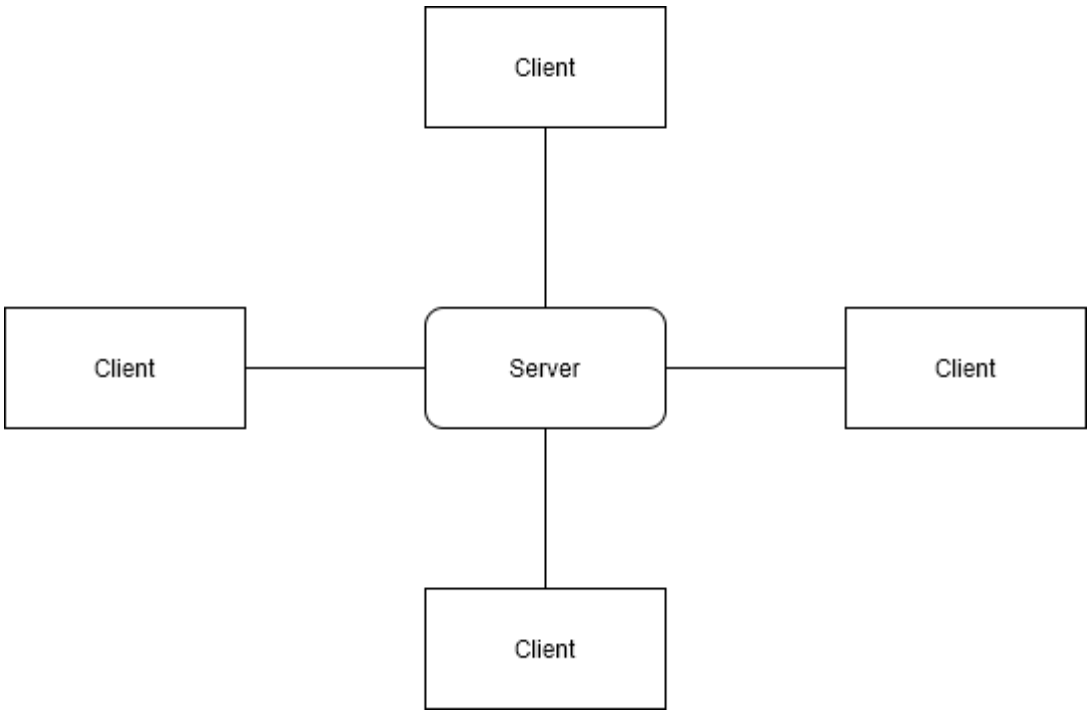


Рисунок 2.1 Структура клієнт-серверної архітектури

Клієнт – це така частина програмного продукту, що відповідає за комунікацію з серверною стороною, а також вміє якимось чином прийняти якусь інформацію від користувача, передати їх на сервер, отримати та відобразити дані в зрозумілому та доступному вигляді.

Сервер – приймає запити від клієнтського коду, обробляє їх, виконує різні операції оптимізації, та надсилає результати назад. Зазвичай, можуть використовуватися також бази даних для зберігання даних.

База даних - це логічно упорядкований набір даних, який, як правило, збирається та отримується в електронному вигляді за допомогою комп’ютерної системи. Там, де бази даних складніші, часто використовують систематизовану архітектуру та методи моделювання.

Система управління базами даних (СУБД) - це програма, яка збирає та аналізує дані, взаємодіючи з кінцевими користувачами, програмами та самою базою даних. СУБД також включає основні інструменти управління базою даних. Вираз "база даних" іноді використовується для позначення всіх систем управління базами даних (СУБД).

Для розробки повноцінного продукту побудованого на клієнт-серверній архітектурі необхідно визначити інструменти та технології, що будуть

використовуватися для кожної частини застосунка. Зокрема, необхідно визначитися з мовами програмування та фреймворками для клієнтської та серверної частин, а також СУБД.

Будемо розглядати найпопулярніших представників кожної з областей:

- Клієнт: React, Angular, Vue
- Сервер: Java, Python, Node.js
- СУБД:

Після розгляду кожної з обраних технологій буде оцінено позитивні та негативні сторони та зроблено висновок про те, яким буде фінальний стек технологій для розробки.

2.2.1 Вибір мови для серверної частини

2.2.1.1 Java

Java є порівняно сучасною мовою програмування, але вона не зовсім нова. Ця універсальна мова програмування святкує своє 26-річчя у 2021 році. Протягом декількох років Java є однією з трьох найпопулярніших мов програмування у світі, і це є повністю заслуженим фактом, що лише засвідчує визнання міжнародним товариством.

Сьогодні Java може похвалитися активною спільнотою і широко використовується як мова для серверного боку розробки для численних проектів, включаючи проекти з що працюють з надзвичайно великою кількістю даних, машинного навчання, усілякі веб-проекти та застосунки для мобільних платформ, насамперед, Android.

Станом на 2021 рік ця мова програмування залишається найчастіше згадуваною в мережі Інтернет (див рис 2.3), що є ще однією ознакою популярності та всеосяжності даної технології.

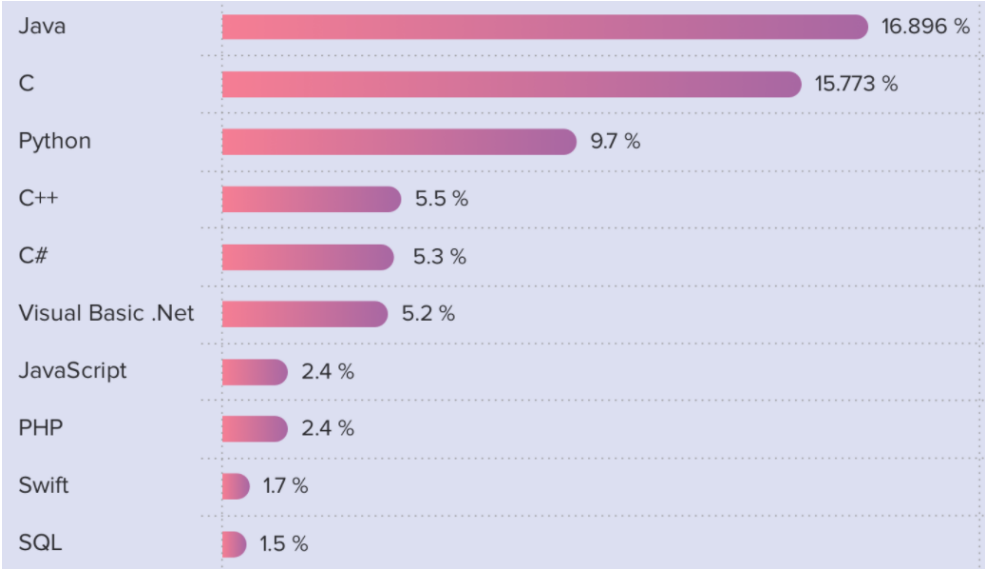


Рис 3.2 Статистика дослідження TIOBE

Одним з головних аспектів, який робить її однією з найбільш привабливих мов для вивчення та подальшого використання, полягає в тому, що вона, чи не найкраще реалізує підходи об'єктно-орієнтованого програмування (ООП). ООП пропонує чітку модульну структуру, що полегшує вирішення складних комплексних проблем. Її модульна структура допомагає програмістам писати універсальний код, що може бути перевикористаний в багатьох сценаріях, що дозволяє вдало уникати повторюваності коду, що з збільшенням кількості розробників та кодової бази ускладнює процес розробки.

У програмуванні на Java для створення об'єктів використовуються класи, що визначають атрибути даних та поведінку (що визначаються методами, закодованими в класі). Крім того, вона дозволяє в повній мірі використовувати такі методики, як абстракція, інкапсуляція, поліморфізм та успадкування, а також найкращі практики та вбудовані пакети, що спрощує написання коду. Оскільки Java-об'єкти не потребують зовнішніх посилань, код написаний на цій є надзвичайно надійним. З неспроста при написанні продуктів, де основний ухил йде на безпеку та захищеність, наприклад, у фінансовій сфері, першим вибором майже завжди постає Java.

Java має англомовний синтаксис, що робить її ідеальною мовою для початківців, які можуть вивчати Java у два етапи - починаючи з основ, а потім переходячи до більш важких просунутих тем. Якщо ви володієте основними знаннями C та C ++, вам не доведеться довго опановувати дану мову.

Хоча Java містить близько п'ятдесяти ключових слів, її інтерфейс прикладного програмування (API) є і розширеним, і багатим - він рясніє численними методами, які можна безпосередньо використовувати в будь-якому коді. Java API включає методи, які можуть задовольнити будь-які цілі, включаючи мережеві операції, підключення до баз даних, синтаксичний аналіз XML, обробку вводу-виводу тощо.

Дана мова програмування є мовою з відкритим кодом, а це означає, що вона абсолютно безкоштовна. Ще однією її чудовою якістю є те, що вона добре задокументована. У її документації є детальне керівництво, яке пояснить проблеми та наштовхне на способи їх вирішення, з якими ви можете зіткнутися під час написання коду на Java.

Це одні з найважливіших причин, чому Java користується популярністю серед розробників, професіоналів та гігантів галузі. Середовище виконання Java (JRE) сумісне майже з усіма пристроями та платформами. Не дивно, чому Java - це всюдисуща сила в галузі саме зараз!

На сьогодні написання майже всіх великих проектів пов’язане з використанням фреймворків. Фреймворк не є абсолютно необхідним: це просто один із доступних інструментів, який допоможе вам краще та швидше писати код. Швидше, оскільки це дозволяє розробникам економити час, повторно використовуючи загальні модулі, щоб зосередитись на інших сферах. Тому, необхідно розглянути один з фреймворків даної мови.

Для Java, одним з найбільш використовуваним та популярним фреймворком являється фреймворк Spring.

Spring - це фреймворк Java з відкритим кодом. Він почав своє існування в 2002 завдяки роботі Рода Джонсона. Був розроблений на принципах введення залежностей та інверсії контролю.

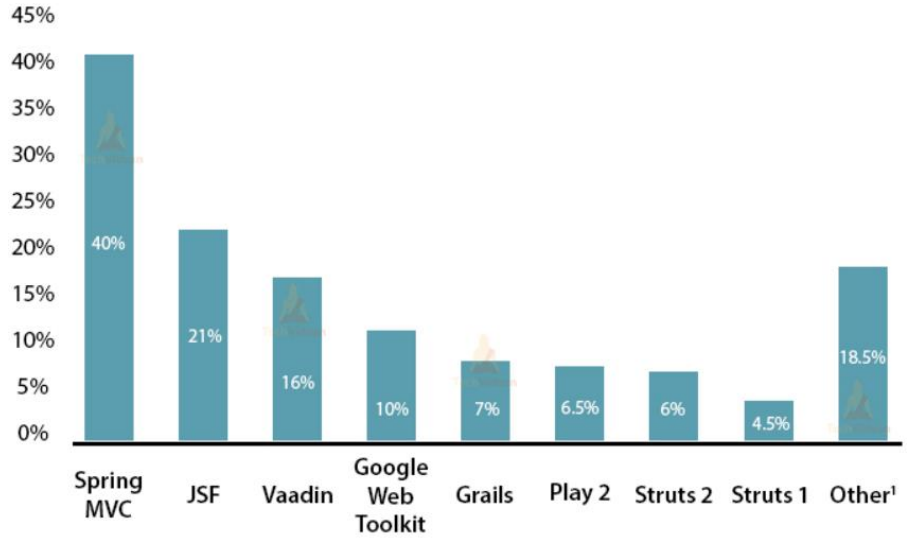


Рисунок 2.4 Рейтинг фреймворків на Java

Як кажуть самі розробники: Spring робить Java простим, сучасним, продуктивним, реактивним. Він відомий своєю інжекцією залежностей та аспектно-орієнтованими особливостями програмування. Насправді це контейнер фреймворків, які дозволяють виконувати завдання будь-якої складності - від роботи з базами даних до процедур тестування.

Він включає в себе величезний вибір модулів, що забезпечують управління транзакціями, автентифікацію, обміном повідомленнями, тестуванням, прив'язкою даних, перетворенням типів, перевірка та обробкою винятків, управлінням ресурсів та подіями, інтернаціоналізацією тощо та різними іншими застосунками, що часто використовуються при розробці.

Переважна більшість компаній віддають перевагу саме цьому фреймворку для побудови корпоративних проетів на Java і це має своє пояснення.

Головною метою фреймворку Spring Boot є скорочення загального часу розробки та підвищення ефективності завдяки встановленню типових налаштувань для модульних та інтеграційних тестів. Якщо ви хочете швидко розпочати роботу, ви можете легко прийняти всі за замовчуванням і повністю уникнути конфігурації XML.

Застосування фреймворку Java Spring:

- Розробка веб-додатків.
- Створення програм будь-якої складності на Java
- Ентерпрайз Java

Розібравши дану мову мову програмування та основну технологію для написання можемо зробити деякі висновки.

До переваг Java можна віднести:

1) Синтаксис подібний до C та C++
2) Використання ООП, що допомагає писати більш структурований код
3) Надійність
4) Велика кількість сторонніх модулів
5) Детальна та чітка документація
6) Перевикористання коду
7) Багатопоточність - програма може виконувати багато завдань одночасно.

Серед недоліків виділимо найважливіші:

1) Велика кількість рядків коду
2) Використовуючи довгі, складні фрагменти, що часто притаманне цій мові, код стає не надто читабельним і складним для подальшої підтримки
3) Займає багато пам’яті і значно повільніше, ніж ті ж самі компільовані мови, такі як C або C ++.
4) Немає фреймворків для використання на клієнтській стороні, що змушує використовувати ще одну мову для написання повноцінного програмного забезпечення веб нахилу

2.2.1.2 Python

Починаючи з 1991 року, мова програмування Python вважалася заповнювачем прогалин, способом писати скрипти, які «автоматизують речі», або швидко створювати прототипи програм, які будуть реалізовані іншими мовами.

Станом на минулий рік - Python був на шляху до того, щоб стати найпопулярнішою мовою кодування у світі. Оскільки такі мови як Fortran та Lisp стрімко занепали, а мови, такі як C та C ++, залишаються стабільними, але без стрімкого росту, такі мови, як Python та JavaScript, різко піднімаються та приваблюють все нову й нову аудиторію.

Python - це мова загального призначення - це означає, що вона може використовуватися для різноманітних типів програмування та розробки програмного забезпечення, крім веб-розробки.

Python можна використовувати для таких речей, як:

- Розробка веб-сторінок та програм для мобільних пристроїв
- Розробка веб-серверів
- Розробка десктопних додатків
- Обробка великих даних
- Виконання «важких» математичних обчислень
- Написання системних скриптів (інструкцій, що вказують комп’ютерній системі щось робити, часто автоматично)

Сама мова має досить невелику кількість вбудованих функцій та різних конструкцій, тому написання перших програм не займе дуже багато часу та зусиль. Python - відмінна мова для викладання програмування завдяки своїй легкості, яка дозволяє новачкам легко її розуміти та швидко починати використовувати її на практиці. Як результат, розробники витрачають менше часу на турботи пов’язані з складністю чи розшифровкою коду, написаного іншими, і більше часу на роздуми над проблемою, яку вони намагаються вирішити.

Python поставляється з надійною стандартною бібліотекою прямо з коробки (без каркасів або інших доповнень), що робить більш ефективним процес написання програм. Стандартна бібліотека також постачає розробникам заздалегідь встановлені «модулі» (файли, що складаються з коду Python), які дозволяють розробникам пропускати процес безлічі корисних та часто вживних функцій самостійно - заощаджуючи час та роблячи код більш організованим.

А ще має місце той факт, що код Python був створений з особливим акцентом на читабельність коду, тому мова зосереджується на англійських ключових словах замість символів та пунктуації. Це означає, що розробнику простіше читати, підтримувати та оновлювати код.

Найпростіший варіант використання Python - це мова скриптів та автоматизації. Також використовується для автоматизації взаємодії з веб-браузерами та графічними інтерфейсами додатків.

Власні бібліотеки Python та сторонні веб-фреймворки забезпечують швидкі та зручні способи створення всього: від простих REST API у декілька рядків коду до повномасштабних веб-сайтів. Останні версії Python мають потужну підтримку асинхронних операцій, дозволяючи сайтам обробляти десятки тисяч запитів в секунду за допомогою спеціальних бібліотек.

Аналіз даних став однією з найбільш швидкозростаючих галузей IT та одним із основних випадків використання Python. Переважна більшість бібліотек, що використовуються для обробки великої кількості інформації та машинного навчання, мають інтерфейси Python, що робить мову найпопулярнішим командним інтерфейсом високого рівня для бібліотек машинного навчання та інших числових алгоритмів.

