УДК 004

Олійник Д.С., здобувач, гр. ВТ-22-1, ФІКТ
Локтікова Т.М., ст. викл. кафедри ІПЗ, ФІКТ
Державний університет «Житомирська політехніка»

ВЕБПЛАТФОРМА ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ З ПІДТРИМКОЮ TELEGRAM-БОТА

У сучасному світі онлайн-освіта стає все більш популярною, оскільки вона дозволяє навчатися у будь-який час та з будь-якого пристрою. Особливо актуальним є вивчення іноземних мов, адже знання мов сьогодні потрібне як для роботи, так і для повсякденного життя. Багато людей користуються мобільними застосунками, вебплатформами та месенджерами для навчання, тому виникає потреба у створенні зручної системи, яка би поєднувала декілька способів взаємодії з користувачем. Саме тому було вирішено розробити вебплатформу для вивчення іноземних мов із підтримкою Telegram-бота, через який користувач може отримувати навчальні матеріали, виконувати вправи та переглядати власний прогрес.

Перед початком розробки було визначено основне призначення системи. Вебзастосунок призначений для проходження мовних курсів, перегляду теоретичних матеріалів, виконання вправ та контролю результатів навчання. Користувач має можливість обирати курс, проходити уроки, виконувати тестові завдання та отримувати статистику власного прогресу. Також система підтримує механізм streaks, який дозволяє відстежувати активність користувача та мотивує регулярно займатися навчанням.

Для створення застосунку було використано клієнт-серверну архітектуру. Такий підхід дозволяє окремо реалізувати серверну частину та клієнтський інтерфейс. Обмін даними між компонентами системи здійснюється через REST API. Це надає можливість підключати не лише вебінтерфейс, а й Telegram-бот.

Серверна частина застосунку реалізована мовою програмування Python із використанням фреймворку FastAPI [1]. Вибір фреймворку пов’язаний із підтримкою асинхронної обробки запитів, високою швидкістю роботи та автоматичною генерацією документації Swagger/OpenAPI. Для роботи з базою даних обрано бібліотеку ORM SQLAlchemy [2], що спрощує створення моделей та взаємодію з таблицями.

Для збереження даних у системі використовується СУБД PostgreSQL. Дана система керування базами даних забезпечує їхнє надійне збереження та підтримує складні зв’язки між сутностями. Під час проєктування було створено структуру бази даних, яка включає таблиці користувачів, курсів, уроків, вправ, результатів тестування та прогресу навчання.

ER-діаграма створеної бази даних представлена на рисунку 1.

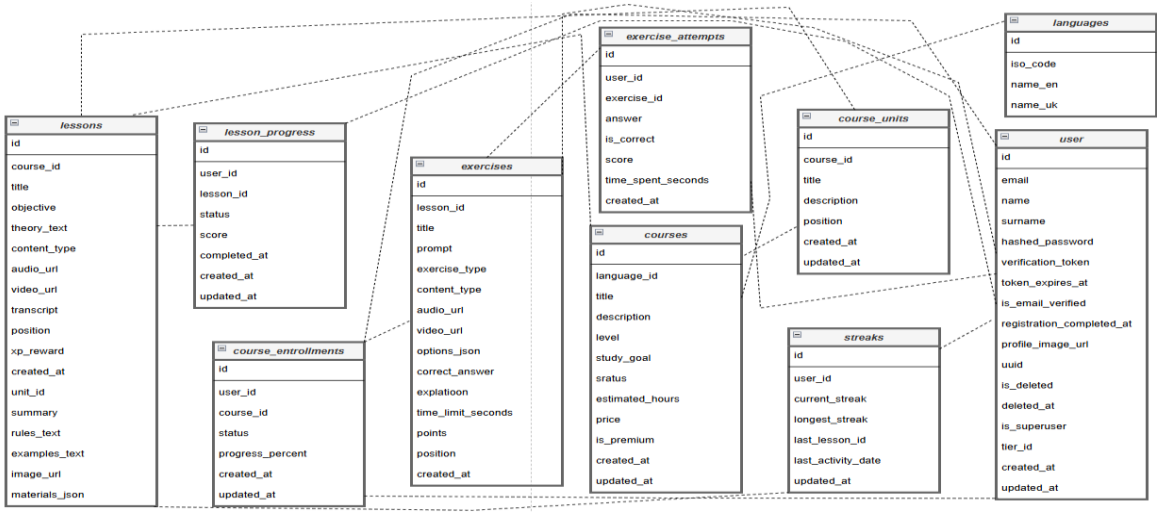

Рис. 1. ER-діаграма бази даних навчальної платформи

Для авторизації користувачів у системі використовуються JWT-токени, які забезпечують доступ до захищених ендпоінтів API. У проєкті також реалізовано Telegram-бот для швидкої взаємодії користувача із платформою. Для розгортання застосунку використовується платформа Docker, а система Redis застосовується для роботи з проміжними даними.

Розробка будь-якого програмного продукту супроводжується можливими помилками, тому важливо проводити тестування програмного забезпечення. Наразі застосунок проходить функціональне та нефункціональне тестування, що дозволяє перевірити коректність роботи системи та її відповідність поставленим вимогам.

Список використаних джерел:

1. FastAPI Documentation FastAPI Documentation : вебсайт. URL: https://fastapi.tiangolo.com/uk/
2. SQLAlchemy Documentation SQLAlchemy Documentation : вебсайт. URL: https://docs.sqlalchemy.org/en/20/

УДК 004

Кравчук М.Р., здобувач, гр. ВТ-22-1,ФІКТ
Лисогор Ю.І., ст. викл. кафедри ІПЗ,ФІКТ
Локтікова Т.М., ст. викл. кафедри ІПЗ, ФІКТ
Державний університет «Житомирська політехніка»

ОСОБЛИВОСТІ ПОБУДОВИ ТА ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ СПОРТИВНИХ ТОВАРІВ

На сьогодні, в епоху цифрової індустрії, сфера торгівлі зазнає змін. Традиційні методи продажу фізичних товарів поступаються місцем електронній комерції (e-commerce), яка здатна забезпечити велике охоплення аудиторії та автоматизацію процесів 24/7. Основою будь-якого інтернет-магазину є надійна архітектура програмного забезпечення, від якої залежить стабільність обробки замовлень, безпека транзакцій та збереження даних користувачів. Актуальність роботи зумовлена необхідністю створення стійкої платформи, яка дозволить автоматизувати процеси реєстрації клієнтів, управління віртуальним кошиком, оформлення замовлень та взаємодію з каталогом товарів.

При проєктуванні системи було обрано класичну багаторівневу архітектуру на базі патерну MVC (Model-View-Controller). Такий підхід дозволяє відокремити бізнес-логіку обробки даних від інтерфейсу користувача, що забезпечує гнучкість розробки та простоту підтримки коду. Основним інструментом розробки обрано мову програмування Java. Вибір обґрунтований її високою продуктивністю, суворою типізацією та надійністю при розробці систем корпоративного рівня. Як базовий фреймворк використано Spring Boot [1]. Він надає потужний інструментарій, включаючи вбудований сервер Apache Tomcat, систему інверсії управління (IoC) та об'єктно-реляційне відображення (Spring Data JPA), що значно пришвидшує процес розробки. Візуальна частина (Frontend) реалізована за допомогою серверного шаблонізатора Thymeleaf та адаптивного CSS-фреймворку Bootstrap 5 [2].

Основу серверної частини становить реляційна модель даних. Оскільки платформа електронної комерції передбачає чіткі зв’язки між користувачами, товарами та історією покупок, було прийнято рішення використовувати систему управління базами даних MySQL. На відміну від NoSQL-рішень, MySQL забезпечує повну відповідність принципам ACID (Atomicity, Consistency, Isolation, Durability), що є критично важливим для фінансових транзакцій та оформлення замовлень. Структура бази даних включає такі основні таблиці: Users – для збереження облікових даних, хешованих паролів та ролей доступу (клієнт, адміністратор); Products – містить детальну інформацію про товари (назва, опис, ціна, категорія, підкатегорія, шлях до зображення); Orders – містить загальну інформацію про оформлені замовлення (контактні дані, адреса доставки, загальна сума, статус); OrderItems – це таблиця-зв'язка, яка фіксує конкретні позиції товарів у межах певного замовлення та їхню вартість на момент покупки.

Серверна частина реалізує повний цикл покупки: від додавання товару в кошик до фіксації замовлення в базі. Особливим архітектурним рішенням є реалізація віртуального кошика на базі механізму сесій (анотація @SessionScope). Це дозволяє створювати ізольоване тимчасове сховище для кожного відвідувача без зайвого навантаження на базу даних. Оформлення замовлення відбувається у транзакційному режимі (@Transactional), що гарантує цілісність даних у разі програмних збоїв. Питання безпеки було вирішено шляхом впровадження модуля Spring Security. Паролі користувачів не зберігаються у відкритому вигляді, а захищаються алгоритмом хешування BCrypt. Система забезпечує захист від несанкціонованого доступу до адміністративних маршрутів та запобігає іншим небезпечним ситуаціям.

Для забезпечення актуальності асортименту реалізовано закритий модуль адміністрування. Авторизований персонал (з роллю ROLE_ADMIN) має доступ до панелі управління, яка дозволяє додавати нові товари в каталог. Особливістю модуля є вбудована система обробки зображень товарів через інтерфейс MultipartFile. Зображення фізично зберігаються у захищеній директорії на сервері, а в базу даних записується лише унікальний ідентифікатор файлу, що робить роботу бази даних швидшою.

У результаті виконання роботи було спроєктовано та реалізовано комплексну архітектуру інтернет-магазину спортивних товарів. Використання мови Java та фреймворку Spring Boot дозволило створити гнучку, надійну та стійку до навантажень систему. Інтеграція реляційної СКБД MySQL та транзакційної бізнес-логіки забезпечила високу цілісність фінансових даних. Розроблене рішення повністю відповідає сучасним вимогам до вебзастосунків електронної комерції: воно є безпечним завдяки механізмам Spring Security, має зручний клієнтський інтерфейс завдяки Thymeleaf та Bootstrap, а також легко піддається подальшому масштабуванню. Перспективи подальшого розвитку проєкту включають впровадження особистого кабінету з історією замовлень та інтеграцію сторонніх платіжних систем для оплати он-лайн.

Список використаних джерел:

1. Spring Boot Reference Documentation [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-boot/index.html.
2. Bootstrap 5 Documentation [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.0/getting-started/introduction/

УДК 004

Ботвін О.Ю., здобувач, гр. ЗІПЗ-22-1, ФІКТ
Лімінович І.Д., асист. кафедри ІПЗ, ФІКТ
Кушнір Н.О., ст. викл. кафедри ІПЗ, ФІКТ
Державний університет «Житомирська політехніка»

ОСОБЛИВОСТІ ПОБУДОВИ ТА ПРОЄКТУВАННЯ ANDROID-ЗАСТОСУНКУ ДЛЯ РОЗВИТКУ СЛОВНИКОВОГО ЗАПАСУ КОРИСТУВАЧІВ

У сучасних умовах мобільні застосунки стали одним із найбільш доступних і зручних засобів навчання. Особливо це стосується вивчення іноземних мов, де важливу роль відіграє регулярне повторення та постійна взаємодія користувача з навчальним матеріалом. Однією з основних проблем під час вивчення нової лексики є складність систематичного повторення слів і контролю власного прогресу. Традиційні способи навчання не завжди забезпечують достатню гнучкість та можливість адаптації до потреб конкретного користувача.

Для розв'язання цієї проблеми доцільним є використання мобільного Android-застосунку для розвитку словникового запасу. Такий застосунок дозволяє користувачу вивчати нові слова, повторювати вже вивчену лексику та отримувати статистику результатів навчання. Важливе значення при цьому має архітектура програмного забезпечення, оскільки саме вона визначає зручність підтримки, можливість розширення функціоналу та стабільність роботи системи.

Під час проєктування архітектури використано рекомендації Android App Architecture. Як основний архітектурний підхід обрано шаблон MVVM (Model–View–ViewModel), який забезпечує розділення інтерфейсу користувача, бізнес-логіки та роботи з даними [1, 2].

У межах MVVM Model відповідає за роботу з даними та бізнес-логіку; View забезпечує відображення інтерфейсу користувача; ViewModel обробляє стан інтерфейсу та взаємодіє з моделлю.

Для підвищення масштабованості система поділяється на окремі шари:

1) Presentation layer — компоненти інтерфейсу та ViewModel;
2) Domain layer — бізнес-логіка та сценарії використання;
3) Data layer — репозиторії та джерела даних.

Для організації доступу до даних використовується патерн Repository, який дозволяє працювати з локальними та віддаленими джерелами через єдиний інтерфейс. Управління станом реалізується за допомогою компонентів Android Jetpack, зокрема ViewModel та Flow або LiveData.

Основними функціональними можливостями системи є:

- створення власних словників;
- додавання та редагування слів;
- проходження вправ;
- планування повторень;
- перегляд статистики навчання;
- синхронізація даних між пристроями.

Розроблюваний застосунок складається з кількох основних модулів.

Модуль керування користувачем забезпечує збереження налаштувань та персоналізацію навчального процесу.

Модуль словникової бази відповідає за зберігання слів, перекладів, транскрипцій та прикладів використання.

Модуль навчання реалізує вправи та взаємодію користувача з навчальним матеріалом.

Модуль інтервального повторення забезпечує автоматичне планування повторень для кращого запам'ятовування слів.

Модуль аналітики призначений для відображення статистики та прогресу користувача.

Модуль локального сховища та синхронізації забезпечує роботу з локальною базою даних і взаємодію з віддаленим сервером.

Склад модулів може змінюватися залежно від вимог до системи та потреб користувачів. Тому одним із важливих етапів проєктування є визначення функціональних і нефункціональних вимог до застосунку.

Запропонована архітектура забезпечує зручність підтримки та подальшого розвитку застосунку. Використання модульного підходу та сучасних архітектурних рішень дозволяє підвищити якість програмного забезпечення та ефективність процесу вивчення словникового запасу користувачами.

Список використаних джерел:

1. Guide to app architecture / Android Developers. URL: https://developer.android.com/topic/architecture (дата звернення: 24.03.2026).

2. Dumbravan A., Price E. Clean Android Architecture: Take a Layered Approach to Writing Clean, Testable, and Decoupled Android Applications. Packt Publishing Ltd, 2022. 416 p. ISBN 978-1803240558

УДК 004

Король В.В., здобувач, гр. ІПЗ-22-2, ФІКТ
Кушнір Н.О., ст. викл. кафедри ІПЗ, ФІКТ
Локтікова Т.М., ст. викл. кафедри ІПЗ, ФІКТ
Державний університет «Житомирська політехніка»

ПОБУДОВА ТА ПРОЄКТУВАННЯ ВЕБСИСТЕМИ УПРАВЛІННЯ ВЗАЄМОВІДНОСИНАМИ З КЛІЄНТАМИ ДЛЯ МАЛОГО БІЗНЕСУ

Цифровізація процесів взаємодії з клієнтами є одним із найбільш важливих напрямів розвитку малого бізнесу. У практичній діяльності невеликих підприємств інформація про клієнтів, угоди, дзвінки та подальші дії зазвичай зберігається у різних месенджерах, таблицях або особистих нотатках працівників. Такий підхід ускладнює контроль виконання замовлень, створює ризик втрати потенційних клієнтів і не дає керівнику об'єктивної картини щодо ефективності роботи менеджерів. Тому актуальною є задача створення веборієнтованої CRM-системи, яка забезпечує централізований облік клієнтської бази, контроль етапів продажу та формування аналітичної інформації для прийняття управлінських рішень.

Метою роботи стало проєктування веборієнтованої CRM-системи для малого бізнесу, яка забезпечує централізоване управління клієнтами, угодами, задачами та аналітичними показниками за рахунок використання багаторівневої архітектури, патерна MVC і реляційної бази даних. Для досягнення поставленої мети було сформовано такі завдання: провести аналіз предметної області та систем-аналогів; визначити функціональні вимоги до власної розробки; обґрунтувати вибір технологічного стеку; спроєктувати архітектуру, об'єктну модель і структуру бази даних; описати алгоритми управління станами угод, валідації даних і розрахунку показників ефективності.

На першому етапі дослідження виконано аналіз існуючих CRM-рішень, орієнтованих на організацію продажів та комунікацію з клієнтами. Для порівняння розглянуто Uspacy як хмарну SaaS-платформу для командної взаємодії та Perfex CRM як self-hosted систему, яка працює на базі PHP-стеку [1, 2]. Обидва рішення мають корисні інструменти для роботи з клієнтами, однак не повністю відповідають потребам малого підприємства, яке потребує спрощеної логіки, контрольованого збереження даних і можливості подальшого розширення функціоналу. В табл. 1 подана порівняльна характеристика існуючих CRM-систем.

Таблиця 1

Параметр	**Uspacy**	**Perfex CRM**
Тип платформи	Хмарна SaaS-платформа	Self-hosted CRM
Доступ до коду	Відсутній	Повний
Гнучкість бізнес-логіки	Обмежена	Середня
Складність освоєння	Середня	Висока
Контроль даних	На боці провайдера	На боці власника сервера

Із табл. 1 випливає, що для малого бізнесу доцільною є розробка власної системи, оскільки вона надасть можливість поєднати простоту інтерфейсу, повний контроль над базою даних і адаптацію під конкретні бізнес-процеси. На відміну від універсальних CRM, проєктована система не перевантажується надлишковими модулями, а зосереджується на ключових діях: веденні клієнтської бази, створенні угод, зміні їх статусів, фіксації нотаток, плануванні задач і перегляді статистики продажів.

Для програмної реалізації обрано мову програмування PHP, оскільки вона широко застосовується для створення вебзастосунків, підтримує серверну обробку запитів, роботу з шаблонами та взаємодію з реляційними базами даних [3]. Як архітектурну основу використано патерн MVC (Model-View-Controller). Модель відповідає за роботу з даними та бізнес-правилами, представлення формує HTML-інтерфейс користувача, а контролер обробляє запити та координує виконання дій системи. Такий поділ зменшує зв'язаність компонентів і спрощує подальшу модифікацію програмного коду.

Архітектура системи побудована як багаторівнева структура. Рівень представлення забезпечує взаємодію користувача із системою через браузер. Рівень контролерів виконує маршрутизацію та первинну перевірку параметрів. Прикладний рівень реалізує сценарії створення клієнтів, угод, задач та нотаток. Доменний рівень містить правила переходу угод між етапами структури продажів і розрахунок показників ефективності. Рівень збереження даних відповідає за роботу з MySQL через параметризовані запити, що знижує ризик SQL-ін'єкцій і підвищує безпеку обробки комерційної інформації [4].

У системі визначено дві основні ролі користувачів: працівник CRM та адміністратор. Працівник створює клієнтів, редагує контактні дані, додає угоди, змінює їх статуси, формує нотатки та планує подальші дії. Адміністратор має розширені права: керує співробітниками, переглядає статистику

продажів, контролює історію взаємодій і може редагувати або видаляти критичні записи. Такий розподіл ролей забезпечує контроль доступу та зменшує ризик несанкціонованих змін у системі.

Логічна модель даних побудована навколо основних сутностей предметної області: Employee, Client, Deal, DealStage, InteractionLog, Note та CRMManager. Сутність Client зберігає інформацію про контрагентів, Deal описує комерційні операції, DealStage визначає етапи структури продажів, InteractionLog фіксує історію комунікацій, а Note використовується для службових коментарів. CRMManager виконує роль сервісного компонента прикладного рівня та координує створення об'єктів, формування статистики і виконання бізнес-операцій.

Фізична структура бази даних реалізується у MySQL. Вона містить таблиці employees, clients, deals, deal_stages, interaction_logs та notes. Між таблицями встановлюються зв'язки типу "один-до-багатьох": один клієнт може мати декілька угод і нотаток, один працівник може відповідати за багато угод, а кожна угода належить до певного етапу продажу. Така структура забезпечує посилальну цілісність, зменшує дублювання інформації та дозволяє швидко формувати вибірки за клієнтами, менеджерами, статусами угод і часовими періодами.

