

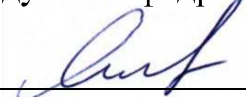
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: «Розроблення інформаційної системи для агенції «Випадкова зустріч»

Виконав: студент групи



Резніченко М.М.

(підпис)

Керівник роботи:

Викладач к.т.н.

(посада, науковий ступінь вчене звання)



Книрік Н.Р.

(підпис)

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

Спеціальність 122 "Комп'ютерні науки"

Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

" ____ " _____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття ступеня вищої освіти "бакалавр"

Студенту Резніченко Миколі Максимовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Розроблення інформаційної системи для агенції «Випадкова зустріч»

Керівник роботи Книрик Наталя Ромуальдівна

Затверджені наказом ректора № ____ -уч від " ____ " березня 2025 року.

2. Термін подання роботи: 20 червня 2025 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів): Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та

5. Перелік презентаційних матеріалів

[illegible]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Книрік Н.Р.	03.02.2025	22.02.2025
2	Книрік Н.Р.	22.02.2025	21.03.2025
3	Книрік Н.Р.	21.03.2025	25.04.2025
4	Книрік Н.Р.	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проєкту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент

(підпис)

Резніченко М.М.

(ПІБ)

Керівник роботи

(підпис)

Книрик Н.Р.

(ПІБ)

АНОТАЦІЯ

У кваліфікаційній роботі розглянуто проектування та розробку інформаційної системи агентства знайомств, яка спрямована на автоматизацію процесів підбору партнерів, організації зустрічей та комунікації між користувачами. У роботі проведено детальний аналіз предметної області та виявлено ключові проблеми існуючих платформ: поверхнева фільтрація, слабка модерація, відсутність механізмів для офлайн-зустрічей, низький рівень персоналізації та зворотного зв'язку.

Запропонована інформаційна система реалізована у вигляді вебзастосунку на основі клієнт-серверної архітектури, з використанням Java, Spring Boot, Hibernate, HTML/CSS/JavaScript та MariaDB. Система забезпечує створення профілю, проходження психологічного тестування, пошук сумісних партнерів, чат, планування реальних зустрічей, ведення статистики та аналітики. Також реалізовані функції модерації, безпеки, підписки, архіву та реферальної системи.

Робота викладена на 74 сторінках машинописного тексту й налічує 16 рисунків, 3 таблиці, список використаних джерел з 10 найменувань та 2 додатки.

Ключові слова: інформаційна система, агентство знайомств, веб-застосунок, інтерфейс користувача

ABSTRACT

The qualification work considers the design and development of an information system for a dating agency, which is aimed at automating the processes of partner selection, meeting organization, and communication between users. The work provides a detailed analysis of the subject area and identifies key problems of existing platforms: superficial filtering, weak moderation, lack of mechanisms for offline meetings, low level of personalization and feedback.

The proposed information system is implemented as a web application based on client-server architecture, using Java, Spring Boot, Hibernate, HTML/CSS/JavaScript, and MariaDB. The system provides profile creation, psychological testing, search for compatible partners, chat, planning real meetings, statistics, and analytics. Moderation, security, subscription, archive, and referral system functions are also implemented.

The work is presented on 74 pages of typewritten text and includes 16 figures, 3 tables, a list of sources used with 10 names, and 2 appendices.

Key-words: information system, dating agency, web application, user interface

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. РОЗРОБКА КОНЦЕПЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДІЯЛЬНОСТІ АГЕНТСТВА ЗУСТРІЧЕЙ	9
1.1 Опис об'єкту дослідження і аналіз предметної області	9
1.2 Формування вимог до інформаційної системи	16
1.3 Розробка концепції інформаційної системи	19
РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ	25
2.1 Загальносистемні рішення	25
2.1.1 Загальний опис системи	25
2.1.2 Розробка діаграми варіантів використання	29
2.2 Рішення з інформаційного забезпечення	31
2.2.1 Визначення сутностей інформаційної системи	32
2.2.2 Розробка логічної моделі даних	35
2.2.3 Розробка фізичної моделі даних	35
2.3 Рішення з програмного забезпечення	38
2.4 Рішення з технічного забезпечення	40
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЕКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ	41
3.1 Розробка інструкції користувача	50
РОЗДІЛ 4. ОХОРОНА ПРАЦІ	51
ВИСНОВКИ	63
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	65
ДОДАТОК А – Програмний код	66
ДОДАТОК Б – Інструкція користувача	74

ВСТУП

У кваліфікаційній роботі розглядається проектування та розробка інформаційної системи для агентства зустрічей із використанням сучасних технологій, таких як Spring Framework для серверної частини та JavaFX/веб-технології для клієнтської. Основна мета дослідження — створення ефективного інструменту, який забезпечить зручність і безпеку взаємодії користувачів, а також надійну обробку та збереження даних.

- Запропонована система надаватиме такі функціональні можливості:
- Перегляд списку клієнтів та їхніх профілів;
- Організація та управління зустрічами між клієнтами;
- Інтегрований чат для комунікації користувачів;
- Аналітичні інструменти для статистичного аналізу взаємодії клієнтів;
- Архівування зустрічей та можливість перегляду історії.

Для досягнення поставленої мети необхідно вирішити ряд завдань: провести аналіз предметної області, дослідити існуючі рішення, визначити основні проблеми та вимоги до системи, розробити концепцію її реалізації, а також здійснити проектування та впровадження програмного забезпечення.

Дослідження охоплює всі етапи розробки: від аналізу до створення прототипу, тестування та оцінки ефективності запропонованого рішення. Це дозволить продемонструвати процес створення інформаційної системи, здатної ефективно автоматизувати роботу агентства зустрічей та покращити взаємодію між клієнтами.

РОЗДІЛ 1 РОЗРОБКА КОНЦЕПЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДІЯЛЬНОСТІ АГЕНТСТВА ЗУСТРІЧЕЙ

1.1 Опис об'єкту дослідження і аналіз предметної області

В умовах глобальної цифровізації суспільства значно зросла роль Інтернету в організації соціальної взаємодії. Мобільний зв'язок, соціальні мережі, цифрові платформи стали не лише зручним, але й звичним середовищем для комунікації, обміну інформацією, спільної діяльності та, зокрема, налагодження міжособистісних стосунків. За даними дослідження, у 2023 році понад 70% українців віком до 35 років щоденно користуються Інтернетом для спілкування, включаючи пошук нових знайомств [1].

Однак ще до епохи цифрових технологій у суспільстві існували альтернативні способи знайомств. Одним з таких традиційних інструментів були агентства знайомств або клуби знайомств, які працювали офлайн та надавали послуги з підбору сумісних партнерів. Такі установи зазвичай мали у своєму розпорядженні анкетні дані клієнтів, проводили особисті співбесіди, іноді — психологічне тестування. Проте ефективність їх роботи була обмежена географічно, мала низьку швидкість оновлення даних, залежала від кваліфікації персоналу та не дозволяла автоматизувати процеси.

З поширенням Інтернету, розвитком мобільних технологій і широким доступом до смартфонів онлайн-знайомства поступово стали частиною повсякденного життя. Сучасні сервіси надають змогу користувачам швидко створювати профілі, налаштовувати фільтри для пошуку потенційних партнерів, спілкуватися у режимі реального часу та навіть отримувати рекомендації на основі заданих параметрів.

Зазвичай організаційна структура подібних сервісів має наступний вигляд:

Адміністративний відділ — координує діяльність агентства, забезпечує стратегічне планування та ухвалення рішень. Він також визначає політику розвитку сервісу, реагує на тенденції ринку та пропонує нові функціональні можливості, такі як інтеграція з іншими платформами чи вдосконалення механізмів рекомендацій, щоб зберегти конкурентоспроможність на ринку онлайн-знайомств.

Відділ підтримки клієнтів — здійснює реєстрацію нових користувачів, супроводжує процес взаємодії та консультує щодо використання сервісу. Цей відділ має важливу роль у забезпеченні зручності для користувачів, підтримці безпеки, а також у вирішенні проблем із фейковими акаунтами та конфліктами між користувачами. Крім того, вони здійснюють навчання нових користувачів та допомагають адаптуватися до специфіки платформи, що є важливим для покращення досвіду користувачів.

Технічний відділ — відповідає за функціонування та стабільність інформаційної системи, включаючи бази даних, сервери та інтерфейси. Це включає моніторинг безпеки платформи, запобігання технічним збоєм, а також впровадження нових технологій для покращення алгоритмів підбору партнерів та оптимізації системи обміну повідомленнями.

Юридичний відділ — контролює дотримання законодавчих вимог, особливо у сфері захисту персональних даних. Він також здійснює юридичний супровід платформи, забезпечує відповідність нормам міжнародного та національного законодавства, що регулює питання конфіденційності, безпеки та інтелектуальної власності.

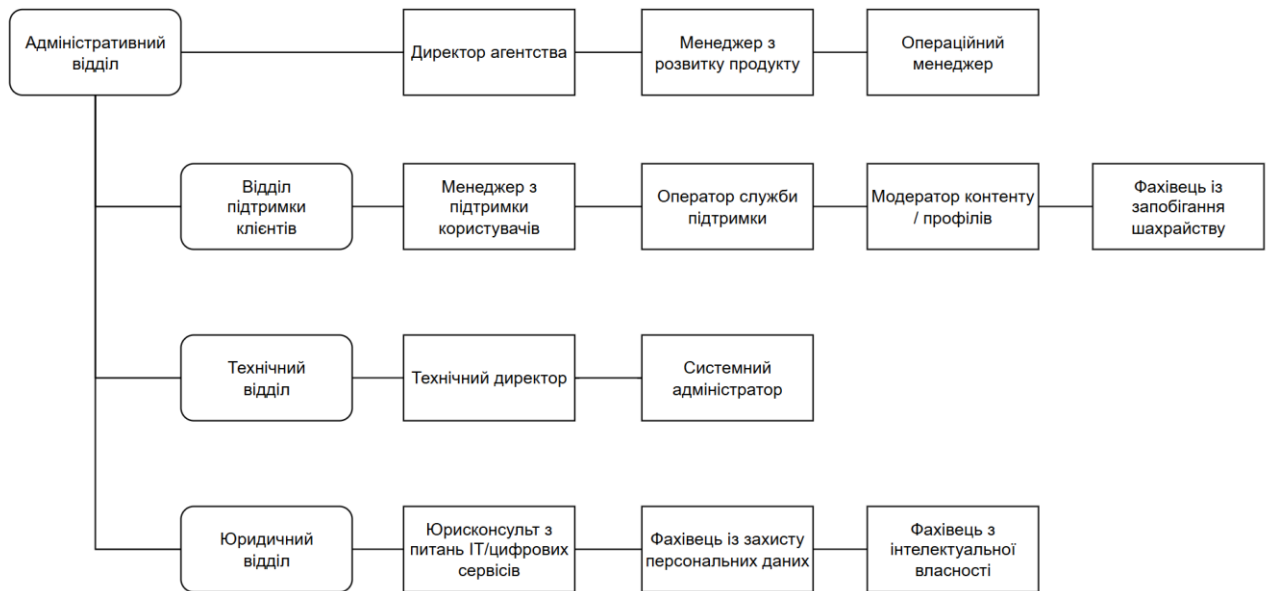


Рис. 1.1 – Організаційна структура агентства

1. Адміністративний відділ

- Директор агентства — загальне керівництво, прийняття стратегічних рішень.
- Менеджер з розвитку продукту (Product Manager) — аналіз ринку, формування бачення розвитку функціоналу платформи.
- Операційний менеджер (Operations Manager) — координація внутрішніх процесів, комунікація між відділами.

2. Відділ підтримки клієнтів

- Менеджер з підтримки користувачів (Customer Support Manager) — організація служби підтримки, контроль за якістю обслуговування.
- Оператор служби підтримки (Support Specialist / Chat Agent) — відповіді на запити користувачів у чаті, email, телефоні.
- Модератор контенту / профілів (Moderator) — перевірка фото, анкет, контроль відповідності до правил.

- Фахівець із запобігання шахрайству (Fraud Prevention Specialist) — виявлення фейкових акаунтів, реагування на підозрілі дії.

3. Технічний відділ

- Технічний директор (СТО) — технічне керівництво, вибір архітектурних рішень.
- Системний адміністратор — підтримка серверів та мережевої інфраструктури.

4. Юридичний відділ

- Юрисконсульт з питань ІТ/цифрових сервісів — супровід проєктів, консультації щодо ІТ-законодавства.
- Фахівець із захисту персональних даних (Data Protection Officer / DPO) — контроль за дотриманням законів про конфіденційність (наприклад, GDPR).

Для цієї структури розглянемо типові рішення з досліджуваної предметної області.

До найвідоміших платформ належать Tinder, Badoo, eHarmony, OkCupid, Hinge та інші. Їхньою беззаперечною перевагою є масштабність: багатомільйонна база користувачів, висока швидкість доступу до анкет, зручні мобільні додатки та велике географічне охоплення. Втім, попри очевидну зручність, ці сервіси не позбавлені суттєвих проблем, які суттєво впливають на якість міжособистісної взаємодії та знижують ефективність у формуванні тривалих стосунків.

