необхідності власноруч прописувати код. Гнучка архітектура застосунку microgen дозволяє швидко додавати нові ресурси та розширювати функціонал.

Підсумовуючі, варто зазначити, що розроблена утиліта не має чітких відповідниках в рамках поставленої задачі. Вона спрощує та пришвидшує розробку в мікросервісів на базі платформи Node.js, що можуть бути згенеровані на базі найпопулярніших технологій цієї платформи. Проте, можливості застосунку обмежуються використанням невеликого переліку найпопулярніших технологій та відсутністю версіонування.

Список використаних джерел

1. Nagpal A. Monolithic vs Microservices Architecture: Advantages, Disadvantages, And Differences. Medium. URL: https://medium.com/@jasminepuno/monolithic-vs-microservices-architecture-advantages-disadvantages-and-differences-2bee6d1da8ca#:~:text=Monolithic%20architecture%20is%20a%20conventional,closely%20connected%20and%20centralized%20system.
2. Global Logic. Mikroservisna arkhitektura dlia pochatkivtsiv. Chastyna I.. GlobalLogic Ukraine. URL: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/
3. Microsoft. What is infrastructure as code (IaC)? - Azure DevOps. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code.
4.Naik A. How to scaffold ExpressJS server and test it. Medium. URL: https://medium.com/craft-academy/how-to-scaffold-expressjs-server-and-test-it-d2a2ab1d30e0
5. INodeJS. Child process | Node.js v22.2.0 Documentation. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/api/child_process.html.

АНАЛІЗ ПАТЕРНІВ ПРОЄКТУВАННЯ У ВЕБРОЗРОБЦІ ТА ЇХ ЗАСТОСУВАННЯ У РОЗРОБЦІ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ СТВОРЕННЯ РОЗКЛАДУ НАВЧАЛЬНОГО ЗАКЛАДУ / ANALYSIS OF DESIGN PATTERNS IN WEB DEVELOPMENT AND THEIR APPLICATION IN THE DEVELOPMENT OF A WEB APPLICATION TO AUTOMATE THE CREATION OF AN EDUCATIONAL INSTITUTION'S TIMETABLE

Борозенний С.О., Мисько Ю.М. / Borozennyi S.O., Mysko Y.M.
Національний університет «Києво-Могилянська академія» / National University of Kyiv-Mohyla Academy
04070, м. Київ, вул. Г. Сковороди, 2, кафедра мультимедійних систем, тел.: 044 425-77-53
E-mail: borozenyi@ukma.edu.ua

This work analyzes design patterns that can be used to create a web application. It also examines how different patterns can solve specific technical problems that developers face in their work. The result is a web application for creating a school timetable.

Сьогодні веброзробка давно перестала бути лише засобом для створення статичних вебсторінок наповнених текстовою інформацією та зображеннями. Натомість, все частіше можна зустріти складні вебзастосунки, здатні забезпечити широкий спектр послуг, як-от електронна комерція, соціальні мережі, графічні редактори та багато інших. Такі застосунки часто містять складну бізнес-логіку та інтеграції з різними сервісами, а отже мають відповідати високим вимогам до продуктивності та масштабованості.

Якщо раніше було достатньо простих скриптів для додавання динаміки на вебсторінку, то тепер, для забезпечення швидкої взаємодії з користувачем потрібна добре продумана та

масштабована система. Тому патерни проєктування стали не просто корисними, а часто необхідними, оскільки вони допомагають розробникам створювати кращу архітектуру застосунків.

Застосування патернів проєктування у розробці вебдодатків може підвищити якість коду та зробити його зрозумілішим для інших розробників. Використання патернів надає можливості для покращення способів створення та проєктування вебзастосунків.

Термін “патерн” є досить абстрактним, тому і немає чіткого визначення для їх опису. Однак, найчастіше для опису патернів, використовують підхід запропонований авторами книги “Design Patterns: Elements of Reusable Object-Oriented Software”[2]. Цей підхід складається з таких пунктів:

- назва та класифікація;
- також відомий як: альтернативні назви цього патерну, якщо такі існують;
- мотивація: опис сценарію використання або проблеми, які вирішує патерн;
- де застосовується: опис ситуації, де можна застосувати патерн;
- структура: візуальне представлення патерну, що показує його основні компоненти та їх взаємозв'язки;
- учасники та взаємодії: опис класів та об'єктів, які потрібні для реалізації патерну;
- наслідки: результати та проблеми, які можуть виникнути в результаті використання патерну;
- реалізація: поради щодо реалізації патерну;
- приклад коду, які ілюструють, як патерн може бути реалізований;
- відомі випадки використання: реальні приклади використання патерну в відомих системах чи бібліотеках;

Класифікація патернів проєктування:

Автори книги “Design Patterns: Elements of Reusable Object-Oriented Software”[1], також відомі, як Банда чотирьох(GoF), пропонують наступну класифікацію патернів за метою їх використання:

- породжуючі патерни: ті, що відповідають за створення об'єктів. До таких відносяться: Singleton, Factory Method, Abstract Factory, Builder, і Prototype;
- структурні патерни: ті, що відповідають за способи побудови зв’язків між об’єктами. Прикладами структурних патернів є: Adapter, Composite, Proxy, Flyweight, Facade, Bridge, і Decorator;
- поведінкові патерни: ті, що відповідають за ефективну комунікацію між об'єктами. До таких відносяться: Observer, Strategy, Command, State, Visitor, Mediator, Memento, Iterator та інші;

“Банда чотирьох” також розділяє патерни в залежності від того, до чого вони застосовуються: до класів чи до об'єктів[2]. Проте, у світі веброзробки, де JavaScript (використовується на 99% вебсайтів для клієнтської частини) і PHP (77% вебсерверів працюють на PHP) є основними мовами програмування і підтримують кілька парадигм програмування, ця класифікація здається не релевантною. Адже патерни активно адаптуються і використовуються у мовах, які не є класичними мовами ООП і можуть зовсім не містити концепції класів.

