**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

01.11 - МР.2224 ”С” 2023.12.07. 005 ПЗ

**КОСТРУБІЦЬКИЙ ІГОР МИКОЛАЙОВИЧ**

**2024 р.**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Механіко – технологічний факультет**

**УДК 656.071.1658.91:629.33**

**ПОГОДЖЕНО**
**Декан механіко - технологічного факультету**

_____ Вячеслав БРАТІШКО
(підпис) (ПІБ)

«____» _____ 2024 р.

**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
**Завідувач кафедри**
_____технічного сервісу та інженерного__
(назва кафедри)
____менеджменту імені М.П. Момотенка_

_____ Іван РОГОВСЬКИЙ
(підпис) (ПІБ)

«____» _____ 2024 р.

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему** Удосконалення інженерного менеджменту оренди автомобілів за_____ _____уподобаннями клієнтів сервісу_____

Спеціальність 274 «Автомобільний транспорт»_____
(код і назва)

Освітня програма «Автомобільний транспорт»_____
(назва)

Орієнтація освітньої програми освітньо-професійна_____
(освітньо-професійна, або освітньо-наукова)

Гарант освітньої програми
___доктор технічних наук, професор _____ Войтюк Валерій Дмитрович
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи
_____к.т.н., доцент _____ _____ Тітова Людмила Леонідівна
(науковий ступінь та вчене звання) (підпис) (ПІБ)

**Виконав** _____ Коструб'яцький Ігор Миколайович
(підпис) (ПІБ)

**КИЇВ – 2024**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Механіко – технологічний факультет**

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри** технічного сервісу та інженерного менеджменту імені М.П. Момотенка

д.т.н., проф. _____ Іван РОГОВСЬКИЙ
(науковий ступінь, вчене звання) (підпис) (ПІБ)

«____» _____ 2024 р.

**З А В Д А Н Н Я**
**ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ**
Коструб'ицькому Ігорю Миколайовичу
(прізвище, ім'я, по батькові)

Спеціальність 274 «Автомобільний транспорт»
(код і назва)

Освітня програма «Автомобільний транспорт»
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна, або освітньо-наукова)

Тема магістерської кваліфікаційної роботи Удосконалення інженерного менеджменту оренди автомобілів за уподобаннями клієнтів сервісу

затверджена наказом ректора НУБіП України від «07» грудня 2023 р. № 2224 «С»
Термін подання завершеної роботи на кафедру
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи Науково – технічна література; результати науково-дослідних робіт по літературних джерелах по вивченню питання інженерного менеджменту оренди автомобілів за уподобаннями клієнтів сервісу

Перелік питань, що підлягають дослідженню:
1. Розділ 1 Аналіз предметної області і постановка задачі
2. Розділ 2 Розробка методу вилучення знань уподобань клієнтів
3 Розділ 3 Проектування моделі даних для представлення вхідної інформації методу вилучення знань уподобань клієнтів
4. Розділ 4 Розробка програмної реалізації методу вилучення знань уподобань клієнтів

Перелік графічного матеріалу Електронна презентація на 18 слайдах

**Дата видачі завдання** «10» листопада 2023 р.

**Керівник магістерської кваліфікаційної роботи** _____ Тітова Л.Л.
(підпис) (прізвище та ініціали)

**Завдання прийняв до виконання** _____ Коструб'ицький І.М.
(підпис) (прізвище та ініціали)

# РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи містить: 77 сторінок, 51 рисунка, 7 таблиці, 50 джерела. Графічна частина магістерської кваліфікаційної роботи розміщується в презентації на 18 слайдах.

**Мета магістерської кваліфікаційної роботи** - дослідження та розробка методу вилучення знань для визначення уподобань клієнтів сервісу оренди автомобілів та аналіз методів вирішення проблеми визначення рекомендацій.

**Предметом досліджування** є метод, що є деяким варіантом рекомендаційної системи, яка дозволить клієнтам підбирати автомобілі для прокату, а адміністратору прокату автомобілів - зробити сервіс більш зручним, а також збільшити прибуток за допомогою кращого підбору засобів пересування для клієнтів.

**Об’єкт досліджування -** оренда автомобілів, а також його програмна реалізація.

Результатом магістерської кваліфікаційної роботи є розроблений метод вилучення знань уподобань клієнтів сервісу оренди автомобілів, а також його програмна реалізація.

Область застосування системи - широке коло користувачів, система орієнтована на в основному на комерційне використання для готового парку автомобілів та сервісів прокату.

ПРОКАТ АВТОМОБІЛІВ, ОРЕНДА, РЕКОМЕНДАЦІЙНА СИСТЕМА, ПЕРСОНАЛІЗАЦІЯ.

**ЗМІСТ**

# ВСТУП

Автомобільна промисловість активно розвивається кожен день, експерти пояснюють подібне збільшення парку насамперед зростанням ринків країн, що розвиваються. Проте, для багатьох автомобіль досі залишається розкішшю, або володіння автомобілем є неможливим або незручним для користивувача.

У цих випадках має місце прокат (або оренда) автомобілів. Прокат транспортних засобів має масу переваг. Одним із них є відсутність необхідності в обслуговуванні та ремонті (ці завдання здійснює прокатний пункт), доступна вартість послуги, обумовлена високою конкуренцією, наявність страхових полісів, сучасне оснащення автомобілів навігаційними системами, дитячими кріслами, кондиціонерами, аудіотехнікою. Тому онлайн-сервіси прокату автомобілів є актуальними сьогодні, такі сервіси дозволяють обрати модель автомобілю та оформити замовлення через інтернет.

Проте, варіантів моделей автомобілів зазвичай пропонується багато, а кожен клієнт має свої власні уподобання. Через це з’являється проблема інформаційного перевантаження користувачів. Звісно, у наші дні в епоху світової глобалізації практично кожна людина має доступ до інформації. Саме тому прийняття важливих рішень - частина повсякденного життя, але з ростом числа варіантів і складності вибору люди можуть вдаватися до допомоги різних сервісів, щоб спростити цей процес.

Рекомендаційні системи - приклад таких сервісів. Вони використовуються для вирішення різних завдань, широко поширені в даний час, і області їх застосування все ще розширюються. На даний момент існує багато алгоритмів розробки рекомендаційних систем, які можуть дозволити вилучати знання для визначення того, які моделі автомобілів необхідно рекомендувати користувачам сервісу прокату. Ці знання будуть корисні і для клієнтів, і для адміністраторів сервісу.

Метою даної магістерської кваліфікаційної роботи є розробка методу вилучення знань для визначення уподобань клієнтів сервісу прокату автомобілів,

який буде використовувати існуючі алгоритми побудування рекомендаційних систем.

В результаті виконання атестаційної роботи був досліджений та розроблений метод вилучення знань уподобань для клієнтів сервісу прокату автомобілів. Даний метод дозволяє користувачам швидко підбирати автомобіль для замовлення за допомогою аналізу даних профілю та попередніх замовлень.

# РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКИ ЗАВДАННЯ

Більшість інтернет сайтів прокату автомобілів обладнані функцією показу своїм користувачам персональних рекомендацій щодо оренди засобів пересування. Рекомендаційні системи прогнозують перевагу автомобіля для конкретного користувача, засновуючи свій прогноз на даних, зазначених користувачем явно, або зібраних з історії його взаємодії з даним сайтом або з різними іншими ресурсами.

Рекомендаційні системи застосовуються в багатьох комерційних проектах для визначення уподобань клієнтів на основі їх дій та даних профілю: Netflix (функція Shuffle Play пропонує користувачам відповідні фільми і серіали), Amazon (використовує кілька видів рекомендаційних систем і одне з застосувань - додаткові пропозиції у вигляді схожих або супутніх товарів), Last.fm (використовує рекомендаційний механізм на основі тегів). За останні роки якість алгоритмів цього типу значно збільшилася. Тому проблема розробки та дослідження методів побудування рекомендаційних систем є актуальною на сьогоднішній день [1].

## 1.1. Аналіз діяльності підприємства

Предметною областю є підприємство, спеціалізація якого – прокат автомобілів (або оренда автомобілів).

Процедура оренди автомобіля формалізована. Клієнт повинен бути дорослішe 21 року. Він повинен пред'явити паспорт і міжнародне водійське посвідчення працівнику сервісу прокату. До моменту укладення договору посвідчення має бути не менше 2 років [2].

Між підприємством та клієнтом складається договір. У договорі обов'язковомають бути вказані такі дані:

- інформація про марку автомобіля;
- державний реєстраційний номер;

- номер двигуна;
- умови, терміни і порядок внесення плати за оренду;
- права орендодавця і орендаря;
- інформація про страхування;
- територія використання автомобіля;
- інформація про обмеження пробігу;
- наявність додаткових послуг;
- відповідальність за неналежне виконання умов оренди;
- умови продовження оренди;
- інша інформація.

Додатково до договору оформляється акт прийому-передачі. Там вказуються характеристики переданого в оренду автомобіля, його пробіг, вартість, а також технічний стан. У цьому акті фіксуються недоліки, наявні в машині на момент її здачі в оренду [2].

Орендодавець перед укладенням договору повинен ознайомити орендаря з правилами використання автомобіля. Крім того, він зобов'язаний у присутності замовника послуги перевірити справність машини.

Капітальний і поточний ремонт автомобіля повинен здійснюватися орендодавцем. Це повинно бути відображено в договорі.

Є декілька сервісів підприємства, в яких можна подивитися та забрати автомобілі. Також є гаряча лінія, за допомогою якої можна отримати консультацію щодо автомобілів, особливостей договору та інше.

Сервіс може пред’явити водія, якщо це потрібно, за додаткові кошти.

### 1.2. Аналіз проблеми інформаційного перевантаження користувачів

Проблема інформаційного перевантаження користувачів інтернет-сервісів прокату автомобілів заважає розвитку бізнесу через нездатність клієнтів самостійно розібратися з вибором автомобілів.

Кількість даних в інформаційному просторі прокату автомобілів зростає набагато швидше, ніж здатність окремого користувача або клієнта обробити їх. В

результаті інформаційне перевантаження стало серйозною проблемою онлайн-сервісів. Найчастіше клієнти звертаються в підтримку за консультацією з приводу вибору засобу пересування, проте не всі онлайн-сервіси прокату готові забезпечити достатню кількість операторів лінії підтримки для задоволення попиту клієнтів. Відповідно, клієнти вибирають менше навантажені онлайн-сервіси з інтегрованими методами отримання знань для визначення переваг клієнтів, якими в більшості випадків є різні види рекомендаційних систем. Рекомендаційні системи можуть підібрати користувачеві автомобіль виходячи з даних його профілю, наприклад, вік, попередні замовлення, стать і так далі.

Тому рішення проблеми інформаційного перевантаження користувачів полягає саме в розробці рекомендаційної системи або методу вилучення знань уподобань клієнтів, яка інтегрується з онлайн-сервісом прокату. Повинен бути запропонований алгоритм, що дозволяє підвищити якість рекомендацій за рахунок врахування інтересів рекомендованого і рекомендованого автомобідів, що в свою чергу частково вирішить проблему інформаційного перевантаження користувачів ресурсу.

### 1.3. Актуальність роботи розробки методу вилучення знань

Проблема побудови методів вилучення знань щодо рекомендацій на даний момент актуальна для багатьох областей і є частиною таких завдань, як пропозиція товарів в інтернет-магазинах, ранжирування результатів видачі в пошукових системах, пошук відповідного контенту в музичних, відеосервісах і ЗМІ, а також засобів пересування для сервісу прокату автомобілів.

В цілому рекомендаційні системи в Інтернеті застосовуються з метою персоналізації контенту - його автоматичного підлаштовування під поточні потреби конкретного користувача. Незважаючи на те, що персональні рекомендації в онлайн-системах з'явилися більше 20 років тому, вдалих прикладів їх використання існує лише пара десятків. Дослідження в цій галузі активно ведуться і в наш час, що головним чином обумовлюється і наявністю невирішених проблем в існуючих методах. Одним із шляхів підвищення точності

рекомендаційної системи є розширення переліку використовуваної при формуванні рекомендацій інформації, зокрема використання контексту.

Рекомендаційна система - це програмний комплекс, який визначає інтереси і переваги користувача і формує пропозиції контенту відповідно до них. Такі системи змінили способи взаємодії програмних систем зі своїми користувачами. Замість надання статистичної інформації, система змінюється, підлаштовується під користувача, збільшуючи ступінь інтерактивності для розширення послуг користувачеві можливостей [3].

Традиційна рекомендаційна система має справу з двома видами сутностей: користувач і об'єкт. Тут користувач - це одержувач рекомендації і джерело даних про переваги, а об'єкт - в залежності від предметної області - товар, фільм, музичний трек, книга, новина, вебсайт, тобто те, що пропонується користувачеві в якості рекомендації [4]. У загальному вигляді завдання рекомендаційної системи можна сформулювати як «визначення об'єкта, раніше невідомого користувачеві (або невикористаного ним протягом будь-якого проміжку часу), але корисного чи цікавого йому в поточному контексті». Існують різні підходи до розробки рекомендаційних систем, які можуть застосовуватися в залежності від:

- доступних даних про користувачів і рекомендованих сутності;
- видів явною і неявній зворотного зв'язку від користувачів;
- предметної області [5].

Використовуючи сучасні методи зберігання і аналізу даних, ефективні рекомендаційні системи дозволяють формувати купівельний інтерес клієнтів.

### 1.4. Порівняння аналогів розроблюваного методу вилучення знань

*Car Recommendation System* - у цьому проекті пропонують впровадити веб-сторінку, де користувачі можуть переглядати певні типи предметів, наприклад автомобілі, та давати свої відгуки про них, явно (великі пальці вгору / вниз, оцінки «подобається» та інше) або неявно (клацання на предмет, витрачаючи час на читання його опису, обмін ним та інше). Потім система буде запускати алгоритми, щоб придумати подібні елементи, щоб показати їх користувачеві, і, за

бажанням, збирати відгуки про якість рекомендації. Алгоритми можуть варіюватися від простих показників подібності до більш складних моделей машинного навчання, таких як SVM. Застосовувались різні теми Data Science, такі як отримання інформації, очищення її для обробки та зберігання, оцінка алгоритмів рекомендацій та впровадження конвеєрів обробки великих даних. Іншими словами, система вивчатиме введені користувачем дані протягом часу та рекомендуватиме машини на основі алгоритмів машинного навчання. Таким чином користувач може розглянути перспективні машини, які відповідають його або її очікуванням [6].

*Recombee* (recombee.com) - сервіс з простою інтеграцією та потужним інтерфейсом адміністратора. Цей сервіс використовувався у багатьох доменах, наприклад у засобах масової інформації (VoD, новини та інше), електронній комерції, дошках оголошень, агрегаторах чи оголошеннях. В основному його можна використовувати в будь-якому домені з каталогом предметів, з якими можуть взаємодіяти користувачі. Користувачі можуть взаємодіяти з елементами різними способами: наприклад, переглядати їх, оцінювати, додавати до них закладки, купувати тощо. І елементи, і користувачі можуть мати різні властивості (метадані), які також використовуються в рекомендаційних моделях. [7].

*Amazon Personalize* - дозволяє розробникам створювати застосування з тією ж технологією машинного навчання (ML), яка використовується Amazon.com для персоналізованих рекомендацій у режимі реального часу - не потрібна експертиза ML.

Amazon Personalize полегшує розробникам створення застосувань, здатних забезпечити широкий спектр можливостей персоналізації, включаючи конкретні рекомендації щодо продукту, персоналізований перереєстрацію продуктів та індивідуальний прямий маркетинг. Amazon Personalize - це повністю керована послуга машинного навчання, яка виходить за межі жорстких систем рекомендацій, заснованих на статичних правилах, і навчає, налаштовує та розгортає власні моделі ML, щоб надавати вподобані рекомендації клієнтам у різних галузях, таких як роздрібна торгівля та медіа та розваги.

Amazon Personalize забезпечує необхідну інфраструктуру та управляє всім конвеєром ML, включаючи обробку даних, виявлення особливостей, використання найкращих алгоритмів та навчання, оптимізацію та розміщення моделей. Легко отримати результати за допомогою Інтерфейсу програмування програм (API) і платити лише за те, що використовується, без мінімальних комісій або попередніх зобов’язань. Усі дані зашифровані, щоб бути приватними та захищеними та використовуються лише для створення рекомендацій для ваших користувачів [8].

### 1.5. Опис об’єкта, для якого необхідна рекомендаційна система

Об'єктом, для якого необхідно створити рекомендаційну систему є сайт (онлайн-сервіс) прокату автомобілів.

В онлайн-сервісі прокату існують чотири ролі: незареєстрований користувач, клієнт, адміністратор та менеджер. При цьому менеджер та адміністратор наслідують функції клієнта, який в свою чергу наслідує можливості незареєстрованого користувача.

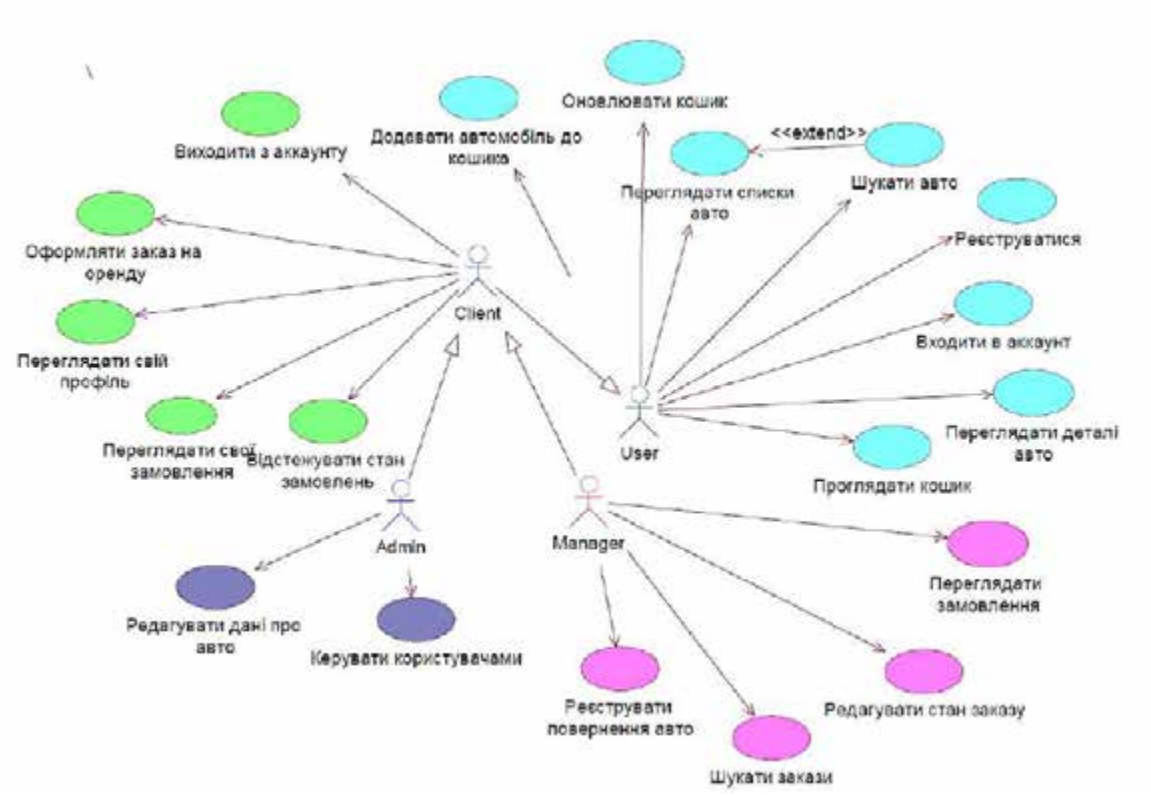



Рис. 1.1. Діаграма варіантів використання для сервісу прокату автомобілів

Можливості незареєстрованого користувача:

- реєстрація на сайті;.
- фільтрація усіх автомобілів  за параметрами;
- перегляд кожного автомобілю докладніше;
- перегляд виробників автомобілів;
- перегляд кожного виробника докладніше;
- здійснювати пошук марки по назві марки.

Можливості клієнта:

- авторизація на сайті;
- блокування свого аккаунту.
- перегляд своїх замовлень;
- перегляд свого профілю;
- оформлення замовлення на прокат.

Можливості адміністратора:

- управління автомобілями (редагування, видалення);
- управління користувачами (блокування, перегляд, зміна ролі).

Можливості менеджера:

- зміна статусу заказів;
- додавання рахунків до заказу;
- пошук всіх замовлень;
- реєстрація повернення автомобілів.

Для створення рекомендаційної системи важливі такі дані як список усіх автомобілів автопарку, профілі клієнтів, а також база усіх замовлень. Важливо,що у кожного замовлення є оцінка від 1 до 5, яку може поставити клієнт. Створенірекомендації будуть корисні не тільки клієнтам, а також менеджерам та адміністраторам сервісу.

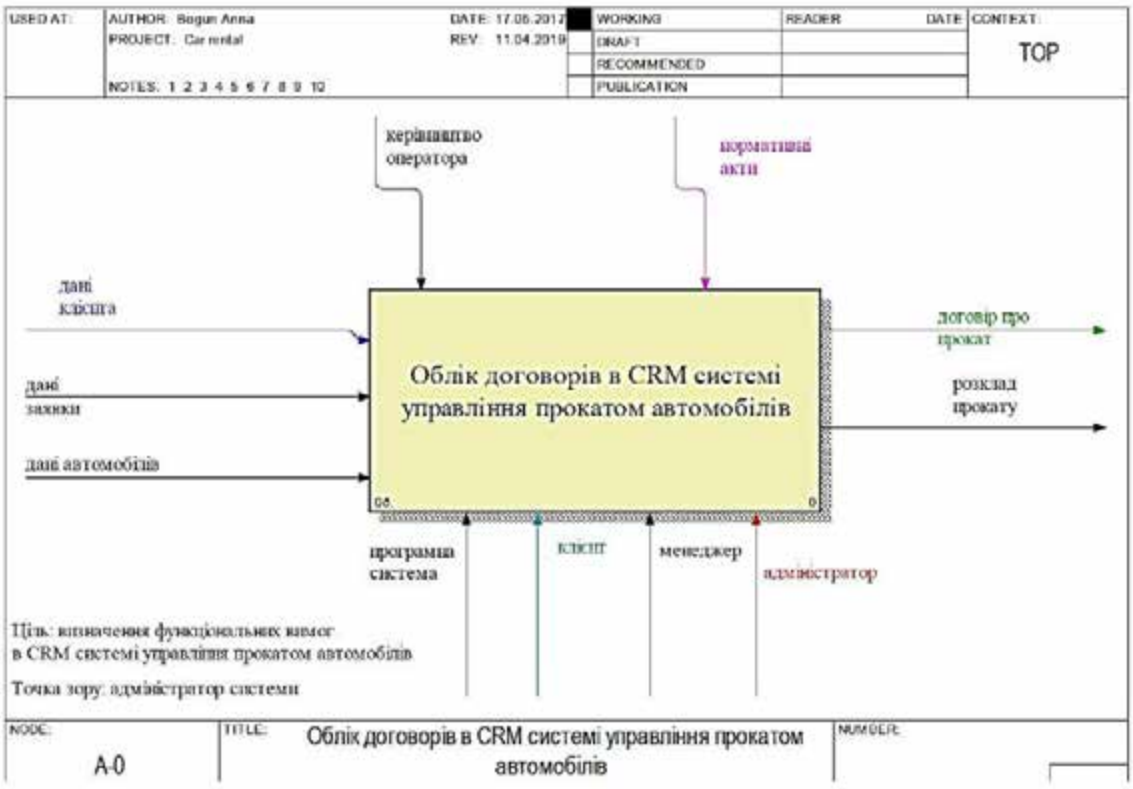



Рис. 1.2. Контексна діаграмма IDEF0 для сервісу прокату автомобілів

### 1.6. Постановка задачі дослідження

Постановка завдання дослідження полягає у аналізі існуючих методів створення рекомендаційних систем з ціллю виявлення найкращих з них для обробки даних онлайн-сервісу прокату автомобілів для створення власного методу вилучення знань.