Однією з найбільш характерних тенденцій для подібних платформ є акцент на візуальні характеристики. Наприклад, Tinder — один із найпопулярніших у світі застосунків для знайомств — використовує механіку швидкого взаємного «свайпу», коли рішення щодо симпатії приймається здебільшого на основі

фотографії. Такий підхід формує культуру поверхневого вибору, де зовнішність відіграє ключову роль, а глибші особистісні якості залишаються поза увагою. У результаті значна частина контактів не переходить у реальне спілкування: згідно з дослідженнями, понад 70% користувачів цієї платформи припиняють діалог одразу після першого обміну повідомленнями [1].

Інша важлива проблема — це безпека користувачів. Попри значну аудиторію та розвинену інфраструктуру, платформи на кшталт Badoo неодноразово зазнавали критики через велику кількість фейкових акаунтів та недостатню ефективність механізмів перевірки особистості. Подібні акаунти створюють сприятливе середовище для онлайн-шахрайства, введення в оману, маніпуляцій, а в окремих випадках — для вчинення злочинів сексуального характеру або виманювання персональних даних.

Згідно з результатами аналітичного дослідження, проведеного у Великій Британії у 2023 році, понад 71% чоловіків та 47% жінок, які користуються додатками для знайомств, зазначили, що стикалися з фейковими профілями під час пошуку партнера. Така статистика свідчить про масштабність проблеми й актуальність її вирішення не лише в межах конкретної платформи, а й для індустрії загалом [2].

Складність користування платформою є ще однією перешкодою, з якою стикаються користувачі. Сервіс eHarmony, орієнтований на створення довготривалих стосунків, вирізняється дуже детальною системою реєстрації: анкета може містити до 200 питань. Хоча така глибина може бути корисною для точнішого підбору пари, вона часто відштовхує новачків, особливо молодь. До того ж, багато функцій eHarmony доступні лише на платній основі, а сам інтерфейс недостатньо інтуїтивний, що робить взаємодію зі службою складною та втомливою [3].

Ще однією серйозною вадою є повна відсутність механізмів організації офлайн-зустрічей. Усі згадані сервіси фактично залишають процес домовленостей про реальну зустріч поза межами власної платформи. Жоден з них не пропонує вбудованої системи планування подій, синхронізації з календарем, інтеграції з картами чи вибору місця для зустрічі. Це часто ускладнює процес переходу з онлайн в офлайн і знижує шанси на успішне продовження знайомства. За результатами дослідження компанії SoulMatcher, понад три чверті користувачів зазначили, що їм бракує такого функціоналу [4].

До переліку проблем слід також віднести ігнорування психологічних чинників при підборі пари. Майже всі популярні сервіси обмежуються лише базовими фільтрами: вік, стать, місцезнаходження та кілька інтересів. Психологічна сумісність, особливості темпераменту, стилі спілкування або емоційні реакції майже не враховуються. Хоча деякі платформи, як-от eHarmony, роблять спроби застосовувати психологічні опитування, вони не завжди ефективно адаптовані до локальних культурних реалій і часто не мають україномовної підтримки, що також знижує їхню популярність серед українських користувачів.

Загалом, аналіз показує, що найбільшою проблемою сучасних платформ є низький рівень персоналізації та брак уваги до психологічних аспектів взаємодії. Саме ці чинники найчастіше призводять до короткотривалих знайомств або повного припинення комунікації вже на початковому етапі. Ці недоліки потребують системного вирішення у межах нового підходу до організації знайомств в цифровому середовищі.

Іншою поширеною проблемою є відсутність контролю якості створених профілів. Користувачі можуть вказувати неправдиву інформацію, а також змінювати поведінку в реальному спілкуванні порівняно з онлайн-образом. Це знижує довіру до сервісу. За дослідженням компанії SoulMatcher, 76%

користувачів додатків для знайомств надають перевагу зустрічам наживо, однак не почуваються впевнено в безпечності організації таких подій [3].

Ще одним важливим аспектом є відсутність зворотного зв'язку, який дозволив би платформі навчатися на основі дій користувача та покращувати алгоритми добору. Успіх платформи залежить не тільки від функціоналу, а й від досвіду користування інтерфейсом, інклюзивності, а також адаптивності до різних вікових груп [4, с. 38].

Отже, в умовах виявлених недоліків виникає потреба у створенні нової інформаційної системи, що об'єднає переваги традиційних методів знайомства (наприклад, оцінку сумісності) з можливостями цифрових технологій (автоматизація, масштабованість, інтерактивність).

На основі проведеного аналізу предметної області зроблено кілька важливих висновків. У сучасному суспільстві знайомства через Інтернет стали повсякденною практикою, а відповідні сервіси перетворилися на один із головних каналів соціальної взаємодії. Водночас, детальний огляд існуючих платформ виявив численні недоліки: поверхневність алгоритмів добору, слабкий рівень безпеки, відсутність функціоналу для організації офлайн-зустрічей, складні або перевантажені інтерфейси, низьку якість персоналізації, а також ігнорування психологічних чинників сумісності між користувачами.

Виявлені недоліки обґрунтовують необхідність створення нової інформаційної системи, яка має об'єднати кращі риси традиційного агентства знайомств (індивідуальний підхід, структурованість процесу, етична відповідальність) з перевагами сучасних цифрових технологій (масштабованість, автоматизація, інтерактивність, доступність). У центрі уваги майбутнього рішення повинна бути людино центричність — врахування психологічних, комунікативних та соціальних особливостей кожного користувача.

У процесі створення інформаційної системи важливим є врахування сучасних підходів до проектування. Інформаційна система розглядається як складна сукупність функціональних, технічних та організаційних компонентів, що повинні бути узгодженими між собою на всіх етапах життєвого циклу. [5]

Проектування подібної системи вимагає дотримання сучасних принципів побудови інформаційних систем: гнучкість, надійність, масштабованість, а також відповідність вимогам безпеки та законодавства.

1.2 Формування вимог до інформаційної системи

На основі аналізу предметної області до інформаційної системи були сформовані наступні вимоги, що забезпечують ефективну підтримку процесів знайомства, підбору партнерів та організації зустрічей між користувачами.

Проектована інформаційна система має функціонувати у вигляді вебзастосунку з клієнтською та серверною частинами, забезпечуючи користувачам зручний інтерфейс для створення профілю, взаємодії з іншими учасниками, призначення зустрічей і оцінки результатів спілкування. Основними завданнями системи є полегшення пошуку сумісних партнерів, організація комунікації між ними та підвищення безпеки взаємодії.

Функціональні вимоги:

- Реєстрація та авторизація користувачів із перевіркою введених даних та верифікацією профілю;
- Створення та редагування профілю користувача з розширеною анкетною, включаючи психологічне тестування;
- Пошук і фільтрація профілів за заданими параметрами (вік, інтереси, сумісність, геолокація тощо);
- Перегляд профілів та можливість надсилання запитів на зустріч;

- Інтегрований інструмент планування зустрічей із вибором дати, часу та місця;
- Вбудований чат для обміну повідомленнями з підтримкою прикріплення зображень;
- Ведення історії взаємодій, відображення статистики, фіксація результатів зустрічей;
- Механізм оцінки взаємодій для вдосконалення алгоритмів підбору;

Нефункціональні вимоги:

- Продуктивність: стабільна робота системи за високого навантаження;
- Безпека: шифрування критичних даних, використання, захист від атак;
- Конфіденційність: відповідність законодавству України щодо захисту персональних даних;
- Масштабованість: модульна архітектура, що дозволяє легко розширювати функціональність;
- Доступність: адаптивний інтерфейс, що коректно працює на ПК, планшетах і смартфонах;
- Надійність: автоматичне резервне копіювання, журналювання помилок, система сповіщень адміністратора;

Програмні вимоги:

- Клієнтська частина із використанням фреймворків для створення сучасного, інтерактивного вебінтерфейсу;
- Серверна частина що забезпечить, логіку обробки запитів та взаємодію з базою даних;

- База даних — для зберігання профілів користувачів, повідомлень, зустрічей, історії взаємодій тощо;

Технічні вимоги:

- Архітектура: клієнт-серверна, з розмежуванням функцій між фронтендом і бекендом;
- Забезпечення доступу: система розмежування прав доступу за ролями (гость, користувач, адміністратор);

Таким чином, розробка системи базується на комплексному підході до організації знайомств в онлайн-середовищі з урахуванням актуальних вимог.

Для реалізації веб-орієнтованої системи агентства знайомств була застосована клієнт-серверна архітектура, що передбачає розділення логіки між клієнтським додатком (веб-інтерфейсом) та сервером. Обрана структура забезпечує масштабованість, гнучкість розробки, зручність тестування та підтримки та використовує наступні технічні засоби:

Серверна частина (хостинг або локальний сервер):

- Процесор: мінімум 4 ядра (рекомендовано Intel Xeon або AMD Ryzen);
- Оперативна пам'ять: від 8 ГБ;
- Жорсткий диск: SSD від 100 ГБ, з можливістю резервного копіювання;
- Операційна система: Windows 10;
- Мережева інфраструктура: безперебійне інтернет-з'єднання, статична IP-адреса або домен;

Клієнтська частина:

- Веббраузери: підтримка сучасних браузерів (Google Chrome, Mozilla Firefox, Microsoft Edge);
- Мобільна підтримка: адаптивний інтерфейс для Android/iOS;

- Засоби розробки та супроводу:
- Інтегроване середовище розробки (IDE): IntelliJ IDEA.

На основі вимог розроблюється концепція інформаційної системи.

1.3 Розробка концепції інформаційної системи

На основі сформованих вимог побудована модель концепції інформаційної системи.

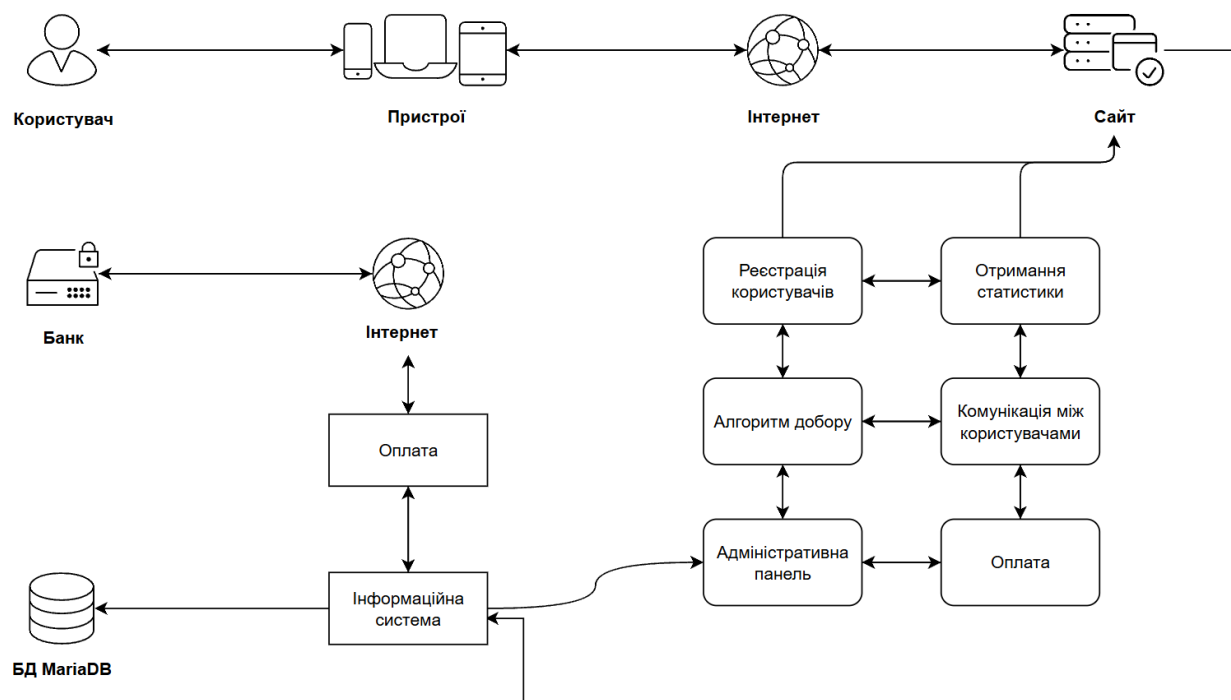


Рис. 1.3 – Модель концепції ІС

Взаємодія користувача з інформаційною системою агентства знайомств здійснюється через вебінтерфейс або мобільну версію платформи, реалізовану як SPA-застосунок (single-page application) з підтримкою адаптивної верстки. Інформаційна система побудована за принципом клієнт-серверної архітектури: інтерфейс користувача відправляє запити через інтернет до серверної частини,

яка відповідає за обробку запитів, зберігання даних, генерацію рекомендацій і організацію взаємодії між учасниками.