Створення шкільного розкладу є складним процесом і вимагає врахування різних факторів як-от: доступності вчителів і аудиторій, потреб і побажань учнів, а також директив Міністерства освіти і науки України. Тож вебзастосунок, який автоматизує цей процесу може значно звільнити час освітян для інших важливих завдань.

Для створення застосунку, який автоматизує процес складання шкільного розкладу було розроблено алгоритм, який відповідає за генерацію розкладу. Цей алгоритм передбачає врахування багатьох різних факторів, зокрема:

- Шкала важкості предметів.
- Побажання вчителів.
- Обмеження за днями.
- Поділ на групи.
- Мінімізація вільних часових проміжків між уроками.

Алгоритм повинен уміти оптимізовувати розклад, щоб уникнути вільних періодів у графіках як учнів, так і вчителів.

Для створення цього клієнт-серверного застосунку з генерації шкільного розкладу було використано підхід товстого клієнта. Це рішення було обумовлено кількома причинами.

1. Перенесення бізнес-логіки на клієнтську сторону
2. Оптимізація продуктивності та реактивності застосунку

При реалызацыъ проекту було використані наступні патерни:

- Стратегія
- Ланцюжок обов'язків
-

Висновки

Патерни дійсно забезпечують значне покращення процесів розробки. Їх використання допомагає розробникам створювати компоненти, що взаємодіють між собою через інтерфейси (контракти), знижуючи тим самим прямі залежності. Це забезпечує більшу гнучкість у розширенні та модифікації системи. Також патерни такі як Спостерігач, ІЗ чи Декоратор сприяють слабкому зв'язуванню компонентів, що робить систему менш чутливою до змін у будь-якій окремій частині коду.

Проте це дослідження також показало, що патерни проєктування не є універсальними рішеннями.

Ключові слова: патерни проєктування, веброзробка, автоматизація.

Використані джерела:

[1] “Мова шаблонів” (“A Pattern Language”), Крістофер Александер, 1977
[2] “Design Patterns: Elements of Reusable Object-Oriented Software”, Еріх Гамма, Річардом Хелмом, Ральфом Джонсон, Джоном Вліссідесом, 1994
[3] “Занурення в патерни проєктування”, Олександр Швець, 2022
[4] “Чиста архітектура”, Роберт Мартін, 2021

ВПОРЯДКОВАНЕ ОБРОБЛЕННЯ ПОВІДОМЛЕНЬ У СИСТЕМАХ З РОЗПОДІЛЕНОЮ АРХІТЕКТУРОЮ / ORDERED MESSAGE PROCESSING IN SYSTEMS WITH A DISTRIBUTED ARCHITECTURE

Давиденко А.М./ Davydenko A.M.
Національний університет «Києво-Могилянська академія» /
National University of "Kyiv-Mohyla Academy" (NaUKMA)
04655, м. Київ, вулиця Григорія Сковороди, 2, НаУКМА, Факультет інформатики,
andrii.davydenko@ukma.edu.ua

This work analyzes challenges relevant for developers of distributed systems, in particular, with regard to ensuring the order of processing of events occurring in the system and its impact on the overall state of the system. It proposes a classification of distributed systems according to the desired characteristics in terms of maintaining and evaluating the current state. An example of using the tools of modern message brokers (such as RabbitMQ and Apache Kafka), which allow the processing of messages by only one consumer, is given.

Вступ

Розподілена система (РС) складається з комп'ютерів (вузлів), що взаємодіють через комунікаційну мережу. На відміну від централізованих систем, де оброблення даних здійснюється одним сервером, у РС завдання розподіляються між кількома вузлами, що забезпечує масштабованість і стійкість. Проте це створює виклики, зокрема паралелізм і складність у підтриманні впорядкованих комунікацій між вузлами. Питання впорядкування повідомлень (подій) залишається актуальним і широко досліджуваним.

У цій роботі розглянуто сучасні виклики в розробці РС, проаналізовано типові сценарії, де необхідне впорядкування повідомлень, та запропоновано класифікацію РС відповідно до вимог щодо збереження стану й обчислень.

Виклики у розробленні розподілених систем

У своїй праці Лампорт запропонував часову логіку відношення "відбулося раніше", яка є основою для концепції часткового впорядкування, або логічного годинника [2]. Цей підхід став базовим для опису алгоритмів, що визначають порядок подій у розподілених системах, і розв'язання проблем синхронізації. Важливим нововведенням стала формалізація поняття причинності в розподілених системах, де впорядкування подій забезпечується за допомогою