<https://youtu.be/GK_6XrduAiA?si=oRFFI4P5AFdupOlO>

https://the-guild.dev/graphql/codegen/docs/guides/angular

https://graphql.org/  
<https://the-guild.dev/graphql/apollo-angular/docs>

https://hasura.io/docs/3.0/graphql-api/queries/nested-queries/#:~:text=You%20can%20use%20the%20object,a%20nested%20or%20related%20type.

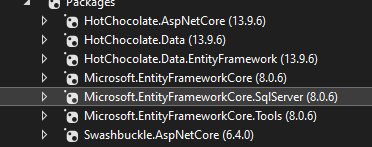
[Presentation.pptx](https://nupakistan-my.sharepoint.com/:p:/g/personal/f219192_cfd_nu_edu_pk/EfuR56CHDtVDvCUnY4GPKeIBMEr6nDVzIXLbJ7dhlKkf4A?e=ckvhCn)

https://preview.keenthemes.com/metronic8/angular/demo1/crafted/widgets/tables

`**Task<>`** in C# represents a unit of work that can be executed asynchronously. It's used to handle operations that may take time to complete, allowing other tasks to run concurrently.

**`ISourceStream`** is used to provide a stream of data that can be asynchronously consumed, making it ideal for real-time data updates in applications like GraphQL subscriptions.

Bogus package used to generate random data



**Apollo:**

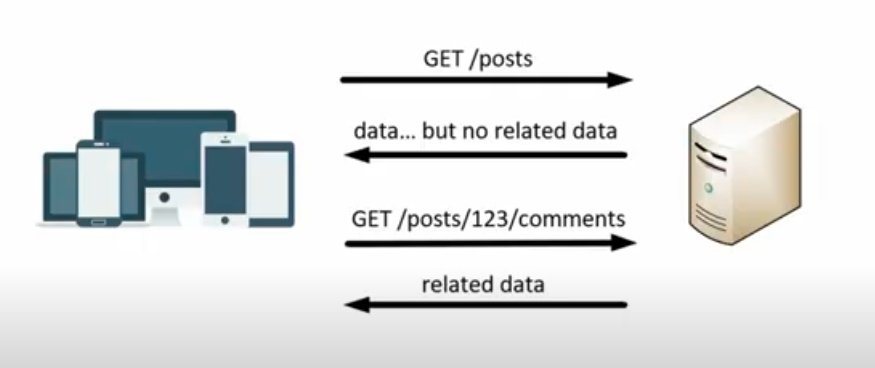
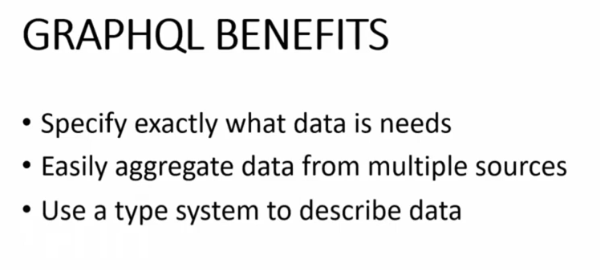
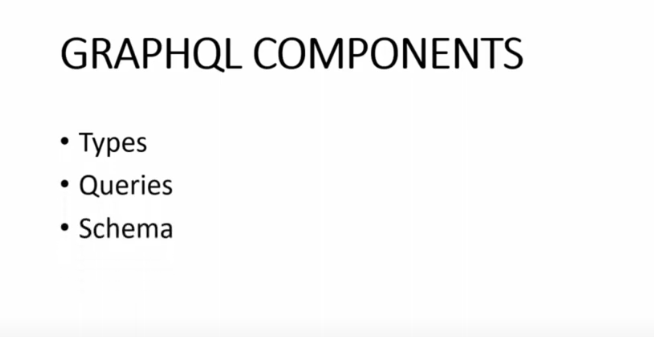
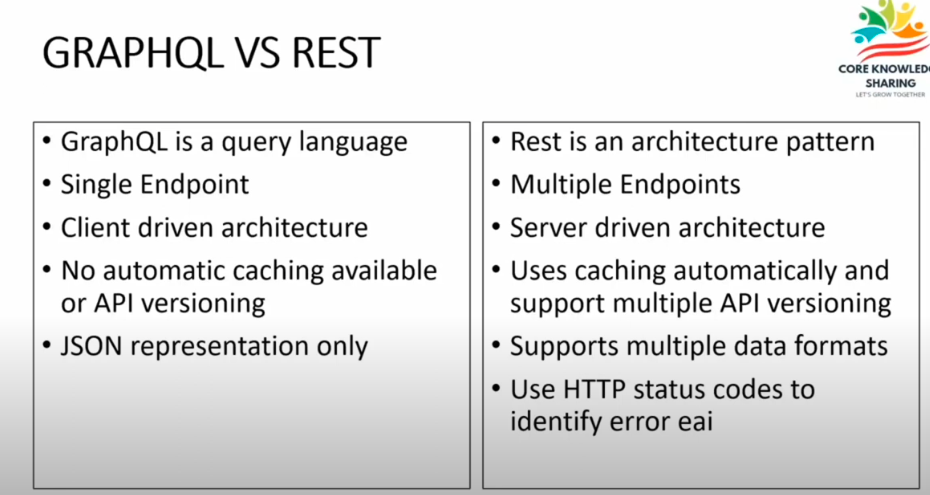
Apollo is a comprehensive platform for building, managing, and querying GraphQL APIs. It provides client and server libraries that help in integrating GraphQL into web and mobile applications. Apollo simplifies state management, data fetching, and caching, enhancing performance and developer productivity. It is widely used for its robust tooling and seamless integration with modern JavaScript frameworks.

