DOI: https://doi.org/10.36910/6775-2524-0560-2023-51-14
УДК 004.42
Христинець Наталія Анатоліївна, к.т.н., доцент
https://orcid.org/0000-0002-4836-7632
Лавренчук Світлана Василівна, к.т.н., доцент
https://orcid.org/0000-0002-5453-3924
Пех Петро Антонович, к.т.н., доцент
https://orcid.org/0000-0002-6327-3319
Євсюк Микола Миколайович, к.т.н., доцент
https://orcid.org/0000-0002-3768-8959
Євсюк Володимир Миколайович, бакалавр
Крулік Юрій Олександрович, бакалавр
Луцький національний технічний університет, м. Луцьк, Україна

ФУНКЦІОНАЛЬНІ АДАПТИВНІ ІНЕРФЕЙСИ З ДИНАМІЧНИМИ КОМПОНЕНТАМИ ДЛЯ ПІДСИСТЕМ ЗБЕРІГАННЯ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

Христинець Н.А., Лавренчук С.В., Пех П.А., Євсюк М.М., Євсюк В.М., Крулік Ю.О. Функціональні адаптивні інтерфейси з динамічними компонентами для підсистем зберігання мультимедійного контенту. В статті розглянуто методи і способи побудови інтерфейсів прикладних програм. Досліджено питання багатопоточності і асинхронного програмування, що включає створення та керування потоками для ефективного обробки та аналізу даних. Такий спосіб керування потоками дозволяє контролювати кількість одночасних потоків, які можуть отримати доступ до обмеженого ресурсу даних та забезпечує ефективність цього доступу. Досліджено класичні атрибути валідації, такі як Required, Range, RegularExpression для перевірки вхідних параметрів, моделей або запитів. Створено власний клас, який реалізує інтерфейс IValidator з пакету FluentValidation та розроблено програму для організації валідації у інтерфейсах прикладних програм.
Ключові слова: API, валідація, багатопоточність, атрибути валідації, мультимедійний контент

Khrystynets N., Lavrenchuk S., Pekh P., Yevsyuk M., Yevsyuk V., Krulik Yu. Functional adaptive interfaces with dynamic components for multimedia content storage subsystems. The article discusses methods and ways of building application program interfaces. The issue of multithreading and asynchronous programming, which includes the creation and management of threads for efficient data processing and analysis, is investigated. This method of thread management allows you to control the number of simultaneous threads that can access a limited data resource and ensures the efficiency of this access. Explored classic validation attributes like Required, Range, RegularExpression to validate input parameters, models or queries. A custom class was created that implements the IValidator interface from the FluentValidation package, and a program was developed to organize validation in application program interfaces.
Keywords: API, validation, multithreading, validation attributes, multimedia content

Постановка задачі. У сучасному цифровому світі розробка програмних додатків інтегрується з великою кількістю зовнішніх сервісів, платформ та систем. Інтерфейси прикладних програм (API) виступають, як міст між різними компонентами програмного забезпечення []. Вони використовуються у багатьох сферах індустрії. У веб-розробках інтерфейси прикладних програм є основним способом взаємодії між різними компонентами програмного забезпечення і можуть надавати доступ до функцій, баз даних, зовнішніх сервісів та іншого функціоналу. В системах електронної комерції інтерфейси прикладних програм дозволяють цим системам взаємодіяти між собою, обмінюватися даними та виконувати спільні операції. Це полегшує автоматизацію бізнес-процесів та сприяє ефективному обміну даними між різними системами. У платіжних сервісах - таких, як наприклад, PayPal – пропонується API для прийому платежів та управління фінансовими операціями, для подальшої інтеграції платіжних функцій у додатки та електронні магазини. Популярні платформи соціальних медіа, такі як Facebook, Instagram та інші, також надають API для доступу до своїх послуг. До таких послуг відносяться публікації мультимедійного контенту, отримання даних про користувачів цих мереж, взаємодії з коментарями та багатьох інших функцій.

Метою дослідження є визначення специфікацій створення програмних інтерфейсів для мультимедійного контенту, валідація даних у програмних інтерфейсах та дослідження багатопоточності прикладних програм.

Новизна дослідження полягає у розробці індивідуальних валідаторів та розробці власних алгоритмів поведінки процесів і потоків в проектуванні інтерфейсів прикладних прогам.

Основна частина. використовуються для забезпечення взаємодії між компонентами програмного забезпечення, сприяють розширенню функціональності та спрощують інтеграцію між

різними системами (рис. 1). Тому, питання розробки продуктивних API за допомогою сучасних засобів: фреймворків, мов програмування є актуальною задачею.

