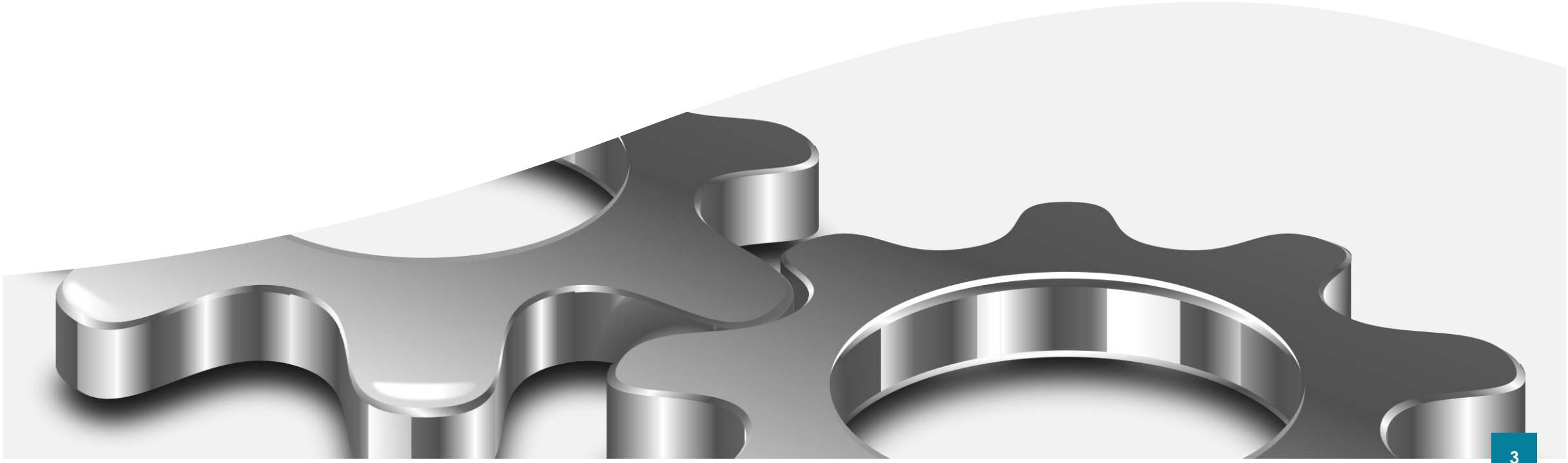
Golang Programming

Building GraphQL Web Services with Go

Where to Find The Code and Materials?

https://github.com/iproduct/coursego

GraphQL



GraphQL – Brief History

- GraphQL is an open-source data query and manipulation language for APIs, and a runtime for fulfilling queries with existing data.
- GraphQL was developed internally by Facebook in 2012 before being publicly released in 2015
- On 7 November 2018, the GraphQL project was moved from Facebook to the newly-established GraphQL Foundation, hosted by the non-profit Linux Foundation.
- Since 2012, GraphQL's rise has followed the adoption timeline as set out by Lee Byron, GraphQL's creator, with accuracy. Byron's goal is to make GraphQL omnipresent across web platforms.