Платформа дозволяє користувачеві пройти реєстрацію та анкетування, сформувати персоналізований профіль, пройти верифікацію та отримати рекомендації потенційно сумісних партнерів. В основі механізму добору лежить поєднання базових фільтрів (вік, стать, локація, інтереси) з поглибленими аналітичними модулями, що враховують результати психологічного тестування, стиль комунікації та інші соціально-поведінкові параметри.

На відміну від більшості існуючих рішень, нова система передбачає інтегрований модуль планування реальних зустрічей. Користувачі можуть обирати запропоновані місця поблизу, узгоджувати час, надсилати запрошення та отримувати автоматичні нагадування. Завдяки цьому платформа не лише сприяє онлайн-комунікації, але й супроводжує користувача до моменту організації реальної зустрічі.

Для підтримки безперервної комунікації реалізовано вбудований чат з можливістю надсилання текстових, голосових та фото повідомлень. Передбачена функція надсилання повідомлень лише після взаємної симпатії або підтвердження запиту на спілкування. Це дозволяє зберігати комфортне та безпечне середовище для користувачів.

Окрему увагу приділено безпеці та модерації: кожен профіль проходить перевірку, а в разі скарг активується система пріоритетного реагування служби підтримки. Уся персональна інформація зберігається в захищеній базі даних PostgreSQL, а кешування виконується через Redis, що забезпечує високу швидкодію системи та стійкість до навантажень.

Платформа підтримує гнучку модель підписки, яка надає доступ до розширеного функціоналу: перегляд статистики взаємодій, розширений пошук, пріоритет у рекомендаціях, доступ до аналітики анкет. Оплата відбувається через

інтеграцію з банківськими системами та платіжними шлюзами, що дозволяє користувачам обирати зручний спосіб.

Для стимулювання залучення нових користувачів реалізовано реферальну систему: користувачі можуть отримувати знижки або бонуси за запрошення друзів. Така модель підтримує органічне зростання спільноти без суттєвих витрат на рекламу.

Платформа фіксує історію активності користувача, зберігає аналітику щодо переглядів анкет, кількості спілкувань, успішних зустрічей тощо. Ці дані доступні в особистому кабінеті у форматі дашборду — наочні графіки, діаграми та показники допомагають користувачеві краще розуміти ефективність свого досвіду знайомств.

Оновлення системи здійснюються на серверному боці, що дозволяє вдосконалювати функціонал без потреби оновлювати застосунок вручну. Це підвищує стабільність роботи платформи та дозволяє швидко адаптуватися до запитів користувачів або змін у законодавстві (наприклад, щодо обробки персональних даних).

У перспективі платформа може бути доповнена міні-застосунками, які надаватимуть розширений функціонал, як-от: віртуальні побачення, групові заходи, тематичні вікторини для зближення, що дозволить ще краще адаптувати платформу до потреб цільової аудиторії.

Система складається з таких основних модулів:

- Модуль користувачів (реєстрація, перегляд профілів, редагування анкет);
- Модуль зустрічей та добору (планування, перегляд, архівування);
- Модуль комунікацій (чат між користувачами);

- Модуль статистики (статистика зустрічей, рейтинг сумісності користувачів);
- Адміністративна панель (модерування користувачів);

Вибір програмних рішень

На основі вимог до функціональності, безпеки, масштабованості та зручності обслуговування обрано такі програмні засоби:

Мова програмування (Back-end):

Java 17/21 — сучасна, типобезпечна мова, що підтримує об'єктно-орієнтоване програмування та має широку екосистему для створення надійних вебдодатків.

Фреймворки (Back-end):

- Spring Boot — дозволяє швидко створювати автономні, масштабовані вебсервіси з мінімальною конфігурацією.
- Hibernate (JPA) — для об'єктно-реляційного відображення (ORM), що забезпечує зручну роботу з базою даних на основі сутностей.

Фронтенд (клієнтська частина):

- HTML5 / CSS3 / JavaScript — базові вебтехнології для побудови інтерфейсу користувача.
- Bootstrap або Tailwind CSS — для адаптивного та сучасного дизайну.
- Thymeleaf — шаблонізатор або бібліотека для побудови динамічного інтерфейсу.

База даних:

MariaDB — надійна реляційна система управління базами даних із відкритим вихідним кодом, сумісна з MySQL, що забезпечує зберігання структурованих даних про користувачів, профілі, зустрічі тощо.

Інтеграції:

- Google Maps API — для вибору місця зустрічі на карті.
- JavaMail API — для надсилання повідомлень електронною поштою (наприклад, підтвердження реєстрації).
- Spring Security — для реалізації автентифікації, авторизації та контролю доступу.
- Інші технології:
- Lombok — для зменшення шаблонного коду в Java-класах.
- Maven— для керування залежностями та збірки проєкту.

Вибір технічних засобів

Для розгортання та експлуатації інформаційної системи необхідно забезпечити відповідне апаратне та мережеве середовище. Орієнтовна конфігурація наведена нижче:

Серверна частина (хостинг або локальний сервер):

- Процесор: мінімум 4 ядра (рекомендовано Intel Xeon або AMD Ryzen).
- Оперативна пам'ять: від 8 ГБ.
- Жорсткий диск: SSD від 100 ГБ, з можливістю резервного копіювання.
- Операційна система: Windows 10.
- Мережева інфраструктура: безперебійне інтернет-з'єднання, статична IP-адреса або домен.

Клієнтська частина:

- Веббраузери: підтримка сучасних браузерів (Google Chrome, Mozilla Firefox, Microsoft Edge).
- Мобільна підтримка: адаптивний інтерфейс для Android/iOS.
- Засоби розробки та супроводу:

- Інтегроване середовище розробки (IDE): IntelliJ IDEA.

На основі проведених досліджень сформовано вимоги на основі яких розроблена концепція інформаційної системи. В проекті буде здійснена реалізація концепції інформаційної системи яка дозволить досягнути поставленої мети з підвищення ефективності об'єкта.

РОЗДІЛ 2 РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Загальносистемні рішення

На основі концептуальної моделі, розробленої в попередньому розділі, сформовано загальносистемні рішення, які визначають архітектуру, структуру, принципи функціонування та основні компоненти інформаційної системи агентства знайомств. Ці рішення є основою для побудови ефективного, масштабованого та безпечного веб орієнтованого сервісу, здатного обслуговувати велику кількість користувачів і забезпечувати комфортну взаємодію на всіх етапах знайомства — від реєстрації до організації офлайн-зустрічей.

2.1.1 Загальний опис системи

Інформаційна система агентства знайомств реалізована як сучасна веб-платформа, заснована на клієнт-серверній архітектурі. Користувачі взаємодіють із системою через веб-інтерфейс. Серверна частина обробляє запити, виконує логіку, зберігає дані та формує відповіді клієнту.

Клієнт при додаванні в БД повинен додати інформацію про себе (фото, ім'я, прізвище, вік, інтереси (характеристики, які риси характеру йому притаманні, хобі), країна, мова, вимоги до партнера (які риси характеру він бажає знайти в свого партнера), вказати стать.

Для кожного клієнта є можливість обрати потенційних кандидатів на зустріч, відсортуювши попередньо кандидатів за рисами характеру.

У кожного клієнта є можливість призначити зустріч з іншим клієнтом. Для цього буде необхідно додати інформацію про час зустрічі, країну зустрічі та ім'я обох клієнтів. Кожну зустріч можливо заархівувати або видалити. Клієнту також

доступна статистика щодо кількості зустрічей за країнами. Для кожної країни буде виводитися кількість зустрічей, що були зареєстровані у ній.

В архіві зберігаються імена клієнтів з пар, що відбулись, та країни, де вони були утворені. З архіву пари видаляти не можна.

Таблиця 2.1 - Матриця подій

Сутність	Подія	Реакція системи
Клієнт	Додавання клієнта	Клієнт при додаванні в БД повинен додати інформацію про себе (фото, ім'я, прізвище, вік, інтереси (характеристики, які риси характеру йому притаманні, хобі), вимоги до партнера (які риси характеру він бажає знайти в свого партнера), вказати стать. Кожна характеристика обирається з наданого списку.
Клієнт	Вивід підходящих кандидатів	Для кожного клієнта є можливість обрати потенційних наречених, відсортувавши попередньо кандидатів за рисами характеру.

Клієнт	Призначення зустрічі	Після обирання відповідних кандидатів, є можливість призначити зустріч. Попередньо для цього буде необхідно додати інформацію про час зустрічі. До зустрічі також буде передаватися інформація про клієнта (хобі, ім'я, прізвище, країна, мова).
Клієнт	Архівування зустрічі	В архів переносяться пари, що відбулись після вдалої зустрічі. З архіву пари видаляти не можна.
Клієнт	Чат	Кожний клієнт має можливість домовитися про зустріч з іншим клієнтом, а потім за бажання призначити йому зустріч.
Клієнт	Зарахування зустрічі до статистики	При плануванні зустрічей, країна цієї зустрічі використовується для

		збору статистичних даних про країни з зустрічами клієнта, відсортованими за зростанням кількості зустрічей що в них відбулися.
Клієнт	Оновлення даних клієнта	
Клієнт	Вивід кількості пар утворених у кожній країні	Статистика щодо кількості утворених у кожній країні пар. У кожного клієнта є можливість переглянути кількість зустрічей для кожної обраної ним країни.

Система орієнтована на такі ключові задачі:

- **створення персоналізованих профілів** з урахуванням характеристик та потреб користувача;
- **підбір сумісних партнерів** за допомогою багаторівневої системи фільтрації та алгоритмів аналізу сумісності;
- **організація та планування зустрічей** із можливістю вибору часу, дати та місця (через інтеграцію з картографічними сервісами);

- **забезпечення безпечної комунікації** між учасниками через вбудований чат із можливістю надсилання повідомлень лише після взаємної згоди;
- **надання статистики та аналітики** про успішність знайомств, кількість контактів, переглядів тощо;
- **модерування профілів** та оперативне реагування на скарги завдяки інструментам адміністративної панелі.

2.1.2 Розробка діаграми варіантів використання

На основі функціональних вимог, сформованих у розділі 1.2, була розроблена діаграма варіантів використання, що ілюструє основні сценарії взаємодії користувачів із інформаційною системою агентства знайомств. Діаграма побудована відповідно до стандартів UML та відображає зв'язки між акторами (користувачами) і варіантами використання (сценаріями дій), які реалізуються в системі.

Діаграма варіантів використання (Use Case Diagram) — це один із типів діаграм у мові моделювання UML, що дозволяє візуально представити функціональність системи з точки зору зовнішнього користувача. Вона показує, які саме дії можуть виконувати користувачі системи, та як ці дії пов'язані між собою і з внутрішніми процесами системи. Основними елементами діаграми є актор (людина або інша система, що взаємодіє з ІС), варіанти використання (функції, доступні актору), а також зв'язки типу включення, розширення та узагальнення між варіантами та акторами. [10]



Рисунок 2. 1 Діаграма варіантів використання

Основними акторами системи є:

Незареєстрований користувач — потенційний клієнт, який має обмежений доступ до функціоналу системи;

Зареєстрований користувач — повноцінний учасник платформи, який має доступ до більшості функцій;

Адміністратор — співробітник, відповідальний за супровід, модерацію та адміністрування системи.

Для кожного з акторів передбачені відповідні сценарії використання:

Незареєстрований користувач може:

- переглядати головну сторінку;
- здійснювати реєстрацію в системі;

- проходити авторизацію (вхід у систему).

Зареєстрований користувач має можливість:

- заповнювати або редагувати анкету;
- переглядати профілі інших користувачів;
- здійснювати пошук за фільтрами;
- призначати зустрічі;
- переглядати історію попередніх взаємодій;
- вести чат із іншими користувачами;
- оплачувати підписку для розширеного доступу до функціоналу.

Адміністратор системи виконує такі дії:

- керує користувачами (створення, редагування, блокування акаунтів);
- переглядає загальну статистику роботи системи;
- створює та експортує звіти для подальшого аналізу ефективності.

Представлена діаграма дозволяє сформувати уявлення про загальну архітектуру функціональної взаємодії користувачів із системою, окреслює межі відповідальності кожного актора та допомагає візуалізувати потоки дій, які підтримує інформаційна система. Такий підхід дозволяє структурувати процес розробки та полегшує подальше моделювання логіки системи в наступних розділах.

Діаграма варіантів використання для бекенду інформаційної системи "Випадкова зустріч" представлена на рисунку 2.1.

2.2 Рішення з інформаційного забезпечення

На основі вимог було виконано проєктування логічної та фізичної структури даних.

Інформаційна модель системи повинна бути узгоджена з архітектурою та технологіями, обраними на етапі загальносистемних рішень. Для реалізації системи передбачається використання реляційної бази даних **MariaDB**, що забезпечує надійне зберігання структурованої інформації та підтримує транзакційність, індексацію, оптимізацію запитів та захист цілісності даних.

Дані про користувачів зберігаються у відповідних таблицях, що охоплюють:

- особисті дані;
- параметри анкет;
- історію спілкування;
- інформацію про призначені зустрічі.

