DOI: 10.32347/2412-9933.2026.65.188-193
УДК 004.9:004.75:656.1

Стужук Ігор Миколайович
ORCID: https://orcid.org/0009-0004-8707-3567
Київський національний університет будівництва і архітектури, Київ, Україна
Здобувач вищої освіти кафедри інформаційних технологій
Науменко Юрій Олександрович
ORCID: https://orcid.org/0000-0002-2631-1048
Київський національний університет будівництва і архітектури, Київ, Україна
Кандидат технічних наук, доцент кафедри інформаційних технологій
Бородавка Євгеній Володимирович
ORCID: https://orcid.org/0000-0002-7476-9387
Київський національний університет будівництва і архітектури, Київ, Україна
Доктор технічних наук, професор, завідувач кафедри інформаційних технологій проєктування та прикладної математики
Азнаурян Ірина Олександрівна
ORCID: https://orcid.org/0000-0002-7085-7291
Київський національний університет будівництва і архітектури, Київ, Україна
Доцентка кафедри фізики

Історія статті:
Надійшла: 30.01.2026
Прийнята: 25.02.2026
Опублікована: 26.03.2026

РОЗРОБКА КРОСПЛАТФОРМНИХ ІНФОРМАЦІЙНИХ СИСТЕМ МІСЬКОЇ МОБІЛЬНОСТІ: АРХІТЕКТУРНІ ТА АЛГОРИТМІЧНІ РІШЕННЯ

Анотація. *Актуальність роботи зумовлена зростанням навантаження на міську транспортну інфраструктуру, необхідністю впровадження екологічно орієнтованих рішень і підвищенням вимог до універсальності й масштабованості програмних систем, що функціонують у межах концепції «розумного міста». У фокусі дослідження – сервіси організації спільних поїздок. Обґрунтовано доцільність використання кросплатформного підходу як засобу зниження витрат на розробку та супровід інформаційних систем, що призначені для різних операційних середовищ. Досліджено архітектурні та алгоритмічні аспекти розробки кросплатформних інформаційних систем міської мобільності. Запропоновано архітектуру клієнт-серверної інформаційної системи, що реалізована з використанням фреймворку .NET MAUI, який забезпечує формування єдиної кодової бази для мобільних та десктопних платформ. Розглянуто застосування патернів MVVM і Dependency Injection для підвищення модульності, підтримуваності та розширюваності системи. Формалізовано задачу пошуку попутників з використанням геопросторових і часових обмежень. Розроблено гібридний алгоритм пошуку поїздок, який поєднує геолокаційні обчислення на ґрунті сферичної метрики і текстову фільтрацію як резервний механізм у разі відмов служб визначення координат. Запропоновано рішення зменшення обчислювальної складності пошуку і забезпечення відмовостійкості підсистеми в умовах нестабільного мережевого з'єднання. Проведено експериментальну оцінку ефективності реалізації, що підтвердила високий (понад 96%) рівень повторного використання коду і прийнятні показники продуктивності на різних платформах. Отримано результати, які можуть бути використані при розробці масштабованих інформаційних систем міської мобільності та інших складних розподілених систем, орієнтованих на кросплатформну експлуатацію.*

Ключові слова: алгоритм пошуку; геолокація; інформаційна система; транспортна логістика; міська мобільність; кросплатформна розробка; клієнт-серверна архітектура; .NET MAUI

Вступ

Перманентне збільшення транспортного навантаження на міські мережі, потреба в екологічно орієнтованих рішеннях і поширення концепції «розумного міста» в напрямку цифрової трансформації актуалізують розробку програмних систем, які здатні надійно і ефективно працювати в умовах складної транспортної інфраструктури і різноманітних користувацьких сценаріїв [1; 2].

Оптимізація транспортних потоків можлива за рахунок впровадження сервісів організації спільних поїздок (carpooling). Проте вирішення цієї задачі потребує не тільки використання коректних алгоритмів обробки геопросторових і часових даних, а й продуманої архітектури кросплатформних інформаційних систем, орієнтованих на одночасну експлуатацію в мобільному та десктопному середовищах [3].

Разом з тим, створення програмного забезпечення для таких систем стикається з технологічними бар'єрами, оскільки користувачі очікують доступу до сервісу з різних пристроїв: смартфонів під час поїздки і персональних комп'ютерів під час планування маршрутів. Проте розробка нативних застосунків окремо для Android (Kotlin), iOS (Swift) і macOS вимагає значних ресурсів. Альтернативним рішенням є використання кросплатформних фреймворків [4; 5].