Таким чином, постановка завдання дослідження зводиться до наступного:

- проведення аналізу методів побудови рекомендаційних систем;
- розробка методу вилучення знань для визначення уподобань клієнтів прокатуавтомобілів на основі проаналізованих методів.

В якості вхідної інформації задача вилучення знань використовує:

- список користувачів у системі з роллю «клієнт» (дані про користувачів системи містять ідентифікаційні дані, а також стать, дату народження);
- список автомобілів доступних для прокату, що можуть бути рекомендовані;

- список попередніх замовлень клієнтів на прокат з оцінками від 1 до 5;
- кількість моделей автомобілів, яка має бути оброблена.

Вихідною інформацією задачі є знання у вигляді продукційної моделі знань. Продукційна модель – це модель, заснована на правилах, що дозволяє уявити знання у вигляді пропозицій типу: «ЯКЩО умова, ТО дія». В нашому випадку умова містить:

- дані моделі автомобілю;
- дані профілю клієнта.

А дією є рішення «рекомендувати» або «не рекомендовути». Формалізований опис задачі можна представити формулою:

$$Z = recommendations\ (p, K, O, C, n)$$
$$p \in P, \qquad (1.1.)$$
$$K \subseteq O$$

де $recommendations$ - функція методу вилучення знань для визначення уподобань клієнтів сервісу прокату автомобілів, який необхідно розробити.

$P$ – множина профілів клієнтів. Кожен елемент множини $P$ є кортежем, який містить атрибути рівнозначні даним профіля клієнта: ідентифікатор клієнта, логін, пароль, email, телефон, ім’я, фамілія, стать, дата народження.

$O$ - множина всіх замовлень клієнтів. Кожен елемент множини $O$ є кортежем, який містить атрибути рівнозначні даним замовлень клієнта: ідентифікатор клієнта, ідентифікатор автомобілю, дата початку прокату, дата завершення прокату, дата замовлення, опис замовлення, статус, номер автомобілю, оцінка замовлення від 1 до 5.

$K$ - підмножина множини замовлень клієнтів $O$, що містить тільки замовлення клієнта p.

$C$ - множина всіх моделей автомобілів, доступних для прокату в сервісі прокату. Кожен елемент множини $C$ є кортежем, який містить атрибути рівнозначні даним опису моделі автомобілю: ідентифікатор моделі автомобілю, марка автомобілю, назва моделі, рік випуску моделі автомобілю, тип кузова,

кількість циліндрів, розмір двигуна, лошадина сила, кількість клапанів, тип приводу, передача, тип, двигуна, тип палива, країна походження, класифікація автомобілів, ціна за оренду на день у доларах США.

$n$ - кількість моделей автомобілів, які мають бути оброблені (рекомендованіабо не рекомендовані) для клієнта p.

$Z$ - множина, що є представленням знань для визначення уподобань клієнтів сервісу прокату автомобілів у вигляді продукційної моделі знань. Кожен елемент множини $Z$ зберігає елементи, що містять список ознак автомобілю та профілю клієнта (умова) та рішення (дія) щодо цієї моделі (рекомендувати або ні).

# РОЗДІЛ 2 РОЗРОБКА МЕТОДУ ВИЛУЧЕННЯ ЗНАНЬ УПОДОБАНЬ КЛІЄНТІВ

## 2.1. Представлення методу вилучення знань як варіанту рекомендаційної системи

Результатом розроблюваного методу є знання, які демонструють уподобання клієнтів сервісу прокату автомобілів, а саме список автомобілів з рішеннями – відповідає автомобіль уподобанням клієнтам чи ні. Ці рішення можна представити як рекомендації - якісь автомобілі будуть рекомендуватись, а якісь ні. Тому розроблюваний метод можна проектувати як деякий варіант рекомендаційної системи.

Системи прогнозування уподобань клієнта - одне з найбільш розповсюджених застосувань інтелектуального аналізу даних та машинного навчання в галузі сервісів підприємств в інтернеті. Такі системи аналізують поведінку користувачів сайтів, після чого може бути прогнозована оцінка припущення користувачем того чи іншого об'єкта рекомендацій. В рамках поставленої задачі об’єктами є моделі автомобілів, а інтернет-сервісом є сервіс прокату автомобілів.

На основі даних користувачів, функціональність сервісу може бути адаптована для кожного конкретного користувача. Таку особливість сервісу прийнято називати персоналізацією.

Рекомендаційні системи допомагають користувачеві орієнтуватися у великому кількісному асортименті, що пропонується сервісом, в випадку обраної предметної області асортимент представляє різноманітну кількість марок автомобілів, які клієнт може взяти в прокат на деякий строк.

Найпопулярніша задача рекомендаційних систем - це передбачення списку релевантного контенту для кожного конкретного користувача, або уподобань клієнтів.

Виключне значення ця задача має для онлайн-прокату, оскільки пропонується автомобіль, який його зацікавив, сервіс прокату може збільшити

свій обсяг продажу.

Ще одне практичне застосування рекомендаційних систем - планування. Наприклад, онлайн-сервіс Amazon використовує прогноз уподобань користувачів, щоб привезти до складу товари, найбільш цікаві для користувачів у цій географічній місцевості ще до фактичного замовлення товарів користувачами, щоб скоротити майбутні логістичні витрати.

Існує декілька методів побудування рекомендаційних систем (рис. 2.1.): колаборативна фільтрація на основі сусідства, колаборативна фільтрація на основі моделі, гібридні моделі, рекомендаційні системи на основі контента (змісту).

Розроблюваний метод вилучення знань має використовувати один чи декілька алгоритмів рекомендаційних систем для того щоб більш якісно спрогнозувати уподобання клієнтів.

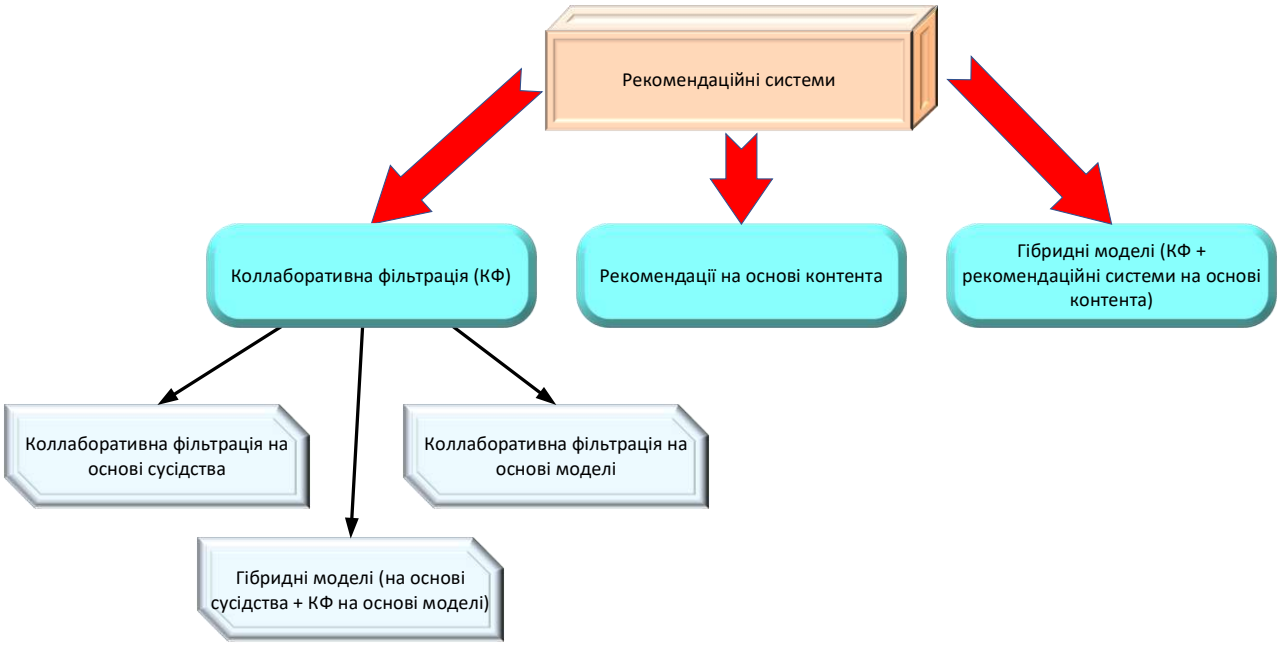



Рис. 2.1. Типи методів побудування рекомендаційних систем

## 2.2. Впровадження в метод вилучення знань алгоритму колаборативної фільтрації

### 2.2.1. Порівняння та вибір методу колаборативної фільтрації для розроблюваного методу

Колаборативна фільтрація - це метод рекомендацій, при якому

аналізується лише реакція користувачів на об'єкти: оцінки, які виставляють користувачі об'єктам. У випадку предметної області будуть аналізуватися оцінки замовлень напрокат автомобілів.

Оцінки можуть бути як явними (клієнт виставляє оцінки замовленням після завершення від 1 до 5, в одному замовленні присутній один автомобіль), так і неявними (в сервісі прокату автомобілів неявних оцінок немає). Чим більше оцінок збирається, тим більше якість прогнозу рекомендацій. Таким чином, користувачі допомагають у фільтраціях об'єктів, тому такий спосіб називається також спільною фільтрацією [9].

Таблиця 2.1.

Порівняння методів колаборативної фільтрації (КФ).

| Категорії КФ | Репрезентативні техніки | Основні переваги | Основні недоліки |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| КФ на основі пам'яті | КФ на основі сусідства (алгоритми КФ на основі елементів / на основі користувачів з кореляцією Пірсона / векторного косинуса) | * легка реалізація | * залежать від рейтингу людей |
| | | * нові дані можна додавати легко та поступово | * зниження продуктивності, колиданих не вистачає |
| | Основані на об’єктах / користувачів | * не треба враховувати зміст елементів, що рекомендуються | * не може рекомендувати для нових користувачів та об’єктів |
| | | * добра масштабованість з елементами, що оцінюються разом | * має обмежену масштабованість для великих наборів даних |
| КФ на основі моделі | Баєсівські сітки довіриКФ | * краще вирішується проблема розрідженості, масштабованості та інших проблем | * дороге моделювання |
| | Кластеризація КФ | | |

| | | | |
|---|---|---|---|
| | КФ на основі MDP (Markov decision processes) | * покращена ефективність прогнозування | * мають компроміс між ефективністю прогнозування та |
| | Латентний семантичний КФ | | масштабованістю |
| | Розріджений аналіз факторів | * дає інтуїтивне обґрунтування рекомендацій | * втратити корисну інформацію для методів зменшення розмірності |
| | КФ з використанням методів зменшення розмірності, наприклад, SVD , PCA | | |
| Гібридні рекомендатори | КФ на основі контенту, наприклад Fab | * долає обмеження КФ та вмісту або інших рекомендаторів | * збільшення складності та витрат на впровадження |
| | КФ з посиленимвмістом | * покращена ефективність прогнозування | * потрібна зовнішня інформація, яказазвичай недоступна |
| | гібридна КФ, що поєднує алгоритми КФ на основі пам'яті тамоделі , наприклад, діагностика особистості | * подолані такі проблеми з КФ, як розрідженість та «сірі вівці» | |

Для розроблюваного методу найкраще підходить колаборативна фільтрація на основі пам’яті, а саме основана на об`єктах (моделях автомобілів) або на користувачах, тому що цей метод не враховує зміст об’єктів і тому працюють швидше, а також враховують масштабованість. Необхідно порівняти ці два типи алгоритмів колаборативної фільтрації і впровадити один із них у розроблюваний метод вилучення знань

### 2.2.2. Порівняння та аналіз алгоритмів колаборативної фільтрації на основівикористувачів та об’єктів та вибір одного з них

#### 2.2.2.1. Аналіз алгоритму колаборативної фільтрації на основі користувачів

Якщо у системі є користувачі та об'єкти (у нашому випадку об’єкти - це автомобілі) та користувачі оцінили деякі об'єкти від 1 до 5, тоді всі оцінки

можна зобразити у вигляді матриці (рис. 2.2).

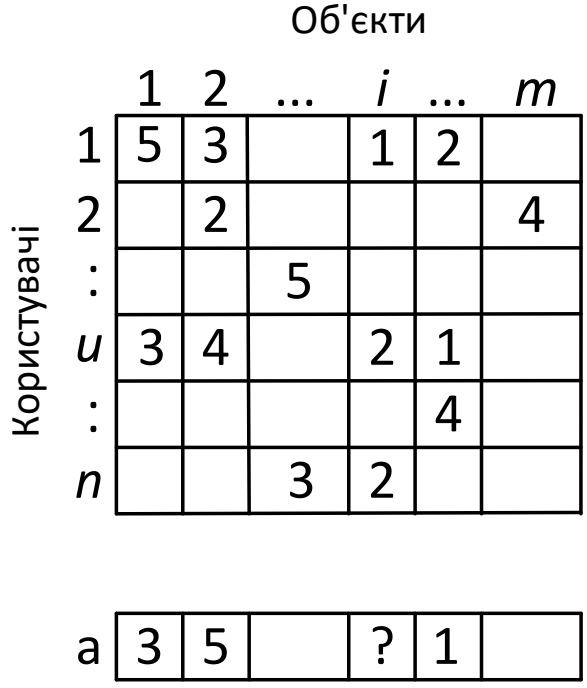



Рис. 2.2. Матриця оцінок

Нехай ім'я клієнт $a$. Задача знаходиться в тому, щоб передбачити, яку оцінку створив клієнт і об'єкт $i$. Будемо розглядати лише клієнта та тих клієнтів, які оцінюють об'єкт $i$. Алгоритм включає в себе 3 кроки:

1. Для кожного клієнта $y$ обчислюються, наскільки його уподобання відповідають інтересам клієнта $a$;
2. Після цього обирається множина клієнтів, найбільш близьких до $a$;
3. Прогнозування оцінки на основі оцінки об'єкта і «сусідами» з попереднього етапу.

*Перший крок*. Кожному клієнту в матриці $R$ відповідає один рядок. Тому буде обчислюватись поблизу векторів-рядків клієнтів. Існує безліч способів підрахунку близькості векторів. Один з найпростіших - косинус між цими віками:

$$sim(u,a) = \frac{\sum_{i=1}^{m} r_{a,i} \cdot r_{u,i}}{\sqrt{\sum_{i=1}^{m} r_{a,i}} \cdot \sqrt{\sum_{i=1}^{m} r_{u,i}}}, \quad (2.1.)$$

Тут $sim(u,a)$ - міра близькості (схожості) користувачів $a$ і $u$, $r_{,}$ - значення матриць $R$:

$u$ - рядок, $i$ - стовпець, $sim(u,a)$ приймає значення з відрізка [0,1]. Якщо користувач не вказав оцінку для якого-небудь об'єкта, відповідне значення

матриць рівно 0. Існує інша міра, близька між віками - коефіцієнт кореляції Пірсона:

$$sim(u,a) = \frac{\sum_{i\in I}(r_{a,i}-\overline{r_a})\cdot(r_{u,i}-\overline{r_u})}{\sqrt{\sum_{i\in I}(r_{a,i}-\overline{r_a})^2}\cdot\sum_{i\in I}(r_{u,i}-\overline{r_u})^2}, \quad (2.2.)$$

Тут $i$ - множина об’єктів, які оцінив як клієнт $a$, так і клієнт $u$. $r_a$и $r_u$ - це середнє оцінювання користувачів $a$ і $u$, відповідно. Чим сильніше відповідають інтереси, тим ближче значення мають близькість до 1. Якщо коефіцієнт кореляції Пірсона заперечує, то інтереси користувачів протилежні. Формули (2.1.) і (2.2.) - найпоширеніші міри поблизу, що використовуються в даному алгоритмі.

Коефіцієнт кореляції Пірсона (формула (2.2.) залишатиметься високим, якщо всі оцінки, що порівнюються користувачами, відрізняються від постійної величини (один користувач оцінює більш сувору, а іншу - надає більші оцінки). Косинус (формула (2.1.) у цьому випадку, якщо видається менше значення. Якщо оцінки двох клієнтів відрізняються пропорційно, то косинус (формула (2.1.) буде більше, кореляція (формула (2.2.) - менше. У обох випадках оцінок користувачів схожими. Отже, яку з цих метрик використовувати - залежить від конкретних завдань.

*Другий крок*. Тепер потрібно вибрати множину $K$, які найбільше схожі на користувачів. Можливо обрати всіх користувачів.

Окрім того, оскільки користувачів з несхожими чи несприятливими перетинами інтересів досить багато, вони будуть негативно впливати на точність прогнозування оцінок об'єкта $i$. Крім того, кількість користувачів впливає на обсяг обчислень на третьому етапі. Одне з можливих рішень - встановити поріг міри близькості, обчисленої на першому етапі: формули (2.1.) та (2.2.).

Користувачі з кількома близькими, підвищуючими порогами, воюють у множині $K$. Інші не будуть входити в цю множину. Але частіше всього вибирається константа $k$. Потім всі користувачі сортуються за зменшенням мір близькості. І в множину $K$ входять і користувачі, найближчі близькі до $a$.

*Третій крок*. Маючи множину $K$ близьких користувачів обчислюється оцінка, яка надіслана користувачем об'єкту і. Помітно, що розглядаються лише

користувачі, які оцінюють об'єкт i. Необхідна оцінка обчислюється за формулою:

$$p_{a,i} = \overline{r_a} + \frac{\sum_{u \in K}(r_{u,i} - \overline{r_u}) \cdot sim(a,u)}{\sum_{u \in K} |sim(a,u)|}, \quad (2.3.)$$

$p_{a,i}$ - це прогнозована оцінка користувача для об'єкта $i$. За підсумками береться його середня оцінка $p_a$, а потім з'являється середня оцінка оцінки інших користувачів із багатьох підприємств $K$ для об'єкта та від них середньої оцінки.

Чим ближче користувач у до користувача $a$ (відповідно до розміру близькості $sim(a,u)$, обчисленої на першому етапі), тим більше його вклад в прогнозування оцінок.

Отже, описаний метод пропонує оцінки об'єктів, які поточний користувач ще не оцінює.

Для того, щоб зробити рекомендацію для користувача, достатньо передбачити оцінку для всіх неоцінених об'єктів та вибрати об'єкти з найбільшою прогнозованою оцінкою.

### 2.2.2.2. Аналіз алгоритму колаборативної фільтрації на основі об’єктів (SlopeOne) та обґрунтування його переваг

Алгоритм фільтрації на основі об’єктів схожий на алгоритм на основі користувачів, він містить три кроки:

1. Для кожного об'єкта $j$ обчислюється, наскільки він схожий на об'єкт $i$, дляякого передбачається оцінка;
2. Вибирається множина об'єктів, найбільш близьких до $i$;
3. Передвіщається оцінка на основі оцінок обраних на другому кроці об'єктів користувачем $a$.

Порівняння об'єктів на першому кроці відбувається, як і в (2.1.) і (2.2.):

$$sim(u,a) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_i}) \cdot (r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_i})^2 \cdot \sum_{u \in U}(r_{u,j} - \overline{r_j})^2 \cdot}}, \quad (2.4.)$$

Тут підсумовування ведеться по множині $U$ користувачів, які оцінили об'єкти $i$ та $j$.

$p_i$- середня оцінка об'єкта $i$.

Другий крок аналогічний другому кроці алгоритму, заснованого на порівнянні користувачів. Основна відмінність в тому, що $K$ – множина близьких до $i$ об'єктів. На третьому кроці вважається зважена середня оцінка схожих об'єктів, отримана користувачем $a$:

$$p_{a,i} = \frac{\sum_{j \in K} r_{u,j} \cdot sim(i,j)}{\sum_{j \in K} |sim(i,j)|}, \quad (2.5.)$$

В алгоритмі, заснованому на порівнянні об'єктів, використовувалася складніша формула (2.3.). Це пов'язано з тим, що середня оцінка об'єкта показує, наскільки добрим є об'єкт, а середня оцінка користувача показує, наскільки строго користувач оцінює об'єкти. Формула (2.3.) обчислює оцінку з урахуванням того, що строгість оцінки різних користувачів, взагалі кажучи, різна.

Алгоритм, заснований на порівнянні об'єктів, має кілька переваг. Через те що уподобання клієнтів часто змінюються, а середня оцінка об'єкта залишається константою, то при порівнянні об'єктів досить порахувати міру близькості кожного з кожним і зберегти в пам'яті.

В частому оновленні цих даних немає необхідності. Заходи близькості можна порахувати, коли в системі немає активних користувачів.

У сучасних системах рекомендацій постійно зростає кількість користувачів і об'єктів, тому настає момент, коли необхідно використовувати розподілену архітектуру системи.

При цьому виникає проблема зберігання даних. Для її вирішення застосовується алгоритми кластеризації для користувачів або об'єктів, використовуючи міру близькості між ними: формули (2.1.), (2.2.), (2.4.). Розбиття користувачів або об'єктів на групи дає ще одну перевагу: оскільки в кластері знаходяться схожі об'єкти, то потрібний об'єкт досить на першому кроці порівнювати з об'єктами зі свого кластера. Це значно скорочує кількість обчислень [10].

Фільтрація заснована на об’єктах має багато переваг. Оскільки рейтинги прогнозуються з використанням рейтингів самої користувача, прогнозовані рейтинги, як правило, набагато більше відповідають іншим рейтингам цього

користувача.

Також подібність на основі об’єктів є загальним вибором для послуг із високим навантаженням.

Рекомендатори на основі об’єктів працюють набагато краще, ніж користувачах, саме більша точність прогнозування методу, що базується на об’єктах, є його головною перевагою.

Враховуючи всі переваги логічно використовувати у методі вилучення знань алгоритму колаборативної фільтрації на основі об’єктів (Slope One). Цей алгоритмбуде не релевантно використовувати для «холодного старту», тобто для тих користувачів, які ще не замовляли у прокаті сервісу автомобілів.

Але для покращення було вирішено додати модифікацію - алгоритм колаборативної фільтрації використовує клієнта такої ж статі з найбільш схожою датою народження на заданого клієнта.

