Technoarete Transactions on Advances in Computer Applications (TTACA)
Volume 2, Issue 4, December 2023
e-ISSN: 2583-3472

Building a Unified Supergraph for Distributed Data Graphs Using Apollo GraphQL Federation

Venkata Thota

Professional: Solution Architect/Lead
Email: vcthota@gmail.com

Abstract

This article discusses the implementation of a Unified Supergraph for Distributed Data Graphs using the Apollo GraphQL Federation architecture, addressing the challenges of efficiently managing complex data sources within contemporary technological landscapes. With the Apollo Federation, multiple GraphQL APIs are seamlessly integrated into a unified Supergraph, providing developers with a scalable, modular, and interconnected approach. This article provides a detailed description of the components of the Apollo Federation, which includes Apollo GraphOS, Supergraph, Subgraph, RoverCLI, Apollo Router, and Managed Federation. Apollo GraphOS, provides a SaaS solution for schema registries, ensuring the integrity of underlying schemas, while Supergraph orchestrates and combines multiple subgraphs into one coherent GraphQL API. The Apollo Router serves as the public-facing interface for the Supergraph, while RoverCLI facilitates precise management and deployment of subgraphs. The Managed Federation simplifies the process of integrating and managing subgraphs, streamlining the process and ensuring consistency. A real-time demonstration of Apollo Federation with Netflix DGS is presented at the end of the article, which illustrates the framework's application in creating a unified Supergraph for Spring Boot applications. Organizations managing distributed data graphs can benefit from the advantages of Managed Federation, including router and composition stability, schema flexibility, and a real-time example using Netflix DGS.

Keywords

Apollo GraphQL Federation, GraphQL APIs, Supergraph, RoverCLI, Apollo Router, GraphQL Architecture, GraphQL Services

INTRODUCTION

In today's technology-driven world, managing complex data sources is a top priority for organizations seeking to stay ahead. The Apollo GraphQL Federation architecture [1], has emerged as a revolutionary solution to address this challenge, offering a paradigm shift in the way organizations integrate and orchestrate their GraphQL APIs. This architecture provides a more efficient way to manage data sources, giving organizations a competitive edge in the dynamic landscape of modern technology.

In this article, we will explore the details of the Apollo Federation and its key components. We will also discuss the numerous benefits it offers. By integrating multiple GraphQL APIs [2] into a single Supergraph, the Apollo Federation allows developers to create scalable, modular, and interconnected APIs. This is not This approach not only enhances application flexibility and agility, but also allows complex data sources to be managed in a more efficient manner.

In the initial section of the article, the Apollo Federation architecture components are discussed, offering a detailed introduction to GraphOS [1], Supergraph [1], Subgraph [1], RoverCLI [1], and Apollo Router [1]. Every component is vital in ensuring the smooth management and integration of GraphQL services.

The subsequent sections deep dive into the functionalities of Apollo GraphOS, the central hub of the architecture, and the Supergraph, a high-level layer orchestrating multiple subgraph. The article sheds light on how the Router and RoverCLI streamline the process of managing and deploying subgraphs, ensuring the integrity and efficiency of GraphQL schema management.

In the latter part of the article, Apollo Federation's managed federation capability is explored in detail. This feature simplifies the integration and management of subgraphs, allowing for autonomous schema publication and dynamic updates. The article walks through the managed federation process, highlighting its role in ensuring a consistent and up-to-date Supergraph. The article then explains the benefits of managed federation, emphasizing the stability it brings to routers, the reliability of Supergraph compositions, and the flexibility it offers for safe schema changes. To demonstrate how these concepts work in real-time, the article concludes with an example of Apollo Federation working with Netflix DGS (Domain Graph Service Framework).[3]

Join us on this exploration of the Apollo GraphQL Federation architecture, where intricate data management meets innovative solutions, transforming the way organizations interact with and harness the power of GraphQL APIs.

