reality, and engineering, as well as by the critical role of dynamic simulation in fields where physical experimentation may be prohibitively expensive or technically infeasible.

Methods. *The Störmer-Verlet method is a numerical integration technique for solving Newton's equations of motion. Originally employed in 1791 by Jean Baptiste Delambre and subsequently refined by Verlet in the 1960s for molecular dynamics, the method provides excellent numerical stability along with time reversibility and preservation of the symplectic structure of phase space, without significant additional computational cost compared to the simple Euler method. A notable constraint is that it operates only with a fixed time step. All computational approaches based on the Verlet method treat simulated objects as systems of particles connected by flexible links rather than as rigid bodies. Based on the requirements of the implementation, the C programming language together with the raylib library were selected as the development tools.*

Results. *Two programs were developed and implemented. The first models a fountain: particles are generated with random masses and initial velocities scaled inversely by mass to ensure physical consistency, with coordinates updated each frame via Verlet integration according to Newton's second law. The second program models gravitational interaction between particles, computing pairwise gravitational forces and accumulating accelerations before updating positions to ensure physical synchronicity. A sub-stepping strategy with time increment $\Delta t = 1/steps$ and a custom double-precision vector type Vector2D were introduced to meet the higher accuracy demands of this task. A collision resolution function was also added, separating overlapping particle pairs by a distance proportional to their masses to realistically approximate momentum transfer, with iteration count bounded to prevent numerical instability.*

Conclusions. *The Störmer–Verlet method demonstrated high performance, sufficient accuracy, and ease of modification across all presented tasks, confirming its suitability for modeling dynamic objects. Particle systems, due to their flexibility, scalability, and ease of implementation, find broad application from cinematography and the gaming industry to science and engineering. Further research is expected to yield more realistic dynamic models and new opportunities for interactive real-time simulation with the involvement of artificial intelligence algorithms.*

Keywords: *Störmer-Verlet method, object modeling, particle systems, computer graphics, collision detection, numerical integration, cross-platform compatibility.*

Одержано редакцією 20.11.2023 р.
Прийнято до публікації 06.12.2023 р.

УДК 004.738.5

DOI 10.31651/2076-5886-2023-1-10-26

PACS 89.20.Ff

ВАСЕНКО Костянтин Олександрович
студент спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького

КРАСНОШЛИК Наталія Олександрівна
кандидат технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: krasnoshlyk@vu.cdu.edu.ua
ORCID 0000-0003-4661-6997

ВЕБ-ОРІЄНТОВАНА СИСТЕМА УПРАВЛІННЯ ЕЛЕКТРОННОЮ ЧЕРГОЮ

Статтю присвячено розробці веб-орієнтованої системи управління електронною

чергою, спрямованої на підвищення ефективності та якості обслуговування клієнтів в організаціях, що надають послуги. Актуальність теми зумовлена зростанням вимог до сервісу в умовах динамічної та конкурентної сфери послуг, а також необхідністю оптимізації процесів обслуговування. У статті проаналізовано наявні системи управління чергами, досліджено технології для розробки користувацького інтерфейсу (HTMX, JavaScript, Tailwind CSS), серверної частини (мова програмування Go) та бази даних (PostgreSQL). Архітектуру системи побудовано на основі патернів MVC, Observer та Singleton, що забезпечує модульність, масштабованість і підтримку комунікації у реальному часі засобами WebSocket. Розроблена система дозволяє підвищити надійність, зручність та швидкодію обслуговування, що сприяє зменшенню часу очікування клієнтів та покращенню їхньої задоволеності. Отримані результати можуть бути використані для впровадження ефективних електронних систем управління чергою в різних галузях та подальших досліджень у цій сфері.

Ключові слова: *веб-орієнтована система, управління електронною чергою, онлайн-сервіси, система керування чергою, автоматизація черг, Go, база даних PostgreSQL.*

Вступ

Сучасний розвиток технологій та вплив Інтернету на різні сфери життя призвели до того, що сфера послуг стала особливо динамічною та конкурентною. Для численних організацій та установ, що надають послуги, надзвичайно важливим стало забезпечення ефективного та зручного обслуговування клієнтів. Втрата високого рівня обслуговування може вести до втрати клієнтів, негативного досвіду та загальної незадоволеності. У цьому контексті розробка та впровадження веб-орієнтованої системи управління електронною чергою є важливим компонентом для покращення сфери послуг.

Мета статті – реалізація веб-орієнтованої системи управління електронною чергою, спрямованої на підвищення ефективності та зручності обслуговування клієнтів в організаціях.

Виклад основного матеріалу

1. Огляд предметної області та постановка задачі

Сфера послуг, також відома як третинний сектор, є важливою складовою сучасної економіки. Вона охоплює широкий спектр галузей, включаючи туризм, готельний та ресторанний бізнес, фінанси, освіту, охорону здоров'я та інформаційні технології. Його основною метою є задоволення потреб клієнтів шляхом надання високоякісних і часто нематеріальних послуг. Сектор еволюціонував з часом, адаптуючись до змін у суспільстві, технологіях та глобальній економіці. Його історичне коріння можна простежити в обміні товарами та допомозі в невеликих громадах, і з тих пір він став наріжним каменем економічного розвитку.

Останніми роками сектор послуг демонструє стабільне зростання, зумовлене змінами у способі життя, рівнях доходів та технологічним прогресом. Глобалізація та діджиталізація ще більше сприяли його розширенню, створюючи нові можливості, такі як електронна комерція та дистанційні послуги. Здатність сектору адаптуватися до цих змін має вирішальне значення для його подальшого успіху.

Традиційні системи керування чергою, хоча і є цінними для покращення обслуговування клієнтів, стикаються з низкою проблем, які впливають на якість обслуговування та рівень задоволеності клієнтів. Одним з основних недоліків є тривалий час очікування, який відчувають клієнти. Традиційні системи управління чергами часто не здатні оптимізувати розподіл часу та персоналу, що призводить до нерівномірного навантаження та затримок в обслуговуванні.

Вимога до клієнтів бути фізично присутніми в місцях обслуговування створює незручності. Крім того, відсутність інструментів для попередньої реєстрації та інформації про стан черги ще більше ускладнює процес обслуговування.

Ці проблеми призводять до незадоволення, шкоди репутації та втрати лояльності. Дослідження показують, що клієнти, які стикаються з тривалим часом очікування, з меншою ймовірністю повернуться, що підкреслює необхідність ефективних стратегій управління чергами [3-5].