Окрему увагу приділено алгоритмічному забезпеченню CRM-системи. Алгоритм валідації перевіряє заповненість обов'язкових полів, коректність електронної адреси, номера телефону та числових значень суми угоди перед записом до бази даних. Алгоритм управління станами контролює перехід угоди між етапами структури продажів і автоматично оновлює часові мітки. Алгоритм логування фіксує кожну важливу дію користувача в журналі взаємодій, що надає можливість відновити повну історію роботи з клієнтом.

Аналітична частина системи базується на розрахунку основних показників ефективності. До них належать кількість створених лідів, кількість активних і закритих угод, сума продажів за період, конверсія угод та прогнозований дохід. Прогнозований дохід визначається шляхом зважування суми кожної угоди на ймовірність її успішного завершення, яка закріплюється за відповідним етапом структури продажів. Завдяки цьому адміністратор отримує не лише фактичні дані, а й орієнтовну оцінку майбутніх фінансових результатів.

Запропонована CRM-система орієнтована на практичне використання у невеликій команді, де важливими є швидке внесення даних, зрозуміла логіка роботи та мінімальна кількість зайвих дій. Вебінтерфейс є адаптивним, щоб працівники могли працювати із системою як на комп'ютері, так і на мобільному пристрої. Для захисту даних передбачається автентифікація користувачів, розмежування прав доступу, хешування паролів і використання параметризованих SQL-запитів. У подальшому архітектура системи дозволяє додати інтеграції з поштою, IP-телефонією, месенджерами або модулем автоматичних нагадувань без повної перебудови ядра програмного комплексу.

У результаті було обґрунтовано необхідність створення власної веборієнтованої CRM-системи для малого бізнесу. Аналіз двох систем-аналогів показав, що хмарні рішення мають обмеження щодо доступу до коду та контролю даних, а універсальні self-hosted CRM можуть бути надмірно складними для малих команд. Запропонована система охоплює 6 основних інформаційних сутностей бази даних, 2 ролі користувачів та ключові модулі управління клієнтами, угодами, нотатками, задачами й аналітикою. Використання PHP, MySQL, архітектурного шаблону MVC та параметризованих SQL-запитів забезпечує структурованість коду, зручність супроводу й розширення системи, а також підвищує базовий рівень захисту даних. Досягнення мети підтверджується тим, що спроєктована модель дозволяє централізувати облік клієнтів, зменшити ризик втрати інформації та сформувати кількісні показники ефективності роботи менеджерів.

Список використаних джерел:

1. Uspacy. CRM, комунікації та робочий простір для бізнесу. **Uspacy**. URL: https://uspacy.ua/ (дата звернення: 03.05.2026).
2. Perfex CRM. Powerful Open Source CRM. **Perfex CRM**. URL: https://www.perfexcrm.com/ (дата звернення: 03.05.2026).
3. PHP Documentation. PHP Manual. **PHP**. URL: https://www.php.net/manual/en/ (дата звернення: 03.05.2026).
4. MySQL Documentation. MySQL 8.4 Reference Manual. **MySQL**. URL: https://dev.mysql.com/doc/refman/8.4/en/ (дата звернення: 03.05.2026)

УДК 004

Миколайчук В.Ю., здобувач, гр. ІПЗ-22-2, ФІКТ
Лімінович І.Д., асист. кафедри ІПЗ, ФІКТ
Локтікова Т.М., ст. викл. кафедри ІПЗ, ФІКТ
Державний університет «Житомирська політехніка»

АРХІТЕКТУРА ТА РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ ПЛАТФОРМИ ДЛЯ ОРГАНІЗАЦІЇ РОЗВАЖАЛЬНИХ ЗАХОДІВ НА БАЗІ KOTLIN MULTIPLATFORM

У сучасному цифровому світі індустрія розваг та управління подіями вимагає створення максимально зручних, швидких та безперебійних користувацьких інтерфейсів. Сьогодні користувачі очікують від цифрових продуктів миттєвої реакції, можливості роботи на різних пристроях та інтуїтивно зрозумілого дизайну. Відповідно, забезпечення присутності сервісу на кількох платформах (вебсайт, iOS та Android) є важливою вимогою. Однак розробка окремих платформозалежних клієнтських застосунків для кожної платформи вимагає значних часових ресурсів, подвоює або потроює витрати на підтримку коду та ускладнює синхронізацію бізнес-логіки.

Метою даної роботи є розробка масштабованого клієнтського програмного забезпечення платформи для організації розважальних заходів із застосуванням технології Kotlin Multiplatform (KMP) [1]. Використання KMP у поєднанні з принципами «чистої архітектури» (Clean Architecture) та архітектурним патерном MVVM (Model-View-ViewModel) дозволяє створити єдину кодову базу для бізнес-логіки системи, забезпечуючи при цьому високу продуктивність та нативний користувацький досвід як у браузері, так і на мобільних пристроях.

На відміну від традиційних кросплатформних фреймворків, які змушують розробників використовувати єдиний підхід для відображення інтерфейсу (що часто призводить до втрати нативного відчуття застосунку), Kotlin Multiplatform пропонує гнучкіший механізм. KMP дозволяє спільно використовувати (share) код бізнес-логіки, маршрутизації та обробки даних між різними цільовими платформами, генеруючи для кожної з них нативний код.

Для розробки даної платформи клієнтський код компілюється у три різні формати, а саме (рис. 1): Android – компіляція у байт-код для JVM; iOS – компіляція у нативні бінарні файли (AArch64) за допомогою Kotlin/Native; Web – компіляція у JavaScript або WebAssembly (Wasm) за допомогою Kotlin/JS або Kotlin/Wasm для запуску безпосередньо у браузері.

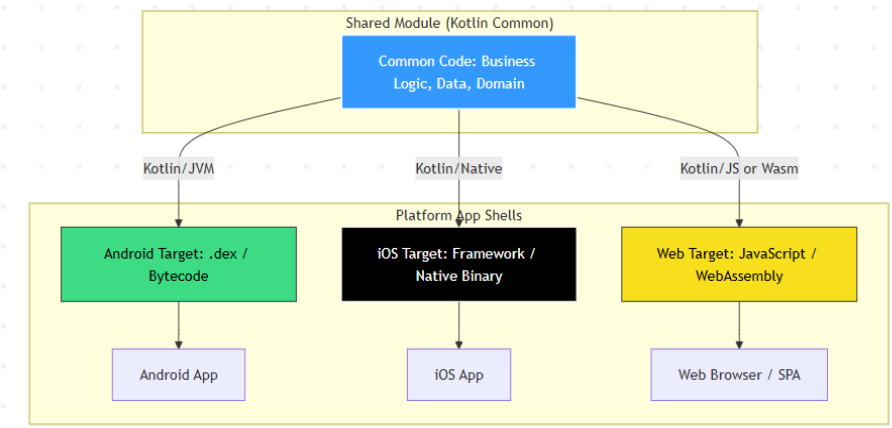

Рис. 1. Схема компіляції та розподілу коду Kotlin Multiplatform між платформами

Такий підхід дозволяє написати ядро застосунку один раз, гарантуючи абсолютно ідентичну поведінку системи на всіх пристроях під час фільтрації подій, валідації квитків або обробки користувацьких сесій.

Концепція Clean Architecture ідеально поєднується з парадигмою Kotlin Multiplatform. Уся клієнтська архітектура поділяється на три чітко відокремлені шари, більшість з яких реалізована у спільному модулі (Shared Module), а саме шар даних, шар домену та шар представлення.

Охарактеризуємо кожен із шарів.

Шар даних (Data Layer) відповідає за комунікацію з мережею та локальне кешування. Цей шар повністю складається зі спільного коду. Для мережевих запитів (взаємодії з REST API) використовується мультиплатформенний HTTP-клієнт Ktor. Для реалізації локального сховища на пристроях клієнтів

застосовуються мультиплатформенні бібліотеки абстракції. Джерело даних повністю приховане від інших шарів системи.

Шар домену (Domain Layer) являє собою найбільш незалежний шар системи, який містить сутності (Entities) та сценарії використання (Use Cases), наприклад, GetEventDetailsUseCase або RegisterForEventUseCase. Цей шар не містить жодних залежностей від платформи (ані від Android SDK, ані від браузерних API) і написаний на чистому Kotlin. Завдяки цьому логіка управління розважальними заходами є повністю універсальною і покривається модульними тестами лише один раз для всіх платформ.

Шар представлення (Presentation Layer) відповідає за відображення інтерфейсу та управління станами екранів. Саме тут архітектура розподіляється на спільний стан та платформозалежне відображення.

У шарі представлення клієнтських застосунків застосовано архітектурний шаблон MVVM. Особливістю даної реалізації є те, що ViewModel також винесена у спільний код (Shared Module) (рис. 2).

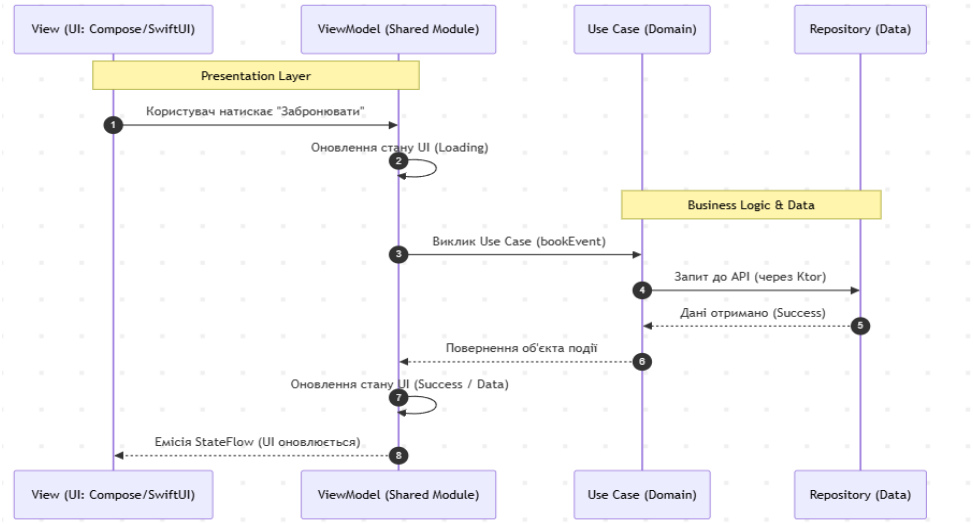

Рис. 2. Потік даних та взаємодія компонентів за патерном MVVM

Model (Модель) отримує дані від шару домену та адаптує їх для відображення.

ViewModel обробляє дії користувача (наприклад, натискання кнопки "Забронювати квиток") і керує станом (State) екрану (завантаження, успіх, помилка). Оскільки ViewModel написана на Kotlin і не прив'язана до конкретного UI-фреймворку, вона спільно використовується вебом та мобільними застосунками.

View (Представлення) являє собою єдиний компонент, який може відрізнятися залежно від платформи. Графічний інтерфейс "підписується" на зміни стану у ViewModel (через StateFlow). Для мобільного застосунку (Android) та вебсайту використовується декларативний UI-фреймворк Compose Multiplatform [2], а для iOS застосовується нативний SwiftUI (або той самий Compose Multiplatform, що дозволяє досягти максимального рівня перевикористання коду).

Платформа для організації розважальних заходів орієнтована на різноманітну аудиторію, тому основну увагу приділено швидкодії та доступності.

Вебзастосунок реалізується як Single Page Application (SPA) за допомогою генерації коду з Kotlin. Взаємодія з DOM-деревом або відмальовка на Canvas (у випадку Compose for Web) відбувається динамічно, без перезавантаження сторінки. Вся логіка фільтрації заходів, пошуку за категоріями та валідації форм реєстрації виконується безпосередньо у браузері користувача завдяки скомпільованому коду, що знижує навантаження на мережу та забезпечує миттєвий відгук інтерфейсу.

Мобільний застосунок гарантує високу продуктивність завдяки відсутності проміжних "мостів" (bridges), які характерні для інших кросплатформних рішень (наприклад, React Native). Застосунок компілюється безпосередньо у машинний код пристрою. Крім того, наявність спільного Data Layer дозволяє реалізувати надійне офлайн-кешування: завантажений розклад заходів або придбані QR-квитки доступні у мобільному застосунку навіть за відсутності під'єднання до Інтернету.

Значну увагу приділено безпеці персональних даних та управлінню сесіями користувачів (організаторів та відвідувачів). Токени доступу (JWT) надійно зберігаються на стороні клієнта за допомогою мультиплатформенних бібліотек шифрування (наприклад, EncryptedSharedPreferences для Android, Keychain для iOS та захищених механізмів браузера для Web). Уся навігація у застосунках

побудована таким чином, що ViewModel автоматично перевіряє права доступу користувача перед ініціалізацією приватних екранів (створення події, перегляд аналітики тощо).

Використання технології Kotlin Multiplatform для розробки клієнтської частини платформи (вебсайту та мобільних застосунків) є інноваційним та економічно виправданим рішенням..

Список використаних джерел:

1. Kotlin Multiplatform Development Documentation [Електронний ресурс] // JetBrains Documentation. – Режим доступу: https://www.jetbrains.com/help/kotlin-multiplatform-dev/get-started.html.

2. Compose Multiplatform UI Framework [Електронний ресурс] // JetBrains Product Page. – Режим доступу: https://www.jetbrains.com/lp/compose-multiplatform/.

УДК 004

Руденко О.І., здобувач, гр. ІПЗ-22-2, ФІКТ
Лімінович І.Д., асист. кафедри ІПЗ, ФІКТ
Локтікова Т.М., ст. викл. кафедри ІПЗ, ФІКТ
Державний університет «Житомирська політехніка»

РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ РОЗВАЖАЛЬНИМИ ЗАХОДАМИ

На сьогодні, в епоху цифрової трансформації, індустрія розваг зазнає суттєвих змін. Традиційні методи організації заходів відходять у минуле, поступаючись місцем високотехнологічним платформам, які здатні обробляти великі масиви даних у режимі реального часу. Найбільш важливою та відповідальною частиною таких платформ є серверна частина (backend), від якої залежить стабільність, безпека та масштабованість усієї системи [1]. Актуальність роботи зумовлена необхідністю створення надійного програмного забезпечення, яке дозволить автоматизувати процеси реєстрації користувачів, створення подій, управління контентом та взаємодію з клієнтськими застосунками через стандартизовані протоколи.

Аналіз сучасних платформ для організації заходів показав, що найбільш популярними аналогами є Eventbrite, Meetup та Ticketmaster [2]. Дані сервіси забезпечують функціонал створення подій, реєстрації користувачів, продажу квитків та взаємодії між організаторами й учасниками заходів. Проте існуючі рішення переважно орієнтовані на комерційне використання та мають обмеження щодо кастомізації серверної логіки й інтеграції сторонніх сервісів. Саме тому актуальним є створення власної серверної платформи, яка забезпечуватиме гнучкість, масштабованість та можливість адаптації під різні сценарії використання.

Метою дослідження є розробка архітектури та програмна реалізація серверної частини вебзастосунку для платформи організації розважальних заходів. Основна увага приділена створенню ефективного API, забезпеченню цілісності даних у реляційній базі та інтеграції хмарних сервісів для збереження медіаконтенту.

При проєктуванні системи було обрано клієнт-серверну архітектуру. Такий підхід дозволяє відокремити бізнес-логіку від інтерфейсу користувача, що забезпечує гнучкість розробки та можливість паралельного під’єднання різних типів клієнтів (вебсайтів, мобільних застосунків). Взаємодія між компонентами реалізована за допомогою архітектурного стилю REST (Representational State Transfer) [1].

Архітектура розробленої серверної частини представлена на рисунку 1.

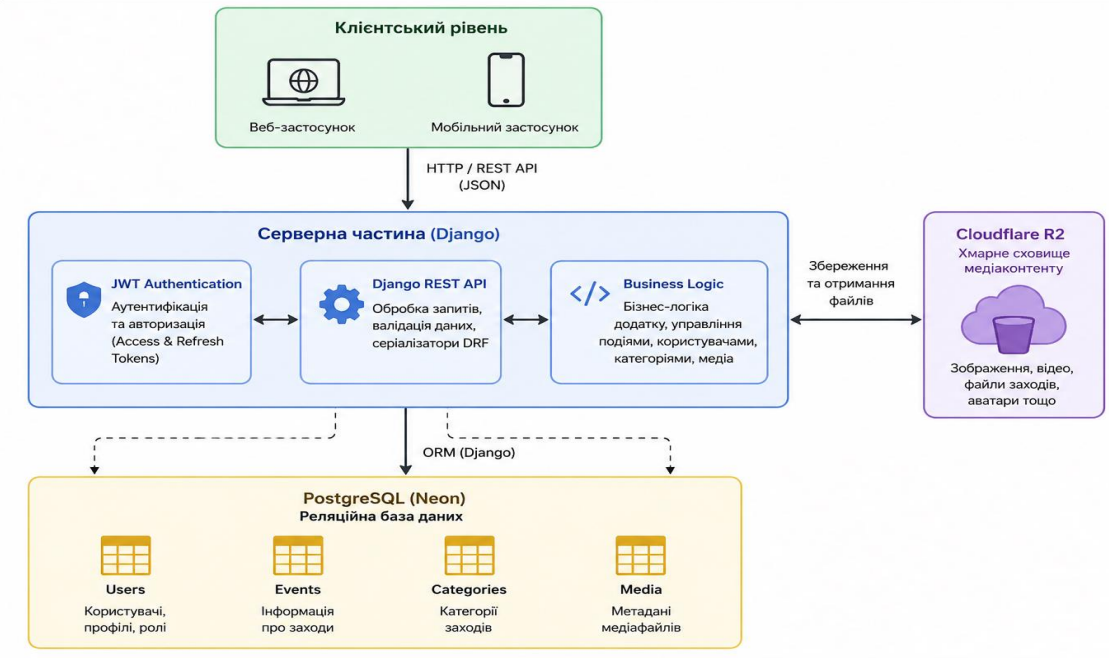

Рис. 1. Архітектура серверної частини платформи для організації заходів

Основним інструментом розробки обрано мову програмування Python. Вибір обґрунтований її високою продуктивністю при розробці складних логічних структур, наявністю великої кількості бібліотек та активною спільнотою. Як базовий вебфреймворк використано Django у поєднанні з Django REST Framework (DRF) [3]. Django надає потужний інструментарій "із коробки", включаючи об'єктно-

реляційне відображення (ORM), систему аутентифікації та адміністративну панель, що значно пришвидшує процес розробки та підвищує надійність коду.

Основою серверної частини є реляційна модель даних. Оскільки платформа передбачає чіткі зв'язки між користувачами, організаторами, заходами та категоріями, було прийнято рішення використовувати систему управління базами даних PostgreSQL. На відміну від NoSQL-рішень, PostgreSQL забезпечує повну відповідність принципам ACID (Atomicity, Consistency, Isolation, Durability), що є надзвичайно важливим для операцій реєстрації на заходи та фінансових транзакцій [4].

Для розгортання бази даних обрано хмарний сервіс Neon. Це серверне (serverless) рішення для PostgreSQL [5], яке дозволяє динамічно масштабувати ресурси залежно від навантаження. Структура бази даних включає такі основні сутності:

Users – для збереження облікових даних, профілів та ролей (відвідувач, організатор, адміністратор);

Events – для збереження детальної інформації про заходи (назва, опис, дата, місце проведення, ліміт учасників);

Categories – для класифікації заходів для зручного пошуку та фільтрації;

Media – для збереження метаданих зображень та відео, які завантажуються користувачами.

