УДК 004.4`242/.8

РЕАЛІЗАЦІЯ ФУНКЦІЙ АВТОРИЗОВАНОГО ДОСТУПУ ДО ХМАРНОГО СХОВИЩА ФАЙЛІВ З ВИКОРИСТАННЯМ GITHUB COPILOT

Лебедь Ю.М., інженер[1]; Суслов С.В., к.т.н., доцент [2]
[1] CCSI Ltd; [2] Національний університет кораблебудування імені адмірала Макарова
[1] Ірландія, Дублін; [2] Україна, Миколаїв
[1] yuriy.lebed@ccsi.ie; [2] sergiy.suslov@nuos.edu.ua

Анотація. *Розглянуто процес застосування штучного інтелекту при розширення функціоналу системи автоматизованого проектування CCSI. Додано функції створення і роботи користувачів з віртуальною індивідуальною файловою системою при розташуванні файлів проектів у загальному хмарному середовищі Azure File Share.*
Ключові слова: *штучний інтелект; хмарне сховище; архітектура програмного забезпечення; тестування.*

Вступ. Компанія CCSI є одним з провідних розробників програмного забезпечення для автоматизованого проектування систем опалення, вентиляції та кондиціонування повітря (HVAC) у світі. Програмне забезпечення для вибору вентиляційних установок (AHU) можна налаштувати відповідно до потреб клієнтів і використовувати онлайн й на ПК [1]. Карта, наведена на рисунку нижче, дає уяву про використання системи протягом однієї доби.



Рисунок – Географічне розташування працюючих користувачів системи AHU

Данні зі створених клієнтами проектів зберігаються у загальному хмарному сховищі. Проектна документація формується шляхом занесення цих даних до шаблонів відповідних документів. Досвід використання системи показав доцільність додати клієнтам можливості з редагування шаблонів. що потребує реалізації авторизованого доступу до файлів у хмарному сховищі

Мета роботи – розширити сервіси для клієнтів AHU шляхом реалізації функцій авторизованого доступу і одночасно відпрацювати технології застосування штучного інтелекту у середовищі розроблення.

Основна частина. Програмне забезпечення проєктування вентиляційних установок AHU розроблено компанією CCSI на основі архітектури ASP.NET MVC [2]. Додавання функцій авторизованого доступу потребувало внесення змін до користувацького інтерфейсу (створення, перегляду, перейменування, завантаження й видалення тек і файлів проектної документації), контролеру і файлового обміну на боці провайдера. GitHub Copilot (з підтримкою LLM) було використано для наступного.

1) Розмітка та скрипти Razor view— UI + JavaScript для навігації по теках, завантаження файлів та викликів API через AJAX.

2) Операції контролера (List, Download, Upload, CreateFolder, DeleteFile, RenameFile), які за шаблоном "Dependency Injection" (DI) [3] делегують роботу до реалізацій через абстракції.

3) На боці провайдера - інтерфейс і реалізації для операцій контролера.

4) Реєстрації DI.

5) Генерація unit-тестів.

6) Налагодження JavaScript стосовно таймінгу подій, делегування подій, імен полів JSON (casing), генерації посилань для завантаження і обробки “anti forgery token”.

Застосування GitHub Copilot полягало у виконанні наступних кроків.

1) Початковий запит згенерувати Razor view для менеджера файлів — Copilot створив код HTML + JS.

2) Запит на потрібні дії контролеру— Copilot спочатку згенерував дії: Azure-орієнтовані, потім запасну файлову реалізацію.

3) Створення абстракції і відповідної реалізації для операцій контролера за шаблоном DI, щоб він був незалежний від бекенда.

4) Інтеграція з Azure Files.

5) Налаштування інверсії залежності DI.

6) Налагодження JavaScript

7) Налаштування метаданих стосовно файлів,

На цей час додана функціональність піддана бета тестуванню і знаходиться у роботоспроможному стані.

Досвід цього процесу розвитку програмної системи з використанням ШІ, а саме GitHub Copilot, можна сформулювати так.

1) Варто робити запити до Copilot невеликими і конкретними, генерувати по одному файлу чи методу й швидко перевіряти їх виконання,

2) Доцільно обмежувати функції контролеру, делегуванням файлових операцій провайдеру, що сприяє тестованості і замінам у бекенді.

3) Потрібно при кожному запиті додавати контекст, як то відкриті файли, інтерфейси, це налаштовує Copilot на стиль коду.

4) Доцільно просити Copilot генерувати тести разом з кодом —це пришвидшує створює моків та набір перевірок.

Висновки. GitHub Copilot виявився ефективним при швидкому прототипуванні та реалізації функцій у комерційному ASP.NET проекті і використаному у цій роботі підході: генерувати UI, генерувати дії контролеру, вводити абстракцію провайдера, реалізувати провайдери (Azure + локальний), реєстрація в DI, ітеративне налагодження JS і серверної JSON-форми, додавання тестів. Втім кінцева відповідальність за перевірку сумісності, безпеки й продуктивності залишається за розробником.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] *Air Handling Units - CCSI Ltd*. (n.d.). CCSI Limited, HVAC and Engineering Software - CCSI Ltd. https://www.ccsi.ie/products/air-handling-units.html
[2] *ASP.NET MVC Pattern | .NET*. (n.d.). Microsoft. https://dotnet.microsoft.com/en-us/apps/aspnet/mvc
[3] *Dependency injection - .NET*. (n.d.). Microsoft Learn: Build skills that open doors in your career. https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection

Lebed Y.M., Suslov S.V.

Implementation of functions for authorized access to cloud file storage using GitHub Copilot

Abstract. *The process of applying artificial intelligence to expand the functionality of the CCSI automated design system is considered. The added functions serve users for creating and working with a virtual individual file system when project files are placed in the Azure File Share cloud environment.*

Keywords: *artificial intelligence; cloud storage; software architecture; testing.*