УДК 004.056: 004.056.53

ОСОБЛИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО РЕАЛІЗУЄ МЕТОД ПОШУКУ ЗА ПРЕФІКСОМ В КРИПТОГРАФІЧНО ЗАХИЩЕНИХ БАЗАХ ДАНИХ

Сергій Лілікович, Віталій Єсін

Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, 61022, Україна
serhii.lilikovych@student.karazin.ua, v.i.yesin@karazin.ua

Надійшла: жовтень 2023. Прийнята: листопад 2023.

***Анотація**: У роботі розглядаються особливості розробки програмного забезпечення (ПЗ), що реалізує метод пошуку за префіксом в криптографічно захищених базах даних. Цей метод є різновидом симетричного шифрування із можливістю пошуку. Він дозволяє виконувати пошук за префіксом серед зашифрованих даних без необхідності їх розшифрування. Такий підхід дозволяє розв'язати проблему збереження конфіденційності даних, що зберігаються на віддалених або хмарних серверах. Однак його використання обумовлює ряд особливостей, які необхідно враховувати при розробці відповідного ПЗ, що його реалізує. У роботі аналізуються вимоги до ПЗ, що розробляється, яке реалізує метод пошуку за префіксом, визначається архітектура ПЗ, обґрунтовується вибір технологій та інструментальних засобів реалізації ПЗ, зокрема, технології ASP.NET, мов програмування Java, JavaScript, PHP, Python, СКБД MongoDB та фреймворку FastAPI, наводиться опис процесу розгортання відповідного програмного забезпечення. Для тестування швидкодії розробленого програмного забезпечення використовувався відомий інструмент для проведення навантажувального тестування Apache JMeter. Отримані оцінки продуктивності запропонованого рішення свідчать про прийнятність часових затримок на обробку відповідних запитів з пошуку даних.*

***Ключові слова:** база даних, шифрування з можливістю пошуку, конфіденційність, програмне забезпечення.*

1. Вступ

Сьогодні зберігання та обробка даних на сторонніх віддалених хмарних серверах знаходить широке застосування. Однак у міру збільшення масштабу, цінності та централізації даних виявляється зворотний бік цього процесу – загострюються проблеми забезпечення безпеки та приватності даних, що викликає серйозне занепокоєння у власників та користувачів даних. Існує виявлений ризик, що дані, які зберігаються в базах даних, можуть бути скомпрометовані [1-2]. Відомим способом розв'язання цієї проблеми та забезпечення конфіденційності даних є їх шифрування. При цьому, використання традиційних методів шифрування стикається зі специфічними труднощами, а саме – як дозволити ненадійним хмарним серверам виконувати пошукові операції так, щоб шукані дані залишалися конфіденційними. В цілому, проблематика пошуку даних у зашифрованих базах даних (БД) викликала великий інтерес, як у наукових колах, так і в цілому, в індустрії інформаційних технологій - IT [2-3]. Для розв'язання питань пошуку потрібної інформації в криптографічно захищених БД були проведені відповідні дослідження, що пов'язані з розробкою нових криптографічних примітивів, нових структур даних для шифрування з можливістю пошуку, розвитком поглядів на безпеку [4-5]. Однак попри велику різноманітність запропонованих на сьогодні рішень, немає домінуючої думки, що універсальною для всіх відомих випадків використання. Як і не існує найбільш захищеної пошукової системи або набору відповідних методів. Тому власники та користувачі даних повинні чітко усвідомлювати наскільки підходять для їх різних застосунків досить широкий існуючий спектр захищених систем БД і які саме компроміси для їхнього варіанта використання, допустимі. Все це стимулювало дослідження в галузі безпечного управління даними та підвищило їх актуальність [2].

У цій роботі розглядаються основні аспекти розробки *(аналіз вимог, вибір патерну системної архітектури, вибір технологій)* та особливості програмного забезпечення (ПЗ), що

реалізує один з методів пошуку даних у криптографічно захищених базах даних, а саме метод пошуку за префіксом, який представлено у роботі [6].

2. Аналіз вимог до програмного забезпечення, що реалізує метод пошуку за префіксом

Програмне забезпечення, що реалізує відповідні різні методи шифрування з можливістю пошуку (*SE – Searchable Encryption*), природно відрізняється набором вимог. Аналіз вимог є критичним для успішної розробки проекту. Для визначення відповідних функціональних і не функціональних вимог, щодо безпеки що пред'являються до ПЗ, можна скористатися діаграмою потоків даних для процесу пошуку в симетричних системах *SE* (див. рис.1):

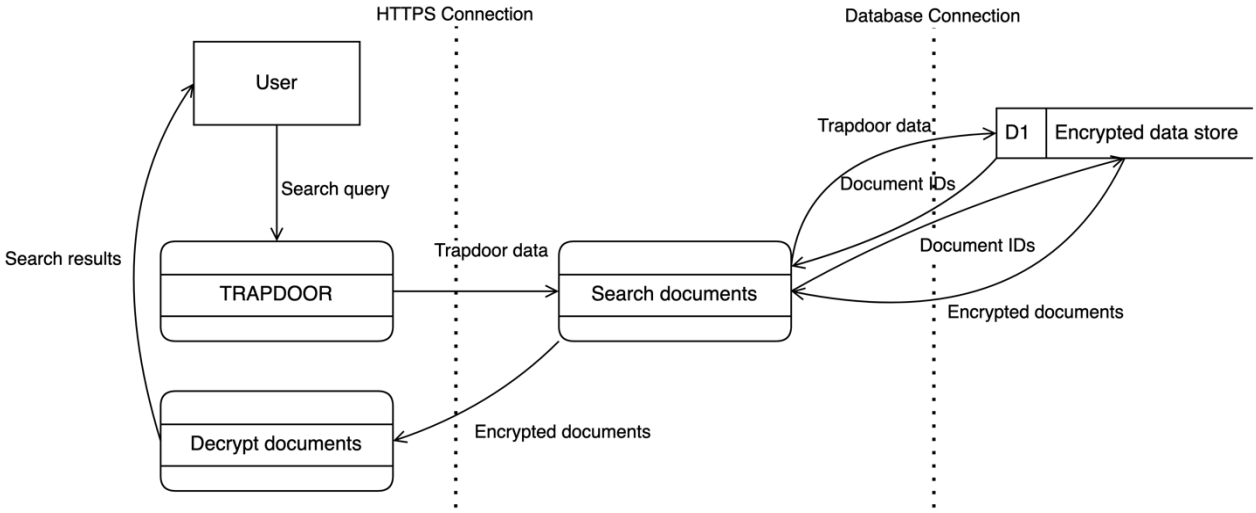

Рис. 1 – Діаграма потоків даних в симетричних системах *SE*
Fig. 1 – Diagram of data flows in symmetric systems *SE*

На діаграмі показано, що пошуковий запит (*search query*) перетворюється на спеціальну «лазівку» (*trapdoor data*) за допомогою алгоритму TRAPDOOR на клієнтському вузлі. «Лазівка» передається на вузол застосунку, який виконує з її використанням запити до БД. Тільки вузол клієнта має доступ до секретних ключів, що використовуються для формування «лазівок».

Виходячи з представленої діаграми, можна ідентифікувати такі загрози:

1. Атака «людина посередині» між вузлами системи, що мають мережеве з'єднання. Зловмисник зможе отримати відомості про «лазівки» та перехопити шифротексти документів (*encrypted documents*). Він також зможе перехопити ідентифікатори відповідних документів (*document IDs*).
2. Атака на алгоритм TRAPDOOR, що дозволить зловмиснику генерувати власні "лазівки" без знання ключової інформації.
3. Атака на вузол сховища даних «*encrypted data store*» – атака «відомого шифртексту», що дозволить зловмиснику отримати зашифровані дані без належної автентифікації, або відомості про зв'язки між документами тощо.