Для ефективної роботи з мультимедійним контентом інтегровано хмарне сховище Cloudflare R2 [6]. Це дозволяє зберігати статичні файли (зображення подій, аватари) поза межами основного сервера, що значно підвищує швидкість завантаження сторінок для кінцевого користувача та зменшує витрати на трафік.

Серверна частина реалізує повний життєвий цикл управління подією: від моменту створення організатором до архівації після завершення. Логіка обробки запитів побудована на серіалізаторах DRF, які відповідають за валідацію вхідних даних та перетворення об'єктів бази даних у формат JSON для передачі клієнту.

Питання безпеки було вирішено шляхом впровадження стандарту JSON Web Token (JWT) [7]. Процес авторизації побудований за такою схемою: клієнт надсилає облікові дані на сервер; сервер перевіряє дані та генерує два токени: Access Token (короткостроковий) та Refresh Token (довгостроковий); для доступу до захищених ресурсів (наприклад, створення нової події) клієнт додає Access Token у заголовок запиту. Така модель дозволяє реалізувати безстанну (stateless) архітектуру, що спрощує масштабування сервера та підвищує рівень захисту від несанкціонованого доступу.

Однією з переваг обраного стеку технологій є потужна вбудована адміністративна панель Django. Для проєкту було виконано кастомізацію адмін-панелі, що дозволило адміністраторам платформи здійснювати оперативну модерацію заходів, керувати списками категорій та контролювати активність користувачів. Реалізовано механізми фільтрації, пошуку та масового редагування записів, що мінімізує час на обслуговування системи.

У результаті виконання роботи було спроєктовано та реалізовано комплексну серверну частину платформи для організації розважальних заходів. Використання мови Python та фреймворку Django дозволило створити гнучку та надійну архітектуру. Інтеграція хмарних сервісів PostgreSQL (Neon) та Cloudflare R2 забезпечила високу доступність даних та швидкість роботи системи.

Розроблене серверне рішення повністю відповідає сучасним вимогам до вебзастосунків: воно є безпечним завдяки використанню JWT, масштабованим завдяки хмарній інфраструктурі та зручним у підтримці завдяки структурованому коду. Перспективи подальшого розвитку проєкту включають впровадження системи push-сповіщень через WebSockets [8] та інтеграцію платіжних шлюзів для автоматизації продажу квитків.

Список використаних джерел:

1. Architectural Styles and the Design of Network-based Software Architectures / Roy Thomas Fielding. Irvine : University of California, 2000. 162 p.

2. Eventbrite. Event Management and Ticketing Platform. URL: https://www.eventbrite.com/.

3. Django Documentation. Django Software Foundation. URL: https://docs.djangoproject.com/.

4. PostgreSQL Documentation. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/.

5. Neon Documentation. Neon. Serverless Postgres Documentation. URL: https://neon.tech/docs/introduction.

6. Cloudflare R2 Documentation. Cloudflare R2 Storage Documentation. URL: https://developers.cloudflare.com/r2/.

7. JWT Introduction. JSON Web Tokens Introduction. URL: https://jwt.io/introduction.

8. MDN Web Docs WebSocket API. MDN Web Docs. WebSocket API. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

УДК 004.7

Зарембицький Є.Ю., здобувач
Чижмотря О.В., ст. викладач
Державний університет «Житомирська політехніка»

ВИКОРИСТАННЯ БІБЛІОТЕКИ TANSTACK QUERY ТА МОВИ TYPESCRIPT ДЛЯ ВЗАЄМОДІЇ API У ВЕБІНТЕРФЕЙСАХ ОНЛАЙН-ШКІЛ

У сучасних інформаційних системах дистанційної освіти швидкість та надійність взаємодії клієнтської частини з сервером є визначальними факторами якості користувацького досвіду. Розробка вебінтерфейсів для онлайн-шкіл потребує обробки значних масивів асинхронних даних, що включають списки викладачів, розклади занять та результати тестувань. Традиційні підходи до управління станом за допомогою стандартних інструментів React часто призводять до надмірної складності коду та виникнення помилок синхронізації [1]. У даній роботі обґрунтовано використання зв'язки TanStack Query та TypeScript як ефективного інструментарію для побудови відмовостійкої архітектури фронтенд-частини.

Архітектурно клієнтська частина платформи побудована за принципом багатошаровості, де кожен рівень виконує ізольовану задачу. На нижньому рівні розташовано шар API-сервісів на основі бібліотеки Axios, який забезпечує низькорівневу комунікацію з сервером, включаючи налаштування базових URL та автоматичну підстановку авторизаційних токенів у заголовки запитів. Над цим шаром розгорнуто систему управління серверним станом на базі TanStack Query, яка виступає інтелектуальним посередником. Такий розподіл обов'язків дозволяє відокремити бізнес-логіку відображення компонентів від технічних деталей виконання мережевих запитів, що суттєво полегшує підтримку та подальше масштабування системи.

Важливим аспектом стабільності платформи є впровадження мови TypeScript для типізації контрактів взаємодії з API. Завдяки використанню строгої типізації стає можливим визначення чітких структур об'єктів передачі даних, що гарантує ідентичність даних на серверній та клієнтській сторонах. Статичний аналіз коду на етапі розробки дозволяє виявити потенційні помилки доступу до невизначених властивостей об'єктів, що є критичним при динамічній зміні ролей користувачів або статусів навчальних курсів [2]. Це створює надійний фундамент для рефакторингу та мінімізує кількість помилок у середовищі виконання.

Центральне місце в оптимізації взаємодії з сервером посідає перехід від імперативного управління станами завантаження до декларативної моделі TanStack Query. Використання механізмів автоматичного кешування та стратегії Stale-While-Revalidate дозволяє системі миттєво повертати дані з локальної пам'яті, одночасно виконуючи фонове оновлення. Це забезпечує безперебійність інтерфейсу та значно зменшує навантаження на сервер за рахунок інтелектуальної дедуплікації запитів. Крім того, вбудовані можливості автоматичних повторних спроб виконання запитів у разі мережевих збоїв гарантують високу доступність освітнього контенту навіть за нестабільного з'єднання [3].

Інтеграція технічного стеку з користувацьким інтерфейсом завершується використанням модульних стилів CSS та систем сповіщень реального часу. Завдяки взаємодії TanStack Query з бібліотекою react-hot-toast, будь-які відхилення у відповідях сервера трансформуються у зрозумілі для учня чи викладача візуальні підказки. Такий підхід не лише покращує UI/UX, а й дозволяє користувачеві отримувати миттєвий зворотний зв'язок щодо статусу його дій, наприклад, успішності запису на урок або збереження оцінки.

Отже, поєднання React, TypeScript та TanStack Query дозволяє реалізувати сучасну архітектуру онлайн-школи, яка поєднує високу продуктивність рендерингу через Virtual DOM із надійним управлінням асинхронними даними. Застосування цих технологій дозволяє оптимізувати мережевий трафік, підвищити стабільність клієнтської частини та забезпечити високу швидкість відгуку інтерфейсу, що є необхідною умовою для функціонування сучасних високонавантажених освітніх платформ.

Список використаних джерел:

1. React. The library for web and native user interfaces: documentation. URL: https://react.dev/ (дата звернення: 11.04.2026).
2. TypeScript Documentation. URL: https://www.typescriptlang.org/docs/ (дата звернення: 11.04.2026).
3. TanStack Query Documentation. URL: https://tanstack.com/query/latest/docs/framework/react/overview (дата звернення: 11.04.2026).

УДК 004.7

Зарембицький Є.Ю., здобувач
Чижмотря О.В., ст. викладач
Державний університет «Житомирська політехніка»

ПРОЄКТУВАННЯ АРХІТЕКТУРИ ВИСОКОНАВАНТЕЖЕНОЇ СИСТЕМИ ВЕБПЛАТФОРМИ ОНЛАЙН-ШКОЛИ НА БАЗІ ФРЕЙМВОРКУ NESTJS

Розробка сучасних освітніх вебплатформ потребує створення масштабованих та відмовостійких систем, здатних обробляти велику кількість запитів у режимі реального часу. Основним завданням такої системи є автоматизація взаємодії між учнями, викладачами та адміністрацією, що включає механізми запису на заняття, комунікації та модерації контенту.

Для забезпечення гнучкого управління доступом у системі запропоновано ієрархічну модель користувацьких ролей:

1. Користувач **–** базова роль, що дозволяє переглядати перелік вчителів за категоріями та подавати заявку на отримання статусу викладача або учня.
2. Учень **– м**ає доступ до функціоналу вибору спеціаліста за предметами, перегляду відгуків та реєстрації на навчальні сесії.
3. Вчитель **– с**пеціалізований акаунт, що активується адміністрацією після перевірки кваліфікації.
4. Модератор – користувач, відповідальний за дотримання правил платформи та моніторинг активності
5. Адміністратор **- з**дійснює повний аудит транзакцій, модерацію заявок, управління списками предметів та контроль за діями користувачів.
6. Власник – суб’єкт з максимальним рівнем повноважень, що здійснює стратегічне управління платформою та фінансовий аудит.

У якості базового інструментарію серверної частини обрано фреймворк NestJS. Це зумовлено необхідністю впровадження модульної архітектури, що дозволяє ізолювати логіку роботи з різними сутностями бази даних. Використання мови TypeScript забезпечує строгу типізацію, що мінімізує кількість логічних помилок на етапі компіляції та спрощує підтримку коду.

Однією з ключових переваг обраного стеку є вбудована підтримка інструментів фільтрації запитів:

Guards **–** використовуються для реалізації механізмів авторизації на основі JWT-токенів та контролю доступу за ролями.

Interceptors – дозволяють трансформувати вихідні дані та логувати дії користувачів для подальшого аудиту.

Pipes – забезпечують валідацію вхідних даних, гарантуючи цілісність інформації перед її потраплянням до контролерів та бази даних.

Застосування модульного підходу NestJS дозволяє створити архітектуру, яка є легкою для масштабування та адаптації під нові вимоги ринку дистанційної освіти. Поєднання TypeScript та інструментарію NestJS забезпечує високу надійність системи та швидкість обробки клієнтських запитів [1].

Важливою складовою проєктованої платформи є застосування принципу модульної декомпозиції, де кожна функціональна область виділяється в окремий незалежний модуль. Такий підхід забезпечує високий рівень інкапсуляції логіки, оскільки внутрішні процеси модуля приховані від інших частин системи, а взаємодія між ними відбувається через чітко визначені інтерфейси та експортовані компоненти. Це забезпечує високу масштабованість: у разі зростання навантаження на певний функціонал, відповідний модуль може бути легко винесений в окремий мікросервіс без радикальної перебудови всієї системи.

Для управління життєвим циклом об'єктів та забезпечення слабкої пов'язаності між компонентами архітектури впроваджено патерн інверсія управління у формі впровадження залежностей. Завдяки цьому підходу, створення примірників сервісів та їх конфігурування делегується вбудованому контейнеру NestJS, що звільняє розробника від необхідності вручну ініціалізувати складні ієрархії об'єктів. Це суттєво підвищує гнучкість системи: наприклад, заміна сервісу відправки повідомлень або перехід на іншу стратегію кешування потребує лише зміни реєстрації провайдера в модулі, не впливаючи на код контролерів, які використовують ці залежності.

Крім того, використання Dependency Injection у поєднанні з TypeScript-інтерфейсами значно спрощує процес модульного тестування. Під час тестування бізнес-логіки реальні сервіси можуть бути легко замінені на mock-об’єкти, що гарантує ізольованість тестів та стабільність розробки на всіх етапах життєвого циклу інформаційної системи.

Список використаних джерел:

1. NestJS. A progressive Node.js framework: documentation. URL: https://docs.nestjs.com/ (дата звернення: 08.04.2026).

УДК 004.7

Ільїн І.В., здобувач
Нерода С.І., викладач
Державний університет «Житомирська політехніка»

АРХІТЕКТУРНІ ПІДХОДИ ДО РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ ДЛЯ УПРАВЛІННЯ IOT-ПРИСТРОЯМИ ЗАСОБАМИ REACT NATIVE

Стрімкий розвиток Інтернету речей (IoT) зумовлює зростання попиту на програмні системи дистанційного моніторингу та керування підключеними пристроями. Мобільні застосунки є одним із ключових інтерфейсів взаємодії користувача з IoT-інфраструктурою, забезпечуючи доступ до телеметричних даних, конфігурування та керування пристроями в режимі реального часу.

Одним із актуальних підходів до розробки таких застосунків є використання фреймворку React Native, який дозволяє створювати кросплатформні рішення для Android та iOS на основі єдиної кодової бази мовою JavaScript. Це суттєво скорочує час розробки та знижує витрати на підтримку програмного продукту порівняно з роздільною нативною розробкою під кожну платформу.

Метою дослідження є порівняльний аналіз монолітного та модульного архітектурних підходів до організації React Native-застосунку для управління IoT-пристроями, а також оцінка ефективності протоколів зв'язку REST API та MQTT у контексті передачі телеметричних даних. Аналіз архітектури проводиться за трьома критеріями: кількість прямих міжмодульних залежностей, час інтеграції нових компонентів та складність тестування окремих підсистем. Порівняння протоколів здійснюється за показниками затримки передачі даних, споживання трафіку та надійності з'єднання в умовах нестабільної мережі.

Для практичної верифікації результатів розроблено прототип мобільного застосунку з модульною архітектурою, який забезпечує моніторинг стану IoT-пристроїв та дистанційне керування ними. Застосунок реалізовано із централізованим керуванням станом через Redux Toolkit, з використанням React Navigation для організації навігації та підтримкою двох режимів зв'язку - REST API і MQTT - для порівняння їх характеристик в однакових умовах.

Попередні результати свідчать, що протокол MQTT демонструє нижчу затримку та менше споживання трафіку порівняно з REST API при передачі частих телеметричних повідомлень, що робить його більш придатним для IoT-сценаріїв із високою частотою оновлення даних. Водночас модульна архітектура забезпечує нижчий рівень міжмодульних залежностей, що позитивно впливає на масштабованість системи при підключенні нових класів IoT-пристроїв і протоколів передачі даних.

Варто зазначити, що запропонований підхід не обмежується конкретним типом IoT-пристроїв і може бути адаптований для широкого кола застосувань - від систем розумного будинку до промислової телеметрії. Модульна організація коду у поєднанні з підтримкою декількох протоколів зв'язку дозволяє розширювати функціональність застосунку без суттєвої переробки існуючої архітектури, що підтверджує універсальність запропонованого рішення.

Отримані результати дозволяють сформулювати практичні рекомендації щодо вибору архітектурного підходу та протоколу зв'язку для розробки мобільних клієнтів IoT-систем на ранніх етапах проєктування.

Список використаних джерел:

1. Пакула А.А., Паламарчук Є.А. Комунікація мобільних додатків на React Native та Bluetooth Low Energy пристроїв. Математичне моделювання. Т. 1(50). 2024. DOI: https://doi.org/10.31319/2519-8106.1(50)2024.304856

2. Зарічук О.Г. Порівняльний аналіз фреймворків для розробки мобільних застосунків: рідні, гібридні чи кросплатформні рішення. Вісник Черкаського державного технологічного університету. Т. 28. № 4. 2023. С. 23–27. URL: https://doaj.org/article/79cbe09ac1c94d179c59e8d90483e400

3. Markowski M., Smołka J. A comparative analysis of the Flutter and React Native frameworks. Journal of Computer Sciences Institute. 2023. Vol. 29. P. 346–351. DOI: https://doi.org/10.35784/jcsi.3794

4. Cicirelli F., Guerrieri A., Vinci A. Smart Monitoring and Control in the Future Internet of Things. Sensors. 2022. Vol. 22. № 1. DOI: https://doi.org/10.3390/s22010027

5. Oliveira R. et al. Evaluating HTTP, MQTT and WebSocket Protocols for IoT Applications. International Journal of Innovative Research and Scientific Studies. 2025. Vol. 8. № 1. P. 679–694. URL: https://www.ijirss.com/index.php/ijirss/article/view/4414

6. React Native Team. The New Architecture is here. React Native Blog. 2024. URL: https://reactnative.dev/blog/2024/10/23/the-new-architecture-is-here

УДК 004.7

Ільїн І.В., здобувач
Лобанчикова Н.М., к.т.н., доцент
Концидайло А.М., ст. викладач
Державний університет «Житомирська політехніка»

ЗАСТОСУВАННЯ ТЕХНІК ТЕСТ-ДИЗАЙНУ ДО ВЕРИФІКАЦІЇ ІГРОВОЇ ЛОГІКИ З РОЗГАЛУЖЕНОЮ СТРУКТУРОЮ СТАНІВ

Тестування програмних систем із розгалуженою логікою переходів між станами є окремою методичною проблемою в інженерії якості програмного забезпечення. Ігрові застосунки жанру Metroidvania становлять показовий клас таких систем: наявність численних комбінацій ігрових умов, залежностей між здібностями персонажа та доступністю локацій, паралельна робота кількох незалежних підсистем, а також нелінійна природа прогресії суттєво ускладнюють визначення мінімально необхідного та водночас достатнього набору тестових сценаріїв. На відміну від типових бізнес-застосунків із лінійними потоками обробки даних, ігровий застосунок цього жанру характеризується комбінаторним вибухом станів: кількість досяжних конфігурацій системи зростає експоненційно зі збільшенням кількості підсистем та механік, що взаємодіють одночасно. Ігри цього жанру - зокрема Castlevania: Symphony of the Night, Blasphemous, Ender Lilies: Quietus of the Knights - демонструють, що некоректна взаємодія навіть двох підсистем (наприклад, системи збереження прогресу та автомата станів гри) може призводити до критичних дефектів, які неможливо виявити без систематичного підходу до проєктування тестів.

Об'єктом дослідження є процес верифікації ігрової логіки застосунків із нелінійною структурою прогресії, предметом - методи систематичного проєктування тестових сценаріїв, орієнтовані на скорочення їх кількості при збереженні повноти покриття вимог. Аналіз існуючих підходів до тестування ігрового програмного забезпечення свідчить про переважання неструктурованого дослідницького тестування (exploratory testing), яке, попри свою практичну цінність, не забезпечує відтворюваності результатів, не дозволяє кількісно оцінити рівень охопленості вимог та ускладнює відстеження регресій при модифікації окремих підсистем. У роботі розглянуто класичні техніки тест-дизайну відповідно до специфіки ігрових компонентів та запропоновано методику їх систематичного застосування.

Діаграма переходів станів (рис. 1) використовується як основний інструмент побудови тестових сценаріїв для автомата керування грою - кожному валідному переходу між режимами (ігровий процес, пауза, меню, діалог) відповідає окремий тест-кейс, тоді як невалідні переходи охоплюються негативними перевірками. Таблиця прийняття рішень застосовується для тестування умовної логіки розблокування локацій: комбінації набутих здібностей персонажа виступають умовами, а зміна стану доступності ігрового середовища - відповідними діями системи. При чотирьох незалежних умовах кількість можливих комбінацій становить шістнадцять; техніка дозволяє виявити надлишкові та суперечливі правила ще на етапі аналізу вимог. Аналіз граничних значень охоплює числові параметри ігрової логіки - показники здоров'я персонажа, лічильники прогресу, координати тригерних зон переходу між локаціями - шляхом перевірки значень на межах допустимих діапазонів та поза ними. Еквівалентне розбиття дозволяє скоротити кількість тестових сценаріїв для перевірки параметризованих ігрових механік: стани системи з однаковою очікуваною поведінкою об'єднуються в класи еквівалентності, що унеможливлює дублювання тестового покриття.

