УДК 004.8
DOI https://doi.org/10.32782/tnv-tech.2024.2.6

АВТОМАТИЗАЦІЯ ПРОЄКТУВАННЯ АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРИЙОМУ КОМУНАЛЬНИХ ПЛАТЕЖІВ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Козачок Ю. А. *– магістр спеціальності «Комп'ютерна інженерія»*
Чорноморського національного університету імені Петра Могили
ORCID ID: 0009-0005-6430-3961

Автоматизація проєктування архітектури програмного забезпечення за допомогою генеративного штучного інтелекту забезпечує ефективне вирішення проблем, які виникають у сучасній розробці програмного забезпечення. У цій статті подано дослідження використання генеративного штучного інтелекту під час проєктування архітектури програмного забезпечення для покращення процесів шляхом використання єдиного стандарту. За допомогою генеративного штучного інтелекту архітектори програмного забезпечення можуть оптимізувати проєкти архітектури відповідно до підвищених вимог до складності та якості сучасних інформаційних систем. Під час роботи були вивчені сучасні дослідження, присвячені використанню нотації проєктування архітектури інформаційної системи. Було проаналізовано нотацію C4, висвітлено та описано основні компоненти даної нотації, які дають змогу привести до єдиного ста`ндарту побудову архітектури інформаційної системи. Окрім того, було досліджено застосування інструментів штучного інтелекту в формуванні архітектури системи. В результаті проведеного аналізу було виконано практичне дослідження, а саме побудована архітектура інформаційної системи прийому комунальних платежів, яка побудована в нотації C4 і складається з чотирьох рівнів: контекст, контейнери, компоненти, код. Кожен рівень було побудовано за допомогою згенерованого коду з використанням інструменту штучного інтелекту ChatGPT в нотації PlantUML та, використовуючи графічний редактор з генерацією діаграм з коду draw.io, відображено у вигляді чотирьох діграм: діаграма контексту, діаграма контейнерів, діаграма компонентів, діаграма коду (класів). В результаті практичного дослідження було проаналізовано побудовану архітектуру інформаційної системи прийому комунальних платежів та надано практичних рекомендацій для покращення якості згенерованих діаграм. Виділено основні переваги застосування ChatGPT та нотації C4 архітекторами для побудови архітектури інформаційної системи та визначено основні недоліки, які потребують подальшої оптимізації.

Ключові слова: *проєктування архітектури програмного забезпечення, штучний інтелект, ChatGPT, автоматизація, стандарти побудови архітектури програмного забезпечення, нотація C4.*

Kozachok Yu. A. Automation of architectural design for an information system for utility payments using artificial intelligence

Automation of software architecture design using generative artificial intelligence provides an effective solution to problems that arise in modern software development. This article presents a study of the use of generative artificial intelligence during the design of software architecture to improve processes through the use of a single standard. With the help of generative artificial intelligence, software architects can optimize architectural projects in accordance with the increased requirements for the complexity and quality of modern information systems. During the work, modern studies devoted to the use of information system architecture design notation were studied. The C4 notation was analyzed, the main components of this notation were highlighted and described, which make it possible to bring the construction of the information system architecture to a single standard. In addition, the use of artificial intelligence tools in the formation of the system architecture was investigated. As a result of the analysis, a practical study was carried out, namely, the architecture of the information system for receiving utility payments was built, which is built in C4 notation and consists of four levels: context, containers, components, code. Each layer was built using generated code using the ChatGPT artificial intelligence tool in PlantUML notation and, using a graphical editor with diagram generation from draw.io code, displayed as four diagrams: context diagram, container diagram, component diagram, code diagram (classes). As a result of the practical study, the constructed architecture of the information system for receiving utility payments was analyzed and practical recommendations were given to improve the quality of the generated diagrams. The main advantages of using ChatGPT and C4 notation

by architects to build the architecture of the information system are highlighted, and the main shortcomings that require further optimization are identified.

