

ПРОЕКТУВАННЯ АРХІТЕКТУРИ УПРАВЛІННЯ ДЖЕРЕЛАМИ ДАНИХ В ГІБРИДНОМУ ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ

Олійник Р.В., к.т.н., доц.

Львівський торговельно-економічний університет

Україна, Львів

oliynykroma@gmail.com

Анотація. Розглянуто задачу проектування архітектури гібридного програмного забезпечення з метою створення гнучкого середовища доступу до даних з використанням сучасних шаблонів програмування. Розроблено структуру ядра гібридного програмного забезпечення з застосуванням структурних та утворюючих шаблонів проектування, що дозволило спростити доступ до локальних або мережесевих постачальників даних відповідно до вимог бізнес-логіки.

Ключові слова: гібридне програмне забезпечення; шаблон проектування; архітектура програмного засобу.

Вступна частина. Сьогодні все більше користувачів використовують гібридні програмні продукти у повсякденному використанні. У свою чергу, гібридними програмними засобами вважаються програмні засоби, які в процесі створення використовують прикладні програмні інтерфейси для доступу до платформенно-залежних бібліотек та компілюються у спеціалізоване для певної платформи програмне рішення в автоматичному режимі з використанням, наприклад, таких фреймворків як Ionic + Capacitor + Angular. Джерелами даних для таких рішень є віддалений доступ до сервера з використанням REST, GraphQL, або платформенно-залежні локальні сховища даних, такі як, SQLite, Indexed DB тощо.

Дослідження, що проводяться в галузі проектування архітектури програмних комплексів, в більшості сфокусовані на проектуванні структури локально- або мережево-орієнтованого програмного забезпечення, зокрема, досліджується доцільність використання мікро-сервісної, сервіс-орієнтованої або монолітної архітектури [1]. Також зустрічаються дослідження структурних і архітектурних шаблонів проектування. Наприклад, Альшаммарі Р. [2] зауважив, що використання добре документованих та загальноприйнятих шаблонів позитивно впливає на якість коду. Проте, в дослідженні Редді С. наголошується, що, в окремих випадках, змішування загальноприйнятих шаблонів може ускладнювати бізнес-логіку. Варто зауважити, що сьогодні бізнес-аналітики все активніше приймають участь у проектуванні архітектури програмного продукту, що, у свою чергу, дає змогу якісно поєднувати архітектуру та шаблони проектування в єдине ціле з метою створення добре документованого масштабованого продукту [5], базуючись на загально прийнятих підходах проектування. Проте, як показав проведений аналіз джерел, більшість напрямків досліджень сфокусовано на розробці або локально орієнтованих програмних комплексів, або мережесевих продуктів без врахування переваг використання гібридних систем. Тому задача проектування архітектури управління потоками даних у гібридних програмних засобах з використанням сучасних шаблонів проектування з метою забезпечення гнучкого доступу до джерел даних є своєчасною та актуальною.

Основна частина. Будь-який програмний продукт, зокрема, гібридне програмне забезпечення, щоб вважатись якісно спроектованим відповідно до вимог бізнес-логіки, повинно забезпечувати сумісність, масштабованість та відповідати вимогам тестування [1, 5].

Гібридне програмне забезпечення, в залежності від завдань, може містити в собі як реалізацію серверної частини для роботи з даними на стороні клієнта, так і комунікацію з веб сервером з використанням REST, тому доцільно при проектуванні архітектури такого програмного засобу зосередитись на проектуванні структури, у якій реалізація логіки створення класів, які управляють потоками даних, такими як операції CRUD, обробкою помилок, тощо, буде прихованою, а сам клас буде створено відповідно до вимог бізнес-логіки на вимогу, отже комунікація повинна проводитись за допомогою методів, які описані в Data Access Object (DAO) інтерфейсі.

