

Міністерство освіти і науки України  
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук  
(повна назва)

Кафедра Системотехніки  
(повна назва)

**АТЕСТАЦІЙНА РОБОТА**  
**Пояснювальна записка**

другий (магістерський)

(освітньо-кваліфікаційний рівень)

ГЮИК.501610.001 ПЗ

(позначення документа)

Розробка методу вилучення знань для визначення уподобань клієнтів сервісу  
оренди автомобілів

(тема)

Виконала: студентка 2 курсу, групи ІТІМ-19-1  
напряму підготовки (спеціальності) \_\_\_\_\_

6.050101 – Комп'ютерні науки

Інформаційні технології проектування

(шифр і назва напряму, спеціальності)

Богун А.Є.

(прізвище, ініціали)

Керівник проф. Ситніков Д.Е.

(прізвище, ініціали)

Рецензент доц. Яськов Г.М.

(прізвище, ініціали)

Допускається до захисту

Зав. кафедри СТ

\_\_\_\_\_ (підпис)

проф. Гребеннік І.В.

(прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

(назва вищого навчального закладу)

Факультет Комп'ютерних наук Кафедра Системотехніки

Напрямок 6.050101 – Комп'ютерні науки

Спеціальність Інформаційні технології проектування

Освітньо-кваліфікаційний рівень другий (магістерський)

**ЗАТВЕРДЖУЮ:**

Зав. кафедри СТ

проф. Гребеннік І.В.

" \_\_\_\_\_ " \_\_\_\_\_ 2020 р.

## ЗАВДАННЯ

### НА АТЕСТАЦІЙНУ РОБОТУ

студентові Богун Анні Євгенівні  
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка методу вилучення знань для визначення уподобань клієнтів сервісу оренди автомобілів

затверджена наказом по університету від " 2 " листопада 2020р. № 689 Ст

2. Термін здачі студентом закінченої роботи 18 грудня 2020 р.

3. Вихідні дані до роботи Функція: Розробка методу вилучення знань уподобань клієнства сервісу оренди автомобілів. Організація даних: база даних. Перелік використовуваних програмних засобів: ОС Microsoft Windows 10, інтегроване середовище програмування IntelliJ IDEA, браузер Google Chrome версії 75, MySQL Workbench, Rational Rose. Технічне забезпечення: комп'ютер для виконання тестів з підтримкою Java та зі встановленим браузером Google Chrome версії 41 та вище й не менш, ніж 1Гб оперативної пам'яті.

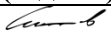
4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити)

4.1 Вступ. 4.2 Аналіз предметної області і постановка задачі. 4.3 Розробка методу вилучення знань уподобань клієнтів. 4.4 Проектування моделі даних для представлення вхідної інформації методу вилучення знань уподобань клієнтів. 4.5 Розробка програмної реалізації методу вилучення знань уподобань клієнтів. 4.6 Висновки.

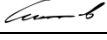
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, плакатів)


5.1 Діаграма прецедентів (1 аркуш формату А4). 5.2 Контексна діаграма IDEF0 (1 аркуш формату А4). 5.3 Схема алгоритму роботи методу вилучення знань уподобань клієнтів (1 аркуш формату А4). 5.4 Діаграма класів-сервісів (1 аркуш формату А4). 5.5 Модель даних (1 аркуш формату А4) .

**6. Консультанти з проекту із зазначенням розділів проекту, що їх стосуються**

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		(підпис)	(дата)
Спеціальна частина	проф. Ситніков Д.Е.		

7. Дата видачі завдання 10.09.2020

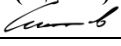
Керівник роботи  проф. Ситніков Д.Е.  
(підпис)

Завдання прийняв до виконання  Богун А.Є.  
(підпис студента)

**КАЛЕНДАРНИЙ ПЛАН**

Пор. №	Назва етапів атестаційної роботи	Термін виконання етапів роботи	Примітка
1.	Отримання завдання на атестаційне проектування	10.09.2020	
2.	Аналіз завдання, пошук літератури та аналогів з теми магістерської атестаційної роботи	12.09.2020	
3.	Опрацювання літератури та аналіз проблеми інформаційного перевантаження користувачів	10.10.2020	
4.	Аналіз існуючих алгоритмів для прогнозування уподобань користувачів	26.10.2020	
5.	Розробка алгоритму вилучення знань уподобань клієнтів	04.11.2020	
6.	Вибір програмних, мовних та технічних засобів для розробки системи	06.11.2020	
7.	Розробка програмної реалізації методу вилучення знань	07.11.2020	
8.	Аналіз результатів, отриманих за допомогою програмного засобу	11.11.2020	
9.	Розробка навантажених тестів для розробленого програмного продукту	20.11.2020	
10.	Оформлення пояснювальної записки та програмної документації	01.12.2020	
11.	Оформлення графічної частини та презентаційних матеріалів комп'ютерного захисту	08.12.2020	
12.	Представлення на рецензування	13.12.2020	
13.	Представлення атестаційної роботи в ДЕК	18.12.2020	

Студент  Богун А.Є.

Керівник роботи  проф. Ситніков Д.Е.  
(підпис)

## РЕФЕРАТ

Пояснювальна записка до атестаційної роботи містить: 107 сторінок, 52 рисунка, 7 таблиці, 7 формул, 25 джерела. Графічна частина атестаційної роботи містить 5 плакатів.

Мета атестаційної роботи – дослідження та розробка методу вилучення знань для визначення уподобань клієнтів сервісу оренди автомобілів та аналіз методів вирішення проблеми визначення рекомендацій.

Предметом розробки 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.

Результатом атестаційної 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, ОRENDA, РЕКОМЕНДАЦІЙНА СИСТЕМА, ПЕРСОНАЛІЗАЦІЯ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ.

## ABSTRACT

The explanatory note to the attestation work contains: 107 pages, 52 figures, 7 tables, 7 formulas, 25 sources. The graphic part of the certification work contains 5 posters.

The purpose of the attestation work is research and development of a method of knowledge extraction to determine the preferences of car rental service customers and analysis of methods for solving the problem of determining recommendations.

The subject of the development is a method that is a variant of the recommendation system, which will allow customers to select cars for rent, and the car rental administrator - to make the service more convenient, as well as increase profits through better selection of vehicles for customers.

The result of the certification work is a developed method of extracting knowledge of the preferences of customers of the car rental service, as well as its software implementation.

The scope of the system is a wide range of users, the system is focused mainly on commercial use for the finished fleet of cars and rental services.

CAR RENTAL, RENTAL, RECOMMENDATION SYSTEM,  
PERSONALIZATION, COLLABORATIVE FILTRATION.

## ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	8
ВСТУП.....	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ.....	11
1.1 Аналіз діяльності підприємства .....	11
1.2 Аналіз проблеми інформаційного перевантаження користувачів.....	12
1.3 Актуальність роботи розробки методу вилучення знань.....	13
1.4 Порівняння аналогів розроблюваного методу вилучення знань .....	14
1.5 Опис об'єкта, для якого необхідна рекомендаційна система .....	15
1.6 Постановка задачі дослідження.....	18
2 РОЗРОБКА МЕТОДУ ВИЛУЧЕННЯ ЗНАНЬ УПОДОБАНЬ КЛІЄНТІВ.....	20
2.1. Представлення методу вилучення знань як варіанту рекомендаційної системи .....	20
2.2 Впровадження в метод вилучення знань алгоритму колаборативної фільтрації.....	21
2.2.1 Порівняння та вибір методу колаборативної фільтрації для розроблюваного методу.....	21
2.2.2 Порівняння та аналіз алгоритмів колаборативної фільтрації на основі користувачів та об'єктів та вибір одного з них .....	24
2.2.2.1 Аналіз алгоритму колаборативної фільтрації на основі користувачів .....	24
2.2.2.2 Аналіз алгоритму колаборативної фільтрації на основі об'єктів (Slope One) та обґрунтування його переваг .....	27
2.3 Впровадження алгоритму рекомендаційних систем на основі вмісту у розроблюваний метод вилучення знань.....	29
2.4 Опис алгоритму розробленого методу вилучення знань уподобань клієнтів	32
3 ПРОЕКТУВАННЯ МОДЕЛІ ДАНИХ ДЛЯ ПРЕДСТАВЛЕННЯ ВХІДНОЇ ІНФОРМАЦІЇ МЕТОДУ ВИЛУЧЕННЯ ЗНАНЬ УПОДОБАНЬ КЛІЄНТІВ .....	34
3.1 Обґрунтування вибору СУБД для проектування бази даних .....	34
3.2 Побудова схеми моделі даних .....	34

3.3 Розробка підтримки цілісності даних .....	40
3.4 Масштабування баз даних .....	41
3.5 Порівняльний аналіз варіантів реалізації баз даних та вибір однієї із них....	42
3.6 Генерація тестових даних .....	43
<b>4 РОЗРОБКА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МЕТОДУ ВИЛУЧЕННЯ ЗНАНЬ УПОДОБАНЬ КЛІЄНТІВ.....</b>	<b>52</b>
4.1 Аналіз і вибір методів реалізації .....	52
4.1.1 Аналіз і обґрунтування вибору типу архітектури додатку .....	52
4.1.1.1 Порівняння типів архітектури додатку .....	52
4.1.1.2 Обґрунтування вибору типу архітектури додатку.....	54
4.1.2 Аналіз і обґрунтування вибору мови програмування .....	54
4.1.2.1 Порівняння мов програмування.....	54
4.1.2.2 Обґрунтування вибору мови програмування Java.....	57
4.1.3 Аналіз і обґрунтування вибору сервера .....	57
4.1.3.1 Порівняння серверів .....	57
4.1.4 Аналіз і обґрунтування вибору фреймворку веб-дизайну Java .....	59
4.1.4.1 Порівняння фреймворків веб-дизайну Java.....	59
4.1.4.2 Обґрунтування вибору фреймворку Spring.....	60
4.2 Опис основних технологій частини програмного додатку.....	60
4.3 Опис запрограмованого методу.....	68
4.4 Тестування розробленого програмного забезпечення .....	73
<b>ВИСНОВКИ .....</b>	<b>76</b>
<b>ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ .....</b>	<b>77</b>
<b>ДОДАТОК А Графічний матеріал .....</b>	<b>1</b>
<b>ДОДАТОК Б Текст програми.....</b>	<b>88</b>
<b>ДОДАТОК В SQL-скрипт бази даних .....</b>	<b>100</b>
<b>ДОДАТОК Г Відомість атестаційної роботи .....</b>	<b>104</b>
<b>ДОДАТОК Д Специфікація.....</b>	<b>106</b>

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ,  
СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

СУБД – система управління базами даних

SQL – Structured Query Language (мова структурованих запитів)

HTTP – HyperText Transfer Protocol

API – Application Programming Interface

JVM – Java Virtual Machine

БД – база даних

CASE – Computer Aided Software Engineering (сукупність методів і засобів автоматизованого проектування інформаційних систем)

ERD – Entity Relationship Diagram (модель даних, що дозволяє описувати концептуальні схеми предметної області)

URL – Uniform Resource Locator (уніфікований покажчик інформаційного ресурсу)

DAO – Data Access Object

API – Application Programming Interface

JSON – JavaScript Object Notation

CRUD – create read update delete, 4 основні функції управління даними «створення, читання, оновлення і видалення».

КФ – колаборативна фільтрація

SVM – метод опорних векторів (support vector machine)

VoD – відео за запитом (Video on Demand)

DS – Data Science

## 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 користувача.

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 атестаційної 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.

## 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

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 доросліше 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'язково мають 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 автомобілів, 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 підлаштування 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 сутностей;
- 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* ([aws.amazon.com/personalize/](https://aws.amazon.com/personalize/)) – 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 прямиий 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 наслідуює можливості незареєстрованого користувача.

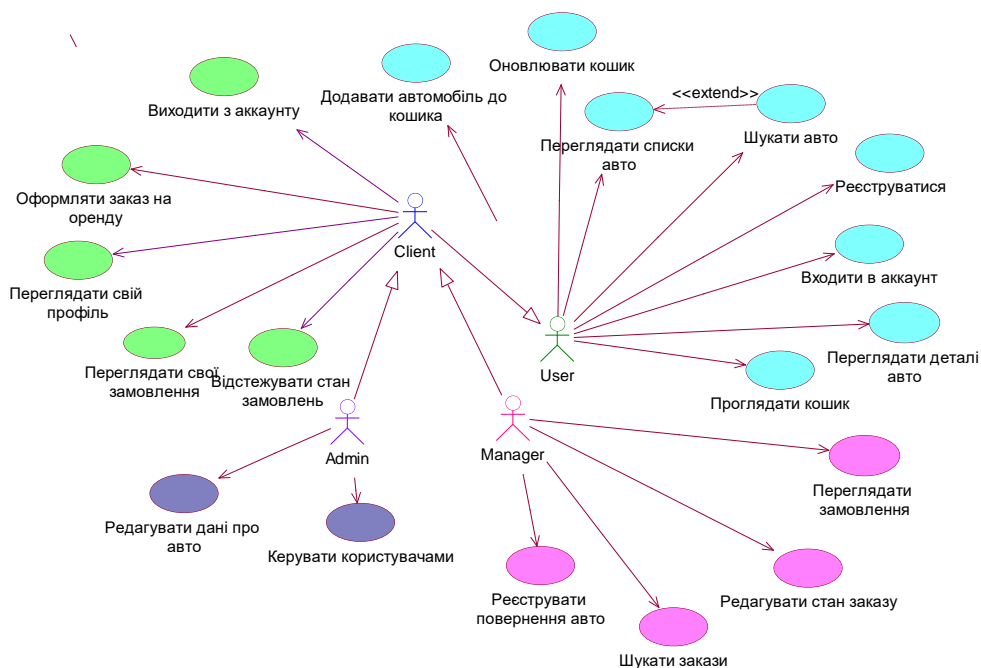


Рисунок 1.1 – Діаграма варіантів використання для сервісу прокату автомобілів

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.