Key words: *software architecture design, artificial intelligence, ChatGPT, automation, software architecture construction standards, C4 notation.*

Постановка проблеми. Сучасні інновації в області програмної архітектури докорінно змінили панораму цієї галузі, наповнивши її надзвичайною кількістю передових технологій, які обіцяють ряд переваг: від підвищення ефективності та автоматизації процесів до поліпшення якості продукції, зниження витрат і забезпечення гнучкості та швидкості адаптації до змін на ринку. Технології, такі як великі дані (Big Data), штучний інтелект (Artificial Intelligence – AI) та інші, істотно розширили наявні технічні можливості та забезпечили архітекторам програмного забезпечення необхідні інструменти для вирішення складних бізнес-задач. Внаслідок цього сам процес проєктування стає складнішим і вимагає великих зусиль у вигляді часу та ресурсів. У таких умовах виникає необхідність в пошуку та дослідженні способів часткової автоматизації певних підпроцесів побудови архітектури інформаційних систем з метою оптимізації та підвищення ефективності розробки.

Аналіз останніх досліджень і публікацій. Останні дослідження розглядають різні підходи до автоматизованого проектування архітектури програмного забезпечення, використовуючи інструменти штучного інтелекту. По-перше, методи, що ґрунтуються на обширних базах знань, широко використовуються в цій галузі [1]. Дослідження рекомендує автоматизувати рутинні завдання розробки програмного забезпечення за допомогою моделей Generative AI.

Додатковою сферою досліджень є технологія автоматичного створення альтернативних рішень у процесі архітектурного проєктування [3]. Ці дослідження підкреслюють важливість охоплення початкових етапів архітектурного проєктування. Також проведено дослідження у сфері інженерії програмного забезпечення, що може автоматизувати синтез та складання архітектур програмного забезпечення, наприклад, за допомогою генетичних алгоритмів [5]. Вони можуть використовуватися для створення архітектур програмного забезпечення з використанням різноманітних проєктних шаблонів. Крім того, використання інструментів штучного інтелекту для генерації шаблонів архітектури програмного забезпечення дозволяє створити гнучку та адаптивну систему для задоволення постійнозмінних потреб у розробці програмного забезпечення.

В області проєктування архітектури також проводились дослідження з використання штучного інтелекту. В результаті аналізу було встановлено, що інструменти штучного інтелекту, такі як ChatGPT, сприяють формуванню архітектурного дизайну, який може гнучко реагувати на технологічні виклики зростання та потребу в масштабованості, що можуть виникнути у майбутньому [6].

Отже, ці останні дослідження вказують на те, що інтерес науковців розширюється у напрямку покращення автоматизованого проектування архітектури програмного забезпечення шляхом використання генеративного штучного інтелекту.

Формулювання мети дослідження. Мета цієї статті полягає в дослідженні потенціалу штучного інтелекту для автоматизації процесу проєктування архітектури програмного забезпечення та побудові архітектури інформаційної системи прийому платежів на оплату комунальних послуг з використанням інструменту штучного інтелекту ChatGPT.

Основні задачі, які вирішуються для досягнення заявленої мети:

1) провести аналіз сучасних методологій і нещодавніх досліджень націлених на автоматизацію проєктування архітектури, зокрема використовуючи генеративний штучний інтелект (Generative AI);

2) спроектувати частину архітектури програмного забезпечення за допомогою інструментів штучного інтелекту для демонстрації його можливостей;

3) проаналізувати якість побудованої архітектури інформаційної системи та зробити висновки щодо можливості інтеграції штучного інтелекту в процеси проєктування архітектури програмного забезпечення.

Виклад основного матеріалу. Проектування архітектури програмного забезпечення є ключовим етапом у процесі розробки програмного продукту, що включає вирішення важливих високорівневих завдань, таких як інтеграція системи або її частини у загальну екосистему та забезпечення її функціональності відповідно до технічних та експлуатаційних вимог. Популярні нотації для візуалізації схем архітектури: C4, ArchiMate, SysML, 4+1, AADL. У цій статті сфокусуємось на нотації моделювання архітектури C4. Моя мета – детально розглянути нотацію моделювання архітектури C4 і надати практичні приклади її застосування, що можуть слугувати в якості орієнтира у моделюванні архітектури різноманітних інформаційних систем.

Нотація моделювання C4 – це метод візуалізації архітектури програмного забезпечення, розроблений Саймоном Брауном [8]. Ця нотація використовується для створення діаграм, які ілюструють архітектурну структуру системи. C4 вирішує проблему складності та незрозумілості традиційних архітектурних діаграм. Метою Саймона Брауна було спростити розуміння архітектури для всіх членів команди. Назва “C4” розшифровується як:

- Context – Контекст,
- Containers – Контейнери,
- Components – Компоненти,
- Code – Код.