Source: https://en.wikipedia.org/wiki/GraphQL

GraphQL

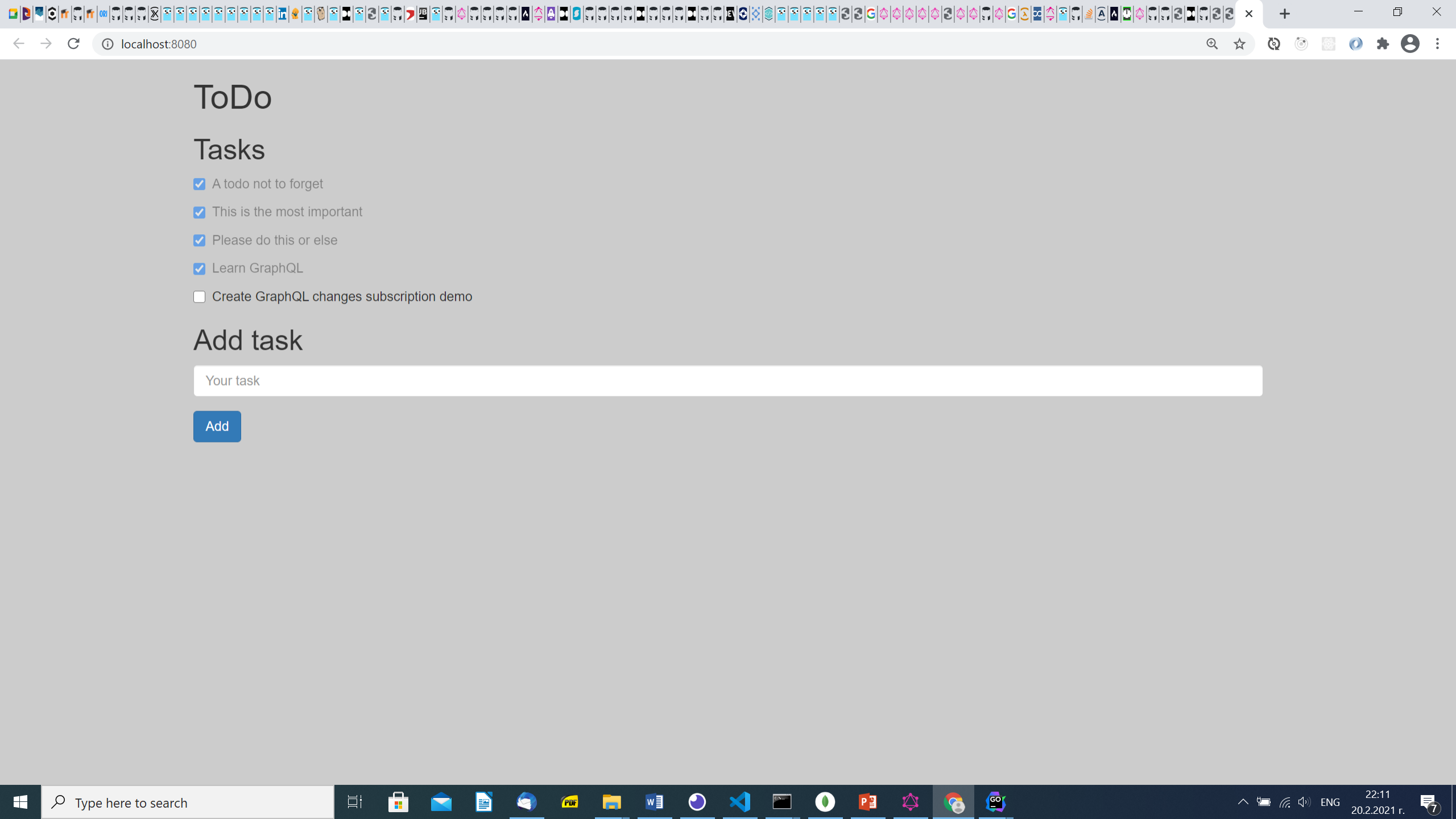
- Similarly to REST and gRPC it allows development of web service APIs
- Clients can define the structure of the data required, and the same structure of the data is returned from the server, therefore preventing excessively large amounts of data from being returned, but this has implications for how effective web caching of query results can be.
- It consists of a type system, query language and execution semantics, static validation, and type introspection.
- GraphQL supports reading, writing (mutating), and subscribing to changes to data (realtime updates)
- GraphQL servers are available for multiple languages, including Haskell, JavaScript, Perl Python, Ruby, Java, C++, C#, Scala, Go, Rust, Elixir, Erlang, PHP, R, and Clojure.

GraphiQL Client

- Download GraphiQL Client at: https://github.com/skevy/graphiql-app/releases
- Examples - query and mutation:

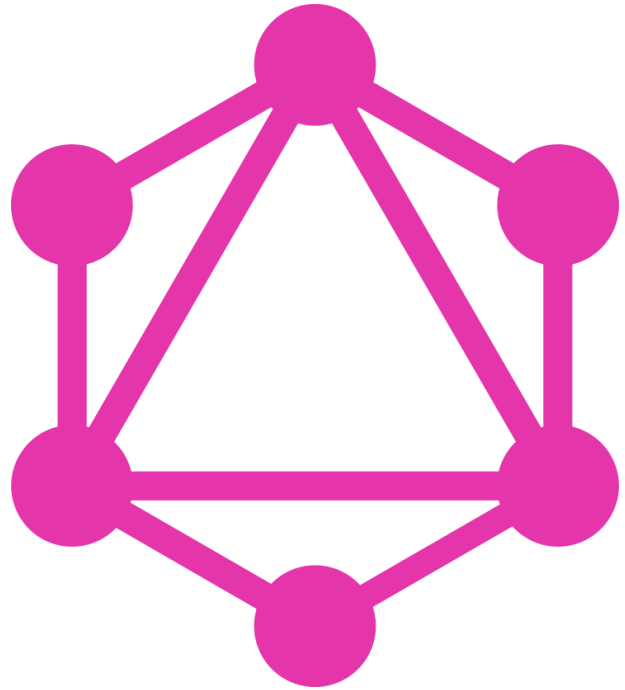
```
query all_todos{
    todoList {
        id
        text
        done
    }
}
```

```
mutation create_todo {
    createTodo(text: "My new todo") {
        id
        text
        done
    }
}
```


localhost:8080
ToDo
Tasks
A todo not to forget
This is the most important
Please do this or else
Learn GraphQL
Create GraphQL changes subscription demo
Add task
Your task
Add
Type here to search
ENG
22:11
20.2.2021 г.