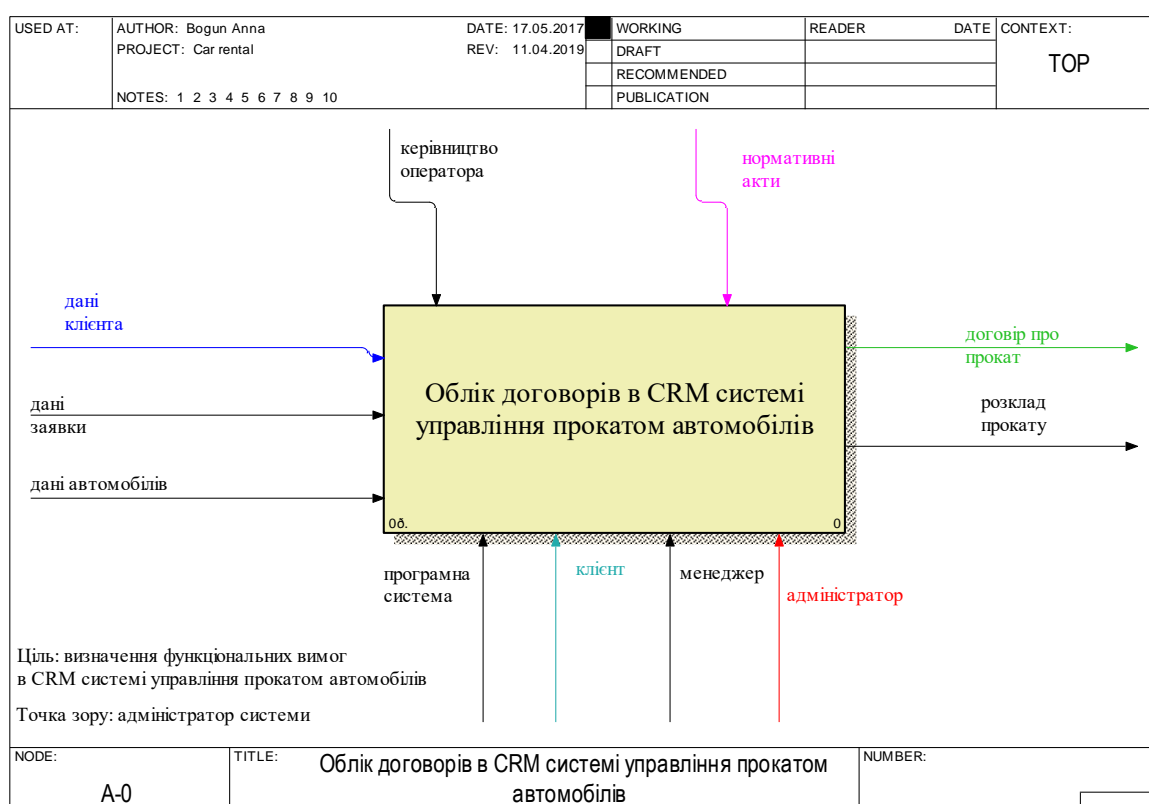


Рисунок 1.2 – Контексна діаграма IDEF0 для сервісу прокату автомобілів

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. Створені рекомендації будуть корисні не тільки клієнтам, а також менеджерам та адміністраторам сервісу.

## 1.6 Постановка задачі дослідження

Постановка задачі 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 задачі дослідження зводиться до наступного:

- проведення аналізу методів побудови рекомендаційних систем;
- розробка методу вилучення знань для визначення уподобань клієнтів прокату автомобілів 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 рекомендувати».

Формалізований опис задачі можна представити формулою:

$$Z = \text{recommendations}(p, K, O, C, n)$$

$$p \in P \quad (1.1)$$

$$K \subseteq O$$

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  $p$ .

$S$  – множина всіх моделей автомобілів, доступних для прокату в сервісі прокату. Кожен елемент множини  $S$  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, лошади́на 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 (рекомендовані або 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).

## 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 контенту (змісту).

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

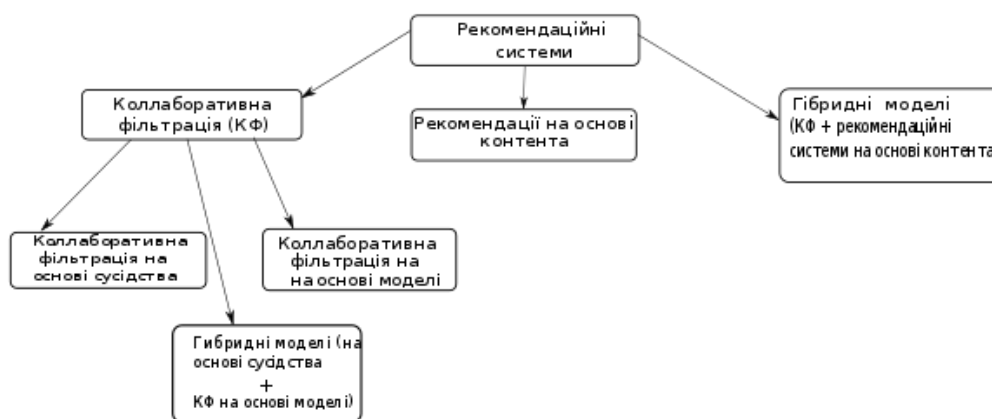


Рисунок 2.1 – Типи методів побудування рекомендаційних систем

## 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 на прокат 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).

Категорія 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, коли даних 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 довіри КФ	* 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

Продовження таблиці 2.1

1	2	3	4
	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 посиленням вмістом	* покращена ефективність прогнозування	* потрібна зовнішня інформація, яка зазвичай недоступна
	гібридна КФ, що поєднує алгоритми КФ на основі пам'яті та моделі , 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 основі користувачів 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).

		Об'єкти					
		1	2	...	$i$	...	$m$
Користувачі	1	5	3		1	2	
	2		2				4
	:			5			
	$u$	3	4		2	1	
	:					4	
	$n$			3	2		
$a$		3	5		?	1	

Рисунок 2.2 – Матриця оцінок

Нехай ім'я клієнт  $a$ . Задача знаходиться в тому, щоб передбачити, яку оцінку створив клієнт  $i$  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  $u$  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  $i$  "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:

$$\text{sim}(u, a) = \frac{\sum_{i=1}^m r_{a,i} \cdot r_{u,i}}{\sqrt{\sum_{i=1}^m r_{a,i}^2} \cdot \sqrt{\sum_{i=1}^m r_{u,i}^2}} \quad (2.1)$$

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

$u$  – рядок,  $i$  – стовець,  $\text{sim}(u, a)$  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:

$$\text{sim}(u, a) = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a) \cdot (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2} \cdot \sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}} \quad (2.2)$$

Тут  $I$  – множина об'єктів, які оцінив як клієнт  $a$ , так і клієнт  $u$ .  $\bar{r}_a$  і  $\bar{r}_u$  – 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 суворо, 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  $k$  користувачів, найближчі близькі до  $a$ .

*Третій крок.* Маючи множину  $K$  близьких користувачів обчислюється оцінка, яка надіслана користувачем об'єкту  $i$ . Помітно, що розглядаються лише користувачі, які оцінюють об'єкт  $i$ . Необхідна оцінка обчислюється за формулою:

$$r_{a,i} = \bar{r}_a + \frac{\sum_{u \in K} (r_{u,i} - \bar{r}_u) \times \text{sim}(a, u)}{\sum_{u \in K} |\text{sim}(a, u)|} \quad (2.3)$$

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

Чим ближче користувач  $u$  до користувача  $a$  (відповідно до розміру близькості  $\text{sim}(a, u)$ , 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 (Slope One) 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  $i$ , для якого 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):

$$sim(u, a) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i) \cdot (r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2 \cdot \sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}} \quad (2.4)$$

Тут підсумовування ведеться по множині  $U$  користувачів, які оцінили об'єкти  $i$  та  $j$ .  $\bar{r}_i$  – 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  $a$ :

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

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 частоту 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 прогноуються 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.

Враховує всі переваги логічно використовувати у методі вилучення знань алгоритму колаборативної фільтрації на основі об'єктів (Slope One). Цей алгоритм буде 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 у розроблюваний 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.

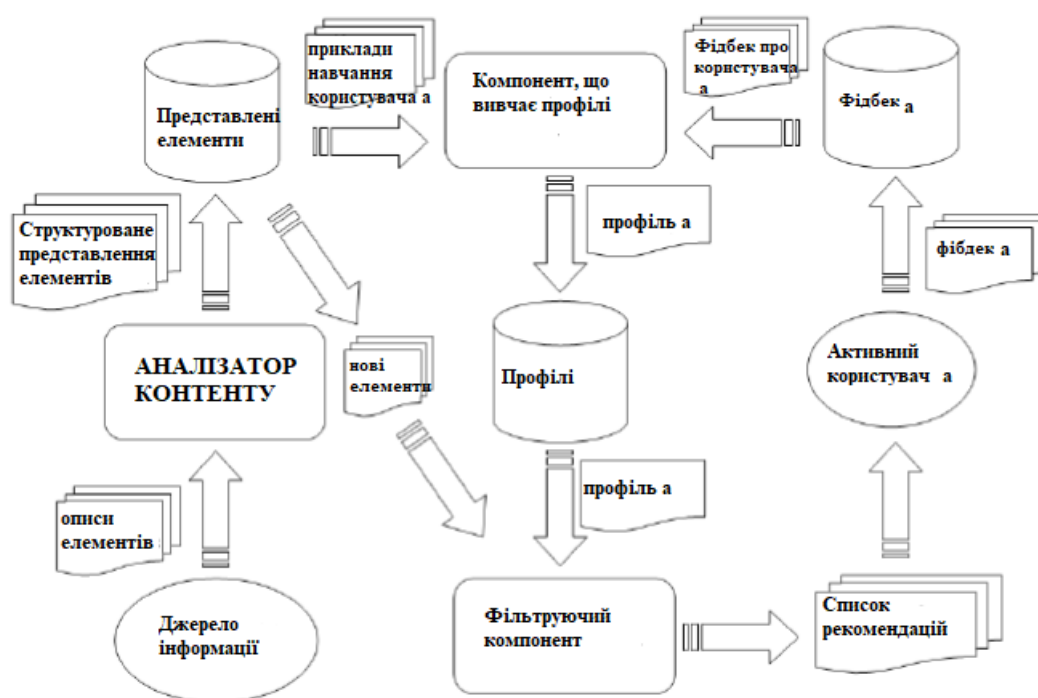


Рисунок 2.3 – Архітектура високого рівня рекомендатора на основі вмісту

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 обирає 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].



Рисунок 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 2.6.

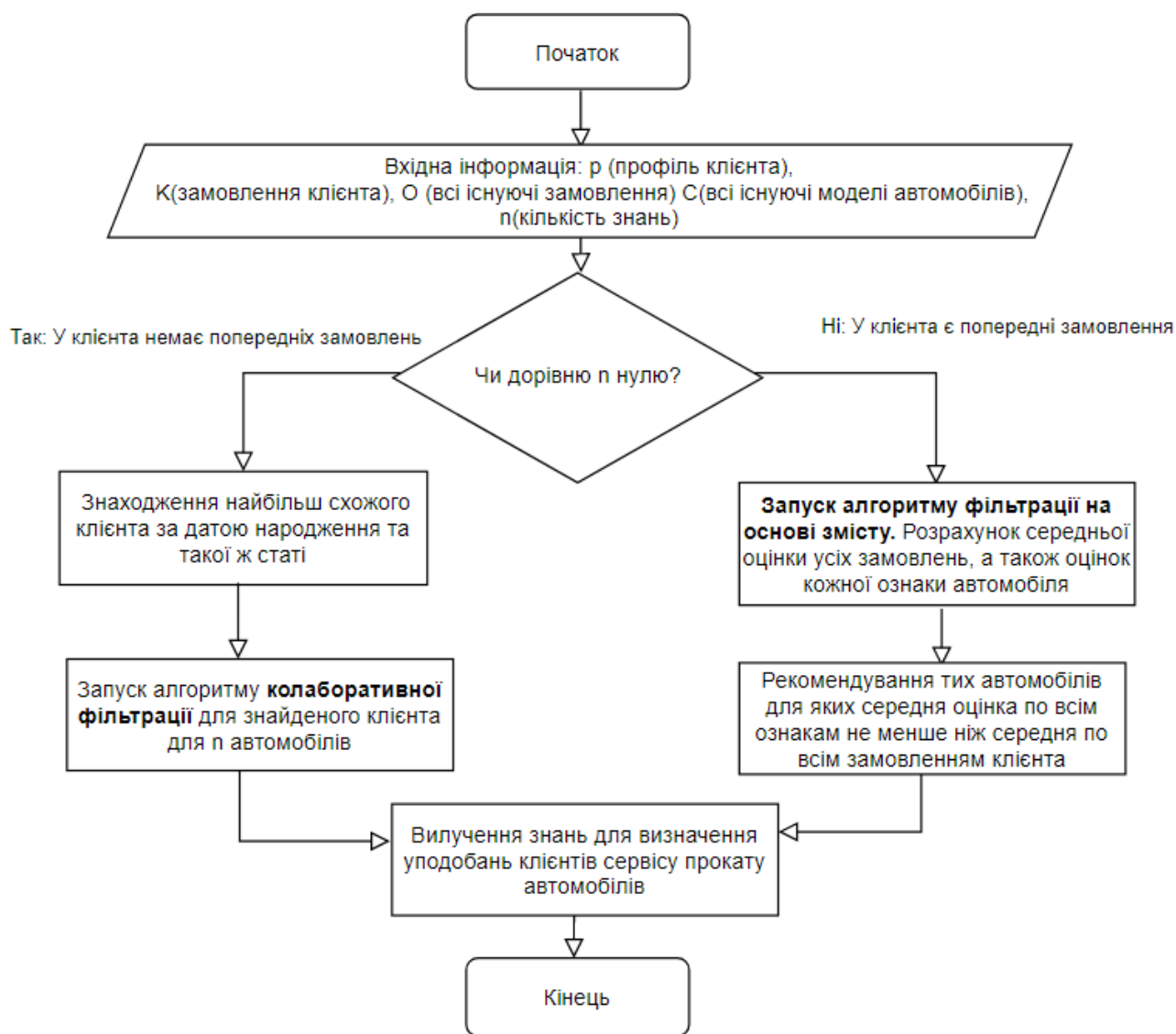


Рисунок 2.5 – Схема алгоритму роботи методу вилучення знань щодо уподобань клієнтів

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.

## 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 підтримуються 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.