Ці чотири “C” представляють чотири рівні абстракції (уявлення), які використовуються в цій нотації для візуалізації архітектури (рис. 1).

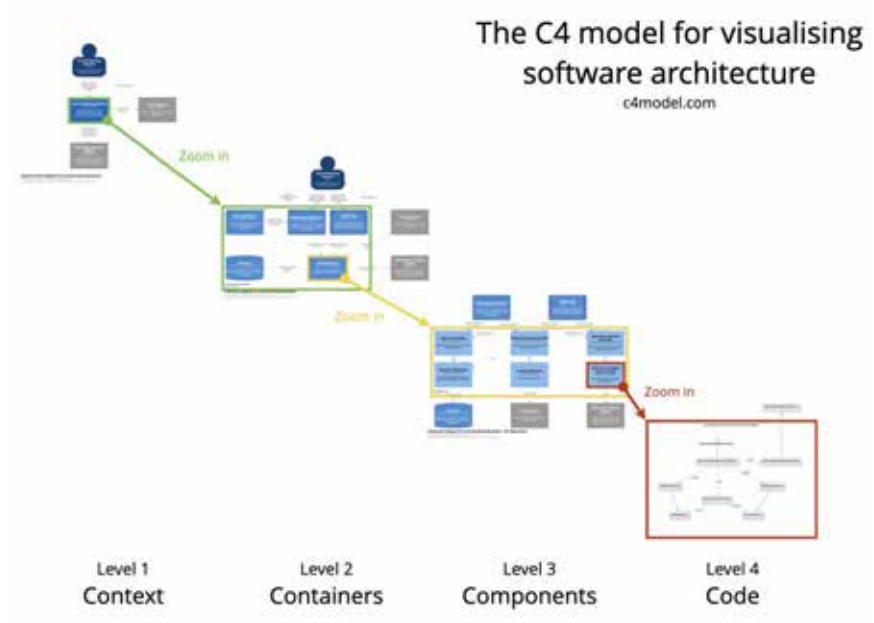

Рис. 1. Рівні абстракції, які використовуються в нотації C4

Рівень 1. Діаграма контексту

Відображає, як система взаємодіє із зовнішніми сутностями (користувачами, зовнішніми системами). Зазвичай використовують на початкових етапах проектування для розуміння загального контексту. Елементи, які використовуються: системи, користувачі, взаємозв'язки.

Рівень 2. Діаграма контейнерів

Описує верхньорівневу архітектуру та технології. Використовується для розуміння технологічного стеку та поділу зон відповідальності.

Елементи, які використовуються: контейнери (наприклад, веб-додатки, бази даних), взаємозв'язки, технології.

Рівень 3. Діаграма компонентів

Деталізує структуру всередині контейнера системи. Тобто детальніше описує лише один контейнер з попереднього рівня. Використовується для проектування та документування внутрішньої структури компонентів системи.

Елементи, які використовуються: компоненти, взаємозв'язки, технології, залежності.

Рівень 4. Діаграма коду

Використовується для низькорівневої деталізації системи. На офіційній сторінці моделі C4 рекомендовано використовувати UML Class diagram/Entity relationship diagram або схожі нотації.

Елементи, які використовуються: класи, інтерфейси, відносини між ними.

Розглянемо процес проєктування архітектури на прикладі інформаційної системи прийому комунальних платежів. Для створення діаграм у нотації C4 можна використовувати:

- графічні інструменти,
- інструменти візуалізації через код з використанням штучного інтелекту.

Метою нашого дослідження є використання інструментів, які дозволяють описувати C4 через код з метою прискорення процесу проектування за рахунок використання штучного інтелекту – ChatGPT. В якості інструменту візуалізації через код було обрано PlantUML.

Роботу з проектування системи слід розпочати із створення діаграми контексту.

Крок 1. Моделювання діаграми контексту

Для створення діаграми контексту необхідно в ChatGPT написати запит з вказанням інформаційної системи для генерації PlantUML коду (рис. 2).