**Hot Chocolate:**

Hot Chocolate is a .NET GraphQL server library that allows developers to create GraphQL endpoints in ASP.NET Core applications. It simplifies the implementation of GraphQL APIs by providing easy-to-use tools and features for schema definition, query execution, and server configuration. Using Hot Chocolate, developers can build efficient and type-safe GraphQL services, enhancing the API's flexibility and performance. It is popular for integrating GraphQL with .NET ecosystems.

**GraphQL in asp.net**

GraphQL is a query language for APIs that allows clients to request exactly the data they need, making it more efficient and flexible compared to REST. It provides a single endpoint for data retrieval and manipulation, enabling more dynamic and efficient interactions with APIs.

* Single Endpoint for exact data
* Benefits of the graphQL
* Components of the graphQL
* Diff rest api vs graphQl

**Steps to run graphql query:**

* Install nugget package of the hotchocolate.aspnetcore
* Write “launchURL”:true in appsetting.json
* Write the given below line in program.cs file:

builder.Services.AddGraphQL()

.AddQueryType<query>()

.AddMutationType<mutation>();

app.MapGraphQL()

**Query and Resolvers:**

**Query:** A query is a request for specific data from the server. It allows you to ask for exactly what you need and get predictable results.

**Resolver:** A resolver is a function that fetches the data for a query. When a query is made, the resolver determines how to fetch the data for each field in the query.

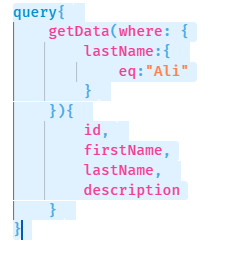
**Mutations:**

In GraphQL, a mutation is an operation that allows you to modify data on the server, such as creating, updating, or deleting records. It's like a query, but it changes data instead of just fetching it.

**Subscriptions:**

In GraphQL, a subscription allows clients to receive real-time updates from the server whenever specific events occur. This enables features like live data feeds and notifications.

**Filtering in GraphQL:**

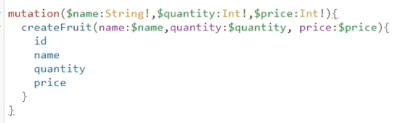
* **Where**
* **OR**A screenshot of a computer

  Description automatically generated
* **And**Same like or only and word change.
* **Not**

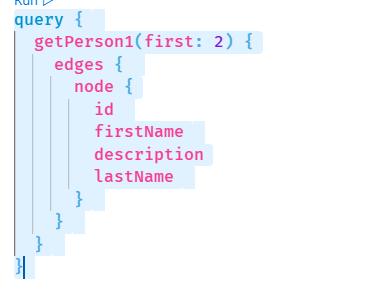
Apply not operator with every eq,contain,greater like this

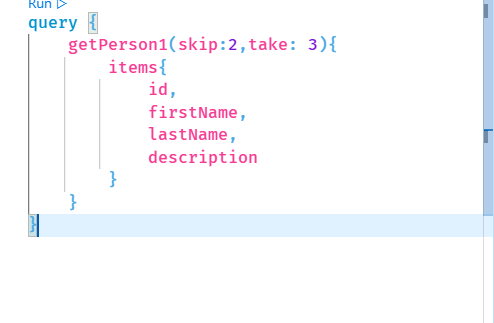


* **Variables**



* **Paging**





**Fragments:**

# Define a fragment for the common fields

fragment UserInfo on User {

id

name

email

}

# Use the fragment in a query

query GetUsers {

users {

...UserInfo

profilePicture

}

}

**Introspection:**

 \_\_schema: Fetches information about the entire schema.

 types: Retrieves all types in the schema.

 name: Gets the name of each type.

 fields: Fetches all fields within each type.

 name: Gets the name of each field.

**Entity Framework in graphQL:**

**Nugget Packages:**

hotChocolate

HotChocolate.data.EntityFramework

**Basic Concepts:**

* **Schema:** Defines the data structure and operations available in the GraphQL API.
* **Types:** Represent different kinds of data (scalar types like Int, String, object types for complex data structures, enums for predefined options).
* **Queries:** Requests to retrieve data from the API.
* **Mutations:** Requests to modify data in the API (create, update, delete).
* **Resolvers:** Functions that handle incoming queries and mutations, fetching data from your data sources (databases, APIs).
* **Arguments:** Optional parameters passed to queries and mutations to filter or control the returned data.
* **Selections:** Specifying the specific fields you want to retrieve within a query.
* **Fragments:** Reusable pieces of GraphQL queries to avoid repetition.
* **Variables:** Allow using dynamic values within queries and mutations.

**Intermediate Concepts:**

* **Directives:** Instructions to modify the behavior of queries and mutations (e.g., @deprecated to mark a field as deprecated).
* **Subscriptions:** Enable real-time data updates through WebSockets.
* **Introspection:** Ability to query the GraphQL schema itself to discover available types and operations.
* **Validation:** Ensuring queries and mutations adhere to the defined schema.
* **Caching:** Optimizing performance by caching frequently requested data.
* **Authorization:** Controlling access to specific data or operations based on user permissions.

**Advanced Concepts:**

* **Federation:** Building GraphQL APIs out of multiple, independent services that can be stitched together.
* **Relay:** A popular pattern for managing data fetching and pagination in GraphQL applications.
* **Persisted Queries:** Caching GraphQL queries on the client-side for performance optimization.
* **GraphQL Extensions:** Allow developers to extend the GraphQL protocol for custom functionality.
* **Schema Stitching:** Combining multiple GraphQL schemas into a single, unified schema.
* **GraphQL Tools:** Various tools for development, testing, and introspection of GraphQL APIs.