Таким чином сучасні процеси урбанізації міського середовища зумовлюють зростання вимог до інформаційних систем, що забезпечують керування сервісами міської мобільності та їх підтримку [6].

Водночас ефективність, масштабованість і надійність функціонування таких систем визначається поєднанням архітектурних і алгоритмічних рішень. Саме це забезпечує актуальність задачі розробки кросплатформної інформаційної системи міської мобільності, що поєднує архітектурну універсальність і надійне алгоритмічне забезпечення пошуку і взаємодії користувачів.

Мета дослідження

Метою дослідження є розробка архітектурних принципів і алгоритмічного забезпечення кросплатформної інформаційної системи міської мобільності, здатної функціонувати в умовах невизначеності геолокаційних даних та забезпечувати адаптивну взаємодію користувача в мобільних і десктопних середовищах.

Аналіз основних досліджень і публікацій

Питання оптимізації транспортних потоків і розробки мобільних сервісів розглядаються у багатьох працях.

Дослідження Majchrzak et al. [5] порівнює продуктивність гібридних (Ionic) і компільованих (Flutter, React Native) рішень, вказуючи на переваги останніх у задачах, пов'язаних з картами [5; 6].

Однак, більшість наявних підходів мають «розрив технологічного стеку», коли клієнтська частина пишеться на JavaScript/Dart, а серверна – на Java/C#. Така фрагментація архітектури ускладнює масштабування, адаптацію систем і оптимізацію складних інформаційних архітектур [7].

Технологія .NET MAUI (Multi-platform App UI) пропонує уніфікований підхід до розробки, що дозволяє використовувати мову C# на всіх рівнях системи [7, 8]. Саме тому в цій роботі досліджується застосування фреймворку .NET MAUI для побудови єдиної екосистеми карпулінгу, що об'єднує мобільний і десктопний досвід.

Виклад основного матеріалу

1. **Архітектура системи**

Інформаційна система міської мобільності побудована за триланковою архітектурою:

– **Клієнт:** застосунок .NET MAUI, що використовує патерн MVVM (Model-View-ViewModel) [9] для розділення логіки та представлення.

– **Сервер:** REST API на базі ASP.NET Core [10].

– **Дані:** реляційна база даних (SQLite [11] для MVP версії) з доступом через Entity Framework Core [12].

Взаємодію компонентів у межах патерну MVVM показано на рис. 1.

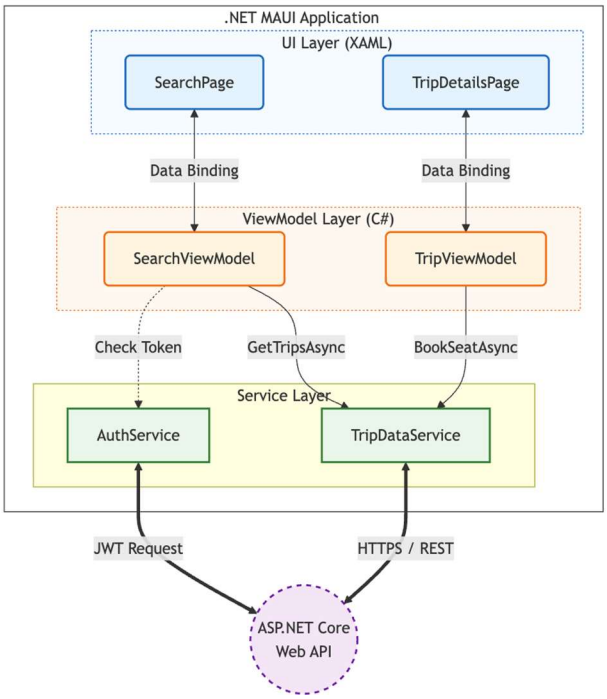

Рисунок 1 – Взаємодія компонентів MVVM у клієнтському застосунку

Особливістю реалізації інформаційної системи карпулінгу є використання механізму Dependency Injection (DI) для абстрагування платформо-залежних функцій. Так збереження токенів авторизації JSON Web Token (JWT) реалізовано через інтерфейс ISecureStorage, який на Android

звертається до системи безпечного зберігання криптографічних ключів KeyStore, а на iOS/macOS – до зашифрованої бази даних Keychain.