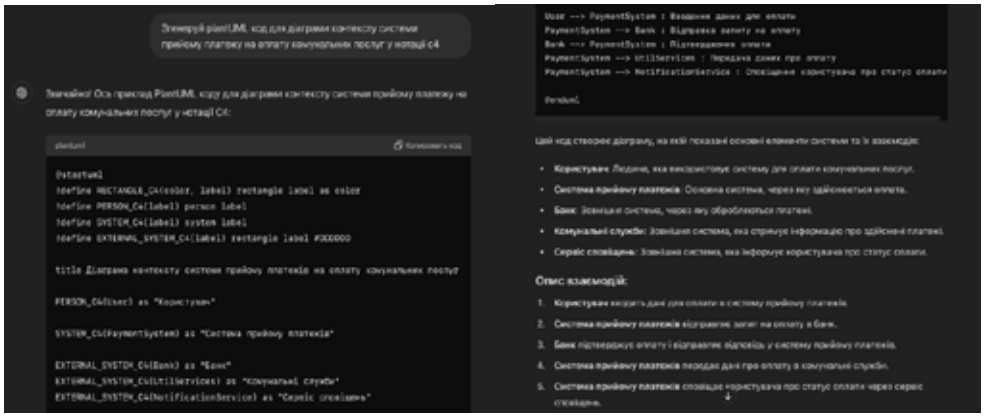

Рис. 2. Генерація PlantUML коду за допомогою ChatGPT

Для побудови діаграми контексту використаємо інструмент візуалізації через код draw.io. Для цього додаємо PlantUML код в конструктор (рис. 3) і генеруємо діаграму контексту (рис. 4).

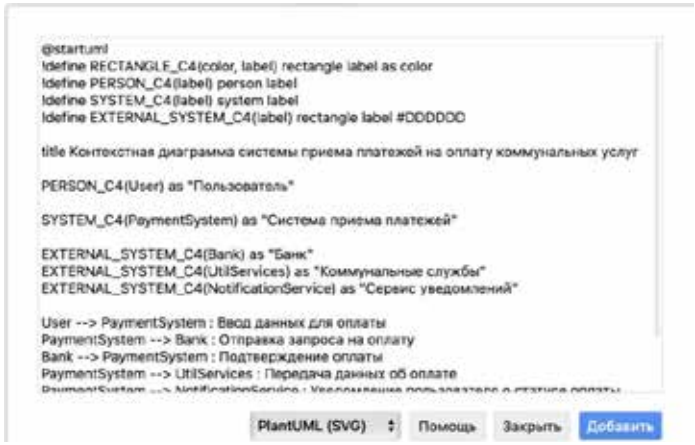

Рис. 3. Конструктор діаграм за допомогою PlantUML коду

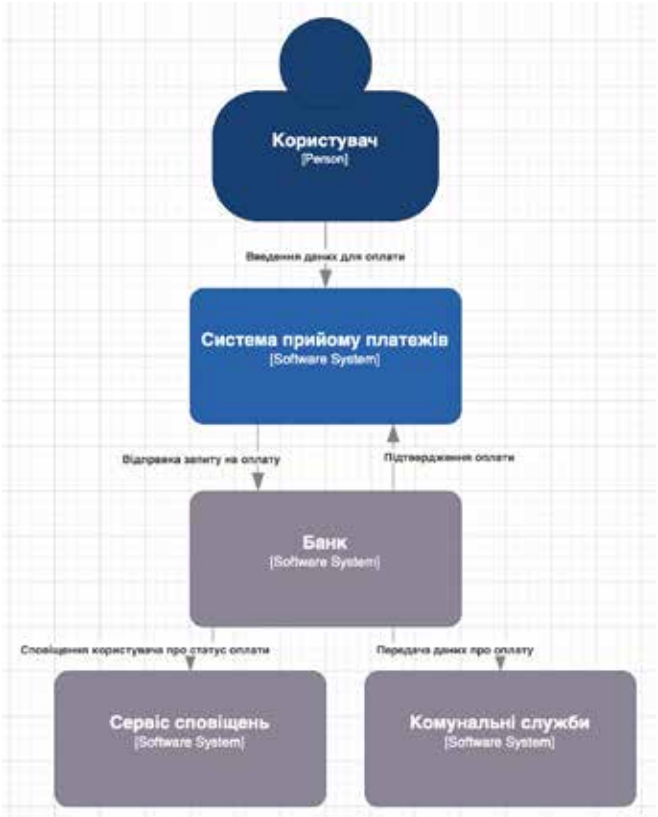

Рис. 4. Згенерована діаграма контексту

Крок 2. Моделювання діаграми контейнерів

Наступним рівнем деталізації в нотації є побудова діаграми контейнерів. Аналогічно першому кроку використаємо ChatGPT для генерації PlantUML коду (рис. 5).