2.2.1 Визначення сутностей інформаційної системи

На основі аналізу предметної області була побудована концептуальна модель даних (ERD)

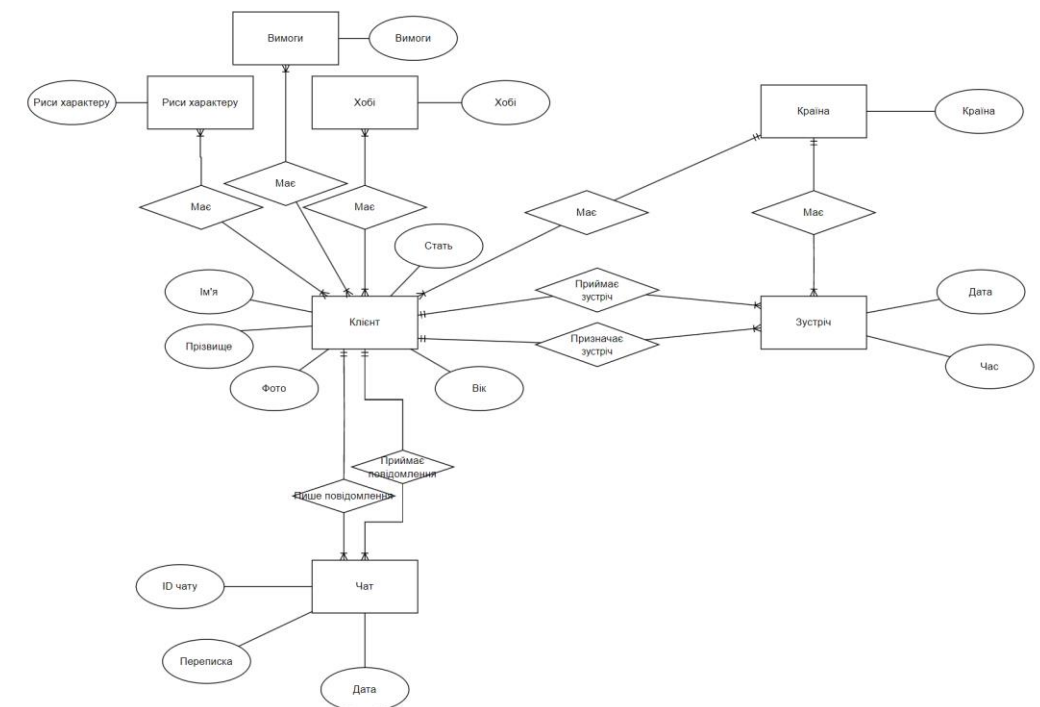


Рис. 2.2 – ER-діаграма сутностей БД

Опис усіх сутностей з усіма атрибутами

Сутність	Атрибути
Клієнт	ID
	Ім'я
	Прізвище
	Фото
	Стать
	Вік
	Дата народження
Зустріч	ID
	Клієнт 1
	Клієнт 2
	Дата
	Час
	Місце
	Країна
Країна	ID країни
	Країна
Риси характеру	ID риси характеру
	Риси характеру
	Клієнт
Хобі	ID хобі
	Клієнт
	Хобі

Вимоги	ID вимоги
	Клієнт
	Вимога
Чат	ID
	Клієнт 1
	Клієнт 2
	Повідомлення
	Час відправлення

На рисунку 4 зображена ER-діаграма сутностей БД у скороченому вигляді – без представлених атрибутів.

2.2.2 Розробка логічної моделі даних

Логічна модель даних включає ідентифікацію сутностей, визначення їхніх атрибутів і зв'язків між ними, а також побудована на базі спроектованої моделі даних. Також логічна модель даних визначає, як ці сутності взаємодіють і зберігаються у базі даних. Розроблена логічна модель предметної області зображена на рисунку 5.

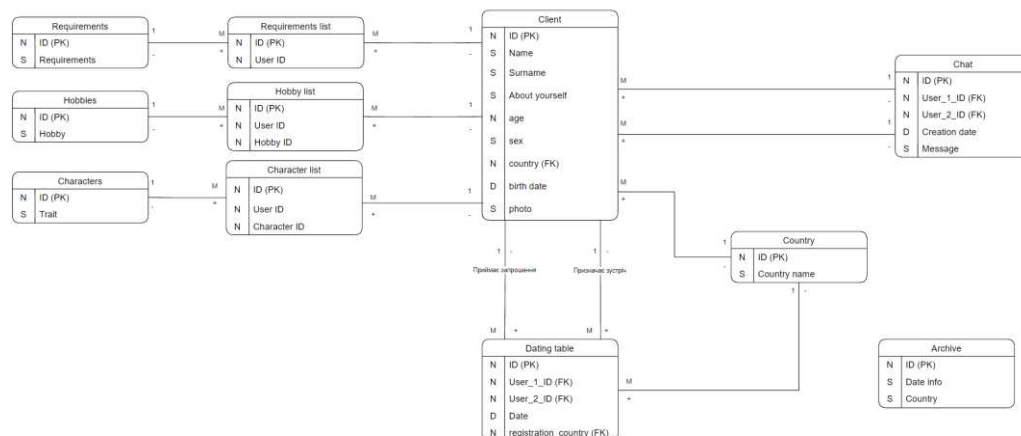


Рис. 2.3 – ER-діаграма логічної моделі

2.2.3 Розробка фізичної моделі даних

Фізична модель даних

Таблиця	Атрибути	Тип даних	Тип ключа	Розмір
client	ID	int	PK	11
	Name	tinytext		
	Surname	tinytext		
	Photo	text		
	Sex	text		
	Age	int		11
	Birthdate	date		
Dating_table	ID	int	PK	11
	User_1_ID	int	FK	11
	User_2_ID	int	FK	11
	Date	datetime		
	Registration_country	int		11
Country	ID	int	PK	11
	Country	text	FK	
characters_list	ID	int	PK	11
	Character_ID	text	FK	
	Client_ID	int	FK	11
Characters	ID		PK	
	Trait			
Requirements_list	ID	int	PK	11
	Requirement_ID	int	FK	11

	Client_ID	int	FK	11
Requirements	ID	int	PK	11
	Requirement	text	FK	
Hobby_list	ID	int	ID	11
	Hobby_ID	text	FK	
	Client_ID	int	FK	11
Hobbies	ID	int	PK	11
	Hobby	text		
Chat	ID	int	PK	11
	Клієнт 1	int	FK	11
	Клієнт 2	int	FK	11
	Повідомлення	text		
	Час відправлення	datetime		

- Таблиця хоббі (hobby_list): має атрибути User ID та Hobby ID, що залежать від посилань на первинні ключі таблиць Client та Hobby.
- Таблиця вимог (requirements_list): має атрибути User ID та Requirement ID, що залежать від посилань на первинні ключі таблиць Client та Requirements.
- Таблиця рис характеру (characters_list): має атрибути User ID та Character ID, що залежать від посилань на первинні ключі таблиць Client та Hobby.
- Таблиця зустрічей (dating_table): має атрибути User 1 ID, User 2 ID та Country, де поля User 1 ID, User 2 ID це посилання на користувачів з таблиці Client, а поле Country містить посилання на однойменну таблицю.
- Таблиця чату (chat): має атрибути User 1 ID, User 2 ID, де поля User 1 ID, User 2 ID це посилання на користувачів з таблиці Client.

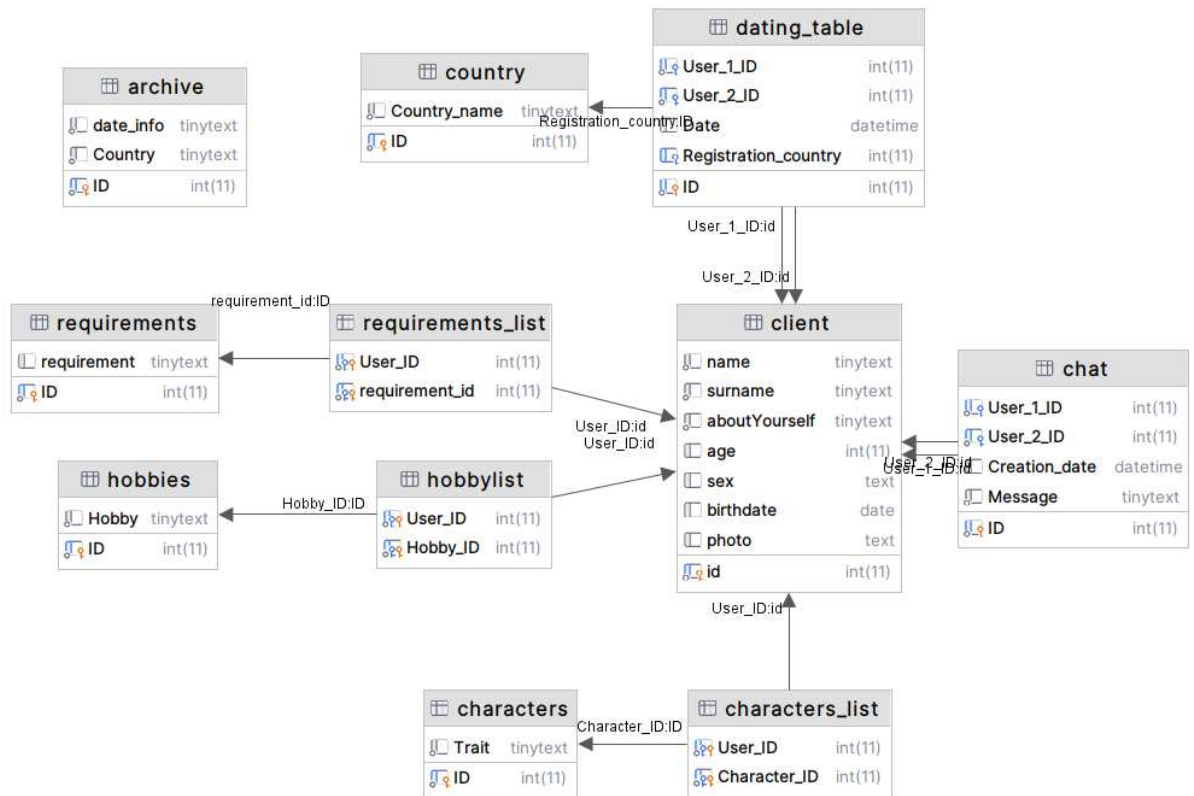


Рис. 2.4 – Фізична модель даних у середовищі розробки

На основі логічної моделі і діаграми сутностей створено статичнцу модель, що представлена діаграмою класів.

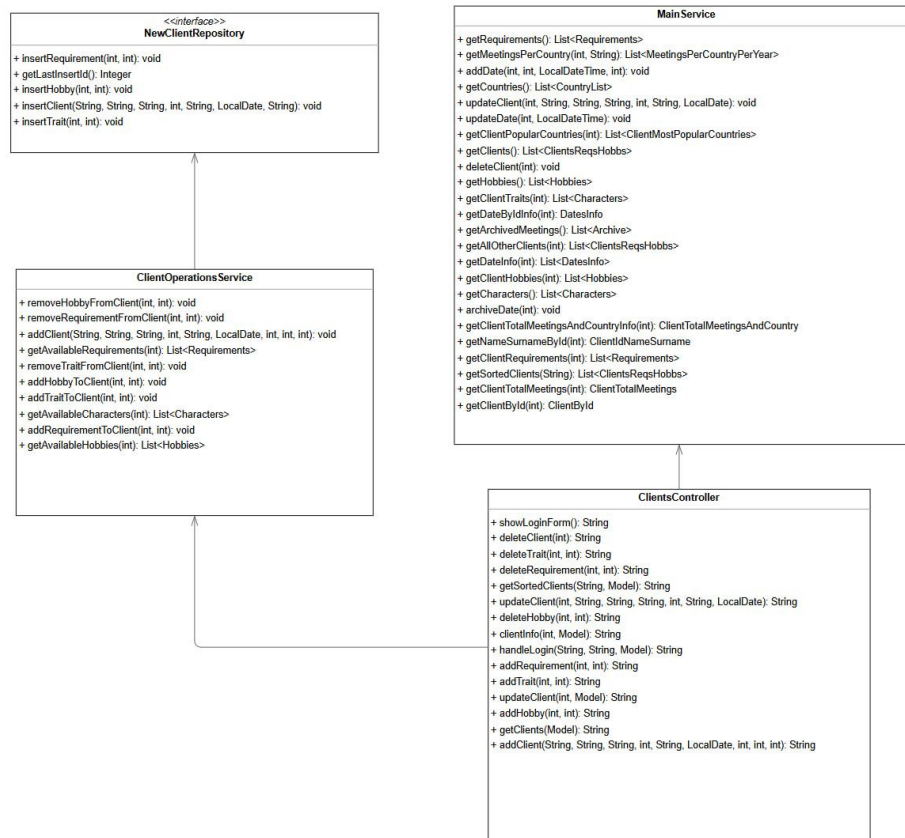


Рис. 2.5 – Діаграма класів застосунку

Для успішної реалізації веб-застосунку було обрано сучасні та зручні інструменти, які забезпечують продуктивність розробки, ефективну інтеграцію компонентів системи та можливість подальшого розширення функціональності.