Електронні черги є сучасним та ефективним рішенням проблем, пов'язаних з традиційним управлінням чергами. Такий підхід пропонує численні переваги, включаючи скорочення часу очікування, краще використання ресурсів та підвищення загальної ефективності обслуговування. Клієнти отримують вигоду від більшого контролю над своїм часом, гнучкості у плануванні та більш безперешкодного обслуговування.

Системи електронної черги продемонстрували успіх у таких галузях, як охорона здоров'я, роздрібна торгівля, державні послуги, фінанси та освіта. Приклади з різних секторів підкреслюють позитивний вплив на задоволеність клієнтів, ефективне управління ресурсами та покращення якості послуг [6-8].

Традиційні системи управління чергами, незважаючи на їх історичне значення, представляють значні виклики, які потребують наукової уваги [3]. Сучасний стан сфери послуг відображає її невід'ємну роль в економічному розвитку, а глобалізація та діджиталізація виступають каталізаторами її зростання. Необхідно усвідомлювати складний взаємозв'язок між сферою послуг, системами управління чергами та динамікою очікувань споживачів, що постійно змінюється. Вивчення психологічних і поведінкових аспектів незадоволеності клієнтів під час очікування може дати цінну інформацію для розробки ефективних систем управління чергами [3-5].

Перехід до електронних черг являє собою зміну парадигми в управлінні послугами. Електронні черги пропонують багатогранне рішення, вирішуючи недоліки традиційних систем. Подальше дослідження могло б заглибитися в нюанси впровадження в різних галузях, вивчаючи фактори, що сприяють успіху систем електронних черг.

Крім того, наукового дослідження потребує вплив електронних черг на ефективність організації, продуктивність працівників та загальний досвід клієнтів. Дослідження в цій галузі можуть включати розробку механізмів оцінки ефективності систем електронної черги та пропонування стратегій для безперешкодної інтеграції між різними секторами послуг.

Таким чином, у майбутньому електронне управління чергою має величезний потенціал, з перспективами подальшого технологічного розвитку та широкого впровадження. Еволюція сфери послуг у поєднанні з проблемами традиційного управління чергами відкриває широкі можливості для подальших досліджень у цій галузі.

Веб-орієнтована система управління електронною чергою втілює структуровану та масштабовану архітектуру, призначену для полегшення ефективної організації заходів та залучення користувачів.

Ефективне управління чергами подій є наріжним каменем функціональності. Система дозволяє динамічно створювати, змінювати та видаляти події. Події можуть бути пов'язані з різними місцями в черзі, кожен з яких відповідає певним часовим рамкам. Користувачі можуть зарезервувати місця для бажаних подій. Деталі подій, включаючи зображення, зберігаються в базі даних, що сприяє динамічному користувацькому інтерфейсу.

Користувачі мають доступ до персоналізованих профілів для перегляду майбутніх подій, зарезервованих місць і управління налаштуваннями облікового запису.

Комунікаційна архітектура системи здатна обробляти взаємодію в режимі реального часу. Використовуючи технологію WebSocket, сервер безперешкодно взаємодіє з підключеними клієнтами, забезпечуючи ефективній обробці паралельних запитів і полегшення комунікації в режимі реального часу.

Архітектура веб-додатку підкреслює добре структуровану модель даних, забезпечуючи узгодженість і керованість. Об'єкти, такі як події, місця в черзі, користувачі та сповіщення організовані систематично. Такий структурований підхід підвищує здатність системи до адаптації та масштабування в міру зростання набору даних. Ця організована модель даних є основою системи, що дозволяє ефективно знаходити інформацію та маніпулювати нею.

Використання JSON веб-токенів (JWT) забезпечує безпечну автентифікацію користувачів, сприяючи створенню надійного та захищеного користувацького середовища.

Включення фонових процесів додає системі рівень автоматизації. Система використовує паралельні підпрограми для ефективного моніторингу місць в чергах. Одна з них зосереджена на перевірці зарезервованих місць в черзі, час яких наближається до їх початку, сприяючи своєчасному сповіщенню користувачів, а інша — на автоматичному видаленні місць, коли їх кінцевий термін вже пройшов. Ці процеси підвищують оперативність і точність реагування системи, не погіршуючи користувацький досвід.

Веб-додаток має систему сповіщень в режимі реального часу, яка попереджає користувачів про наближення часу зарезервованого місця. Зв'язок через WebSocket забезпечує миттєві сповіщення підключеним клієнтам, а користувачі можуть налаштувати параметри сповіщень у налаштуваннях свого облікового запису.

Архітектура фронтенду надає пріоритет модульності. Шаблони, створені за допомогою сторонньої бібліотеки, абстрагують базові технології, що дозволило створити користувацький інтерфейс з урахуванням адаптивності, забезпечуючи послідовний і цікавий досвід роботи на різних пристроях і з різними розмірами екранів.

2. Архітектура проєкту

Веб-орієнтована система управління електронною чергою розроблена як монолітний веб-додаток. У монолітній архітектурі всі компоненти програми, включаючи інтерфейс користувача, бізнес-логіку та управління даними, тісно інтегровані в єдину кодову базу і працюють як єдиний веб-додаток. Такий підхід спрощує розробку та розгортання, але може мати проблеми з масштабуванням у міру зростання веб-додатку.

На рішення прийняти монолітну архітектуру вплинули такі фактори, як розмір проєкту, його складність та особисті уподобання. Монолітні архітектури часто підходять для малих і середніх додатків, де переваги простоти і легкості розробки переважують потенційні проблеми, пов'язані з масштабуванням і гнучкістю.

Проєкт має чітко визначену архітектуру, яка використовує патерни Model-View-Controller (MVC), Observer та Singleton для структурування та покращення різних аспектів веб-додатку (рис. 1).

Патерн "Модель-Представлення-Контролер" (MVC) [15]:

- Модель (M):
  - Модель інкапсулює дані та бізнес-логіку веб-додатку.

- Цей компонент включає такі сервіси, як управління користувачами, обробка подій, резервування слотів і сповіщення.
- Модель взаємодіє з базою даних, забезпечуючи пошук, оновлення та видалення даних.

