УДК 004.42

ЧАСТКОВА АВТОМАТИЗАЦІЯ ЕТАПІВ РОЗРОБКИ ПЗ ЗА ДОПОМОГОЮ ФУНКЦІОНАЛУ AI

Кондус О. С., науковий керівник Ткаченко О. М.

У сучасних умовах стрімкої еволюції IT-сфери, дедалі актуальнішим стає використання інтелектуальних інструментів для автоматизації рутинних завдань розробки програмного забезпечення. Зі зростанням обсягів коду, потребою у швидкому оновленні документації, створенні тестів та забезпеченні якості виникає необхідність інтеграції новітніх AI-рішень безпосередньо у процеси розробки. У межах роботи було досліджено можливості провідних мовних моделей – GPT-4, Claude, Deepseek, Gemini та Grok – з метою визначення їхньої ефективності в автоматизації типових завдань розробника. Основною метою дослідження стало створення розширення для середовища Visual Studio Code, яке забезпечує зручну взаємодію з мовними моделями під час написання коду.

Плагін реалізовано мовою JavaScript із використанням API OpenAI та інших сторонніх провайдерів, що дозволяє здійснювати вибір мовної моделі залежно від конкретного завдання. Користувач може викликати необхідну функцію як із контекстного меню, так і через командний рядок редактора. Наприклад, достатньо виділити функцію, клацнути правою кнопкою миші та обрати команду «Генерувати документацію», після чого плагін сформує структурований коментар, який одразу можна інтегрувати у код. Для формування запитів використовується система динамічних підказок, адаптованих під різні задачі: від написання документації у форматі Javadoc до рефакторингу коду згідно з принципами SOLID. Це забезпечує гнучкість і адаптивність взаємодії з моделлю відповідно до потреб користувача.

Особливу увагу приділено підтримці кількох мовних моделей. ChatGPT-3.5 забезпечує найшвидшу генерацію відповідей за помірну вартість, що робить його ефективним інструментом для щоденної роботи. Grok, хоч і працює повільніше, здатен виконувати глибокий технічний аналіз, тоді як Deepseek є найдешевшим варіантом, хоч і з відчутними затримками у відповіді. У плагіні реалізовано як ручний, так і автоматичний вибір моделі, що дозволяє досягти балансу між швидкістю, вартістю та глибиною аналізу залежно від контексту завдання.

Інтерфейс плагіна побудовано з використанням Webview API, що дозволяє візуалізувати згенеровані блоки документації, коду чи тестів безпосередньо у вікні розширення. Користувач має змогу переглянути результат, перегенерувати його, скопіювати або вставити у файл одним кліком. Архітектура плагіна побудована з чітким розділенням відповідальностей між частинами, що відповідають за інтерфейсну взаємодію, та частинами, що формують запити й обробляють відповіді. Такий підхід забезпечує масштабованість і гнучкість рішення.

Серед ключових функцій плагіна – генерація документації до методів, рефакторинг відповідно до принципів SOLID, створення юніт-тестів, генерація коду за сигнатурою методу, пояснення логіки коду у вигляді коментарів, а також перевірка відповідності принципам об'єктно-орієнтованого дизайну із рекомендаціями та автоматичним виправленням. На рисунку 1 наведено приклад використання функції генерації документації, де класс EmployeeController автоматично доповнюється коментарями у форматі Javadoc, що охоплюють опис, призначення параметрів та типи повернених значеннь. Це значно прискорює створення якісної документації та покращує читабельність коду.

