МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ШЕВЧУК ЮРІЙ АНАТОЛІЙОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д. т. н., доцент
_____ Т. В. Нескородєва
«_____» _____ 2022р.

РОЗРОБКА І ДОСЛІДЖЕННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ СФЕРИ НАДАННЯ ПОСЛУГ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Бабаков Р. М., доцент кафедри
інформаціних технологій,
д.т.н., доцент
_____
(підпис)

Оцінка: _____ / _____ /_____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця 2022

АНОТАЦІЯ

Шевчук Ю.А. Розробка додатку по наданню послуг та рекомендаційної системи для нього. Спеціальність 122 “Комп’ютерні науки”, Освітня програма “Комп’ютерні технології обробки даних”. Донецький національний університет імені Василя Стуса, Вінниця, 2022.

Кваліфікаційна робота містить 91 с. тексту, 40 рисунків, 6 таблиць, посилання на 8 літературних джерел, та 1 додаток.

У кваліфікаційній роботі: досліджено існуючі рішення у галузі рекомендаційних систем. Розглянуто загальний підхід до розробки рекомендаційної системи та архітектуру рекомендаційної системи. Після чого розроблено рекомендаційну систему по наданню послуг.

Ключові слова: КОРИСТУВАЧ, ПРОГРАМА, .ПОСЛУГА, РЕКОМЕНДАЦІЙНА СИСТЕМА, ЗАПИТИ, ДІАГРАМА.

ANNOTATION

Shevchuk Y.A. Development of an application for the provision of services and a recommendation system for it. Specialty 122 "Computer science", Educational program "Computer data processing technologies". Vasyl Stus Donetsk National University, Vinnytsia, 2022.

The qualification work contains 91 pages. text, 40 figures, 6 tables, references to 8 literary sources, and 1 appendix.

In the qualification work: existing solutions in the field of recommender systems are investigated. The general approach to the development of the recommender system and the architecture of the recommender system are considered. After that, a recommendation system for the provision of services was developed.

Keywords: USER, PROGRAM, .SERVICE, RECOMMENDATION SYSTEM, REQUESTS, DIAGRAM.

Зміст

ВСТУП

Останнім часом стає все більш популярним створення систем, які здатні вгадувати переваги та потреби користувачів та на основі цього пропонувати відповідні рішення. Такі системи, які називаються рекомендаційними, мають безліч застосувань. Великі компанії, такі як Amazon, Apple, eBay, Pandora і т.д., використовують рекомендаційні системи у складі своїх сервісів. Що важливо, рекомендації формуються для конкретного користувача, виходячи з його особистих даних та інтересів. Такий підхід спрощує користувачеві роботу з великим обсягом даних, допомагає скоротити час пошуку потрібного рішення, забезпечуючи релевантність інформації. Проте найчастіше рекомендаційні системи розробляються на вирішення завдань конкретної області. Наприклад, Amazon спеціалізується на продуктах, що продаються ним, Netflix на фільмах тощо.

Великі компанії мають необхідні кошти та кваліфікованих фахівців для того, щоб розробити власну рекомендаційну систему. Компанії ж, які тільки починають розвиватися і мають необхідність створення сервісу рекомендацій, часто не мають необхідних ресурсів і часу для цього. Розробка універсальної рекомендаційної системи, що адаптується до різних областей, може вирішити цю проблему. Можна відзначити, що при розробці рекомендаційних систем застосовуються загальноприйняті походи, що ґрунтуються на реалізації таких методів машинного навчання як колаборативна фільтрація та фільтрація на основі змісту та ін., що дозволяє поставити завдання реалізації універсального рішення, що базується на використанні цих методів. Налаштування на конкретні предметні області може бути виконане на основі моделей, що створюються з використанням даних, що надаються компаніями-клієнтами (користувачами).

Сервіс рекомендацій, що налаштовується на декілька областей застосування, повинен мати в основі не тільки модель рекомендаційної системи, її предметної області та засоби збору даних про користувачів

системи, а й включати додаткові модулі її підтримки, зокрема засоби аналізу даних. Дані, що використовуються для вироблення рекомендацій, повинні відповідати певним вимогам: актуальність, правильність, унікальність, повнота, структурованість тощо. Тільки за наявності якісних даних можливе прийняття рішень, які найбільш повно відповідають потребам користувачів.

Як показав аналіз, на даний момент не існує рішень, які б врахувати всі вимоги до системи.

Актуальність теми курсової роботи зумовлена тим, що в умовах розширення сфер надання різноманітних послуг, потребує все більше фінансових затрат та людських ресурсів, тому є необхідність у розробці універсальної рекомендаційної системи по наданню послуг.

Мета курсової роботи – розробка додатку по наданню послуг та рекомендаційної системи для нього.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– Провести аналіз існуючих рішень в області рекомендаційних систем;
– здійснити постановку задачі;
– вибрати інструменти для розробки системи;
– розробити систему по наданню послуг та рекомендаційної із рекомендаціями.

Об'єктом дослідження є програмне забезпечення по наданню послуг.

Предметом дослідження є методи та засоби рекомендаційних систем по наданню послуг.

Наукова новизна. На основі дослідженого матеріалу було розроблено продукт по наданню послуг та рекомендаційної системи для нього. Головною ідеєю розробки даного продукту було безкоштовність та масовість у використанні, так як даний продукт можна легко переносити на будь-які операційні системи сімейства Windows.

Розділ 1. Аналіз предметної області

1.1 Аналіз існуючих рішень у галузі рекомендаційних систем

Останнім часом популярність рекомендаційних систем стала дуже великою. Це пояснюється прагненням компаній спростити вибір користувача у тій чи іншій ситуації та отримати від цього прибуток. З огляду на велику затребуваність систем, які автоматично формують пропозиції користувачам з урахуванням їх інтересів, робляться спроби розробки універсальних рекомендаційних систем.

На даний момент існують такі рішення, пов'язані зі створенням рекомендаційних систем:

– По-перше, це послуги з модулем рекомендацій для певної предметної області. Такий підхід використовується, наприклад, у Apple, Netflix, Amazon та інших компаніях.

– По-друге, це дослідження, створені задля створення універсальної рекомендаційної системи, що забезпечує роботу з різними предметними областями (Unresyst); різні бібліотеки, у яких реалізовані методи, які можна використовуватиме створення власної рекомендаційної системи.

– Третім рішенням є універсальні рекомендаційні системи (Apache Prediction IO), які є закінченим програмним продуктом.

Для порівняння даних рішень було виділено такі критерії:

1) налаштування системи на будь-яку предметну область (універсальність);

2) можливість інтеграції системи в продукт сторонніх розробників (використання системи як вбудованого модуля у власний сервіс);

3) наявність компонентів збору даних для подальшого формування оцінок;

4) наявність модуля аналізу для забезпечення якості даних та рекомендацій;

5) наявність готового програмного продукту над ринком програмного забезпечення.

Порівняння існуючих рішень наведено у табл. 1. Використовується така шкала для оцінювання рішень:

0 – рішення не виконує зазначеної функції;

1 – рішення частково виконує вказану функцію;

2 – рішення повністю виконує вказану функцію.

Таблиця 1. Порівняння існуючих рішень

Критерії	*Рекомендаційні системи у складі готових сервісів*	*Unresyst (дослідницький прототип)*	*Бібліотеки (Crab, LensKit, RankSys)*	*Prediction IO*
Універсальність системи	0	2	1	2
Можливість ітеграції	0	0	0	1
Наявність компонентів збору даних	2	0	0	0
Наявність модуля аналізу	1	0	0	0
Наявність продукту над ринком	0	0	2	2

Універсальність забезпечується системою Unresyst, бібліотеками (платформами) для розробки рекомендаційних систем, Prediction IO. Однак Unresyst не є готовим програмним продуктом, має лише розроблений універсальний метод формування рекомендацій без збирання та аналізу даних. Бібліотеки не є модулями, що вбудовуються, потрібно писати код для того, щоб вони могли приймати дані; крім того, бібліотеки не є масштабованими. Порівняння показує, що Prediction IO найбільше задовольняє поставленим

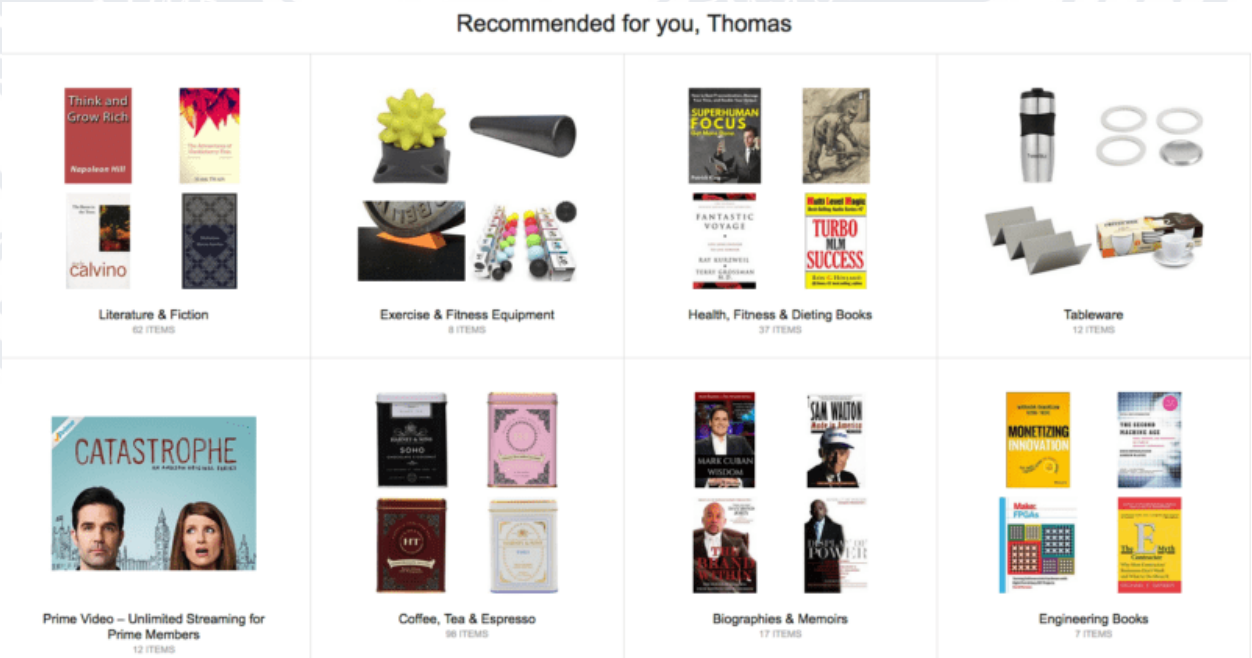
вимогам: його можна використовувати як модуль, що вбудовується в системи сторонніх розробників, проте він не інтегрується повністю з основним сервісом. Крім того, у ньому відсутні модулі збору даних та аналізу, які необхідні при створенні рекомендаційної системи (готові сервіси, що мають модуль рекомендацій, обов'язково мають ці кошти).

Таким чином, виявлені у існуючих рішень обмеження вказують на актуальність розробки універсальної рекомендаційної системи, яка здатна вбудовуватися в сервіс клієнта, має компоненти збору та аналізу даних для забезпечення якості даних і результатів, що використовуються.

Приклади компаній, які використовують систему рекомендацій

Amazon.com

Amazon.com використовує рекомендації спільної фільтрації по елементах на більшості сторінок свого веб-сайту та в кампаніях електронної пошти. За даними McKinsey, 35% покупок Amazon здійснюються завдяки системам рекомендацій. Кілька прикладів, коли Amazon використовує системи рекомендацій



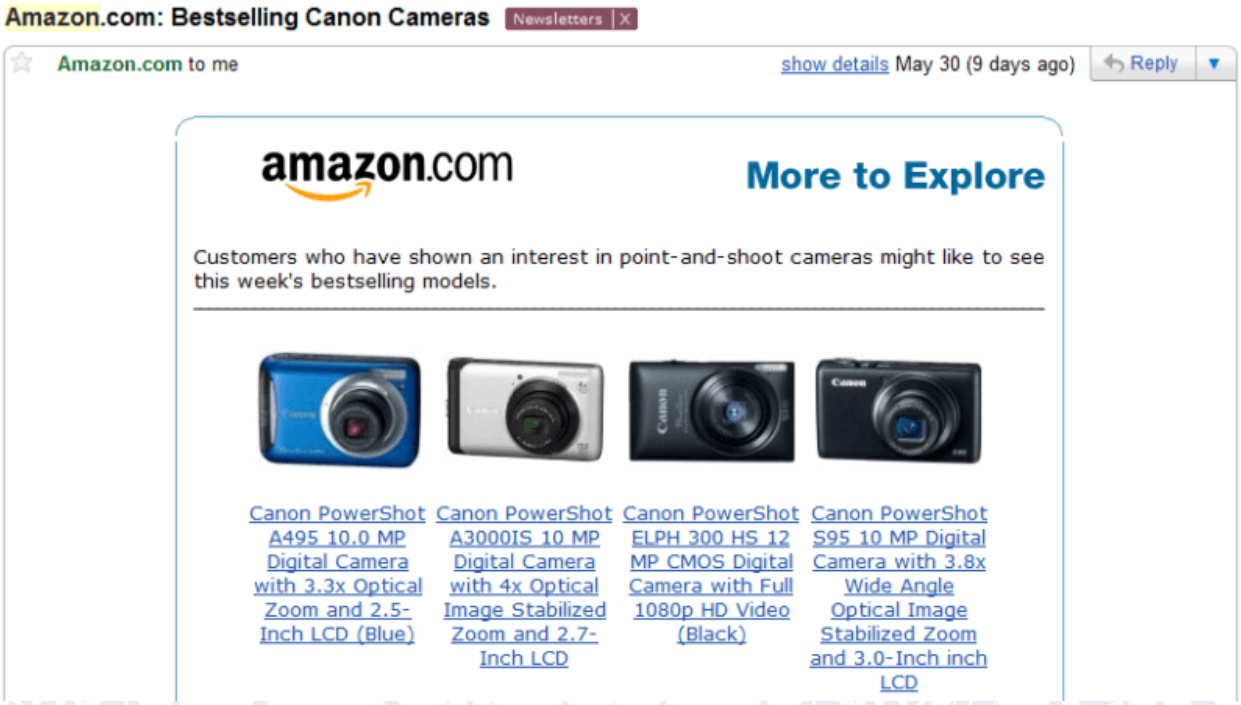


Рис. 1.1, 1,2 – Приклад рекомендаційної системи Amazon.com

Netflix

Netflix — це ще одна компанія, що керується даними, яка використовує системи рекомендацій для підвищення задоволеності клієнтів. Те саме дослідження Mckinsey, яке ми згадували вище, підкреслює, що 75% переглядів Netflix зумовлені рекомендаціями. Насправді Netflix настільки одержимий наданням найкращих результатів для користувачів, що вони провели змагання з вивчення даних під назвою Netflix Prize, де той, хто має найточніший алгоритм рекомендації фільмів, виграє приз у розмірі 1 000 000 доларів США.

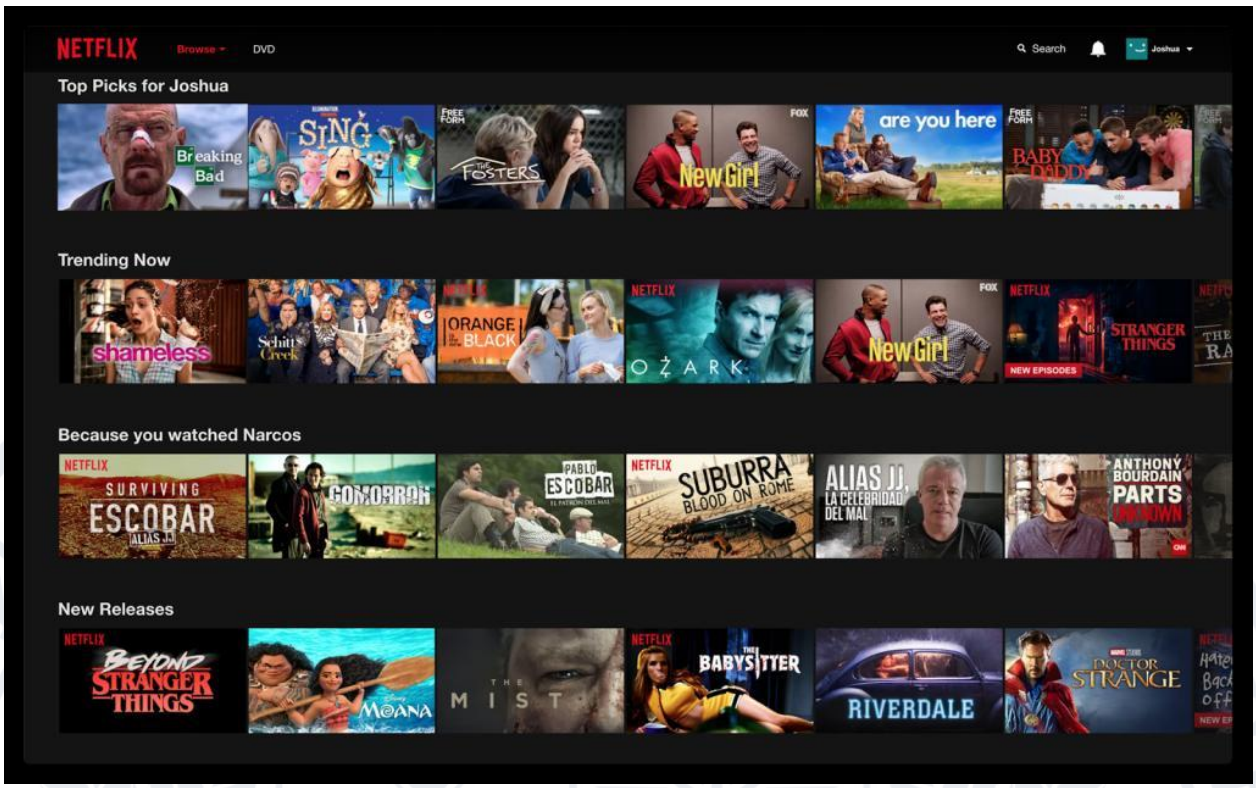


Рис. 1.3 – Приклад рекомендаційної системи Netflix

Spotify

Щотижня Spotify створює новий налаштований список відтворення для кожного підписника під назвою «Discover Weekly». Це персоналізований список із 30 пісень на основі унікальних музичних смаків користувачів. Придбання Echo Nest, музичного інтелектуального стартапу та аналітики даних, дозволило їм створити систему музичних рекомендацій, яка використовує три різні типи моделі рекомендацій:

- **Спільна фільтрація:** фільтрація пісень шляхом порівняння даних про прослуховування користувачами з історією прослуховування інших користувачів.
- **Обробка природної мови:** пошук в Інтернеті інформації про певних виконавців і пісні. Потім кожному виконавцю чи пісні призначається динамічний список найпопулярніших термінів, який змінюється щодня та зважується за релевантністю. Потім механізм визначає, чи схожі два музичні твори або виконавці.

- **Аналіз аудіофайлу:** алгоритм аналізує характеристики кожного окремого аудіофайлу, включаючи темп, гучність, тональність і тактовий розмір, і дає відповідні рекомендації.