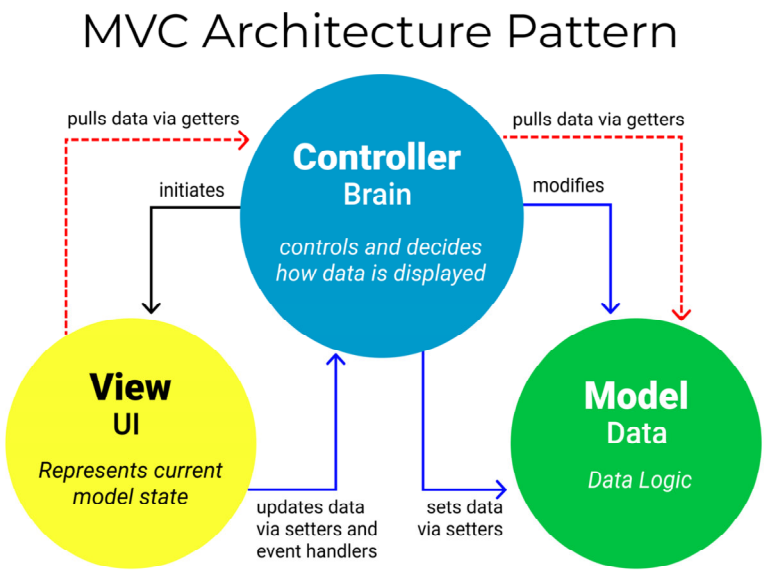

Рис. 1. Архітектурний шаблон MVC

– Представлення (V):
  - Представлення керує користувацьким інтерфейсом і логікою представлення.
  - Він включає HTML-шаблони, статичні ресурси (CSS, JS) і код на стороні клієнта.
  - Представлення отримує оновлення від контролера і відображає відповідну інформацію користувачам.
– Контролер (C):
  - Компонент Контролер обробляє дані, введені користувачем, оновлює Модель і повідомляє про зміни у Представлення.
  - Контролери обробляють HTTP-запити, виконують бізнес-логіку і оновлюють модель на основі взаємодії з користувачем.

Патерн Observer (Спостерігач) [2].

Патерн Спостерігач використовується для полегшення комунікації у реальному часі та оновлень у системі. З'єднання WebSocket діють як спостерігачі, підписуючись на відповідні події в веб-додатку. Модель (суб'єкт) сповіщає підключених клієнтів WebSocket (спостерігачів) про оновлення, забезпечуючи синхронізацію між користувачами в режимі реального часу.

Патерн Singleton [2].

Патерн Singleton використовується для забезпечення єдиного існування критично важливих компонентів у всьому веб-додатку. З'єднання з базою даних, кероване службою баз даних, реалізовано як синглтон, що забезпечує централізовану точку для взаємодії з базою даних.

Таким чином, поєднання патернів MVC, Observer та Singleton створює надійну архітектуру веб-додатку. Це сприяє наступному.

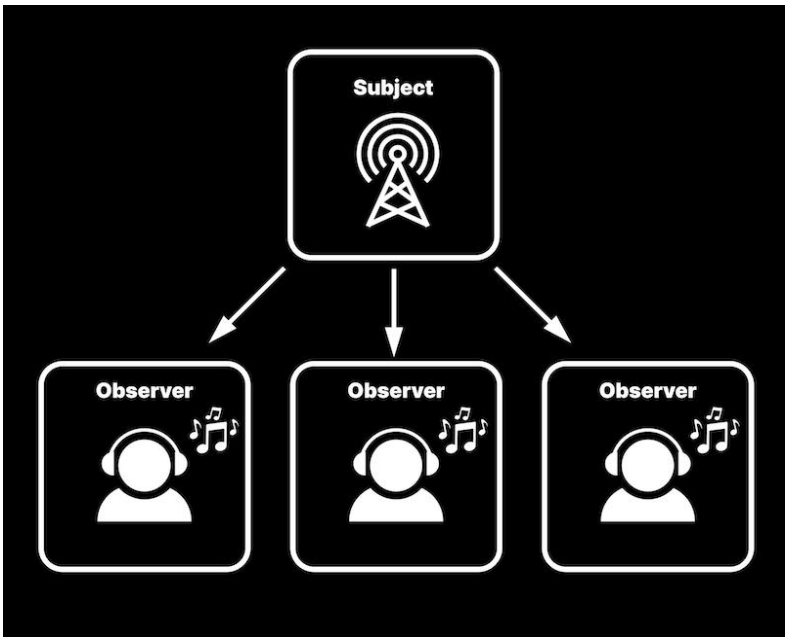

Рис. 2. Патерн Observer

1. Модульність та розподіл завдань.

Патерн MVC забезпечує чіткий розподіл завдань, дозволяючи незалежну розробку та підтримку кожного компонента. Модульність досягається за рахунок розподілу обов'язків між моделлю, представленням та контролером, що сприяє повторному використанню коду.

2. Спілкування в реальному часі.

Патерн Observer підвищує адаптивність користувацького інтерфейсу і забезпечує безперебійну роботу в реальному часі для користувачів, які очікують на сповіщення.

3. Централізоване управління ресурсами.

Доступ до налаштувань конфігурації або інших спільних ресурсів здійснюється через екземпляри Singleton, що забезпечує узгодженість та єдину точку контролю.