Як приклад для розгляду фреймворку на Python було обрано Django, оскільки він є найбільш популярним.

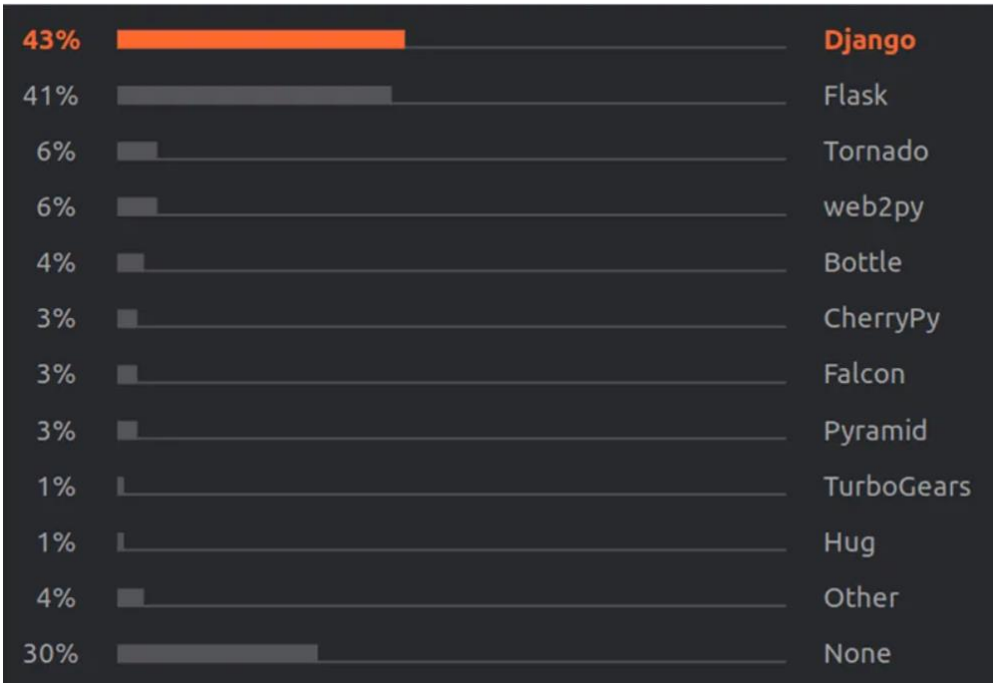


Рисунок 2.5 Рейтинг популярності фреймворків Python

Django - це веб-фреймворк з відкритим вихідним кодом, який використовується для швидкого запуску та розвитку прагматичних, здатних до змін безпечних веб-сайтів.

Основна мета фреймворку Django - дозволити розробникам зосередитися на створенні нових компонентів програми, а не витрачати час на вже розроблені. Дане рішення є повнофункціональним без використання зовнішніх бібліотек та модулів, на відміну від багатьох інших фреймворків. Він вирішує багато проблем, що виникають при веб-розробці, дозволяє користувачам зосередитися саме на розробці компонентів, необхідних для їх кінцевих продуктів.

За допомогою Django ви можете розробляти проекти будь-якого розміру та навантаженності, будь то простий веб-сайт або веб-програма з великим навантаженням. Він повністю масштабований, тому ви можете створювати програми, які обробляють інтенсивний трафік та великий обсяг інформації. Також важливою особливістю є крос-платформенність, що означає, що ваш проект може базуватися на Mac, Linux або PC. Він працює з більшістю основних баз даних і дозволяє використовувати базу даних, яка більше підходить для конкретного проекту, або навіть кілька баз даних одночасно.

Django має власну систему імен для всіх функцій та компонентів (наприклад, відповіді HTTP називаються "поданнями"). Він також має адміністративну панель, з якою легше працювати, ніж у Lavarel або Yii, та інші технічні особливості, зокрема:

- Простий синтаксис;
- Власний веб-сервер;
- Архітектура ядра MVC;
- ORM (Object Relational Mapper);
- Бібліотеки HTTP;
- Підтримка проміжного програмного забезпечення;
- Тестовий фреймворк Python.

Таким чином можемо сформувати переваги та недоліки використання Python. До переваг можемо віднести:

1) Простий синтаксис;
2) Може підтримувати безліч стилів програмування
3) Використання в машинному навчанні та штучному інтелекті
4) Легкість в вивченні та використанні
5) Великий вибір готових рішень у вигляді пакетів
6) Можливість створювати прототипи та перевіряти ідеї набагато швидше ніж в інших мовах

Недоліки цієї мови:

1) Інтерпретована мова, тому ви можете виявити, що вона працює повільніше, ніж деякі інші популярні мови.
2) Проблеми з впровадженням паралельності
3) Не ідеально підходить для завдань, що використовують велику кількість пам’яті
4) Немає популярних рішень для клієнтської сторони

2.2.1.3 JavaScript

JavaScript спочатку був створений, щоб «оживити веб-сторінки». Програми на цій мові називаються скриптами. Їх можна записати прямо в HTML розмітку веб-сторінки та запускати автоматично під час її завантаження. Скрипти передаються та виконуються як звичайний текст. Їм не потрібна спеціальна підготовка чи компіляція для запуску. У цьому аспекті JavaScript сильно відрізняється від тієї самої Java.

Найбільш ранні втілення JavaScript були розроблені наприкінці 1990-х для веб-браузера Netscape Navigator. У той час веб-сторінки були статичними, пропонуючи мало взаємодії з користувачами, окрім клацання посилань та завантаження нових сторінок. Вперше JavaScript активував анімацію, адаптивний контент та валідацію форми на сторінці без запиту на сервер.

Сьогодні код написаний на цій мові може виконуватися не лише у браузері, але і на сервері, або фактично на будь-якому пристрої, що має спеціальну програму, яка називається движок JavaScript. У всіх браузерах є вбудований движок, який іноді називають «віртуальною машиною JavaScript».

Популярним прикладом є V8 від Google. Він використовується, наприклад, у Chrome та Node.js. Ось дуже спрощений вигляд того, як він працює:

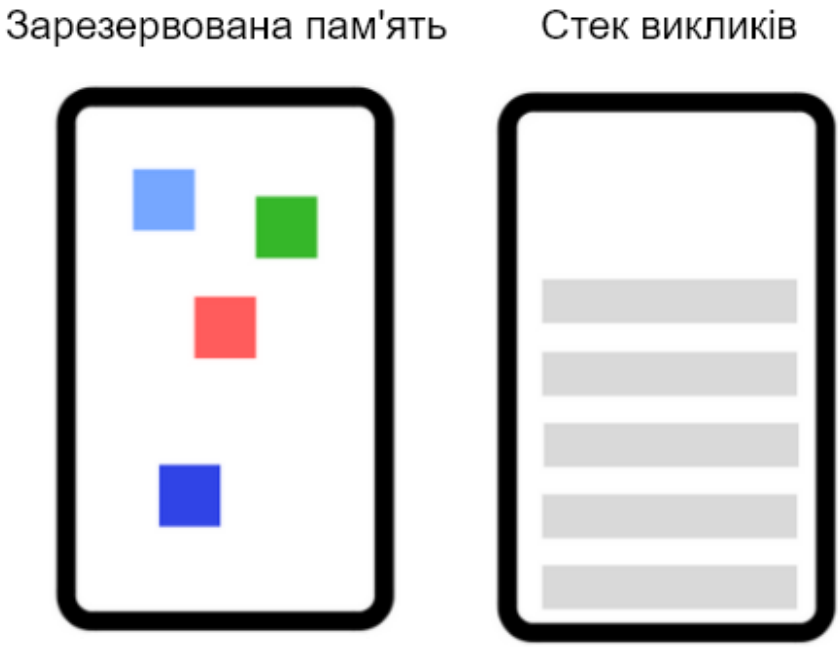


Рисунок 2.6 Структура движка V8

Даний движок складається з двох основних компонентів:

- Зарезервованої пам'яті - тут відбувається виділення пам'яті
- Стек викликів - тут знаходяться виклики під час виконання вашого коду

JavaScript - це однопотокова мова програмування, що означає, що вона має єдиний стек викликів. Тому він може виконувати лишу одну річ за один раз.

Стек викликів - це структура даних, яка в основному фіксує, де в програмі ми знаходимось. Якщо ми вступаємо в функцію, ми ставимо її на вершину стека. Якщо ми повертаємося з функції, ми викидаємо її з вверху стека. Це все, що може зробити стек.

Сучасний JavaScript - це "безпечна" мова програмування. Він не забезпечує низькорівневий доступ до пам'яті або процесора, оскільки першочергово був створений для браузерів, які цього не потребують, і лише згодом став чи не найгнучкішою та мультипарадигменною мовою, яку можна використовувати чи не у всіх галузях створення програмного забезпечення.

Можливості JavaScript сильно залежать від середовища, в якому він працює. Наприклад, Node.js підтримує функції, які дозволяють JavaScript читати / писати довільні файли на фізичний носій, виконувати мережеві запити тощо та взагалі розробляти масштабовані рішення для серверної сторони.

JavaScript на сервері використовується за допомогою платформи Node.js. Це середовище виконання, побудоване на Google Chrome V8. Використання Javascript є гарною новиною, адже тепер розробники можуть використовувати одне і те ж рішення як для написання клієнтської частини, так і для серверних веб-програм, над якими вони працюють

Згідно з опитуванням Stack Overflow у 2019 році, Node.js зараз є найпопулярнішим інструментом у категорії «Фреймворки, бібліотеки та інструменти» з 50% позитивних відповідей.

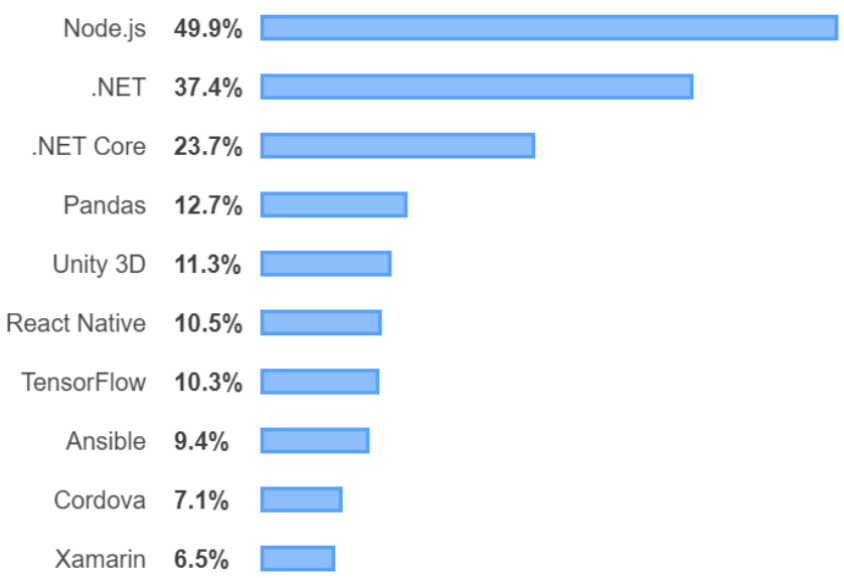


Рисунок 2.7 – Рейтинг фреймворків у 2020 році

Node.js є досить швидким, навіть, без використання, таких популярних, серед серверних мов програмування, потоків. Він може одночасно обробляти багато з'єднань завдяки своїй однопоточній архітектурі, керованій подіями.

Тим часом багато веб-платформ створюють новий потік щоразу, коли надходить запит. Це використовує багато оперативної пам'яті, що призводить до нижчої швидкості. А якщо користувачів, а значить й потоків стає дуже багато, то часто з’являється «потоковий голод». Це такий стан системи, коли більшість часу витрачається не на виконання обчислень, роботи з базою даних, а на постійне переключання між потоками. З іншого боку, Node.js використовує цикл подій та зворотні виклики для операцій вводу-виводу.

Приблизно, цю модель можна порівняти з рестораном. Офіціант приймає ваше замовлення, повертає його шеф-кухареві і продовжує приймати замовлення від інших клієнтів. Тож, він не просто чекає, поки шеф-кухар зварить вам їжу, а продовжує обробляти запити інших клієнтів. Саме так працює Node.js - він може обробляти тисячі з'єднань одночасно, використовуючи ресурси більш ефективно ніж типові синхронні та блокуючі системи.

Node.js став одним із найкращих варіантів для багатьох галузей. Важко встояти перед простотою, яку пропонує Node.js. Однак у кожної технології є деякі свої плюси і мінуси.

Головні переваги:

- Висока продуктивність для програм реального часу
- Легка масштабованість для сучасних додатків
- Написання коду на клієнтській та на серверній стороні використовуючи одну й ту ж мову програмування
- Чи не найбільша спільнота серед усіх мов програмування
- Легка для вивчення мова
- Скорочує час завантаження за допомогою швидкого кешування
- Допомагає у створенні крос-платформних додатків
- Швидка розробка

Основні недоліки:

- Зниження продуктивності під час складних обчислень
- Повна залежність від функцій зворотнього виклику ставить під загрозу якість коду
- Якість багатьох модулів з менеджера пакетів залишає бажати кращого

2.2.2 Вибір технології для користувацької частини

Що стосується розробки клієнтської частни, то є три основні компоненти: HTML, CSS та JavaScript. Кожен з них має вирішальне значення для створення та відображення веб-сторінки такою, якою вона є. HTML - це структура та зміст сайту, CSS (каскадні таблиці стилів) робить її приємною додаючи різні стили. Наприклад, рамки, кольори, відступи – це все робиться за допомогою стилів, і, нарешті, JavaScript - це те, що забезпечує його інтерактивність. Кожна частина працює рука об руку, коли справа стосується створення веб-сайтів.

JavaScript – та мова, яка у браузері може робити все, що стосується маніпулювання веб-сторінками, взаємодії з користувачем та веб-сервером.

Наприклад:

- Додавати на сторінку новий HTML, заміняти наявний, змінювати стилі.

- Реагувати на дії користувача, такі як: рух чи кліки миші, натисканням клавіш і тд.
- Надсилати запити через мережу на віддалені сервери, скачувати та завантажувати файли (так звані технології AJAX та COMET).
- Взаємодіяти з куками.
- Запам’ятовувати будь-які дані на стороні клієнта використовуючи «локальне сховище».

Тому, для розробки цієї частини продукту розглянемо три найпопулярніші рішення – вони всі є бібліотеками чи фремворками Javascript.

2.2.2.1 React.js

React.js - найпопулярніша бібліотека для створення інтерфейсів для веб-додатків. React.js або Reactjs або просто React - це різні способи представлення React.js. Дана бібліотека є програмним забезпеченням з відкритим кодом, яка використовується для побудови користувальницьких інтерфейсів спеціально для односторінкових додатків. Вона використовується для обробки шару відображення для веб- і мобільних додатків. React також дозволяє створювати багаторазові компоненти, які можна згодом перевикористовувати. Вперше React був створений Джорданом Уолком, інженером-програмістом, що працює у Facebook.

React дозволяє розробникам створювати великі веб-програми, які можуть змінювати дані, не перезавантажуючи сторінку. Основна мета React - бути швидким, масштабованим і простим. Він працює лише на рівні користувальницьких інтерфейсів програми. Це відповідає відображенню в шаблоні MVC.

У React замість того, щоб використовувати звичайний JavaScript для створення шаблонів, він використовує JSX. JSX - це простий JavaScript, що дозволяє HTML-теги і використовує цей синтаксис для візуалізації підкомпонентів. Синтаксис HTML обробляється у виклики JavaScript. Таким чином в результаті обробки залишається «чистий» Javascript.

Головною причиною чому саме дана бібліотека є такою популярною є принцип «Virtual DOM».

DOM розшифровується як „Модель об’єкта документа”. Простіше кажучи, це структуроване представлення елементів HTML, які присутні на веб-сторінці або веб-програмі. Він містить кожний елемента HTML, присутній у веб-документі. Це дуже корисно, оскільки дозволяє змінювати вміст будь-якого елемента за допомогою JavaScript.

Оновлення DOM не є важким процесом, для цього необхідно лише знати декілька функції Javascript, таких як:

- getElementById()
- getElementByClass()

Вони дозволяють просто змінити елемент за його ідентифікатором чи классом відповідно. Це працює чудово, але давайте розглянемо випадок, коли цих елементів у нас багато.

Під час написання наведеного вище коду трапляються такі речі:

1) Браузер аналізує HTML, щоб знайти вузол з цим ідентифікатором.
2) Він видаляє дочірній елемент цього конкретного елемента.
3) Оновлює елемент (DOM) із оновленим значенням.
4) Перераховує CSS для батьківського та дочірнього вузлів.
5) Обновляє макет.
6) Обходить все HTML дерево і відображає його на екрані (браузері).

