

Backend Coding Challenge

Solytic

Submitted by:

Nouman Ullah Qureshi

iamnouman95@gmail.com

Pre-requisites:

To compile and run the code provided following mandatory checklist should be installed first:

- .NET Core 5.0.
- GraphQL (HotChocolate Playground).
- EntityFramework Core.
- Visual Studio 2019.
- Microsoft SQL Server.
- Google Chrome or Firefox Web Browser.



HotChocolate.AspNetCore by ChilliCream authors and contributors

v11.3.5

This package contains the GraphQL ASP.NET Core middleware for Hot Chocolate. Moreover, this package includes the Banana Cake Pop middleware, which provides you with our beloved GraphQL IDE middleware.



HotChocolate.AspNetCore.Playground by ChilliCream authors and contributors

v10.5.5

Contains a GraphQL Playground for ASP .Net core that can be used with the Hot Chocolate GraphQL server.



HotChocolate.Data by ChilliCream authors and contributors

v11.3.5

Contains ready to use extensions for data management in HotChocolate. This includes filtering, projections and sorting



Microsoft.EntityFrameworkCore by Microsoft

v5.0.9

Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other...



Microsoft.EntityFrameworkCore.SqlServer by Microsoft

v5.0.9

Microsoft SQL Server database provider for Entity Framework Core.



Microsoft.EntityFrameworkCore.Tools by Microsoft

v5.0.9

Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

Architecture:

The Web Application architecture is built on the following mentioned tech-stack:

- GraphQL WebAPI -> C# .NET Core 5.0.
- Front end -> HotChocolate Playground.
- Server-side -> C#.
- Unit Tests -> xUnit

GraphQL API Configuration:

In order to connect to database and use the GraphQL WebAPI, first open appsettings.json and configure the database connection string. The default is at root local (.) and database name is 'GraphQL'.

Adjust/Update it as your need as the EntityFramework Core will create a database & table 'Customer' in the database.

Afterwards, open Package Manager Console in VS and add migrations from the following commands:

- Add-Migration mig1
- update-database

```
appsettings.json  + X
Schema: https://json.schemastore.org/appsettings.json
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft": "Warning",
6        "Microsoft.Hosting.Lifetime": "Information"
7      }
8    },
9    "AllowedHosts": "*",
10   "ConnectionStrings": {
11     "myconn": "server=.; database=GraphQL;Trusted_Connection=True;"
12   }
13 }
14
```

Sql Server Database schema:

Databases

System Databases

Database Snapshots

GraphQL

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.__EFMigrationsHistory

dbo.Customer

Columns

Id (PK, int, not null)

Email (nvarchar(128), null)

Name (nvarchar(128), null)

Code (int, null)

Status (int, not null)

CreatedAt (datetime2(7), not null)

IsBlocked (bit, not null)

Keys

Constraints

Triggers

Indexes

Statistics

select * from Customer

100 %

Results Messages

	Id	Email	Name	Code	Status	CreatedAt	IsBlocked
1	1	nouman@gmail.com	Nouman Ullah Qureshi	NULL	1	2021-08-26 21:55:54.4785628	0
2	3	NULL	New Customer3	NULL	1	2020-08-16 22:00:00.0000000	0
3	5	new@gmail.com	New Customer1	NULL	1	2020-08-17 22:00:00.0000000	0
4	41	dummy@gmail.com	Dummy	NULL	1	2021-08-27 15:49:33.3357487	0
5	44	new@gmail.com	New Customer5	NULL	1	2021-08-27 16:26:36.4290380	0

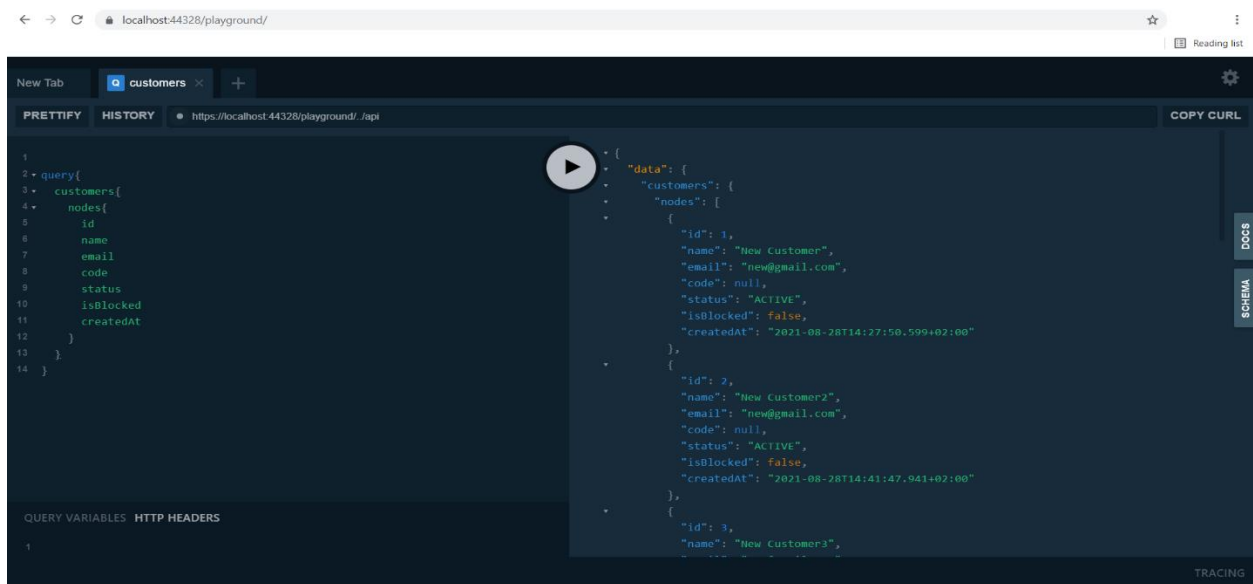
WebApi Usage:

There are CRUD operations available in the WebApi to access namely:

- GetAll
- Create
- Update
- Delete

Syntax:

GetAll (Pagination added, CustomerConnection node will be fetched):

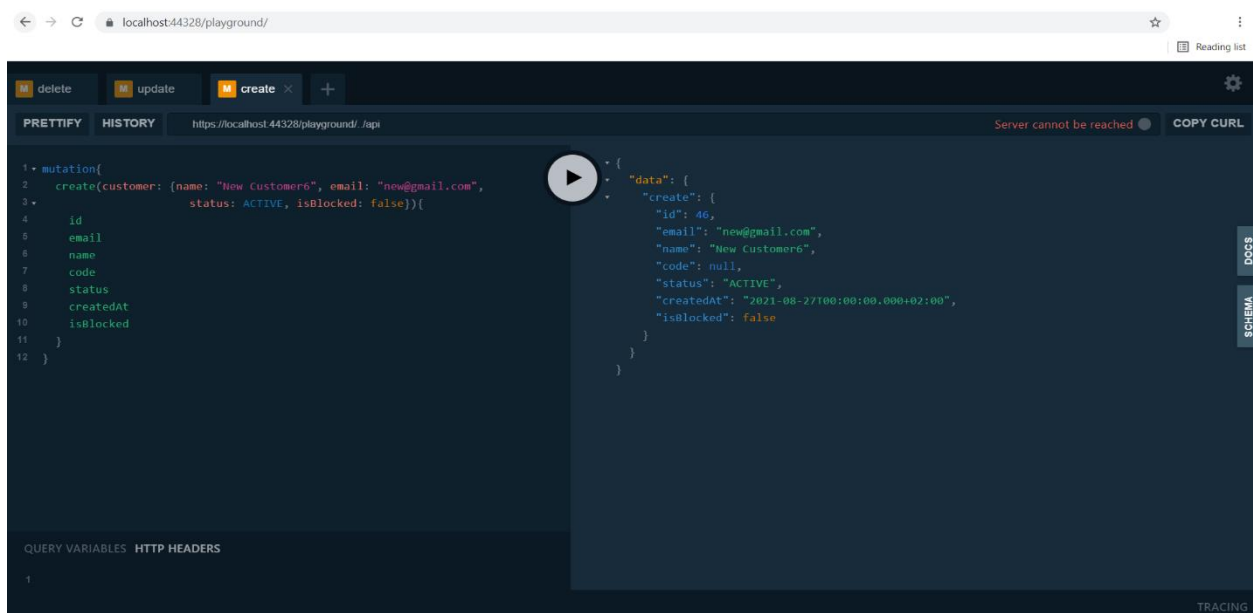


The screenshot shows the GraphQL Playground interface. The query editor on the left contains a query to fetch a list of customers. The response on the right shows a JSON object with a 'data' field containing a list of customer nodes. The interface includes tabs for 'PRETTIFY', 'HISTORY', and 'COPY CURL'. The URL bar shows 'localhost:44328/playground/'.

```
1 query {
2   customers {
3     nodes {
4       id
5       name
6       email
7       code
8       status
9       isBlocked
10      createdAt
11    }
12  }
13 }
```

```
{
  "data": {
    "customers": {
      "nodes": [
        {
          "id": 1,
          "name": "New Customer",
          "email": "new@gmail.com",
          "code": null,
          "status": "ACTIVE",
          "isBlocked": false,
          "createdAt": "2021-08-28T14:27:50.599+02:00"
        },
        {
          "id": 2,
          "name": "New Customer2",
          "email": "new@gmail.com",
          "code": null,
          "status": "ACTIVE",
          "isBlocked": false,
          "createdAt": "2021-08-28T14:41:47.941+02:00"
        },
        {
          "id": 3,
          "name": "New Customer3",
          "email": "new@gmail.com",
          "code": null,
          "status": "ACTIVE",
          "isBlocked": false,
          "createdAt": "2021-08-28T14:41:47.941+02:00"
        }
      ]
    }
  }
}
```