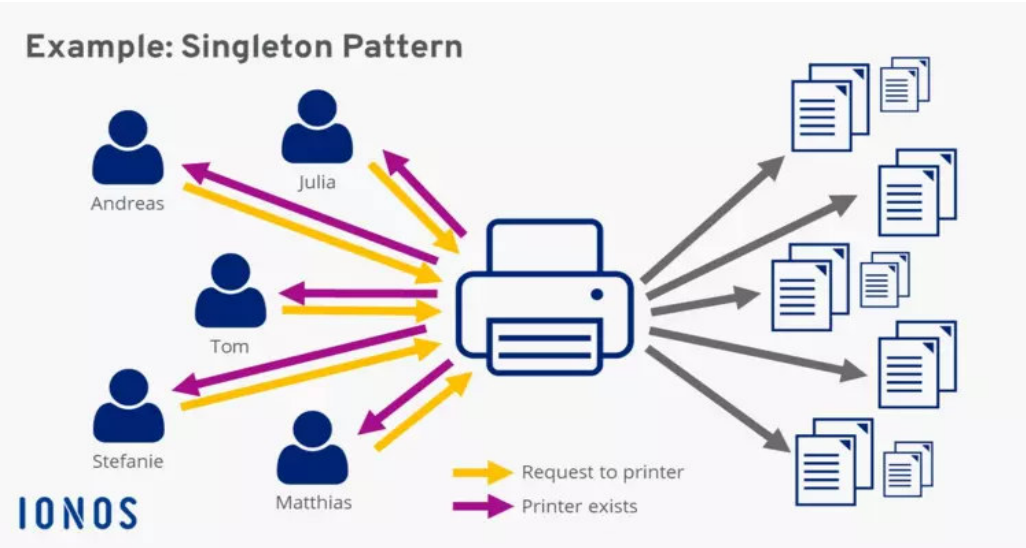

Рис. 3. Патерн Singleton

3. Технології для серверної сторони веб-сервісу

Мова Go, зазвичай відома як Golang, - це мова програмування з відкритим вихідним кодом, розроблена інженерами Google Робертом Гріземером, Робом Пайком та Кеном Томпсоном у 2009 році. Вона набула популярності завдяки своїй простоті та ефективності [1].

Переваги використання Go наступні:

- мінімалістичний синтаксис та простий підхід Go підвищують зрозумілість коду, сприяючи швидшій розробці та легшій підтримці;
- простота – Go скорочує час навчання, сприяючи швидкому освоєнню і роблячи її доступною для розробників;
- ідіоматичний стиль кодування – Go сприяє підтримці коду, роблячи перегляд коду простим, а помилки легко ідентифікуються;
- Go компілює безпосередньо в машинний код, забезпечуючи швидку та ефективну роботу двійкових файлів. Швидкий процес компіляції сприяє прискоренню циклів розробки;
- система статичної типізації Go виявляє помилки під час компіляції, підвищуючи надійність коду та зручність його підтримки;
- мова включає в себе обширну стандартну бібліотеку, яка охоплює різноманітні функціональні можливості, просуваючи філософію "батарейки в комплекті" для спрощення розробки;
- модель рівночасності (Concurrency) Go робить її придатною для додатків, що потребують паралельної обробки та обробки великої кількості одночасних з'єднань, роблячи її більш доступною та менш схильною до помилок, ніж традиційна багатопотоковість;
- Go постачається з чудовим інструментарієм, включаючи вбудоване тестування, інструменти форматування (go fmt) та утиліти для профілювання, що підвищує продуктивність та якість коду.

Мінуси використання Go:

- розгорнуте оброблення помилок у Go може бути багатослівною у порівнянні з мовами з виключеннями (exception), що потенційно забруднює код;
- хоча у Go є непогані веб-фреймворки, їх можна вважати менш розвинутими порівняно з аналогічними фреймворками в інших мовах. Постійний розвиток та внески спільноти спрямовані на вирішення цієї проблеми.

PostgreSQL, яку часто називають Postgres, – це надійна, багатофункціональна реляційна система управління базами даних з відкритим вихідним кодом, яка пропонує переконливе рішення для широкого спектру застосувань. Спочатку розроблена в Каліфорнійському університеті в Берклі, PostgreSQL за кілька десятиліть перетворилася на багатофункціональну і керовану спільнотою систему баз даних. Її гнучкість, дотримання стандартів і сильна підтримка спільноти роблять її кращим вибором для багатьох розробників. Хоча існують такі міркування, як крива навчання та потенційні нюанси продуктивності, загальні переваги часто переважають над цими занепокоєннями [12].

PostgreSQL розповсюджується на умовах ліцензії PostgreSQL, що дозволяє вільне використання, модифікацію та розповсюдження програмного забезпечення.

Переваги PostgreSQL:

- відкритий вихідний код PostgreSQL означає, що вона є вільно доступною, що робить її привабливим вибором для проєктів з обмеженим бюджетом;
- PostgreSQL суворо дотримується стандартів SQL, забезпечуючи сумісність з широким спектром додатків та інструментів;
- розробники можуть розширювати PostgreSQL, додаючи власні типи даних, функції та оператори, що робить її легко адаптованою до конкретних вимог проекту;
- відома своєю надійністю, PostgreSQL має репутацію системи, яка рідко дає збої або спричиняє пошкодження даних, забезпечуючи стабільне середовище для критично важливих додатків;
- PostgreSQL відмінно справляється з рівночасними транзакціями, зберігаючи при цьому відповідність стандартам ACID (атомарність, узгодженість, ізоляція, довговічність), що робить її придатною для додатків з високим рівнем одночасного використання;
- підтримка типів даних JSON/JSONB, віконні функції та розширені можливості індексування роблять PostgreSQL універсальною;
- PostgreSQL має активну спільноту, яка сприяє постійній розробці, вирішенню проблем та створенню розширень.

Недоліки PostgreSQL:

- для новачків в управлінні базами даних PostgreSQL може мати крутішу криву навчання порівняно з простішими системами;
- хоча PostgreSQL є потужною, деякі бенчмарки показують, що вона може бути дещо повільнішою, ніж деякі інші бази даних для певних робочих навантажень, хоча різниця в продуктивності може бути непомітною для невеликих проектів;
- хоча PostgreSQL уникає деяких проблемних поведінок за замовчуванням, які спостерігаються в інших базах даних, її оптимальне налаштування для конкретних випадків використання може вимагати певного досвіду.

PostgreSQL є відмінним варіантом для проєктів, які шукають надійну, розширювану систему управління реляційними базами даних з відкритим вихідним кодом.

4. Реалізація веб-сервісу

HTMX – це веб-бібліотека, призначена для спрощення розробки динамічних та інтерактивних веб-додатків, використовуючи можливості HTML, а не покладаючись на фреймворки JavaScript. Ця бібліотека має на меті забезпечити мінімалістичний і простий підхід до створення веб-додатків [10].