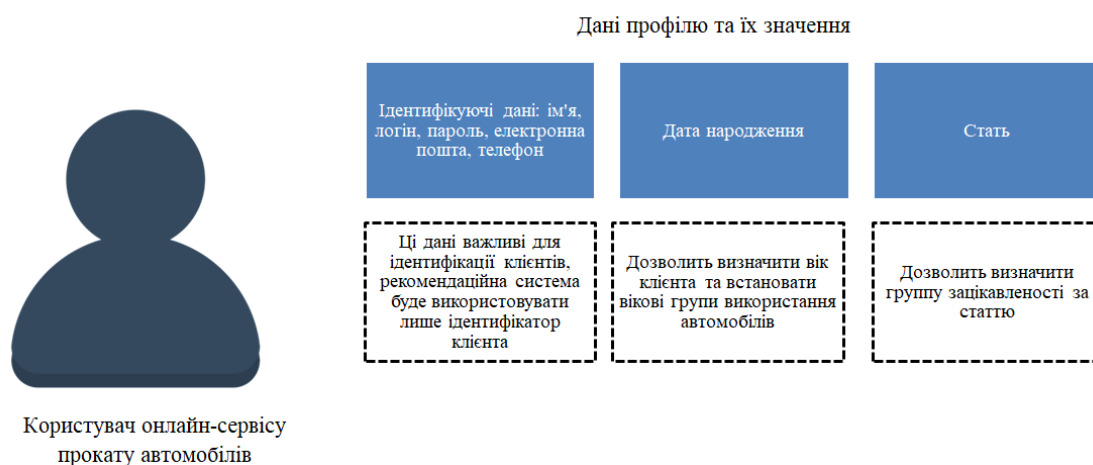


Рисунок 3.1 – Схема даних клієнта

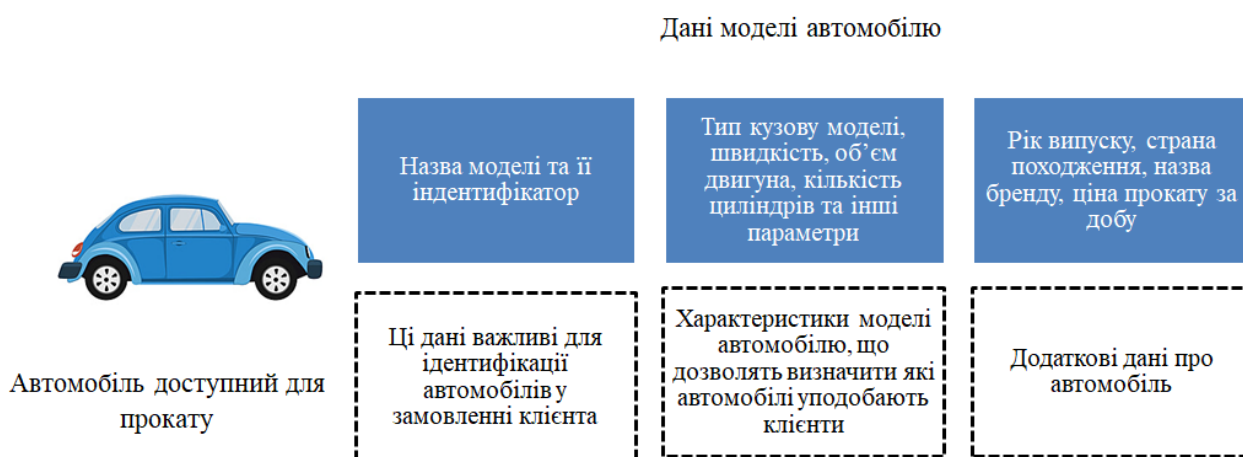


Рисунок 3.2 – Схема даних моделі автомобілю

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

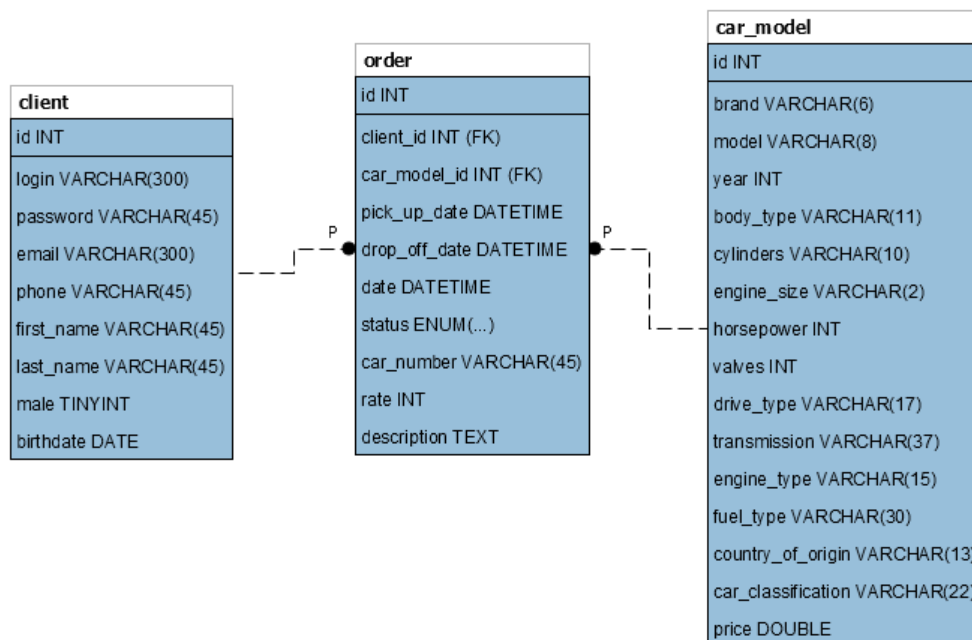


Рисунок 3.3 – Логічна модель даних програмної системи

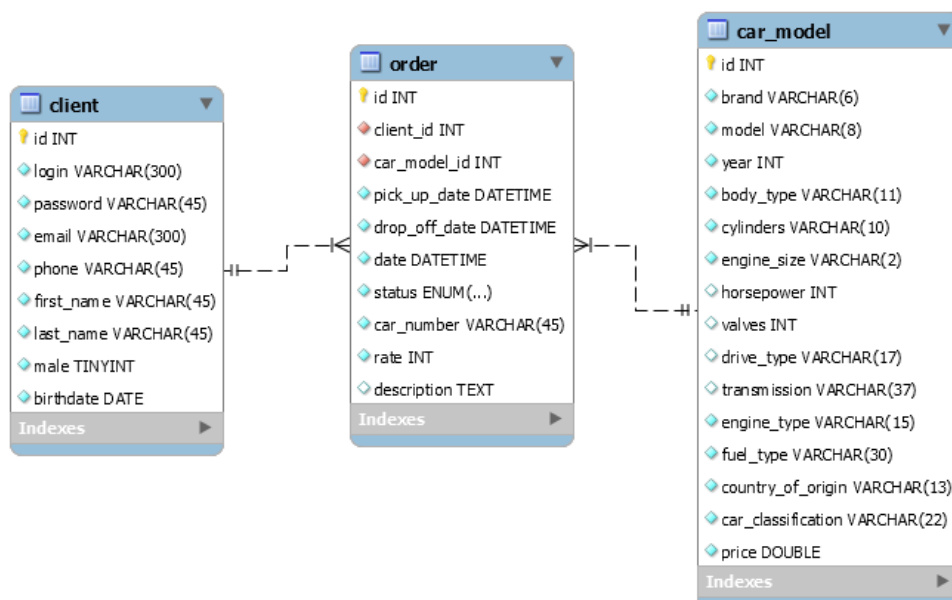


Рисунок 3.4 – Фізична модель даних програмної системи

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

### 3.2 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 детальний опис цих таблиць, а саме: назви атрибутів та їх опис, а також приклад даних.

Таблиця 3.1 – Відомості про типи таблиць-сутностей

Імя таблиці	Опис
car_model	Опис деякої моделі автомобілю. Наприклад, Mazda Atenza Sport 2.0. Містить дані про тип кузова 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 замовлення та інше.
client	Сутність користувача системи, містить всі дані про нього, такі як імя, 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

Продовження таблиці 3.2

1	2	3	4
4	year	Рік випуску моделі автомобілю	1995, 2000
5	body_type	Тип кузова	Sedan, Wagon, Hatchback, Coupe
6	cylinders	Кількість циліндрів	I4, I6
7	engine_size	Розмір двигуна	1, 2
8	horsepower	Лошадина сила	170, 240, 138
9	valves	Кількість клапанів	16, 24
10	drive_type	Тип приводу	rear wheel drive, all wheel drive, four wheel drive, front wheel drive
11	transmission	Передача	5-speed manual, 4-speed automatic, continuously variable-speed automatic
12	engine_type	Тип двигуна	gas, hybrid, diesel, flex-fuel (FFV)
13	fuel_type	Тип палива	regular unleaded, premium unleaded (required), flex-fuel (unleaded/E85), diesel fuel, premium unleaded (recommended)
14	country_of_origin	Країна походження	Germany, United States, Japan
15	car_classification	Класифікація автомобілів	Compact car, Full-size pickup truck
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

Продовження таблиці 3.2

1	2	3	4
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	Nina
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_id	Ідентифікатор моделі автомобілю	7, 8, 45312
3	pick_up_date	Дата взяття автомобілю у прокат	2020-07-17 14:32:34, 2020-03-13 07:02:42
4	drop_off_date	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	AX 4321 AB
7	rate	Оцінка замовлення клієнтом від 1 до 5	1, 2, 5

### 3.3 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 оновляти 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 3.2 – 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

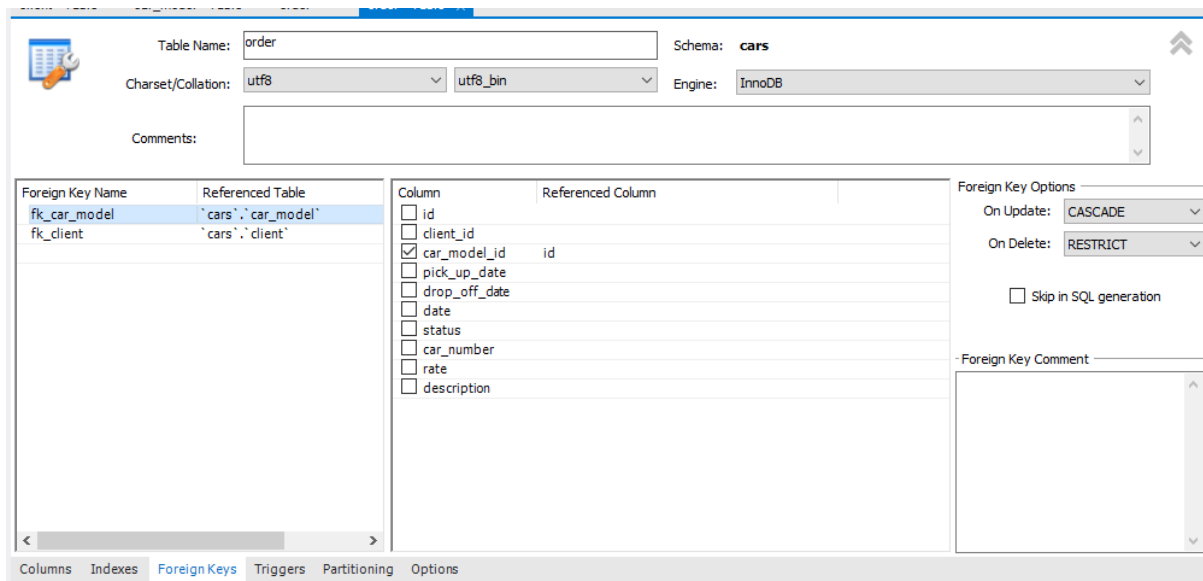


Рисунок 3.5 – Створення зовнішніх ключів для таблиці order

### 3.4 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 що усі таблички лежать 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 розподілено 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. Foreign data wrapper. Далі треба створити віддалений сервер та вказати його як foreign data wrapper.

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

### 3.5 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 InnoDB. 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 підвищують надійність 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 посилювальної цілісності і не потребує написання

тригерів для цього. InnoDB більш популярний тип, він також має гарну швидкість і є дуже безпечним.

### 3.6 Генерація тестових даних

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

На рисунку 3.6 продемонстровано знайдену базу даних з сайту [www.teoalida.com/cardatabase](http://www.teoalida.com/cardatabase).

The screenshot displays the homepage of 'Teoalida's Car Database', which offers a wide range of automobile data for download in Excel, CSV, and SQL formats. The site is organized into a grid of database categories, each with a thumbnail image of a data spreadsheet and a brief description of its contents.

Key databases listed include:

- American Year Make Model Trim Specs:** 1990-present, 63 makes, 8000+ model years, 60000+ model trims, 96 columns of specifications.
- European German car database:** 1946-present, 93 makes, 5000+ models, 130000+ versions, 239 columns of specifications.
- Middle East (GCC) Car Database:** 1990-present, 90+ makes, 900+ models, 12000+ engines, 26 columns of specifications.
- India Car Database:** 1990s-present, 50 makes, 700+ models, 5000+ variants, 188 columns of specs and features.
- Japan Car Database:** 1986-present, 74 makes, 1400+ models, 90000+ versions, 55 columns of specifications.
- Car Models List:** Worldwide list, with mentions for car sold in Europe and America, 1945-present, 210+ brands, 5700+ models.
- United Kingdom car database:** 1980-present, 103 makes, 2500+ models, 80000+ versions, 52 columns of specifications.
- American Year Make Model:** Last release: 29 April 2020, Download sample XLSX, CSV, SQL. All cars, trucks, vans and SUVs sold in United States & Canada, 1945-present, 98 brands.

The website also features a navigation menu with options like 'By type', 'Worldwide', 'America', 'Europe', 'Middle East', 'Asia', 'Motorcycles', 'Other databases', 'FAQ', 'Custom', 'Web scraping', 'About', and 'Contact'.

Рисунок 3.6 – Teoalida's Car Database. The most updated automobile database

Ці дані «брудні», вони містять багато інформації, не потрібної для розробки методу вилучення знань щодо рекомендацій. Імпортуємо ці дані у тимчасову таблицю cars\_sample (рисунок 3.7).

1 • SELECT \* FROM car\_data.cars\_sample;

ID	Make	Model	Year	Trim (description)	MSRP	Invoice	Colors exterior	Colors interior	Body type	Total seating	L
213	BMW	3 Series	1990	325i 2dr Convertible					Convertible	1:	
216	BMW	3 Series	1990	325i 4dr Sedan					Sedan	1:	
222	BMW	3 Series	1990	325iX 4dr Sedan AWD					Sedan	1:	
218	BMW	3 Series	1990	325is 2dr Coupe					Coupe	1:	
215	BMW	3 Series	1990	325i 2dr Coupe					Coupe	1:	
220	BMW	3 Series	1990	325iX 2dr Coupe AWD					Coupe	1:	
231	BMW	3 Series	1990	M3 2dr Coupe					Coupe	1:	
160	BMW	3 Series	1992	318i 4dr Sedan			Iceland Green Metallic();Alpine White II();Broca...		Sedan	1:	
163	BMW	3 Series	1992	325i 4dr Sedan			Laguna Green Metallic();Iceland Green Metallic();...		Sedan	1:	
158	BMW	3 Series	1992	318i 2dr Convertible			Glacier Blue Metallic();Diamond Black Metallic();I...		Convertible	1:	
162	BMW	3 Series	1992	325i 2dr Convertible			Sterling Silver Metallic();Mauritius Blue Metallic();...		Convertible	1:	
165	BMW	3 Series	1992	318is 2dr Coupe			Calypso Red Metallic();Sterling Silver Metallic();I...		Coupe	1:	
167	BMW	3 Series	1992	325is 2dr Coupe			Calypso Red Metallic();Brilliant Red();Lazure Blu...		Coupe	1:	
104	BMW	3 Series	1994	318i 2dr Convertible			Samoa Blue Metallic();Jet Black();Samana Beige ...		Convertible	1:	
108	BMW	3 Series	1994	325i 2dr Convertible			Brocade Red Metallic();Jet Black();Brilliant Red();...		Convertible	1:	
100	BMW	3 Series	1994	318i 4dr Sedan			Samoa Blue Metallic();Brilliant Red();Alpine Whit...		Sedan	1:	
102	BMW	3 Series	1994	325i 4dr Sedan			Samoa Blue Metallic();Morea Green Metallic();Al...		Sedan	1:	
106	BMW	3 Series	1994	318is 2dr Coupe			Oxford Green Metallic();Calypso Red Metallic();...		Coupe	1:	
110	BMW	3 Series	1994	325is 2dr Coupe			Mauritius Blue Metallic();Arctic Gray Metallic();Ar...		Coupe	1:	
34	BMW	3 Series	1996	318i 4dr Sedan					Sedan	1:	
42	BMW	3 Series	1996	328i 4dr Sedan					Sedan	1:	
32	BMW	3 Series	1996	318i 2dr Convertible			Bright Red();Jet Black();Alpine White();Samoa B...		Convertible	1:	
38	BMW	3 Series	1996	328i 2dr Convertible			Bright Red();Jet Black();Alpine White();Boston ...		Convertible	1:	
45	BMW	3 Series	1996	318i 2dr Coupe			Jet Black();Calypso Red Metallic();Bright Red();...		Coupe	1:	

Рисунок 3.7 – Таблиця cars\_sample з брудними даними моделей автомобілів

Видалимо непотрібні колонки, переменуємо інші щоб отримати таблицю car\_model. Ціну за прокат на добу сгенеруємо по формулі (формула знайдена досвідним шляхом і використовується для генерації тестових даних):

$$Price = engine\_size \cdot 10 + \frac{horsepower}{10} \quad (3.1)$$

```
1 update car_model set price = engine_size*10+horsepower/10
```

Рисунок 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.