OVERVIEW OF APOLLO SYSTEM COMPONENTS AND FLOW

In Figure.1, we can see how the Apollo Federation architecture enables the smooth integration of multiple GraphQL APIs into a single Supergraph. This innovative approach helps organizations to efficiently combine and manage various GraphQL services, creating a connected and

e-ISSN: 2583-3472

cohesive data layer. With the Apollo Federation, developers can lay a strong foundation for building scalable, modular, and interconnected GraphQL APIs, which enhances their application's agility and flexibility. This architecture enables efficient management and composition of complex data sources, making it possible to access unified and dynamic data with GraphQL, thus unlocking the potential for more efficient and unified data access.

The following diagram Figure.1 illustrates the key components of the Apollo Federation architecture, their relationships, and how data flows through the architecture.

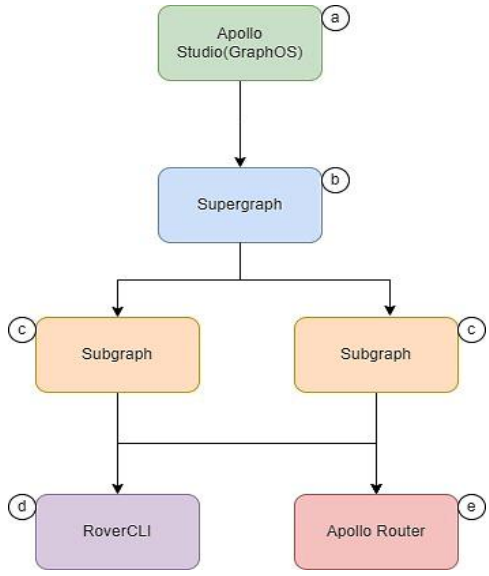

Figure. 1. A set of Apollo components for the Federation

Apollo GraphOS:

The Apollo GraphOS serves as the foundation for the entire architectural framework. GraphOS is a Software as a Service (SaaS) solution that acts as a robust schema registry for orchestrating GraphQL data. Supergraph composition is primarily concerned with ensuring the integrity and coherence of the underlying schemas.

The software provides a sophisticated managed mode that is tailored specifically for Apollo Federation. As a result, it provides a seamless solution for dynamically evolving and expanding GraphQL APIs. In order to ensure continuous, uninterrupted service delivery, Apollo GraphOS has been meticulously designed to enable users to make changes and enhancements to their Supergraph without any disruption in service.

An integral part of the GraphOS ecosystem is the GraphOS Studio, which is a web-based interface that is at the heart of the platform. By using this tool, users are able to interact with the GraphOS environment in a variety of ways. A number of features are available through this software, including the ability to establish organizational structures, which serves as a foundation for future operations. Furthermore, GraphOS Studio provides users with the capability to explore and visualize schemas, facilitating a more in-depth understanding of the data structures and relationships in their Supergraph.

Supergraph:

A Supergraph is a high-level layer that orchestrates and combines multiple Subgraphs to form the unified GraphQL API. In addition to Apollo GraphOS, it is connected to various subgraphs. An innovative concept called a "Supergraph" arises when these individual subgraphs are strategically stitched together to form a unified, higher-level GraphQL schema. This aggregation of subgraphs is orchestrated in a manner that facilitates their cooperation, enabling clients to interact with the Supergraph as if it were one cohesive GraphQL schema. The Supergraph facilitates efficient and streamlined data retrieval while providing clients with an integrated API that aggregates data from all constituent subgraphs. In essence, the Supergraph provides a consolidated and centralized view of the data and functionality of an application, abstracting the complexity of interacting with multiple subgraphs. A federated approach to GraphQL architecture simplifies client-side queries, optimizes data retrieval, and enhances the overall developer and client experience.