Переваги HTMX:

- HTMX забезпечує легкий і мінімалістичний підхід до створення динамічних веб-додатків. Він дозволяє розробникам покращити HTML за допомогою динамічної поведінки без необхідності використання великого коду JavaScript;
- HTMX залишається вірним синтаксису, орієнтованому на HTML, роблячи його доступним і читабельним. Такий підхід дозволяє розробникам зосередитися на структурі документа, що призводить до більш чистого і зручного для підтримки коду;
- HTMX пропонує вбудовану підтримку історії, що забезпечує безперешкодну навігацію та покращує користувацький досвід. Це особливо корисно для

односторінкових додатків, де користувачі очікують плавних переходів між різними представленнями;
- бібліотека спрощує процес збору вхідних значень з форм, полегшуючи обробку вхідних даних користувача без написання складного JavaScript коду;
- HTMX дозволяє розробникам використовувати фільтри подій, забезпечуючи контроль над тим, які події запускають запити. Це може підвищити продуктивність, зменшивши кількість непотрібних звернень до сервера;
- HTMX слідує філософії бути цілісною і відшліфованою функцією, яка може бути легко інтегрована в різні проєкти, сприяючи узгодженості і простоті використання;
- користувачі повідомляють про прискорення завантаження сторінок, зменшення використання пам'яті браузера та можливість ефективніше реалізовувати нові функції після переходу на HTMX;
- HTMX виявився цінним для нових програмістів, оскільки усуває необхідність у складному JavaScript-коді, дозволяючи їм зосередитися на своїх улюблених мовах програмування;
- реальні приклади демонструють успішний перехід з різних фреймворків на HTMX, підкреслюючи такі переваги, як швидше завантаження сторінок, зменшення використання пам'яті, менша кодова база та менша кількість залежностей від JavaScript.

Мінуси HTMX:

- HTMX не так широко відомий і прийнятий, як основні фреймворки для односторінкових додатків (SPA), такі як React або Vue.js. Це може бути недоліком для розробників, які шукають можливості на ринку праці, оскільки знання популярних фреймворків часто користуються більшим попитом;
- деякі розробники висловлюють занепокоєння щодо читабельності коду при використанні HTML. Підхід, орієнтований на HTML, може сприйматися як нетрадиційний, і людям, які звикли до традиційної розробки, орієнтованої на JavaScript, може бути складно адаптуватися;
- HTMX може не підходити для всіх типів проєктів. Хоча він чудово спрощує розробку певних веб-додатків, він може бути не найкращим варіантом для проєктів зі специфічними вимогами, які вимагають більш розширеної екосистеми JavaScript.

HTMX є чудовою альтернативою для розробників, які шукають спрощений, HTML-орієнтований підхід до створення динамічних веб-додатків. Хоча він може не підходити для кожного проєкту або вподобань розробника, його сильні сторони полягають у простоті, сумісності з серверними фреймворками та здатності надавати ефективну, динамічну веб-функціональність без значної залежності від JavaScript.

JavaScript, часто скорочено JS, – це мова програмування високого рівня, відома насамперед своїми можливостями у створенні динамічних та інтерактивних веб-додатків. Розроблена спочатку для веб-браузерів, вона перетворилася на універсальну мову, що використовується в різних сферах, включаючи веб-розробку, серверне програмування і навіть розробку десктопних додатків [11].

JavaScript продовжує залишатися домінуючою мовою для веб-розробки, оскільки переважна більшість веб-сайтів використовує її можливості для інтерактивності на стороні клієнта. Такі фреймворки, як React та Vue.js, зробили революцію у веб-розробці, запровадивши архітектуру на основі компонентів, спростивши розробку інтерфейсів та покращивши користувацький досвід. Node.js набув значної популярності, надаючи розробникам можливість використовувати JavaScript для

програмування на стороні сервера, уніфікуючи кодові бази та полегшуючи повностекову розробку.

Переваги JavaScript:

- JavaScript широко підтримується всіма браузерами, що робить його основною мовою для веб-розробки. Його універсальність виходить за межі Інтернету, а Node.js дозволяє писати сценарії на стороні сервера;
- асинхронна природа JavaScript дозволяє виконувати операції без блокування, забезпечуючи безперебійну та швидку роботу користувачів завдяки одночасному виконанню декількох завдань;
- мова може похвалитися великою екосистемою бібліотек і фреймворків (наприклад, React, Angular, Vue.js), які задовольняють різні потреби розробників, прискорюючи розробку і розширюючи функціональність;
- JavaScript дозволяє маніпулювати об'єктною моделлю документа (DOM), що дозволяє динамічно змінювати вміст, структуру та стиль веб-сторінки на основі взаємодії з користувачем;
- спільнота JavaScript є потужною, пропонує велику документацію, форуми та онлайн-ресурси, які допомагають у навчанні та вирішенні проблем;
- здатність JavaScript працювати на різних платформах полегшує створення крос-платформних додатків зі спільними кодовими базами.

Недоліки JavaScript:

- хоча JavaScript широко підтримується, неузгодженість між браузерами може призвести до проблем сумісності, що вимагатиме додаткового коду для забезпечення кросбраузерної функціональності;
- керування асинхронними операціями за допомогою зворотних викликів (callback) або обіцянок (Promise) може призвести до створення складних структур коду, що потенційно може спричинити проблеми з читабельністю та обслуговуванням;
- як клієнтська мова, JavaScript може бути вразливою до вразливостей безпеки, таких як міжсайтовий скриптинг (XSS), якщо не захищена належним чином;
- значна залежність від JavaScript при виконанні великих операцій може вплинути на час завантаження сторінки та загальну продуктивність, що вимагає стратегій оптимізації;
- гнучкість JavaScript та екосистема, що постійно розвивається, можуть стати складним завданням для початківців, особливо при роботі з різними бібліотеками та фреймворками.

Tailwind CSS – це CSS-фреймворк, який спрощує процес стилізації веб-додатків, надаючи набір попередньо визначених класів утиліт. Розробники часто повідомляють про підвищення продуктивності та пришвидшення циклів розробки при використанні Tailwind завдяки інтуїтивно зрозумілим класам утиліт та узгодженим шаблонам проєктування [9].

Tailwind отримує високу оцінку за надання позитивного досвіду розробки, особливо для проєктів, де швидке створення прототипів та ітерації мають вирішальне значення. Tailwind був прийнятий великими компаніями та проєктами, демонструючи його придатність для великомасштабних додатків. Приклади включають OpenAI, GitHub та багато інших.

Переваги Tailwind CSS:

- підхід Tailwind, орієнтований на утиліти, прискорює розробку, пропонуючи повний набір класів для загальних стилів, усуваючи необхідність у великому користувацькому CSS;
- фреймворк підтримує єдину мову дизайну для всіх проєктів, полегшуючи командам співпрацю та підтримуючи єдиний візуальний стиль;
- Tailwind включає адаптивні класи утиліт, які спрощують створення адаптивних макетів, забезпечуючи оптимальний користувацький досвід на різних пристроях і розмірах екранів;
- надаючи багатий набір налаштувань за замовчуванням, Tailwind дозволяє розробникам легко налаштовувати стилі, конфігуруючи фреймворк відповідно до конкретних вимог проекту;
- простота Tailwind робить його доступним для розробників з різним рівнем досвіду. Початківці можуть швидко освоїти фреймворк і використовувати його, в той час як досвідчені користувачі можуть скористатися його гнучкістю;
- Tailwind має активну спільноту, яка пропонує ресурси, плагіни та розширення, що розширюють його можливості. Екосистема підтримує постійне навчання та вдосконалення.