The screenshot shows the 'generatedata.com' interface. At the top, there's a logo and navigation links for 'Generate', 'About', 'News', and 'Donate'. The table name 'client' is entered in a search box. Below it, 'COUNTRY-SPECIFIC DATA' is set to 'United States'. The 'DATA SET' section contains a table with 8 columns: Order, Table Column, Data Type, Examples, Options, Help, and Del. The rows are configured as follows:

Order	Table Column	Data Type	Examples	Options	Help	Del
1	login	Fixed Number of Words	No examples available.	Generate # 1 words	?	<input type="checkbox"/>
2	password	Fixed Number of Words	No examples available.	Generate # 1 words	?	<input type="checkbox"/>
3	email	Email	No examples available.	No options available.	?	<input type="checkbox"/>
4	phone	Phone / Fax	UK	(01xxxx) xxxxx (01xxx) xxxxxx (01x1) xxx xxx	?	<input type="checkbox"/>
5	first_name	Names	Alex (any gender)	Name	?	<input type="checkbox"/>
6	last_name	Names	Smith (surname)	Surname	?	<input type="checkbox"/>
7	male	Boolean	0 or 1	0 1	?	<input type="checkbox"/>
8	birthdate	Date	MySQL datetime	From: 10/01/1960 To: 10/01/2000 Format code: Y-m-d	?	<input type="checkbox"/>

At the bottom of the table, there are 'Add' and 'Row(s)' buttons.

Рисунок 3.9 – Генерація даних для таблиці client

The screenshot shows the 'EXPORT TYPES' section of the website. It has tabs for CSV, Excel, HTML, JSON, LDIF, Programming Language, SQL, and XML. The 'SQL' tab is selected. The configuration is as follows:

- Database table name: user
- Database Type: MySQL
- Misc Options:
  - Include CREATE TABLE query
  - Include DROP TABLE query
  - Enclose table and field names with backquotes
- Statement Type:  INSERT  INSERT IGNORE  UPDATE
- INSERT batch size: 10
- Primary Key:  None  Add default auto-increment column

At the bottom, there are options for 'Generate' (100 rows), 'Generate in-page' (selected), 'New window/tab', 'Prompt to download', and 'Zip?'. A large green 'Generate' button is on the right.

Рисунок 3.10 – Завдання додаткових налаштувань генерації даних для таблиці client

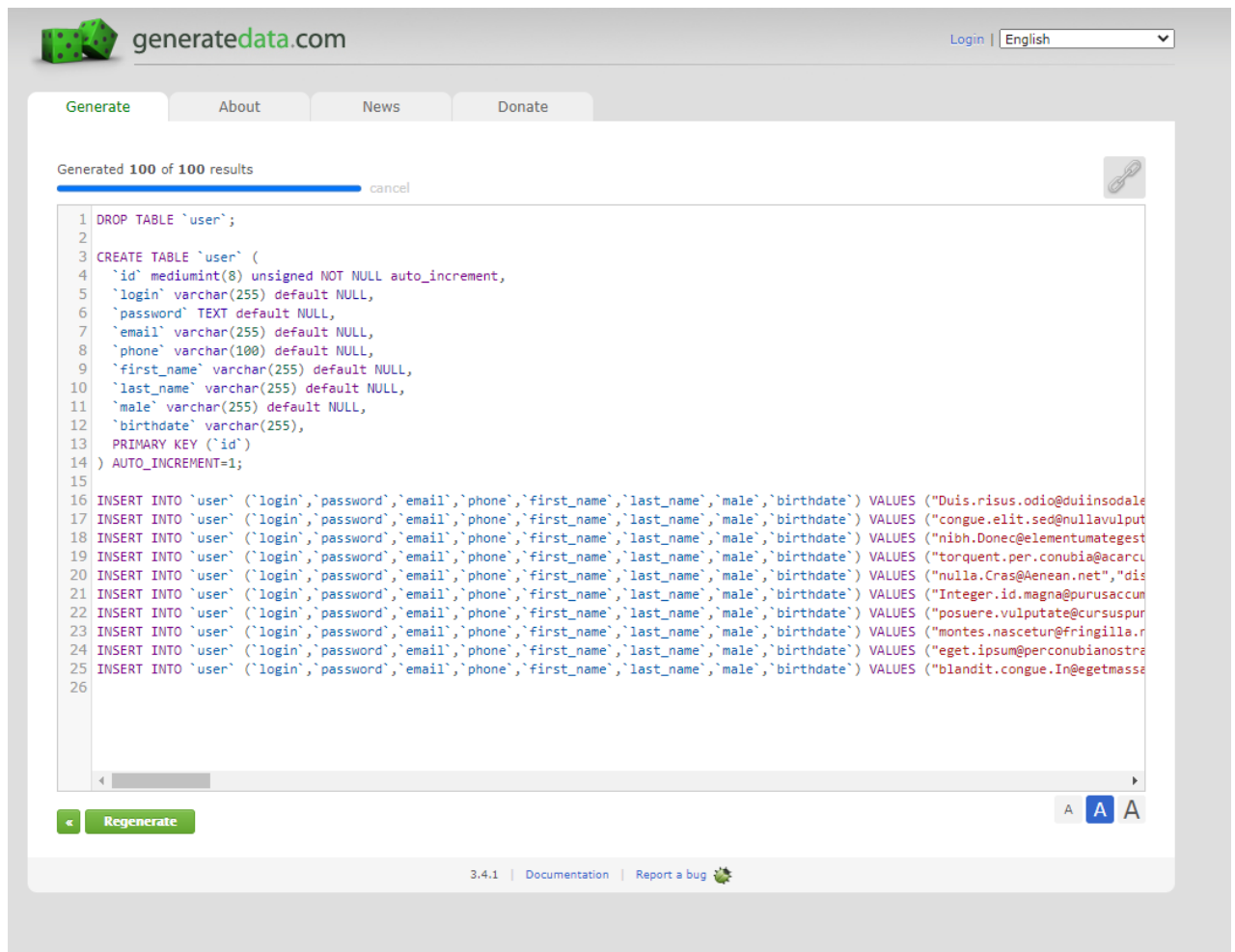


Рисунок 3.11 – Сгенеровані дані для таблиці client (100 записів)

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 + 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 йдуть за порядком (рисунок 3.13).

```

1 • SET @row_number = 0;
2 • UPDATE `client` SET id = (@row_number:=@row_number + 1)
3 • ORDER BY id;
4
5 • select * from client;

```

id	login	password	email	phone	first_name	last_name	male	birth_date
1	admin	admin	sjfhdsjk@mail.com	7759081	Anna	Bena	0	NULL
2	manager	manager	rtrew	trew	Lena	Motra	0	NULL
42	vitae.odio@eu.edu	et	malesuada.augue.ut@velitQuisque.co.uk	0877 794 9016	Kelsie	Swanson	1	197
43	interdum.enim.non@Integer.co.uk	Cum	Pellentesque.habitant.morbi@Nundectuspede....	(01860) 71540	Otto	Russell	1	199
44	malesuada.augue.ut@mattisvelit.net	nec,	ornare@mauris.org	(016977) 7119	Fritz	Crosby	0	199
45	Curae.Phasellus@atlacus.co.uk	mauris	parturient@magnaet.org	(017124) 30276	Whitney	Jacobson	1	197
46	Nam.consequat.dolor@etnetuset.ca	elit,	Donec.tempor@malesuadaaugueut.org	(01029) 589904	Barry	Russell	1	198
47	commodo.hendrerit.Donec@ipsumleo.edu	fringilla	lorem.eget.mollis@non.org	076 1824 0371	Serena	Cooke	0	197
48	tempus.eu.ligula@Suspendisstristiqueque.c...	nunc,	ac.mattis.semper@vitaealiquam.net	0912 155 9953	Carissa	Santiago	1	197
49	In.faucibus.Morbi@mollisPhaselluslibero.co.uk	auctor	amet@sempercursusInteger.com	(0101) 592 0414	Florence	Best	1	196
50	parturient.montes.nascetur@sit.org	luctus	Nunc.mauris.sapien@ornaretortor.net	070 0437 9538	Trevor	Daniels	1	196
51	ipsum@eros.net	vel	non.ante.bibendum@malesuada.net	0883 063 4493	Wayne	Woodward	0	196
52	eget.metus.In@ornarelibero.edu	ipsum	aliquet.molestie.tellus@necurna.org	(016977) 4756	Tiger	Wagner	1	199
53	Sed@Sedeueros.com	convallis	non.dapibus.rutrum@ultricesornare.co.uk	0830 830 8786	Mason	Buchanan	1	199
54	In.ornare.sagittis@eueratsemper.edu	euismod	justo@arcuVestibulum.net	056 3193 9007	Owen	Tyler	0	198
55	Vivamus@loremvitae.com	feugiat	erat.eget.ipsum@Aenean.com	(015460) 92537	Garrett	Brown	1	196
56	mi@elitEtiam.co.uk	Ut	tristique.neque.venenatis@Suspendisstriqu...	(016749) 93296	Cameron	Carrillo	0	197

Рисунок 3.12 – До регенерації ідентифікаторів для таблиці клієнта

```

1 • SET @row_number = 0;
2 • UPDATE `client` SET id = (@row_number:=@row_number + 1)
3 • ORDER BY id;
4
5 • select * from client;

```

id	login	password	email	phone	first_name	last_name	male	birth_date
1	admin	admin	sjfhdsjk@mail.com	7759081	Anna	Bena	0	NULL
2	manager	manager	rtrew	trew	Lena	Motra	0	NULL
3	vitae.odio@eu.edu	et	malesuada.augue.ut@velitQuisque.co.uk	0877 794 9016	Kelsie	Swanson	1	197
4	interdum.enim.non@Integer.co.uk	Cum	Pellentesque.habitant.morbi@Nundectuspede....	(01860) 71540	Otto	Russell	1	199
5	malesuada.augue.ut@mattisvelit.net	nec,	ornare@mauris.org	(016977) 7119	Fritz	Crosby	0	199
6	Curae.Phasellus@atlacus.co.uk	mauris	parturient@magnaet.org	(017124) 30276	Whitney	Jacobson	1	197
7	Nam.consequat.dolor@etnetuset.ca	elit,	Donec.tempor@malesuadaaugueut.org	(01029) 589904	Barry	Russell	1	198
8	commodo.hendrerit.Donec@ipsumleo.edu	fringilla	lorem.eget.mollis@non.org	076 1824 0371	Serena	Cooke	0	197
9	tempus.eu.ligula@Suspendisstristiqueque.c...	nunc,	ac.mattis.semper@vitaealiquam.net	0912 155 9953	Carissa	Santiago	1	197
10	In.faucibus.Morbi@mollisPhaselluslibero.co.uk	auctor	amet@sempercursusInteger.com	(0101) 592 0414	Florence	Best	1	196
11	parturient.montes.nascetur@sit.org	luctus	Nunc.mauris.sapien@ornaretortor.net	070 0437 9538	Trevor	Daniels	1	196
12	ipsum@eros.net	vel	non.ante.bibendum@malesuada.net	0883 063 4493	Wayne	Woodward	0	196
13	eget.metus.In@ornarelibero.edu	ipsum	aliquet.molestie.tellus@necurna.org	(016977) 4756	Tiger	Wagner	1	199
14	Sed@Sedeueros.com	convallis	non.dapibus.rutrum@ultricesornare.co.uk	0830 830 8786	Mason	Buchanan	1	199
15	In.ornare.sagittis@eueratsemper.edu	euismod	justo@arcuVestibulum.net	056 3193 9007	Owen	Tyler	0	198
16	Vivamus@loremvitae.com	feugiat	erat.eget.ipsum@Aenean.com	(015460) 92537	Garrett	Brown	1	196
17	mi@elitEtiam.co.uk	Ut	tristique.neque.venenatis@Suspendisstriqu...	(016749) 93296	Cameron	Carrillo	0	197
18	gravidam@Donectincidunt.ca	Etiam	pharetra.sed.hendrerit@magnaetortor.com	0845 46 45	Aidan	Snow	1	197
19	Duis@vulputatemauris.co.uk	massa.	lorem@natoque.co.uk	(019682) 71729	Courtney	Briggs	0	198
20	Sed.molestie.Sed@aliquamenimnec.ca	neque	eu@maurisSuspendisse.edu	0800 1111	Damon	Reed	0	196
21	Morbi@ametanteVivamus.org	condime...	neque@euismodindolor.org	07624 619807	Sacha	Rowland	1	199
22	ultrices.Vivamus@fermentummetusAenean.com	in	a.nunc@Integer.com	07624 856595	Patricia	Fischer	1	198
23	congue.elit.sed@nullavulputate.co.uk	risus.	non@diam.com	0389 911 6583	Valentine	Rutledge	0	197
24	pede@infelis.co.uk	eu	mi@quamdignissimpharetra.co.uk	07624 703077	Claudia	Sharp	1	199

Рисунок 3.13 – Після регенерації ідентифікаторів для таблиці клієнта

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

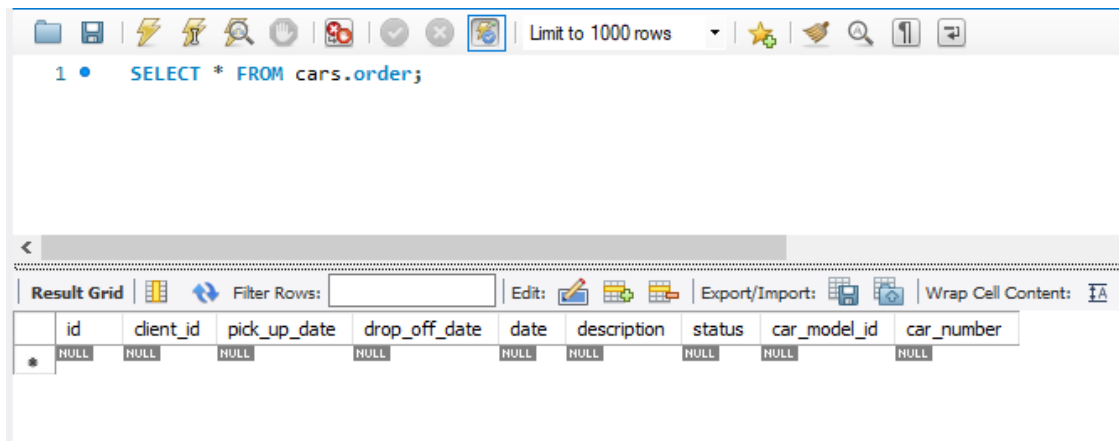


Рисунок 3.14 – Порожня таблиця замовлень

Сгенеруємо дані для таблиці замовлень за допомогою сервісу GenerateData.