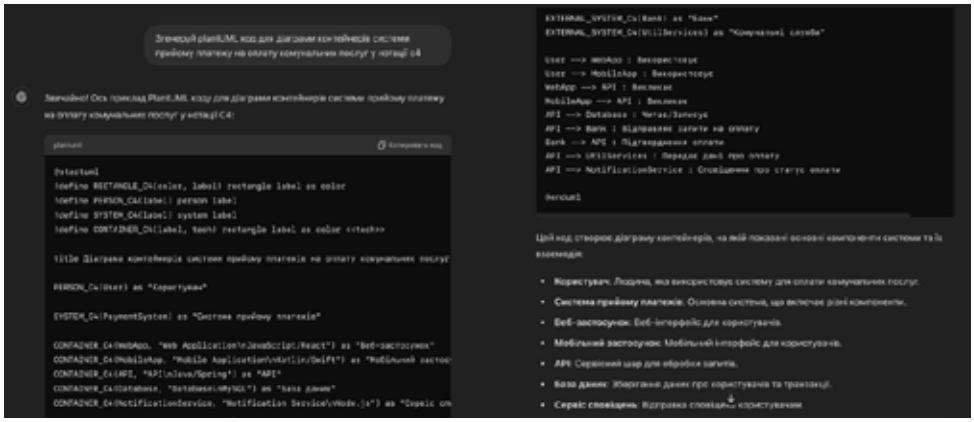

Рис. 5. Генерація PlantUML коду за допомогою ChatGPT

Для побудови діаграми контексту переходимо в draw.io, додаємо PlantUML код в конструктор і генеруємо діаграму контейнерів (рис. 6).

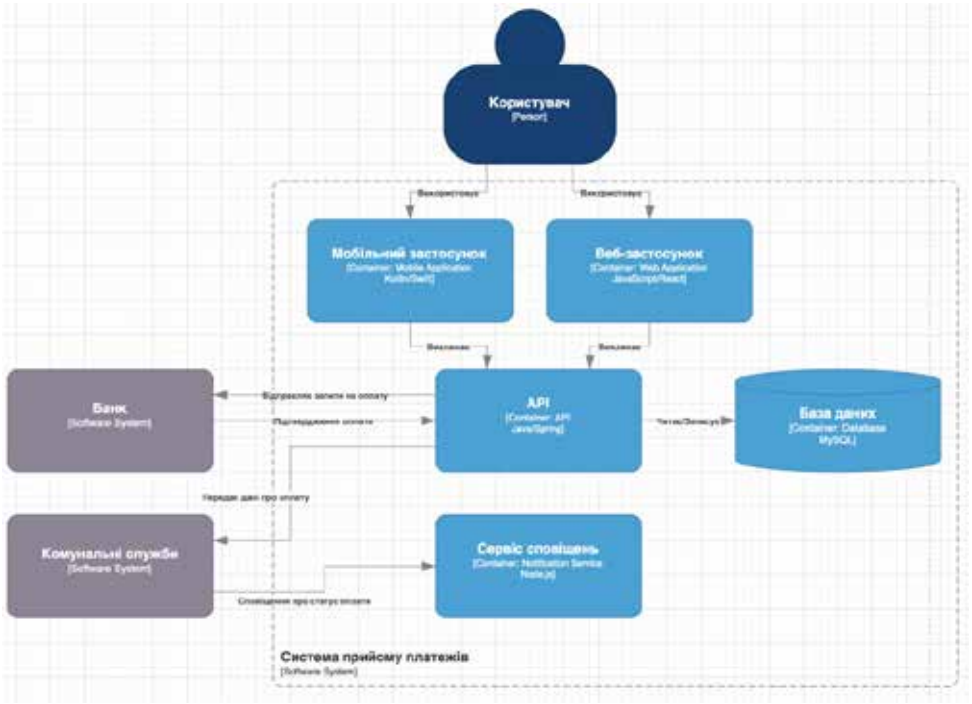

Рис. 6. Згенерована діаграма контейнерів

Крок 3. Моделювання діаграми компонентів

Діаграма компонентів є деталізацією всіх контейнерів з попередньої моделі (діаграми контейнерів). В якості прикладу розглянемо побудову діаграми компонентів з деталізацією контейнера “API”. Згенеруємо PlantUML код в ChatGPT (рис. 7, 8), перейдемо в draw.io, додаємо PlantUML код в конструктор і генеруємо діаграму контейнерів (рис. 9).