2.3 Рішення з програмного забезпечення

Проектована інформаційна система агентства знайомств є веб орієнтованим застосунком, реалізованим за клієнт-серверною архітектурою. На основі функціональних, нефункціональних та інформаційних вимог сформовано набір програмних засобів, що забезпечують розробку, запуск і супровід системи.

Мова програмування (Back-end):

Java 17/21 — сучасна, типобезпечна мова, що підтримує об'єктно-орієнтоване програмування та має широку екосистему для створення надійних вебдодатків.

Фреймворки (Back-end):

- Spring Boot — дозволяє швидко створювати автономні, масштабовані вебсервіси з мінімальною конфігурацією.
- Hibernate (JPA) — для об'єктно-реляційного відображення (ORM), що забезпечує зручну роботу з базою даних на основі сутностей.

Фронтенд (клієнтська частина):

- HTML5 / CSS3 / JavaScript — базові вебтехнології для побудови інтерфейсу користувача.
- Bootstrap або Tailwind CSS — для адаптивного та сучасного дизайну.
- Thymeleaf— шаблонізатор або бібліотека для побудови динамічного інтерфейсу.

База даних:

MariaDB — надійна реляційна система управління базами даних із відкритим вихідним кодом, сумісна з MySQL, що забезпечує зберігання структурованих даних про користувачів, профілі, зустрічі тощо.

Інтеграції:

- Google Maps API — для вибору місця зустрічі на карті.
- JavaMail API — для надсилання повідомлень електронною поштою (наприклад, підтвердження реєстрації).
- Spring Security — для реалізації автентифікації, авторизації та контролю доступу.
- Інші технології:
- Lombok — для зменшення шаблонного коду в Java-класах.
- Maven— для керування залежностями та збірки проєкту.

2.4 Рішення з технічного забезпечення

Для реалізації інформаційної системи агентства знайомств обов'язковим є підбір відповідного технічного забезпечення. Його мета — гарантувати безперебійну роботу сервісу, збереження даних, швидкий доступ до функціоналу платформи, а також можливість масштабування при збільшенні навантаження.

Вибір технічних засобів

Для розгортання та експлуатації інформаційної системи необхідно забезпечити відповідне апаратне та мережеве середовище. Орієнтовна конфігурація наведена нижче:

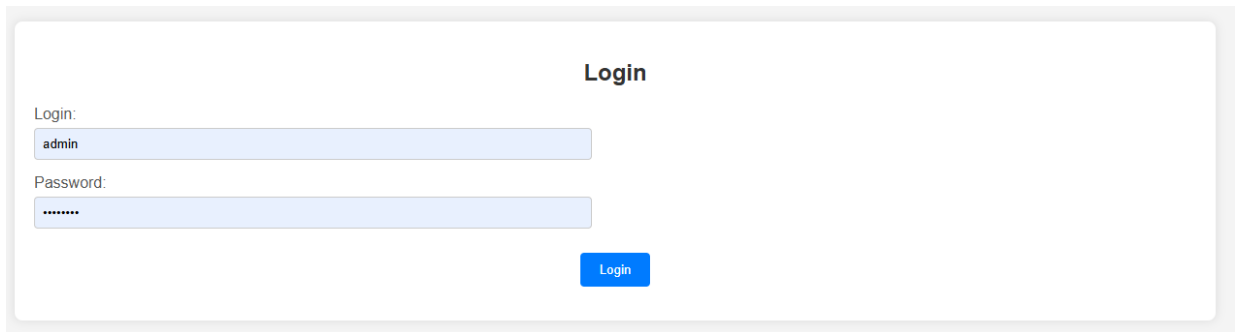
Серверна частина (хостинг або локальний сервер):

- Процесор: мінімум 4 ядра (рекомендовано Intel Xeon або AMD Ryzen).
- Оперативна пам'ять: від 8 ГБ.
- Жорсткий диск: SSD від 100 ГБ, з можливістю резервного копіювання.
- Операційна система: Windows 10.
- Мережева інфраструктура: безперебійне інтернет-з'єднання, статична IP-адреса або домен.

Клієнтська частина:

- Веббраузери: підтримка сучасних браузерів (Google Chrome, Mozilla Firefox, Microsoft Edge).
- Мобільна підтримка: адаптивний інтерфейс для Android/iOS.
- Засоби розробки та супроводу:
- Інтегроване середовище розробки (IDE): IntelliJ IDEA.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОЕКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ



The image shows a web form for logging in. The title 'Login' is centered at the top. Below it, there are two input fields. The first is labeled 'Login:' and contains the text 'admin'. The second is labeled 'Password:' and contains masked characters '*****'. A blue button labeled 'Login' is positioned at the bottom right of the form.

Рис. 3.1 – Сторінка авторизації

На головній сторінці відображено перелік усіх користувачів з їх ознаками.

У кожного користувача є можливість подивитися інформацію щодо зустрічей та більш детальну інформацію про користувача.

Лістинг 3.1 – Запит з виведення усіх клієнтів

```
public interface ClientRepository extends JpaRepository<Client, Integer> {
    @Query(value = ""
SELECT
    C.id,
    C.name,
    C.surname,
    C.aboutYourself,
    TIMESTAMPDIFF(YEAR, C.birthdate, CURDATE()) AS age,
    C.sex,
    C.birthdate,
    GROUP_CONCAT(DISTINCT H.Hobby ORDER BY H.Hobby SEPARATOR ', ') AS hobby,
    GROUP_CONCAT(DISTINCT R.Requirement ORDER BY R.Requirement SEPARATOR ', ')
AS requirement
FROM
    client C
    JOIN hobbylist HL ON HL.User_ID = C.id
    JOIN hobbies H ON HL.Hobby_ID = H.ID
    JOIN requirements_list RL ON RL.User_ID = C.id
    JOIN requirements R ON R.ID = RL.requirement_id
GROUP BY
```

```

        C.id, C.name, C.surname, C.aboutYourself, C.birthdate, C.sex, R.Requirement
ORDER BY
    C.ID
    """, nativeQuery = true)
    List<ClientsReqsHobbs> getClientsWithReqsHobbs();
}

```

Client List										
id	First Name	Last Name	About yourself	age	sex	birthdate	hobby	requirement	client info	dates info
1	John	Johnson	Abou John	20	male	2004-05-01	Cooking	test req	About client	Look dates
2	Alice	Doe	About Alice	20	female	2004-04-29	Gardening	Clever	About client	Look dates
3	John	Doe	About John	30	male	1994-05-07	Gardening	test req	About client	Look dates
4	Jane	Smith	About Jane	24	female	1999-10-15	Reading	test req	About client	Look dates
5	Alice	Johnson	About Alice	40	female	1984-03-22	Cooking	Clever	About client	Look dates
6	John	Smith	About John	30	male	1994-05-07	Coding	test req	About client	Look dates
7	Jane	Smith	About Jane	24	female	1999-10-15	Apple	test req	About client	Look dates
8	Alice	Johnson	About Alice	40	female	1984-03-22	Gardening	Clever	About client	Look dates
9	Michael	Brown	About Michael	34	male	1989-07-18	Reading	test req	About client	Look dates
10	Emily	Davis	About Emily	27	female	1996-12-03	Computer games	test req	About client	Look dates
11	David	Martinez	About David	44	male	1979-09-29	Coding	test req	About client	Look dates

Рис. 3.2 – Початкова сторінка з виведенням усіх клієнтів

Для зручності використання є можливість відсортувати клієнтів за обраним значенням, наприклад за ім'ям.

Лістинг 3.2 – Запит з сортування клієнтів за обраним параметром

```

public interface SortedClientsRepository extends JpaRepository<Client, Integer>{
    @Query(value = ""
    SELECT
    C.id,
    C.name,

```

```

        C.surname,
        C.aboutYourself,
        TIMESTAMPDIFF(YEAR, C.birthdate, CURDATE()) AS age,
        C.sex,
        C.birthdate,
            H.Hobby,
            R.Requirement
    FROM\s
        client C, hobbies H, hobbylist HL, requirements R, requirements_list RL
    WHERE\s
        HL.User_ID = C.id\s
        AND HL.Hobby_ID = H.ID\s
        AND RL.User_ID = C.id\s
        AND R.ID = RL.requirement_id
    ORDER BY
    CASE ?1
        WHEN 'id' THEN CAST(C.id AS SIGNED)
        WHEN 'name' THEN C.name
        WHEN 'surname' THEN C.surname
        WHEN 'aboutYourself' THEN C.aboutYourself
        WHEN 'age' THEN TIMESTAMPDIFF(YEAR, C.birthdate, CURDATE())
        WHEN 'sex' THEN C.sex
        WHEN 'birthdate' THEN C.birthdate
        WHEN 'hobby' THEN H.Hobby

    END
    """, nativeQuery = true)

    List<ClientsReqsHobbs> getClientsWithReqsHobbsSortByName(String sortBy);
}

```

<u>id</u>	<u>First Name</u>	<u>Last Name</u>	<u>About yourself</u>
2	Alice	Doe	About Alice
5	Alice	Johnson	About Alice
8	Alice	Johnson	About Alice
31	David	Martinez	About David
11	David	Martinez	About David
10	Emily	Davis	About Emily
7	Jane	Smith	About Jane
4	Jane	Smith	About Jane
1	John	Johnson	About John
3	John	Doe	About John
6	John	Smith	About John

Рис. 3.3 – Сортвання клієнтів на головній сторінці

А також є форма для додавання нового клієнта

Лістинг 3.2 - Запит з додавання нового клієнта

```
public interface NewClientRepository extends JpaRepository<Client, Integer> {

    @Modifying
    @Query(value = "INSERT INTO client (name, surname, aboutYourself, age, sex,
birthdate, photo) VALUES (?1, ?2, ?3, ?4, ?5, ?6, ?7)", nativeQuery = true)
    void insertClient(String firstName, String lastName, String aboutYourself,
int age, String sex, LocalDate birthdate, String photo);

    @Query(value = "SELECT LAST_INSERT_ID()", nativeQuery = true)
    Integer getLastInsertId();
}
```



```

    @Modifying
    @Query(value = "INSERT INTO hobbylist (User_ID, Hobby_ID) VALUES (?1, ?2)",
nativeQuery = true)
    void insertHobby(int userId, int hobbyId);

    @Modifying
    @Query(value = "INSERT INTO requirements_list (User_ID, requirement_id)
VALUES (?1, ?2)", nativeQuery = true)
    void insertRequirement(int userId, int requirementId);

    @Modifying
    @Query(value = "INSERT INTO characters_list (User_ID, Character_ID) VALUES
(?1, ?2)", nativeQuery = true)
    void insertTrait(int userId, int characterId);
}

```

Рис. 3.4 – Форма додавання нового клієнта

На сторінці перегляду зустрічей виведено список усіх зустрічей з участю клієнта, а також є форма для додавання нової зустрічі з можливістю призначити час, обрати іншого клієнта з існуючих у БД, а також наявних країн. Також є можливість заархівувати зустріч.

Лістинг 3.4 – Запит для виведення зустрічей клієнта

```

public interface ClientDateInfoRepository extends JpaRepository<Client, Integer>
{

```

```

@Query(value = """
SELECT
DT.id,
CONCAT(C1.name, ' ', C1.surname) AS client1,
C1.id AS client1_id,
CONCAT(C2.name, ' ', C2.surname) AS client2,
C2.id AS client2_id,
DT.Date,
CNTR.Country_name AS `Country`
FROM dating_table DT
LEFT JOIN client C1 ON DT.User_1_ID = C1.id
LEFT JOIN client C2 ON DT.User_2_ID = C2.id
LEFT JOIN country CNTR ON CNTR.ID = DT.Registration_country
WHERE DT.User_1_ID = ?1 OR DT.User_2_ID = ?1
""", nativeQuery = true)
List<DatesInfo> getDateInfo(int clientid);

}

```

[Go to Home](#)
[About client](#)

Client Information

Date ID	Made the Appointment	Got the Appointment	Date	Country	Edit	Archive
3	Alice Doe	Jane Smith	2024-05-07 17:00:00.0	Ukraine	Edit date	Archive date
5	Alice Doe	Jane Smith	2024-05-08 10:16:54.0	Poland	Edit date	Archive date
21	Alice Doe	Jane Smith	2024-05-21 17:55:00.0	Ukraine	Edit date	Archive date
27	Alice Doe	William Taylor	2024-05-25 11:03:00.0	USA	Edit date	Archive date
31	Alice Doe	Emily Davis	2024-06-07 06:15:00.0	Poland	Edit date	Archive date
32	Alice Doe	Emily Davis	2024-06-13 12:05:00.0	Hungary	Edit date	Archive date
73	Alice Doe	David Martinez	2024-12-06 00:00:00.0	USA	Edit date	Archive date

Рис. 3.5 – Сторінка перегляду зустрічей

На сторінці заархівованих зустрічей виведено усі заархівовані зустрічі між клієнтами, а також країни в яких вони відбувалися.