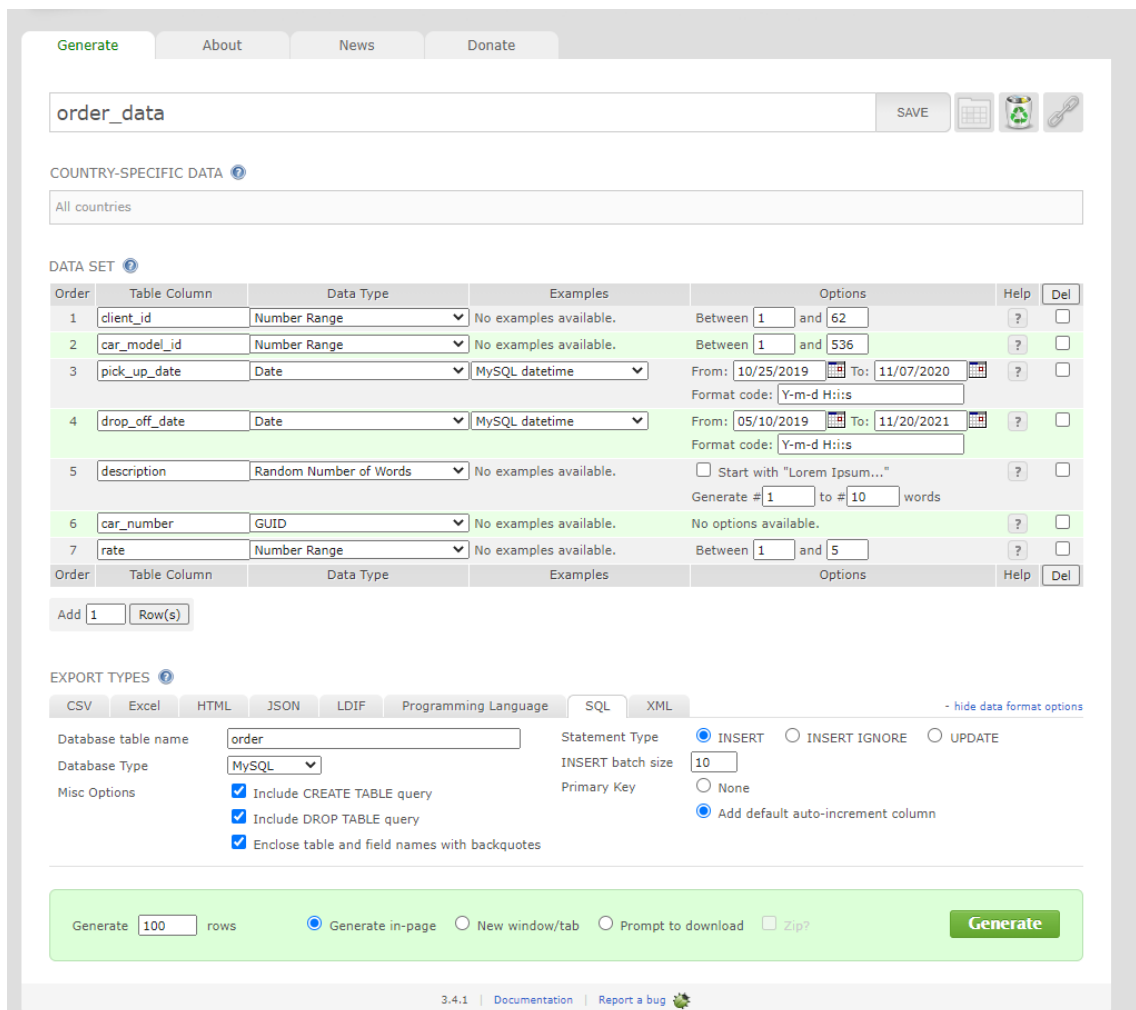


Рисунок 3.15 – Генерація даних для таблиці замовлень

The screenshot shows the website **generatedata.com** with a navigation bar (Generate, About, News, Donate) and a user interface for generating data. A progress bar indicates "Generated 100 of 100 results". The main content area displays SQL code for creating a table named 'order' and inserting 100 rows of data. The code includes a DROP TABLE statement, a CREATE TABLE statement with columns: id (mediumint(8) unsigned NOT NULL auto\_increment), client\_id (mediumint default NULL), car\_model\_id (mediumint default NULL), pick\_up\_date (varchar(255)), drop\_off\_date (varchar(255)), description (TEXT default NULL), car\_number (varchar(36) NOT NULL), and rate (mediumint default NULL). The PRIMARY KEY is on 'id'. The INSERT statements use random values for the columns.

```

1 DROP TABLE `order`;
2
3 CREATE TABLE `order` (
4   `id` mediumint(8) unsigned NOT NULL auto_increment,
5   `client_id` mediumint default NULL,
6   `car_model_id` mediumint default NULL,
7   `pick_up_date` varchar(255),
8   `drop_off_date` varchar(255),
9   `description` TEXT default NULL,
10  `car_number` varchar(36) NOT NULL,
11  `rate` mediumint default NULL,
12  PRIMARY KEY (`id`)
13 ) AUTO_INCREMENT=1;
14
15 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (6,260,"2020-07-17 13:06:55","2020-11-17 01:01:36","libero nec ligula consectetur rhoncus. Nullam v...",96A253);
16 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (28,273,"2020-07-17 13:06:55","2020-11-17 01:01:36","et, euismod",FD4DC5);
17 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (38,394,"2020-07-17 13:06:55","2020-11-05 05:01:16","luctus lobortis. Class aptent taciti sociosqu",7E202F6);
18 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (15,308,"2020-07-17 13:06:55","2020-11-13 16:52:07","a feugiat tellus lorem eu metus.",BCC28F);
19 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (62,499,"2020-07-17 13:06:55","2020-07-07 06:49:59","est arcu ac",2BD801);
20 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (53,443,"2020-07-17 13:06:55","2020-06-11 00:14:37","non, luctus sit",57C8DD);
21 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (18,144,"2019-12-30 15:52:23","2020-08-09 08:38:46","quis diam. Pellentesque habitant morbi tristique ...",96A253);
22 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (6,471,"2020-10-25 20:32:39","2020-10-25 20:32:39","vitae, erat. Vivamus nisi. Mauris",0D5353);
23 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (37,427,"2019-11-13 08:47:20","2020-11-17 10:01:13","facilisi.",944D60);
24 INSERT INTO `order` (`client_id`,`car_model_id`,`pick_up_date`,`drop_off_date`,`description`,`car_number`,`rate`) VALUES (40,232,"2019-11-14 01:19:48","2020-09-15 22:32:45","sed, facilisi vitae, orci. Phasellus dapibus quam",2FBAC4);
25

```

At the bottom, there are buttons for "Regenerate" and "A A A" (font size controls). The footer includes "3.4.1 | Documentation | Report a bug".

Рисунок 3.16 – Результат генерації даних для таблиці замовлень

1 • `SELECT * FROM cars.order;`

id	client_id	car_model_id	pick_up_date	drop_off_date	date	description	status	car_nur
102	48	433	2020-02-17 13:06:55	2020-11-17 01:01:36	2020-10-25 20:31:34	libero nec ligula consectetur rhoncus. Nullam v...	completed	96A253
103	23	432	2019-11-11 00:24:11	2020-09-02 09:34:00	2020-10-25 20:31:34	et, euismod	completed	FD4DC5
104	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	7E202F6
105	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
106	28	289	2019-11-28 21:50:39	2020-07-10 18:34:29	2020-10-25 20:31:34	mattis. Cras eget nisi dictum augue malesuada ...	completed	56BE34E
107	23	451	2019-12-31 12:45:54	2020-07-07 06:49:59	2020-10-25 20:31:34	est arcu ac	completed	2BD801
108	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
109	1	40	2019-12-30 15:52:23	2020-08-09 08:38:46	2020-10-25 20:31:34	quis diam. Pellentesque habitant morbi tristique ...	completed	96A253
110	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
111	58	367	2019-11-13 08:47:20	2020-11-17 10:01:13	2020-10-25 20:31:34	facilisi.	completed	944D60
112	48	433	2020-02-17 13:06:55	2020-11-17 01:01:36	2020-10-25 20:32:39	libero nec ligula consectetur rhoncus. Nullam v...	completed	96A253
123	23	432	2019-11-11 00:24:11	2020-09-02 09:34:00	2020-10-25 20:32:39	et, euismod	completed	FD4DC5
124	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	7E202F6
125	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
126	28	289	2019-11-28 21:50:39	2020-07-10 18:34:29	2020-10-25 20:32:39	mattis. Cras eget nisi dictum augue malesuada ...	completed	56BE34E
127	23	451	2019-12-31 12:45:54	2020-07-07 06:49:59	2020-10-25 20:32:39	est arcu ac	completed	2BD801
128	54	448	2020-04-27 03:36:01	2020-06-11 00:14:37	2020-10-25 20:32:39	non, luctus sit	completed	57C8DD
129	1	40	2019-12-30 15:52:23	2020-08-09 08:38:46	2020-10-25 20:32:39	quis diam. Pellentesque habitant morbi tristique ...	completed	96A253
130	15	324	2019-11-14 01:19:48	2020-09-15 22:32:45	2020-10-25 20:32:39	vitae, erat. Vivamus nisi. Mauris	completed	0D5353
131	58	367	2019-11-13 08:47:20	2020-11-17 10:01:13	2020-10-25 20:32:39	facilisi.	completed	944D60
142	10	66	2020-01-19 02:56:26	2020-08-16 12:53:36	2020-10-25 20:35:36	sed, facilisi vitae, orci. Phasellus dapibus quam	completed	2FBAC4

Рисунок 3.17 – Сгенеровані дані у таблиці замовлен

Отже, було сгенеровано 1020 замовлень (рисунок 3.18).

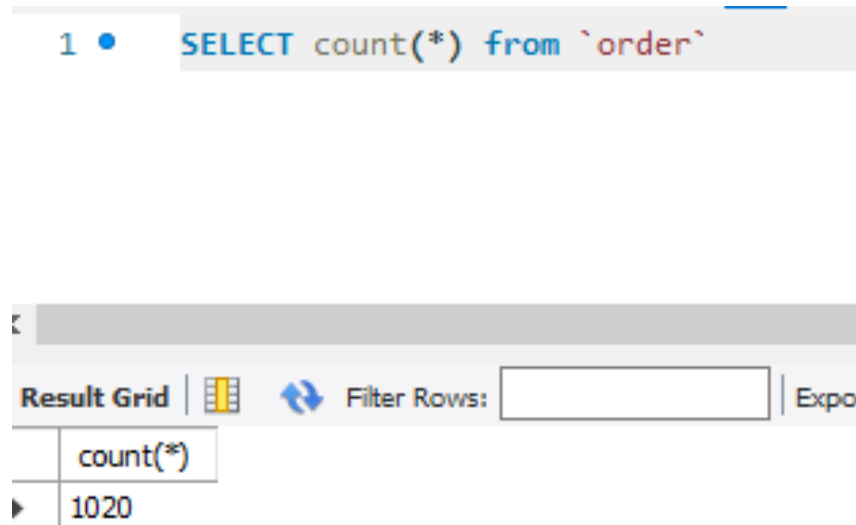


Рисунок 3.18 – Кількість сгенерованх замовлень

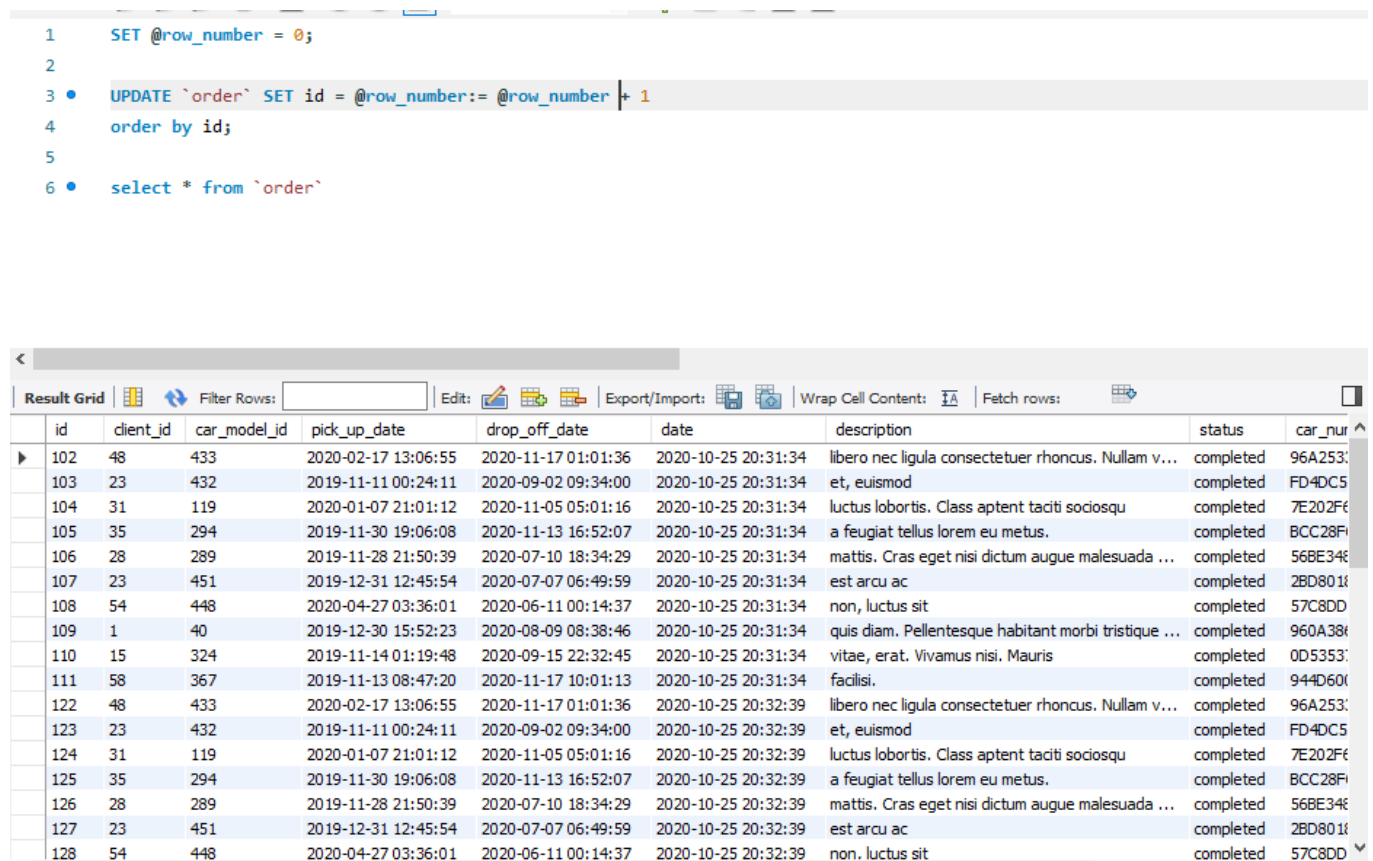


Рисунок 3.19 – До регенерації ідентифікаторів для таблиці замовлень

```

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_num
1	48	433	2020-02-17 13:06:55	2020-11-17 01:01:36	2020-10-25 20:31:34	libero nec ligula consectetur 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	7E202F6
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	BCC28F6
5	28	289	2019-11-28 21:50:39	2020-07-10 18:34:29	2020-10-25 20:31:34	mattis. Cras eget nisi dictum augue malesuada ...	completed	56BE34E
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	2BD801f
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 08:38:46	2020-10-25 20:31:34	quis diam. Pellentesque habitant morbi tristique ...	completed	960A38f
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	944D60f
11	48	433	2020-02-17 13:06:55	2020-11-17 01:01:36	2020-10-25 20:32:39	libero nec ligula consectetur 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	7E202F6
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	BCC28F6
15	28	289	2019-11-28 21:50:39	2020-07-10 18:34:29	2020-10-25 20:32:39	mattis. Cras eget nisi dictum augue malesuada ...	completed	56BE34E

Рисунок 3.20 – Після регенерації ідентифікаторів для таблиці замовлень

## 4 РОЗРОБКА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МЕТОДУ ВИЛУЧЕННЯ ЗНАНЬ УПОДОБАНЬ КЛІЄНТІВ

### 4.1 Аналіз і вибір методів реалізації

#### 4.1.1 Аналіз і обґрунтування вибору типу архітектури додатку

##### 4.1.1.1 Порівняння типів архітектури додатку

###### SOAP (Простий протокол доступу до об'єктів)

Протокол SOAP для простого доступу до об'єктів (SOAP) – це тип архітектури 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 архітектури пов'язаний із взаємовідносинами клієнта та сервера та способом збереження стану. Архітектура REST базується на стилі архітектури 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, архітектура 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:

- використання 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 пропускну 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 архітектури 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 уникайти 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 мову програмування.

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. [18].

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 користувацьких інтерфейсів [18].

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

##### 4.1.3.1 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 що Tomcat підтримує деякі функції Java EE нестандартно,

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 відсутності 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 майбутньому і це є суттєвим недоліком цього серверу [19].

#### 4.1.3.2 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 продукту 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

##### 4.1.4.1 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 архітектура 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 у таблиці баз даних. В основному це пов'язано з базами даних [20].

#### 4.1.4.2 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-фреймворком 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 4.2 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.