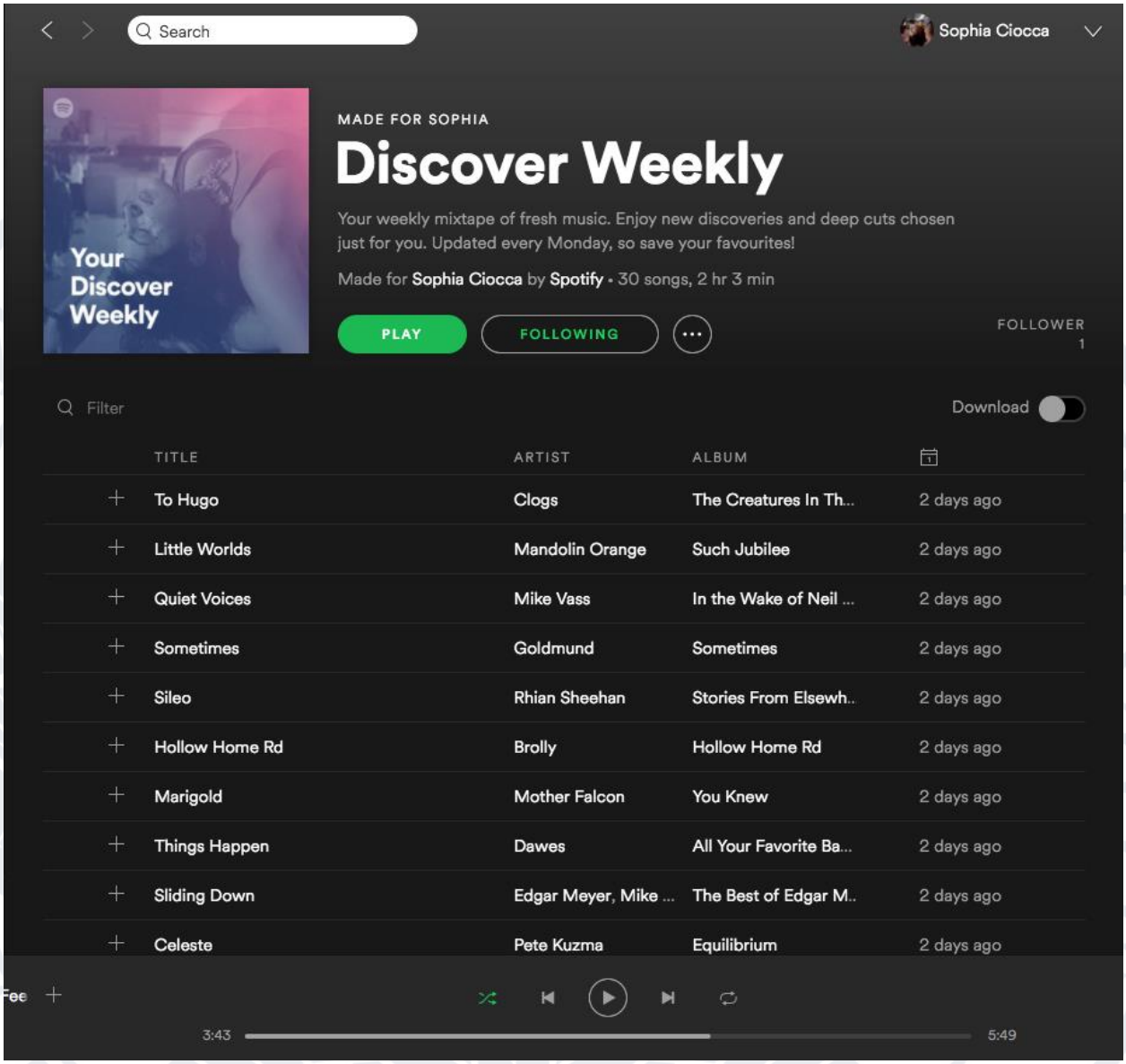


Рис. 1.4 – Приклад рекомендаційної системи Spotify

Linkedin

Як і будь-який інший канал соціальних мереж, LinkedIn також використовує рекомендації типу «Ви також можете знати» або «Вам також може сподобатися».

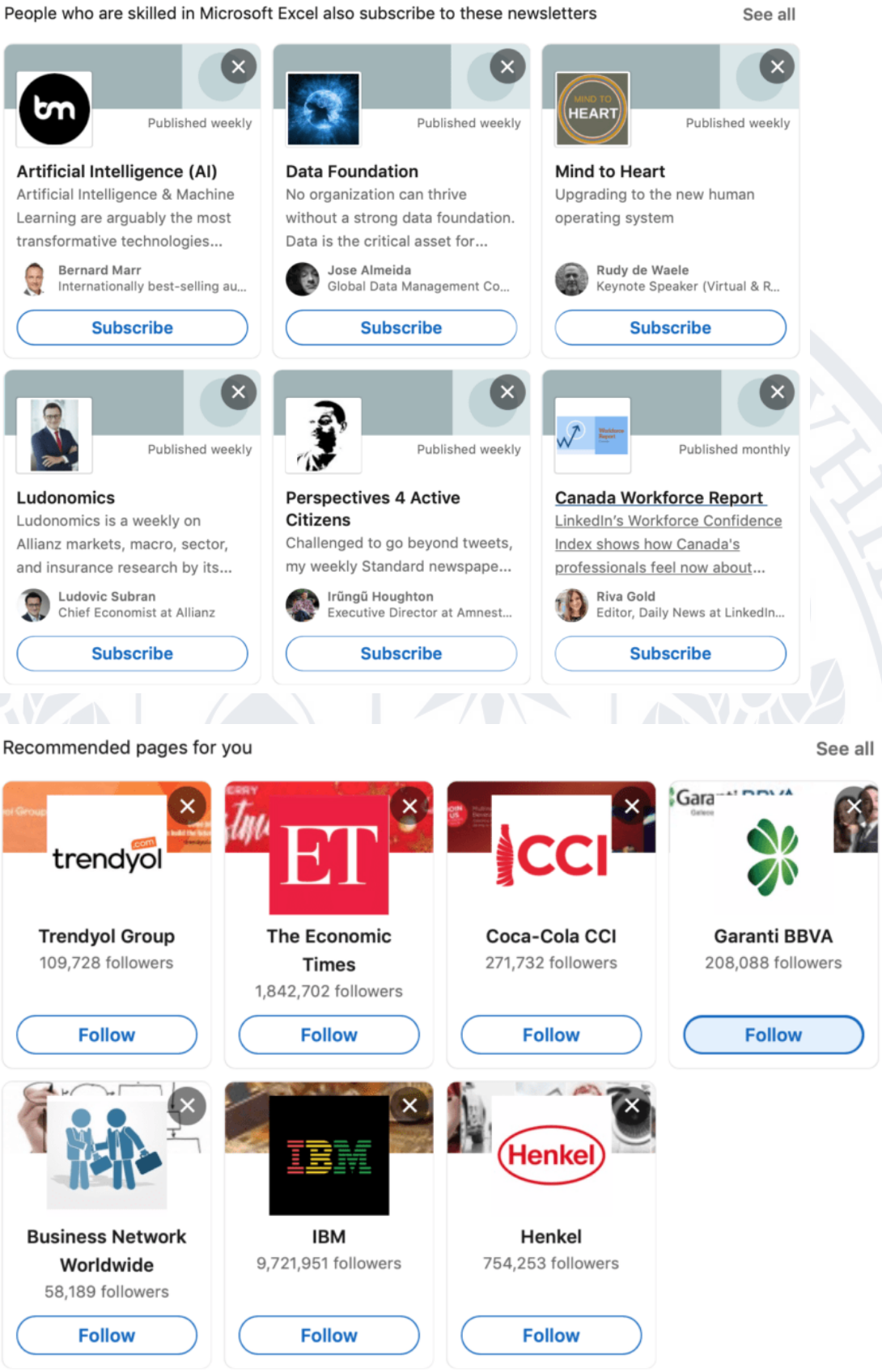


Рис. 1.5, 1.6 – Приклад рекомендаційної системи Linkedin

1.2 Загальний підхід до розробки рекомендаційної системи

Загальна схема роботи з рекомендаційною системою у межах запропонованого підходу показано на рис. 1. Пропонується інтегрувати систему з основним сервісом компанії, що пропонує товари чи послуги, налаштувавши її на відповідну область через створення моделі предметної області на основі даних, які отримують від основного сервісу у певному форматі.

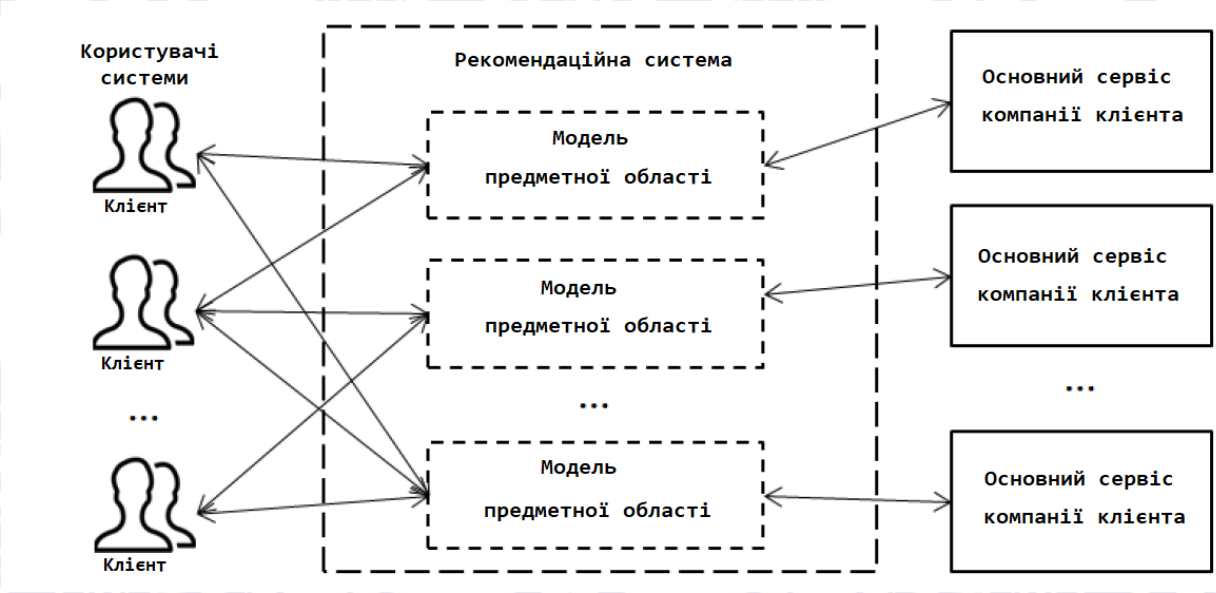


Рис. 1.7 – Схема інтеграції рекомендаційної системи з основними сервісами компаній-клієнтів

Універсальна рекомендаційна система включає дві підсистеми, що взаємодіють з основним сервісом: власне рекомендаційна система, яка забезпечує збір даних та вироблення рекомендацій; модуль обробки та інтелектуального аналізу даних.

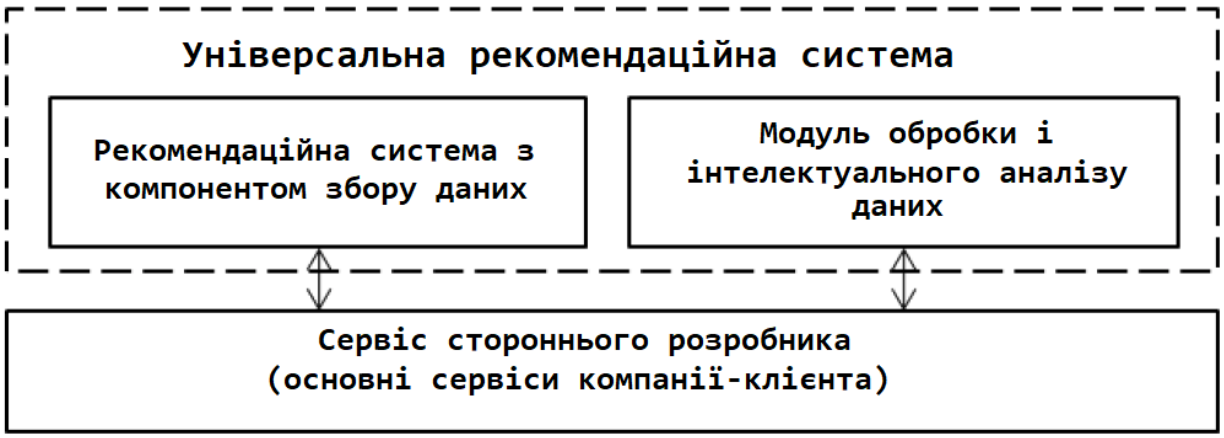


Рис. 1.8 – Підсистеми універсальної рекомендаційної системи

Ці дві підсистеми можуть працювати незалежно одна від одної. Однак доповнення рекомендаційної системи засобами аналізу даних суттєво підвищує якість її роботи.

1.3 Архітектура рекомендаційної системи

Під час проектування архітектури системи, що взаємодіє з сервісом та користувачами, було обрано мікросервісний підхід. Мікросервісний підхід – це особливий спосіб розробки програмних систем, який знаходить широке застосування останніми роками. Мікросервісна архітектура передбачає розробку програмних додатків як набору незалежних невеликих сервісів, що розгортаються, кожен з яких запускає унікальний процес і забезпечує комунікацію за допомогою чітко визначеного, легкого механізму. Завдяки гнучкості та масштабованості ці архітектурні рішення є найбільш підходящими, коли потрібно включити підтримку для цілого ряду платформ та пристроїв різних типів.

Основні переваги застосування мікросервісів:

– архітектура надає розробникам свободу самостійно розробляти та розгортати служби;

– мікросервіс може бути розроблений невеликою командою;

– код для різних служб може бути написаний різними мовами;

– проста інтеграція та автоматичне розгортання (з використанням інструментів безперервної інтеграції з відкритим вихідним кодом, таких як Jenkins, Hudson тощо);

– легкість розуміння та модифікації коду;

– можливість використання розробниками новітніх технологій;

– простота масштабування та інтеграції зі сторонніми сервісами.

Але цей підхід має свої недоліки – розробникам доводиться мати справу зі збільшенням складності робіт під час проектування, реалізації та супроводу розподіленої системи, зокрема:

– розподілена архітектура привносить додаткову складність, оскільки розробникам доводиться балансувати навантаження, підтримувати стійкість до відмови тощо;

– розробники повинні докласти додаткових зусиль для впровадження та підтримки механізмів взаємодії між сервісами, обробки різних форматів повідомлень, реалізації більш дорогих віддалених викликів, повинні використовувати грубіші віддалені API-інтерфейси;

– завдяки розподіленому розгортанню тестування може стати складним та стомлюючим;

– при «перерозподілі обов'язків» між компонентами можуть виникнути проблеми, неузгоджена робота може призвести до дублювання зусиль; обробка функцій, що охоплюють більше однієї служби, потребує взаємодії та співробітництва між різними командами;

– збільшення кількості послуг може призвести до інформаційних бар'єрів;

– коли кількість послуг збільшується, інтеграція та управління системою загалом може стати складною;

– архітектура зазвичай призводить до збільшення споживання пам'яті та ін.

Таким чином, етап проектування архітектури, вибору технологій для реалізації компонентів системи стає найскладнішим та найвідповідальнішим етапом розробки.

У системі виділені такі завдання, реалізовані як окремі мікросервіси: збір даних; формування оцінки об'єктів; видача рекомендацій.

Для службових обчислень обрано архітектуру, засновану на кластерах, що забезпечують прозорість системи. В результаті цього підвищується надійність, збалансованість та продуктивність платформи.

Загальна архітектура рекомендаційної системи представлена рис. 1.3.

Як основний кластер для зберігання та обробки даних використовується програмна платформа Hadoop – система, яка складається з безлічі різних компонентів, що у сукупності створюють єдину платформу.

Ця платформа має відкритий вихідний код і призначена для зберігання та обробки даних, які надто великі для одного конкретного пристрою або сервера. Сила Hadoop полягає в її здатності масштабуватись по тисячах товарних серверів, які не поділяють пам'ять або дисковий простір.

Hadoop делегує завдання через робочі сервери, які називаються "робочими вузлами" або "підлеглими вузлами", суттєво використовуючи можливості кожного пристрою та запускаючи їх одночасно. Для цілей роботи використовується розподілена файлова система Hadoop (HDFS) – масштабована файлова система, яка розподіляє та зберігає дані на всіх машинах у кластері Hadoop (групі серверів), що дозволяє зберігати величезні файли.

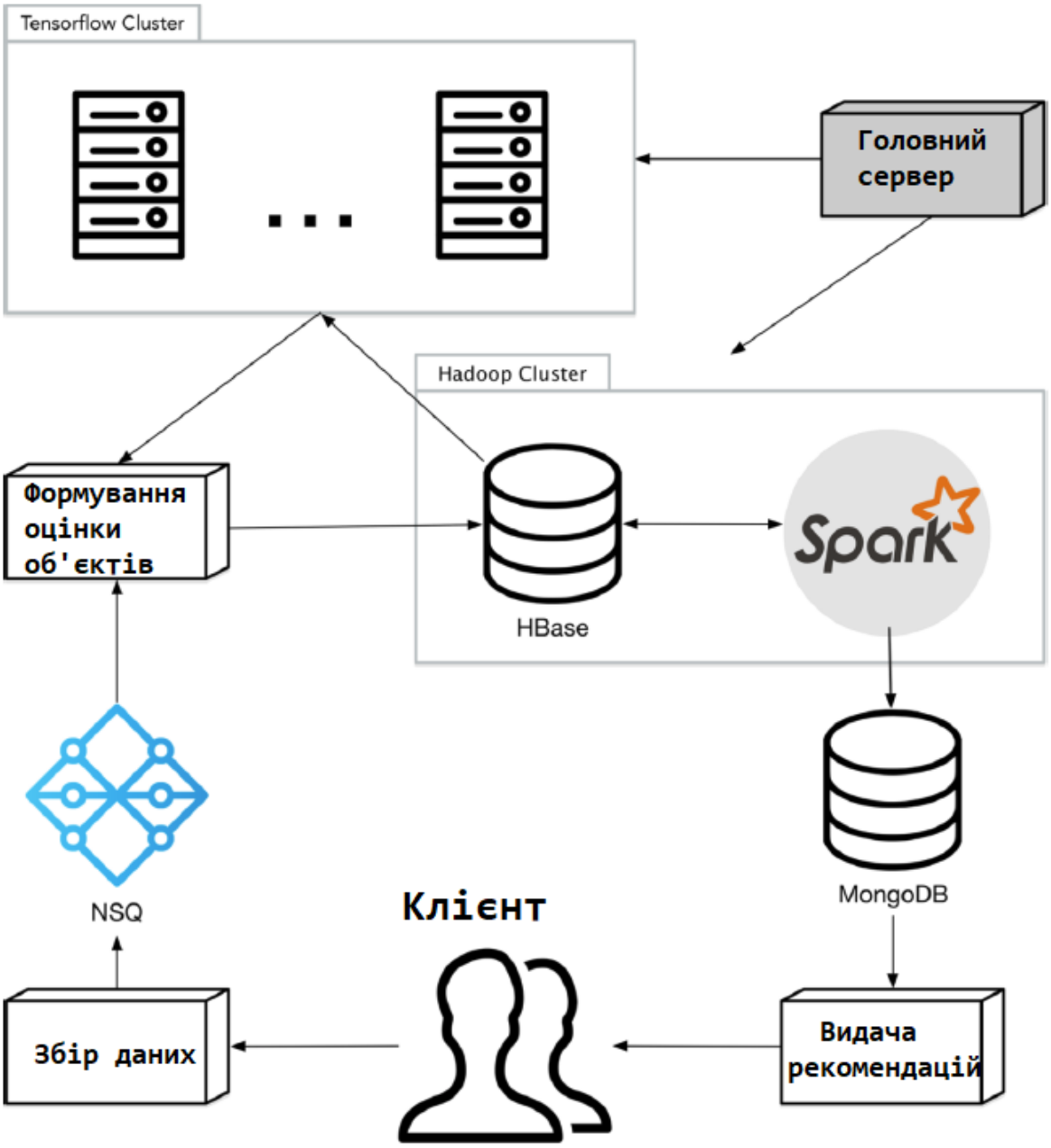


Рис. 1.9 – Спрощена архітектура рекомендаційної системи

На кластері Hadoop як база даних використовується Apache Hbase. Дана база даних є колонковою, працює на HDFS. Вона масштабується зберігання великих обсягів розріджених даних. У схемі Big Data вона відповідає категорії сховища та є альтернативним або додатковим варіантом сховища даних.