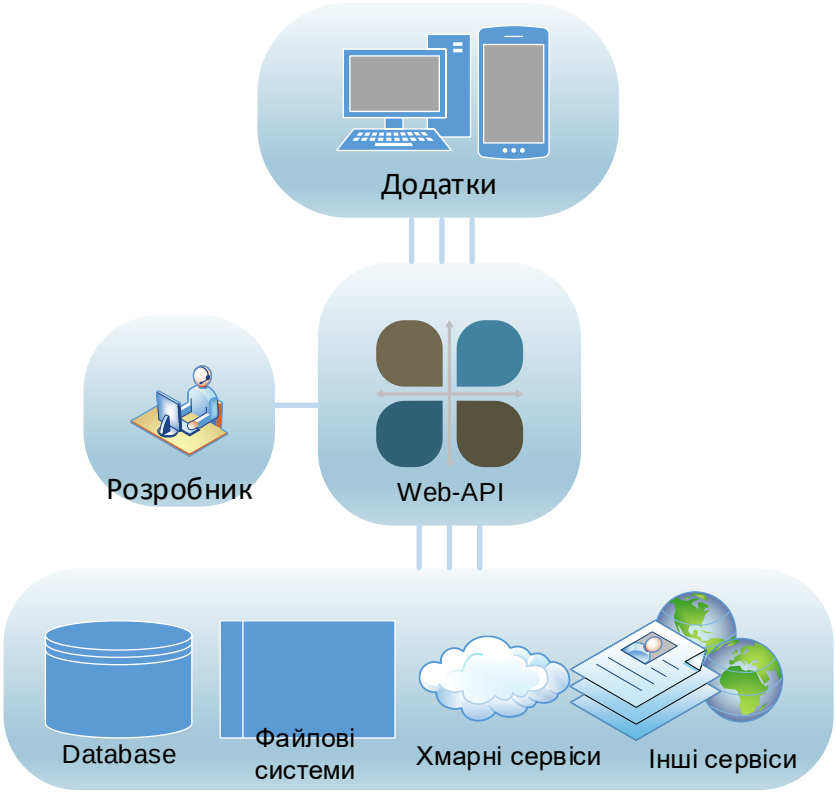

Рис. 1. Взаємодія елементів системи API-інтерфейсів

Створення мультимедійного контенту в веб-інтерфейсах включає в себе використання різноманітних технологій та інструментів для додавання графіки, звуку, відео та іншого мультимедійного вмісту на веб-сторінки. Основні елементи, які ви можете використовувати для створення мультимедійного контенту, включають зображення, аудіо- та відео-елементи, анімацію та відео. Важливим моментом організації ефективного API є багатопоточність, тобто здатність програм обробляти багато потоків одночасно при роботі з мультимедійним контентом [1].

Використання багатопоточності дозволяє програмі виконувати багато завдань паралельно, що може покращити продуктивність і швидкість відгуку програми на користувацькі дії. В роботі будемо акцентувати увагу на елементах розробки веб-API у фреймворку ASP.NET Web API на платформі .NET. Використаємо засоби мови програмування C#, яка надає різні механізми та бібліотеки для роботи з потоками.

Загалом, створення інтерфейсів забезпечується певним алгоритмом. Спочатку іде розробка користувацького інтерфейсу (UI) програми. Вона охоплює створення графічного вигляду програми, розташування елементів керування на формі, обробку подій і взаємодію з користувачем. Для програмування інтерфейсів можуть використовуватись різні інструменти та технології, такі як бібліотеки графічного інтерфейсу користувача (GUI), мови програмування для розробки десктопних або веб-додатків, фреймворки. Важливим моментом організації ефективного API є багатопоточність, тобто здатність програм обробляти багато потоків одночасно. Потік є окремим шляхом виконання програмного коду.

Використання багатопоточності дозволяє програмі виконувати багато завдань паралельно, що може покращити продуктивність і швидкість відгуку програми на користувацькі дії. В роботі будемо акцентувати увагу на елементах розробки веб-API у фреймворку ASP.NET Web API на платформі .NET. Використаємо засоби мови програмування C#, яка надає різні механізми та бібліотеки для роботи з потоками. Ці прості кроки можуть служити вихідною точкою для створення мультимедійного контенту в веб-інтерфейсах.

Специфікації API. Визначення специфікацій та етапи створення веб-API можуть розрізнятися залежно від конкретного проекту та підходу розробки [2]. Однак, є загальні етапи створення веб-API, серед яких: розробка механізмів маршрутизації для визначення шляхів до API та методи обробки запитів до додатку; процес перетворення об'єктів або структур даних в формат, який може бути збережений та переданий через мережу та інші. При проектуванні веб-API важливо враховувати особливості обробки потоків для забезпечення ефективності, масштабованості та безпеки.

