УДК 004.4

О. В. Войцеховська
А. К. Рижков

АНАЛІЗ АРХІТЕКТУРИ РОЗПОДІЛЕНОГО СЛАБКОЗВ'ЯЗАНОГО СЕРВЕРНОГО ДОДАТКУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Вінницький національний технічний університет

Анотація

В роботі проведено аналіз методів взаємодії між клієнтською та серверною частинами додатку. Для різних частин додатку використано технологію, яка мінімізує використання трафіку, сприяє кешуванню та високій швидкодії.

Ключові слова: авторизація, GraphQL, REST API, клієнт-сервер.

Abstract

The analysis of methods of interaction between the client and the server is carried out in this paper. For different parts of the application, technology is used that minimizes the use of traffic, promotes caching and speed.

Keywords: authorization, GraphQL, REST API, client-server.

Вступ

В даний час клієнт-серверні додатки є найпопулярнішим способом взаємодії в інформаційних системах. У зв'язку з цим постає питання способів взаємодії між частинами інформаційної системи. Ця взаємодія повинна сприяти мінімізації трафіку, шляхом зменшення кількості запитів та передачі лише необхідних даних, а також захисту інформації. Для різних частин додатку, оптимальними є різні методи клієнт-серверної взаємодії. На сьогодні найбільш популярними технологіями такої взаємодії є мова запитів для роботи з даними GraphQL[1] та архітектурний підхід REST API [2] для покращеного використання протоколу HTTP.

Для того, щоб оптимізувати взаємодію між клієнтом та сервером через протокол HTTP, необхідно використати набір архітектурних обмежень REST (Representational state transfer) та специфікацію GraphQL, кожен з яких має свої переваги та недоліки. Тому потрібно вдало їх поєднувати.

Результати досліджень

Використання клієнт-серверної архітектури у сучасному програмному забезпеченні забезпечує централізований доступ до даних та їх захист, що запобігає дублюванню серверного коду на клієнтах. Окрім цього, така архітектура забезпечує мінімальні вимоги до комп'ютерів, на яких встановлений клієнт, за рахунок винесення всіх обрахунків на сервер. З точки зору розробника, така система є достатньо гнучкою для розширення, а отже дозволить зручно додавати нові функціональні можливості та усувати знайдені дефекти. Також у системі з таким підходом набагато простіше підвищити захист локальної мережі. [3]

Описана вище архітектура зображена на рис. 1. На рівні доступу до даних описані моделі сутностей та логіка роботи з базою даних. Після того як цей рівень отримує інформацію із бази даних, вона передається на рівень вище – рівень бізнес логіки. У рівні бізнес логіки відбуваються маніпуляції над отриманими даними, інтеграції зі сторонніми сервісами та перевірки доступу. Після чого інформація передається на рівень програмного інтерфейсу додатку або в веб-додаток.

Звісно, у такого підходу є і недоліки, зокрема обслуговування серверів набагато важче, ніж обслуговування клієнтських машин. Але у цьому випадку такі проблеми не є суттєвими. При використанні правильного стеку технологій, а також грамотній реалізації, усі мінуси можна усунути, а переваги лише посилити. [4]

Слід враховувати, що немає чіткого розділення обладнання на клієнтське та серверне. Просто така архітектура дає можливість перерозподілити і оптимізувати завантаженість і розподілити функціональність між робочими станціями.

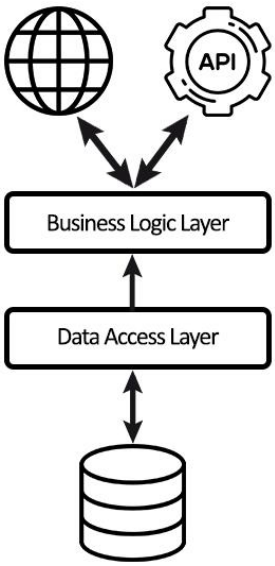

Рисунок 1 – Узагальнена схема багаторівневої архітектури програмного продукту

Багаторівнева архітектура за рахунок своєї гнучкості може легко розширюватись за допомогою встановлення групи додаткових серверних частин. Подібна віртуальна архітектура дозволяє суттєво підвищити ефективність функціонування інформаційних систем, а також виконати оптимізоване розділення частини її програмно-апаратних ресурсів.

Нюанси взаємодії системи клієнт-сервер дозволяють користувачам швидко розділяти визначені функціональні можливості та обчислювальне навантаження між підключеними клієнтськими веб-продуктами з серверними додатками при використанні різних систем тестування.

Висновки

Для забезпечення високого рівня продуктивності, та оптимізації усіх програмних процесів при проектуванні слабкозв'язаного серверного додатку будуть використовуватись надійні підходи до проектування програмного продукту, що дасть можливість легко масштабувати серверну частину.

Актуальні та перспективні технології GraphQL та REST API дадуть можливості ефективно та швидко взаємодіяти з серверною частиною додатку. REST API простіше у використанні, а GraphQL, дає можливість отримувати лише ту частину даних, яка потрібна у конкретний момент. Грамотне поєднання цих технологій дозволить використовувати лише їх переваги.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Introduction to GraphQL [Електронний ресурс] – Режим доступу до ресурсу: https://graphql.org/learn/

2. OpenAPI Specification. Introductions Definitions and more [Електронний ресурс] – Режим доступу до ресурсу: https://spec.openapis.org/oas/v3.1.0

3. Clent-Server Model [Електронний ресурс] – Режим доступу до ресурсу:https://www.geeksforgeeks.org/client-server-model/

4. Client Server Computing [Електронний ресурс] – Режим доступу до ресурсу:https://www.tutorialspoint.com/Client-Server-Computing

Войцеховська Олена Валеріївна – кандидат технічних наук, доцент кафедри обчислювальної техніки, Вінницький національний технічний університет, Вінниця

Рижков Андрій Костянтинович – студент групи 1КІ-20МС, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: ryzhkovanruha@gmail.com.

Voytsekhovska Olena V. **—** PhD, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University

Ryzhkov Andrii K. – students, 1KI-20MS, Faculty of information Technologies and Computer Engeneering, Vinnytsa National Technical University, email: ryzhkovanruha@gmail.com.