Learning GraphQL - https://graphql.org/learn/

- Queries and Mutations – Fields, Arguments, Aliases, Fragments, Operation Name, Variables, Directives, Mutations, Inline Fragments
- Schemas and Types - Type System, Type Language, Object Types and Fields, Arguments, The Query and Mutation Types, Scalar Types, Enumeration Types, Lists and Non-Null, Interfaces, Union Types, Input Types
- Validation
- Execution
- Introspection



GraphiQL

File View Help

Untitled Query 1

GraphQL Endpoint http://localhost:8080/graphql

Method GET

Edit HTTP Headers

GraphiQL

Prettify History

```
query Luke {
  human(id: "1000") {
    name
    # Queries can have comments!
    friends {
      id
      name
      appearsIn
    }
  }
}

{
  leftComparison: hero(episode: EMPIRE) {
    ...comparisonFields
  }
  rightComparison: hero(episode: JEDI) {
    ...comparisonFields
  }
}
fragment comparisonFields on Character {
  name
  appearsIn
  friends {
    id
    name
  }
}
{
  empireHero: hero(episode: EMPIRE) {
    name
  }
  jediHero: hero(episode: JEDI) {
    name
  }
}

query Hero($episode: Episode!, $withFriends: Boolean = false) {
  hero(episode: $episode) {
    name
    friends @include(if: $withFriends) {
      name
    }
  }
}
```

QUERY VARIABLES


```
{
  "data": {
    "leftComparison": {
      "appearsIn": [
        "NEWHOPE",
        "EMPIRE",
        "JEDI"
      ],
      "friends": [
        {
          "id": "1002",
          "name": "Han Solo"
        },
        {
          "id": "1003",
          "name": "Leia Organa"
        },
        {
          "id": "2000",
          "name": "C-3PO"
        },
        {
          "id": "2001",
          "name": "R2-D2"
        }
      ],
      "name": "Luke Skywalker"
    },
    "rightComparison": {
      "appearsIn": [
        "NEWHOPE",
        "EMPIRE",
        "JEDI"
      ],
      "friends": [
        {
          "id": "1000",
          "name": "Luke Skywalker"
        },
        {
          "id": "1002",
          "name": "Han Solo"
        },
        {
          "id": "1003",
```


Schema **Query**

Search Query...