Таким чином, до програмного забезпечення висуваються такі нефункціональні вимоги:

1. Секретний ключ повинен використовуватись та зберігатись тільки на клієнтському вузлі.
2. Пошуковий індекс та документи, що зберігаються в БД, повинні бути криптографічно захищені, а їх зміст не повинен розкриватись на будь-якому вузлі системи, окрім клієнтського.
3. Алгоритм TRAPDOOR повинен бути криптографічно стійким.

4. Канали зв'язку між вузлами системи повинні бути захищеними.

Набір функціональних вимог та інших не функціональних вимог визначаються в залежності від конкретних завдань, які повинна вирішувати програмна система. При аналізі вимог корисно враховувати існуючі моделі безпеки для методів *SE* – вони можуть стати джерелом додаткових вимог до програмної системи. Деякі з цих моделей описані в [7]. В рамках даної роботи щодо функціональних вимог достатньо визначити, що серед них є вимога надавати користувачу можливість пошуку в криптографічно захищений БД за префіксом, з використанням обраного методу *SE*.

3. Визначення системної архітектури ПЗ

В застосунках де передбачається використання шифрування з можливістю пошуку доцільно використовувати 3-х рівневу клієнт-серверну архітектуру *(як шаблон або патерн (pattern))*, за допомогою якої розробники можуть створювати гнучке та повторно використовуване ПЗ. Такий архітектурний шаблон допомагає структурувати програми, які можна розкласти на групи підзавдань, у яких кожна група підзавдань знаходиться на певному рівні абстракції [8]. В цьому випадку такий патерн клієнт-серверної архітектури дозволяє перенести криптографічні функції на бік клієнтського застосунку, доступ до даних може бути обмеженим за допомогою серверу БД, а сервер застосунку може приховати деталі взаємодії з пошуковим індексом та БД. Приклад діаграми розгортання застосунку, що використовує такий патерн системної архітектури для реалізації пошуку за методом SE показано на рис.2.

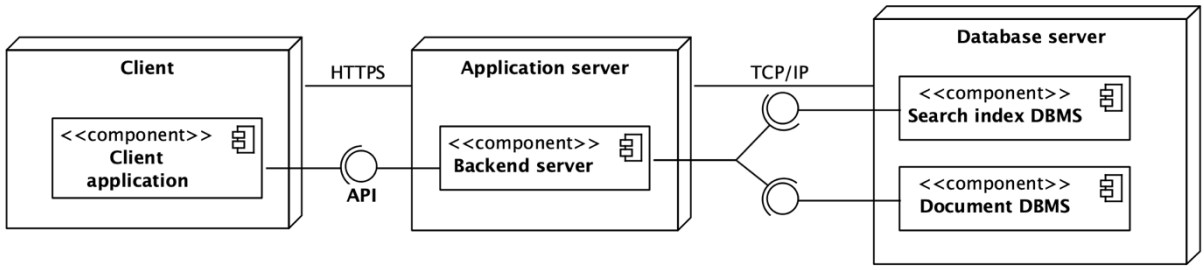

Рис. 2 – Діаграма розгортання ПЗ
Fig. 2 - Software deployment diagram

На діаграмі видно, що у порівнянні з типовою діаграмою розгортання 3-х рівневого застосунку, на фізичному рівні використання SE не призводить до значних змін в архітектурі, окрім можливого додавання нового компонента БД, що може зберігати пошуковий індекс.

4. Вибір технологій реалізації ПЗ

Основною особливістю при виборі технологій реалізації ПЗ пошуку в криптографічно захищених БД є необхідність враховувати, що окрім зашифрованих документів, для забезпечення можливості пошуку методами симетричного *SE* необхідно додатково зберігати та обслуговувати пошуковий індекс, за рахунок якого стає можливою операція пошуку без розшифрування даних. У випадку використання методу пошуку за префіксом, індекс має деревовидну структуру, що впливає на вибір типу СКБД *(системи управління/керування базами даних* СКБД/СУБД*, від англ. Database Management System, DBMS))* для його зберігання.

У разі використання реляційної СКБД, зберігання деревовидної структури даних потребує додаткових супутніх витрат на її обслуговування. В деяких СКБД, таких як *PostgreSQL* і *MongoDB,* деревовидні структури даних підтримуються на рівні розширень або в основному функціоналі, що впливає на вибір конкретного рішення для зберігання даних пошукового індексу. З урахуванням цієї особливості, застосовується документ-орієнтована *MongoDB*, яка

підтримує не тільки збереження деревовидних структур даних, а й виконання складних запитів до них, що може бути використано для оптимізації операцій з індексом. Одночасно з цим, зберігання великих за обсягом записів в *MongoDB* значно знижує швидкодію роботи з такою БД, тому у випадку використання цієї СКБД з документами великого обсягу, необхідно забезпечити їх окреме зберігання поза межами БД індексу.

Особливості вибору інших технологій реалізації полягають у необхідності вибору сучасних підтримуваних технологій, з достатньою кількістю інструментів підтримки якості, наприклад аналізаторів коду, сканерів вразливостей тощо. Далі розглядається обґрунтування вибору інструментальних засобів ПЗ для конкретної реалізації.

4.1 Вибір інструментальних засобів

На сьогодні ринок надає широкий вибір технологій реалізації програмного забезпечення. В якості потенційних засобів розробки програмного забезпечення розглядалися наступні технології: - *Java, JavaScript, Python, ASP.NET, PHP*. Список з цих технологій був складений виходячи з практичної можливості використання кожної з них. При цьому в процесі вибору інструментарію були сформульовані відповідні критерії:

1. Багатоплатформність – один і той же програмний код повинен покрити якомога більше платформ.
2. Доступність – розробка повинна бути максимально дешевою.
3. Простота розробки – час розробки має бути мінімальним.
4. Наявність зручного та сучасного GUI-фреймворку для швидкої розробки графічного інтерфейсу.
5. Наявність простих інструментів статичного аналізу коду для підтримки його якості.

Детальніше розглянемо властивості зазначених засобів розробки.

Мова програмування Java

Java – це об’єктно орієнтована мова програмування, яка була вперше представлена в 1995 році компанією Sun Microsystems як ключовий компонент платформи Java. З 2009 року розвитком та підтримкою мови займається компанія Oracle, яка того року придбала Sun Microsystems. Офіційна реалізація програм на Java компілюється в байт-код, який потім виконується віртуальною машиною, специфічною для конкретної платформи.

Oracle надає компілятор *Java* та віртуальну машину Java, які відповідають специфікаціям *Java Community Process*, і це відбувається під ліцензією *GNU General Public License*.

Синтаксис Java схожий на C та C++, але з істотними відмінностями. Розробники технології врахували досвід розробки за допомогою цих двох мов, що призвело до усунення можливості виникнення деяких конфліктних ситуацій, які могли виникнути через помилки програміста, і спростило процес розробки об’єктно орієнтованих програм. Багато з тих дій, які вимагаються в C/C++, тепер виконує віртуальна машина. Головною метою *Java* завжди було створення мови, яка була б платформонезалежною, і це визначає її обмеження в роботі з апаратним забезпеченням порівняно з, наприклад, C++. Таким чином, швидкість роботи програми може бути меншою. Проте, *Java* надає можливість викликати підпрограми, написані іншими мовами програмування, коли це необхідно. Приклад програми мовою *Java* в середовищі *IntelliJ IDEA* наведено на рис. 3.

Java мала значний вплив на розвиток мови програмування J++, яку розробляла компанія Microsoft. Роботу над J++ було припинено через судовий позов, поданий *Sun Microsystems*, оскільки ця мова програмування була модифікацією *Java*. Пізніше, при створенні нової платформи .NET компанією Microsoft, була представлена мова J#, яка спрощувала міграцію програмістів, які вже володіли *J++* або *Java*, на нову платформу. З часом нова мова програ-

мування C# стала основною мовою платформи .NET, позичивши багато ідей з Java. J# включали востаннє в версію *Microsoft Visual Studio 2005*.

