МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Черкаський національний університет
імені Богдана Хмельницького
MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
Bohdan Khmelnytsky
National University of Cherkasy

ISSN 2076-5886 (Print)

ВІСНИК ЧЕРКАСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО

Серія
ПРИКЛАДНА МАТЕМАТИКА. ІНФОРМАТИКА

BULLETIN OF THE CHERKASY BOHDAN KHMELNYTSKY NATIONAL UNIVERSITY

APPLIED MATHEMATICS. INFORMATICS

ВИПУСК 1
ISSUE 1

Черкаси, 2023
Cherkasy, 2023

Засновник, редакція, видавець і виготовлювач –
Черкаський національний університет імені Богдана Хмельницького.
Свідоцтво про державну реєстрацію КВ № 16161-4633 ПР від 11.12.2009.
Свідоцтво про державну реєстрацію КВ № 23973-13813 Р від 21.05.2019.

Журнал розрахований на математиків, спеціалістів у галузі ІТ, викладачів, науковців, аспірантів, студентів.

Випуск № 1 наукового журналу «Вісник Черкаського національного університету імені Богдана Хмельницького. Серія «Прикладна математика. Інформатика» рекомендовано до друку та до поширення через мережу Інтернет Вченою радою Черкаського національного університету імені Богдана Хмельницького (протокол № 6 від 21.12.2023 року).

Журнал індексується Google Scholar.

За дотримання права інтелектуальної власності, достовірність матеріалів та обґрунтування висновків відповідають автори.

Адреса редакційної колегії:
18031, Черкаси, бул. Шевченка, 79
Черкаський національний університет імені Богдана Хмельницького, корпус № 3, к. 307
e-mail: ami.cdu@ukr.net

З електронною версією журналу можна ознайомитися за адресою: http://ami-ejournal.cdu.edu.ua/

Founder, editorial, publisher and manufacturer –
Bohdan Khmelnytsky National University of Cherkasy.
State registration certificate: KV No. 16161-4633 PR dated 11.12.2009.
State registration certificate: KV No. 23973-13813 R dated 21.05.2019.

This journal is meant for mathematicians, IT specialists, teachers, researchers, postgraduates and students.

Issue № 1 of the scientific journal «Bulletin of the Cherkasy Bohdan Khmelnytsky national university. Applied mathematics. Informatics» is recommended for publication and dissemination through the Internet by the Academic Council of Bohdan Khmelnytsky National University of Cherkasy (protocol number 6 dated 21.12.2023).

The journal is indexed in Google Scholar.

The authors are responsible for the observance of the intellectual property right, for the reliability of the materials and for the substantiation of the conclusions.

Editorial office address:
18031, Cherkasy, Shevchenko Blvd., 79
Bohdan Khmelnytsky National University of Cherkasy, building 3, ap. 307
e-mail: ami.cdu@ukr.net

All electronic versions of articles are available on the website edition http://ami-ejournal.cdu.edu.ua/

СЕКЦІЯ «ПРИКЛАДНА МАТЕМАТИКА»

УДК 519.6:004.92

DOI 10.31651/2076-5886-2023-1-4-10

PACS 02.60.Cb, 45.50.-j, 07.05.Tp

ДЗЮБА Вікторія Анатоліївна
кандидат технічних наук, викладач кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: viktoriya.dzyuba15@gmail.com
ORCID 0000-0003-1655-0333

ЧАЛИЙ Антон Анатолійович
студент спеціальності «Прикладна математика» Черкаського національного університету імені Богдана Хмельницького
e-mail: anton.chaliy@gmail.com

ЗАСТОСУВАННЯ СИСТЕМ ЧАСТИНОК ДЛЯ МОДЕЛЮВАННЯ ОБ'ЄКТІВ ДИНАМІЧНОЇ ПРИРОДИ

У статті досліджується застосування систем частинок для моделювання об'єктів динамічної природи з використанням методу Стьормера–Верле. Система частинок є ефективним інструментом для створення реалістичних моделей складних природних явищ, таких як потоки рідини, диму, вибухи та інші динамічні процеси. Метод Стьормера–Верле, який використовується для інтегрування рівнянь руху, забезпечує високу чисельну стійкість і точність, що пояснює його успішне практичне застосування для задач молекулярної динаміки та комп'ютерної графіки.

Представлено архітектуру програми для моделювання фонтану та гравітаційної взаємодії частинок, реалізованої за допомогою мови програмування C та бібліотеки raylib. Основні етапи роботи системи включають генерацію, оновлення стану, візуалізацію та відображення частинок, що дозволяє точно моделювати їхню фізичну взаємодію та змінювати параметри. Було також додано обчислення колізій між частинками, що дозволяє більш реалістично передавати імпульс між об'єктами.

Ключові слова: *метод Стьормера-Верле, моделювання об'єктів, системи частинок, комп'ютерна графіка, колізії, мультиплатформність.*

Вступ

Моделювання об'єктів динамічної природи – одна з найскладніших задач у комп'ютерній графіці, фізиці та інженерії, після чого такі об'єкти мають високий ступінь складності та варіативності. Динамічні системи – це, зокрема, потоки рідини і газів, процеси горіння, еластичні та в'язкопружні об'єкти, вибухи, дим, дощ і сніг. Моделювання таких об'єктів у реальному часі, із врахуванням їхньої фізичної поведінки та візуальної достовірності, є ключовим аспектом сучасної анімації, симуляції та ігрових механізмів. У такому контексті системи частинок зарекомендували себе як ефективний підхід для створення реалістичних моделей об'єктів динамічної природи. Оскільки, структура такої системи розроблена для опису та управління великою кількістю дрібних часток, які разом представляють складний

об’єкт або явище. Тобто, такий підхід дає змогу моделювати об’єкти та процеси, які складно представити традиційними способами моделювання, що дозволяє досягти високого рівня деталізації [1].

Сучасні технології все частіше потребують реалістичної та ефективної візуалізації природних явищ та динамічних процесів, які мають складні фізичні властивості і постійно змінюються в часі. Саме тому пошук нових методів та вдосконалення існуючих підходів до моделювання таких об’єктів є критично важливим завданням для науковців, інженерів та розробників. Актуальність дослідження підсилюється потребою у реалістичності симуляцій для таких сфер, як кінематограф, комп'ютерні ігри, віртуальна реальність та інженерія. Здатність моделювати природні явища з високою точністю дає можливість створювати реалістичні та захоплюючі візуальні ефекти, підвищувати залучення користувачів та покращувати досвід взаємодії з системою. Крім того, симуляція динамічних процесів відіграє важливу роль у наукових та інженерних дослідженнях, зокрема в галузях, де експериментальні дослідження можуть бути складними або навіть неможливими через складні умови або високі витрати.

Таким чином, дослідження у сфері моделювання об’єктів динамічної природи є актуальним як для практичних програм у сучасних технологіях, так і для фундаментальних наукових досліджень.

Метою статті є розробка та реалізація програм для моделювання об’єктів динамічної природи, за допомогою методу Стьормера-Верле.

Виклад основного матеріалу

Метод Стьормера-Верле це чисельний метод, який використовується для інтегрування Ньютонівських рівнянь руху. Він часто використовується для розрахунку траєкторій частинок у моделюванні молекулярної динаміки та комп'ютерній графіці. Алгоритм був вперше використаний у 1791 році Жаном Батистом Деламбре і з тих пір був модифікований багато разів, востаннє Лу Верле в 1960-х роках для використання в молекулярній динаміці. Інтегрування по Верле забезпечує хорошу чисельну стійкість, а також інші властивості, важливі для фізичних систем, такі як оборотність у часі та збереження симплектичної форми на фазовому просторі, без значних додаткових обчислювальних витрат порівняно з простим методом Ейлера [2, 3].

Відмітимо специфіку використання розрахункових підходів для методу Верле – всі вони є системами частинок, об’єднаними гнучкими зв’язками, а не твердими тілами [4, 5].

Серед недоліків цього методу є те, що він працює лише з постійним кроком. Для накладання обмежень на систему частинок можна вираховувати сили, що діють на кожну окрему частинку і відповідно визначати прискорення, що діє на неї в той момент. Таким чином створюючи, наприклад пружину.

На основі поставленої мети, були висунуті такі вимоги до технології реалізації: можливість використання складних структур даних; зручність у роботі з векторами; наявність бібліотек для роботи з графікою; мультиплатформність; простий синтаксис. Враховуючи вимоги було обрано мову програмування C та бібліотеку raylib.

Розглянемо тривіальний приклад програми, в якій потрібно реалізувати модель «фонтану». Основні вимоги до структури програми наступні:

- малювати частинки з заданими характеристиками у вікні;
- створювати та видаляти частинки;
- рахувати їх координати в кожному моменті часу залежно від координат в попередньому та сил, що на них діють;

Архітектурно програма складатиметься із наступних компонентів: ініціалізації, головного циклу, функції для створення частинок, функції для видалення частинок, функції для оновлення фізики системи частинок та функції для відображення.

Для простоти частинки зберігатимуться в однозв'язному списку, а самі вони будуть структурами типу Particle з полями, які визначають поточні координати, попередні координати та масу, від якої також залежить їх розмір при відображені. Головний цикл працює з частотою 60 кадрів на секунду, обробляє завершення програми та викликає функції update_particles та draw_particles.

У функції update_particles викликається процедура генерації частинок – emit_particles в якій створюється п'ять частинок з координатами внизу середини екрана, випадковою масою, та випадкові попередні координати, які залежать від маси та визначають початкові напрямок руху та швидкість. Якщо всім точкам з різною масою задати однакову швидкість, то виходитиме, що частинки з більшою масою на початку матимуть незрівнянно більшу кінетичну енергію і вилітатимуть за межі екрану, тоді як малі частинки майже одразу падатимуть. Для виправлення цього, використано вектор швидкості, за допомогою якого вираховуються і записуються попередні координати ми ділимо на масу частинки. Після виклику функції emit_particles слідує виклик функції apply_particle_physics в якій відбувається обрахунок прискорення за другим законом Ньютона та нових координат. Далі йде виклик функції delete_particles, що видаляє точки, які вилетіли за межі екрану.

На наступному етапі, після функції update_particles слідує блок коду відповідальний за відображення. В ньому, між викликами функцій BeginDrawing та EndDrawing ми очищаємо екран, викликаємо функцію draw_particles та виводимо FPS.

На рис. 1 видно як частинки підлітають вгору, сповільнюються, а потім падають вниз, а відповідно легші частинки підлітають вище, згідно поставленої мети.

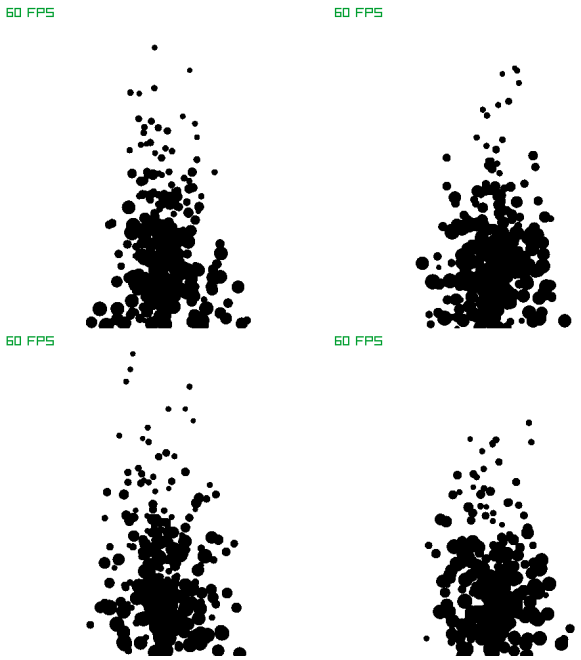

Рис. 1. Графічна інтерпретація моделі «фонтан»

Розглянемо структурні особливості при розробці програми яка реалізує гравітаційну взаємодію частинок. Загальна архітектура майже не зазнає змін відносно

попередньої програми, але є великі відмінності в деяких деталях. В структурі Particle з'явилось поле прискорення. Головною відмінністю є функція apply_particle_physics, яка тепер обраховує іншу, складнішу взаємодію. Дана функія спочатку обнуляє значення прискорення для кожної частинки, після чого перебирає всі можливі пари, обраховує прискорення, яке надає сила гравітації між двома частинками та акумулює його у відповідному полі. Лише після підрахунку всіх прискорень, переходимо до підрахунку нових координат. Це важливо для синхронності, щоб в один момент часу частинка знаходилась лише в одному місці.

Також слід зазначити, що дана задача вимагає більшої точності обчислень, для чого була створена змінна steps яка визначає за скільки кроків буде обчислено нові координати в одному кадрі. Тобто замість одного кроку з $\Delta t = 1$ виконується steps кроків з $\Delta t = 1/steps$.

Зазначимо, що при моделюванні фонтану використовувалась бібліотечна структура Vector2, яка має поля типу float, точності яких не вистачає в даній задачі. Тому було написано маленьке розширення для бібліотеки raylib, в якому було додано структуру Vector2D з полями типу double та переписані базові операції над векторами для неї.

На рис. 2 можна побачити, що й справді, частинки з більшою масою (вони мають і більший радіус) рухаються з меншою швидкістю та амплітудою по відношенню до менших частинок.

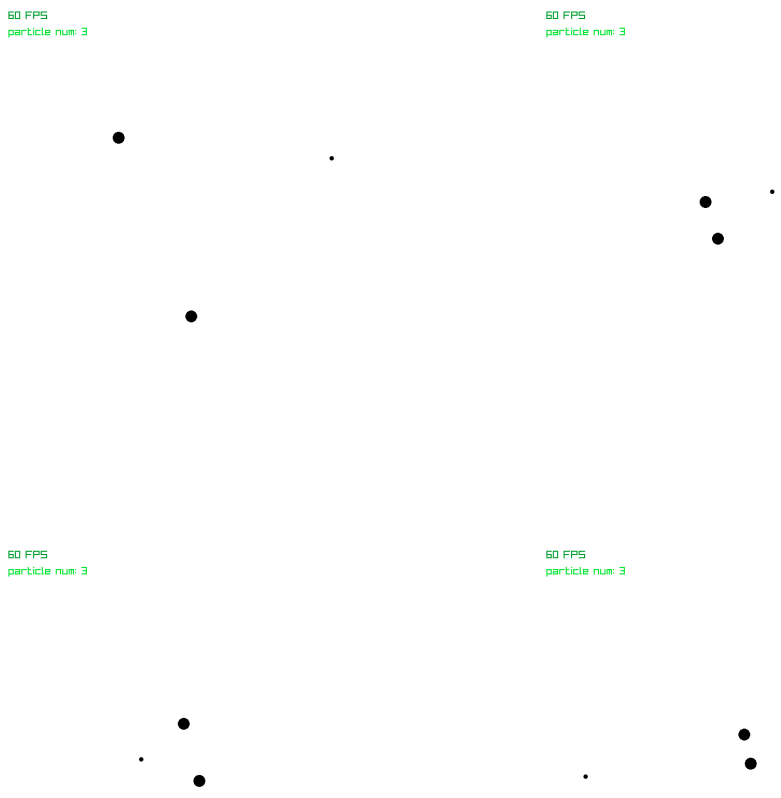

Рис. 2. Візуалізація руху частинок під дією сили тяжіння.

Додамо до цієї програми колізію між частинками. для цього була створена функція compute_collision, яка розводить кожну пару частинок на допустиму дистанцію

пропорційно до мас цих частинок. Це дозволяє більш реалістично імітувати передачу імпульсу.

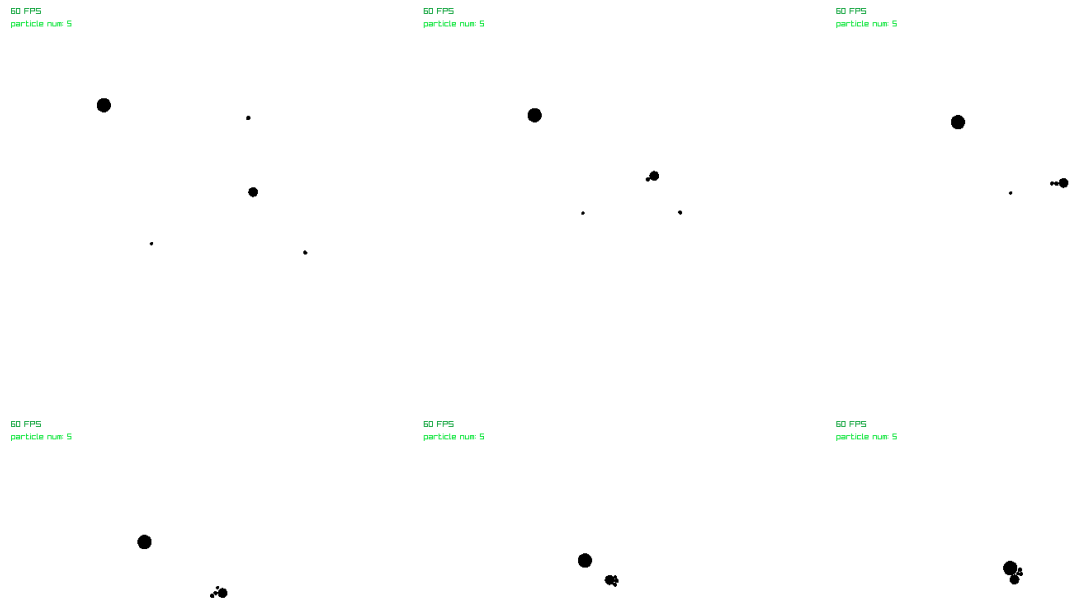

Рис. 3. Взаємодія частинок з колізіями

Дотримання колізій відбувається із заданою точністю, а не абсолютно, це зроблено для суттєвого зменшення кількості обрахунків. Також для обмеження кількості обрахунків та уникнення несправного функціонування програми кількість ітерацій при обрахунку колізій обмежена. Тому що, розрахунок взаємодії частинок з колізіями вимагає більшої точності ніж без колізій. При недостатній точності частинки починають набирати швидкість, обертатись і за рахунок цього розлітатись в різні боки, що менш схоже на взаємодію між частинками в природі.

Таким чином, можна виокремити загальні переваги у використанні даного методу для задач моделювання об’єктів динамічної природи, а саме: гнучкість і масштабованість (можна налаштувати параметри окремих часток, що досягають ефекту різної складності – від невеликих об’єктів до масштабних симуляцій); простота у реалізації; реалістичність (через симуляцію великої кількості елементів можна імітувати складну поведінку об'єктів природи, створюючи реалістичні анімації).

Висновки

Системи частинок – це потужний і ефективний інструмент для моделювання об’єктів динамічної природи, що дозволяє імітувати складні процеси за допомогою великої кількості простих елементів. Завдяки своїй гнучкості, масштабованості та всій простоті реалізації, такі системи знаходять застосування в численних галузях, від кінематографа до науки та інженерії. Використання описаних систем дозволяє досягти високого рівня реалістичності та надати нові можливості для візуалізації складних явищ.

Беручи до уваги, вище сказане, в даній статті було досліджено метод Стьормера-Верле, розроблені та реалізовані програми, які за допомогою цього методу моделюють об’єкти динамічної природи. Загалом метод Стьормера-Верле добре показав себе в поставлених задачах за рахунок гарної швидкодії, достатньої точності та легкості

модифікування, підтвердив свою репутацію ефективного алгоритму тому може бути рекомендованим для вирішення подібних задач.

Прогнозується, що подальші дослідження дозволять розробити більш реалістичні моделі динамічних процесів з підвищеною точністю, а також забезпечити нові можливості для інтерактивної роботи з такими симуляціями в реальному часі із залученням алгоритмів штучного інтелекту.

Список використаної літератури:

1. Swope W. C., Andersen H. C., Berens P. H., Wilson K. R. A computer simulation method for the calculation of equilibrium constants for the association of simple models of biological molecules. The Journal of Chemical Physics. 1982. Vol. 76, № 1. P. 637–649. DOI: 10.1063/1.442716
2. Erleben K., Sporring J., Henriksen K., Dohlmann H. Physics-Based Animation. Hingham: Charles River Media, 2005. 576 p.
3. Jakobsen T. Advanced character physics. Proceedings of the Game Developers Conference. San Jose, 2001. P. 383–401.
4. Jiang X., Ren H., He X. Research on anchor chain visualization for a ship anchoring simulation training system. PLoS ONE. 2020. Vol. 15, № 10. e0237563. DOI: 10.1371/journal.pone.0237563
5. Neumaier A. Mathematical Model Building. Modeling Languages in Mathematical Optimization / ed. J. Kallrath. Applied Optimization, Vol. 88. Boston: Kluwer, 2004. P. 37–43. URL: https://www.mat.univie.ac.at/~neum/model.html

References:

1. Swope, W. C., Andersen, H. C., Berens, P. H., & Wilson, K. R. (1982). A computer simulation method for the calculation of equilibrium constants for the association of simple models of biological molecules. The Journal of Chemical Physics, 76(1), 637–649. https://doi.org/10.1063/1.442716
2. Erleben, K., Sporring, J., Henriksen, K., & Dohlmann, H. (2005). Physics-Based Animation. Charles River Media.
3. Jakobsen, T. (2001). Advanced character physics. Proceedings of the Game Developers Conference, 383–401.
4. Jiang, X., Ren, H., & He, X. (2020). Research on anchor chain visualization for a ship anchoring simulation training system. PLoS ONE, 15(10), e0237563. https://doi.org/10.1371/journal.pone.0237563
5. Neumaier, A. (2004). Mathematical model building. In J. Kallrath (Ed.), Modeling Languages in Mathematical Optimization (Applied Optimization, Vol. 88, pp. 37–43). Kluwer. https://www.mat.univie.ac.at/~neum/model.html

DZYUBA Viktoriya,
Candidate of Technical Sciences, Lecturer, The Bohdan Khmelnytsky National University of Cherkasy
CHALYI Anton,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

APPLICATION OF PARTICLE SYSTEMS FOR MODELING OBJECTS OF A DYNAMIC NATURE

Summary. Introduction. *The modeling of dynamically behaving objects represents one of the most challenging problems in computer graphics, physics, and engineering. Dynamic systems encompass a wide range of phenomena, including fluid and gas flows, combustion processes, explosions, smoke, rain, and snow. Simulating such objects in real time, while preserving their physical behavior and visual plausibility, is a key aspect of modern animation, simulation, and game mechanics. Particle systems have established themselves as a highly effective approach to creating realistic models of such objects, enabling a level of detail that is difficult to achieve through conventional modeling techniques. The relevance of this research is reinforced by the growing demand for realistic visualization of natural phenomena in cinematography, computer games, virtual*

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.

