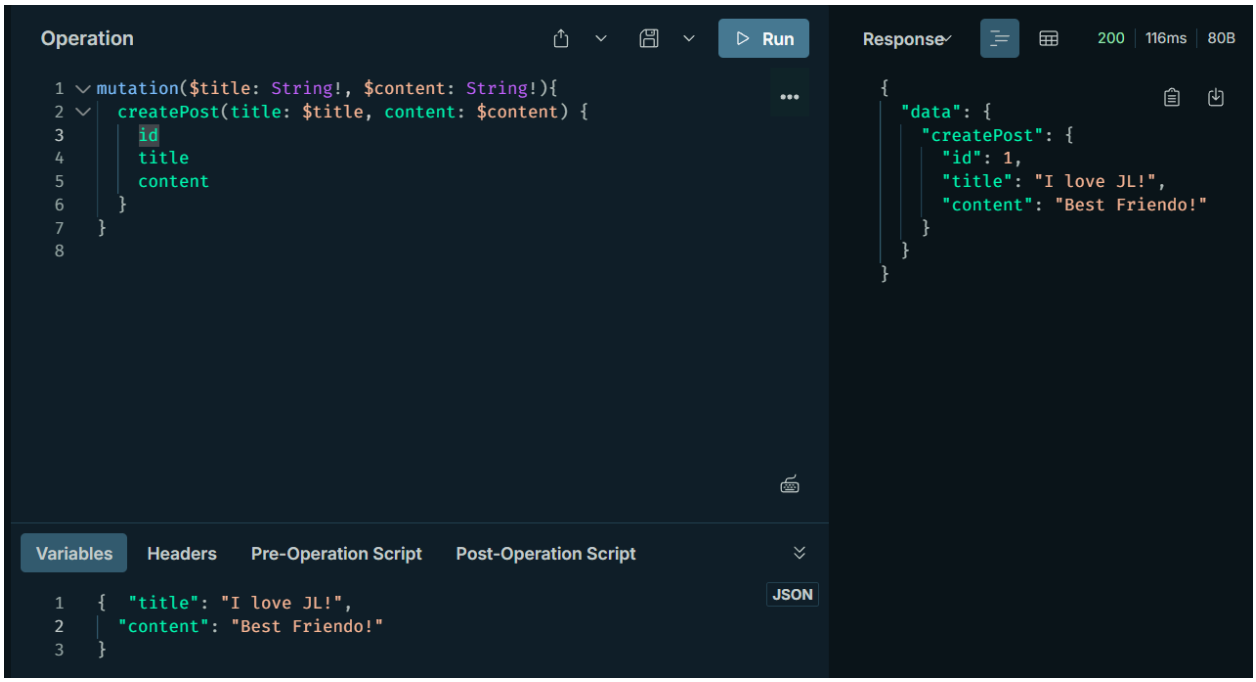
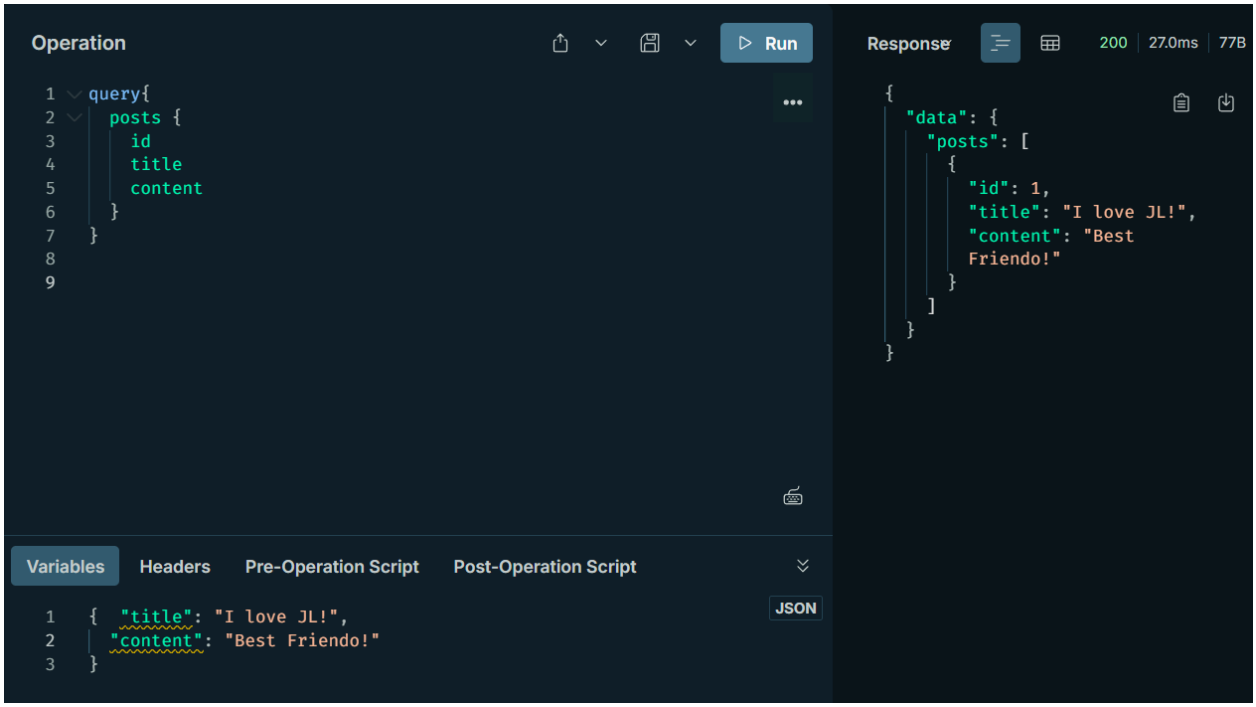