Іншою технологією, що має широкі можливості як клієнтських, так і серверних застосунків є *JavaScript*.

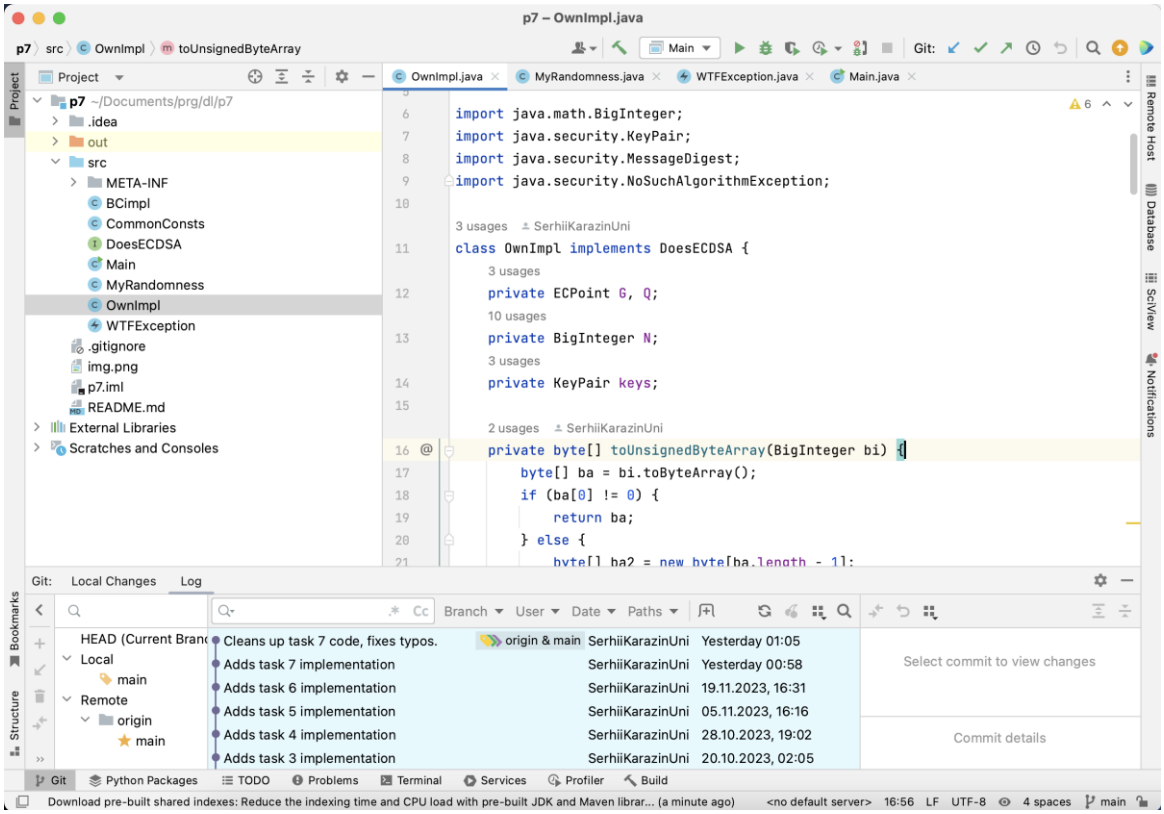

Рис. 3 – Приклад програми мовою *Java* (*в середовищі IntelliJ IDEA)*

Fig. 3 – Example of *Java* program (*in the environment IntelliJ IDEA*)

Мова програмування JavaScript

JavaScript є однією з найпоширеніших мов програмування у світі. Вона підтримується всіма основними браузерами, включаючи *Chrome, Firefox, Edge* та *Safari*. *JS* також є основою для багатьох популярних фреймворків, таких як *React, Angular* та *Vue.JS*.

Серед основних переваг *JavaScript* можна назвати такі:

1. Має простий синтаксис, подібний до C та C++.
2. Є широко поширеною мовою, що робить її легко доступною для розробників.
3. Де-факто є стандартом для сучасних веб-застосунків, підтримується більшістю веб-браузерів.
4. Деякі реалізації *JavaScript* є проектами відкритого коду, що дозволяє розробникам робити свій внесок у розвиток мови, проводити незалежні аудити безпеки тощо.

Серед основних недоліків *JavaScrip*t виділяють:

1. Програми на *JavaScript* інтерпретуються на стороні клієнта, а отже передаються туди у вигляді відкритого тексту. Це може впливати на безпеку застосунків, адже клієнт може модифікувати код.

2. *JavaScript* є динамічною мовою без статичної типізації, що може призводити до помилок.

Приклад програми мовою JavaScript, в середовищі *IntelliJ IDEA* наведено на рис. 4. Для JavaScript існують фреймворки та численні інструменти підтримки якості коду, що можуть значно полегшити розробку програмного забезпечення роботи. Одним з них є *Vue.JS* – фрей-

мворк для створення односторінкових веб-застосунків, що прискорює розробку. Іншим фреймворком, що може бути використаним в роботі, є *Vuetify* – фреймворк для створення інтерфейсів користувача на основі *Vue.JS*. *Vuetify* надає широкий спектр готових компонентів, які можна використовувати для швидкого створення красивих і зручних у використанні інтерфейсів.

Альтернативою до використання *JavaScript* є технології на платформі .NET, зокрема *ASP.NET*.