## 2.3. Впровадження алгоритму рекомендаційних систем на основі вмісту урозроблюваний метод вилучення знань

Було вирішено, що розроблюваний метод вилучення знань буде використовувати алгоритм колаборативної фільтрації на основі об’єктів (Slope One), але цей метод не враховує вміст об’єктів, а саме характеристики моделей автомобілів, а саме країну походження, лоша дину силу, кількість циліндрів та інше.

Системи, що реалізують підхід, що ґрунтується на вмісті, аналізують набір документів та / або описи елементів, які раніше були оцінені користувачем, та створюють модельний профіль інтересів користувача на основі особливостей об’єктів, оцінених цим користувачем. структуроване представлення інтересів користувачів, прийняте для рекомендації нових цікавих предметів.

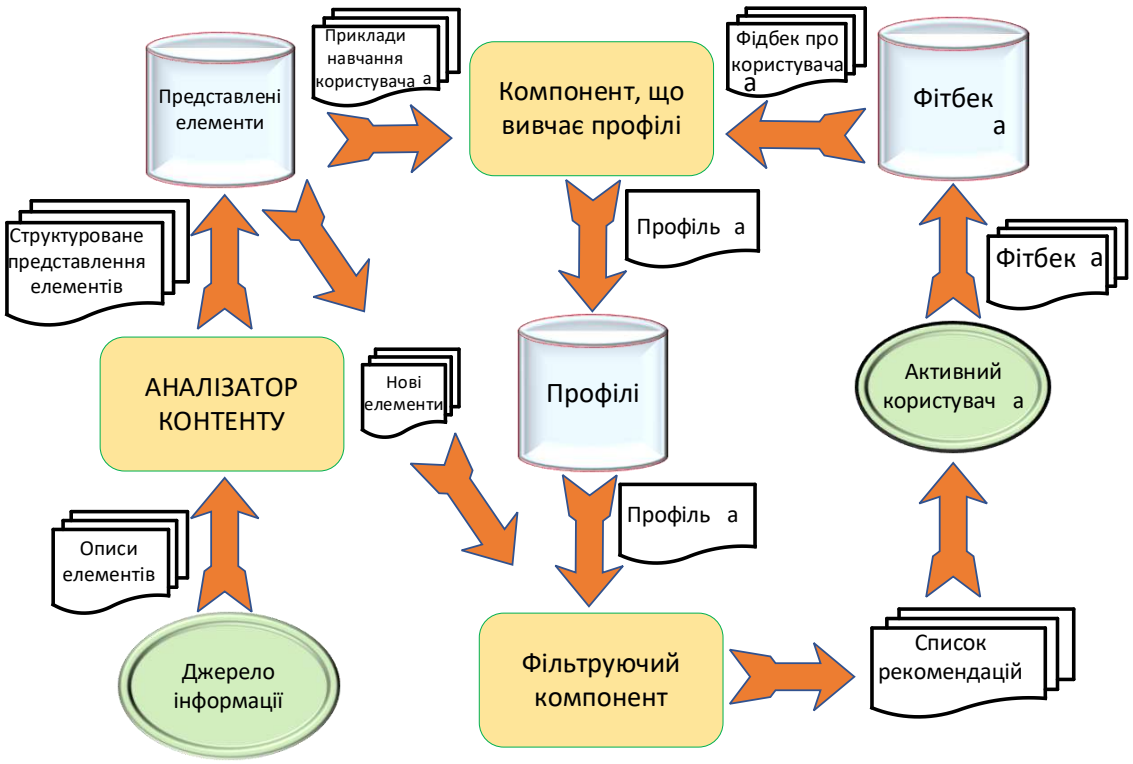



Рис. 2.3. Структура високого рівня рекомендатора на основі вмісту

Процес рекомендації в основному полягає у зіставленні атрибутів профілю

клієнта з атрибутами об'єкта вмісту. Результатом є релевантність судження, яке відображає рівень зацікавленості користувача в цьому об'єкті. Якщо профіль точно відображає вподобання клієнтів, це має надзвичайну перевагу для ефективності процесу доступу до інформації.

Наприклад, його можна використовувати для фільтрування результатів пошуку, вирішуючи, чи зацікавлений користувач у певній веб-сторінці, чи ні, і, в негативному випадку, запобігаючи його показу.

Алгоритми фільтрації на основі контексту засновані на описі об'єкта та профілі переваг клієнта. Ці методи краще підходять для ситуацій, коли є відомі дані предмет рекомендації (в нашому випадку, модель автомобілів), не про клієнта.

Зміст - орієнтовані рекомендації розглядають рекомендації як специфічну для клієнту проблему класифікації і створюють класифікатор для уподобань і небажані варіанти користувача на основі характеристик моделі автомобілю. У цій системі ключові слова використовуються для опису моделей автомобілів, а

обліковий запис клієнта використовується для визначення типу об'єкту, який подобається цьому клієнту.

Отже, ці алгоритми намагаються рекомендувати об’єкти, які схожі на ті, які клієнт обирав в минулому або вивчає в даний час. Зокрема, різні об’єкти-кандидати порівнюються з елементами, раніше оціненими клієнтом, і рекомендуються найбільш підходящі об’єкти.

Головна проблема з фільтрацією такого типу полягає в тому, чи може метод прогнозу уподобань дізнатися переваг клієнтів з дій клієнтів щодо одного походження змісту і аналізувати їх в інших типах змісту. Даний метод використовує опис або атрибути об’єктів, з якими взаємодіяв клієнт, для рекомендації схожих об’єктів. В такому випадку залежність створюється тільки на основі попереднього вибору клієнта, що робить цей алгоритм надійним, щоб уникнути проблеми холодного старту.

Наприклад, деякий клієнт замовляв автомобіль для прокату на певний строк певного типу, йому будуть рекомендуватися автомобілі з відповідним типом. Назва, рік випуску, марка, іншими словами контент, опис автомобіля, також допомагають ідентифікувати схожий за контентом автомобіль.

При такому підході зміст продукту вже оцінюється на основі уподобань клієнта (профіль клієнта), а значення ознак об’єкта - це неявні методи, які будуть використовуватися для створення профілю об’єкта. Потім оцінка об’єкта прогнозується з використанням обох профілів, і можуть бути зроблені рекомендації уподобань клієнтів [12].

Для вилучення знань уподобань клієнтів необхідно створити облікові записи (профілі) клієнта та об’єктів. Для цього є дві групи вхідних даних:

- таблиця уподобань клієнта, що показує відношення клієнт-об’єкт;
- профіль клієнта, що показує відношення об’єкт-атрибут, набір значень атрибутів об’єкта.

Тоді на основі цих даних створюється профіль клієнта, який показує відношення клієнт-атрибут, через це маємо набір значень атрибутів об’єкту деякого типу, яким найбільше віддає перевагу клієнт [13].

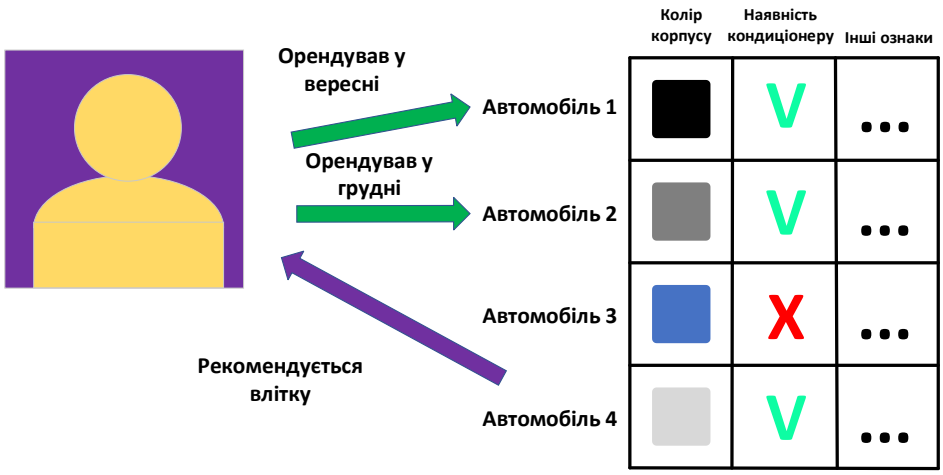



Рис. 2.4. Візуалізація підходу зі створенням профілю користувача і профілю автомобіля для прокату

Алгоритм рекомендаційних систем на основі вмісту логічно використовувати для тих клієнтів, які вже робили замовлення у сервісі прокату автомобілів, бо вони мають базу моделей автомобілів, які їм сподобалися або не сподобалися.

Таким чином можна досягти більш якісного прогнозу рекомендацій.

## 2.4. Опис алгоритму розробленого методу вилучення знань уподобань клієнтів

Розроблений метод є гібридним. У залежності від того чи є минулі замовлення у клієнта алгоритм колаборативної фільтрації на основі об’єктів (Slope One) або фільтрацію на основі вмісту на основі оцінок.

Схему роботи алгоритму методу вилучення знань щодо уподобань клієнтів прокату автомобілів наведено на рисунку 2.5.

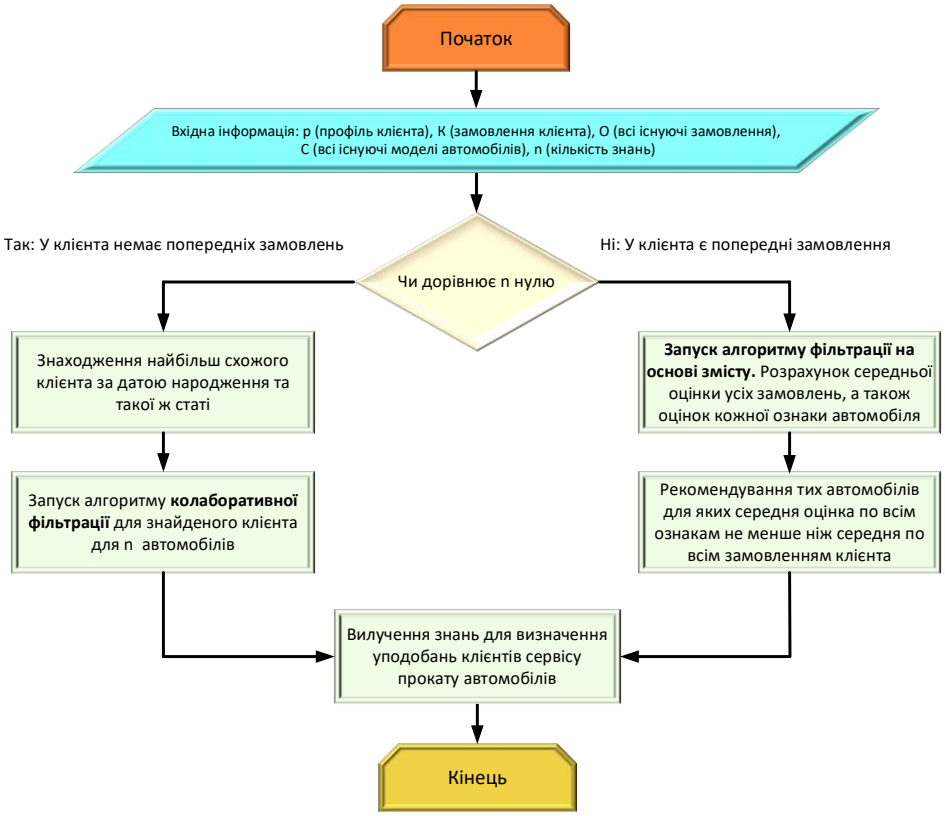



Рис. 2.5. Схема алгоритму роботи методу вилучення знань щодо уподобань клієнтів

Розроблений метод має модифікацію для алгоритму колаборативної фільтрації. Рекомендації складаються не для заданого користувача, а для користувача такої ж статі та найближчої дати народження у якої є минулі замовлення. Така модифікація допомагає поліпшити якість прогнозу уподобань клієнтів, а також частино вирішує проблему холодного старту.

# РОЗДІЛ 3 ПРОЕКТУВАННЯ МОДЕЛІ ДАНИХ ДЛЯ ПРЕДСТАВЛЕННЯ ВХІДНОЇ ІНФОРМАЦІЇ МЕТОДУ ВИЛУЧЕННЯ ЗНАНЬ УПОДОБАНЬ КЛІЄНТІВ

## 3.1. Обґрунтування вибору СУБД для проектування бази даних

$MySQL$ - це найпоширеніша повноцінна серверна СУБД. $MySQL$ дуже функціональна, що вільно розповсюджується СУБД, яка успішно працює з різними сайтами і веб додатками. Навчитися використанню цієї СУБД досить просто, так як на просторах інтернету легко знайти більшу кількість інформації.

Незважаючи на те, що в ній не реалізований весь $SQL$ функціонал, $MySQL$ пропонує досить багато інструментів для розробки додатків. Так як це серверна СУБД, додатки для доступу до даних, на відміну від $SQLite$ працюють зі службами $MySQL$.

$MySQL$ має декілька переваг. По-перше, це простота в роботі, тому що налагодити $MySQL$ на машині досить просто. Існують різноманітні програми, наприклад $MySQLWorkbench$, дозволяє досить легко працювати з БД.

По-друге, ця СУБД має багатий функціонал - $MySQL$ підтримує більшість функціоналу $SQL$. Також ця система є дуже безпечною, відомо що велика кількість функцій забезпечують безпеку, які підтримується за замовчуванням. $MySQL$ легко працює з великими обсягами даних і легко масштабується. Спрощення деяких стандартів дозволяє $MySQL$ значно збільшити продуктивність.

## 3.2. Побудова схеми моделі даних

Модель сутність-зв'язок ($Entity - Relationship\ Model$, $ERM$, $ER$-модель) дозволяє описувати концептуальні схеми предметної області при проектуванні баз даних. Вона допомагає виділити основні сутності предметної області, атрибути сутностей і зв'язку між сутностями.

Необхідно проаналізувати вхідні дані користувачів (їх профілі), а також описи автомобілів, що доступні для оренди, щоб визначити які дані повинні бути збережені у базі даних. Також необхідно визначити дані, які більш важливі для

роботи методу побудування рекомендаційної системи.

На рисунку 3.1. проілюстровано дані клієнта у вигляді схеми. Для кожної категорії даних описано чому ці дані важливі і яку роль вони можливо матимуть для розробки майбутнього методу.

На рисунку 3.2. проілюстровано важливість даних автомобілів, що доступні для прокату в сервісі. Наприклад, ціна за добу прокату має важливу роль для уточнення діапазону ціни яка підходить конкретному клієнту.

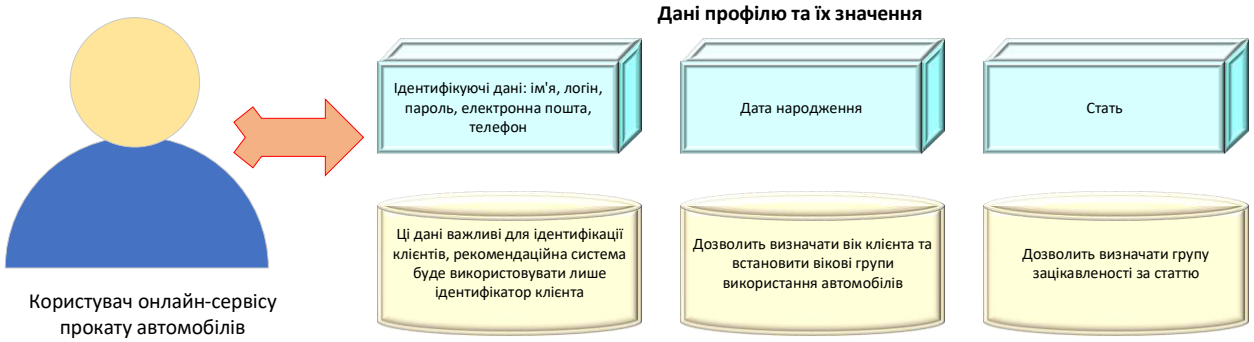



Рис. 3.1. Схема даних клієнта

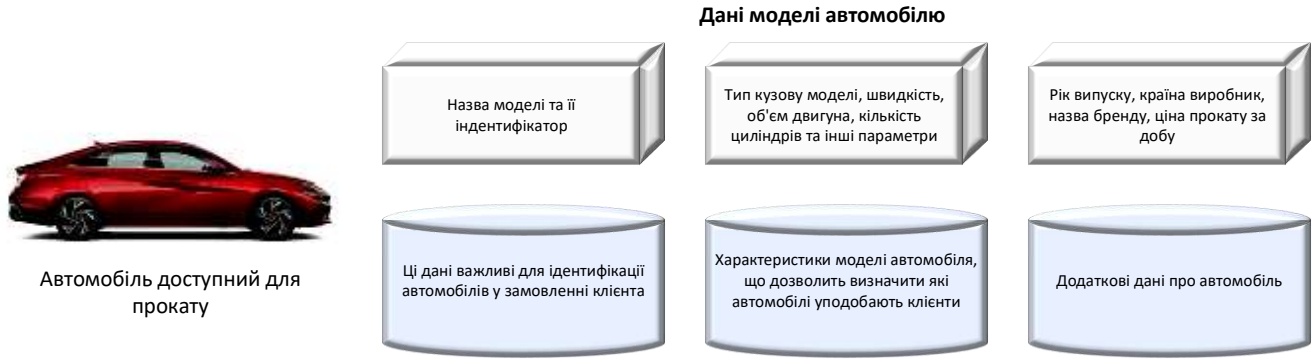



Рис. 3.2. Схема даних моделі автомобілю

На рисунку 3.3. зображено логічну модель системи прокату, а на рисунку 3.4. - фізичну. Для роботи майбутнього алгоритму достатньо трьох таблиць, а саметаблиць клієнта, замовлень та моделей автомобілів.

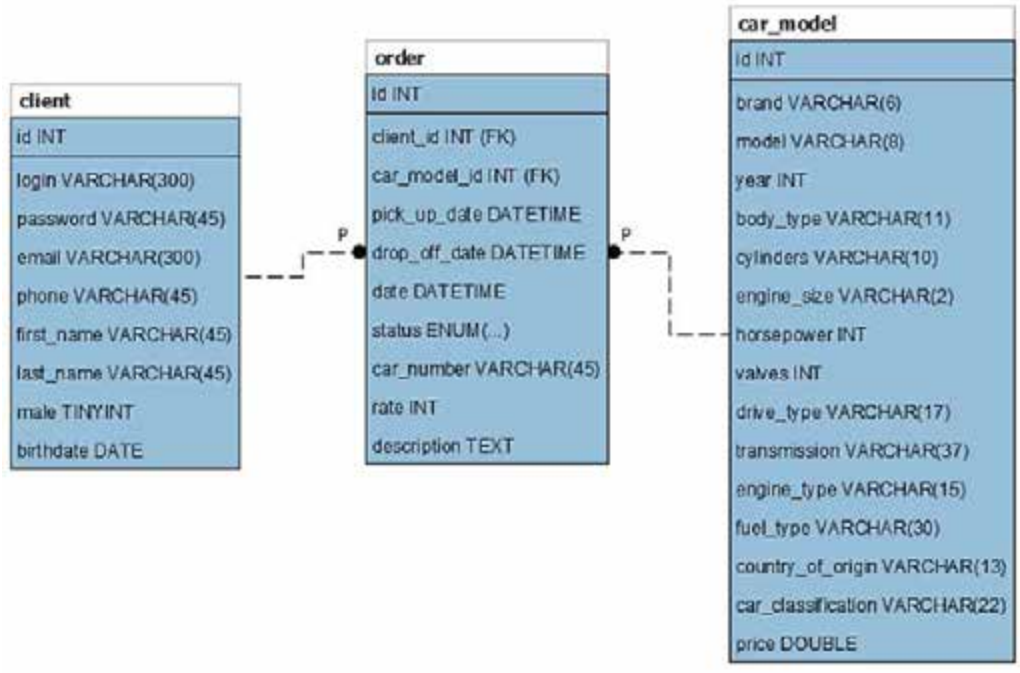



Рис. 3.3. Логічна модель даних програмної системи

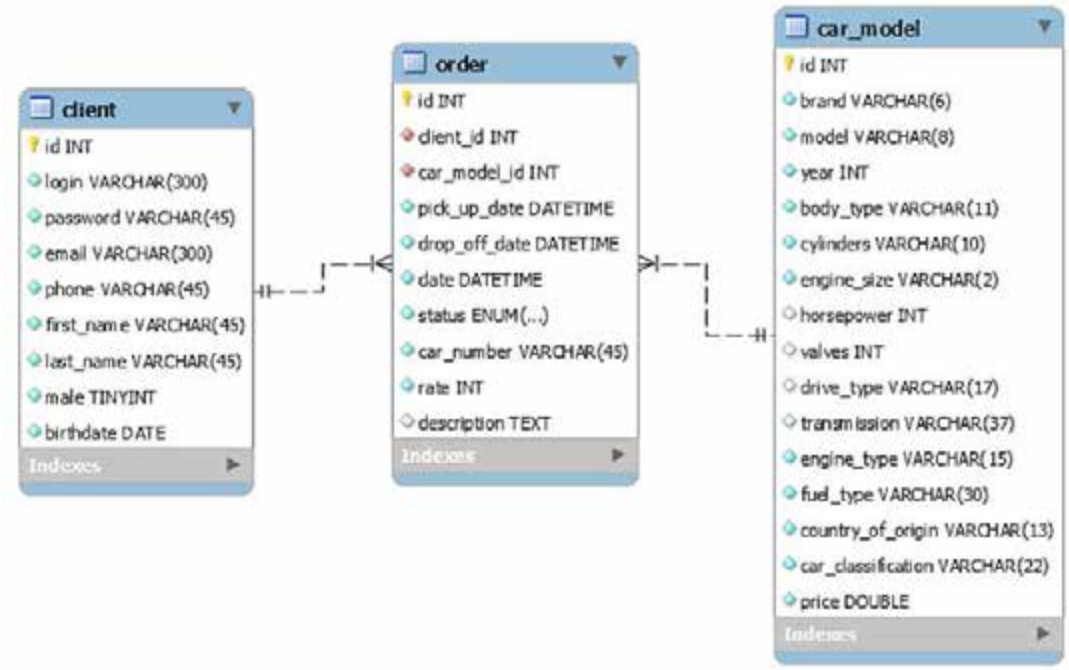



Рис. 3.4. Фізична модель даних програмної системи

Дана модель сутність-зв'язок знаходиться в третій нормальній формі.

### 3.3. Відомості про типи таблиць-сутностей

У таблиці 3.1. подано загальну інформацію щодо кожної таблиці бази даних прокату автомобілів. У таблиці 3.2. подано більш детальних опис цих таблиць, а саме: назви атрибутів та їх опис, а також приклад даних.

Таблиця 3.1.

Відомості про типи таблиць-сутностей

| Ім’я таблиці | Опис |
|---|---|
| *car_model* | Опис деякої моделі автомобілю. Наприклад, *Mazda Atenza Sport* 2.0. Містить дані про тип кузову моделі, швидкість, об’єм двигуна, кількість циліндрів та інші деталі. |
| *order* | Заказ на договір оренди. Містить дані про клієнта, що замовив, дату створення заказу, статус замовленні та інше. |
| *client* | Сутність користувача системи, містить всі дані про нього, такі як ім’я, логін, стать, день народження та інше. |

Таблиця 3.2.