На відміну від десктопних API, веб-API мають кілька відмінних рис, оскільки вони спрямовані на різні платформи та середовища. Застосування протоколів HTTP, REST у веб-API використовуються для комунікації між веб-сервером та клієнтськими додатками, а протоколи десктопних API – це, як правило, TCP/IP або named pipes. Особливістю веб-API є і середовище виконання: зазвичай вони виконуються на веб-сервері або в хмарному середовищі і обслуговують багато клієнтів одночасно, а десктопні API виконуються безпосередньо на комп'ютері користувача, часто навіть відокремлено від сервера. Проте, найважливішою особливістю, на нашу думку, є те, що веб-API потребують ретельного розгляду з питань безпеки, оскільки вони працюють в мережі Інтернет і піддаються різним загрозам. Десктопні API також потребують заходів безпеки, але вони працюють у більш контрольованому середовищі, що зменшує загрози безпеки таких API.

В даній роботі детально акцентована увага на ті особливості обробки потоків, які стосуються надійності, безпеки та ефективності веб-API: валідацію даних та організацію різних механізмів багатопоточності.

Визначення вимог до інтерфейсу. Першим етапом створення інтерфейсів прикладних програм є формування ідеї щодо розробки функціоналу і архітектури додатку, а також підбір фреймворку для подальшої реалізації [3]. На цьому етапі важливо сформувати уявлення, які дані потрібно буде передавати та отримувати через розроблений інтерфейс. Вхідні дані, які надходять до веб-додатку, повинні бути перевірені на коректність та безпеку перед обробкою.

Важливо врахувати на цьому етапі, які будуть права доступу до API, а також розробити схеми маршрутів, структури даних та об'єктів, які будуть передаватися через API. Це дозволить уникнути вразливостей, таких як SQL-ін'єкції, XSS-атаки та інші види зловмисного впливу на систему.

Правила вдалого керування процесами і потоками. Погане проектування поведінки процесів і потоків може статися у різних ситуаціях. Особливо часто воно виникає в таких випадках [4]:

1) Якщо розробник не має достатнього досвіду або розуміння багатопотоковості. У такому разі є висока імовірність неправильно оцінити взаємодію між потоками і зробити помилки у проектуванні.

2) Використання спільного доступу до ресурсів. Коли кілька потоків мають доступ до спільних ресурсів, таких як змінні, об'єкти або файли, неправильне керування доступом може призвести до ситуацій гонок, коли потоки конфліктують між собою і працюють з непередбачуваними даними.

Нижче поданий фрагмент коду засобами ASP.NET мовою C#, у якому при записі в файл, відсутнє належне керування доступом до файлової системи:

```
public FileManager(string filePath)
{
_filePath = filePath;
}

public void WriteToFile(string content)
{
var fileStream = File.OpenWrite(_filePath);
var writer = new StreamWriter(fileStream);
writer.Write(content);
writer.Close();
fileStream.Close();
}
}
```

Код відкриває файл для запису, але не використовує механізми синхронізації, щоб уникнути одночасного доступу до файлу з багатьох потоків або процесів. Це може призвести до ситуації, коли кілька потоків або процесів намагаються записати в файл одночасно, що може призвести до конфліктів і некоректного запису даних.

Логіка коду web-API. Усі логічні структурні елементи коду [5], їх призначення та приклади реалізації для створення веб-API подані в таблиці 1:

Таблиця 1 – Систематизація структурних елементів веб-API

Структурний елемент	**Призначення**	**Способи реалізації (приклади)**
Routing	Визначення URL та відповідних методів обробки запитів	За допомогою атрибутів маршрутизації [HttpGet], [HttpPost], [HttpPut], [HttpDelete], методів MapRoute або MapGet, MapPost та ін.
Controllers	Контролери приймають запити, обробляють їх і повертають відповіді.	Можна створювати шляхом створення класу, який наслідує від класу Controller або ApiController
Models	Класи, які представляють дані, що передаються в запиті або повертаються відповіддю.	Моделі визначають структуру і валідацію даних, а також можуть містити логіку для операцій з даними. (Наприклад, у веб-додатку можна створити клас Product, який представляє модель продукту з властивостями Id, Name і Price)
Services	Класи, які містять бізнес-логіку додатку.	Наприклад, можна оголосити інтерфейс IProductService, який визначає методи, необхідні для роботи з продуктами, а потім реалізувати цей інтерфейс в класі ProductService, який отримує залежність IRepository<Product> через конструктор.
Filters	Класи, які застосовуються, щоб виконувати певні перевірки або операції до або після обробки запиту.	Зазвичай використовують такі типи фільтрів: Authorization Filters, Action Filters, Result Filters, Exception Filters
Результати (Results):	Класи, які визначають типи відповідей, що повертаються від методів контролерів.	Результатом обробки може бути JSON-об'єкт, XML-документ, HTML-сторінку або статус коду. Типи результатів: ViewResult, JsonResult, FileResult, RedirectResult, ContentResult та ін.
Configuration	Налаштування та параметризація веб-API.	Файл конфігурації окремо від коду. Наприклад, можна використати файл appsettings.json, який зберігає ключ-значення пари параметрів конфігураці