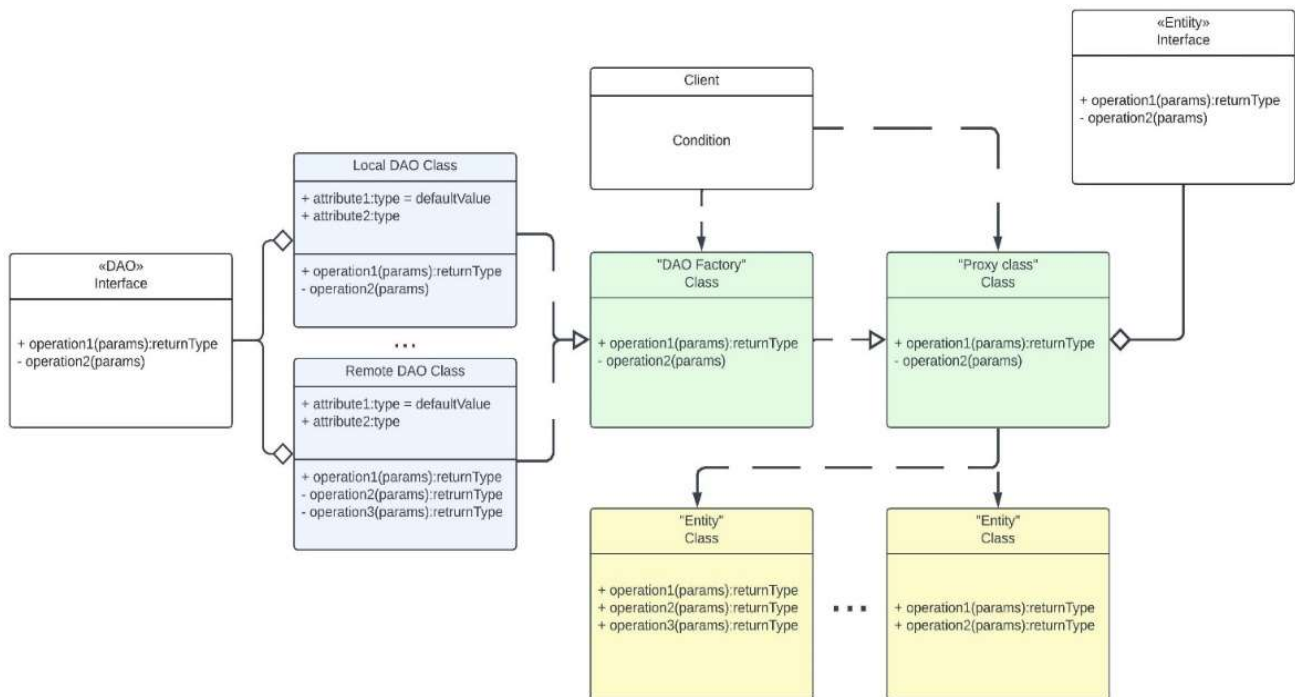


рис. 1. Діаграма взаємозв'язків між компонентами гібридної архітектури

Як відомо, задача створення об'єктів на вимогу належать до шаблонів створення (Creational Patterns) [3]. Такі шаблони дозволяють абстрагувати створення екземплярів класів, приховуючи складні логічні процеси створення за простим інтерфейсом доступу до даних. Цей підхід можна вважати особливо корисним у гібридних системах, де об'єкти або класи можуть мати різну реалізацію залежно від платформи, конфігурації, offline-, online- first підходів або інших факторів [3]. В більш загальному випадку зміна бізнес логіки з offline first до Online зводиться до використання одних і тих самих методів взаємодії контролера і моделі даних. Тому для забезпечення проектованої гнучкості зміни логіки роботи з даними доцільно використати шаблон проектування фабрика (Factory) [4]. Цей шаблон за допомогою правильно спроектованого інтерфейсу доступу до даних Data Access Object Interface (DAO Interface), задовольнятиме потреби доступу як offline first, так і Online реалізації гібридної програми [4]. У свою чергу, класи повинні імплементувати цей інтерфейс для коректної роботи з локальними (LocalDAO) так і віддаленими даними (RemoteDao) (рис .1). Фабрика класів генеруватиме той, чи іншого клас відповідно до вимог користувача або бізнес-логіки в будь-який момент життєвого циклу або на етапі ініціалізації програмного продукту.

