Гідлевський Д.В., здобувач,
Державний університет «Житомирська політехніка»

РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ УПРАВЛІННЯ РЕСТОРАННИМ ВЕБ-ДОДАТКОМ

У сучасних умовах розвитку ресторанного бізнесу веб-додатки є ключовим інструментом для оптимізації замовлень, управління меню, збору відгуків та підвищення задоволеності клієнтів загалом. ЕмануельМелесе [1] у своїй статті підкреслює, що відгуки й коментарі клієнтів, а саме їхні оцінки, можна застосовувати для формування персоналізованих рекомендацій. Це стимулює гостей робити повторні замовлення, підвищує середній чек і формує довгострокову лояльність. Застосування технологій машинного навчання і статистичних моделей дає можливість аналізувати поведінку користувачів і попит на позиції меню. Одним із підходів до реалізації таких рекомендацій є колективна фільтрація за допомогою методу невід'ємної матричної факторизації (NMF), який дозволяє аналізувати вподобання, щоб запропонувати страви, що ймовірно їм сподобаються [2].

Суть цього підходу полягає в тому, що всі оцінки відвідувачів (наприклад, від 1 до 5 балів) представляються у вигляді великої матриці, елементи якої є невід'ємними. NMF «розкладає» цю матрицю на дві менші, кожна з яких містить приховані характеристики: одна відповідає за узагальнені «смаки» користувачів, а інша описує «властивості» страв. Зіставлення цих характеристик дає змогу передбачити, яку оцінку може поставити клієнт тій чи іншій позиції меню, навіть якщо він раніше її не замовляв. Таким чином, система персоналізованої рекомендації пропонує позиції, що потенційно сподобаються відвідувачам.

Для оптимізації операційних процесів ресторану важливо враховувати не лише рекомендації страв для клієнтів, але й сезонні та трендові зміни попиту[3]. Для цього використовують аналіз часових рядів за допомогою моделі ARIMA (AutoRegressiveIntegratedMovingAverage). Це методика, призначенавиявляти й описуватизакономірності у часових рядах, де дані (наприклад, щоденнізамовлення) фіксують на регулярнихпроміжках. ARIMA зазвичай описується трьома параметрами:

1. AR (авторегресія) — показує, як минулі значення впливають на поточне;
2. I (інтегрування) — віднімає сусідні спостереження, щоб позбутися глобального тренду й отримати «більш рівний» ряд.
3. MA (ковзне середнє) — нівелює випадкові коливання, згладжуючи шум у даних.

Для ресторану це означає, що, маючи історію замовлень (наприклад, за кожну годину чи день), можна навчити модель ARIMA, щоб вона виявляла приховані закономірності. На основі такої аналітики ресторан може заздалегідь визначати періоди зростання чи спаду попиту й заздалегідь планувати обсяги закупівель, кількість персоналу або інші ресурси. Це допомагає ресторану краще планувати запаси продуктів в пікові години.

Щоб правильно використати методики вище, важливо вибрати правильну архітектуру. Тоді у єдиній системі можливо забезпечити високу продуктивність оброблення замовлень, належну роботу з базою даних, простоту внесення змін у бізнес-логіку, а також можливість швидко оновлювати алгоритми машинного навчання.
Умовно систему можна поділити на дві основні частини:

1. Бекенд, реалізований на Java з використанням SpringBoot [4]. Цей фреймворк дає змогу швидко виконувати CRUD-операції, впроваджувати авторизацію, зручно взаємодіяти з базою даних і налаштовувати авторизацію та аутентифікацію. Окрім того, SpringBoot має розвинені засоби автоконфігурації, що полегшує розгортання веб-додатку.
2. Аналітичний модуль, який потрібно винести в окремий сервіс на Python. У ньому реалізовано методи NMF (для формування рекомендацій) і ARIMA (для прогнозування попиту). Така архітектура є кращою, завдяки можливості швидко змінювати алгоритми машинного навчання без ризику впливу на базову інфраструктуру обробки замовлень і керування меню, іншою перевагою є можливість використовувати Python окремо, що відкриває доступ до великої екосистеми бібліотек.

Також важливою причиною впровадження мікросервісної архітектури - зменшення ризику помилок при оновленні системи та можливість швидко впроваджувати зміни. Потрібно запровадити безперервну інтеграцію й розгортання (CI/CD). У межах цього підходу створюються автоматизовані процеси збірки й тестування коду, які підвищують якість розробки й знижують кількість збоїв.

Контейнеризація (наприклад, використовуючи Docker) дає змогу запускати різні програми у відокремлених середовищах, а використання систем оркестрації (Kubernetes та подібних) забезпечує гнучке масштабування та оновлення окремих модулів.Завдяки цьому, можна додавати нові алгоритми або змінювати логіку бекенду без тривалої зупинки роботи ресторанного застосунку. Це має значення у ситуаціях, коли виникає необхідність швидко виправити критичні помилки.

Ресторан мусить дбати й про збереження даних клієнтів. Безпека є важливою. Уся інформація про замовлення та вподобання користувачів має бути захищеною від доступу сторонніх осіб. Розробка же має включати створення механізмів шифрування даних під час передачі між сервісами та зберігання, обмежувати доступ до критичних сервісів і вести моніторинг підозрілих запитів.

Для шифрування даних під час передавання налаштовують протокол TLS (TransportLayerSecurity), щоб уникнути перехоплення даних, а також застосовують алгоритми шифрування на рівні бази даних.

Найкращим рішенням є заздалегідь передбачити резервне копіювання та відновлення БД, а також проводити регулярний аудит, щоб убезпечити систему від потенційних вразливостей. Використовуючи таке рішення, вдається уникнути витоків конфіденційних даних і втримувати довіру клієнтів, адже репутація є вирішальним фактором для будь-якого бізнесу.

Таким чином, формується мікросервісна архітектура, де кожен компонент розв'язує свій набір завдань. Для збереження інформації про замовлення, користувачів та оцінки слід використовувати реляційну СУБД (наприклад,

PostgreSQL), яка забезпечить необхідний рівень консистентності даних. Взаємодія між сервісами буде здійснюватися через REST API, що спростить обмін даними у форматі JSON в зашифрованому форматі.

Отже, проєктування такої структури з окремими модулями для операційних завдань і алгоритмів машинного навчання відповідає вимогам інтелектуальної системи і дає змогу суттєво підвищити ефективність планування та обслуговування клієнтів.

Список використаних джерел:

1. Melese A. Food and restaurant recommendation system using hybrid filtering mechanism / A. Melese // North American Academic Research Journal. — 2021. — Vol. 4, № 4. — C. 268-281.
2. Kim J. Restaurant recommender system based on customer preference and rating data / J. Kim, S. Lee // IEEE Access. — 2023. — DOI: 10.1109/ACCESS.2023.3324971.
3. Schmidt A. Machine learning based restaurant sales forecasting / A. Schmidt, M. W. Ul Kabir, M. T. Hoque // Machine Learning and Knowledge Extraction. — 2022. — Vol. 4, № 1. — C. 105-130.
4. Spring Boot: documentation [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-boot (дата звернення: 02.25.2025).