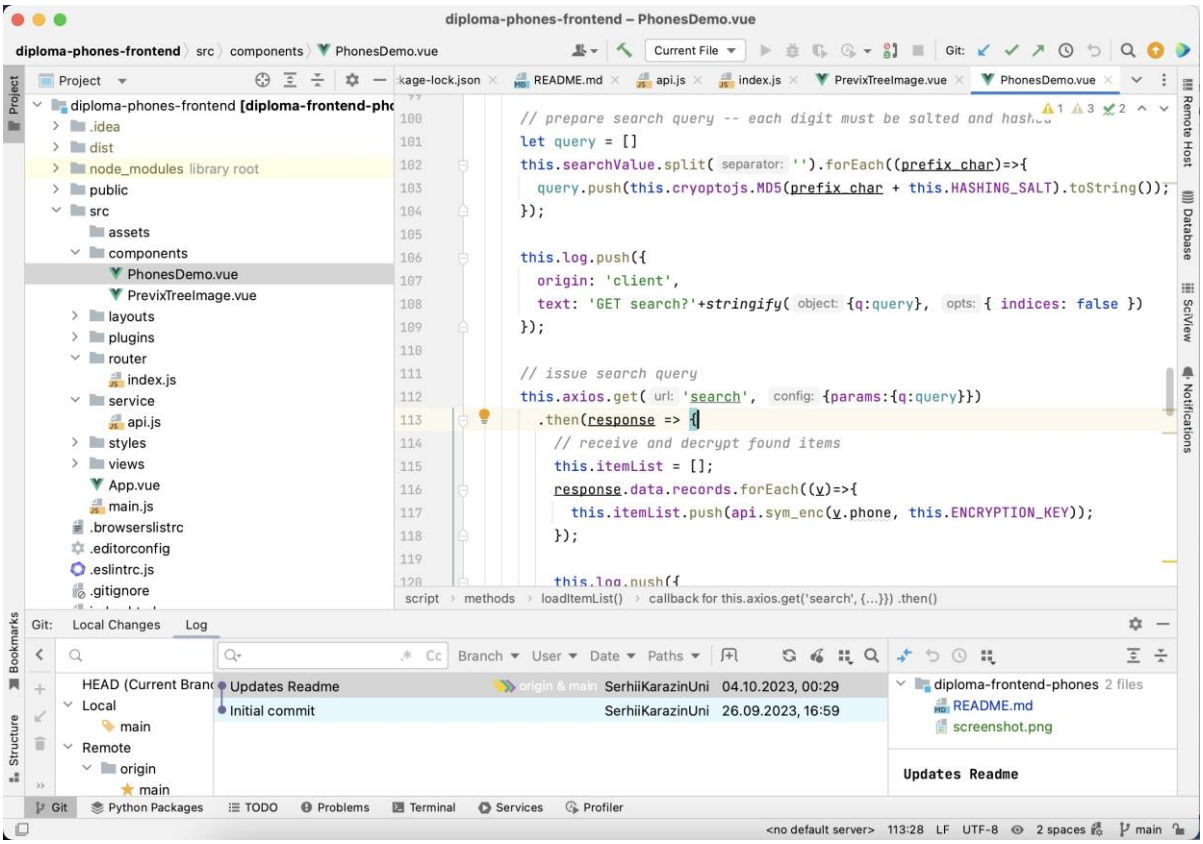

Рис. 4 – Приклад програми мовою *JavaScript (в середовищі IntelliJ IDEA)*

Fig. 4 – Example of *JavaScript* program (*in the environment IntelliJ IDEA*)

Технологія ASP.NET

ASP.NET – це технологія створення веб застосунків і веб сервісів від компанії Microsoft, і вона є важливою частиною платформи *Microsoft.NET*. Ця технологія являє собою еволюцію попередньої версії – *Microsoft ASP*. На сьогодні останньою версією *ASP.NET* є *ASP.NET 6*.

ASP.NET має багато спільних рис зі своєю попередницею ASP, що дозволяє розробникам переходити на *ASP.NET* досить легко. Проте внутрішня структура *ASP.NET* суттєво відрізняється від *ASP*, оскільки *ASP.NET* базується на платформі .NET і використовує всі переваги цієї платформи. Приклад програми для платформи *ASP.NET* в середовищі Microsoft *Visual Studio* наведено на рис. 5.

Microsoft повністю переробила *ASP.NET*, використовуючи Common Language Runtime як основу, яка є фундаментальною для всіх застосунків *Microsoft .NET*. Один із головних плюсів *ASP.NET* полягає в тому, що розробники можуть писати код для *ASP.NET*, використовуючи практично будь-яку мову програмування, доступну в *.NET Framework* (такі як C#).

Основною перевагою *ASP.NET* є його продуктивність. Під час першого запиту код компілюється і зберігається в спеціальному кеші, що дозволяє його швидше виконувати в подальших запитах, і не потребує синтаксичного аналізу (*parsing*), оптимізації та інших проміжних операцій, що заощаджує час.

Іншою популярною технологією є *Python*.

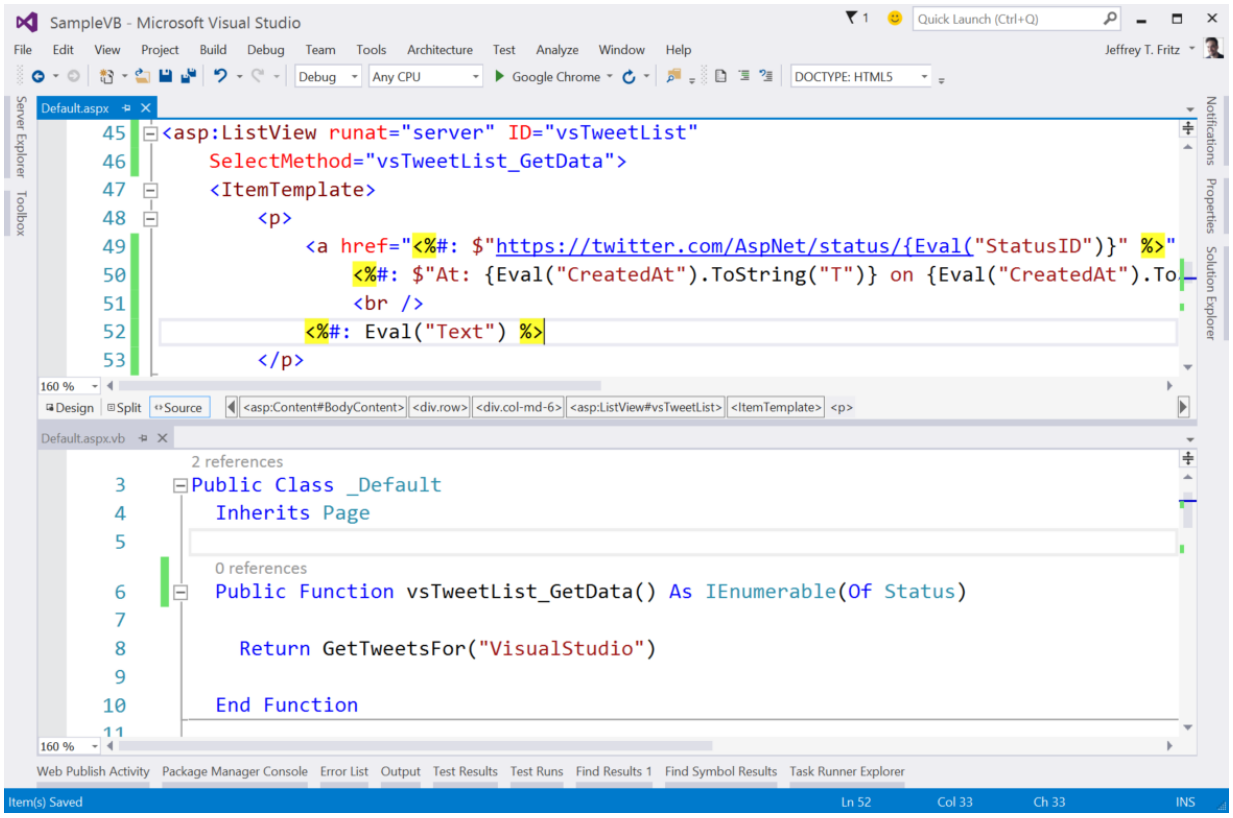

Рис. 5 – Приклад застосування технології *ASP.NET* в середовищі *Visual Studio*

Fig. 5 – Application example of the implementation of *ASP.NET* technology in the middle of *Visual Studio*

Мова програмування Python

Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. *Python* підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор *Python* та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах.

В мові програмування *Python* підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі:

1. Доступний для розробників різного рівня досвіду синтаксис мови.
2. Переносність програм (що властиве більшості інтерпретованих мов).
3. Зручний для розв'язання математичних задач (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини).
4. Відкритий код.

До недоліків *Python* можна віднести:

1. Низьку швидкодію (*у порівнянні з мовами більш низького рівня*).
2. Відсутність статичної типізації.

Для Python існують інструменти підтримки якості коду, та фреймворки, що можуть значно полегшити розробку програмного забезпечення прототипу. Одним з них є *FastAPI* – високопродуктивний веб-фреймворк для створення *API*. Приклад програми мовою *Python* з використанням *FastAPI*, в середовищі *IntelliJ IDEA* наведено на рис. 6.