Застосування механізмів MVVM та Dependency Injection забезпечує відокремлення бізнес-логіки від платформо-залежних реалізацій, підвищує модульність, підтримуваність і розширюваність програмної системи. Такий підхід узгоджується із сучасними принципами проєктування адаптивних і масштабованих архітектур складних інформаційних систем [9 – 12].

Обрана архітектурна організація дозволяє нарощувати функціональні можливості без суттєвих змін у вже реалізованих модулях. Така властивість відповідає концепціям повторного використання компонентів і поетапної еволюції програмних систем, характерним для сучасних підходів до розроблення складних інформаційних платформ [9; 12].

Взаємодія між клієнтськими компонентами, службами доступу до даних та платформо-залежними модулями реалізована за принципами сервісної декомпозиції, що забезпечує незалежність розвитку окремих частин системи та спрощує їх інтеграцію. Такий підхід відповідає сучасним моделям сервісно-орієнтованої архітектури програмних систем [9; 10].

2. **Пошук попутників**

Задачу пошуку попутників подамо у вигляді кортежу:

$$S = \langle U, T, R, \Phi \rangle,$$

де $U = \{u_1, u_2, \dots, u_n\}$ – множина користувачів системи; $R = \{r_1, r_2, \dots, r_k\}$ – множина запитів на бронювання; $T = \{t_1, t_2, \dots, t_m\}$ – множина активних поїздок, створених водіями; $\Phi = \{\varphi_1, \varphi_2, \dots, \varphi_p\}$ – функція відповідності, що визначає релевантність поїздки запиту.

Кожна поїздка $t_m \in T$ характеризується набором параметрів:

$$t_m = \langle L_{start}, L_{end}, \tau_{dep}, C_{total}, C_{free}, P \rangle,$$

де L_{start}, L_{end} – координати пунктів відправлення і призначення; τ_{dep} – час відправлення; C_{total} – загальна місткість транспортного засобу; C_{free} – кількість доступних місць; P – вартість поїздки.

Запит пасажира q_j визначається як:

$$q_j = \langle l_{pas}, \tau_{req}, \triangle\tau, \delta \rangle,$$

де l_{pas} – поточна локація пасажира; τ_{req} – бажаний час відправлення, $\triangle\tau$ – допустиме часове відхилення; δ – максимально допустима просторова відстань (тут 5 км).

Умовою валідності підбору поїздки є виконання:

$$\begin{cases} dist\left(L_{start}^{(m)}, l_{pas}^{(j)}\right) \le \delta, \\ \left|\tau_{dep}^{(m)} - \tau_{req}^{(j)}\right| \le \triangle\tau, \\ c_{free}^{(m)} > 0, \end{cases} \quad (1)$$

де $dist(a, b)$ – функція обчислення відстані між точками на сфері [13; 14].

3. **Алгоритм пошуку поїздок**

Основним модулем системи карпулінгу є пошук поїздок.

Стандартні підходи, що базуються виключно на Geocoding API [15; 16], показали вразливість у разі помилок мережі. В таких умовах координати міст локації пасажирів можуть не визначатися, що робить пошук неможливим.

Розроблений гібридний алгоритм працює у двох режимах: геопросторовому і текстовому.

Геопросторовий режим для розрахунку відстані між точкою відправлення і координатами міст локації пасажирів використовує формулу [17; 18]:

$$d = 2R \cdot arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot sin^2\left(\frac{\Delta\lambda}{2}\right)}\right),$$

де: d – відстань між точками вздовж поверхні Землі; R – радіус Землі (≈ 6371 км); φ_1, φ_2 – широти точок у радіанах; $\Delta\varphi$ – різниця широт; $\Delta\lambda$ – різниця довгот.

Такий підхід забезпечує коректність вимірювання у глобальних координатах і широко використовується у транспортних та навігаційних інформаційних системах.

Текстовий режим (Fallback) використовується у разі відсутності координат. В цьому випадку виконується нормалізований текстовий пошук за індексованими полями бази даних.

Фрагмент реалізації сервісу пошуку мовою C#:

```
public async Task<IEnumerable<Trip>>
SearchTripsAsync(string fromCity, double lat, double
lon)
{
// Спроба пошуку за координатами (радіус 10 км)
if (lat.HasValue && lon.HasValue && lat != 0)
{
var trips = await _context.Trips.ToListAsync();
return trips.Where(t => CalculateDistance(lat.Value,
lon.Value, t.Lat, t.Lon) <= 10);
}

// Fallback: Текстовий пошук
return await _context.Trips.Where(t =>
t.FromCity.ToLower().Contains(fromCity.ToLower()))
.ToListAsync();
}
```

Запропонований гібридний алгоритм мінімізує обчислювальну складність перевірки першої нерівності системи (1) шляхом попередньої фільтрації множини T за індексованими текстовими полями (назви міст), що дозволяє знизити складність пошуку з $O(N)$ до $O(\log N)$ у більшості сценаріїв.

