УДК 004.42:004.415.2

А. С. Галіброда
В. В. Кабачій

СТВОРЕННЯ REST API ДЛЯ ТАСК-МЕНЕДЖЕРА: ДОСЛІДЖЕННЯ АРХІТЕКТУРНИХ РІШЕНЬ

Вінницький національний технічний університет

Анотація

У дослідженні запропоновано підхід до проєктування та реалізації REST API для веб-застосунку таск-менеджера – інструменту, який дозволяє ефективно керувати особистими й командними завданнями. Розробка таких систем потребує використання сучасних архітектурних підходів, що забезпечують масштабованість, безпечність і зручність у подальшому супроводі. У межах дослідження проаналізовано основні принципи архітектури REST, серед яких: визначення ресурсів, використання стандартних HTTP-методів та структурованість даних у форматі JSON.

Особливу увагу приділено багаторівневій архітектурі застосунку, зокрема розмежуванню логіки обробки запитів (контролери), бізнес-логіки (сервіси) та доступу до даних (репозиторії), що, як очікується, забезпечить високу модульність системи. У якості технологічної основи обрано фреймворк Spring Boot, який дає змогу зменшити обсяг конфігураційного коду та забезпечити швидкий запуск застосунку. У межах дослідження також приділено увагу аспектам безпеки, зокрема використанню JWT для реалізації авторизації користувачів без збереження сесій на сервері.

У результаті проведеного аналізу сформовано концептуальну модель архітектури, що в подальшому може бути використана для реалізації високоякісного серверного інтерфейсу системи управління задачами. Такий підхід забезпечить гнучкість, розширюваність і адаптивність до змін функціональних вимог під час розвитку застосунку.

Ключові слова: REST API, Spring, таск-менеджер, архітектура, JWT, безпека, Java, база даних.

Abstract

This paper proposes an approach to designing and implementing a REST API for a task management web application – an essential tool for efficiently organizing personal and team tasks. The development of such systems requires the use of modern architectural approaches that ensure scalability, security, and ease of maintenance. The study analyzes the fundamental principles of REST architecture, including clear resource identification, the use of standard HTTP methods, stateless interactions, and structured data in JSON format.

Special attention is given to a multi-layered application architecture, in particular the separation of request processing logic (controllers), business logic (services), and data access logic (repositories), which is expected to ensure a high level of system modularity. The Spring Boot framework was chosen as the technological foundation, allowing reduced configuration overhead and rapid application deployment. The study also addresses security aspects, particularly the use of JWT to implement user authorization without storing sessions on the server.

As a result of the analysis, a conceptual architectural model was developed, which may be used in the future for implementing a high-quality server interface for a task management system. This approach ensures flexibility, extensibility, and adaptability to changing functional requirements during application evolution.

Keywords: REST API, Spring, task manager, architecture, JWT, security, Java, database.

Вступ

У сучасному програмному забезпеченні веб-застосунки стали невід'ємною частиною як особистої, так і професійної діяльності. Вони охоплюють широкий спектр функцій: від соціального спілкування до управління ресурсами, проєктами та завданнями. Особливої популярності набули системи керування задачами – так звані таск-менеджери, які допомагають користувачам організовувати власну діяльність, встановлювати дедлайни, контролювати прогрес виконання завдань та координувати командну роботу. Їх використання сприяє підвищенню особистої ефективності, а в корпоративному середовищі – оптимізації бізнес-процесів, покращенню комунікації та підвищенню прозорості проєктної діяльності.

У зв’язку з ускладненням вимог до програмних рішень зростає потреба у створенні інтерфейсів, які можуть масштабуватися, бути безпечними, надійними та легко інтегруватися з іншими компонентами цифрової інфраструктури. Саме тому на особливу увагу заслуговує розробка REST API – серверної частини застосунку, яка відповідає за обробку запитів, доступ до даних і виконання бізнес-логіки. REST-архітектура, з її уніфікованими підходами до роботи з ресурсами, сумісністю з HTTP-протоколом і широкою підтримкою серед веб-фреймворків, є на сьогодні одним із найкращих рішень для створення такого інтерфейсу.

Актуальність обраної теми полягає у необхідності розробки надійного та зручного у використанні серверного API, який дозволяє ефективно об’єднувати клієнтські інтерфейси – як веб, так і мобільні – із бекендом, що виконує основну логіку управління задачами. При цьому надзвичайно важливим є не лише написання коду, а й грамотний вибір архітектурних рішень, які безпосередньо впливають на стабільність, продуктивність, безпеку та можливість подальшого розвитку системи.

Метою цієї роботи є дослідження архітектурних підходів до створення REST API для таск-менеджера, аналіз сучасних технологій і патернів, що можуть бути використані для реалізації такого застосунку, а також обґрунтування вибору фреймворку Spring як основи для реалізації прототипу системи.

Дослідження

Сучасні системи управління задачами повинні відповідати низці ключових вимог: бути масштабованими, безпечними, гнучкими у впровадженні нових функцій та зручними у використанні. Щоб досягти цього, необхідно на етапі проєктування API приймати архітектурні рішення, які дозволять у майбутньому не лише забезпечити стабільність системи, але й адаптувати її до можливого зростання навантаження, кількості користувачів та змін у функціональних вимогах.