Для обробки даних на кластері використовується Apache Spark. Spark – це середовище виконання, яке працює з файловою системою, щоб розподіляти дані по кластеру та обробляти ці дані паралельно. Spark також приймає набір інструкцій із програми, написаної розробником. Це середовище підтримує

різні мови програмування (Java, Python та Scala). Цей кластер використовує бібліотеку MLlib.

Для навчання нейронної мережі, що формує оцінки об'єктів за даними користувачів, використовується кластер TensorFlow. TensorFlow – одна з найпопулярніших бібліотек для машинного навчання. Ця бібліотека використовує тензори та графи як основу. Кластер TensorFlow – це кілька комп'ютерів, на яких запущено сервер TensorFlow, об'єднаних майстер-сервером.

Первинне завдання сервісу збору даних – отримати дані про переглянуті об'єкти від користувачів. Для очікування даних від клієнта потрібний окремий сервіс. Після надсилання даних користувача сервер збору даних ці дані потрапляють на сервер NSQ – розподіленої платформи обміну повідомленнями у часі. Ця платформа підтримує розподілені та децентралізовані топології, забезпечуючи відмовостійкість та високу доступність у поєднанні з надійною гарантією доставки повідомлень, а також горизонтально масштабується. Вбудоване виявлення полегшує додавання вузлів у кластер, підтримує доставку повідомлень. Від NSQ дані розподіляються на мікросервіси формування оцінки об'єктів, де є навчена модель нейронної мережі, яка формує оцінку, якщо вона відсутня, і накопичує пакет повідомлень, щоб надіслати інформацію на сервер бази даних.

Архітектура сервісу формування пропозицій включає три компоненти: головний сервер, кластер Hadoop та компонент видачі рекомендацій. Сервіс формування пропозицій починає свою роботу у Hadoop-кластері, де за сигналом головного сервера відбувається формування пропозицій для користувачів на платформі Apache Spark. Сформовані пропозиції передаються на сервер баз даних MongoDB, де відбувається реплікація даних інші сервери MongoDB до роботи з мікросервісом видачі рекомендацій. Реплікація – процес копіювання та синхронізації даних на кількох серверах – забезпечує надмірність підвищення доступності даних, відмовостійкісті.

При необхідності клієнт (додаток основного сервісу) надсилає запит на отримання пропозицій для користувача. Логіка обробки наданих рекомендацій та видача їх кінцевому користувачеві реалізується розробниками програми самостійно.

Таким чином, система працює з двома СУБД: для збору даних – HBase; для надання рекомендацій – MongoDB. HBase забезпечує швидкий доступ до всієї таблиці для їх обробки. MongoDB має досить велике значення RPS (request per second), що дозволяє обробити велику кількість запитів, а також легко масштабується для потреб мікросервісів.

Обмін даними між сервером та клієнтом необхідний у двох випадках: при зборі даних про переглянуті об'єкти та при видачі рекомендацій. Весь обмін даними передбачає використання REST API.

1.4 Вибір технологій

Для побудови користувацького інтерфейсу будемо використовувати середовище розробки Visual Studio 2019 та мову програмування C#.

C#, вимовляється як "C-sharp", є об'єктно-орієнтованою мовою програмування від Microsoft, яка дозволяє розробникам створювати програми, які працюють на платформі .NET. C# бере свій початок у сімействі мов програмування C і має багато тих самих характеристик, що й у C і C++, а також у Java та JavaScript.

Мова C# була розроблена в Microsoft переважно Андерсом Хейлсбергом, Скоттом Вільтамутом і Пітером Голдом. Microsoft випустила першу широко розповсюджену реалізацію C# у липні 2000 року як частину своєї ініціативи .NET framework. C# мав бути простою, сучасною мовою програмування загального призначення, яку можна було б використовувати для розробки компонентів програмного забезпечення для розподіленого

середовища. Нещодавно випущена C# підкреслила портативність вихідного коду з підтримкою як розміщених, так і вбудованих систем.

C# надає базову, читабельну мову для побудови логіки додатків, приховуючи при цьому основну складність властивих цій мові можливостей. Наразі мова стандартизована відповідно до специфікації ISO/IEC 23270: Інформаційні технології -- Мови програмування -- C#. Специфікація спочатку ґрунтувалася на поданні Hewlett-Packard, Intel і Microsoft. Це вже третє видання, яке вийшло у 2018 році.

З моменту свого появи C# набула широкого поширення і є де-факто мовою програмування для більшості розробок на базі Windows. Цю мову разом із платформою .NET також можна використовувати для розробки додатків для систем під управлінням Linux, macOS, iOS або Android, хоча C# використовується переважно для розробки додатків Windows.

C# вважається строго типізованою мовою, що означає, що кожна змінна та константа мають тип, як і вирази, які обчислюють значення. Тип описує структуру та поведінку даних. Це важливо під час визначення та роботи зі змінними, які можна розглядати як екземпляри типів. C# підтримує дві категорії типів:

Типи значень. Змінні, визначені типами значень, містять свої дані безпосередньо - кожна змінна має власну копію даних і ізольована від інших змінних. Робота однієї змінної типу значення не впливає на інші змінні типу значення. C# підтримує п'ять підкатегорій типів значень: прості типи, типи структур, типи перерахувань, типи значень із можливістю обнулення та типи значень кортежу.

Довідкові типи. Змінні, визначені за допомогою посилальних типів, зберігають лише посилання на свої дані, які називаються об’єктами. Типи посилань дозволяють двом змінним посилатися на один і той же об’єкт, що означає, що операції над однією змінною можуть впливати на об’єкт, на який

посилається інша змінна. C# підтримує чотири підкатегорії посилальних типів: типи класів, типи інтерфейсів, типи масивів і типи делегатів.

Під час створення програм C# розробники можуть використовувати оголошення типів для створення нових типів. Оголошення типу визначає назву та члени нового типу. Оголошення типів базуються на шести підкатегоріях, доступних для типів значень і посилань. Вони включають типи структур, типи enum, типи значень кортежу, типи класів, типи інтерфейсів і типи делегатів..NET Framework - це платформа для розробників із відкритим кодом, яку можна використовувати для створення широкого кола програм. Цей безкоштовний крос-платформний фреймворк підтримує декілька мов і має великі бібліотеки коду, які спрощують створення додатків для мобільних пристроїв, робочих столів та Інтернету.

C# розроблено для роботи з платформою Microsoft .NET, екосистемою програмного забезпечення для розробки, компіляції та запуску програмного коду. Платформа включає загальномовне середовище виконання (CLR) і набір бібліотек класів.

CLR запускає код і надає служби, які дозволяють і вдосконалюють розробку додатків і кросплатформенний дизайн. Він також пропонує високорівневу підтримку мов програмування, таких як C#, F# і Visual Basic.

Коли розробник створює програму C#, вихідний код компілюється в проміжну мову (IL), яка відповідає стандарту Common Language Infrastructure. Код IL та інші ресурси програми зберігаються в збірці, яка завантажується в CLR під час запуску програми. CLR перетворює код IL на рідні машинні інструкції за допомогою процесу своєчасної компіляції.

Платформи CLR і .NET також включають функції, які допомагають оптимізувати та покращити розробку програм:

- Асинхронний код. Забезпечує спрощений підхід до асинхронного програмування з виконанням коду на основі виділення зовнішніх ресурсів і виконання завдань.
- Атрибути. Дозволяє розробникам вставляти додаткову описову інформацію в метадані, які можна отримати за допомогою служб відображення часу виконання.
- Аналізатори коду. Перевіряє код на наявність проблем із якістю та стилем.
- Делегати. Визначає типи, які представляють посилання на методи з певними списками параметрів і типами повернення.
- Події. Надсилає сповіщення, які сигналізують про виконання дій об’єкта.
- Вивіз сміття. Керує розподілом і звільненням пам’яті програми, позбавляючи розробників необхідності писати код для керування пам’яттю.
- Родові типи. Дозволяє розробникам визначати типобезпечні структури даних, не прив’язуючись до фактичних типів даних.
- Паралельне програмування. Дозволяє одночасно виконувати кілька потоків, щоб використовувати ресурси обробки системи.
- Рефлексія. Дозволяє динамічно створювати екземпляр типу, прив’язувати тип до існуючого об’єкта або отримувати тип з об’єкта та отримувати доступ до його компонентів.
- Тип системи. Встановлює загальну систему типів, яка визначає, як типи оголошуються, використовуються та керуються, одночасно надаючи підтримку для виведення типу.
- Небезпечний код. Надає небезпечний контекст, який дозволяє розробникам включати неперевірений код у свої програми. Це не означає, що код небезпечний, лише те, що його неможливо перевірити, як і більшість коду.

.NET Framework також підтримує Language-Integrated Query (LINQ), набір технологій, які інтегрують можливості запитів безпосередньо в мову C#. На додаток до спрощення доступу до даних, LINQ пропонує розробникам послідовний досвід доступу до даних з об’єктів, реляційних баз даних або ресурсів Extensible Markup Language.

Для розробки інформаційної бази використовувався Microsoft Access. Microsoft Access — це програма реляційної бази даних, типова база даних містить таблиці, запити, форми та звіти. За допомогою Microsoft Access ви можете легко впорядковувати, зберігати та отримувати дані. Є кілька переваг або переваг використання Microsoft Access.

Нижче наведено 10 переваг використання Microsoft Access:

1. Access пропонує доступне рішення для малого та середнього бізнесу та невеликих команд у великих організаціях.
2. Його легше вивчити та використовувати, ніж клієнт-серверну базу даних.
3. Його легко імпортувати та експортувати в інші програми Microsoft Office, такі як Excel.
4. База даних Access може посилатися на зовнішні бази даних і надсилати запити та звітувати про результати.
5. Користувачі можуть підтримувати базу даних Access на сервері або настільному комп’ютері (вам не обов’язково використовувати хмарне рішення).
6. Користувачі можуть розробляти та контролювати базу даних, а не залежати від зовнішніх джерел або консультантів.
7. Access дозволяє автоматизувати за допомогою різних типів макросів.
8. Звіти можна виводити у форматі PDF (Portable Document Format).

9. Користувачі можуть використовувати SQL (Structured Query Language) для взаємодії з базою даних. Бази даних Access також можна перенести на SQL Server.
10. Microsoft пропонує кілька шаблонів баз даних Access для початку.

Давайте розглянемо деякі ключові функції Microsoft Access.

- Бази даних Access можуть містити одну або кілька таблиць для зберігання даних

У Access дані зберігаються в таблицях (подібно до електронних таблиць). Таблиця може містити багато полів для розділення даних. Поле в таблиці можна налаштувати для різних типів даних і дозволити або заборонити користувачам вводити певну інформацію.

База даних Microsoft Access може містити кілька таблиць, і ви можете зв’язати таблиці за ключовим полем.

- Дані можна імпортувати з Excel та інших баз даних

Ви можете імпортувати дані з Excel у Microsoft Access або з іншої зовнішньої бази даних. У Access ви також можете посилатися на зовнішні бази даних, а не імпортувати дані. Ви все ще зможете створювати запити та звіти, використовуючи зв’язані таблиці.

- Форми можна створювати для введення або перегляду даних

Ви можете створювати форми Access, які слугуватимуть інтерфейсом для введення або редагування даних у ваших таблицях. База даних Access може містити кілька форм із відображенням різних полів. Форми можуть відображати один запис за раз або кілька записів. Ви можете налаштувати зовнішній вигляд форм і включити кнопки навігації.

- Користувачі можуть створювати та запускати запити для отримання даних

У Access ви можете створювати запити для відображення даних, які відповідають певним критеріям. Зберігати та повторно використовувати запити, а також використовувати їх як джерело для звітів або експортувати в Excel, дуже просто

У Access є кілька типів запитів, зокрема:

- Виберіть запити
- Видалити запити
- Оновлення запитів
- Зробіть запити до таблиці
- Додавання запитів
- Перехресні запити

- Звіти можна розробити та роздрукувати або вивести у формат PDF

Звіти Access значно відрізняються від звітів Excel. Вони можуть включати кілька полів з однієї або кількох таблиць і включати групування, проміжні підсумки та загальні підсумки. Ви також можете легко експортувати звіти у формат PDF (Portable Document Format). Звіти про доступ можуть містити логотип компанії, рядки для розділення записів і навіть підзвіти.

- Бази даних Access можна автоматизувати за допомогою макросів

Access містить конструктор макросів для створення макросів для автоматизації повторюваних завдань.

- Бази даних Access можна автоматизувати за допомогою VBA

Ви також можете автоматизувати Access, написавши макроси у VBA (Visual Basic для програм). Access — це єдина програма Microsoft Office, яка пропонує два методи розробки та запуску макросів.

- Користувачі можуть взаємодіяти з базами даних Access за допомогою SQL

Ви також можете взаємодіяти з Microsoft Access за допомогою SQL (мова структурованих запитів). Ви можете писати запити в SQL або використовувати сітку розробки запитів. Новіші версії Access використовують SQL у серверній частині.

- Access має невисоку ціну та входить до складу окремих ліцензій Microsoft

Залежно від ліцензії Microsoft Access може входити до інших ваших програм Microsoft (наприклад, ліцензія Microsoft Business може включати Access). Його також можна придбати як окрему програму за низькою ціною порівняно з альтернативними базами даних.

Висновок

Microsoft Access пропонує недороге потужне рішення для малих і середніх підприємств або команд у великій організації, яким потрібно створити базу даних і керувати нею. Оскільки його можна автоматизувати кількома способами та налаштувати відповідно до потреб клієнта чи користувача, його можна ефективно використовувати в багатьох сценаріях.

1.5 Типи рекомендаційних систем

Персоналізована система рекомендацій більш детально аналізує дані користувачів, їх покупки, рейтинг і стосунки з іншими користувачами. Таким чином кожен користувач отримає індивідуальні рекомендації.

Найпопулярнішими типами персоналізованих систем рекомендацій є фільтрація на основі вмісту та спільна фільтрація.

На основі вмісту

Системи рекомендацій на основі вмісту використовують елементи або метадані користувачів для створення конкретних рекомендацій. Спостерігається історія покупок користувача. Наприклад, якщо користувач уже прочитав книгу одного автора або купив продукт певної марки, припускається, що клієнт віддає перевагу цьому автору або цій марці, і існує

ймовірність того, що користувач придбає подібний продукт у майбутнє. Припустімо, що Дженні любить науково-фантастичні книги, а її улюбленим письменником є Волтер Джон Вільямс. Якщо вона прочитає книгу Арістоя, то її рекомендованою книгою буде Станція Ангелів, також науково-фантастична книга, написана Уолтером Джоном Вільямсом.

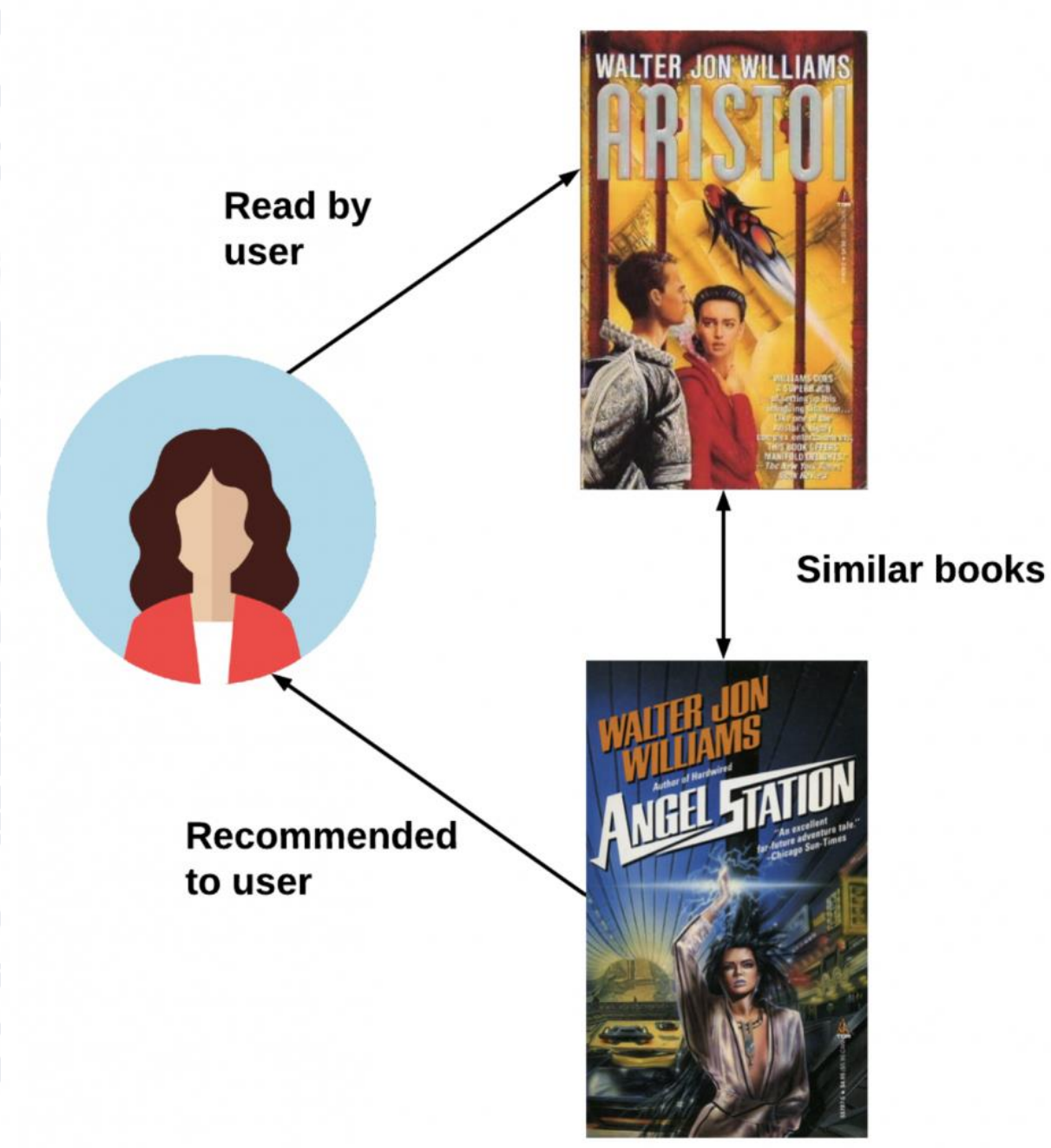


Рис. 1.10 – Система рекомендацій на основі вмісту

Спільна фільтрація на практиці дає кращі результати, ніж підхід на основі вмісту. Можливо, це тому, що в результатах немає такої різноманітності, як у спільній фільтрації.