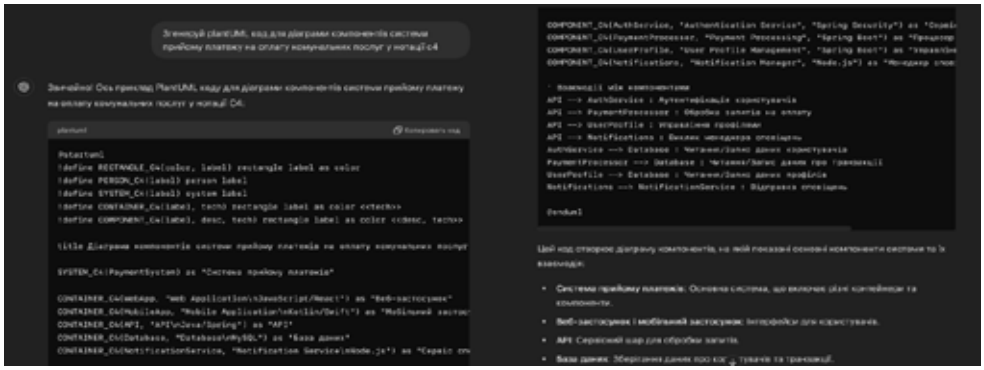

Рис. 7. Генерація PlantUML коду за допомогою ChatGPT

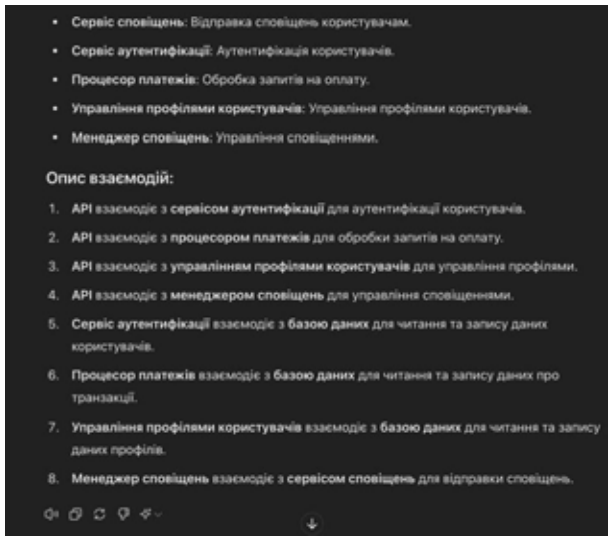

Рис. 8. Генерація PlantUML коду за допомогою ChatGPT

Рис. 9. Згенерована діаграма компонентів

Крок 4. Моделювання діаграми коду

Завершальним етапом проектування архітектури інформаційної системи в нотації C4 є побудова діаграми коду. Це необов’язковий рівень деталізації, який часто доступний за запитом. В ідеалі цю діаграму було б автоматично створено за допомогою середи програмного забезпечення, але якості повноти проектування розглянемо на прикладі найважливіших і складних компонентів інформаційної системи прийому комунальних платежів. Як і в попередніх кроках згенеруємо PlantUML код в ChatGPT (рис. 10, 11), перейдемо в draw.io, додаємо PlantUML код в конструктор і генеруємо діаграму контейнерів (рис. 12).