Лістинг 3.4 – Запит для виведення заархівованих зустрічей

```
public interface ArchiveRepository extends JpaRepository<Archive, Integer> {  
    @Query(value = ""  
    SELECT  
        A.ID,  
        A.date_info,  
        A.Country  
    FROM archive A  
    """, nativeQuery = true)  
    List<Archive> getArchivedMeetings();  
}
```

Archived Meetings	
Date Info	Country
Test Testovich, Test Testovich, 2024-06-06	Poland
David Martinez, Michael Brown, 2024-07-06	Poland
David Martinez, William Taylor, 2024-06-21	USA
Alice Johnson, David Martinez, 2024-07-10	USA
Alice Johnson, John Smith, 2024-05-21	USA

Рис. 3.6 – Сторінка перегляду архівованих зустрічей

На сторінці подробиць про клієнта виведено усі його дані, а також є можливість їх оновити або видалити клієнта. Крім того, виведено найчастішу країну у якій у клієнта відбувалися зустрічі, а також статистику за усіма країнами з його зустрічей.

Статистичні запити для виведення статистики клієнта надані на лістингах 3.5-3.7

Лістинг 3.5 – Країни зустрічей клієнта

```
public interface ClientPopularCountriesRepository extends JpaRepository<Client,  
Integer> {
```

```

        @Query(value = ""
SELECT
    COUNT(*) AS totalMeetings,
    CNTR.Country_name AS mostCommonCountry
FROM
    dating_table DT
LEFT JOIN country CNTR ON CNTR.ID = DT.Registration_country
WHERE
    DT.User_1_ID = ?1 OR DT.User_2_ID = ?1
GROUP BY
    CNTR.Country_name
ORDER BY
    totalMeetings DESC
"", nativeQuery = true)
    List<ClientMostPopularCountries> getClientPopularCountriesById(int
clientid);
}

```

Лістинг 3.6 – Сума зустрічей клієнта за країнами

```

public interface ClientTotalMeetingsAndCountryByIdRepository extends
JpaRepository<Client, Integer> {
    @Query(value = ""
SELECT
        COUNT(*) AS totalMeetings,
        CNTR.Country_name AS mostCommonCountry
FROM
        dating_table DT
JOIN
        country CNTR ON CNTR.ID = DT.Registration_country
WHERE
        DT.User_1_ID = ?1 OR DT.User_2_ID = ?1

"", nativeQuery = true)
    ClientTotalMeetingsAndCountry getClientInfoById(int clientid);
}

```

Лістинг 3.7 – Сума зустрічей клієнта за країнами за рік

```

public interface MeetingsByCountryPerYear extends JpaRepository<Client, Integer>
{
    @Query(value = ""

```

```

SELECT
    DATE_FORMAT(DT.`Date`, '%Y-%m') AS month,
    COUNT(*) AS totalMeetings,
    CNTR.Country_name AS country
FROM
    dating_table DT
LEFT JOIN
    country CNTR ON CNTR.ID = DT.Registration_country
WHERE
    YEAR(DT.`Date`) = ?1
    AND CNTR.Country_name = ?2
GROUP BY
    DATE_FORMAT(DT.`Date`, '%Y-%m'), CNTR.Country_name
ORDER BY
    month
""", nativeQuery = true)
    List<MeetingsPerCountryPerYear> getMeetingsForCountryPerYear(int year,
String country);
}

```

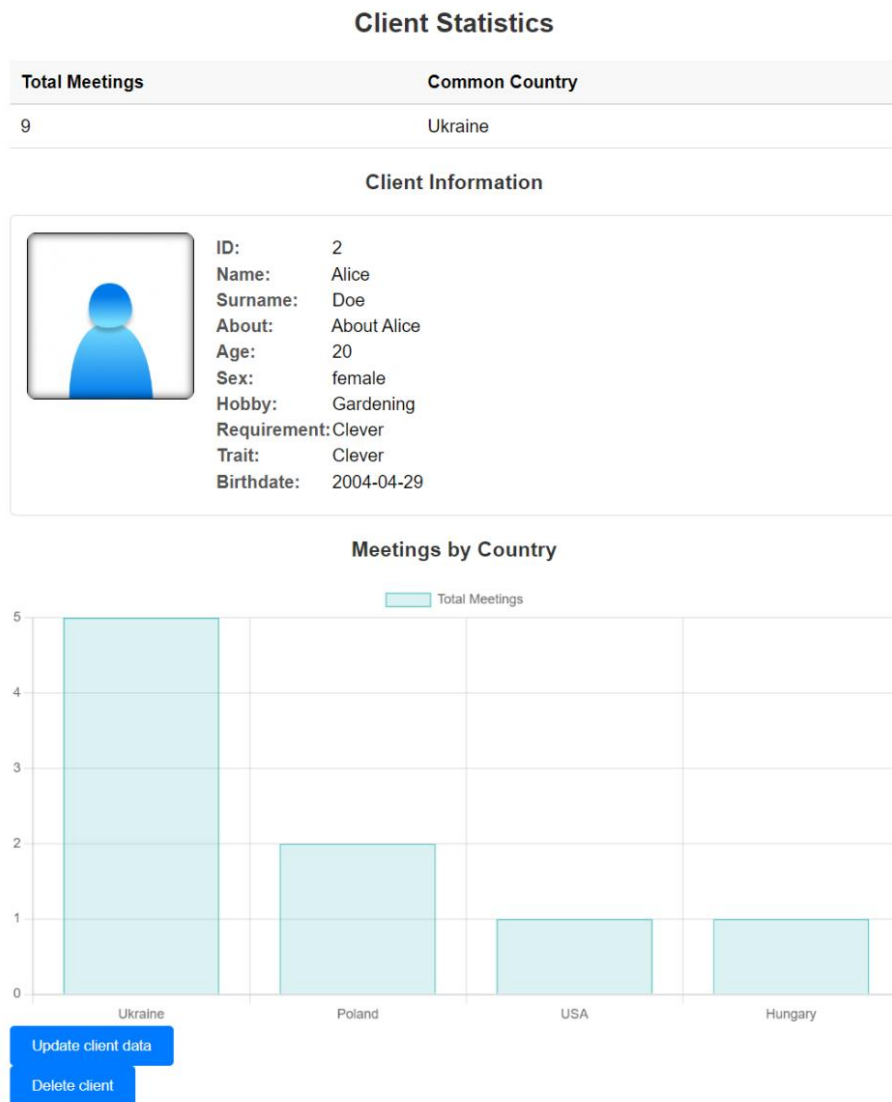


Рис. 3.7 – Сторінка для перегляду загальних та статистичних даних клієнта

На сторінці оновлення даних клієнта виведено його існуючі дані у полях для їх можливої зручної зміни.

Лістинг 3.8 – Запит для оновлення даних клієнта

```
public interface UpdateClientRepository extends JpaRepository<Client, Integer> {  
  
    @Modifying  
    @Transactional  
    @Query(value = "UPDATE client SET name = ?2, surname = ?3, aboutYourself =  
?4, age = ?5, sex = ?6, birthdate = ?7 WHERE id = ?1", nativeQuery = true)  
    void updateClient(int id, String firstName, String lastName, String  
aboutYourself, int age, String sex, LocalDate birthdate);  
}
```

```

    @Modifying
    @Transactional
    @Query(value = "UPDATE hobbylist SET Hobby_ID = ?2 WHERE User_ID = ?1",
nativeQuery = true)
    void updateHobby(int userId, List<Integer> hobbyId);

    @Modifying
    @Transactional
    @Query(value = "UPDATE requirements_list SET requirement_id = ?2 WHERE
User_ID = ?1", nativeQuery = true)
    void updateRequirement(int userId, List<Integer> requirementId);

    @Modifying
    @Transactional
    @Query(value = "UPDATE characters_list SET Character_ID = ?2 WHERE User_ID =
?1", nativeQuery = true)
    void updateTrait(int userId, List<Integer> traitId);
}

```

Update Client

First Name:
 Last Name:
 About Yourself:
 Age:
 Sex:
 Birthdate:

Update Basic Info

Hobbies

Hobby Actions
 Coding
 Add Hobby:

Requirements

Requirement Actions
 Clever
 Add Requirement:

Traits

Trait Actions
 Clever
 Add Trait:

Рис. 3.8 – Сторінка для оновлення даних клієнта

На сторінці чату виведено активні чати та клієнти з якими можна розпочати спілкування. Запити для забезпечення функціоналу чату наведені на лістингах 3.9-3.10

Лістинг 3.9 – Запит для виведення активних чатів клієнта

```
public interface activeChatsRepository extends JpaRepository<Client, Integer> {
    @Query(value = ""
SELECT
    C.id,
    CONCAT(C.name, ' ', C.surname) AS clientNameSurname
FROM client C
WHERE C.id IN (
    SELECT User_1_ID FROM chat WHERE User_2_ID = ?1
    UNION ALL
```



```

        SELECT User_2_ID FROM chat WHERE User_1_ID = ?1
    )
    "", nativeQuery = true)
    List<ClientIdNameSurname> getAllActiveChatsClientNameSurname(int clientid);
}

```

Лістинг 3.10 – Запит для виведення можливих нових чатів

```

public interface clientsForNewChatRepository extends JpaRepository<Client,
Integer> {
    @Query(value = ""
SELECT
    C.id,
    CONCAT(C.name, ' ', C.surname) AS clientNameSurname
FROM client C
WHERE C.id NOT IN (
    SELECT User_1_ID FROM chat WHERE User_2_ID = ?1
    UNION ALL
    SELECT User_2_ID FROM chat WHERE User_1_ID = ?1
) AND C.id != ?1
    "", nativeQuery = true)
    List<ClientIdNameSurname> getAllClientNameSurname(int clientid);
}

```

Active chats		
Id	Client	#
1	John Johnson	Show chat
7	Jane Smith	Show chat
9	Michael Brown	Show chat

Available clients		
Id	Client	#
3	John Doe	Create new chat
4	Jane Smith	Create new chat
5	Alice Johnson	Create new chat
6	John Smith	Create new chat

Рис. 3.9 – Сторінка перегляду чатів клієнта

Сторінка активного чату дає можливість надсилати повідомлення іншому користувачу та видаляти свої повідомлення за бажання.

Лістинг 3.11 – Запит для виведення повідомлень між певними клієнтами

```

public interface existingChat extends JpaRepository<Client, Integer> {
    @Query(value = ""

```

```

SELECT
    chat.ID,
    chat.User_1_ID,
    CONCAT(client1.name, ' ', client1.surname) AS client1NameSurname,
    chat.User_2_ID,
    CONCAT(client2.name, ' ', client2.surname) AS client2NameSurname,
    chat.Creation_date,
    chat.Message
FROM chat
JOIN client client1 ON chat.User_1_ID = client1.id
JOIN client client2 ON chat.User_2_ID = client2.id
WHERE (chat.User_1_ID = ?1 AND chat.User_2_ID = ?2) OR (chat.User_1_ID = ?2 AND
chat.User_2_ID = ?1)
ORDER BY chat.Creation_date
"", nativeQuery = true)
    List<ExistingChat> getExistingChat(int firstClientId, int secondClientId);

}

```

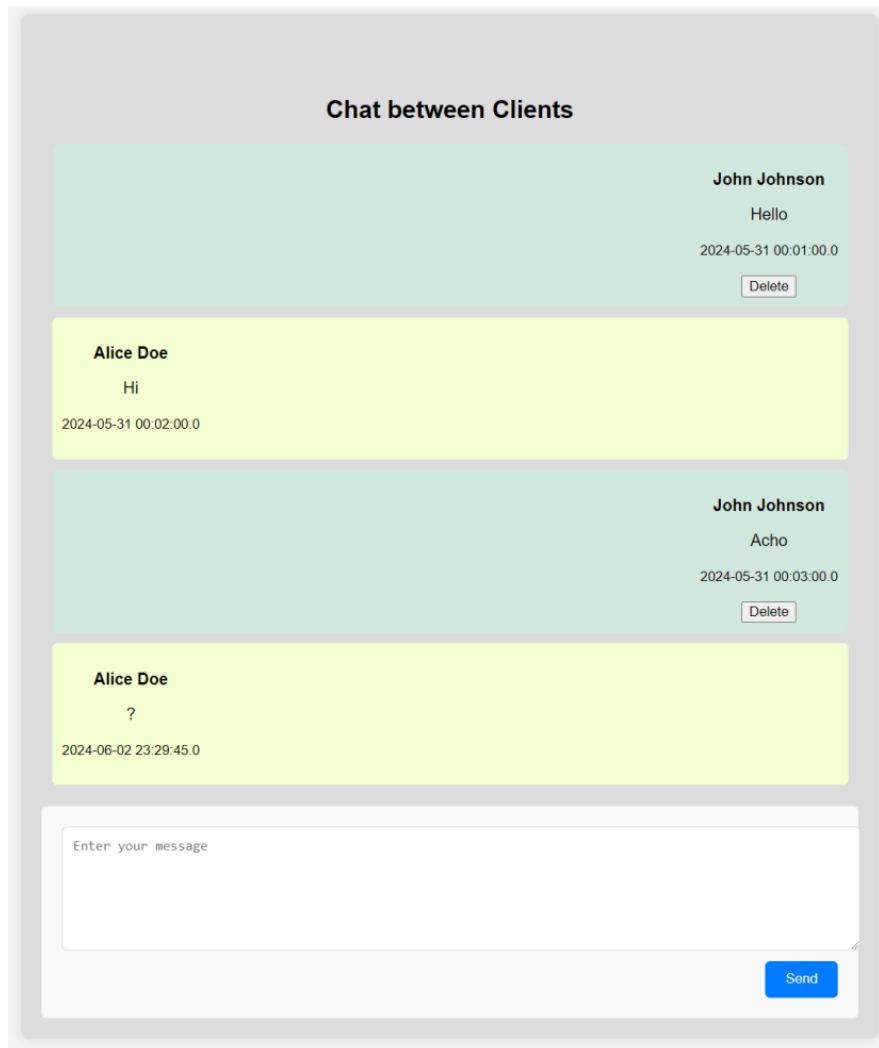


Рис. 3.10 – Активний чат між користувачами

Сторінка зі статистикою дозволяє переглянути кількість активних зустрічей кожного місяця у окремій активній вибраній країні за введений рік