Subgraph:

As part of a federated architecture, individual GraphQL APIs are referred to as "subgraphs." Subgraphs are self-contained GraphQL schemas and functionality that are typically associated with specific microservices and data sources within the application's architecture. Subgraphs have their own types, queries, mutations, and data structures.

RoverCLI:

As an important component of the Apollo Federation architecture, RoverCLI serves as a robust command-line utility meticulously designed for the precise management and deployment of Subgraphs. The GraphQL command-line interface provides a comprehensive suite of essential functionalities that together enhance the integrity and efficiency of the GraphQL schema management process.

Through RoverCLI admins performs the critical function of schema linting and validation as one of its core capabilities. As a result, GraphQL schemas conform to predefined standards and adhere to best practices, which reduces the likelihood of errors and inconsistencies within Supergraph. The robust schema checks contribute significantly to the reliability and maintainability of the entire GraphQL ecosystem.

Additionally, RoverCLI commands simplifies the complex process of schema publication. GraphQL enables users to propagate changes across their GraphQL infrastructure efficiently by providing a seamless mechanism for schema deployment. This agility in schema management is instrumental in supporting a dynamic and responsive GraphQL ecosystem.

Router:

Apollo Router is a crucial component of the Apollo Federation architecture. It serves as the public-facing entry point for your Supergraph which is a combination of multiple

e-ISSN: 2583-3472

GraphQL subgraphs. As an essential component, it operates at a high level of efficiency, efficiently handling incoming GraphQL operations via a routing mechanism that is intelligent and discerning. A seamless data flow is orchestrated by skillfully directing each query to its respective subgraph within the Supergraph.

A notable feature of the Apollo Router is its ability to retrieve the Supergraph definition from Apollo Studio, which performs subgraph calls in response to client requests. This integration enhances the router's capabilities and ensures that the Supergraph remains current and responsive to evolving data requirements.

Apollo Router's orchestration ensures a seamless and coherent aggregation of data from different sources, allowing clients to interact with the Supergraph as if it were a unified GraphQL server. Furthermore, Apollo Router simplifies the client-server interaction without imposing complex client-side configurations, making it a powerful tool for aggregating GraphQL data.

APOLLO MANAGED FEDERATION USING GRAPHOS (APOLLO STUDIO-WEB INTERFACE) [4]:

Figure. 2 presents the various system components and highlights the key benefits of integrating GraphOS. One significant advantage of Apollo Federation is its managed federation feature, which simplifies the integration and management of subgraphs. Managed federation ensures a streamlined process by enabling each subgraph to independently publish its schema to GraphOS. GraphOS then performs schema composition validation to ensure a successful merging of subgraphs into a unified Supergraph.

GraphOS offers a reliable and efficient managed federation support which makes it easy for GraphQL developers to integrate and manage subgraphs in Apollo Federation setup. With managed federation, each subgraph independently publishes its schema to GraphOS, thereby simplifying the process. The GraphOS system then validates the schema composition to ensure that all subgraphs have merged successfully into a unified Supergraph.

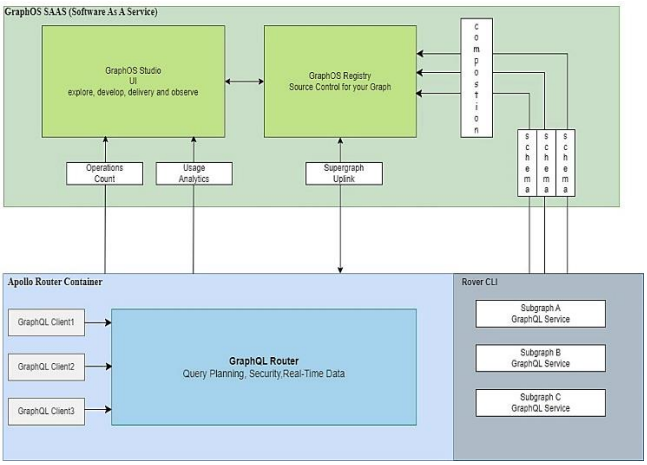

Figure. 2. GraphOS System Architecture

The process of managed federation with Apollo GraphOS involves several key steps:

1. **Subgraph Schema Publication:**
   - Each subgraph in Apollo Federation setup publishes its schema to GraphOS. The schema represents the data and functionality provided by the subgraph.
   - Subgraphs can independently update their schemas during the publication process.
2. **Schema Composition Validation:**
   - In the validation process, GraphOS plays an important role. In order to ensure that the subgraph schemas can be effectively combined into a coherent Supergraph schema, it meticulously examines the schemas of each subgraph.
   - This step ensures that the Supergraph can provide a unified API while avoiding conflicts or inconsistencies between the subgraphs.
3. **Supergraph Configuration Update:**
   - GraphOS updates the configuration of your Supergraph upon successful validation and composition. It is this configuration that determines how the subgraphs relate to one another and how they fit together.
   - GraphOS ensures that the Supergraph accurately reflects the combined data and functionality of the subgraphs.
4. **Special Endpoint (Uplink):**
   - The latest Supergraph configuration is available at a special endpoint called the "uplink."
   - To keep track of any changes to the Supergraph's structure, Apollo Router periodically polls this uplink.

Managed federation with Apollo GraphOS simplifies the orchestration of Apollo Federation architecture. Subgraph schemas are automatically published, validated, and maintained, ensuring that Supergraph remains consistent and up-to-date.

In addition to simplifying the development and management of complex GraphQL architectures, this approach offers a level of confidence in the stability and cohesiveness of Supergraph, enabling admins to scale and maintain GraphQL services more easily.

Benefits of Managed Federation [5]

Managed federation brings several benefits to organizations adopting the Apollo Federation architecture:

Router Stability: Modifying subgraph schemas or adding/removing entire subgraphs doesn't require router modifications or redeployment, maximizing uptime.

Composition Stability: Router updates always lead to valid Supergraph's compositions, ensuring consistency and reliability.

Schema Flexibility: Externalized configuration allows for safe schema changes, such as migrating types or fields between subgraphs, enhancing the safety and flexibility of GraphQL schemas.

Technoarete Transactions on Advances in Computer Applications (TTACA)
Volume 2, Issue 4, December 2023
e-ISSN: 2583-3472

REAL-TIME EXAMPLE: APOLLO FEDERATION WITH NETFLIX DGS (DOMAIN GRAPH SERVICE) [3][6]

The DGS [3] Framework, standing for Domain Graph Service Framework, provides a robust GraphQL [6] server framework tailored specifically for Spring Boot applications [4]. A framework developed and maintained by the Netflix team, this framework offers an array of essential features and capabilities designed to simplify the development of GraphQL servers within the Spring Boot framework.

The diagram in Figure. 3 shows an example of federation, where multiple GraphQL APIs (represented as subgraphs), such as Accounts, Products, Inventory, and Reviews, are integrated to create a unified Supergraph. The process of amalgamation and orchestration is facilitated and managed using RoverCLI to ensure a seamless and coherent composition of the Supergraph.