No Description

FIELDS

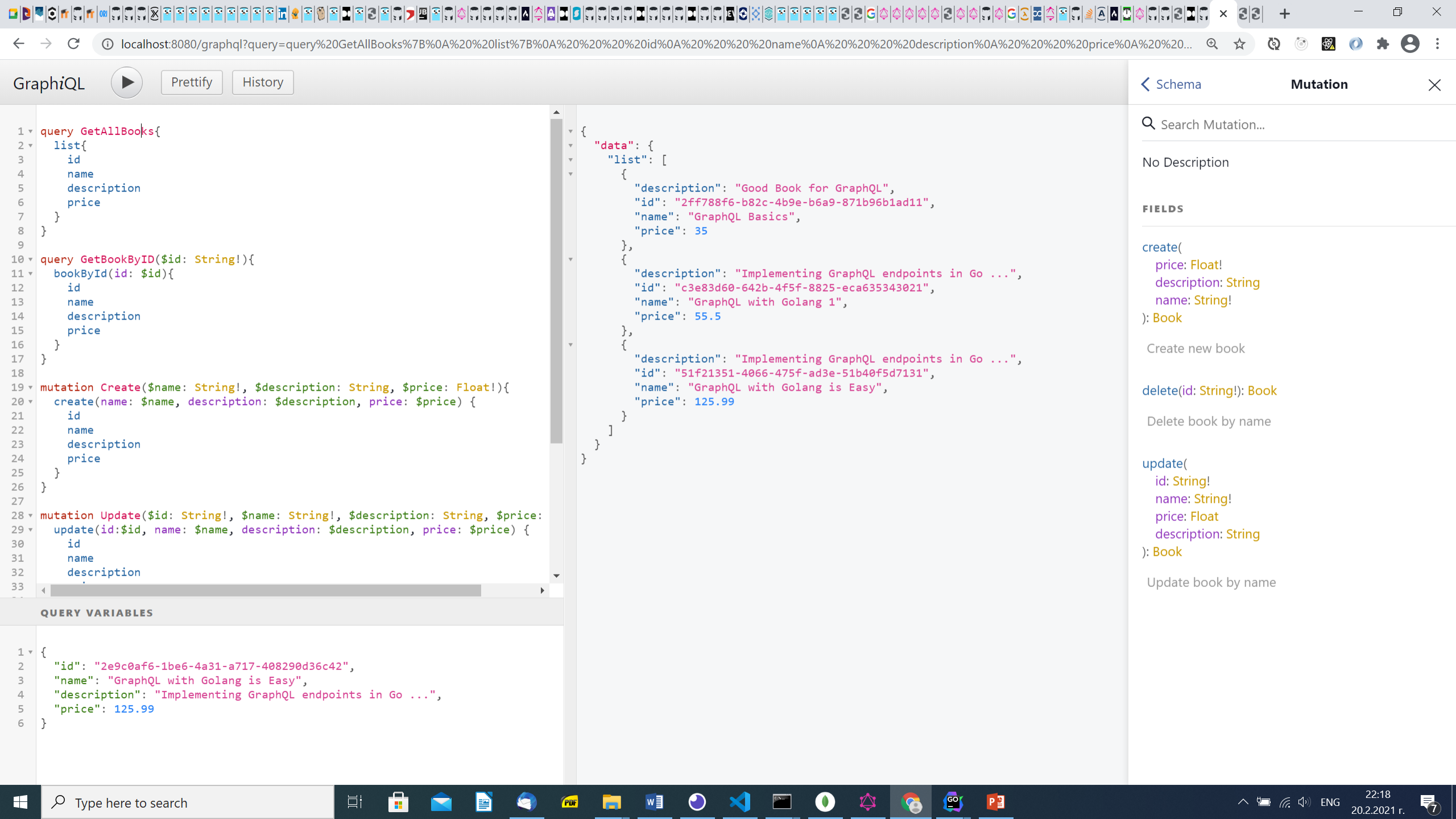
droid(id: String!): Droid

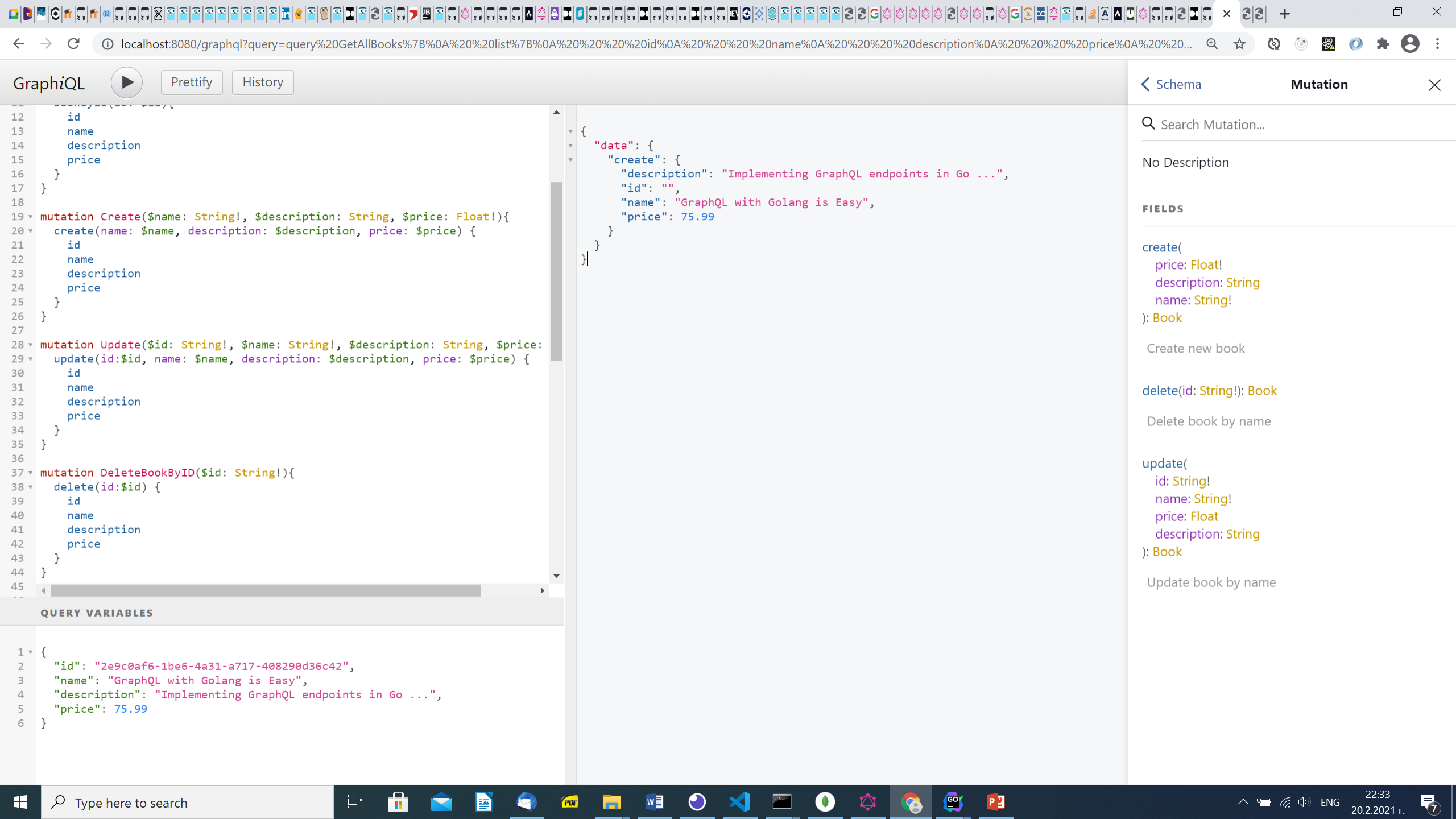