Лістинг 3.12 – Сума зустрічей клієнта за країнами за рік

```
public interface MeetingsByCountryPerYear extends JpaRepository<Client, Integer>
{
    @Query(value = ""
SELECT
    DATE_FORMAT(DT.`Date`, '%Y-%m') AS month,
    COUNT(*) AS totalMeetings,
    CNTR.Country_name AS country
FROM
    dating_table DT
```

```

LEFT JOIN
    country CNTR ON CNTR.ID = DT.Registration_country
WHERE
    YEAR(DT.`Date`) = ?1
    AND CNTR.Country_name = ?2
GROUP BY
    DATE_FORMAT(DT.`Date`, '%Y-%m'), CNTR.Country_name
ORDER BY
    month
""", nativeQuery = true)
    List<MeetingsPerCountryPerYear> getMeetingsForCountryPerYear(int year,
String country);
}

```

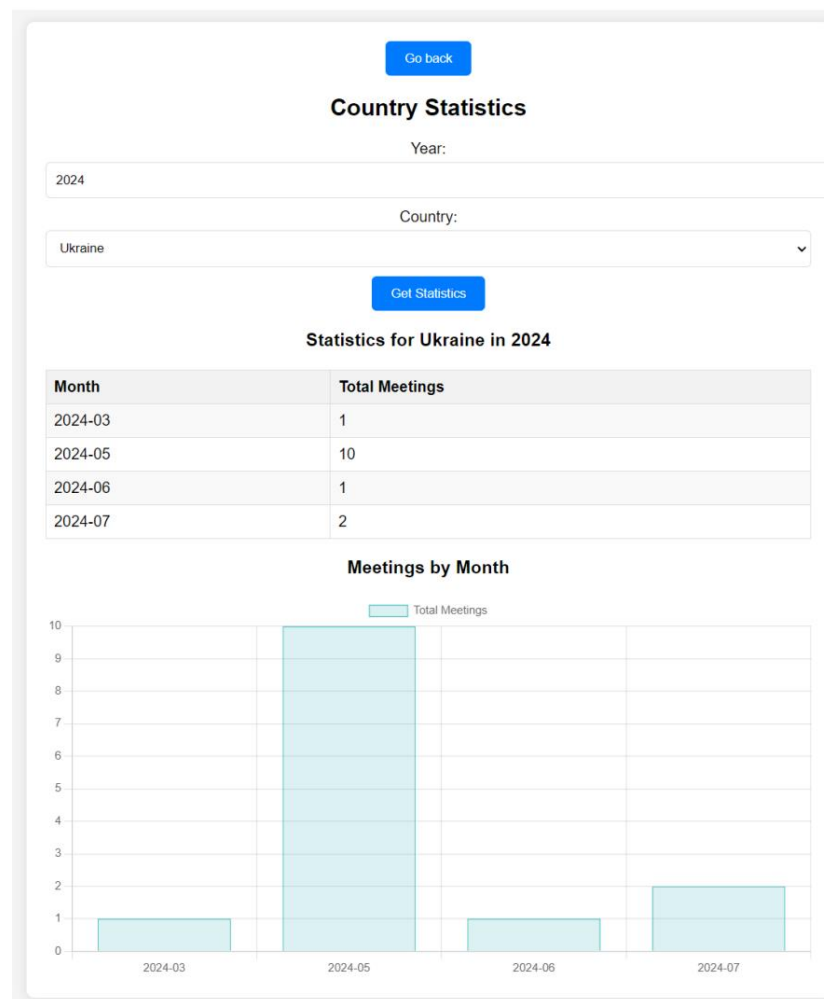


Рис. 3.11 – Сторінка перегляду статистичних даних

Першою сторінкою яку бачить користувач є `index.html`. Коли користувач взаємодіє з об'єктами на цій сторінці, викликаються методи у класі `Controller` які відповідають функціональності веб-інтерфейсу. Взаємодія методів з класу `Controller` з базою даних предметної області відбувається за допомогою методів-сервісів, які в свою чергу викликають інтерфейси для взаємодії застосунку з базою даних.

Як приклад, розглянемо клас `getAllOtherClients()` викликає інтерфейс `clientsWithoutId.getClientsWithoutId(id)`; передає в нього отриману змінну `id`. Викликаний інтерфейс передає SQL-запит до БД та повертає оброблені за допомогою інтерфейсу `DTO (Data Transfer Object)` дані з БД до сервісу, який передає їх назад до контролеру, а той в свою чергу використовує їх для побудову контенту сторінки.

Загальний опис методів у класі `Service`

- `getClients()`: Цей метод отримує список всіх клієнтів разом з їхніми вимогами та хобі.
- `getAllOtherClients(int id)`: Цей метод повертає список всіх клієнтів, окрім того, що вказано в параметрі `id`.
- `getHobbies()`: Цей метод повертає список усіх доступних хобі.
- `getRequirements()`: Цей метод повертає список усіх доступних вимог.
- `getCharacters()`: Цей метод повертає список усіх доступних рис характеру.
- `getClientById(int id)`: Цей метод повертає конкретного клієнта на основі його ідентифікатора.
- `getNameSurnameById(int id)`: Цей метод отримує ім'я та прізвище клієнта на основі його ідентифікатора.
- `getDateInfo(int id)`: Цей метод повертає список дат, пов'язаних з певним клієнтом (ймовірні дати зустрічей).

- `getClientTotalMeetingsAndCountryInfo(int id)`: Цей метод повертає інформацію про загальну кількість зустрічей клієнта та його країну.
- `getClientPopularCountries(int id)`: Цей метод повертає список найпопулярніших країн, які відвідував певний клієнт.
- `getMeetingsPerCountry(int id, String country)`: Цей метод повертає список зустрічей, які клієнти провели у певній країні за обраний рік.
- `getCountries()`: Цей метод повертає список усіх країн.
- `addDate(int firstid, int secondid, дата LocalDateTime, int country)`: Цей метод додає запис про дату (ймовірно, зустріч) між двома клієнтами в певній країні.
- `addClient(String firstName, String lastName, String aboutYourself, int age, String sex, LocalDate дата народження, int hobbyId, int requirementId, int traitId)`: Цей метод створює новий запис клієнта з його даними, хобі, вимогами та рисами.
- `getSortedClients(String sortBy)`: Цей метод повертає відсортований список клієнтів з їхніми вимогами та хобі на основі наданих критеріїв сортування.
- `getArchivedMeetings()`: Цей метод повертає список всіх заархівованих зустрічей.
- `updateClient(int id, String firstName, String lastName, String aboutYourself, int age, String sex, LocalDate дата народження, int hobby, int requirement, int trait)`: Цей метод оновлює дані, хобі, вимоги та риси конкретного клієнта.
- `archiveDate(int dateId)`: Цей метод переміщує запис дати (ймовірно, зустрічі) до архіву і видаляє його з основної таблиці.
- `deleteClient(int id)`: Цей метод видаляє запис клієнта разом з пов'язаними з ним хобі, вимогами та рисами.

- getClientTotalMeetings(int id): Цей метод повертає загальну кількість зустрічей, які мав клієнт.

3.1 Розробка інструкції користувача

Інструкція користувача є важливим елементом впровадження інформаційної системи, оскільки забезпечує ефективну взаємодію між користувачем і системою. Інструкція представлена у Додатку Б.

-

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

Охорона праці є важливою складовою організації безпечних умов виконання професійної діяльності працівників в ІТ-сфері. Метою системи охорони праці є збереження життя, здоров'я, працездатності персоналу та створення належних санітарно-гігієнічних умов праці. Розробка інформаційної системи агентства знайомств здійснюється в умовах офісного середовища, що передбачає використання комп'ютерної техніки, офісного обладнання, мережевої інфраструктури та відповідного програмного забезпечення.

Діяльність програміста або інженера-програміста, що виконує такі завдання, підпадає під класифікацію робіт з нормальними умовами праці. Проте при тривалій роботі за комп'ютером можливе виникнення психоемоційного та фізичного навантаження, тому до організації робочого процесу мають бути висунуті певні вимоги.

Робоче місце розробника інформаційної системи повинне відповідати вимогам нормативних документів, зокрема ДСанПіН 3.3.2.007-98 «Гігієнічні вимоги до відеодисплейних терміналів (ВДТ) і персональних електронно-обчислювальних машин». Згідно з ними:

Освітлення має бути комбінованим: природним і штучним. Рекомендована яскравість — 300–500 лк.

Мікроклімат у приміщенні підтримується на рівні температури 20–24 °С, відносної вологості 40–60%.

Акустичний комфорт забезпечується рівнем шуму, що не перевищує 50 дБА.

Ергономіка робочого місця передбачає:

використання стільця з регулюванням висоти, нахилу спинки та наявністю підлокітників;

правильне розташування монітора на відстані 50–70 см від очей на рівні або трохи нижче рівня очей користувача;

використання підставки для ніг (за необхідності).

Робочі місця обладнані персональними комп'ютерами, мережевими комутаторами, джерелами безперебійного живлення та іншими пристроями, що працюють від електромережі. Тому велике значення має забезпечення **електробезпеки**, згідно з вимогами ПУЕ (Правила улаштування електроустановок) та ПТЕЕС (Правила технічної експлуатації електроустановок споживачів).

Категорія електробезпеки — **II група**. Працівники повинні пройти інструктаж з електробезпеки, а також мати доступ до захисних засобів (діелектричні килимки, справні розетки, захисне заземлення корпусів пристроїв).

Офісне приміщення має бути обладнане первинними засобами пожежогасіння: порошковими або вуглекислотними вогнегасниками, пожежною сигналізацією, евакуаційними схемами. Основними причинами пожеж в ІТ-офісах є коротке замикання, перевантаження електромережі, використання несправного обладнання. Пожежна безпека забезпечується шляхом:

- регулярного технічного огляду електроприладів;
- встановлення автоматичних вимикачів;
- наявності інструкцій з евакуації;
- щорічного проходження працівниками інструктажів з пожежної безпеки.

Згідно з нормами охорони праці в галузі ІТ, робота за комп'ютером не повинна перевищувати:

- 4 години поспіль без перерви;
- 6 годин на день загалом — для жінок;
- 8 годин — для осіб, що займаються розробкою ПЗ.

Рекомендується проводити перерви кожні 50 хвилин тривалістю 10–15 хвилин, під час яких варто робити вправи для очей, розминку для зняття м'язової напруги та змінювати положення тіла.

У роботі розробників важливе значення має психоемоційна стабільність. Постійна робота з інформацією, жорсткі строки виконання завдань та технічна відповідальність можуть призводити до перенавантаження та стресу. Тому в офісі повинна бути створена сприятлива атмосфера: зони відпочинку, доступ до питної води, гнучкий графік роботи, а також можливість віддаленої роботи за необхідності.

Таким чином, організація безпечних умов праці під час розробки інформаційної системи агентства знайомств передбачає дотримання комплексу заходів з охорони праці, електробезпеки, пожежної безпеки, забезпечення комфортного мікроклімату та ергономіки робочого місця. Виконання зазначених вимог сприяє не лише збереженню здоров'я працівників, а й підвищенню їх продуктивності та якості результату.

ВИСНОВКИ

У результаті розробки проєкту інформаційної системи агентства знайомств було здійснено детальний аналіз предметної області, виявлено актуальні проблеми сучасних онлайн-сервісів знайомств, такі як недостатня безпека, відсутність персоналізованого підбору, низький рівень верифікації користувачів та відсутність механізмів планування реальних зустрічей. На основі виявлених недоліків було сформовано вимоги до нової інформаційної системи, яка враховує як функціональні, так і нефункціональні аспекти — безпеку, масштабованість, зручність використання та адаптивність інтерфейсу.