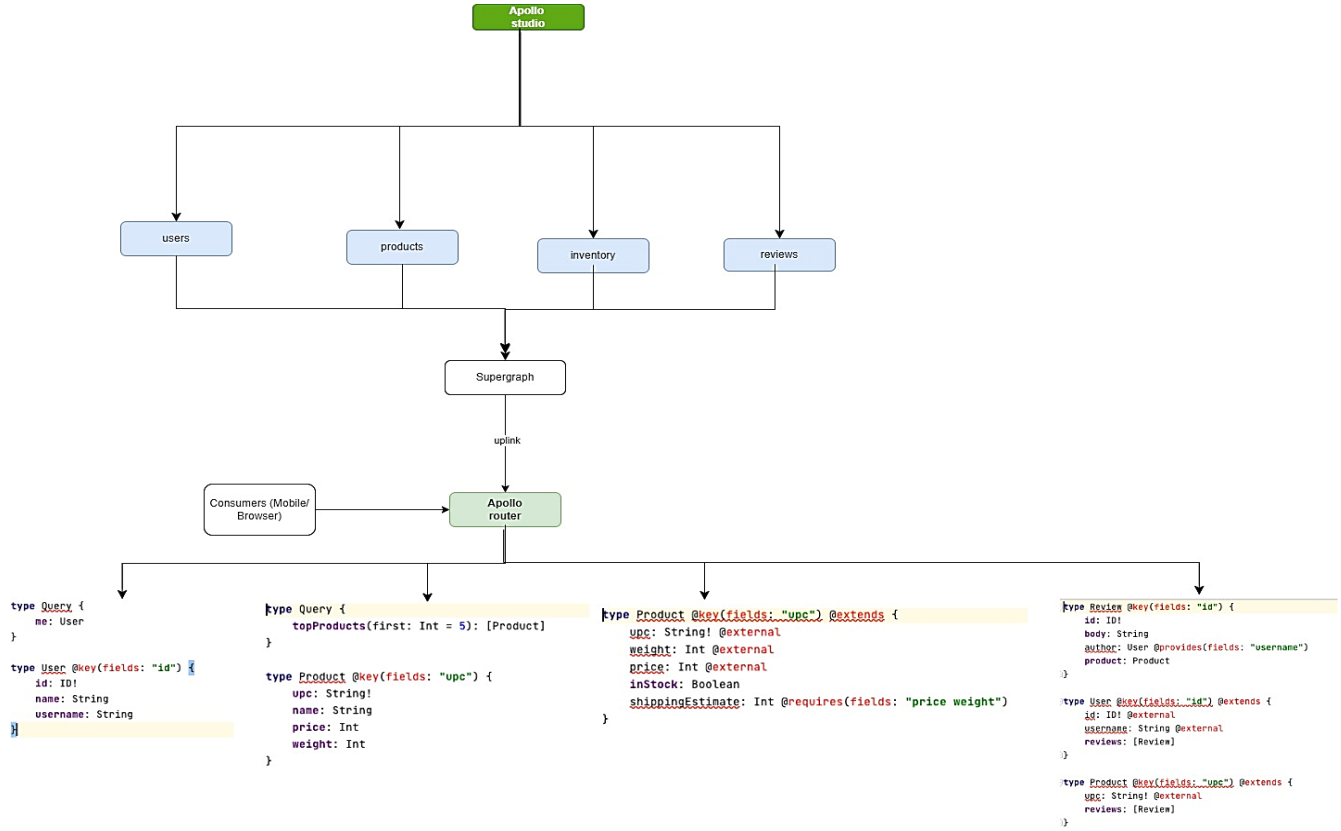

Figure.3. Different Subgraph Services and GraphOS execution flow

RELATIONSHIPS BETWEEN SUBGRAPH SCHEMAS [5]:

Figure. 4. illustrates the relationships between the User, Review, and Product subgraphs to facilitate client queries.