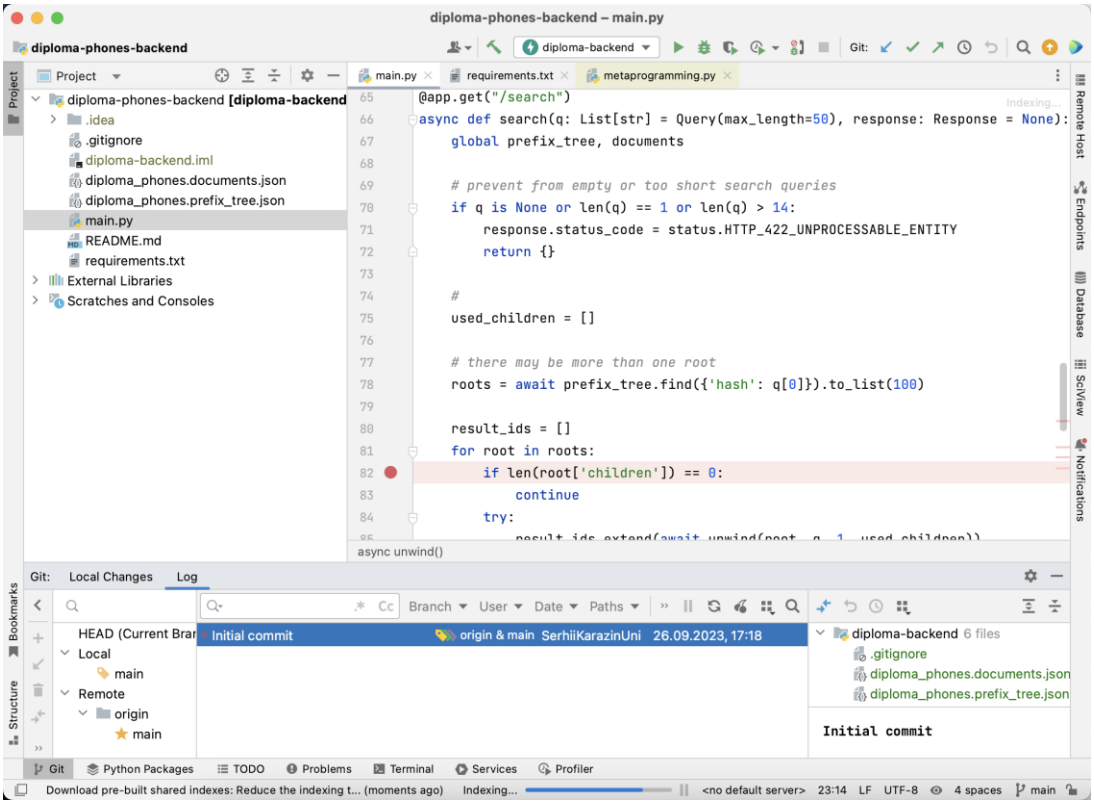

Рис. 6 – Приклад програми мовою *Python* (*в середовищі IntelliJ IDEA)*

Fig. 6 – Application example of *Python* program (*in the environment IntelliJ IDEA*)

Важливі властивості *FastAPI*:

1. Один з найшвидших веб фреймворків для *Python*. Він досягає високої продуктивності шляхом використання асинхронного запиту та відповіді, а також оптимізації коду.
2. Має простий і інтуїтивно зрозумілий синтаксис. Він також має ряд вбудованих функцій, які спрощують створення *API*.
3. Має вбудовані функції для документування *API* та тестування.

Крім того, при розробці веб-додатків однією з найпопулярніших є технологія *PHP*.

Мова програмування PHP

PHP – це скриптова мова програмування, створена для генерації *HTML*-сторінок на стороні веб сервера. *PHP* є однією з найпоширеніших мов, використовуваних у сфері веб розробки, разом з *Java, .NET, Python* та *Ruby*.

PHP була розроблена в 1994 році, а наразі є однією з найпоширеніших мов програмування для веб розробки. Велика кількість веб сайтів, включаючи популярні соціальні мережі, електронні комерційні платформи та блоги, побудовані з використанням *PHP*. Однією з переваг *PHP* є те, що вона підтримується більшістю хостинг-провайдерів, що робить її доступною для широкого кола веб розробників.

Завдяки вдосконаленням у внутрішній архітектурі та оптимізаціям, зараз *PHP* здатна конкурувати з іншими популярними мовами програмування, такими як *Java* та *Python*. Швидкодія стала особливо помітною в *PHP 7*, яка отримала значне прискорення завдяки оптимізації в роботі з пам'яттю й виконанню опкоду.

Основні переваги *PHP* включають:

1. *PHP* – мова програмування з синтаксисом, схожим на мови з родини «C».
2. Спрямованість на веб розробку – *PHP* призначена для розв'язання задач, пов'язаних з веб розробкою, і не вимагає від програміста глибоких знань інших технологій.
3. Висока популярність серед веб-розробників та спільноти користувачів.

4. Ліцензія відкритого ПЗ, що дозволяє вільне використання і розповсюдження.

Приклад програми мовою *PHP*, в середовищі *IntelliJ IDEA* наведено на рис. 7.

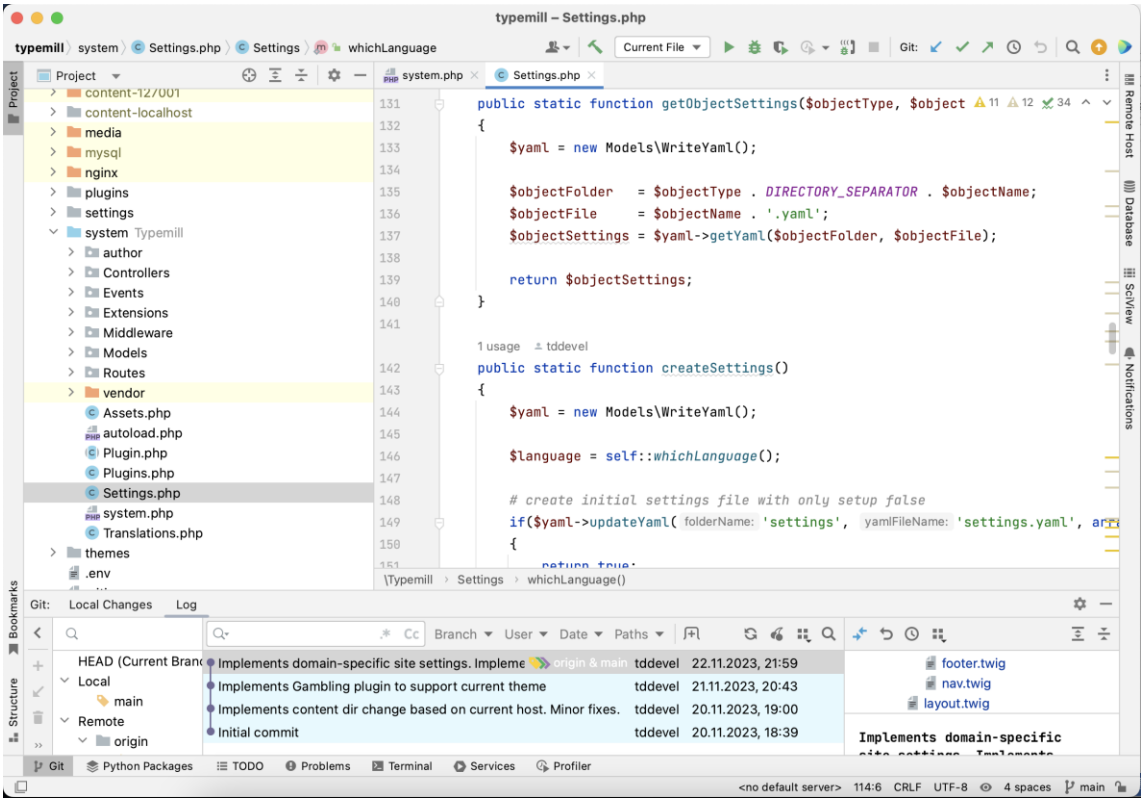

Рис. 7 – Приклад програми мовою *PHP* (*в середовищі IntelliJ IDEA*)

Fig. 7 – Application example of *PHP* program (*in the environment IntelliJ IDEA*)

Серед недоліків можна виділити:

1. Забезпечення безпеки. PHP в минулому мав деякі серйозні проблеми із забезпеченням безпекою (*наприклад, такі як вразливості SQL-ін'єкції, вразливості з крос-сайт скриптінгом тощо; відтоді в PHP були внесені покращення, такі як впровадження підготованих виразів (prepared statements) для SQL та бібліотеки PDO, але питання безпеки є актуальним для PHP і сьогодні*).