Розроблена концепція інформаційної системи орієнтована на автоматизацію процесів взаємодії між користувачами: реєстрації, заповнення анкет, планування зустрічей, ведення статистики й адміністрування системи. В результаті проєктування були створені моделі даних, описано архітектуру програмного продукту, розроблено програмні та технічні рішення.

На основі створеної концепції реалізовано технічний проєкт системи, який включає модулі користувацької взаємодії, адміністрування, аналітики, а також чат і модуль планування зустрічей із прив'язкою до геолокації. Програмна реалізація базується на сучасних технологіях — Java, Spring Boot, MariaDB, HTML/CSS — що забезпечує гнучкість, розширюваність і високий рівень захисту даних.

Таким чином, мету дипломної роботи досягнуто. Створена інформаційна система може бути впроваджена в практичну діяльність агентства знайомств, що суттєво підвищить ефективність організації знайомств, покращить користувацький досвід і сприятиме формуванню безпечного цифрового середовища для міжособистісної взаємодії.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Київський міжнародний інститут соціології (КМІС). Онлайн-знайомства в Україні: тенденції та поведінка користувачів. – 2023.
2. Statista. Share of adults in the United Kingdom who have encountered fake profiles on dating apps as of December 2023, by gender [Електронний ресурс] // Statista. – Режим доступу: <https://www.statista.com/statistics/1466464/britain-dating-app-encounter-fake-profiles-by-gender/> –.
3. Пронь О. М. Інтернет-платформи соціальної взаємодії: переваги та недоліки // Вісник НТУУ «КПІ». Серія: Соціологія. — 2022. — №1..
4. eHarmony. User feedback and technical documentation. — Internal Review, 2020.
5. SoulMatcher. Зростаючі тенденції в онлайн-знайомствах: Висновки та статистика [Електронний ресурс]. — Режим доступу: <https://soulmatcher.app/uk/blog/growing-trends-in-online-dating-insights-and-statistics/>
6. Марченко А. В. Проектування інформаційних систем. — Суми: СумДУ, 2016. — 90 с. — Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/content-20160217112601.pdf
7. Грекул В. І. Проектування інформаційних систем. Практикум. — М.: Національний відкритий університет «ІНТУІТ», 2012.
8. Ременяк Л. В. Проектування інформаційних систем: конспект лекцій. — Одеса: Одеський державний екологічний університет, 2016.

9. Фаулер М. UML. Основи. — 3-тє вид. — Пер. з англ. А. Петухова. — СПб.: Символ-Плюс, 2005
10. Проектування програмного забезпечення засобами UML. 3.1
Діаграма варіантів використання — КПІ —Режим доступу:
http://mmsa.kpi.ua/sites/default/files/disciplines/%D0%A0%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0%20%D1%96%20%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC/didkovska_m_v_testing_lecture_3.pdf

ДОДАТОК А

Код програмних модулів системи:

Лістинг 1 – ClientsController.java

```
@org.springframework.stereotype.Controller
@AllArgsConstructor
public class ClientsController {

    private MainService mainService;
    private ClientOperationsService clientOperations;

    @GetMapping("/")
    public String getClients(Model model) {
        model.addAttribute("clients", mainService.getClients());
        model.addAttribute("hobbies", mainService.getHobbies());
        model.addAttribute("requirements", mainService.getRequirements());
        model.addAttribute("characters", mainService.getCharacters());
        return "index";
    }

    @GetMapping("/login")
    public String showLoginForm() {
        return "login";
    }

    @PostMapping("/login")
    public String handleLogin(
        @RequestParam String username,
        @RequestParam String password,
        Model model
    ) {
        if ("admin".equals(username) && "password".equals(password)) {
            model.addAttribute("message", "Успешный вход!");
            return "index";
        } else {
            model.addAttribute("error", "Неверный логин или пароль");
            return "login";
        }
    }
}
```

```

    }
}

@GetMapping("/client_info")
public String clientInfo(@RequestParam int id, Model model) {
    model.addAttribute("clientStatistics",
mainService.getClientTotalMeetingsAndCountryInfo(id));
    model.addAttribute("clientInfo", mainService.getClientById(id));
    model.addAttribute("clientPopularCountries",
mainService.getClientPopularCountries(id));

    List<ClientMostPopularCountries> popularCountries =
mainService.getClientPopularCountries(id);
    int totalMeetings =
mainService.getClientTotalMeetings(id).getTotalMeetings();

    ObjectMapper objectMapper = new ObjectMapper();
    try {
        String countriesJson = objectMapper.writeValueAsString(
popularCountries.stream().map(ClientMostPopularCountries::getMostCommonCountry).
collect(Collectors.toList())
        );
        String meetingsJson = objectMapper.writeValueAsString(
popularCountries.stream().map(ClientMostPopularCountries::getTotalMeetings).coll
ect(Collectors.toList())
        );
        model.addAttribute("countriesJson", countriesJson);
        model.addAttribute("meetingsJson", meetingsJson);
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }

    model.addAttribute("totalMeetings", totalMeetings);

    return "client_info";
}

```

```

@GetMapping("/update_client")
public String updateClient(@RequestParam int id, Model model) {
    model.addAttribute("clientInfo", mainService.getClientById(id));
    model.addAttribute("clientHobbies", mainService.getClientHobbies(id));
    model.addAttribute("clientRequirements",
mainService.getClientRequirements(id));
    model.addAttribute("clientTraits", mainService.getClientTraits(id));
    model.addAttribute("clientAvailableHobbies",
clientOperations.getAvailableHobbies(id));
    model.addAttribute("clientAvailableRequirements",
clientOperations.getAvailableRequirements(id));
    model.addAttribute("clientAvailableTraits",
clientOperations.getAvailableCharacters(id));
    model.addAttribute("hobbies", mainService.getHobbies());
    model.addAttribute("requirements", mainService.getRequirements());
    model.addAttribute("characters", mainService.getCharacters());
    return "update_client";
}

@PostMapping("/add_hobby")
public String addHobby(@RequestParam int clientId, @RequestParam int
hobbyId) {
    clientOperations.addHobbyToClient(clientId, hobbyId);
    return "redirect:/update_client?id=" + clientId;
}

@PostMapping("/delete_hobby")
public String deleteHobby(@RequestParam int clientId, @RequestParam int
hobbyId) {
    clientOperations.removeHobbyFromClient(clientId, hobbyId);
    return "redirect:/update_client?id=" + clientId;
}

@PostMapping("/add_requirement")
public String addRequirement(@RequestParam int clientId, @RequestParam int

```



```

requirementId) {
    clientOperations.addRequirementToClient(clientId, requirementId);
    return "redirect:/update_client?id=" + clientId;
}

@PostMapping("/delete_requirement")
public String deleteRequirement(@RequestParam int clientId, @RequestParam
int requirementId) {
    clientOperations.removeRequirementFromClient(clientId, requirementId);
    return "redirect:/update_client?id=" + clientId;
}

@PostMapping("/add_trait")
public String addTrait(@RequestParam int clientId, @RequestParam int
traitId) {
    clientOperations.addTraitToClient(clientId, traitId);
    return "redirect:/update_client?id=" + clientId;
}

@PostMapping("/delete_trait")
public String deleteTrait(@RequestParam int clientId, @RequestParam int
traitId) {
    clientOperations.removeTraitFromClient(clientId, traitId);
    return "redirect:/update_client?id=" + clientId;
}

@PostMapping("/add_client")
public String addClient(@RequestParam String firstName,
                        @RequestParam String lastName,
                        @RequestParam String aboutYourself,
                        @RequestParam int age,
                        @RequestParam String sex,
                        @RequestParam @DateTimeFormat(pattern = "yyyy-MM-
dd") LocalDate birthdate,
                        @RequestParam int hobby,
                        @RequestParam int requirement,

```

```

        @RequestParam int trait) {
            clientOperations.addClient(firstName, lastName, aboutYourself, age, sex,
birthdate, hobby, requirement, trait);
            return "redirect:/";
        }

@PostMapping("/update_client")
public String updateClient(@RequestParam int id,
                           @RequestParam String firstName,
                           @RequestParam String lastName,
                           @RequestParam String aboutYourself,
                           @RequestParam int age,
                           @RequestParam String sex,
                           @RequestParam @DateTimeFormat(pattern = "yyyy-MM-
dd") LocalDate birthdate) {
    mainService.updateClient(id, firstName, lastName, aboutYourself, age,
sex, birthdate);
    return "redirect:/client_info?id=" + id;
}

@PostMapping("/delete_client")
public String deleteClient(@RequestParam int id) {
    mainService.deleteClient(id);
    return "redirect:/";
}

@GetMapping("/sortedClients")
public String getSortedClients(@RequestParam(required = false) String sort,
Model model) {
    if (sort != null) {
        model.addAttribute("clients", mainService.getSortedClients(sort));
        model.addAttribute("hobbies", mainService.getHobbies());
        model.addAttribute("requirements", mainService.getRequirements());
        model.addAttribute("characters", mainService.getCharacters());
    }
    if (Objects.equals(sort, "id")){

```

```

        return getClients(model);
    }
    return "index";
}

}

```

Лістинг 2 – ClientOperationsService.java

```

@Service
@AllArgsConstructor
public class ClientOperationsService {
    private final NewClientRepository newClientRepository;
    private final AddHobbyRepository addHobbyRepository;
    private final DeleteHobbyRepository deleteHobbyRepository;
    private final AddRequirementRepository addRequirementRepository;
    private final DeleteRequirementRepository deleteRequirementRepository;
    private final AddTraitRepository addTraitRepository;
    private final DeleteTraitRepository deleteTraitRepository;
    private final AvailableHobbiesRepository availableHobbiesRepository;
    private final AvailableTraitsRepository availableTraitsRepository;
    private final AvailableRequirementsRepository
availableRequirementsRepository;

    @Transactional
    public void addClient(String firstName, String lastName, String
aboutYourself, int age, String sex, LocalDate birthdate, int hobbyId, int
requirementId, int traitId) {
        newClientRepository.insertClient(firstName, lastName, aboutYourself,
age, sex, birthdate, "images/client_photo.jpg");
        Integer clientId = newClientRepository.getLastInsertId();
        newClientRepository.insertHobby(clientId, hobbyId);
        newClientRepository.insertRequirement(clientId, requirementId);
        newClientRepository.insertTrait(clientId, traitId);
    }

    @Transactional
    public void addHobbyToClient(int clientId, int hobbyId) {
        addHobbyRepository.insertHobby(clientId, hobbyId);
    }
}

```

```

@Transactional
public void removeHobbyFromClient(int clientId, int hobbyId) {
    deleteHobbyRepository.deleteHobby(clientId, hobbyId);
}

@Transactional
public void addRequirementToClient(int clientId, int requirementId) {
    addRequirementRepository.insertRequirement(clientId, requirementId);
}

@Transactional
public void removeRequirementFromClient(int clientId, int requirementId) {
    deleteRequirementRepository.deleteRequirement(clientId, requirementId);
}

@Transactional
public void addTraitToClient(int clientId, int traitId) {
    addTraitRepository.insertTrait(clientId, traitId);
}

@Transactional
public void removeTraitFromClient(int clientId, int traitId) {
    deleteTraitRepository.deleteTrait(clientId, traitId);
}

public List<Hobbies> getAvailableHobbies(int clientId) {
    return
availableHobbiesRepository.findAvailableHobbiesByClientId(clientId);
}

public List<Requirements> getAvailableRequirements(int clientId) {
    return
availableRequirementsRepository.findAvailableRequirementsByClientId(clientId);
}

public List<Characters> getAvailableCharacters(int clientId) {
    return

```

```
availableTraitsRepository.findAvailableTraitsByClientId(clientId);  
    }  
}
```

ДОДАТОК Б

Інструкція користувача

Інструкція складається з кількох основних частин:

1. Вимоги до середовища:

- наявність стабільного інтернет-з'єднання;
- підтримка сучасного веббраузера (Google Chrome, Mozilla Firefox, Microsoft Edge);
- для мобільної версії — пристрої на базі Android або iOS.

2. Вхід у систему:

- незареєстровані користувачі мають доступ до головної сторінки з описом сервісу;
- для повного доступу необхідно пройти реєстрацію через форму із підтвердженням e-mail;
- після реєстрації здійснюється авторизація в особистому кабінеті.

3. Створення та редагування профілю:

- користувач заповнює анкету з особистими даними (ім'я, вік, стать, фото, інтереси тощо);
- додатково проходить психологічне тестування для оцінки сумісності;
- усі дані можна редагувати в будь-який час.

4. Пошук та взаємодія:

- пошук профілів здійснюється за допомогою фільтрів (вік, місто, інтереси тощо);
- для взаємодії доступні функції надсилання запитів на зустріч, відкриття чату після підтвердження обома сторонами.

5. Призначення зустрічей:

- користувач може запропонувати зустріч, вибравши час і місце; обидві сторони отримують сповіщення з нагадуванням.