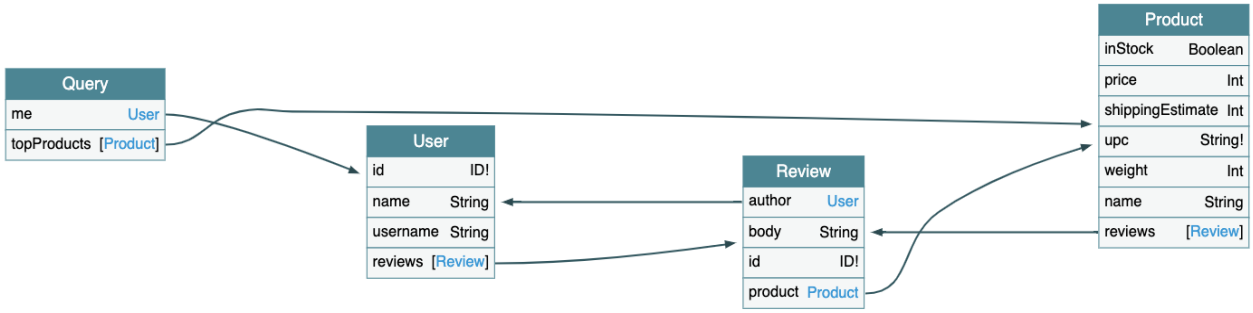

Figure.4. Subgraph ER diagram for queries and mutations

Technoarete Transactions on Advances in Computer Applications (TTACA)
Volume 2, Issue 4, December 2023
e-ISSN: 2583-3472

APOLLO FEDERATION TESTING:

Figure.5 and Figure.6 demonstrate the execution of a Federation[5] query. They illustrate how the association between different subgraphs works.