Недоліки Tailwind CSS:

- широке використання утилітарних класів у HTML-файлах може призвести до безладу в розмітці, що погіршує читабельність. Розробники повинні дотримуватися балансу між корисністю та зручністю обслуговування;
- деякі розробники віддають перевагу традиційному підходу до написання власного, ручного CSS для точного контролю над стилем. Підхід Tailwind, орієнтований на практичність, може не відповідати уподобанням кожного;
- хоча Tailwind простий у використанні, освоєння розширених функцій і налаштувань може вимагати глибшого розуміння конфігурації та опцій фреймворку.

5. Функціональні можливості веб-додатку

Розробка серверної частини веб-додатку відбувається за модульною та масштабованою архітектурою. Для створення надійного веб-сервера використовується мова програмування Go та веб-інструментарій Gorilla. Архітектура сервера складається з декількох компонентів, серед яких можна виділити наступні.

Основна функція ініціалізує сервер, створює екземпляри контролерів, обробників і сервісів, а також встановлює необхідні конфігурації.

Маршрутизатор та проміжне програмне забезпечення (middleware): маршрутизатор Gorilla Mux використовується для обробки вхідних HTTP-запитів і перенаправлення їх до відповідних обробників; проміжне програмне забезпечення, наприклад, для автентифікації, реалізовано для виконання дій перед передачею запитів до фактичних обробників.

Служба бази даних: сервер підключається до бази даних PostgreSQL за допомогою бібліотеки GORM ORM. Пакет бази даних містить службу бази даних, яка відповідає за ініціалізацію бази даних, перевірку працездатності та надання посилання на екземпляр бази даних GORM.

Сервер WebSocket: до складу сервера входить сервер WebSocket для полегшення зв'язку з клієнтами в режимі реального часу. З'єднання WebSocket оновлюються за допомогою бібліотеки Gorilla WebSocket.

Серверна розробка веб-додатку зосереджена навколо добре структурованої та модульної архітектури, що підтримує CRUD-операції через RESTful кінцеві точки

(endpoints). Реалізація забезпечує масштабованість, ремонтопридатність та можливості комунікації в режимі реального часу за допомогою технології WebSocket. Такий підхід полегшує безперебійну взаємодію між клієнтами та сервером, забезпечуючи надійну основу для системи управління чергою [13,14].

Цей набір маршрутів надає повну функціональність для взаємодії з системою. Зокрема, можливості керування користувачами, а саме реєстрація нових користувачів, авторизація та вихід із системи (табл. 1).

Таблиця 1

Керування користувачами		
/login	GET/POST	Обробляє вхід користувача
/signup	GET/POST	Керує реєстрацією користувачів
/logout	POST	Керує виходом користувача з системи

Тут представлені маршрути, які дозволяють створювати, редагувати, видаляти події та отримувати інформацію про них (табл. 2). Також є можливість завантаження зображень для подій та відображення сторінкованих результатів.

Таблиця 2

Керування подіями		
/event/new	GET	Відображає форму для створення нової події
/event	POST	Створює нову подію
/event/{id}/upload	GET	Відображає форму для завантаження зображень події
/event/{id}/edit	GET	Створює форму для редагування існуючої події
/event	PUT	Оновлює існуючу подію
/event/{id}	GET	Отримати детальну інформацію про певну подію

Продовження таблиця 2

/events/{page}	GET	Отримує посторінкові події
/event/{id}	DELETE	Видаляє певну подію

Таблиця 3 містить маршрути для резервування, скасування та управління місцями. Вони дозволяють додавати та видаляти місця в черзі для подій, а також отримувати інформацію про них.

У таблиці 4 зібрано маршрути для управління профілем користувача, його подіями, налаштуваннями, параметрами безпеки та зовнішнім виглядом. Це надає можливість перегляду, редагування та видалення облікового запису, а також налаштування різних аспектів облікового запису.

Присутні також маршрути для отримання списку сповіщень та позначення їх як прихованих (табл. 5). Це дозволяє користувачеві керувати своїми сповіщеннями у системі.

Таблиця 3

Керування місцями		
/slot/{id}/reserve	PUT	Резервує місце
/slot/{id}/cancel	PUT	Звільняє зарезервоване місце
/event/{id}/add	GET	Відображає форму для додавання місць до події
/event/{id}/slots	GET	Отримує місця для певної події
/event/add	POST	Створює нове місце в черзі
/slot/{id}	DELETE	Видаляє певне місце в черзі

Таблиця 4

Керування користувачами		
/user	GET	Відображає профіль користувача
/user/{id}	DELETE	Видаляє обліковий запис користувача
/user/events	GET	Відображає події створені користувачем
/user/slots	GET	Відображає зарезервовані користувачем місця
/user/settings	GET	Відображає налаштування користувача
/user/account	GET/PUT	Керує налаштуваннями облікового запису користувача
/user/security	GET/PUT	Керує параметрами безпеки користувача
/user/appearance	GET	Відображає налаштування зовнішнього вигляду користувача
/user/notifications	GET/PUT	Керує налаштуваннями сповіщень користувача

Таблиця 5

Керування сповіщеннями		
/notifications	GET	Відображає список сповіщень
/notifications/{id}	PUT	Позначає сповіщення як приховане

У таблиці 6 можна побачити маршрути для роботи з WebSocket, які використовуються для передачі сповіщень у реальному часі, а також для завантаження зображень на сервер.

Таблиця 6

Інші маршрути		
/ws	WebSocket	Керує з'єднаннями WebSocket для сповіщень у реальному часі
/upload	POST	Зберігає зображення на сервері

Кожен маршрут має відповідний HTTP метод та функціональність, яка дозволяє виконувати певні операції з даними. Такий підхід забезпечує гнучкість та розширюваність системи для майбутніх змін та вдосконалень.