2. Складні назви деяких поширених функцій, таких як *htmlspecialchars*, *mysql_select_db*, *nl2br* тощо. У *PHP* можуть існувати кілька функцій, які виконують схожі завдання, і це може призвести до плутанини та невизначеності у виборі правильної функції. Крім того, у різних функціях *PHP* можуть існувати різні стандарти назв функцій та порядку аргументів, що може призвести до плутанини та помилок.

Для *PHP* існують численні фреймворки та інструменти підтримки якості коду, проте в швидкодії *Python* зазвичай має перевагу над *PHP*.

Порівняння технологій

Для порівняння та вибору технологій можна прибігти до методу експертних оцінок за критеріями, які були сформульовані вище. Для цього розглянуті технології були оцінені за трьома критеріями: - кросплатформність, доступність, та простота розробки. В табл.1 наведені оцінки за кожним з критеріїв в діапазоні [1;5] для кожної із технологій.

З урахуванням отриманих оцінок прийнято рішення про використання *Python* з фреймворком *FastAPI* для реалізації сервера застосунку та *JavaScript* з фреймворком *Vuetify* для клієнтського ПЗ. В якості СКБД/СУІБ може бути обрана будь-яка з підтриманих в *Python*, але з міркувань простоти реалізації роботи з деревовидними структурами даних була обрана *MongoDB*.

Таблиця 1 – Порівняння інструментальних засобів реалізації ПЗ
Table 1 – Comparison of software implementation tools

Критерій/ Засіб	Крос-платформність	Доступність	Простота розробки	Сума
Java	5 – Виконується у віртуальній машині, що доступна для більшості платформ	5 – Вільна ліцензія, багато безкоштовних інструментів	3 – Висока складність для швидкої розробки	13
JavaScript	5 – Скриптова мова програмування	5 – Вільна ліцензія, відкритий код, багато безкоштовних інструментів	4 – Можуть бути проблеми сумісності при виконанні в різних браузерах	14
ASP.NET	3 – Виконується у віртуальній машині з обмеженою підтримкою	4 – Вільна ліцензія, відкритий код	3 – Висока складність для швидкої розробки	10
PHP	5 – Скриптова мова програмування	5 – Вільна ліцензія, відкритий код, багато безкоштовних інструментів	5 – Підтримує SDLC швидкої розробки	15
Python	5 – Скриптова мова програмування	5 – Вільна ліцензія, відкритий код, багато безкоштовних інструментів	5 – Підтримує SDLC швидкої розробки	15

Таким чином, за результатами огляду різних технологій реалізації, діаграма розгортання застосунку котра реалізує пошук з використанням запропонованого методу *SE*, набула наступного вигляду (рис. 8).

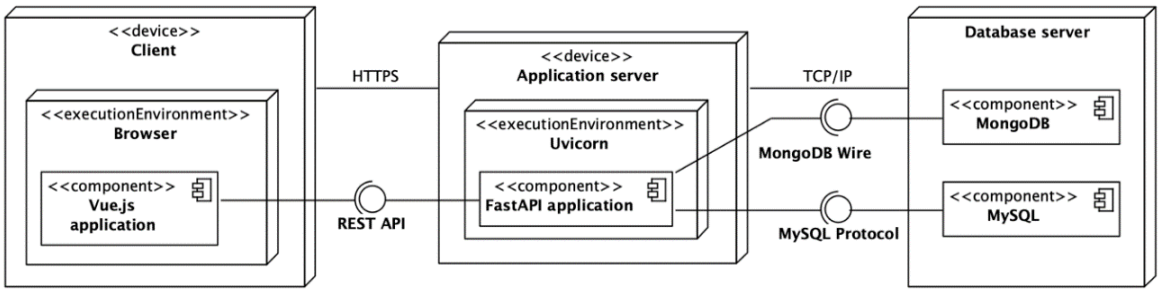

Рис. 8 – Уточнена діаграма розгортання для застосунку
Fig. 8 – Adjusted application deployment diagram

Вимоги до апаратного і програмного забезпечення

Обраний набір інструментів реалізації ПЗ визначає початкові вимоги до ПЗ і апаратного забезпечення кожного з вузлів майбутньої системи. Для коректної роботи ПЗ вузла *«Client»* необхідно забезпечити функціонування сучасної версії веб-браузера, тому в залежності від операційної системи (ОС) висуваються наступні вимоги (див.табл. 2).

Таблиця 2 – Вимоги до вузлу *«Client»* в залежності від ОС
Table 2 – Requirements for the "Client" node depending on the OS used

ОС	Вимоги
Windows	Win 10 або новіша, 4 ГБ ОЗУ (або більше), не менше 128 ГБ дискового простору. Браузер Google Chrome або будь-який інший сумісний.
Mac	MacOS Catalina 10.15 або новіша, 4 ГБ ОЗУ, не менше 128 ГБ дискового простору. Браузер Google Chrome або будь-який, сумісний.
Linux	64-бітна система Ubuntu 18.04, Debian 10, OpenSUSE 15.2, Fedora Linux 32 або новіші, 4 ГБ ОЗУ (чи більше), не менше 128 ГБ дискового простору, підтримка інструкцій SSE3. Браузер Google Chrome або будь-який інший сумісний.
Android	Android 8.0 або новіша, 1 ГБ оперативної пам'яті (чи більше), 32 ГБ дискового простору (або більше). Браузер Google Chrome або будь-який інший, сумісний.

ПЗ вузла *«Application server»* використовує мову *Python*, яка дозволяє виконувати однаковий код на різних платформах, тому для цього вузла висуваються однакові вимоги для всіх підтриманих платформ (*Windows, Linux, MacOS або будь-яка інша, що підтримує Python 3.11*):

1. 4 ГБ оперативної пам'яті (ОЗУ), не менше 128 ГБ дискового простору.
2. Інтерпретатор *Python* 3.11 (або сумісний).
3. Бібліотеки залежностей: FastAPI 0.103.2 (або сумісна), *PyMongo* 4.5.0 (*або сумісна*), python-dotenv 1.0.0 (*або сумісна*).

Для вузла сервера баз даних необхідно забезпечити функціонування сучасної версії СКБД *MongoDB*, тому висуваються такі вимоги:

1. 4 ГБ оперативної пам'яті (*або більше*), не менше 10 ГБ дискового простору (в залежності від обсягу даних що зберігаються, ця вимога може бути більше).
2. Процесор 64-бітної архітектури.
3. MongoDB 7.0.1 community (*або будь-яка сумісна*).

При цьому, хмарне рішення *MongoDB Atlas* надає декілька варіантів готових віртуальних машин для простого розгортання СКБД. Один з таких варіантів – *«M0 Sandbox»*. Він має суттєві обмеження в ресурсах, проте задовольняє мінімальним вимогам до програмного та апаратного забезпечення прототипу.

Процес інсталяції програмного забезпечення

Для ознайомлення з ПЗ прототипу користувачам необхідно перейти за адресою *https://se-uavs.lilikovych.name* – окрім браузера *Google Chrome* (*або сумісного*) з боку користувача встановлювати будь-яке інше ПЗ не потрібно.

Вихідний код програмного забезпечення прототипу доступний за адресами:
https://github.com/SerhiiKarazinUni/diploma-uavs-frontend (інтерфейс користувача).
https://github.com/SerhiiKarazinUni/diploma-uavs-backend (ПЗ вузла серверу застосунку).