Це відповідає сучасним вимогам до алгоритмічної адаптації і зменшення обчислювальної складності в складних інформаційних системах [12].

4. **Проєктування взаємодії компонентів**

Логіку процесу бронювання місць показано на рис. 2.

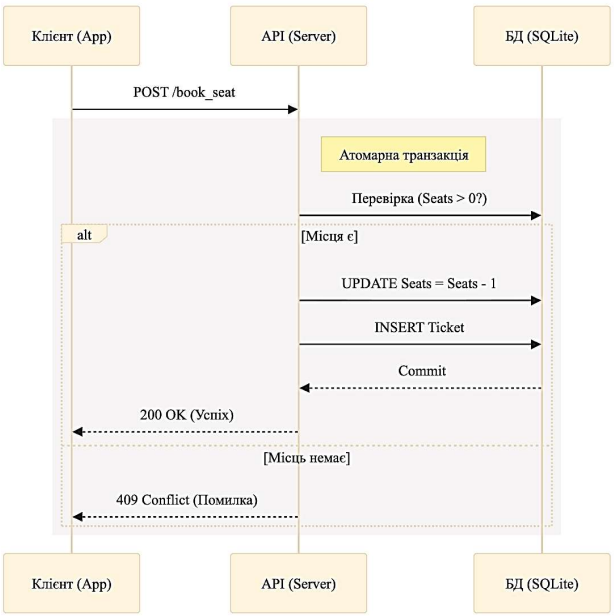

Рисунок 2 – Діаграма послідовності процесу бронювання місця

Для забезпечення цілісності даних під час бронювання місць розроблено протокол взаємодії між клієнтом і сервером, який враховує можливі затримки мобільної мережі.

5. **Адаптація інтерфейсу (MacCatalyst)**

Під час перенесення мобільного застосунку на macOS виявлено проблему відображення навігації. Стандартний TabBar на десктопі часто приховується системою. Для вирішення цієї проблеми в роботі застосовано модифікацію файлу Info.plist, що примусово встановлює режим сумісності з iPad (UIDeviceFamily = 1, 2), зберігаючи звичну навігацію для користувачів ПК.

Результати дослідження

1. Аналіз повторного використання коду в системі

Розроблена система була протестована на пристроях під керуванням Android 14 і macOS Sonoma.

Результати аналізу повторного використання коду в розробленій системі наведено в табл. 1.

Використання фреймворку .NET MAUI дозволило:

– сформувати єдину кодову базу для мобільних і десктопних платформ;

– досягти рівня повторного використання програмного коду понад 96%, що відповідає сучасним підходам до проєктування адаптивних і масштабованих архітектур складних інформаційних систем [11; 12].

Таблиця 1 – ***Результати аналізу кодової бази***

Компонент архітектури	Мова реалізації	Тип коду	Частка у спільном у проєкті
Моделі даних (DTO)	C#	Кросплат-формний	100%
Бізнес-логіка (ViewModels)	C#	Кросплат-формний	100%
Інтерфейс користувача (UI)	XAML	Кросплат-формний	100%
Сервіси доступу до API	C#	Кросплат-формний	100%
Абстракція сховища (SecureStorage)	C#	Адаптер	90%
Конфігурація платформ	XML/Plist	Нативний	0%
Загальний показник системи			>96%

2. Аналіз ефективності використання ресурсів

Окрім оцінки спільного коду, в цій роботі було проведено порівняльний аналіз споживання оперативної пам'яті (RAM) і часу «холодного» старту застосунку на різних платформах.

Результати тестування продуктивності системи, що проводилося на пристроях Google Pixel 7 (Android) і iPhone 13 (iOS) наведені в табл. 2.