Опис атрибутів таблиць прокату автомобілів

| № | Назва атрибуту | Опис атрибуту | Приклад даних |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| Таблиця *car_model* | | | |
| 1 | *id* | Ідентифікатор моделі автомобілю | 1, 2, 4321 |
| 2 | *brand* | Марка автомобілю | BMW, Ford |
| 3 | *model* | Назва моделі | 3 Series, F-150 |
| 4 | *year* | Рік випуску моделі автомобілю | 1995, 2000 |
| 5 | *body_type* | Тип кузова | Седан, Універсал, Хетчбек, Купе |
| 6 | *cylinders* | Кількість циліндрів | I4, I6 |
| 7 | *engine_size* | Розмір двигуна | 1, 2 |

| | | | |
|---|---|---|---|
| 8 | *horsepower* | Кінська сила | 170, 240,138 |
| 9 | *valves* | Кількість клапанів | 16, 24 |
| 10 | *drive_type* | Тип приводу | задній привід, повний привід, передній привід |
| 11 | *transmissio* | Передача | 5-ступінчаста механічна, 4-ступінчаста автоматична, безступінчатий автомат |
| 12 | *engine_type* | Тип двигуна | газ, гібрид, дизель, flex-fuel (FFV) |
| 13 | *fuel_type* | Тип палива | звичайний неетилований, неетилований преміум-клас (обов'язково), (неетилований/E85), дизельне паливо, неетилований преміум-клас (рекомендовано) |
| 14 | *country_of_* | Країна походження | Німеччина, США, Японія |
| 15 | *car_classifi* | Класифікація автомобілів | Компактний автомобіль, повнорозмірний пікап |
| 16 | *price* | Ціна за оренду на день у доларах США | 63,5, 58,2, 72,5 |
| Таблиця *client* | | | |
| 1 | *id* | Ідентифікатор клієнта | 1, 2, 4321 |
| 2 | *login* | Логін клієнта | Nina_smith |
| 3 | *password* | Пароль | 4aya314екма |
| 4 | *email* | Електронна пошта клієнта | test@gmail.com |
| 5 | *phone* | Телефон клієнта | 380951057843 |
| 6 | *first_name* | Ім'я | *Igor* |
| 7 | *last_name* | Фамілія | *Smith* |
| 8 | *birthdate* | День народження | 09.09.2000 |
| Таблиця *order* | | | |
| 1 | *client_id* | Ідентифікатор клієнта | 1, 7,4321 |
| 2 | *car_model_i* | Ідентифікатор моделі автомобілю | 7, 8, 45312 |

| | | | |
|---|---|---|---|
| 3 | *pick_up_dat* | Дата взяття автомобілю у прокат | 2020-07-17 14:32:34, 2020-03-13 07:02:42 |
| 4 | *drop_off_d* | Дата повернення автомобілю у прокат | 2019-06-28 01:39:55, 2021-03-25 02:16:39 |
| 5 | *description* | Опис замовлення | Особистий запит клієнта - наявність дитячого крісла |
| 6 | *car_number* | Номер автомобілю | AA 4321 AB |
| 7 | *rate* | Оцінка замовлення клієнтом від 1 до 5 | 1, 2, 5 |

### 3.4. Розробка підтримки цілісності даних

Для того щоб забезпечити цілісність даних для таблиць типу *MyISAM* потрібно створити тригери *before update* та *before delete* які будуть забороняти видалення запису з батьківської таблиці, якщо цей запис використовується у дочірній таблиці (реалізувати *restrict on delete*) та будуть обновляти запис в усіх дочірніх таблицях, якщо цей запис було оновлено в батьківській (реалізувати *cascade on update*). Це потрібно зробити для усіх зв'язків між основними таблицями та дочірніми.

Для того щоб забезпечити цілісність даних для таблиць типу *InnoDB* потрібно підчас створення фізичної моделі зв'язати таблиці за допомогою зовнішніх ключів.

Таблиці типу *MyISAM* не надають можливості створювати зовнішні ключі, тому необхідно створювати тригери щоб підтримувати цілісність даних. Для таблиць типу *InnoDB* окрім просто зовнішніх ключів потрібно в скрипті створення таблиць бази даних указати такі обмеження як:

*Cascade* дозволяє видалення/оновлення запису з видаленням/оновленням записів з відповідним значенням зовнішнього ключа.

*Restrict* забороняє видалення/оновлення запису, якщо маються записи з відповідним значенням зовнішнього ключа.

Доцільно використати *Cascade on update* та *Restrict on delete*.

Таблиця 3.3.

Приклади зв’язків між таблицями

| Номер зв’язку | Перша таблиця (зовнішній ключ) | Друга таблиця (первинний ключ) | *SQL*-інструкція | Тип посилальної цілісності |
|---|---|---|---|---|
| 1 | *order* | *car_model* | *Update* | *Cascade* |
| | | | *Delete* | *Restrict* |
| 2 | *order* | *client* | *Update* | *Cascade* |
| | | | *Delete* | *Restrict* |

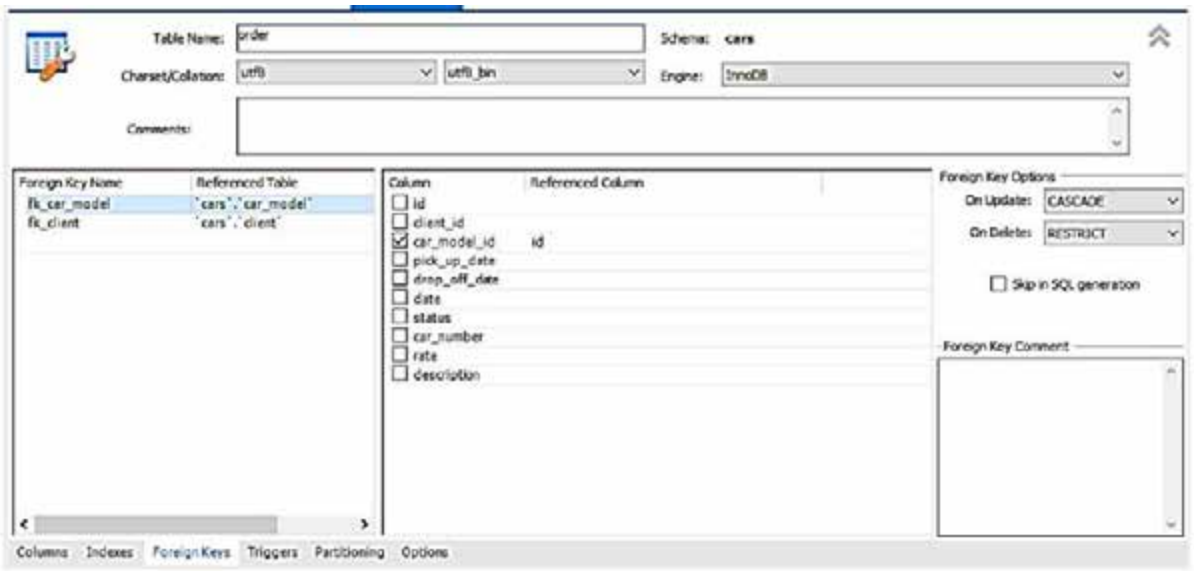



Рис. 3.5. Створення зовнішніх ключів для таблиці *order*

### 3.5. Масштабування баз даних

Вертикальний шардинг та горизонтальний шардинг.

Вертикальний шардинг це коли одну велику таблицю ми поділяємо на декілька по-менше за якимось принципом. При цьому усі шарди залишаються на тому ж інстансі.

З горизонтальним шардингом усе також, але різниця у тому шо усі табличкилежать у інших базах на іншому інстансі.

Вертикальний шардинг:

БД має 3 основні таблиці, що включають не менш ніж 2000 записів.

Вертикальний шардинг розбиває одну велику таблицю на декілька менших. Це можна зробити створив необхідну кількість таблиць та успадкував їх від основної. Наприклад. Є таблиця $client$ і від неї буде успадковане 2 таблиці: $client\_1$ та $client\_2$. У першу таблицю будуть вставлятися тільки парні записи (у яких $client\_id$ парне число), а у другу не парні. При спадкуванні таблиць успадковані таблиці не получають від основної обмеження, тригери та індекси. Все це необхідно створити знов для кожної меншої таблиці.

У випадку з горизонтальним шардингом успадкування таблиць не відбувається. Шардинг буде розподіллено по різним серверам.

У скрипті створення основної таблиці $user$ треба прописати $CHECK$ ($client\_id$) - це то поле по якому записи будуть ділитися та розподілятися.

Потрібно настроїти шардинг на основному сервері. Для того щоб виконати це потрібно підключити $extention$. Стороння оболонка даних. Далі треба створитивіддалений сервер та вказати його як $foreign\ data\ wrapper$.

Після цього на основному сервері треба створити таблицю $user\_1$ та у скрпиті створення вказати що розміщена вона буде на віддаленому сервері, який тільки що будо створено. Далі доцільно буде створити представлення, за допомогою оператора $union\ all$ об’єднати записи з 2х відділених серверів та користуватися однією великою таблицею.

### 3.6. Порівняльний аналіз варіантів реалізації баз даних та вибір однієї із них

В процесі виконання курсового проекту були створені 2 реалізації однієї інформаційної системи. На основі таблиць типу $MyISAM$ та типу I$nnoDB$. Обі два типу таблиць мають свої переваги та недоліки. У $MyISAM$ є повнотекстовий пошук що істотно прискорює виконання запитів пошуку. Це важливо для даної ІС тому що це сайт інтернет магазину аптеки с великим каталогом товарів та виробників цих товарів. В день велика кількість користувачів відвідує сайт  з ціллю придбати необхідний товар, але спочатку его ще треба знайти. Якщо кожен раз коли користувач шукає щось БД читає весь

великий каталог товарів пошук проходив би дуже повільно.

Також у $MyISAM$ операції читання виконуються швидше ніж у $InnoDB$. Це теж важливо для цієї ІС. Але в $MyISAM$ не підтримуються зовнішні ключі та транзакції. Зовнішні ключі це дуже зручно та наочно, вони забезпечують цілісність даних. Щоб забезпечити цілісність даних у $MyISAM$ потрібно зробити індекси та тригери що реалізують ту підтримку, яку виконують зовнішні ключі у $InnoDB$.

Транзакція ($Transaction$) – це блок операторів $SQL$, який в разі помилки в одному запиті, повертається до попереднього стану ($Rollback$), і тільки в разі виконання всіх запитів підтверджується ($Commit$). Транзакції істотно підвищуютьнадійність роботи БД, але за цю надійність треба платити швидкістю.

Також групова функція $count(*)$ яка була достатньо часту використана у запитах даної ІС у $MyISAM$ виконується швидше ніж у $InnoDB$.

Було вибрано тип реалізації бази даних $InnoDB$, він є більш універсальним та підтримує вбудовані механізми посилальної цілісності і не потребує написання тригерів для цього. $InnoDB$ більш популярний тип, він також має гарну швидкістьі є дуже безпечним.

### 3.7. Генерація тестових даних

Для вирішення та тестування поставленої задачі необхідні тестові дані. Спочатку згенеруємо дані для моделей автомобілів, спробуємо знайти базу реальних моделей автомобілів в інтернетному просторі.

На рисунку 3.6. продемонстровано знайдену базу даних з сайту .

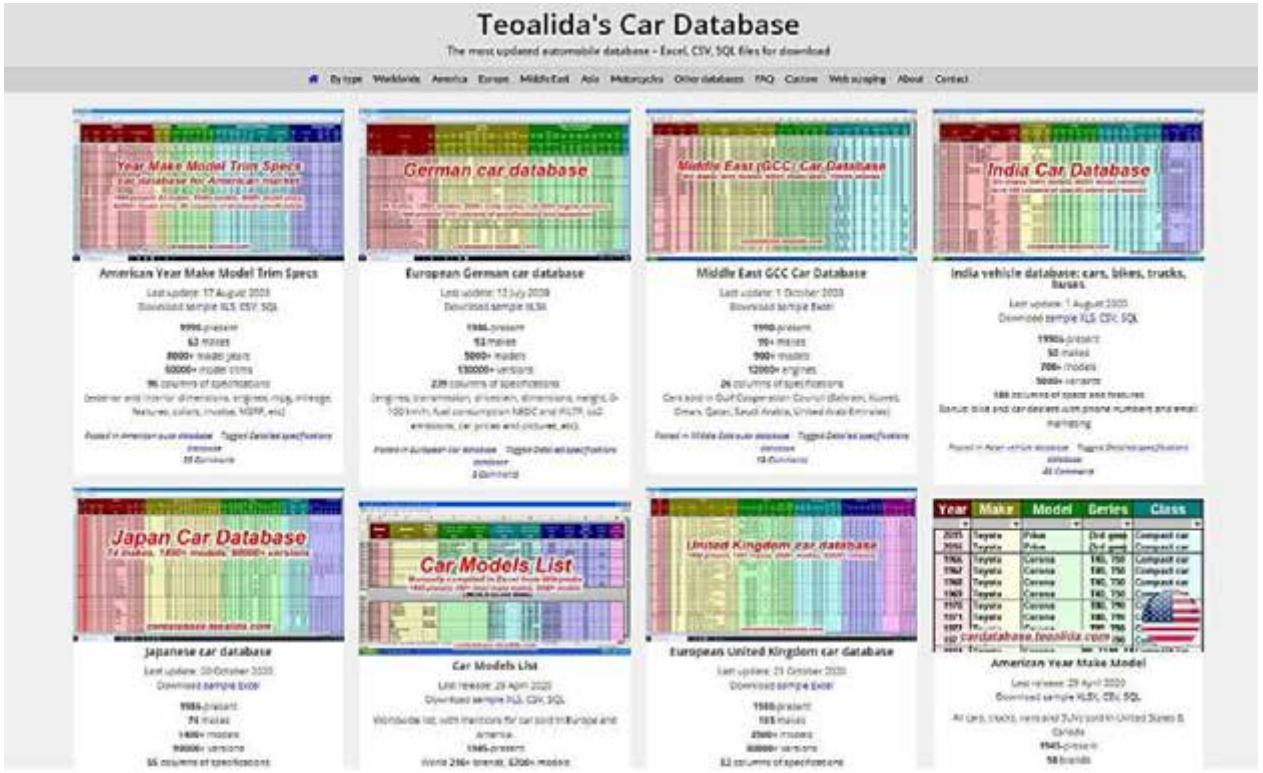
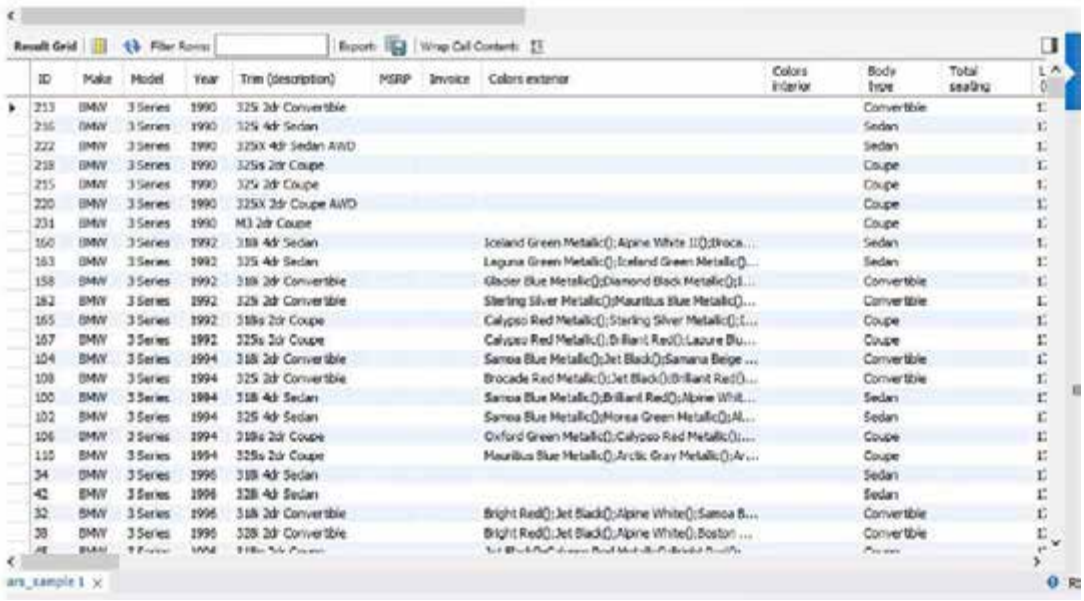



Рис. 3.6. Найновіша автомобільна база даних автомобілів Teoalida

Ці дані «брудні», вони містять багато інформації, не потрібної для розробки методу вилучення знань щодо рекомендацій. Імпортуємо ці дані у тимчасовутаблицю *cars_sample* (рис. 3.7.).



| ID | Make | Model | Year | Trim (description) | MSRP | Invoice | Colors exterior | Colors interior | Body type | Total seating |
|---|---|---|---|---|---|---|---|---|---|---|
| 213 | BMW | 3 Series | 1990 | 325 2dr Convertible | | | | | Convertible | |
| 216 | BMW | 3 Series | 1990 | 325 4dr Sedan | | | | | Sedan | |
| 222 | BMW | 3 Series | 1990 | 325iX 4dr Sedan AWD | | | | | Sedan | |
| 218 | BMW | 3 Series | 1990 | 325is 2dr Coupe | | | | | Coupe | |
| 215 | BMW | 3 Series | 1990 | 325i 2dr Coupe | | | | | Coupe | |
| 220 | BMW | 3 Series | 1990 | 325iX 2dr Coupe AWD | | | | | Coupe | |
| 231 | BMW | 3 Series | 1990 | M3 2dr Coupe | | | | | Coupe | |
| 160 | BMW | 3 Series | 1992 | 318i 4dr Sedan | | | Iceland Green Metallic();Alpine White II();Broca... | | Sedan | |
| 163 | BMW | 3 Series | 1992 | 325i 4dr Sedan | | | Laguna Green Metallic();Iceland Green Metallic()... | | Sedan | |
| 158 | BMW | 3 Series | 1992 | 318i 2dr Convertible | | | Glacier Blue Metallic();Diamond Black Metallic();I... | | Convertible | |
| 162 | BMW | 3 Series | 1992 | 325i 2dr Convertible | | | Sterling Silver Metallic();Mauritius Blue Metallic()... | | Convertible | |
| 165 | BMW | 3 Series | 1992 | 318is 2dr Coupe | | | Calypso Red Metallic();Sterling Silver Metallic();I... | | Coupe | |
| 167 | BMW | 3 Series | 1992 | 325is 2dr Coupe | | | Calypso Red Metallic();Brilliant Red();Laguna Blu... | | Coupe | |
| 104 | BMW | 3 Series | 1994 | 318i 2dr Convertible | | | Samoa Blue Metallic();Jet Black();Sahara Beige ... | | Convertible | |
| 108 | BMW | 3 Series | 1994 | 325i 2dr Convertible | | | Brocade Red Metallic();Jet Black();Brilliant Red();... | | Convertible | |
| 100 | BMW | 3 Series | 1994 | 318i 4dr Sedan | | | Samoa Blue Metallic();Brilliant Red();Alpine Whit... | | Sedan | |
| 102 | BMW | 3 Series | 1994 | 325i 4dr Sedan | | | Samoa Blue Metallic();Monsa Green Metallic();Al... | | Sedan | |
| 106 | BMW | 3 Series | 1994 | 318is 2dr Coupe | | | Oxford Green Metallic();Calypso Red Metallic();... | | Coupe | |
| 110 | BMW | 3 Series | 1994 | 325is 2dr Coupe | | | Mauritius Blue Metallic();Arctic Gray Metallic();Ar... | | Coupe | |
| 34 | BMW | 3 Series | 1996 | 318i 4dr Sedan | | | | | Sedan | |
| 42 | BMW | 3 Series | 1996 | 328i 4dr Sedan | | | | | Sedan | |
| 32 | BMW | 3 Series | 1996 | 318i 2dr Convertible | | | Bright Red();Jet Black();Alpine White();Samoa B... | | Convertible | |
| 38 | BMW | 3 Series | 1996 | 328i 2dr Convertible | | | Bright Red();Jet Black();Alpine White();Boston ... | | Convertible | |

Рис. 3.7. Таблиця *cars_sample* з брудними даними моделей автомобілів

Видалимо непотрібні колонки, перейменуємо інші щоб отримати таблицю *car_model*. Ціну за прокат на добу згенеровану по формулі (формула знайдена

досвідним шляхом і використовується для генерації тестових даних):

$$Price = engine_{size} \cdot 10 + \frac{horsepower}{10}, \quad (3.1.)$$

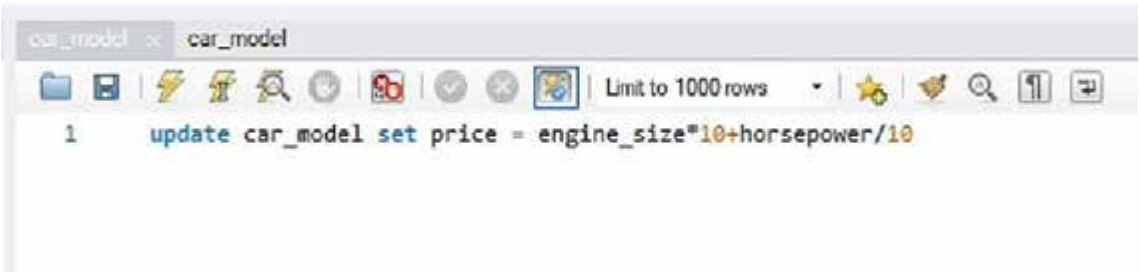



Рис. 3.8. Оновлення ціни за добу прокату у таблиці $car\_model$

Тестові дані для таблиць $client$ та $order$ сгенеруємо за допомогою онлайн-сервісу generatedata.com. $GenerateData$ – це інструмент з відкритим кодом, написаний на $PHP$, $Javascript$ та $MySQL$. Сервіс є безкоштовним, дозволяє можете створити 100 записів одночасно, працює через браузер.