```

1 server:
2   port: 8083
3 spring:
4   application:
5     name: CarRental
6   datasource:
7     driver-class-name: com.mysql.cj.jdbc.Driver
8     password: |
9     testWhileIdle: true
10    url: jdbc:mysql://localhost:3306/cars?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
11    username: root
12    validationQuery: SELECT 1
13  jpa:
14    hibernate:
15      ddl-auto: none
16      naming:
17        implicit-strategy: org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyHbmImpl
18        physical-strategy: org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy
19    properties:
20      hibernate:
21        dialect: org.hibernate.dialect.MySQL5Dialect
22    show-sql: true

```

Рисунок 4.1 – Вміст application.yml

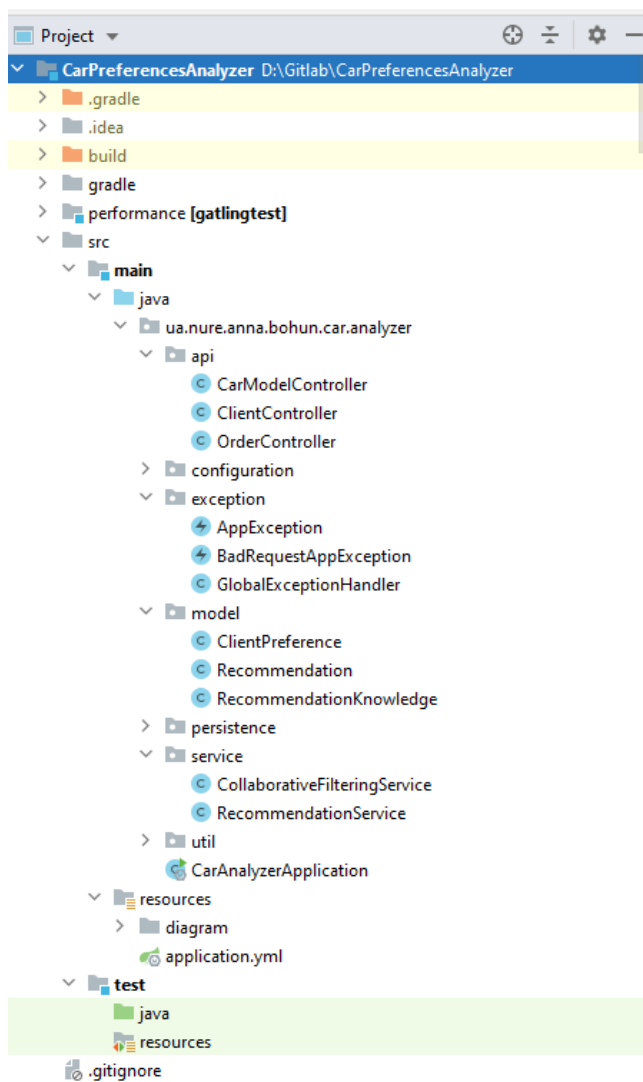


Рисунок 4.2 – Структура Java-проекту

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 фремоворк 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.

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.

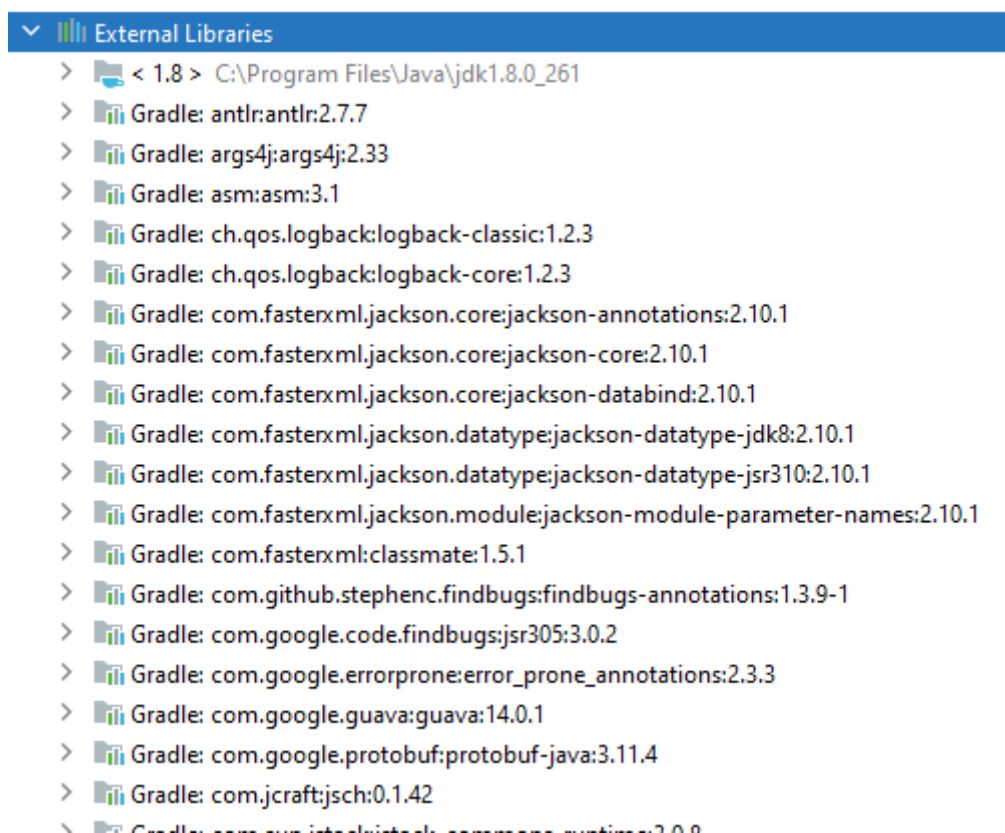


Рисунок 4.3 – Gradle-залежності проекту

Для доступу в базу даних використовується JDBC та такий ORM-фреймворк як 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).

```

2020-11-29 19:53:29.229 INFO 6280 --- [nio-8083-exec-7] u.n.a.b.c.a.s.RecommendationService : Average order rate for client 62 is 3.1852631578947367
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_, carmodel0_.brand as brand3_0_, carmodel0_.car_classification as car_clas4_0_, carmodel0_.cou
Hibernate: select count(carmodel0_.id) as col_0_0_ from car_model carmodel0_
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_0_, carmodel0_.brand as brand3_0_0_, carmodel0_.car_classification as car_clas4_0_0_, carmodi
2020-11-29 19:53:29.273 INFO 6280 --- [nio-8083-exec-7] u.n.a.b.c.a.s.RecommendationService : Car preference map for client62 is ClientPreference(priceRate={21.0=4.0, ;
Hibernate: select client0_.id as id1_1_0_, client0_.birthdate as birthdat2_1_0_, client0_.email as email3_1_0_, client0_.first_name as first_na4_1_0_, client0_.last_name as l
Hibernate: select orders0_.client_id as client_i9_2_0_, orders0_.id as id1_2_0_, orders0_.id as id1_2_1_, orders0_.car_model_id as car_model8_2_2_1_, orders0_.car_number as car_
2020-11-29 19:53:48.574 INFO 6280 --- [nio-8083-exec-8] u.n.a.b.c.a.s.RecommendationService : Average order rate for client 62 is 3.1852631578947367
Hibernate: select carmodel0_.id as id1_0_, carmodel0_.body_type as body_typ2_0_, carmodel0_.brand as brand3_0_, carmodel0_.car_classification as car_clas4_0_, carmodel0_.cou
Hibernate: select count(carmodel0_.id) as col_0_0_ from car_model carmodel0_

```

Рисунок 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 (і'мя 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.

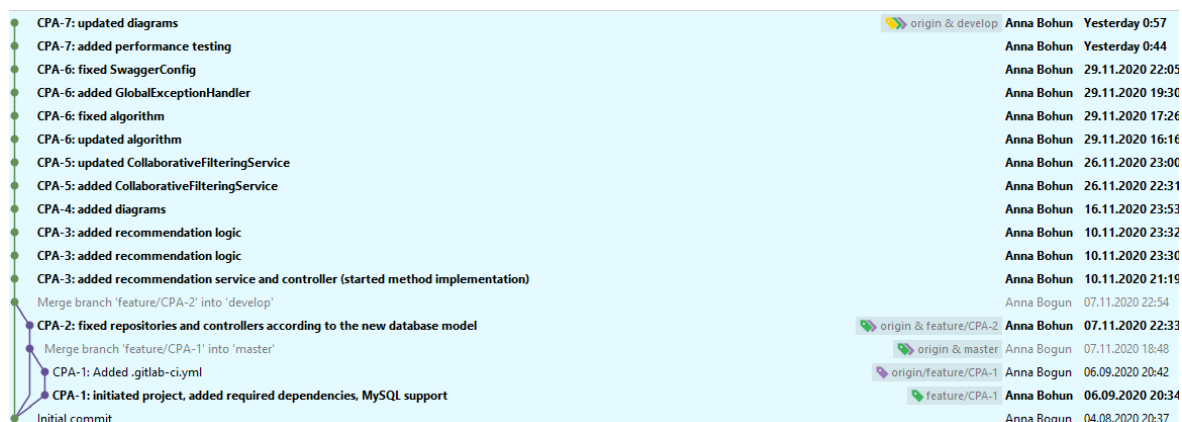


Рисунок 4.5 – Історія комітів додатку

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.

**CarsAnalyzer**  
Project ID: 20339943

19 Commits | 4 Branches | 0 Tags | 15.3 MB Files | 15.4 MB Storage

master | carsanalyzer / +

Merge branch 'develop' into 'master'  
Anna Bogun authored 10 seconds ago

README | CI/CD configuration | Add LICENSE | Add CHANGELOG | Add CONTRIBUTING | Add Kubernetes cluster

Name	Last commit	Last update
gradle/wrapper	CPA-3: added recommendation logic	2 weeks ago
performance	CPA-7: added performance testing	1 day ago
src/main	CPA-7: updated diagrams	1 day ago
.gitignore	CPA-3: added recommendation logic	2 weeks ago
.gitlab-ci.yml	CPA-1: Added .gitlab-ci.yml	2 months ago
README.md	CPA-7: added performance testing	1 day ago
build.gradle	CPA-6: fixed SwaggerConfig	1 day ago
gradlew	CPA-1: initiated project, added required dependencies, MySQL support	2 months ago
gradlew.bat	CPA-1: initiated project, added required dependencies, MySQL support	2 months ago
settings.gradle	CPA-7: added performance testing	1 day ago

README.md

**CarsAnalyzer**

This application is designed for calculating recommendations for car rental clients.

Рисунок 4.6 – Огляд проекту в GitLab

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

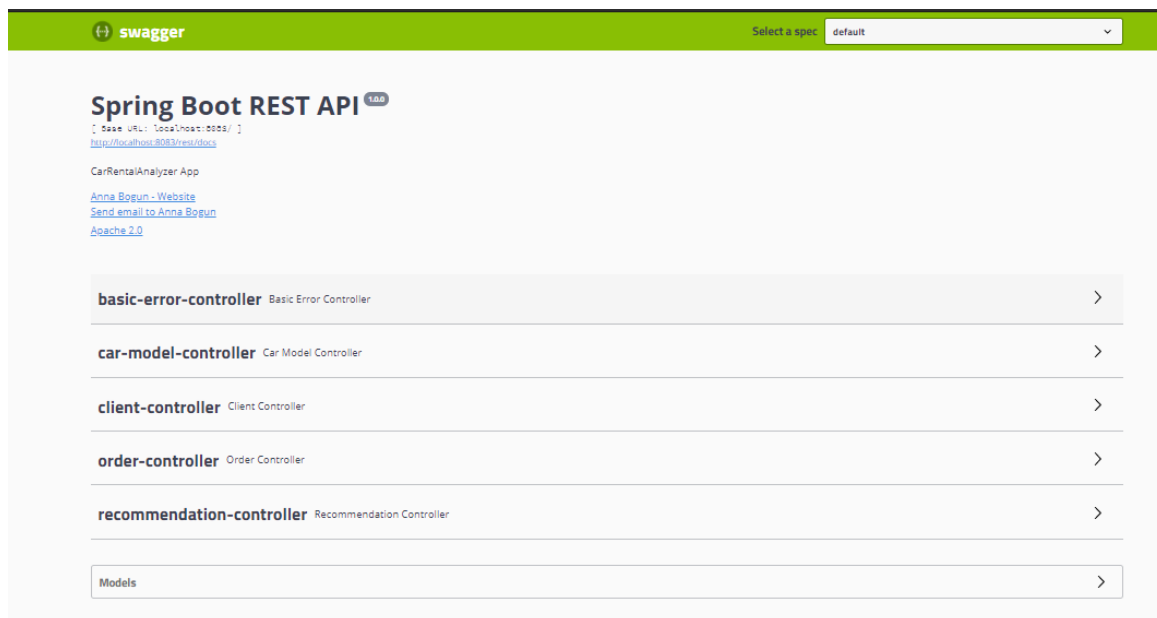


Рисунок 4.7 – Головна сторінка Swagger

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

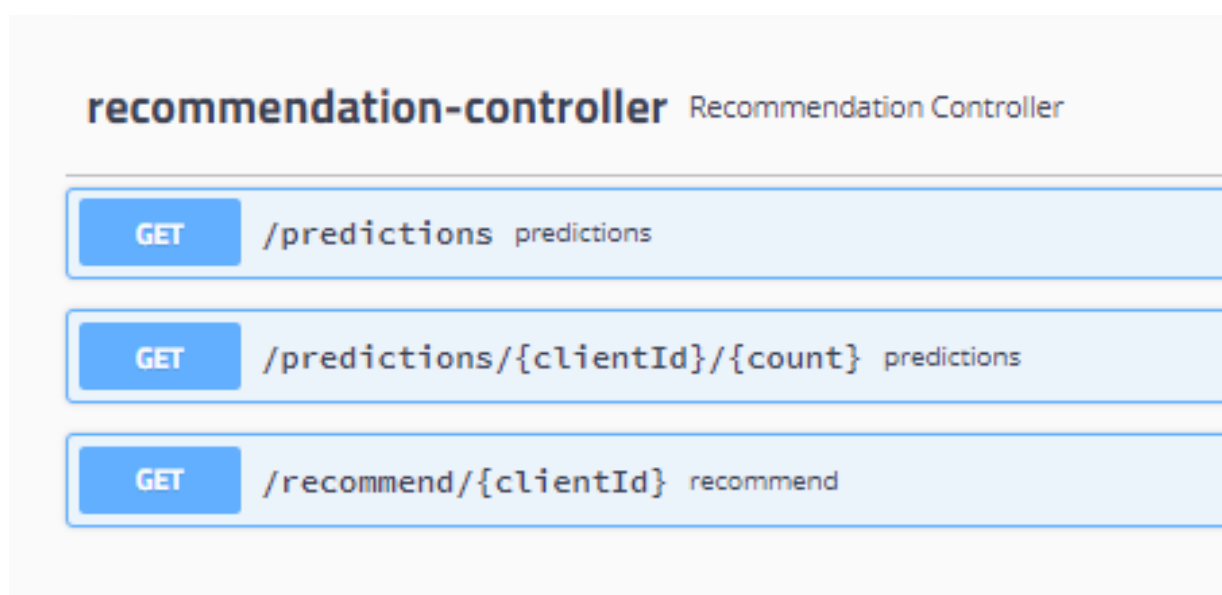


Рисунок 4.8 – Перелік ендпоінтів для контролеру рекомендацій

client-controller Client Controller	
GET	/clients/ getAll
GET	/clients/{id} getOrderById
DELETE	/clients/{id} deleteById

Рисунок 4.9 – API client-controller

GET /recommend/{clientId} recommend

Parameters Try it out

Name	Description
<b>clientId</b> * required integer (path)	clientId
count integer (query)	count

Responses Response content type: application/json

Code	Description
200	<p>OK</p> <p>Example Value   Model</p> <pre>{   "clientId": 0,   "recommendationList": [     {       "carId": 0,       "isRecommended": true     }   ] }</pre>
401	Unauthorized
403	Forbidden
404	Not Found

Рисунок 4.10 – GET-метод client-контролеру

## Models