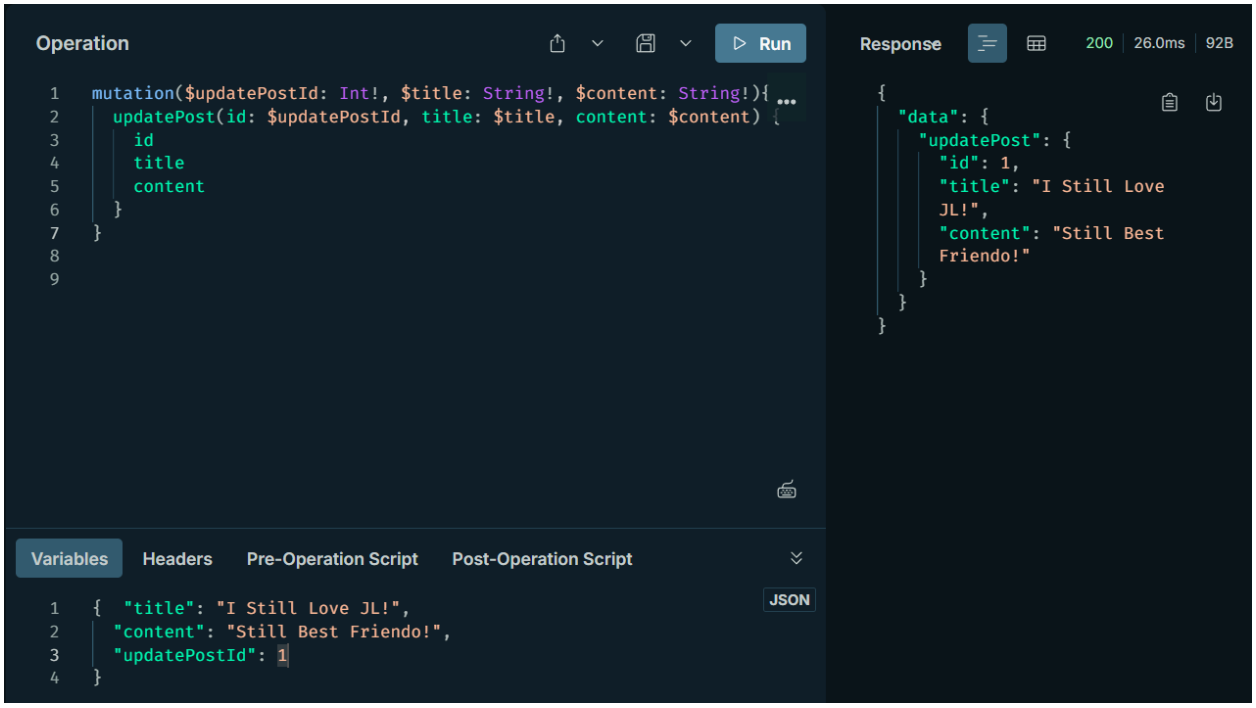
Post - Services
Creation Example:



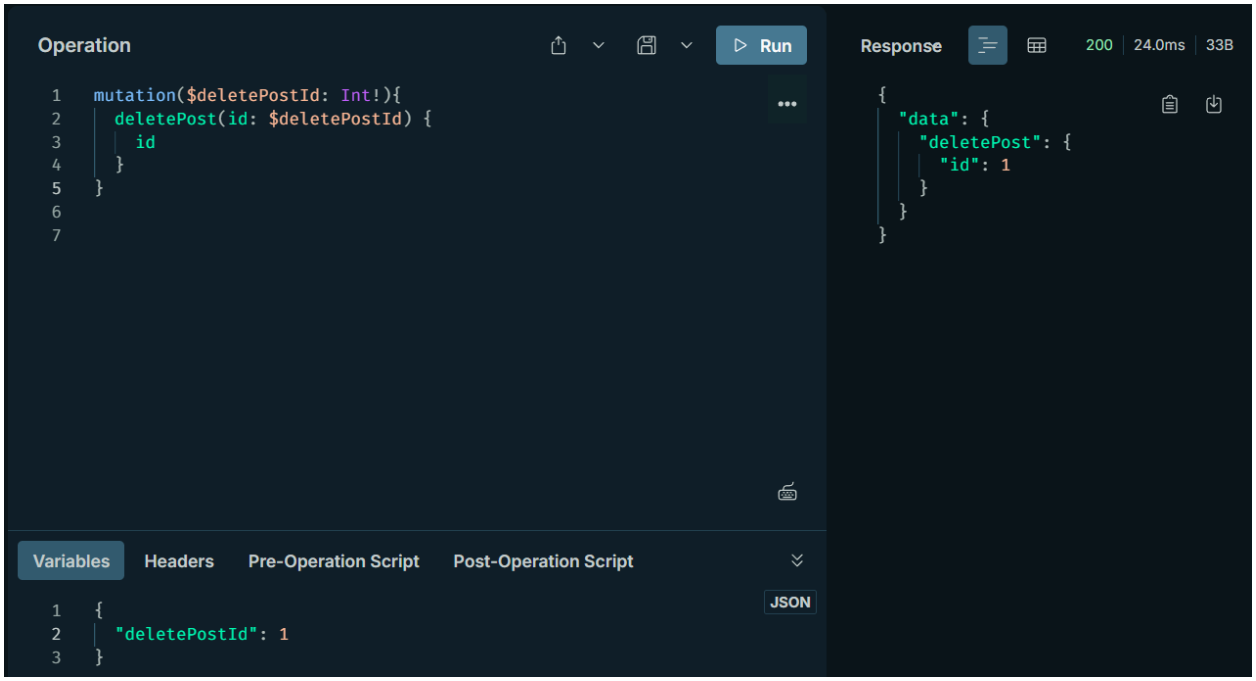
Read Example:



Update Example:

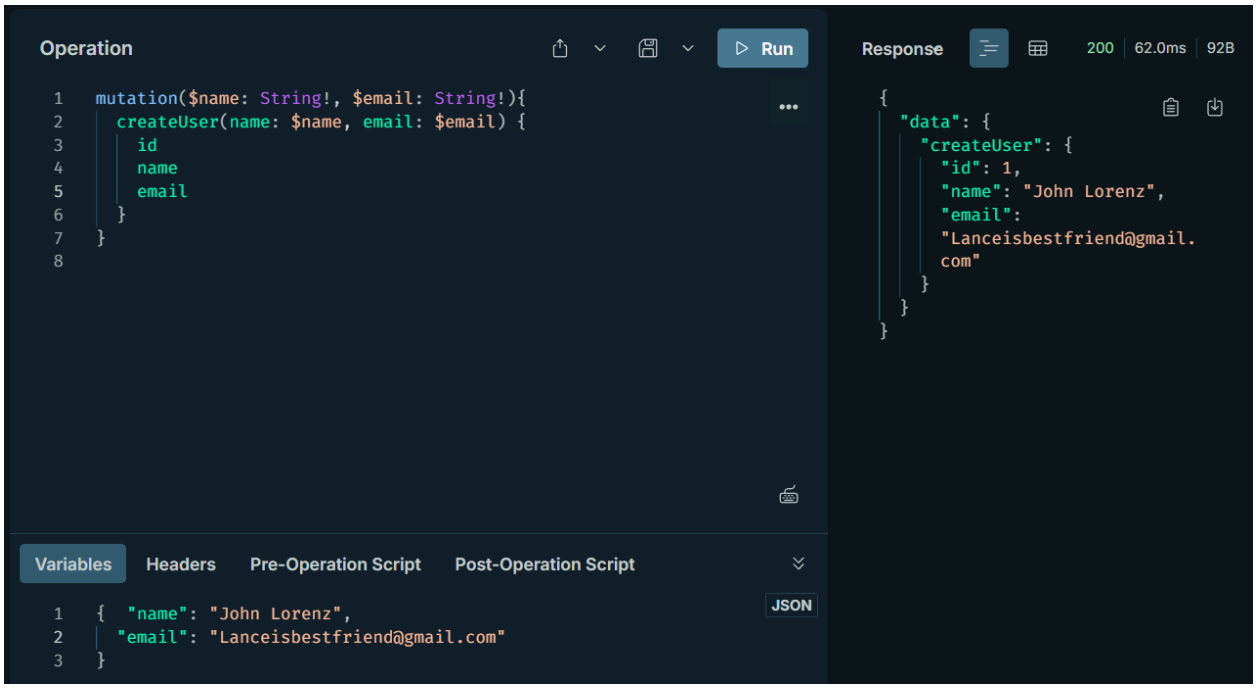


Delete Example:

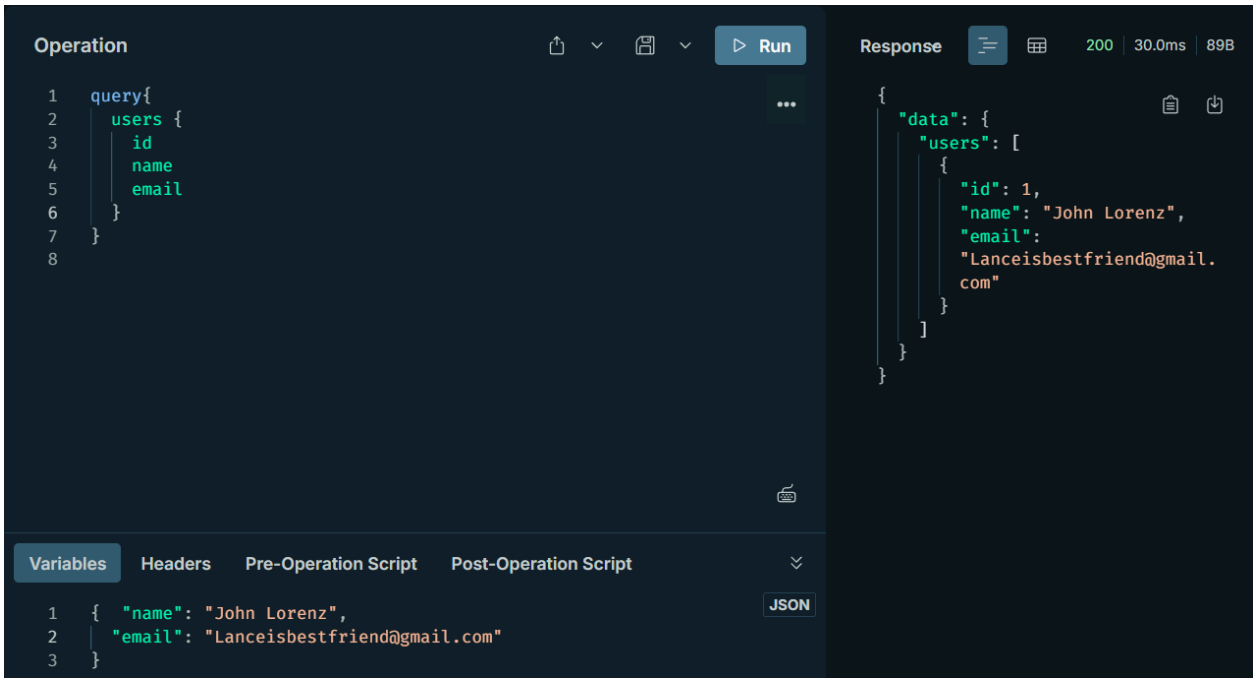


Users - Services

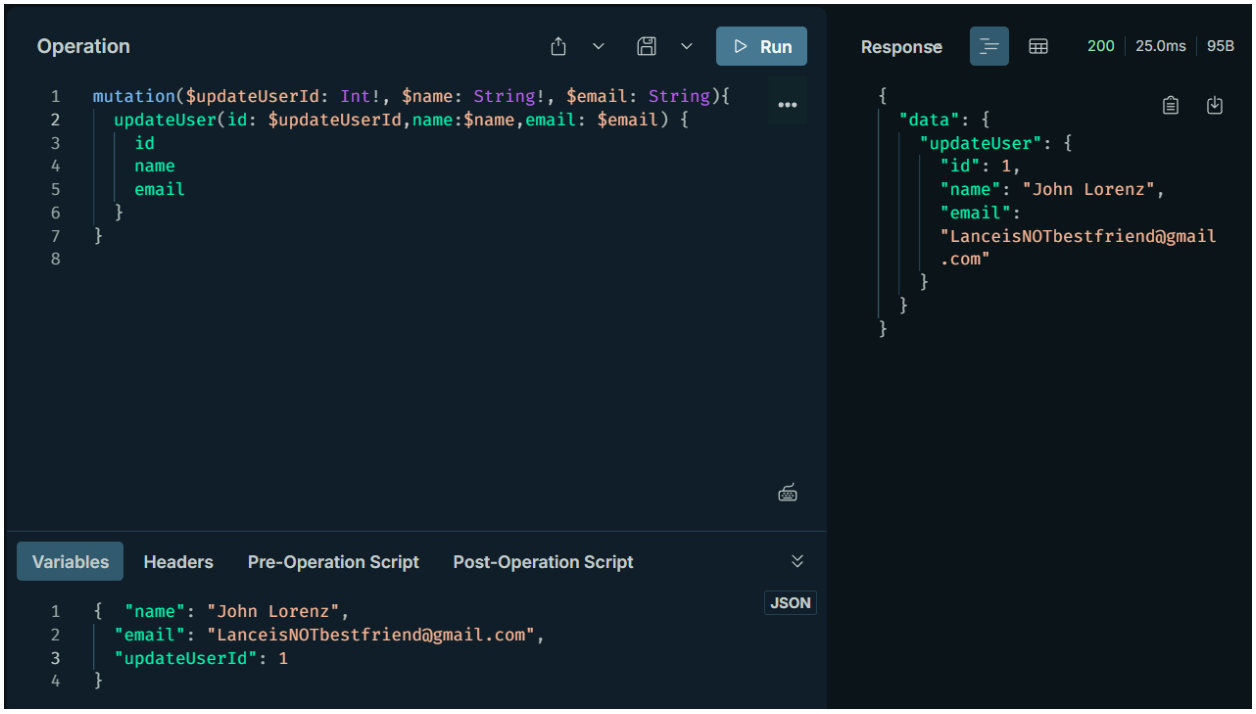
Create Users



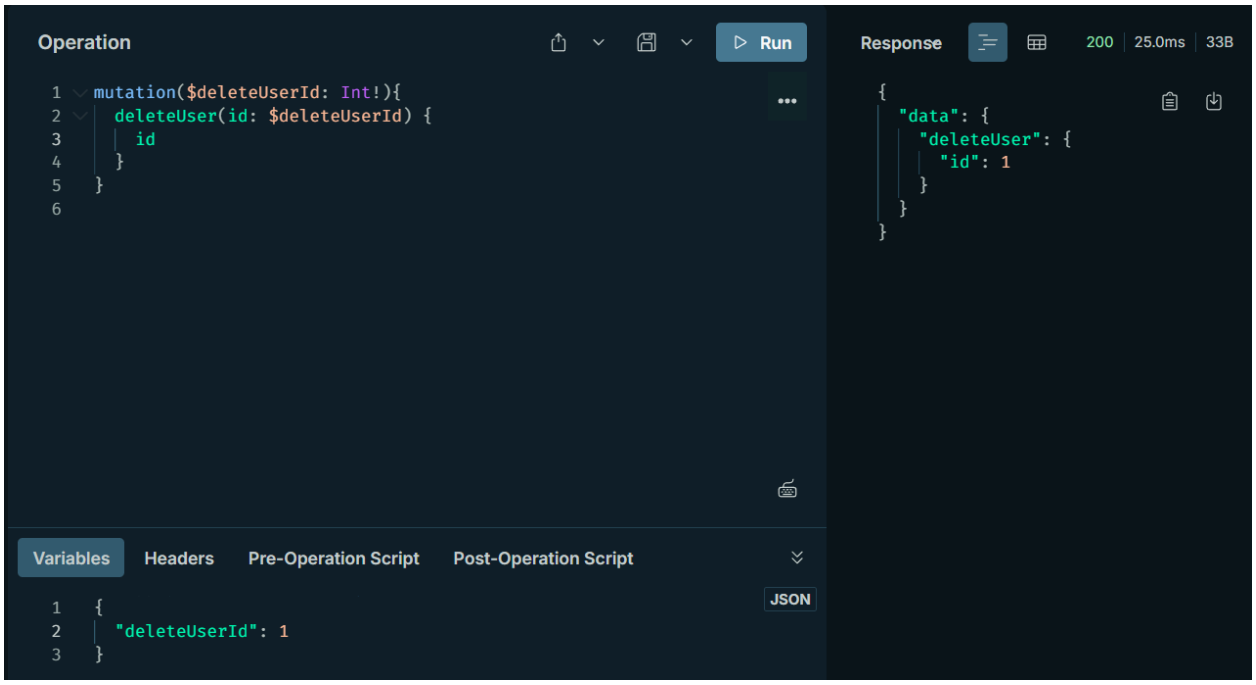
Read Users



Update Users



Delete Users



Reflection

This activity gave me some hands-on experience in building microservices with Node.js, Prisma, and Apollo Server. One of the main takeaways was really grasping the concept of database migrations and why they matter. Migrations are essential for tracking changes to the database schema, ensuring everything stays consistent across different environments, and making it easier to tweak the structure without risking data loss. Prisma's migration system made this process a breeze, allowing for smooth schema evolution.

Another key point was diving into GraphQL for CRUD operations. Unlike REST, which has set endpoints for each action, GraphQL offers a more flexible way to fetch data by letting clients request only the fields they need. This approach cuts down on both over-fetching and under-fetching, boosting efficiency. Setting up separate microservices for users and posts really highlighted the perks of a modular and scalable system architecture, where each service can work independently but still communicate when necessary.

All in all, this exercise really deepened my understanding of microservice architecture, database management, and the benefits of using GraphQL in today’s application development landscape.