Проміжне програмне забезпечення відіграє ключову роль в обробці автентифікації, оновленні контексту запиту, управлінні файлами cookie та покращенні загальної функціональності сервера. Ці компоненти проміжного програмного забезпечення в сукупності сприяють створенню безпечної, зручної та ефективної системи. Основні його обов'язки включають наступне.

1. Певні статичні шляхи (наприклад, /css, /img) виключаються з автентифікації, щоб дозволити публічний доступ.

2. Обробка файлів cookie:

- проміжне програмне забезпечення перевіряє наявність токенів автентифікації (Access-Token і Refresh-Token) у вхідних запитах;
- якщо токени не знайдено, для початкового запиту встановлюється шлях за замовчуванням (/) для перенаправлення користувача після успішної аутентифікації.

3. Статус автентифікації:

- проміжне програмне забезпечення перевіряє автентичність токена доступу, дозволяючи доступ лише автентифікованим користувачам;
- якщо токен недійсний або термін його дії закінчився, проміжне програмне забезпечення намагається оновити токени за допомогою Refresh-Token.

4. Контекст і оновлення файлів cookie:

- проміжне програмне забезпечення оновлює контекст запиту зі статусом автентифікації та ідентифікатором користувача, дозволяючи наступним обробникам отримати доступ до цієї інформації;
- у разі оновлення токенів проміжне програмне забезпечення оновлює файли cookie з новими значеннями Access-Token і Refresh-Token;
- при виході з системи або виникненні проблем з автентифікацією, проміжне програмне забезпечення очищає файли cookie, забезпечуючи чистий стан.

5. Інтеграція проміжного програмного забезпечення в маршрутизацію:

- проміжне програмне забезпечення додається до маршрутизатора, щоб забезпечити його виконання до того, як воно досягне фактичних обробників запитів;
- проміжне ПЗ визначає дозволи користувача і діє відповідно до них;
- проміжне програмне забезпечення керує файлом cookie початкового шляху, зберігаючи попереднє відвідане місце користувача перед автентифікацією.

Клієнтська частина програми розроблена для простоти та реактивності. Вона генерується серверним рендерингом з використанням HTML-шаблонів. Такий підхід гарантує, що контент динамічно створюється на сервері до того, як він потрапить до клієнта.

Щоб покращити серверний HTML, ми інтегрували HTMX, легку бібліотеку JavaScript. Вона полегшує динамічну взаємодію, дозволяючи оновлювати певні частини сторінки без необхідності перезавантаження всієї сторінки. Це забезпечує більш плавну та швидку роботу користувача.

Tailwind CSS використовується для стилізації, забезпечуючи підхід, орієнтований на практичність. Цей фреймворк спрощує процес стилізації, дозволяючи вносити зміни в дизайн швидко і послідовно. Tailwind покращує візуальну привабливість клієнтських компонентів.

JavaScript лежить в основі покращення поведінки на стороні клієнта. Ця універсальна мова використовується для додавання інтерактивності, обробки користувацького вводу та покращення загальної динамічної поведінки системи. Крім того, окремі бібліотеки JavaScript вибірково інтегруються для забезпечення унікальної функціональності.

Технологія WebSocket інтегрована для встановлення з'єднання між клієнтом і сервером у режимі реального часу. Це забезпечує миттєві оновлення та сповіщення, сприяючи більш інтерактивному та динамічному користувацькому досвіду. WebSocket підвищує ефективність зв'язку, виходячи за рамки традиційних циклів запит-відповідь.

Події HTMX і дії на стороні сервера використовуються для обробки взаємодії з користувачем, запускаючи динамічні оновлення на основі даних, введених користувачем. Користувацькі клієнтські скрипти використовуються вибірково для задоволення конкретних вимог, таких як перевірка форм, обробка даних або управління змінами стану на стороні клієнта.

Таким чином, ці технології разом використовуються для створення адаптивної, динамічної та безпечної системи управління електронною чергою.

Висновки

У даній роботі була проведена розробка веб-орієнтованої системи управління електронною чергою з метою підвищення ефективності та зручності обслуговування клієнтів в організаціях. Результати дослідження свідчать про важливість впровадження сучасних технологій у сферу послуг для підтримки конкурентоспроможності та задоволення потреб сучасного споживача.

Розробка веб-додатку включала в себе глибоке вивчення процесів розробки користувацької та серверної частини, а також впровадження системи управління базою даних. Результатом є зручний користувацький інтерфейс та надійна серверна частина, що сприятиме оптимальній взаємодії з різними компонентами системи.

Практичне значення отриманих результатів полягає в тому, що впровадження розробленої системи управління електронною чергою може значно підвищити якість обслуговування клієнтів для численних організацій у сфері послуг. Зменшення часу очікування та підвищення загальної задоволеності клієнтів сприятиме збереженню та приваблению нових клієнтів, підвищенню конкурентоспроможності та покращенню репутації компаній.

Отже, результати розробки можуть слугувати основою для подальших удосконалень у сфері управління чергою та впровадження подібних систем у різних галузях послуг, сприяючи подальшому розвитку сучасних технологій та покращенню якості обслуговування клієнтів.

Список використаної літератури:

1. The Go Programming Language: Documentation [Електронний ресурс]. – Режим доступу: https://go.dev/doc/.
2. Refactoring Guru: The Catalog of Design Patterns [Електронний ресурс]. – Режим доступу: https://refactoring.guru/design-patterns/catalog.
3. Yusuf M. O., Blessing N., Kazeem A. O. Queuing Theory and Customer Satisfaction: A Review of Performance, Trends and Application in Banking Practice (A Study of First Bank Plc Gwagwalada, Abuja Branch) // European Journal of Business and Management. – 2015. – Vol. 7, № 35.
4. Desta Al. Z., Belete T. H. The Influence of Waiting Lines Management on Customer Satisfaction in Commercial Bank of Ethiopia // Financial Markets, Institutions and Risks. – 2019. – Vol. 3, № 3. – P. 5–12.

5. Bidari A., Jafarnejad S., Alaei Faradonbeh N. Effect of Queue Management System on Patient Satisfaction in Emergency Department; a Randomized Controlled Trial // Archives of Academic Emergency Medicine. – 2021.
6. Qtrac: 7 Examples of Successful Enterprise Queuing Solutions [Електронний ресурс]. – Режим доступу: https://qtrac.com/blog/7-examples-of-successful-enterprise-queuing-solutions/.
7. Міграційна служба Рівненщини відновила сервіс «Електронна черга» [Електронний ресурс] / Рівненська обласна державна адміністрація. – Режим доступу: https://www.rv.gov.ua/news/na-rivnenshchini-vidnovleno-robotu-servisu-elektronna-cherga.
8. У Львівській ОДА відбулось засідання робочої групи щодо впровадження електронної черги на пункті пропуску «Краковець-Корчова» [Електронний ресурс] / Львівська обласна державна адміністрація. – Режим доступу: https://old.loda.gov.ua/news?id=47088.
9. Tailwind CSS: Utility-First Fundamentals [Електронний ресурс]. – Режим доступу: https://tailwindcss.com/docs/utility-first.
10. HTMX: Essays [Електронний ресурс]. – Режим доступу: https://htmx.org/essays/.
11. Mozilla Developer Network: JavaScript [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.
12. Riggs S., Ciolli G. PostgreSQL 14 Administration Cookbook: Over 175 Proven Recipes for Database Administrators to Manage Enterprise Databases Effectively. – Birmingham : Packt Publishing, 2022. – 608 p.
13. Sturgeon P., Bohill L. Build APIs You Won't Hate: Everyone and Their Dog Wants an API, So You Should Probably Learn How to Build Them. – [s.l.] : Philip J. Sturgeon, 2015. – 188 p.
14. Microsoft REST API Guidelines [Електронний ресурс] / Microsoft. – Режим доступу: https://github.com/microsoft/api-guidelines/blob/vNext/Guidelines.md.
15. Mozilla Developer Network: MVC [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/MVC.

References:

1. The Go Programming Language. Documentation. Retrieved from https://go.dev/doc/
2. Refactoring Guru. The catalog of design patterns. Retrieved from https://refactoring.guru/design-patterns/catalog
3. Yusuf, M. O., Blessing, N., & Kazeem, A. O. (2015). Queuing theory and customer satisfaction: A review of performance, trends and application in banking practice (A study of First Bank Plc Gwagwalada, Abuja Branch). European Journal of Business and Management, 7(35).
4. Desta, A. Z., & Belete, T. H. (2019). The influence of waiting lines management on customer satisfaction in Commercial Bank of Ethiopia. Financial Markets, Institutions and Risks, 3(3), 5-12.
5. Bidari, A., Jafarnejad, S., & Alaei Faradonbeh, N. (2021). Effect of queue management system on patient satisfaction in emergency department: A randomized controlled trial. Archives of Academic Emergency Medicine.
6. Qtrac. 7 examples of successful enterprise queuing solutions. Retrieved from https://qtrac.com/blog/7-examples-of-successful-enterprise-queuing-solutions/
7. Rivne Regional State Administration. The Migration Service of Rivne Region has resumed the "Electronic Queue" service. Retrieved from https://www.rv.gov.ua/news/na-rivnenshchini-vidnovleno-robotu-servisu-elektronna-cherga
8. Lviv Regional State Administration. A working group meeting on the implementation of the electronic queue at the "Krakovec-Korchowa" checkpoint was held in Lviv RSA. Retrieved from https://old.loda.gov.ua/news?id=47088
9. Tailwind CSS. Utility-first fundamentals. Retrieved from https://tailwindcss.com/docs/utility-first
10. HTMX. Essays. Retrieved from https://htmx.org/essays/
11. Mozilla Developer Network. JavaScript. Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript
12. Riggs, S., & Ciolli, G. (2022). PostgreSQL 14 administration cookbook: Over 175 proven recipes for database administrators to manage enterprise databases effectively. Packt Publishing.
13. Sturgeon, P., & Bohill, L. (2015). Build APIs you won't hate: Everyone and their dog wants an API, so you should probably learn how to build them. Philip J. Sturgeon.
14. Microsoft. Microsoft REST API guidelines. Retrieved from https://github.com/microsoft/api-guidelines/blob/vNext/Guidelines.md
15. Mozilla Developer Network. MVC. Retrieved from https://developer.mozilla.org/en-US/docs/Glossary/MVC

VASENKO Kostiantyn,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
KRASNOSHLYK Nataliya,
Candidate of Technical Sciences, Associate Professor, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

WEB-ORIENTED ELECTRONIC QUEUE MANAGEMENT SYSTEM

Summary. Introduction. *This article presents the development of a web-based electronic queue management system aimed at improving the efficiency and quality of customer service in organizations that provide services. The relevance of the topic is driven by increasing service demands in a dynamic and competitive services sector, as well as the need to optimize service processes. The article analyzes existing queue management systems and explores technologies for developing the user interface, server-side, and database. The proposed solutions enhance the system's reliability, usability, and performance, ultimately reducing customer wait times and improving customer satisfaction. The results obtained can be applied for the implementation of effective electronic queue management systems in various industries and for further research in this area.*

The modern development of technologies and the influence of the Internet on various areas of life have made the service sector particularly dynamic and competitive. For numerous organizations and institutions that provide services, ensuring effective and convenient customer service has become extremely important. A decline in service quality can lead to the loss of customers, negative experiences, and overall dissatisfaction. In this context, the development and implementation of a web-based electronic queue management system is a crucial component for improving the service industry.

The purpose *of this article is to develop a web-based electronic queue management system aimed at improving the efficiency and convenience of customer service in organizations.*

Results. *In this work, a web-based electronic queue management system was developed with the aim of enhancing the efficiency and convenience of customer service in organizations. The results indicate the importance of implementing modern technologies in the service sector to support competitiveness and meet the needs of the modern consumer.*

The development of the web application included a thorough study of the processes for developing both the client and server sides, as well as the implementation of a database management system. The outcome is a user-friendly interface and a reliable server component that facilitates optimal interaction with various components of the system.

The practical significance of the obtained results lies in the fact that the implementation of the developed electronic queue management system can significantly improve the quality of customer service for numerous organizations in the service sector. Reducing waiting times and increasing overall customer satisfaction will contribute to retaining and attracting new customers, enhancing competitiveness, and improving the reputation of companies.

Conclusion. *Thus, the results of this research can serve as a foundation for further improvements in queue management and the implementation of similar systems in various service sectors, fostering the continued development of modern technologies and improving the quality of customer service.*

Keywords: *web-oriented system, electronic queue management, online services, queue management system, queue automation, Go, PostgreSQL database.*

Одержано редакцією 15.11.2023 р.
Прийнято до публікації 06.12.2023 р.