Недоліки контент-орієнтованого підходу:

- Існує так зване явище бульбашки фільтра. Якщо користувач читає книгу про політичну ідеологію і йому рекомендують книги, пов’язані з цією ідеологією, він опиниться в «міхурі своїх попередніх інтересів».
- Щоб отримати найкращу рекомендацію, потрібно зібрати багато даних про користувача та його вподобання
- На практиці існує 20% товарів, які привертають увагу 70-80% користувачів, і 70-80% товарів, які привертають увагу 20% користувачів. Мета Recommender — представити інші продукти, які на перший погляд недоступні користувачам. У підході, що базується на вмісті, ця мета не досягається так само добре, як у спільному фільтруванні.

Спільна фільтрація

Ідея спільної фільтрації проста: поведінка групи користувачів використовується для надання рекомендацій іншим користувачам. Оскільки рекомендація базується на вподобаннях інших користувачів, вона називається спільною.

Існує два типи спільної фільтрації: на основі пам’яті та на основі моделі.

На основі пам'яті

Методи на основі пам’яті застосовуються до необроблених даних без попередньої обробки. Їх легко реалізувати, а отримані рекомендації, як

правило, легко пояснити. Кожного разу необхідно робити прогнози над усіма даними, що уповільнює рекомендувач.

Існує два типи: спільна фільтрація на основі користувачів і на основі елементів.

- На основі користувачів – «Користувачам, які схожі на вас, також сподобалося…» Продукти рекомендовані користувачеві на основі того факту, що вони були придбані / сподобалися користувачам, схожим на спостережуваного користувача. Якщо ми говоримо, що користувачі схожі, що це означає? Наприклад, Дженні і Том люблять науково-фантастичні книги. Коли з'явиться нова науково-фантастична книга, і Дженні купить цю книгу, оскільки Том також любить науково-фантастичні книги, ми можемо порекомендувати книгу, яку купила Дженні.

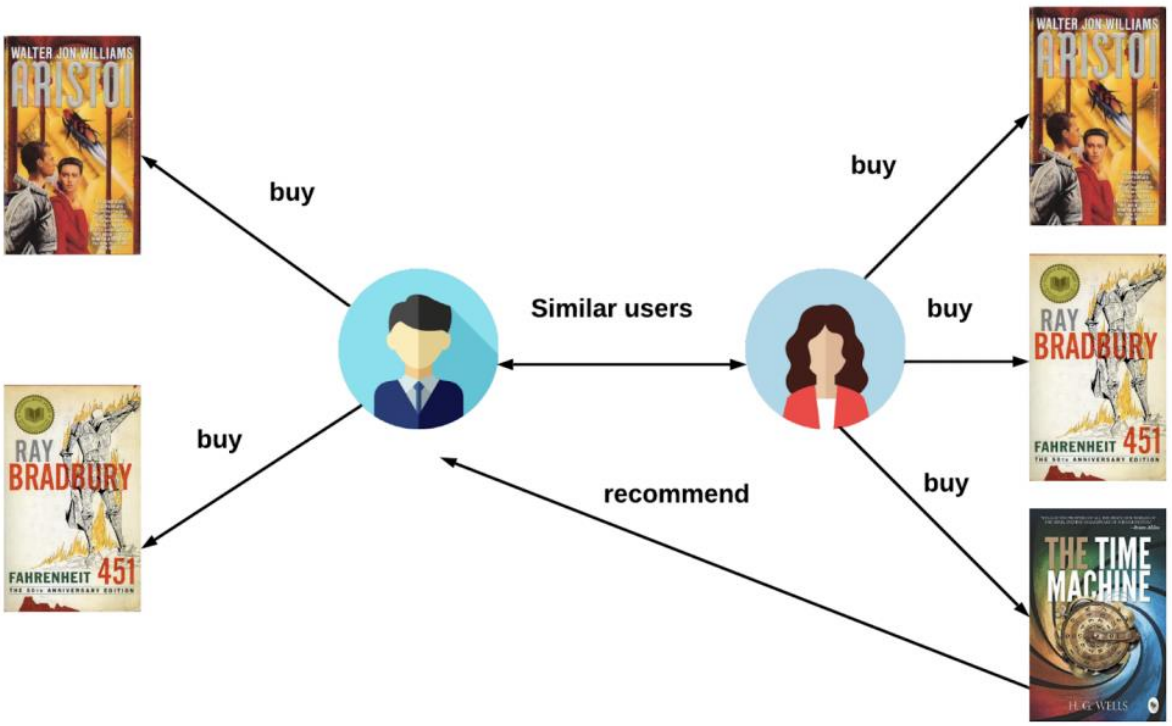


Рис. 1.11 – Рекомендована система спільної фільтрації на основі користувача

- На основі предмета – «Користувачам, яким сподобався цей предмет, також сподобався…» Якщо Джон, Роберт і Дженні високо оцінили науково-фантастичні книги «451 градус за Фаренгейтом» і «Машина часу», наприклад, дали 5 зірок, тоді, коли Том купує книгу «451 градус за Фаренгейтом», тоді книга «The машина часу також рекомендована йому, оскільки система визначила книги як схожі на основі оцінок користувачів.

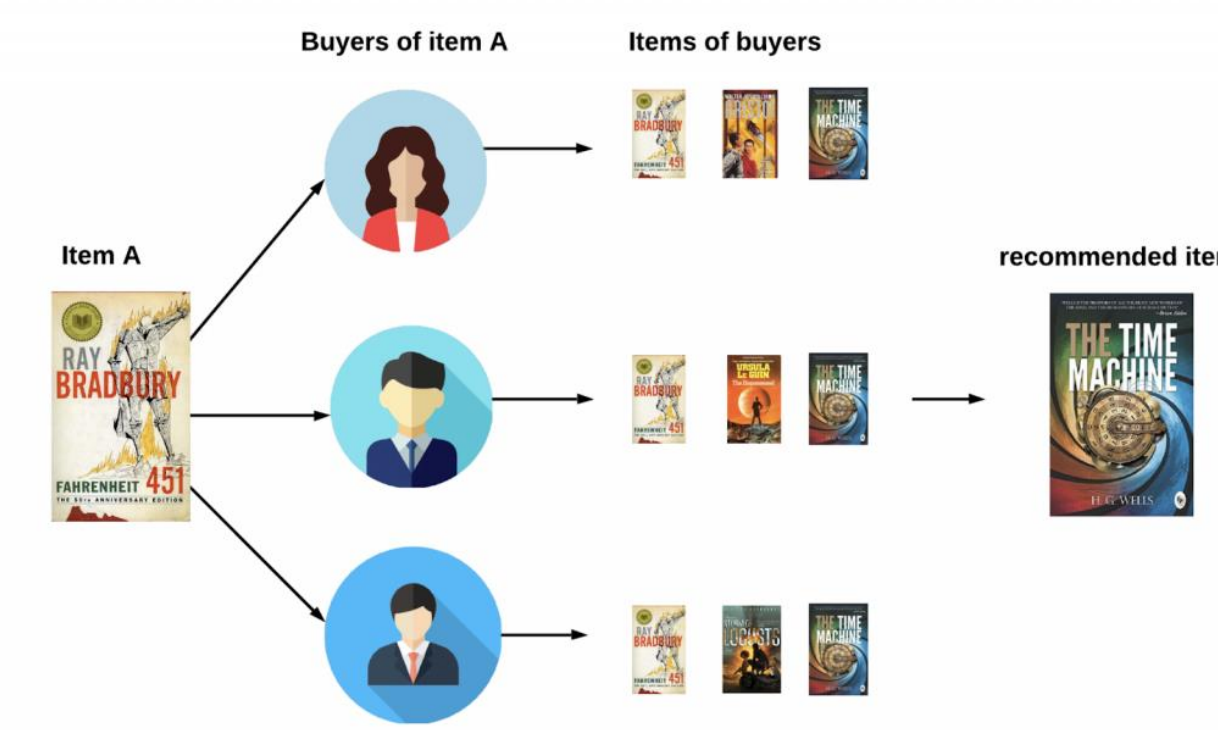


Рис. 1.12 – Рекомендована система спільної фільтрації на основі елементів

Як обчислити подібність між користувачами та предметами?

На відміну від підходу, заснованого на вмісті, де використовуються метадані про користувачів або елементи, підхід, заснований на спільному фільтруванні пам’яті, спостерігає за поведінкою користувачів, напр. чи сподобався або оцінив товар користувачеві, чи сподобався або оцінив товар певний користувач.

Наприклад, ідея полягає в тому, щоб порекомендувати Роберту нову науково-фантастичну книгу.

Кроки:

1. Створення матриці оцінки товару користувача
2. Створення матриці подібності між користувачами
    - Косинусна подібність обчислюється (альтернативи: скоригована косинусна подібність, подібність Пірсона, кореляція рангу списоносця) між кожними двома користувачами. Таким чином виходить матриця користувач-користувач. Ця матриця менша, ніж початкова матриця оцінки елементів користувача.

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

3. Пошук схожих користувачів
    - У матриці користувач-користувач спостерігаються користувачі, найбільш схожі на Роберта
4. Кандидатське покоління
    - Коли знайдено найбільш схожих користувачів Роберта, ми розглядаємо всі книги, які ці користувачі прочитали, і оцінки, які вони поставили.
5. Оцінка кандидатів
    - Залежно від оцінок книги розташовуються від тих, які сподобалися найбільш схожим користувачам Роберта, до тих, які сподобалися їм найменше.
    - Результати нормалізуються (за шкалою від 0 до 1)

6. Фільтрування кандидатів
    - Перевіряється, чи Роберт уже купив якусь із цих книг. Ті книжки треба виключити, бо він уже їх прочитав.

Розрахунок подібності між елементами виконується таким же чином і має ті самі кроки, що й подібність між користувачами.

Порівняння підходів, орієнтованих на користувача та на основі предметів

Подібність між елементами більш стабільна, ніж подібність між користувачами, тому що підручник з математики завжди буде підручником з математики, але користувач може змінити свою думку, наприклад, те, що йому сподобалося минулого тижня, може не сподобатися наступного тижня. Ще одна перевага полягає в тому, що продуктів менше, ніж користувачів. Це призводить до висновку, що матриця елемент-предмет із балами подібності буде меншою, ніж матриця користувач-користувач. Крім того, на основі елементів є кращим підходом, якщо новий користувач відвідує сайт, тоді як підхід на основі користувачів у цьому випадку проблематичний.

На основі моделі

Ці моделі були розроблені з використанням алгоритмів машинного навчання. Створюється модель і на основі неї, а не всіх даних, даються рекомендації, що прискорює роботу системи. Цей підхід забезпечує кращу масштабованість. У цьому підході часто використовується зменшення розмірності. Найвідомішим типом цього підходу є матрична факторизація.

Матрична факторізація

Якщо є відгук від користувача, наприклад, користувач переглянув певний фільм або прочитав певну книгу та дав оцінку, яку можна представити

у формі матриці, де кожен рядок представляє конкретного користувача, а кожен стовпець представляє конкретний предмет. Оскільки практично неможливо, щоб користувач оцінив кожен елемент, ця матриця матиме багато незаповнених значень. Це називається розрідженістю. Методи матричної факторизації використовуються для пошуку набору прихованих факторів і визначення уподобань користувача за допомогою цих факторів. Про приховану інформацію можна повідомляти, аналізуючи поведінку користувача. Приховані фактори інакше називають особливостями.

Чому факторізація?

Матриця оцінок є добутком двох менших матриць – матриці елемент-ознака та матриця характеристик користувача.

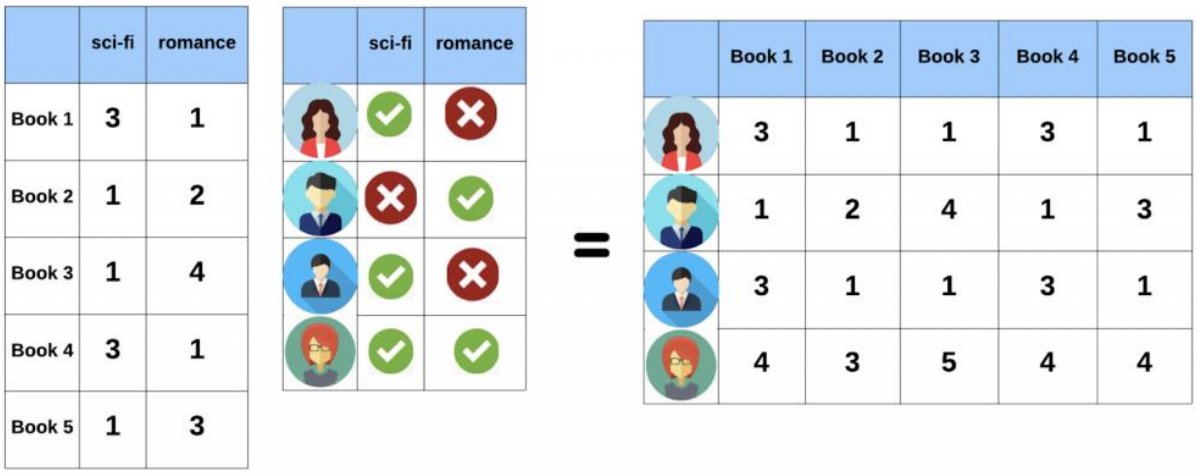


Рис. 1.13 – Матрична факторізація

Етапи розкладання матриці на множники:

1. Ініціалізація випадкового користувача та матриці елементів
2. Матриця оцінок отримується множенням користувача та транспонованої матриці елементів
3. Метою матричної факторизації є мінімізація функції втрат (різниця в рейтингах прогнозованої та фактичної матриць має бути мінімальною). Кожен рейтинг можна описати як скалярний добуток рядка в матриці користувача та стовпця в матриці елементів.

$$\min_{Q*,P*} \sum_{(u,i)\in K} (r_{ui} - P_u^T Q_i)^2 + \lambda(||Q_i||^2 + ||P_u||^2)$$

Рис. 1.14 – функція мінімізації втрат

Де K – набір (u,i) пар, r(u,i) – оцінка елемента і користувачем u, а λ – регуляризаційний термін (використовується для уникнення переобладнання).

1. Щоб мінімізувати функцію втрат, ми можемо застосувати стохастичний градієнтний спуск (SGD) або чергування найменших квадратів (ALS). Обидва методи можна використовувати для поступового оновлення моделі, коли надходить новий рейтинг. SGD є швидшим і точнішим, ніж ALS.

Гібридні рекомендації

Вони являють собою комбінацію різних рекомендацій. Припускається, що комбінація кількох різних рекомендацій дасть кращі результати, ніж один алгоритм.

1.6 Постановка задачі

Огляд сучасних підходів та засобів до проектування та розроблення програмного забезпечення дозволив обрати для створення власної системи ефективні технології та інструментальні засоби: IDE – Visual Studio 2019; Мова програмування – C#.

На основі виконаного аналізу предметної області можна сформулювати постановку задачі.

Розробити систему, по наданню послуг та рекомендаційної системи для нього. Забезпечити внесення даних про співробітників, клієнтів, послуги та рекомендації.

Розробити зручний графічний інтерфейс для роботи з програмою. Основні дії та взаємодія між користувачем та системою повинні супроводжуватися відповідними повідомленнями для користувача.

Створити навігаційне меню для можливості швидкого та зручного отримання доступу до потрібної функції в системі.

Розробити супровідну документацію до створеної системи.

Дипломна робота припускає розробку додатка засобами об’єктно-орієнтованого середовища програмування .

Реалізація додатка виконується з використанням технологій .NET та мови програмування c#. Результат –Windows додаток.

Вона повинна містити в собі:

Класи додатку:

– клієнти ;
– співробітники;
– послуги;
– рекомендації;
– звернення клієнтів.

Введення:

Вводиться і редагується: інформація про співробітників, клієнтів, послуги та рекомендації.

Фіксація:

– фіксуються звернення ;
– фіксуються вирішені послуги.

Результат роботи:

– Надання послуг клієнтам.

Серед усіх функцій, які виконує система можна виділити загальні операції:

- реакція програми на вибір меню користувача;
- реакція програми на натискання кнопок в програмі;
- реакція програми на введення неправильних даних.

Функціональні вимоги:

- можливість додавати, редагувати та видаляти дані про співробітників, клієнтів, послуги та рекомендації;
- фіксація звернень клієнтів на надання послуг;
- фіксація виконаних послуг
- .

Нефункціональні вимоги:

- для роботи програми на комп'ютері повинна бути встановлена бібліотека класів .NET Framework 4.7;
- для забезпечення роботи програми потрібно мати лише клавіатуру та мишку.

Розділ 2. Розробка додатку

2.1 Аналіз вимог. Use-case діаграми. Основні прецеденти

Згідно з поставленою задачею можна висунути такий список вимог до проекту:

Таблиця 2.1 – Функціональні вимоги до додатку «Додаток по наданню послуг»

Вимоги	Опис
REQ-1	Система повинна дозволяти користувачеві додати та редагувати інформацію про клієнтів
REQ-2	Система повинна дозволяти користувачеві додати та редагувати інформацію про співробітників
REQ-3	Система повинна дозволяти користувачеві додати та редагувати інформацію про послуги
REQ-4	Система повинна дозволяти користувачеві додати та редагувати інформацію про рекомендації послуги
REQ-5	Система повинна дозволяти здійснювати реєстрацію послуги
REQ-6	Система повинна дозволяти здійснювати фіксацію виконання послуги

Таблиця 2.2 – Нефункціональні вимоги до додатку «Додаток по наданню послуг»

Вимоги	Опис
REQ-7	Додаток повинен мати простий дизайн та зручну навігації
REQ-8	Поля повинні бути не порожніми, унікальними відносно вже існуючих записів

Таблиця 2.3 – Актори та цілі додатку «Додаток по наданню послуг»

Актори	**Цілі**
Користувач	Мета співробітника полягає в роботі з системою.
База даних	Мета бази даних полягає у зберігання інформації

Таблиця 2.4 – Опис варіантів використання додатка «Додаток по наданню послуг»

Варіант використання	**Ім'я**	**Опис**
UC1	Вивід каталогу співробітників	Дозволяє користувачеві вивести каталог всіх співробітників
UC2	Додати співробітника	Дозволяє користувачеві додати нового співробітника
UC3	Редагувати співробітника	Дозволяє користувачеві редагувати інформацію вибраного із списку співробітника
UC4	Вивід каталогу послуг	Дозволяє користувачеві вивести каталог всіх послуг
UC5	Додати послугу	Дозволяє користувачеві додати нову послугу
UC6	Редагувати послугу	Дозволяє користувачеві редагувати інформацію вибраної із списку послуги
UC7	Вивід каталогу клієнтів	Дозволяє користувачу вивести список всіх клієнтів
UC8	Додати клієнта	Дозволяє користувачу додати нового клієнта в систему
UC9	Редагувати клієнта	Дозволяє користувачеві редагувати інформацію вибраного із списку клієнта

UC10	Рекомендаційні послуги	Дозволяє користувачу вивести список послуг в залежності від введених параметрів
UC11	Додати послугу	Дозволяє користувачу додати нову послугу
UC12	Редагувати послугу	Дозволяє користувачеві редагувати інформацію про вибрану рекомендацію

З поставлених вимог (див. розділ 1) тепер можна виставити повний опис вимог із сценаріями, що будуть вхідними даними для візуального моделювання мовою UML.