Класи, які реалізують DAO Interface, відповідають лише за реалізацію набору методів описаних в інтерфейсі, тому потрібна додаткова маніпуляція з даними. Проте проектована

структура у чистому вигляді є такою, що обмежує гнучкість архітектури через інтерфейс, який регламентує певний набір методів. Подальше дослідження показало, що для усунення цього недоліку можна шаблон Factory спробувати замінити на шаблон Builder, який конструюватиме клас «на вимогу» з використанням додаткових параметрів, але реалізація такого підходу у системі, що використовує різні середовища для роботи з даними, може бути ускладненою, адже проектована архітектура повинна являти собою «генератор» типових класів, що реалізують лише DAO Interface з різною реалізацією методів, але самі виклики мають мати можливість бути перехопленими для подальших маніпуляцій. Таким чином, доцільно спроектувати роботу з DAO шляхом введення в розроблювану архітектуру певного класу заступника (Proxy), що контролюватиме доступ до DAO об'єкта. Введення такого класу дозволяє вести роботу з класами, що реалізують DAO Interface через додатковий проміжний шар, а саме Proxy клас, що також реалізовуватиме DAO Interface, але, окрім того, дозволить маніпуляції перед викликами сконструюваних фабрикою методів. (рис. 1). Proxy може імплементувати певний Entity інтерфейс, який реалізовуватиме специфічну поведінку, яка, в свою чергу, делегуватиме завдання або перехоплюватиме всі виклики до оригінального об'єкта, дозволяючи виконувати додаткові маніпуляції з даними, а оригінальні класи залишатимуться чистими і міститимуть лише реалізацію описаних в DAO інтерфейсі методів.

Висновки. Таким чином, поєднання створюючих та структурних шаблонів проектування при проектуванні архітектури управління даних у гібридному програмному забезпеченні здатне якісно вплинути як на масштабованість, підтримку продукту, так і зменшити час відлагодження програмного продукту в цілому, оскільки проектована структура міститиме чіткі зони відповідальності при роботі з даними. Окрім того, поєднання шаблонів проектування Factory та Proxy дозволяє уникнути використання більш складної організації програмного коду, що, у свою чергу, дозволяє вести розробку сучасних програмних систем, які можуть реалізовувати в собі різну логіку поведінки в залежності від платформи чи офлайн або онлайн сховищ даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Морозов, А., Вакалюк, Т., Кубрак, Ю., Зосімович, Д. (2022). Аналіз факторів впливу на архітектури програмних систем. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 1, 44–52. <https://doi.org/10.32782/IT/2022-1-76>
- [2] Альшаммарі, Р., Хаммад, М. (2019). Ефективність шаблонів проектування в об'єктно-орієнтованій програмній інженерії: систематичний огляд літератури. *Журнал систем і програмного забезпечення*, 151.
- [3] Соні, Д., Нахар, В. (2015). Оглядова стаття про шаблони проектування. *Міжнародний журнал інженерних досліджень і технологій*, 4(02), 935-938.
- [4] Factory Method. (n.d.). <https://refactoring.guru/design-patterns/factory-method>.
- [5] Гобов, Д. А., Шевченко, Н. Ю. (2024). Визначення архітектури вимог до ІТ-рішення як бізнес-аналітичного продукту. *Сучасний стан наукових досліджень та технологій в промисловості*, 1(27), 26-38. <https://doi.org/10.30837/ITSSI.2024.27.026>.

Oliinyk R.V.

Designing Architecture for Data Source Management in Hybrid Software

Abstract. The task of designing the architecture of hybrid software has been considered with the aim of creating a flexible data access environment using modern programming patterns. The structure of the core of the hybrid software has been developed using structural and creational design patterns, which has simplified access to local or network data providers in accordance with the requirements of business logic.

Keywords: Hybrid software; design pattern; software architecture; interface