Інсталяція сервера застосунку

Для інсталяції власної копії ПЗ вузла *"Application server"* необхідно:

1. Встановити *Git* та інтерпретатор *Python* версії 3.11 (*або новішої сумісної*).
2. Виконати команду *«git clone https://github.com/SerhiiKarazinUni/diploma-uavs-backend»*, перейти в директорію *«diploma-uavs-backend»*.
3. Виконати команду встановлення залежностей *«py -m pip install -r requirements.txt»*.
4. Окремо встановити сервер *Uvicorn* за допомогою команди *«py -m pip install uvicorn[standard]»*.
5. Перейменувати файл *"example.env"* в *".env"*, налаштувати актуальні значення змінних оточення.
6. Після збереження файлу, сервер застосунку можна запустити командою *"py -m uvicorn main:app --port 3000 --host 127.0.0.1"*.

Після виконання цих дій, ПЗ сервера застосунку буде доступне по протоколу *HTTP* на порт 3000.

Інсталяція користувальницького інтерфейсу застосунку

Для встановлення ПЗ користувальницького інтерфейсу необхідно:

1. Встановити Git та актуальну версію Node.JS.
2. Виконати команду *"git clone https://github.com/SerhiiKarazinUni/diploma-uavs-frontend"*, перейти в директорію *"diploma-uavs-frontend"*.
3. Виконати команду встановлення залежностей *"npm install"*.
4. Відкрити для редагування файл *"src/plugins/index.js"*, та в рядках 14 і 15 налаштувати відповідну адресу сервера застосунку. Крім того, необхідно переконатися, що в ряд-

ку 16 встановлено актуальне значення токена клієнта, таке що відповідає *"API_TOKEN"* з налаштувань сервера. Зберегти файл.

5. Виконати команду *"npm run dev -- --port 80"* .

Після виконання цих дій, ПЗ вузла «Client» буде доступне по протоколу *HTTP* на порту 80, як показано на рис. 9.

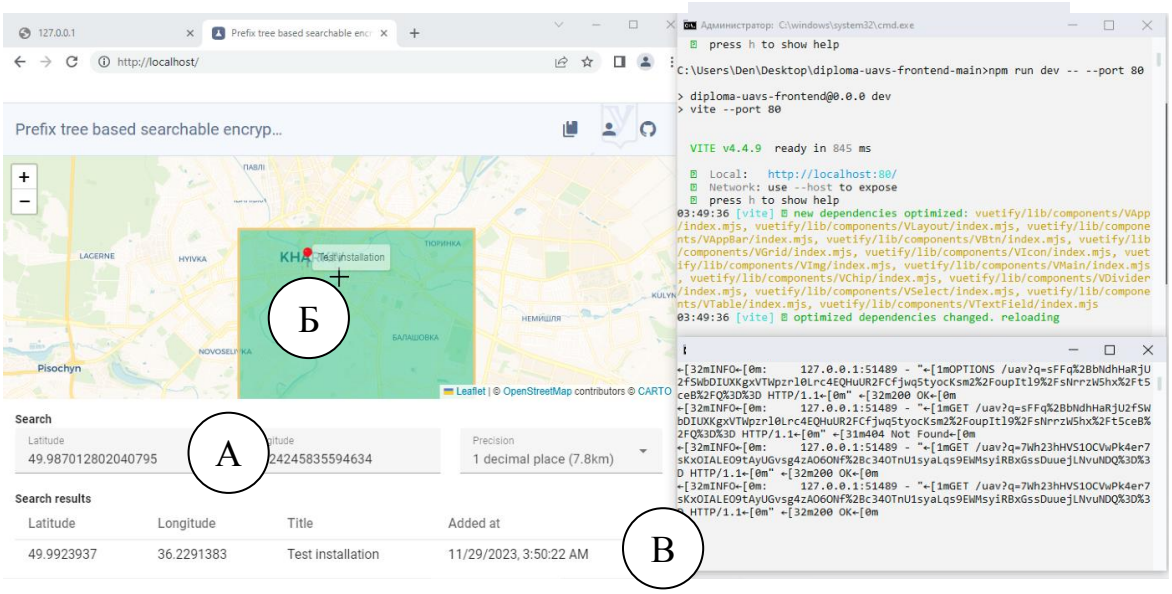

Рис. 9 – Вікно браузера з користувальницьким інтерфейсом та вікна терміналів
Fig. 9 – Browser window with user interface and terminals windows

Літерами на рисунку позначено: (А) панель пошуку, де можна задати або переглянути поточні координати центральної точки, навколо якої відбувається пошук, (Б) інтерактивну карту, яка відображає поточну центральну точку, зону пошуку, та дозволяє подвійним кліком змінити місцезнаходження центральної точки, (В) результати пошуку у вигляді таблиці. Додатково в інтерфейсі застосунку доступне поле вводу для створення нового об'єкту на карті, та вікно відомостей про обмін даних між клієнтським застосунком та сервером застосунку, де можна бачити, що в ході пошуку не відбувається обміну даними пошукового запиту у відкритому вигляді, і дані про збережені документи (*відомості про точки на карті*) надходять до клієнтського застосунку в зашифрованому вигляді. Для тестування швидкодії ПЗ прототипу, використано *Apache JMeter*, який був сконфигурирован, як показано на рис. 10.

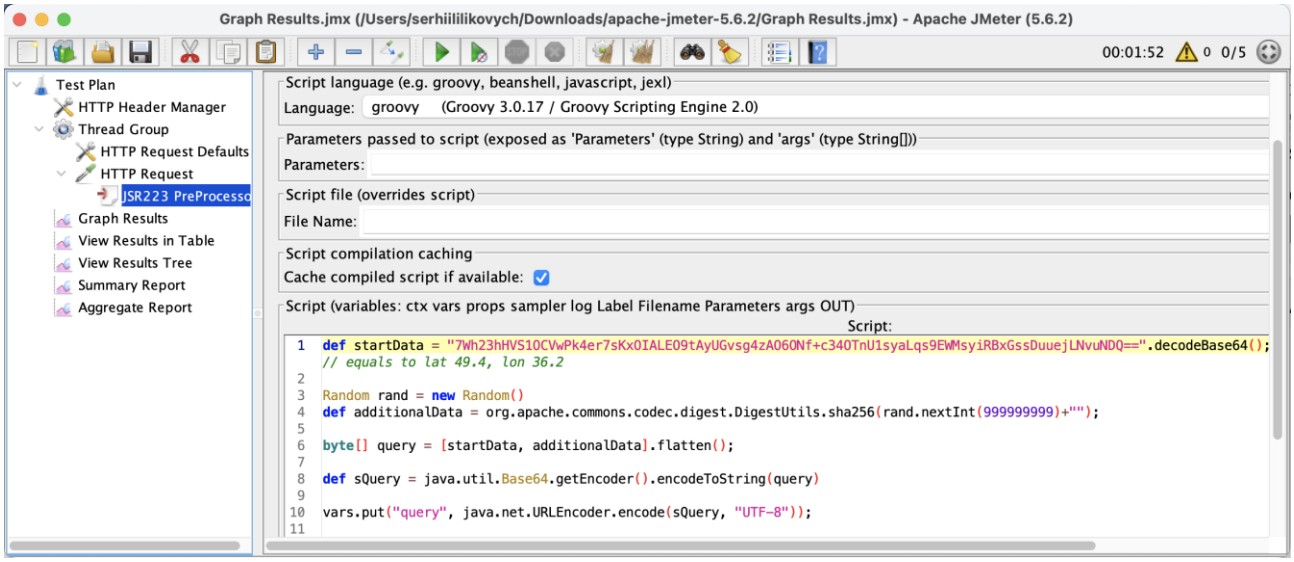

Рис. 10 – Інтерфейс *Apache JMeter* в ході тестування швидкодії
Fig. 10 – *Apache JMeter* interface during performance testing

З рис.10 видно, що тестування складається з одної групи потоків (*thread group*), яка виконує *HTTP*-запити таким чином, що кожен наступний відрізняється від попередніх, за що відповідає спеціально створений об'єкт *"JSR223 PreProcessor"* – програма, яка генерує випадкові, але коректні «*лазівки*». Група потоків сконфігурована таким чином, що імітує одночасне виконання пошукових запитів від імені п'яти користувачів. Кожен користувач виконує по 200 запитів, тобто сумарне навантаження досягає 1000 пошукових запитів. Графік пропускної здатності прототипу ПЗ за результатами тестування швидкодії *Apache JMeter*, показано нижче, на рис.11. Як можна бачити, пропускна здатність ПЗ прототипу – 535.26 запити на 1 хв, а отже приблизно 8.9 запитів на секунду, або 0.11 секунд на один запит.

