УДК 004.7

Колесник О.А., здобувач,
Піонтківський В.І., асистент
Державний університет «Житомирська політехніка»

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ БРОНЮВАННЯ УПРАВЛІННЯ ТА АНАЛІТИКИ У СФЕРІ ПОСЛУГ

У сучасних умовах високої конкуренції в сфері послуг веб-застосунки відіграють ключову роль у залученні нових клієнтів, підвищенні якості їх обслуговування, обробці даних для детальної аналітики бізнес-процесів та отриманні зворотного зв'язку. Такі системи дозволяють не лише оптимізувати й скоротити операційні процеси, пов'язані з ручною обробкою інформації, але й забезпечують бізнесу конкурентоспроможність на цифровому ринку, тобто досягнення лідерських позицій у пошуковій видачі за ключовими словами чи регіоном.

Аналіз даного сегменту дає змогу виокремити основні бізнес вимоги для такого застосунку:

1. Залучення клієнтів та користувацький досвід.

Висока позиція у пошуковій видачі забезпечує стабільний потік нових відвідувачів, однак наступним важливим етапом є перетворення цих відвідувачів на реальних клієнтів. Для досягнення високої конверсії необхідно впровадити спрощену систему реєстрації та бронювання, що мінімізує кількість кроків до оформлення послуги. Важливу роль відіграє також інтуїтивно зрозумілий, візуально привабливий та зручний інтерфейс, який формує довіру користувачів, покращує їх враження від взаємодії із системою та підштовхує до повторного використання сервісу.

2. Автоматизація бізнес процесів.

Максимальне скорочення ручної обробки інформації (наприклад, бронювання, оформлення замовлень, формування рахунків та управлінських звітів) підвищить ефективність та зменшить операційні витрати. Автоматичне формування детальних звітів дозволить керівництву оперативно реагувати на всі процеси та швидше приймати ключові рішення в розвитку власного бізнесу.

3. Гнучкість та розширення.

При проєктуванні таких систем необхідно заздалегідь закладати високу архітектурну гнучкість і масштабованість. Це вимагає врахування як потенційного якісного зростання бізнесу (впровадження

нових послуг та функціоналу) так і кількісного розширення (збільшення мережі закладів, кількості користувачів та обсягів даних). Також не варто забувати про такі технічні деталі, як можливість легкої інтеграції нових модулів, вони мають бути закладені ще на початковому етапі проєктування, не обмежуючись лише поточним списком бізнес-вимог.

Для задоволення цих ключових потреб системи важливо знайти ефективні архітектурні рішення, які в подальшому будуть сприяти тільки розширенню програмного продукту.

Бекенд системи пропоную реалізовувати на мовою програмування Java із застосуванням фреймворку Spring Boot, який забезпечить надійну побудову серверної логіки та масштабованої архітектури. Використання Spring Boot спрощує процес створення та підтримку основних операцій, налаштування взаємодії з базами даних, а також впровадження механізмів безпеки (аутентифікації та авторизації). Завдяки великому набору вбудованих модулів і автоматичній конфігурації, Spring Boot значно зменшує час на розгортання застосунку та підвищує стабільність роботи[2].

Фронтенд частину системи доцільно реалізувати з використанням JavaScript-фреймворку Next.js, який створений на базі бібліотеки React та забезпечує високу продуктивність, зручність розробки й SEO-оптимізацію завдяки можливості серверному рендерингу. Архітектура Next.js дозволяє ефективно поєднувати статичні та динамічні сторінки, що прискорює завантаження сторінки та забезпечить позитивний користувацький досвід. Завдяки вбудованій маршрутизації, системі кешування та підтримці API-роутів, фреймворк значно спрощує розробку в компонентно-орієнтованому стилі.

Додатково, використання React-компонентів дає змогу створювати модульний, легко масштабований користувацький інтерфейс, який можна швидко розширювати відповідно до нових функціональних вимог[1].

Також важливим елементом є вровадження на ранньому етапі CI/CD (Continuous Integration / Continuous Delivery), що є надзвичайно важливим для сучасних застосунків, оскільки дозволяє автоматизувати процеси збірки, тестування та розгортання нових версій системи. Такий підхід забезпечує безперервну інтеграцію змін у коді та їх поступову доставку на віддалені сервери, що мінімізує ризики виникнення помилок у продакшні, тому що від цього напряму залежить стабільний потік клієнтів.

Особливо важливо, підкреслити що маленькі, але регулярні оновлення виявляються набагато ефективнішими за великі релізи.

Кожна невелика зміна легше тестується, швидше впроваджується та простіше відкатується у разі виявлення проблем. Крім того, постійні оновлення дозволяють швидко отримувати зворотний зв'язок від користувачів і адаптувати функціонал під нагальні потреби[3].

Додатково система має включати модуль інтеграції з Telegram, у рамках якого спеціальний бот автоматично надсилатиме сповіщення як користувачам, так і працівникам. Такий бот оперативно інформуватиме про нові бронювання, зміни в графіку роботи, нагадування та інші важливі події. Використання Telegram- це ефективне інфраструктурне рішення, яке забезпечить швидку та зручну комунікацію без потреби розробляти окремі механізми для доставки повідомлень, що суттєво економить час і ресурси, підвищуючи загальний рівень сервісу та взаємодії між адміністраторами й клієнтами.

Також окремо слід виділити модуль генерації контенту, що використовує існуючі лінгвістичні моделі для створення та оптимізації тематичних статей. Такий підхід дозволяє автоматизувати контент-маркетинг, швидко формувати матеріали для сайту та покращувати SEO-показники без значних витрат часу копірайтерів.

Отже, впровадження сучасної інформаційної системи з автоматизацією бронювання, аналітики та AI-підтримкою дозволяє підвищити ефективність сервісу, оптимізувати бізнес-процеси та забезпечити високий рівень задоволеності користувачів.

Список використаних джерел:

1. Next.js:: documentation [Електронний ресурс]. – Режим доступу https://nextjs.org/docs (дата звернення 21.11.2025)

2. Spring Boot: documentation [Електронний ресурс]. – Режим доступу: https://spring.io/projects/spring-boot (дата звернення: 25.02.2025).

3. Towards cost-benefit evaluation for continuous software engineering activities [Електронний ресурс]. – Режим доступу: https://link.springer.com/article/10.1007/s10664-022-10191-w (дата звернення 20.11.2025)