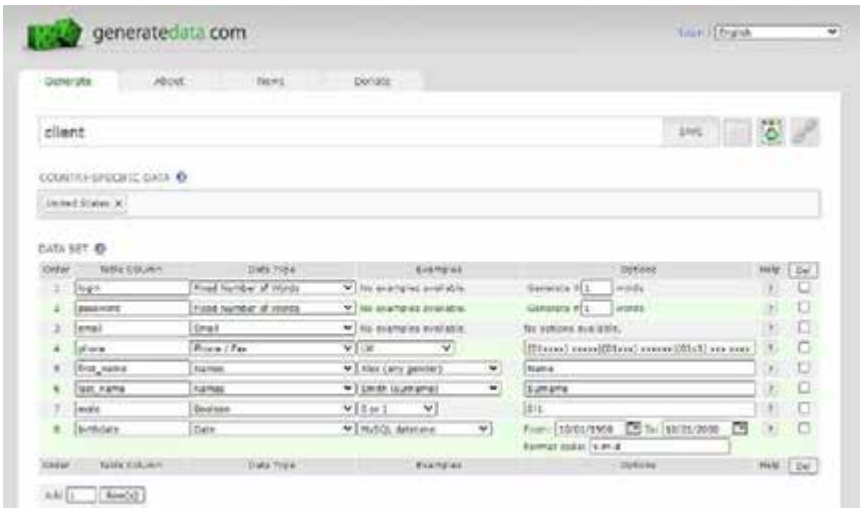



Рис. 3.9. Генерація даних для таблиці c$lient$


Рис. 3.10. Завдання додаткових налаштувань генерації даних для таблиці $client$

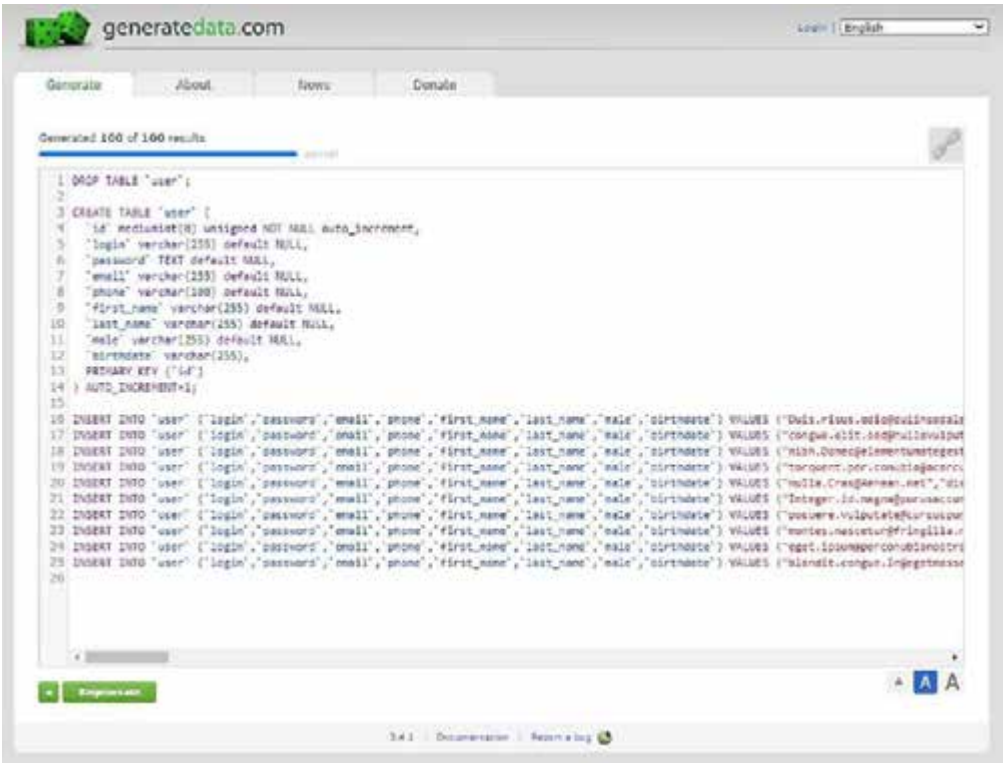



Рис. 3.11. Згенеровані дані для таблиці $client$ (100 записів)

Як видно на рисунку 3.12. ідентифікатори клієнтів йдуть не по порядку, але це необхідна умова для генерації замовлень, де буде вказуватись діапазон натуральних чисел. Для цього наступний виконаємо $SQL$ код:

$SET\ SQL\_SAFE\_UPDATES = 0;$

$SET\ @row\_number = 0;$

$UPDATE\ "order" SET\ id = @row\_number := @row\_number 6 +$
$1\ order\ by\ id;$

Ми задаємо $SQL\_SAFE\_UPDATES$ як нуль тому що його увімкнення призводить до того, що оператори $UPDATE$ і $DELETE$ видають помилку, якщо вони не вказують обмеження ключа в частині $WHERE$ або не надають $LIMIT$.

Після виконання цих операцій ідентифікатори у таблиці клієнта ідуть за порядком (рисунок 3.13.).

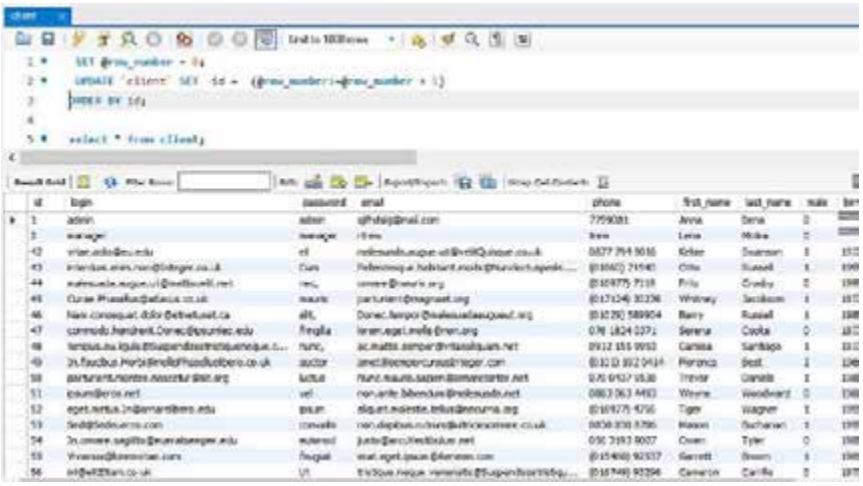


Рис. 3.12. До регенерації ідентифікаторів для таблиці клієнта

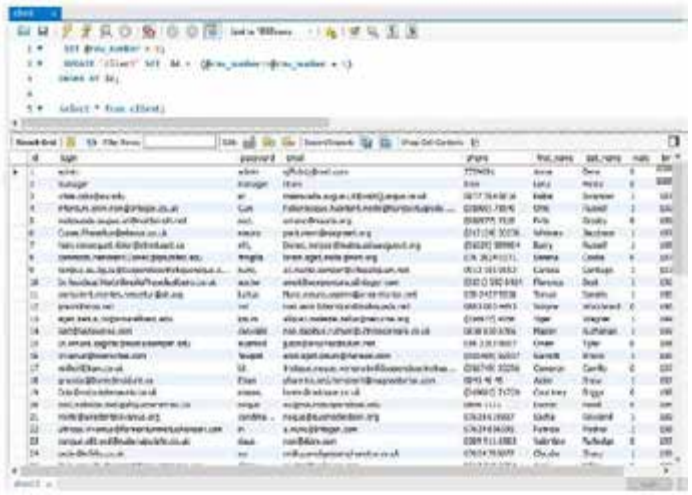


Рис. 3.13. Після регенерації ідентифікаторів для таблиці клієнта

Наступним кроком необхідно згенерувати дані для таблиці *order*, яка на даний момент є порожньою (рисунок 3.14.).

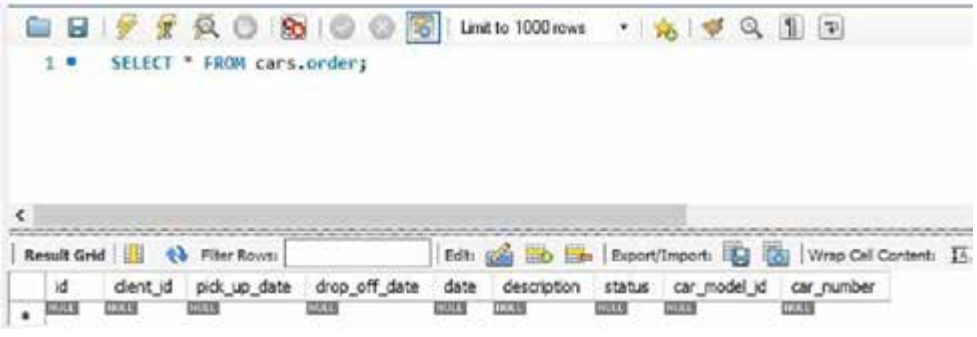



Рис. 3.14. Порожня таблиця замовлень

Згенеруємо дані для таблиці замовлень за допомогою сервісу *GenerateData*.

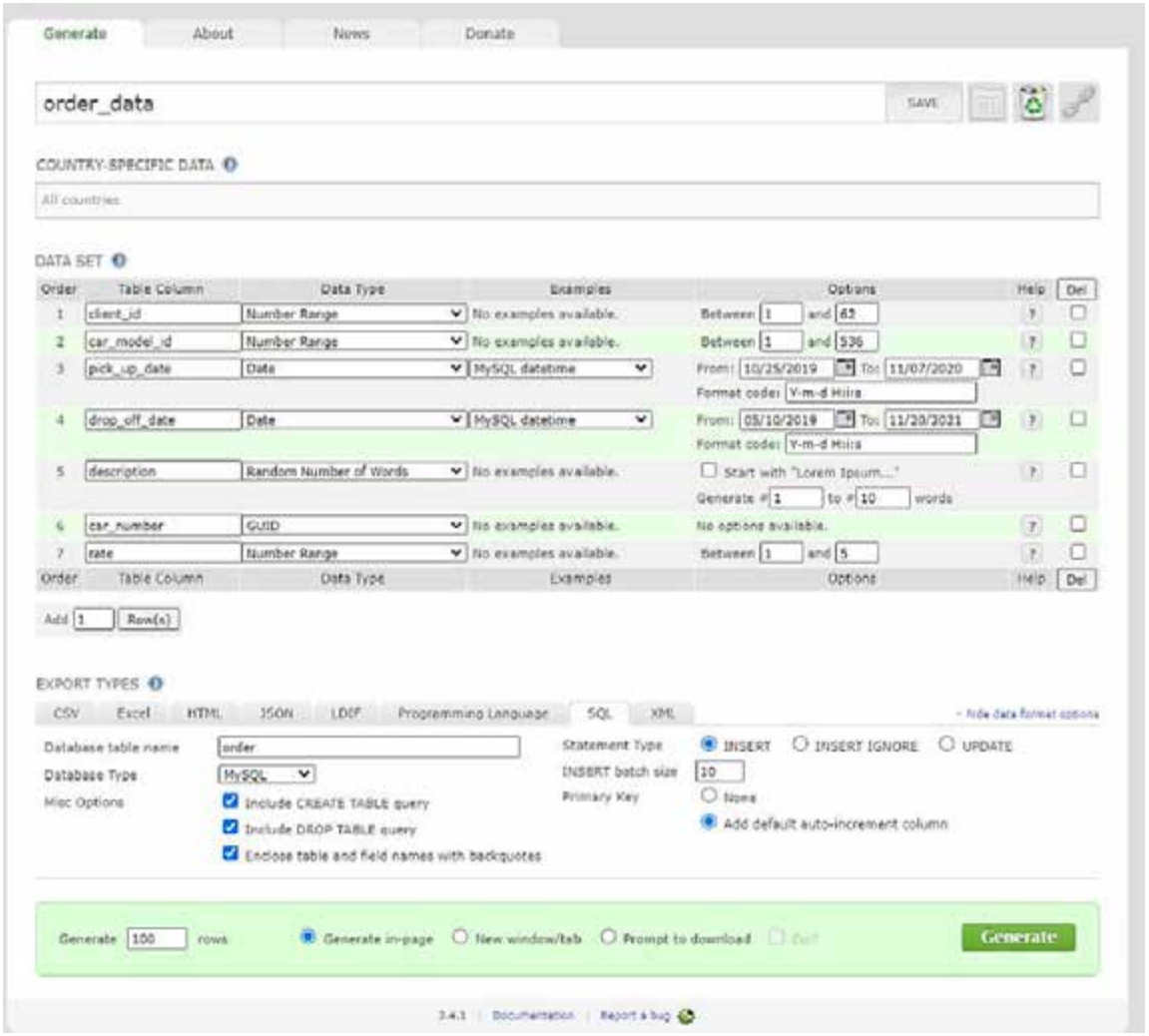



Рис. 3.15. Генерація даних для таблиці замовлень

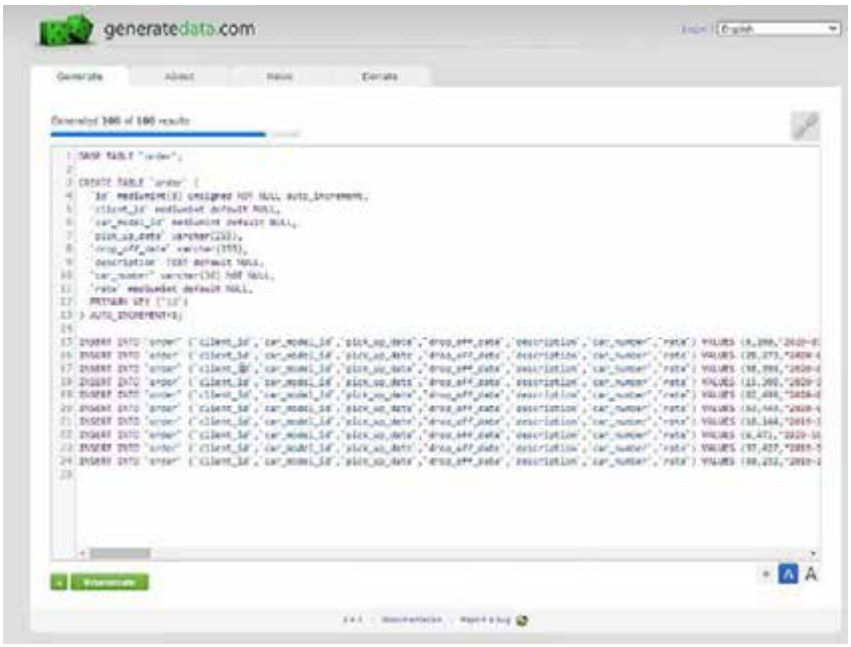



Рис. 3.16. Результат генерації даних для таблиці замовлень

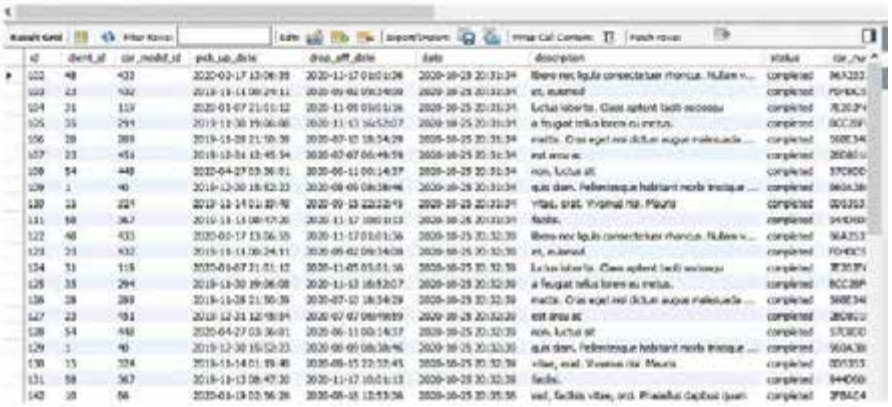


Рис. 3.17. Згенеровані дані у таблиці замовлень

Отже, було згенеровано 1020 замовлень (рисунок 3.18.).

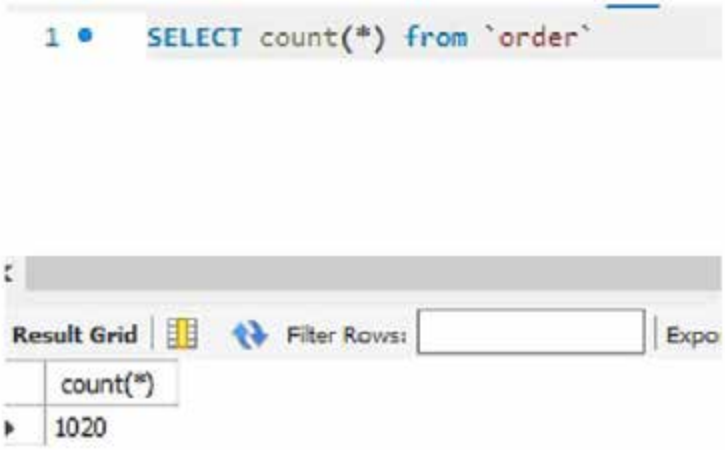



Рис. 3.18. Кількість згенерованих замовлень

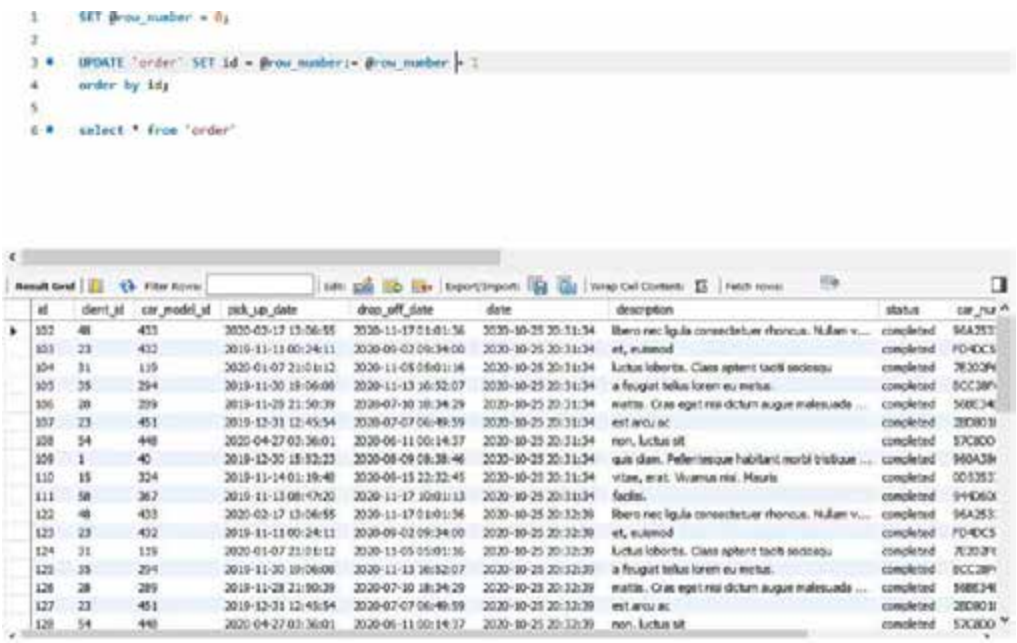
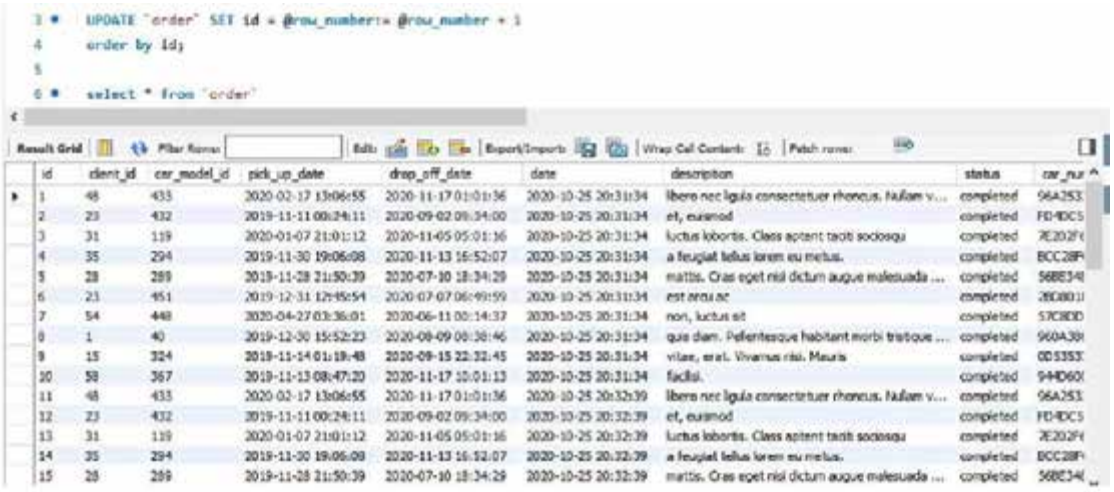



Рис. 3.19. До регенерації ідентифікаторів для таблиці замовлень



```
3 •  UPDATE `order` SET id = @row_number:= @row_number + 1
4    order by id;
5
6 •  select * from `order`
```

| id | client_id | car_model_id | pick_up_date | drop_off_date | date | description | status | car_nu |
|---|---|---|---|---|---|---|---|---|
| 1 | 48 | 433 | 2020-02-17 13:06:55 | 2020-11-17 01:01:36 | 2020-10-25 20:31:34 | libero nec ligula consectetuer rhoncus. Nullam v... | completed | 96A253 |
| 2 | 23 | 432 | 2019-11-11 00:24:11 | 2020-09-02 09:34:00 | 2020-10-25 20:31:34 | et, euismod | completed | FD4DC5 |
| 3 | 31 | 119 | 2020-01-07 21:01:12 | 2020-11-05 05:01:16 | 2020-10-25 20:31:34 | luctus lobortis. Class aptent taciti sociosqu | completed | 7E202F |
| 4 | 35 | 294 | 2019-11-30 19:06:08 | 2020-11-13 16:52:07 | 2020-10-25 20:31:34 | a feugiat tellus lorem eu metus. | completed | BCC28F |
| 5 | 28 | 289 | 2019-11-28 21:50:39 | 2020-07-10 18:34:29 | 2020-10-25 20:31:34 | mattis. Cras eget nisl dictum augue malesuada ... | completed | 56BE34 |
| 6 | 23 | 451 | 2019-12-31 12:45:54 | 2020-07-07 06:49:59 | 2020-10-25 20:31:34 | est arcu ac | completed | 2B0B01 |
| 7 | 54 | 448 | 2020-04-27 03:36:01 | 2020-06-11 00:14:37 | 2020-10-25 20:31:34 | non, luctus sit | completed | 57C8DD |
| 8 | 1 | 40 | 2019-12-30 15:52:23 | 2020-08-09 00:38:46 | 2020-10-25 20:31:34 | quis diam. Pellentesque habitant morbi tristique ... | completed | 960A38 |
| 9 | 15 | 324 | 2019-11-14 01:19:48 | 2020-09-15 22:32:45 | 2020-10-25 20:31:34 | vitae, erat. Vivamus nisi. Mauris | completed | 0D5353 |
| 10 | 58 | 367 | 2019-11-13 08:47:20 | 2020-11-17 10:01:13 | 2020-10-25 20:31:34 | facilisi. | completed | 94MD60 |
| 11 | 48 | 433 | 2020-02-17 13:06:55 | 2020-11-17 01:01:36 | 2020-10-25 20:32:39 | libero nec ligula consectetuer rhoncus. Nullam v... | completed | 96A253 |
| 12 | 23 | 432 | 2019-11-11 00:24:11 | 2020-09-02 09:34:00 | 2020-10-25 20:32:39 | et, euismod | completed | FD4DC5 |
| 13 | 31 | 119 | 2020-01-07 21:01:12 | 2020-11-05 05:01:16 | 2020-10-25 20:32:39 | luctus lobortis. Class aptent taciti sociosqu | completed | 7E202F |
| 14 | 35 | 294 | 2019-11-30 19:06:08 | 2020-11-13 16:52:07 | 2020-10-25 20:32:39 | a feugiat tellus lorem eu metus. | completed | BCC28F |
| 15 | 28 | 289 | 2019-11-28 21:50:39 | 2020-07-10 18:34:29 | 2020-10-25 20:32:39 | mattis. Cras eget nisl dictum augue malesuada ... | completed | 56BE34 |

Рис. 3.20. Після регенерації ідентифікаторів для таблиці замовлень

# РОЗДІЛ 4 РОЗРОБКА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МЕТОДУ ВИЛУЧЕННЯ ЗНАНЬ УПОДОБАНЬ КЛІЄНТІВ

## 4.1. Аналіз і вибір методів реалізації

### 4.1.1. Аналіз і обґрунтування вибору типу структури додатку

**Порівняння типів структури додатку**

$SOAP$ (Простий протокол доступу до об’єктів)

Протокол $SOAP$ для простого доступу до об’єктів ($SOAP$) - це тип структури додатку, що є також протоколом обміну повідомленнями, який дозволяє програмам спілкуватися за допомогою $HTTP$ та $XML$. Він представляє принципово безсторонню односторонню парадигму обміну повідомленнями між вузлами.