UC1 Вивід каталогу співробітників

Актор: користувач.

Ціль актора: вивести інформацію про всіх співробітників системи.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Співробітники».

Післяумова: система відображає екран для виведення списку всіх зареєстрованих співробітників системи.

UC2 Додати співробітника

Актор: користувач.

Ціль актора: додати нового співробітника.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про співробітника натискає на кнопку «Додати».

Післяумова: система додає нового співробітника та відображає екран із списком всіх співробітників.

UC3 Редагувати співробітника

Актор: користувач.

Ціль актора: редагувати інформацію про вибраного співробітника.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку співробітника.

Післяумова: система відображає екран для редагування інформації про вибраного співробітника.

UC4 Вивід каталогу послуг

Актор: користувач.

Ціль актора: вивести інформацію про послуги.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Послуги».

Післяумова: система відображає екран для виведення списку всіх послуг.

UC5 Додати послугу

Актор: користувач.

Ціль актора: додати нову послугу.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про послугу натискає на кнопку «Додати».

Післяумова: система додає нову послугу і відображає екран із списком всіх послуг.

UC6 Редагувати послугу

Актор: користувач.

Ціль актора: редагувати вибрану із списку послугу.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку послугу.

Післяумова: система відображає екран для редагування інформації про вибрану послугу.

UC7 Вивід каталогу клієнтів

Актор: користувач.

Ціль актора: вивести інформацію про всіх клієнтів.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Клієнти».

Післяумова: система відображає екран для виведення списку всіх клієнтів.

UC8 Додати клієнта

Актор: користувач.

Ціль актора: додати нового клієнта.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про клієнта натискає на кнопку «Додати».

Післяумова: система додає нового клієнта та відображає екран із списком всіх клієнтів.

UC9 Редагувати клієнта

Актор: користувач.

Ціль актора: редагувати вибраного із списку клієнта.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку клієнта.

Післяумова: система відображає екран для редагування інформації про вибраного клієнта.

UC10 Рекомендаційні послуги

Актор: користувач.

Ціль актора: вивести рекомендації послуг в залежності від умов клієнта.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Рекомендації по підбору послуг».

Післяумова: система відображає екран для підбору послуг в залежності від вибраних параметрів.

UC11 Додати послугу клієнта із рекомендацій

Актор: користувач.

Ціль актора: додати нову послугу.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію та вибравши послугу із рекомендацій натискає на кнопку «Додати».

Післяумова: система додає нову послугу клієнта.

UC12 Редагувати послугу клієнта

Актор: користувач.

Ціль актора: редагувати вибрану із послугу клієнта.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку послугу клієнта.

Післяумова: система відображає екран для редагування інформації про вибрану послугу клієнта із можливістю рекомендацій послуг.

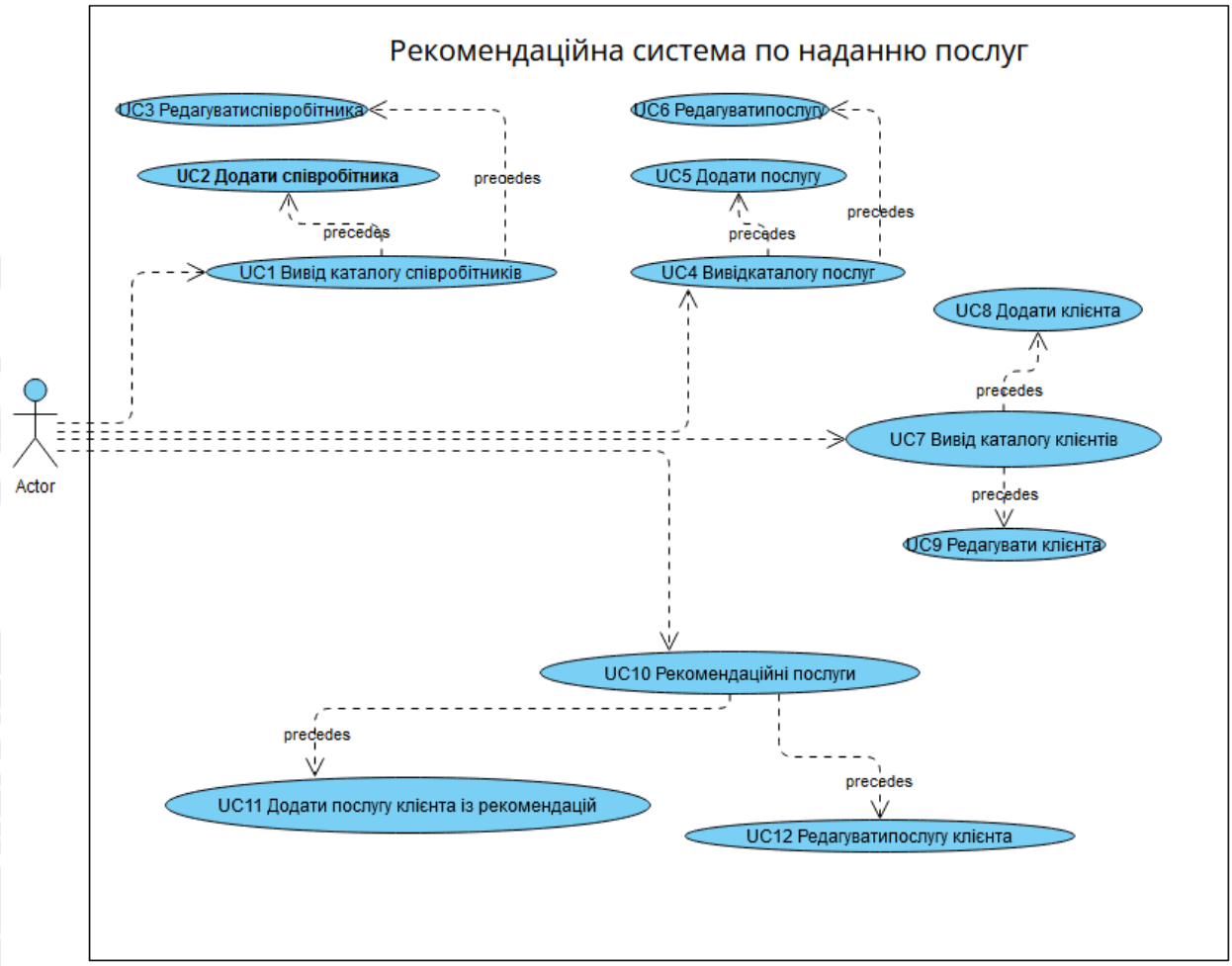


Рис. 2.1 – Діаграма use-case

2.2 Особливості розробки бази даних. ERD діаграма з описанням сутностей

Діаграма взаємозв'язків сутностей — це тип блок-схеми, яка дає змогу проілюструвати, як сутності (люди, об'єкти чи концепції) пов'язані одна з одною в системі. Щоб отримати інтуїтивно зрозумілу картину системи, діаграми ER використовують набір символів, таких як трикутники, прямокутники, ромби, овали та лінії, які відображають зв'язки між об'єктами. Типова діаграма сутності відображає граматичну структуру: сутності виражаються іменниками, а зв'язки – дієсловами.

Діаграма зв’язків сутностей була популярним інструментом візуалізації даних протягом десятиліть. Ще в 1970-х роках Пітер Чен, професор Університету Карнегі-Меллона, розробив ER-моделювання для проектування баз даних. Інструмент швидко злетів. Наприкінці 1970-х років Чарльз Бахман і A.P.G. Браун розширив підхід Чена, щоб побудувати вдосконалену версію діаграми ER, яка є досить близькою до того, що ми використовуємо сьогодні.Згідно завдання будуємо ERD діаграму.

Коли використовувати діаграму зв’язку сутності

Діаграми зв’язків сутностей мають різноманітні застосування. Ось приклад того, що ви можете робити з ERD:

- **Навчання своїх колег**: ERD — це потужні інструменти для навчання ваших товаришів по команді зв’язкам між системами чи об’єктами в організаційній структурі.
- **Проектування та усунення неполадок баз даних**. Багато команд використовують діаграми ER як інструмент розробки бази даних для моделювання баз даних взаємозв’язків. Розробники програмного забезпечення використовують діаграми ER, щоб визначити вимоги до проекту перед початком роботи. ERD також є популярним вибором для усунення несправностей існуючих баз даних, що дозволяє групам знаходити та вирішувати логічні проблеми.
- **Аналіз даних**: Дослідники часто використовують діаграми взаємозв’язків сутностей для налаштування та аналізу баз даних.

- **Інформація про нових найманих**: ERD є візуальними, тож вони є простим способом продемонструвати інформацію для нових працівників і швидко навчити їх працювати.
- **Створення документації**: перш ніж змінити процес, багато команд створюють діаграми ER, щоб документувати існуючі процеси. Таким чином вони можуть переконатися, що вжито всіх необхідних заходів на випадок, якщо їм доведеться повернутися до попереднього процесу.

У яких областях найчастіше використовуються діаграми зв'язків сутностей?

Діаграми взаємозв'язків сутностей є популярним способом візуалізації інформації в різних областях.

Інженери-програмісти покладаються на ER-діаграми для розробки або усунення несправностей інформаційних баз даних. Складання ERD часто є першим кроком у діагностиці проблеми з базою даних.

У сфері бізнес-інформації інструменти діаграм ER часто використовуються для розробки, аналізу або очищення реляційних баз даних. Вони корисні для оптимізації процесів, розуміння залежностей і покращення результатів.

Дослідники використовують ERD для роботи зі структурованими даними. Багато досліджень починаються з великого обсягу необроблених даних із шумом. Потім дослідники можуть використовувати діаграму зв'язків сутностей, щоб створити корисну базу даних, яка дозволить їм краще розуміти свої дані.

Як намалювати діаграму зв'язку сутності

1. **Вирішення рівню деталізації, який включити до діаграми.** Залежно від цілей можна намалювати ERD на високому рівні або збільшити масштаб. Деякі моделі є концептуальними, тоді як інші є логічними або фізичними. Концептуальні діаграми містять найменшу кількість деталей і розроблені, щоб показати загальний обсяг моделі. Логічні діаграми містять більше деталей, таких як операційні та транзакційні сутності. Фізичні діаграми є найдетальнішими, вони збільшують технологію, необхідну для створення бази даних.

2. **Ідентифікація всіх об’єктів системи.** Простіше кажучи, сутність - це іменник. Це будь-яка річ, яку можна визначити, наприклад особа, об’єкт, поняття чи подія з відповідними даними, пов’язаними з нею. Сутністю може бути клієнт, студент, продукт, собака, будинок або олівець. Як правило, прямокутники представляють сутності у вашому ERD.

3. **Створення прямокутника для всіх сутностей і назва його.** Назви мають бути чіткими та інтуїтивно зрозумілими.

4. **Визначення відносин між суб’єктами.** Якщо сутності є іменниками, зв’язки є дієсловами. Відносини – це те, як сутності діють одна на одну або пов’язані одна з одною. Наприклад, скажімо, клієнт підписується на безкоштовну пробну версію. Суб’єктами будуть клієнт і безкоштовна пробна версія, а відносини – реєстрація. Як правило, ромби представляють відносини у ERD.

5. **З’єднання всіх відносин лініями або стрілками.** Лінії демонструють, як різні сутності на діаграмі зв’язків пов’язані між собою.

6. **Налаштування діаграми зв’язку суб’єктів відповідно до потреб.** Можна додавати фігури, змінювати кольори, робити наліпки та малювати стрілки.

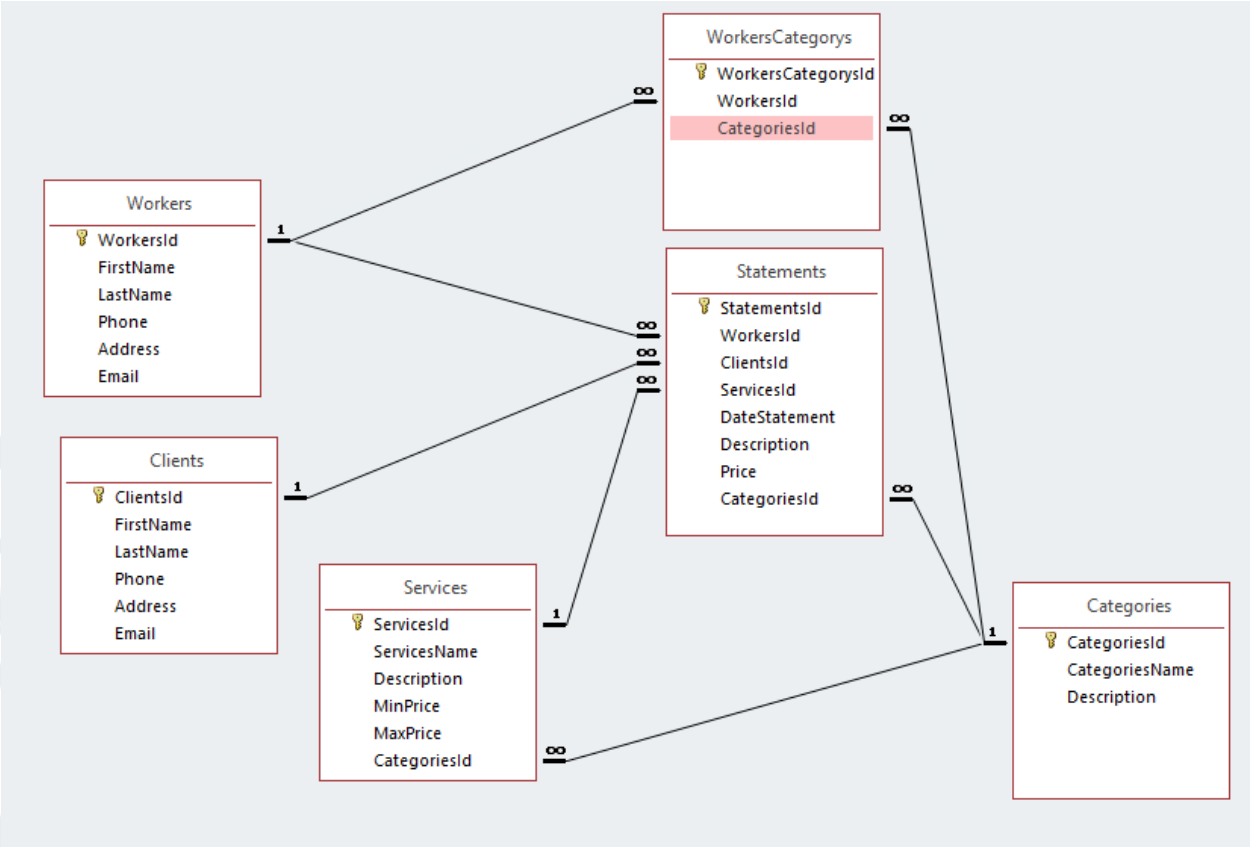


Рис. 2.2 – ERD діаграма

Як ми бачимо на рис. 2.3 наша база даних складається із 6 сутностей. Кожна сутність має свою таблицю, а саме:

1. Таблиця «Clients» – зберігає інформацію про постояльців готелю.
2. Таблиця «Categories» - зберігає інформацію про категорії послуг.
3. Таблиця «Statements» - зберігає інформацію про звернення клієнтів на надання послуг.
4. Таблиця «Services» - зберігає інформацію про послуги, що надаються.
5. Таблиця «Workers» - зберігає інформацію про співробітників.
6. Таблиця «WorkersCategorys» - зберігає інформацію про послуги, що надають співробітники.

2.3 Особливість реалізації бізнес логіки – діаграма домена.

Будуємо діаграму класів домена. Кожен клас описує конкретну таблицю бази даних для зручного опрацювання даних (рис. 2.3).

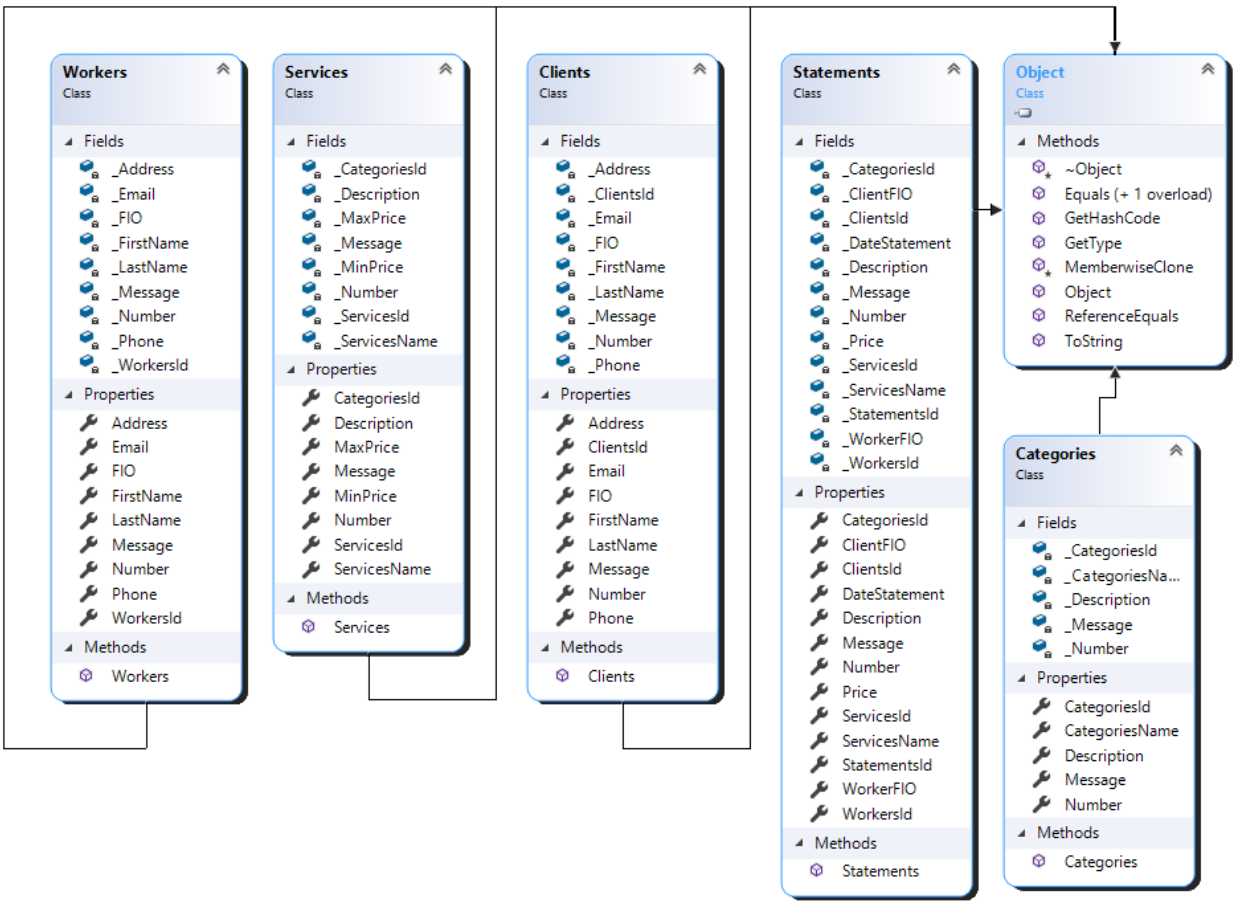


Рис. 2.3 – Діаграма класів домена

Як можна побачити із рис. 2.3, у проекті реалізовано рівно стільки класів, скільки і сутностей у базі даних.