Організація тестового процесу передбачає складання тест-плану з визначеними критеріями входу та виходу, пріоритизацію сценаріїв на основі матриці ризиків (Risk = Impact × Probability) та групування тест-кейсів у test suite за ознакою підсистеми. Ризик-орієнтований підхід дозволяє зосередити тестові зусилля на функціональності з найвищим поєднанням ймовірності відмови та критичності наслідків - наприклад, система збереження прогресу та переходи між локаціями мають вищий пріоритет, ніж допоміжні візуальні ефекти. Моніторинг якості здійснюється через стандартні метрики: щільність дефектів, покриття переходів станів, відсоток виконаних та успішно пройдених сценаріїв. Виявлені дефекти фіксуються у структурованих баг-репортах із зазначенням кроків відтворення, фактичного та очікуваного результату, критичності, пріоритету виправлення та поточного статусу в life cycle дефекту відповідно до стандарту IEEE 829.

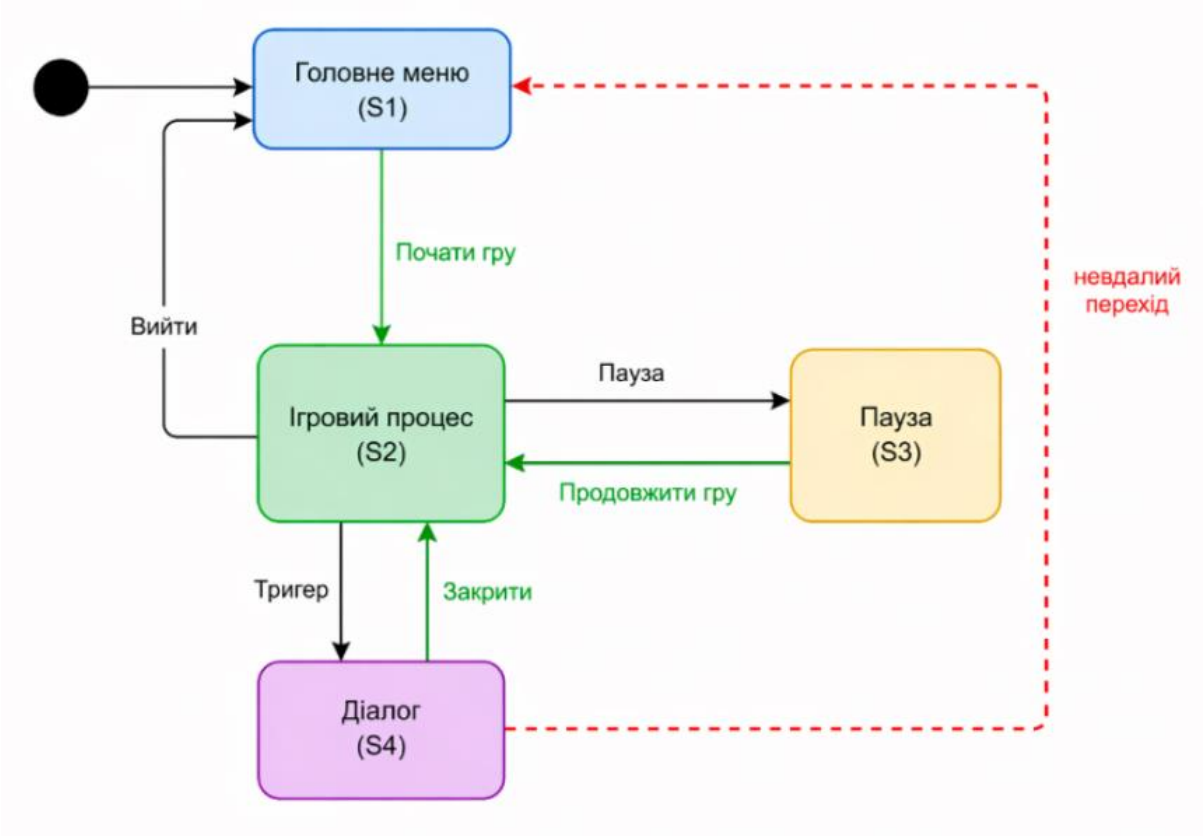

Рис. 1. Діаграма переходів станів ігрового застосунку: валідні переходи (суцільна лінія) та невалідний перехід S1>>>S4 як приклад негативного тест-кейсу

Систематичне застосування технік тест-дизайну до верифікації ігрової логіки з розгалуженою структурою станів дозволяє перейти від неструктурованого до відтворюваного та документованого процесу тестування. Запропонований підхід забезпечує кількісно вимірюване покриття вимог, скорочує кількість надлишкових тестових сценаріїв та створює основу для регресійного тестування при розширенні функціональності. Результати роботи можуть бути використані як методична основа для організації процесу забезпечення якості в проєктах розробки ігрових застосунків із подієво-орієнтованою архітектурою та складною структурою станів незалежно від конкретної реалізаційної платформи

Список використаних джерел:

1. Nystrom R. Game Programming Patterns / R. Nystrom. - Genever Benning, 2014. - 345 p. - URL: https://gameprogrammingpatterns.com

2. Martin R. C. OO Design Quality Metrics: An Analysis of Dependencies / R. C. Martin. - Object Mentor, 1994. - URL: https://condor.depaul.edu/dmumaugh/OOT/Design-Principles/oodmetrc.pdf

3. Oliveira R. A Framework for Metroidvania Games / R. Oliveira та ін. // Proceedings of SBGames 2020. - Recife, Brazil, 2020. - P. 844–853.

4. Folmer E. Component Based Game Development - A Solution to Escalating Costs and Expanding Deadlines? / E. Folmer // Component-Based Software Engineering: 10th International Symposium, CBSE 2007. - Springer, 2007. - P. 66–73.

УДК 004.35:004.42

Левицький А.А., викладач каф. КН
Левківський В.Л., доктор філософії, доцент
Державний університет «Житомирська політехніка»

ГРАФІЧНИЙ ПЛАНШЕТ ЯК ПРИСТРІЙ ВВЕДЕННЯ ДЛЯ КОМП'ЮТЕРНИХ ІГОР

Цифровий малюнок та 3D-скульптинг є поширеними для створення комп'ютерного контенту. Графічні планшети, які є затребуваними для цих напрямів, стали більш доступними в порівнянні із минулим десятиліттям. Можна припустити, що частка користувачів з графічними планшетами стає все більшою, тому є сенс створювати програмні продукти, що не пов'язані з професійною діяльністю.

Комп'ютерні ігри, які часто потребують нових рішень, можуть застосувати потенціал пристрою, однак ритм-ігри – єдиний жанр подібних додатків, у яких графічний планшет використовується явно. Постає питання щодо популяризації пристрою серед розробників ігор інших жанрів.

Графічний планшет – вид дигітайзера, що конвертує положення пера відносно активної області в координати курсора на моніторі комп'ютера. Перо графічного планшета – спеціальний стилус, який зчитує натискання та тонкощі рухів людини за допомогою рухомого наконечника.

Пристрій виконує введення одночасно кількох видів сигналів до комп'ютера:

- положення пера (двовимірний вектор) – координати пера на активній області пристрою, що залежать від параметра пристрою LPI (lines per inch) та розмірів активної області;
- сила натискання (float) – значення сили натискання пера на активну область пристрою, що залежить від максимально допустимого значення (зазвичай 2048 та більше); сприймається комп'ютером як ліва кнопка миші;
- нахил (двовимірний вектор) – значення нахилу пера відносно активної області пристрою, що залежить від максимального кута нахилу пера; не завжди підтримується планшетами;
- кнопки планшета та пера (boolean) – кнопки, що розташовані на відповідних складових пристрою та імітують звичайні натискання клавіш або їх комбінацій.

Для графічних планшетів існують драйвери, які дають можливість налаштувати пристрій необхідним чином. Вони суттєво відрізняються за своєю роботою залежно від моделі, а також оновлюються з великою періодичністю, тому ігрові додатки не мають доступу до повної інформації пристрою. У поширених операційних системах існують власні драйвери для графічних планшетів, які дають змогу отримати силу натискання, нахил та дані кнопок пера, але без налаштувань інших аспектів. Це полегшує роботу розробників із сигналами пристрою.

Ігрові додатки розпізнають графічний планшет як маніпулятор типу комп'ютерної миші. На цій підставі, наприклад, для ритм-ігор планшет стає кращим пристроєм введення, адже дає користувачу кращий контроль ситуації під час гри. Однак пристрій має більше сигналів для застосування. Для прикладу буде представлено сценарії, де їх використання буде доречним:

1. Ритм-ігри. Силу натискання можна застосувати у створенні нових елементів ігор такого жанру. Приклад: за допомогою кольорового градієнта або різниці розміру лінії, по якій потрібно провести пером у такт композиції, гравець повинен притиснути перо з необхідною силою.
2. Стрибок в іграх з ігровими персонажами. Силу натискання можна конвертувати в силу стрибка персонажа, що є більш контрольованим у порівнянні з обрахунком часу затискання клавіші.
3. Створення та розміщення об'єктів на сцені. Сила натискання має потенціал до зміни розміру об'єкта, а нахил – до його повороту.

Програмною базою для прикладу можливості реалізації механік слугуватимуть операційна система Windows 11, що має вбудований драйвер Windows Ink, та ігровий рушій Unity.

Unity має підтримку графічних планшетів через пакет Input System [1]. Він використовує Windows Ink для зчитування сили натискання, нахилу та кнопок пера. Для цих дій у пакеті реалізований клас Pen, в якому визначені сигнали як параметри: Pen.pressure, Pen.tilt та до чотирьох кнопок пера відповідно.

Отже, графічний планшет є повноцінним пристроєм введення, який можна використати для створення або доповнення ігрових додатків. Він підтримується операційними системами та дає можливість створювати нові механіки й переосмислювати існуючі за допомогою сигналів, що не відтворюються іншою периферією. В доповіді було описано графічний планшет, наведено аргументацію щодо доцільності його використання в ігрових додатках, а також представлено три сценарії застосування сигналів планшета.

Список використаних джерел:

1. Unity Manual. Input System: Pen, tablet, and stylus support. URL: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.19/manual/Pen.html

УДК 004.7

Школьна А.Л., здобувач
Савіцький Р.С., ст. викладач
Толстой І.А., ст. викладач
Державний університет "Житомирська політехніка"

ВИКОРИСТАННЯ ШІ-АСИСТЕНТА ДЛЯ АВТОМАТИЗАЦІЇ ПЕРЕВІРКИ ТА ОПТИМІЗАЦІЇ 3D-МОДЕЛЕЙ У ВЕБСИСТЕМАХ

Проаналізовано основні проблеми користувачів за налаштування параметрів друку та запропоновано модель інтелектуальної системи з автоматичним виявленням дефектів геометрії та оптимізацією витрат матеріалів. Визначено роль чат-бота на основі штучного інтелекту як інструменту підтримки прийняття рішень для підвищення ефективності адитивного виробництва.

Сучасні вебсервіси для роботи з 3D-друком здебільшого забезпечують лише базову взаємодію з обладнанням з залишенням самостійного налаштування параметрів за користувачем. Це створює труднощі для користувачів без достатнього досвіду через потребу у врахуванні великої кількості факторів процесу підготовки моделі до друку, зокрема температурних режимів, швидкості друку, щільності заповнення та конструктивних особливостей моделі.

Проблема ускладнюється частою наявністю помилок геометрії у STL-моделях – негерметичних поверхонь, самоперетинів, пошкоджених або некоректно орієнтованих полігонів. Такі дефекти призводять до помилок на етапі розрізання моделі на шари (слайсингу) або до отримання неякісного виробу.

Для вирішення зазначених проблем запропоновано інтеграцію ШІ-асистента з аналізом завантаженої моделі та автоматичним виявленням типових дефектів геометрії. На основі аналізу система формує рекомендації щодо виправлення моделі або коригування параметрів друку. Додатково передбачено використання накопичених даних про попередні успішні друки для підбору оптимального матеріалу та базових налаштувань.

Окрему увагу приділено контролю безпеки. Запропонований підхід передбачає перевірку згенерованого G-коду перед передачею на пристрій для виявлення потенційно небезпечних команд з негативним впливом на роботу обладнання.

Архітектура запропонованої системи побудована за принципом діалогової взаємодії з користувачем через текстовий інтерфейс чат-бота, інтегрованого у вебсервіс керування 3D-друком. Користувач завантажує STL-модель та отримує покрокові поради з підготовки до друку у форматі природної мови без потреби у вивченні технічної документації слайсера. ШІ-асистент аналізує запит користувача з урахуванням характеристик моделі та параметрів обладнання для формування персоналізованих рекомендацій.

Ефективність ШІ-асистента залежить від накопичення бази даних попередніх друків з фіксацією успішних та невдалих результатів. Кожен друк фіксується у системі з повним набором параметрів (температура екструдера та столу, швидкість друку, висота шару, відсоток заповнення) разом з оцінкою якості виробу від користувача.

Технічна реалізація ШІ-асистента передбачає використання великої мовної моделі для обробки природномовних запитів користувача разом зі спеціалізованими алгоритмами аналізу геометрії STL-моделей через бібліотеки комп'ютерної графіки. Перевірка G-коду виконується через парсер команд з порівнянням значень параметрів з допустимими діапазонами для конкретного обладнання. Поточні обмеження запропонованого підходу охоплюють залежність якості рекомендацій від обсягу накопиченої навчальної бази, потребу в стабільному з'єднанні з сервером ШІ-асистента та обмежену підтримку специфічних типів обладнання у початковій версії системи.

Практичне застосування такого підходу забезпечує зменшення кількості помилок під час підготовки моделей до друку, скорочення часу налаштування та підвищення стабільності результатів, особливо для користувачів початкового рівня.

Інтеграція ШІ-асистента у веб-системи керування 3D-друком є доцільним кроком у напрямі підвищення доступності адитивних технологій. Автоматизація перевірки моделей та надання рекомендацій щодо налаштувань дозволяє спростити процес підготовки до друку, зменшити кількість браку та підвищити ефективність використання ресурсів

Список використаних джерел:

1. Офіційний сайт BambuLab: https://shop.bambulab.com.ua/bambu-connect

УДК 004.7

Ломоносов І.О.
Савіцький Р.С.
Толстой І.А.
Державний університет «Житомирська політехніка»

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ГРАВЦІВ D&D З МОДУЛЕМ АВТОМАТИЗОВАНОГО ЗБОРУ ДАНИХ

Проаналізовано потреби спільноти настільно-рольових ігор (HPI) та запропоновано структуру інформаційної системи для систематизації ігрового контенту. Розглянуто використання методів вебскрапінгу для наповнення бази даних бестіарію, заклинань та предметів. Визначено переваги централізованого збереження ігрових параметрів для швидкого доступу під час сесій.

Популярність Dungeons & Dragons зумовлює появу великої кількості цифрових інструментів, проте більшість з них мають обмежений функціонал або складний поріг входження для локальних спільнот. Основна проблема полягає у розрізненості даних з розміщенням офіційних джерел, аматорських доповнень (homebrew) та мультимедійних ресурсів на різних платформах.

Запропонована система базується на модульній архітектурі з підсистемою парсингу даних як центральним елементом. Використання автоматизованих скриптів забезпечує збір актуальної інформації з відкритих реєстрів (SRD) з пришвидшенням процесу наповнення бази даних системи. Такий підхід вирішує проблему ручного введення характеристик монстрів чи описів механік з частим супроводом помилками.

Інтерфейс системи для Гейм-майстрів забезпечує керування сценаріями у режимі реального часу. Організація даних за категоріями (бестіарій, магія, спорядження) забезпечує швидкий пошук для підтримки динаміки ігрового процесу.

Архітектура системи побудована за принципом клієнт-серверного застосунку з розділенням на три рівні. Клієнтський рівень реалізовано як вебінтерфейс на основі сучасних JavaScript-фреймворків з адаптацією під роботу з планшетами та смартфонами для зручності використання під час ігрових сесій. Серверний рівень забезпечує обробку запитів через REST API з аутентифікацією користувачів за ролями. Рівень зберігання даних базується на реляційній базі даних з нормалізованою структурою таблиць для ігрових сутностей.

Модуль автоматизованого збору даних реалізує конвеєр обробки інформації з відкритих джерел HPI-контенту. Парсер аналізує HTML-сторінки офіційного SRD з виділенням структурованих елементів (характеристики монстрів, тексти заклинань, опис властивостей предметів) через CSS-селектори та XPath-вирази. Зібрані дані проходять етап нормалізації з приведенням до уніфікованої схеми бази даних та валідацією значень числових атрибутів (рівень небезпеки, бонуси атаки, кількість кубиків шкоди). Періодичний запуск парсера за розкладом забезпечує синхронізацію локальної бази даних з оновленнями офіційного джерела без ручного втручання адміністратора.

Майстер гри отримує набір інструментів для оперативного управління ігровим сеансом. Швидкий пошук за бестіарієм забезпечує миттєвий доступ до характеристик монстрів за назвою, рівнем небезпеки або типом істоти з відображенням повного блоку статистик у форматі офіційного посібника. Конструктор зустрічей розраховує загальний рівень небезпеки бою на основі обраних монстрів та порівнює його з рівнем партії гравців для оцінки балансу складності. Механізм відстеження ініціативи фіксує черговість ходів учасників бою з автоматичним підрахунком тривалості ефектів заклинань та станів персонажів. Гравці отримують доступ до особистих листів персонажів з можливістю редагування інвентарю, відстеження здоров'я та підготовлених заклинань у реальному часі під час сесії.

Практичне використання системи забезпечує зменшення часу підготовки до ігрових сесій, підвищення динаміки ігрового процесу та централізоване зберігання історії кампанії для подальшого аналізу та документування.

Розробка спеціалізованої інформаційної системи дозволяє структурувати великі обсяги ігрових даних та забезпечити гравцям швидкий доступ до необхідної інформації. Автоматизація збору даних за допомогою парсингу мінімізує людський фактор та підвищує актуальність контенту.

Список використаних джерел:

1. D&D 5th Edition Systems Reference Document (SRD). URL: https://dnd.wizards.com/resources/systems-reference-document

УДК 004.7

Ломоносов І.О.
Савіцький Р.С., ст. викладач
Толстой І.А., ст. викладач
Державний університет «Житомирська політехніка»

ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ЛОКАЛІЗАЦІЇ ТА ГЕНЕРАЦІЇ КОНТЕНТУ В СИСТЕМАХ ПІДТРИМКИ НРІ

Розглянуто питання інтеграції ШІ-асистентів у вебсистеми для гравців D&D. Визначено роль штучного інтелекту в автоматизованому перекладі специфічної ігрової термінології та наданні рекомендацій щодо адаптації ігрових механік. Проаналізовано підходи до обробки природної мови для покращення користувацького досвіду.

Одним з головних бар'єрів для розвитку D&D-спільноти в Україні виступає відсутність повної офіційної локалізації правил та додаткових матеріалів. Велика кількість контенту публікується англійською мовою з ускладненням гри для новачків. Використання традиційних систем перекладу часто викривляє ігрову термінологію з чітко визначеними механічними значеннями.

Для вирішення проблеми запропоновано інтеграцію ШІ-асистента на базі великих мовних моделей (LLM). На відміну від звичайних перекладачів, ШІ враховує контекст правил D&D 5e з коректною адаптацією назв характеристик, станів персонажів та описів заклинань.

Додатково ШІ-модуль виконує роль інтелектуального помічника з генерацією ідей для сюжетів та розрахунком балансу бойових зіткнень. Система аналізує запит користувача природною мовою та видає структуровану відповідь на основі бази даних правил. Такий підхід знижує навантаження на майстра гри з зосередженням на творчому аспекті гри.

Архітектура ШІ-модуля побудована на принципі контекстно-залежного запиту до великої мовної моделі через REST API. Перед формуванням відповіді система додає до запиту користувача структурований контекст з ігрової бази даних, що охоплює офіційні правила D&D 5e, словник специфічних термінів з прийнятими українськими відповідниками та історію поточної ігрової кампанії. Такий підхід отримав назву Retrieval-Augmented Generation (RAG) та забезпечує генерацію відповідей з прив'язкою до конкретного ігрового контексту замість використання загальних знань мовної моделі. Кешування часто використовуваних запитів зменшує навантаження на API мовної моделі та витрати на її використання.