```
CarModel ▾ {
  bodyType           string
  brand              string
  carClassification string
  countryOfOrigin   string
  cylinders          string
  driveType         string
  engineSize        string
  engineType        string
  fuelType          string
  horsepower        integer($int32)
  id                integer($int32)
  model             string
  price             number($double)
  transmission      string
  valves            integer($int32)
  year              integer($int32)
}
```

```
Client ▾ {
  birthdate          string($date)
  email              string
  firstName          string
  id                integer($int32)
  lastName           string
  login             string
  male              boolean
  orders             > [...]
  password           string
  phone             string
}
```

Рисунок 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)

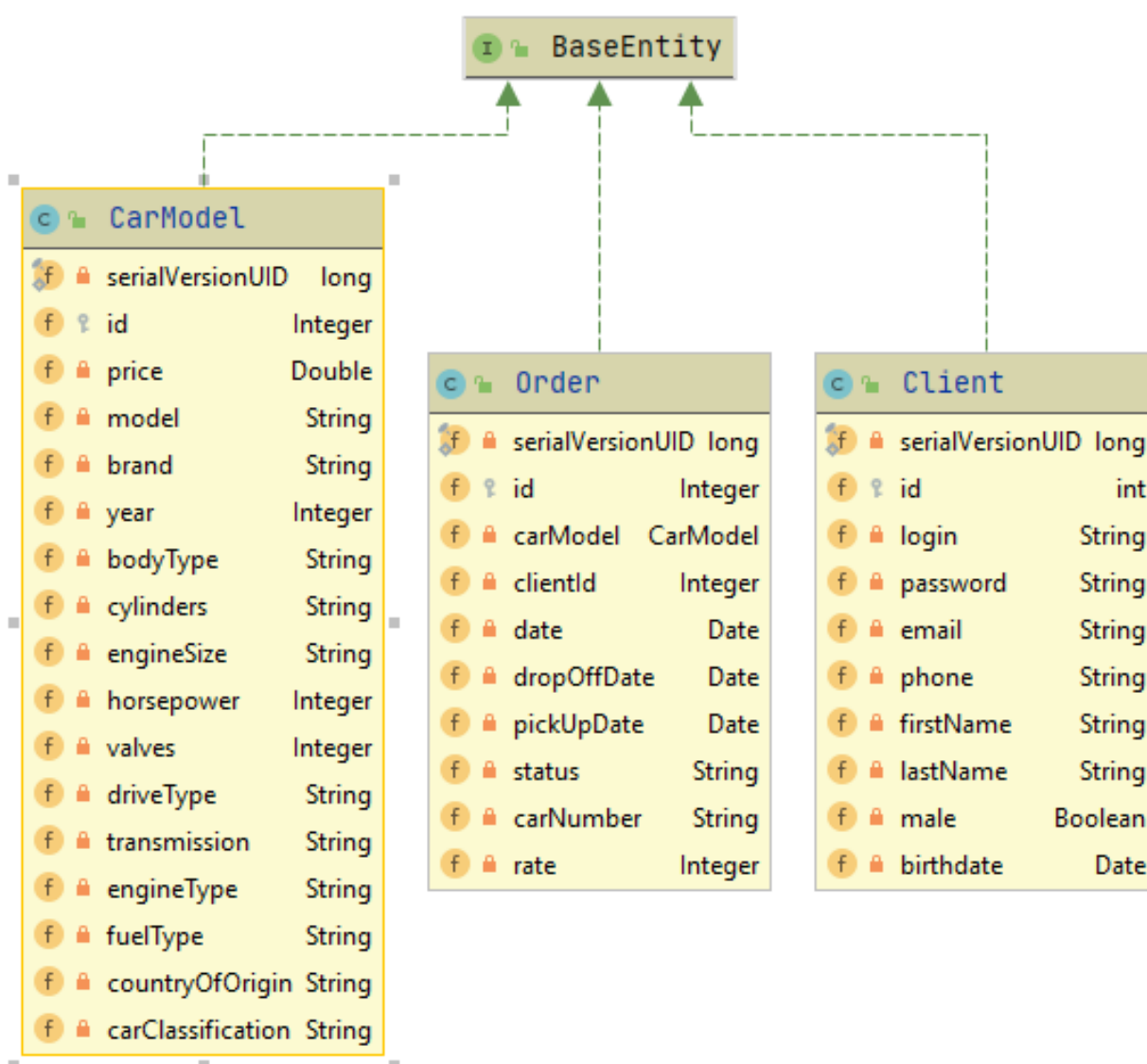


Рисунок 4.12 – Діаграма класів-сутностей

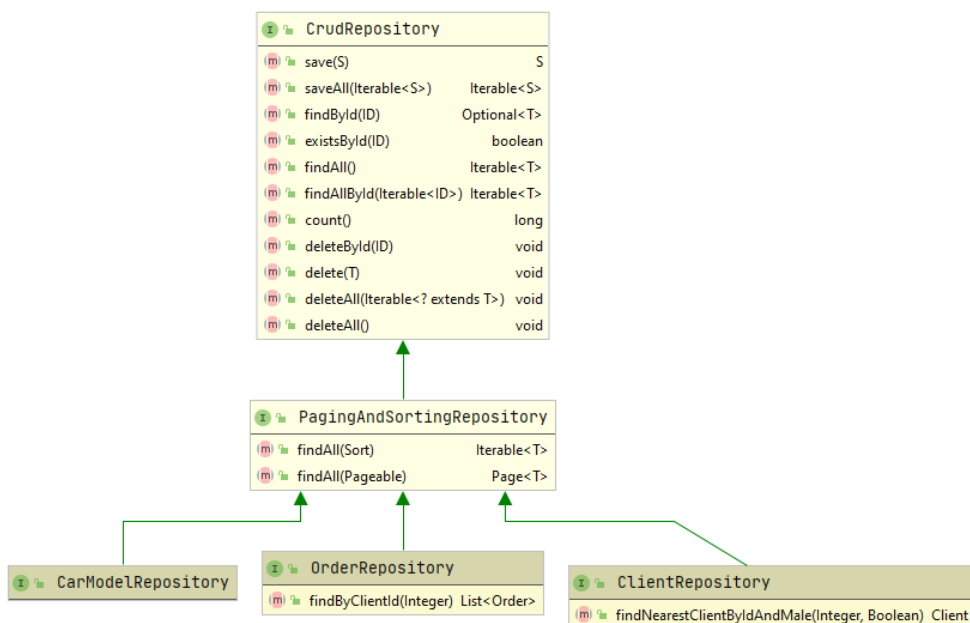


Рисунок 4.13 – Діаграма класів-репозиторіїв

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

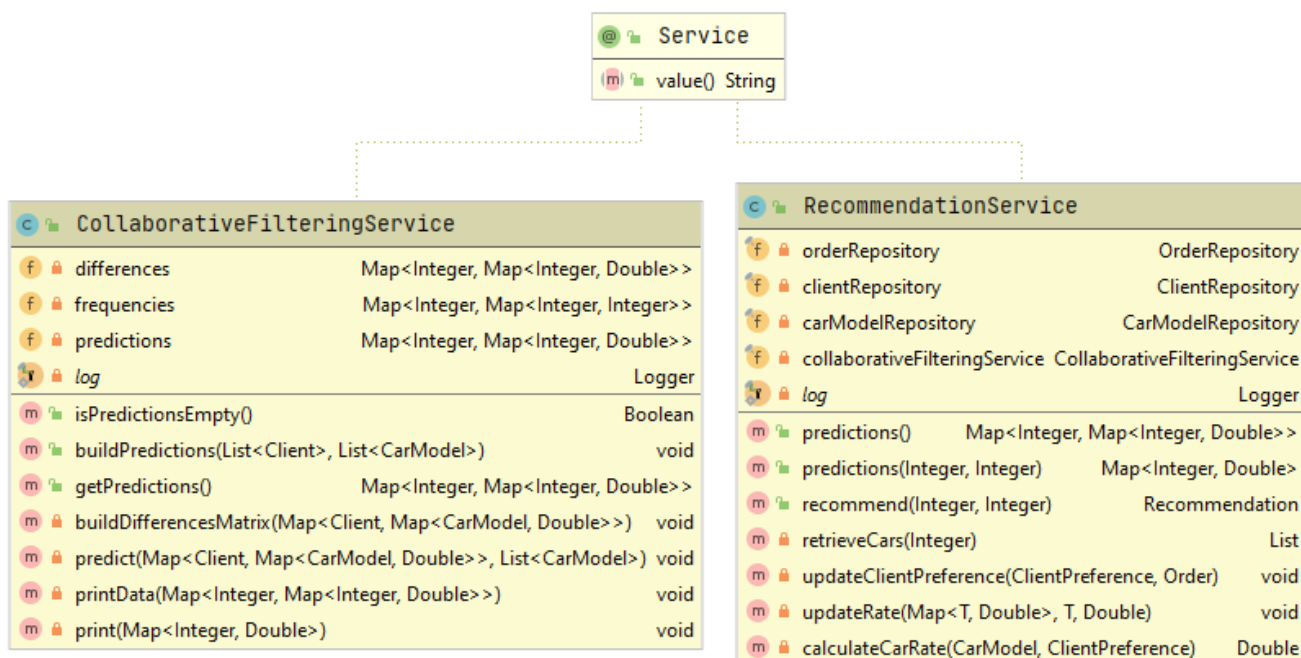


Рисунок 4.14 – Діаграма класів-сервісів

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 переліковані класи продемонстровано на рисунку 4.14 у вигляді діаграми. Сервіси та контролери використовують додаткові класи-моделі (рисунок 4.16)

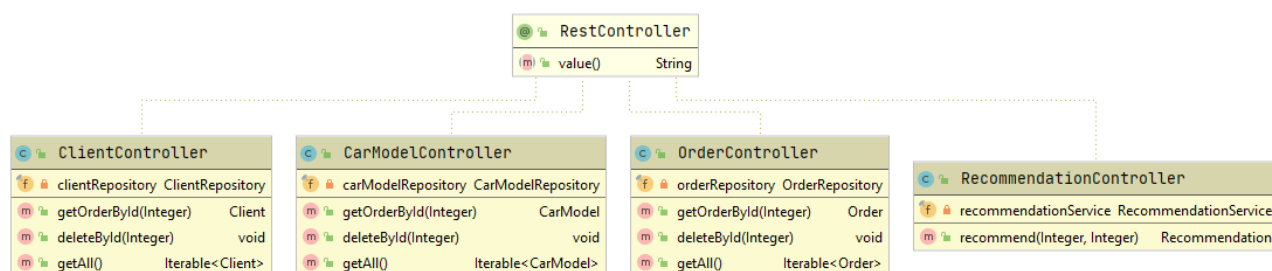


Рисунок 4.15 – Діаграма класів-контролерів

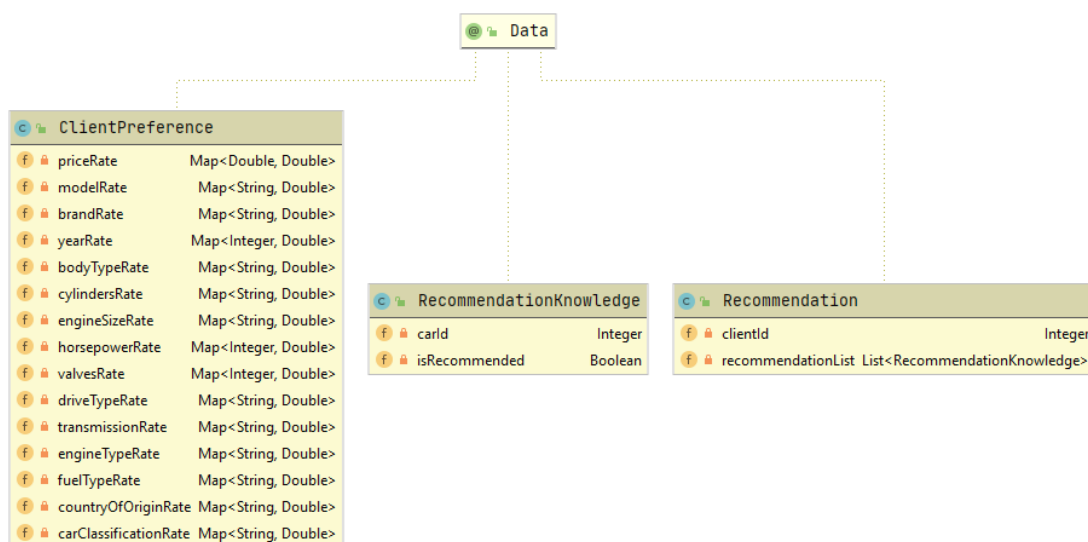


Рисунок 4.16 – Діаграма класів-моделей

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.

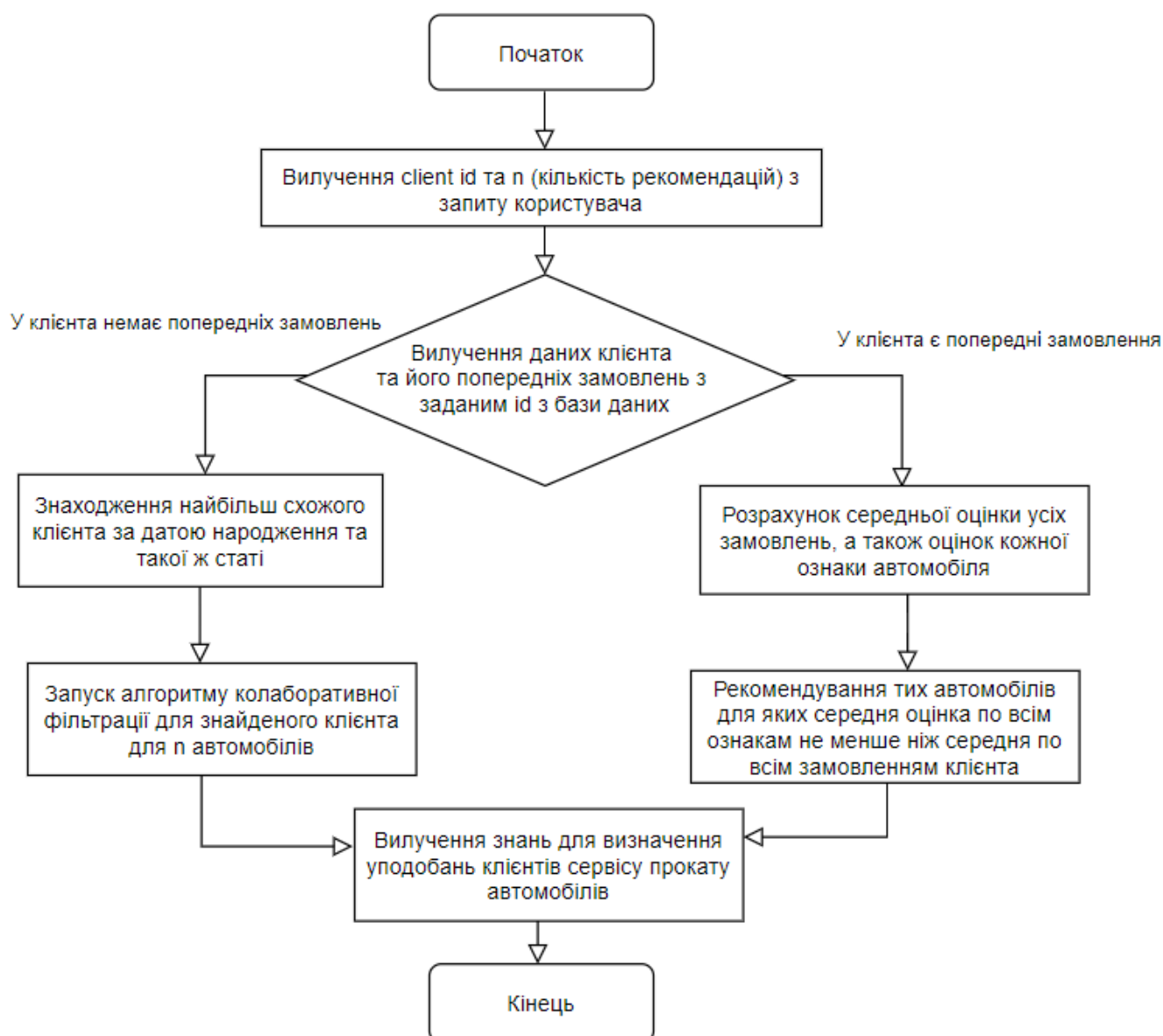


Рисунок 4.17 – Схема алгоритму роботи програми методу вилучення знань щодо уподобань клієнтів

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);
}
```