У центрі дослідження перебуває архітектурний стиль REST (Representational State Transfer), який завдяки своїй стандартизованості, сумісності з HTTP-протоколом та концептуальній простоті є базовим підходом для побудови API в більшості сучасних веб-сервісів. REST передбачає представлення кожної сутності системи – користувача, задачі, проєкту – як ресурсу, з яким можна взаємодіяти за допомогою чітко визначених HTTP-методів. Дотримання принципів REST дозволить у майбутньому сформувати логічну та передбачувану структуру API. Це спрощує тестування, налагодження та супровід застосунку [1].

Окремо варто розглянути використання багаторівневої архітектури. Запропоновано структурувати застосунок на три основні логічні рівні: контролери, які приймають та обробляють HTTP-запити; сервіси, що містять бізнес-логіку застосунку; та репозиторії, які забезпечують взаємодію з базою даних (рис. 1). Такий підхід дозволяє не лише підвищити модульність системи, а й досягти чіткого розмежування відповідальностей, що значно полегшить розширення функціональності у майбутньому. Наприклад, змінюючи спосіб обробки певних бізнес-правил, розробник зможе модифікувати лише сервісний шар без необхідності втручання у контролери або сховище даних.

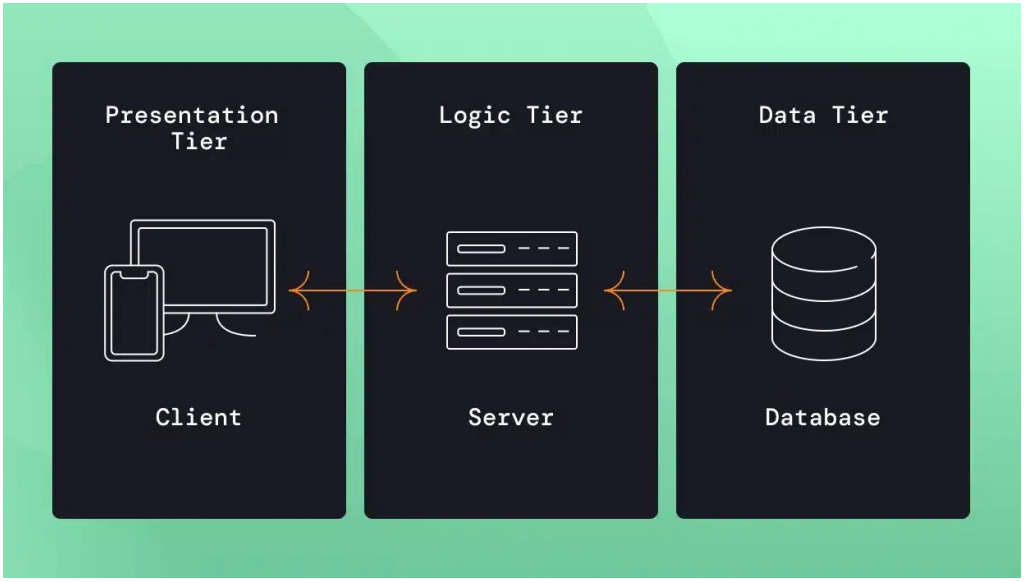

Рисунок 1 – Трирівневий застосунок

Однією з ключових переваг трирівневої архітектури застосунків є її очевидна ефективність у порівнянні з дворівневою моделлю. Такий підхід забезпечує масштабованість, адже кожен рівень (контролерний, логічний, даних) може масштабуватись окремо залежно від навантаження. Завдяки чіткому розмежуванню відповідальностей покращується підтримка системи: зміни в одному рівні рідко впливають на інші, що дозволяє швидше вносити правки та оновлення. Гнучкість реалізації полягає в можливості заміни технологій всередині одного рівня без зміни всієї системи, за умови збереження інтерфейсних контрактів між шарами. Безпека також зростає – можна впроваджувати багаторівневі механізми захисту, ізолювати критичні дані на глибинних шарах та проводити валідацію на кількох рівнях, що зменшує ризики, пов'язані з вразливостями. Компоненти логічного рівня можуть бути повторно використані в інших застосунках через REST API. Крім того, така архітектура сприяє спеціалізації розробників: команди можуть зосереджуватись на окремих шарах (фронтенд, бекенд, база даних), що оптимізує процес розробки та підвищує ефективність. У сукупності ці переваги пояснюють, чому трирівнева архітектура стала стандартом для більшості сучасних застосунків[2].