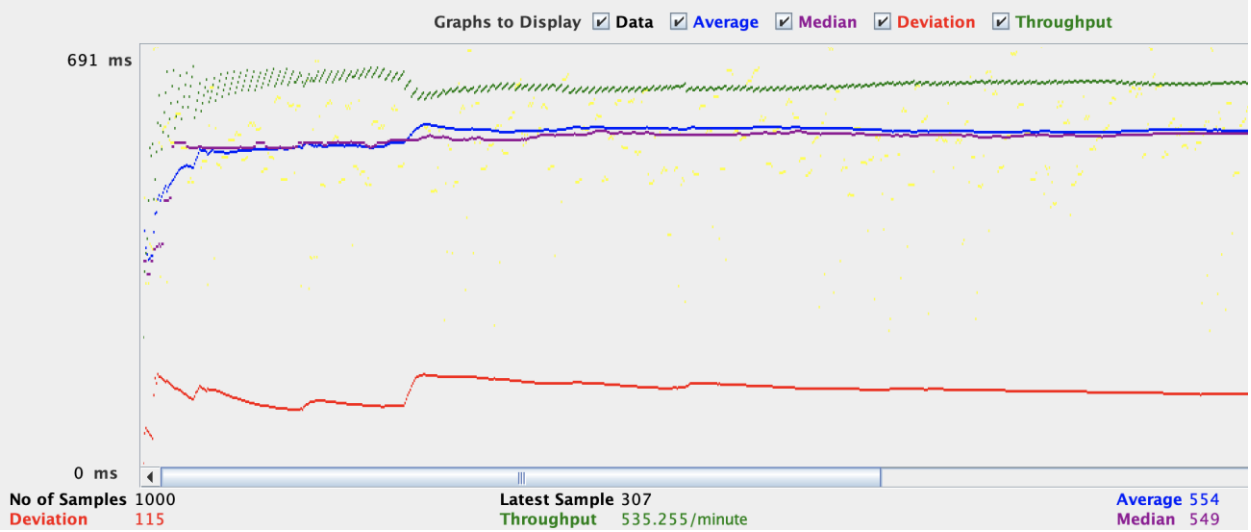

Рис. 11 – Пропускна здатність прототипу ПЗ (*за результатами Apache JMeter)*
Fig. 11 – Bandwidth of the software prototype *(according to the results of Apache JMeter)*

3. Висновки

При розробці ПЗ, яке забезпечує пошук за допомогою розглянутого методу були:

1. Сформульовані вимоги до нього, що враховують результати проведеного аналізу загроз та відомих моделей безпеки.
2. Визначено архітектуру та інструментальні засоби для реалізації ПЗ.
3. Розроблено програмну модель відповідного прототипу додатку/рішення.
4. Проведено оцінку продуктивності запропонованого рішення. Середній час обробки запиту під час тестування швидкодії склав 0.11 с.

Список літератури

[1] SoK: Cryptographically Protected Database Search / Benjamin Fuller [et al.] // 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017. – Mode of access: https://doi.org/10.1109/sp.2017.10 (date of access: 05.01.2024)

[2] Technique for Searching Data in a Cryptographically Protected SQL Database / Vitalii Yesin [et al.] // Applied Sciences. – 2023. – Vol. 13, no. 20. – P. 11525. – Mode of access: https://doi.org/10.3390/app132011525 (date of access: 05.01.2024)

[3] Azraoui M. Framework for Searchable Encryption with SQL Databases / Monir Azraoui, Melek Önen, Refik Molva // 8th International Conference on Cloud Computing and Services Science, Funchal, Madeira, Portugal, 19–21 March 2018. – Mode of access: https://doi.org/10.5220/0006666100570067 (date of access: 05.01.2024)

[4] A Survey of Provably Secure Searchable Encryption / Christoph Bösch [et al.] // ACM Computing Surveys. – 2015. – Vol. 47, no. 2. – P. 1–51. – Mode of access: https://doi.org/10.1145/2636328 (date of access: 05.01.2024)

[5] Dynamic Verifiable Encrypted Keyword Search Using Bitmap Index and Homomorphic MAC / Rajkumar Ramasamy [et al.] // 2017 IEEE 4th International Conference on Cyber-Security and Cloud Computing (CSCloud), New York, NY, 26–28 June 2017.– Mode of access: https://doi.org/10.1109/cscloud.2017.47 (date of access: 05.01.2024)

[6] Лілікович С.О. Метод пошуку за префіксом в зашифрованих базах даних / Лілікович С.О., Єсін В.І. // Комп'ютерне моделювання в наукоємних технологіях : Зб. наук. пр. міжнар. науково-техн. конф., Харків, 25–27 жовт. 2023 р.– С. 105–108.

[7] Handa R. Searchable encryption: A survey on privacy preserving search schemes on encrypted outsourced data / Rohit Handa, C. Rama Krishna, Naveen Aggarwal // Concurrency and Computation: Practice and Experience. – 2019. – P. e5201. – Mode of access: https://doi.org/10.1002/cpe.5201 (date of access: 05.01.2024)
[8] Pattern-Oriented Software Architecture Volume 1: A System of Patterns / Frank Buschmann [et al.]: Wiley, 1996. – 476 p.

Received: on October 2023. **Accepted:** on November 2023.

Authors:
Serhii Lilikovych, CSD Student (magistrate), Department of Security of Information Systems and Technologies, V.N. Karazin Kharkiv National University, Ukraine.
ORCID: https://orcid.org/0009-0003-4407-1281
E-mail: serhii.lilikovych@student.karazin.ua
Vitalii Yesin, Doctor of Engineering Sciences, Full Professor, Department of Security of Information Systems and Technologies, V.N. Karazin Kharkiv National University, Ukraine.
ORCID: https://orcid.org/0000-0003-1977-7269
E-mail: v.i.yesin@karazin.ua

Features of software implementing the prefix search method in cryptographically protected databases.

Abstract. The article addresses the specific considerations associated with the development of software implementing the prefix search method in cryptographically protected databases. This method is a variant of symmetric searchable encryption, which allows search among the encrypted data. The prefix search method allows searching for prefixes among encrypted data without the need for decryption. Such an approach resolves the issue of maintaining data confidentiality stored on remote or cloud servers. However, its usage introduces a set of issues that must be considered during the development of the corresponding software. The paper analyzes the requirements for software that implements the prefix search method, defines the software architecture, and justifies the choice of technologies and tools for software implementation, including *ASP.NET*, *Java*, *JavaScript*, *PHP*, *Python* programming languages, *MongoDB* database management system, and the *FastAPI* framework. A description of the deployment process of the corresponding software is provided. To assess the performance of the developed software, the well-known *Apache JMeter* tool for conducting load testing was utilized. The obtained performance evaluations of the proposed solution indicate acceptable time delays in processing relevant data search queries.

Keywords: *Database, Searchable Encryption, Confidentiality, Software.*