2.4 Особливості розробки рівня UI

User interface (UI) елементи – це частини, які дизайнери використовують для створення програм або веб-сайтів. Вони додають інтерактивність в інтерфейс користувача, надаючи користувачеві точки зіткнення при навігації по них. Кнопки, смуги прокручування, пункти меню та чекбокси.

Інтерфейсу користувача (UI) використовує UI елементи для створення візуальної мови і забезпечення узгодженості продукту, що робить його

зручним для користувача і простим у навігації без особливих зусиль з боку користувача.

UI елементи зазвичай поділяються на одну з наступних чотирьох груп:

– елементи керування введенням (Input Controls) – дозволяють користувачам вводити інформацію до системи.

– компоненти навігації (Navigation Components) – допомагають користувачам переміщатися продуктом або веб-сайтом. Загальні навігаційні компоненти включають панелі вкладок та головне меню програми.

– інформаційні компоненти (Informational Components) – діляться інформацією з користувачем.

– контейнери (Containers) – містять зв’язаний контент разом.

В даному проекті для розробки користувацького інтерфейсу я використав Windows Forms.

Windows Forms - це платформа користувача інтерфейсу для створення класичних додатків Windows. Вона забезпечує один з найефективніших способів створення класичних додатків за допомогою візуального конструктора в Visual Studio.

На рис. 2.4 показана діаграма програми «Додаток по наданню послуг» з правами адміністратора системи.

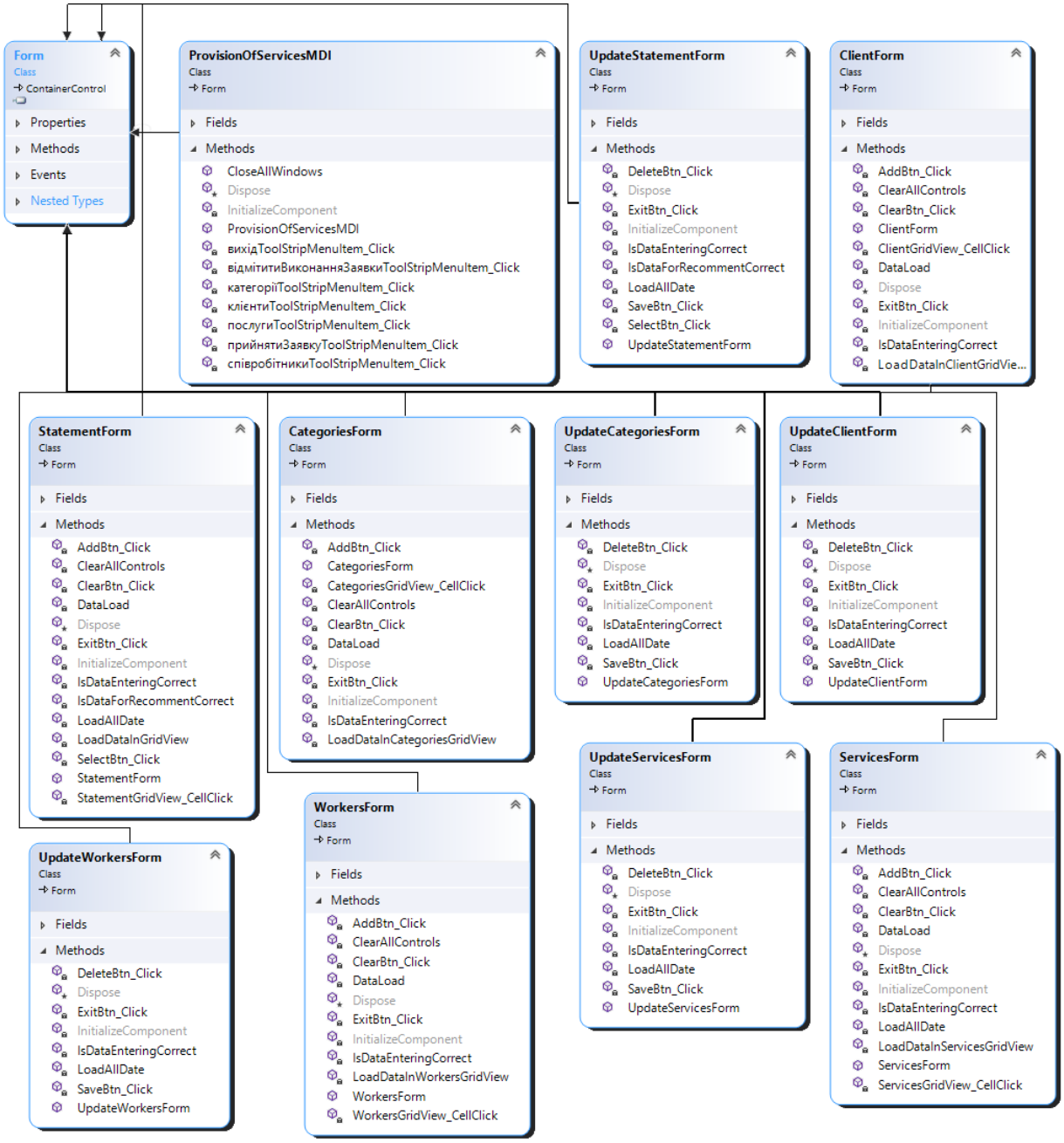


Рис. 2.4 – Діаграма інтерфейсу програми «Додаток по наданню послуг»

Програма складається з дванадцяти класів рівня UI та є похідними від класу Form, тобто мають графічний інтерфейс.

Клас «ProvisionOfServicesMDI» становить головне вікно програми. Найважливішим елементом управління в головному вікні є розташування головного меню, за допомогою якого можна відкривати інші форми та вийти з програми.

Клас «ClientForm» призначений для опрацювання даних про постояльців.

Клас «UpdateClientForm» призначений для редагування інформації про постояльців.

Клас «CategoriesForm» призначений для опрацювання даних категорій послуг.

Клас «UpdateCategoriesForm» призначений для редагування інформації вибраної категорії із списку.

Клас «WorkerForm» призначений для опрацювання про співробітників.

Клас «UpdateWorkerForm» призначений для редагування інформації вибраного із списку співробітника.

Клас «ServicesForm» призначений для опрацювання даних послуг системи.

Клас «UpdateServicesForm» призначений для редагування інформації послуг системи.

Клас «StatementForm» призначений для фіксації звернення клієнтів на надання рекомендаційних послуг.

Клас «UpdateStatementForm» призначений для редагування інформації про звернення клієнтів.

Створення об'єктів інших класів та виклик їх методів відбувається в результаті взаємодії користувача з елементами графічного інтерфейсу програми.

2.5 Висновок

Побудова сучасних інформаційних система займає дуже багато часу. Вона починається з аналізу предметної області, детального планування

системи, описання вимог, моделювання її поведінки за допомогою uml діаграм. Визначається шаблони, які будуть використовуватись при розробці.

Після планування починається стадія розробки. Розробка починається зі створення користувацького інферфейсу і закінчується базою даних.

При розробці слід враховувати, що вимоги змінюються швидко і потрібно будувати систему так, щоб вона була гнучкою.

Розробка системи виконується по окремим компонентам. Кожний створений компонент потрібно детально тестувати, щоб мінімізувати помилки на наступних етапах розробки.

Якщо дотримуватися всіх вищеперерахованих вимог, то можна побудувати гнучку і стійку інформаційну систему.

Розділ 3. Реалізація додатку

3.1 Розробка програмних модулів системи

В даному підрозділі наведений код роботи програми, що був використаний для реалізації проекту. Для зменшення кількості коментарів у програмі всі назви класів, об’єктів у програмі мають зрозумілі назви.

Після постановки задачі, маючи всі необхідні дані, проводимо детальний аналіз роботи. Спочатку засобами СУБД було побудовано модель бази даних.

Сучасні бази даних зберігають великі об’єми інформації, тому обробляти їх вручну переглядаючи та редагуючи дані в таблицях, стає дуже важко важко, тому для підвищення ефективності користування базами даних застосовують запити.

Запит– це засіб отримання інформації з бази даних.

Для написання запитів було використано мову SQL (Structured Query Language).

В базі даних, було розроблено необхідні запити на додавання інформації до таблиць, редагування та видалення. Всі запити були реалізовані безпосередньо із розробленого додатку.

Після того, як базу даних було створено під’єднаємо її за допомогою засобів середовища Visual Studio 2019 для розробки подальшої системи. Усі таблиці пізніше будуть підключатись до форм, на яких можливе додавання, редагування та видалення даних.

Для під’єднання БД до середовища Microsoft Visual Studio 2019 у файлі проекту "App.config" створюємо змінну "CONNECT" та задаємо значення параметрів (лістинг 3.1).

Лістинг 3.1 Параметри змінної "CONNECT"

```
<add key="CONNECT" value="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=|DataDirectory|\DataBase.mdb;"/>
```

Для роботи з створеною базою даних використовуємо простір імен System.Data. OleDb.

Для того, щоб створити меню, додаємо елемент menuStrip у головне вікно проекту (рис. 3.1).

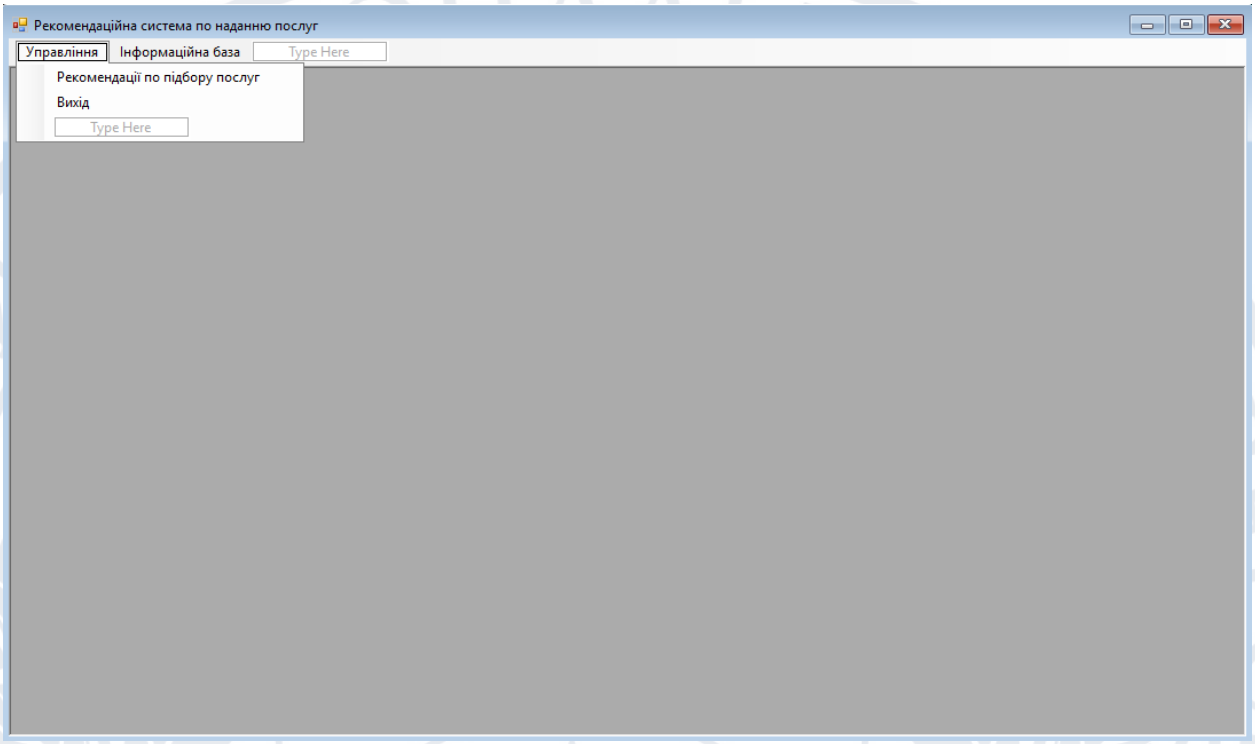


Рис. 3.1 – Головне меню програми

На кожен із пунктів меню додано певний код, який відповідає за створення екземплярів форм та закриття попередньо відкритого вікна. Аналогічний код застосовуємо для всіх пунктів меню (рис. 3.2).

```
1 reference
private void відмітитиВиконанняЗаявкиToolStripMenuItem_Click(object sender, EventArgs e) {
    CloseAllWindows();
    NoteStatementForm noteStatementForm = new NoteStatementForm();
    noteStatementForm.MdiParent = this;
    noteStatementForm.WindowState = FormWindowState.Maximized;
    noteStatementForm.Show();
}
```

Рис. 3.2 – Програмування пунктів меню

Для роботи з базою даних було створено відповідні класи, всі вони розміщені у папці із назвою «Providers» в проекті рішення.

Для з’єднання з базою даних використовується метод «Open» екземпляру класу «OleDbConnection». Для закриття з’єднання з базою даних використовується метод «Close» того ж екземпляру класу.

Для опрацювання даних про постояльців був створений клас «ClientProvider», який містить в собі 5 публічних методів для опрацювання інформації: додавання інформації про нового клієнта, вибірка всіх клієнтів, вибірка конкретного клієнта по його ідентифікатору, редагування вибраного клієнта із списку, видалення інформації про клієнта та ін.

Код методу для додавання нового клієнта представлений на рис. 3.3.

```
public void InsertClient(string LastName, string FirstName, string Phone, string Address, string Email) {
  string SqlString = "INSERT INTO Client (LastName, FirstName, Phone, Address, " +
    "Email) Values(?, ?, ?, ?, ?)";

  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      cmd.Parameters.AddWithValue("LastName", LastName);
      cmd.Parameters.AddWithValue("FirstName", FirstName);
      cmd.Parameters.AddWithValue("Phone", Phone);
      cmd.Parameters.AddWithValue("Address", Address);
      cmd.Parameters.AddWithValue("Email", Email);
      conn.Open();
      cmd.ExecuteNonQuery();
      conn.Close();
    }
  }
}
```

Рис. 3.3 – Метод «InsertClient» для додавання нового клієнта

Код методу для редагування даних клієнта показаний на рис. 3.4.

```
1 reference
public void UpdateClient(string LastName, string FirstName, string Phone, string Address, string Email, int ClientId) {
  string SqlString = "UPDATE Client SET FirstName=?, LastName=?, Phone=?, " +
Address=?, Email=? WHERE ClientId=?";

  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      cmd.Parameters.AddWithValue("FirstName", FirstName);
      cmd.Parameters.AddWithValue("LastName", LastName);
      cmd.Parameters.AddWithValue("Phone", Phone);
      cmd.Parameters.AddWithValue("Address", Address);
      cmd.Parameters.AddWithValue("Email", Email);
      cmd.Parameters.AddWithValue("ClientId", ClientId);
      conn.Open();
      cmd.ExecuteNonQuery();
      conn.Close();
    }
  }
}
```

Рис. 3.4 – Метод «UpdateClient» для редагування інформації про клієнта

Код методу для вибірки всіх клієнтів представлений на рис. 3.5.

```
6 references
public List<Client> GetAllClient() {
    int i = 0;
    string SqlString = "SELECT ClientId, FirstName, LastName, Phone, Address, " +
      "Email FROM Client";

    List<Client> listAllClient = new List<Client>();
    using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
            conn.Open();
            using (OleDbDataReader reader = cmd.ExecuteReader()) {
                while (reader.Read()) {
                    Client oneClient = new Client();
                    oneClient.Number = ++i;
                    oneClient.ClientId = Convert.ToInt32(reader["ClientId"].ToString());
                    oneClient.FirstName = reader["FirstName"].ToString();
                    oneClient.LastName = reader["LastName"].ToString();
                    oneClient.FIO = oneClient.LastName + " " + oneClient.FirstName;
                    oneClient.Phone = reader["Phone"].ToString();
                    oneClient.Address = reader["Address"].ToString();
                    oneClient.Email = reader["Email"].ToString();

                    listAllClient.Add(oneClient);
                }
            }
            conn.Close();
        }
    }

    if (listAllClient.Count == 0) {
        Client noClient = new Client();
        noClient.ClientId = 0;
        noClient.Message = NamesMy.NoDataNames.NoDataInClient;
        listAllClient.Add(noClient);
    }
    return listAllClient;
}
```

Рис. 3.5 – Метод «GetAllClient» для вибірки всіх клієнтів

Код для вибору клієнта із списку показаний на рис. 3.6.

```
public Client SelectedClientByClientId(int ClientId) {
  string SqlString = "SELECT ClientId, FirstName, LastName, Phone, Address, " +
    "Email FROM Client Where ClientId=" + ClientId.ToString();

  Client oneClient = new Client();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          oneClient.ClientId = Convert.ToInt32(reader["ClientId"].ToString());
          oneClient.FirstName = reader["FirstName"].ToString();
          oneClient.LastName = reader["LastName"].ToString();
          oneClient.FIO = oneClient.LastName + " " + oneClient.FirstName;
          oneClient.Phone = reader["Phone"].ToString();
          oneClient.Address = reader["Address"].ToString();
          oneClient.Email = reader["Email"].ToString();
        }
      }
    }
    conn.Close();
  }
  return oneClient;
}
```

Рис. 3:6 – Метод «SelectedClientByClientId» для вибору клієнта із списку

Для видалення даних про клієнта був розроблений метод «DeleteClientByClientId», що показаний на рис. 3.7.

```
1 reference
public void DeleteClientByClientId(int ClientId) {
  string SqlString = "DELETE FROM Client WHERE ClientId=" + ClientId.ToString();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      cmd.ExecuteNonQuery();
      conn.Close();
    }
  }
}
```

Рис. 3:7 – Метод «DeleteClientByClientId» для видалення інформації вибраного із списку клієнта

Наступним кроком було створення форми для надання рекомендаційних послуг (рис. 3.8).