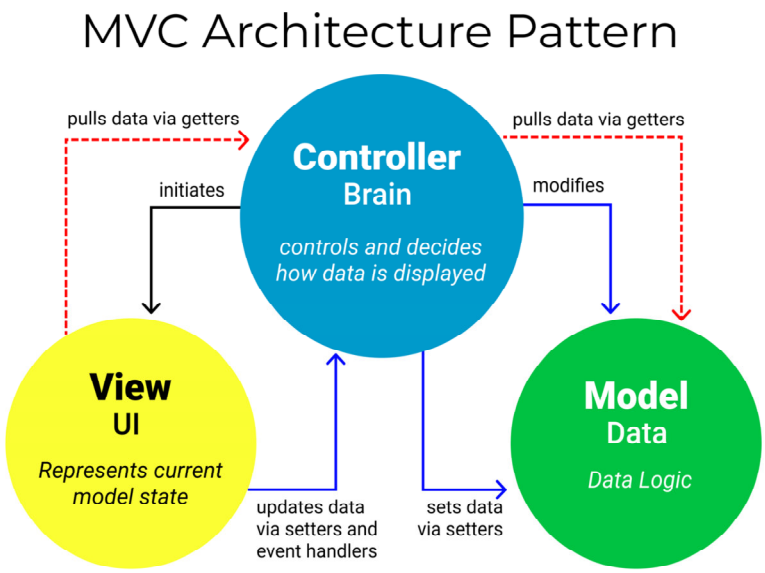

Рис. 1. Архітектурний шаблон MVC

– 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.

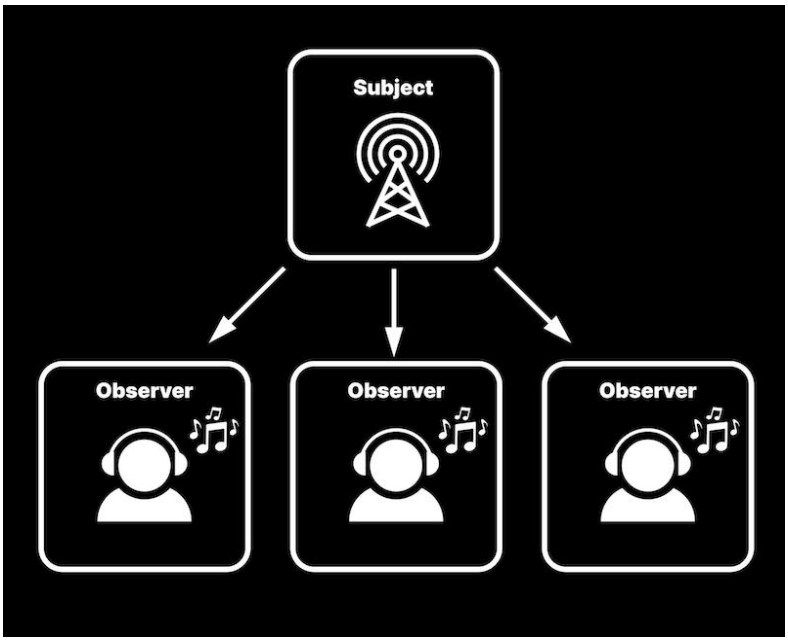

Рис. 2. Патерн Observer

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.

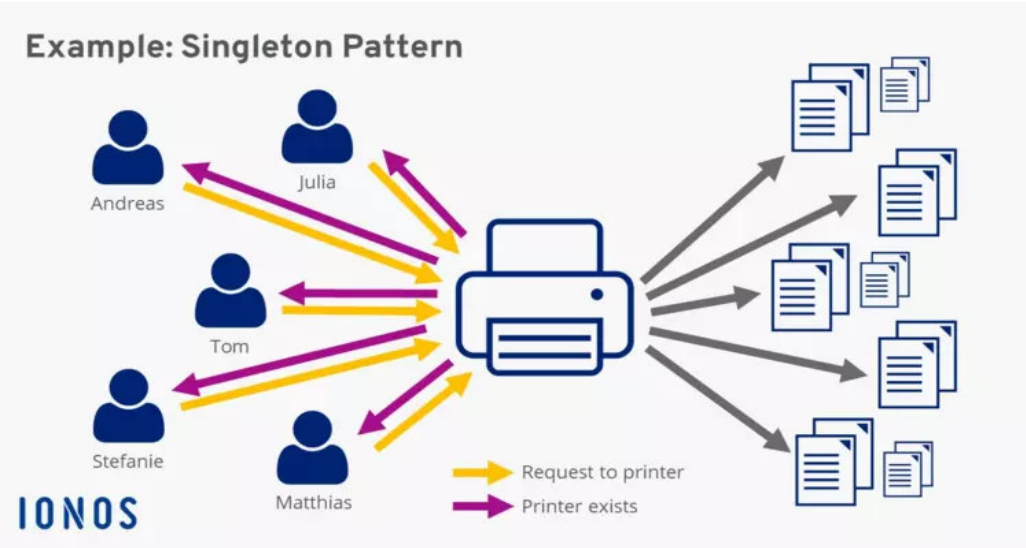

Рис. 3. Патерн Singleton

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.

УДК 004.35:007.52

DOI 10.31651/2076-5886-2023-1-27-40

PACS 07.05.Tp, 07.07.Df

ДОМІНІЧЕНКО Віталій Станіславович
студент спеціальності «Інформаційні система та технології» Черкаського національного університету імені Богдана Хмельницького
e-mail: dominichenko_v_s@vu.cdu.edu.ua

ПІСКУН Олександр Варфоломійович
кандидат технічних наук, доцент, завідувач кафедри прикладної математики та інформатики, Черкаський національний університет ім. Б. Хмельницького
e-mail: piskun@ukr.net
ORCID 0000-0001-5334-6337

ГЛАДКА Людмила Іванівна
к.ф.-м.н., доцент кафедри автоматизації та комп'ютерно-інтегрованих технологій, Черкаський національний університет імені Б. Хмельницького
e-mail: l_i_gladka@vu.cdu.edu.ua
ORCID 0000-0002-7030-9666

РОЗРОБКА АПАРАТНО-ПРОГРАМНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЧНОЇ ГРИ НА «СПІВАЮЧІЙ» ЧАШІ З ДИСТАНЦІЙНИМ КЕРУВАННЯМ ЧЕРЕЗ WI-FI

У статті описано розробку апаратно-програмної системи автоматизованої гри на «співаючій» чаші з дистанційним керуванням через Wi-Fi. Сучасний темп життя призводить до зростання рівня стресу серед людей, що обумовлює актуальність засобів для релаксації, зокрема «співаючих» чаш. Проте традиційна гра на такій чаші потребує постійної присутності людини. Розроблена система усуває цей недолік, автоматизуючи процес гри з можливістю дистанційного керування через веб-інтерфейс. Апаратну основу становить плата розробки NodeMCU ESP8266 з вбудованим Wi-Fi-модулем та електромагніт, керований через MOSFET-транзистор. Реалізовано три режими роботи: «Manual» (поодинокий удар), «UnpredictaBell» (псевдовипадкові удари у заданих діапазонах) та «DreamDive» (поступове затухання з подальшим наростанням для медитативного пробудження). Веб-інтерфейс розроблено як односторінковий застосунок на базі бібліотеки Preact.js та забезпечує зручне керування з будь-якого пристрою, оснащеного браузером. Пристрій підтримує два режими Wi-Fi: точка доступу та клієнт локальної мережі. Взаємодія між інтерфейсом і пристроєм здійснюється через REST API на базі асинхронного HTTP-сервера ESPAsyncWebServer. У статті описано архітектуру системи, обґрунтування вибору апаратних компонентів, алгоритми трьох режимів керування та процес розробки веб-інтерфейсу. Результати тестування підтверджують коректність роботи системи та відповідність усім поставленим вимогам.

Ключові слова: *«співаюча» чаша, Інтернет речей, NodeMCU ESP8266, веб-інтерфейс, вбудовані системи, RTOS, релаксація.*

Вступ

Сучасний ритм життя, насичений постійними стресовими навантаженнями, спонукає мільйони людей до пошуку ефективних засобів психологічного

розвантаження. Згідно зі статистикою Statista [1], хронічний стрес є одним із провідних чинників, що негативно впливають на продуктивність праці, якість міжособистісних стосунків та стан здоров'я загалом. Як наслідок, зростає популярність практик, заснованих на медитації та звукотерапії.

Серед інструментів звукотерапії особливе місце посідає «співаюча» чаша – стародавній музичний інструмент, поширений у різноманітних культурах Сходу. Як показано в дослідженні Goldsby et al. [2], регулярне прослуховування звуків «співаючої» чаші позитивно впливає на настрій, знижує рівень напруженості та покращує суб'єктивне відчуття добробуту. Коливання чаші породжують звук із багатим спектром обертонів у широкому діапазоні частот, що відрізняє її від більшості сучасних електронних засобів релаксації. Проте практичне застосування чаші потребує постійної уваги та фізичної присутності людини, що суттєво обмежує сценарії її використання – зокрема, унеможливлює застосування чаші під час засинання чи медитації без партнера.

Завдання автоматизації процесу гри на чаші з наданням можливості дистанційного керування через бездротову мережу є актуальним і вирішується у цій роботі. Запропонована апаратно-програмна система поєднує можливості платформи Arduino, Wi-Fi-модуля ESP8266, електромагнітного приводу та динамічного веб-інтерфейсу, реалізуючи тим самим повноцінний IoT-пристрій побутового призначення.

Метою роботи є розробка і практична реалізація апаратно-програмної системи автоматизованої гри на «співаючій» чаші, що забезпечує дистанційне керування режимами гри через Wi-Fi-мережу з використанням веб-інтерфейсу, а також дослідження ефективності обраних апаратних та програмних рішень і підтвердження їх достатності для задоволення визначених вимог.

Виклад основного матеріалу

1. Огляд предметної галузі та обґрунтування підходу

1.1. «Співаюча» чаша як акустичний об'єкт

«Співаюча» чаша є різновидом стоячого дзвону і в тих чи інших формах існує в багатьох культурах світу (рис. 1). Найпоширеніші чаші виготовляються зі сплавів металів методом кування або лиття. Звучання чаші утворюється або шляхом удару по її стінці медіатором, або рівномірними обертальними рухами медіатора по зовнішньому краю – метод, що викликає явище резонансу («заводить» чашу). У цій роботі реалізовано метод удару, оскільки він технічно придатний для автоматизації за допомогою електромагнітного привода з прийнятною складністю механізму.

Акустичний сигнал чаші характеризується широким спектром обертонів у діапазоні приблизно від 100 Гц до 8 кГц (рис. 2). Після удару звук поступово затухає впродовж десятків секунд. Важливою особливістю є те, що сила удару безпосередньо впливає на амплітуду коливань і, відповідно, на гучність та тривалість звучання, проте не змінює суттєво спектральний склад сигналу. Ця лінійна залежність між механічним впливом і акустичним результатом є фундаментальною для алгоритмів керування, реалізованих у роботі: управляючи лише силою удару та інтервалом між ударами, можна створювати різноманітні акустичні ефекти.

1.2. Підхід на основі Інтернету речей

Системи дистанційного керування фізичними пристроями є усталеним напрямком у галузі Інтернету речей (IoT). Концепція IoT, детально описана в роботах Gubbi et al. [3] та Atzori et al. [9], передбачає інтеграцію фізичних пристроїв до мережі з метою обміну даними та дистанційного управління ними. Atzori et al. визначають ключові компоненти IoT-системи: сенсори та виконавчі механізми, мережевий рівень передачі

даних і рівень застосунків. У розробленій системі ці компоненти представлені відповідно: електромагнітним приводом (виконавчий механізм), Wi-Fi-модулем ESP8266 (мережевий рівень) і веб-інтерфейсом на базі REST API (рівень застосунків).

Рис. 1. Різноманіття форм «співаючих» чаш

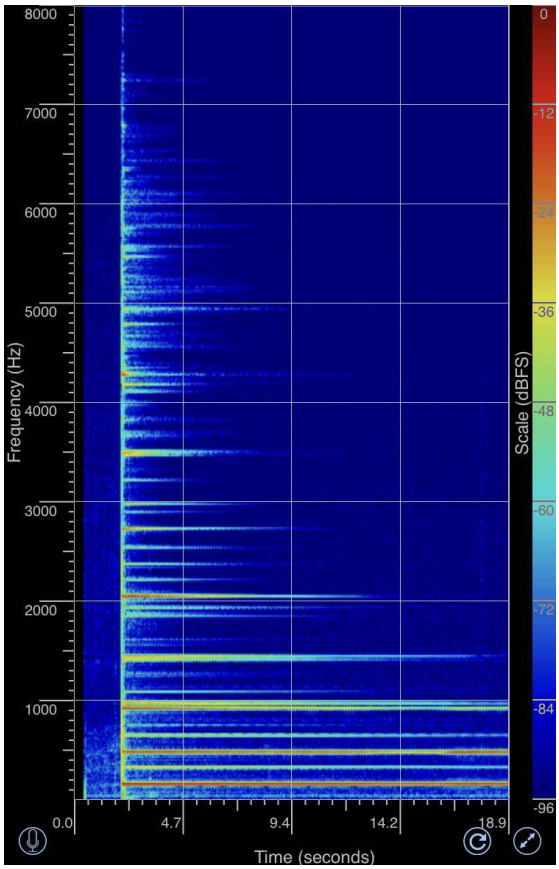

Рис. 2. Спектрограма звуку «співаючої» чаші, що демонструє широкий спектр обертонів

Для організації взаємодії між клієнтом і вбудованим сервером обрано архітектурний стиль REST (Representational State Transfer), описаний у докторській дисертації Філдінга [4]. Основними принципами REST є: клієнт-серверна архітектура, відсутність стану сесії на сервері (stateless), підтримка кешування, уніформний інтерфейс та багатошарова система. У контексті мікроконтролерної системи з обмеженими ресурсами ці принципи є особливо цінними: відсутність сесійного стану суттєво знижує споживання оперативної пам'яті, а уніформний інтерфейс спрощує реалізацію клієнта.

1.3. Вибір апаратної платформи

Платформа Arduino обрана як базова завдяки розвиненій екосистемі бібліотек, широкій документованості та великій спільноті розробників. Порівняно з іншими мікроконтролерними платформами, Arduino-сумісні плати мають нижчий поріг входження для швидкого прототипування без жертвування функціональністю. Для проектів класу IoT, де потрібна інтеграція Wi-Fi, природним вибором є сімейство ESP8266/ESP32 завдяки вбудованому бездротовому інтерфейсу та підтримці Arduino IDE через сторонній пакет плат.

Протягом розробки системи змінилося два апаратних покоління: від конфігурації Arduino Mega 2560 + ESP8266 (де обидва модулі є фізично окремими і взаємодіють через UART) до плати NodeMCU ESP8266, де мікроконтролер і Wi-Fi-чип є єдиним пристроєм із нативною підтримкою мережевого стека та файлової системи. Детальне порівняння платформ наведено в наступному розділі.

2. Постановка задачі

Необхідно розробити апаратно-програмну систему, яка відповідає таким функціональним вимогам:

1. Автоматизоване керування електромагнітним ударним механізмом для гри на «співаючій» чаші у щонайменше трьох режимах із різними характеристиками сили та частоти ударів.
2. Дистанційне керування пристроєм через динамічний веб-інтерфейс за протоколом HTTP без перезавантаження сторінки при зміні параметрів.
3. Підтримка двох режимів Wi-Fi: точки доступу (AP) для первинного налаштування та клієнта зовнішньої мережі (STA) для інтеграції в домашню мережу.
4. Зберігання параметрів Wi-Fi-підключення в енергонезалежній пам'яті з автоматичним відновленням з'єднання після перезапуску.
5. Підтримка одночасного підключення кількох HTTP-клієнтів без порушення стабільності роботи виконавчого механізму.
6. Коректне адаптивне відображення веб-інтерфейсу на мобільних пристроях та настільних комп'ютерах.

Нефункціональними обмеженнями системи є апаратні ресурси плати NodeMCU ESP8266: 128 КБ оперативної пам'яті (RAM) та 4 МБ флеш-пам'яті [5], з яких частина відводиться під файлову систему SPIFFS для розміщення файлів веб-інтерфейсу. Крім того, відсутність на платі апаратного годинника реального часу (RTC) і неможливість синхронізації часу без підключення до Інтернет вносять додаткові обмеження на реалізацію будильникових функцій.

Вхідними параметрами системи є: режим роботи, значення сили та інтервалу ударів (цілочисельні в діапазоні 1–5), час завершення (мітка часу, передана клієнтом у вигляді дельти від millis()), дані підключення до Wi-Fi (SSID та пароль). Вихідним є

механічний удар по чаші через електромагнітний привод, тривалість активації якого визначається обраною силою у відповідності до наперед встановлених констант.

3. Апаратна складова системи

3.1. Порівняльний аналіз апаратних платформ і вибір NodeMCU ESP8266

На початковому етапі проекту використовувався модифікований контролер Arduino Mega 2560 з вбудованим Wi-Fi-модулем ESP8266 на одній платі. Обидва компоненти – процесор ATmega2560 і чіп ESP8266 – на такій платі можуть працювати як спільно (через UART-з'єднання), так і незалежно. Для спільного режиму ESP8266 прошивається прошивкою NonOS SDK, після чого ATmega2560 управляє ним за допомогою AT-команд.

Проте ця конфігурація виявила ряд принципових обмежень. По-перше, відсутність підтримки асинхронного HTTP-сервера призводила до взаємного блокування: під час обробки HTTP-запиту ATmega2560 не міг одночасно керувати GPIO для електромагніту. По-друге, весь HTML/CSS/JavaScript код доводилося зберігати безпосередньо у Flash-рядках у тексті програми через функцію F(""), що унеможливлювало використання сучасних JavaScript-фреймворків. По-третє, програмування плати вимагало перемикання DIP-перемикачів для зміни режиму роботи між ATmega та ESP8266.

Перехід на плату NodeMCU ESP8266 version 1.0 (рис. 3) усунув усі ці проблеми. Плата побудована на модулі ESP-12E, що містить мікросхему ESP8266 з 32-бітним RISC-мікропроцесором Tensilica Xtensa LX106. Процесор підтримує операційну систему реального часу (RTOS) і працює на тактовій частоті від 80 до 160 МГц [5]. Планувальник RTOS гарантує детерміновану обробку конкурентних задач: Wi-Fi-стека, HTTP-сервера та керування GPIO одночасно і без взаємного блокування. NodeMCU також має 128 КБ оперативної пам'яті та 4 МБ флеш-пам'яті, що є достатнім для зберігання файлів SPA-застосунку у файловій системі SPIFFS.

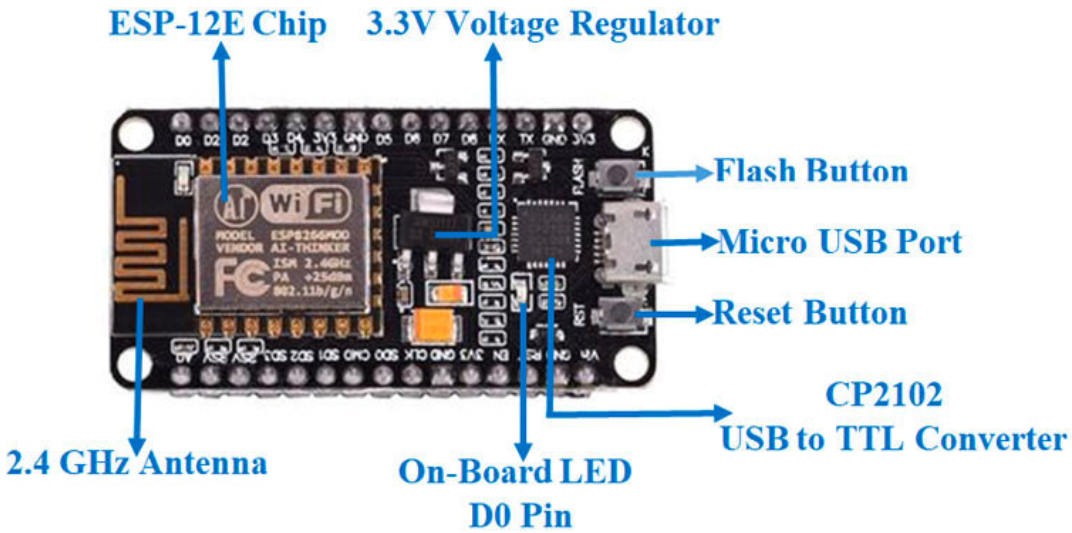

Рис. 3. Плата розробки NodeMCU version 1.0

Ключові переваги NodeMCU ESP8266 перед конфігурацією Arduino Mega 2560 + ESP8266 зведено в таблиці 1.

Плата повністю сумісна з Arduino IDE завдяки підтримці пакету ESP8266 Board Package. Для завантаження файлів веб-інтерфейсу у SPIFFS використовується стороннє розширення ESP8266 Filesystem Uploader для Arduino IDE.

Таблиця 1

Порівняння апаратних платформ

Характеристика	***Arduino Mega 2560 + ESP8266***	***NodeMCU ESP8266***
Файлова система	Відсутня (тільки Flash-рядки F(""))	SPIFFS (до 3 МБ для файлів застосунку)
Асинхронний HTTP-сервер	Не підтримується	Підтримується (ESPAsyncWebServer)
Кількість фізичних плат	2 + перемикач режимів	1
Тактова частота	16 МГц (ATmega2560)	80–160 МГц
Програмування	DIP-перемикачі + 2 окремих завантаження	Один кабель micro-USB
Wi-Fi-взаємодія	Через AT-команди по UART	Нативна підтримка в SDK
Оперативна пам'ять	8 КБ (ATmega2560)	128 КБ
Підтримка SPA-застосунків	Неможлива	Повноцінна (хостинг JS/HTML/CSS)
Конкурентна обробка задач	Відсутня (послідовне виконання)	RTOS-планувальник

3.2. Виконавчий механізм та схема керування

Механічним елементом, що здійснює удари по стінці чаші, є електромагніт із номінальною вантажопідйомністю 55 кг, робочою напругою 12 В та струмом споживання до 340 мА. Безпосереднє керування електромагнітом від GPIO-виходу NodeMCU неможливе з двох причин: максимальна вихідна напруга GPIO становить 3,3 В при максимальному струмі ~12 мА, тоді як електромагніт потребує 12 В / 340 мА; крім того, необхідна гальванічна ізоляція індуктивного навантаження від чутливої мікроконтролерної схеми для захисту від зворотних ЕРС.

Обидві проблеми вирішено застосуванням модуля комутації на базі польового MOSFET-транзистора F5305S (рис. 4) з вбудованою оптичною розв'язкою [6]. Допустима напруга керуючого сигналу: 3–20 В (сумісно з 3,3 В NodeMCU). Вихідні параметри: 5–36 В при тривалому струмі до 5 А (максимальний – 20 А). Сигнал рівня логічної «1» (3,3 В) від GPIO-піна D7 перемикає MOSFET, підключаючи обмотку електромагніту до джерела живлення 12 В.

Рис. 4. Модуль MOSFET-транзистора F5305S

Вся система живиться від одного джерела постійного струму напругою 12 В, підключеного до плати розширення NodeMCU. Плата розширення містить

понижуючий конвертер 12 В → 5 В / 800 мА, з якого через вбудований стабілізатор NodeMCU формується напруга 3,3 В для мікроконтролера. Таким чином, обидва компоненти – плата розробки та електромагніт – живляться від одного джерела, що виключає необхідність окремих блоків живлення у кінцевому пристрої.