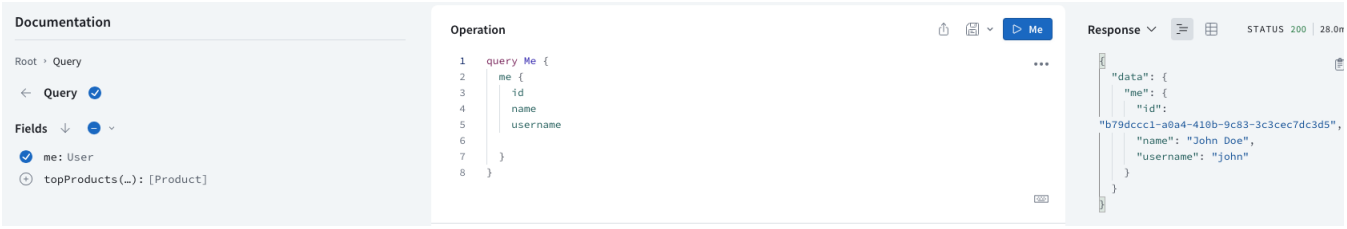

Figure.5. Sample federation

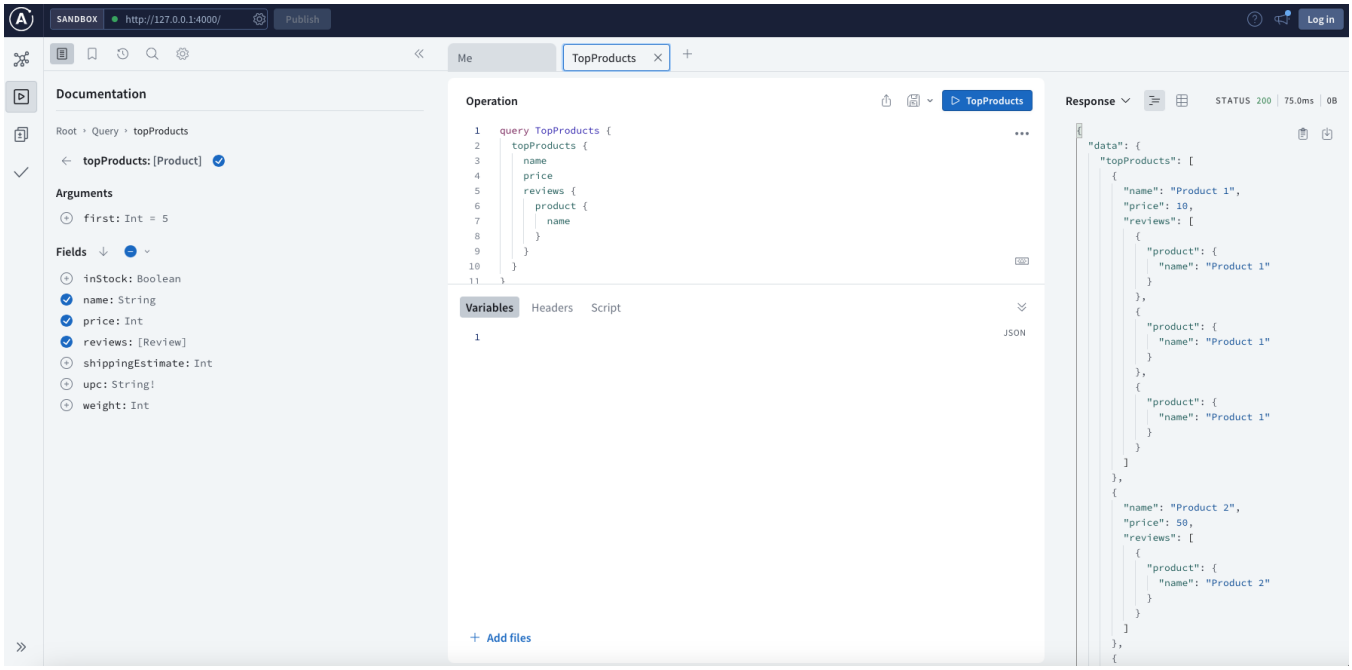

Figure.6. Sample federation

SOURCE CODE REFERENCE:

Refer to the below repository for a complete working example.

https://github.com/explorecloudsolutions/netflix-dgs-federation2

CONCLUSION:

In conclusion, the implementation of a Unified Supergraph for Distributed Data Graphs using the Apollo GraphQL Federation architecture provides organizations with a powerful solution for efficiently managing complex data sources. The Apollo Federation, with its components such as Apollo GraphOS, Supergraph, Subgraph, RoverCLI, Apollo Router, and Managed Federation, enables the creation of scalable, modular, and interconnected GraphQL APIs. The managed federation feature simplifies the integration and management of subgraphs, offering benefits such as router and composition stability, schema flexibility, and a streamlined development process. The real-time example with Netflix DGS demonstrates the practical application of Apollo Federation in creating a unified Supergraph for Spring Boot applications. Overall, organizations adopting Apollo Federation can enhance the agility, flexibility, and efficiency of their GraphQL-based applications, making data access more unified and dynamic.

ACKNOWLEDGEMENTS

I want to express our sincere appreciation and gratitude to the Apollo GraphQL Community (https://community.apollographql.com/). This is a vibrant and collaborative ecosystem of developers, contributors, and enthusiasts. The success of Apollo GraphQL as a cutting-edge technology is a direct result of the collective efforts and passion of this community. I would like to thank all the developers and contributors for their dedication to advancing GraphQL technology, for their insightful discussions, and for their invaluable contributions to the open-source Apollo

e-ISSN: 2583-3472

projects. Your collaboration and shared expertise continue to shape the future of GraphQL, empowering developers worldwide. Together, we celebrate the spirit of innovation, openness, and knowledge-sharing that defines the Apollo GraphQL Community.

REFERENCES

[1] Apollo, GraphQL, 2023. [Online]. Available: https://www.apollographql.com/docs/
[2] IBM, GraphQL, 2023. [Online]. Available: https://www.ibm.com/docs/en/scis?topic=reference-graphql
[3] Netflix DGS, [Online]. Available: https://netflix.github.io/dgs/
[4] GraphQL Summit in San Deigo, Oct 10, 2023, [online] Available: https://www.apollographql.com/tutorials/summit-2022/wrapup
[5] GraphQL setup with Federation2, [Online]. Available: https://www.apollographql.com/docs/federation/quickstart/setup/
[6] V. Spasev, I. Dimitrovski and I. Kitanovski, "An Overview of GraphQL: Core Features and Architecture", [online] Available: https://repository.ukim.mk/bitstream/20.500.12188/9488/1/an-overview-of-graphql-core-features-and--architecture.pdf