Таблиця 2 – ***Показники продуктивності системи***

Показник	Платформа		
	Android	iOS	macOS
Час запуску (Cold Start)	1.2 с	0.9 с	0.6 с
Споживання RAM (Спокій)	85 МБ	72 МБ	110 МБ
Споживання RAM (Активна карта)	145 МБ	130 МБ	185 МБ
Розмір інсталяційного пакету	35 МБ (.apk)	42 МБ (.ipa)	55 МБ (.app)

З таблиці видно, що використання єдиного стеку технологій дозволило мінімізувати дублювання коду. Реалізований алгоритм пошуку показав стабільну роботу: час відгуку під час текстового пошуку склав менше 150 мс, за використання геопошуку – менше 200 мс. Ці показники узгоджуються з вимогами до якості програмного забезпечення за стандартом ISO/IEC 25010 [17].

Слід також відзначити ефективність роботи Garbage Collector у .NET 8.

Під час тестування списку з 1000 елементів поїздок (Virtualization у CollectionView):

– не було зафіксовано візуальних затримок;

– частота оновлення трималася на рівні 58–60 кадрів/сек.

Висновки

Наукова новизна роботи полягає у формалізації задачі пошуку попутників у вигляді кортежної моделі, що уніфікує представлення користувачів, запитів та транспортних подій у межах єдиного інформаційного простору.

Розроблений гібридний алгоритм відповідності забезпечує адаптивний перехід між геопросторовим і текстовим режимами пошуку в умовах невизначеності або відсутності координатних даних. На відміну від існуючих підходів, запропонований метод поєднує архітектурну повторну використовуваність програмних компонентів із алгоритмічною стійкістю до відмов зовнішніх сервісів.

На відміну від існуючих сервісів карпулінгу, де геопросторовий модуль жорстко залежить від доступності зовнішніх API, у запропонованому забезпечується безперервність функціонування за рахунок внутрішнього механізму алгоритмічного резервування.

У межах цього дослідження реалізовано кросплатформну інформаційну систему карпулінгу як багатокомпонентну систему міської мобільності, що здатна функціонувати на різних апаратно-програмних платформах і в умовах нестабільного мережевого середовища. Запропоновані архітектурні рішення забезпечують масштабованість системи та можливість її подальшої модернізації, зокрема переходу до промислових систем керування базами даних.

Практична реалізація гібридного алгоритму, що поєднує геопросторові обчислення з резервною текстовою фільтрацією, забезпечила безперервність роботи сервісу навіть у випадках недоступності служб геолокації. Проведені експериментальні дослідження показали зменшення часу пошуку та підвищення ефективності відбору релевантних поїздок; у структурованих сценаріях використання індексації спостерігалося наближення складності пошуку до логарифмічної.

Отримані результати можуть бути використані під час проєктування інших класів розподілених сервісних платформ у сфері міської мобільності та інтелектуальних транспортних систем, де критичними є адаптивність, повторне використання архітектурних компонентів і стійкість до невизначеності даних.

Подальші дослідження доцільно спрямувати на врахування проміжних точок маршруту, що розширить можливості оптимізації та персоналізації транспортних сервісів.

Конфлікт інтересів. Автори підтверджують відсутність фінансових, особистих чи інших інтересів, що можуть розглядатися як потенційний конфлікт інтересів щодо публікації цієї статті.

Фінансування. Дослідження було проведено без фінансової підтримки.

Доступність даних. Усі дані доступні в цифровій або графічній формі в основному тексті рукопису.

Використання штучного інтелекту. Автори підтверджують, що при створенні даної роботи вони не використовували інструментальні засоби штучного інтелекту.

Список використаних джерел / References

1. Shaheen, S., & Cohen, A. (2019). Smart cities and urban mobility. *International Journal of Urban Sciences*, 23, 12–28.
2. Batty, M. (2013). The new science of cities. MIT Press. 510 p.
3. Chan, N. D., & Shaheen, S. A. (2012). Ridesharing in North America: Past, present, and future. *Transport Reviews*, 32(1), 93–112.
4. Majchrzak, T. A., Biørn-Hansen, A., & Grønli, T. M. (2015). Comprehensive analysis of cross-platform app approaches. *2015 IEEE International Conference on Mobile Services*, 402–409.
5. Biørn-Hansen, A., Grønli, T. M., & Ghinea, G. (2019). Animations in cross-platform mobile applications. *Sensors*, 19(9), 2081.
6. Albino, V., Berardi, U., & Dangelico, R. (2015). Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of Urban Technology*, 22(1), 3–21.
7. Riabchun, Yu. V., Kurinskyi, O. V., Dolia, O. V., & Fesan, A. O. (2025). Optimization and adaptation of neural networks based on existing architectures. *Management of Development of Complex Systems*, (61), 210–218. [in Ukrainian].
8. Microsoft. (2025). .NET Multi-platform App UI documentation. https://learn.microsoft.com/dotnet/maui
9. Sommerville, I. (2016). Software engineering (10th ed.). Pearson. 816 p.
10. Fowler, M. (2002). Patterns of enterprise application architecture. Addison-Wesley. 560 p.
11. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley. 395 p.
12. Riabchun, Yu., Kurinsky, O., Palamarchuk, D., et al. (2025). Optimization and adaptation of neural networks based on existing architectures. *2025 IEEE 5th International Conference on Smart Information Systems and Technologies (SIST)*, 798–804.