ШІ-модуль підтримує генерацію різних типів ігрового контенту за запитом майстра гри. Для опису локацій система формує атмосферний текст з урахуванням заданого жанру (фентезі, готика, кіберпанк), типу місцевості (підземелля, місто, ліс) та рівня небезпеки. Генерація неігрових персонажів охоплює створення імені, характеру, біографії та мовної манери з прив'язкою до раси та класу персонажа. Для побудови сюжетних ліній система пропонує варіанти зав'язки, основних подій та можливих розвилок розгортання історії. Усі згенеровані елементи зберігаються у базі даних кампанії з можливістю подальшого редагування майстром гри.

Якість перекладу ігрової термінології забезпечується через двоетапний процес валідації. На першому етапі мовна модель виконує переклад з урахуванням контексту правил, на другому етапі система звіряє переклад зі словником затверджених україномовних еквівалентів та виправляє розбіжності з підтримкою консистентності термінології у межах кампанії. Поточні обмеження ШІ-модуля охоплюють залежність від якості мовної моделі та її обмежень з контекстним вікном, потребу у стабільному з'єднанні з API мовної моделі та витрати на токени для активних користувачів. Перспективним напрямом розвитку виступає використання локально розміщених мовних моделей з адаптацією під специфіку D&D-термінології через тонке налаштування на корпусі ігрових текстів.

Інтеграція ШІ-асистента в інформаційну систему для поціновувачів D&D виступає ефективним інструментом для подолання мовного бар'єру та автоматизації рутинних завдань. Використання інтелектуальних алгоритмів перекладу забезпечує високу точність передачі ігрових механік зі сприянням популяризації гри.

Список використаних джерел:

1. OpenAI API Documentation. Text Generation and Translation. URL: https://platform.openai.com/docs/guides/text-generation

УДК 621.396

Терпак А.Р., студент
Прокопенко В.В., аспірант
Морозов Д.С., ст. викладач
Державний університет «Житомирська політехніка»

ВПЛИВ ДОДАТКОВИХ ЄМНІСНИХ ЕЛЕМЕНТІВ НА ХАРАКТЕРИСТИКИ АПЕРТУРНО-ЗВ'ЯЗАНИХ ПАТЧ-АНТЕН З КРУГОВОЮ ПОЛЯРИЗАЦІЄЮ

Багатошарові апертурно-зв'язані патч-антени з круговою поляризацією широко застосовуються у складі антенних решіток X-діапазону завдяки можливості просторового розділення випромінюючого елемента та мікросмужкової мережі живлення [1]. Подібна структура дозволяє зменшити паразитний вплив мікросмужкової діаграмоутворюючої схеми на характеристики випромінювання та забезпечує додаткові можливості оптимізації коефіцієнта відбиття та осьового співвідношення (AR) [2].

У роботі досліджувалась багатошарова апертурно-зв'язана патч-антена з хрестоподібною щілиною збудження. Верхній випромінюючий шар було виконано на основі НВЧ-ламінату RT/duroid 6002 товщиною 1,524 мм. Нижній шар мікросмужкової лінії живлення реалізовано на фольгованому ламінаті Rogers RO4350B товщиною 0,508 мм. Товщина металізації в обох шарах становила 0,035 мм. Між шарами розташовувався металізований екран із X-подібною щілиною збудження.

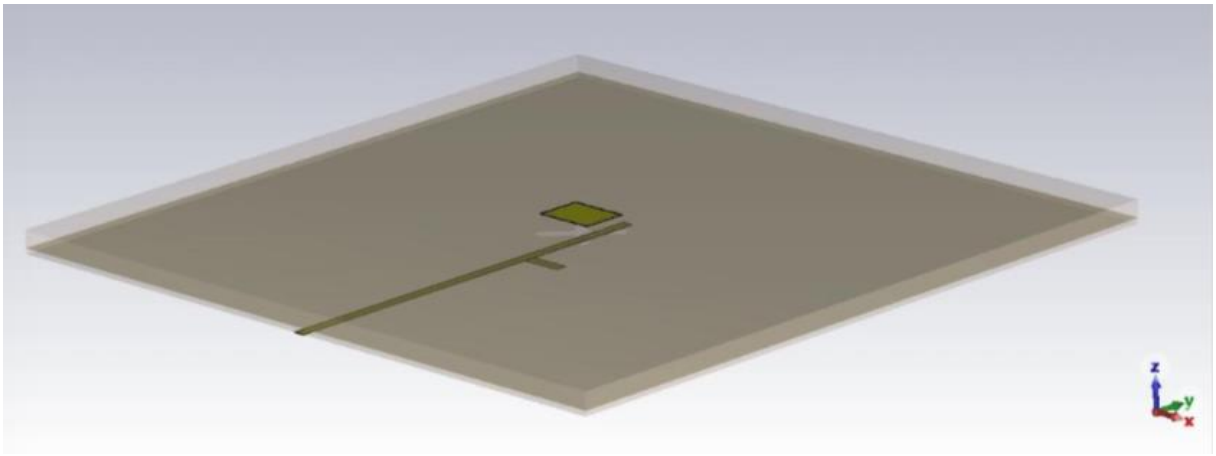

Рис. 1. Модель багатошарової апертурно-зв'язаної патч-антени з хрестоподібною щілиною

Для формування кругової поляризації використовувалась X-подібна щілина з різною довжиною плечей 5,1 мм та 6,1 мм. Розмір випромінюючого патча становив 3,8×5,1 мм, що є суттєво меншим за розміри класичних прямокутних патчів для аналогічного частотного діапазону та використовуваного діелектричного матеріалу. Додаткове узгодження мікросмужкової лінії живлення і системи патч-щілина здійснювалось за допомогою перпендикулярного стаба довжиною 3,7 мм, розташованого на відстані 6,15 мм від центра щілини збудження.

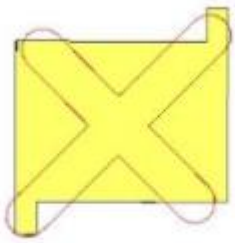

Рис. 2. Взаємне перекриття хрестоподібної щілини збудження патчем і його додатковими ємнісними елементами

Одним із ключових факторів формування кругової поляризації в апертурно-зв'язаних патч-антенах є взаємодія двох ортогональних резонансних мод, параметри яких визначаються співвідношенням

геометричних розмірів щілини та патча [3]. При цьому робоча смуга з низьким AR формується двома резонансними піками, пов'язаними з плечами хрестоподібної щілини [4, 5].

У роботі досліджувався вплив додаткових локальних ємнісних елементів на характеристики антени. Як узгоджуючі елементи використовувались мініатюрні металізовані стаби розміром 0.5×0.5 мм, розташовані на поверхні випромінюючого патча навпроти довшого плеча X-подібної щілини.

Результати електродинамічного моделювання в середовищі CST Studio Suite показали, що введення додаткових локальних ємнісних неоднорідностей дозволяє змінити положення низькочастотного резонансного піка, пов'язаного з довшим плечем щілини збудження. Додавання локальної ємності призвело до зміщення робочої смуги з низьким осьовим відношенням у нижню частину діапазону частот. Зокрема, смуга частот з AR менше 3 dB була зміщена з діапазону 9.85–10.55 ГГц до 9.6–9,8 ГГц.

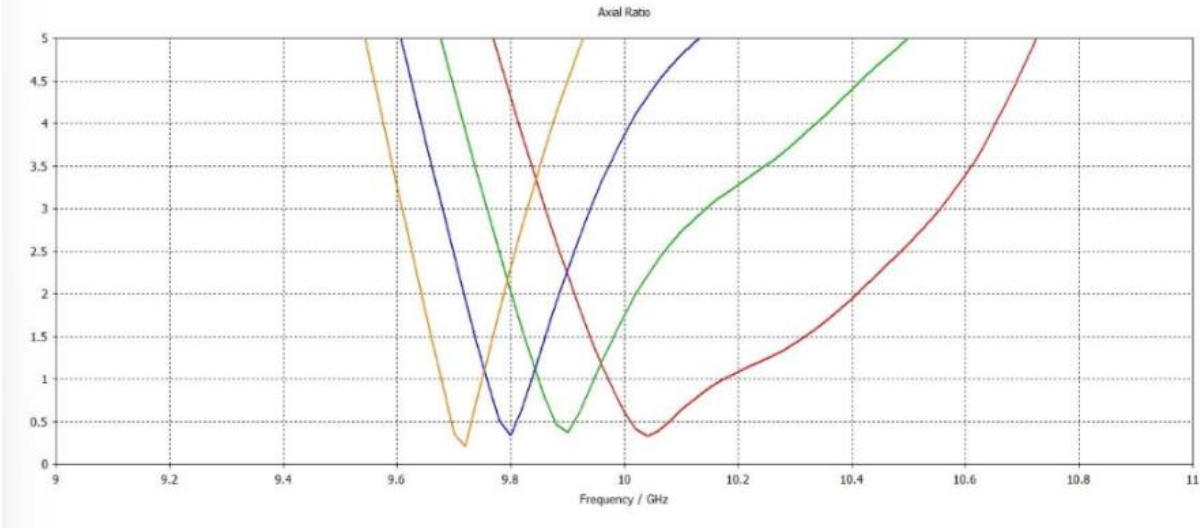

Рис. 3. Графік залежностей AR від частоти для патч-антени кругової поляризації з додатковими ємнісними елементами різної довжини

На Рисунку 3 продемонстрований вплив довжини ємнісних елементів на AR патч-антени кругової поляризації. Червона крива відповідає моделі без ємнісних елементів, зелена – ємнісні елементи квадратної форми з довжиною сторони 0,2 мм, синя – 0,4 мм, жовта – 0,5 мм.

Встановлено, що внесення додаткових ємнісних елементів призводить до зміни характеристик узгодження системи «патч–щілина–мікросмужкова лінія». Для компенсації зміни вхідного імпедансу виконувалась додаткова оптимізація параметрів узгоджуючого мікросмужкового стаба, що розміщений на мікросмужковій лінії живлення перед щілиною. Збільшення його довжини дозволило компенсувати вплив додаткової ємності патча та забезпечити низький рівень коефіцієнта відбиття S11 у робочому діапазоні частот.

Отримані результати показують, що використання локальних ємнісних елементів на поверхні випромінюючого патча може бути ефективним методом тонкого налаштування характеристик кругової поляризації апертурно-зв'язаних антенних елементів без суттєвої зміни геометрії щілинного збуджувача. Запропонований підхід може бути використаний при проектуванні багатошарових антенних решіток з високими вимогами до осьового відношення та кросполяризаційної розв'язки в системах супутникового зв'язку та радіолокації

Список використаних джерел:

1. Cao W., Wang C., Wang Y., Yuan B. A wideband circularly polarized aperture-coupled stacked microstrip antenna with enhanced axial-ratio bandwidth. Sensors. 2020. Vol. 20. No. 3. Art. no. 939. DOI: 10.3390/s20030939.
2. Deng C., Li Y., Zhang Z., Feng Z. A wideband sequential-phase fed circularly polarized patch antenna array. IEEE Antennas and Wireless Propagation Letters. 2014. Vol. 13. P. 1399–1402. DOI: 10.1109/LAWP.2014.2343939.
3. Row J.-S., Yeh C.-K. Wideband circularly polarized aperture-coupled stacked patch antenna. Microwave and Optical Technology Letters. 2009. Vol. 51. No. 12. P. 2907–2911. DOI: 10.1002/mop.24770.
4. Fakharian M. M., Rezaei P., Orouji A. A compact aperture-coupled microstrip antenna with circular polarization for X-band applications. Progress In Electromagnetics Research C. 2018. Vol. 83. P. 161–170. DOI: 10.2528/PIERC18032403.
5. Ta S. X., Park I. Wideband circularly polarized crossed-slot coupled microstrip antenna using a single feed. Applied Sciences. 2019. Vol. 9. No. 3. Art. no. 408. DOI: 10.3390/app9030408

УДК 621.396

Загацький В.А., студент
Козяр Я.А., аспірант
Морозов Д.С., ст. викладач
Державний університет «Житомирська політехніка»

КОМПЕНСАЦІЯ ПАРАЗИТНИХ НЕОДНОРІДНОСТЕЙ У МІКРОСМУЖКОВИХ ДІАГРАМОУТВОРЮЮЧИХ СХЕМАХ АНТЕННИХ РЕШІТОК

Проектування багатошарових антенних решіток X- та Ku-діапазонів із мікросмужковими діаграмоутворюючими мережами вимагає забезпечення високої точності виготовлення елементів мікросмужкової структури. Особливо критичними є геометричні параметри ліній живлення, дільників потужності, узгоджуючих трансформаторів та компенсаційних елементів, оскільки навіть незначні зміни їх розмірів можуть призводити до помітного погіршення коефіцієнта відбиття, нерівномірності фазового розподілу та зростання паразитних втрат у діаграмоутворюючій мережі.

Для швидкого прототипіювання подібних антенних систем часто застосовується фрезерування на станках з ЧПК поверхонь фольгованих НВЧ-ламінатів. Цей підхід дозволяє оперативно виготовляти дослідні зразки антенних решіток без використання фотолітографії та промислових технологій травлення. Разом з тим технологічні обмеження обробки суттєво впливають на точність формування геометрії мікросмужкових структур, особливо в X- та Ku-діапазонах, де характерні розміри елементів узгодження та неоднорідностей становлять десяті частки міліметра.

Одним із ключових технологічних факторів є діаметр мікрофрези, який визначає мінімально можливий радіус внутрішнього скруглення контурів мікросмужкової лінії. Для частот X- та Ku-діапазонів забезпечення необхідних характеристик антенної решітки зазвичай вимагає точності позиціонування інструмента на рівні 0,05–0,1 мм. При цьому реальна геометрія виготовленої структури суттєво відрізняється від ідеалізованої моделі, що використовується під час електродинамічного синтезу.

Особливо помітний вплив технологічних обмежень спостерігається в областях локальних неоднорідностей мікросмужкової мережі, зокрема у Т-подібних дільниках потужності, чвертьхвилевих трансформаторах та 90-градусних поворотах мікросмужкових ліній. У класичних електродинамічних моделях подібні елементи часто розглядаються як структури з ідеально гострими внутрішніми кутами та точно визначеними параметрами компенсаційних елементів [1]. Під час фрезерування на станках з ЧПК такі геометричні особливості замінюються скругленнями, радіус яких визначається діаметром використовуваної фрези.

У роботі досліджується вплив радіуса фрезерного інструмента на характеристики узгодження мікросмужкової діаграмоутворюючої мережі антенної решітки. Основну увагу приділено аналізу змін електродинамічних параметрів локальних неоднорідностей, що виникають внаслідок технологічних обмежень обробки. В роботі досліджувалась модель мікросмужкової лінії виконаної на фольгованому ламінаті RO4350B товщиною 0,508 мм, товщина шару металізації 0,035 мм. Ширина мікросмужкової лінії з хвилевим опором 50 Ом складає 0,91 мм.

Одним з основних елементів мікросмужкової лінії живлення антенної решітки є Т-подібний дільник. У Т-подібних дільниках потужності скруглення вершини трикутного узгоджуючого вирізу (УВ) призводить до зміни його компенсаційних властивостей [2]. У класичній конфігурації трикутний виріз використовується для часткової компенсації паразитної ємності, яка виникає в області Т-подібного розгалуження через локальне розширення провідника мікросмужкової лінії. Збільшення радіуса скруглення вершини вирізу зменшує ефективність компенсації паразитної ємності та погіршує характеристики коефіцієнта відбиття дільника потужності [3].

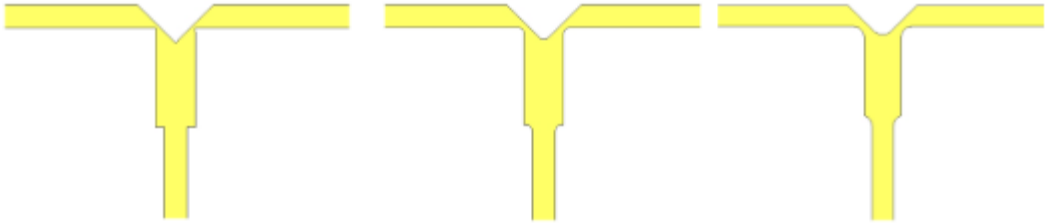

Рис. 1. Електродинамічні моделі Т-подібного мікросмужкового дільника з УВ глибиною 1,4 мм без врахування радіусу фрези при виготовленні (зліва), з радіусом фрези 0,25 мм (в центрі), з радіусом фрези 0,5 мм (справа)

Результати моделювання показують, що для компенсації впливу технологічного скруглення необхідно збільшувати глибину УВ. Подібна модифікація дозволяє частково відновити компенсаційні властивості неоднорідності та забезпечити прийнятний рівень узгодження в робочому діапазоні частот. Встановлено, що оптимальна глибина УВ залежить як від хвильового опору мікросмужкових ліній, так і від діаметра використовуваної мікрофрези.

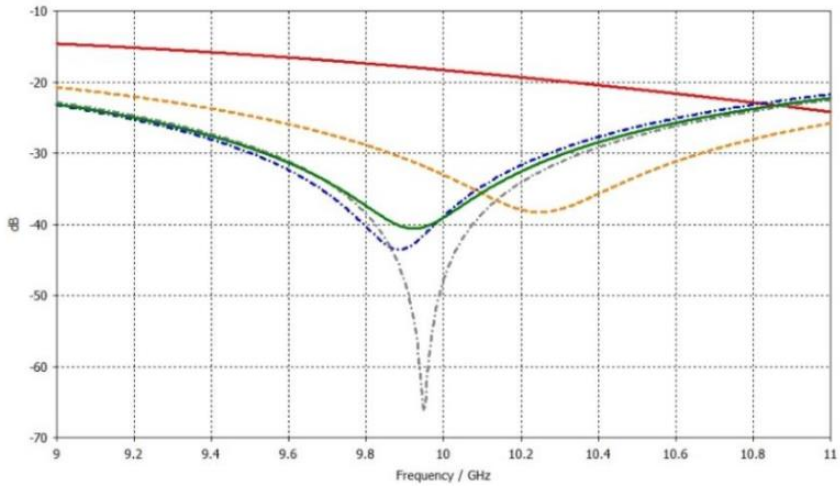

Рис. 2. Залежність коефіцієнта відбиття S11 моделі Т-подібного мікросмужкового дільника від параметрів УВ

На Рисунку 2 Залежність коефіцієнта відбиття S11 моделі Т-подібного мікросмужкового дільника від параметрів УВ. Так модель без УВ (червоний колір) має відносно низьке узгодження. Модель дільника з УВ глибиною 1,4 мм без врахування радіусу фрези при виготовленні (синій колір) має оптимальний узгодження по входу. Модель дільника з УВ глибиною 1,4 мм і радіусом фрези 0,25 мм (сірий колір) має дуже високий рівень узгодження, що пояснюється впливом скруглень на кути повороту самих мікросмужок. Модель дільника з УВ глибиною 1,4 мм і радіусом фрези 0,5 мм (помаранчевий колір) вже має значно меньший рівень узгодження, бо радіус фрези не дає змогу виконати трикутний УВ потрібної глибини. Модель дільника з УВ глибиною 1,5 мм і радіусом фрези 0,5 мм (зелений колір) компенсує на 0,1 мм глибину УВ і має блтизький до оптимального рівень узгодження.

Аналогічні ефекти спостерігаються і в областях 90-градусних поворотів мікросмужкових ліній. Відомо, що прямий поворот мікросмужки створює локальну паразитну ємність, викликану концентрацією електричного поля у внутрішньому куті неоднорідності. Для компенсації цієї паразитної ємності традиційно використовуються повороти зі зрізаним кутом, які дозволяють зменшити локальне розширення області струмів та покращити характеристики узгодження.