Поєднуючи односторонній обмін з функціями, передбаченими основним транспортним протоколом та / або конкретною інформацією про додатки, $SOAP$ може використовуватися для створення більш складних взаємодій, таких як запит

/ відповідь, запит / множинна відповідь. Процес виклику веб-служб дуже важливий, тому протокол $SOAP$ встановлений для обміну повідомленнями між постачальниками послуг та споживачами. Це структурований формат повідомлення $XML$ для обміну даними в розподіленому середовищі. Він використовує базовий транспортний протокол ($HTTP$, $SMTP$ тощо) через прив'язку.

Повідомлення $SOAP$ має структуру, яка характеризується двома специфічними для $SOAP$ піделементами загального конверта $SOAP$ ($env:Envelope$), а саме заголовком $SOAP$ ($env:Header$) та текстом $SOAP$ ($env:Body$).

$SOAP$ – це легкий незалежний протокол. Він незалежний і легкий, оскільки не має значення, з якої ОС чи з якої платформи використовується служба: він реагує однаково на будь-якій платформі чи ОС.

Все це можливо завдяки протоколам $XML$ та $HTTP$. Існує два типи запитів на обмін повідомленнями $SOAP$: віддалений виклик процедур ($RPC$) та запит на

документ [17].

*REST* (Representational state transfer) – тип структури пов’язаний із взаємовідносинами клієнта та сервера та способом збереження стану. Структура REST базується на стилі структури клієнт / сервер. Таким чином, запити та відповіді будуються на основі процесу передачі ресурсів. Усі ресурси ідентифікуються за допомогою унікального уніфікованого ідентифікатора ресурсу (URI), який зазвичай представляє документ, що фіксує стан ресурсу.

Як правило, Структура стилю REST набагато легша в порівнянні з SOAP. Він не вимагає включення до повідомлення форматів, таких як заголовки, як це потрібно в архітектурі SOAP. З іншого боку, він аналізує JSON – зручну для читання мову, призначену для обміну даними та полегшення аналізу та використання комп’ютером. За оцінками, це приблизно в сто разів швидше, ніж XML.

Простий документ JSON Існує кілька принципів, необхідних для проектування веб-служби RESTFul. Адресабельність – це принцип REST, коли набори даних змодельовані для роботи як ресурси, позначені URI.

Уніфікований інтерфейс вимагає, щоб інтерфейс був єдиним і стандартним, що використовується для доступу до ресурсів, тобто з використанням фіксованого набору методів HTTP. Якщо дизайнер послуг дотримується цих принципів, то майже гарантовано, що додаток REST буде простим і легким.

REST має дуже багато переваг:

- використовування меншого формату повідомлень та забезпечення економічної ефективності з часом та кращу продуктивність завдяки повідомленням JSON;
- підтримка комунікації без стану;
- просто вивчити та реалізувати;
- ефективно використовує методи HTTP;
- легка пропускна здатність, оскільки його повідомлення проходить у форматі JSON;

- для безпеки він використовує стандарти HTTP;
- REST може споживати будь-який клієнт;
- Робить дані доступними як ресурс.

Але REST має також ряд недоліків:

- він не підходить для великої кількості даних;
- REST ненадійний;
- запити REST (особливо GET) не підходять для великої кількості даних;
- затримка часу обробки запитів та використання пропускної здатності.

#### 4.1.1.2. Обґрунтування вибору типу структури додатку

В якості моделі для реалізації даної системи було вирішено використовувати REST архітектуру. REST більше займається операціями, які можна виконувати над веб-сутностями, в свою чергу SOAP зосереджений на тому, як отримати віддалений доступ до об'єктів та керувати ними. REST використовує основні операції HTTP (GET, POST тощо), і саме це робить його ідеальним кандидатом для обраної програмної системи. SOAP використовує XML і надає набір функцій, які дозволяють практикам визначати контракти між споживачами та постачальниками, що нажаль не підходить для обраної предметної області сервісу прокату автомобілів, бо необхідно уникати використання SOAP у тих ситуаціях, коли пропускна здатність дуже обмежена.

SOAP повинен передавати інформацію про об'єкти та їх стани за допомогою XML Infoset. Зазвичай ці моделі даних серіалізуються як текстовий XML. Це порівняно із типовими реалізаціями REST споживає значно більшу пропускну здатність. Усі обмеження SOAP, які так добре працюють у корпоративному світі, є контрпродуктивними при націлюванні на відкритий Інтернет. У сучасному діловому кліматі, який зазвичай не вимагає використання жорстких довгострокових контрактів між клієнтами та серверами, такі обмеження взагалі не корисні. Тим не менше, SOAP все ще активно використовується, але просто не так у загальнодоступних веб-API [17].

### 4.1.2. Аналіз і обґрунтування вибору мови програмування

#### 4.1.2.1. Порівняння мов програмування

*Java*. Java є мовою комп'ютерного програмування загального призначення, яка є синхронною, заснованою на класах, об'єктно-орієнтованою та спеціально розробленою для забезпечення якомога меншої залежності від реалізації. Вона вважається ідеальною мовою для вивчення розробниками та програмістами. Час роботи компіляції програми – 1,89 секунд, а пам'яті, що використовується в секунду, – 6,01 Мб. В даний час це найпопулярніша мова програмування, і вона знову зайняла найкращу позицію в ОС Android, хоча вона трохи знизилася. Кілька років тому, Java використовувалась для мобільних додатків корпоративного рівня, для створення настільних програм та для встановлення програм Android на планшетах та смартфонах.

Таблиця 4.1.

Недоліки та переваги Java

| Переваги | Недоліки |
|---|---|
| Java була розроблена таким чином, щоб бути простою у використанні, записі, компіляції, налагодженні, ніж альтернативні мови | Значно більше споживання пам'яті, ніж C або C ++. |
| Створює стандартну програму та код багаторазового використання | Типовий зовнішній вигляд програм графічного інтерфейсу |
| Має можливість простого переходу від однієї автоматичної системи обробки даних до іншої | Мова єдиної парадигми |

*PHP*. PHP додатково називають препроцесором гіпертексту, який може бути мовою сценаріїв на стороні служби, призначеною для розробки Інтернету, проте він також використовується як мова програмування загального

призначення. Інтернет-розробники повинні вивчити Hypertext Preprocessor, широко відому мовупрограмування.

За допомогою PHP можна надзвичайно швидко та без зусиль збільшити веб- програму. Тривалість роботи, яку використовує компілятор для виконання логіки, становить 27,64 секунд, а використовувана пам’ять – 2,57 Мб.

PHP – це фактична основа багатьох сильних систем управління вмістом, як, наприклад, Word Press.

Таблиця 4.2.

Недоліки та переваги PHP

| Переваги | Недоліки |
|---|---|
| Зручне втручання | Багато ручної роботи |
| Швидкий доступ до бази даних | Потрібне додаткове сховище |
| Надзвичайно корисні варіанти текстового процесу | Немає офіційних механізмів обробки помилок |

*JavaScript*. JavaScript є зрозумілою мовою програмування на високому рівні. Це мова, яка додатково характеризується як динамічна, слабко набрана, заснована на прототипі та багатопарадигма. JavaScript надзвичайно практичний, оскільки ця мова може надзвичайно допомогти у формуванні комунікацій для вашого веб- сайту.

Можуть використовуватися різні за стилем фреймворки в JavaScript для побудови інтерфейсу користувача. Це об’єктно-орієнтована мова сценаріїв для компіляції займає 6,52 секунд пам’ять, яку використовує компілятор, становить 4,59 мб.

Надзвичайно важливо зрозуміти, що таке JavaScript для створення інтерактивних веб-сторінок. JavaScript застосовується для включення анімації на веб-сторінках, завантаження сучасних зображень, сценаріїв або об'єктів на веб-сторінку та створення високочутливих користувальницьких інтерфейсів.

Таблиця 4.3.

Недоліки та переваги JavaScript

| Переваги | Недоліки |
|---|---|
| Простота створення веб-сторінки | Щоб створити щось, що навіть нагадує веб-сторінку, потрібно багато часу |
| Швидка передача в результаті тексту стискається | Це не так гнучко, як альтернативні розробники веб-сторінок, такі як Dreamweaver |

#### 4.1.2.2. Обґрунтування вибору мови програмування Java

Java має суттєві переваги перед іншими мовами та середовищами, що робить її придатною практично для будь-якого завдання програмування.

Java легко вивчити. Java була розроблена для простоти у використанні, а тому її легко писати, компілювати, налагоджувати та вивчати, ніж інші мови програмування.

Java є об'єктно-орієнтованою. Це дозволяє створювати модульні програми та багаторазовий код.

Java не залежить від платформи. Однією з найважливіших переваг Java є її здатність легко переходити від однієї комп’ютерної системи до іншої. Можливість запускати одну і ту ж програму на багатьох різних системах має вирішальне значення для програмного забезпечення World Wide Web, і Java досягає цього завдяки незалежності від платформи як на вихідному, так і на двійковому рівнях.

Завдяки надійності, простоті використання, крос-платформним можливостям та функціям безпеки Java стала мовою вибору для надання світових Інтернет-рішень.

### 4.1.3. Аналіз і обґрунтування вибору сервера

**Порівняння серверів**

*Tomcat.* Tomcat – це найпопулярніший сервер додатків, що

використовується з веб-додатками Java, розроблений фондом програмного забезпечення Apache. Деякі джерела стверджують, що частка ринку Tomcat на ринку перевищує 60% усіх розгортань сервера додатків Java.

Однак існує певна плутанина щодо заслуг Tomcat як сервера додатків. Специфікація під назвою Java EE точно визначає функціональність серверів додатків, але Tomcat не реалізує всіх функцій, необхідних для сервера додатків Java EE. Точним заголовком Tomcat є «веб-сервер», або «контейнер сервлетів». Незважаючи на те щоTomcat підтримує деякі функції Java EE нестандандартно, все одно є можливість використовувати більшість із цих функцій. Для цього потрібно включити їх як додаткові сторонні залежності у свою програму.

Tomcat є зрілим, добре задокументованим і найбільш широко використовуваним сервером додатків Java. Завдяки хорошій документації та відсутність підручників про це в Інтернеті, Tomcat є серйозним претендентом на роль сервера додатків майже у всіх веб-додатках Java.

*Jetty*. Jetty – це ще один сервер додатків (розроблений Eclipse Foundation), який технічно не є повнофункціональним контейнером Java EE. Як і Tomcat, він не має підтримки багатьох функцій Java EE. І як і Tomcat, він надає можливість використовувати більшість функцій, включаючи додаткові сторонні залежності.

Незважаючи на те, що частка Jetty на ринку не наближається до Tomcat, цей сервер все ще широко використовується в галузі. Дві основні точки продажів – це компактність та невелика площа. Ці фактори роблять Jetty чудовим рішенням для обмеженого середовища та для вбудовування в інші продукти.

*GlassFish*. GlassFish – це повнофункціональний та сертифікований сервер додатків Java EE, розроблений Oracle. Таким чином, GlassFish є більш важкою вагою, ніж Tomcat або Jetty – і, можливо, трохи складніше в експлуатації.

Насправді GlassFish – це більше, ніж просто загальний сервер додатків Java EE. Це довідкова реалізація стандарту Java EE. Це означає, що GlassFish використовується для демонстрації можливостей Java EE, і він отримує внески від тих самих людей, які визначають стандарти Java EE. Тому GlassFish завжди підтримуватиме найновіші функції Java EE. Це є великою перевагою цього

сервера.

Недоліком GlassFish є відсутність комерційної підтримки. У міру того, як деякий Java-проект росте та стає фінансово успішним, стає питання отримання довгострокової підтримки, оновлення безпеки, підтримки клієнтів для свого сервера додатків. GlassFish не надає такої комерційної підтримки в майбутньом і це є суттєвим недоліком цього серверу [19].

**Обґрунтування вибору сервера ApacheTomcat**

Як сервер для розробки був обраний сервер ApacheTomcat, версія 9.0.40, що є останньою на момент розробки додатку.

Tomcat є якісним та перевіреним продуктом, має величезну користувальну базу, це означає, що основні недоліки та проблеми цього сервісу вже були виправлені або покращені, також, всі можливі труднощі з цим продуктом вже обговорені на таких IT-форумах як StackOverFlow та інші. Крім цього, спільнота користувачів Apache та самі розробники продукти створили багато документації та гайдів.

Одним із переваг цього серверу є підтримка Maven через плагіни, а також сумісність з таким найпопулярнішим Java фреймворком як Spring. Розгортання веб-додатку на Tomcat є неймовірно простою та зручною.

Також, у порівнянні з розглянутими аналогами, Tomcat добре підтримує велику кількість навантажених з'єднань.

### 4.1.4. Аналіз і обґрунтування вибору фреймворку веб-дизайну Java

**Порівняння фреймворків веб-дизайну Java**

*Spring фреймворк*. Spring пропонує універсальні роботи для додатків, заснованих на Java, на всіх шарах (один рівень - окремий додаток Java, веб-рівень у веб-додатку та рівень корпоративного рівня - Enterprise Java Beans). Spring Framework передбачає близько 20 модулів, які можна використовувати на основі вимог заявки.

Головна Структура Spring Frame, яка спочатку є основним контейнером, а

потім після інтеграції доступу до даних та Web MVC. Тепер основний контейнер складається з модулів Core, Beans, Context та Expression Language. Основний модуль надає функції інжекції залежностей, а модуль Bean - фабричний шаблон Bean. Розміри модуля контексту на компактній основі, передбачені модулями Core і Beans, і це носій для доступу до будь-яких визначених та налаштованих об'єктів. Модуль Appearance Language пропонує потужну мову виразів для запитів та управління графіком об’єктів під час виконання. Рівень доступу / інтеграції даних містить JDBC, ORM, OXM та JMS. Веб-рівень включає модулі Web, Web-Servlet, Web-Struts та Web-Portlet.

*Фреймворк Struts.* Цей фреймворк розділяє веб-систему на три шари: Model, View та Controller. Модель містить JavaBeans, EJB. Представлення складається з файлів JSP, контролер поступається Action.

*Фреймворк Hibernate.* Цей фреймворк модерує ускладнення та проблеми під час маніпулювання даними JDBC та SQL. Він професійно відображає класи Java утаблиці баз даних. В основному це пов'язано з базами даних [20].

**Обґрунтування вибору фреймворку Spring**

Spring пропонує надійний спосіб обробки бізнес-об’єктів і надихає на практику, наприклад, програмування інтерфейсів, а не навчальні класи. Він програмує набагато більш загальний функціональний модуль управління документами, заснований на вдосконаленому передбачуваному управлінні життєвим циклом.

Spring and Struts надають допомогу користувачеві в розробці, налагодженні та тестуванні програмного забезпечення, але Spring має більш переваг, він є модульним легким каркас, який дозволяє вибірково використовувати будь-який його модуль у верхній частині Spring Core та забезпечує власну абстракцію винятку JDBC, що ще більше спрощує обробку винятків у програмі.

### 4.2. Опис основних технологій частини програмного додатку

Отже, даний проект є Java web-додатком, що використовує Apache Tomcat

для розгортання та MySQL базу даних як сховище. Основним веб-фреймворков є Spring Boot.

На рисунку 4.1 продемонстровано вміст application.yml – головний файл додатку, що містить конфігурації бази даних та серверу (такі як порт). На рисунку 3.1. можна побачити структуру проекту – пакети та класи.

Головним пакетом проєкту є ua.nure.anna.bohun.car.analyzer, який містить інші пакети. Пакет api містить контролери, що надають API REST-сервісу, пакет exception містить класи помилок, пакет service - сервіси, а пакет repository містить класи для зв’язку з базою даних.


Рис. 4.1. Вміст application.yml

Spring Boot дозволяє легко створювати автономні додатки без необхідності використання сервлетів, XML-конфігурацій (наприклад, дескриптор розгортання web.xml, файл опису зв’язку з базою даних context.xml та інше). Цей фремворк підтримує майже всі види серверів та message-брокерів, він значно полегшує створення веб-додатків. Для налаштування проекту використовуються клас CarAnalyzerConfig та файл конфігурацій application.yml. В останньому вказуються порт та назва додатку.

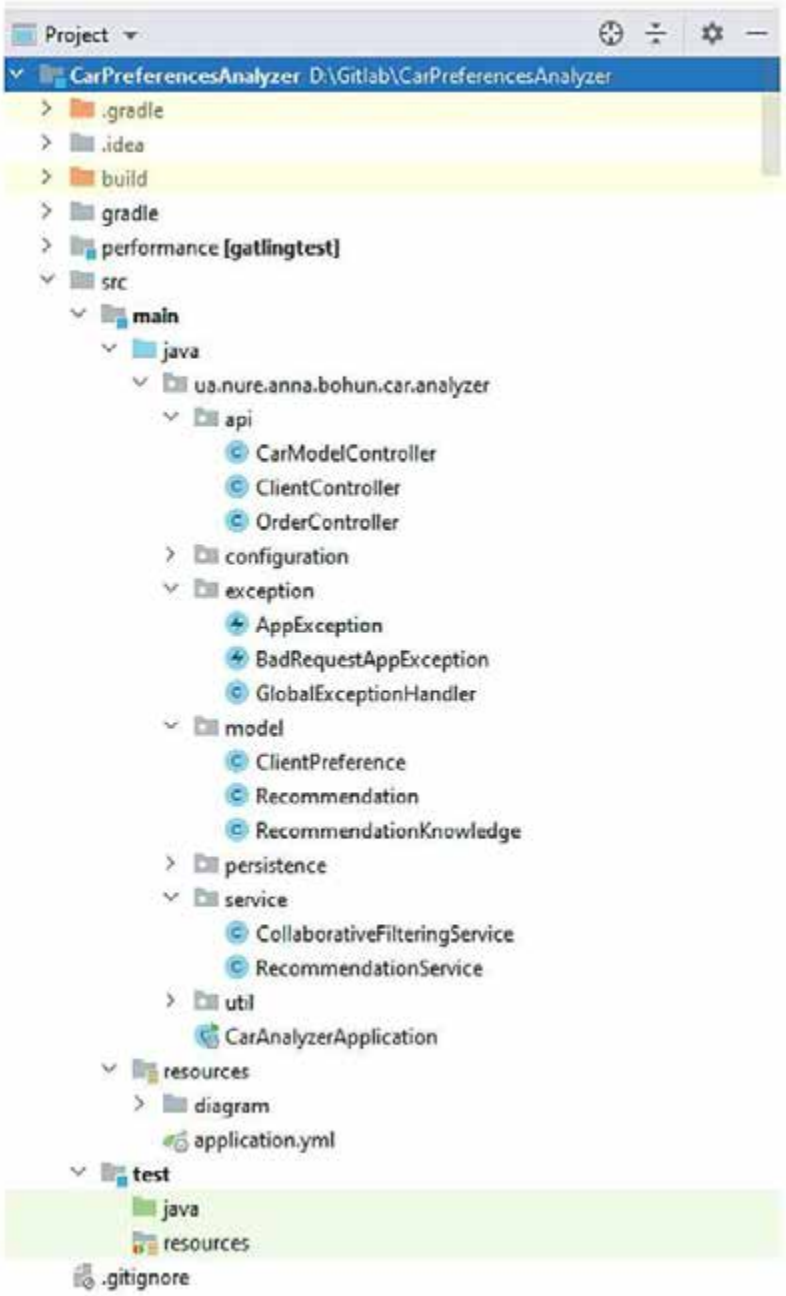



Рис. 4.2. Структура Java-проекту

Для швидкого збирання та тестування проекту використовується Gradle. Gradle – це інструмент автоматизації побудови проектів Java з відкритим кодом, який розроблений для того, щоб бути досить гнучким для створення майже будь-якого типу програмного забезпечення.

Gradle використовує репозиторії бібліотек Maven. Достатньо вказати назву та версію необхідної залежності в build.gradle і бібліотека може використовуватись у проекті.

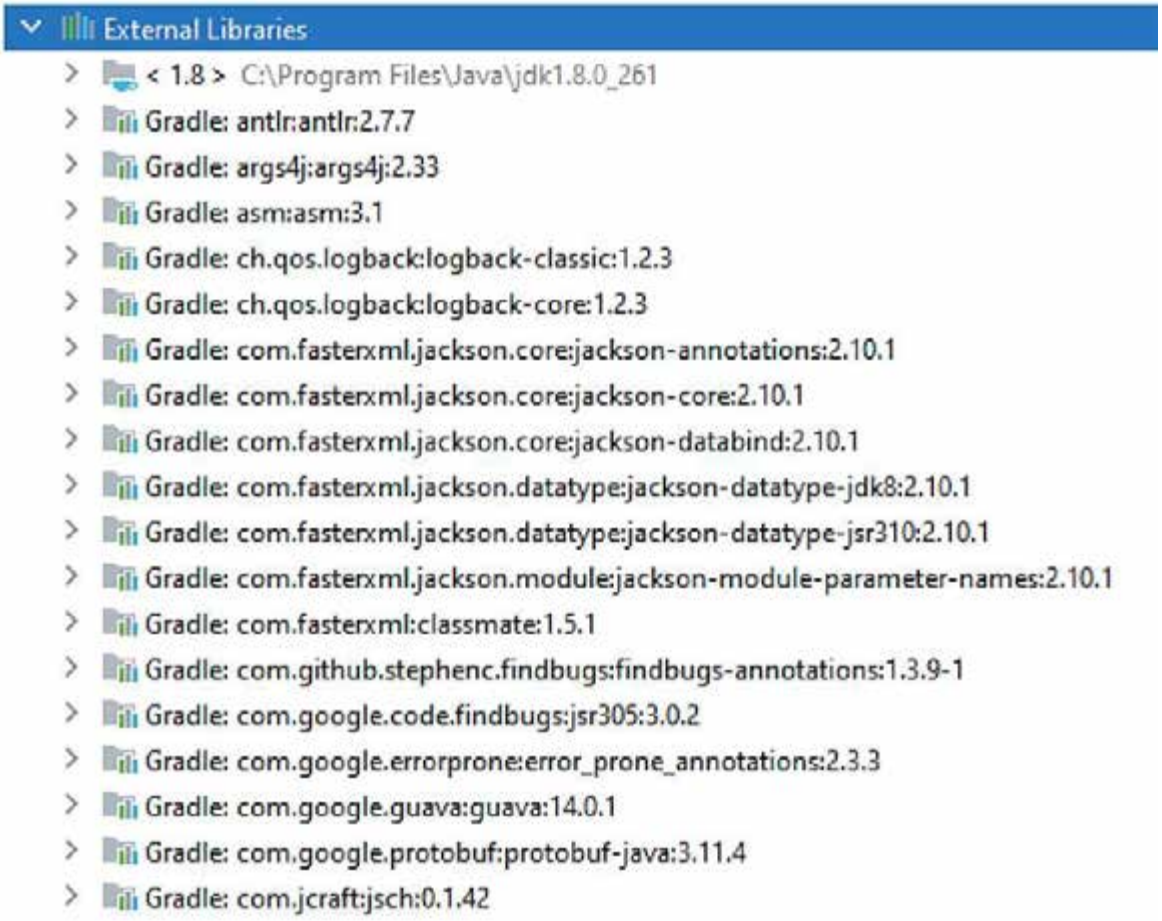



Рис. 4.3. Gradle-залежності проекту

Для доступу в базу даних використовується JDBC та такий ORM-фреймворкяк Hibernate, а точніше Spring-надбудова над ним – Spring Data. Окрім цього використовується JPA – Java Persistence API, API для визначення сутностей та типів даних їх параметрів. JDBC використовується у випадках коли необхідно написати складний SQL-запит, також він працює трохи швидше та дозволяє повністю контролювати транзакції та повернені дані.