hero(episode: Episode): Character

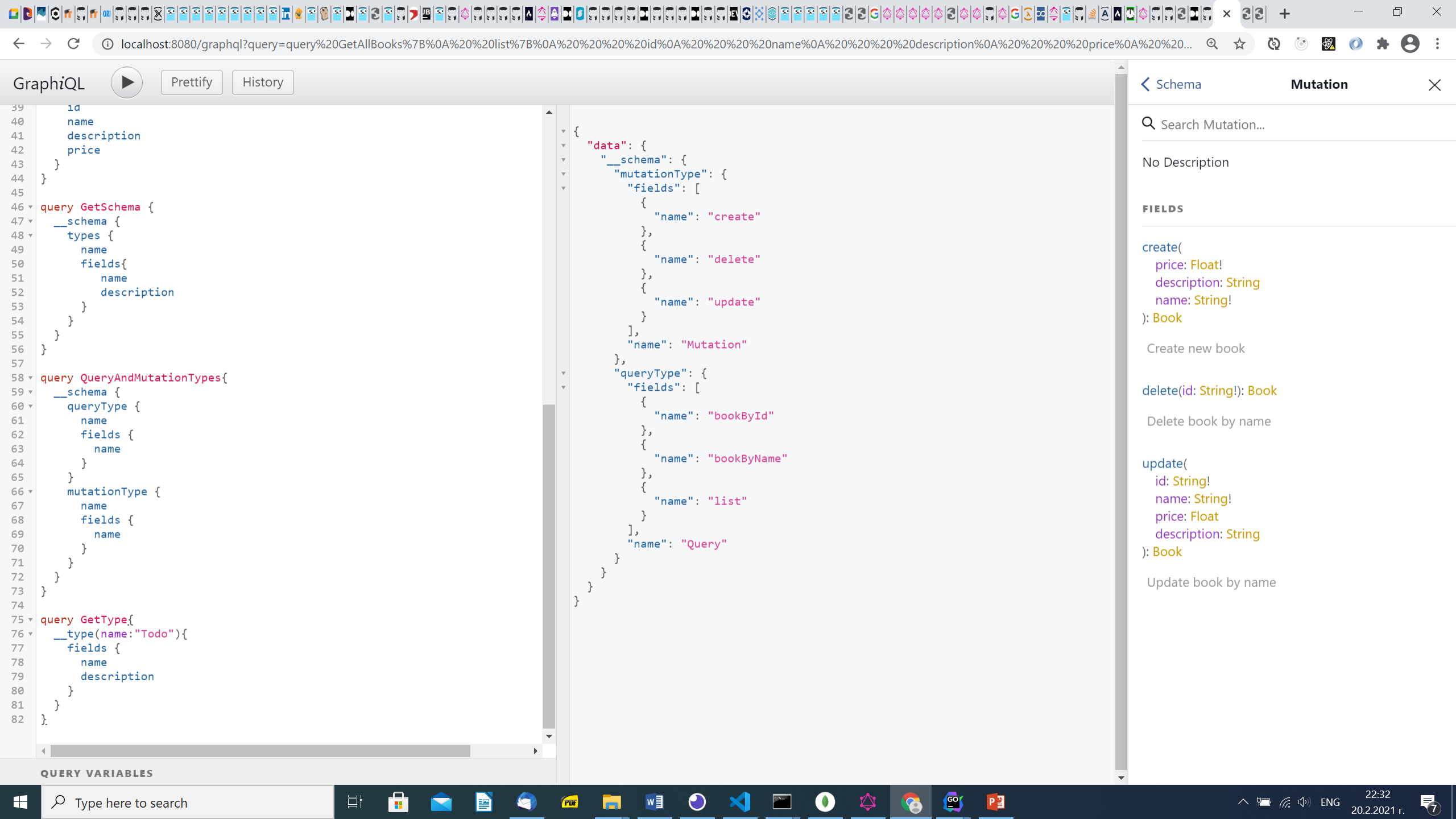
human(id: String!): Human

Type here to search

ENG 23:05 20.2.2021 г.