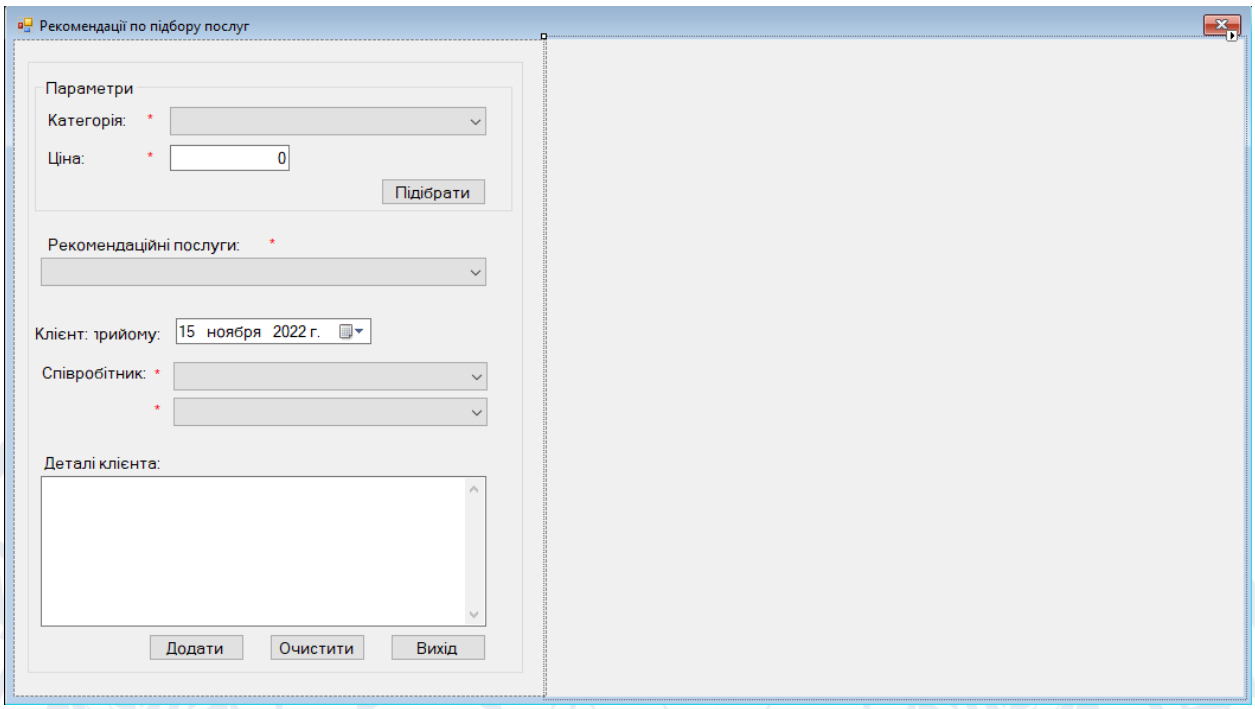


Рис. 3.8 – Форма для надання рекомендаційних послуг

При завантаженні форми про реєстрацію клієнтів у конструкторі форми викликається метод «LoadAllDate», який завантажує дані про клієнтів, співробітників та номери кімнат у випадаючі списки (рис. 3.9).

```
1 reference
private void LoadAllDate() {
    _CategoriesList = _CategoriesProvider.GetAllCategories();
    CategoriesCBox.DataSource = _CategoriesList;
    CategoriesCBox.ValueMember = "CategoriesId";
    CategoriesCBox.DisplayMember = "CategoriesName";

    _ClientList = _ClientProvider.GetAllClients();
    ClientCBox.DataSource = _ClientList;
    ClientCBox.ValueMember = "ClientsId";
    ClientCBox.DisplayMember = "FIO";

    _ServicesList = _ServicesProvider.GetAllServices();
    ServicesCBox.DataSource = _ServicesList;
    ServicesCBox.ValueMember = "ServicesId";
    ServicesCBox.DisplayMember = "ServicesName";

    _WorkerList = _WorkerProvider.GetAllWorkers();
    WorkerCBox.DataSource = _WorkerList;
    WorkerCBox.ValueMember = "WorkersId";
    WorkerCBox.DisplayMember = "FIO";
}
```

Рис. 3.9 – Метод для заповнення даних у випадаючі списки

Для додавання інформацію про нову реєстрацію клієнта у формі вікна реалізовано метод «AddBtn_Click», що спрацьовує на натискання кнопки «Додати» (рис. 3.10).

```
1 reference
private void AddBtn_Click(object sender, EventArgs e) {
    if (IsDataEnteringCorrect()) {
        _StatementProvider.InsertStatements(DescriptionTBox.Text,
          Convert.ToInt32(ClientCBox.SelectedValue),
          Convert.ToInt32(WorkerCBox.SelectedValue),
          Convert.ToInt32(ServicesCBox.SelectedValue),
          Convert.ToDouble(PriceTBox.Text),
          DateStatement.Value,
          Convert.ToInt32(CategoriesCBox.SelectedValue));
        DataLoad();
        ClearAllControls();
    }
}
```

Рис. 3.10 – Метод для додавання інформації про надання рекомендаційних послуг

Як можна побачити, в методі на рис. 3.10 викликається метод «IsDataEnteringCorrect», що призначений для перевірки даних на правильність введення, для подальшого їх зберігання в базі даних. Код методу «IsDataEnteringCorrect» показаний на рис. 3.11.

```
1 reference
private bool IsDataEnteringCorrect() {
    bool isCorrect = true;
    if (Convert.ToInt32(ClientCBox.SelectedValue) > 0) {
        ClientValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
        ClientValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
    }
    if (Convert.ToInt32(WorkerCBox.SelectedValue) > 0) {
        WorkerValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
        WorkerValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
    }
    if (Convert.ToInt32(ServicesCBox.SelectedValue) > 0) {
        ServicesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
        ServicesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
    }
    return isCorrect;
}
```

Рис. 3.11 – Метод для перевірки введених даних

При потребі можна видалити інформацію про реєстрацію. За видалення відповідає подія, яка викликається у формі «UpdateStatementForm», та спрацьовує після натискання лівою кнопкою мишки по кнопці «Видалити». Після вибору запису появляється повідомлення, де необхідно підтвердити видалення (рис. 3.12).

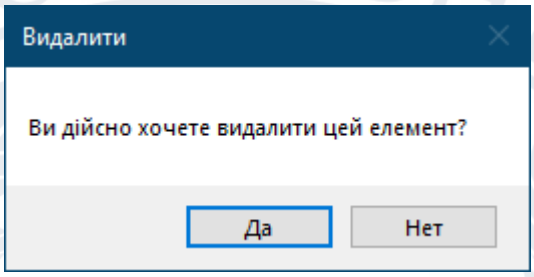


Рис. 3.12 – Підтвердження видалення

Код події на видалення показаний на рис. 3.13.

```
1 reference
private void SaveBtn_Click(object sender, EventArgs e) {
    if (IsDataEnteringCorrect()) {
        _StatementProvider.UpdateStatements(DescriptionTBox.Text, Convert.ToInt32(ClientCBox.SelectedValue), Convert.ToInt32(WorkerCBox.SelectedValue),
            Convert.ToInt32(ServicesCBox.SelectedValue), DateStatement.Value, _StatementId);
        this.Close();
    }
}
```

Рис. 3.13 – Подія «DeleteBtn_Click» для видалення звернення

Отже, в даному розділі було проведено частковий опис основним моментів функціонування розробленого програмного забезпечення.

3.2 Результати функціонального тестування розробленого додатку

Тестування — це процес аналізу та дослідження, який надає змогу виявити інформацію про якість продукту відносно умов, в яких він буде застосовуватись. Методика тестування також включає в себе процес пошуку дефектів, помилок, несправностей. Також це є випробуванням програмних складових з метою оцінити готовність програмного продукту до використання. Результат тестування оцінюється за наступними критеріями:

— відповідність вимогам, які надавалися розробниками та проектувальниками;

— відповідність вихідних даних;

— прийнятний час виконання функцій;

— практичність;

— відповідність вимогам замовника.

Кількість тестів навіть для простих програмних компонентів може бути ледь не нескінченним, тому тактика тестування має полягати в тому, що будуть проведені тільки необхідні тести з урахуванням доступного часу та ресурсів. Як результат, програмні засоби тестуються стандартним виконанням програми з метою виявлення багів, помилок або інших дефектів.

Існує багато видів тестування: одні зазвичай виконують самі розробники, а інші — спеціалізовані групи. В нашому ж випадку буде використовуватись тестування системи.

Тестування системи — це виконання програмного забезпечення в його остаточної конфігурації, інтегрованого з іншими програмними та апаратними системами.

Одним із способів вивчення поставленого питання є дослідження методики «чорної скриньки», Основна роль тестування методів «чорної скриньки» – це інтерфейс програмного забезпечення. Ці тести демонструють:

— як будуть виконуватись функції програми;

— як будуть прийматися вихідні дані;

— як будуть формуватись результати;

— як буде збережена цілісність зовнішньої інформації.

Тестування програмних засобів буде доречно проводити використовуючи методику «чорної скриньки».

Вона базується на використанні шаблонів тестування або ж тест-кейсів. Це означає, що буде створено декілька ситуацій у яких перевіряється працездатність додатку, коректності основних функцій.

Таблиця 3.1 – Тест-кейси

<table>
<tr><td rowspan="3">Код тест-кейса</td><td colspan="3">Опис тест-кейса</td></tr>
<tr><td colspan="2">Хід тестування</td><td>Очікуваний результат</td></tr>
<tr><td>Дата тестування</td><td>Результат</td><td>Примітка</td></tr>
<tr><td rowspan="3">001</td><td colspan="3">Перевірка запуску додатку</td></tr>
<tr><td colspan="2">1. Запустити додаток.</td><td>Вивід відповідного вікна із функціоналом для роботи користувача</td></tr>
<tr><td>19.10.2022</td><td>Пройдено</td><td>–</td></tr>
<tr><td rowspan="3">002</td><td colspan="3">Додавання інформації про нового клієнта</td></tr>
<tr><td colspan="2">1. Запустити додаток;
2. Додати клієнта;
3. Редагувати дані клієнта;
4. Видалити клієнта.</td><td>Список клієнтів готелю відображається відповідно до введених користувачем даних</td></tr>
<tr><td>19.10.2022</td><td>Пройдено</td><td>–</td></tr>
<tr><td rowspan="3">003</td><td colspan="3">Надання рекомендаційних послуг</td></tr>
<tr><td colspan="2">1. Запустити додаток;
2. Додати інформацію про надання послуги;
3. Вибір послуги із рекомендованих;
4. Додавання інформації про надання послуги.</td><td>Список рекомендованих послуг відображається коректно в залежності від наданих даних клієнта. Вибрана послуга для клієнта зберігається та виводиться на екран</td></tr>
<tr><td>19.10.2022</td><td>Пройдено</td><td>–</td></tr>
</table>

3.3 Інструкція користувачеві програми

3.3.1 Опис процедури розгортання програмного продукту, створеного на платформі .NET

Важливим фактором, який необхідно врахувати при розробці програмного забезпечення є потреба в ресурсах.

Мінімальні вимоги до апаратних засобів для запуску та функціонування розробленого програмного забезпечення:

– Процесор Pentium IV/Xeon2.4Ггц.
– Оперативна пам'ять: 1024 Мб вище.
– Вільний дисковий простір менше 120Мб.

Вимоги до програмних засобів:

– Операційна система сімейства Windows: починаючи з Windows XP – до Windows 10.
– Наявність бібліотеки класів .NET framework 4.7 або вище.

Розгортання програмного продукту на комп'ютері користувача у вигляді автономного застосунку:

1. Створіть на цільовому диску каталог для застосунку, наприклад «Система управління готелем»;
2. Скопіюйте каталог «Debug» із проекту.
3. Для перевірки коректності запуску програми виконайте подвійне клацання лівою кнопкою миші на файлі «ProvisionOfServices.exe» (операційна система Windows).

3.3.2 Використання програмного продукту

Запуск програми

Запуск програми в операційній системі сімейства Windows здійснюється одним з стандартних способів:

а) подвійним клацанням лівою кнопкою миші на ярлику програми;

б) викликом контекстного меню з вибором його пункту "Відкрити";

в) натисканням кнопки "Пуск" панелі завдань із подальшим вибором пункту "Усі програми" та подвійним клацанням лівою кнопкою миші на ярлику програми.

Вхід користувача в систему

Після запуску додатка користувача системи буде відкрите головне вікно програми з основним меню (рис.3.14).

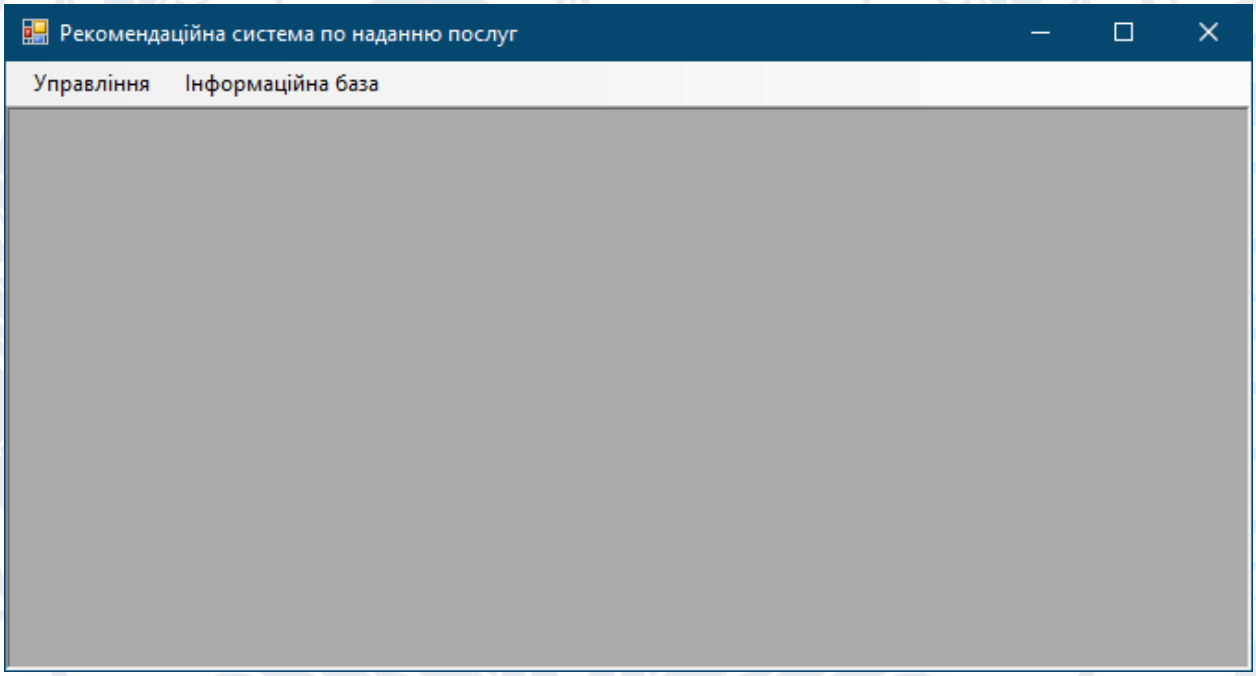


Рис. 3.14 – Головне меню програми

Для реєстрації звернень клієнтів на надання послуг необхідно наповнити інформаційну систему, а саме: інформацію про співробітників, клієнтів, послуги та рекомендації.

Програма дозволяє також редагувати та видаляти введені дані про співробітників, клієнтів, послуги та рекомендації.

Для початку потрібно додати інформацію про постояльців. Для цього необхідно перейти по меню програми «Інформаційна база» –> «Клієнти», у відповідному вікні (рис. 3.15) ввести всі необхідні дані про клієнта та

натиснути кнопку «Додати». Новий постоялець з’явиться у правій частині екрану у списку всіх клієнтів.

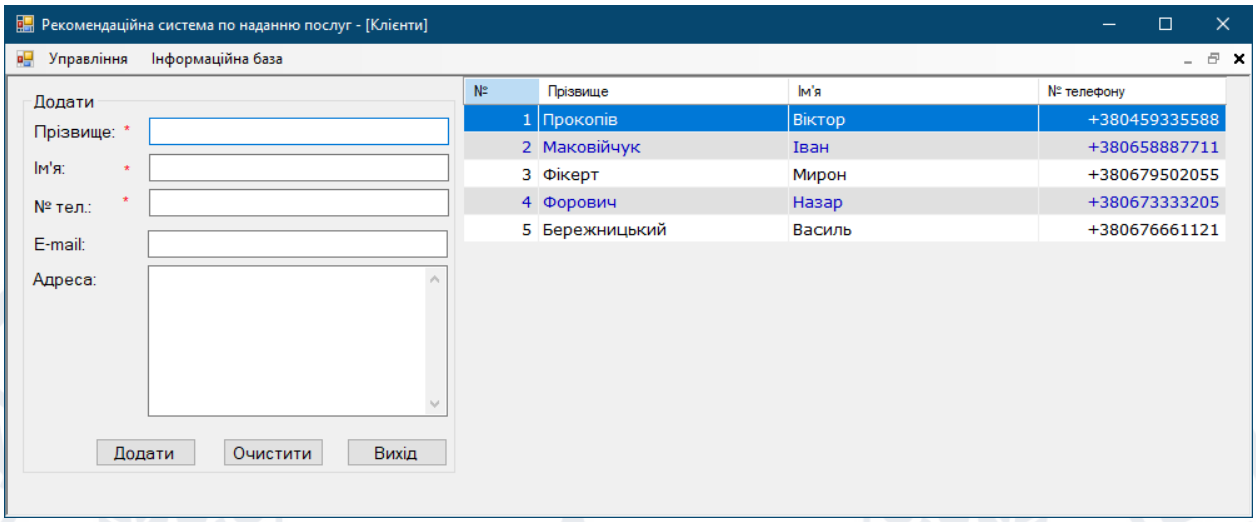


Рис. 3.15 – Вікно для опрацювання даних про клієнтів

Також реалізована можливість редагування та видалення даних у випадку якщо це є необхідним. На рис 3.16 зображено вікно для редагування даних про клієнта.

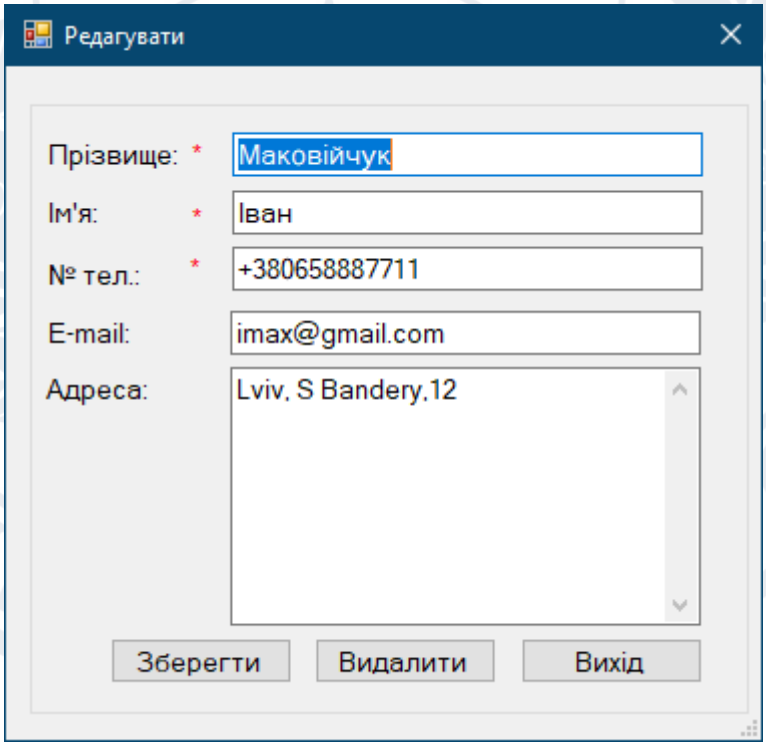


Рис. 3.16 – Вікно для редагування даних

Далі необхідно заповнити базу співробітників, вікно для опрацювання даних про співробітників показано на рис. 3.17.

