

**Architecture of Application=>**

The user is making fetchthread request through port 8000

1. **GraphQL API Layer:**
   * **Queries and Mutations:**
     + The Queries handle fetching data, such as FetchThreads, with resolvers connecting to services or data sources.
     + The Mutations handle data modifications, such as creating or updating records, with their respective resolvers.
   * **GraphQL Schema: (**The layer before queries & mutation**)**  
     Defines the data structure (Thread, User) and relationships between entities, such as:
     + Thread: Contains fields like id, title, description, and user.
     + User: Contains fields like id, name, email, and profile.
2. **Service Layer:**
   * Divided into individual services such as:
     + **Post Service:** Manages thread-related logic.
     + **User Service:** Manages user-related operations.
   * Each service encapsulates business logic and acts as an intermediary between the GraphQL resolvers and the database.
3. **Database Layer:**
   * **Prisma ORM:**
     + Acts as the bridge between the application and the PostgreSQL database.
     + Simplifies database queries and manages schema migrations.
   * **PostgreSQL:**
     + Relational database used to store and manage data.
4. **Real-Time Communication:**
   * **Socket.IO:**  
     Likely used for real-time updates, such as notifying users of new threads or changes.
5. **External Integration:**
   * **REST API (gRPC):**
     + Represents an additional communication layer or service for integrating with external systems.
6. **Port and Endpoint:**
   * The GraphQL API is exposed on PORT:8000 with an endpoint /graphql.

**Data Flow:**

1. A client sends a GraphQL query (e.g., FetchThreads) to /graphql.
2. The query is resolved through the appropriate resolver, which invokes a service method (e.g., PostService.getAllThreads()).
3. The service interacts with Prisma to fetch or modify data in the PostgreSQL database.
4. The response is sent back to the client.

# Setting up Graphql Server | Complete GraphQL Series

Yarn init

-setup typescript envirnement

- create simple express server

- install apolo graphql server & setup graphql server

- setup docker => postgress, redis => write docker-compose.yml

Enter cmd “docker compose up”

This will pull postrgrss image

# Graphql Prisma Postgresql Setup | Complete GraphQL Series

In previos installed postgrss DB now we will connecting this postgress DB to our application using prisma

* Run docker <docker compose up>
* Connect postgress running in docker with prisma
* Create lib/db.ts file in this file => help to interact with DB
* Created one user in DB for testing

# Refactoring GraphQL Code | Complete GraphQL Series

* Created ‘user’ folder for user related code & same for ‘graphql’ folder graphql related code

# Authentication with GraphQL Server

Creating Authentication in graphQL it is different from normal’

# JWT Authentication and Context in GraphQL