Spring Data набагато легше для використання, використовується в інших випадках.

Для логування використовується slf4j – одну з найпопулярніших бібліотек для написання логів, підтримує yml-, xml- та properties-файли конфігурацій (у проекті використовується останній тип).

Рис. 4.4. Приклад логів додатку

Для управління версіями проекту використовувався Git – розподілена система керування версіями. Git дозволяє зберігати зміни проекту за допомогою «комітів» – кожний з них являє собою список змін файлів (оновлення, переміщення, видалення), дані про автора коміту (ім’я та електронна пошта), дату створення та повідомлення-коментар.

Також ця система контролю версій в підтримує розгалуження. Кожна гілка має свою історію комітів і являє собою альтернативний розвиток основної гілки, яка має назву master.

На рисунку 4.5. наведено частину історії комітів. Для наочного відображення використовується вбудований механізм інтеграції з Git в IntelliJ IDEA. Для цього також можна використовувати GitBash (термінал) або утиліту Gitk.


Рис. 4.5. Історія комітів додатку

Також використовувався GitLab – веб-інструмент життєвого циклу DevOps з відкритим вихідним кодом, що представляє систему управління репозиторіями коду для Git із власною вікою, систематичним відслідковуванням помилок, CI/CD пайплайном та іншими функціями.

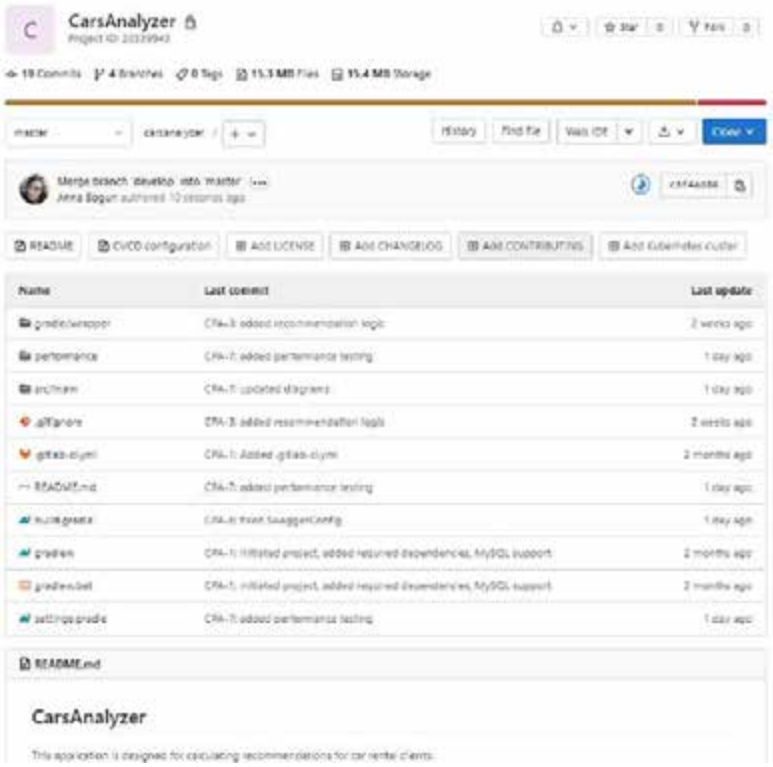



Рис. 4.6. Огляд проекту в GitLab

Для наочного відображення API (програмний інтерфейс додатку) було використано утиліту Swagger. Ця утиліта інтегрована інтегрована з фреймворком Spring, достатньо лише додати необхідні класи конфігурацій.

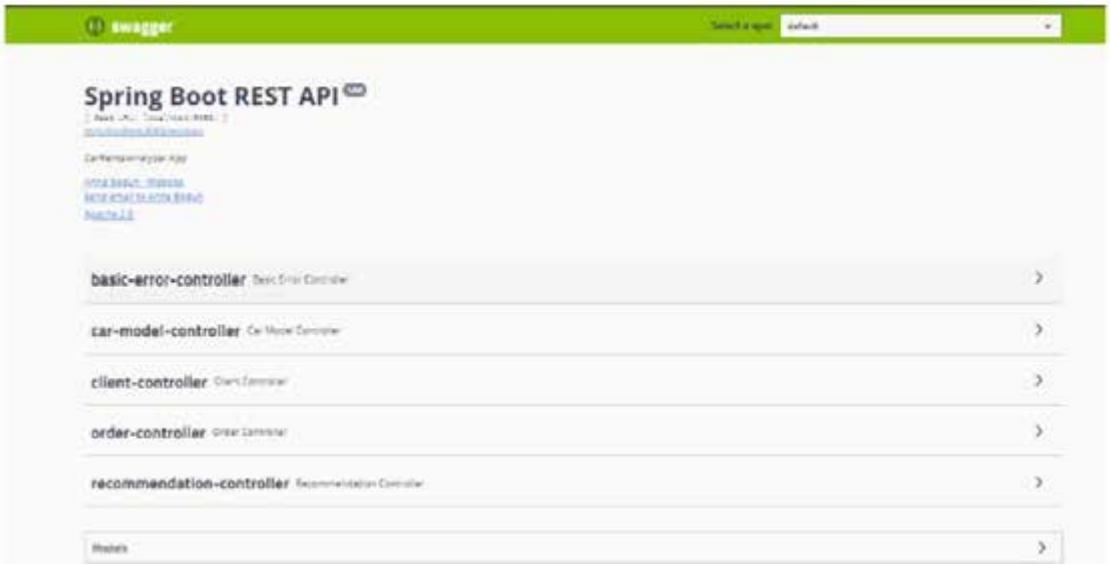



Рис. 4.7. Головна сторінка Swagger

За допомогою Swagger можна подивитись список REST-контролерів додатку, API кожного з них, також є можливість подивитись список та склад моделей (сутностей).

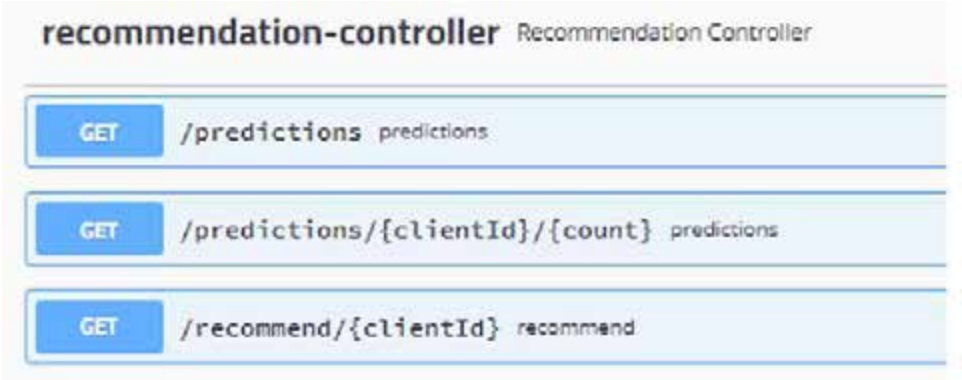



Рис. 4.8. Перелік ендпоінтів для контролеру рекомендацій


Рис. 4.9. API client-controller

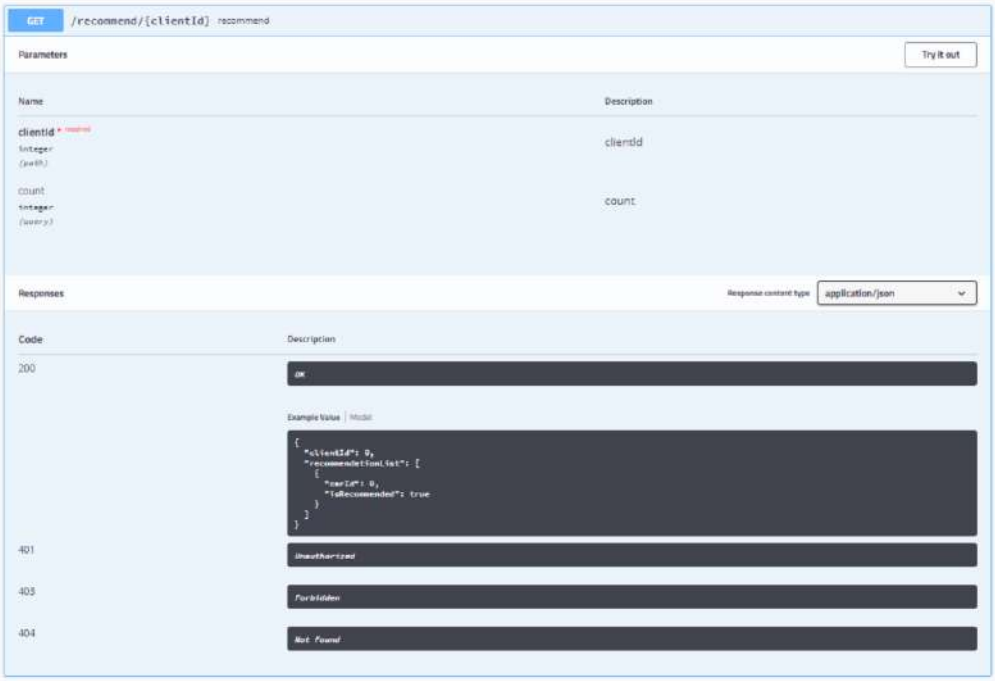



Рис. 4.10. GET-метод client-контролеру


Рис. 4.11. Моделі в Swagger

### 4.3. Опис запрограмованого методу

Для роботи з базою даних було створено три класи-сутності відповідно кожній таблиці – CarModel, Order, Client. На рисунку 4.12. зображено діаграму класів-сутностей.

Для кожного класу-сутності був створений клас-репозиторій, що містить CRUD-методи для таблиць (рисунок 4.13.)

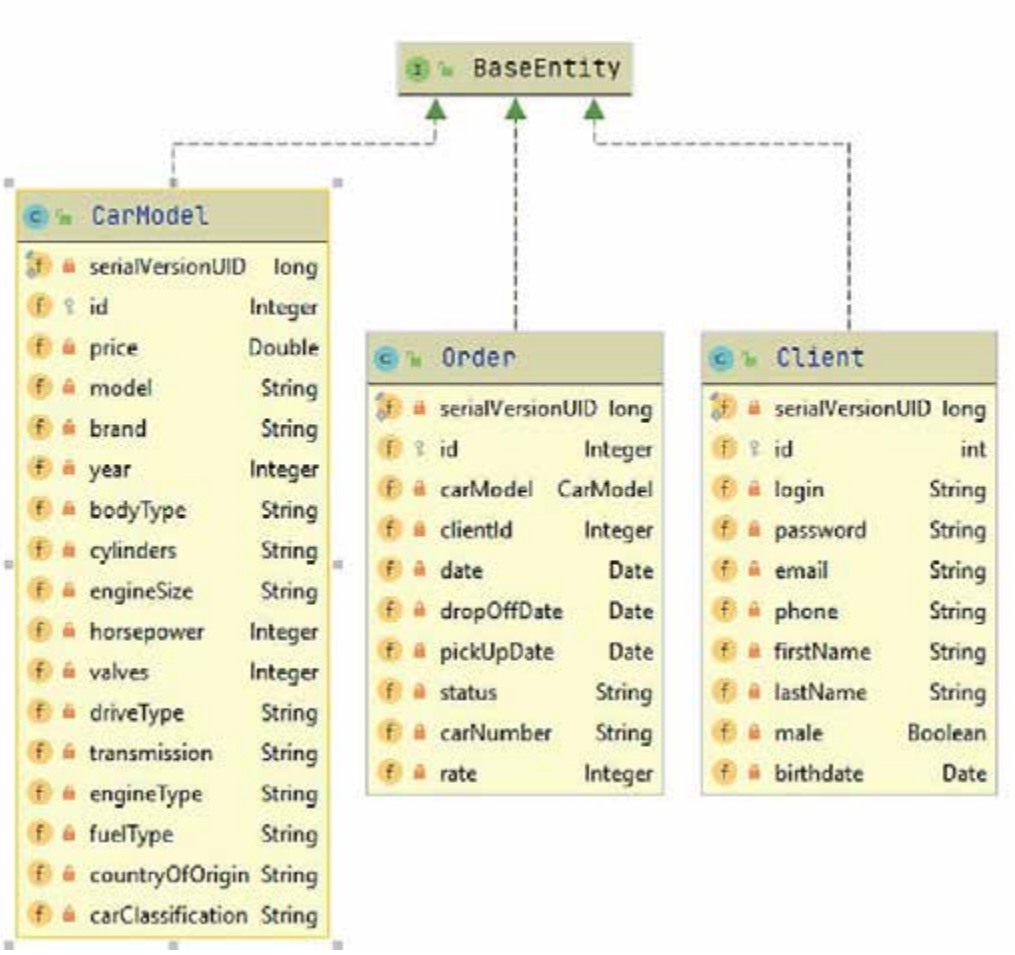



Рис. 4.12. Діаграма класів-сутностей

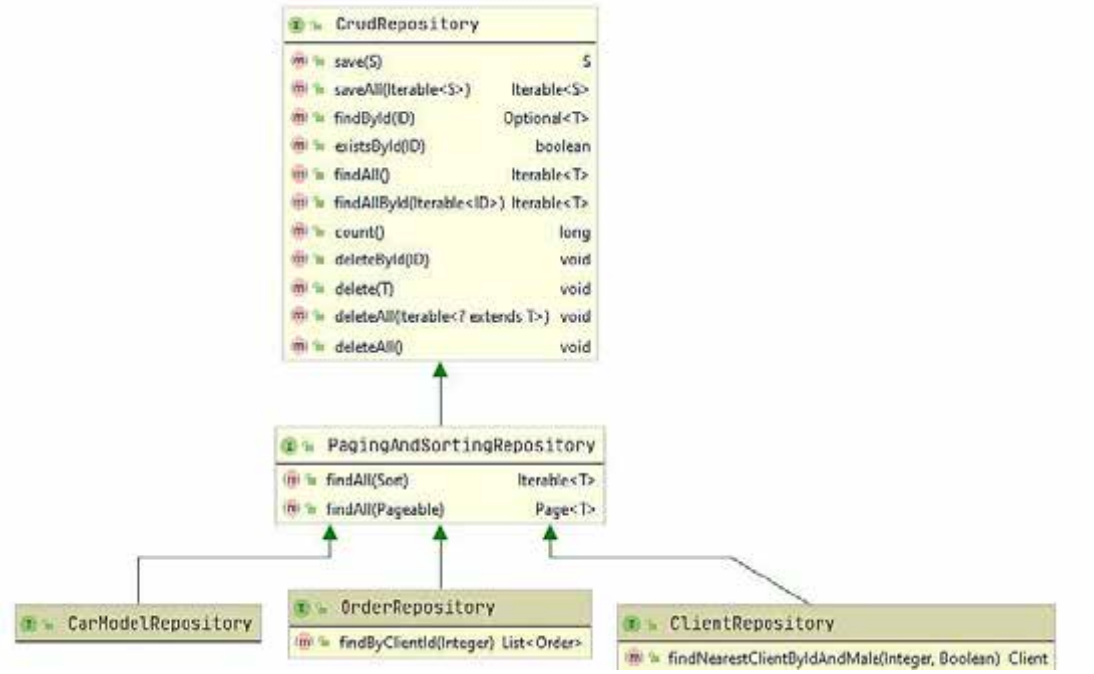



Рис. 4.13. Діаграма класів-репозиторіїв

Також було створено два сервіси – один для алгоритму колаборативної фільтрації та інший для вилучення рекомендацій (рисунок 4.14).

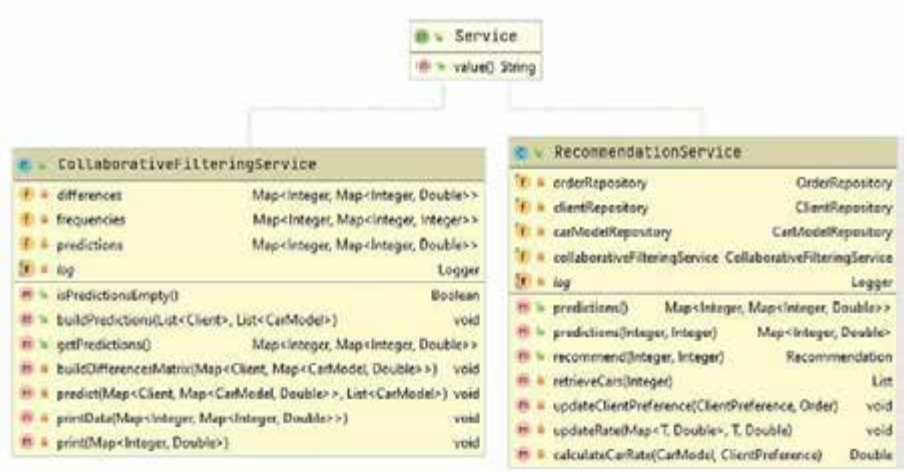



Рис. 4.14. Діаграма класів-сервісів

Для реалізації API сервісу були створені класи-контролери. Усі контролери окрім контролеру рекомендацій надають ендпоінти для додавання та вилучення нових записів.

Контролер рекомендацій у свою чергу надає API для визначення знань щодо рекомендацій клієнтів. Всі перераховані класи продемонстровано на рисунку 4.14. у вигляді діаграми. Сервіси та контролери використовують додаткові класи- моделі (рис. 4.16.)


Рис. 4.15. Діаграма класів-контролерів

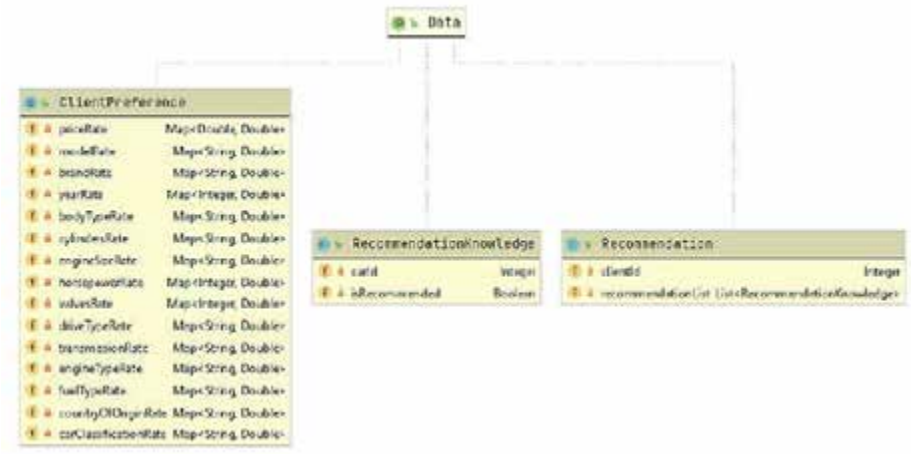



Рис. 4.16. Діаграма класів-моделей

У класі RecommendationService Було запрограмовано гібридний метод рекомендаційної системи, що поєднує колаборативну фільтрацію та рекомендації за змістом за допомогою оцінювання.

На рисунку 4.16 наведено приклад результату цього методу. Він містить ідентифікатор клієнта, а також список рекомендацій, що містить ідентифікатори моделей автомобілів та чи рекомендовано їх клієнту. Результат являю собою продукційну модель представлення знань.

Схему роботи алгоритму методу вилучення знань щодо уподобань клієнтів прокату автомобілів наведено на рисунку 4.17.