Отож, як ми зараз знаємо, оновлення DOM включає не лише зміну вмісту, але й набагато більше операцій прив’язаних до нього. Також перерахунок CSS і зміна макетів включає складні алгоритми, і вони впливають на продуктивність. Тому, React має інший підхід до вирішення цього питання, оскільки використовує Віртуальний DOM.

У даній технології існує віртуальний DOM, який нагадує полегшену копію фактичного DOM. Отже, для кожного об’єкта, що існує у вихідному DOM, існує об’єкт для цього у віртуальному. Це точно той ж самий DOM, але

без має можливості безпосередньо змінювати макет документа. Маніпулювання DOM відбувається повільно, але маніпулювання віртуальним DOM відбувається швидко, оскільки на екрані нічого не малюється.

Щоразу, коли ми щось змінюємо у нашому файлі JSX, усі об’єкти, що знаходяться у віртуальному DOM, оновлюються. Хоча може здатися, що це неефективно, але вартість не настільки значна, оскільки оновлення віртуального DOM не займає багато часу. Реакт підтримує два віртуальних DOM в кожен момент часу, один містить оновлений Virtual DOM і один, який є просто версією попереднього оновлення цієї оновленої віртуальної DOM. Він порівнює попередньо оновлену версію з оновленою Virtual DOM і з’ясовує, що саме змінилося в DOM. Як тільки React дізнається, що саме змінилося, він оновлює лише ці об'єкти, на реальному DOM. Це значно покращує продуктивність і є основною причиною того, чому сама дана бібліотека є такою швидкою та продуктивною.

Перевагами Реакту є:

1) Простота в освоєнні і використанні
2) Велика громада
3) Може використовуватися для створення мобільних додатків
4) Продуктивність
5) Багаторазові компоненти
6) Додатки легко тестувати

Недоліками можно відзначити:

1) Високий темп розробки, що може привести до помилок
2) Не зовсім повна документація
3) JSX не для всіх є зрозумілим на початку

2.2.2.2 Angular.js

Angular - це частина екосистеми JavaScript і один із найпоширеніших засобів розробки програмного забезпечення сьогодні. Вперше він був представлений Google у 2009 році, і спільнота програмістів з самого його

зародження дуже полюбила його. 30,7 відсотка розробників програмного забезпечення в даний час використовують AngularJS та останню версію Angular 2+ для побудови користувальницьких інтерфейсів, згідно з опитуванням StackOverflow 2019 року. За даними NG-Conf 2019, група розробників Angular зросла на 50% з початку 2019 року порівняно з 2018 роком.

Ключовою перевагою даного фреймворку у 2010 році було те, що він дозволив перетворити документи на основі HTML в інтерактивний текст. HTML, мова веб-розмітки, завжди була статичною до створення AngularJS, що означало, що користувачі не могли регулярно спілкуватися з інтерфейсами на HTML-сторінках. Існувало декілька методів створення динамічних односторінкових додатків (SPA), але вони були занадто складними для простого проектування. Користувачі отримували веб-сторінки зі складними типами та елементами, але завдяки фреймворку AngularJS, який мінімізував зусилля, що потрібні для розробки динамічного контенту, відбувся значний розвиток клієнтської сторони, що передусім, дозволило втілювати в життя ідеї, що зі звичайним HTML були б неможливими.

У вересні 2016 року Google випустив Angular 2. Це була повна переробка фреймворку тією ж командою, що більше відповідає сучасніим вимогам Інтернету. Різниця між версіями була настільки радикальною, що не можна було просто оновитися з однієї на іншу. Міграція програми на Angular 2 вимагала багато модифікацій через досить велику різницю в синтаксисі. Тож у подальших оновленнях команда розробників Angular розробила методи та інструменти міграції, що зробили даний перехід менш болючим та більш простим.

З періодичною швидкістю виходять все нові й нові версії, що покращують використання фреймворку та вирішують проблеми, що виникають у програмістів. Версія 11, останнє оновлення, була представлена в листопаді 2020 року. Випуск приніс багато вдосконалень, що стосуються фреймворку та його платформи.

Найголовнішою є пришвидшений час збірки проекту, що має надзвичайно важливе значення при розробці.

Додатки на Angular побудовані з використанням мови TypeScript, що забезпечує більш високий рівень безпеки, так як підтримує типи (примітиви, інтерфейси і т.д.). Це допомагає відловлювати та усувати помилки на початку написання коду.

На відміну від CoffeeScript або Dart, TypeScript не є окремою мовою. За допомогою TypeScript ви можете легко взяти наявний код ES5 або ES2015 + JS, і він скомпілює його на основі того, що ви налаштовуєте. Він повністю підтримує основні функції ES2015 та ES2016 / ES2017, такі як декоратори або async / await.

Angular в своїй основі використовую таке архітектурне рішення як MVC. Angular не просить розробників розділити додаток на різні компоненти MVC та побудувати код, який може їх об'єднати. Навпаки, він лише просить розділити програму і сам піклується про все інше.

Рисунок 2.9 Модель MVC

Компоненти MVC:

- Модель - це компонент відповідає за дані, а також визначає структуру програми.
- Представлення - це компонент відповідає за взаємодію з користувачем. Це HTML-шаблон, який повертає сервер після обробки запиту. Тобто, код цього компонента визначає зовнішній вигляд програми і способи його використання

- Контролер - цей компонент відповідає за зв'язок між моделью та представленням. Код компонента визначає, як сайт реагує на дії користувача, а також, обробляє вхідні запити. У фреймворку це може полягати у визначенні конкретних URL, на які потрапляє користувач при переході по посиланню або при натисканні кнопки

Angular забезпечує легку розробку, оскільки позбавляє потреби в непотрібному коді. Він має спрощену архітектуру MVC, що робить записування геттерів та сетерів непотрібним. Директивами може керувати інша команда, оскільки вони не є частиною коду програми.

Переваги використання Angular:

- Двостороння прив'язка даних
- Директиви
- Ін’єкція залежності
- Підтримується Google
- Велика громада та екосистема
- Використання TypeScript

Недоліки даного рішення:

- Забагато версій
- Зниження популярності
- Багато шаблонного коду
- Крутіша крива навчання(погано підходить для новачків)
- Не найкраща документація
- Проблеми з міграцією версій

2.2.3 Вибір СУБД

2.2.3.1 MongoDB

MongoDB - це документо-орієнтована база даних NoSQL, яка використовується для зберігання даних великого обсягу. Замість

використання таблиць і рядків, як у традиційних реляційних базах даних, MongoDB використовує колекції та документи. Документи складаються з пар ключ-значення, які є основною одиницею даних у MongoDB. Колекції містять набори документів та функції, що є еквівалентом реляційних таблиць баз даних.

Ключові поняття, що використовуються в данній технології [20]:

- База даних - це контейнер для колекцій, як у реляційних базах данних, де він є контейнером для таблиць. Кожна база даних отримує власний набір файлів у файловій системі. Сервер MongoDB може зберігати кілька баз даних.
- Колекція - це групування документів MongoDB. Колекція - це еквівалент таблиці, яка створюється в будь-яких інших СУБД, таких як Oracle або MS SQL. Колекція існує в межах однієї бази даних. Як видно з вступу, колекції не застосовують жодної структури.
- Документ - Запис у колекції MongoDB в основному називається документом. Документ, у свою чергу, буде складатися з назви поля та значень.
- JSON - це зручний для читання формат простого тексту для вираження структурованих даних. На даний момент JSON підтримується багатьма мовами програмування і став стандартним форматом для представлення даних.
- Курсор - це вказівник на набір результатів запиту. Клієнти можуть перебирати курсор для отримання результатів.
- Поле - пара ім'я-значення в документі. Документ має нуль або більше полів. Поля аналогічні стовпцям у реляційних базах даних.
- _id - це обов’язкове поле для кожного документа MongoDB, воно представляє собою унікальне значення в документі MongoDB. Поле _id схоже на первинний ключ документа

```
{
    "_id": ObjectId("5c54e647499d4242fc007c12"),
    "name": "Barbara Palmer",
    "contact": "555-555-7654",
    "dob": "19-09-1980",
    "address": [
        {
            "_id": ObjectId("5c55ad84499d4230cc004934"),
            "address 1": "135 Sycamore Dr.",
            "address 2": "",
            "postal_code": 55345,
            "city": "Tulsa ",
            "state": "Oklahoma"
        },
        {
            "_id": ObjectId("5c55b102499d4230cc004935"),
            "address 1": "88 Mongoose Lane",
            "address 2": "Apt 9",
            "postal_code": 55335,
            "city": "albuquerque",
            "state": "New Mexico"
        }
    ]
}
```


Рисунок 2.10 Структура документа в MongoDB

Дана СУБД підтримує багато різних типів даних, найчастіше використовуваними є наступні:

- Рядок
- Ціле число
- Логічне
- Масив
- Мітки часу
- Об’єкт
- Дата
- Ідентифікатор об’єкту

У цій технології кожна база даних містить колекції, які, в свою чергу, містять документи. Кожен документ може бути різним із різною кількістю полів. Розмір та зміст кожного документа можуть відрізнятися один від

одного. Кожна конкретна структура документа більше відповідає тому, як розробники будують свої класи та об'єкти у відповідних мовах програмування. Рядки (або документи, як їх називають у MongoDB), не повинні мати схему, визначену заздалегідь. Натомість поля можна додавати чи видаляти на льоту. Модель організації даних, доступна в MongoDB, дозволяє легше представляти ієрархічні відносини, зберігати масиви та інші більш складні структури ніж в реляційних аналогах [19].

Нижче наведено кілька причин використання MongoDB:

- Орієнтований на документ
- Спеціальні запити - за полем, діапазоном та регулярним виразом
- Індексація
- Реплікація
- Простота в використанні
- Швидке налаштування
- Гнучкість

Основні недоліки:

- Проблеми при виконанні транзакцій
- З’єднання документів не завжди є простою операцією
- Обмежений обсяг даних для документа(16 МБ для документа)
- Високе використання пам'яті
- Можливе копіювання даних

2.2.3.2 PostgreSQL

PostgreSQL - це потужна об'єктно-реляційна система баз даних з відкритим кодом, яка застосовує мову SQL. Дана розробка відома своєю надійністю, швидкістю та продуктивністю. Вона дозволяє безпечно зберігати великі та складні дані. Це допомагає розробникам створювати найскладніші програми, запускати адміністративні завдання та створювати цілісні середовища. PostgreSQL стає використовуваною базою даних для все більшої кількості підприємств.

Починаючи з 1986 року, коли PostgreSQL був створений, у нього було багато прихильників та критиків. Щоб зберегти назву «найдосконалішої бази даних», Postgres регулярно оновлює себе новими функціями. В даний час дана технологія займає 4 місце за популярністю серед сотень баз даних у всьому світі згідно з рейтингом DB-Engines.

Більшість реляційних систем управління базами даних (СУБД) використовують SQL як мову програмування для обробки даних, що зберігаються в табличній формі.

Таблиця SQL складається з рядків і стовпців, які позначені міткою. Реляційна база даних складається з таблиць, які зв'язані між собою спільними стовпцями, що дозволяє пов’язувати таблиці між собою і використовувати властивості реляційних баз.

Наприклад, з’єднавши таблиці, можемо налаштувати систему так, щоб видалення запису з однієї таблиці вилучало відповідний рядок і з іншої також і тд.

SQL пропонує безліч функцій та стратегій для ефективного та надійного управління даними. Доступно кілька систем управління реляційними базами даних (наприклад, MySQL, PostgreSQL, SQL Server). Кожна з них використовує дещо змінений синтаксис даної мови.

Для початку роботи необхідно створити базу данних, підключитися до неї, і почати зі створення таблиць. Кожна таблиця описується за допомогою структури, яка описує які будуть наявні поля, їх типи, обмеження та залежності [24].

PostgreSQL підтримує такі типи даних:

- Різні типи тексту
- Числові типи
- Дати та час
- XML
- JSON
- Логічний тип

- Біти
- Двійкові дані
- Масиви
- UUID
- JSON
- Спеціальні типи даних для зберігання мережевої адреси та геометричних даних

Обмеження SQL - це правила, яким повинні слідувати дані таблиці. Тип даних, які можна ввести в таблицю, може бути обмеженим. Це означає, що дані в таблиці точні та надійні. Дія додання нових даних припиняється, якщо існує конфлікт між обмеженням та даними [18].

Обмеження можуть застосовуватися на рівні стовпців або таблиць. Обмеження на рівні стовпців застосовуються до одного стовпця, тоді як обмеження на рівні таблиці застосовуються до всієї таблиці.

У SQL часто використовуються такі обмеження:

- «NOT NULL» - гарантує, що стовпець не може мати значення «NULL»
- «UNIQUE» - гарантує, що всі значення в стовпці різні
- «PRIMARY KEY» - поєднання «NOT NULL» та «UNIQUE». Унікально ідентифікує кожен рядок у таблиці.
- «FOREIGN KEY» - допомагає запобігає діям, які руйнують зв'язки між таблицями
- «CHECK» - переконується, що нові значення в стовпці відповідають певній умові
- «DEFAULT» - встановлює значення за замовчуванням для стовпця, якщо значення не вказано
- «CREATE INDEX» - використовується для створення та отримання даних з бази даних дуже швидко використовуючи різні алгоритми, наприклад, бінарного дерева.

Після створення таблиці доступ та взаємодія з нею відбувається через SQL-запити. З'являється можливість отримувати, додавати, змінювати та видаляти рядки з таблиці.

```
sonery=# SELECT * FROM orders;

order_id | product_id | order_qty |    date
---------+------------+-----------+------------
     101 |       1001 |         2 | 2021-01-04
     102 |       1423 |         1 | 2021-01-04
     103 |       1260 |         5 | 2021-01-13
     104 |       1590 |         5 | 2021-05-13

(4 rows)
```

Рисунок 2.11 SQL-запит отримання даних

Також варто відзначити таку корисну функцію цієї СУБД як трігери. Тригер - це код SQL, що автоматично виконується у відповідь на певні події в певній таблиці. Він використовується для підтримки цілісності даних. Тригер в SQL працює подібно до реального тригера. Тобто виконання однією дії провокує виконання іншої.

Одним із прикладів тригера може бути створення тригера, який наприклад на додання рядка в таблицю, додає запис логування про створення в таблицю логів.

Таким чином, виділимо головні переваги та недоліки даної технології:

Переваги даної реляційної СУБД:

- Велика кількість типів даних
- Впровадження обмежень та регулювання даних, що додаються
- Потужні методи індексації
- Найвищий рівень надійності даних
- Підтримка JSON, XML
- Паралелізація запитів на читання
- Повноцінна та вичерпна документація
- Підтримує тригери
- Поєднання таблиць

Недоліки:

- Ефективність буде значно повільнішою при обробці великих обсягів даних
- Зв’язки між таблицями можуть ставати громіздкими
- Необхідне знання мови SQL
- Встановлення є непростим для початківців

ВИСНОВОК ДО РОЗДІЛУ 2

В даному розділі було проаналізовано найпопулярніші технології для створення веб-сервісу. Було розглянуто принципи роботи, способи використання, переваги та недоліки кожної з них. Керуючись отриманою інформацією можемо зробити висновок про те, які засоби найкраще підходять для вирішення поставленої задачі.

Таким чином, для написання клієнтської частини додатку прийнято рішення використовувати мову бібліотеку React. Це пов’язано з тим, що саме даний рішення використовує віртуальну об’єктну модель документа, що дозволяє їй найшвидше справлятися з задачами відображення вмісту сторінки та найшвидше реагувати на події та зміни на ній. Також саме цей інструмент налічує найбільшу спільноту користувачів, що зменшує час вирішення будь-якої проблеми, що може виникнути під час розробки програмного забезпечення.

Для написання серверної частини було обрано платформу Node.js. Головною та найбільш значною причиною для саме такого вибору було те, що написання коду саме за допомогою цього інструменту виконується за допомогою мови програмування JavaScript. Це мова, яка використовується для клієнтської частини, тому обравши саме її вилучається необхідність вчити ще одну мову, що значно сповільнило б час розробки. А також, саме це рішення є одним з найбільш гнучким та масштабованим, що робить розробку більш простою та дозволяє по ходу написання безболісно змінювати основні компоненти програмного забезпечення.

Оскільки розроблювана соціальна мережа буде працювати з великою кількістю однорідних даних та дуже важливою є цілісність та консистентність збережуваної інформації, було прийнято вирішення використовувати SQL рішення, бо саме СУБД такого типу можуть забезпечити ці запити.

Конкретною СУБД було обрано PostgreSQL через велику кількість різноманітних типів даних, швидкі поєднання таблиць та регулярні оновлення, які додають нові функції та вирішують проблеми, які помічає товариство, а також, через дуже детальну та зрозумілу документацію, що значно спрощує використання даної технології.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ТА РОЗРОБКА МОДЕЛІ

В даному розділі ми спроєктуємо та розробимо модель соціальної мережі, використовуючи обраний стек технологій. Розробимо покроково клієнтський інтерфейс та серверну частину.

3.1 Вимоги до можливостей додатку

Спочатку розпишемо повністю функціонал додатку, що буде реалізовано, щоб під час розробки опиратись на цей пункт й не упустити жодного важливого моменту.

При вході на веб-ресурс користувач може:

- Зареєструватися
- Ввійти в раніше створений аккаунт
- Продовжити використання ресурсом не авторизуючись

Якщо клієнт обирає реєстрацію – перед ним з'являється форма, яку він має заповнити. Вона має бути не надто великою, щоб користувачу не набридло вводити дані. Оптимальною кількістю полів будемо вважати 4, серед яких точно мають бути представлені ввід юзернейма та пароля. Після успішної реєстрації має бути перенаправлення на сторінку входу, де можна використати введені дані та ввійти в свій новостворений профіль.

У випаду вибору входу – з'являється форма, де необхідно ввести, юзернейм та пароль, якщо передані дані коректні – користувач успішно авторизується та перенаправляється на свою особисту сторінку.

У випадку коли користувач нехтує авторизацією – він все ще може використовувати сайт, але його функції мають бути дещо урізані. До таких функції можна віднести, наприклад, створення публікацій, уподобання публікацій інших людей та слідкування за користувачами.

Безперечно має бути функціонал, щоб з будь-якої частини сервісу можна було за один клік перейти на найголовніші сторінки:

- Домашня
- Особиста
- Повідомлень
- Налаштувань
- Пошуку користувачів
- Рекомендації користувачів

Найкраще це можна досягти за допомогою навігаційної стрічки, на якій будуть знаходитись посилання на дані частини додатку. Вона має бути завжди показана на екрані користувача.

На своїй сторінці користувач має бачити загальну інформацію про себе, фото та свої обрані інтереси та публікації. Мати можливість змінювати ці дані, створювати, видаляти та редагувати свої дописи. Також, має бути можливість бачити людей, яких він відслідковує, та людей, що відслідковують його.

На домашній сторінці має бути показано публікації людей, за якими слідкує авторизований користувач, причому, в порядку спадання відносно часу завантаження на ресурс, тобто нові – завжди зверху.

На сторінці повідомлень мають бути показані всі діалоги юзера, має бути можливість писати повідомлення та бачити час відправлення кожного з повідомлень. Бажано, щоб повідомлення приходили в реальному часі й не було необхідності перезавантажувати сторінку.

На сторінці налаштувань має бути можливість зміни особистих данних, місцяположення, інтересів, зміни фото профілю, а також відображення тимчасового стану профілю.

На сторінці пошуку людей має бути можливість вибору інтересів для пошуку, а також, місцеположення та інших фільтрів, а також можливість, щоб сервіс сам запропонував вам користувачів використовуючи вашу інформацію.

Пошукова стрічка має шукати користувачів с нікнеймом, який відповідає наданій послідовності символів.

В загальному, інтерфейс має бути зрозумілим та простим в використанні, виконання запитів має виконуватися максимально швидко щоб не погіршувати користувацький досвід.

3.2 Розробка системи

3.2.1 Розробка СУБД

Розробку веб-сервісу можна починати як з написання клієнтської та і з серверної. Почнемо з серверної оскільки для проектування так буде набагато краще. Для зберігання даних ми прийняли рішення використовувати PostgreSQL СУБД, оскільки вона належить до SQL рішень, то найправильнішим буде почати з проектування структури бази даних. Для приємної та продуктивної роботи будемо використовувати DBeaver - безкоштовний мультиплатформенний інструмент баз даних для розробників, адміністраторів баз даних та аналітиків. Він підтримує будь-яку базу даних, яка має драйвер JDBC (що в основному означає – будь-яку). Має багато функцій, включаючи редактор метаданих, редактор SQL, експорт / імпорт / міграція даних тощо. Почнемо з найпростішого – створимо структуру таблиць грунтуючись на функціоналі продукту. Таким чином їх вийшло 13.

Рисунок 3.1 Спрощена структура БД

Наведемо типовий скрипт SQL, що використовувався для створення таблиць:

```
CREATE TABLE Messages
(
  message_id BIGSERIAL,
  room_id INTEGER NOT NULL,
  sender_id INTEGER NOT NULL,
  context TEXT NOT NULL,
  send_at TIMESTAMP NOT NULL
);

ALTER TABLE Messages ADD CONSTRAINT pk_message_id
  PRIMARY KEY (message_id);

ALTER TABLE Messages ADD CONSTRAINT fk_sender_id_user_info
  FOREIGN KEY (sender_id) REFERENCES user_info (user_id)
  ON DELETE CASCADE;

ALTER TABLE Messages ADD CONSTRAINT fk_room_id
  FOREIGN KEY (room_id) REFERENCES room (room_id)
  ON DELETE CASCADE;

ALTER TABLE Messages ADD COLUMN is_read boolean;

ALTER TABLE Messages ALTER is_read
SET
DEFAULT FALSE;


ALTER TABLE Messages ALTER send_at
SET
DEFAULT NOW
();
```

Рисунок 3.2 Структура таблиці повідомлень

Спочатку задаємо необхідні поля, їх типи та, якщо потрібно, обмеження. Далі додаємо зв’язки між таблицями – з’єднали сутності користувача та чату з повідомленням. Додали умову, щоб у випадку видалення чату – видалялися всі повідомлення в ньому, щоб вивільнити пам’ять та не зберігати застарілі дані. Також, можемо побачити, що для поля «is_read», що показує чи було прочитане відправлене повідомлення додали значення за замовчуванням FALSE, що буде помічати щойно відправлене повідомлення як не прочитане, а для поля «send_at» додали також значення за замовчуванням, але в даному випадку – це виклик функції «DATE», що поверне коректний теперішній час.

Варто зазначити, що для коректної реалізації системи необхідно було використано всі три види зв’язків між таблицями [16]. Розглянемо використання кожної з них.

При взаємозв’язку «один до одного» один запис у таблиці пов’язаний з одним і лише одним записом в іншій таблиці. В нашому випадку, це до прикладу таблиці «User_info» та «Person». Оскільки дані дві сутності просто розділяють дані користувача на персональну інформацію та інформацію аккаунта, але по-суті стосуються лише одного користувача.

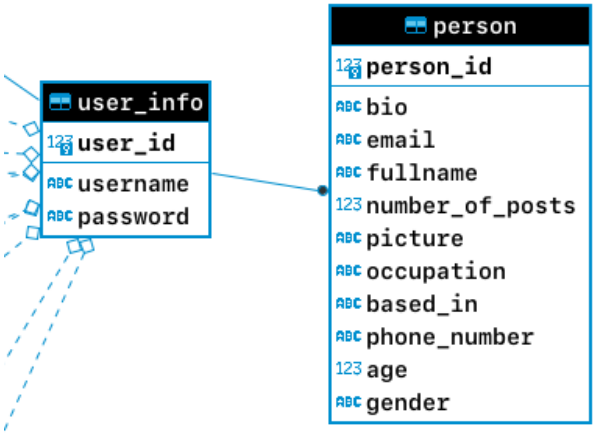


Рисунок 3.3 Зв’язок «один до одного»

Наступним типом є «один до багатьох» - коли один запис у таблиці може бути пов'язаний з одним або кількома записами в іншій таблиці. Саме цей зв’язок є найчастіше використовуваним в розробленому продукту, оскільки, користувач може створювати багато постів, але у кожного з них може бути лише один автор, так само з вподобаннями – кожен може вподобати багато дописів, але кожен допис може бути вподобаним одним акаунтом не більше одного разу. Таким чином, цей вид комунікації використовується для, наприклад, таблиць: «User_info» та «Post», «User_info» та «Like».

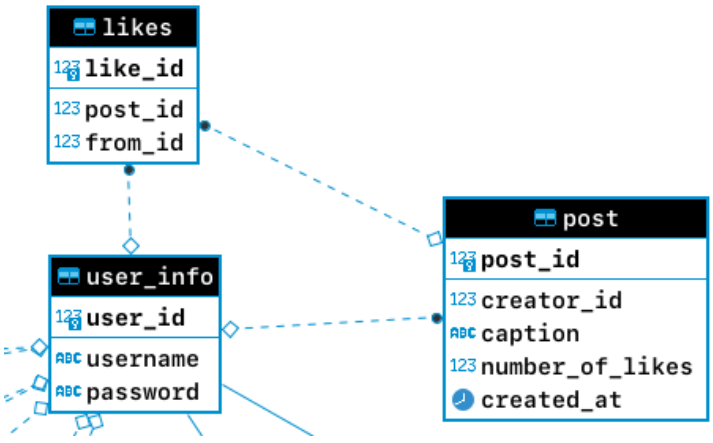


Рисунок 3.4 Зв’язок «один до багатьох»

Найцікавішим та найбільш важким для імплементації та розуміння є «багато до багатьох» [15]. Цей зв’язок використовується коли кілька записів у таблиці пов'язано з кількома записами в іншій таблиці. Тут можуть бути два різних випадки:

- Зв’язок таблиці з самою собою
- Зв’язок таблиці з іншою

Обидва варіанти було успішно використано, при написанні архітектури бази даних. Для такого типу зв’язку створюється окрема таблиця, що має, щонайменше, містити ідентифікатори обох сутностей

Перший варіант було застосовано на прикладі таблиці «Follow», дана таблиця поєднує сутності юзерів. Дана таблиця відповідає за відслідковування. Дана таблиця має два поля, кожне з яких посилається на ідентифікатор користувача. Таким чином ми поєднуємо того, хто відслідковує і того, хто є відслідковуваним, також, дана комбінація унікальна, оскільки неможливо та немає сенсу відслідковувати когось декілька разів.

Варіант з різними таблицями було використано для того, щоб користувачі мали можливість обирати мови, якими вони володіють, та ця опція була не обмежена лише одним варіантом. Тобто маючи таблицю людей та таблицю можливих до вибору мов, кожен юзер може знати більше одної мови та кожна мова може бути знаною більше ніж одним юзером. Таким чином, було створено таблицю «Mtm_user_language».

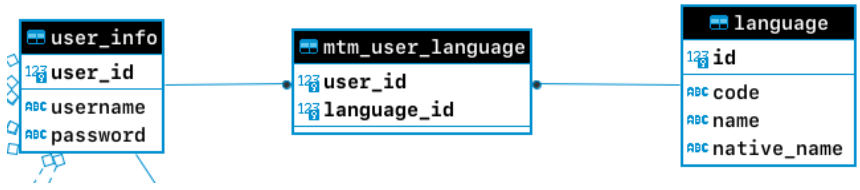


Рисунок 3.5 Зв’язок «багато до багатьох»

Для правильного поєднання було додано два зовнішніх ключі:

- fk_language_id – для зв’язку з таблицею мов
- fk_user_id – для зв’язку з таблицею клієнтів

Рисунок 3.6 Зовнішні ключі в таблиці «mtm_user_language»

3.2.2 Розробка серверної частини

Після розробки структури даних можемо перейти до серверної частини. Звичайно, писати веб-сервіси можна й на платформі Node.js, не використовуючи жодних додаткових ресурсів, але найчастіше додатково використовується один із фреймворків. Вони корисні тим, що скорочують час написання коду за рахунок вбудованої імплементації функціоналу, який довелося би писати заново. Було обрано фреймворк Express - мінімалістичний і гнучкий веб-фреймворк для додатків Node.js, що надає великий набір функцій для мобільних і веб-додатків. Маючи в своєму розпорядженні безліч службових методів HTTP і проміжних оброблювачів, створити надійний API можна швидко і легко.

```
const path = require('path');
const express = require('express');
const app = express();


app.use(express.static(path.join(__dirname, 'public')))


app.get('/', (req, res) => {
    res.send({ message: 'Hello WWW!' });
});


app.listen(3333, () => {
    console.log('Application listening on port 3333!');
});
```

Рисунок 3.7 Базовий веб-сервіс на Express

Маршрут являє собою кінцеву точку, до якої можуть звертатися користувачі. Він асоціюється з HTTP-дієсловом (наприклад, GET, POST і ін.) та отримує шлях URL. Крім того, він отримує функцію, яка викликається при зверненні до цієї кінцевої точки.

Отже, коли користувач виконує запит GET до домашньої сторінки, тобто localhost: 3333, викликається передана функція і відображається фраза "Hello WWW!".

Тепер, коли сервер запущений, можемо звернутися до нього в браузері. Перейдемо за адресою http://localhost:3333/, перед нами відображається очікуване повідомлення, що свідчить про коректне налаштування серверу:

Для створення проекту спочатку створимо порожню директорію, ініціалізуємо його та завантажимо пакет Express. Далі, створивши застосунок Express, ми налаштовуємо на якому порті він буде приймати запити, та напишемо тестовий маршрут, щоб перевірити, що початкове налаштування правильно і ми можемо переходити до розробки функціоналу.

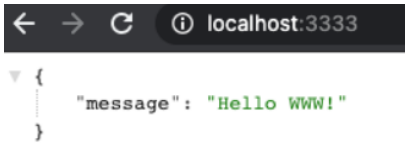


Рисунок 3.8 Домашня сторінка тестового сервісу

Розпочнемо з забезпечення користувачу можливості не вводити дані свого профілю щоразу після перезавантаження сторінки та входу на сайт. Для цього нам при кожному запиті ідентифікувати поточного користувача – для цього нам необхідно реалізувати аутентифікацію та авторизацію [23].

Аутентифікація - це процес ідентифікації користувачів та перевірки того, чи вони претендують бути не собою задля шахрайських намірів. Одним з найпоширеніших і очевидних факторів автентифікації особи є пароль. Якщо він збігається з обліковими даними пароля, це означає, що особа є дійсною, і система надає доступ користувачеві.

Цікаво, що на системах, які працюють без пароля, багато хто використовує сучасні методи автентифікації, такі як одноразові паролі (OTP) за допомогою SMS або електронною поштою, єдиний вхід (SSO), багатофакторна автентифікація (MFA).

Авторизація відбувається після успішної аутентифікації користувача. Йдеться про надання повного або часткового права доступу до таких ресурсів, як база даних та інша важлива інформація для виконання роботи. Також за допомогою авторизації сервер розуміє який саме клієнт намагається звернутися до нього і розуміє, які дані мають бути доступні саме для нього.

Аутентифікацію було реалізовано за допомогою юзернейма та пароля – тобто, коли користувач вводить дані параметри, відправляється запит і сервер перевіряє отримані дані звіряючи з тими, що в базі, якщо дані коректні, то юзера аутентифіковано. Задля подальшого розуміння введемо поняття кукі та сесій

Файл cookie - це невеликий файл максимальним розміром 4 КБ, який веб-сервер зберігає на клієнтському комп'ютері. Він відправляється разом з іншими заголовками HTTP. Після встановлення файлу всі наступні запити на сторінку повертають ім’я та значення файлу. Файл кукі можна прочитати лише з домену, з якого він виданий.

Сесія - це глобальна змінна, що зберігається на сервері. Кожній сесії присвоюється унікальний ідентифікатор, який використовується для отримання збережених значень. Щоразу, коли створюється сесія, файл кукі, що містить унікальний ідентифікатор сеансу, зберігається на комп'ютері користувача та повертається з кожним запитом на сервер. Сесії здатні зберігати відносно великі дані порівняно з файлами кукі.

Таким чином процедура авторизації в розробленому веб-сервісі виглядає наступним чином:

1) Юзер відвідує сторінку логіну та вводить нікнейм та пароль
2) Сервер продивляється заголовки і бачить, що кукі немає
3) Сервер створює сесію, куди додає ідентифікатор користувача, записує її в базу данних та відправляє ідентифікатор сесії в заголовку «Set-Header», таким чином виставляючи кукі в браузері клієнта
4) Юзер робить інші запити на сервіс
5) Сервер знаходить кукі, а в них – ідентифікатор сесії, робить запит до бази, щоб перевірити чи дійсно така сесія існує, в разі якщо – так, користувача авторизований, якщо ні – то ні, і сервер на основі цих даних розуміє які дані доступні та можуть бути надані
6) Це повторюється з кожним іншим запитом

Всю роботу з сесіями та куками реалізовано за допомогою бібліотек «express-session» та «Passport.js». Вони представляють собою зручний інструмент для створення, перевірки та збереження в базу даних сесій.

```
app.use(
  session({
    store: new pgSession({
      pool: pgPool,
      tableName: 'session',
    }),
    secret: process.env.FOO_COOKIE_SECRET,
    saveUninitialized: false,
    resave: false,
    cookie: { maxAge: 30 * 24 * 60 * 60 * 1000 }, // 30 days
  })
);
```

Рисунок 3.9 – Ініціалізація сесій

За допомогою параметрів ми можемо конфігурувати налаштування створення сесій. Ви зазначаємо назву таблиці куди їх зберігати, секретний текст, що буде використовуватися для генерації випадкового ідентифікатора, час життя кукі та інші параметри.

Коли ви вводите URL-адресу та натискаєте кнопку вводу, браузер, по суті, надсилає HTTP-запит на сервер, який пов'язаний з цією URL-адресою. Потім сервер відповідає HTTP-відповіддю і може надіслати щось на зразок файлу HTML.

Після встановлення зв’язку та здійснення транзакції зв’язок перестає існувати. Більше не може відбуватися спілкування між браузером та сервером.

Що, якщо нам потрібно створити чат, де ми хочемо отримувати нові повідомлення? З використанням HTTP нам довелося б продовжувати оновлювати сторінку, щоб побачити нові повідомлення. Однак це не зручно для користувачів. Для цієї мети було прийняти використовувати таку технологію як вебсокети.

Вебсокети дозволяють здійснювати двонаправлений (дуплексний) зв’язок. Це означає, що ми можемо змусити клієнта та сервер спілкуватися між собою, не перериваючи з’єднання. Більше ми не повинні покладатися на протокол HTTP для спілкування в режимі реального часу. Навпаки, ми “оновлюємо” протокол HTTP, щоб ми могли підтримувати зв’язок між клієнтом та сервером.

Коли WebSocket вперше запускається, він надсилає простий HTTP-запит на вказану нами URL-адресу. Звідти HTTP-запит “оновлюється” до сокета TCP, по суті, захищеного тунелю для проходження даних після здійснення HTTP-рукостискання. Рукостискання можна сприймати як "домовленість" між клієнтом та сервером про підтримку зв'язку [27].

Для використання вебсокетів є широкий вибір бібліотек, було обрану найстабільнішу та найбезпечнішу з поміж усіх – «ws» [26].

Існує чотири основних типи подій, які відбуваються з вебсокетами: «Відкрити», «Повідомлення», «Закрити» та «Помилка».

- Відкриття - після встановлення зв'язку між клієнтом та сервером відкрита подія запускається. Якщо необхідно зробити якусь дію, коли з’єднання встановлено, ми можемо визначити функцію, що буде виконана після спрацювання події.
- Повідомлення - кожен раз, коли сервер хоче надіслати деякі дані, це називається "повідомленням", оскільки це буквально нове повідомлення, яке ваш браузер отримає від сервера. Повідомленнями можуть бути звичайний текст, двійкові дані, JSON, дані зображень та багато іншого.
- Закриття- щоразу, коли зв’язок між клієнтом і сервером закривається, спрацьовує дана подія. З’єднання може бути розірвано через поганий зв’язок або через визначення закритої події. Після того, як з'єднання буде закрито, клієнт і сервер не зможуть обмінюватися подальшими повідомленнями.
- Помилка - Щоразу, коли під час спілкування між клієнтом та сервером виникає помилка, спрацьовує подія Error. Ми можемо отримати детальну інформацію про помилку та способи її усунення, визначивши функцію, що буде виконана у випадку цієї події.

Події запускаються, коли щось трапляється. Вони реагують на певні ситуації, що відбуваються.

З іншого боку, є дії. Дії ініціативні та мають намір. Дії - це те, що запускається, коли користувач хоче щось зробити.

Є дві основні дії: «Надсилання» та «Закриття» [25].

- Надсилання - використовується коли ми хочемо надіслати якусь інформацію. В нашому випадку – будь-яка спроба надіслати повідомлення викликає цю функцію і так дане повідомлення потрапляє на сервер, де оброблюється
- Закриття - використовується для закриття вебсокетного з’єднання за допомогою функції close (). Цей метод є по суті «прощальним рукостисканням». Після того, як зв’язок буде закрито, його слід відновити, якщо буде відбуватися подальше спілкування.

Загальний принцип роботи вебсокетного з’єднання в розробленій системі:

1) Сервер запускає вебсокет сервіс
2) Клієнт провокує з’єднання передаючи URL веб-сервісу
3) З’єднання встановлено
4) Кожна з сторін представлена своїми обробниками подій та діями, які працюють один з одним

3.2.3 Розробка клієнтської частини

Для розробки клієнтської частини використовувався React.js. Оскільки зараз найпопулярнішим та найзручнішим для використання є односторінкові додатки - додатки, яким не потрібно перезавантажувати сторінку під час її використання та які працюють в браузері. Для коректної роботи такого підходу обов’язково треба використовувати якийсь менеджер стану. Було прийнято рішення зупинитися на найпопулярнішому та найкраще задокументованому – Redux. Він є контейнером для управління станом додатку. Він не прив'язаний безпосередньо до React.js і може також використовуватися з іншими js- бібліотеками і фреймворками.

Ключові складові Redux:

- Сховище: зберігає стан додатка
- Дії: деякий набір інформації, який виходить від додатка до сховища і який вказує, що саме потрібно зробити. Для передачі цієї інформації у сховища викликається метод dispatch ().
- Творці дій: функції, які створюють дії
- Ред’юсер: функція (або кілька функцій), яка отримує дію і відповідно до цього дією змінює стан сховища

Загальну схему взаємодії елементів архітектури Redux можна виразити таким чином:

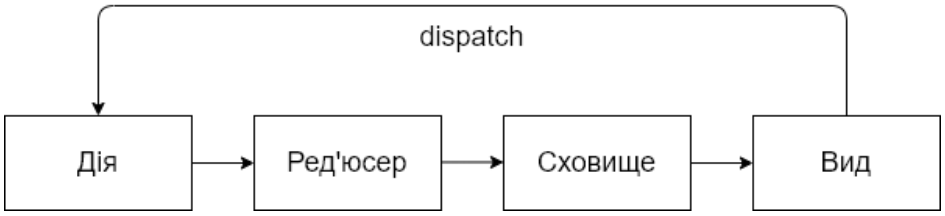


Рисунок 3.10 Архітектура Redux

З виду (тобто з компонентів React) ми посилаємо дію, ця дію отримує функція ред’юсер, яка відповідно до дією оновлює стан сховища. Потім компоненти React застосовують оновлений стан зі сховища для відображення актуальних даних, оскільки будь-яка змінна в сховищі перемальовує компонент, який використовує його.

```
import { UPDATE_BIO, ADD_INTERESTS, ADD_PICTURES, EDIT_CURRENT } from './types';

const initialState = {
  id: null,
  username: null,
  picture: null,
  interests: null,
};

const currentPageReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'SET_CURRENT_PAGE':
      return {
        ...state,
        id: action.payload.id,
        username: action.payload.username,
        picture: action.payload.picture,
      };

    case ADD_INTERESTS: {
      const { interests } = action.payload;
      return { ...state, interests };
    }

    case EDIT_CURRENT: {
      const { updatedFields } = action.payload;
      return { ...state, ...updatedFields };
    }
  }
};

export default currentPageReducer;
```

Рис 3.11 Приклад ред`юсера

Клієнтська частина для покращення розуміння коду та розділення обов’язків була поділена наступним чином:

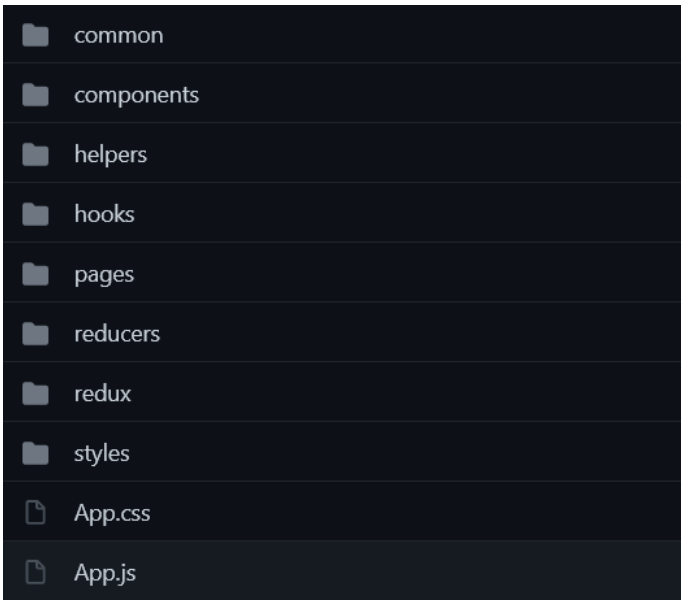


Рисунок 3.12 Структура клієнтської частини мережі

Найголовнішою частиною тут є файл App.js – який є початковим для будь-якого React застосунку. З найголовнішого можемо відмітити маршрутизацію, яка описується саме в ньому. Ми визначаємо URL та

обираємо для нього компонент який буде відображено. Також є можливість редіректів, що є корисним в випадку, коли у користувача немає досить прав для відображення якоїсь сторінки. На рисунку 3.11 показані декілька використовуваних маршрутів.

```
<Router>
        <div className='container'>
          <Switch>
            <Redirect from='/' exact to={`/${userInfo.username}`} />
            <Redirect from='/login' exact to={`/${userInfo.username}`} />
            <Redirect from='/signup' exact to={`/${userInfo.username}`} />
            <Route
              exact
              path='/account'
              render={(props) => <Settings {...props} />}
            />
            <Route
              exact
              path='/search'
              render={(props) => <Search {...props} />}
            />
            <Route exact path='/home' render={(props) => <Home {...props} />} />
            <Route render={(props) => <NotFound {...props} />} />
          </Switch>
        </div>
      </Router>
```

Рисунок 3.13 Маршрутизація в React

Кожна з директорій відповідає тільки за свій функціонал і може бути перевикористана. Наприклад, в директорії з сторінками зберігаються різні маршрути соціальної мережі в ній відбувається підключення компонентів, що показуються на екрані.

Найбільшою є папка з компонентами оскільки візуальна складова всіх сторінок знаходиться саме тут. Кожен файл представлений в JSX форматі, що дозволяє поєднувати синтаксис мови розмітки HTML та мови програмування JavaScript.

JSX - це розширення синтаксису до Javascript, що використовується в ReactJS, що дозволяє писати Javascript, схожий на HTML. Іншими словами, це свого роду шаблонізатор, але з підтримкою Javascript. Даний синтаксис дуже схожий на HTML. Після компіляції він перекладається на звичайні виклики функцій Javascript.

Основними перевагами використання саме цього розширення, а не просто коду, написаного на «чистому» JavaScript:

- Спрощує створення компонентів
- Швидше ніж JavaScript, оскільки підлягає оптимізації
- Дозволяє зберігати логіку та шаблон разом

```
import React, { useState, useEffect } from 'react';

const Post = () => {
  const loggedInUser = useSelector((state) => state.loggedInUser);
  let likeButtonClasses = liked ? 'fa fa-heart liked' : 'fa fa-heart';
  const [loginModal, setLoginModal] = useState(false);
  useEffect(() => {
    setLiked(likes?.alreadyLiked);
  }, [likes]);

  if (!likes) return <div></div>;

  return (
    <div>
      <div className='POST__body'>
        <div className='POST__title'>
          <div
            className='POST__title__left'
            onClick={() => history.push(`/${username}`)}
          >
            <img src={picture} alt='avatar' className='POST__profile_picture' />
            <div className='POST__header'>
              <h3 className='POST__fullname'>{fullname}</h3>
              <h4 className='POST__username'>@{username}</h4>
            </div>
          </div>
    </div>
```

Рисунок 3.14 JSX-файл

Під час написання застосунку активно використовувалися гаки або хуки – це така абстракція, що дозволяє писати більш простий для розуміння код. Раніше, щоб керувати станом необхідно було обов'язково використовувати класові компоненти, але з часом, додання функціональних компонентів, повністю змінило все, оскільки, вони більш зрозумілі і легші в написанні. Таким чином, необхідно було, винайти інструмент, що управлял би станом в саме таких компонентах. І в команди розробників даної бібліотеки це вийшло, так з'явилися гаки. Таким чином, оскільки у класових компонентів більше не бкло ніяких можливостей, що не було б у функціональних – тепер можна писати весь застосунок лише на функціональних компонентах.

Хуки бувають двох типів – стандарті та власноруч написані. Продемонструємо по одному прикладу використання кожного варіанту [22].

Стандартний гачок useState - він використовується для управління

локальним станом у функціональних компонентах, приймає початковий стан як аргумент і повертає, використовуючи деструктурування масиву, дві змінні, які можна назвати як ви того забажаєте. Тоді як перша змінна є фактичним станом, друга змінна є функцією оновлення стану шляхом надання нового стану. Тобто, є лише один варіант змінити фактичне значення зміної – викликати відповідну функцію, і передати туди нове значення. Цікавим та корисним є те, що при виклику функцією наявний доступ до початкового стану змінної.

```
function App() {
  const [list, setList] = React.useState(INITIAL_LIST);

  function onRemoveItem(id) {
    const newList = list.filter(item => item.id !== id);
    setList(newList);
  }

  return (
    <ul>
      {list.map(item => (
        <li key={item.id}>
          <a href={item.url}>{item.title}</a>
          <button type="button" onClick={() => onRemoveItem(item.id)}>
            Remove
          </button>
        </li>
      ))}
    </ul>
  );
}
```

Рисунок 3.15 Використання стандартного гака «useState»

Для написання своїх гачків необхідно зрозуміти як вони працюють, але, загалом, це просто функція, що може приймати значення та повертати, а всередині себе використовувати стандартні гаки [21]. Найчастіше ця частина коду буде часто перевикористовуватися, тому її й виносять окремо.

Найчастіше використовуваним розробленим гаком є useFetch, який спрощує процес відправки запиту на сервер та отримання відповіді. Без використання його, всі нижче описані дії довелося б робити окремо для кожного запиту, щоб в значній мірі збільшило б кодову базу і зменшило читабельність.

- Відловлювання помилок
- Відслідковування чи запит вже закінчився
- Конвертація JSON

Більша частина компонентів була розроблена власноруч, оскільки жодна з готових реалізацій повністю не влаштовувала своїм дизайном та не зовсім підходила під розроблюваний продукт. Для деяких часто використовуваних елементів було використано бібліотеку компонентів Material UI, яка добре себе зарекомендувала, була розроблена Google та використовується в їх та не лише їх додатках та застосунках [29].

Material UI - це бібліотека компонентів для React, наповнена потужними компонентами, які можна використовувати у своїх проектах безкоштовно. Material може надати доступ до надійних попередньо стилізовані компонентів які допоможуть в розробці. Також дана бібліотека дозволяє повністю змінювати їх вид за допомогою стилів.

Рисунок 3.16 Компоненти Material UI

ВИСНОВОК ДО РОЗДІЛУ 3

В даному розділі було виконано опис небхідних дій для створення повноцінної соціьаної мережі. Використовуючи обрані раніше інструменти була розроблена загальна архітектура, що дозволяла починаючи з початку, постійно покращувати функціонал, додаючи нові функції та можливості для користувача.

В основі всього проектування лежала побудова стабільної та надійної бази даних, що б в свою чергу гарантувала безпечність даних. Дана ціль була успішно досягнута. Через відносини між таблицями відносно легко було інтегрувати нові функції під час виконання завдання.

Контролери для оброблення запитів на серверному боці також виправдали всі сподівання оскільки с масштабуванням та розробкою не виникало жодних проблем, як, і з написання клієнтського коду, оскільки використовувана бібліотека повністю побудована на компонентах, які можно легко перевикористовувати в різних місцях застосунку.

Таким чином, весь задумани функціонал було реалізовано, соціальна мережа з адаптивним пошуком користувачів була розроблена.

РОЗДІЛ 4
ПРЕДСТАВЛЕННЯ РОЗРОБЛЕНОЇ СОЦІАЛЬНОЇ МЕРЕЖІ

4.1 Демонстрація роботи

В результаті розробки було створено соціальну мережу для пошуку знайомств. Оскільки однією, з найголовніших характеристик будь-якого веб-ресурсу є привітливий користувацький інтерфейс та швидка робота, цим двум аспектам було приділено максимум уваги.

Щоб почати свою роботу з цим додатком, необхідно запустити сервіс, який в свою чергу пов'язаний як з базою даних, так і з клієнтським застосунком. Для цього необхідно встановити Node.js, версіїї не нижче за 12. Після успішного старту, необхідно перейти в браузер за посиланням «localhost:3000»

При переході на даний ресурс, відкриється сторінка логування, оскільки це перший захід на сайт.

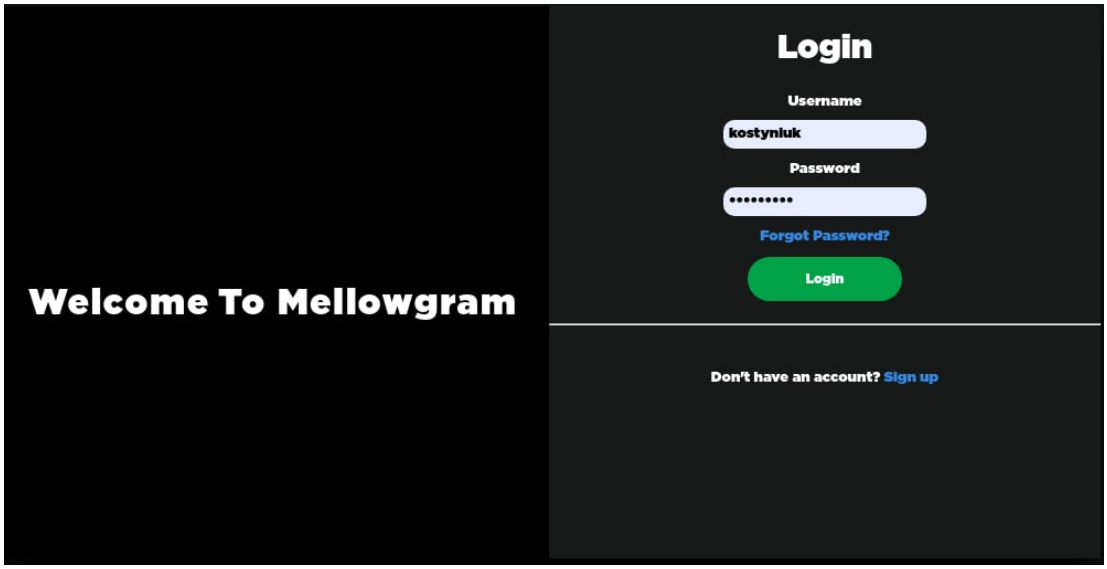


Рисунок 4.1 Віконце логування

Як бачимо, є можливість або ввести необхідну для входу інформацію в спеціальні поля, або зареєструватись. Вибравши останню опцію, ми потрапляємо на сторінку реєстрації.

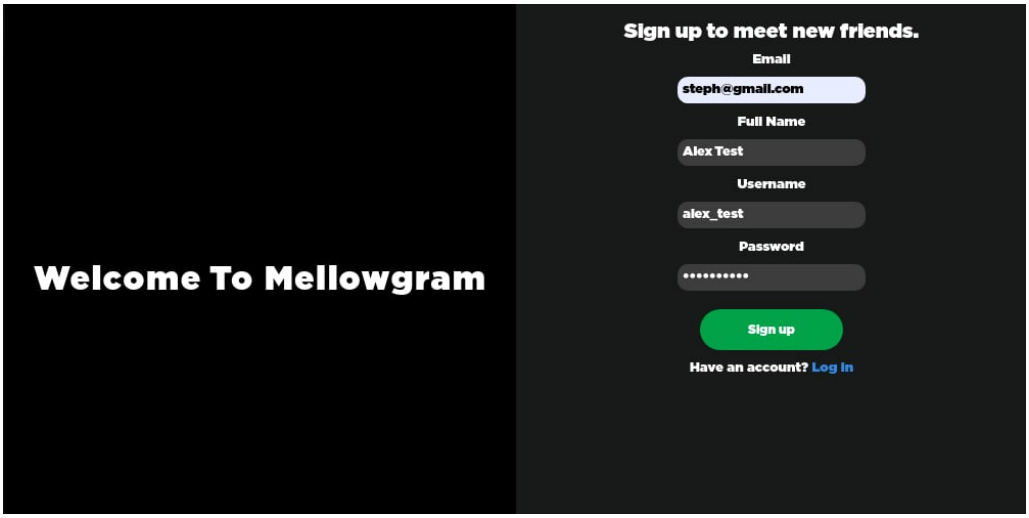


Рисунок 4.1 Віконце реєстрації

Тут ми маємо заповнити всі поля і закінчити реєстрацію. Після її закінчення система аавтоматично перенаправить вас знову на сторінку логування.

На сторінці логування користувач, у випадку неправильно наданої інформації, отримає помилку, про некоректність даних, у іншому випадку, буде перенаправлений на його особисту сторінку.

Особиста сторінка розділена на два блоки:

- Інфомації користувача
- Дописів

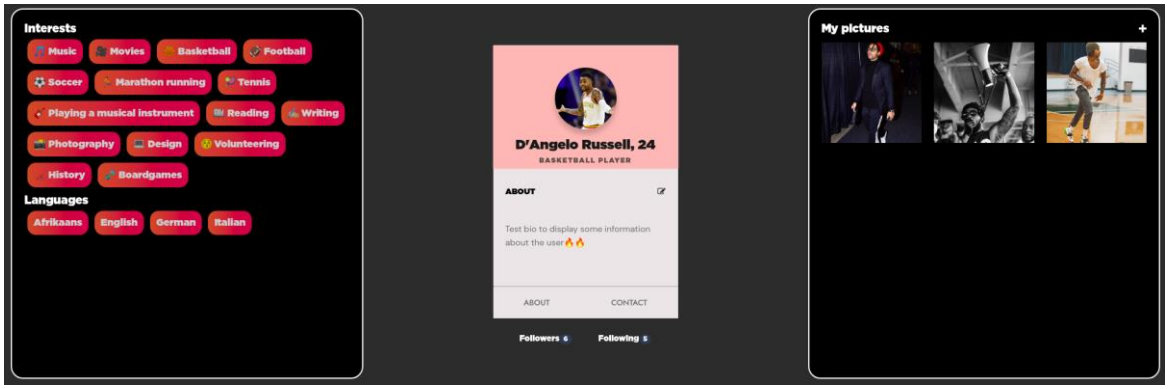


Рисунок 4.3 Блок інформації користувача

В перщому знаходиться інформація про користувача, така як: ім’я, вік, професія, коротка інформація про нього, інтереси, мови, якими він володіє та додані ним фото.

Якщо користувач знаходиться на своїй сторінці, то в нього з’являються додаткові можливості. Наприклад, додати фотографії, змінити інформацію про себе.

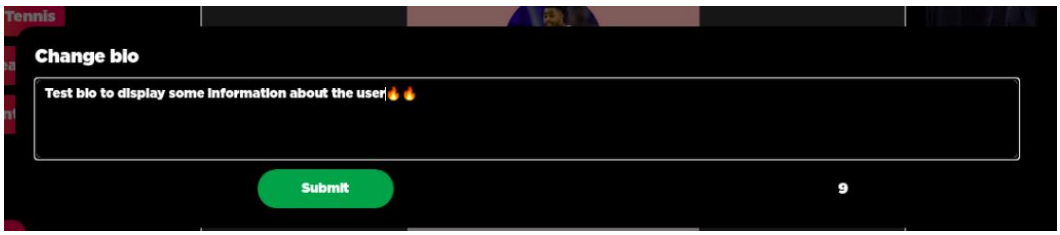


Рисунок 4.4 Віконце зміни інформації

Оскільки дана інформація є коротким описом, то регламентовано максимально можливу кількість символів – 50.

Також в цьому блоці показуються люди, які слідкують за цим користувачем, та профілі, які він сам відслідковує

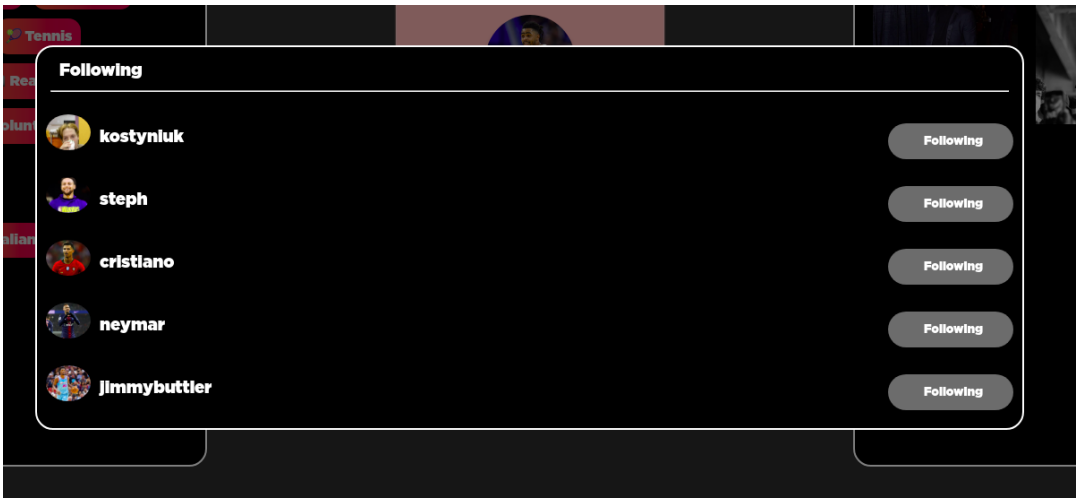


Рисунок 4.5 Віконце відслідковуваних користувачів

При натисканні на користувача, ви будете перенаправлені на його сторінку. Також є можливість почати його відслідковувати прям з цього місця соціальної мережі.

Блок з дописами показує дописи користувача, починаючи від найновіших. Кожен пост має дату публікації, показник кількості вподобань та можливість продивитися людей, що його вподобали.

Також є можливість створювати нові дописи.

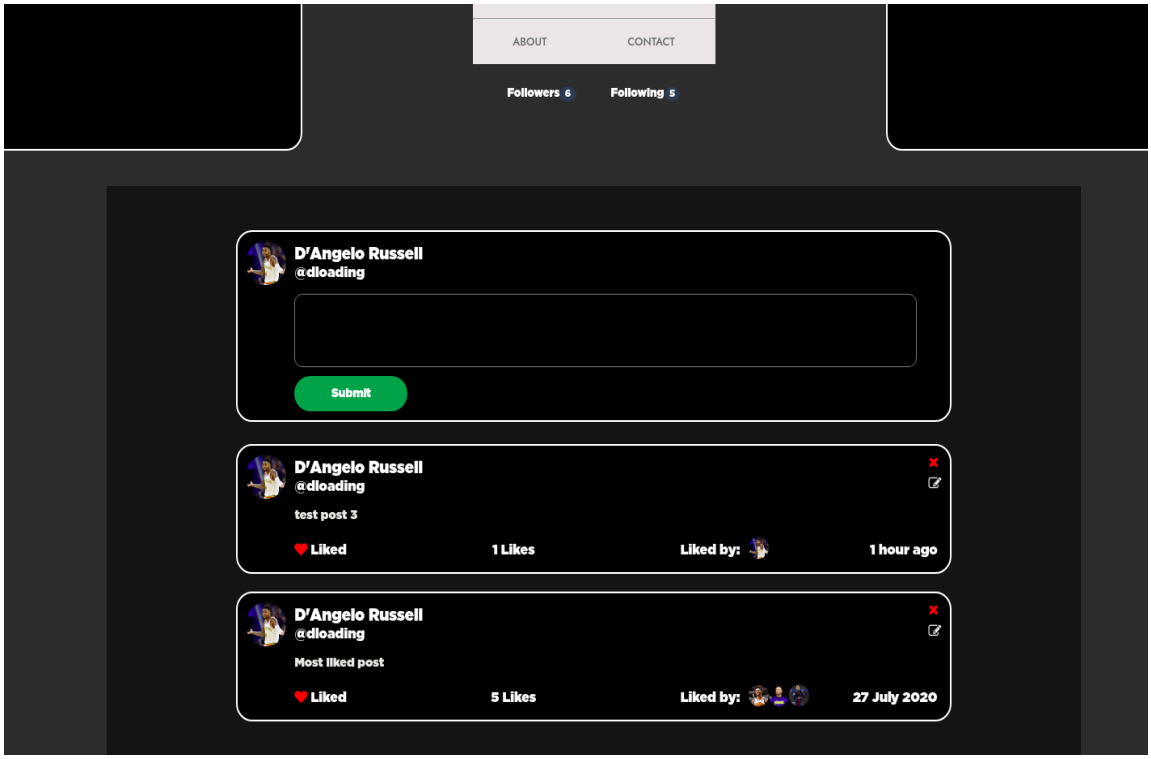


Рисунок 4.5 Блок дописів

Після ознайомлення з домашньою сторінкою користувач може перейти до налаштувань. Це відбувається за рахунок заголовка швидкого доступу, що знаходиться на екрані на будь-якій сторінці даного веб-ресурсу.

Рисунок 4.6 Заголовок швидкого доступу

Можемо помітити, що окрім кнопки налаштувань, присутня можливість знайти будь-якого користувача за його нікнеймом за допомогою пошукача в лівій частині. Також можливість перейти на домашню, особисті сторінки та сторінки пошуку та повідомлень. Над кнопкою повідомлень знаходиться покажчик, що сигналізує кількість непрочитаних повідомлень користувача.

Перейшовши до налаштувань, можемо обрати що саме хочемо налаштувати :

- Локацію
- Налаштування профілю
- Змінити інтереси
- Змінити мови спілкування

- Змінити пароль
- Видалити аккаунт

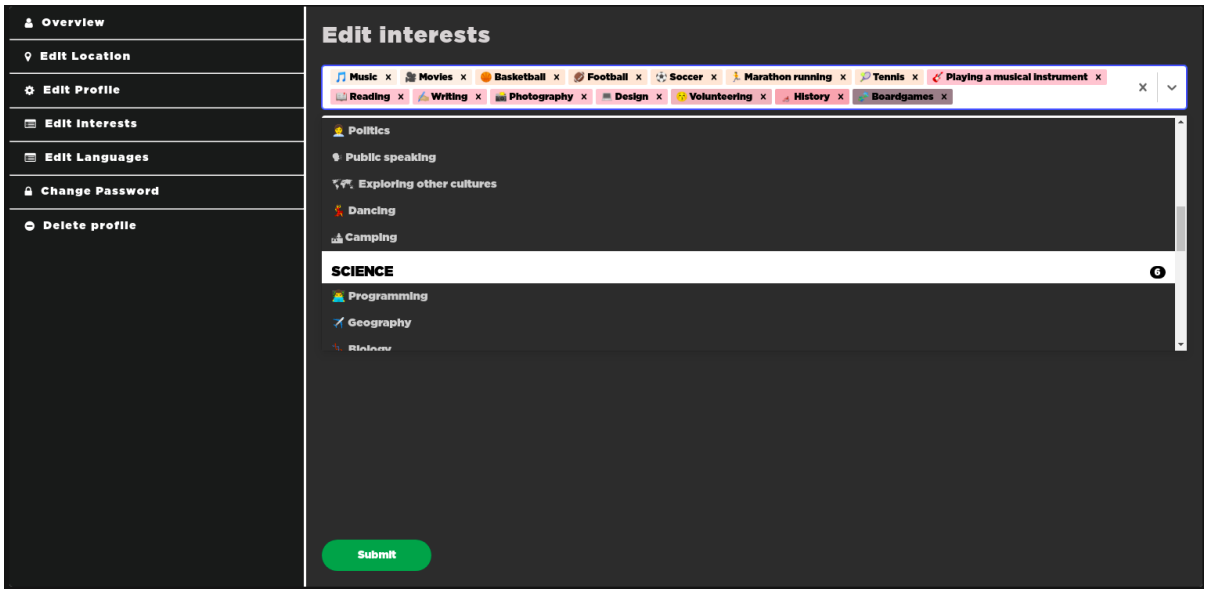


Рисунок 4.7 Сторінка налаштувань

Зупинимось детальніше на можливості зміни інтересів, оскільки, це одна з найважливіших функцій при знаходженні нових знайомств. Вони тут зібрані в окремі категорії, щоб було простіше шукати. Вибір є досить великим, присутні більше ніж 50 різноманітних опцій. Є можливість вибрати довільну кількість без будь-яких обмежень.

Перейдемо на домашню сторінку – сторінку, де відображаються дописи людей, за якими слідкує користувач. Дописи розміщені в порядку останього завантаження.

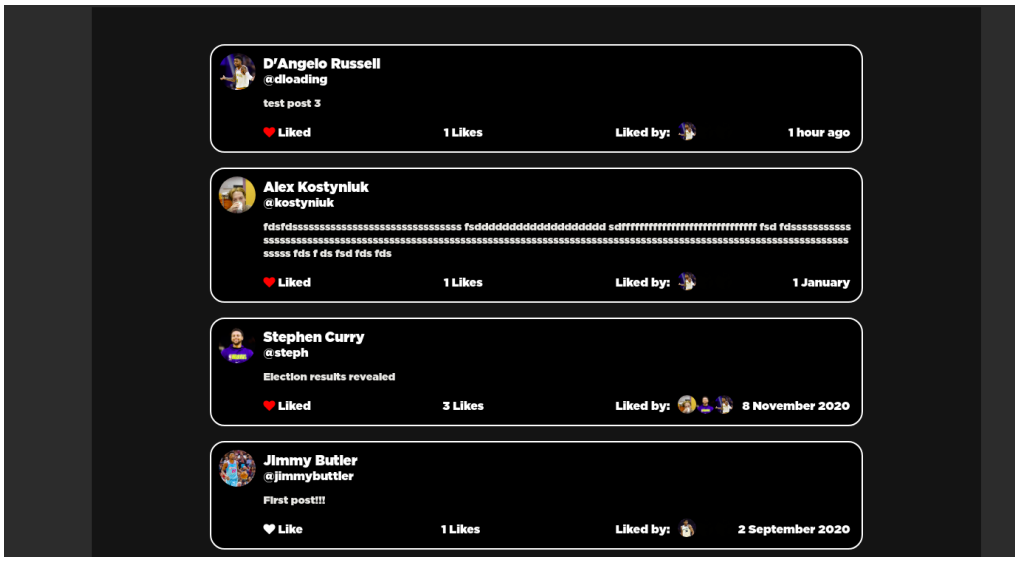


Рисунок 4.8 Домашня сторінка

При відвідуванні чиєїсь сторінки наявна кнопка для відправлення повідомлень.

Рисунок 4.9 Фрагмент сторінки користувача

При натиску на дану кнопку відбудеться відкриття сторінки з повідомленнями, там де показуються всі активні чати і є можливість писати повідомлення. Зазначимо, що у неавторизованих користувачів такої можливості немає.

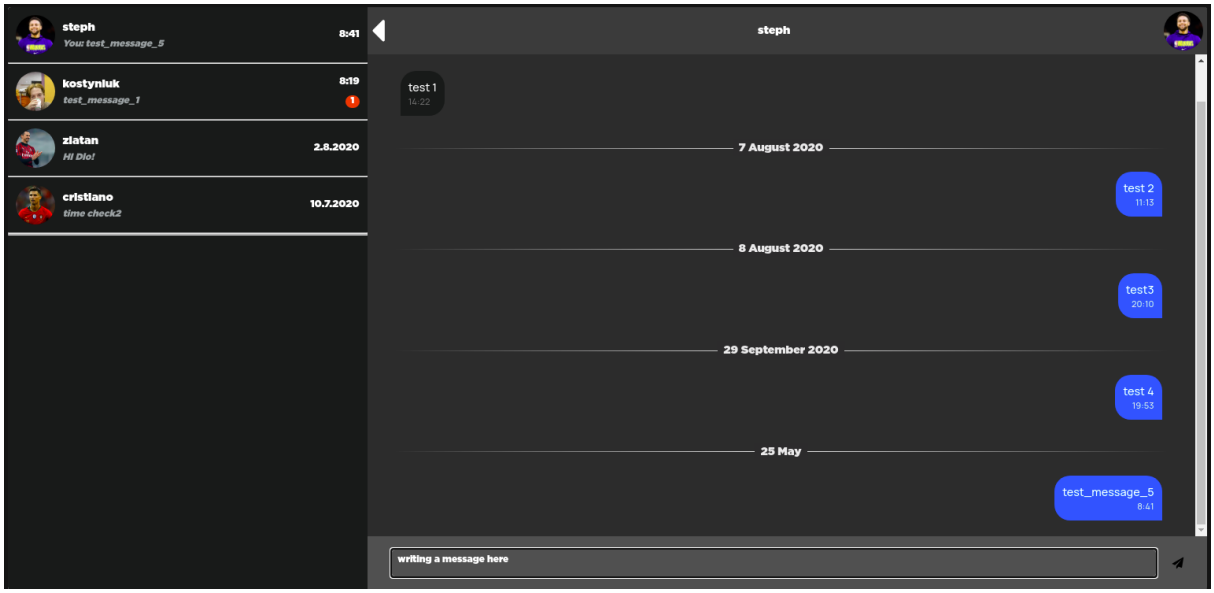


Рисунок 4.10 Сторінка повідомлень

Як бачимо вона є досить інформативною, оскільки відображає багато корисної інформації. Наприклад, чи є непрочитані повідомлення в якомусь чаті, останнє повідомлення, час та автор його відправки з кожного чату, кожне повідомлення маркується окремим кольором та місцеположенням, що не дає можливості заплутатись в їх приналежності. Також є роздільники, що показують дату відправки, а також час відправлення повідомлень.

Повідомлення з’являються без перезавантаження сторінки. При надходженні нового повідомлення з’являється нотифікація в верхній частині екрану, щоб користувач відразу знав про надходження.

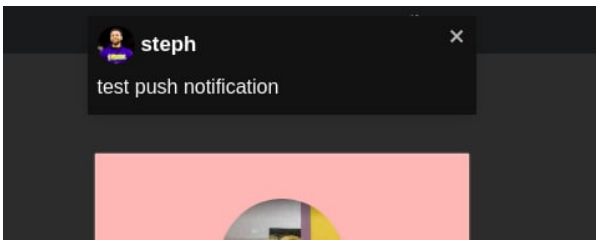


Рисунок 4.11 Пуш-нотифікація

Головною сторінкою виступає сторінка пошуку знайомств. Як зазначалося раніше є дві опції по роботі з нею:

- Адаптивний пошук
- Ручний пошук

За вибір конкретного типу пошуку відповідає спеціальний слайдер.

При виборі адаптивного пошуку всі поля є неактивними, оскільки в них немає жодного сенсу тому, що вся інфомація для пошуку береться з інформації авторизованого користувача.

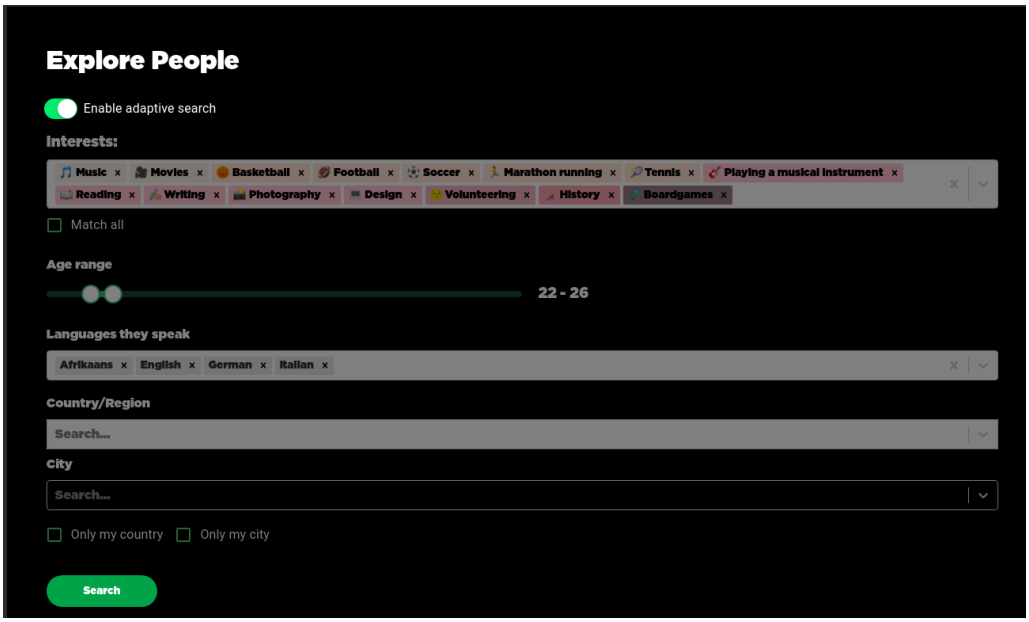


Рисунок 4.12 Адаптивний пошук

В результаті отримаємо список користувачів, що система вважає підходящими для знайомства. Список складається з карток користувачів, на яких відображена основна їхня інформація.

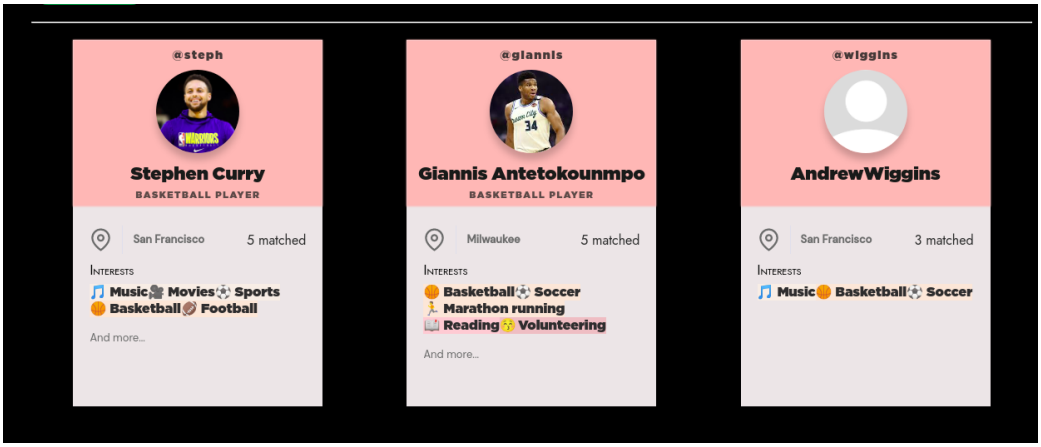


Рисунок 4.13 Результат адаптивного пошуку

При ручному пошуці, користувач автоматично задає критерії для пошуку – серед них:

- Інтереси
- Вік
- Мови спілкування
- Країна проживання
- Місто проживання
- Можливість вибирати тільки користувачів зі своєї країни або свого міста
- Можливість обирати тільки користувачів, в яких присутні повністю всі обрані інтереси

В результаті система знаходить підходящих користувачів і відображає їх на екрані.

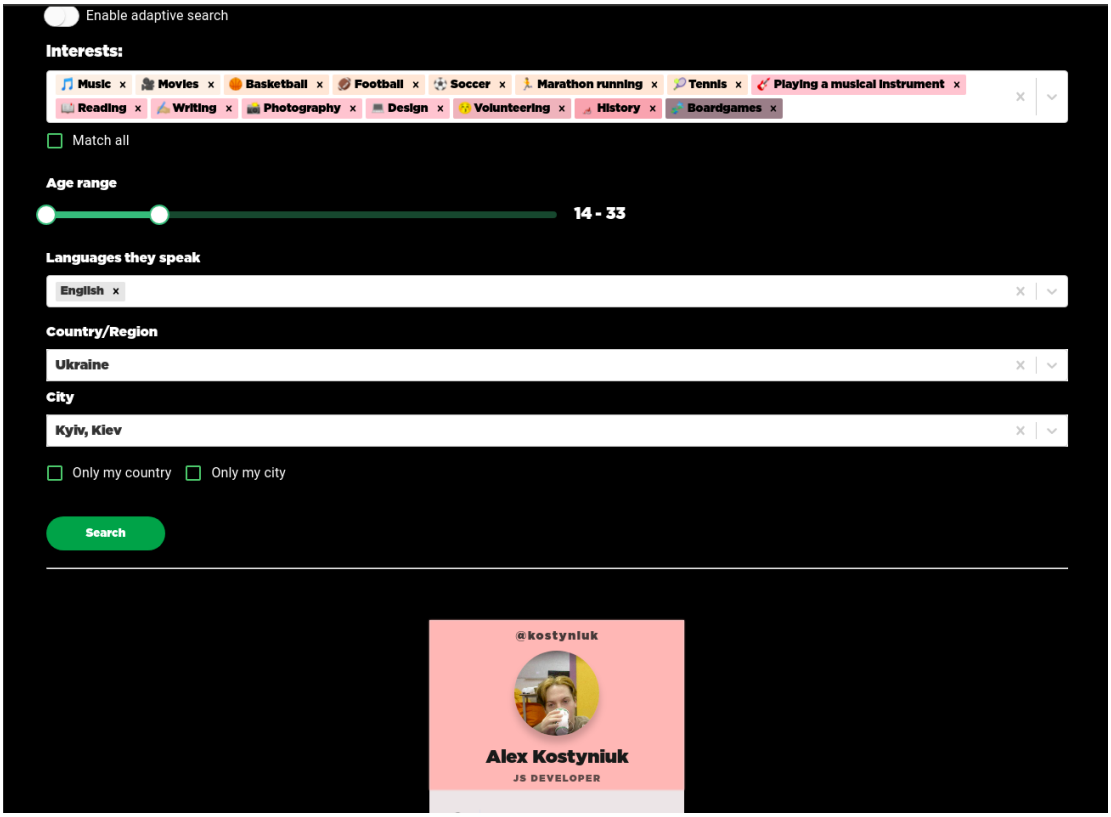


Рисунок 4.14 Ручний пошук користувачів

ВИСНОВОК ДО РОЗДІЛУ 4

В даному розділі було описано та продемонстровано використання написаної соціальної мережі. Були заповнені тестові дані для перевірки коректності результатів. За допомогою створення декількох користувачів було перевірено весь функціонал додатку

Було протестовано роботу на декількох операційних системах:

- Windows 10
- MacOS Catalina
- IOS 14

Також було перевірено коректність роботи в різних браузерах:

- Google Chrome 90
- Firefox 83
- Edge 90

Кожен з тестових запусків не виявив жодних недоліків, оскільки при розробці, було приділено увагу сумісності створюваного продукту з середовищами, в яких він буде використовуватися

.

ВИСНОВКИ

В першому розділі бакалаврського проєкту було розглянуто основні тенденції та способи створення соціальних мереж для пошуку знайомств. Розглянуто конкурентні рішення, їх основний функціонал, виокремлено позитивні підходи, що стали в нагоді під час розробки, будучи певним орієнтиром. Також, приділено увагу негативним частинам, з метою уникнення їх в проєкті. Задано основний функціонал, який згодом був реалізований.

В другому розділі було проаналізовано основні тенденції до створення веб-сервісів, а також, проведений огляд найпопулярніших технологій для його реалізації. Було виділено по 2-3 кандидати для написання клієнтської та серверної частини, а також 2 кандидати для організації СУБД Зіставивши всі переваги та недоліки кожної з них, було прийнято рішення про використання технологій, що найкраще підходять для розроблюваної системи – Node.js, React.js та PostgreSQL.

В третьому розділі було приділено багато уваги на вивчення обраних інструментів, знайомство з інфраструктурою та бібліотеками. В результаті, за допомогою отриманих знань та вмінь, було спроектовано структуру даних, яка якнайкраще дозволяє зберігання інформації та її обробку. Розроблена структура є масштабована, про що свідчить те, що під час розробки, з’являлися все нові й нові ідеї, що були без проблем успішно втілені в життя. Серверна частина побудована на контролерах, що дозволяє, також, нарощувати функціонал в майбутньому. Клієнтський код створений на перевикористовуваних компонентах, що зменшує час розробки та допомагає в написанні нового функціоналу за рахунок використання старих базових конструкцій.

В четвертому розділі було показано приклад використання створеної соціальної мережі, продемонстровано принципи роботи та розглянуто основний функціонал. Також було перевірено коректну роботу розробки на різних пристроях та середовищах запуску.

В результаті виконання даного дипломного проєкту було створено продукт, що являє собою соціальну мережу для пошуку знайомств. Обравши підходящі технології було досягнута швидку роботу сайту та приємний користувацький інтерфейс, що дозволяє навіть новому користувачу без проблема орієнтуватися в ньому.

Було вирішено проблеми пошуку знайомств в мережі Інтернет базуючись не лише на якійсь системі, що не зрозуміло як підбирає рекомендації. Розроблена система дозволяє користувачу дуже гнучко задавати необхідні вподобання, не обмежуючись лише місцем положенням та віком. Додання можливості створювати публікації дозволяє краще зрозуміти чи бажаєте ви знайомитися чи ні, щоб часто важко сказати, дивлячись тільки на фото.

Створена соціальна мережа значно краще виконує функцію сервісу для пошуку знаймоств ніж зазначені конкуренти з першого розділу. Цей факт лише підтверджує те, що система може виконувати своє призначення у реальному світі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. JavaScript: The Definitive Guide, 7th Edition / David Flanagan // O'Reilly Media, Inc. – 2020 – https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/.
2. Software Engineering at Google / Titus Winters, Tom Manshreck, Hyrum Wright // O'Reilly Media, Inc. – 2020 – https://www.oreilly.com/library/view/software-engineering-at/9781492082781/.
3. A Smarter Way to Learn JavaScript: The new tech-assisted approach that requires half the effort/ Mark Myers// CreateSpace Independent Publishing Platform– 2014 – https://www.amazon.de/dp/1497408180?tag=hackr03c-21&geniuslink=true.
4. Eloquent JavaScript: A Modern Introduction to Programming/ Marjin Haverbeke// No Starch Press – 2015 – https://www.amazon.de/dp/1593275846?tag=hackr03c-21&geniuslink=true.
5. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Moderns JS Libraries/ Eric Elliott // O'Reilly – 2014 – https://www.amazon.de/dp/1491950293?tag=hackr03c-21&geniuslink=true.
6. Introduction to Programming Using Java/ David J. Eck// Hobart and William Smith Colleges – 2006 – https://www.iitk.ac.in/esc101/share/downloads/javanotes5.pdf.
7. Teach yourself Java in 21 days / Laura Lemay// Sams.net – 2017 – https://www.cs.cmu.edu/afs/cs.cmu.edu/user/gchen/www/download/java/LearnJava.pdf.
8. Learning Python / Marc Lutz // O'Reilly – 2009 – https://cfm.ehu.es/ricardo/docs/python/Learning_Python.pdf
9. Python Basics / David Amos, Dan Bader // Real Python– 2020 – https://static.realpython.com/python-basics-sample-chapters.pdf.
10. Getting started with Angular / Minko Gechev // Packt – 2017 – http://englishonlineclub.com/pdf/Getting%20Started%20with%20Angular%20(Second%20Edition)%20[EnglishOnlineClub.com].pdf.

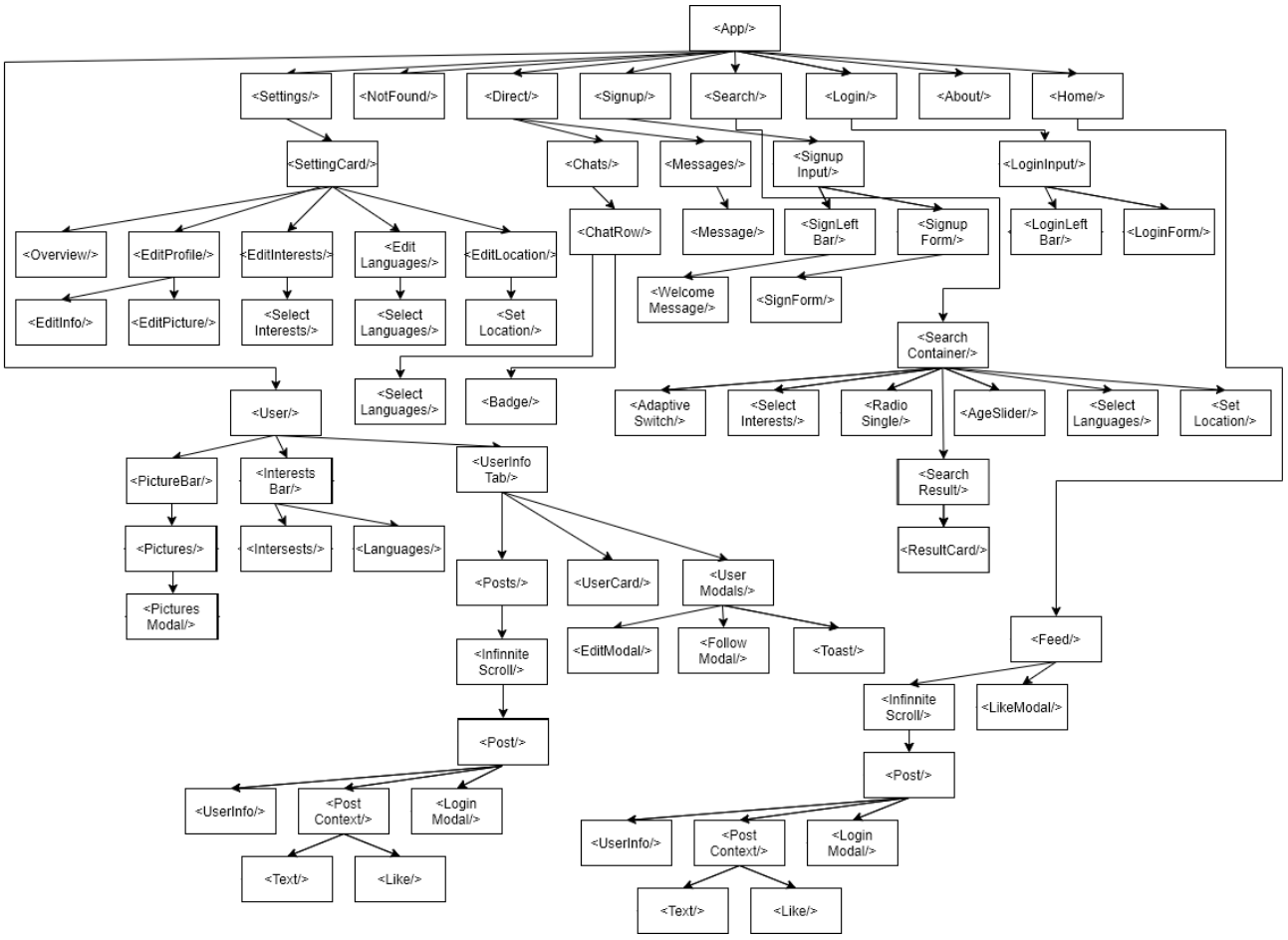
11. What And Why React.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.c-sharpcorner.com/article/what-and-why-reactjs/. (дата звернення 3.03.2021)
12. What is Python? Everything You Need to Know - Skillcrush [Електронний ресурс] – Режим доступу до ресурсу: https://skillcrush.com/blog/what-is-python/. (дата звернення 1.03.2021)
13. Django explained [Електронний ресурс] – Режим доступу до ресурсу: https://programmingwithmosh.com/react/react-virtual-dom-explained/#:~:text=Virtual%20DOM%20is%20a%20virtual%20representation%20of%20the%20real%20DOM.&text=The%20virtual%20DOM%20then%20sends,the%20versions%20of%20virtual%20DOM/. (дата звернення 2.03.2021)
14. Rest API Design Book / Marc Masse // O'Reilly – 2017 – https://pepa.holla.cz/wp-content/uploads/2016/01/REST-API-Design-Rulebook.pdf.
15. Нормализация отношений. Шесть нормальных форм [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/254773/ . (дата звернення 8.03.2021)
16. Many (to many) relations among the stars [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/swlh/many-to-many-relations-among-the-stars-1728ba18a2d0 /. (дата звернення 10.03.2021)
17. Rest API Design Book/ Phil Spector // Stat – 2015 – https://www.stat.berkeley.edu/~spector/sql.pdf.
18. Learning SQL / Alan Beaulieu // O'Reilly – 2012 – http://www.r-5.org/files/books/computers/languages/sql/mysql/Alan_Beaulieu-Learning_SQL-EN.pdf.
19. MongoDB in Action / Kyle Banker, Peter Bakkum // Manning – 2016 – https://pepa.holla.cz/wp-content/uploads/2016/07/MongoDB-in-Action-2nd-Edition.pdf.
20. MongoDB In-Depth [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/analytics-vidhya/mongodb-in-depth-part-1-59f5aeda87cc (дата звернення 13.03.2021)

21. How React hooks work - in depth [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/eliav2/how-react-hooks-work-in-depth-17o9. (дата звернення 17.03.2021)
22. Frustrations with React Hooks [Електронний ресурс] – Режим доступу до ресурсу: https://blog.logrocket.com/react-hooks-frustrations/. (дата звернення 30.04.2021)
23. Express/Node introduction - Learn web development [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction/. (дата звернення 3.04.2021)
24. PostgreSQL server programming / Usama Dar, Hannu Krossing // Packt – 2015 – http://sd.blackball.lv/library/PostgreSQL_Server_Programming_Second_Edition_(2015).pdf
25. Web Socket/ Sanele Msweli // O'Reilly – 2015 – https://html.scribdassets.com/87qv27neio6urk7x/images/1-0e962a11a8.pdf
26. Research of Web Real-Time Communication Based on Web Socket / Liu Qigang // Shangai University – 2012 – https://www.researchgate.net/publication/276491172_Research_of_Web_Real-Time_Communication_Based_on_Web_Socket/
27. Node.js and Websockets best practices checklist [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/voodoo-engineering/websockets-on-production-with-node-js-bdc82d07bb9f/. (дата звернення 21.04.2021)
28. Material-UI: A popular React UI framework [Електронний ресурс] – Режим доступу до ресурсу: https://material-ui.com/. (дата звернення 12.04.2021)
29. Getting Started With Material-UI For React [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/codingthesmartway-com-blog/getting-started-with-material-ui-for-react-material-design-for-react-364b2688b555. (дата звернення 14.04.2021)
30. Clean Code / Robert C. Martin / Prentice Hall – 2008 – https://enos.itcollege.ee/~jpoial/oop/naited/Clean%20Code.pdf

Додаток А
СТРУКТУРНА СХЕМА РОБОТИ

до дипломного проєкту
на тему: «Соціальна мережа з адаптивною рекомендацією користувачів»

Київ – 2021 року

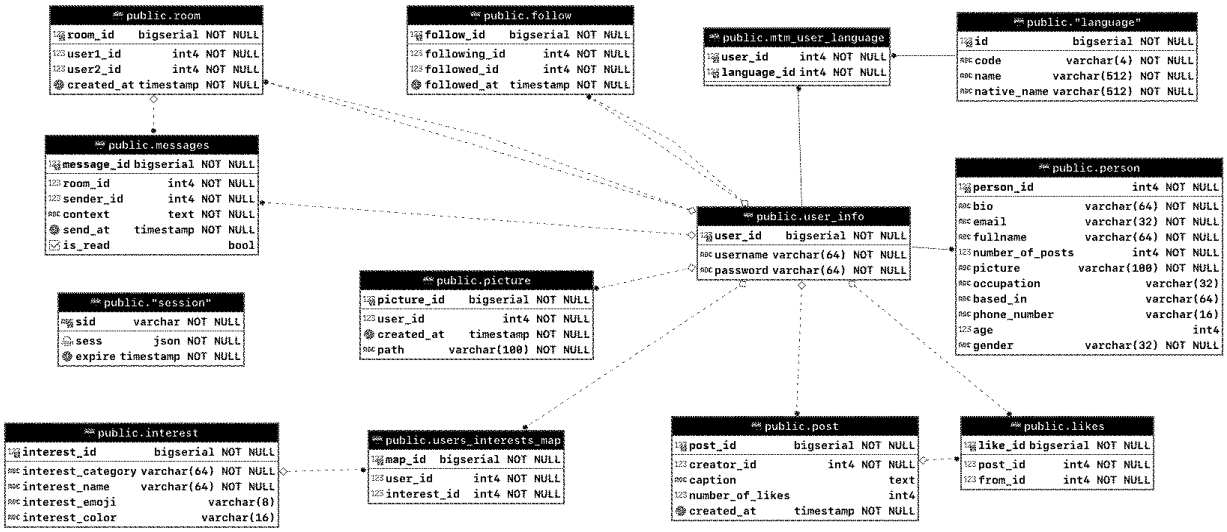


Зм.	*Арк.*	*№ документа*	*Підпис*	*Дата*			
Розробив		*Костинюк О. А*			*Соціальна мережа з адаптивною рекомендацією користувачів Структурна схема моделі*	*Літ.*	*Аркуш*	*Аркушів*
Перевірив		*Алєнін О.І .*				Т		1
						НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІО - 71		
Н. контр.		*Симоненко В.П.*						
Затв.		*Стіренко С.Г.*						

Додаток Б
ФУНКЦІОНАЛЬНА СХЕМА ДАННИХ

до дипломного проєкту
на тему: «Соціальна мережа з адаптивною рекомендацією користувачів»

Київ – 2021 року

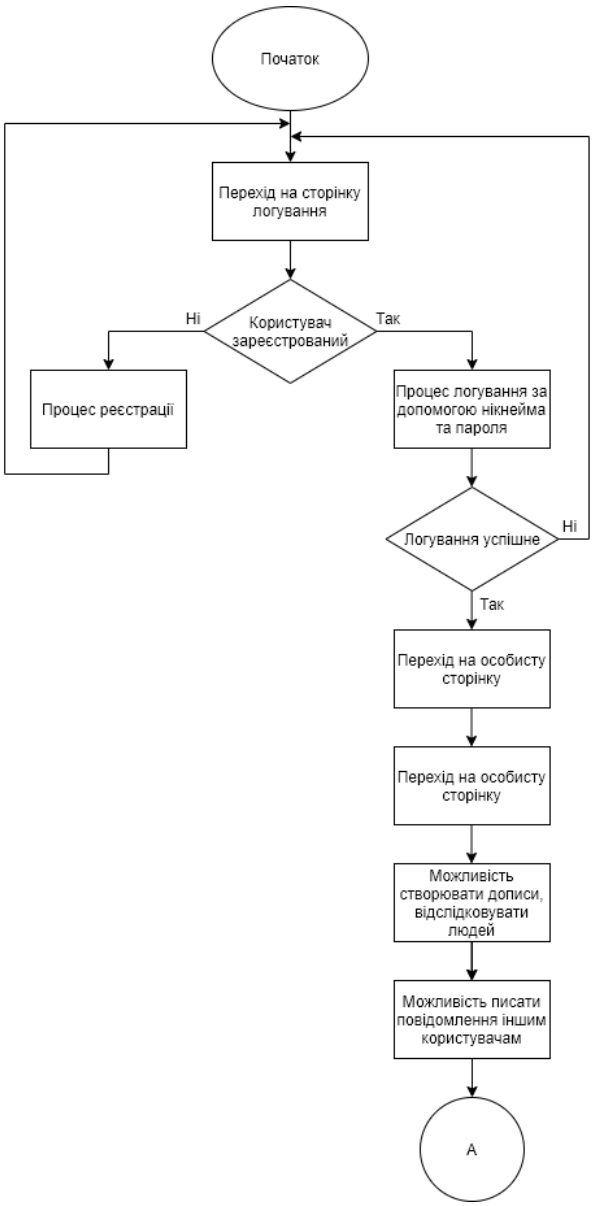


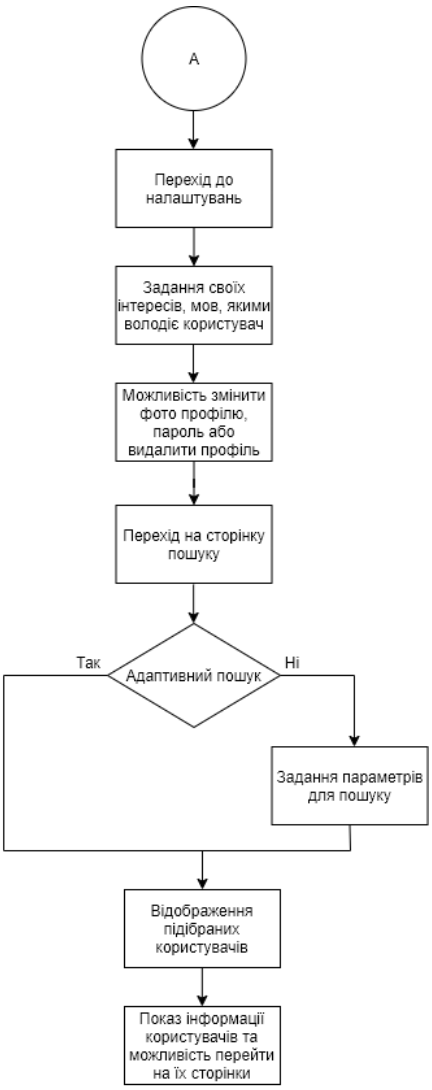
					ІАЛЦ.467100.005 Д2	
Зм.	*Арк.*	*№ документа*	*Підпис.*	*Дата*		
Розробив		*Костинюк О.А..*			*Соціальна мережа з адаптивною рекомендацією користувачів Функціональна схема данних*	*Літ.* / *Аркуш* / *Аркушів*: Т / 1 / 1
Перевірив		*Алєнін О.І.*				
Н.контр.		*Симоненко В.П.*				*НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІО - 71*
Затв.		*Стіренко С.Г.*				

Додаток В
ПРИНЦИПОВА СХЕМА РОБОТИ

до дипломного проєкту
на тему: «Соціальна мережа з адаптивною рекомендацією користувачів»

Київ – 2021 року





					ІАЛЦ.467100.006 ДЗ	
Зм.	*Арк.*	*№ документа*	*Підп.*	*Дата*		
Розробив		*Костинюк О.А..*			*Соціальна мережа з адаптивною рекомендацією користувачівПринципова схема роботи*	*Літ.* Т	*Аркуш* 1	*Аркушів* 1
Перевірив		*Алєнін О.І..*						
Н.контр.		*Симоненко В.П.*				*НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІО - 71*		
Затв.		*Стіренко С.Г.*						