Під час фрезерування внутрішній кут повороту неминуче набуває додаткового скруглення, радіус якого визначається геометрією інструмента. Це призводить до появи додаткової паразитної ємності та зміни оптимальних параметрів компенсаційного зрізу. Результати дослідження показали, що зі збільшенням радіуса внутрішнього скруглення необхідно збільшувати величину компенсаційного зрізу для збереження мінімального рівня відбиття в області повороту.

Отримані результати демонструють необхідність врахування реальної геометрії фрезерування на станку з ЧПК вже на етапі електродинамічного синтезу мікросмужкової діаграмоутворюючої мережі. Використання ідеалізованих моделей без урахування радіуса інструмента може призводити до суттєвого розходження між результатами моделювання та характеристиками реального зразка антенної решітки.

Запропонований підхід дозволяє підвищити точність швидкого прототипіювання багатошарових антенних решіток X- та Ku-діапазонів та забезпечити більш передбачувані характеристики узгодження мікросмужкових діаграмоутворюючих мереж при використанні доступного фрезерного обладнання з ЧПК.

Список використаних джерел:

1. Lin D.-B., Wang M.-H., Huang Y.-L. Smooth bend structures using hybrid compensation for common-mode noise reduction. Applied Sciences. 2022. Vol. 12. No. 13. Art. no. 6479. DOI: 10.3390/app12136479.
2. Mao J., Xu Y., Zhang R., Yang M. Analysis of microstrip discontinuities and their compensation via the finite difference time domain method. International Conference on Microwave Technology and Computational Electromagnetics. 2009. P. 381–384. DOI: 10.1049/cp.2009.1309
3. Barzdenas V., Vasjanov A. Applying characteristic impedance compensation cut-outs to full radio frequency chains in multi-layer printed circuit board designs. Sensors. 2024. Vol. 24. No. 2. Art. no. 675. DOI: 10.3390/s24020675.

УДК 004

Ковальчук М.М., здобувач
Желізко В.В., викладач каф. комп'ютерних наук
Державний університет «Житомирська політехніка»

РОЗРОБКА ВЕБ-СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ЮВЕЛІРНОГО ВИРОБНИЦТВА НА ОСНОВІ NODE.JS ТА MYSQL

Ювелірне підприємство витрачає від 40 до 70 % собівартості виробу на дорогоцінні метали та каміння, тому точність обліку безпосередньо визначає фінансовий результат. На практиці менеджер виробничої дільниці може дізна-тися про перевитрату золота лише наприкінці місяця — після ручного зведення декількох таблиць. За цей час підприємство втрачає можливість оперативно відреагувати: скоригувати технологічну норму, зупинити партію або замовити сировину. Завдання полягає у створенні централізованої системи, яка відстежує кожну виробничу операцію без затримки між подією та відображенням у звіті.

Метою роботи є розробка концепції та програмного прототипу системи підтримки прийняття рішень (СППР) «JewelryDecision» для середніх ювелірних підприємств. Систему реалізовано на серверній платформі Node.js (v18 LTS) із використанням СУБД MySQL та REST API архітектури, що забезпечує масштабованість, надійність і зручний доступ до виробничих даних.

Ключову роль у системі відіграє реляційна модель бази даних, нормалі-зована до третьої нормальної форми, що охоплює чотири основні сутності: ProductionBatch (виробничі партії), MaterialUsage (облік витрат матеріалів), Forecast (прогнозні значення виробничих обсягів) та Reports (аналітичні звіти та KPI-показники). REST API реалізує CRUD-операції для кожної сутності, забезпе-чуючи гнучку інтеграцію з клієнтськими додатками та BI-інструментами. Така архітектура дозволяє централізовано зберігати та обробляти дані про всі етапи виробничого циклу без прив'язки до конкретного інтерфейсу.

Функціональне тестування підтвердило коректність роботи системи: CRUD-операції для всіх чотирьох сутностей виконуються без помилок; усі ендпоінти API повертають коректні відповіді відповідно до специфікації при навантаженні до 50 одночасних запитів. Розроблена система «JewelryDecision» забезпечує перехід ювелірного підприємства від розрізнених таблиць до централізованої цифрової платформи управління виробництвом.

Список використаних джерел:

1. Портна О.В. Управління витратами виробничих підприємств: теорія та пра-ктика. Харків : Монограф, 2018. 284 с.
2. Node.js Documentation. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 20.04.2026).
3. MySQL 8.0 Reference Manual. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 20.04.2026).
4. Brown E. Web Development with Node.js and Express: Leveraging the JavaScript Stack. 2nd ed. Sebastopol : O'Reilly Media, 2019. 332 p. URL: https://www. oreilly.com/library/view/web-development-with/9781492053507/ (дата звернен-ня: 20.04.2026).
5. Turban E., Sharda R., Delen D. Decision Support and Business Intelligence Systems. 9th ed. Upper Saddle River : Pearson, 2010. 912 p. URL: https:// www.researchgate.net/publication/258099211_Decision_Support_and_Business_ Intelligence_Systems_9th_Edition (дата звернення: 20.04.2026).
6. Ситник В.Ф. Системи підтримки прийняття рішень : навч. посіб. Київ : КНЕУ, 2004. 614 с. URL: https://www.lib.ugi.edu.ua/cgi-bin/koha/opac-detail.pl? biblionumber=2011 (дата звернення: 20.04.2026).
7. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures : doctoral dissertation. Irvine : University of California, 2000. URL: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm (дата звернення: 20.04.2026).

УДК 004.7

Можаровський Я.С., здобувач
Лобанчикова Н.М., к.т.н., доцент
Концидайло А.М., ст. викладач
Державний університет «Житомирська політехніка»

СПЕЦИФІКА ТЕСТУВАННЯ ІГРОВОГО БАЛАНСУ ТА МЕХАНІК У METROIDVANIA

Специфіка тестування у жанрі Metroidvania полягає в тому, що стандартні методи забезпечення якості (QA) потребують адаптації до умов нелінійного геймплею. Якість у цьому контексті визначається як «придатність для використання або призначення» і передбачає задоволення потреб гравця через стабільну функціональність та вивірений баланс. Досягнення цього стану в процесі життєвого циклу розробки (SDLC) вимагає глибокої інтеграції технік тест-дизайну в архітектурне планування рівнів.

На відміну від лінійних ігор, де послідовність проходження є фіксованою, Metroidvania будується на принципі взаємозалежності локацій та здібностей персонажа. Гравець постійно повертається до раніше відвіданих зон із новими можливостями, що створює розгалужену мережу залежностей між елементами ігрового світу. Саме ця архітектурна особливість унеможливлює застосування стандартних лінійних сценаріїв тестування і вимагає системного підходу до верифікації кожного переходу між станами.

Методи тест-дизайну, що застосовуються для верифікації механік Metroidvania, доцільно об'єднати у такі групи: діаграми переходу станів – для візуалізації еволюції можливостей персонажа та перевірки валідності доступів, що запобігає передчасному потраплянню у закриті зони; аналіз граничних значень – для контролю поступового некерованого зростання сили персонажа шляхом тестування стійкості балансу в екстремальних точках спорядження; таблиці прийняття рішень – для систематичної перевірки логічних умов доступу та складних синергій між отриманими навичками, що підтримує релевантність ігрового виклику; регресійні набори для критичних шляхів – для верифікації прохідності світу після коригування параметрів руху, щоб уникнути станів «soft-lock».

Особливе місце посідає верифікація фізичних колізій та взаємодій рушія. Аналіз специфічних технік пересування, як-от damage boost, дозволяє сформувати тест-кейси на основі досвіду speedrun-спільнот для виявлення вразливостей ігрової логіки. Таким чином команда QA виявляє дефекти, що виникають лише при специфічних комбінаціях вхідних параметрів та дій гравця.

Особливої уваги у життєвому циклі розробки потребує регресійне тестування параметрів руху персонажа. Зміна висоти стрибка чи швидкості пересування має каскадний вплив на прохідність усієї карти: локація, що була доступна з певними параметрами, може стати недосяжною після технічного виправлення. Тому після кожного коригування фізичних характеристик необхідно повторно верифікувати критичні точки доступу по всьому ігровому світу, щоб унеможливити стан повного блокування проходження. Це зумовлено тим, що ігровий світ у даному жанрі функціонує як цілісна взаємозалежна система, де модифікація одного параметра здатна порушити логіку прогресії у віддаленій частині карти.

Важливим аспектом є валідація суб'єктивного сприйняття складності. Статичні методи, зокрема аудит дизайн-документації та карт рівнів, дозволяють верифікувати коректність ігрового темпу ще до етапу активного кодування. Динамічне тестування, у свою чергу, спрямоване на перевірку інформативності навігації та відповідності візуальних елементів інтерфейсу очікуванням користувача. Окремо перевіряється економіка ресурсів: надмірна їх кількість знижує напруження, тоді як дефіцит може унеможливити проходження. Баланс між цими крайнощами є одним із ключових критеріїв якості ігрового досвіду.

Раннє виявлення помилок у балансі значно скорочує витрати на виправлення дефектів у межах життєвого циклу розробки. Описана методологія є масштабованою і може бути адаптована для інших жанрів із нелінійною прогресією, зокрема ігор із відкритим світом та процедурно генерованими рівнями. Таким чином, ефективне тестування Metroidvania базується на інтеграції технік тест-дизайну в планування – через застосування діаграм станів, аналізу граничних значень та регресійних наборів, що мінімізує ризики порушення логіки прогресії в умовах нелінійного геймплею.

Список використаних джерел:

1. Bycer J. Metroidvania: A Genre of Exploration and Progression. Game Design Deep Dive. Boca Raton : CRC Press, 2021. 214 p.

2. ISO/IEC/IEEE 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Geneva : ISO, 2011. 34 p.

3. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Hoboken : John Wiley & Sons, 2011. 256 p.

УДК 004.7

Можаровський Я.С., здобувач
Концидайло А.М., ст. викладач
Державний університет «Житомирська політехніка»

ТЕОРЕТИКО-ПРАКТИЧНІ АСПЕКТИ ВПРОВАДЖЕННЯ ЗОВНІШНЬОГО ТЕСТУВАННЯ В ЦИКЛ РОЗРОБКИ ПЗ

Актуальність дослідження зумовлена наявною проблемою зниження об’єктивності контролю якості в межах закритих циклів розробки через когнітивні упередження внутрішніх команд. Пропонується розгляд зовнішнього тестування як необхідного інструменту незалежної верифікації, що дозволяє виявити дефекти, які ігноруються через надмірну обізнаність із внутрішньою архітектурою системи. Ефективність такого підходу базується на застосуванні методів чорної скриньки, або Black-box testing, де тестувальник моделює поведінку реального користувача без використання знань про програмний код.

Методологічна перевага зовнішнього аудиту полягає у диверсифікації тестових середовищ. Зовнішні агенції оперують значно ширшою матрицею апаратного забезпечення та конфігурацій операційних систем порівняно з внутрішніми відділами. Згідно з вимогами стандарту ISO/IEC/IEEE 29119, такий підхід забезпечує необхідну повноту тестового покриття в умовах варіативності середовищ виконання [1]. Це дозволяє виявляти специфічні дефекти продуктивності на нішевих пристроях та помилки у бізнес-логіці, які внутрішня команда може помилково вважати допустимими технічними обмеженнями.

Особливе місце у структурі зовнішнього тестування посідає аудит безпеки. Залучення незалежних експертів забезпечує неупереджену перевірку системи на вразливість до атак класу SQL-ін’єкцій, міжсайтового скриптингу, відомого як Cross-Site Scripting, та порушень механізмів автентифікації. Результати такого аналізу мають високий рівень валідності для міжнародних сертифікаційних органів, серед яких ISO або PCI DSS. Це безпосередньо впливає на комерційну привабливість продукту та рівень довіри кінцевих споживачів.

Економічна доцільність зовнішнього тестування обґрунтовується класичною моделлю вартості помилки Баррі Боема, згідно з якою витрати на усунення дефекту зростають експоненціально на кожному наступному етапі життєвого циклу розробки [2]. Залучення зовнішнього ресурсу на етапі підготовки фінального кандидата на випуск, або Release Candidate, діє як критичний фільтр, що запобігає потраплянню помилок у продуктивне середовище.

Таким чином, інтеграція зовнішнього тестування у загальну стратегію забезпечення якості трансформує витрати з категорії операційних видатків у стратегічні інвестиції. Це гарантує стабільність цифрових активів, мінімізує ризики репутаційних втрат та забезпечує високу якість програмного продукту у висококонкурентному ринковому середовищі.

Список використаних джерел:

1. ISO/IEC/IEEE 29119-1:2022. Software and systems engineering — Software testing — Part 1: General concepts. International Organization for Standardization, 2022.
2. Boehm B. W. Software Engineering Economics. Prentice-Hall, 1981. 767 p.

УДК 004.7

Можаровський Я.С., здобувач
Концидайло А.М., ст. викладач
Державний університет «Житомирська політехніка»

ТЕСТУВАННЯ БЕЗПЕКИ ВЕБДОДАТКІВ

У сучасних умовах глобальної цифровізації безпека вебдодатків стає фундаментальним аспектом життєздатності інформаційних систем. Оскільки вебтехнології є основним інтерфейсом взаємодії між користувачем та корпоративними даними, вони перетворюються на першочергову ціль для кіберзлочинців. Проблема захисту ускладнюється постійним оновленням методів атак та впровадженням гнучких методологій розробки, де швидкість часто переважає над безпекою [1]. Науковий підхід до тестування передбачає побудову комплексної стратегії верифікації захисних механізмів на кожному етапі створення програмного забезпечення.

Центральним елементом такої стратегії є ідентифікація критичних вразливостей, що входять до міжнародних рейтингів. Особливу увагу приділяють механізмам контролю доступу та автентифікації, оскільки їх компрометація призводить до повного захоплення облікових записів. Окрім цього, критичне значення має перевірка вхідних даних на предмет можливості проведення ін’єкцій коду, що дозволяють зловмисникам маніпулювати базами даних або виконувати довільні команди на сервері [2]. Ефективність тестування залежить від поєднання методів аналізу коду та зовнішнього сканування системи.

Сучасна парадигма безпеки вимагає впровадження автоматизованих інструментів у конвеєр безперервної інтеграції. Статичний аналіз дозволяє виявляти потенційно небезпечні конструкції на етапі написання, тоді як динамічне сканування аналізує поведінку додатка під час його виконання. Проте автоматизація не здатна замінити експертне ручне тестування на проникнення, яке залишається єдиним способом виявлення складних логічних вразливостей. Логічні помилки в бізнес-процесах часто залишаються непомітними для алгоритмічних сканерів, що потребує залучення фахівців з етичного хакінгу.

На завершення варто підкреслити, що тестування безпеки повинно мати циклічний характер. Постійна поява нових методів експлуатації робить одноразовий аудит неефективним у довгостроковій перспективі. Тільки через інтеграцію культури безпеки у процес розробки та регулярну перевірку стійкості системи можна досягти стабільного рівня захищеності цифрових активів [3]. Таким чином, наукове обґрунтування методів тестування є невід'ємною частиною створення сучасного та надійного програмного продукту.

Список використаних джерел:

1. Stallings W. Computer Security: Principles and Practice / William Stallings, Lawrie Brown. – London: Pearson Education, 2021. – 816 с.
2. Stuttard D. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws / Dafydd Stuttard, Marcus Pinto. – Indianapolis: John Wiley & Sons, Inc., 2011. – 912 с.
3. Engebretson P. The Basics of Hacking and Penetration Testing / Patrick Engebretson. – Waltham: Syngress, 2013. – 238 с.

УДК 004

Ковалюк І.Е., здобувач
Любченко Д.В., викладач каф. комп'ютерних наук
Державний університет «Житомирська політехніка»

ОПТИМІЗАЦІЯ ВІДОБРАЖЕННЯ ГЕОПРОСТОРОВИХ ДАНИХ У МОБІЛЬНИХ ЗАСТОСУНКАХ ДЛЯ МОНІТОРИНГУ СУДЕН

Відображення великих масивів геопросторових даних у мобільних застосунках є однією з ключових технічних проблем сучасної розробки. У контексті моніторингу морських суден картографічний модуль одночасно обробляє тисячі динамічних об'єктів — позиції суден, маршрутні треки, портові зони та навігаційні попередження. Фреймворк MapKit, що є стандартним картографічним рішенням для платформи iOS, надає потужний інструментарій для роботи з геоданими, однак без цілеспрямованої оптимізації демонструє суттєве зниження продуктивності при одночасному рендерингу 500 і більше анотацій [1].

Основна проблема полягає у тому, що стандартна реалізація MKAnnotationView у MapKit не передбачає механізмів автоматичного пулінгу та кластеризації об'єктів. При завантаженні AIS-даних реального трафіку для акваторії середньої інтенсивності кількість одночасно активних суден може перевищувати 2000 одиниць, що призводить до падіння частоти кадрів нижче 20 FPS та підвищеного енергоспоживання пристрою [2, 3]. Дослідження у галузі мобільної картографії підтверджують, що деградація UX при відображенні більше 300 некластеризованих маркерів є критичною з точки зору сприйняття користувача [4].

Метою роботи є розробка та дослідження методів оптимізації відображення геопросторових даних у iOS застосунку для моніторингу морських суден на основі фреймворку MapKit. Для досягнення мети вирішувались такі завдання: аналіз існуючих алгоритмів кластеризації геоданих; дослідження механізму повторного використання анотацій (annotation reuse); реалізація динамічної LOD-стратегії (Level of Detail) залежно від масштабу карти; оцінка продуктивності різних підходів за метриками FPS, використання пам'яті та часу відгуку [5].

У роботі досліджено та реалізовано три основні підходи до оптимізації. По-перше, застосовано алгоритм кластеризації на основі grid-розбиття географічного простору з динамічним розміром комірок залежно від рівня зуму карти — це дозволило скоротити кількість видимих анотацій до керованих 50–80 об'єктів на екрані [6]. По-друге, впроваджено механізм пулінгу анотацій через стандартний API dequeueReusableAnnotationView(withIdentifier:), аналогічний підходу UITableView, що зменшило кількість операцій виділення пам'яті на 73% [7]. По-третє, реалізовано LOD-стратегію: на низьких рівнях зуму відображаються спрощені іконки суден, на високих — деталізовані об'єкти з курсом, швидкістю та статусом [8].

Порівняльне тестування проводилось на пристроях iPhone 13 та iPhone SE (2022) із набором реальних AIS-даних, що містив 3 500 суднових позицій. Без оптимізації середня частота кадрів під час скролінгу карти складала 18 FPS при піковому споживанні пам'яті 310 МБ. Після впровадження комплексу запропонованих методів показник FPS зріс до 58–60, а споживання пам'яті скоротилось до 87 МБ, тобто на 72%. Час первинного завантаження шару суден зменшився з 4.2 с до 0.9 с завдяки асинхронній обробці геоданих у фоновому потоці з використанням Swift Concurrency (async/await) [9].

Проведене дослідження підтверджує, що комплексне застосування кластеризації, пулінгу анотацій та LOD-стратегії дозволяє досягти плавного відображення геопросторових даних навіть при масштабах морського трафіку реального часу. Запропонований підхід є універсальним та може бути адаптований для інших галузей, де необхідне відображення великої кількості динамічних об'єктів на карті, — зокрема, для систем моніторингу авіаційного трафіку, логістичних платформ та застосунків екстреного реагування [10]. Результати роботи впроваджено у розроблений iOS застосунок для моніторингу морських суден і підтверджено актами тестування.

Список використаних джерел:

1. Apple Inc. MapKit — Annotating a Map [Електронний ресурс]. – Apple Developer Documentation, 2023. – Режим доступу: https://developer.apple.com/documentation/mapkit/annotating_a_map
2. Hu J. Performance Optimization Strategies for Large-Scale Geospatial Data Rendering on Mobile Platforms / J. Hu, R. Chen // Journal of Mobile Computing. – 2022. – Vol. 14, № 3. – P. 45–61.
3. AIS Live. Global AIS Traffic Statistics 2023 [Електронний ресурс]. – Режим доступу: https://www.aislive.com/statistics
4. Nielsen J. Mobile UX: Performance Thresholds for Map Interactions / J. Nielsen // Nielsen Norman Group Report. – 2021. – P. 12–19.