Рисунок 4.18 – SQL-запит у класі ClientRepository

```
1 • SELECT * FROM client WHERE id <> 1 AND male = 1
2 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 794 9016	Kelsie	Swanson	1	1972-06-17

Рисунок 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.

```
GET http://localhost:8083/recommend/62?count=100
```

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies Code

Body Cookies Headers (5) Test Results 200 OK 528 ms 3.7 KB Save Response

```
1 {
2   "clientId": 62,
3   "recommendationList": [
4     {
5       "carId": 101,
6       "isRecommended": false
7     },
8     {
9       "carId": 102,
10      "isRecommended": false
11    },
12    {
13      "carId": 103,
14      "isRecommended": false
15    },
16    {
17      "carId": 104,
18      "isRecommended": false
19    }
20  ]
21 }
```

Рисунок 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 дозволя 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.

```
2
3 import io.gatling.core.Predef._
4 import io.gatling.http.Predef._
5
6 class GatlingRecommendApiTest extends Simulation {
7
8     val httpConf = http.baseUrl("http://localhost:8083")
9     | .header("Accept", "application/json")
10    val repeatCount = 10
11    val random = scala.util.Random
12
13    def reccomend(): Any = {
14        repeat(repeatCount) {
15            exec(http("Recommend API test")
16                | .get("/recommend/" + random.nextInt(63)+ "?count=30")
17                | .check(status.is(200)))
18        }
19    }
20
21    val scn = scenario("Recommend API test")
22    | .exec(reccomend())
23
24    setUp(
25        | scn.inject(atOnceUsers(1))
26        | ).protocols(httpConf)
27
28 }
```

Рисунок 4.21 – Приклад навантаженого тесту з використанням Gatling

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



Рисунок 4.22 – Сгенерований звіт з результатами тестування Car API

#### > Client API test

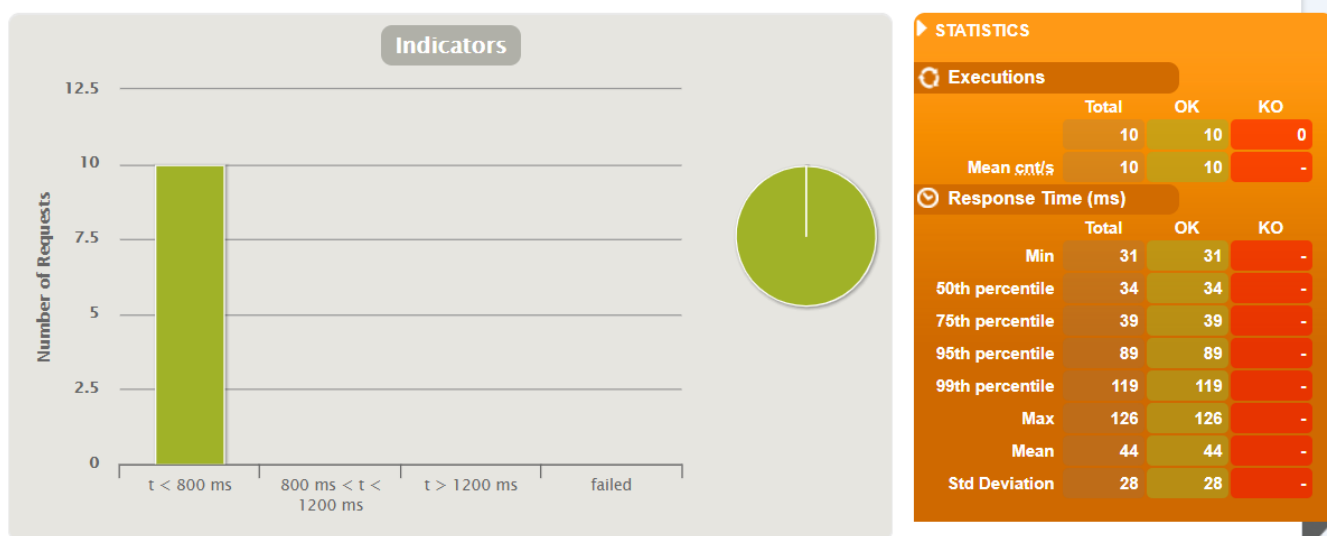


Рисунок 4.23 – Сгенерований звіт з результатами тестування Client API

### > Order API test

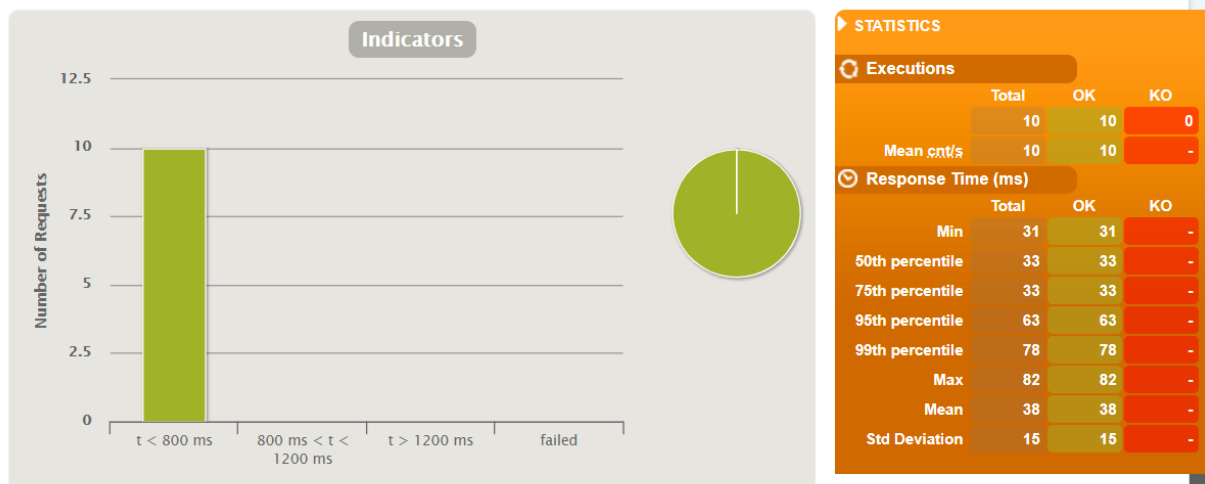


Рисунок 4.24 – Сгенерований звіт з результатами тестування Order API

### > Recommend API test

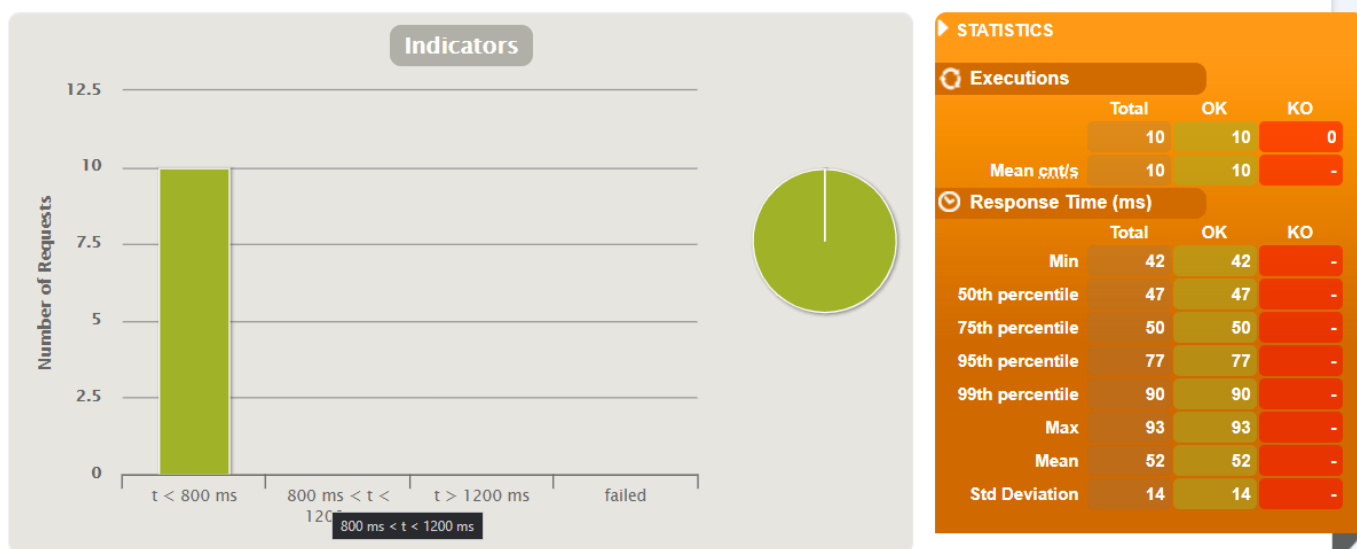


Рисунок 4.25 – Сгенерований звіт з результатами тестування Recommend API

## ВИСНОВКИ

В ході виконання даної атестаційної 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 метод модифікація, 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.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Бізнес з прокату автомобілів. [Електронний ресурс] / PRO-BIZNES – Режим доступу: [www / URL: http://pro-biznes.com.ua/idei-dlya-biznesu/biznes-z-prokату-avtomobiliv.html](http://pro-biznes.com.ua/idei-dlya-biznesu/biznes-z-prokату-avtomobiliv.html) – Дата звернення: 10.10.2020 р. – Загол. з екрану.
2. 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](https://zakon.rada.gov.ua/laws/show/435-15/page13) – Дата звернення: 21.10.2020 р. – Загол. з екрану.
3. Джонс М. Т. Принципы работы рекомендательных механизмов Интернета. [Електронний ресурс]. IBM developerWorks, 2014. URL: <https://www.ibm.com/developerworks/ru/library/os-recommender1/>.
4. Пономарев А. В. Обзор методов учета контекста в системах коллаборативной фильтрации // Труды СПИИРАН, 2013. № 7 (30), С. 169-188.
5. Михайловский Н. Анатомия рекомендательных сервисов. [Электронный ресурс]. Centrobite, 2013, URL: <http://centrobite.ru/blog/>.
6. Car Recommendation System Version 1.0. [Електронний ресурс] / Nars cars – Режим доступу: [www / URL: https://seniorproject.cis.fiu.edu/seniorprojects/car-recommendation-system/](https://seniorproject.cis.fiu.edu/seniorprojects/car-recommendation-system/) – Дата звернення: 08.10.2020 р. – Загол. з екрану.
7. Recombee. [Електронний ресурс] / Nars cars – Режим доступу: [www / URL: https://www.recombee.com/product-recommendations.html](https://www.recombee.com/product-recommendations.html) – Дата звернення: 10.10.2020 р. – Загол. з екрану.
8. Amazon Personalize. [Електронний ресурс] / Nars cars – Режим доступу: [www / URL: https://aws.amazon.com/personalize/](https://aws.amazon.com/personalize/) – Дата звернення: 10.10.2020 р. – Загол. з екрану.
9. Xiaoyuan Su, Taghi M. Khoshgoftaar A Survey of Collaborative Filtering Techniques. SuXiaoyuan, M. Khoshgoftaar Taghi [Електронний ресурс]. – Режим доступу: <http://www.hindawi.com/journals/aai/2009/421425/>
10. Su X., Khoshgoftaar T.M. A Survey of Collaborative Filtering Techniques; Advances in Artificial Intelligence, 2009.

11. Panagiotis Symeonidis, Andreas Zioupos. Matrix and Tensor Factorization. Techniques for Recommender Systems /P. Symeonidis – Free University of Bozen-Bolzano, Italy: «Springer», 2016. – 101 с.
12. P. Lops, M. Gemmis, G. Semeraro Recommender Systems Handbook Springer-Verlag, 2011.
13. C. C. Aggarwal, Recommender Systems: The Textbook. Springer. C: 837, 2016
14. Melville P., Sindhvani V. Recommender systems; Encyclopedia of Machine Learning, 2010
15. 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.
16. Oliver Theobald. Machine Learning: Make Your Own Recommender System (MachineLearningFromScratchBook3).
17. Web Services: A Comparison of Soap and Rest Services. [Текст] / Festim Halili, Erenis Ramadani – Tetova: University of Tetova. 2018 – 175-183 с.
18. 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.
19. Top Java Application Servers: Tomcat vs. Jetty vs. GlassFish vs. WildFly. [Электронный ресурс] / Stackify – Режим доступа: [www / URL: https://stackify.com/tomcat-vs-jetty-vs-glassfish-vs-wildfly/](http://www.stackify.com/tomcat-vs-jetty-vs-glassfish-vs-wildfly/) – Дата звернення: 01.12.2020 р. – Загол. з екрану.
20. 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.
21. Гослінг, Джеймс. The Java Language Environment. Design Goals of the Java TM Programming Language. [Текст] / Джеймс Гослінг. – Санта-Клара: Sun Microsystems, Inc., 1996 – 548 с.
22. 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.

23. 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.

24. 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)

25. Порівняння засобів створення пайплайнів Apache для розподіленої обробки даних. [Текст] / Анна Богун. – Дніпро: V Всеукраїнська науково-практична конференція MEICS-2020, 2020 – 104-105 с.