УДК 004.031.2

С.Г. Московко

НАДАННЯ ДОСТУПУ ДО ПРЕДМЕТНО-ОРІЄНТОВАНИХ СИСТЕМ

Вінницький національний технічний університет

Анотація

У цій роботі проводиться детальний аналіз підходів до надання доступу до предметно-орієнтованих систем. Розглядаються такі підходи: адаптер, фасад, шина повідомлень. Для кожного підходу аналізуються його переваги та недоліки, а також наведені приклади реалізації.

Ключові слова: предметно-орієнтоване проектування, зовнішній доступ, адаптер, фасад, шина повідомлень.

Abstract

This paper analyzes approaches for providing access to domain-driven design systems in detail. It considers the following approaches: adapter, facade and message bus implementation. For each approach, the paper analyzes advantages, disadvantages, and examples of implementation..

Keywords: domain-driven design, external access, adapter, facade, message bus.

Вступ

Предметно-орієнтований дизайн (Domain-Driven Design, DDD) — це підхід до розробки програмного забезпечення, який фокусується на бізнес-домені системи. DDD рекомендує розбивати систему на домен, який відповідає за бізнес-логіку, і інфраструктуру, яка відповідає за реалізацію цієї логіки. Одним з важливих аспектів реалізації DDD-систем є надання доступу до них ззовні. Це може бути необхідно для взаємодії з іншими системами, для використання системи через веб-інтерфейс або для автоматизації завдань. Реалізація зовнішнього доступу до DDD-системи має враховувати такі фактори, як безпека, версійність і моніторинг. У цій роботі буде проведено аналіз підходів до надання доступу до DDD-систем. Буде розглянуто такі підходи: адаптер, фасад, шина повідомлень. Для кожного підходу будуть розглянуті його переваги та недоліки.

Метою роботи аналіз підходів до надання доступу до DDD-систем з метою підвищення ефективності алгоритмічних та програмних засобів, що проектуються. Робота спрямована на детальний аналіз підходів по параметрам: адаптер, фасад, шина повідомлень.

Практична цінність цієї роботи полягає в тому, що вона допомагає розробникам DDD-систем вибрати правильний підхід до надання доступу до них ззовні. Це дозволяє забезпечити безпеку, версіонність і моніторинг системи, а також зробити її більш гнучкою та масштабованою.

Актуальність цієї роботи обумовлена поширенням DDD-систем. DDD-системи є популярним вибором для розробки програмного забезпечення, яке потребує підтримки бізнес-домену. Надання доступу до таких систем ззовні є важливим аспектом їх реалізації.

Результати роботи

Забезпечення доступу до предметно-орієнтованих систем для зовнішніх користувачів і компонентів є критично важливим завданням. Для досягнення цієї мети, існують різні підходи та патерни, які можна використовувати. У цій роботі ми розглянемо чотири з них: фасад, адаптер, шина повідомлень. Кожен з цих підходів має свої переваги та недоліки і може бути вибраний в залежності від конкретних вимог і контексту проекту.

Паттерн "Фасад" (Façade) - це структурний паттерн проектування, який надає простий і однорідний інтерфейс для взаємодії з складнішою системою, скриваючи деталі роботи цієї системи від клієнта. Основною метою паттерну "Фасад" є спрощення взаємодії із системою, зниження складності та підвищення рівня абстракції для клієнта. У предметно-орієнтованих системах паттерн "Фасад" може бути реалізований у вигляді спеціалізованого сервісу або аплікаційного рівня. Його завдання - надати зовнішньому світу простий та зрозумілий інтерфейс для взаємодії з комплексною доменною моделлю. Наприклад, в системі для управління банківськими рахунками, "Фасад" може надавати методи для створення рахунку, здійснення переказів та перевірки балансу, приховуючи всі деталі доменної логіки, такі як обробка транзакцій, перевірки безпеки та інше.

Переваги паттерну "Фасад":

- Зниження складності: Даний паттерн спрощує взаємодію з доменною моделлю, що робить систему більш зрозумілою і підтримуваною;
- Забезпечення однорідного інтерфейсу: Фасад надає єдиний інтерфейс для всіх операцій, що спрощує роботу з системою;
- Забезпечення роботи з доменною моделлю: Внутрішні деталі доменної моделі залишаються прихованими, забезпечуючи їхню незалежність та безпеку.

Недоліки паттерну "Фасад":

- Додатковий шар абстракції: Додавання фасаду може призвести до збільшення складності системи через наявність ще одного шару абстракції;
- Потреба в оновленнях: Зміни в доменній моделі можуть вимагати оновлення фасаду, що може бути трудомістким завданням.

У контексті версійності, паттерн "Фасад" може зменшити вплив змін в доменній моделі на зовнішні інтерфейси, але в той же час може затримати впровадження нових можливостей, оскільки фасад повинен буде адаптуватися до змін. З точки зору безпеки, паттерн "Фасад" може допомагати в централізованому контролі доступу до функціональності системи, але важливо дбати про безпеку самого фасаду, оскільки він є точкою входу. При моніторингу системи, фасад може служити як точка для вимірювання та моніторингу використання певних функціональностей, що допомагає у виявленні проблем та оптимізації системи.