localhost:8080/graphql?query=query%20GetAllBooks%7B%0A%20%20list%7B%0A%20%20%20%20id%0A%20%20%20%20name%0A%20%20%20%20description%0A%20%20%20%20price%0A%20%20...
GraphiQL
Prettify
History
query GetAllBooks{
list{
id
name
description
price
}
}
query GetBookByID($id: String!){
bookById(id: $id){
id
name
description
price
}
}
mutation Create($name: String!, $description: String, $price: Float!){
create(name: $name, description: $description, price: $price) {
id
name
description
price
}
}
mutation Update($id: String!, $name: String!, $description: String, $price:
update(id:$id, name: $name, description: $description, price: $price) {
id
name
description
{
"data": {
"list": [
{
"description": "Good Book for GraphQL",
"id": "2ff788f6-b82c-4b9e-b6a9-871b96b1ad11",
"name": "GraphQL Basics",
"price": 35
},
{
"description": "Implementing GraphQL endpoints in Go ...",
"id": "c3e83d60-642b-4f5f-8825-eca635343021",
"name": "GraphQL with Golang 1",
"price": 55.5
},
{
"description": "Implementing GraphQL endpoints in Go ...",
"id": "51f21351-4066-475f-ad3e-51b40f5d7131",
"name": "GraphQL with Golang is Easy",
"price": 125.99
}
]
}
}
QUERY VARIABLES
{
"id": "2e9c0af6-1be6-4a31-a717-408290d36c42",
"name": "GraphQL with Golang is Easy",
"description": "Implementing GraphQL endpoints in Go ...",
"price": 125.99
}
Schema
Mutation
Search Mutation...
No Description
FIELDS
create(
price: Float!
description: String
name: String!
): Book
Create new book
delete(id: String!): Book
Delete book by name
update(
id: String!
name: String!
price: Float
description: String
): Book
Update book by name
Type here to search
ENG
22:18
20.2.2021 г.


localhost:8080/graphql?query=query%20GetAllBooks%7B%0A%20%20list%7B%0A%20%20%20%20id%0A%20%20%20%20name%0A%20%20%20%20description%0A%20%20%20%20price%0A%20%20...
GraphiQL
Prettify
History
id
name
description
price
}
}
mutation Create($name: String!, $description: String, $price: Float!){
create(name: $name, description: $description, price: $price) {
id
name
description
price
}
}
mutation Update($id: String!, $name: String!, $description: String, $price:
update(id:$id, name: $name, description: $description, price: $price) {
id
name
description
price
}
}
mutation DeleteBookByID($id: String!){
delete(id:$id) {
id
name
description
price
}
}
QUERY VARIABLES
{
"id": "2e9c0af6-1be6-4a31-a717-408290d36c42",
"name": "GraphQL with Golang is Easy",
"description": "Implementing GraphQL endpoints in Go ...",
"price": 75.99
}
{
"data": {
"create": {
"description": "Implementing GraphQL endpoints in Go ...",
"id": "",
"name": "GraphQL with Golang is Easy",
"price": 75.99
}
}
}
Schema
Mutation
Search Mutation...
No Description
FIELDS
create(
price: Float!
description: String
name: String!
): Book
Create new book
delete(id: String!): Book
Delete book by name
update(
id: String!
name: String!
price: Float
description: String
): Book
Update book by name
Type here to search
ENG
22:33
20.2.2021 г.


localhost:8080/graphql?query=query%20GetAllBooks%7B%0A%20%20list%7B%0A%20%20%20%20id%0A%20%20%20%20name%0A%20%20%20%20description%0A%20%20%20%20price%0A%20%20...
GraphiQL
Prettify
History
id
name
description
price
}
}
query GetSchema {
__schema {
types {
name
fields{
name
description
}
}
}
}
query QueryAndMutationTypes{
__schema {
queryType {
name
fields {
name
}
}
mutationType {
name
fields {
name
}
}
}
}
query GetType{
__type(name:"Todo"){
fields {
name
description
}
}
}
{
"data": {
"__schema": {
"mutationType": {
"fields": [
{
"name": "create"
},
{
"name": "delete"
},
{
"name": "update"
}
],
"name": "Mutation"
},
"queryType": {
"fields": [
{
"name": "bookById"
},
{
"name": "bookByName"
},
{
"name": "list"
}
],
"name": "Query"
}
}
}
}
QUERY VARIABLES
Schema
Mutation
Search Mutation...
No Description
FIELDS
create(
price: Float!
description: String
name: String!
): Book
Create new book
delete(id: String!): Book
Delete book by name
update(
id: String!
name: String!
price: Float
description: String
): Book
Update book by name
Type here to search
ENG
22:32
20.2.2021 г.