Як основу для реалізації REST API доцільне використання фреймворку Spring Boot. Цей фреймворк дозволяє уникнути надлишкової конфігурації, забезпечує автоматичне підключення необхідних компонентів та підтримує інтеграцію з широким спектром технологій, включно з Spring Security, Spring Data та іншими. Очікується, що завдяки Spring Boot буде спрощено процес ініціалізації проєкту, запуску веб-серверу, реалізації обробки запитів і тестування системи. Spring Boot побудований на основі фреймворку Spring та включає всі його можливості, водночас спрощуючи процес конфігурації. Він став одним з найуживаніших інструментів багатьох розробників завдяки можливості швидкої розробки, що дозволяє зосередитися на бізнес-логіці замість витрачання часу на налаштування. Spring Boot є фреймворком, орієнтованим на мікросервіси, який забезпечує швидку розробку готових до використання застосунків [3].

Безпека займає важливе місце в архітектурі будь-якого веб-застосунку, що працює з персональними даними користувачів. У межах дослідження було розглянуто можливість застосування механізму автентифікації та авторизації на основі JSON Web Token (JWT). Цей підхід дозволяє реалізувати авторизацію: після входу користувач отримує токен, який передається в заголовку кожного запиту. Токен містить у зашифрованому вигляді дані про роль користувача, його ідентифікатор тощо. На відміну від традиційного зберігання сесій на сервері, JWT забезпечує легку масштабованість і знижує навантаження на серверну частину, що особливо важливо в умовах великої кількості одночасних користувачів [4].

Також варто вибрати систему управління базами даних. PostgreSQL запропоновано як оптимальний варіант для реляційного моделювання даних. Ця СУБД підтримує транзакції, цілісність даних, гнучке індексування та складні запити, що дозволить ефективно обробляти інформацію про користувачів, задачі, коментарі, проєкти тощо. У поєднанні з технологією Spring Data JPA це дозволяє значно спростити реалізацію запитів до БД, делегуючи створення SQL-операцій фреймворку на основі сигнатур методів. Це також сприятиме скороченню коду і зменшенню ймовірності помилок [5].

Таким чином, результати дослідження сформували концептуальну основу для реалізації REST API таск-менеджера. Обрані архітектурні рішення є актуальними, обґрунтованими й орієнтованими на довгострокову підтримку, розвиток і масштабування системи.

Висновок

У ході дослідження було обґрунтовано архітектурні підходи до створення REST API для таск-менеджера, що можуть забезпечити масштабованість, безпеку та зручність подальшої підтримки системи. REST-архітектура дозволяє організувати чітку взаємодію між клієнтом і сервером, а трирівнева структура застосунку – розмежувати відповідальності та підвищити гнучкість розробки.

Фреймворк Spring Boot розглядається як оптимальна основа для реалізації завдяки автоматизації конфігурацій та підтримці необхідних модулів. Передбачене використання JWT для авторизації забезпечує безпечний і незалежний від сесій підхід до контролю доступу. Вибір PostgreSQL дозволяє ефективно працювати з реляційними структурами.

Таким чином, сформовані рішення можуть стати надійною основою для майбутньої реалізації таск-менеджера з сучасною, функціональною та розширюваною серверною частиною.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. GeeksforGeeks. Best practices while making REST APIs in Spring Boot application [Електронний ресурс] / GeeksforGeeks. – Режим доступу: https://www.geeksforgeeks.org/best-practices-while-making-rest-apis-in-spring-boot-application/ – Дата звернення: 05.05.2025.
2. vFunction. Understanding 3-Tier Architecture: Benefits, Examples & More [Електронний ресурс] / vFunction. – Режим доступу: https://vfunction.com/blog/3-tier-application/#toc-heading-6 – Дата звернення: 16.05.2025
3. GeeksforGeeks. Introduction to Spring Boot [Електронний ресурс] / GeeksforGeeks. – Режим доступу: https://www.geeksforgeeks.org/introduction-to-spring-boot/ – Дата звернення: 16.05.2025.
4. Onu V. Implementing JWT authentication in a simple Spring Boot application with Java [Електронний ресурс] / Victor Onu // Medium. – Режим доступу: https://medium.com/@victoronu/implementing-jwt-authentication-in-a-simple-spring-boot-application-with-java-b3135dbdb17b – Дата звернення: 06.05.2025.
5. HackerNoon. Using Postgres effectively in Spring Boot applications [Електронний ресурс] / HackerNoon. – Режим доступу: https://hackernoon.com/using-postgres-effectively-in-spring-boot-applications – Дата звернення: 06.05.2025.

Галіброда Анатолій Сергійович – студент групи 1ІСТ-24м, кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, Вінниця. e-mail: galibrodaa@gmail.com.

Науковий керівник: ***Кабачій Владислав Володимирович*** – к. т. н., доцент кафедри Автоматизації та інтелектуальних інформаційних технологій, Вінницький національний технічний університет, м. Вінниця, e-mail: vkabachiy@gmail.com

Halibroda Anatolii Serhiiovych – student Group 1IST-24m, Department of Automation and Intelligent Information Technologies, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, Ukraine, e-mail: galibrodaa@gmail.com

Supervisor: **Kabachii Vladyslav Volodymyrovych**, PhD in Technical Sciences, Associate Professor of the Department of Automation and Intelligent Information Technologies, Vinnytsia National Technical University, Vinnytsia, Ukraine, e-mail: vkabachiy@gmail.com