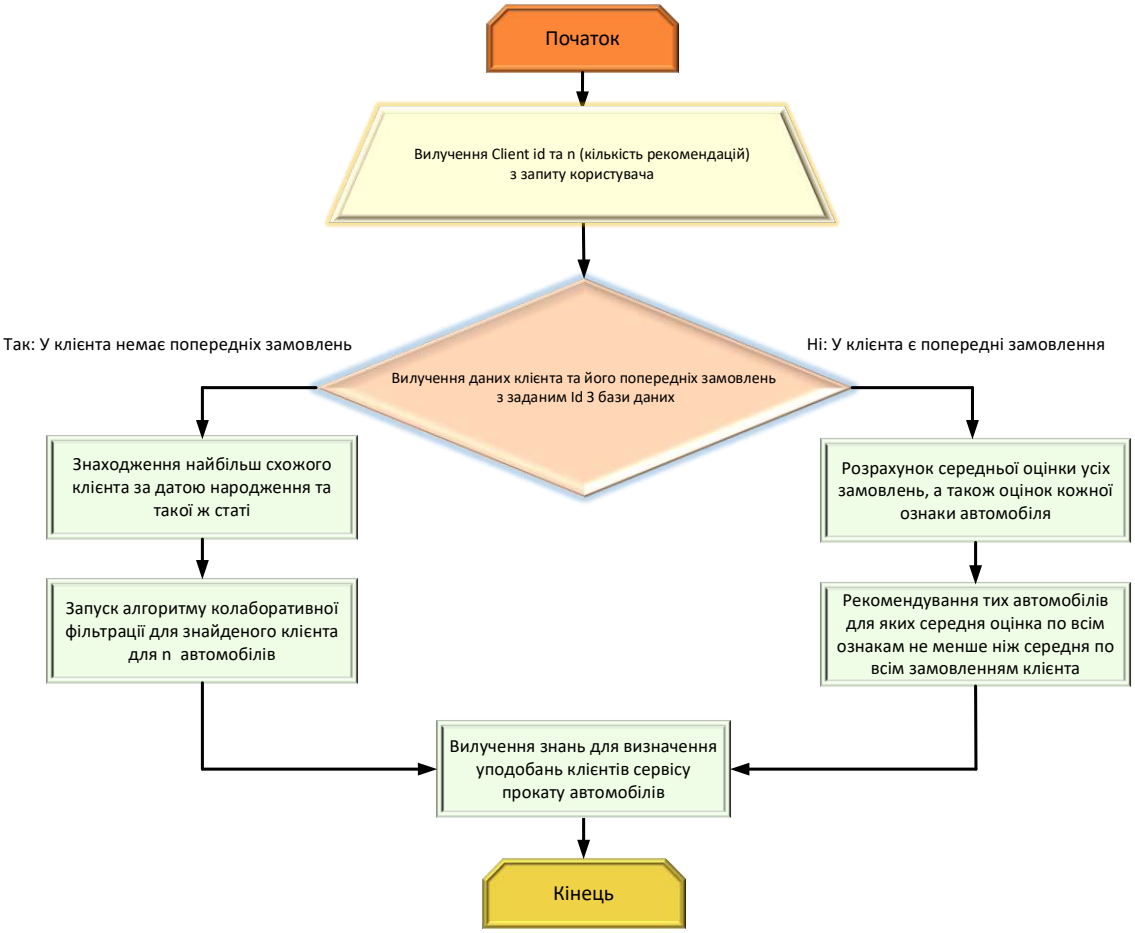



Рис. 4.17. Схема алгоритму роботи програми методу вилучення знаньщодо уподобань клієнтів

Знаходження найбільш схожого клієнта такої ж статі за датою народження було реалізовано в репозиторії для клієнта (рисунок 4.18) за допомогою SQL-запиту, продемонстрованого на рисунку 4.19.

```
@Repository
public interface ClientRepository extends PagingAndSortingRepository<Client, Integer> {
    @Query(value = "SELECT * FROM client WHERE id <> :id  AND male = :male " +
            "ORDER BY ABS( DATEDIFF( birthdate, (SELECT c.birthdate FROM client c WHERE c.id = :id))) LIMIT 1",
            nativeQuery = true)
    Client findNearestClientByIdAndMale(@Param("id") Integer id, @Param("male") Boolean male);
}
```

Рис. 4.18. SQL-запит у класі ClientRepository

```
SELECT * FROM client WHERE id <> 1  AND male = 1
ORDER BY ABS( DATEDIFF( birthdate, (SELECT c.birthdate FROM client c WHERE c.id = 1))) LIMIT 1
```

| id | login | password | email | phone | first_name | last_name | male | birthdate |
|---|---|---|---|---|---|---|---|---|
| 3 | vitae.odio@eu.edu | et | malesuada.augue.ut@velitQuisque.co.uk | 0877 764 9016 | Kelsie | Swanson | 1 | 1972-06-17 |

Рис. 4.19. SQL-запит для знаходження найбільш схожого клієнта такої ж статі за датою народження

Результат виконання ендпоінту, що запускає метод вилучення знань уподобань, продемонстровано на рисунку 4.20. Ендпоінт приймає такі значення як ідентифікатор клієнта, а також кількість знань які необхідно повернути. Результат роботи ендпоінту являє собою знання у продукційному представленні у форматі JSON.

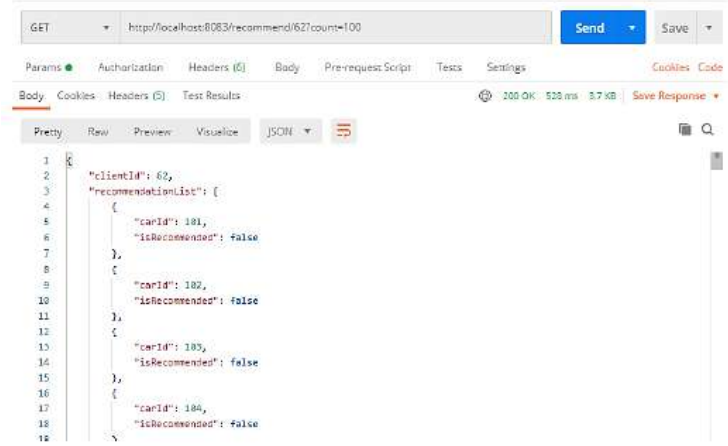



Рис. 4.20. Приклад результату роботи рекомендаційного методу

### 4.4. Тестування розробленого програмного забезпечення

Для тестування програмного продукту використовувалось декілька фрейворків. Перший з них це Gatling. Gatling - це framework для проведення

навантажувального тестування. Він заснований на трьох технологіях: Scala, Akka і Netty. В контексті даної задачі використовувалась мова програмування Scala.

Gatling дозволяє швидко створювати тести, які не тільки тестують API сервіси з точки зору правильного виконання запитів, а також з точки зору швидкості виконання цих запитів. Загалом було створено шість тестів для тестування розробленого програмного додатку.

Gatling генерує звіти, які зручно читати користувачу. Вони демонструють час роботи протестованих запитів у мілісекундах.

```
import io.gatling.core.Predef._
import io.gatling.http.Predef._

class GatlingRecommendApiTest extends Simulation {

  val httpConf = http.baseUrl("http://localhost:8083")
    .header("Accept", "application/json")
  val repeatCount = 10
  val random = scala.util.Random

  def reccomend(): Any = {
    repeat(repeatCount) {
      exec(http("Recommend API test")
        .get("/recommend/" + random.nextInt(63)+ "?count=30")
        .check(status.is(200)))
    }
  }

  val scn = scenario("Recommend API test")
    .exec(reccomend())

  setUp(
    scn.inject(atOnceUsers(1))
  ).protocols(httpConf)

}
```

Рис. 4.21. Приклад навантаженого тесту з використанням Gatling

Всі розроблені тести успішно пройшли. На рис. 4.22. - 4.25. наведено згенеровані звіти що містять результати тестування запитів. Як можна побачити, всі запити виконуються успішно, а також виконуються менш ніж 800 мілісекунд.


Рис. 4.22. Згенерований звіт з результатами тестування автомобіль API

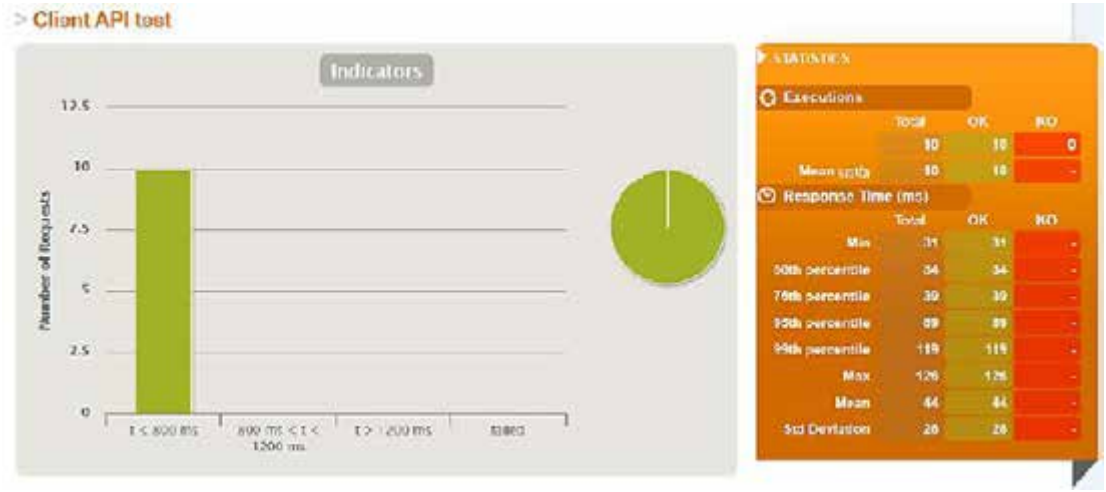



Рис. 4.23. Згенерований звіт з результатами тестування Клієнт API


Рис. 4.24. Згенерований звіт з результатами тестування порядку API


Рис. 4.25. Згенерований звіт з результатами тестування Рекомендація API

# ВИСНОВКИ

В ході виконання даної магістерської кваліфікаційної роботи був проведений аналіз предметної області, огляд існуючих аналогів систем побудування рекомендаційних систем, виділено їх переваги та недоліки. Проведений огляд показав необхідність і актуальність розробки методу вилучення знань уподобань клієнтів сервісу прокату автомобілів. Була виявлена та проаналізована проблема, яка потребує свого рішення, бо саме інформаційне перевантаження стало серйозною проблемою онлайн-сервісів прокату автомобілів, рішенням якого і є розробка методу вилучення знань для визначення уподобань клієнтів сервісу прокату автомобілів.

Проаналізовано такі алгоритми як колаборативна фільтрація та рекомендації на основі змісту, на основі цих алгоритмів був розроблений метод вилучення знань для обраної предметної області. Розроблений метод використовує алгоритм колаборативної фільтрації на основі об’єктів SlopeOne у випадку, якщо для клієнта, для якого вилучаються знання, не існує попередніх замовлень. Була додана покращуюча методика модифікації, яка проводить алгоритм колаборативної фільтрації для клієнта такої ж статі з найбільш схожою датою народження з заданим клієнтом, така модифікація дозволяє побороти проблему «холодного старту».

У випадку, якщо клієнт вже має замовлення у сервісі прокату автомобілів, використовується рекомендаційний метод на основі змісту. Цей метод аналізує ознаки моделей автомобілів попередніх замовлень і за допомогою оцінювання вилучає знання щодо уподобань конкретного клієнта, таким чином роблячи прогноз рекомендацій.

Розроблений метод був реалізований у вигляді програмного забезпечення, який надає користувачу можливість побачити знання щодо уподобань моделей автомобілів для прокату. Програмний додаток написаний на мові програмування Java за допомогою декількох фреймворків таких як Spring Boot.

Програмне забезпечення було протестовано за допомогою

навантажувального тестування, яке показало, що всі запити тривають менше ніж 800 мс, що говорить про якісну роботу розробленого програмного продукту.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Павленко В. М., Богдан В. І. Гібридні силові установки для сучасних автомобілів. Вісн. Вінниц. політехн. ін-ту. 2019. № 5. С. 108-111.
2. Тімков О. М., Григорашенко О. В. Поява гібридних силових установок на транспортних засобах. Вісник Донецької академії автомобільного транспорт. 2015. № 1. С. 42-47.
3. О. В. Бажинов та ін. «Гібридні автомобілі»: монографія / Харк. нац. автомоб.-дор. ун-т. Х.: Крок, 2008. 327 с.
4. О. М. Артюх, О. В. Дударенко, А. Ю. Сосик, А. В. Щербина. «ДВЗ з нетрадиційними робочим циклами. Напрямки розвитку транспортних енергетичних установок» / Укл.: Запоріжжя : ЗНТУ, 2019. 82 с.
5. Осетров О. О., Кравченко С.С., Чучуменко Б.С. «Обґрунтування параметрів послідовної гібридної силової установки легкового автомобіля». Двигуни внутрішнього згоряння. 2022. №1. С.78-85 DOI: 10.20998/0419-8719.2022.1.10
6. В.П. Кужель, Д.С. Стаднійчук «Сучасні гібридні силові установки для легкових автомобілів» / Матеріали VI міжнародної науково-практичної конференції «Сучасні технології та перспективи розвитку автомобільного транспорту», 21–23 жовтня, 2013 р.: Збірник наукових праць. - Вінниця : ВНТУ, 2013. – С. 145 – 147.
7. Автомобільний транспорт: Сб. науч. тр. Вип.17. – Харків : РИО ХНАДУ, 2005. - С. 103-107.
8. В.П. Кужель, О.В. Харчук «Проблеми та перспективи експлуатації електромобілів на території України» / Науково-технічна конференція Вінницького національного технічного університету. XLV Науково-технічна конференція факультету машинобудування та транспорту, 10-11 березня 2016 р. : Збірник наукових праць / Вінницький національний технічний університет. – Вінниця: ВНТУ, 2016.
9. В.П. Кужель, В.В. Красиленко «Основні проблеми експлуатації електромобілів в Україні та шляхи їх вирішення» / //Матеріали VIII міжнародної науково-практичної конференції «Сучасні технології та

перспективи розвитку автомобільного транспорту», 19–21 жовтня, 2015 р.: Збірник наукових праць / Міністерство освіти і науки України, Вінницький національний технічний університет [та інш.]. – Вінниця: ВНТУ, 2015. – С. 132 – 135. 5.

10. В.П. Кужель, Д.П. Комар, А.А. Кашканова «Варіанти застосування гібридних силових установок на автомобілях» / Матеріали X міжнародної науково-практичної конференції «Сучасні технології та перспективи розвитку автомобільного транспорту», 23–25 жовтня, 2017 р.: Збірник наукових праць / Вінницький національний технічний університет [та інш.]. - Вінниця: ВНТУ, 2017. -С. 116 – 119.
11. Колеснікова Є.Б., Колесніков В.О. «Технологічні тенденції та дизайн в автомобілебудуванні». Матеріали VIII-ої міжнародної науково-практичної інтернет-конференції «Проблеми і перспективи розвитку автомобільного транспорту». 14-15 квітня 2020 року: збірник наукових праць. / Міністерство освіти і науки України, Вінницький національний технічний університет [та інш.]. - Вінниця: ВНТУ, 2020. С. 190 - 203.
12. Василенко О. Є., Безруков В. О., Шуліка С. О., Знова О. І., Іщенко Б. М., Колесніков В. О. «Нові технологічні тенденції в автомобільному транспорті» / Матеріали VII-ї Міжнародної науково-технічної інтернет-конференції «Проблеми та перспективи розвитку автомобільного транспорту», 8 - 10 квітня 2019 р., м. Вінниця. С. 13 - 24.
13. Колесников В.А. «Водневі технології. Частина 2. Вантажні водневі автомобілі». Матеріали VIII-ої міжнародно науково-практичної інтернет-конференції «Проблеми і перспективи розвитку автомобільного транспорту» (Materials of VIII-th international scientific practical internet-conference «Problems and prospects of automobile transport»). 14-15 квітня 2020 року: збірник наукових праць. / Міністерство освіти і науки України, Вінницький національний технічний університет [та інш.]. –- Вінниця: ВНТУ, 2020. С. 158 - 165.

14. Балицький О.І., Колесніков В.О., Іщенко Б.М. «Передумови створення водневої інфраструктури для транспортної галузі». Частина 2. «Problems and prospects of automobile transport»). 14-15 квітня 2020 року: збірник наукових праць. / Міністерство освіти і науки України, Вінницький національний технічний університет [та інш.]. - Вінниця: ВНТУ, 2020. С. 31 - 45.
15. Ставицький О.В., Стадник Л.Г., Колесніков В.О. Концепція автомобіля майбутнього // Матеріали VI-ї Міжнародної науково-технічної інтернет-конференції «Проблеми та перспективи розвитку автомобільного транспорту», 12-13 квітня 2018 р., м. Вінниця. - С. 181 - 189.
16. Стадник Л.Д., Колесніков В.О. Сонячні батареї, як допоміжне обладнання для електромобілів // Матеріали VI-ї Міжнародної науково-технічної інтернет-конференції «Проблеми та перспективи розвитку автомобільного транспорту», 12-13 квітня 2018 р., м. Вінниця. - С. 198 - 202.
17. Цимбалюк П.Ю., Колесніков В.О. Системи зв'язку транспортних засобів // Матеріали VI-ї Міжнародної науково-технічної інтернет-конференції «Проблеми та перспективи розвитку автомобільного транспорту», 12-13 квітня 2018 р., м. Вінниця. - С. 204 - 208.
18. Ярченко Б.В., Стадник Л.Д., Колесніков В.О. «Нові технології в сучасних автомобілях» // Матеріали VI-ї Міжнародної науково-технічної інтернет-конференції «Проблеми та перспективи розвитку автомобільного транспорту», 12-13 квітня 2018 р., м. Вінниця. - С. 216 - 223.
19. Колесніков В.О., Шуліка С.О., Гаврилюк М.Р. «Мастильні матеріали для транспортної галузі та енергомашинобудування. Частина 2. Приклади випробувань. Матеріали VIII-ої міжнародної науково-практичної інтернет-конференції 86 «Проблеми і перспективи розвитку автомобільного транспорту». 14-15 квітня 2020 року: збірник наукових праць. / Міністерство освіти і науки України, Вінницький національний технічний університет [та інш.]. - Вінниця: ВНТУ, 2020. С. 179 - 189.
20. Olexiy Balitskii, Valerii Kolesnikov «Identification of Wear Products in the Automotive Tribotechnical System Using Computer Vision Methods, Artificial

Intelligence and Big Data» // 2019 XIth International Scientific and Practical Conference on Electronics and Information Technologies (ELIT) September 16 – 18, 2019, Lviv, Ukraine. P. 24 - 27.

21. Toyota Prius. URL:CC BY-SA 3.0, https://commo ns.wikimedia .org/w/index.ph p?curid=141028.
22. Car of the year Japan. URL:http:/ /www.jcoty .org/record /coty2009/.
23. North American Car, Utility and Truck of the Year Awards. URL:https://northamericancaroftheyear.org/winners-of-the-2019-north-american-car-utility-and-truck-of-the-year-announced/.
24. Риб'янець С. Р.; Бахмут М. І.; Колесніков В. О. Приклади застосування адитивних технологій в автомобілебудуванні. X-та міжн. науково-практичн. конф., 14- 15 квітня 2022 р.: матеріали. Вінниця: ВНТУ, 2022. С. 247–253.
25. Колесніков Валерій Олександрович, Колеснікова Єлизавета Борисівна. Перспективи використання технологій ігрового рушія Unreal Engine 5 в моушн дизайне. Актуальні питання, проблеми та перспективи розвитку науки та освіти: І Всеукраїнська міждисциплінарна науково-практич конф., 27-28 квітня 2022 р. Полтава: матеріали. Вид-во ДЗ «Луганський національний університет імені Тараса Шевченка», 2022. С. 17–20.
26. Верецун А. В., Ануфрієв В. А., Колесніков В. О. Висвітлення деяких недоліків та переваг гібридних та водневих автомобілів. Проблеми і перспективи розвитку автомобільного транспорту: XI-та міжн. науковопрактичн. конф., 13-14 квітня 2023 р.: матеріали. Вінниця: ВНТУ, 2023. С. 71-74.
27. Колесніков В. О., Балицький О. І., Гаврилюк М. Р., Іваськевич Л. М. Застосування комп’ютерного програмного комплексу для візуалізації шорсткості поверхні деталей в транспортній галузі та енергомашинобудуванні. Проблеми і перспективи розвитку автомобільного транспорту: XI-та міжн. науково-практичн. конф., 13–14 квітня 2023 р.: матеріали. Вінниця: ВНТУ, 2023. С. 179–184. ISBN 978-966-641-929-6.

28. Конверсія легкового автомобіля в гібридний / О.В. Бажинов, В.Я. Двадненко, М. Хакім; під. ред. О.В. Бажинова – Харків: ХНАДУ, 2014 – 160 с.
29. Мауш Хаким. Гібридна силова установка конверсійного автомобіля / Мауш Хакім // Матеріали науково – практичної конференції «Інформаційні технології і мехатроніки» 15 квітня 2014 р. – Харків. – С. 81 – 82
30. Бізнес з прокату автомобілів. [Електронний ресурс] / PRO-BIZNES – Режим доступу:– Дата звернення: 10.10.2020 р. – Загол. з екрану.
31. Цивільний кодекс України / Глава 58 НАЙМ (ОРЕНДА). [Електронний ресурс] / Законодавство України – Режим доступу: www / URL: https://zakon.rada.gov.ua/laws/show/435-15/page13 – Дата звернення: 21.10.2020 р. – Загол. з екрану.
32. Джонс М. Т. «Принципи роботи рекомендованих механізмів». [Електронний ресурс]. IBM developerWorks, 2014. URL: https://www.ibm.com/developerworks/ru/library/os-recommender1/.
33. Пономарев А. В. Обзор методов учета контекста в системах коллаборативной фильтрации // Труды СПИИРАН, 2013. № 7 (30), С. 169-188.
34. Михайловский Н. «Анотомія рекомендованих сервісів» [Електронний ресурс]. Centrobit, 2016.
35. Car Recommendation System Version 1.0. [Електронний ресурс] / Nars cars – Режим доступу: www / URL: https://seniorproject.cis.fiu.edu/seniorprojects/car-recommendation-system/ – Дата звернення: 08.10.2020 р. – Загол. з екрану.
36. Recombee. [Електронний ресурс] / Nars cars – Режим доступу: www / URL: https://www.recombee.com/product-recommendations.html – Дата звернення: 10.10.2020 р. – Загол. з екрану.
37. Amazon Personalize. [Електронний ресурс] / Nars cars – Режим доступу: www / URL: https://aws.amazon.com/personalize/ – Дата звернення: 10.10.2020 р. – Загол. з екрану.
38. Panagiotis Symeonidis, Andreas Zioupos. Matrix and Tensor Factorization. Techniques for Recommender Systems /P. Symeonidis – Free University of Bozen-Bolzano, Italy: «Springer», 2016. – 101 с.
39. P. Lops, M. Gemmis, G. Semeraro Recommender Systems Handbook Springer-Verlag, 2017.
40. C. C. Aggarwal, Recommender Systems: The Textbook. Springer. С: 837, 2016
41. Melville P., Sindhwani V. Recommender systems; Encyclopedia of Machine Learning, 2016

42. Stefan nadschläger, Analysis of GoF Design Patterns used in Knowledge Processing Systems, Institute for Application Oriented Knowledge Processing. [Текст] /Stefan nadschläger, Josef küng. – Linz: Johannes Kepler University, 2017 – 22 с.
43. Oliver Theobald. Machine Learning: Make Your Own Recommender System (MachineLearningFromScratchBook3).
44. Web Services: A Comparison of Soap and Rest Services. [Текст] / Festim Halili, Erenis Ramadani – Tetova: University of Tetova. 2018 – 175-183 с.
45. Comparison of Programming Languages: Review. [Текст] / Naveen Reddy K P, Geyavalli .Y, Sujani .D, Rajesh S M – International Journal of Computer Science & Communication. 2018 – 113-122 с.
46. Top Java Application Servers: Tomcat vs. Jetty vs. GlassFish vs. WildFly. [Електронний ресурс] / Stackify – Режим доступу: www / URL: https://stackify.com/tomcat-vs-jetty-vs-glassfish-vs-wildfly/ – Дата звернення: 01.12.2020 р. – Загол. з екрану.
47. Java web design frameworks: Review of java frameworks for web applications. [Текст] / Dr. Tejinder Singh – International Journal of Advance Research In Science And Engineering. 2015 – 592-595 с.
48. Tu Nguyen. Java Spring Framework in developing the Knowledge Article Management application. [Текст] / Tu Nguyen. – Helsinki: Helsinki Metropolia University of Applied Sciences, 2018 – 46 с.
49. Stefan nadschläger, Analysis of GoF Design Patterns used in Knowledge Processing Systems, Institute for Application Oriented Knowledge Processing. [Текст] /Stefan nadschläger, Josef küng. – Linz: Johannes Kepler University, 2017 – 22 с.
50. Sitnikov D. Assessment of extended aggregated association rules / Sitnikov D., Ryabov O., Titova O., Kovalenko A. // Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT’2018, 24 27 May, 2018, Kyiv, Ukraine, page 95 99. (IEEE Xplore, Scopus)
51. Порівняння засобів створення пайплайнів Apache для розподіленої обробки даних. [Текст] / Анна Богун. – Дніпро: V Всеукраїнська науково- практична конференція MEICS-2020, 2020 – 104-105 с.