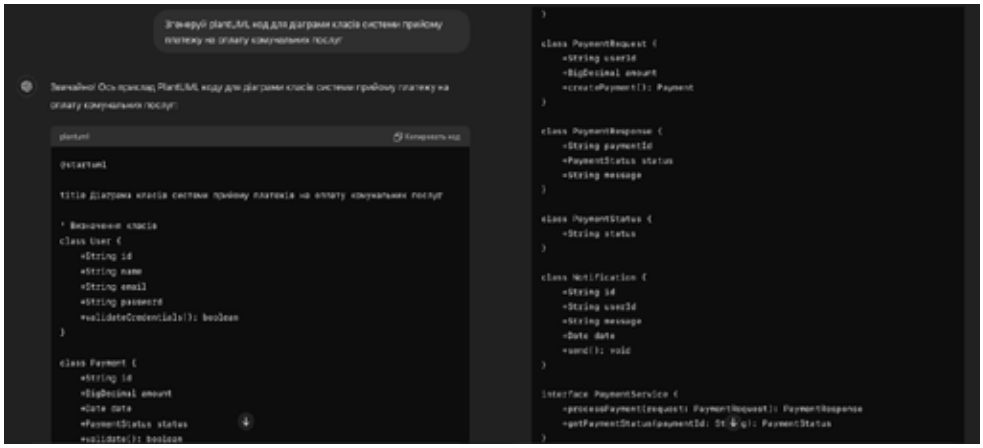

Рис. 10. Генерація PlantUML коду за допомогою ChatGPT

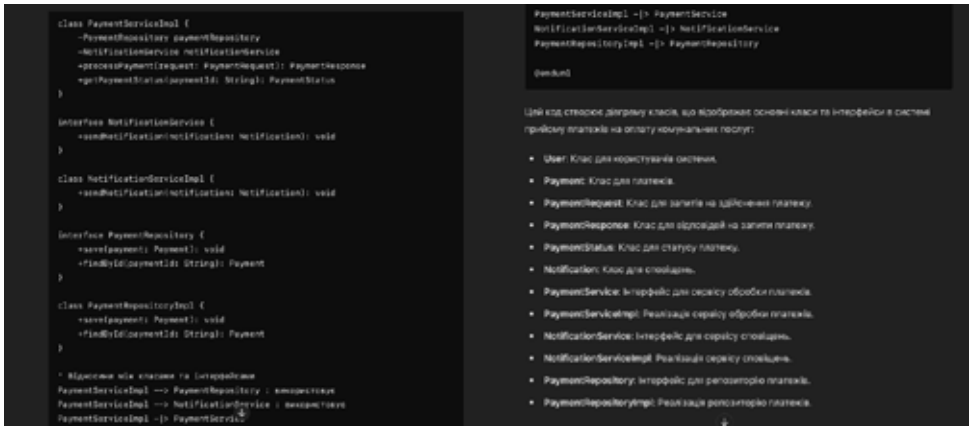

Рис. 11. Генерація PlantUML коду за допомогою ChatGPT

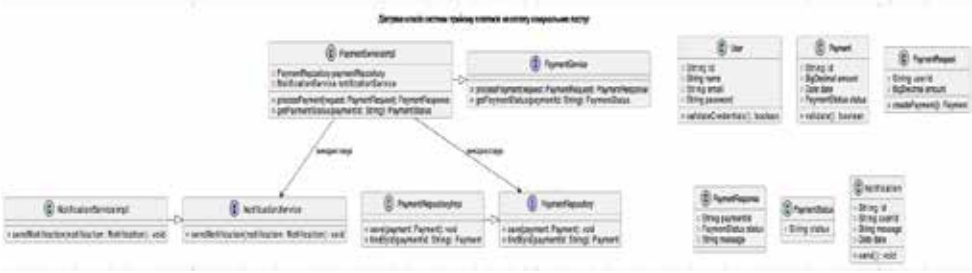

Рис. 12. Згенерована діаграма коду

Проаналізувавши згенеровані діаграми є необхідність втручання в процес архітектора, адже ChatGPT не може повністю врахувати всі особливості інформаційної системи. Також, для зручності користування діаграмами було сформовано ряд правил:

– використання однакового розташування елементів;

– генерація невеликих діаграми (при необхідності можна розбити на декілька та деталізувати);

– діаграма має бути розміром не більше А4, тобто повністю поміщатись на екран монітора або могла бути роздрукована для обговорення;

– якщо в діаграмі занадто багато елементів, об'єднувати подібні елементи і деталізувати в окремих діаграмах.

Не зважаючи на недоліки, отримані діаграми можна використовувати в подальшому проектуванні інформаційної системи прийому комунальних платежів та використовувати ChatGPT для деталізації інших модулів інформаційної системи.