Create:



The screenshot shows the GraphQL Playground interface. The query editor on the left contains a mutation query to create a new customer. The response on the right shows an error message: 'Server cannot be reached'. The interface includes tabs for 'PRETTIFY', 'HISTORY', and 'COPY CURL'. The URL bar shows 'localhost:44328/playground/'.

```
1 mutation {
2   create(customer: {name: "New Customer6", email: "new@gmail.com",
3     status: ACTIVE, isBlocked: false}) {
4     id
5     email
6     name
7     code
8     status
9     createdAt
10    isBlocked
11  }
12 }
```

```
{
  "data": {
    "create": {
      "id": 46,
      "email": "new@gmail.com",
      "name": "New Customer6",
      "code": null,
      "status": "ACTIVE",
      "createdAt": "2021-08-27T00:00:00.000+02:00",
      "isBlocked": false
    }
  }
}
```

Server cannot be reached

Update:

The screenshot shows the GraphQL Playground interface in a web browser. The address bar displays `localhost:44328/playground/`. The interface includes tabs for `delete`, `update`, and `update` (selected). The URL bar shows `https://localhost:44328/playground/ .api`. The main editor contains the following GraphQL query:

```
1
2 mutation{
3   update(customer: {id:46, name: "New Customer7"}){
4     id
5     email
6     name
7     code
8     status
9     createdAt
10    isBlocked
11  }
12 }
```

The right-hand pane displays the JSON response:

```
{
  "data": {
    "update": {
      "id": 46,
      "email": "new@gmail.com",
      "name": "New Customer7",
      "code": null,
      "status": "ACTIVE",
      "createdAt": "2021-08-27T00:00:00.000+02:00",
      "isBlocked": false
    }
  }
}
```

At the bottom, there are tabs for `QUERY VARIABLES` and `HTTP HEADERS`, and a `TRACING` button in the bottom right corner.

Delete:

The screenshot shows the GraphQL Playground interface in a web browser. The address bar displays `localhost:44328/playground/`. The interface includes tabs for `delete` and `delete` (selected). The URL bar shows `https://localhost:44328/playground/ .api`. The main editor contains the following GraphQL query:

```
1 mutation{
2   delete(deleteVM: {id: 44})
3 }
```

The right-hand pane displays the JSON response:

```
{
  "data": {
    "delete": true
  }
}
```

At the bottom, there are tabs for `QUERY VARIABLES` and `HTTP HEADERS`, and a `TRACING` button in the bottom right corner.

Filter & Sorting(HotChocolate.Data functions also added use where, order etc to fetch data):

The screenshot shows the GraphQL Playground interface. The query editor on the left contains the following query:

```
1 query{
2   customers(where: {name: {eq: "New Customer9"}}){
3     nodes{
4       id
5       name
6       email
7       code
8       status
9       isBlocked
10      createdAt
11    }
12  }
13 }
14 }
```

The JSON response is displayed on the right:

```
{
  "data": {
    "customers": {
      "nodes": [
        {
          "id": 32,
          "name": "New Customer9",
          "email": "new@gmail.com",
          "code": null,
          "status": "ACTIVE",
          "isBlocked": false,
          "createdAt": "2021-08-28T14:41:53.684+02:00"
        }
      ]
    }
  }
}
```

The interface includes tabs for PRETTIFY, HISTORY, and COPY CURL. The URL bar shows localhost:44328/playground/. The bottom status bar indicates TRACING.

The screenshot shows the GraphQL Playground interface. The query editor on the left contains the following query:

```
1 query{
2   customers(where: {name: {endsWith: "3"}}, order: {id: DESC}){
3     nodes{
4       id
5       name
6       email
7       code
8       status
9       isBlocked
10      createdAt
11    }
12  }
13 }
14 }
```

The JSON response is displayed on the right:

```
{
  "data": {
    "customers": {
      "nodes": [
        {
          "id": 46,
          "name": "New Customer23",
          "email": "new@gmail.com",
          "code": null,
          "status": "ACTIVE",
          "isBlocked": false,
          "createdAt": "2021-08-28T14:41:53.684+02:00"
        },
        {
          "id": 36,
          "name": "New Customer13",
          "email": "new@gmail.com",
          "code": null,
          "status": "ACTIVE",
          "isBlocked": false,
          "createdAt": "2021-08-28T14:41:53.684+02:00"
        },
        {
          "id": 3,
          "name": "New Customer3",
          "email": "new@gmail.com",
          "code": null,
          "status": "ACTIVE",
          "isBlocked": false,
          "createdAt": "2021-08-28T14:41:53.684+02:00"
        }
      ]
    }
  }
}
```

The interface includes tabs for PRETTIFY, HISTORY, and COPY CURL. The URL bar shows localhost:44328/playground/. The bottom status bar indicates TRACING.