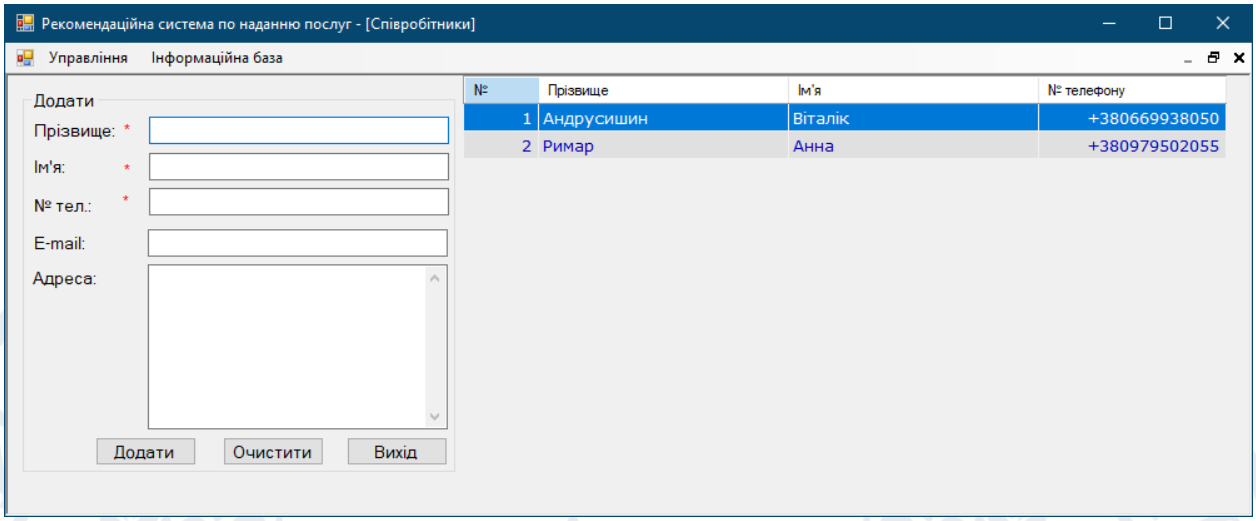


Рис. 3.17 – Вікно для опрацювання даних про співробітників

Дані про співробітників можна відредагувати, вибравши у правій частині необхідного співробітника натисканням лівої кнопки мишки. Після цього з’явиться вікно для редагування даних вибраного співробітника колл-центру (рис. 3.17).

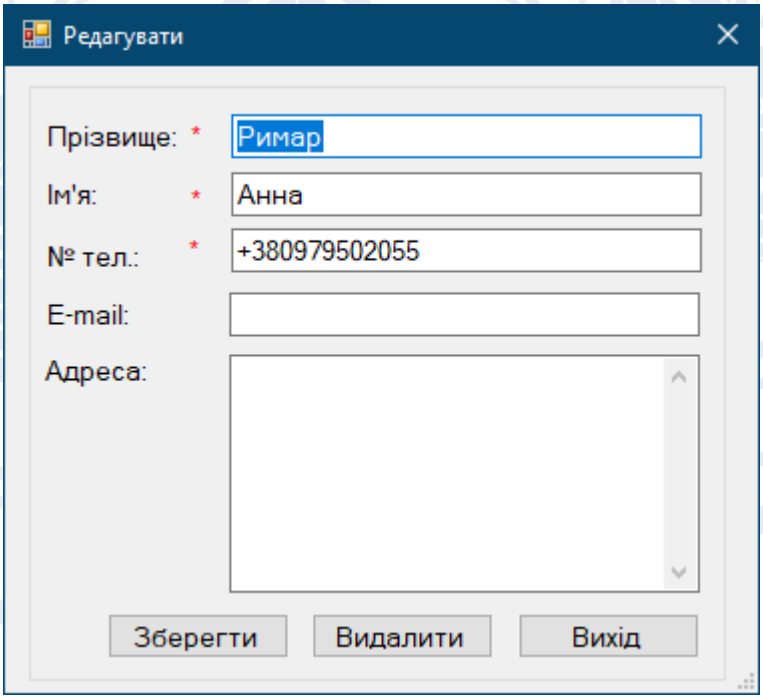


Рис. 3.17 – Вікно для редагування даних вибраного співробітника

Також у програмі є функція додавання послуг. Для того, щоб додати нову послугу, користувачу програми необхідно перейти до пункту меню «Інформаційна база» –> «Послуги», після чого з’явиться відповідне вікно (рис. 3.18).

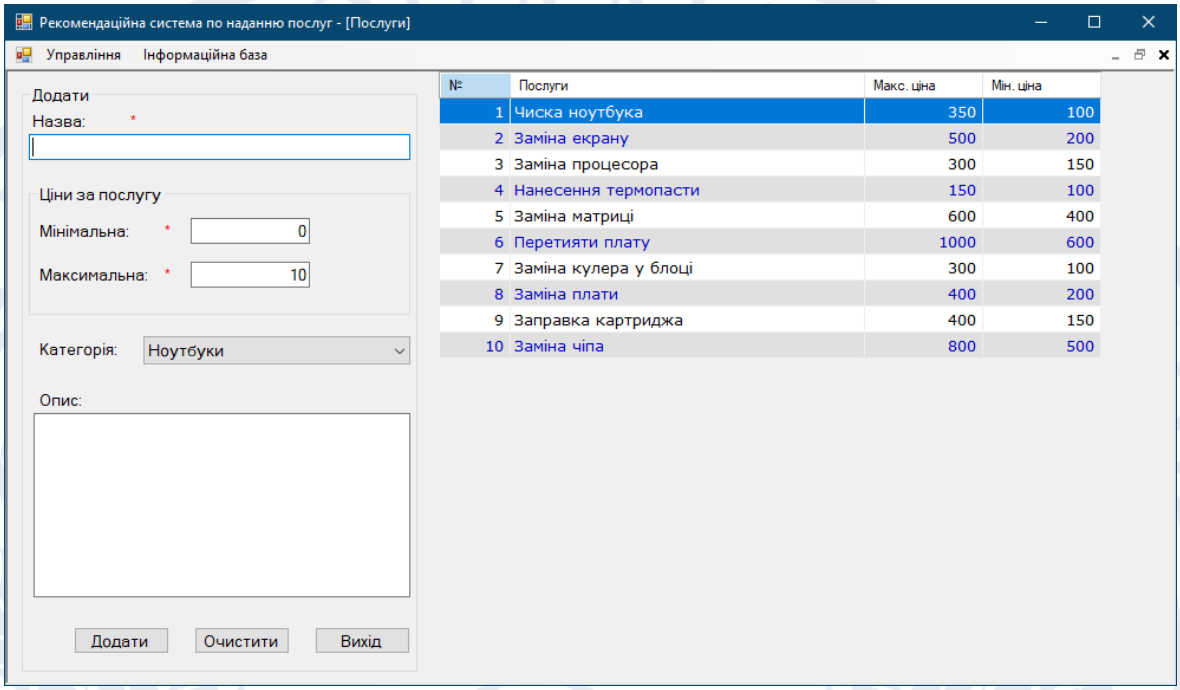


Рис. 3.18 – Вікно для опрацювання даних про послуги

Дані про послуги також можна редагувати, для цього необхідно вибрати у правій частині відповідну послугу, після чого зявиться вікно, що показане на рис. 3.19.

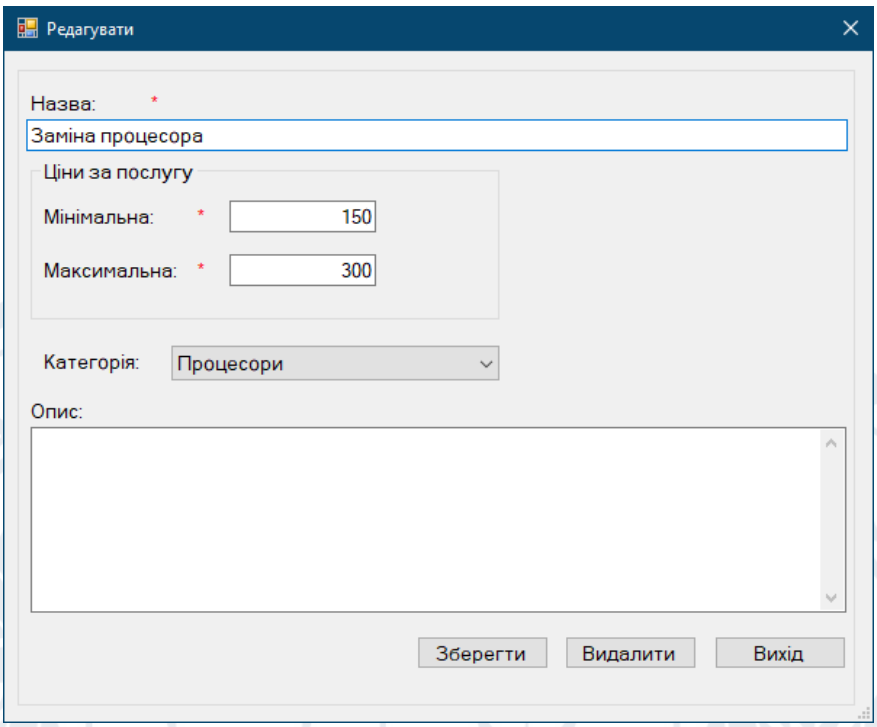


Рис. 3.19 – Вікно для редагування даних послуги

Для підбору рекомендаційних послуг для клієнтів необхідно перейти по меню програми «Управління» –> «Рекомендації по підбору послуг», після чого відкриється вінкно, що представлено на рис. 3.20. Після натискання кнопки підібрати, система підбере необхідні послуги для клієнта та всіх спеціалістів, що можуть їх надати.

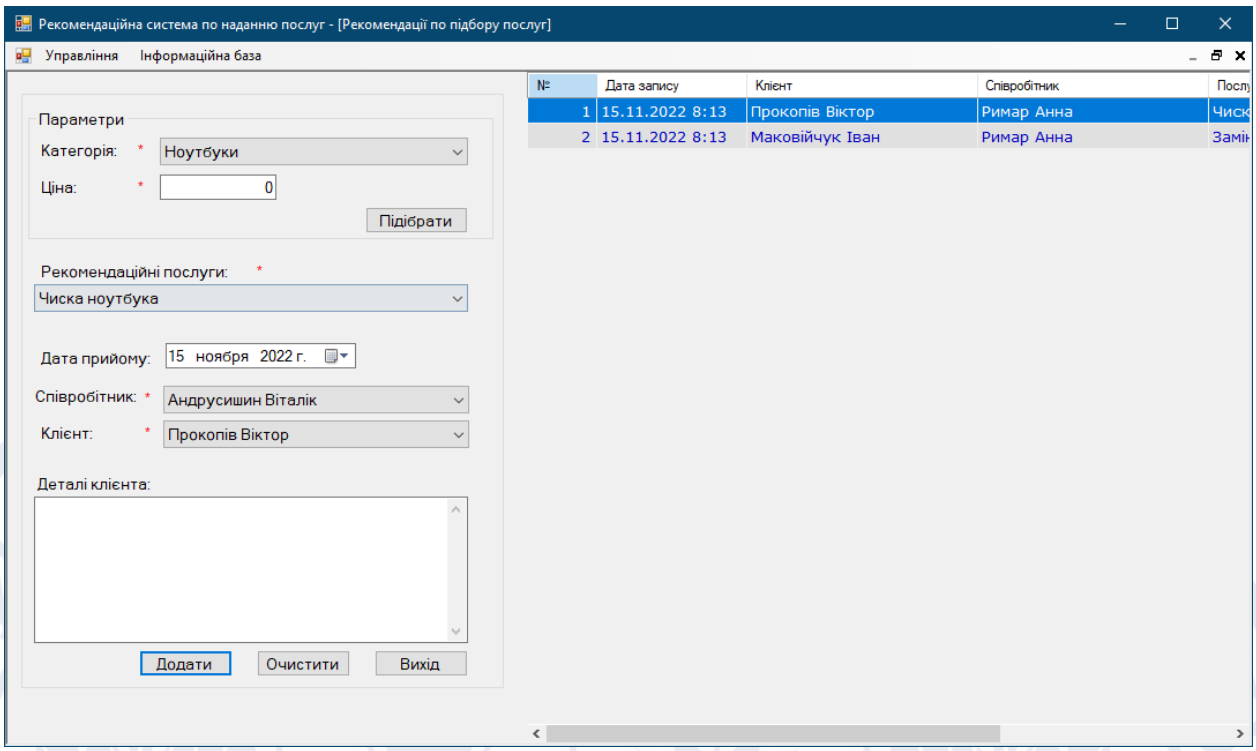


Рис. 3.20 – Вікно для підбору рекомендованих послуг та спеціалістів, що їх надають

Також, у програмі реалізована можливість редагування збереженої послуги, якщо в цьому виникне потреба (рис. 3.21).

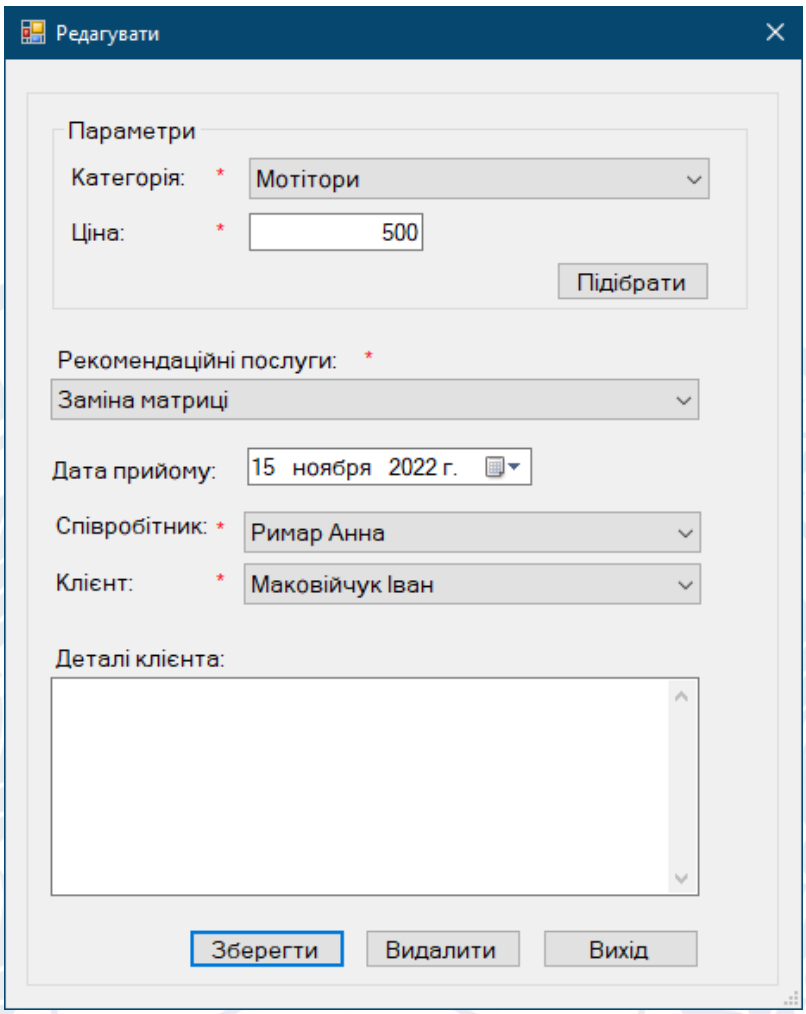


Рис. 3.21 – Редагування послуги

Для виходу із програми необхідно перейти по меню «Управління» –> «Вихід».

Треба сказати, що дана система є не сильно функціональною, але вона є досить простою в користуванні. ЇЇ інтерфейс є інтуїтивно зрозумілим для користувача.

Тестування програми успішно проведено та не було виявлено жодних помилок системи.

3.4 Висновок

В ході виконання роботи мовою C# в середовищі Visual Studio 2019 реалізовано додаток для служби підтримки клієнтів інтернет-сервісу.

Реалізовано такі функціональні вимоги:

- можливість додавати, редагувати та видаляти дані про клієнтів;
- можливість додавати, редагувати та видаляти дані про співробітників;
- можливість додавати, редагувати та видаляти дані про послуги;
- можливість додавати, редагувати та видаляти дані про рекомендації послуг;
- можливість подбору послуг для клієнтів в залежності від введених даних.

ВИСНОВКИ

Даний проект розроблявся для рекомендаційної системи по наданню послуг клієнтам. Розробку системи було виконано у середовищі Microsoft Visual Studio 2019 при використанні мови програмування C # та СУБД MS Access. Дана інформаційна система повинна значно полегшити роботу менеджерів сервісів по наданню послуг, а саме тим, тим, що розроблене ПЗ має зручний перегляд даних, додавання та вилучення записів.

У розробленому продукту реалізовано такі функціональні можливості:

- можливість додавати, редагувати та видаляти дані про клієнтів;
- можливість додавати, редагувати та видаляти дані про співробітників;
- можливість додавати, редагувати та видаляти дані про послуги;
- можливість додавати, редагувати та видаляти дані про рекомендації послуг;
- можливість подбору послуг для клієнтів в залежності від введених даних.

У першому розділі було проведено аналіз предметної області.

У другій частині роботи розглянуто дослідження системи та описано її архітектуру. Зроблено аналіз вимог до програмного забезпечення та побудовано use-case діаграми основних прецедентів.

У третій частині було здійснено розробку програмного забезпечення та проведено тестування. Також, було розроблено інструкцію користувача програми. Результати тестування були успішними, помилок не було виявлено.

В результаті проведеної роботи вирішено актуальне технічне завдання для рекомендаційної системи по наданню послуг клієнтам. У ході виконання роботи отримано такі основні наукові та практичні результати:

1. Розроблено гнучку систему, призначену для надання послуг клієнтам.

2. Розроблені методи в даній роботі можуть бути використані для широкого класу завдань, тому можливий подальший розвиток розробленого програмного забезпечення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. SOLID: The First 5 Principles of Object Oriented Design: [Електронний ресурс] – Режим доступу: https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

2. Побудова ER-діаграм : [Електронний ресурс] // Режим доступу: https://app.diagrams.net/.

3. Литвинов О.А., Карпенко Н.В. Тестування інформаційних систем: модульне, інтеграційне, системне [Текст] – Д.: Ліра, 2016. – 283 с.

4. Литвинов О.А., Герасимов В.В., Карпенко Н.В. Об’єктно-орієнтована розробка інформаційних систем [Текст] – Д.: Ліра, 2018. – 448 с.

5. Програма для роботи з базами даних Microsoft Access: [Електронний ресурс] // Режим доступу: https://www.microsoft.com/uk-ua/microsoft-365/access.

6. Джон Скит, C# для професіоналів: тонкощі програмування, 608 ст. ISBN 978-5-8459-1909-0, 978-1-617-29134-0; 2014, Вільямс.

7. Мова програмування C# і платформа .NET Core [Електронний ресурс] Режим доступу: https://metanit.com/sharp/tutorial/1.1.php

8. What's new in .NET 4.7 [Електронний ресурс] Режим доступу: https://docs.microsoft.com/en-us/dotnet/core/dotnet-five.

Шевчук Юрій Анатолійович
Прізвище, ім'я по батькові

Факультет інформаційних та прикладних технологій
Факультет

122 Комп'ютерні науки
Шифр і назва спеціальності

Комп'ютерні технології обробки даних
Освітня програма

ДЕКЛАРАЦІЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (магістерська) робота на тему «Розробка і дослідження рекомендаційної системи для сфери надання послуг» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувалась іншими особами, а також дані та інформація не отримувалась у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____ _____

(дата) (підпис здобувача освіти)