GraphQL in Go

GraphQL Starter Example

```
import ("net/http“; "github.com/graphql-go/graphql“; gqhandler "github.com/graphql-go/graphql-go-handler")
var queryType = graphql.NewObject(graphql.ObjectConfig {      // ... })
var Schema, _ = graphql.NewSchema(graphql.SchemaConfig{   // ... })

func main() {
        // create a graphl-go HTTP handler with our previously defined schema
        // and we also set it to return pretty JSON output
        h := gqhandler.New(&gqhandler.Config{
                Schema: &Schema,
                Pretty: true,
        })

        // serve a GraphQL endpoint at `/graphql`
        http.Handle("/graphql", h)

        // and serve!
        http.ListenAndServe(":8080", nil)
}
```

GraphQL Schema Configuration – Code First

```
var queryType = graphql.NewObject(graphql.ObjectConfig{
        Name: "RootQuery",
        Fields: graphql.Fields{
                "latestPost": &graphql.Field{
                        Type: graphql.String,
                        Resolve: func(p graphql.ResolveParams) (interface{}, error) {
                                return "Hello World!", nil
                        },
                },
                "postsCount": &graphql.Field {
                        Type: graphql.Int,
                        Resolve: func(p graphql.ResolveParams) (interface{}, error) {
                                return rand.Intn(100), nil
                        },
                },
        },
})
var Schema, _ = graphql.NewSchema(graphql.SchemaConfig{
        Query: queryType,
})
```

GraphQL TODOs Example

GraphQL Todos: Model

```
package model

var TodoList []Todo

type Todo struct {
        ID   string    `json:id`
        Text string    `json:text`
        Done bool      `json:done`
}
```

GraphQL Todos: Schema Objects

```
var todoType = graphql.NewObject(graphql.ObjectConfig{
        Name: "Todo",
        Fields: graphql.Fields{
                "id": &graphql.Field{
                        Type: graphql.String,
                },
                "text": &graphql.Field{
                        Type: graphql.String,
                },
                "done": &graphql.Field{
                        Type: graphql.Boolean,
                },
        },
})
```

GraphQL Todos: RootQuery

```
var rootQuery = graphql.NewObject(graphql.ObjectConfig{
    Name: "RootQuery",
    Fields: graphql.Fields{
        "list": &graphql.Field{
            Type:        graphql.NewList(todoType),
            Description: "list of all todos",
            Resolve: func(params graphql.ResolveParams) (interface{}, error) {
                return model.TodoList, nil
            },
        },
    },
})
```

GraphQL Todos: RootQuery + Field Arguments

```
"todo": &graphql.Field{
        Type:        todoType,
        Args: graphql.FieldConfigArgument{
                "id": &graphql.ArgumentConfig{
                        Type: graphql.NewNonNull(graphql.String),
                },
        },
        Resolve: func(params graphql.ResolveParams) (interface{}, error) {
                idQuery, ok := params.Args["id"].(string)
                if ok {
                        for _, todo := range model.TodoList {
                                if todo.ID == idQuery {
                                        return todo, nil
                                }
                        }
                }
                return nil, fmt.Errorf("TODO with id='%s' not found", params.Args["id"])
        },
},
```

GraphQL Todos: RootQuery + Field Arguments

```
"todo": &graphql.Field{
        Type:        todoType,
        Args: graphql.FieldConfigArgument{
                "id": &graphql.ArgumentConfig{
                        Type: graphql.NewNonNull(graphql.String),
                },
        },
        Resolve: func(params graphql.ResolveParams) (interface{}, error) {
                idQuery, ok := params.Args["id"].(string)
                if ok {
                        for _, todo := range model.TodoList {
                                if todo.ID == idQuery {
                                        return todo, nil
                                }
                        }
                }
                return nil, fmt.Errorf("TODO with id='%s' not found", params.Args["id"])
        },
},
```

GraphQL Todos: RootMutation – Create TODO – I

```
var rootMutation = graphql.NewObject(graphql.ObjectConfig{
        Name: "RootMutation",
        Fields: graphql.Fields{
            "create": &graphql.Field{
                Type:        todoType,
                Description: "Create new todo.",
                Args: graphql.FieldConfigArgument{
                    "text": &graphql.ArgumentConfig{
                        Type: graphql.NewNonNull(graphql.String),
                    },
                },
```