5. Finley T. Clustering Algorithms for Geographic Data / T. Finley // ACM SIGSPATIAL. – 2020. – Vol. 8, № 2. – P. 33–50.
6. Zhang W. Grid-Based Spatial Clustering for Real-Time Mobile Mapping / W. Zhang, L. Park // Proceedings of IEEE MDM. – 2021. – P. 112–120.
7. Apple Inc. MKAnnotationView – dequeueReusableAnnotationView [Електронний ресурс]. – Apple Developer Documentation, 2023. – Режим доступу: https://developer.apple.com/documentation/mapkit/mkmapview/1452045-dequeuereusableannotationview
8. Сисоєв О. М. Рівні деталізації у геоінформаційних системах: методи і реалізація / О. М. Сисоєв. – Харків: ХНУРЕ, 2020. – 198 с.
9. Apple Inc. Swift Concurrency: Async/Await [Електронний ресурс]. – Apple Developer Documentation, 2023. – Режим доступу: https://developer.apple.com/documentation/swift/concurrency
10. Goodchild M. F. Geographic Information Science and the Era of Big Data / M. F. Goodchild // Annals of the American Association of Geographers. – 2022. – Vol. 112, № 1. – P. 1–12.

УДК 004.7

Ярмоленко А.М., здобувач
Талавер О.В., асистент кафедри
Державний університет «Житомирська політехніка»

АРХІТЕКТУРНИЙ АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ SFU-СЕРВЕРА ДЛЯ ПОБУДОВИ СИСТЕМИ ВІДЕОКОНФЕРЕНЦЗВ'ЯЗКУ НА ОСНОВІ WebRTC

Розвиток дистанційних форм комунікації в умовах цифровізації суспільства зумовлює зростаючий попит на надійні та масштабовані системи відеоконференцзв'язку. Особливої актуальності ця проблематика набула у зв'язку із широким впровадженням дистанційного навчання, віддаленої роботи та онлайн-взаємодії у різних галузях. Водночас більшість існуючих комерційних рішень не задовольняють вимог щодо конфіденційності даних, гнучкості налаштування та економічної доступності для освітніх і некомерційних організацій. У цьому контексті актуальним є дослідження можливостей побудови власної інфраструктури відеозв'язку на основі відкритих технологій.

Протокол WebRTC (Web Real-Time Communication) є відкритим стандартом, що забезпечує передачу медіаданих у реальному часі безпосередньо між браузерами без потреби у сторонніх плагінах [1]. Однак при збільшенні кількості учасників сеансу зв'язку виникають суттєві обмеження, пов'язані з топологією з'єднань. Виділяють три основні архітектурні підходи до організації багатоточкових відеоконференцій: Mesh (p2p), MCU (Multipoint Control Unit) та SFU (Selective Forwarding Unit) [2].

У топології Mesh кожен учасник встановлює пряме з'єднання з кожним іншим, що призводить до експоненційного зростання навантаження на клієнтські пристрої: при n учасниках кожен надсилає (n−1) потоків і приймає (n−1) потоків. На практиці така схема є прийнятною лише для 2–3 учасників. MCU сервер натомість отримує всі потоки, декодує, мікширує і надсилає єдиний вихідний потік кожному учаснику, що знімає навантаження з клієнта, але вимагає значних обчислювальних ресурсів на стороні сервера та вносить затримку через перекодування.

SFU є компромісним рішенням: сервер отримує медіапотоки від кожного учасника і вибірково перенаправляє їх іншим без декодування та мікшування. Це дозволяє суттєво знизити серверне навантаження порівняно з MCU, зберігши при цьому масштабованість, недосяжну для Mesh-топології. Затримка при такому підході мінімальна, оскільки перекодування відсутнє. Саме SFU-архітектура є домінуючою у сучасних комерційних та відкритих рішеннях для відеоконференцзв'язку.

Серед актуальних SFU-реалізацій з відкритим кодом виділяють mediasoup, LiveKit та Janus [3]. Порівняльний аналіз за ключовими параметрами наведено в таблиці 1.

Таблиця 1

Порівняльний аналіз SFU-рішень

Критерій	**mediasoup**	**LiveKit**	**Janus**
Мова реалізації	*C++ / Node.js*	*Go*	*C*
Підтримка SimulCast	*Так*	*Так*	*Так*
Вбудований сигналінг	*Ні*	*Так*	*Частково*
Гнучкість API	*Висока*	*Середня*	*Середня*
Складність інтеграції	*Висока*	*Низька*	*Середня*
Документація	*Детальна*	*Детальна*	*Достатня*

За результатами аналізу для реалізації у складі дипломного проєкту обрано бібліотеку mediasoup. Ключовими чинниками вибору є: висока продуктивність завдяки C++-ядру, повний контроль над логікою сигналізації, відсутність зайвих абстракцій та широкі можливості інтеграції з Node.js-бекендом. Попри вищий поріг входу порівняно з LiveKit, mediasoup надає розробнику максимальну гнучкість у побудові власної архітектури.

Таким чином, проведений аналіз підтверджує доцільність застосування SFU-архітектури на основі mediasoup для побудови масштабованої системи відеоконференцзв'язку. Подальші дослідження передбачають практичну реалізацію серверної та клієнтської частин системи, тестування під навантаженням та оцінку якості медіапотоку за метриками RTT, jitter та втрати пакетів.

Список використаних джерел:

1. MDN Web Docs. WebRTC API [Електронний ресурс]. MDN. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API.
2. Levent-Levi T. WebRTC Multiparty Video Architecture [Електронний ресурс]. BlogGeek.me. Режим доступу: https://bloggeek.me/webrtc-multiparty-video/.
3. Mediasoup. mediasoup Documentation [Електронний ресурс]. mediasoup. Режим доступу: https://mediasoup.org/documentation/.

УДК 004.4

Тарасюк М.О., здобувач
Чижмотря О.В., ст. викладач
Державний університет «Житомирська політехніка»

АВТОМАТИЗАЦІЯ ОНЛАЙН-БРОНЮВАННЯ У ВЕБСЕРВІСАХ СФЕРИ BEAUTY-ПОСЛУГ

На сьогодні значна кількість студій краси продовжує використовувати ручний спосіб ведення записів через телефонні дзвінки, соціальні мережі або месенджери. Такий формат роботи часто створює додаткові труднощі як для клієнтів, так і для адміністраторів. Постійна обробка повідомлень, уточнення вільного часу та коригування графіка можуть призводити до помилок під час бронювання, накладок у розкладі або втрати частини заявок. Крім того, користувачі не завжди мають можливість швидко отримати актуальну інформацію про доступні послуги чи вільні години для запису. Одним із найбільш ефективних способів вирішення цієї проблеми є створення вебплатформи з функцією автоматизованого онлайн-бронювання. Така система дозволяє користувачам самостійно переглядати список доступних послуг, ознайомлюватися з інформацією про майстрів, обирати зручну дату та час візиту, а також підтверджувати запис без необхідності додаткового спілкування з адміністратором. Це значно спрощує процес взаємодії між клієнтом і студією та дозволяє скоротити час обробки заявок. Запропонований підхід передбачає використання серверної системи для автоматичного оновлення графіків роботи майстрів та синхронізації бронювань у режимі реального часу. Завдяки цьому користувачі бачать лише актуальну інформацію про доступні часові проміжки, що допомагає уникнути дублювання записів. Додатково така система може надсилати автоматичні повідомлення або нагадування про майбутній візит, зміну часу запису чи його скасування. Це дозволяє зменшити кількість пропущених записів та покращити організацію робочого процесу.

Особливу увагу має бути приділено зручності користувацького інтерфейсу. Для вебсервісів у сфері beauty-послуг важливим є простий та зрозумілий дизайн, оскільки більшість користувачів здійснює запис саме через мобільні пристрої. Адаптивний інтерфейс забезпечує коректну роботу системи на смартфонах, планшетах та персональних комп'ютерах, що робить використання сервісу більш комфортним. Також важливо забезпечити швидке завантаження сторінок та зрозумілу навігацію між розділами платформи, оскільки це безпосередньо впливає на зручність користування вебсервісом.Технічна реалізація вебплатформи може базуватись на сучасних вебтехнологіях для створення клієнтської та серверної частин застосунку. Для зберігання інформації про користувачів, послуги, графіки роботи та бронювання використовуються системи керування базами даних. Також важливим аспектом є забезпечення безпеки даних користувачів шляхом реалізації авторизації, перевірки введених даних та контролю доступу до системи. Використання сучасних засобів захисту дозволяє забезпечити конфіденційність персональної інформації та стабільність роботи вебплатформи.

Практичне використання автоматизованих систем онлайн-запису дозволяє значно покращити організацію роботи beauty-сервісів, зменшити кількість помилок під час бронювання та забезпечити більш швидку й зручну взаємодію між студією та клієнтами. Крім того, цифровізація процесу запису сприяє підвищенню конкурентоспроможності студій краси та покращенню якості обслуговування користувачів.

Використання цифрових систем онлайн-бронювання є важливим напрямом розвитку сучасних beauty сервісів. Автоматизація процесу запису дозволяє оптимізувати роботу студій краси, знизити навантаження на адміністраторів та забезпечити комфортніші умови для користувачів. Застосування сучасних вебтехнологій сприяє підвищенню якості сервісу та покращенню ефективності організації роботи у сфері beauty-послуг. автоматизовані рекомендації щодо популярних процедур. Це дозволяє не лише покращити взаємодію з клієнтами, а й забезпечити більш ефективне управління діяльністю студії та аналіз попиту на послуги.

Додатковою перевагою впровадження подібних вебплатформ є можливість централізованого зберігання інформації про клієнтів, їхні вподобання та історію відвідувань. Це дозволяє майстрам швидше адаптувати послуги під потреби користувача та забезпечувати більш персоналізований підхід до обслуговування. Наявність такої інформації також спрощує повторний запис клієнтів та покращує загальний рівень взаємодії між користувачем і студією. Перспективним напрямом розвитку подібних систем є подальше розширення функціоналу AI-консультанта. Зокрема, можливою є інтеграція інструментів аналізу зображень для підбору стилю макіяжу на основі фотографії користувача, а також реалізація систем рекомендацій з урахуванням актуальних трендів beauty-індустрії. Це дозволить зробити вебплатформи більш інтерактивними, підвищити рівень персоналізації сервісу та покращити якість надання послуг.

УДК 004.4

Тарасюк М.О., здобувач
Чижмотря О.В., ст. викладач
Державний університет «Житомирська політехніка»

ВЕБПЛАТФОРМА СТУДІЇ ВІЗАЖИСТІВ З AI-КОНСУЛЬТАНТОМ

Сучасний розвиток цифрових технологій суттєво впливає на сферу надання послуг, зокрема й на індустрію краси. Вебресурси студій візажистів поступово стають не лише засобом представлення інформації про послуги, а й повноцінним інструментом взаємодії з клієнтами. Проте більшість існуючих платформ виконують переважно інформаційну функцію та не забезпечують достатнього рівня автоматизації процесів обслуговування. Користувачі можуть переглядати приклади робіт, ознайомлюватися з цінами або контактною інформацією, однак процес підбору відповідного образу та запису на послугу часто залишається складним і потребує додаткового спілкування з адміністратором. У результаті клієнт змушений самостійно аналізувати, який тип макіяжу найбільше відповідає його потребам, формату заходу чи особливостям зовнішності. Особливо актуальною ця проблема є для нових користувачів, які не мають достатнього досвіду у виборі послуг у сфері візажу. Відсутність персоналізованих рекомендацій може призводити до невдалого вибору образу або непорозумінь між клієнтом і майстром. Крім цього, значна частина часу адміністраторів витрачається на обробку однотипних запитів щодо вартості послуг, підбору часу для запису, уточнення деталей макіяжу чи рекомендацій стосовно стилю образу. Це створює додаткове навантаження на персонал студії та знижує ефективність організації роботи. Також користувачі часто змушені переходити до сторонніх месенджерів для узгодження запису, що робить процес взаємодії менш зручним і більш тривалим. Для вирішення зазначених проблем доцільним є створення вебплатформи студії візажистів з інтегрованим AI консультантом. Основною метою такої системи є автоматизація процесу взаємодії між клієнтом і студією, підвищення якості консультаційної підтримки та спрощення процедури вибору послуг. AI-консультант дозволяє аналізувати текстові запити користувачів та формувати персоналізовані рекомендації щодо типу макіяжу, стилю образу та відповідних послуг студії. Система може враховувати формат події, побажання клієнта, кольоротип зовнішності та інші індивідуальні особливості. Завдяки цьому користувач отримує більш точні рекомендації та може швидше визначитися з вибором послуги.

Важливою перевагою AI-консультанта є можливість роботи в режимі реального часу через чат інтерфейс. Користувач може поставити запитання щодо послуг, вартості, особливостей макіяжу або підготовки до процедури та отримати швидку відповідь без участі адміністратора. Це дозволяє значно скоротити час очікування відповіді та підвищити доступність сервісу. Крім того, автоматизована консультаційна підтримка дає можливість студії обслуговувати більшу кількість клієнтів одночасно без збільшення навантаження на персонал. Використання AI-технологій також сприяє покращенню користувацького досвіду, оскільки взаємодія із системою стає більш зрозумілою, швидкою та зручною.

Окрему увагу в роботі приділено автоматизації процесу бронювання послуг. Запропонована система передбачає інтеграцію календаря записів із графіком роботи майстрів, що дозволяє користувачам самостійно переглядати доступний час та обирати потрібного спеціаліста. Після вибору послуги клієнт може підтвердити запис без необхідності телефонного дзвінка або додаткового листування. Такий підхід дозволяє зменшити кількість помилок під час формування записів, уникнути накладання часу бронювання та оптимізувати роботу студії. Для користувачів це забезпечує більш простий та швидкий процес взаємодії із сервісом, а для персоналу - зменшення адміністративного навантаження. Архітектура вебплатформи побудована за принципом клієнт-серверної взаємодії із використанням сучасних вебтехнологій. Клієнтська частина відповідає за відображення адаптивного інтерфейсу та забезпечення зручної взаємодії користувача із системою на різних типах пристроїв. Серверна частина здійснює обробку даних, авторизацію користувачів, роботу AI-консультанта та взаємодію з базою даних. Для реалізації AI функціоналу використовується інтеграція з сервісами штучного інтелекту, які забезпечують обробку природномовних запитів та генерацію рекомендацій у зрозумілому форматі. Додатково система може накопичувати інформацію про попередні записи, популярні послуги та оцінки клієнтів для подальшого вдосконалення персоналізації сервісу. Технічна реалізація вебплатформи передбачає використання сучасних фреймворків веброзробки, серверних API та систем керування базами даних. Водночас важливим аспектом є забезпечення стабільної роботи сервісу при одночасній роботі великої кількості користувачів, а також захист персональних даних клієнтів. Серед основних обмежень системи можна виділити залежність якості рекомендацій від точності введених даних користувача та можливі труднощі у складних індивідуальних випадках, де може знадобитися консультація безпосередньо з майстром. Незважаючи на це, використання AI-консультанта у вебплатформі студії візажистів є перспективним рішенням, яке дозволяє підвищити ефективність роботи студії, автоматизувати ключові процеси та покращити якість обслуговування клієнтів.

УДК 654.15:364-7

Грицюк В.І., магістр
Державний університет «Житомирська політехніка»

ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА УПРАВЛІННЯ ТАРИФНИМИ ПАКЕТАМИ ТА НАДАННЯМ ПОСЛУГ МОБІЛЬНОГО ЗВ'ЯЗКУ

В умовах стрімкого розвитку телекомунікаційної галузі мобільний зв’язок є однією з ключових складових інформаційної інфраструктури суспільства. Люди всього світу щодня відправляють десятки тисяч смс, роблять мільйони дзвінків, здійснюють сотні мільйонів пошукових запитів через мобільні мережі. Постійне зростання кількості абонентів, різноманіття тарифних пакетів та послуг мобільного зв’язку зумовлюють необхідність використання ефективних інформаційно-аналітичних систем для їх управління та оптимізації.

Метою роботи є розробка та дослідження інформаційно-аналітичної системи управління тарифними пакетами та наданням послуг мобільного зв’язку, яка забезпечує підтримку прийняття управлінських рішень на основі аналізу даних про споживання послуг абонентами.

Проведений аналіз предметної області, тарифних планів операторів мобільного зв’язку, існуючих підходів до управління тарифними планами та інформаційних систем, що використовуються операторами мобільного зв’язку в результаті якого виявлено переваги та недоліки існуючих систем, визначено основні функціональні вимоги до інформаційно-аналітичної системи.

Головні характеристики такої системи — доступність та гнучкість. Значної уваги потребує автоматизація процесів формування, модифікація та контроль тарифних пакетів, аналіз поведінки абонентів, прогнозування попиту на послуги та оцінювання ефективності тарифних рішень. Запропонована архітектура інформаційно-аналітичної системи, рис.1, яка базується на використанні сучасних технологій обробки даних, реляційних баз даних та аналітичних модулів. Реалізовано механізми збору, зберігання та обробки інформації щодо тарифів, послуг та активності абонентів. Особливу увагу приділено модулю аналітики, який дозволяє здійснювати сегментацію абонентів, виявляти закономірності споживання послуг та формувати рекомендації щодо оптимізації тарифних пакетів.

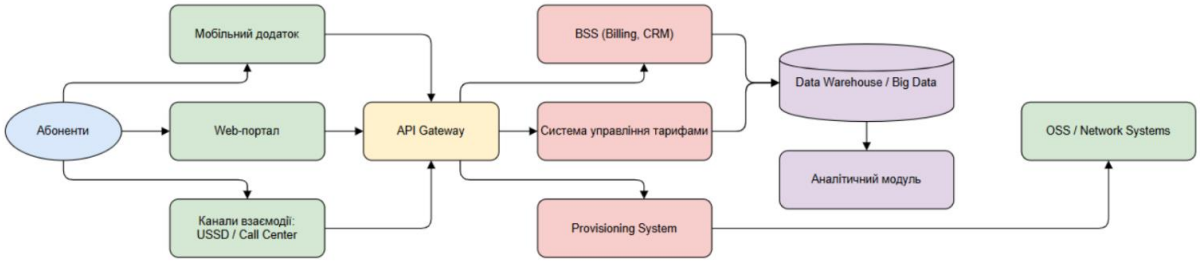

Рис. 1. Архітектура інформаційно-аналітичної системи управління тарифними пакетами та наданням послуг мобільного зв'язку

Передбачається, що система збирає інформацію про кількість одиниць спожитих послуг і формує вартість кожної операції. Система виставляє рахунок користувачеві за принципом передплати або післяплати. У разі передоплати необхідно контролювати стан рахунку, блокувати надання послуг, якщо рахунок не поповнений, сповіщати користувачів.

Для роумінгу використовується облік одразу двох операторів (домашній і гостьовий), які взаємодіють між собою.

У результаті впровадження розробленої системи підвищується гнучкість тарифної політики оператора мобільного зв’язку, зменшуються витрати на адміністрування послуг та покращується якість обслуговування абонентів з урахуванням європейського регулювання. Практичне значення роботи полягає у можливості використання розробленої системи в діяльності операторів мобільного зв’язку для підтримки стратегічного та оперативного управління.

Список використаних джерел:

1. A basic study and review of UML versions and UML tools. 2022. URL: https://medium.com/@okworchukwuebube/a-basic-study-and-review-of-uml-ersionsand-uml-tools-af2c85e606 (дата звернення 04.04.2026 р.).