3.3. Метод кодування сили удару

Оскільки плата NodeMCU ESP8266 не забезпечує аналогового керування силою вихідного струму GPIO (лише напругою), а типова реалізація ШІМ-регулювання струму в індуктивному навантаженні потребувала б додаткового LC-фільтра, у роботі застосовано альтернативний метод – часове кодування сили удару: сила удару визначається тривалістю утримання електромагніту у ввімкненому стані.

При короткому імпульсі (мала тривалість) сердечник електромагніта не встигає суттєво розігнатися до моменту механічного контакту з медіатором, забезпечуючи слабкий удар. При тривалому імпульсі (максимальне значення – до 350 мс) сердечник набирає значну швидкість і наносить сильний удар по чаші. Після завершення імпульсу пружний елемент конструкції повертає медіатор у вихідне положення, готуючи систему до наступного удару. Межі діапазону тривалостей встановлено експериментально: MIN_STRENGTH = 150 мс, MAX_STRENGTH = 350 мс. Такий підхід дозволяє обійти апаратне обмеження платформи без додаткових схемотехнічних рішень.

4. Алгоритм керування

4.1. Загальна структура програми

Загальна структура програми відповідає стандартній схемі Arduino-застосунку з функціями setup() та loop(), розширеною засобами асинхронного програмування RTOS. Функція setup() виконує одноразову ініціалізацію: налаштування GPIO-піна D7 як виходу, монтування файлової системи SPIFFS, зчитування конфігурації з JSON-файлу, спробу підключення до збереженої Wi-Fi-мережі та запуск асинхронного HTTP-сервера з реєстрацією обробників запитів. Функція loop() реалізує основний цикл: перевіряє поточний активний режим і виконує відповідні дії для керування електромагнітом – формує імпульси потрібної тривалості та витримує задані інтервали між ними.

Логіка ініціалізації Wi-Fi реалізована за принципом «спроба – відкат». Якщо у JSON-конфігурації збережено дані зовнішньої мережі (SSID та пароль), система намагається підключитися в режимі STA протягом фіксованого таймауту. У разі невдачі (мережа недоступна або змінила пароль) конфігурація очищується, і система переходить у режим точки доступу (AP) із фіксованим SSID «MAUMVI». Такий підхід забезпечує роботу пристрою в будь-яких умовах без необхідності ручного скидання налаштувань.

4.2. Режим «Manual»

Режим «Manual» призначений для поодиноких ударів на розсуд користувача та реалізує найпростішу форму взаємодії. Після отримання HTTP GET-запиту /api/hit?holdTime=<мс> сервер зчитує параметр holdTime – тривалість активації електромагніту в мілісекундах – і передає його в чергу задачі керування приводом. Задача встановлює GPIO D7 у стан HIGH на задану тривалість, після чого повертає його в LOW.

З боку веб-інтерфейсу сила удару задається тривалістю утримання кнопки «Hit the Bowl»: JavaScript-обробник фіксує момент натискання (mousedown / touchstart) і момент відпускання (mouseup / touchend), обчислює різницю в мілісекундах (обмежену значенням MAX_STRENGTH), після чого надсилає її як параметр holdTime. Таким

чином, між зусиллям утримання кнопки і силою механічного удару існує прямий, інтуїтивно зрозумілий зв'язок. Навколо кнопки у веб-інтерфейсі анімується коло наростаючої яскравості, що дає користувачу візуальний зворотний зв'язок про накопичену силу.

4.3. Режим «UnpredictaBell»

Режим «UnpredictaBell» реалізує автономну гру з псевдовипадковим характером ударів, що наближає автоматичну гру до живої. Монотонна регулярна послідовність ударів дратує слухача, тому введення контрольованої варіативності є принциповою вимогою до алгоритму.

Користувач задає два цілочисельних параметри в діапазоні від 1 до 5: центральне значення сили удару та центральне значення інтервалу між ударами. Ці параметри лінійно масштабуються до фізичних діапазонів: тривалість удару – від MIN_STRENGTH = 150 мс до MAX_STRENGTH = 350 мс; інтервал між ударами – від MIN_INTERVAL = 3000 мс до MAX_INTERVAL = 15000 мс. Після масштабування фактичні значення тривалості удару та інтервалу для кожного наступного удару генеруються за допомогою функції random() в певному симетричному околі центрального значення. Це запобігає монотонності звучання і водночас зберігає загальний характер гри, заданий користувачем.

4.4. Режим «DreamDive»

Режим «DreamDive» є найскладнішим алгоритмічно і орієнтований на сценарій поступового засинання під звуки чаші з наступним плавним пробудженням у заданий час. Алгоритм складається з трьох послідовних фаз:

Фаза 1 – затухання (Fade Out). Починаючи від поточних параметрів сили та інтервалу (успадкованих від режиму «UnpredictaBell» або заданих за замовчуванням), система поступово зменшує силу ударів і збільшує паузи між ними. Тривалість фази задається користувачем у хвилинах (параметр transitionTime). Зміна параметрів відбувається лінійно з кожним наступним ударом – від початкових значень до мінімально допустимих.

Фаза 2 – очікування (Awaiting). Система не виконує ударів. Тривалість цієї фази автоматично розраховується сервером як різниця між заданим часом завершення режиму та сумою тривалостей фаз 1 і 3, що дозволяє точно дотримати час завершення незалежно від тривалості переходів.

Фаза 3 – наростання (Fade In). Система виконує дзеркальний до Фази 1 процес: від мінімальних значень параметрів до початкових. Тривалість також дорівнює transitionTime.

Оскільки NodeMCU не має доступу до реального астрономічного часу без підключення до Інтернет, час завершення режиму передається клієнтом у вигляді дельт відносно поточної мілісекундної мітки пристрою (millis()). Клієнтський браузер конвертує бажаний час будильника в мілісекунди відносно Epoch, обчислює дельту від поточного millis() плати та передає окремі тривалості для кожної з трьох фаз. Таким чином, плата не потребує синхронізації часу, а браузер бере на себе роль перекладача між астрономічним часом і відносними мілісекундами пристрою.

Після завершення режиму передбачено опціональний автоматичний перехід до режиму «UnpredictaBell» (параметр toSwitch), що дозволяє продовжити акустичний супровід після пробудження без додаткових дій з боку користувача.

4.5. Збереження конфігурації у файловій системі

Зберігання параметрів Wi-Fi-підключення здійснюється через JSON-файл у

файловій системі SPIFFS флеш-пам'яті. Такий підхід обрано замість EEPROM з кількох міркувань: ресурс перезаписів флеш-пам'яті (понад 30 мільйонів циклів [8]) значно перевищує ресурс EEPROM (100 тисяч циклів); JSON-формат забезпечує зручну роботу зі структурованими даними без ручного управління адресами; файлова система дозволяє зберігати будь-яку кількість файлів конфігурації, не обмежуючись фіксованим обсягом EEPROM. Збереження відбувається лише за фактом зміни параметрів Wi-Fi-підключення, що мінімізує кількість операцій запису і подовжує ресурс пам'яті.

5. Веб-інтерфейс

5.1. Архітектура та технологічний стек

Веб-інтерфейс реалізовано як односторінковий застосунок (SPA – Single Page Application) на основі бібліотеки Preact.js [7]. Preact.js є легковаговим аналогом React.js з ідентичним API, проте значно меншим розміром бандлу (близько 3 КБ у стисненому вигляді порівняно з ~40 КБ у React), що є критично важливим в умовах обмеженої флеш-пам'яті NodeMCU. Компонентна модель Preact дозволяє декларативно описувати UI і автоматично оновлювати лише ті частини DOM, стан яких змінився, – без ручного маніпулювання HTML і без перезавантаження сторінки.

Скомпільовані статичні файли (HTML, CSS, JavaScript) зберігаються у файловій системі SPIFFS і роздаються клієнту асинхронним HTTP-сервером ESPAsyncWebServer як статичний контент. Завдяки кешуванню браузером статичних ресурсів після першого завантаження, подальша взаємодія з пристроєм генерує лише невеликі JSON-запити, що мінімізує навантаження на вбудований сервер.

Відмова від шаблонного серверного рендерингу (SSR) на користь SPA зумовлена двома принциповими чинниками. По-перше, SSR потребує генерації нової HTML-сторінки для кожного запиту, що навантажує процесор NodeMCU. По-друге, при SSR будь-яке оновлення стану (наприклад, зміна відсотка завершеності прогрес-бару DreamDive) вимагає повного перезавантаження сторінки, тоді як у SPA оновлюється лише відповідний компонент.

5.2. REST API системи

Комунікація між браузерним клієнтом і сервером здійснюється виключно через HTTP GET-запити до задокументованих ендпоінтів. Вибір методу GET (замість POST/PUT) обумовлений простотою реалізації на стороні ESPAsyncWebServer та мінімізацією обсягу коду парсингу на мікроконтролері. Параметри запитів передаються як рядок запиту (query string).

При отриманні запиту обробник зчитує параметри рядка запиту, вносить зміни до глобального стану системи та повертає HTTP 200 без тіла відповіді – за винятком /api/status, що повертає JSON-об'єкт із поточним режимом, значеннями параметрів та відсотком завершеності поточної фази. Клієнт опитує цей ендпоінт з фіксованим інтервалом (polling) для оновлення інтерфейсу.

5.3. Функціональні можливості та особливості UX

Фінальна версія веб-інтерфейсу (рис. 6) містить такі елементи управління:

- панель вкладок для перемикання між трьома режимами роботи;
- режим «Manual»: велика кругла кнопка «Hit the Bowl» із анімованим світловим індикатором наростання сили удару при утриманні;
- режим «UnpredictaBell»: два сегментних перемикачі (1–5) для сили та інтервалу, кнопки START/STOP;

– режим «DreamDive»: годинний вибірник часу завершення, параметр тривалості переходів у хвилинах, перемикач опції «Перейти до UnpredictaBell після завершення»; після запуску – три-сегментний прогрес-бар із назвою поточної фази та відсотком її виконання, кнопка STOP;
– модальне вікно Wi-Fi: поля для SSID та пароля, кнопка Connect.

Рис. 5. Приклади GET-запитів, що надсилаються веб-інтерфейсом

Рис. 6. Остаточний UI/UX дизайн веб-інтерфейсу

Усі зміни стану відображаються у реальному часі без перезавантаження сторінки. Інтерфейс спроектовано з урахуванням принципів адаптивного дизайну: усі елементи оптимізовані для взаємодії пальцем на сенсорному екрані мобільного пристрою, а ширина блоку обмежена з центруванням для коректного відображення на широких моніторах.

6. Результати тестування та аналіз

6.1. Функціональне тестування

У ході функціонального тестування перевірялися такі аспекти роботи системи:

Коректність режимів гри. Усі три режими перевірено на відповідність заданим параметрам. Режим «Manual» коректно передає тривалість утримання кнопки у вигляді параметра holdTime і відтворює удар відповідної сили. Режим «UnpredictaBell» формує псевдовипадкові послідовності в заданих діапазонах. Режим «DreamDive» коректно переходить між фазами та завершується в заданий час.

Перемикання режимів. Зміна режиму в процесі активної роботи зупиняє поточний цикл ударів і запускає новий без зависань і збоїв.

Wi-Fi-підключення. Система коректно запускається в режимі AP при першому включенні. Після введення даних зовнішньої мережі та перезапуску система підключається в режимі STA і обслуговує HTTP-запити за статичною IP-адресою.

Збереження конфігурації. Параметри Wi-Fi-підключення успішно зчитуються з JSON-файлу SPIFFS після кожного перезапуску плати.

Часова точність режиму «DreamDive». Перевірено відповідність часу завершення режиму заданому значенню будильника з урахуванням дельта-передачі від браузера. Відхилення склало ±1–2 секунди за 15-хвилинний цикл, що є прийнятним для даного застосування.

6.2. Аналіз проблеми асинхронності та її вирішення

Ключовою технічною проблемою прототипної версії системи (на Arduino Mega 2560 із синхронним сервером) була взаємна блокуванність HTTP-сервера та виконавчого механізму. Синхронна обробка HTTP-запиту займала весь процесорний час, під час якого PWM/GPIO-сигнал для електромагніта не оновлювався. Як вимушений захід у прототипі запроваджувалася примусова 2-секундна затримка між запитами («лоадер»), що компенсувала нестабільність за рахунок суттєвого зниження швидкодії інтерфейсу.

Перехід до NodeMCU ESP8266 з асинхронним сервером ESPAsyncWebServer усунув цю проблему на архітектурному рівні: планувальник RTOS перемикає контекст між задачами обслуговування Wi-Fi-стека, HTTP-сервера та керування GPIO, гарантуючи своєчасне виконання кожної без блокування інших. У фінальній версії затримка «лоадера» відсутня, а під час активної роботи виконавчого механізму веб-сервер продовжує відповідати на запити без затримки.

Зведені результати тестування наведено в таблиці 2.

Таблиця 2

Зведені результати функціонального тестування системи

Параметр тестування	**Очікуваний результат**	**Фактичний результат**
1	2	3
Запуск у режимі AP (без збереженої конфігурації)	SSID «MAUMVI» доступний	Виконано
Підключення до зовнішньої мережі (STA)	Статична IP доступна	Виконано
Відновлення підключення після перезапуску	Автоматичне відновлення	Виконано
Одночасні підключення ≥2 клієнтів	Без збоїв	Виконано

Продовження таблиці 2

1	2	3
Відповідь /api/hit під час активного удару	Без затримки	Виконано (асинхронний сервер)
Прогрес-бар «DreamDive» (точність часу)	±5 с за 15 хв	±1–2 с
Адаптивність: мобільний / ПК	Коректне відображення	Виконано
Збереження конфігурації SPIFFS	Дані між перезапусками	Виконано

6.3. Обговорення обмежень

Серед виявлених обмежень системи варто зазначити такі. По-перше, відсутність на платі NodeMCU апаратного аналогового управління виходом GPIO унеможливила реалізацію плавного ШІМ-регулювання сили удару, замість якого запроваджено метод часового кодування. Це спрощення виявилося практично достатнім, проте теоретично обмежує плавність регулювання. По-друге, NodeMCU не підтримує синхронізацію астрономічного часу без підключення до Інтернет, що потребує передавання часових параметрів від браузерного клієнта й унеможливлює роботу будильникового режиму без пристрою-клієнта в момент запуску. По-третє, розмір SPIFFS-розділу обмежує загальний обсяг файлів веб-інтерфейсу, що зумовлює вибір на користь легковагових бібліотек.

Висновки

У роботі розроблено апаратно-програмну систему для автоматизованої гри на «співаючій» чаші з дистанційним керуванням через Wi-Fi. Система включає:

- обґрунтований вибір апаратної бази (NodeMCU ESP8266) на основі порівняльного аналізу з конфігурацією Arduino Mega 2560 + ESP8266 за критеріями асинхронності, підтримки файлової системи, зручності програмування та обчислювальної потужності;
- реалізацію електромагнітного ударного механізму з керуванням через MOSFET-транзистор F5305S і методом часового кодування сили удару;
- три режими роботи – «Manual», «UnpredictaBell» та «DreamDive» – що охоплюють широкий спектр сценаріїв використання від поодинокого удару до медитативного будильника з нелінійним профілем наростання/затухання;
- REST API на базі асинхронного HTTP-сервера ESPAsyncWebServer для стабільної паралельної обробки HTTP-запитів і керування виконавчим механізмом;
- динамічний SPA-веб-інтерфейс на Preact.js із підтримкою адаптивного відображення та одночасного підключення кількох клієнтів.

Наукова новизна роботи полягає в такому: (1) запропоновано та обґрунтовано метод часового кодування сили електромагнітного удару як альтернативу ШІМ-регулюванню у разі апаратної неможливості аналогового управління виходом GPIO мікроконтролера; (2) розроблено алгоритм «DreamDive» з лінійним профілем наростання/затухання параметрів гри, прив'язаним до астрономічного часу через дельта-передачу від браузерного клієнта; (3) обґрунтовано і реалізовано архітектуру IoT-системи акустичного пристрою побутового призначення на базі RTOS-мікроконтролера з асинхронним HTTP-сервером, що розв'язує проблему конкуренції між задачами мережевого стека і керування виконавчим механізмом.

Перспективами подальшого розвитку є: інтеграція мікрофонного датчика для адаптивного регулювання сили удару за акустичним зворотним зв'язком; розширення алгоритмів генерації ритмічних патернів із застосуванням методів машинного навчання; реалізація WebSocket-з'єднання замість поточного polling-підходу для оновлення стану інтерфейсу в реальному часі; перехід на платформу ESP32 для збільшення обчислювальних ресурсів і розширення функціональності.

Список використаної літератури

1. Statista. Stress and burnout – Statistics & Facts [Електронний ресурс]. – 2023. – Режим доступу: https://www.statista.com/topics/2099/stress-and-burnout/
2. Goldsby T.L., Goldsby M.E., McWalters M., Mills P.J. Effects of Singing Bowl Sound Meditation on Mood, Tension, and Well-Being: An Observational Study // Journal of Evidence-Based Complementary & Alternative Medicine. – 2017. – Vol. 22(3). – P. 401-406.
3. Gubbi J., Buyya R., Marusic S., Palaniswami M. Internet of Things (IoT): A vision, architectural elements, and future directions // Future Generation Computer Systems. – 2013. – Vol. 29(7). – P. 1645-1660.
4. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures : doctoral dissertation. – University of California, Irvine, 2000. – 162 p.
5. NodeMCU ESP8266 [Електронний ресурс]. – components101.com. – 2020. – Режим доступу: https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet
6. IRF5305S Datasheet – International Rectifier [Електронний ресурс]. – 1999. – Режим доступу: https://html.alldatasheet.com/html-pdf/68168/IRF/IRF5305S/47/1/IRF5305S.html
7. Preact – Getting Started [Електронний ресурс]. – preactjs.com. – 2023. – Режим доступу: https://preactjs.com/guide/v10/getting-started/
8. ESP8266 Arduino Core documentation – Filesystem [Електронний ресурс]. – 2017. – Режим доступу: https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html
9. Atzori L., Iera A., Morabito G. The Internet of Things: A survey // Computer Networks. – 2010. – Vol. 54(15). – P. 2787-2805.

References

1. Statista. (2023). Stress and burnout – Statistics & Facts. https://www.statista.com/topics/2099/stress-and-burnout/
2. Goldsby T.L., Goldsby M.E., McWalters M., Mills P.J. (2017). Effects of Singing Bowl Sound Meditation on Mood, Tension, and Well-Being: An Observational Study. Journal of Evidence-Based Complementary & Alternative Medicine, 22(3), 401-406.
3. Gubbi J., Buyya R., Marusic S., Palaniswami M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.
4. Fielding R.T. (2000). Architectural Styles and the Design of Network-based Software Architectures [Doctoral dissertation]. University of California, Irvine.
5. NodeMCU ESP8266. (2020). components101.com. https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet
6. IRF5305S Datasheet. (1999). International Rectifier. https://html.alldatasheet.com/html-pdf/68168/IRF/IRF5305S/47/1/IRF5305S.html
7. Preact – Getting Started. (2023). preactjs.com. https://preactjs.com/guide/v10/getting-started/
8. ESP8266 Arduino Core documentation – Filesystem. (2017). https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html
9. Atzori L., Iera A., Morabito G. (2010). The Internet of Things: A survey. Computer Networks, 54(15), 2787-2805.

DOMINICHENKO Vitaliy,
Student, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
PISKUN Oleksandr,
Candidate of Technical Sciences, Associate Professor, Head of Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
HLADKA Liudmyla,
PhD in Physical and Mathematical Sciences, Associate Professor of the Department of Automation and Computer-Integrated Technologies, The Bohdan Khmelnytsky National

University of Cherkasy, Ukraine

DEVELOPMENT OF A HARDWARE-SOFTWARE SYSTEM FOR AUTOMATED PLAYING OF A SINGING BOWL WITH REMOTE WI-FI CONTROL

Summary. ***Introduction.*** *Modern living conditions lead to increased chronic stress levels among a significant portion of the population. As shown by Goldsby et al. [2], systematic exposure to singing bowl sounds positively affects mood, reduces tension and improves subjective well-being. However, traditional playing of a singing bowl requires constant human presence, which limits its use in unattended scenarios such as falling asleep or solo meditation. The development of an automated system capable of playing the bowl independently while allowing remote control is therefore a relevant engineering task at the intersection of IoT systems and wellness technology.*

Purpose. *The purpose of this work is to develop and practically implement a hardware-software system for automated playing of a singing bowl with remote Wi-Fi control, and to investigate the effectiveness of the selected hardware and software solutions.*

Results. *The system is built on the NodeMCU ESP8266 development board, selected through comparative analysis against the Arduino Mega 2560 + ESP8266 configuration. NodeMCU's native RTOS enables concurrent asynchronous HTTP server operation and GPIO actuator control; its built-in SPIFFS file system hosts SPA web interface assets; and its single-board architecture simplifies development and deployment. The electromagnetic actuator, switched through an F5305S MOSFET module with optical isolation, implements strike strength control through temporal encoding: activation pulse duration (150–350 ms) determines strike strength, compensating for the inability to perform analog current control via NodeMCU GPIO. Three operating modes are implemented: Manual (single strike on demand, strength proportional to button hold duration), UnpredictaBell (pseudo-random autonomous playing within user-defined strength and interval ranges), and DreamDive (gradual fade-out followed by fade-in timed to a user-specified alarm, suitable for sleep and wake scenarios). Since NodeMCU has no access to astronomical time without Internet connectivity, the DreamDive mode receives timing parameters as millisecond deltas calculated by the browser client from Unix Epoch. The web interface is a Single Page Application built with Preact.js (a 3 KB lightweight React alternative) and communicates with the device via a REST API on ESPAsyncWebServer. Configuration persistence uses JSON files in SPIFFS flash memory, providing over 30 million write cycles. Functional testing confirmed correct operation across all modes, stable Wi-Fi in both AP and STA modes, concurrent multi-client support without instability, and DreamDive timing accuracy of ±1–2 seconds over a 15-minute period.*

Conclusion. *A fully functional hardware-software system for automated singing bowl playing with remote Wi-Fi control has been developed and validated. The scientific contributions are: (1) a temporal encoding method for electromagnetic actuator force as an alternative to PWM under GPIO analog output constraints; (2) the DreamDive algorithm with a linear fade profile tied to astronomical time via browser-side delta computation; (3) an RTOS-based IoT architecture resolving the concurrency conflict between network stack and actuator control tasks. Future work includes acoustic feedback via a microphone sensor, machine-learning-based rhythm pattern generation, WebSocket-based real-time state synchronization, and migration to the ESP32 platform for greater resources.*

Keywords: *singing bowl, Internet of Things, NodeMCU ESP8266, web interface, electromagnetic actuator, MOSFET, RTOS, embedded systems, stress reduction.*

Одержано редакцією 17.11.2023 р.
Прийнято до публікації 06.12.2023 р.

УДК 378:517

DOI 10.31651/2076-5886-2023-1-41-48

PACS

БОСОВСЬКИЙ Микола Васильович,
кандидат педагогічних наук, доцент,
доцент кафедри математики та методики навчання математики,
Черкаський національний університет імені Богдана Хмельницького
e-mail: bosovskyy@gmail.com
ORCID: 0000-0003-1187-5550

СЕРДЮК Зоя Олексіївна,
кандидат педагогічних наук, доцент,
завідувач кафедри математики та методики навчання математики,
Черкаський національний університет імені Богдана Хмельницького
e-mail: serdyuk_z@ukr.net
ORCID: 0000-0002-9376-4346

АСИМПТОТИЧНА ОЦІНКА ДЕЯКИХ РЕКУРЕНТНИХ ПОСЛІДОВНОСТЕЙ

У статті розглянуто деякі властивості рекурентних послідовностей та уточнено результати для одного класу нескінченно малих послідовностей. Матеріали можуть бути використані в курсах «Додаткові розділи математичного аналізу», «Чисельні методи», «Алгоритмізація та програмування» тощо для підготовки студентів спеціальностей різних математичних та технічних спеціальностей.

Ключові слова: *рекурентні послідовності, нескінченно мала послідовність, асимптотичний метод, теорема.*

Постановка проблеми. Підготовка сучасних фахівців найчастіше вимагає симбіозу знань з різних галузей. Нинішній інженер, програміст, математик має добре поєднувати знання з математики та програмування. Тобто, грунтовні знання з математичних та ІТ дисциплін, а також вдале їх застосування, є гарантією формування професійних навичок у таких спеціалістів. Зокрема, рекурентні співвідношення мають надзвичайно важливе значення для програмування. Вони використовуються у аналізі алгоритмів, наближених обчисленнях, динамічному програмуванні, тощо. Під час розгляду рекурентних послідовностей однією з головних задач є її розв'язання, тобто необхідно виразити x_n через n. Ця задача не завжди розв'язується. Тому постає питання уточнення n для знаходження x_n в залежності від того, як обрано перший член x_0.

Аналіз останніх досліджень та публікацій. Різні аспекти використання рекурентних послідовностей досліджено у роботах А.П. Креневича [1], Т.О.Коротєєвої [2], В.М. Пивоварчик, О.М. Яковлєва, О.М. Болдирєва [3] та інші. Питанням підвищення ефективності викладання курсу вищої математики та математичного аналізу й удосконаленням методики його навчання студентів ЗВО різних технічних спеціальностей присвячені роботи М. Бородіна, Л. Васяк, К. Власенко, М. Жалдак [4], Т. Крилової, О. Кондратьєвої, В. Клочка [5], І. Лов'янової, О. Потапової, С. Федорової, С. Федосєєва та ін.

Мета статті – розглянути деякі властивості рекурентних послідовностей та уточнити результати для одного класу нескінченно малих послідовностей.

Виклад основного матеріалу. Часто трапляється, що під час розв'язання тієї чи

іншої задачі, потрібно обчислити визначену яким-небудь чином величину, а це обчислення призводить до надмірно великої кількості операцій, так що їх виконання стає досить складним або ж практично неможливим. У таких випадках справжньою знахідкою може виявитися який-небудь інший метод отримання додаткової інформації про цю величину, який дозволяє знайти її значення хоча б наближено. Такий метод (як зазначив Лаплас) «тим більш точний, чим більш потрібний»; ми отримуємо тим більш точне наближення до шуканої величини, чим більше дій потрібно для її прямого обчислення.

У даному випадку ми говоримо про асимптотичну оцінку або асимптотичну формулу.

Метою асимптотичних методів є отримання *O*-оцінок і *o*-оцінок у випадках, коли скористатися визначенням функції для дуже великих (або дуже малих) значень аргументу досить важко. Трапляється навіть так, що визначення функції настільки важке, що вже для звичайних значень змінної легше отримати асимптотичну інформацію, ніж будь-яку іншу.

Ні *O*-оцінка, ні *o*-оцінка в їх звичайному вигляді безпосередньо не застосовні для обчислювальних цілей. Однак, майже у всіх випадках, де є такі оцінки, можна, переглянувши доведення, замінити *O*-оцінки нерівностями, які містять числові сталі.

Для цього на кожному етапі наших дій ми повинні вказувати визначені числа або функції з визначеними властивостями там, де при отриманні асимптотичних оцінок ми обмежились доведенням існування таких чисел або функцій.

У даній роботі уточнюється результат для одного класу рекурентних послідовностей, які отримані раніше в статті С.С. Линчук «Про швидкість збіжності рекурентних послідовностей», [6, с. 72-79], в якій розглядається один клас нескінченно малих рекурентних послідовностей.

Згідно з цієї роботою, *n*-й член послідовності виражається через *n* і φ(x). Функція φ(x) отримується з функціонального рівняння $\varphi(f(x)) = \varphi(x) - 11$, $x \in (0, x_0)$.

Асимптотика загального члену одного класу нескінченно малих рекурентних послідовностей. У роботі [6] для одного класу нескінченно малих рекурентних послідовностей в досить загальній ситуації вивчено асимптотичну поведінку їх загального члену. При цьому отримано узагальнення відомих результатів в цьому напрямку.

Скрізь далі асимптотичні рівності для послідовностей розглядаються при $n \to \infty$.

Теорема 1. Нехай функція $f(x)$ неперервна на деякому інтервалі $(0, x_0)$ і задовольняє умовам:

а) $0 < f(x) < x$ при $0 < x < x_0$;

б) $f(x) = x - ax^2 + bx^3 + cx^4 + 0\left(x^4\right)$ при $x \to 0$, де a –додатна, a, b і c – довільні сталі.

Тоді для рекурентної послідовності $x_n = f(x_{n-1})$, $x_1 \in (0, x_0)$ при $n \to \infty$ має місце рівність

$$x_n = \frac{1}{an} + \frac{b - a^2}{a^3} \cdot \frac{\ln n}{n^2} + \frac{\varphi(x_1)}{an^2} + \frac{(b - a^2)^2}{a^5} \cdot \frac{\ln^2 n}{n^3} + o\left(\frac{\ln^2 n}{n^3}\right), \qquad (1)$$

де $\varphi(x)$ – деяка функція, визначена на $(0, x_0)$, яка залежить від $f(x)$ і задовольняє співвідношенню $\varphi(f(x)) = \varphi(x) - 1$ при $x \in (0, x_0)$ (2)

Розглянемо приклади.

1) Нехай $f(x)=x-x^2$. Тоді для відповідної рекурентної послідовності $\{x_n\}$, в якій $0<x_1<1$, маємо:

$$x_n=\frac{1}{n}-\frac{\ln n}{n^2}+\frac{\varphi(x_1)}{n^2}+\frac{\ln^2 n}{n^3}+o\left(\frac{\ln^2 n}{n^3}\right)$$

2) Для послідовності $x_n=\ln(1+x_{n-1}), x_1>0$, при $n\to\infty$ правильні рівність:

$$x_n=\frac{2}{n}+\frac{2}{3}\cdot\frac{\ln n}{n^2}+\frac{2\varphi(x_1)}{n^2}+\frac{2\ln^2 n}{9n^3}+o\left(\frac{\ln^2 n}{n^3}\right)$$

Теорема 2. Нехай функція $f(x)$ неперервна на деякому інтервалі $(0;x_0)$ і задовольняє умовам:

а) $0<f(x)<x$ при $0<x<x_0$.

б) $f(x)=x+\sum_{k=2}^{m+2} f_k x^k+o(x^{m+2})$ при $x\to 0$

де m – деяке натуральне число, а $f_k\,(k=2,...,m+2)$ – дійсні числа, причому $f_2<0$.

Тоді для рекурентної послідовності $x_n=f(x_{n-1}), x_1\in(0,x_0)$ при $n\to\infty$ має місце рівність:

$$x_n=\sum_{i=1}^{m}\sum_{k=0}^{i=1} a_i^{(k)}\frac{\ln^k n}{n^i}+a_{m+1}^{(m)}\frac{\ln^m n}{n^{m+1}}+o\left(\frac{\ln^m n}{n^{m+1}}\right), \qquad (3)$$

де $a_i^{(k)}(i=1,...,m;k=0,...,i-1)$ і $a_{m+1}^{(m)}$ – деякі константи.

Таким чином згідно [6], рекурентна послідовність $\{x_n\}$ може бути вивчена за допомогою теорем 1 і 2.

Уточнення деяких асимптотичних рівностей для загального члену деяких рекурентних послідовностей.

Теорема 1.1. Нехай функція $f(x)$ неперервна на деякому інтервалі $(0;x_0)$ і задовольняє умовам:

а) $0<f(x)<x$ при $0<x<x_0$

б) $f(x)=x-ax^2+bx^3$ при $x\to 0$, де a – додатна, а b – довільна стала,

тоді для рекурентної послідовності $x_n=f(x_{n-1}),\ x_1\in(0,x_0)$ при $n\to\infty$ має місце рівність:

$$x_n=\frac{1}{an}+\frac{2(b-a^2)}{a^3}\frac{\ln n}{n^2}+\frac{2(b-a)^2}{a^5}\frac{\ln^2 n}{n^3}+o(\frac{\ln^2 n}{n^3}) \qquad (4)$$

Доведення.

З умови а) випливає, що $\{x\}_{n=1}^{\infty}$ – нескінченно мала монотонно спадна послідовність додатних чисел. Використавши двічі теорему Штольца і умову б), отримуємо:

$$\lim_{n\to\infty} nx_n = l \lim_{n\to\infty} \frac{n}{x_n^{-1}} = \lim_{n\to\infty} \frac{x_n x_{n+1}}{x_n - x_{n+1}} = \frac{1}{a},\ x_n = \frac{1}{an} + o\left(\frac{1}{n}\right) \quad (5)$$

$$\lim_{n\to\infty} \frac{n^2}{\ln n}\left(x_n - \frac{1}{an}\right) = \frac{1}{a}\lim_{n\to\infty} \frac{n - \frac{1}{ax_n}}{\ln n} = \frac{b-a^2}{a^3}$$

$$x_n = \frac{1}{an} + \frac{b-a^2}{a^3}\frac{\ln n}{n^2} + o(\frac{\ln n}{n^2}) \text{ при } n \to \infty \quad (6)$$

$$x_{n+1} = \frac{1}{an} + \frac{b-a^2}{a}\ln n \frac{1}{a^2 n^2} + o(\frac{\ln n}{n^2}) = x_n + \frac{b-a^2}{a}\ln n x_n^2 + o(\frac{\ln n}{n^2}) =$$

$$= \frac{1}{an} + \frac{b-a^2}{a^3}\frac{\ln n}{n^2} + \frac{b-a^2}{a}\ln n(\frac{1}{an} + \frac{b-a^2}{a^3}\frac{\ln n}{n^2})^2 + o(\frac{\ln n}{n^2}) =$$

$$= \frac{1}{an} + \frac{b-a^2}{a^3}\frac{\ln n}{n^2} + \frac{b-a^2}{a}\ln n(\frac{1}{a^2 n^2} + \frac{2(b-a^2)}{a^5}\frac{\ln^2 n}{n^3} + \frac{(b-a^2)^2 \ln^2 n}{a^6 n^4}) + o(\frac{\ln n}{n^2}) =$$

$$= \frac{1}{an} + \frac{b-a^2}{a^3}\frac{\ln n}{n^3} + \frac{(b-a^2)\ln n}{a^3 n^2} + \frac{2(b-a^2)\ln^2 n}{a^5 n^3} + \frac{(b-a^2)^3 \ln^3 n}{a^7 n^4} =$$

$$= \frac{1}{an} + \frac{2(b-a^2)}{a^3}\frac{\ln n}{n^2} + \frac{2(b-a^2)^2}{a^5}\frac{\ln^2 n}{n^3} + o(\frac{\ln^2 n}{n^3}).$$

Теорему доведено.

Розглянемо приклади:

1)* Нехай $f(x) = x - x^2$. Тоді для відповідної рекурентної послідовності $\{x_n\}$, в якій $0 < x < 1$, маємо:

$$x_n = \frac{1}{n} - \frac{2\ln n}{n^2} + \frac{2\ln^2 n}{n^3} + o\left(\frac{\ln^2 n}{n^3}\right)$$

2)* Для послідовності $x_n = \ln(1 + x_{n-1}), x > 0$, при $n \to \infty$ правильна рівність:

$$x_n = \frac{2}{n} + \frac{2\left(-\frac{1}{4}\right)}{\frac{1}{32}} \cdot \frac{\ln n}{n^2} + \frac{2\left(0 - \frac{1}{4}\right)^2 \ln^2 n}{\frac{1}{32} \cdot n^3} + o\left(\frac{\ln^2 n}{n^3}\right) =$$

$$\frac{2}{n} + \frac{4\ln n}{n^2} + \frac{4\ln^2 n}{n^3} + o\left(\frac{\ln^2 n}{n^3}\right),$$

тобто

$$x_n = \frac{2}{n} + \frac{4\ln n}{n^2} + \frac{4\ln^2 n}{n^3} + o\left(\frac{\ln^2 n}{n^3}\right).$$

Теорема 2.1. Нехай функція $f(x)$ неперервна на деякому інтервалі $(0; x_0)$ і задовольняє умовам:

а) $0 < f(x) < x$ при $0 < x < x_0$;

б) $f(x) = x - ax^2 + bx^3$ при $x \to 0$, де a – додатна, а b – довільна стала,

тоді для рекурентної послідовності $x_n = f(x_{n-1})$ $x \in (0, x_0)$ при $n \to \infty$ має місце рівність

$$x_n = \frac{1}{an} + \frac{b-a^2}{a^3}\frac{\ln n}{n^2} + \frac{1}{an^2}(-1) + o(\frac{\ln^2 n}{n^3}) \qquad (7)$$

Доведення.

З умови а) випливає, що $\{x_n\}_{n=1}^{\infty}$ – нескінченно мала монотонно спадна послідовність додатних чисел. Використавши двічі теорему Штольца і умову б), отримуємо:

$$\lim_{n\to\infty} nx_n = \lim_{n\to\infty} \frac{n}{x_n^{-1}} = \lim_{n\to\infty} \frac{x_n x_{n+1}}{x_n - x_{n+1}} = \frac{1}{a};\ x_n = \frac{1}{a_n} + o(\frac{1}{n}); \qquad (8)$$

$$\lim_{n\to\infty} \frac{n^2}{\ln n}(x_n - \frac{1}{a_n}) = \frac{1}{a}\lim_{n\to\infty} \frac{n - \frac{1}{ax_n}}{\ln n} = \frac{b-a^2}{a^3};$$

$$x_n = \frac{1}{an} + \frac{b-a^2}{a^3}\frac{\ln n}{n^2} + o(\frac{\ln n}{n^2}) \text{ при } n \to \infty \qquad (9)$$

$$x_n^2 = \frac{1}{a^2n^2} + \frac{2(b-a^2)}{a^4}\frac{\ln n}{n^3} + \frac{(b-a^2)^2 \ln^2 n}{a^6 n^4};$$

$$x_n^3 = \frac{1}{a^3n^3} + 3\frac{1}{a^2n^2}\frac{b-a^2}{a^3}\frac{\ln n}{n^2} + 3\frac{1}{an}\frac{(b-a^2)\ln^2 n}{a^6n^4} + \frac{(b-a^2)^3 \ln^3 n}{a^9 n^6}.$$

Оскільки, $x_n = f(x_{n-1})$, то

$$x_{n+1} = x_n - ax_n^2 + bx_4^3 = \frac{1}{an} + \frac{b-a^2}{a^3}\frac{\ln n}{n^2} - \frac{1}{an^2} - \frac{2(b-a^2)\ln n}{a^3n^3} - \frac{(b-a^2)^2\ln^2 n}{a^5n^4} +$$

$$+ \frac{b}{a^3n^3} + \frac{3b(b-a^2)\ln n}{a^5n^4} + + \frac{3b(b-a^2)^2\ln^2 n}{a^7n^5} + \frac{b(b-a^2)^3\ln^3 n}{a^9n^6} =$$

$$= \frac{1}{an} + \frac{b-a^2}{a^3n^2}\ln n + \frac{1}{an^2}(-1) + o(\frac{\ln^2 n}{n^3})$$

Теорему доведено.

Розглянемо приклади:

1)** Нехай $f(x) = x - x^2$. Тоді для відповідної рекурентної послідовності $\{x_n\}$, в якій $0 < x < 1$, маємо:

$$x_n = \frac{1}{n} - \frac{\ln n}{n^2} - \frac{1}{n^2} + o\left(\frac{\ln^2 n}{n^3}\right)$$

2)** Для послідовності $x_n = \ln(1 + x_{n-1}), x > 0$, при $n \to \infty$ правильна рівність:

$$x_n = \frac{2}{n} - \frac{\left(\frac{1}{4}\right)}{\frac{1}{8}} \cdot \frac{\ln n}{n^2} - \frac{2}{n^2} + o\left(\frac{\ln^2 n}{n^3}\right) = \frac{2}{n} - \frac{2\ln n}{n^2} - \frac{2}{n^2} + o\left(\frac{\ln^2 n}{n^3}\right),$$

тобто $x_n = \frac{2}{n} - \frac{2\ln n}{n^2} - \frac{2}{n^2} + o\left(\frac{\ln^2 n}{n^3}\right)$.

Наслідки та ілюстративні приклади до теорем 1.1 та 2.1.

При визначенні x_n через n не враховується x_1, тому щоб врахувати x_1 потрібно підрахунок провести не до n, як пропонується в теоремах 1.1, 2.1, а до $n+k-1$. Для визначення k пропонується наслідок 1 та наслідок 2.

Наслідок 1. У теоремі 1.1 загальний член послідовності знаходився за формулою:

$$x_n = \frac{1}{an} + \frac{2(b-a^2)}{a^3}\frac{\ln n}{n^2} + \frac{2(b-a)^2}{a^5}\frac{\ln^2 n}{n^3} + o(\frac{\ln^2 n}{n^3}) .$$

При врахуванні x_1, загальний член послідовності x_n може бути представлений формулою:

$$x_n = \frac{1}{a(n+k-1)} + \frac{2\left(b-a^2\right)}{a^3}\frac{\ln\left(n+k-1\right)}{\left(n+k-1\right)^2} + \frac{2\left(b-a^2\right)^2}{a^5}\frac{\ln^2\left(n+k-1\right)}{\left(n+k-1\right)^3} + o\left(\frac{\ln^2 n}{n^3}\right),$$

де k залежить від x_1 і визначається з рівності:

$$x_1 = \frac{1}{ak} + \frac{2\left(b-a^2\right)}{a^3}\frac{\ln k}{k^2} + \frac{2\left(b-a^2\right)^2}{a^5}\frac{\ln^2 k}{k^3}.$$

Наслідок 2.

У теоремі 2.1 загальний член послідовності знаходився за формулою:

$$x_n = \frac{1}{an} + \frac{b-a^2}{a^3}\frac{\ln n}{n^2} + \frac{1}{an^2}(-1) + o(\frac{\ln^2 n}{n^3}) .$$

При врахуванні x_1, загальний член послідовності x_n може бути представлений формулою:

$$x_n = \frac{1}{a(n+k-1)} + \frac{b-a^2}{a^3}\frac{\ln\left(n+k-1\right)}{\left(n+k-1\right)^2} - \frac{1}{a\left(n+k-1\right)^2} + o\left(\frac{\ln^2 n}{n^3}\right),$$

де k залежить від x_1 і визначається з рівності: $x_1 = \frac{1}{ak} + \frac{b-a^2}{a^3}\frac{\ln k}{k^2} - \frac{1}{ak^2}$.

Наведемо приклади, які ілюструють висновки, отримані в теоремах 1.1, 2.1. **У** таблицях 1 і 2 для різних значень n підраховано точне значення x_n, наближене значення з теоремами 1.1 та 2.1. Таблиця 1 ілюструє приклад 1. Таблиця 2 ілюструє приклад 2**.**

Висновки. У даній роботі сформульовано та доведено Теорему 1.1 та Теорему 2.1, які є уточненням результатів Теорем 1 та 2 [4]. Висновки теорем перевірені експериментально, про що говорить таблиця 1 та 2. При досить великих n експериментальні дані майже співпадають з точними, що вказує на правильність теорем.

Список використаних джерел

1. Креневич А.П. Алгоритми і структури даних. Підручник. – К.: ВПЦ "Київський Університет", 2021. – 200 с.
2. Коротєєва Т. О. Алгоритми та структури даних: навч. Посібник. Львів: Львівської політехніки, 2014. – 280 с.
3. В. М. Пивоварчик, О. М. Яковлєва, О. М. Болдарєва. Дискретна математика (частина 1). Навчальний посібник. – Одеса : ПНПУ імені К. Д. Ушинського, 2022. – 145 с.
4. Жалдак М. І., Г. О. Михалін, С. Я. Деканов. Математичний аналіз з елементами інформаційних технологій: навчальний посібник. – К. : Редакції газет природничо-математичного циклу, 2012. – 128 с.
5. Клочко В. І., З. В. Бондаренко. Формування знань майбутніх інженерів з інформаційних технологій розв'язування диференціальних рівнянь : монографія – Вінниця: ВНТУ, 2010. – 216 с
6. Линчук С.С. Про швидкість збіжності рекурентних послідовностей. Математика, Випуск 11 (1997), С. 72-79.
7. Дороговцев А. Я. Математичний аналіз: підручник. К.: Либідь, 1993. – 320 с.

References.

1. Krenevich, A. (2021). Algorithms and data structures: [textbook] [in Ukrainian].
2. Koroteeva, T. (2014). Algorithms and data structures: [textbook] [in Ukrainian].
3. Pyvovarchyk, V., Yakovleva, O., Boldareva, O. (2022). Discrete mathematics (part 1): [textbook] [in Ukrainian].
4. Zhaldak, M., Mikhalin, G., Dekanov, S. (2012). Mathematical analysis with elements of information technology: [textbook] [in Ukrainian].
5. Klochko, V., Bondarenko, Z. (2010). Formation of knowledge of future engineers on information technologies for solving differential equations: [monograph] [in Ukrainian].
6. Lynchuk, S. (1997). On the rate of convergence of recurrent sequences. Mathematics, Issue 11.
7. Dorogovtsev, A. (1993). Mathematical analysis: [textbook] [in Ukrainian].

BOSOVSKIY Mykola,
PhD (Pedagogical Sciences), Associate Professor of the Department of Mathematics and Methods of Learning of Mathematics,Cherkasy Bohdan Khmelnytsky National University
SERDIUK Zoia,
PhD (Pedagogical Sciences), Associate Professor of the Department of Mathematics and Methods of Learning of Mathematics,Cherkasy Bohdan Khmelnytsky National University

ASYMPTOTIC ESTIMATION OF SOME RECURRENT SEQUENCES

Abstract. Introduction. *The training of modern specialists often requires a symbiosis of knowledge from different fields. Basic knowledge of mathematical and IT disciplines, as well as their successful application, is a guarantee of the formation of professional skills in such specialists. Recurrence relations are extremely important for programming. They are used in algorithm analysis, approximate calculations, dynamic programming, etc. When considering recurrent sequences, one of the main problems is its solution, that is, it is necessary to express xn in terms of n. This task is not always solved. Therefore, the question arises of refining n to find xn depending on how the first term x0 is chosen.*

Originality. *The purpose of asymptotic methods is to obtain O-estimates and o-estimates in cases where it is quite difficult to use the function definition for very large (or very small) values of the argument. Sometimes it is easier to obtain asymptotic information than any other.*

Neither the O-score nor the o-score in their usual form are directly applicable for computational purposes. However, in almost all cases where such estimates are available, it is possible, after reviewing the proof, to replace the O-estimates with inequalities that contain numerical constants. For this, at each stage of our actions, we must indicate certain numbers or functions with certain properties where, when obtaining asymptotic estimates, we limited ourselves to proving the existence of such numbers or functions. In this work, the result for one class of recurrent infinitesimal

recurrent sequences is refined.

Conclusion. *In this article, Theorem 1.1 and Theorem 2.1 are formulated and proved, which are refinements of the results of Theorems 1 and 2.*

The conclusions of the theorems have been tested experimentally, which is shown in Tables 1 and 2. For sufficiently large experimental data, they almost coincide with the exact ones, which indicates the correctness of the theorems

Одержано редакцією 25.08.2023 р.
Прийнято до публікації 27.09.2023 р.

СЕКЦІЯ «ІНФОРМАТИКА»

УДК 004.738.5:378.14

DOI 10.31651/2076-5886-2023-1-49-60

PACS 89.20.Hh

ХОВАЙБА Дарина Євгенівна
студентка спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького
e-mail:
tovstopiat.daryna1618@vu.cdu.edu.ua

ДІДКОВСЬКИЙ Руслан Михайлович
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОГО ВЕБ-САЙТУ КАФЕДРИ ЗАКЛАДУ ВИЩОЇ ОСВІТИ

Розглядається процес розробки інформаційного веб-сайту для кафедри прикладної математики та інформатики закладу вищої освіти. Описано етапи, що включають аналіз вимог до сайту, проектування структури та дизайну, розробку інтерактивного прототипу у Figma та налаштування CMS-платформи. Веб-сайт має на меті забезпечити ефективну комунікацію між студентами, викладачами та адміністрацією, а також підвищити імідж навчального закладу. Основні розділи сайту включають інформацію про кафедру, ресурси для студентів, новини та контакти. Використання сучасних графічних елементів, шрифтів та кольорових схем сприяє привабливості та функціональності сайту. Особливу увагу приділено вибору CMS для управління контентом, для якого обрано WordPress завдяки його простоті, функціональності та широкій підтримці спільноти. Фінальні етапи включають встановлення та налаштування WordPress на хостингу, що завершує процес розробки. Стаття демонструє ключові аспекти створення ефективного веб-ресурсу для освітньої установи.

Ключові слова: *проектування веб-сайту, інформаційний веб-сайт, Figma, WordPress, CMS, освітній процес, комунікація.*

Вступ

В сучасному цифровому світі, інформаційні технології мають величезне значення для будь-якої сфери діяльності, включаючи освіту. Веб-сайти стають необхідним інструментом для закладів вищої освіти, оскільки вони надають можливість ефективно комунікувати зі студентами, викладачами та іншими зацікавленими сторонами. Основна мета цієї роботи полягає в розробці та реалізації веб-сайту для кафедри, що дозволить поліпшити комунікацію, забезпечити доступ до необхідної інформації та покращити взаємодію зі студентами, викладачами та іншими зацікавленими особами.

Мета статті - проектування та реалізація інформаційного веб-сайту для кафедри закладу вищої освіти.

Виклад основного матеріалу

1. Аналіз вимог до веб-сайту кафедри ЗВО та підготовка до розробки

Веб-сайти підрозділів закладів вищої освіти сприяють покращенню комунікації, доступу до інформації та навчання, а також мають певний вплив на студентів та викладачів. Веб-сайти в освітній сфері створюють нові можливості для ефективної комунікації та взаємодії між різними учасниками навчального процесу. Студенти, викладачі та адміністрація мають змогу обмінюватися інформацією, спілкуватися та співпрацювати через електронні форми зворотного зв'язку, форуми, чати чи інші інтерактивні елементи веб-сайту. Це допомагає покращити комунікацію в межах навчального закладу, залучити студентів до активної участі в дискусіях та сприяти взаєморозумінню.

Сайти в наш час стають центральним джерелом інформації для вступників, студентів, викладачів та інших зацікавлених осіб. На них розміщуються актуальні дані про навчальні програми, курси, розклад занять, шаблони, матеріали та ресурси для навчання. Це дозволяє студентам легко знаходити потрібну інформацію, бути в курсі останніх оновлень та забезпечує доступність освітнього матеріалу навіть поза класними кімнатами. Вони також можуть містити додаткові розділи зі статтями, дослідженнями та іншими ресурсами, що поглиблюють знання студентів та стимулюють їх академічний розвиток, а ще інформацію про викладачів, їхню освіту, які предмети вони викладають та їхні публікації.

Привабливий та функціональний веб-сайт допомагає навчальному закладу підняти свій імідж та привернути увагу потенційних студентів. Сучасний веб-дизайн, зручність навігації, якісний контент та актуальна інформація створюють позитивне враження про навчальний заклад. Веб-сайт може бути використаний як ефективний інструмент маркетингу для просування навчального закладу та спеціальностей, привертання нових студентів та збільшення популярності навчальної програми – це дуже важливо як впродовж навчання, так і в період вступної кампанії, адже абітурієнти шукають інформацію про університети саме в інтернеті.

Веб-сайти в освітній сфері мають велику роль та значення. Вони створюють нові можливості для комунікації, забезпечують доступ до інформації та навчальних ресурсів, розширюють можливості навчання та сприяють підвищенню ефективності та продуктивності навчальних закладів. Веб-сайти також допомагають покращити імідж навчального закладу та залучити нових студентів. Враховуючи швидкий розвиток технологій та зміни у способах навчання, веб-сайти відіграють все більш важливу роль у сучасній освітній системі.

Структура сайту – це ієрархія розміщення матеріалів, що розміщуються на сайті. Чітка та логічно структурована інформація дає такі переваги: відвідувачам легко знайти необхідну інформацію та і самому адміністратору сайту легше доповнювати і оновлювати матеріали, не створюючи плутанини, тому визначення основних розділів та структури сайту кафедри є важливим етапом проектування інформаційного веб-сайту. Інформація на сайті повинна бути чітко структурована та оптимізована для мінімізації кліків для доступу до потрібної інформації.

Відвідувач заходить на веб-сайт кафедри ЗВО з метою швидкої та ефективної пошуку необхідної інформації (щодо навчального закладу, доступних професій, навчальних планів). Він бажає здійснити цю операцію швидко і економно витратити свій час, тому на сайті має бути своя система, аналогічна порядку в архівах, де кожна справа знаходиться на своєму визначеному місці.

Навігація сайту (меню) повинна відображати інформаційну структуру сайту, бути простою, зрозумілою у використанні і знаходитися в помітному місці сторінки. Назви в

меню повинні повідомляти користувачеві, що після натискання на відповідне посилання, він потрапить в потрібний йому розділ чи категорію.

Основні розділи, які мають бути на сайті кафедри, включають (Рис. 1):

а) головна сторінка: це стартова сторінка сайту, на якій буде розміщена загальна інформація про кафедру, важливі новини та оголошення;

б) про кафедру: в цьому розділі буде розміщена детальна інформація про кафедру, її мету, завдання, склад викладачів, інформація про вісник та наукові досягнення;

в) студенту: в цьому розділі будуть розміщені корисні ресурси та інформація для студентів, включаючи розклад занять, розклад заліків та екзаменів, посилання на анкетування, матеріали для захисту робіт, робочі програми навчальних дисциплін та інформація про вибіркові дисципліни;

г) вступнику: в цьому розділі буде розміщена актуальна інформація про вступ, інформація про спеціальності кафедри та навчальні плани;

ґ) новини: в цьому розділі будуть новини університету, навчально-наукового інституту та кафедри;

д) контакти: у даному розділі будуть наведені контактні дані кафедри, посилання на соціальні мережі, форма зворотнього зв'язку та інформація про розташування.

Рис 1. Структура майбутнього веб-сайту

2. Створення складових елементів дизайну

Складові елементи дизайну включають в себе графічні елементи, зображення, шрифти і кольорову палітру. Графічні елементи повинні утримувати увагу відвідувачів, не відволікаючи їх від інформаційного контенту веб-сайту. Рекомендується обмежувати кількість використовуваних кольорів до трьох, аби кольорова палітра не мала надмірного впливу на очі. Вибір кольорів важливий для визначення фірмового стилю кафедри і суттєвий для зорового сприйняття інформації відвідувачами.

Графічні елементи мають відображати зміст сайту, а не бути лише прикрасою. Застосування анімованих заставок, великої кількості відео та надмірної анімації може викликати негативні емоції у користувачів, оскільки ці ефекти можуть заважати швидкому доступу до необхідної інформації, затягувати час і відволікати увагу. Крім того, це може суттєво впливати на час завантаження веб-сайту.

Підбір оптимальних шрифтів та кольорів для сайту залежить від багатьох факторів, таких як тема, цільова аудиторія, бренд-ідентичність та особливості кафедри прикладної математики та інформатики. Згідно тематики були підібрані такі шрифти:

1) Для заголовків та акцентів було обрано сучасний та виразний sans-serif шрифт: Roboto Slab.

2) Для текстового контенту буде використаний читабельний та простий serif шрифт: Open Sans.

Це чудова шрифтова пара, яку часто використовують для корпоративних чи ділових сайтів і до того ж, з метою мінімізації витрат на розробку перевагою є ліцензія free for commercial use, що дозволяє законно використовувати шрифти для особистого чи комерційного проєкту без оплати.

Важливо також враховувати доступність та контрастність шрифтів та кольорів, щоб забезпечити зручну читабельність для всіх користувачів. Рекомендується використовувати як основний колір, який відповідає бренд-ідентичності кафедри, темно синій (#192A56) у поєднанні з сірувато-білим (#F4F5F8). Як акцентний колір підібрано світло-синій (#2482BF), який вказує на зв'язок з областю математики та інформатики. Сині кольори використовуються на ділових веб-сайтах з метою створення професійного та довірливого враження на користувачів. Синій колір асоціюється зі стабільністю, спокоєм та довірою, що робить його ідеальним вибором для сайтів, пов'язаних з освітою, науковими дослідженнями та інформаційними ресурсами.

Логотип для сайту був підібраний згідно тематики на безкоштовному сайті flaticon.com та допрацьований за допомогою редактору Adobe Illustrator. Так як це сайт кафедри прикладної математики та інформатики, то настільний комп'ютер з технічним мозком на заставці і смартфоном поряд чудово відображає суть і має кілька символічних значень, а саме:

1) відображення спеціалізації кафедри;
2) посилання на сучасні технології;
3) зв'язок зі зручністю та доступністю;
4) визначення ідентичності та впізнаваності.

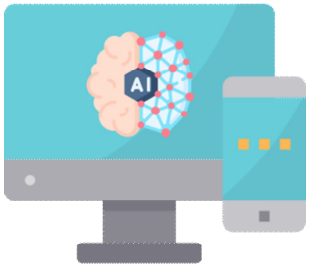

Рис. 2 Логотип сайту кафедри

3. Прототипування та розробка фінального дизайну сайту

Прототипування – це схематичне зображення блоків та тексту з котрих складається сайт. Малюється для того, щоб було візуально видно розташування елементів і навігацію. Створення прототипу — це один із кроків, який передує запуску фінальної пропозиції. Фактично, створення прототипу майбутнього продукту, або сервісу має кілька переваг:

1) можливість оцінити технічну доцільність;
2) можливість покращити якість веб-сайту;
3) можливість продемонструвати свою ідею більш наглядно;
4) зменшити ризик;
5) імітувати реальний продукт;
6) отримати чіткий зворотній зв’язок.

Для розробки мінімального прототипу було використано Figma і елементи «Текст» та «Прямокутник».

Головна сторінка є ключовою частиною сайту, оскільки вона першою зустрічає відвідувачів. У фінальному дизайні головної сторінки потрібно забезпечити зручну навігацію, привабливий вигляд, відображення ключової інформації та ефективну презентацію контенту згідно узгодженого прототипу. Кожна внутрішня сторінка має бути розроблена з урахуванням єдиної візуальної концепції головної сторінки. Важливо забезпечити послідовність та спрощену навігацію між сторінками, використовуючи зручне меню, посилання та інші навігаційні елементи. Всі графічні елементи мають підкреслювати тематику кафедри прикладної математики та інформатики. Наприклад, це можуть бути ілюстрації з математичними символами, фотографії студентів та викладачів, або графіки, пов'язані з областями досліджень викладачів кафедри.

Також не варто забувати за адаптивність, оскільки сайти використовуються на різних пристроях і екранах, важливо розробити фінальний дизайн, який буде коректно відображати контент на різних розмірах екранів.

Отже, у Figma були створені всі сторінки сайту і наповнені відповідним контентом, дотримана тематика та патерни продуманого дизайну. Також для зручності майбутньої верстки був створений UI Kit.

UI Kit, або User Interface Kit (набір елементів інтерфейсу користувача), є набором графічних елементів, компонентів і інструментів, які використовуються для розробки і стилізації інтерфейсів користувача, веб-сайту тощо.

Ще один важливий етап – розробка адаптивного дизайну для сайту кафедри прикладної математики та інформатики - це забезпечить оптимальний перегляд та взаємодію з сайтом на різних пристроях та розмірах екранів, оскільки в наш час пошук ресурсів в інтернеті здійснюється не тільки через комп’ютери. Адаптивний дизайн дозволяє оптимізувати відображення контенту для різних пристроїв, таких як комп'ютери, планшети та смартфони. У Figma був розроблений дизайн всіх екранів для мобільної версії, за цим зразком сайт буде адаптований для планшетів.

4. Створення інтерактивного прототипу в розділі Prototype

Створення інтерактивного прототипу в розділі "Prototype" в Figma дозволяє візуалізувати та перевірити функціонал та взаємодію елементів на майбутньому веб-сайті перед фактичною реалізацією та верстkою. Інтерактивний прототип дозволяє створити динамічну модель сайту чи додатку, де фокус група чи команда розробників можуть взаємодіяти з елементами, переходити між сторінками, заповнювати форми та спостерігати за анімацією, щоб перевірити логіку сайту та коректність роботи.

Основні кроки створення інтерактивного прототипу в розділі "Prototype" Figma:

1. Визначити перехідні точки: потрібно встановити, які елементи на сторінці будуть взаємодіяти з користувачем. Це зазвичай кнопки, посилання, меню, форми та інші елементи, на які користувач може клікати або взаємодіяти якимось іншим чином.
2. Додати перехідні зв'язки: використовуючи інструменти Figma, необхідно з'єднати перехідні точки, створюючи зв'язки між елементами, що взаємодіють.
3. Визначити типи переходів: далі треба встановити типи переходів між сторінками або елементами. Також можна встановити анімаційні ефекти деяких елементів для більшої інтерактивності та залучення користувачів до перегляду сайту.
4. Додати взаємодію: треба встановити взаємодію з елементами, які можуть бути пов’язані, такими як кнопки, форми або меню. Наприклад, можливість заповнювати поля форми, переходити на іншу сторінку після натискання кнопки тощо.

5. Перегляд та тестування прототипу: щоб перевірити, чи функціонують всі переходи та взаємодії треба перейти в режим перегляду, і переконатись, що користувачеві надається зручний та приємний досвід взаємодії з сайтом та всі кнопки та навігація працюють коректно, згідно задуму.

Створення інтерактивного прототипу дозволяє зробити передбачувану перевірку та оцінку функціоналу та дизайну майбутнього веб-сайту перед його верстkою, також це дозволяє отримати більш чітке уявлення про кінцевий продукт та забезпечити високу якість та задоволення користувачів.

За посиланням: http://surl.li/inkta можна переглянути проєкт та протестувати функціонал сайту в розділі Prototype.

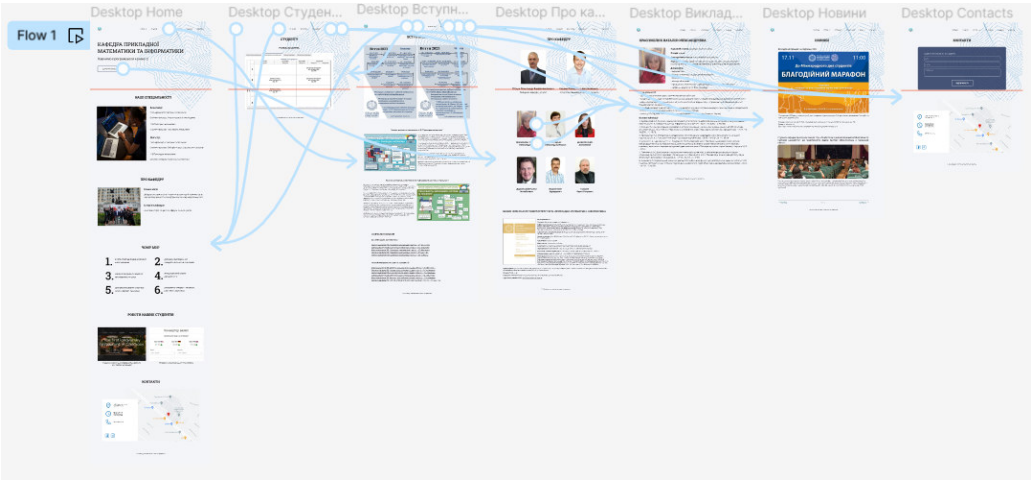

Рис 3. Переходи між сторінками в розділі Prototype

5. Вибір та початкові налаштування платформи для розміщення веб-сайту

Веб-сайт освітнього закладу розробляється відповідно до технічного завдання з метою забезпечити можливість людям без практичного досвіду ефективно управляти своїм веб-ресурсом, тому для створення сайту була вибрана система управління контентом (CMS). На даний момент топ-3 CMS системи:

– WordPress є платформою для управління вмістом, створеною для швидкої та простої розробки веб-сайтів. Заснована на PHP та MySQL, вона початково була спрямована на блоги, але з часом стала популярною для розробки різноманітних веб-сайтів. Це одна з найпопулярніших CMS систем у світі - проста у встановленні та використанні, має велику спільноту користувачів, безліч тем і плагінів для розширення функціональності, а також має хорошу технічну підтримку. WordPress підходить для різних типів веб-сайтів, від особистих блогів до корпоративних порталів, але для великих та складних проєктів є певні обмеження.
– Joomla також є популярною CMS системою з великою кількістю функцій та можливостей, яка позиціонується як середньорівневий варіант між WordPress та Drupal. Вона спрямована на гнучкість та розширення і підходить для середньо-розмірних до великих веб-сайтів, таких як корпоративні портали, онлайн-магазини, соціальні мережі тощо. Joomla має добру підтримку мультимовності та має розширену систему дозволів, але інтерфейс не такий інтуїтивно зрозумілий як у WordPress, тому на розробку може піти більше часу.
– Drupal є потужною CMS системою, яка зазвичай використовується для складних та великих веб-проектів. Вона забезпечує високий рівень гнучкості та

розширюваності, а також має високий рівень безпеки, розширену систему модулів та широкі можливості для створення користувацьких функцій, для розробки сайту на цій системі треба значно вищий рівень знань та певна експертність. Drupal підходить для корпоративних веб-сайтів, урядових порталів, академічних сайтів тощо.

– Вибір між цими CMS системами залежить від конкретних потреб проекту, розміру та складності веб-сайту, рівня технічних знань команди проекту та інших факторів.

Для розробки сайту кафедри прикладної математики та інформатики було обрано CMS систему WordPress з декількох причин.

Популярність та загально прийнятність: велика кількість веб-сайтів, від особистих блогів до корпоративних ресурсів, використовують WordPress, до того ж більша частина університетських сайтів та ресурсів розроблені саме на цій системі.

Простота використання: WordPress відомий своєю легкістю встановлення та використання. Це особливо важливо для веб-сайту кафедри, оскільки може бути різний рівень технічної підготовки користувачів. Простий інтерфейс системи дозволить персоналу, який відповідатиме за ведення сайту легко додавати та оновлювати контент без глибоких технічних знань.

Багатофункціональність: система забезпечує широкий функціонал за замовчуванням, такі як система коментарів, керування користувачами, інтеграція з соціальними мережами та багато іншого.

Дизайн та кастомізація: система має велику екосистему тем та плагінів, які дозволяють швидко налаштовувати та розширювати функціонал веб-сайту. Це дозволить легко впроваджувати необхідні функції для кафедри без значних зусиль та глибоких технічних знань.

SEO-налаштування: дана система має ефективні та розширені інструменти для SEO-налаштувань, які дозволяють оптимізувати сайт для пошукових систем та підняти його в пошуку вище - чітко визначені URL-адреси, мета-теги, карти сайту та інші функції сприяють покращенню позицій у пошукових результатах.

Безпека: команда WordPress активно веде роботу з питань безпеки та регулярно випускає оновлення. Це гарантує стабільну та безпечну роботу веб-сайту, що є важливим для проекту в освітньому сегменті.

Технічна підтримка: WordPress має велику та активну спільноту користувачів та розробників, а отже завжди є доступ до допомоги, документації та ресурсів для вирішення будь-яких питань або проблем. Для навчальних закладів це є важливим аспектом для успішного ведення сайту.

З урахуванням цих факторів, вибір WordPress для розробки сайту кафедри прикладної математики та інформатики є обґрунтованим та може сприяти успішній та легкій реалізації поставленого завдання.

6. Інсталяція та налаштування WordPress на хостингу

Щоб встановити систему управління WordPress, слід завантажити останню версію дистрибутиву з офіційного веб-сайту www.wordpress.org та налаштувати її. Під час початкового створення веб-сайту на основі WordPress необхідно зареєструвати доменне ім'я та веб-хостинг, на якому буде розміщено сайт.

При виборі домена слід враховувати наступне:

– домен повинен бути легко запам'ятовуваним;
– мати просте ім’я;
– відображати ідею веб-сайту, до якого він належить;

– містити ключові слова, зрозумілі для пошукових систем.

Після огляду різних хостингів вибір пав на хостинг www.ho.ua, адже він має тарифний план, який повністю задовольняє потреби даної розробки. Надалі сайт буде перенесений на хостинг університету. Доменне ім'я обиралось на самому хостингу з доступних варіантів і це - www.pmi.ho.ua. У панелі управління хостинг-провайдера у розділі бази даних треба створити нову базу даних для майбутнього WordPress-сайту та записати назву бази даних, ім'я користувача та пароль, оскільки вони будуть потрібні під час налаштування сайту на WordPress. Керування базою даних сайту виконується через відповідний акаунт на сервісі phpMyAdmin.

Для підключення до свого хостинг-серверу використаємо FTP-клієнт FileZilla та перейдемо до каталогу, в якому необхідно встановити WordPress і завантажити всі файли WordPress на сервер за допомогою FTP-клієнта.

Підключення бази даних до сайту відбувається за допомогою файлу wp-config-sample.php, даний файл є в кореневому каталозі сайту. Його назву потрібно змінити на wp-config.php та відкрити в будь-якому текстовому редакторі, щоб внести дані нашої бази даних, а саме назву, логін та пароль, які ми вводили при створенні бази даних на хостінгу.

У веб-браузері переходимо до доменного імені сайту – з'явиться екран встановлення WordPress. Обираємо мову встановлення та продовжуємо, на наступній сторінці вводимо майбутню назву сайту, а також інформацію про адміністратора, включаючи обліковий запис користувача та пароль. Натискаємо кнопку "Встановити WordPress" і чекаємо, поки процес установки завершиться, після цього нас перекине на Майстерню сайту.

Далі необхідно увійти в адміністративний розділ WordPress (wp-admin) за допомогою облікового запису, який був вказаний під час налаштування (веб-браузер - адреса "yourdomain.com/wp-admin/) та ввести свої облікові дані (ім'я користувача та пароль), щоб увійти в адміністративний розділ.

Після успішного входу в адміністративний розділ WordPress можна налаштовувати вигляд та функціональність сайту, встановлювати теми та плагіни, додавати контент та виконувати інші дії для створення і керування веб-сайтом.

7. Вибір теми та додаткових плагінів для веб-сайту кафедри

Для розробки сайту було обрано тему Astra, яка відома своєю високою яскравістю та швидкістю, що робить її ідеальною для будь-яких веб-сайтів. Astra пропонує сучасний і красивий дизайн, який можна легко налаштувати відповідно до індивідуальних потреб користувача. Завдяки своїй зручності для мобільних пристроїв і гнучкості ця тема ідеально підходить для створення різноманітних веб-проектів. Крім того, Astra оптимізовано для SEO та продуктивності, де це можливо, забезпечуючи високий рівень швидкості та ефективності завантаження сторінок і підвищення рейтингу сайту у пошукових системах.

Щоб зробити верстку сайту згідно розробленого дизайну, а не за стандартним шаблоном теми, то всі атрибути теми було видалено, а лишились тільки меню з конкретними налаштуваннями та футер, які відображаються однаковими для всіх сторінок сайту.

Для коректної роботі сайту та обраної теми необхідно було встановити додаткові розширення (плагіни), які не будуть до всього заважати роботі один одного:

– Elementor – це потужний конструктор сторінок WordPress, який дозволяє редагувати вміст сторінки в реальному часі. У конструкторі сайтів Elementor можна знайти усе,

що потрібно: візуальний конструктор сторінок, дизайн, який враховує кожен піксель, можливість адаптивного редагування для мобільних пристроїв та багато інших корисних функцій;

- Essential Addons for Elementor – це бібліотека додаткових елементів і функцій для Elementor. Він розширює функціональні можливості стандартного конструктора, дозволяючи додавати різноманітні елементи стилю та функціональність;
- Cookie Notice & Compliance for GDPR / CCPA – цей плагін важливий для веб-сайтів у зв'язку з вимогами щодо конфіденційності даних, таких як GDPR та CCPA. Він дозволяє легко додати попередження про файли cookie та гарантує відповідність законодавства з питань конфіденційності;
- Robin Image Optimizer – це інструмент оптимізації зображень для покращення швидкодії веб-сайту. Він допомагає автоматично стискати та оптимізувати зображення для покращення завантаження сторінки;
- Solid Security Basic – це плагін, який забезпечує безпеку веб-сайту, він надає різноманітні інструменти для захисту від хакерів, атак та інших загроз безпеці сайту;
- BetterDocs – це плагін, який спрощує та поліпшує процес створення та управління документацією для веб-сайтів. Заснований на ідеї полегшення доступу користувачів до інформації, BetterDocs дозволяє легко створювати структуровані документаційні сторінки та бібліотеки з ілюстраціями та вкладеними відеороликами. Зручний конструктор інтерфейсу дозволяє швидко оформити стильні та інтуїтивно зрозумілі документи;
- WP Fastest Cache – це плагін кешування, який допомагає прискорити завантаження сторінок веб-сайту для відвідувачів за рахунок збереження кешу та оптимізації роботи веб-сайту;
- Contact Form 7 – це безкоштовний та дуже популярний плагін для WordPress, який дозволяє легко створювати та управляти формами зв'язку на веб-сайті. Заснований на WordPress Shortcode API, CF7 забезпечує зручний інтерфейс для створення та керування різними формами;
- WOW Style Contact Form 7 – це плагін, який розширює функціональність плагіну Contact Form 7, надає додаткові стилі, анімації та можливості кастомізації для форм зворотнього зв'язку, надає додаткові інструменти для вдосконалення зовнішнього вигляду та функціональності форм;
- Premium Addons for Elementor – це розширення для плагіна Elementor, яке додає додаткові функції та елементи із преміальних до конструктора сторінок Elementor для WordPress. Цей плагін дозволяє розширити можливості веб-дизайну та розробки, додавши нові блоки, ефекти та інструменти для побудови зручних та стильних сторінок.

Кожен із цих плагінів виконує конкретні функції для покращення продуктивності, безпеки та зручності веб-сайту на платформі WordPress.

8. Верстка веб-сайту на платформі wordpress та налаштування сторінок

Верстка сайту згідно дизайну виконувалась через плагін Elementor – це зручний у використанні інструмент для побудови сторінок на WordPress. Щоб запустити свій сайт, всі доступні віджети розміщуються на лівій бічній панелі, і все можна редагувати візуально та перетягуванням. Там є 45+ елементів для будь-яких потреб дизайну сторінки, поділені на “Basic,” “General” і “WordPress”, також за допомогою плагіна Essential Addons for Elementor цей функціонал був розширений і тому доступно більше

елементів. Всі сторінки та секції можна зберігати для повторного використання, як шаблони.

Після створення сторінки слід вибрати опцію "Редагувати з Elementor", щоб відкрити графічний конструктор сторінок. Перед початком верстки необхідно виконати загальні налаштування сайту, а саме:

- глобальні кольори – визначення фірмових кольорів сайту;
- глобальні шрифти –конфігурація шрифтів для текстів ;
- типографія – налаштування параметрів різних текстових елементів на сайті, таких як звичайний текст, текст посилань та заголовки шести рівнів;
- ідентичність сайту – встановлення фірмових елементів вашого сайту, таких як назва сайту, короткий опис, логотип і фавікон;
- фон – визначення фону для всього сайту, у даному випадку це світлий колір з геометричним візерунком;
- переходи між сторінками – налаштування значень ширини в пікселях, коли вміст сайту масштабується під екрани планшетів або смартфонів.

Структура будь-якої сторінки в Elementor: секція > стовпчик > відмет, згідно цього правила для створення дизайну сторінки власноруч треба почати зі створення нової секції, вибрати потрібну кількість колонок. Після вибору структури в редакторі ви побачите порожній розділ. Натискання кнопки "+" у будь-якій колонці відкриє бічну панель з віджетами. Основна ідея полягає в тому, щоб перетягувати потрібні віджети до колонок, поетапно складаючи структуру вашого сайту. Синя тінь від віджету допомагатиме точно визначити місце, куди він буде поміщений.

Таким чином зроблена верстка всіх сторінок сайту і також посилання між ними через меню чи якорі. Посилання на веб-сайт кафедри: https://pmi.ho.ua.

Адаптив сайту для різних пристроїв відбувається також через плагін Elementor, для цього необхідно перейти в адаптивний режим натиснувши відповідну кнопку внизу зліва на панелі, де за замовчуванням доступні три стандартні екрани для десктопу, планшету та мобільного, але при потребі через плагін Essential Addons for Elementor можна додати ще різні модифікації, якщо це потрібно. В спеціальному редакторі для налаштування адаптивну можна переглянути та налаштувати всі елементи сайту в різних розширеннях екрану.

Для того, щоб налаштувати певний елемент сайту необхідно зробити такі кроки:

1. Натиснути на потрібний елемент.
2. Справа на панелі перейти на сторінку «Зміст».
3. В кожному пункті, який можна налаштувати стоїть іконка, яка позначає тип адаптивну і це можна налаштовувати під конкретний екран.
4. Перевірити чи не змінилась іконка та зберегти зміни.

У Elementor також можна приховувати секції, колонки або навіть окремі віджети деяких типів пристроїв. Щоб приховати елемент, треба перейти у його параметрах на бічній панелі на вкладку «Розширений» > розділ «Адаптивний». У ньому три вимикачі, які приховують елемент на комп’ютері, планшеті чи телефоні. У редакторі прихований елемент відображатиметься сірим кольором, а перейти в режим перегляду, то цей елемент взагалі пропаде.

Висновки

У статті було проведено дослідження щодо проектування та реалізації

інформаційного веб-сайту для кафедри закладу вищої освіти. Проектування та реалізація інформаційного веб-сайту кафедри закладу вищої освіти є важливим та актуальним завданням, яке сприяє поліпшенню освітнього процесу, комунікації та престижу навчального закладу. Результати дослідження можуть бути використані в практичній діяльності для розробки та впровадження веб-сайту кафедри з використанням інструментів, таких як Figma та WordPress.

Основні результати проведеної роботи полягають у наступному:

- розглянуто існуючі роботи та підходи до проектування веб-сайтів кафедр вищих навчальних закладів, виконаний аналіз предметної області;
- проаналізовані вимоги до веб-сайту кафедри, виконаний підбір кольорів і шрифтів для сайту;
- розроблено структуру сайту та виконано проектування макетів сторінок, розроблений дизайн сайту у Figma;
- опанувана CMS WordPress;
- реалізований дизайн сайту на CMS WordPress та розміщений веб-сайт на хостингу.

Список використаної літератури:

1. Content Management System (CMS). WordPress Documentation. URL: https://wordpress.org/documentation/ (дата звернення: 01.11.2023).
2. WordPress: офіційний сайт. URL: https://wordpress.org (дата звернення: 01.11.2023).
3. FTP (File Transfer Protocol). MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Glossary/FTP (дата звернення: 01.11.2023).
4. Ховайба Д. Є. Розробка та реалізація інформаційної системи для закладу харчування. Актуальні проблеми природничих і гуманітарних наук у дослідженнях молодих учених : матеріали XXV Всеукраїнської наукової конференції молодих учених (10–11 травня 2023 р.). Черкаси, 2023. С. 1140–1141.
5. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. Berkeley : New Riders, 2014. 216 p.
6. Структура WordPress: анатомія двигуна. Sebweo. URL: https://sebweo.com/struktura-wordpress/ (дата звернення: 01.11.2023).
7. Figma: офіційна документація. URL: https://help.figma.com (дата звернення: 01.11.2023).

References:

1. WordPress Documentation. (2023). Content management system (CMS). https://wordpress.org/documentation/
2. WordPress. (2023). Official website. https://wordpress.org
3. MDN Web Docs. (2023). FTP (File Transfer Protocol). https://developer.mozilla.org/en-US/docs/Glossary/FTP
4. Khovayba, D. Ye. (2023). Development and implementation of an information system for a catering facility. In Current Problems of Natural and Human Sciences in Research by Young Scientists: Proceedings of the XXV All-Ukrainian Scientific Conference of Young Scientists (May 10–11, 2023, pp. 1140–1141). Cherkasy.
5. Krug, S. (2014). Don't make me think, revisited: A common sense approach to web usability (3rd ed.). New Riders.
6. Sebweo. (2023). WordPress structure: Anatomy of the engine. https://sebweo.com/struktura-wordpress/
7. Figma. (2023). Official documentation. https://help.figma.com

KHOVAYBA Daryna,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIDKOWSKY Ruslan,
Doctor of Technical Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

DESIGN AND IMPLEMENTATION OF AN INFORMATIONAL WEBSITE FOR A DEPARTMENT OF A HIGHER EDUCATION INSTITUTION

Summary. Introduction. *In today's digital world, information technology plays a crucial role in every field, including education. Websites have become essential tools for higher education institutions as they provide an effective means of communication with students, faculty, and other stakeholders. The primary goal of this work is to design and implement a website for a department, which will enhance communication, ensure access to necessary information, and improve interaction with students, faculty, and other interested parties.*

The purpose *of this article is to design and implement an informational website for a department of a higher education institution.*

Results. *The article examines the process of developing an informational website for the Department of Applied Mathematics and Informatics at a higher education institution. It describes the stages involved, including requirement analysis for the site, designing the structure and layout, creating an interactive prototype in Figma, and setting up the CMS platform. The website aims to ensure effective communication between students, faculty, and administration, as well as enhance the institution's image. Key sections of the site include information about the department, resources for students, news, and contact details. The use of modern graphic elements, fonts, and color schemes contributes to the site's attractiveness and functionality. Special attention is given to the choice of CMS for content management, with WordPress selected for its simplicity, functionality, and extensive community support. The final stages include the installation and configuration of WordPress on a web hosting server, completing the development process. The article highlights key aspects of creating an effective web resource for an educational institution.*

Conclusion. *The article presents a study on the design and implementation of an informational website for a department of a higher education institution. Designing and implementing such a website is an important and relevant task that contributes to the improvement of the educational process, communication, and the prestige of the institution. The results of the study can be applied in practical activities for the development and deployment of a department's website using tools such as Figma and WordPress.*

The main outcomes of the work include:

- *a review of existing work and approaches to designing websites for higher education departments, along with an analysis of the subject area;*
- *an analysis of the requirements for the department's website, including the selection of colors and fonts for the site;*
- *the development of the website structure and the design of page layouts, with the website design created in Figma;*
- *proficiency in the CMS WordPress was developed and applied;*
- *the implementation of the website design on CMS WordPress and the deployment of the website on a web hosting server.*

Keywords: *website design, informational website, Figma, WordPress, CMS, educational process, communication.*

Одержано редакцією 14.11.2023 р.
Прийнято до публікації 06.12.2023 р.

УДК 004.932.2:004.85

DOI 10.31651/2076-5886-2023-1-61-68

PACS 07.05.Mh

БИЛИМЕНКО Анна Володимирівна
студентка спеціальності «Прикладна математика» Черкаського національного університету імені Богдана Хмельницького
e-mail: bylymenko.anna1619@vu.cdu.edu.ua

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

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ВИЗНАЧЕННЯ ПОРОДИ СОБАКИ ЗА ДОПОМОГОЮ НЕЙРОННОЇ МЕРЕЖІ

У статті розглядається розробка інтелектуальної системи автоматичного визначення породи собаки на основі згорткових нейронних мереж. Описано теоретичні засади класифікації зображень із застосуванням глибокого навчання: принципи роботи згорткових і пулінгових шарів, функції активації ReLU та Softmax, механізм зворотного поширення помилки. Проведено огляд ключових бібліотек Python, що використовувались у роботі: Pandas – для попередньої обробки даних, TensorFlow – для побудови та навчання нейронної мережі, Streamlit – для розгортання веб-інтерфейсу. Описано практичну реалізацію системи: формування навчальної вибірки з 1888 зображень 21 породи собак, архітектуру послідовної згорткової мережі з чотирма блоками «згортка – підвибірка» та двома повнозв'язними шарами, а також процес навчання протягом 30 епох. За результатами тестування досягнуто точність класифікації 93.64%. Веб-додаток розгорнуто на платформі Streamlit Cloud з використанням репозиторію GitHub. Виконано серії практичних тестувань, що підтвердили працездатність системи для порід, представлених у навчальній вибірці, та прогнозовану поведінку для відсутніх порід. Запропонований підхід може бути поширений на інші задачі класифікації зображень у галузі комп'ютерного зору.

Ключові слова: *згорткова нейронна мережа, машинне навчання, класифікація зображень, розпізнавання порід собак, TensorFlow, Streamlit, глибоке навчання, веб-додаток, штучний інтелект.*

Вступ

Штучні нейронні мережі, зокрема згорткові, зарекомендували себе як потужний інструмент для аналізу зображень. Їхні алгоритми, натхненні біологічними процесами, дозволяють виявляти складні візерунки в даних. Згорткові мережі послідовно обробляють зображення, витягуючи все більш абстрактні ознаки.

Мета статті – теоретично обґрунтувати застосування згорткових нейронних мереж для розпізнавання порід собак.

Виклад основного матеріалу

1. Класифікація зображень за допомогою згорткових нейронних мереж

Штучний інтелект, особливо машинного навчання, досяг значних висот завдяки розвитку згорткових нейронних мереж. Ці мережі, натхненні біологічними процесами, виявилися надзвичайно ефективними для аналізу зображень та інших типів даних.

Однією з ключових особливостей згорткових мереж є їх здатність автоматично виявляти важливі ознаки в даних. Завдяки процесу зворотного поширення помилки,

мережа вчиться налаштовувати свої параметри, щоб краще розпізнавати ці ознаки. Це дозволяє їй виконувати різноманітні завдання, від класифікації зображень до розпізнавання мови.[1].

Важливу роль у згорткових мережах відіграють згорткові шари, які застосовують фільтри для виявлення різних ознак. Пулінгові шари, в свою чергу, зменшують розмірність даних і запобігають перенавчанню. Функція активації ReLU додає нелінійність в мережу, що дозволяє їй моделювати складні залежності.

Архітектура згорткових мереж зазвичай складається з чергування згорткових і пулінгових шарів, за якими можуть йти повнозв'язні шари для класифікації. Вибір розміру фільтрів, кількості шарів та інших параметрів мережі є важливим завданням і впливає на її продуктивність.

Згорткові нейронні мережі знайшли широке застосування в різних галузях, таких як комп'ютерний зір, обробка природної мови, медична діагностика та багато інших. Їхньою перевагою є здатність автоматично витягувати інформативні ознаки з великих обсягів даних.

Ключові переваги згорткових нейронних мереж:

- Автоматичне виявлення ознак.
- Інваріантність до зсувів.
- Ефективність обробки великих обсягів даних.

Застосування:

- Розпізнавання образів (облич, об'єктів)
- Сегментація зображень
- Обробка природної мови
- Медична діагностика
- Автономні автомобілі

Згорткові нейронні мережі є одним з найпотужніших інструментів сучасного машинного навчання і продовжують розвиватися, відкриваючи нові можливості для вирішення складних задач.

2. Задача класифікації

Задача класифікації в контексті машинного навчання полягає в тому, щоб призначити вхідному об’єкту або даним певний "клас" або "мітку" з попередньо визначеного набору можливих класів.

Для виконання задачі класифікації необхідно мати набір прикладів даних, для яких відомі правильні класифікації або мітки. Ці дані використовуються для тренування моделі машинного навчання, яка вивчає зв'язки між вхідними ознаками та відповідними класами.

Після тренування моделі вона може бути застосована до нових, раніше не використованих даних для прогнозування класу, до якого вони належать. Задача класифікації має широке застосування в багатьох галузях, включаючи комп’ютерне зорове сприйняття, медицину, фінанси, рекламу, безпеку, та багато інших областей, де необхідно приймати рішення на основі вхідних даних. [2].

3. Огляд бібліотек для реалізації нейронних мереж і для створення веб-додатків та допоміжних інструментів

На сьогоднішній день, завдяки популярності нейронних мереж на мові Python існує велика кількість бібліотек, які надають можливість конструювання різноманітних нейронних мереж на досить різних рівнях абстракції. Для огляду було обрано три бібліотеки: Pandas, Tensorflow та Streamlit.

Pandas – це високорівнева Python бібліотека для аналізу даних. Вона побудована поверх більш низькорівневої бібліотеки NumPy, що додає їй продуктивності. Ця бібліотека є одною з найбільш просунутих, що швидко розвивається, для обробки та аналізу даних. Pandas – одна із дуже зручних бібліотек всередині мови Python, які допомагають працювати з великою кількістю табличних даних (досить часто тренувальні і тестувальні вибірки виглядають, як .csv-таблиці мільйонами рядків і колонок параметрів) є бібліотека Pandas. Вона дозволяє дуже швидко завантажувати дані, препроцесити їх (готувати у відповідний формат), щоб у зручному вигляді відправляти на опрацювання алгоритмом, який, наприклад, вибрали з бібліотеки scikit-learn.

Бібліотека Pandas пропонує дві основні структури даних: DataFrame і Series. DataFrame – це двовимірна структура даних у вигляді таблиці, яка складається зі стовпців із позначеними осями для роботи з табличними даними. Крім того, Series є одновимірним позначеним масивом об'єкта, який містить значення з позначеними осями.

Бібліотека Pandas може бути корисною в реальних додатках, включаючи фінансове моделювання, наукові обчислення, аналіз даних, візуалізацію даних, та машинне навчання.

Бібліотека Pandas надає розробникам простий у використанні інструмент аналізу даних для виконання маніпуляцій з даними, їх фільтрації та обробки в Python. Здатність бібліотеки обробляти величезну кількість даних, повна документація, та велика кількість функцій, роблять її потужним інструментом для аналізу даних.

Дана бібліотека – це opensource бібліотека, тобто вихідний код є у відкритому доступі та розміщений на GitHub. Користувачі можуть додавати туди свій код: вносити зміни, доповнювати методи власним кодом, оновлювати розділи, тощо.

TensorFlow – це бібліотека з відкритим кодом, яка дозволяє розробникам і дослідникам створювати та розгортати різні моделі машинного навчання та системи глибокого навчання. Вона розроблена Google і є однією з найпопулярніших бібліотек, що використовуються в програмах машинного навчання та штучного інтелекту.

Однією з ключових особливостей TensorFlow є здатність обробляти великі обсяги даних. Бібліотека підтримує розподілене обчислення, дозволяючи обробляти великі набори даних на кількох машинах, що скорочує час, необхідний для навчання моделей. Крім того, TensorFlow має широкий спектр готових моделей, наприклад як модель розпізнавання зображень і мови.

Іншою важливою особливістю TensorFlow є те, що вона підтримує кілька платформ, включаючи мобільні та вбудовані пристрої, а також пропонує підтримку різних мов програмування, включаючи Python, C++ і Java. Ця бібліотека також надає широкий набір інструментів для розгортання моделей у різних середовищах.

На відміну від інших бібліотек DL, які в основному орієнтовані на дослідження (наприклад, Theano), TensorFlow був розроблений для використання як у системах досліджень, так і у розробці та виробництві програмного забезпечення.

Streamlit − це бібліотека Python з відкритим вихідним кодом, яка дозволяє легко створювати та ділитися чудовими спеціальними веб-програмами для машинного навчання та обробки даних. Лише за кілька хвилин ви можете створити та розгорнути потужну програму для обробки даних. [3].

Три найважливіші функції, які надає Streamlit:

1. Використання сценаріїв: можна створити інтерактивну веб-програму за допомогою лише кількох рядків коду, і якщо внести будь-які зміни до свого коду, то побачите автоматичні оновлення у веб-додатку. Це відбувається з API, що надається цим фреймворком.

2. Взаємодія: додати віджети до вашого веб-додатку так само просто, як оголосити змінні в Python.
3. Миттєве розгортання: платформа Streamlit допомагає легко розгортати програми та керувати ними. [4].

Крім того, Streamlit можна використовувати з популярними бібліотеками Python, такими як NumPy, Pandas і Matplotlib, що робить її універсальним інструментом для проектів з обробки даних.

Streamlit також може легко обробляти великі набори даних і складні обчислення. Її можна легко розгорнути на різних хмарних платформах, включаючи Amazon Web Services (AWS), Google Cloud Platform (GCP) і Microsoft Azure. Streamlit – це чудовий інструмент для розробників, яким потрібен стислий, спрощений та ефективний спосіб створення веб-додатків для аналізу даних, машинного навчання та візуалізації даних.

Основна мета Streamlit – це спростити процес розробки веб-додатків і усунути складність багатьох мов, фреймворків, тощо. За допомогою Streamlit розробники можуть створювати власні веб-інтерфейси зі свого коду Python без використання HTML, CSS або JavaScript.

Однією з важливих особливостей Streamlit є його механізм кешування, який дозволяє користувачам кешувати обчислення та використовувати їх належним чином. Ще однією значною перевагою використання Streamlit є його універсальність у створенні різних типів веб-додатків, керованих даними.

Streamlit є здатність перетворювати статичні блокноти Jupyter в інтерактивні веб-додатки. Ноутбуки Jupyter широко використовуються в машинному навчанні для відтворення коду, спільного використання, та співпраці. Однак Jupyter Notebooks може бути важко розгорнути та поділитися з нетехнічними зацікавленими сторонами. Streamlit вирішує цю проблему, дозволяючи конвертувати Jupyter Notebooks в інтерактивні веб-програми за допомогою кількох простих команд.

Streamlit працює на всіх основних операційних системах, таких як Windows, macOS і Linux, і підтримує Python 3.6 і вище.

Підсумовуючи, Streamlit забезпечує універсальний спосіб створення веб-додатків з мінімальними зусиллями та чудовими варіантами розгортання. Саме бібліотеку Streamlit було обрано для створення веб-додатку.

4. Практична реалізація системи розпізнавання порід собак

4.1. Підготовка набору даних

Першим етапом практичної реалізації є формування навчальної вибірки. Для автоматизованого збору зображень використано бібліотеку Python `bing-image-downloader`, яка здійснює асинхронне завантаження зображень із пошукового сервісу Bing за заданим запитом. Команда запуску має вигляд:

```
python test.py <назва_породи>
```

Для кожної з 21 породи собак було завантажено близько 100 зображень. Загальний обсяг сформованого набору даних склав 1888 файлів. Після завантаження виконано ручну фільтрацію нерелевантних зображень та уніфікацію форматів файлів до вимог Google Colaboratory (JPEG, PNG, GIF, BMP). Підготовлений набір даних завантажено на Google Диск для подальшого використання у середовищі Colaboratory. Підключення Google Диску виконується наступним кодом:

```
from google.colab import drive
drive.mount('/content/drive')
```

Набір даних розподілено на навчальну (90%) та перевірочну (10%) вибірки за допомогою функції `image_dataset_from_directory` з параметром `validation_split=0.1`. Розмір вхідних зображень уніфіковано до 100×100 пікселів.

4.2. Архітектура та навчання нейронної мережі

Для класифікації порід собак побудовано згорткову нейронну мережу послідовної архітектури (Sequential). Мережа складається з чотирьох блоків «згортковий шар – шар підвибірки» з фільтрами розмірністю 5×5 та кількістю карт ознак 16, 32, 64 і 128 відповідно, після яких розташовано два повнозв'язні шари з 1024 та 256 нейронами. У всіх згорткових та повнозв'язних шарах використано функцію активації ReLU; після кожного повнозв'язного шару застосовано Dropout з коефіцієнтом 0.2 для запобігання перенавчанню. Вихідний шар містить 21 нейрон (за кількістю класів) з функцією активації `Softmax`. Зведена архітектура мережі подана в табл. 1.

Таблиця 1

Архітектура згорткової нейронної мережі

Шар	Тип	Параметри
1	Conv2D	16 фільтрів, 5×5, ReLU
2	MaxPooling2D	2×2
3	Conv2D	32 фільтри, 5×5, ReLU
4	MaxPooling2D	2×2
5	Conv2D	64 фільтри, 5×5, ReLU
6	MaxPooling2D	2×2
7	Conv2D	128 фільтрів, 5×5, ReLU
8	MaxPooling2D	2×2
9	Dense + Dropout	1024 нейрони, p=0.2
10	Dense + Dropout	256 нейронів, p=0.2
11	Dense (вихідний)	21 нейрон, Softmax

Навчання виконувалось протягом 30 епох з використанням навчальної та перевірочної вибірок. Оцінку якості навченої моделі здійснено на тестовому наборі за допомогою методу `model.evaluate`. За результатами тестування точність класифікації склала 93.64%, що підтверджує прийнятну якість навченої моделі для практичного застосування (рис. 1). Динаміка частки правильних відповідей на навчальній та перевірочній вибірках упродовж 30 епох навчання свідчить про стійке зростання точності моделі без ознак суттєвого перенавчання (рис. 2).

Навчену модель збережено у форматі HDF5:

```
model.save("dog_breeds.h5")
```

4.3. Розгортання веб-додатку

Для розгортання веб-інтерфейсу використано бібліотеку Streamlit. Основний файл застосунку `web-for-classification.py` реалізує наступну логіку: завантаження збереженої моделі; попередню обробку завантаженого користувачем зображення (зміна розміру до 100×100 пікселів, нормалізація); передачу оброблених даних до моделі та виведення п'яти найбільш імовірних порід із зазначенням ймовірності кожної з них. Ключовий фрагмент інтерфейсу виглядає наступним чином:

```
st.title('Класифікація зображень')
img = load_image()
```

```
result = st.button('Розпізнати зображення')
if result:
    x = preprocess_image(img)
    preds = model.predict(x)
    print_predictions(preds)
```

```
[ ]  # Оцінюємо якість навчання моделі на тестових даних
     scores = model.evaluate(test_dataset, verbose=1)

     8/8 [==============================] - 17s 30ms/step - loss: 0.3898 - accuracy: 0.9364

[ ]  print("Частка правильних відповідей на тестових даних, у відсотках:", round(scores[1] * 100, 4))

     Частка правильних відповідей на тестових даних, у відсотках: 93.6441
```

Рис. 1. Оцінка якості отриманих результатів

```
plt.plot(history.history['accuracy'],
         label='Частка правильних відповідей на навчальному наборі')
plt.plot(history.history['val_accuracy'],
         label='Частка правильних відповідей на перевірочному наборі')
plt.xlabel('Епоха навчання')
plt.ylabel('Частка правильних відповідей')
plt.legend()
plt.show()
```

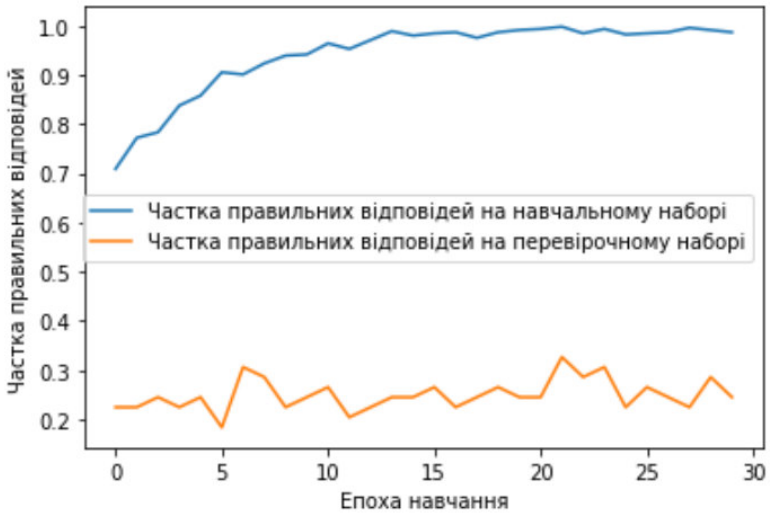

Рис. 2. Динаміка зміни частки правильних відповідей

Публікацію застосунку виконано через платформу Streamlit Cloud з використанням репозиторію GitHub. До репозиторію завантажено файли: `web-for-classification.py` (код застосунку), `dog_breeds.h5` (навчена модель) та `requirements.txt` (перелік залежностей). Розгортання здійснюється через вебінтерфейс Streamlit Cloud після підключення облікового запису GitHub та вказівки шляху до основного файлу. Зовнішній вигляд розгорнутого веб-додатку наведено на рис. 3.

Для перевірки практичної ефективності системи було виконано три серії тестувань: розпізнавання зображень із навчального набору (точність наближається до 100%), розпізнавання нових зображень порід, що входять до набору даних (хаскі розпізнано з імовірністю 94.08%; джек-расел-тер'єр – з імовірністю 99.98%), та розпізнавання породи, відсутньої в наборі даних (бультер'єр). В останньому випадку модель розподілила ймовірність між візуально схожими породами (бульдог – 41.33%, такса – 40.67%), що є прогнозованою поведінкою системи з фіксованим набором класів.

Висновки

Згорткові нейронні мережі є ефективним інструментом для розпізнавання порід собак. Завдяки здатності автоматично виявляти характерні риси об’єктів на зображеннях, ці мережі досягають високої точності класифікації. Для реалізації такого розпізнавання було використано сучасні бібліотеки Python, такі як TensorFlow та Streamlit, які забезпечили зручність розробки та гнучкість у налаштуванні моделей. Запропонований підхід може бути застосований для створення інших веб-додатків, що базуються на технологіях глибокого навчання, та відкриває широкі перспективи для розвитку інноваційних рішень.

Список використаної літератури:

1. Згорткова нейронна мережа. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа (дата звернення: 01.10.2023).
2. Patterson J., Gibson A. Deep Learning: A Practitioner's Approach. Sebastopol : O'Reilly Media, 2017. 520 p.
3. Streamlit Documentation. URL: https://docs.streamlit.io/ (дата звернення: 01.10.2023).
4. LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998. Vol. 86, No. 11. P. 2278–2324. DOI: 10.1109/5.726791.
5. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems. 2012. Vol. 25. P. 1097-1105.
6. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556. 2014. URL: https://arxiv.org/abs/1409.1556.
7. Khosla A., Jayadevaprakash N., Yao B., Fei-Fei L. Novel Dataset for Fine-Grained Image Categorization. Proceedings of the 1st Workshop on Fine-Grained Visual Categorization (FGVC), CVPR. Colorado Springs, 2011.
8. Parkhi O. M., Vedaldi A., Zisserman A., Jawahar C. V. Cats and Dogs. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Providence, 2012. P. 3498-3505. DOI: 10.1109/CVPR.2012.6248092.

References:

1. Convolutional neural network. Wikipedia. URL: https://en.wikipedia.org/wiki/Convolutional_neural_network (accessed: 01.10.2023).
2. Patterson J., Gibson A. Deep Learning: A Practitioner's Approach. Sebastopol: O'Reilly Media, 2017. 520 p.
3. Streamlit Documentation. URL: https://docs.streamlit.io/ (accessed: 01.10.2023).
4. LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998. Vol. 86, No. 11. P. 2278–2324. DOI: 10.1109/5.726791.
5. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems. 2012. Vol. 25. P. 1097-1105.
6. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556. 2014. URL: https://arxiv.org/abs/1409.1556.
7. Khosla A., Jayadevaprakash N., Yao B., Fei-Fei L. Novel Dataset for Fine-Grained Image Categorization. Proceedings of the 1st Workshop on Fine-Grained Visual Categorization (FGVC), CVPR. Colorado Springs, 2011.
8. Parkhi O. M., Vedaldi A., Zisserman A., Jawahar C. V. Cats and Dogs. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Providence, 2012. P. 3498–3505. DOI: 10.1109/CVPR.2012.6248092.

BYLYMENKO Anna,
Student, Department of Informatics and Applied Mathematics, 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

AN INTELLIGENT SYSTEM FOR DOG BREED RECOGNITION USING CONVOLUTIONAL NEURAL NETWORKS

Summary. Introduction. *Convolutional neural networks (CNNs) have established themselves as a powerful tool for image analysis. Inspired by biological visual processing, they automatically detect complex patterns in data by sequentially extracting increasingly abstract features. The growing demand for AI-powered interactive applications makes automated image recognition systems particularly relevant across a wide range of domains, including veterinary practice, animal shelters, and mobile applications.*

The Purpose *of the article is to theoretically justify and practically demonstrate the use of convolutional neural networks for dog breed identification, and to describe the full development cycle of a CNN-based web application – from dataset preparation to deployment.*

Results. *The article presents a review of image classification methods based on deep learning, with a focus on the architecture and training principles of CNNs. The roles of convolutional layers, pooling layers, the ReLU activation function, and the backpropagation algorithm are discussed. Three Python libraries central to the project are reviewed: Pandas for data preprocessing, TensorFlow for building and training the neural network model, and Streamlit for rapid development and deployment of the web interface.*

The practical implementation covers all stages of system development. A dataset of 1,888 images across 21 dog breeds was collected using the bing-image-downloader library and manually filtered. The images were split into training (90%) and validation (10%) sets. The CNN architecture follows a sequential structure consisting of four convolutional-pooling blocks with 16, 32, 64, and 128 filters respectively, followed by two fully connected layers (1,024 and 256 neurons) with Dropout regularization ($p = 0.2$), and a Softmax output layer with 21 neurons. The model was trained for 30 epochs and achieved a classification accuracy of 93.64% on the test set. Practical testing confirmed high accuracy for breeds present in the training data (e.g., Husky – 94.08%, Jack Russell Terrier – 99.98%) and predictable probabilistic distribution for breeds outside the dataset.

The web application was deployed via Streamlit Cloud integrated with a GitHub repository, providing a user-friendly interface for uploading images and displaying the top five predicted breeds with corresponding probabilities.

Conclusion. *Convolutional neural networks are an effective tool for dog breed classification tasks. The combination of TensorFlow for model development and Streamlit for interface deployment provides an efficient and accessible pipeline for building machine learning–based web applications. The proposed approach can be readily extended to other image recognition tasks and illustrates the practical potential of deep learning in real-world automated recognition systems.*

Keywords: *convolutional neural network, machine learning, image classification, dog breed recognition, TensorFlow, Streamlit, deep learning, web application, artificial intelligence.*

Одержано редакцією 03.11.2023 р.
Прийнято до публікації 06.12.2023 р.

УДК 004.738.5:339.1

DOI 10.31651/2076-5886-2023-1-69-78

PACS 89.20.Hh

БРАГІНЕЦЬ Ростислав Леонідович
студент спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького
e-mail: rostyslav.brahinets@gmail.com

ДІДКОВСЬКИЙ Руслан Михайлович
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ ОРГАНІЗАЦІЇ ОНЛАЙН ПОКУПОК ТА ПРОДАЖУ ТОВАРІВ

Розглядається процес розробки веб-сервісу для організації онлайн-покупок та продажу товарів. Проведено аналіз існуючих аналогічних платформ — «Rozetka» та «Prom» — і визначено функціональні вимоги до сервісу, що розробляється. Обґрунтовано вибір стеку технологій: Java та Spring Boot для серверної частини, TypeScript і Angular — для клієнтської, PostgreSQL — як система управління базами даних, Flyway — для міграцій схеми БД, Stripe — для обробки платіжних транзакцій. Описано архітектуру RESTful API, що забезпечує взаємодію між клієнтом і сервером, наведено ключові фрагменти коду серверної та клієнтської частин. Реалізовано чотири режими роботи сервісу: гість, авторизований покупець, адміністратор та супер-адміністратор. Функціонал включає реєстрацію й авторизацію користувачів, перегляд і сортування товарів за категоріями, кошик покупок, оплату банківською карткою з автоматичним генеруванням звіту про покупку, а також адміністративний інтерфейс для управління товарами, категоріями та обліковими записами. Наведено демонстрацію роботи сервісу з відповідними ілюстраціями.

Ключові слова: *онлайн-покупки, веб-сервіс, інтернет-магазин, PostgreSQL, Spring Boot, Angular, TypeScript, REST API, Stripe, електронна комерція.*

Вступ

В сучасному світі люди постійно щось купують. Це може бути щось життєво необхідним, а може бути просто заради розваги. Розвиток сучасних технологій постійно спрощує життя людей. Навіть така річ, як покупка чогось, сьогодні може відбуватися дуже швидко та зручно. Для цього не потрібно виходити з дому. Достатньо лише мати доступ до Інтернету. Онлайн-покупки є дуже популярними в сучасному світі, а ще більшої актуальності, навіть необхідності, вони набули через пандемію COVID-19 та збройний конфлікт в Україні.

Мета статті полягає у тому, щоб показати процес розробки веб-сервісу для організації онлайн покупок та продажу товарів. Для досягнення поставленої мети потрібно виконати наступні завдання:

1) розглянути існуючі сервіси для продажу та покупок товарів;
2) дослідити технології для роботи з базою даних;
3) визначити технологічний стек для серверної частини сервісу;
4) розглянути технології для клієнтської частини сервісу;

5) описати та продемонструвати процес розробки й функціонал веб-сервісу.

Виклад основного матеріалу

1. Використані технології та мови програмування

Розглянемо існуючі сервіси для продажу та покупок у мережі Інтернет. Зокрема візьмемо такі інтернет-магазини, як «Rozetka» та «Prom».

«Rozetka» та «Prom» є двома популярними онлайн-магазинами в Україні, які надають платформу для покупки та продажу товарів через Інтернет. Обидва магазини мають великий вибір товарів і надійну репутацію серед споживачів.

«Rozetka» є одним з найбільших онлайн-магазинів в Україні. Він пропонує широкий асортимент товарів, включаючи електроніку, побутову техніку, одяг, косметику та інше. Клієнти можуть знайти продукти різних брендів і виробників на платформі «Rozetka». Магазин також відомий своєю швидкою доставкою та якісним обслуговуванням клієнтів.

«Prom» є іншим популярним онлайн-магазином в Україні. Ця платформа також пропонує широкий асортимент товарів від різних продавців. «Prom» орієнтований на бізнес-клієнтів та підприємства, а також на приватних споживачів. Він надає можливості для оптової торгівлі та бізнесу, включаючи спеціальні умови для великих замовлень.

Як і багато інших онлайн-магазинів, обидва цих магазини забезпечують зручність покупок через Інтернет, широкий вибір товарів, можливість порівняння цін та відгуків клієнтів, а також швидку та надійну доставку. Крім того, вони мають розвинені системи оплати і забезпечують безпеку та конфіденційність покупців.

Онлайн-магазини, які мають широкий асортимент товарів, такі як «Rozetka» та «Prom», зазвичай мають структуровану інформаційну архітектуру. Вони організовують свої товари в категорії та підкатегорії для полегшення навігації користувачів. На головній сторінці зазвичай розміщено акційні пропозиції, найбільш популярні товари або нові надходження. Крім того, обидва магазини мають систему пошуку, яка дозволяє користувачам швидко знаходити потрібні товари.

«Rozetka» та «Prom» надають широкий спектр функціональних можливостей для зручності покупця. Це можуть бути фільтри для пошуку, порівняння товарів, рейтинги та відгуки клієнтів, які допомагають зробити обґрунтований вибір. Також, можуть бути надані інструменти для створення облікових записів, збереження списків бажань, отримання сповіщень про акції та знижки.

Обидва магазини звичайно мають привабливий та інтуїтивно зрозумілий інтерфейс користувача. Це означає, що вони намагаються забезпечити легку навігацію, зрозумілі піктограми та кнопки, чітке розміщення товарів та інші деталі, щоб забезпечити зручне та ефективне користування сайтом.

Онлайн-магазини використовують різні технології для своєї розробки та функціонування.

Для розробки клієнтської частини сайту (фронтенду) можуть використовуватись такі технології, як HTML, CSS та JavaScript. Фреймворки та бібліотеки, такі як Angular, React або Vue.js, можуть бути використані для побудови більш складних інтерфейсів користувача.

Для обробки бізнес-логіки та управління даними використовуються різні технології. Наприклад, популярні мови програмування, такі як Java, C#, Python або PHP, можуть бути використані для написання серверної частини. Використання фреймворків, таких як Spring (для Java) або Django (для Python), дозволяє прискорити розробку та забезпечити кращу структуру проекту.

Для зберігання та керування даними магазинів можуть використовуватись реляційні бази даних, наприклад, MySQL або PostgreSQL. Також можуть використовуватись нереляційні (NoSQL) бази даних, такі як MongoDB або Redis, для певних аспектів, наприклад, кешування або швидкодії.

Деякі магазини можуть використовувати спеціалізовані CMS для управління контентом на сайті. Популярні CMS, такі як WordPress або Drupal, можуть бути налаштовані для реалізації функціональності магазину.

Для прийому платежів та інтеграції з платіжними системами магазини можуть використовувати API платіжних провайдерів, таких як PayPal, Stripe або LiqPay.

Під час розробки власного веб-сервісу було поставлено наступні завдання:

1. Розробити меню для реєстрації.
2. Розробити меню для авторизації.
3. Розробити головні меню для неавторизованого гостя, авторизованого покупця, адміністратора та супер-адміністратора.
4. Розробити меню з категоріями товарів для всіх користувачів.
5. Розробити меню з товарами для всіх користувачів.
6. Розробити меню профілю та його редагування для авторизованих користувачів.
7. Додати кошик для авторизованого покупця.
8. Додати оплату товарів для авторизованого покупця із генеруванням звіту про покупку.
9. Розробити меню для додавання та видалення унікальних номерів адміністратора.
10. Розробити меню для перегляду та видалення зареєстрованих користувачів.
11. Розробити меню для додавання, редагування та видалення категорій товарів.
12. Розробити меню для додавання, редагування та видалення товарів.

Для розробки власного сервісу було використано Java та Spring для серверної частини, TypeScript та Angular для розробки клієнтської частини і базу даних PostgreSQL. Для обробки платежів обрано Stripe.

Stripe – це платіжна платформа, яка надає розробникам та бізнесам інструменти для прийому онлайн-платежів. Вона дозволяє підприємствам приймати платежі через Інтернет і забезпечує безпечну та зручну інтеграцію платіжних функцій у веб-сайти та додатки.

Stripe дозволяє приймати різні види платежів, включаючи кредитні та дебетові картки (Visa, Mastercard, American Express), цифрові гаманці (Apple Pay, Google Pay), банківські перекази та інші способи платежів.

Розробники можуть використовувати Stripe для інтеграції платіжних функцій безпосередньо у свої веб-сайти або мобільні додатки. Stripe надає зручне API та набір готових бібліотек для різних мов програмування, що спрощує процес інтеграції.

TypeScript є мовою програмування, розробленою компанією Microsoft в 2012 році [11]. Вона позиціонується як розширення мови JavaScript і призначена для розробки веб-застосунків.

Gradle є системою автоматичного збирання, яка поєднує принципи Apache Ant та Apache Maven. Вона використовує предметно-орієнтовану мову (DSL) на основі мови Groovy для налаштування та опису залежностей проекту. Головна мета Gradle – забезпечити зручне підключення необхідних бібліотек та залежностей.

У випадку з серверною стороною веб-сервісу, Gradle використовується для підключення необхідних бібліотек. Код, який визначає залежності та налаштування проекту, міститься у файлі build.gradle.

Для запуску серверної сторони веб-сервісу використовується фреймворк Spring Boot. У класі ApplicationStarter, в методі main(), ініціалізується серверна частина, а саме

локальний сервер Tomcat, який дозволяє працювати з веб-сервісом без підключення до Інтернету.

REST (Representational State Transfer) – це підхід до архітектури мережевих протоколів, який надає доступ до інформаційних ресурсів. Він був описаний та популяризований Роєм Філдінгом у 2000 році, одним з творців протоколу HTTP. REST базується на принципах функціонування Всесвітньої павутини та можливостях протоколу HTTP. Філдінг розробив REST одночасно з HTTP 1.1, ґрунтуючись на попередньому протоколі HTTP 1.0.

Комунікація між клієнтом та сервером у RESTful API є безстановною, що означає, що кожен запит клієнта до сервера містить усю необхідну інформацію для обробки цього запиту. Сервер не зберігає жодної інформації про стан клієнта між запитами.

2. Реалізація сервісу

Для створення сервісу необхідно створити сервер, який буде працювати з нашою базою даних. Додавати до неї інформацію, видаляти та редагувати її, а також просто переглядати. І потрібно створити інтерфейс клієнтської сторони. Він має відображати інформацію про товари, давати можливість взаємодіяти з цими товарами та мати всі необхідні функції.

Серверна частина побудована за трирівневою REST-архітектурою: контролери обробляють HTTP-запити, сервіси реалізують бізнес-логіку, репозиторії відповідають за доступ до бази даних PostgreSQL. Нижче наведено ключові фрагменти реалізації.

Розглянемо кілька частин всього функціоналу сервісу. Зокрема для серверної сторони існує метод update(), у класі ProductRepository, який оновлює дані певного продукту. Він приймає ідентифікаційний номер товару, який потрібно оновити, і об’єкт з оновленими даними. Далі за допомогою Spring Data JDBC дані про товар оновлюються в базі даних. Для цього використовується SQL-скрипт. Код цього методу наведено нижче.

```
@Override
public void update(long id, Product product) {
    jdbcTemplate.update(
        "UPDATE product "
            + "SET name=:name, describe=:describe, price=:price, "
            + "barcode=:barcode, in_stock=:in_stock, image=:image "
            + "WHERE id=:id",
        Map.ofEntries(
            Map.entry("name", product.getName()),
            Map.entry("describe", product.getDescribe()),
            Map.entry("price", product.getPrice()),
            Map.entry("barcode", product.getBarcode()),
            Map.entry("in_stock", product.isInStock()),
            Map.entry("image", product.getImage()),
            Map.entry("id", id)
        )
    );
}
```

Метод saveProduct(), у класі ProductController, додає новий товар до бази даних. Він приймає об’єкт нового товару. В методі перевіряється чи отриманий об’єкт містить зображення товару. Якщо його немає, то встановлюється стандартне зображення. І вже потім новий товар додається до бази даних.

Для категорій існує клас-валідатор CategoryValidator. Він має кілька методів validate() для перевірки категорії. Перший приймає в якості параметрів

ідентифікаційний номер категорії та список всіх категорій, які існують. Потім створюється порожній список для ідентифікаційний номерів. Які за допомогою циклу отримуються від кожної категорії і записуються у цей список. Далі перевіряється чи місить цей список ідентифікаційний номер, отриманий методом у параметрах. Якщо він існує, то нічого не відбувається. Але якщо такого номера немає, то виникає помилка.

Інший метод validate() приймає назву категорії та список всіх існуючих категорій. Метод перевіряє чи назва не порожня. І викидає помилку, якщо це не так. Далі перевіряється чи існує вже така назва категорії, так само, як і з ідентифікаційним номером. Якщо її не існує, то виникає помилка.

Кожен користувач має якусь роль: клієнт або адміністратор. Є сервіс RoleService в якому є метод findByName(), який повертає об'єкт ролі за назвою. Він приймає назву ролі, які треба знайти і звертається до репозиторія RoleRepository, щоб отримати роль. Код класу наведено нижче.

```
package com.shop.role;
import org.springframework.stereotype.Service;
import java.util.Optional;
@Service
public class RoleService {
    private final RoleRepository roleRepository;
    public RoleService(RoleRepository roleRepository) {
        this.roleRepository = roleRepository;
    }
    public Role findByName(String name) {
        Optional<Role> role = roleRepository.findByName(name);
        return role.orElseGet(Role::new);
    }
}
```

Розглянемо тепер функціонал клієнтської частини. Для сторінки авторизації є розмітка на HTML. Вона формує на веб-сторінці потрібні елементи. А за допомогою стилів CSS відображає їх у красивому для користувача вигляді. На сторінці авторизації є назва сторінки, форма з полями для введення електронної адреси або номера телефону і пароля. Також є кнопки для підтвердження авторизації та переходу на сторінку реєстрації.

На всіх веб-сторінках є кнопки. Для них існує компонент «shop-button». Його можна помістити на сторінку за допомогою тега в HTML <shop-button>. Цей тег заміняється розміткою, яка наведена нижче.

```
<button
  class="btn"
  type="button"
  [ngStyle]="{'background-color': color}"
  (click)="onClick()">
  {{text}}
</button>
```

Для такої заміни існує клас «ButtonComponent». Він вказує, яку розмітку та стилі використовувати для компонента «shop-button». Компонент має кілька властивостей, які потрібно задавати під час його використання. Потрібно вказувати текст, який має відображатися на кнопці і колір самої кнопки. Також кнопка має обробник подій, який

спрацьовує під час натискання на неї. А що саме має зробити кнопка потрібно теж вказати під час використання компонента. Код цього класу наведено нижче.

```
import {Component, EventEmitter, Input, OnInit, Output} from '@angular/core';
@Component({
  selector: 'shop-button',
  templateUrl: './button.component.html',
  styleUrls: ['./button.component.css']
})
export class ButtonComponent implements OnInit {
  @Input() text: string;
  @Input() color: string;
  @Output() btnClick;
  constructor() {
    this.btnClick = new EventEmitter();
  }
  ngOnInit(): void {
  }
  onClick(): void {
    this.btnClick.emit();
  }
}
```

Для того щоб звертатися до сервера та працювати з користувачами існує сервіс UserService. Він використовує HttpClient щоб посилали запити на сервер. Цей сервіс дозволяє отримувати всіх користувачів, певного користувача по його ідентифікаційному номеру, оновлювати дані користувача та видаляти його.

3. Використання сервісу

Ознайомимося з розробленим сервісом та розглянемо його можливості. Сервіс має чотири режими роботи: гість, користувач, адміністратор та супер-адміністратор. Для кожного відображаються різні головні сторінки. На них відображаються випадкові товари та меню, відповідне для кожного режиму (рис. 1).

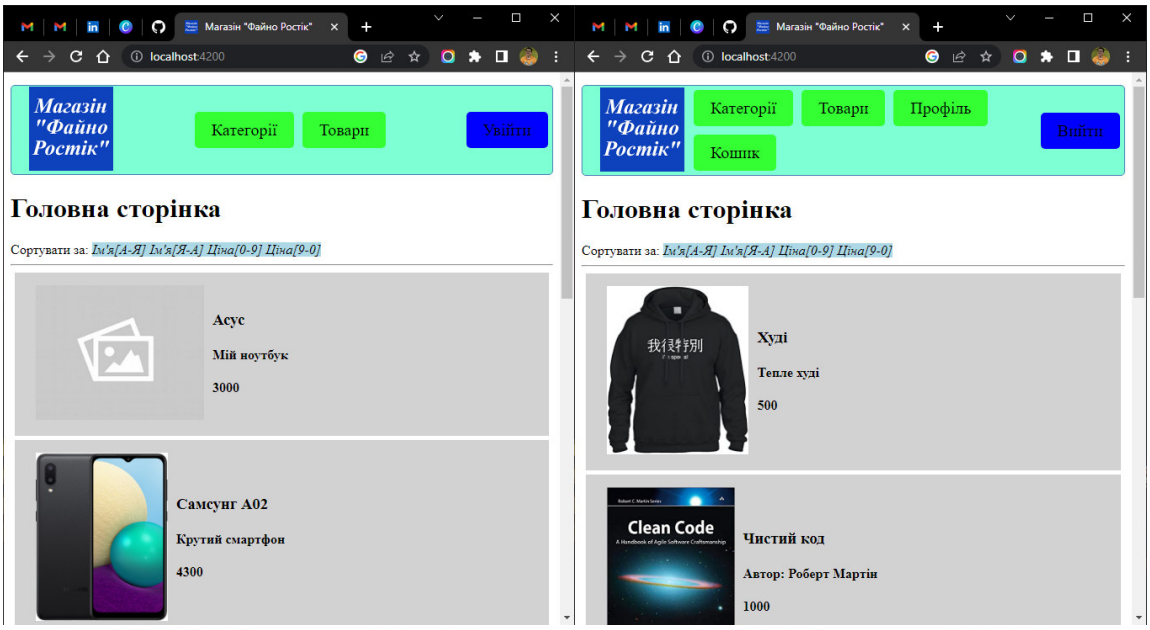

Рис. 1. Головні сторінки для гостя та авторизованого покупця

На веб-сторінці авторизації є поля для введення електронної адреси або номера телефону та пароля і кнопка підтвердження (рис. 2). На веб-сторінці реєстрації є поля для введення прізвища та ім’я, електронної адреси, номера телефону, пароля та підтвердження пароля, номеру адміністратора, кнопка підтвердження і кнопка переходу на веб-сторінку авторизації.

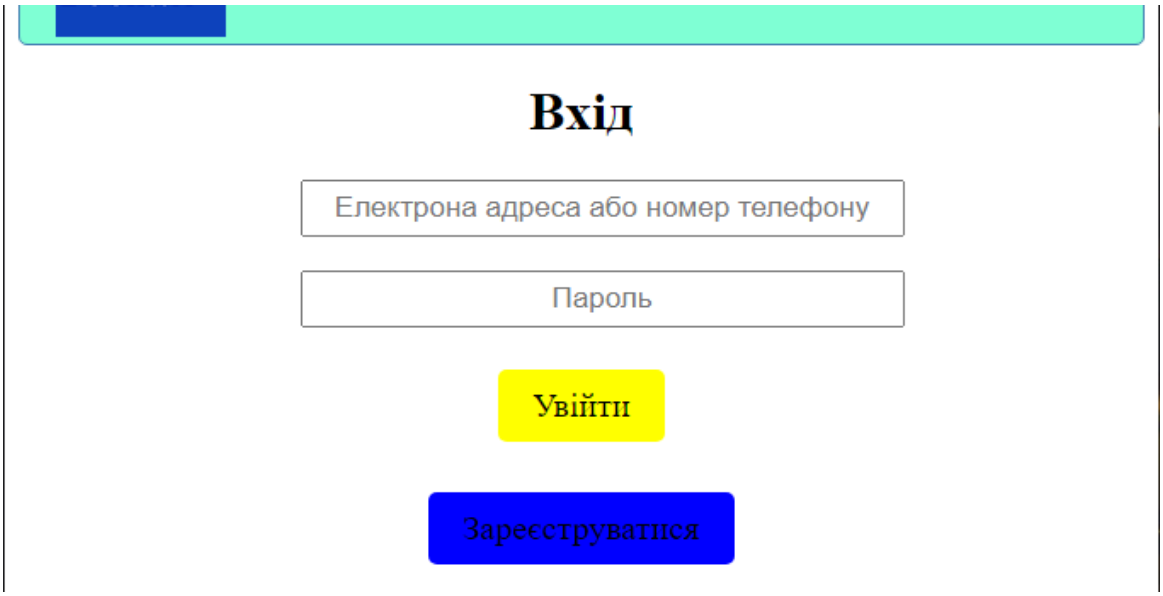

Рис. 2. Сторінка авторизації

На веб-сторінці «Товари» є кнопки переходу на інші веб-сторінки та кілька випадкових товарів. Також товари можна сортувати за назвою та ціною у порядку зростання або спадання (рис. 3). Якщо користувач не авторизований, то на кожній сторінці є кнопка входу. А якщо авторизований, то кнопка виходу. Товари можуть бути присутні та відсутні на складі. Про це вказується на веб-сторінці з товаром. Якщо товар відсутній на складі, то купити його неможливо. Кожен товар належить до якоїсь категорії. На веб-сторінці «Категорії» є кнопки переходу на інші веб-сторінки та список усіх категорій товарів.

Рис. 3. Сторінка «Товари»

Адміністратор може додавати нові товари та категорії. Також адміністратор може видаляти існуючі товари та категорії. Ще адміністратор може редагувати існуючі товари та категорії. Додати товар можна із зображенням або без. Якщо його немає, то встановлюється стандартне зображення автоматично.

На веб-сторінці «Профіль» є посилання для переходу на інші веб-сторінки та інформація про користувача, яку було введено під час реєстрації. Користувач може змінювати інформацію про себе.

Кожен покупець має кошик, куди додаються товари, які він хоче купити (рис. 4). Після додавання товарів до кошика їх можна купити. Після натискання на кнопку «Оплатити» з'являється форма для введення реквізитів банківської карти й кнопка «Оплатити». Після оплати пропонується завантажити звіт про покупку (рис. 5).

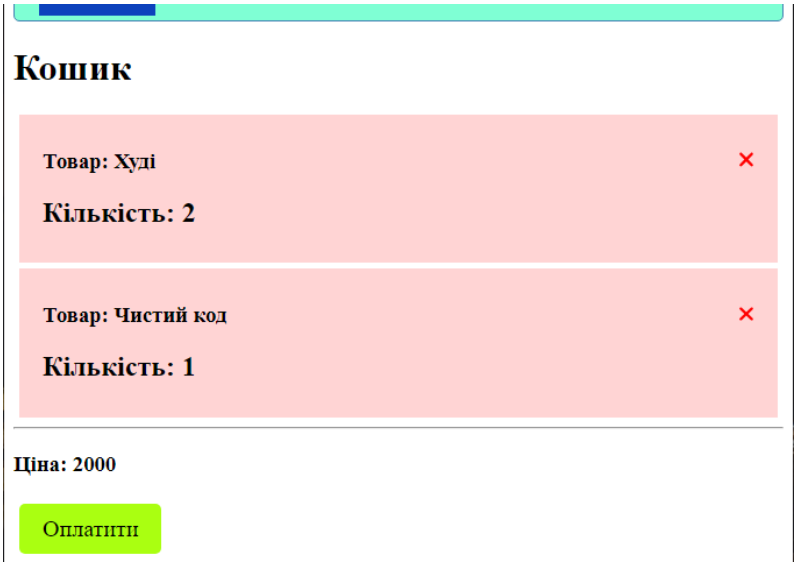

Рис. 4. Сторінка кошика з товарами

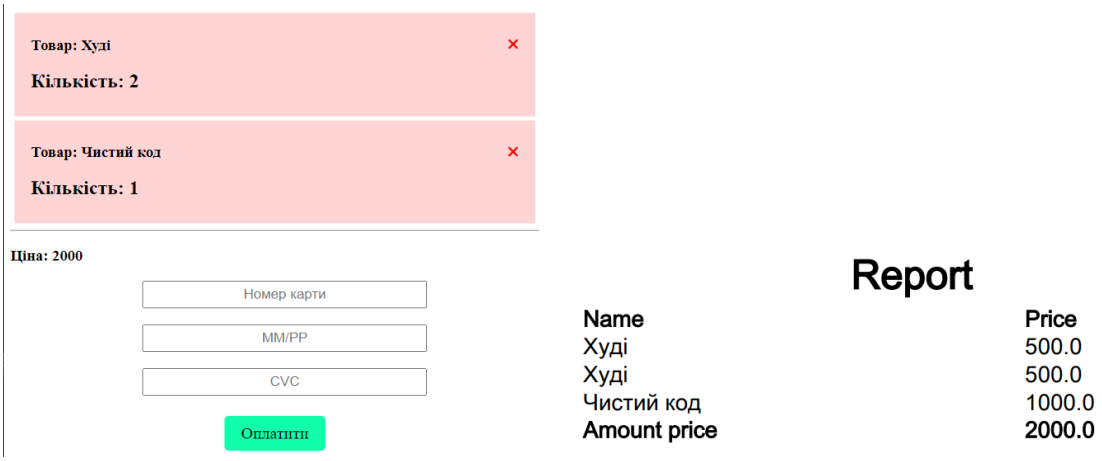

Рис. 5. Форма оплати та згенерований звіт про покупку

Супер-адміністратор може переглядати всіх зареєстрованих користувачів. На веб-сторінці конкретного користувача є кнопка переходу на інші веб-сторінки та інформація про користувача, яку було введено під час реєстрації. А також відображається роль даного користувача. Він може бути адміністратором або клієнтом. Також супер-адміністратор має «Адмін-панель». На ній є кнопки переходу на веб-сторінки для видалення користувачів і додавання та видалення унікального номера адміністратора.

Весь код можна знайти за посиланнями на репозиторії на GitHub: https://github.com/rbrahinets/shop-backend/ та https://github.com/rbrahinets/shop-frontend/.

Висновки.

У статті розглянуто процес проектування та реалізації веб-сервісу для організації онлайн-покупок і продажу товарів. Проведено порівняльний аналіз існуючих аналогічних платформ – «Rozetka» та «Prom» – і сформовано функціональні вимоги до розробленого сервісу. Обґрунтовано вибір технологічного стеку: Java і Spring Boot для серверної частини, TypeScript та Angular – для клієнтської, PostgreSQL – як система управління базами даних, Flyway – для керування міграціями схеми, Stripe – для обробки платіжних транзакцій. Реалізовано RESTful API та чотири рольові режими роботи: гість, авторизований покупець, адміністратор і супер-адміністратор. Наведено ключові фрагменти коду серверної та клієнтської частин і продемонстровано повний функціонал сервісу. Результати підтверджують, що розроблений сервіс є повнофункціональним аналогом сучасних торговельних платформ і може слугувати основою для комерційних рішень у сфері електронної торгівлі.

Список використаної літератури:

1. Schildt H. Java: The Complete Reference. 9th ed. – New York : McGraw-Hill Education, 2014. – 1312 p.
2. Spring Framework. Core Technologies. Official Reference Documentation. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/core.html (дата звернення: 01.06.2023).
3. Spring Security. Official Reference Documentation. URL: https://docs.spring.io/spring-security/reference/ (дата звернення: 01.06.2023).
4. Spring Data. Official Reference Documentation. URL: https://docs.spring.io/spring-data/commons/docs/current/reference/html/ (дата звернення: 01.06.2023).
5. Spring Data JDBC. Official Reference Documentation. URL: https://docs.spring.io/spring-data/jdbc/docs/current/reference/html/ (дата звернення: 01.06.2023).
6. Obe R., Hsu L. PostgreSQL: Up and Running. 3rd ed. – Sebastopol : O'Reilly Media, 2017. – 336 p.
7. Flyway. Official Documentation. URL: https://flywaydb.org/documentation/ (дата звернення: 01.06.2023).
8. Stripe. Developer Documentation. URL: https://stripe.com/docs (дата звернення: 01.06.2023).
9. MDN Web Docs. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 01.06.2023).
10. MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 01.06.2023).
11. TypeScript. Official Documentation. URL: https://www.typescriptlang.org/docs/ (дата звернення: 01.06.2023).
12. Angular. Official Documentation. URL: https://angular.io/docs (дата звернення: 01.06.2023).
13. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Ph.D. dissertation. University of California, Irvine, 2000. URL: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm (дата звернення: 01.06.2023).

References:

1. Schildt, H. (2014). Java: The Complete Reference (9th ed.). New York: McGraw-Hill Education. 1312 p.
2. Spring Framework Core Technologies. Official Reference Documentation. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/core.html
3. Spring Security Official Reference Documentation. URL: https://docs.spring.io/spring-security/reference/
4. Spring Data Official Reference Documentation. URL: https://docs.spring.io/spring-data/commons/docs/current/reference/html/
5. Spring Data JDBC Official Reference Documentation. URL: https://docs.spring.io/spring-data/jdbc/docs/current/reference/html/
6. Obe, R., Hsu, L. (2017). PostgreSQL: Up and Running (3rd ed.). Sebastopol: O'Reilly Media. 336 p.
7. Flyway Official Documentation. URL: https://flywaydb.org/documentation/
8. Stripe Developer Documentation. URL: https://stripe.com/docs
9. MDN Web Docs. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML
10. MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS

11. TypeScript Official Documentation. URL: https://www.typescriptlang.org/docs/
12. Angular Official Documentation. URL: https://angular.io/docs
13. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Ph.D. dissertation, University of California, Irvine. URL: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm

BRAHINETS Rostyslav,
Student, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIDKOWSKY Ruslan,
Doctor of Technical Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

DEVELOPMENT OF A WEB SERVICE FOR ONLINE SHOPPING AND PRODUCT SALES

Summary. Introduction. *In today's digital world, online commerce has become an essential component of everyday life, further accelerated by the COVID-19 pandemic and ongoing armed conflict in Ukraine. The growing demand for convenient, secure, and scalable online trading platforms motivates the development of purpose-built web services. The primary goal of this work is to design and implement a full-stack web service supporting both the purchase and sale of goods through the Internet.*

Purpose. *The purpose of this article is to present the process of developing a web service for online shopping and product sales: from analysis of existing analogues and technology selection to implementation and functional demonstration of the finished service.*

Results. *The article examines existing Ukrainian e-commerce platforms – "Rozetka" and "Prom" – and identifies functional requirements for the service under development. The technology stack is justified: Java and Spring Boot for the server side, TypeScript and Angular for the client side, PostgreSQL as the relational database, Flyway for database migration management, and Stripe for secure online payment processing. The RESTful API architecture ensuring stateless client-server communication is described, and key code fragments from both server-side and client-side components are provided. The implemented web service supports four operation modes – guest, authorized buyer, administrator, and super-administrator – each with a dedicated interface. Core functionality includes user registration and authentication, product browsing with category-based navigation and sorting, a shopping cart, bank card payment with automatic purchase report generation, and a full administrative interface for managing products, categories, and user accounts. The payment module was tested using Stripe's sandbox environment, confirming correct transaction processing.*

Conclusion. *The article presents the design and implementation of a web service for online shopping and product sales. The results of the study can be applied in the development of commercial trading platforms and the improvement of existing e-commerce solutions. The main outcomes include a comparative analysis of analogous services, justification of the selected technology stack, full implementation of the service using Java/Spring Boot and TypeScript/Angular, integration of the Stripe payment platform, and demonstration of all major functional modules.*

Keywords: online shopping, web service, internet store, PostgreSQL, Spring Boot, Angular, TypeScript, REST API, Stripe, e-commerce.

Одержано редакцією 20.10.2023 р.
Прийнято до публікації 06.12.2023 р.

ЗМІСТ

СЕКЦІЯ «ПРИКЛАДНА МАТЕМАТИКА»

СЕКЦІЯ «ІНФОРМАТИКА»

CONTENTS

APPLIED MATHEMATICS SECTION

INFORMATICS SECTION

ВІСНИК ЧЕРКАСЬКОГО УНИВЕРСИТЕТУ

Серія Прикладна математика. Інформатика
№1.2023

Відповідальний за випуск
Пасічний М.О., Головня Б.П.

Відповідальний секретар
Сердюк О.А.

Комп'ютерне верстання
Сердюк О.А.

Підписано до друку 22.12.2023.
Формат 60х84/в. Папір офсет. Друк офсет. Гарнітура Times.
Ум. друк. арк. 10,1. Наклад 100 прим.