GraphQL Todos: RootMutation – Create TODO – II

```
                Resolve: func(params graphql.ResolveParams) (interface{}, error) {
                        text, ok := params.Args["text"].(string)
                        if ok {
                                newTodo := model.Todo{
                                        ID:   uuid.New().String(),
                                        Text: text,
                                        Done: false,
                                }
                                model.TodoList = append(model.TodoList, newTodo)
                                return newTodo, nil
                        }
                        return nil, fmt.Errorf("error updating todo: %v", params.Args["text"])
                },
        },
```

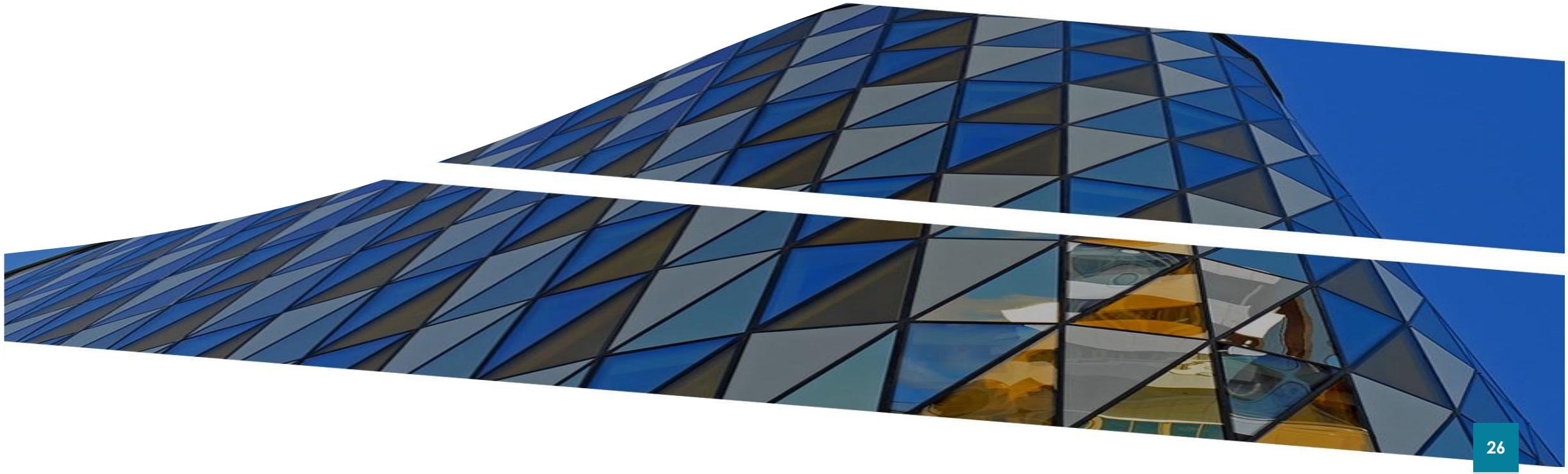
GraphQL Todos: RootMutation – Update TODO – I

```
"update": &graphql.Field{
        Type:        todoType,
        Description: "Update todo done status.",
        Args: graphql.FieldConfigArgument{
                "id": &graphql.ArgumentConfig{
                        Type: graphql.NewNonNull(graphql.String),
                },
                "done": &graphql.ArgumentConfig{
                        Type: graphql.NewNonNull(graphql.Boolean),
                },
        },
```

GraphQL Todos: RootMutation – Update TODO – II

```
                Resolve: func(params graphql.ResolveParams) (interface{}, error) {
                        id, ok1 := params.Args["id"].(string)
                        done, ok2 := params.Args["done"].(bool)
                        if ok1 && ok2 {
                                // Search list for todo with id and change the done status
                                for i := 0; i < len(model.TodoList); i++ {
                                        if model.TodoList[i].ID == id {
                                                model.TodoList[i].Done = done
                                                return model.TodoList[i], nil
                                        }
                                }
                                return nil, fmt.Errorf("error updating todo: id='%s' not found", id)
                        }
                        return nil, fmt.Errorf("error updating todo with id='%v'", params.Args["id"])
                },
        },
    },
})
```

More examples



There is More About GraphQL …

- GraphQL subscriptions with Go, GQLgen and MongoDB – https://www.freecodecamp.org/news/https-medium-com-anshap1719-graphql-subscriptions-with-go-gqlgen-and-mongodb-5e008fc46451/
- Genrating Golang code with GQLgen (schema-first approach) – https://github.com/99designs/gqlgen, https://gqlgen.com/getting-started/
- Using Golang Relay GrpahQL server implementation – https://relay.dev/, https://github.com/graphql-go/relay, https://github.com/sogko/golang-relay-starter-kit

Source: https://en.wikipedia.org/wiki/GraphQL

Thank's for Your Attention!

Trayan Iliev

IPT – Intellectual Products & Technologies

http://iproduct.org/

http://robolearn.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD