

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Сайт знайомств»

Виконав студент групи ІПЗ-20-3
ГРУШЕВИЦЬКИЙ Сергій Віталійович

Керівник роботи:
ГОРДІЄНКО Ірина Валеріївна

Консультант:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
ЄФРЕМОВ Юрій Миколайович

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

«ЗАТВЕРДЖУЮ»

Завідувач кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГРУШЕВИЦЬКИЙ Сергій Віталійович**

Керівник роботи: **ГОРДІЄНКО Ірина Валеріївна**

Тема роботи: **«Сайт знайомств»,**

затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74/с**

Вихідні дані для роботи: **розробка сайту знайомств.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Гордієнко І. В., Чижмотря О. Г.	27.02.2024р.	27.02.2024р.
2	Гордієнко І. В., Чижмотря О. Г.	27.02.2024р.	27.02.2024р.
3	Гордієнко І. В., Чижмотря О. Г.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Пояснювальна записка з кваліфікаційної роботи на тему «Сайт знайомств» містить текстову частину на 80 сторінках друкованого тексту, 11 сторінок додатків, 35 рисунків та 34 таблиці. Список використаних джерел включає 30 найменувань і займає 3 сторінки. Загальний обсяг роботи становить 91 сторінок.

Основною метою є розробка сайту знайомств, що спрямований на вдосконалення процесів пошуку нових знайомств, забезпечення зручності та ефективності для користувачів, а також усунення недоліків інших платформ для знайомств.

Було визначено основні етапи і завдання розробки, включаючи визначення вимог користувачів, створення прототипу системи, аналіз аналогічних програмних рішень, вибір засобів реалізації, розробку структури бази даних та алгоритмів функціонування системи. Обрано відповідну архітектуру та розроблено модульну структуру системи з об'єктно-орієнтованими компонентами. Особлива увага була приділена детальному опису програмної реалізації сайту, включаючи розробку інтерфейсу користувача, інтеграцію функціональних модулів та оптимізацію продуктивності. Описано процеси реєстрації та авторизації користувачів, створення профілів, пошуку партнерів, обміну повідомленнями, а також методи, які забезпечують зручність використання та інтуїтивність інтерфейсу для кінцевих користувачів.

КЛЮЧОВІ СЛОВА: САЙТ ЗНАЙОМСТВ, ВЕБ-ДОДАТОК, ПОШУК, КОРИСТУВАЧ, СИСТЕМА.

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Сайт знайомств	Літ.	Арк.	Аркушів
Розроб.		Грушевицький С.В.						
Керівник		Гордієнко І.В.					3	91
Рецензент		Єфремов Ю.М.				Житомирська політехніка, група. ІПЗ-20-3		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The explanatory note for the qualification work on the topic "Dating Site" includes 80 pages of text, 11 pages of appendices, 35 figures, and 34 tables. The list of references contains 30 titles and occupies 3 pages. The total volume of the work is 91 pages.

The main goal is to create a dating site aimed at improving the processes of finding new acquaintances, ensuring convenience and efficiency for users, and eliminating the shortcomings of existing platforms.

The main stages of development included determining user requirements, creating a system prototype, and analyzing similar software solutions. The selection of implementation tools and the development of the database structure were also important components of the development process.

The appropriate architecture was chosen, and a modular system structure with object-oriented components was developed. Special attention was paid to the detailed description of the software implementation of the site, including the development of the user interface.

The integration of functional modules and performance optimization were also important stages of the work. The processes of user registration and authorization, profile creation, partner search, and messaging were described in detail.

Methods that ensure ease of use and an intuitive interface for end-users were also thoroughly considered.

KEYWORDS: DATING SITE, WEB APPLICATION, SEARCH, USER, SYSTEM.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ВЕБ-ОРІЄНТОВАНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ САЙТУ ЗНАЙОМСТВ	8
1.1 Постановка задачі	8
1.2 Аналіз аналогів програмного продукту	9
1.3 Вибір архітектури веб-орієнтованої системи	13
1.4 Обґрунтування вибору інструментальних засобів	18
Висновки до першого розділу	19
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ СИСТЕМИ САЙТУ ЗНАЙОМСТВ	21
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	21
2.2 Розробка бази даних системи	29
2.3 Проектування та реалізація алгоритмів роботи системи.....	34
Висновки до другого розділу.....	48
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З САЙТОМ ЗНАЙОМСТВ	50
3.1 Порядок встановлення та налаштування параметрів системи.....	50
3.2 Структура інтерфейсу вебдодатку та порядок роботи	51
3.3 Тестування роботи та використання сайту	69
Висновки до третього розділу	75
ВИСНОВКИ.....	76
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	78
ДОДАТКИ.....	81

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

СУБД – Система управління базами даних

SQL – Structure Query Language

JSON – JavaScript Object Notation

API – Application Programming Interface

JS – JavaScript

ПК – Первинний ключ

ЗК – Зовнішній ключ

ПЗ – програмне забезпечення

БД – База даних

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

У сучасному світі, на фоні стрімкого розвитку інформаційних технологій, сфера ІТ визначає нові тренди у різних сферах людської діяльності. Однією з актуальних тем є розробка сайтів для пошуку нових знайомств, які відображають сучасні вимоги та очікування спільноти.

Актуальність теми полягає в тому, що в сучасному світі, де люди все більше використовують інтернет для соціальних взаємодій, сайти для знайомств стають необхідністю. Зростаючий інтерес до цього напрямку визначає необхідність вивчення та розробки нових інструментів, що полегшать та зроблять ефективнішим процес знайомств через мережу.

Метою дослідження є розробка та реалізація сайту для пошуку нових знайомств, що враховує сучасні потреби користувачів та використовує передові технології в галузі веб-розробки. Для досягнення поставленої мети необхідно вирішити комплекс завдань, спрямованих на проведення аналізу, визначення архітектури, вибір технологій та розробку необхідного програмного забезпечення.

Завдання дослідження включають:

- Аналіз існуючих сайтів знайомств, їхніх функцій та недоліків.
- Визначення вимог до системи та користувацького інтерфейсу.
- Розробка архітектури вебдодатку.
- Вибір та обґрунтування використання сучасних технологій для розробки сайту.
- Реалізація функцій реєстрації, авторизації, створення профілю, налаштування та отримання рекомендацій та інтеграція чату.
- Проведення тестування та оптимізація роботи системи.

Об'єктом дослідження є розробка, проектування та реалізація сайту для знайомств за певними критеріями.

Предметом дослідження є використання веб-орієнтованих технологій задля створення сайту знайомств.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ВЕБ-ОРІЄНТОВАНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ САЙТУ ЗНАЙОМСТВ

1.1 Постановка задачі

У сучасному світі, де соціальні взаємодії здебільшого здійснюються онлайн, розробка сайту для пошуку нових друзів є актуальним завданням. Завдання проекту полягало в тому, щоб створити ефективний та інтуїтивно зрозумілий інструмент для полегшення та розширення можливостей соціальної взаємодії для користувачів в онлайн-середовищі.

Етапи та завдання розробки:

1. Вимоги до програми та визначення функцій:

- Проведення аналізу ринку та конкурентів для визначення основних функцій.
- Визначити вимоги до інтерфейсу користувача та зручності використання.

2. Розробити дизайн програми:

- Створення дизайну основних частин програми відповідно до сучасних тенденцій.
- Розробка адміністративної частини керування користувачами та контентом.

3. Проектування та розробка архітектури програми:

- Розробка архітектури сервера для забезпечення ефективного обміну даними.
- Розробка клієнтської архітектури для забезпечення швидкої та зручної взаємодії користувача з програмою.

4. Розробка компонентів програми, сторінок і функціоналу:

- Надання функціональних можливостей для зберігання та обробки інформації про користувачів та їх профілі.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

- Реалізація функціоналу пошуку, фільтрації та сортування профілів.
- Впровадження системи рекомендацій та аналізу інтересів для покращення взаємодії.

5. Тестування роботи програми:

- Проведіть тестування продуктивності кожного компонента та функціональності.
- Виявлення та виправлення помилок і недоліків для забезпечення стабільної та безаварійної роботи.

Результатом виконання завдань має стати веб-додаток, який забезпечить ефективний та захоплюючий процес пошуку нових знайомств для користувачів в онлайн-середовищі.

1.2 Аналіз аналогів програмного продукту

Для сайту знайомств було проведено аналіз аналогів програмного забезпечення серед існуючих сайтів, а також визначено їхні переваги та недоліки.

У зв'язку зі зміною світових тенденцій виникла необхідність створити зручний сайт для пошуку нових знайомих, який міг би пропонувати гнучкі фільтри за інтересами та цілями, самостійно підбираючи відповідних кандидатів.

Одним з таких онлайн-додатків є Tinder, ця програма популярна не лише в Україні, а й у всьому світі [24]. Тому ви можете знайти пару практично в будь-якій точці світу. Ще одна перевага — геолокація знаходить потенційних партнерів поблизу (мінімальна відстань — 1 км, але її можна збільшити до кількох сотень кілометрів). Базовий інтерфейс, можливість додати профіль Instagram (щоб не завантажувати кілька фотографій) і захист від надокучливих користувачів роблять цю програму однією з найкращих для онлайн-знайомств. Так, лише взаємно перевірені користувачі можуть почати чат.

Переваги:

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

- Спілкування, тільки коли симпатія взаємна;
- Інтуїтивно зрозумілий інтерфейс, що робить його доступним для людей будь-якого віку та технічного досвіду.;
- Має ряд функцій безпеки, які допомагають захистити користувачів від небажаного спілкування та шахрайства;
- Може показати вам користувачів, які знаходяться поблизу вас, що робить його зручним для пошуку партнерів для побачень або зустрічей;

Недоліки:

- Заохочує поверхнєве ставлення до знайомств. Люди часто роблять вибір партнера на основі лише кількох фотографій і короткого опису;
- Багато функцій Tinder доступні лише за платну підписку;
- Недостатньо конкретна геолокація;
- Обмежена можливість редагування профілю;

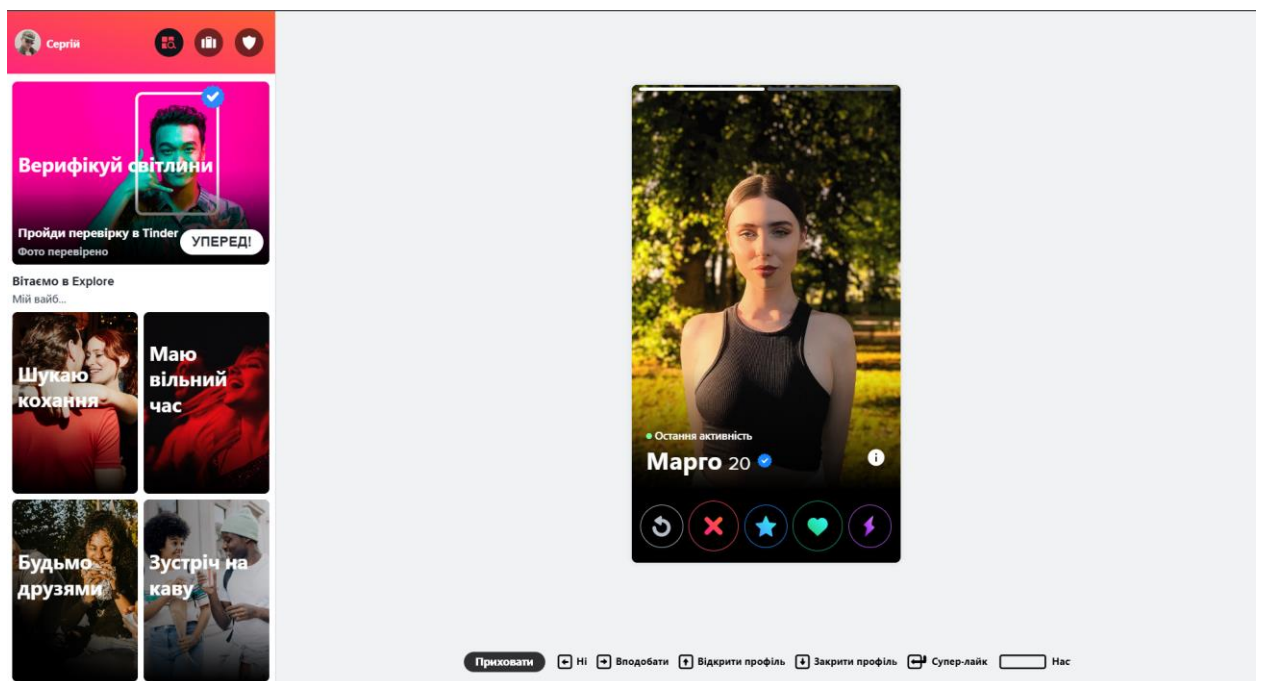


Рисунок 1.1 – Веб-додаток для знайомств «Tinder»

Іншим схожим додатком є Badoo, це сайт знайомств, у якого практично стільки ж переваг, скільки й недоліків [25]. Брак інформації в анкеті компенсується великою кількістю користувачів із різних країн, що буде корисно тим, хто хоче підтягнути іноземну мову з носієм. Цікавою функцією

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

програми є пошук свого двійника серед зовні схожих людей. Також можна ввімкнути місце розташування і подивитися, хто поруч.

Переваги:

- Можна використовувати безкоштовно, що робить його доступним для людей з будь-яким бюджетом.
- Має функцію показати вам користувачів, які знаходяться поблизу вас, що робить його зручним для пошуку партнерів для побачень або зустрічей;
- Має ряд функцій безпеки, які допомагають захистити користувачів від небажаного спілкування та шахрайства.

Недоліки:

- Брак інформації про користувачів що зумовлено обмеженою можливістю редагування профілю;
- Деякі функції доступні лише за платну підписку;
- Мале різноманіття фільтрів для рекомендацій;
- Функціонал обмежено за кількістю використань за певний час;

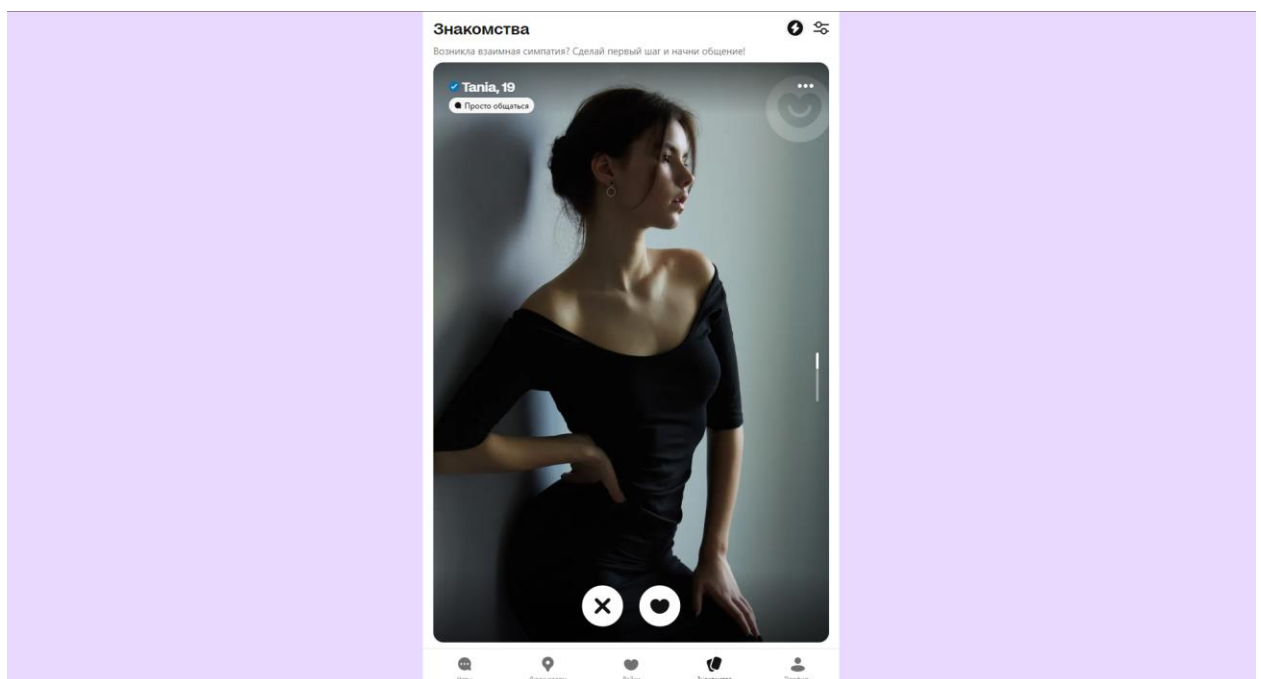


Рисунок 1.2 – Веб-додаток для знайомств «Badoo»

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Останнім аналогом є сайт Waytohey, це відкритий і всебічний сайт, який вітає усіх бажаючих стати його користувачем: як гетеро-, так і гомосексуалів, небінарних осіб і будь-яких інших представників ЛГБТК + спільноти [26]. Важливою метою на сьогодні є створення вільного від нетерпимості і упереджень простору. Тому Way to Hey розвивається як сайт, доступний абсолютно кожному, хто перебуває в пошуку, враховуючи різноманітність їх вподобань.

Переваги:

- Пропонує більш детальний профіль, ніж інші сайти знайомств, що дає вам можливість краще познайомитися з людьми, перш ніж зв'язатися з ними;
- Використовує систему відповідності, яка допомагає вам знайти людей, які поділяють ваші цінності та інтереси;
- Пропонує можливість віртуальних побачень, щоб ви могли познайомитися з людиною, перш ніж зустрітися з нею в реальному житті;
- Не має реклами, що робить його більш приємним для користування;

Недоліки:

- Платний сайт знайомств, тому вам потрібно буде оформити підписку, щоб використовувати його;
- У нього може менше функцій, ніж у інших сайтів знайомств;
- Не передбачає можливість його використання на мобільних пристроях;
- Не проводиться процедура затвердження фотографій користувачів, тому можливий високий відсоток появи фальшивих профілів з потенційно неприпустимими або рекламними зображеннями

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

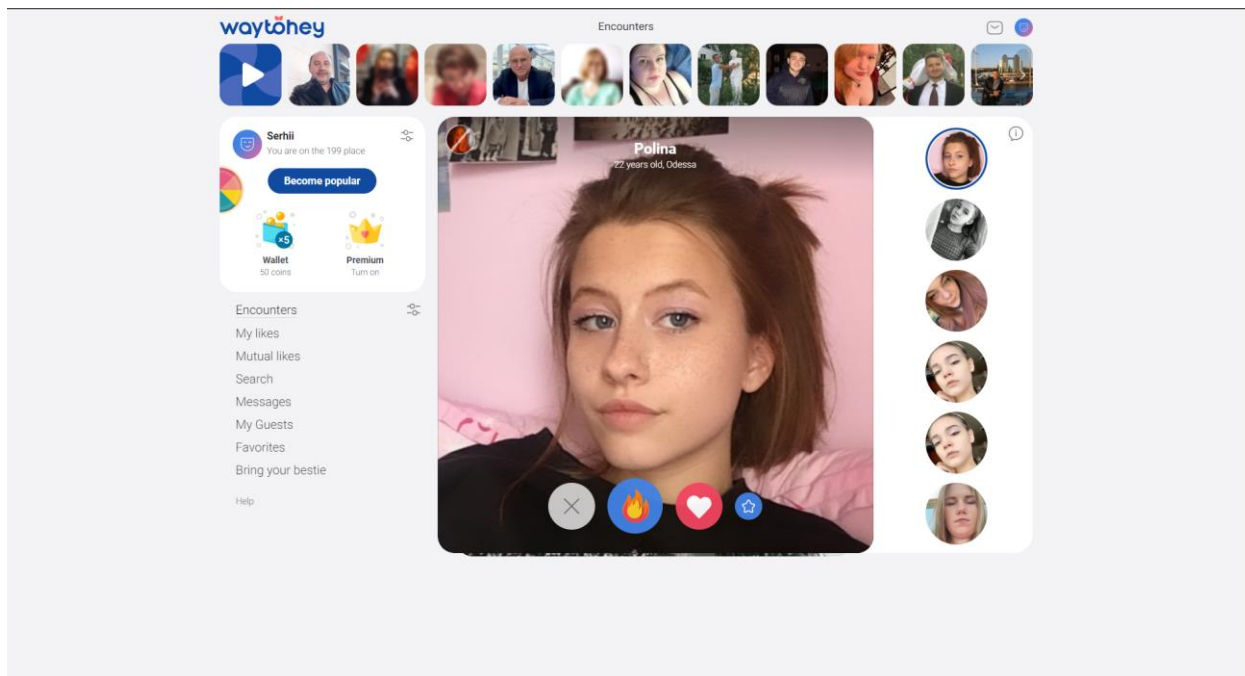


Рисунок 1.3 – Сайт для знайомств «Waytohey»

Порівнюючи Tinder, Badoo і Waytohey, можна сказати, що жоден з них не ідеальний для того, щоб знайти нових друзів.

Моя програма матиме низку переваг, завдяки яким вона виділиться серед конкурентів і запозичить у неї найкращі функції.

По-перше, для зменшення кількості фейкових профілів буде використана система перевірки.

По-друге, буде розроблена система модерації для запобігання нав'язливим повідомленням і образливій поведінці.

По-третє, пошукова система на основі інтересів, яка допомагає людям знаходити партнерів зі схожими цінностями та вподобаннями.

Нарешті, можливість проводити віртуальні події, які допомагають людям знайомитися з новими людьми.

1.3 Вибір архітектури веб-орієнтованої системи

Розробка веб-орієнтованої системи для знайомств є завданням, що вимагає глибокого розуміння потреб користувачів та ефективного використання сучасних технологій [16]. Створення такої платформи вимагає

уважного підходу до вибору архітектури, яка забезпечить не тільки зручність користувачам, але й ефективність системи в цілому [13].

Однією з ключових вимог до архітектури веб-системи для знайомств є масштабованість. З урахуванням того, що популярність таких платформ може стрімко зростати, важливо обрати архітектурний підхід, який дозволить ефективно масштабувати систему під ростуче навантаження [21].

Ще однією важливою характеристикою є висока продуктивність системи. Користувачі системи для знайомств очікують миттєвих реакцій та ефективної обробки запитів, тому вибір архітектури повинен сприяти оптимальному використанню ресурсів та швидкому виконанню завдань.

Забезпечення безпеки є ще однією важливою складовою. З огляду на особливості сервісів для знайомств, захист персональної інформації та запобігання шахрайству стають невід'ємною частиною архітектурного підходу.

Крім того, архітектура повинна бути гнучкою та легко розширюваною, щоб враховувати зміни в потребах користувачів та технологічній сфері. Важливо вибрати такий підхід, що дозволяє ефективно впроваджувати нові функціональності та адаптуватися до змін на ринку онлайн-знайомств.

Для того щоб детальніше описати своє програмне забезпечення було коротко описано про твердження: “Веб-додаток” та “API”, їхні типи.

Веб-додаток (web application) є програмним забезпеченням, призначеним для використання через веб-браузер. Він дозволяє користувачам здійснювати різноманітні дії, такі як перегляд вмісту, взаємодія з елементами інтерфейсу, або виконання конкретних завдань, не вимагаючи встановлення додатків на локальний комп'ютер. Веб-додатки можуть бути розгорнуті на веб-серверах та надавати доступ користувачам через мережу Інтернет або внутрішню корпоративну мережу.

Існують різні типи веб-додатків:

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

- Статичні веб-додатки: Вони відображають фіксовану інформацію та не мають можливості інтерактивної взаємодії з користувачем.
- Динамічні веб-додатки: Ці додатки можуть реагувати на дії користувача та динамічно оновлювати вміст без повного перезавантаження сторінки.

API (інтерфейс програмування застосунків) – це набір правил та протоколів, які визначають, як програми можуть взаємодіяти між собою. В контексті веб-додатків, API визначає, як різні компоненти програмного забезпечення можуть обмінюватися даними та функціональністю через мережу.

Існують різні типи API:

RESTful API (Representational State Transfer): Використовує принципи HTTP-протоколу для обміну даними між клієнтом і сервером. Часто використовується для взаємодії веб-додатків та мобільних додатків з серверами.

SOAP API (Simple Object Access Protocol): Використовує XML для обміну повідомленнями між різними системами. В основному використовується у корпоративних середовищах.

GraphQL API: Забезпечує більш гнучку взаємодію з сервером, дозволяючи клієнту визначати, які саме дані йому потрібні.

JSON-RPC та XML-RPC: Використовуються для віддаленого виклику процедур (RPC) між віддаленими системами.

Було проаналізовано три види архітектури такі як: монолітна, мікросервісна та безсерверна.

Монолітна архітектура є класичним методом розробки програмного забезпечення, де весь додаток створюється як єдина технологічна система. У такій архітектурі всі компоненти взаємодіють між собою, і розгортання відбувається на одному сервері або групі серверів.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Переваги монолітної архітектури включають його простоту в розробці та тестуванні, оскільки всі компоненти взаємодіють безпосередньо між собою. Це полегшує вирішення проблем. Крім того, управління монолітними додатками може бути менш складним, оскільки вони працюють на єдиній технологічній платформі.

Проте монолітні додатки можуть зазнавати проблем при масштабуванні через обмежену можливість розширення. Складна структура може ускладнити розробку та підтримку. Зокрема, ця архітектура найбільше підходить для невеликих та середніх проектів з стабільним функціоналом.

Отже, вибір між монолітною та мікросервісною архітектурою залежить від розміру та складності проекту. Моноліт може бути оптимальним для менших проектів, тоді як мікросервіси найбільше підходять для великих та зростаючих проектів, які потребують гнучкості та можливості масштабування.

Мікросервісна архітектура є альтернативою монолітному підходу, де весь додаток розглядається як одна система. У порівнянні з монолітом, мікросервіси є невеликими, слабо пов'язаними модулями, що можуть змінюватися незалежно.

Основна ідея мікросервісної архітектури полягає в тому, щоб розділити бізнес-логіку додатка на малі сервіси, які можна розгортати та використовувати незалежно. Це дозволяє ефективно випускати нові версії та оптимізувати релізи.

Мікросервіси – це невеликі, незалежні модулі, які можуть легко змінюватися. Кожен мікросервіс може відповідати за конкретну область функціоналу. Розглянемо розробку системи електронної комерції як приклад мікросервісної архітектури. Різні мікросервіси можуть відповідати за автентифікацію, кошик та білінг.

Переваги мікросервісів включають модульність, надійність, змінюваність, масштабованість та тестованість. Вони також дозволяють

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

незалежну роботу команд, що працюють над різними сервісами, і сприяють більшій увазі до якості.

Проте, існують і недоліки мікросервісної архітектури. Зростання витрат на інфраструктуру, додатковий рівень комунікації між командами та складніше середовище розробки стають завданнями, які варто врахувати.

Таким чином, вибір між мікросервісами та іншими архітектурними підходами повинен враховувати конкретні потреби та розмір проекту, а також готовність вирішувати додаткові завдання, пов'язані зі складністю та координацією.

Безсерверна архітектура, або архітектура "без сервера", представляє собою стратегію розробки програмного забезпечення, яка автоматизує процеси розгортання за допомогою хмарних технологій. Незважаючи на назву, вона не відмовляється від серверної частини, але вирізняється відсутністю прямого взаємодії з серверами як з окремими елементами інфраструктури.

В безсерверній архітектурі розробка та взаємодія між сервісами ставиться в центр уваги, а інфраструктурні аспекти обробляються постачальником хмарних послуг. Платиться лише за використання інфраструктури під час її активного використання.

Переваги безсерверної архітектури включають гнучкість, абстракцію від операційної системи, легкий поріг входу та надійність. Проте, існують і недоліки, такі як обмежена гнучкість управління масштабуванням, ускладнене налагодження взаємодії між компонентами, залежність від конкретного постачальника послуг та можливість каскадної відмови в системі.

Таким чином, вибір між безсерверною архітектурою та іншими підходами повинен залежати від конкретних потреб проекту, а також готовності вирішувати виникаючі завдання, пов'язані зі складністю та координацією.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Вибрано монолітну архітектуру через її простоту в розробці та тестуванні. RESTful API було вибрано для обміну даними між клієнтом і сервером.

1.4 Обґрунтування вибору інструментальних засобів

Обравши мову програмування PHP, я врахував її велику популярність та високу ефективність для веб-розробки [8]. PHP відома своєю простотою в освоєнні та широким спектром функцій, що полегшить розробку та підтримку веб-сайту.

Symfony став обраним фреймворком з урахуванням його репутації у веб-розробці [1]. Він надає структурну організацію коду та швидкість розробки завдяки великому набору готових компонентів. Використання Symfony спростить процес створення функціональностей сайту та забезпечить його стабільність.

MySQL обраний як СУБД з урахуванням його надійності та продуктивності [27]. Взаємодія PHP з MySQL вважається стандартом у веб-розробці, і це забезпечить оптимальну продуктивність та ефективність роботи з базою даних.

З огляду на те, що на старті передбачається малий обсяг користувачів, такий стек технологій є оптимальним, надійним та легким для управління. Такий вибір дозволить ефективно реалізувати основний функціонал сайту знайомств та легко масштабувати його в майбутньому при необхідності [19].

React JS обраний для реалізації клієнтської частини веб-сайту [7]. Це сучасна бібліотека JavaScript, яка дозволяє створювати швидкі та інтерактивні інтерфейси користувача. Використання React JS сприяє підвищенню продуктивності завдяки віртуальному DOM і компонентній архітектурі, що дозволяє легко керувати станом додатку та повторно використовувати компоненти.

Material-UI використовується як бібліотека компонентів для стилізації веб-сайту знайомств [2]. Вона надає набір готових компонентів, розроблених

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

відповідно до принципів Material Design, що дозволяє створювати сучасний та естетично привабливий інтерфейс користувача. Використання Material-UI сприяє прискоренню розробки, оскільки більшість стилів і компонентів вже готові до використання, а також забезпечує адаптивність дизайну для різних пристроїв.

Такий комплекс технологій (рис. 1.4) дозволяє створити веб-сайт знайомств із зручним та привабливим інтерфейсом, підтримкою адаптивного дизайну та ефективною взаємодією користувачів. Обраний стек забезпечить високу якість та ефективність розробки в умовах невеликої кількості користувачів на початковому етапі.



Рисунок 1.4 – Використані технології

Висновки до першого розділу

У першому розділі була розглянута розробка веб-орієнтованої платформи для знайомств, в якій визначалися основні етапи та завдання розробки, аналізувались аналоги програмного продукту та обиралась архітектура веб-системи.

У процесі постановки задачі визначення вимог та функцій сайту проводився аналіз ринку та конкурентів, а також встановлювались вимоги до інтерфейсу та зручності використання. Проектування дизайну сайту включало створення сучасного інтерфейсу та адаптацію його під різні розміри екранів.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Під час аналізу аналогів, таких як Tinder, Badoo та Waytohey, виявлено їхні переваги та недоліки. Цей аналіз вказав на важливі аспекти, які можна врахувати у власній розробці, такі як система верифікації, модерації та пошуку за інтересами.

При виборі архітектури враховувались ключові фактори, такі як масштабованість, продуктивність та безпека. Обрана монолітна архітектура та RESTful API визначають основу для надійного та ефективного взаємодії клієнта та серверу.

Отже, було створено чіткий план розробки платформи для знайомств, взявши до уваги найкращі практики сучасних сервісів, а також обрати оптимальну архітектуру для її реалізації. Застосування вивчених принципів та технік вказує на те, що розроблений продукт може стати ефективним та конкурентоздатним у сфері онлайн-знайомств.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ СИСТЕМИ САЙТУ ЗНАЙОМСТВ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Для забезпечення безперебійного функціонування та відповідності встановленим стандартам і вимогам, було сформульовано ключові вимоги до сайту для знайомств. Ці вимоги спрямовані на забезпечення надійності та ефективності роботи платформи, а також на дотримання всіх необхідних стандартів і вимог.

Бізнес вимоги

1. Ціль проєкту: Створення вебдодатка для ефективного пошуку нових знайомств завдяки аналізу відповідності профілів вимогам та комунікацією між користувачами.
2. Представлення проєкту: Вебдодаток, який надає зручний інтерфейс для створення, керування та відстеження профілів користувачів. Користувачі можуть реєструватися, створювати облікові записи, знаходити потенційних партнерів, призначати зустрічі та спілкуватися через систему.

Вимоги користувачів

1. Інтуїтивно зрозумілий інтерфейс користувача.
2. Можливість створювати та редагувати власний профіль.
3. Персоналізовані рекомендації для кожного користувача.
4. Функціонал для блокування небажаних користувачів.

Функціональні вимоги

1. Зручний інтерфейс.
2. Реєстрація та авторизація користувачів.
3. Персоналізовані рекомендації за вибраними критеріями.
4. Створення та редагування профілю користувача.
5. Завантаження та видалення фото у профілі.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

6. Чат для спілкування між користувачами.
7. Можливість блокування користувачів.
8. Забезпечення безпеки даних та конфіденційності.
9. Можливість відновлення паролю.
- 10.Адміністративна панель для модерації та управління контентом.

Нефункціональні вимоги

1. Безпека: Захист особистих даних користувачів.
2. Швидкодія: Оптимізована продуктивність та швидкий відгук системи на дії користувачів.
3. Сумісність: Підтримка різних браузерів.
4. Зручність: Простий та інтуїтивно зрозумілий інтерфейс для користувачів.
5. Масштабованість: Можливість розширення функціональностей та підтримка великої кількості користувачів.
6. Конфіденційність: Гарантія конфіденційності та безпеки даних користувачів.
7. Адміністративна зручність: Інтуїтивно зрозумілий інтерфейс адмін-панелі для ефективного управління.
8. Стабільність: Надійна робота системи без збоїв та помилок.

Аналіз потреб користувачів та визначення ключових ролей дозволили сформувати діаграми використання сайту для знайомств з боку користувача (рис. 2.1) та адміністратора (рис. 2.2).

Роль адміністратора є розширеною роллю користувача з додатковими правами. Адміністратор має змогу змінювати існуючі профілі. Крім того, адміністратор може змінювати, додавати та видаляти хобі та стилі життя.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		22



Рисунок 2.1 – Варіанти використання сайту з ролю Користувача

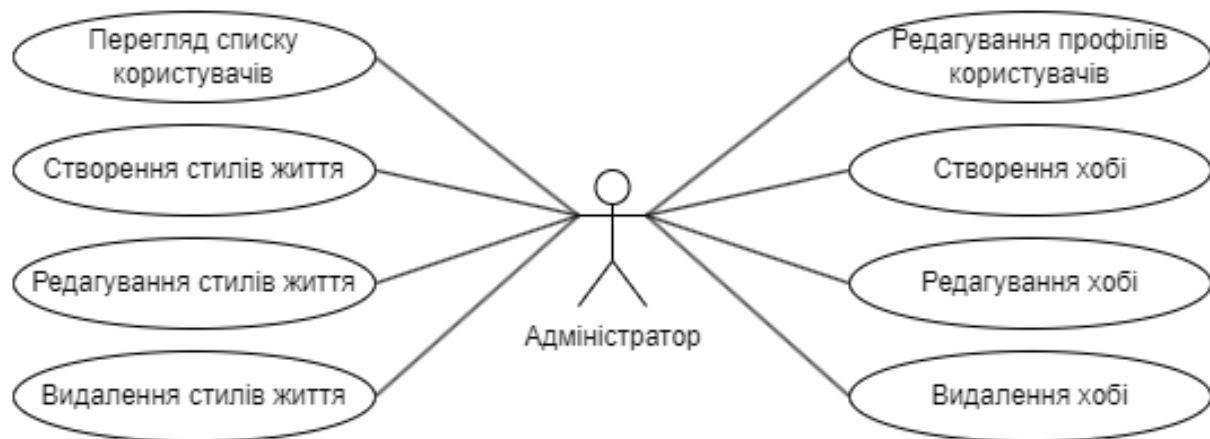


Рисунок 2.2 – Варіанти використання сайту з ролю Адміністратор

Прецедент: Реєстрація користувача

Опис: Реєстрація користувача та заповнення профілю для отримання повного доступу до функціоналу сайту.

Актори: Користувач.

Передумови: Користувач не зареєстрований.

Основний сценарій:

1. Користувач заповнює реєстраційні данні та напискає зареєструватись.
2. Користувач проходить перевірку пошти.
3. Користувач заповнює основну інформацію.
4. Користувач вибирає свій стиль життя.
5. Користувач вибирає свої хобі.
6. Користувач додає власні фото.
7. Система підтверджує активацію акаунту.

Альтернативний сценарій:

1. Користувач входить до акаунту з незаповненим профілем або не повністю заповненим профілем.
2. Користувач завершує заповнення профілю.
3. Система підтверджує активацію акаунту.

Прецедент: Налаштування рекомендацій

Опис: Налаштування рекомендацій для формування списку рекомендованих користувачів за параметрами.

Актори: Користувач.

Передумови: Користувач повинен мати активований акаунт.

Основний сценарій:

1. Користувач заходить на сторінку налаштувань рекомендацій.
2. Користувач вводить базові налаштування.
3. Користувач вибирає бажаний стиль життя.
4. Користувач вибирає бажані хобі.

Альтернативний сценарій:

1. Користувач заходить на сторінку налаштувань рекомендацій.
2. Користувач вводить базові налаштування.
3. Користувач вибирає бажаний стиль життя або хобі.

Прецедент: Додавання користувачів до вподобаних

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Опис: Додавання користувача до списку вподобаних з пропозицією створити чат.

Актори: Користувач.

Передумови: Користувач повинен мати активований акаунт та заповнені налаштування рекомендацій.

Основний сценарій:

1. Користувач заходить на сторінку рекомендацій.
2. Користувач переглядає картку рекомендованого користувача.
3. Користувач додає його до симпатій.
4. Система надсилає запрошення на створення спільного чату.

Альтернативний сценарій:

1. Користувач заходить на сторінку чорного списку.
2. Користувач переглядає картку користувача, якого він додавав до чорного списку.
3. Користувач додає його до симпатій.
4. Система надсилає запрошення на створення спільного чату.

Прецедент: Додавання користувачів до чорного списку

Опис: Додавання користувача до чорного списку з автоматичним заповненням причини блокування.

Актори: Користувач.

Передумови: Користувач повинен мати активований акаунт.

Основний сценарій:

1. Користувач заходить на сторінку рекомендацій.
2. Користувач переглядає картку рекомендованого користувача.
3. Користувач додає його до чорного списку.
4. Система заповнює причину додавання до чорного списку.

Альтернативний сценарій:

1. Користувач заходить на сторінку симпатій.
2. Користувач переглядає картку користувача, який його вподобав.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

3. Користувач додає його до чорного списку.
4. Система заповнює причину додавання до чорного списку.

Прецедент: Створення чату

Опис: Створення чату з користувачем який вподобав вас.

Актори: Користувач.

Передумови: Користувач повинен мати активований акаунт.

Основний сценарій:

1. Користувач заходить на сторінку симпатій.
2. Користувач переглядає картку користувача, який його вподобав.
3. Користувач приймає запрошення на створення чату.
4. Система створює чат між двома користувачами.

Альтернативний сценарій:

1. Користувач заходить на сторінку рекомендацій.
2. Користувач додає іншого користувача до симпатій.
3. Система надсилає запрошення на створення спільного чату.
4. Користувач якому надійшло запрошення приймає його.
5. Система створює чат між двома користувачами

Спираючись на дані, отримані та сформовані під час проведення аналізу вимог та способів використання програмного продукту, можна створити базову діаграму класів (рис. 2.3).

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

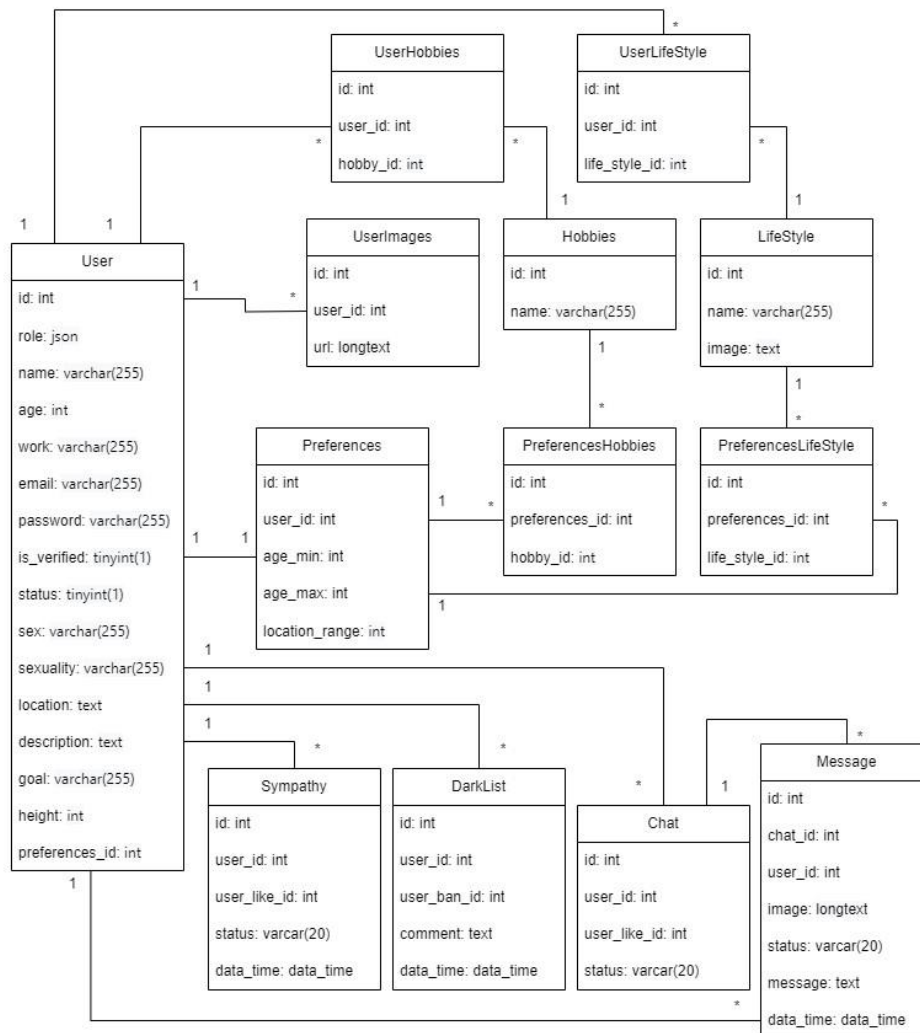


Рисунок 2.3 – Діаграма класів

На основі зібраної та опрацьованої інформації під час аналізу вимог та сценаріїв використання програмного продукту було створено базову діаграму класів. Нижче продемонстровано головні класи системи:

1. **User** – представляє користувача системи та містить інформацію про користувача, таку як ім'я, вік, робота, електронна пошта, пароль, роль, статус верифікації, стать, гендер, місцезнаходження, опис, цілі, зріст та статус активації. Користувач може взаємодіяти з хобі, стилями життя, зображеннями, симпатіями, чорним списком, чатами та повідомленнями.
2. **UserHobbies** – представляє хобі користувача та містить інформацію про користувача і його хобі, такі як ідентифікатори користувача та хобі.

3. **UserLifeStyle** – представляє стиль життя користувача та містить інформацію про користувача і його стиль життя, такі як ідентифікатори користувача та стилю життя.
4. **UserImages** – містить зображення користувачів та інформацію про користувача і його зображення, такі як ідентифікатор користувача та URL зображення.
5. **Hobbies** – містить інформацію про всі можливі хобі, такі як назва.
6. **LifeStyle** – містить інформацію про всі можливі стилі життя, такі як назва та зображення.
7. **Preferences** – містить інформацію про вподобання користувачів, такі як ідентифікатор користувача, мінімальний та максимальний вік, діапазон місцезнаходження.
8. **PreferencesHobbies** – представляє вподобання користувача щодо хобі та містить інформацію про ідентифікатор вподобань і хобі.
9. **PreferencesLifeStyle** – представляє вподобання користувача щодо стилів життя та містить інформацію про ідентифікатор вподобань і стиль життя.
10. **Sympathy** – представляє симпатії користувачів та містить інформацію про користувача, об'єкт симпатії, статус та дату/час.
11. **DarkList** – представляє чорний список користувача та містить інформацію про користувача, заблокованого користувача, коментар та дату/час.
12. **Chat** – представляє чати між користувачами та містить інформацію про користувача, об'єкт симпатії, статус та дату/час.
13. **Message** – містить інформацію про повідомлення в чатах, такі як ідентифікатор чату, користувача, зображення, статус, текст повідомлення та дату/час.

Зв'язки між класами:

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

- Клас **User** має зв'язок "один до багатьох" з класами **UserHobbies**, **UserLifeStyle**, **UserImages**, **Preferences**, **Sympathy**, **DarkList**, **Chat**, **Message**.
- Клас **Hobbies** має зв'язок "один до багатьох" з класами **UserHobbies** та **PreferencesHobbies**.
- Клас **LifeStyle** має зв'язок "один до багатьох" з класами **UserLifeStyle** та **PreferencesLifeStyle**.
- Клас **Preferences** має зв'язок "один до одного" з класом **User**, один до багатьох з класами **PreferencesHobbies** та **PreferencesLifeStyle**.
- Клас **Sympathy** має зв'язок "багато до одного" з класом **User**.
- Клас **DarkList** має зв'язок "багато до одного" з класом **User**.
- Клас **Chat** має зв'язок "багато до одного" з класом **User**.
- Клас **Message** має зв'язок "багато до одного" з класом **Chat**.

2.2 Розробка бази даних системи

Розгляд бази даних системи знайомств – це невід'ємна складова вебдодатку [9]. Вона включає в себе чотирнадцять таблиць:

- Користувачі (Users)
- Хобі користувачів (UserHobbies)
- Спосіб життя користувачів (UserLifeStyle)
- Зображення користувачів (UserImages)
- Хобі (Hobbies)
- Спосіб життя (LifeStyle)
- Вподобання (Preferences)
- Вподобання щодо хобі (PreferencesHobbies)
- Вподобання щодо стилю життя (PreferencesLifeStyle)
- Симпатії (Sympathy)
- Чорний список (DarkList)
- Чати (Chat)
- Повідомлення (Message)

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

Кожна таблиця охоплює різні аспекти системи.

Таблиця «Користувачі» – (табл. 2.1), є ключовим елементом бази даних вебдодатку системи знайомств, що зберігає інформацію про користувачів, включаючи їхні особисті дані, контактні дані, вподобання та інші важливі характеристики, які визначають профілі користувачів.

Таблиця 2.1.

Структура таблиці "Користувачі"

Поле	Тип даних	ПК	ЗК	Призначення
id	INTEGER	+	+	Ідентифікатор користувача
role	JSON	-	-	Роль користувача
name	VARCHAR(255)	-	-	Ім'я користувача
age	INT	-	-	Вік користувача
work	VARCHAR(255)	-	-	Місце роботи
email	VARCHAR(255)	-	+	Електронна пошта
password	VARCHAR(255)	-	-	Пароль
is_verified	TINYINT(1)	-	-	Статус верифікації
status	TINYINT(1)	-	-	Статус користувача
sex	VARCHAR(255)	-	-	Стать користувача
sexuality	VARCHAR(255)	-	-	Сексуальна орієнтація
location	TEXT	-	-	Локація
description	TEXT	-	-	Опис
goal	VARCHAR(255)	-	-	Мета знайомств
height	INT	-	-	Зріст користувача
preferences_id	INT	-	+	Ідентифікатор вподобань

Таблиця «Хобі користувачів» – (табл. 2.2), відображає зв'язок між користувачами та їхніми хобі, що дозволяє зберігати інформацію про інтереси користувачів та сприяти більш точному підбору відповідних пар для знайомств.

Таблиця 2.2.

Структура таблиці "Хобі користувачів"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор хобі користувача
user_id	INT	-	+	Ідентифікатор користувача
hobby_id	INT	-	+	Ідентифікатор хобі

Таблиця «Спосіб життя користувачів» – (табл. 2.3), містить дані про звички та стиль життя користувачів, що допомагає у формуванні більш детального профілю користувача і підвищує точність відповідності в системі знайомств.

Таблиця 2.3.

Структура таблиці "Спосіб життя"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор способу життя
user_id	INT	-	+	Ідентифікатор користувача
life_style_id	INT	-	+	Ідентифікатор способу життя

Таблиця «Зображення користувачів» – (табл. 2.4), зберігає URL-адреси зображень профілів користувачів, що дозволяє відображати фотографії у профілях та під час пошуку пар.

Таблиця 2.4.

Структура таблиці "Зображення користувачів"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор зображення
user_id	INT	-	+	Ідентифікатор користувача
url	LONGTEXT	-	-	URL зображення

Таблиця «Хобі» – (табл. 2.5), містить інформацію про можливі хобі, які можуть бути обрані користувачами, створюючи базу даних для відображення інтересів користувачів.

Таблиця 2.5.

Структура таблиці "Хобі"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор хобі
name	VARCHAR(255)	-	-	Назва хобі

Таблиця «Спосіб життя» – (табл. 2.6), зберігає перелік можливих стилів життя, які можуть бути пов'язані з користувачами, що допомагає у формуванні відповідного профілю та підбору пар.

Таблиця 2.6.

Структура таблиці "Спосіб життя"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор способу життя
name	VARCHAR(255)	-	-	Назва способу життя
image	TEXT	-	-	Зображення

Таблиця «Вподобання» – (табл. 2.7), містить інформацію про вподобання користувачів щодо віку, локації та інших параметрів, що сприяє точному підбору пар для знайомств.

Таблиця 2.7.

Структура таблиці "Вподобання"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор вподобань
user_id	INT	-	+	Ідентифікатор користувача
age_min	INT	-	-	Мінімальний вік
age_max	INT	-	-	Максимальний вік
location_range	INT	-	-	Діапазон локації

Таблиця «Вподобання щодо хобі» – (табл. 2.8), зберігає зв'язок між вподобаннями користувачів та хобі, що допомагає системі більш точно підбирати пари за інтересами.

Таблиця 2.8.

Структура таблиці "Вподобання щодо хобі"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор вподобань щодо хобі
preferences_id	INT	-	+	Ідентифікатор вподобань
hobby_id	INT	-	+	Ідентифікатор хобі

Таблиця «Вподобання щодо стилю життя» – (табл. 2.9), відображає зв'язок між вподобаннями користувачів та стилями життя, забезпечуючи більш детальний підбір пар на основі спільних життєвих уподобань.

Таблиця 2.9.

Структура таблиці "Вподобання щодо стилю життя"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор вподобань щодо стилю життя
preferences_id	INT	-	+	Ідентифікатор вподобань
life_style_id	INT	-	+	Ідентифікатор стилю життя

Таблиця «Симпатії» – (табл. 2.10), містить дані про симпатії між користувачами, що дозволяє фіксувати взаємний інтерес та відображати можливі пари для знайомств.

Таблиця 2.10.

Структура таблиці "Симпатії"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор симпатії
user_id	INT	-	+	Ідентифікатор користувача
user_like_id	INT	-	+	Ідентифікатор користувача, якого подобають

status	VARCHAR(20)	-	-	Статус симпатії
data_time	DATETIME	-	-	Дата та час симпатії

Таблиця «Чорний список» – (табл. 2.11), зберігає інформацію про користувачів, які були додані до чорного списку іншими користувачами, забезпечуючи можливість блокування небажаних контактів.

Таблиця 2.11.

Структура таблиці "Чорний список"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор у чорному списку
user_id	INT	-	+	Ідентифікатор користувача
user_ban_id	INT	-	+	Ідентифікатор користувача, що заборонений
comment	TEXT	-	-	Коментар
data_time	DATETIME	-	-	Дата та час додавання

Таблиця «Чати» – (табл. 2.12), містить дані про чати між користувачами, що дозволяє зберігати історію повідомлень та взаємодії між користувачами в рамках системи.

Таблиця 2.12.

Структура таблиці "Чати"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор чату
user_id	INT	-	+	Ідентифікатор користувача
user_like_id	INT	-	+	Ідентифікатор користувача, якого подобають
status	VARCHAR(20)	-	-	Статус чату

Таблиця «Повідомлення» – (табл. 2.13), зберігає всі повідомлення, відправлені між користувачами в чатах, включаючи текстові повідомлення та зображення, забезпечуючи зручну комунікацію в системі.

Таблиця 2.13.

Структура таблиці "Повідомлення"

Поле	Тип даних	ПК	ЗК	Призначення
id	INT	+	+	Ідентифікатор повідомлення
chat_id	INT	-	+	Ідентифікатор чату
image	LONGTEXT	-	-	Зображення
status	VARCHAR(20)	-	-	Статус повідомлення
message	TEXT	-	-	Текст повідомлення
data_time	DATETIME	-	-	Дата та час повідомлення

Загалом, структура бази даних сайту знайомств охоплює ключові сутності користувачів, уподобань, хобі, стилів життя та повідомлень, що

забезпечує ефективне зберігання та обробку даних, необхідних для функціонування системи. Нижче наведено ER-діаграму бази даних (рис. 2.4).

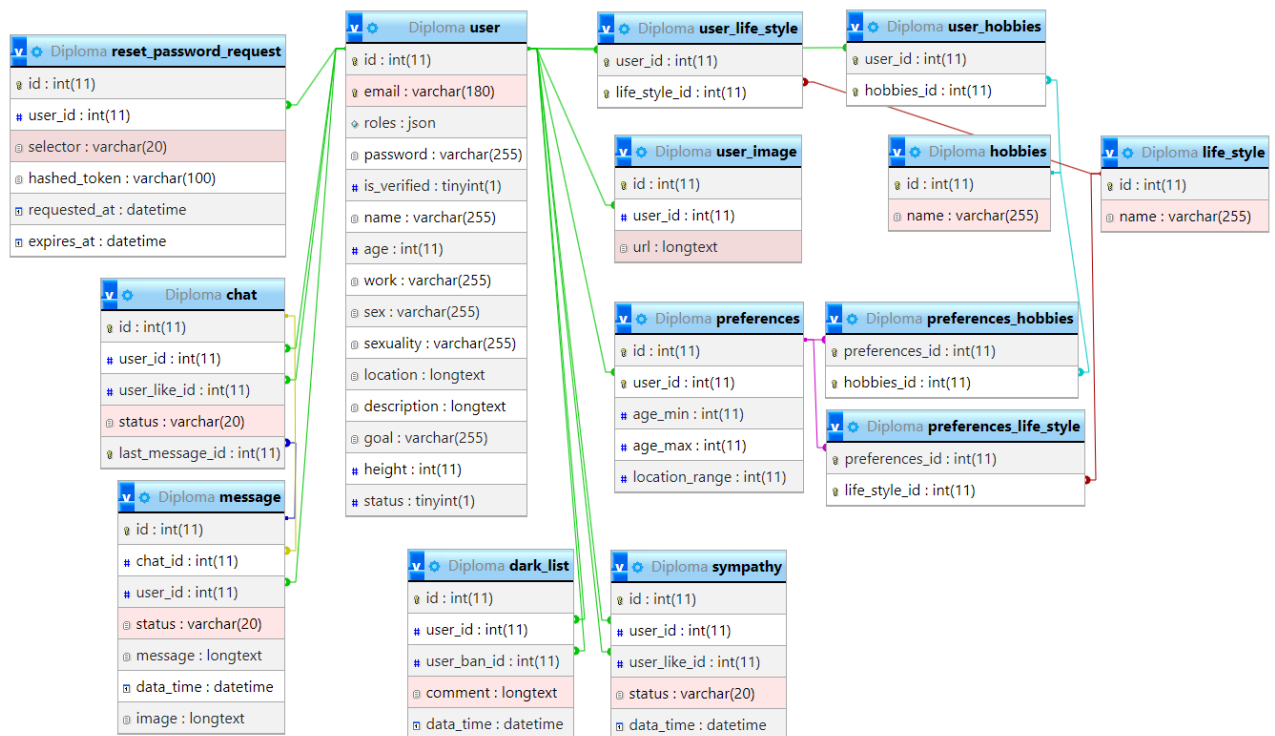


Рисунок 2.4 – ER-діаграма бази даних

2.3 Проєктування та реалізація алгоритмів роботи системи

Потрапивши на сайт, користувач бачить сторінку авторизації, з якої він може перейти до свого акаунту або перейти на сторінку реєстрації. Також з цієї сторінки користувач може перейти до відновлення паролю.

Одною з найважливіших функцій сайту є процес реєстрації та заповнення даних профілю (рис. 2.5), яка містить в собі декілька кроків. Спочатку користувач повинен заповнити на сторінці реєстрації свою пошту та пароль після чого йому потрібно підтвердити свою пошту через лист, який йому надійде. Натиснувши на кнопку підтвердження пошти, користувача перенаправить на сторінку профілю. На цій сторінці користувач повинен пройти п'ять кроків заповнення профілю. Спочатку заповнюється основна інформація про користувача, така як ім'я, вік, ріст, стать, гендер. Після цього користувач вибирає власний стиль життя та хобі. Наступним пунктом користувач дає доступ до місцезнаходження та заповнює цілі використання

сайту. І нарешті додає свої власні фото до профілю. Перевіривши те, що всі необхідні данні заповнені система активує акаунт.

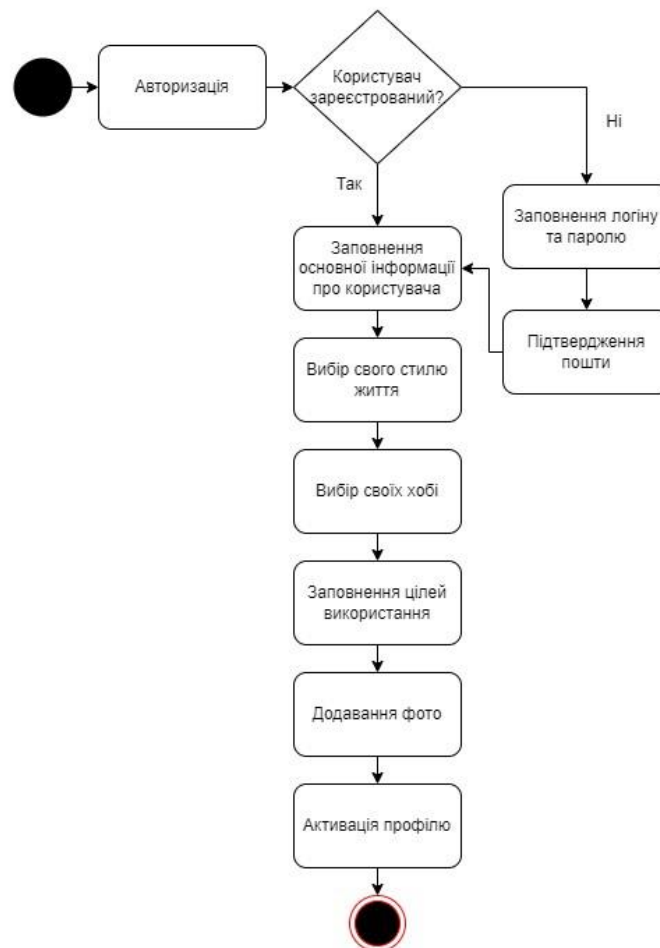


Рисунок 2.5 – Процес реєстрації та заповнення даних профілю

Користувач починає з заповнення налаштувань рекомендацій на спеціальній формі. Він обирає бажаний вік потенційних партнерів та діапазон пошуку, їх хобі та стилі життя, щоб отримувати рекомендації відповідно до цих параметрів.

Після налаштування рекомендацій користувач переходить на сторінку з рекомендаціями. Тут відображаються картки користувачів з фотографіями та основною інформацією про них. На кожній картці є кнопки для додавання користувача до симпатій (тобто відправлення запрошення в чат) або до чорного списку.

Якщо користувач додає когось до вподобань, він може перейти на сторінку вподобань, де видно користувачів, які вподобали його, або тих, кого

вподобав він. Якщо обидва користувачі вподобали один одного, то система створює між ними чат. Якщо користувач вподобав когось, він також має можливість скасувати своє вподобання, додавши користувача до чорного списку.

Також є окрема сторінка чорного списку, де користувач може переглянути всіх, кого він додав до цього списку, та видалити когось із нього за потреби, додавши його до вподобань.

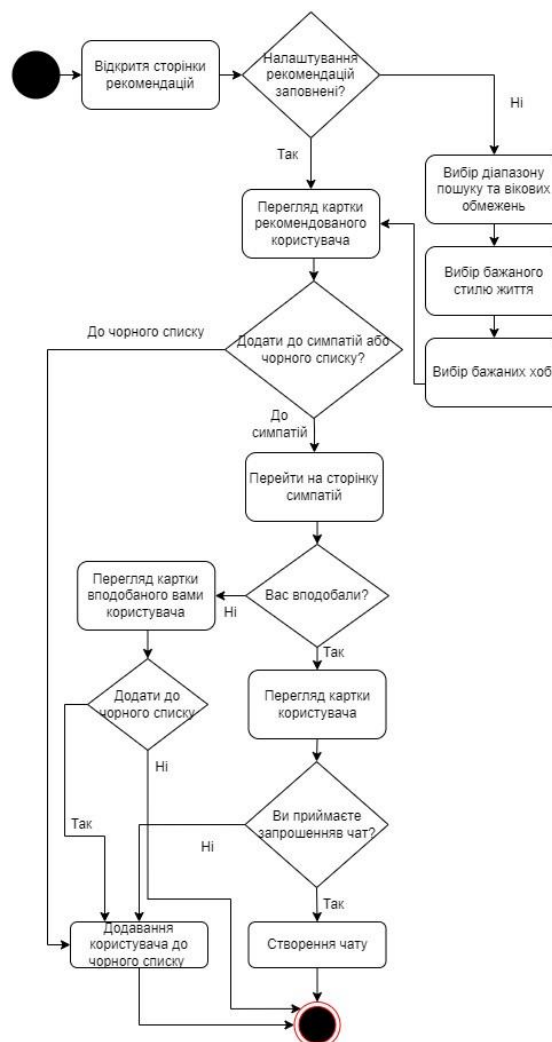


Рисунок 2.6 – Процес створення чату між користувачами

2.4 Реалізація функціоналу сайту знайомств

Архітектура проекту ґрунтується на використанні фреймворку Symfony, що гарантує надійність та гнучкість завдяки модульній структурі та застосуванню перевірених підходів розробки. Основні компоненти включають

архітектурний патерн MVC, різні бандли Symfony (зокрема Doctrine) та інтеграцію з React.js для фронтенд-частини.

Архітектурний патерн MVC розділяє логіку додатку на Model (Модель), що відповідає за дані та бізнес-логіку з використанням Doctrine ORM; View (Вид), який відображає дані користувачам (фронтенд на React.js); та Controller (Контролер), що обробляє запити користувачів та взаємодіє з моделлю у Symfony.

Крім того, використовуються бандли Symfony, такі як Symfony Security (для аутентифікації та авторизації), EasyAdmin (для створення адміністративної панелі) та Mercure (для реалізації завантаження повідомлень у чаті в реальному часі). Інтеграція з React.js відбувається через Webpack Encore. Структура бази даних включає основні сутності для сайту знайомств, що забезпечує ефективне зберігання та обробку даних.

Алгоритм рекомендацій базується на виборі користувачів для поточного користувача на основі його уподобань. Спочатку метод `index` отримує поточного користувача та його уподобання. Користувачеві уподобання включають мінімальний та максимальний вік бажаних партнерів, колекцію хобі та стилів життя, а також місцезнаходження. З місцезнаходження витягуються координати (широта і довгота):

```
$user = $this->getUser();
$preferences = $user->getPreferences();
$ageMin = $preferences->getAgeMin();
$ageMax = $preferences->getAgeMax();
$hobbiesCollection = $preferences->getHobbies();
$location = $user->getLocation();
$locationRange = $preferences->getLocationRange();
list($lat1, $lon1) = explode(",", $location, 2);
```

Хобі та стилі життя конвертуються в масиви ідентифікаторів для подальшої обробки:

```
$hobbiesArray = $hobbiesCollection->map(function($hobby) {
    return $hobby->getId();
})->toArray();
$lifestylesCollection = $preferences->getLifestyles();
$lifestylesArray = $lifestylesCollection->map(function($lifestyle) {
    return $lifestyle->getId();
})->toArray();
```

Потім репозиторій користувачів викликає метод `findByAgeHobbiesAndLifestyles`, який повертає список користувачів, що відповідають віковому діапазону, хобі та стилям життя, виключаючи поточного користувача:

```
$users = $userRepository->findByAgeHobbiesAndLifestyles($ageMin, $ageMax, $user->getId(), $hobbiesArray, $lifestylesArray);
```

Після отримання списку користувачів, алгоритм перевіряє, чи користувач, який рекомендується, не знаходиться у списку симпатій або у чорному списку поточного користувача. Для цього використовуються репозиторії симпатій та чорного списку:

```
$data = [];
foreach ($users as $recommendation) {
    if (!$doctrine->getRepository(Sympathy::class)->isUserInSympathy($user->getId(), $recommendation->getId())
        && !$doctrine->getRepository(DarkList::class)->isUserInDarkList($user->getId(), $recommendation->getId())) {
```

Крім того, перевіряється, чи відстань між поточним користувачем та користувачем, який рекомендується, не перевищує допустимий діапазон місцезнаходження. Відстань розраховується за допомогою функції `haversineDistance`, яка обчислює відстань між двома точками за їх координатами:

```
list($lat2, $lon2) = explode(",", $recommendation->getLocation(), 2);
if ($this->haversineDistance($lat1, $lon1, $lat2, $lon2) <= $locationRange) {
    $data[] = $recommendation;
}
}
```

Якщо всі умови виконуються, користувач додається до списку рекомендацій. Наприкінці, список користувачів нормалізується для JSON-відповіді, включаючи тільки певні атрибути, такі як ідентифікатор, електронну пошту, ім'я, вік, робота, стать, сексуальна орієнтація, місцезнаходження, опис, мета, зріст, стилі життя, хобі та зображення користувача:

```
$data = $serializer->normalize($data, null, [
    AbstractNormalizer::ATTRIBUTES => [
        'id', 'email', 'name', 'age', 'work', 'sex', 'sexuality', 'location', 'description', 'goal',
        'height', 'lifeStyles' => ['name'], 'Hobbies' => ['name'], 'userImages' => ['url']
    ]
]);
```

Результат повертається як JSON-відповідь:

```
return new JsonResponse($data);
```

Код даного методу наведено у додатку А.

Алгоритм пошуку користувачів для рекомендацій розпочинається з формування запиту, який буде об'єкт запиту для вибору користувачів. Спершу встановлюються основні критерії віку, виключення поточного користувача і додаються відповідні параметри для мінімального віку, максимального віку та ідентифікатора користувача. Це реалізується за допомогою методу `createQueryBuilder('u')`, де встановлюються умови `where` для мінімального і максимального віку, а також виключення поточного користувача:

```
$qb = $this->createQueryBuilder('u')
->where('u.age >= :minAge')
->andWhere('u.age <= :maxAge')
->andWhere('u.id != :userId')
->setParameter('minAge', $minAge)
->setParameter('maxAge', $maxAge)
->setParameter('userId', $userId);
```

Далі, якщо масив хобі не порожній, алгоритм приєднує таблицю хобі до запиту, перевіряючи наявність хобі у користувачів, і встановлює відповідні параметри. Це досягається додаванням `leftJoin` для хобі та встановленням параметра `hobbies`:

```
if (!empty($hobbies)) {
    $qb->leftJoin('u.Hobbies', 'h', Join::WITH, 'h.id IN (:hobbies)')
    ->setParameter('hobbies', $hobbies);
}
```

Аналогічно, якщо масив стилів життя не порожній, додається таблиця стилів життя та встановлюються відповідні параметри. Це реалізовано наступним чином:

```
if (!empty($lifestyles)) {
    $qb->leftJoin('u.lifeStyles', 'l', Join::WITH, 'l.id IN (:lifestyles)')
    ->setParameter('lifestyles', $lifestyles);
}
```

Потім алгоритм отримує поточного користувача з бази даних для визначення його статі, сексуальної орієнтації та цілі. На основі цих значень

додаються додаткові умови до запиту. Виклик методу find отримує поточного користувача, після чого визначаються його характеристики:

```
$currentUser = $this->find($userId);  
$currentUserSex = $currentUser->getSex();  
currentUserSexuality = $currentUser->getSexuality();  
$currentUserGoal = $currentUser->getGoal();
```

Якщо користувач зацікавлений у серйозних стосунках, алгоритм додає умови для пошуку користувачів з протилежною статтю та гетеросексуальною орієнтацією, або тією ж статтю та гомосексуальною орієнтацією, залежно від статі і орієнтація поточного користувача. Також додається умова, що ціль користувача має відповідати серйозним стосункам:

```
if ($currentUserGoal === 'Серйозні стосунки') {  
    if ($currentUserSex === 'Жінка' && $currentUserSexuality === 'Гетеросексуал(ка)') {  
        $qb->andWhere('u.sex = :male')  
        ->andWhere('u.sexuality = :heterosexual')  
        ->setParameter('male', 'Чоловік')  
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');  
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality === 'Гетеросексуал(ка)')  
    {  
        $qb->andWhere('u.sex = :female')  
        ->andWhere('u.sexuality = :heterosexual')  
        ->setParameter('female', 'Жінка')  
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');  
    } elseif ($currentUserSex === 'Жінка' && $currentUserSexuality === 'Гомосексуаліст(ка)')  
    {  
        $qb->andWhere('u.sex = :female')  
        ->andWhere('u.sexuality = :homosexual')  
        ->setParameter('female', 'Жінка')  
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');  
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality ===  
'Гомосексуаліст(ка)') {  
        $qb->andWhere('u.sex = :male')  
        ->andWhere('u.sexuality = :homosexual')  
        ->setParameter('male', 'Чоловік')  
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');  
    }  
}  
if ($currentUserGoal === 'Серйозні стосунки') {  
    $qb->andWhere('u.goal = :serious')  
    ->setParameter('serious', 'Серйозні стосунки');  
}
```

Якщо ціль користувача — несерйозні стосунки або короткостроковий роман, алгоритм застосовує аналогічні умови для статі та сексуальної орієнтації, а також додає умову для відповідної цілі:

```

if ($currentUserGoal === 'Несерйозні стосунки') {
    if ($currentUserSex === 'Жінка' && $currentUserSexuality === 'Гетеросексуал(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality === 'Гетеросексуал(ка)')
    {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Жінка' && $currentUserSexuality === 'Гомосексуаліст(ка)')
    {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality ===
'Гомосексуаліст(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    }
}
}
if ($currentUserGoal === 'Несерйозні стосунки') {
    $qb->andWhere('u.goal = :casual')
    ->setParameter('casual', 'Несерйозні стосунки');
}

```

Також для короткострокового роману та нових друзів додаються відповідні умови:

```

if ($currentUserGoal === 'Короткостроковий роман') {
    if ($currentUserSex === 'Жінка' && $currentUserSexuality === 'Гетеросексуал(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality === 'Гетеросексуал(ка)')
    {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Жінка' && $currentUserSexuality === 'Гомосексуаліст(ка)')
    {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    }
}

```

```

    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality ===
'Гомосексуаліст(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    }
}
if ($currentUserGoal === 'Короткостроковий роман') {
    $qb->andWhere('u.goal = :shortTerm')
    ->setParameter('shortTerm', 'Короткостроковий роман');
}

if ($currentUserGoal === 'Нові друзі') {
    $qb->andWhere('u.goal = :newFriends')
    ->setParameter('newFriends', 'Нові друзі');
}

```

Після встановлення всіх умов, алгоритм додає до запиту підрахунок кількості спільних хобі та стилів життя, а також загальної кількості спільних інтересів. Користувачі групуються за ідентифікатором і сортуються за кількістю спільних інтересів у порядку спадання:

```

$qqb->addSelect('COUNT(DISTINCT h.id) AS HIDDEN hobbiesCount')
->addSelect('COUNT(DISTINCT l.id) AS HIDDEN lifestylesCount')
->addSelect('COUNT(DISTINCT h.id) + COUNT(DISTINCT l.id) AS HIDDEN
totalCommonCount')
->groupBy('u.id')
->orderBy('totalCommonCount', 'DESC');

```

Наприкінці запит виконується, і результати повертаються як масив користувачів, які відповідають усім встановленим критеріям:

```
return $qb->getQuery()->getResult();
```

Код даного методу наведено у додатку А.

Алгоритм додавання користувача до чорного списку починається з визначення маршруту addDarkList, який обробляє POST-запит на додавання користувача до чорного списку за його ідентифікатором. Викликається метод addDarkList, який приймає об'єкти ManagerRegistry, UserRepository і ідентифікатор користувача. Спочатку отримується менеджер сутностей для подальшої роботи з базою даних:

```
$entityManager = $doctrine->getManager();
```

Далі отримується поточний користувач і користувач, якого необхідно додати до чорного списку. Поточного користувача отримують через метод

getUser, а користувача для додавання до чорного списку знаходять за допомогою репозиторію користувачів за його ідентифікатором:

```
$user = $this->getUser();  
$userBan = $userRepository->find($id);
```

Потім перевіряється, чи користувач вже знаходиться в чорному списку, і чи є користувач у списку симпатій поточного користувача. Для цього використовуються відповідні методи репозиторіїв:

```
$darkLists = $doctrine->getRepository(DarkList::class)->isUserInDarkList(  
    $user->getId(),  
    $userBan->getId()  
);  
$sympathy = $doctrine->getRepository(Sympathy::class)->isUserInSympathy(  
    $user->getId(),  
    $userBan->getId()  
);
```

Якщо користувач вже знаходиться в чорному списку, повертається відповідь з цими даними:

```
if ($darkLists) {  
    return new JsonResponse($darkLists);  
}
```

Якщо користувач знаходиться в списку симпатій, цей запис видаляється з бази даних:

```
if ($sympathy) {  
    $entityManager->remove($sympathy);  
    $entityManager->flush();  
}
```

Далі створюється новий запис в чорному списку. Встановлюються поточний користувач, користувач, якого додають до чорного списку, коментар, і поточний час. Ці дані зберігаються в базі даних за допомогою методу persist і flush менеджера сутностей:

```
$darkList = new DarkList();  
$darkList->setUser($user);  
$darkList->setUserBan($userBan);  
$darkList->setComment("НЕ СПОДОБАВСЯ");  
$darkList->setDateTime(new \DateTimeImmutable());  
$entityManager->persist($darkList);  
$entityManager->flush();
```

Наприкінці повертається відповідь з новим записом в чорному списку:

```
return new JsonResponse($darkList);
```

Код даного методу наведено у додатку А.

Алгоритм додавання користувача до симпатій починається з визначення маршруту addSympathy, який обробляє POST-запит на додавання користувача до списку симпатій за його ідентифікатором. Викликається метод addSympathy, який приймає об'єкти ManagerRegistry, UserRepository і ідентифікатор користувача. Спочатку отримується менеджер сутностей для подальшої роботи з базою даних:

```
$entityManager = $doctrine->getManager();
```

Далі отримується поточний користувач і користувач, якого необхідно додати до симпатій. Поточного користувача отримують через метод getUser, а користувача для додавання до симпатій знаходять за допомогою репозиторію користувачів за його ідентифікатором:

```
$user = $this->getUser();
```

```
$userLike = $userRepository->find($id);
```

Потім перевіряється, чи користувач вже знаходиться у списку симпатій, і чи є користувач у чорному списку поточного користувача. Для цього використовуються відповідні методи репозиторіїв:

```
$sympathy = $doctrine->getRepository(Sympathy::class)->isUserInSympathy(
    $user->getId(),
    $userLike->getId()
);
```

```
$darkLists = $doctrine->getRepository(DarkList::class)->isUserInDarkList(
    $user->getId(),
    $userLike->getId()
);
```

Якщо користувач вже знаходиться у списку симпатій, повертається відповідь з цими даними:

```
if ($sympathy) {
    return new JsonResponse($sympathy);
}
```

Якщо користувач знаходиться у чорному списку, цей запис видаляється з бази даних:

```
if ($darkLists) {
    $entityManager->remove($darkLists);
    $entityManager->flush();
}
```

Далі створюється новий запис у списку симпатій. Встановлюються поточний користувач, користувач, якого додають до симпатій, статус "В

ОЧІКУВАННІ", і поточний час. Ці дані зберігаються в базі даних за допомогою методу persist і flush менеджера сутностей:

```
$sympathy = new Sympathy();  
$sympathy->setUser($user);  
$sympathy->setUserLike($userLike);  
$sympathy->setStatus("В ОЧІКУВАННІ");  
$sympathy->setDateTime(new \DateTimeImmutable());  
$entityManager->persist($sympathy);  
$entityManager->flush();
```

Наприкінці повертається відповідь з новим записом у списку симпатій:

```
return new JsonResponse($sympathy);
```

Код даного методу наведено у додатку А.

Алгоритм створення чату починається з визначення маршруту createChat, який обробляє запити на створення нового чату з користувачем, ідентифікатор якого передається у запиті. Метод createChat приймає об'єкти ManagerRegistry, Request і ідентифікатор користувача. Спочатку отримується менеджер сутностей для роботи з базою даних:

```
$entityManager = $doctrine->getManager();
```

Далі витягується користувач, з яким буде створюватися чат, за допомогою репозиторію користувачів, а також поточний користувач:

```
$otherUser = $doctrine->getRepository(User::class)->find($id);  
$user = $this->getUser();
```

Якщо користувач, з яким планується створення чату, не знайдений, викидається виняток:

```
if (is_null($otherUser)) {  
    throw new \Exception("Користувача не знайдено");  
}
```

Якщо користувач намагається створити чат з самим собою, викидається виняток:

```
if ($otherUser->getId() === $this->getUser()->getId()) {  
    throw new \Exception("Це глибоко, але ви не можете створити розмову з самим собою");  
}
```

Потім перевіряється, чи існує вже чат між поточним користувачем і користувачем, з яким планується створення чату. Для цього використовується відповідний метод репозиторію чатів:

```
$chat = $doctrine->getRepository(Chat::class)->findChatByUsers(  
    $otherUser->getId(),  
    $this->getUser()->getId()
```

);

Якщо чат вже існує, викидається виняток:

```
if ($chat) {  
    throw new \Exception("Чат вже існує");  
}
```

Створюється новий об'єкт чату, встановлюється його статус, поточний користувач і користувач, з яким створюється чат:

```
$chat = new Chat();  
$chat->setStatus('Новий чат');  
$chat->setUser($this->getUser());  
$chat->setUserLike($otherUser);
```

Перевіряється, чи існує симпатія між поточним користувачем і користувачем, з яким створюється чат. Для цього використовується відповідний метод репозиторію симпатій:

```
$sympathy = $doctrine->getRepository(Sympathy::class)->isUserInSympathy(  
    $user->getId(),  
    $otherUser->getId()  
);
```

Розпочинається транзакція для збереження змін у базі даних. Якщо симпатія існує, вона видаляється. Чат зберігається у базі даних, після чого транзакція завершується успішно. Якщо під час збереження виникає помилка, транзакція відкатується, і викидається виняток:

```
$entityManager->getConnection()->beginTransaction();  
try {  
    $entityManager->persist($chat);  
    if ($sympathy) {  
        $entityManager->remove($sympathy);  
    }  
    $entityManager->flush();  
    $entityManager->commit();  
} catch (\Exception $e) {  
    $entityManager->rollback();  
    throw $e;  
}
```

Наприкінці повертається JSON-відповідь з ідентифікатором створеного чату та статусом HTTP 201 Created:

```
return $this->json([  
    'id' => $chat->getId()  
], Response::HTTP_CREATED, [], []);
```

Код даного методу наведено у додатку А.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Метод `newMessage` призначений для надсилання повідомлення у вказаний чат. Спочатку визначається маршрут, який обробляє POST-запити на адресу `/chat/{id}`. Метод приймає інтерфейс `HubInterface`, об'єкт запиту, ідентифікатор чату, інтерфейс серіалізатора та реєстр менеджерів. Менеджер сутностей отримується для роботи з базою даних:

```
$entityManager = $doctrine->getManager();
```

Знаходиться об'єкт чату за переданим ідентифікатором, а також витягується поточний користувач:

```
$chat = $doctrine->getRepository(Chat::class)->find($id);
```

```
$user = $this->getUser();
```

Текст повідомлення отримується з запиту, а також перевіряється наявність зображення у файлах запиту:

```
$text = $request->get('message', null);
```

```
$image = $request->files->get('image', null);
```

Створюється новий об'єкт повідомлення, встановлюються його текст, статус, користувач, що надіслав повідомлення, та дата і час створення:

```
$message = new Message();
```

```
$message->setMessage($text);
```

```
$message->setStatus('UNREAD');
```

```
$message->setUser($user);
```

```
$message->setDateTime(new \DateTimeImmutable());
```

Якщо у повідомленні є зображення, воно зберігається у директорії `images` з унікальним іменем, і це ім'я додається до об'єкта повідомлення:

```
if($image){
```

```
    $newImageName = uniqid() . '.' . $image->guessExtension();
```

```
    $image->move('images', $newImageName);
```

```
    $message->setImage($newImageName);
```

```
}
```

Повідомлення додається до чату, а також оновлюється останнє повідомлення у чаті:

```
$chat->addMessage($message);
```

```
$chat->setLastMessage($message);
```

Повідомлення і чат зберігаються у базі даних:

```
$entityManager->persist($message);
```

```
$entityManager->persist($chat);
```

```
$entityManager->flush();
```

Повідомлення серіалізується для подальшої публікації, вказується, що воно надіслане іншим користувачем (не поточним):

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

```
$message->setMine(false);
$messageSerialized = $serializer->serialize($message, 'json', [
    'attributes' => ['id', 'image', 'message', 'dateTime', 'mine', 'chat' => ['id']]
]);
```

Отримується інший учасник чату для надсилання йому оновлення:

```
$participants = [$chat->getUser(), $chat->getUserLike()];
$otherUser = null;
foreach ($participants as $participant) {
    if ($user !== $participant) {
        $otherUser = $participant;
        break;
    }
}
```

Оновлення публікується у хаб, де воно стає доступним для обох користувачів чату:

```
$update = new Update(
    [
        sprintf("/chat/%s", $chat->getId()),
        sprintf("/chat/%s", $otherUser->getEmail()),
    ],
    $messageSerialized,
    true
);
$hub->publish($update);
```

Зрештою, повідомлення повертається у JSON-форматі з атрибутом, що воно надіслане поточним користувачем:

```
$message->setMine(true);
return $this->json($message, Response::HTTP_CREATED, [], [
    'attributes' => ['id', 'image', 'message', 'dateTime', 'mine']
]);
```

Код даного методу наведено у додатку А.

Висновки до другого розділу

В процесі розробки сайту було створено кілька ключових алгоритмів для забезпечення користувачів необхідними функціями. Зокрема, описано алгоритм пошуку користувачів для рекомендацій, який базується на вікових обмеженнях, спільних інтересах і стилях життя, а також враховує мету знайомства та сексуальну орієнтацію користувачів.

Також було реалізовано алгоритм додавання користувачів до чорного списку, який перевіряє наявність користувача у списку симпатій і видаляє його, якщо він там присутній, перед додаванням до чорного списку.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

Аналогічно, алгоритм додавання симпатій перевіряє наявність користувача в чорному списку і видаляє його звідти при додаванні до списку симпатій.

Розглянуто алгоритм створення чату, який перевіряє існування чату між двома користувачами та створює новий чат, якщо він не існує, видаляючи при цьому наявні симпатії. Нарешті, описано метод надсилання повідомлення, який дозволяє користувачам обмінюватися повідомленнями у чаті, зберігати ці повідомлення та оновлювати інформацію у реальному часі за допомогою публікації оновлень у хаб.

Таким чином, спроектовані алгоритми забезпечують ефективне управління взаємодією користувачів на платформі, зокрема, додаючи можливості для рекомендацій, блокування, симпатій та обміну повідомленнями, що сприяє покращенню користувацького досвіду та підвищенню функціональності системи.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З САЙТОМ ЗНАЙОМСТВ

3.1 Порядок встановлення та налаштування параметрів системи

Для успішного встановлення та налаштування системи, спочатку необхідно ознайомитись із діаграмою розгортання (рис. 3.1), яка показує, як компоненти системи взаємодіють між собою та які апаратні і програмні ресурси використовуються.

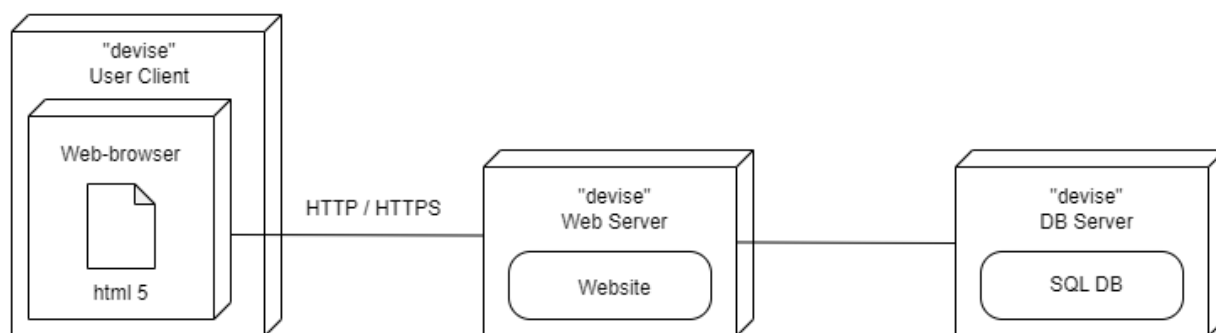


Рисунок 3.1 – Діаграма розгортання

Для коректної роботи сайту знайомств необхідно налаштувати робоче середовище для запуску компонентів системи. На робочій машині повинні бути встановлені наступні версії компонентів та засобів:

PHP (версія 8.2.10) – мова програмування серверної частини.

Symfony (версія 7.0.5) – фреймворк PHP, для розробки веб-додатків.

React (версія 18.3.1) – бібліотека для створення користувацького інтерфейсу.

Mercure (версія 0.9.0) – протокол для оновлення даних в реальному часі[4].

npm (версія 10.2.4) – для керування залежностями React-проєкту.

Composer (версія 2.6.2) – менеджер пакетів для PHP.

MySQL (версія 5.7) – система управління реляційними базами даних.

Розгортання проєкту на основі Symfony вимагає налаштування конфігураційних файлів. Використання файлу .env сприяє зручному управлінню змінними середовища для проєкту.

```

APP_ENV=dev
APP_SECRET=ce08d26219bb2fb69793920ad8f0ea98
DATABASE_URL="mysql://root:@127.0.0.1:3306/Diploma?serverVersion=8.0.32&charset=utf8mb4"
MAILER_DSN=smtp://1c59aae0efd2e3:dd457de545c89d@sandbox.smtp.mailtrap.io:2525
MERCURE_URL=http://localhost:3000/.well-known/mercure
MERCURE_PUBLIC_URL=http://localhost:3000/.well-known/mercure
MERCURE_JWT_SECRET=HrushevitskiySerhiiVitaliyovichS

```

- APP_ENV - Вказує на те, що середовище додатку встановлене у режимі розробки.
- APP_SECRET - Секретний ключ додатку, який використовується для підпису куки та інших даних безпеки.
- DATABASE_URL - Посилання на базу даних MySQL, включаючи ім'я користувача, пароль, хост, порт, назву бази даних, версію сервера і кодування.
- MAILER_DSN - Налаштування відправлення пошти через сервер SMTP (в даному випадку, через сервіс Mailtrap) з використанням певних облікових даних [14].
- MERCURE_URL - URL-адреса сервера Mercure, яка використовується для реалізації сповіщень у реальному часі [29].
- MERCURE_PUBLIC_URL - Публічна URL-адреса сервера Mercure для підписки на сповіщення.
- MERCURE_JWT_SECRET - Секретний ключ, який використовується для генерації JWT токенів для сервера Mercure [28].

3.2 Структура інтерфейсу вебдодатку та порядок роботи

Для початку роботи з сайтом користувач повинен авторизуватися. Спершу він може зареєструвати новий обліковий запис за допомогою форми реєстрації (рис. 3.2).

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Реєстрація

👁

Погодитися з умовами ☐

Зареєструватися

Вже маєте акаунт?

Рисунок 3.2 – Форма реєстрації

Після введення коректних облікових даних користувачу надсилається повідомлення (рис. 3.3) про необхідність верифікації електронної пошти.

Підтвердження пошти

Будь ласка, підтвердіть вашу електронну пошту,
щоб продовжити.

Я підтвердив

Рисунок 3.3 – Повідомлення про необхідність верифікації акаунту

Користувач отримує електронний лист із спеціальним посиланням для верифікації (рис. 3.4). Це посилання містить унікальний токен, який є ключем доступу до процесу верифікації і має обмежений строк дії — одну годину. Це зменшує ризик зловживання посиланням у разі його перехоплення. Процес верифікації електронної пошти побудований з урахуванням важливих аспектів безпеки, щоб гарантувати, що лише правомірні користувачі можуть підтвердити свою електронну адресу. Після успішної верифікації користувач отримує доступ до функціоналу сайту.

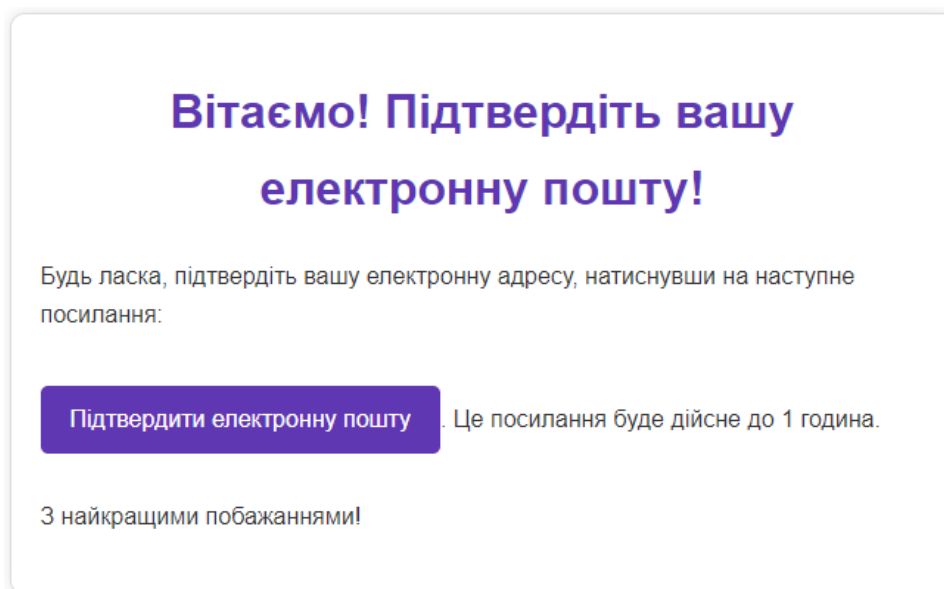


Рисунок 3.4 – Електронний лист з посиланням для верифікації облікового запису

Якщо користувач вже зареєстрований, він може скористатися формою авторизації (рис. 3.5).

Рисунок 3.5 – Форма авторизації користувача

Якщо користувач забув пароль, він може перейти на сторінку відновлення пароля за посиланням "Забули пароль?" (рис. 3.6).

Рисунок 3.6 – Форма запиту на відновлення пароля

SymfonyCast/reset-password-bundle надає можливість користувачам скинути свої паролі за допомогою токенів. Користувач вводить свою електронну адресу в спеціальну форму на сторінці запиту скидання пароля. Система перевіряє наявність користувача із цією електронною адресою і, якщо він існує, генерує токен для скидання пароля. Цей токен разом з інструкціями відправляється на електронну пошту користувача (рис. 3.7).

Лист для скидання паролю надіслано

Якщо існує обліковий запис з вашою електронною адресою, на вашу пошту було надіслано лист з посиланням для скидання паролю. Це посилання буде дійсне 1 година.

Якщо ви не отримали листа, перевірте папку зі спамом або [спробуйте знову](#).

Рисунок 3.7 – Повідомлення про відправлення листа

Користувач отримує лист з посиланням для скидання пароля (рис. 3.8).

Привіт!

Щоб скинути свій пароль, будь ласка, відвідайте наступне посилання:

<http://127.0.0.1:8000/reset-password/reset/zevXfkr9hJWjimprwLcUcvU1vmaaI5BRXRIKGpO7>

У вас є 1 година для підтвердження.

З найкращими побажаннями!

Рисунок 3.8 – Лист з посиланням для скидання пароля

Перейшовши за цим посиланням, він вводить новий пароль у формі на відповідній сторінці (рис. 3.9). Система перевіряє валідність токена, і якщо все в порядку, хешує новий пароль та зберігає його в базі даних. Після успішного скидання пароля сесія очищується, щоб запобігти повторному використанню токена.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

Рисунок 3.9 – Форма зміни паролю

Система авторизації забезпечує високий рівень безпеки, захищаючи токени, паролі і сесії, що запобігає несанкціонованому доступу. Увійшовши до облікового запису, користувач може почати користуватися функціоналом сайту.

Після реєстрації користувач потрапляє на сторінку заповнення профілю (рис. 3.10). На цій сторінці користувач може ввести свою основну інформацію для профілю. Після введення даних, їх можна зберегти, натиснувши кнопку "Зберегти". Користувачу доступні поля для введення ім'я, віку, зросту, статі та сексуальної орієнтації.

Профіль

Основне
Стиль життя
Хобі
Цілі
Фото
Профіль

Основна інформація

Рисунок 3.10 – Форма заповнення базової інформації про користувача

Наступною відображає форму (рис. 3.11), на якій користувач може вибрати свій стиль життя. Після завантаження сторінки відбувається запит до API для отримання інформації про стилі життя користувача. Після цього користувач може вибрати бажані стилі з чекбоксів та натиснути кнопку "Зберегти" для збереження змін.

Профіль

Основне Стиль життя Хобі Цілі Фото Профіль

Стиль життя

☐ Не курю ☐ Курю ☐ Не вживаю алкоголь ☐ Вживаю алкоголь ☐ Без кофеїну
☐ Жайворонок ☐ Сова ☐ Веганство ☐ Люблю м'ясо ☐ Їм усе
☐ Займаюсь спортом ☐ Люблю тварин ☐ Екологічний спосіб життя ☐ Мініمالізм
☐ Цифровий кочівник ☐ Фрілансер ☐ Мандрівник ☐ Здорове харчування
☐ Суспільно активний ☐ Волонтер ☐ Сімейно орієнтований ☐ Кар'єрист
☐ Інтроверт ☐ Екстраверт ☐ Технологічний ентузіаст ☐ Віддалений працівник
☐ Інвестор ☐ Художник ☐ Музикант ☐ Геймер

ЗБЕРЕГТИ

ПОВЕРНУТИСЬ **ДАЛІ**

Рисунок 3.11 – Форма вибору свого стилю життя

Наступною відображає форму (рис. 3.12), на якій користувач може вибрати та додати свої хобі. Під час завантаження сторінки відбувається запит до API для отримання інформації про хобі користувача. Після цього користувач може вибрати бажані хобі зі списку або ввести нові хобі в поле введення. Після вибору хобі користувач може натиснути кнопку "Зберегти" для збереження змін.

Профіль

Основне Стиль життя Хобі Цілі Фото Профіль

Хобі

Обрати хобі

Читання × Музика × Біг × Караоке × ×

Для додавання нового хобі натисніть Enter

ЗБЕРЕГТИ

ПОВЕРНУТИСЬ ДАЛІ

Рисунок 3.12 – Форма вибору своїх хобі

Наступною відображає форму (рис. 3.13), де користувач може вказати своє місце роботи, цілі, опис та місцезнаходження. Користувач може внести зміни до цих даних і зберегти їх за допомогою кнопки "Зберегти". Після заповнення полів користувач може натиснути кнопку "Зберегти" для збереження змін.

Рисунок 3.13 – Форма заповнення цілі використання сайту

Наступною відображає форму (рис. 3.14), завантаження фото користувача. На сторінці відображаються завантажені фото та зона перетягування для додавання нових зображень. Користувач може перетягнути файли та вибрати їх для завантаження. Завантажені зображення відображаються у вигляді карточок з можливістю видалення. Користувач може видалити будь-яке зображення, натиснувши на кнопку "Видалити" під кожним зображенням.

Профіль


 Основне


 Стиль життя


 Хобі


 Цілі


 Фото


 Профіль

Фото

Перетягніть файли сюди або натисніть для завантаження











ПОВЕРНУТИСЬ

ДАЛІ

Рис. 3.14 – Форма додавання фото користувача

Пройшовши усі етапи заповнення профілю користувач може перейти на будь-яку сторінку сайту або переглянути щойно заповнений профіль. На сторінці (рис. 3.15) відображено картку з профілем користувача, яка містить інформацію про нього, таку як ім'я, вік, опис, мета, зріст, стать, гендер, робота, хобі та стиль життя. Фотографії користувача відображаються у вигляді каруселі, де можна прогортати зображення.



Боб 58

Про мене: Незважаючи на те, що я невисокий, я точно знаю, де мені потрібно бути

Мета: Серйозні стосунки

Зріст: 143 см

Стать: Чоловік

Гендер: Гомосексуаліст(ка)

Робота: Сценарист

Хобі: Читання Музика Біг Караоке

Стиль життя: Курю Вживаю алкоголь Їм усе Мінімалізм Волонтер Кар'єрист

Музикант

Рисунок 3.15 – Форма додавання фото користувача

Для отримання рекомендацій спочатку потрібно налаштувати їх. При відвідуванні сторінки налаштування рекомендацій користувач отримує можливість заповнити форму основних налаштувань (рис. 3.16). Користувач може задати вікові обмеження за допомогою повзунка, який відображає обрані межі віку. Також користувач може ввести діапазон відстані пошуку у форматі кілометрів. Користувач може натиснути кнопку "Зберегти" для збереження змін.

Рисунок 3.16 – Форма заповнення основних налаштувань рекомендацій

На цій сторінці користувач бачить форму для вибору бажаного стилю життя (рис. 3.17) у рекомендованих користувачів. Користувач може обирати стиль життя який він хоче бачити у рекомендованих користувачів зі списку вже наявних. Після того, як користувач вибрав свої уподобання у хобі та натиснув кнопку "Зберегти", дані відправляються на сервер для збереження.

Рисунок 3.17 – Форма вибору бажаного стилю життя у рекомендованих користувачів

На цій сторінці користувач бачить форму для вибору хобі (рис. 3.18) у рекомендованих користувачів. Користувач може обирати кілька хобі зі списку вже наявних опцій. Користувач також може вводити нові хобі, просто натискаючи клавішу "Enter". Після того, як користувач вибрав свої уподобання у хобі та натиснув кнопку "Зберегти", дані відправляються на сервер для збереження.

Рисунок 3.18 – Форма вибору бажаних хобі у рекомендованих користувачів

Після завантаження сторінки рекомендацій (рис. 3.19) користувач бачить картку з інформацією про рекомендованих користувачів. Кожна картка містить фотографії користувача, його ім'я, вік та короткий опис.

Користувач може перегортати картки, вибираючи чи хоче він продовжити спілкування з обраним користувачем чи ні. Він може натискати на іконки "Вподобання" або "Не цікаво" для позначення своїх симпатій та відсутності інтересу до користувача, а також може переглядати більше інформації про користувача, натиснувши відповідну кнопку.

Крім того, користувачу надається можливість перейти до налаштувань свого профілю, натиснувши відповідну кнопку, якщо він не знайшов спільноти з обраними користувачами.

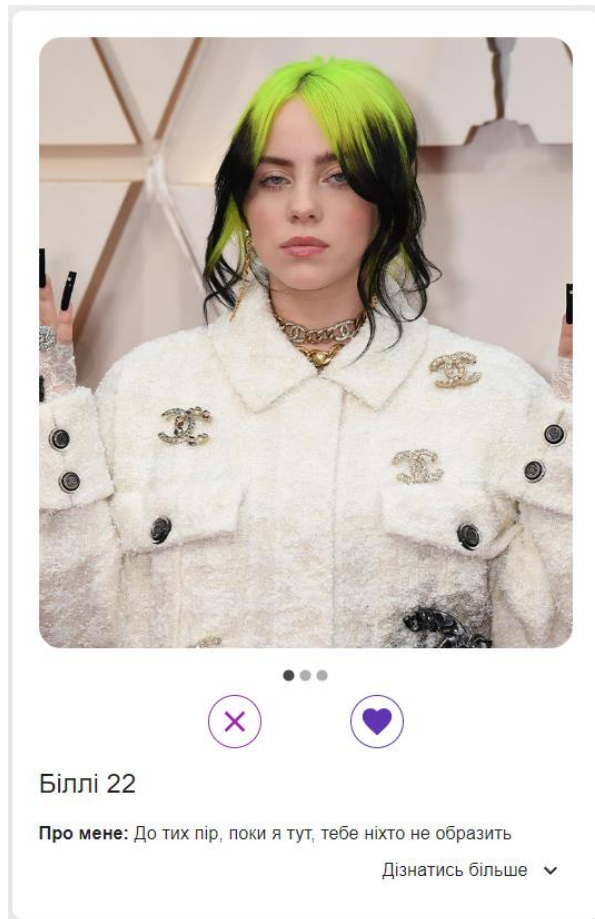


Рисунок 3.19 – Сторінка рекомендацій

На сторінці симпатії (рис. 3.20) користувач бачить картки інших користувачів, які його вподобали або він їх вподобав. Кожна картка містить фотографії користувача, його ім'я, вік та короткий опис. Користувач може перегортати картки за допомогою кнопок навігації.

Крім основних даних, користувач може натискати на іконки "Вподобання" або "Не цікаво" для зміни свого вибору. Також для кожної картки є можливість переглядати додаткову інформацію про користувача, розкриваючи розгорнутий вигляд. Після взаємного інтересу користувач може створити чат з обраним користувачем.

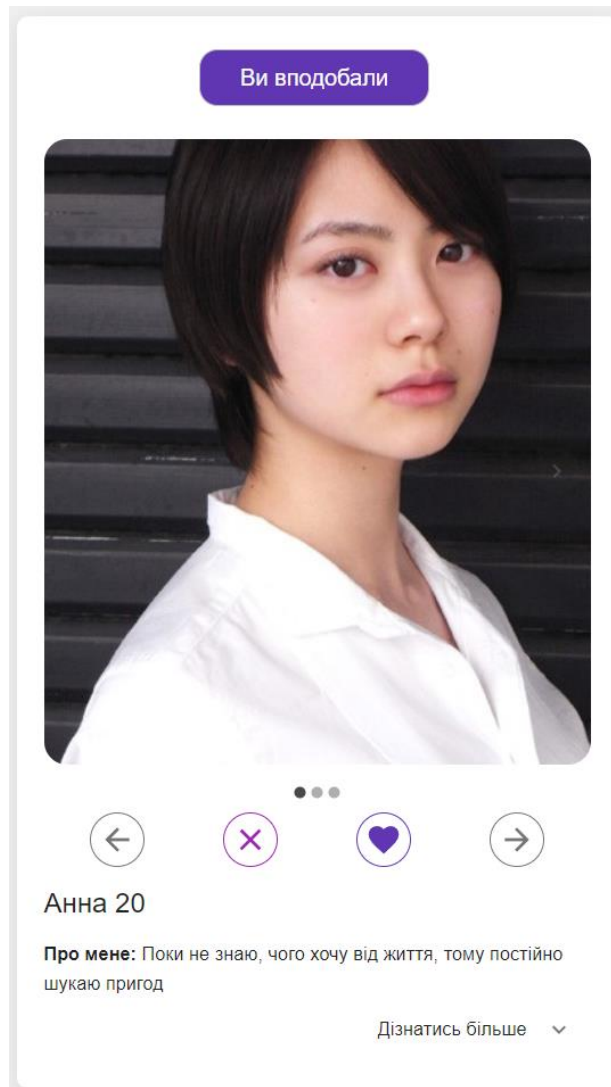


Рисунок 3.20 – Сторінка симпатій

На сторінці чорного списку (рис. 3.21) користувач бачить картки інших користувачів, яких він додав до чорного списку. Кожна картка містить фотографії користувача, його ім'я, вік та короткий опис. Користувач може перегортати картки за допомогою кнопок навігації.

Крім основних даних, користувач може натискати на іконки "Вподобання" або "Не цікаво" для зміни свого вибору. Також для кожної картки є можливість переглядати додаткову інформацію про користувача, розкриваючи розгорнутий вигляд.

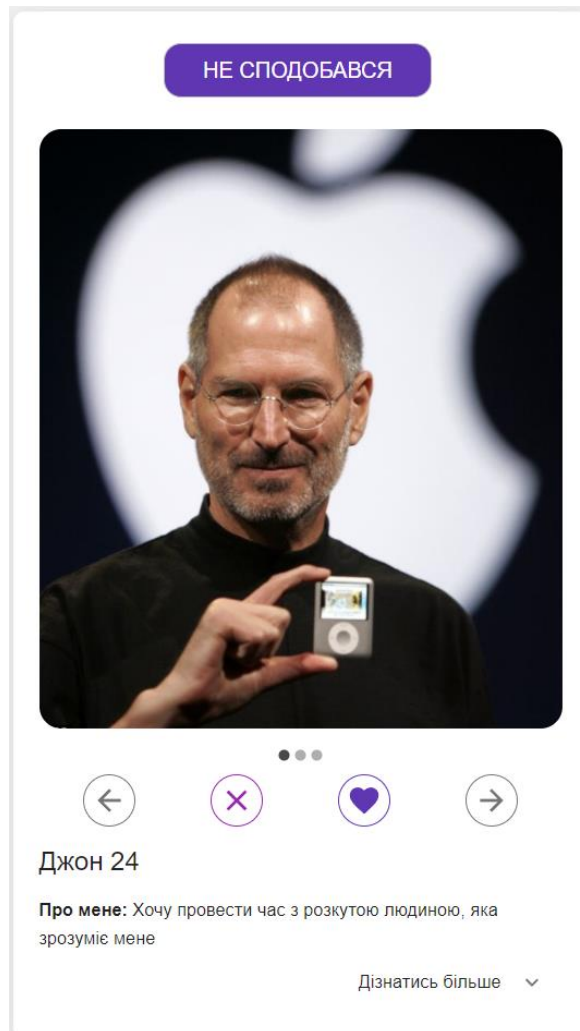


Рисунок 3.21 – Сторінка чорного списку

На сторінці чатів користувача (рис. 3.22) відображаються усі чати користувача. При натисненні на чат він відкривається та завантажуються повідомлення. Користувач може надіслати повідомлення в чат, та в режимі реального часу спостерігати за надісланими та отриманими повідомленнями. Також користувач може видалити чат що призведе до додавання користувача з яким вівся чат до чорного списку.

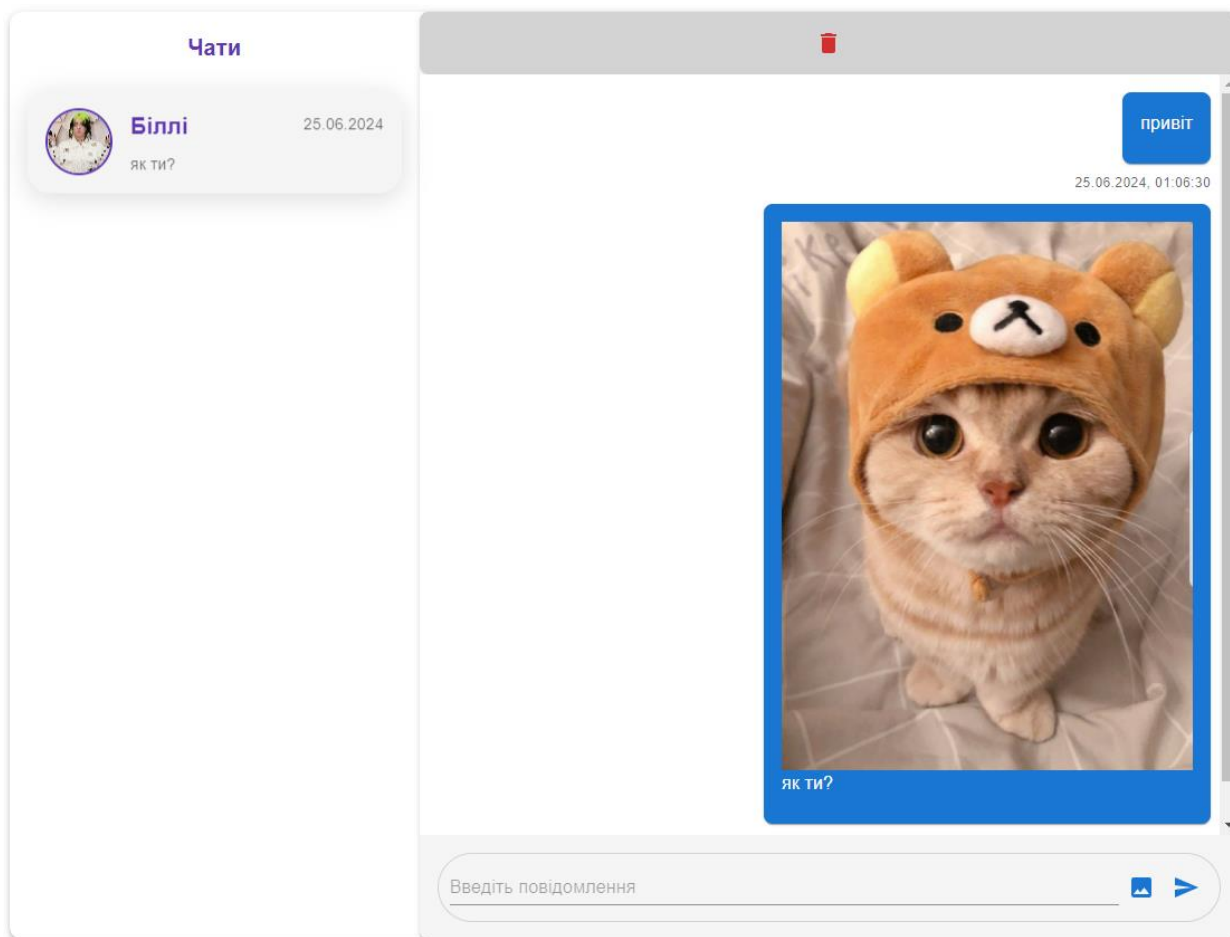


Рисунок 3.22 – Сторінка чатів

На сторінці користувачі в адмін панелі (рис. 3.23), адміністратор може керувати обліковими записами користувачів. Тут представлений CRUD (створення, читання, оновлення, видалення) інтерфейс для сутності User.

User Створити User

☐	ID	Name	Email	Description	
☐	1	Сергій	renfrenb@gmail.com	Переглянути вміст	...
☐	2	Матвій	mat@gmail.com	Переглянути вміст	...
☐	5	Даша	sten@gmail.com	Переглянути вміст	...
☐	19	Анна	anna@gmail.com	Переглянути вміст	...
☐	21	Джон	jonger@gmail.com	Переглянути вміст	...
☐	22	Зендея	zimb@gmail.com	Переглянути вміст	...
☐	23	Кара	cara@gmail.com	Переглянути вміст	...
☐	24	Том	tom@gmail.com	Переглянути вміст	...
☐	25	Тіль	till@gmail.com	Переглянути вміст	...
☐	29	Біллі	billie@gmail.com	Переглянути вміст	...
☐	30	Боб	chaca@gmail.com	Переглянути вміст	...

11 результатів

Рисунок 3.23 – Сторінка користувачі в адмін панелі

На сторінці хобі в адмін панелі (рис. 3.24), адміністратор може керувати хобі. На цій сторінці використовується CRUD (створення, читання, оновлення, видалення) інтерфейс для сутності Hobbies.

Hobbies

Створити Hobbies

☐	ID	Name	
☐	1	Фітнес	...
☐	2	Йога	...
☐	3	Танці	...
☐	4	Відеоігри	...
☐	5	Плавання	...
☐	6	Читання	...
☐	7	Фотографія	...
☐	8	Малювання	...
☐	9	Письмо	...
☐	10	Музика	...
☐	11	Кулінарія	...
☐	12	Велоспорт	...
☐	13	Туризм	...
☐	14	Подорожі	...
☐	15	Кросфіт	...
☐	17	Шиття	...
☐	18	Малювання графіті	...
☐	19	Колекціонування	...
☐	20	Кіно	...
☐	22	Блогінг	...

38 результатів

Попередня

1

2

Наступна

Рисунок 3.24 – Сторінка хобі в адмін панелі

На сторінці стилі життя в адмін панелі (рис. 3.25), адміністратор може керувати стилями життя. На цій сторінці використовується CRUD (створення, читання, оновлення, видалення) інтерфейс для сутності LifeStyle.

☐	ID	Name	
☐	1	Не курю	...
☐	2	Курю	...
☐	3	Не вживаю алкоголь	...
☐	4	Вживаю алкоголь	...
☐	5	Без кофеїну	...
☐	6	Жайворонок	...
☐	7	Сова	...
☐	8	Веганство	...
☐	9	Люблю м'ясо	...
☐	10	Ім усе	...
☐	11	Займаюсь спортом	...
☐	12	Люблю тварин	...
☐	13	Екологічний спосіб життя	...
☐	14	Мінімалізм	...
☐	15	Цифровий кочівник	...
☐	16	Фрілансер	...
☐	17	Мандрівник	...
☐	18	Здорове харчування	...
☐	19	Суспільно активний	...
☐	20	Волонтер	...

30 результатів

< Попередня 1 2 Наступна >

Рисунок 3.25 – Сторінка стилі життя в адмін панелі

3.3 Тестування роботи та використання сайту

Перевірка роботи сайту є ключовим етапом. Якість та надійність програми, а також задоволення користувачів від її функціоналу, залежать від результатів тестування. Цей процес допомагає виявити та усунути потенційні помилки та проблеми, що забезпечує стабільну роботу програми. Нижче представлений чек-лист для тестування сайту знайомств:

Таблиця 3.1

Форма реєстрації

№	Опис тесту	Результат
1	Відкрити форму реєстрації	Форма реєстрації відкривається
2	Ввести валідні дані у всі поля форми	Дані приймаються системою
3	Ввести невалідний email	Відображається повідомлення про помилку
4	Залишити поле пароля порожнім	Відображається повідомлення про необхідність заповнення
5	Ввести короткий пароль (менше 6 символів)	Відображається повідомлення про мінімальну довжину пароля
6	Підтвердити реєстрацію з валідними даними	Користувач отримує повідомлення про необхідність верифікації
7	Підтвердити реєстрацію з невалідними даними	Відображається відповідне повідомлення про помилку

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

Таблиця 3.2

Повідомлення про верифікацію електронної пошти

№	Опис тесту	Результат
1	Завершити реєстрацію з валідними даними	Користувач отримує повідомлення про необхідність верифікації
2	Перевірити, чи надіслано email для верифікації	Електронний лист надсилається на вказану адресу
3	Перевірити вміст електронного листа	Лист містить унікальне посилання з токеном

Таблиця 3.3

Верифікація електронної пошти

№	Опис тесту	Результат
1	Перейти за посиланням в електронному листі	Користувач верифікований, доступ до функціоналу сайту відкритий
2	Використати посилання з токеном після закінчення терміну його дії	Відображається повідомлення про недійсний токен
3	Використати неправильний токен	Відображається повідомлення про помилку

Таблиця 3.4

Форма авторизації

№	Опис тесту	Результат
1	Відкрити форму авторизації	Форма авторизації відкривається
2	Ввести валідні дані (email та пароль)	Користувач авторизується
3	Ввести невалідні дані (невірний email або пароль)	Відображається повідомлення про помилку
4	Відкрити сторінку відновлення пароля через посилання "Забули пароль?"	Сторінка відновлення пароля відкривається

Таблиця 3.5

Відновлення пароля

№	Опис тесту	Результат
1	Відкрити сторінку запиту на скидання пароля	Сторінка запиту відкривається
2	Ввести валідний email у форму запиту	Користувач отримує повідомлення про відправку листа
3	Ввести невалідний email	Відображається повідомлення про помилку
4	Перевірити, чи надіслано email з посиланням на скидання пароля	Лист надсилається на вказану адресу
5	Перейти за посиланням у листі	Відкривається форма для зміни пароля
6	Ввести новий валідний пароль у форму зміни пароля	Пароль успішно змінюється, користувач повідомлений про успішну операцію
7	Використати неправильний або прострочений токен	Відображається повідомлення про помилку

Таблиця 3.6

Сторінка заповнення профілю

№	Опис тесту	Результат
1	Відкрити сторінку заповнення профілю	Сторінка відкривається
2	Ввести валідні дані у всі поля	Дані приймаються системою
3	Залишити одне або кілька полів порожніми	Відображається повідомлення про необхідність заповнення
4	Натиснути кнопку "Зберегти" після введення валідних даних	Дані зберігаються, користувач повідомлений про успішне збереження
5	Ввести некоректні дані (наприклад, текст у полі "вік")	Відображається повідомлення про помилку

Таблиця 3.7

Форма вибору стилю життя

№	Опис тесту	Результат
1	Відкрити форму вибору стилю життя	Форма відкривається
2	Перевірити завантаження стилів життя з API	Стилі життя успішно завантажуються
3	Вибрати кілька стилів життя за допомогою чекбоксів	Стилі вибираються
4	Залишити всі чекбокси порожніми та натиснути "Зберегти"	Відображається повідомлення про обов'язковий вибір
5	Вибрати стилі життя та натиснути "Зберегти"	Зміни успішно зберігаються
6	Перевірити, чи зберігаються обрані стилі життя після перезавантаження сторінки	Обрані стилі життя зберігаються

Таблиця 3.8

Форма вибору хобі

№	Опис тесту	Результат
1	Відкрити форму вибору хобі	Форма відкривається
2	Перевірити завантаження хобі з API	Хобі успішно завантажуються
3	Вибрати кілька хобі зі списку	Хобі вибираються
4	Ввести нове хобі у полі введення та натиснути "Enter"	Нове хобі додається до списку
5	Залишити всі поля порожніми та натиснути "Зберегти"	Відображається повідомлення про обов'язковий вибір
6	Вибрати хобі та натиснути "Зберегти"	Зміни успішно зберігаються
7	Перевірити, чи зберігаються обрані хобі після перезавантаження сторінки	Обрані хобі зберігаються

Таблиця 3.9

Форма введення місця роботи, цілей, опису та місцезнаходження

№	Опис тесту	Результат
1	Відкрити форму введення місця роботи, цілей, опису та місцезнаходження	Форма відкривається
2	Ввести валідні дані у всі поля	Дані приймаються системою

3	Залишити одне або кілька полів порожніми та натиснути "Зберегти"	Відображається повідомлення про обов'язкове заповнення
4	Ввести некоректні дані (наприклад, спеціальні символи у полі "Місце роботи")	Відображається повідомлення про помилку
5	Ввести валідні дані та натиснути "Зберегти"	Зміни успішно зберігаються
6	Перевірити, чи зберігаються дані після перезавантаження сторінки	Дані зберігаються

Таблиця 3.10

Форма завантаження фото

№	Опис тесту	Результат
1	Відкрити форму завантаження фото	Форма відкривається
2	Перетягнути файли у зону завантаження	Файли завантажуються
3	Натиснути кнопку "Видалити" під кожним зображенням	Зображення видаляється
4	Завантажити файл некоректного формату (не зображення)	Відображається повідомлення про помилку
5	Перевірити, чи відображаються завантажені зображення у вигляді карточок	Зображення відображаються у вигляді карточок
6	Перевірити, чи видаляються зображення після натискання кнопки "Видалити"	Зображення успішно видаляються

Таблиця 3.11

Картка профілю користувача

№	Опис тесту	Результат
1	Відкрити сторінку з картою профілю користувача	Сторінка відкривається
2	Перевірити відображення інформації про користувача	Інформація відображається коректно
3	Переглянути фотографії у каруселі	Фотографії можна перегортати
4	Перевірити коректність відображення інформації (ім'я, вік, опис, мета і т.д.)	Інформація відображається коректно

Таблиця 3.12

Налаштування рекомендацій

№	Опис тесту	Результат
1	Відкрити сторінку налаштування рекомендацій	Сторінка відкривається
2	Налаштувати вікові обмеження за допомогою повзунка	Обрані межі віку відображаються коректно
3	Ввести діапазон відстані у форматі кілометрів	Дані приймаються системою
4	Натиснути "Зберегти" для збереження налаштувань	Налаштування успішно зберігаються
5	Перевірити, чи зберігаються налаштування після перезавантаження сторінки	Налаштування зберігаються

Таблиця 3.13

Форма вибору стилю життя у рекомендованих користувачів

№	Опис тесту	Результат
1	Відкрити сторінку вибору стилю життя для рекомендованих користувачів	Сторінка відкривається
2	Вибрати кілька стилів життя зі списку	Стилі вибираються
3	Натиснути "Зберегти" для збереження вибору	Зміни успішно зберігаються
4	Перевірити, чи зберігаються обрані стилі після перезавантаження сторінки	Обрані стилі зберігаються

Таблиця 3.14

Форма вибору хобі у рекомендованих користувачів

№	Опис тесту	Результат
1	Відкрити сторінку вибору хобі для рекомендованих користувачів	Сторінка відкривається
2	Вибрати кілька хобі зі списку	Хобі вибираються
3	Ввести нове хобі у полі введення та натиснути "Enter"	Нове хобі додається до списку
4	Натиснути "Зберегти" для збереження вибору	Зміни успішно зберігаються
5	Перевірити, чи зберігаються обрані хобі після перезавантаження сторінки	Обрані хобі зберігаються

Таблиця 3.15

Сторінка рекомендацій

№	Опис тесту	Результат
1	Відкрити сторінку рекомендацій	Сторінка відкривається
2	Переглянути картки рекомендованих користувачів	Картки відображаються коректно
3	Перегортати картки користувачів	Картки перегортаються
4	Натиснути на іконки "Вподобання" або "Не цікаво"	Симпатія або відсутність інтересу відображаються
5	Натиснути кнопку для перегляду більше інформації про користувача	Відображається розгорнута інформація
6	Перейти до налаштувань профілю з сторінки рекомендацій	Налаштування профілю відкриваються

Таблиця 3.16

Сторінка симпатій

№	Опис тесту	Результат
1	Відкрити сторінку симпатій	Сторінка відкривається
2	Переглянути картки користувачів, які сподобалися або були вподобані	Картки відображаються коректно
3	Перегортати картки користувачів	Картки перегортаються
4	Натиснути на іконки "Вподобання" або "Не цікаво"	Симпатія або відсутність інтересу відображаються
5	Натиснути кнопку для перегляду більше інформації про користувача	Відображається розгорнута інформація

6	Створити чат з обраним користувачем після взаємного інтересу	Чат створюється
---	--	-----------------

Таблиця 3.17

Сторінка чорного списку

№	Опис тесту	Результат
1	Відкрити сторінку чорного списку	Сторінка відкривається
2	Переглянути картки користувачів, яких додано до чорного списку	Картки відображаються коректно
3	Перегортати картки користувачів	Картки перегортаються
4	Натиснути на іконки "Вподобання" або "Не цікаво"	Симпатія або відсутність інтересу відображаються
5	Натиснути кнопку для перегляду більше інформації про користувача	Відображається розгорнута інформація

Таблиця 3.18

Сторінка чатів

№	Опис тесту	Результат
1	Відкрити сторінку чатів	Сторінка відкривається
2	Переглянути список чатів	Список відображається коректно
3	Відкрити чат, натиснувши на нього	Чат відкривається, повідомлення завантажуються
4	Надіслати повідомлення в чат	Повідомлення надсилається і відображається у чаті
5	Перевірити оновлення чату у реальному часі	Повідомлення оновлюються у реальному часі
6	Видалити чат	Чат видаляється, користувач додається до чорного списку

Таблиця 3.19

Адмін панель: сторінка користувачів

№	Опис тесту	Результат
1	Відкрити сторінку керування користувачами	Сторінка відкривається
2	Створити нового користувача	Користувач успішно створюється
3	Прочитати інформацію про користувача	Інформація відображається коректно
4	Оновити інформацію про користувача	Інформація успішно оновлюється
5	Видалити користувача	Користувач успішно видаляється

Таблиця 3.20

Адмін панель: сторінка хобі

№	Опис тесту	Результат
1	Відкрити сторінку керування хобі	Сторінка відкривається
2	Створити нове хобі	Хобі успішно створюється
3	Прочитати інформацію про хобі	Інформація відображається коректно
4	Оновити інформацію про хобі	Інформація успішно оновлюється
5	Видалити хобі	Хобі успішно видаляється

Адмін панель: сторінка стилів життя

№	Опис тесту	Результат
1	Відкрити сторінку керування стилями життя	Сторінка відкривається
2	Створити новий стиль життя	Стиль життя успішно створюється
3	Прочитати інформацію про стиль життя	Інформація відображається коректно
4	Оновити інформацію про стиль життя	Інформація успішно оновлюється
5	Видалити стиль життя	Стиль життя успішно видаляється

Висновки до третього розділу

У першому підрозділі було докладно розглянуто налаштування, функціонування та тестування сайту знайомств. В ньому показано процес налаштування конфігурації бази даних та використання необхідних технологій для побудови системи, а також їх версії.

У другому підрозділі було описано програмний продукт, включаючи деталі про процеси взаємодії між користувачами та інші аспекти функціонування сайту знайомств. Було розглянуто ключові функціональні можливості, такі як реєстрація та авторизація користувачів, створення та редагування профілів, додавання фотографій, налаштування параметрів рекомендацій.

У третьому підрозділі було описано процес тестування сайту знайомств. Проведено ручне тестування для перевірки коректності роботи всіх функцій сайту. Було складено чекліст, що включає тестування авторизації та реєстрації, роботи профілю користувача, створення та редагування профілів, завантаження фотографій, налаштування рекомендацій, панелі адміністратора. Проведене тестування допомогло виявити та виправити помилки, що забезпечило стабільну та ефективну роботу сайту.

ВИСНОВКИ

У ході роботи було розроблено та досліджено сайт знайомств з метою створення ефективної платформи для взаємодії користувачів. Основний результат полягає в успішному створенні та впровадженні функціональної системи, яка дозволяє користувачам реєструватися, заповнювати профіль, додавати фотографії, отримання рекомендацій та взаємодіяти з іншими користувачами через чати.

Спочатку було визначено мету створення сайту знайомств, проведено огляд існуючих платформ для знайомств та висвітлено їх недоліки. Було підкреслено важливість забезпечення розширених опцій пошуку та створення зручного й безпечного інтерфейсу для користувачів. Також було розглянуто архітектуру вебдодатку, обрані технології та компоненти, які забезпечують успішну реалізацію концепції сайту знайомств.

Далі було проаналізовано функціональні та нефункціональні вимоги до системи, створено діаграми використання та описано структуру бази даних. Було визначено основні алгоритми роботи системи, такі як реєстрація, авторизація, відновлення пароля та налаштування рекомендацій. Також детально описано реалізацію програмного коду, що підвищило ефективність та надійність системи.

Розробка сайту охоплювала реалізацію таких функцій: реєстрація та авторизація користувачів, налаштування профілю з додаванням основної інформації, вибір стилю життя та хобі, завантаження фотографій, налаштування рекомендацій, а також можливість переглядати та взаємодіяти з рекомендованими користувачами. Система також включає можливість відновлення пароля, що забезпечує додатковий рівень безпеки для користувачів. Адміністративна панель дозволяє керувати користувачами, хобі та стилями життя, забезпечуючи гнучкість у управлінні контентом сайту.

Проведене тестування підтвердило коректну роботу всіх реалізованих функцій. Система авторизації забезпечує високий рівень безпеки, запобігаючи

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

несанкціонованому доступу, а інтерфейс користувача забезпечує зручну та інтуїтивно зрозумілу взаємодію з платформою. Адміністративні функції надають необхідні інструменти для ефективного управління користувачами та контентом, що сприяє стабільній роботі сайту.

Практична цінність результатів роботи полягає у створенні платформи, яка надає користувачам зручний та безпечний спосіб знайомства та взаємодії. Розроблені функції відповідають завданням і забезпечують високу ефективність роботи системи.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Symfony [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://symfony.com.ua/doc/current/index.html>.
2. Material UI [Електронний ресурс] – Режим доступу до ресурсу: <https://mui.com/material-ui/>.
3. React Redux [Електронний ресурс] – Режим доступу до ресурсу: <https://react-redux.js.org/introduction/getting-started>.
4. Instant Messaging in Symfony: A Deep Dive into Mercure Integration [Електронний ресурс] // Medium. – 2023. – Режим доступу до ресурсу: <https://medium.com/@skowron.dev/instant-messaging-in-symfony-a-deep-dive-into-mercure-integration-06ed9bbb87a7>.
5. Роберт Мартін. Чистий код. – Харків: Фабула, 2019. – 416 с.
6. Роберт Мартін. Чиста архітектура. – Харків: Фабула, 2019. – 368 с.
7. React [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.legacy.reactjs.org/docs/hooks-state.html>.
8. Kevin Tatroe. Programming PHP: Creating Dynamic Web Pages / Kevin Tatroe, Peter MacIntyre., 2020. – 542 с.
9. Гайна Г. А. Основи проектування баз даних / Георгій Анатолійович Гайна. – Київ: Кондор, 2018. – 204 с.
10. Глабець І. Використання архітектурного шаблону MVC при проектуванні програмного забезпечення [Електронний ресурс] / І. Глабець. – 2013. – Режим доступу до ресурсу: <https://api.core.ac.uk/oai/oai:elartu.tntu.edu.ua:123456789/9470>.
11. Дідковська М. В. Тестування: Основні визначення, аксіоми та принципи. Текст лекцій. Частина I / М. В. Дідковська, Ю. О. Тимошенко. – МОН України: ННК НТУУ «КПІ». Кафедра математичних методів системного аналізу, 2010. – 62 с.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

12. Дудзяний І. М. Об'єктно-орієнтоване моделювання програмних систем / Ігор Михайлович Дудзяний. – Львів: ЛНУ ім. Івана Франка, 2007. – 107 с.
13. Боков О. Г. Аналіз архітектури сучасних ВЕБ-додатків [Електронний ресурс] / О. Г. Боков – Режим доступу до ресурсу: <https://openarchive.nure.ua/handle/document/21107>.
14. Gerardus Blokdyk. SMTP A Complete Guide - 2021 Edition / Gerardus Blokdyk., 2021. – 303 с.
15. Єфремов М. Ф. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ UML [Електронний ресурс] / М. Ф. Єфремов, Ю. М. Єфремов, В. М. Єфремов. – 2016. – Режим доступу до ресурсу: <http://eztuir.ztu.edu.ua/123456789/3296>.
16. Мартін Р. Чиста архітектура: Мистецтво створення програмного забезпечення / Роберт Мартін. – Харків: Фабула, 2019. – 368 с. – (Видання друге).
17. Швець О. Занурення в ПАТЕРНИ ПРОЕКТУВАННЯ / Олександр Швець., 2022. – 396 с.
18. Етапи розробки веб-додатків [Електронний ресурс] // TDS Digital Solutions. – 2023. – Режим доступу до ресурсу: <https://terentevdesignstudio.com/blog/etapi-rozrobki-veb-dodatviv/>.
19. Створення веб-додатку: основні аспекти ефективної розробки веб-сайту [Електронний ресурс] // Webbook Studio. – 2023. – Режим доступу до ресурсу: <https://webbookstudio.com/ua/articles/building-a-web-application-key-considerations-for-efferctive-website-development/>.
20. Створення веб-додатків з використанням принципів SOLID та Clean Code [Електронний ресурс] // ІТ рейтинг України. – 2024. – Режим доступу до ресурсу: <https://it-rating.ua/stvorenniya-veb-dodatktiv-z-vikoristannyam-printsipiv-solid-ta-clean-code>.

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

21. Як обрати архітектуру для веб додатку [Електронний ресурс] // (Hillel IT News. – 2022. – Режим доступу до ресурсу: <https://blog.ithillel.ua/articles/web-application-architecture>.
22. Anthony Accomazzo. Fullstack React: The Complete Guide to ReactJS and Friends / Anthony Accomazzo, Nate Murray, Ari Lerner., 2017. – 836 с.
23. Martin Kleppmann. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems / Martin Kleppmann., 2017. – 611 с.
24. Tinder [Електронний ресурс] – Режим доступу до ресурсу: <https://tinder.com/>.
25. Badoo [Електронний ресурс] – Режим доступу до ресурсу: <https://badoo.com/uk>.
26. Waytohey [Електронний ресурс] – Режим доступу до ресурсу: <https://waytohey.com/>.
27. MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/>.
28. Introduction to JSON Web Tokens [Електронний ресурс] – Режим доступу до ресурсу: <https://jwt.io/introduction>.
29. Mercure [Електронний ресурс] – Режим доступу до ресурсу: <https://mercure.rocks/docs>.
30. PhpMyAdmin [Електронний ресурс] – Режим доступу до ресурсу: <https://www.phpmyadmin.net/docs/>.

ДОДАТКИ

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

Лістинг методу haversineDistance:

```
function haversineDistance($lat1, $lon1, $lat2, $lon2) {
    $earthRadius = 6371; // Earth's radius in kilometers
    $dLat = deg2rad($lat2 - $lat1);
    $dLon = deg2rad($lon2 - $lon1);
    $a = sin($dLat / 2) * sin($dLat / 2) + cos(deg2rad($lat1)) * cos(deg2rad($lat2)) * sin($dLon
/ 2) * sin($dLon / 2);
    $c = 2 * asin(sqrt($a));
    $distance = $earthRadius * $c;
    return $distance;
}
```

Лістинг методу index:

```
#[Route('/recommendation', name: 'app_recommendation')]
public function index(SerializerInterface $serializer, UserRepository $userRepository,
ManagerRegistry $doctrine): Response
{
    $user = $this->getUser();
    $preferences = $user->getPreferences();
    $ageMin = $preferences->getAgeMin();
    $ageMax = $preferences->getAgeMax();
    $hobbiesCollection = $preferences->getHobbies();
    $location = $user->getLocation();
    $locationRange = $preferences->getLocationRange();
    list($lat1, $lon1) = explode(",", $location, 2);
    $hobbiesArray = $hobbiesCollection->map(function($hobby) {
        return $hobby->getId();
    })->toArray();
    $lifestylesCollection = $preferences->getLifestyles();
    $lifestylesArray = $lifestylesCollection->map(function($lifestyle) {
        return $lifestyle->getId();
    })->toArray();
    $users = $userRepository->findByAgeHobbiesAndLifestyles($ageMin, $ageMax, $user-
>getId(), $hobbiesArray, $lifestylesArray);
    $data = [];
    foreach ($users as $recommendation){
        if(!$doctrine->getRepository(Sympathy::class)->isUserInSympathy($user->getId(),
$recommendation->getId())
            && !$doctrine->getRepository(DarkList::class)->isUserInDarkList($user->getId(),
$recommendation->getId())){
            list($lat2, $lon2) = explode(",", $recommendation->getLocation(), 2);
            if ($this->haversineDistance($lat1, $lon1, $lat2, $lon2) <= $locationRange)
                $data[] = $recommendation;
        }
    }
    $data = $serializer->normalize($data, null, [
        AbstractNormalizer::ATTRIBUTES => [
            'id',
            'email',

```

```

        'name',
        'age',
        'work',
        'sex',
        'sexuality',
        'location',
        'description',
        'goal',
        'height',
        'lifeStyles' => ['name'],
        'Hobbies' => ['name'],
        'userImages' => ['url']
    ]
});
return new JsonResponse($data);
}

```

Лістинг методу addDarkList:

```

#[Route('/adddarklist/{id}', name: 'addDarkList', methods: "POST")]
public function addDarkList(ManagerRegistry $doctrine, UserRepository $userRepository,int
$id): Response
{
    $entityManager = $doctrine->getManager();
    $user = $this->getUser();
    $userBan = $userRepository->find($id);
    $darkLists = $doctrine->getRepository(DarkList::class)->isUserInDarkList(
        $user->getId(),
        $userBan->getId()
    );
    $sympathy = $doctrine->getRepository(Sympathy::class)->isUserInSympathy(
        $user->getId(),
        $userBan->getId()
    );
    if ($darkLists) {
        return new JsonResponse($darkLists);
    }
    if ($sympathy) {
        $entityManager->remove($sympathy);
        $entityManager->flush();
    }
    $darkList = new DarkList();
    $darkList->setUser($user);
    $darkList->setUserBan($userBan);
    $darkList->setComment("НЕ СПОДОБАВСЯ");
    $darkList->setDateTime(new \DateTimeImmutable());
    $entityManager->persist($darkList);
    $entityManager->flush();
    return new JsonResponse($darkList);
}

```

Лістинг методу addSympathy:

```

#[Route('/sympathy/{id}', name: 'addSympathy', methods: "POST")]

```

```

public function addSympathy(ManagerRegistry $doctrine, UserRepository $userRepository, int
$id): Response
{
    $entityManager = $doctrine->getManager();
    $user = $this->getUser();
    $userLike = $userRepository->find($id);
    $sympathy = $doctrine->getRepository(Sympathy::class)->isUserInSympathy(
        $user->getId(),
        $userLike->getId()
    );
    $darkLists = $doctrine->getRepository(DarkList::class)->isUserInDarkList(
        $user->getId(),
        $userLike->getId()
    );
    if ($sympathy) {
        return new JsonResponse($sympathy);
    }
    if ($darkLists) {
        $entityManager->remove($darkLists);
        $entityManager->flush();
    }
    $sympathy = new Sympathy();
    $sympathy->setUser($user);
    $sympathy->setUserLike($userLike);
    $sympathy->setStatus("Б ОЧИУВАНИ");
    $sympathy->setDateTime(new \DateTimeImmutable());
    $entityManager->persist($sympathy);
    $entityManager->flush();
    return new JsonResponse($sympathy);
}

```

Лістинг методу `findByAgeHobbiesAndLifestyles`:

```

public function findByAgeHobbiesAndLifestyles(int $minAge, int $maxAge, int $userId, array
$hobbies, array $lifestyles): array
{
    $qb = $this->createQueryBuilder('u')
        ->where('u.age >= :minAge')
        ->andWhere('u.age <= :maxAge')
        ->andWhere('u.id != :userId')
        ->setParameter('minAge', $minAge)
        ->setParameter('maxAge', $maxAge)
        ->setParameter('userId', $userId);
    if (!empty($hobbies)) {
        $qb->leftJoin('u.Hobbies', 'h', Join::WITH, 'h.id IN (:hobbies)')
            ->setParameter('hobbies', $hobbies);
    }
}

```

```

}
if (!empty($lifestyles)) {
    $qb->leftJoin('u.lifeStyles', 'l', Join::WITH, 'l.id IN (:lifestyles)')
    ->setParameter('lifestyles', $lifestyles);
}
$currentUser = $this->find($userId);
$currentUserSex = $currentUser->getSex();
$currentUserSexuality = $currentUser->getSexuality();
$currentUserGoal = $currentUser->getGoal();
if ($currentUserGoal === 'Серйозні стосунки') {
    if ($currentUserSex === 'Жінка' && $currentUserSexuality === 'Гетеросексуал(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality ===
'Гетеросексуал(ка)') {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Жінка' && $currentUserSexuality ===
'Гомосексуаліст(ка)') {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality ===
'Гомосексуаліст(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    }
}

```

```

}
if ($currentUserGoal === 'Серйозні стосунки') {
    $qb->andWhere('u.goal = :serious')
    ->setParameter('serious', 'Серйозні стосунки');
}
if ($currentUserGoal === 'Несерйозні стосунки') {
    if ($currentUserSex === 'Жінка' && $currentUserSexuality === 'Гетеросексуал(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality ===
'Гетеросексуал(ка)') {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Жінка' && $currentUserSexuality ===
'Гомосексуаліст(ка)') {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality ===
'Гомосексуаліст(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    }
}
}
if ($currentUserGoal === 'Несерйозні стосунки') {
    $qb->andWhere('u.goal = :casual')
    ->setParameter('casual', 'Несерйозні стосунки');
}

```

```

}
if ($currentUserGoal === 'Короткостроковий роман') {
    if ($currentUserSex === 'Жінка' && $currentUserSexuality === 'Гетеросексуал(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality ===
'Гетеросексуал(ка)') {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :heterosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('heterosexual', 'Гетеросексуал(ка)');
    } elseif ($currentUserSex === 'Жінка' && $currentUserSexuality ===
'Гомосексуаліст(ка)') {
        $qb->andWhere('u.sex = :female')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('female', 'Жінка')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    } elseif ($currentUserSex === 'Чоловік' && $currentUserSexuality ===
'Гомосексуаліст(ка)') {
        $qb->andWhere('u.sex = :male')
        ->andWhere('u.sexuality = :homosexual')
        ->setParameter('male', 'Чоловік')
        ->setParameter('homosexual', 'Гомосексуаліст(ка)');
    }
}
}
if ($currentUserGoal === 'Короткостроковий роман') {
    $qb->andWhere('u.goal = :shortTerm')
    ->setParameter('shortTerm', 'Короткостроковий роман');
}

if ($currentUserGoal === 'Нові друзі') {
    $qb->andWhere('u.goal = :newFriends')

```

```

        ->setParameter('newFriends', 'Нові друзі');
    }
    $qb->addSelect('COUNT(DISTINCT h.id) AS HIDDEN hobbiesCount')
        ->addSelect('COUNT(DISTINCT l.id) AS HIDDEN lifestylesCount')
        ->addSelect('COUNT(DISTINCT h.id) + COUNT(DISTINCT l.id) AS HIDDEN
totalCommonCount')
        ->groupBy('u.id')
        ->orderBy('totalCommonCount', 'DESC');
    return $qb->getQuery()->getResult();
}

```

Лістинг методу createChat:

```

#[Route('/chat/create/{id}', name: 'newChat')]
public function createChat(ManagerRegistry $doctrine, Request $request, $id)
{
    $entityManager = $doctrine->getManager();
    $otherUser = $doctrine->getRepository(User::class)->find($id);
    $user = $this->getUser();
    if (is_null($otherUser)) {
        throw new \Exception("Користувача не знайдено");
    }
    if ($otherUser->getId() === $this->getUser()->getId()) {
        throw new \Exception("Це глибоко, але ви не можете створити розмову з самим собою");
    }
    $chat = $doctrine->getRepository(Chat::class)->findChatByUsers(
        $otherUser->getId(),
        $this->getUser()->getId()
    );
    if ($chat) {
        throw new \Exception("Чат вже існує");
    }
    $chat = new Chat();
    $chat->setStatus('Новий чат');
    $chat->setUser($this->getUser());
    $chat->setUserLike($otherUser);
    $sympathy = $doctrine->getRepository(Sympathy::class)->isUserInSympathy(
        $user->getId(),
        $otherUser->getId()
    );
    $entityManager->getConnection()->beginTransaction();
    try {
        $entityManager->persist($chat);
        if ($sympathy) {
            $entityManager->remove($sympathy);
        }
        $entityManager->flush();
    }
}

```

```

        $entityManager->commit();
    } catch (\Exception $e) {
        $entityManager->rollback();
        throw $e;
    }
    return $this->json([
        'id' => $chat->getId()
    ], Response::HTTP_CREATED, [], []);
}

```

Лістинг методу newMessage:

```

#[Route('/chat/{id}', name: 'newMessage', methods: ["POST"])]
public function newMessage(HubInterface $hub, Request $request, $id, SerializerInterface
$serializer, ManagerRegistry $doctrine): Response
{
    $entityManager = $doctrine->getManager();
    $chat = $doctrine->getRepository(Chat::class)->find($id);
    $user = $this->getUser();
    $text = $request->get('message', null);
    $image = $request->files->get('image', null);
    $message = new Message();
    $message->setMessage($text);
    $message->setStatus('UNREAD');
    $message->setUser($user);
    $message->setDateTime(new \DateTimeImmutable());
    if($image){
        $newImageName = uniqid() . '.' . $image->guessExtension();
        $image->move('images', $newImageName);
        $message->setImage($newImageName);
    }
    $chat->addMessage($message);
    $chat->setLastMessage($message);
    $entityManager->persist($message);
    $entityManager->persist($chat);
    $entityManager->flush();

    $message->setMine(false);
    $messageSerialized = $serializer->serialize($message, 'json', [
        'attributes' => ['id', 'image', 'message', 'dateTime', 'mine', 'chat' => ['id']]
    ]);
    $participants = [$chat->getUser(), $chat->getUserLike()];
    $otherUser = null;
    foreach ($participants as $participant) {
        if ($user !== $participant) {
            $otherUser = $participant;
            break;
        }
    }
    $update = new Update(
        [
            sprintf("/chat/%s", $chat->getId()),
            sprintf("/chat/%s", $otherUser->getEmail()),

```

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

    ],
    $messageSerialized,
    true
);
$hub->publish($update);
$message->setMine(true);
return $this->json($message, Response::HTTP_CREATED, [], [
    'attributes' => ['id', 'image', 'message', 'dateTime', 'mine']
]);
}

```

Лістинг методу newMessage:

```

#[Route('/chat/{id}', name: 'newMessage', methods: ["POST"])]
public function newMessage(HubInterface $hub, Request $request, $id, SerializerInterface
$serializer, ManagerRegistry $doctrine): Response
{
    $entityManager = $doctrine->getManager();
    $chat = $doctrine->getRepository(Chat::class)->find($id);
    $user = $this->getUser();
    $text = $request->get('message', null);
    $image = $request->files->get('image', null);
    $message = new Message();
    $message->setMessage($text);
    $message->setStatus('UNREAD');
    $message->setUser($user);
    $message->setDateTime(new \DateTimeImmutable());
    if($image){
        $newImageName = uniqid() . '.' . $image->guessExtension();
        $image->move('images', $newImageName);
        $message->setImage($newImageName);
    }
    $chat->addMessage($message);
    $chat->setLastMessage($message);
    $entityManager->persist($message);
    $entityManager->persist($chat);
    $entityManager->flush();
    $message->setMine(false);
    $messageSerialized = $serializer->serialize($message, 'json', [
        'attributes' => ['id', 'image', 'message', 'dateTime', 'mine', 'chat' => ['id']]
    ]);
    $participants = [$chat->getUser(), $chat->getUserLike()];
    $otherUser = null;
    foreach ($participants as $participant) {
        if ($user !== $participant) {
            $otherUser = $participant;
            break;
        }
    }

    $update = new Update(
        [
            sprintf("/chat/%s", $chat->getId()),

```

```

        sprintf("/chat/%s", $otherUser->getEmail()),
    ],
    $messageSerialized,
    true
);
$hub->publish($update);

$message->setMine(true);
return $this->json($message, Response::HTTP_CREATED, [], [
    'attributes' => ['id', 'image', 'message', 'dateTime', 'mine']
]);
}

```

		Грушевицький С.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Гордієнко І. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		91