Висновки. В рамках даної роботі було:

1) проведено аналіз сучасних методологій, які використовуються в автоматизації проєктування архітектури, зокрема інструмент генеративного штучного інтелекту ChatGPT;

2) спроектовано частину архітектури інформаційної системи прийому комунальних платежів в нотації С4 за допомогою ChatGPT та draw.io;

3) проаналізовано якість побудованої архітектури інформаційної системи. В рамках цього завдання було виконано аналіз згенерованих діаграм та виявлено недоліки, які враховано в подальшому проектуванні.

Загалом, в ході дослідження було виявленно, що С4 є понятною та доступною нотацією моделювання архітектури. Діаграми легко читаються всіма учасниками за рахунок того, що містить обмежений позначений набір елементів. Використання інструменту штучного інтелекту ChatGPT значно спрощує архітектору задачу побудови якісної архітектури програмного забезпечення та дозволяє автоматизувати частину процесу. Графічний редактор draw.io, підтримуючи інтеграцію з PlantUML, надає змогу за допомогою текстового опису згенерувати графічне зображення та в подальшому інтегрувати з іншими рішеннями.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. Bengio Y., Courville A., Vincent P. Representation learning: a review and new perspectives. ieee transactions on pattern analysis and machine intelligence. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2013. Vol. 35, no. 8. P. 1798–1828.

2. Lee Y.-G. A Study on The Development of Automatic Design Alternatives Generation Technology Used in the Early Stages of Architectural Design. *Asia-pacific Journal of Convergent Research Interchange*. 2021. Vol. 7, no. 3. P. 123–132.

3. Li Z., Liang P., Avgeriou P. Application of knowledge-based approaches in software architecture: A systematic mapping study. *Information and Software Technology*. 2013. Vol. 55, no. 5. P. 777–794.

4. Nascimento N., Cowan R., Alencar P. Artificial intelligence versus software engineers: an evidence-based assessment focusing on non- functional requirements. 2023. P. 1–21.

5. Model Checking Software Architecture Design. *Home Page*. URL: https://doi.org/10.1109/hase.2012.12

6. Vathsavayi S., Hadaytullah H., Koskimies K. Interleaving human and search-based software architecture design. *Proceedings of the Estonian Academy of Sciences*. 2013. Vol. 62, no. 1. P. 16.

7. Harman M., Mansouri S. A., Zhang Y. Search-based software engineering. *ACM Computing Surveys*. 2012. Vol. 45, no. 1. P. 1–61. URL: https://doi.org/10.1145/2379776.2379787.

8. The C4 model for visualising software architecture. *The C4 model for visualising software architecture*. URL: https://c4model.com/#SystemContextDiagram (дата звернення: 25.05.2024).

REFERENCES:

1. Bengio Y., Courville A., Vincent P. (2013). Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 35(8).

2. Lee Y.-G. (2021) A study on the development of automatic design alternatives generation technology used in the early stages of architectural design. *Asia-pacific Journal of Convergent Research Interchange,* 7(3), 123–132. https://doi.org/10.47116/apjcri.2021.03.11

3. Li Z., Liang P., Avgeriou P. (2013) Application of knowledge-based approaches in software architecture: a systematic mapping study. *Information and Software Technology*, 55(5), 777–794. https://doi.org/10.1016/j.infsof.2012.11.005

4. Nascimento N., Cowan R., Alencar P. (2023) Artificial intelligence versus software engineers: an evidence-based assessment focusing on non- functional requirements. https://doi.org/10.21203/rs.3.rs-3126005/v1

5. Model checking software architecture design. (2012) https://doi.org/10.1109/hase.2012.12

6. Vathsavayi S., Hadaytullah H., Koskimies K. (2013) Interleaving human and search-based software architecture design. *Proceedings of the Estonian Academy of Sciences*, 62(1), 16. https://doi.org/10.3176/proc.2013.1.03

7. Harman M., Mansouri S. A., Zhang Y. (2012) Search-based software engineering. *ACM Computing Surveys*. 45(1), 1–61. https://doi.org/10.1145/2379776.2379787ted :24.04.2024 p.

8. *The C4 model for visualising software architecture*. (б. д.). The C4 model for visualising software architecture. https://c4model.com/#SystemContextDiagram