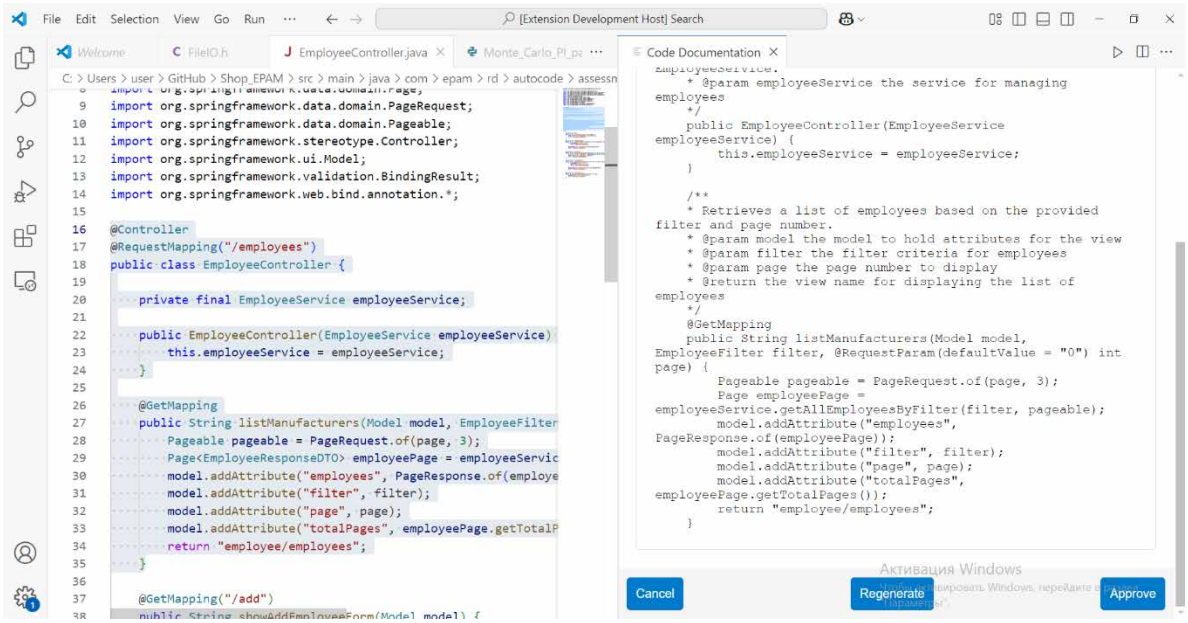

Рисунок 1 – Функція генерації документації

Запропоноване рішення дає змогу значно зменшити час, витрачений на документацію, тестування та рефакторинг, а також підвищити якість архітектури завдяки автоматизованому дотриманню принципів розробки. Нові члени команди швидше ознайомлюються з кодом завдяки коментарям та структурованій документації. У майбутньому плагін може бути розширено за рахунок інтеграції з інструментами аналізу коду, такими як SonarQube чи Checkstyle, що дозволить автоматично діагностувати помилки та генерувати пропозиції щодо їх усунення. Планується також підтримка інших форматів документації – reStructuredText, Markdown тощо, що забезпечить універсальність плагіна під різні потреби команд. Ще одним напрямом розвитку є інтеграція з CI/CD-системами, як-от Jenkins чи GitLab CI, де плагін зможе автоматично створювати коментарі до змін або запускати генерацію тестів у рамках конвеєра.

Таким чином, автоматизація рутинних етапів розробки за допомогою мовних моделей відкриває нову парадигму в інженерії програмного забезпечення, де швидкість, якість і автоматизація стають взаємопов’язаними елементами ефективного робочого процесу. Розробники можуть зосередитися на вирішенні творчих і складних задач, довіривши рутинні операції інтелектуальним помічникам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Code documentation: Benefits, challenges, and tips for success. Swimm. URL: https://swimm.io/learn/code-documentation-benefits-challenges-and-tips-for-success/.

2. Introducing ChatGPT. OpenAI. URL: https://openai.com/index/chatgpt/.

3. Martin, R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.

4. Microsoft. Visual Studio Code API Documentation. URL: https://code.visualstudio.com/api.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ РОЗРОБКИ КОМП'ЮТЕРНИХ СИСТЕМ

ЗБІРНИК НАУКОВИХ ПРАЦЬ ЗА МАТЕРІАЛАМИ
VII ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
СТУДЕНТІВ І АСПІРАНТІВ
24 квітня 2025 року

Київ 2025

УДК 004

Відповідальна за випуск: М.І. Лендєл

Збірник наукових праць за матеріалами VII Всеукраїнської науково-практичної конференції студентів і аспірантів «ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ РОЗРОБКИ КОМП'ЮТЕРНИХ СИСТЕМ '2025», 24 квітня 2025 року, НУБіП України, Київ. – 452 с. (електронне видання)

Відповідальність за зміст публікацій несуть автори.