Паттерн адаптер (Adapter) - це структурний паттерн проектування, який дозволяє об'єднувати інтерфейси двох несумісних систем. Він дозволяє класам працювати разом, не змінюючи їхніх кодових осередків. У DDD системах адаптер може використовуватися для інтеграції інших систем або сервісів в додаток. Наприклад, у великому корпоративному додатку може бути потреба взаємодіяти з зовнішніми системами, такими як платіжні шлюзи або соціальні мережі. Для цього можна створити адаптер, який надає зручний інтерфейс для внутрішньої системи додатку і перетворює запити та дані так, щоб вони відповідали інтерфейсу зовнішньої системи.

Переваги паттерну адаптер:

- Розширюваність: Додавання нових інтеграцій або зміна інтерфейсів зовнішніх систем не впливає на внутрішній код додатку;
- Підтримка існуючих систем: Дозволяє використовувати старі або існуючі системи без необхідності їх повної модифікації;
- Спрощення тестування: Дозволяє створювати моки (заглушки) для тестування без залучення зовнішніх систем.

Недоліки паттерну адаптер:

- Додаткова складність: Адаптери можуть додавати додатковий рівень абстракції, що впливає на виконавчу швидкість та ресурси системи.
- Потреба у підтримці: Адаптери потребують підтримки та обслуговування, особливо у випадку змін у зовнішніх системах.

Адаптери можуть впливати на безпеку системи, оскільки надають доступ до зовнішніх ресурсів. Необхідно гарантувати, що адаптери мають обмежений доступ та використовують механізми аутентифікації та авторизації. Адаптери потребують моніторингу для відстеження їхньої продуктивності

та правильності інтеграції з зовнішніми системами. Важливо мати механізми ведення журналів та моніторингу помилок для швидкого виявлення проблем. Також у контексті версійності проблеми можуть виникнути при оновленні адаптера під нову версію зовнішньої системи, оскільки інтерфейси можуть змінюватись.

Паттерн шина повідомлень (Message Bus) - це архітектурним паттерном, який використовується для обміну повідомленнями між різними компонентами або службами в системі. В цьому паттерні існує централізований сервіс або складова, яка допомагає іншим частинам системи комунікувати між собою, не прямо взаємодіючи один з одним.

Переваги паттерну шина повідомлень:

- Розділення компонентів: Дозволяє розділити доменні логіку від інфраструктурних аспектів, що сприяє підтримці чистоти DDD.
- Асинхронність: Дозволяє обробляти події асинхронно, що підвищує відзивчивість системи та масштабованість.

Недоліки паттерну шина повідомлень:

- Версіонність: При зміні схеми повідомлень або обробників можуть виникати проблеми з обратною сумісністю та версіонністю.
- Безпека: Неправильна конфігурація шини повідомлень може призвести до можливих проблем з безпекою, таких як витік інформації або атаки на шину.
- Моніторинг: Складно контролювати та моніторити потік повідомлень та обробників, що може ускладнити відладку та аналіз проблем.

Висновки

У цій роботі було розглянуто ключовий аспект предметно-орієнтованого дизайну (DDD) - надання доступу до системи ззовні. Ми дослідили чотири різні підходи: фасад, адаптер та шина повідомлень кожен із яких має свої переваги та обмеження. Ці підходи можуть бути використані в залежності від конкретних вимог та контексту проекту.

Проте важливо зауважити, що в багатьох ситуаціях комбінування різних підходів може бути найкращим рішенням для надання доступу до предметно-орієнтованої системи. Наприклад, можна використовувати фасад для надання спрощеного інтерфейсу для клієнтів, адаптери для інтеграції з іншими системами та шину повідомлень для асинхронної комунікації між компонентами системи. Ця комбінація може дозволити досягти більшої гнучкості, безпеки та продуктивності в системі.

Усі розглянуті підходи вимагають ретельного аналізу і проектування для визначення, який з них найкраще підходить для конкретного завдання. На вибір підходу також можуть впливати фактори, такі як розмір системи, потреби користувачів, технічні можливості та бюджетні обмеження. В кінцевому підсумку, вибір підходу до надання доступу до предметно-орієнтованої системи важливий етап проектування, який впливає на загальну якість та продуктивність системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Vernon V. Implementing Domain-Driven Design : Addison-Wesley Professional, 2013. 656 с. ISBN 978-0321834577.
2. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software, 2003. 320 с. ISBN 9780321125217.
3. Facade. – URL: https://refactoring.guru/design-patterns/facade
4. Adapter. – URL: https://refactoring.guru/design-patterns/adapter
5. Event-based Microservices: Message Bus. – URL: https://medium.com/usertesting-engineering/event-based-microservices-message-bus-5b4157d5a35d

Московко Сергій Геннадійович — факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: smoskovko@icloud.com.

Moskovko Serhii G. — Department of intelligent information technologies and automation, Vinnytsia National Technical University, Vinnytsia, e-mail: smoskovko@icloud.com.