13. Haversine, R. (1918). Great-circle distance between two points on a sphere.
14. Sinnott, R. W. (1984). Virtues of the haversine. *Sky and Telescope*, 68(2), 159.
15. Google. (n.d.). Geocoding API documentation. https://developers.google.com/maps/documentation/geocoding
16. OpenStreetMap Foundation. (n.d.). Nominatim geocoding service. https://nominatim.org/
17. International Organization for Standardization. (2011). Systems and software quality models (ISO/IEC Standard No. 25010:2011).
18. Pressman, R. S. (2015). Software engineering: A practitioner's approach (8th ed.). McGraw-Hill. 976 p.

Ihor Stuzhuk
ORCID: https://orcid.org/0009-0004-8707-3567
Kyiv National University of Construction and Architecture, Kyiv, Ukraine
Graduate Student of the Department of Information Technologies
Yuriy Naumenko
ORCID: https://orcid.org/0000-0002-2631-1048
Kyiv National University of Construction and Architecture, Kyiv, Ukraine
PhD, Associate Professor of the Department of Information Technologies
Yevhenii Borodavka
ORCID: https://orcid.org/0000-0002-7476-9387
Kyiv National University of Construction and Architecture, Kyiv, Ukraine
D.Sc., Professor, Head of the Department of Information Technologies of Design and Applied Mathematics
Iryna Aznaurian
ORCID: https://orcid.org/0000-0002-7085-7291
Kyiv National University of Construction and Architecture, Kyiv, Ukraine
Associate Professor of the Department of Physics

DEVELOPMENT OF CROSS-PLATFORM INFORMATION SYSTEMS FOR URBAN MOBILITY: ARCHITECTURAL AND ALGORITHMIC SOLUTIONS

Abstract. *The relevance of the work stems from the increasing load on urban transport infrastructure, the need to implement environmentally oriented solutions, and the growing requirements for the versatility and scalability of software systems operating within the "smart city" concept. The study focuses on ride-sharing services. The paper substantiates the feasibility of a cross-platform approach as a means to reduce the costs of developing and maintaining information systems designed for different operating environments. Architectural and algorithmic aspects of developing cross-platform information systems for urban mobility are studied. The architecture of a client-server information system is proposed and implemented using the .NET MAUI framework, which enables a single codebase for mobile and desktop platforms. The application of MVVM and Dependency Injection patterns to enhance the system's modularity, maintainability, and extensibility is considered. The problem of finding fellow travelers subject to geospatial and temporal constraints is formalized. A hybrid trip search algorithm is developed that combines geolocation calculations based on a spherical metric with text filtering as a backup mechanism in case of coordinate-determination service failures. The proposed solution allows for reducing the computational complexity of the search and ensures the fault tolerance of the subsystem under unstable network conditions. An experimental assessment of the implementation efficiency was carried out, confirming a high (over 96%) level of code reuse and acceptable performance indicators on different platforms. Results have been obtained that can be used in the development of scalable urban mobility information systems and other complex distributed systems focused on cross-platform operation.*

Keywords: .NET MAUI; client-server architecture; cross-platform development; geolocation; information system; search algorithm; transport logistics; urban mobility.

Посилання на публікацію

APA Stuzhuk, I., Naumenko, Yu., Borodavka, Ye., & Aznaurian, I. (2026). Development of Cross-Platform Information Systems for Urban Mobility: Architectural and Algorithmic Solutions. *Management of Development of Complex Systems*, 65, 188–193, dx.doi.org\10.32347/2412-9933.2026.65.188-193.

ДСТУ Стужук І. М., Науменко Ю. О., Бородавка Є. В., Азнаурян І. О. Розробка кросплатформних інформаційних систем міської мобільності: архітектурні та алгоритмічні рішення. *Управління розвитком складних систем.* Київ, 2026. № 65. С. 188 – 193, dx.doi.org\10.32347/2412-9933.2026.65.188-193.