Подана у таблиці логічна структура веб-API допомагає організувати код будь-якого веб-додатку і визначити, як взаємодіють його різні компоненти. Вона допомагає зрозуміти, як дані перетікають через різні шари архітектури додатку і як вони повинні опрацьовуватись в процесі обробки HTTP-запитів.

Валідація. У ASP.NET Web API можна використовувати класичні атрибути валідації, такі як Required, Range, RegularExpression та інші, для перевірки вхідних параметрів, моделей або запитів на валідність.

Також, можна створити власні правила валідації, які відповідають специфічним потребам додатку. Зокрема, було створено власний клас, який реалізує інтерфейс IValidator (або успадкується від класу AbstractValidator) з пакету FluentValidation. Для валідації об'єктів типу XProject створено валідатор XProjectValidator, який перевіряє, чи поле Name не є пустим і має довжину від 1 до 50 символів, а поле Email є обов'язковим та має коректний формат електронної пошти.

```
public class XProjectValidator : AbstractValidator<XProject>
{
   public XProjectValidator()
   {
      RuleFor(x => x.Name)
         .NotEmpty().WithMessage("Name is required")
         .Length(1, 50).WithMessage("Name must be between 1 and 50 characters");

      RuleFor(x => x.Email)
         .NotEmpty().WithMessage("Email is required")
         .EmailAddress().WithMessage("Email is not valid");
   }
}
```

Розробка індивідуальних валідаторів має ряд переваг. Перше – це гнучкість в плані власних правил валідації, які відповідають специфічним потребам додатку. Друге - це персоналізація повідомлень: власною валідацією можна визначити самостійно прописані повідомлення про помилки, які повертаються клієнту. Ну і найголовніше – контроль над валідацією.

Висновки. ASP.NET надає різні механізми для реалізації багатопоточності, які дозволяють ефективно обробляти багато запитів одночасно. Ці механізми допомагають забезпечити багатопоточну обробку запитів і підвищити продуктивність веб-API, дозволяючи одночасно обробляти багато запитів без блокування основного потоку виконання. Досліджено, що розробка власної валідації може бути вигідною для багатьох веб-додатків, особливо якщо вони мають специфічні вимоги, або потребують гнучкості та контролю над процесом валідації.

Список бібліографічного опису

1. Кривонос, О. М., Кривонос, М. О. (2021). Приклад реалізації задачі з використанням багатопоточності на мові С. Прикладні системи та технології в інформаційному суспільстві, 131-134.
2. Кравченко, С. В. (2023). Аналіз сучасного фреймворка ASP. NET CORE для WEB-додатків..
3. Лавренюк, А. М., Куссуль, Н. М., Шелестов, А. Ю., & Скакун, С. В. (2019). Інформаційні технології (частина І).
4. Маліновський, В. І., Куперштейн, Л. М., & Каплун, В. А. (2022). Аналіз основних інформаційних загроз і впливів у сучасних мікроконтролерних системах (аналітичний огляд). Оптико-електронні інформаційно-енергетичні технології, 44(2), 100-113.
5. Романець, А., & Козбур, Г. В. (2022). Безпека соцмережі під час аутентифікації користувача. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології "Тернопільського національного технічного університету імені Івана Пулюя, 45-45.

References

1. Kryvonos, O. M., Kryvonos, M. O. (2021). An example of problem implementation using multithreading in the C language. Applied systems and technologies in the information society, 131-134.
2. Kravchenko, S. V. (2023). Analysis of the modern ASP framework. NET CORE for web-Applications.
3. Lavrenyuk, A. M., Kussul, N. M., Shelestov, A. Yu., & Skakun, S. V. (2019). Information technologies (part I).
4. Malinovskyi, V. I., Kuperstein, L. M., & Kaplun, V. A. (2022). Analysis of the main information threats and influences in modern microcontroller systems (analytical review). Optical-electronic information and energy technologies, 44(2), 100-113.
5. Romanets, A., & Kozbur, G. V. (2022). Social network security during user authentication. Materials of the X scientific and technical conference "Information models, systems and technologies" of Ternopil National Technical University named after Ivan Pulyu, 45-45.