2. UML Activity Diagram Tutorial URL: https://www.lucidchart.com/pages/umlactivity-diagram (дата звернення 04.04.2026 р.).

3. Білінгові системи: як працює діджитал у телеком-бізнесі. URL: https://wezom.com.ua/ua/blog/billingovye-sistemy-kak-rabotaet-didzhital-v-telekom-biznese (дата звернення 04.04.2026 р.).

4. Нечитайло Б. Роль персоналізованих цінових пропозицій у підвищенні конкурентоспроможності телекомунікаційних компаній. Економіка та суспільство. 2024. № 66.. С.1-7. DOI: https://doi.org/10.32782/2524-0072/2024-66-30 (дата звернення 04.04.2026 р.).

5. Подзігун С.М., Білошкурська Н.В., Омельяненко В.А., Корнєєва О.О. Аналіз конкурентоспроможності мобільних операторів України на основі бенчмаркінгу: оцінка ринку та формування інноваційного бізнес-середовища. Здобутки економіки: перспективи та інновації. 2024. №11. С.1-27. DOI: https://doi.org/10.5281/zenodo.13937048 (дата звернення 04.04.2026 р.).

УДК 004

Держанівський А.О., здобувач, гр. ЗІПЗ-22-1, ФІКТ
Піонтківський В.І., асист. кафедри ІПЗ, ФІКТ
Державний університет «Житомирська політехніка»

ОПИС ЗАСТОСУВАННЯ ПРОДУКТУ SUPABASE ДЛЯ ВИКОРИСТАННЯ В МОБІЛЬНІЙ РОЗРОБЦІ

Supabase [1] є сучасною хмарною платформою класу Backend-as-a-Service (BaaS), яка призначена для швидкого створення серверної частини веб та мобільних застосунків. Платформа надає готовий набір інструментів для роботи з базою даних, авторизацією користувачів, зберіганням файлів, API та синхронізацією даних у реальному часі. Основною перевагою Supabase є можливість значно скоротити час розробки мобільного застосунку завдяки використанню готової інфраструктури замість створення власного серверного рішення.

В основі Supabase лежить реляційна система керування базами даних PostgreSQL, яка є однією з найпоширеніших та найнадійніших СУБД у світі. Використання PostgreSQL дозволяє розробникам створювати складні структури даних, налаштовувати зв'язки між таблицями, використовувати SQL-запити та забезпечувати високу продуктивність роботи застосунку. Для мобільної розробки це особливо важливо, оскільки більшість сучасних застосунків потребують централізованого зберігання даних та синхронізації між пристроями.

Однією з ключових функцій Supabase є система аутентифікації користувачів. Платформа підтримує реєстрацію через електронну пошту та пароль, а також інтеграцію із зовнішніми сервісами авторизації, такими як Google, Apple, GitHub та іншими. У мобільних застосунках це дозволяє швидко реалізувати функції входу та реєстрації без необхідності створення власної серверної логіки для управління користувачами.

Ще однією важливою можливістю платформи є підтримка технології Realtime. Вона забезпечує миттєву синхронізацію даних між клієнтськими пристроями та сервером. Це особливо корисно для створення чатів, систем обміну повідомленнями, POS-систем, сервісів бронювання або будь-яких застосунків, де інформація повинна оновлюватися в реальному часі. Наприклад, у системі управління магазином продавці можуть одразу бачити нові замовлення або зміни залишків товарів без необхідності вручну оновлювати сторінку чи виконувати повторні запити до сервера.

Supabase також надає сервіс Storage для зберігання файлів. У мобільних застосунках це дозволяє завантажувати фотографії, документи, відео або фото користувачів без використання сторонніх файлових сервісів. Файли можуть зберігатися у спеціальних контейнерах, а доступ до них контролюється через систему прав доступу. Наприклад, у соціальному застосунку користувач може завантажити фотографію профілю, а у POS-системі — фото товару або електронний чек.

Однією з переваг Supabase є автоматичне створення REST API та GraphQL API на основі структури бази даних. Це означає, що після створення таблиць у PostgreSQL платформа автоматично генерує API для роботи з даними. Мобільний застосунок може виконувати операції створення, читання, оновлення та видалення записів без написання окремого серверного коду.

Для забезпечення безпеки даних Supabase використовує механізм Row Level Security (RLS). Він дозволяє задавати правила доступу до окремих записів у таблицях бази даних. Наприклад, користувач мобільного застосунку може бачити лише власні замовлення або особисті повідомлення, тоді як адміністратор має доступ до всіх записів. Це особливо важливо для комерційних систем, де необхідно забезпечити конфіденційність даних та контроль доступу відповідно до ролей користувачів.

Supabase підтримує офіційні SDK для різних платформ мобільної розробки, зокрема Kotlin для Android, Swift для iOS та Flutter/Dart для кросплатформної розробки. Це дозволяє інтегрувати платформу безпосередньо у мобільний застосунок та працювати з базою даних, авторизацією й файлами через зручні програмні інтерфейси.

У сучасній мобільній розробці Supabase часто використовується для створення CRM-систем, POS-застосунків, соціальних мереж, месенджерів, сервісів доставки, систем бронювання та маркетплейсів. Завдяки поєднанню бази даних, авторизації, файлового сховища та API в одному сервісі платформа дозволяє зосередитися на створенні інтерфейсу та бізнес-логіки застосунку, не витрачаючи значний час на налаштування серверної інфраструктури.

Таким чином, Supabase є ефективним рішенням для мобільної розробки, яке поєднує простоту використання, високу швидкість розгортання та широкі можливості масштабування. Використання цієї платформи дозволяє значно скоротити час створення мобільного застосунку, підвищити надійність роботи сервера та забезпечити безпечну взаємодію користувачів із даними.

Список використаних джерел:

1. Supabase. URL: https://supabase.com.

УДК 004.78

Платонов М.С., здобувач
Лобанчикова Н.М., к.т.н., доцент
Державний університет «Житомирська політехніка»

ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ДИЗАЙНЕРСЬКОГО ОДЯГУ НА БАЗІ ASP.NET CORE MVC

Стрімкий розвиток електронної комерції у fashion-індустрії зумовлює нагальну потребу у створенні спеціалізованих вебплатформ, здатних ефективно презентувати продукцію та автоматизувати процеси продажу. Локальні дизайнерські бренди, зокрема «ELARIUM», стикаються з суттєвими обмеженнями при використанні соціальних мереж як основного каналу збуту через відсутність автоматизованого обліку залишків, неможливість повноцінної фільтрації товарів, а також відсутність єдиної клієнтської бази. Разом з тим, використання універсальних CMS (WordPress, Shopify) супроводжується надмірною технічною складністю та обмеженнями кастомізації. Тому розробка спеціалізованого інтернет-магазину на базі ASP.NET Core MVC є актуальним та практично значущим завданням [1].

Метою роботи є проєктування та програмна реалізація інтернет-магазину дизайнерського одягу «ELARIUM» для автоматизації процесів презентації колекцій, управління каталогом та оформлення замовлень. Об'єктом дослідження є процеси розробки програмного забезпечення для електронної комерції у сфері fashion-retail; предметом є моделі, методи, архітектурні підходи та програмні засоби розробки вебзастосунку на платформі .NET 8.0.

Для досягнення мети проведено порівняльний аналіз існуючих аналогів, зокрема аналіз мінімалістичного сайту plaa4444.com демонструє високу швидкість, але позбавлений галереї та пошуку; optimusgang.com дозволив визначити, що сайт побудовано на CMS і має обмежену інформативність карток товарів; міжнародної платформи misbhv.com дозволив виділити реалізацію преміальної візуальної презентації, проте це є технічно надлишковим для молодого локального бренду. Системний аналіз виявив потребу у розробці «легкого», «швидкого» та «кастомізованого рішення» з повноцінною галереєю [2].

В основу системи «ELARIUM» покладено архітектурний патерн MVC (Model-View-Controller) на базі фреймворку ASP.NET Core 8.0. Обрана монолітна багаторівнева архітектура включає шар доступу до даних (Entity Framework Core + PostgreSQL), шар бізнес-логіки (контролери та сервіси) та шар представлення (Razor Views + Bootstrap 5). Такий вибір зумовлений вимогами до SEO-оптимізації, що найкраще забезпечується серверним рендерингом (SSR) та необхідністю високої швидкодії при обслуговуванні десктопних галерей.

Система реалізує три основні підсистеми. Перша – клієнтський модуль (HomeController): каталог товарів із динамічною фільтрацією за категорією та текстовим пошуком через IQueryable-запити до PostgreSQL, детальна сторінка товару з інтерактивною галереєю та функцією масштабування (Zoom 2.5x), реалізованою засобами Vanilla JavaScript без зовнішніх бібліотек. Друга – сесійний кошик (CartController): серіалізація стану кошика у JSON із збереженням у Distributed Memory Cache (2 год.), підтримка різних розмірів одного товару як окремих позицій. Третя – адміністративна панель (ProductsController): повний CRUD над товарами, захищений атрибутом [Authorize(Roles = "Admin")], алгоритм двоетапного множинного завантаження фотографій галереї.

Модель даних включає три сутності: Product (Id, Name, Description, Price, ImageUrl, Sizes, Category, IsAvailable, Images), ProductImage (Id, Url, ProductId з CASCADE-видаленням) та CartItem – денормалізована сесійна модель без таблиці в БД. Схема PostgreSQL еволюціонувала через шість міграцій EF Core (підхід Code First). Eager Loading через .Include(p => p.Images) забезпечує завантаження галереї за один SQL-запит, уникаючи проблеми N+1 [3].

Безпека реалізована засобами ASP.NET Core Identity: хешування паролів (PBKDF2), рольова модель доступу (RBAC), захист форм від CSRF через [ValidateAntiForgeryToken], сесійні cookie позначено як HttpOnly та IsEssential. Адміністративний обліковий запис ініціалізується автоматично при запуску застосунку в Program.cs, що гарантує готовність системи без ручного налаштування.

Таким чином, розроблений програмний додаток є кастомним рішенням, яке поєднує преміальну візуальну презентацію рівня enterprise-платформ із технічною легкістю та низькими вимогами до хостингу, і може бути впроваджене в комерційну діяльність бренду.

Список використаних джерел:

1. Microsoft. ASP.NET Core documentation. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/aspnet/core/ (дата звернення: 10.04.2025).
2. Документація Microsoft ASP.NET Core. URL: https://docs.microsoft.com/aspnet/core (дата звернення: 10.04.2026).
3. Smith J. Entity Framework Core in Action. 2nd ed. Manning Publications, 2022. 665 с.

УДК 004.738.5:687

Платонов М.С., здобувач
Лобанчикова Н.М., к.т.н., доц.
Державний університет «Житомирська політехніка»

ЗАСТОСУВАННЯ СУЧАСНИХ ТЕХНОЛОГІЙ ЕЛЕКТРОННОЇ КОМЕРЦІЇ У FASHION-РИТЕЙЛІ

Сучасний ринок fashion-ритейлу переживає кардинальну трансформацію під впливом цифровізації. Зміна споживчої поведінки та розвиток мобільних технологій зумовили те, що наявність власного вебресурсу стала не просто конкурентною перевагою, а обов'язковою умовою виживання комерційного бренду. Особливо гостро ця проблема постає перед локальними дизайнерськими марками, які здебільшого покладаються на соціальні мережі як основний канал комунікації з клієнтами. Проте такий підхід має системні обмеження: алгоритмічне стиснення якості фотографій унеможливлює передачу фактури тканин, відсутня автоматизація складського обліку, а ручна обробка замовлень через месенджери неминуче призводить до помилок та втрати клієнтів [1].

Аналіз ринку рішень для електронної комерції у сегменті дизайнерського одягу виявив характерне протиріччя. З одного боку існують технічно прості, але функціонально бідні платформи (*plaa4444.com*), позбавлені галереї, пошуку та особистого кабінету. З іншого – надпотужні *enterprise*-рішення (*misbhv.com*) з глобальною локалізацією та мультивалютністю, що є економічно недоцільним для молодого локального бренду. Платформи на базі CMS (*optimusgang.com*) займають проміжну позицію, але страждають від надлишкового коду та обмежень кастомізації. Таким чином, сформована потреба у «золотій середині»: спеціалізованому рішенні з преміальною візуальною презентацією та мінімальними вимогами до ресурсів [2]. Відповіддю на цей запит стала розробка інтернет-магазину «ELARIUM» на базі ASP.NET Core MVC - платформи, яка забезпечує серверний рендеринг (SSR) для SEO-оптимізації та миттєвого відображення контенту. До технологічного стеку входять мова C# зі строгою статичною типізацією, СУБД PostgreSQL, ORM Entity Framework Core (підхід Code First), фреймворк Bootstrap 5 та Vanilla JavaScript для клієнтської частини. Вибір PostgreSQL обґрунтований підтримкою ACID-транзакцій, ефективністю JOIN-запитів при фільтрації великих каталогів та надійністю інтеграції з ASP.NET Core Identity.

Ключовою конкурентною перевагою розробленого рішення є інтерактивна десктопна галерея з функцією масштабування (Zoom 2.5x), реалізованою виключно засобами Vanilla JavaScript без підключення зовнішніх бібліотек. При наведенні курсору обчислюються відносні координати у відсотках від розмірів контейнера, які встановлюються як трансформаційна точка CSS-властивості *transform-origin*. Це дозволяє клієнту детально вивчити фактуру тканини, якість швів та фурнітуру безпосередньо в браузері без додаткових HTTP-запитів – функціонал, реалізація якого у аналогів потребує важких комерційних бібліотек.

Система безпеки реалізована на базі ASP.NET Core Identity та включає: рольову модель доступу (Гість / Адміністратор), хешування паролів за алгоритмом PBKDF2, захист від CSRF-атак через антипідробні токени ([ValidateAntiForgeryToken]), а також сесійний кошик із серіалізацією стану у JSON та збереженням у Distributed Memory Cache. Адміністративна панель автоматично захищена атрибутом [Authorize(Roles = "Admin")] на рівні контролера та недоступна для звичайних користувачів. Важливою архітектурною особливістю є автоматична ініціалізація адміністративного облікового запису при кожному запуску застосунку, що усуває необхідність ручного налаштування при розгортанні. Розроблений програмний продукт має модульну структуру та відкритий потенціал для масштабування. У наступних ітераціях заплановано: виділення сутності Category в окрему таблицю з полями управління відображенням; додавання таблиць OrderHeaders та OrderDetails для повного циклу оформлення замовлень; розширення профілю ApplicationUser (успадкування від IdentityUser) для збереження адрес доставки та перегляду історії покупок; інтеграція із платіжними шлюзами. Модульна MVC-архітектура без надлишкового коду CMS забезпечує легкість розширення без ризику порушення існуючої функціональності.

Практичне значення роботи полягає у створенні повністю працездатного програмного продукту, готового до впровадження в комерційну діяльність бренду «ELARIUM». Його використання дозволить централізувати облік товарів, автоматизувати процес наповнення каталогу (адміністратор самостійно завантажує фото без залучення IT-спеціалістів), підвищити рівень обслуговування клієнтів та забезпечити преміальну презентацію продукції в онлайн-просторі.

Список використаних джерел:

1. Kingsnorth S. Digital Marketing Strategy: An Integrated Approach to Online Marketing. 3rd ed. Kogan Page, 2022. 344 с.
2. Документація Microsoft ASP.NET Core Identity. URL: https://docs.microsoft.com/aspnet/core/security/authentication/identity (дата звернення: 10.04.2026).
3. Офіційна документація PostgreSQL. URL: https://www.postgresql.org/docs/ (дата звернення: 12.04.2026).

УДК 681.518:612.822

Янчук Е.А., магістрант
Нікітчук Т.М., к.т.н., доцент
Державний університет «Житомирська політехніка»

МЕТРОЛОГІЧНІ АСПЕКТИ МУЛЬТИМОДАЛЬНИХ ІНТЕРФЕЙСІВ МОЗОК-КОМП'ЮТЕР ІЗ ЗАСТОСУВАННЯМ МЕТОДІВ НЕЙРОФОТОНІКИ

Розвиток неінвазивних інтерфейсів мозок-комп'ютер (ІМК) на сучасному етапі стикається з проблемою недостатньої точності та стабільності зчитування інформації. Традиційні системи на основі електроенцефалографії (ЕЕГ) мають високу часову роздільну здатність, але низьку просторову точність та чутливість до артефактів. Перспективним шляхом вирішення цієї проблеми є впровадження мультимодальних систем, що поєднують ЕЕГ з методами нейрофотоніки, зокрема функціональною ближньою інфрачервоною спектроскопією (fNIRS).

Мультимодальні системи (ЕЕГ + fNIRS) – це один із найефективніших шляхів подолання обмежень неінвазивних ІМК. Вони забезпечують безпрецедентну точність зчитування, поєднуючи переваги обох технологій

Метою роботи є аналіз інформаційно-вимірювальних характеристик гібридних ІМК та обґрунтування переваг використання методів нейрофотоніки для підвищення надійності вимірювань біосигналів.

Нейрофотоніка базується на вимірюванні змін поглинання світла в ближньому інфрачервоному діапазоні, (700–900 нм) залежно від концентрації окси- та дезоксигемоглобіну в корі головного мозку. На відміну від ЕЕГ, яка вимірює електричну активність нейронів, fNIRS фіксує метаболічну відповідь, що забезпечує вищу стійкість до електромагнітних завад та м'язових артефактів.

Впровадження нейрофотонного каналу вимірювання дозволяє вирішити ряд критичних метрологічних завдань:

– зменшити кількість помилкових спрацювань системи за рахунок перехресної верифікації сигналів електричної та оптичної природи;

– підвищити просторову роздільну здатність локалізації активних зон мозку до 5-10 мм, що суттєво перевершує можливості стандартних ЕЕГ-електродів;

– забезпечити стабільну роботу вимірювальної системи в умовах динамічних навантажень та тривалого моніторингу психофізіологічного стану.

Основною метрологічною проблемою таких гібридних систем є затримка гемодинамічної відповіді (до 2-5 секунд порівняно з мілісекундною реакцією ЕЕГ). Це потребує розробки спеціалізованих алгоритмів предиктивного аналізу. Застосування методів цифрової обробки сигналів та глибокого навчання дозволяє ефективно синхронізувати ЕЕГ та fNIRS дані, фільтрувати шуми та виділяти корисний сигнал навіть в умовах інтенсивних зовнішніх завад. Зокрема, використання алгоритмів на базі згорткових нейронних мереж (CNN) дозволяє автоматизувати процес класифікації патернів мозкової активності з точністю понад 90%.

Висновки. Поєднання методів нейрофотоніки та класичної електроенцефалографії в рамках єдиної інформаційно-вимірювальної системи дозволяє створити новий клас високонадійних інтерфейсів мозок-комп'ютер. Це відкриває широкі можливості для створення більш точних асистивних технологій, медичних реабілітаційних комплексів та систем об'єктивного моніторингу стану операторів складних технічних систем, мінімізуючи ризики хибних вимірювань.

Список використаних джерел:

1. Янчук Е.А., Нікітчук Т.М., Коренівська О.Л. Інтерфейси мозок–комп'ютер: сучасні тенденції розвитку. Державний університет «Житомирська політехніка», 2024.
2. Elashmawi W.H. et al. A Comprehensive Review on Brain–Computer Interface (BCI). Applied Sciences, 2024.
3. Hossain K.M. et al. Status of Deep Learning for EEG-based Brain–Computer Interfaces. Frontiers in Computational Neuroscience, 2023.
4. Ding Y. et al. EEG-based Brain–Computer Interface Enables Real-time Noninvasive Robotic Control. Nature Communications, 2025.
5. Технології нейроінтерфейсів та їх застосування при неврологічних порушеннях. Режим доступу: https://mpclinic.com.ua/tekhnolohii-neyrointerfeysiv-ta-ikh-zastosuvannia-pry-nevrolohichnykh-porushenniakh/