|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ |  |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |  |

**Институт информационных технологий**

КАФЕДРА ИНСТРУМЕНТАЛЬНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИиППО)

ПрактическАЯ РАБОТА №6

по дисциплине «Интерфейсы прикладного программирования»

Выполнил студент группы ИКБО-24-20 Постнов Н.С.

Принял Зарипов Евгений Анатольевич.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2022г. *подпись студента*

«Зачтено» «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г. *подпись преподавателя*

Москва 2022

**О Г Л А В Л Е Н И Е**

Цель работы ………………………………………………………………………………….....3

Теоретическое введение……………………………………………………………………......3

Выполнение практической работы...................…………………………………………..........3

Вывод.............................................................................................................................................4

# Практическая работа №7

**Цель работы**

Целью данной практической работы является знакомство обучающихся с набирающим популярность современным подходом к проектированию и реализации API на основе графовых моделей и с реализующей данный подход технологией на основе спецификации GraphQL.

**Постановка задачи**

Используя теоретические сведения из данной практической работы, открытые интернет-источники, официальную документацию по GraphQL необходимо, с использованием SDL создать схему, реализовать сервер и клиента GraphQL для следующей бизнес-задачи:

**Создание приложения для хранения информации об автомобилях.** Схема должна реализовывать возможность хранения краткой информации об автомобиле (поле title), уникальном идентификаторе автомобиля(поле id), информации о бренде автомобиля (поле brand), текущей цены автомобиля (поле price), возраста автомобиля (поле age).

**Ход работы**

На созданном в теоретическом введении Node.js сервере (Рисунки 36-39) переопределяется схема под индивидуальный вариант (Листинг 4)

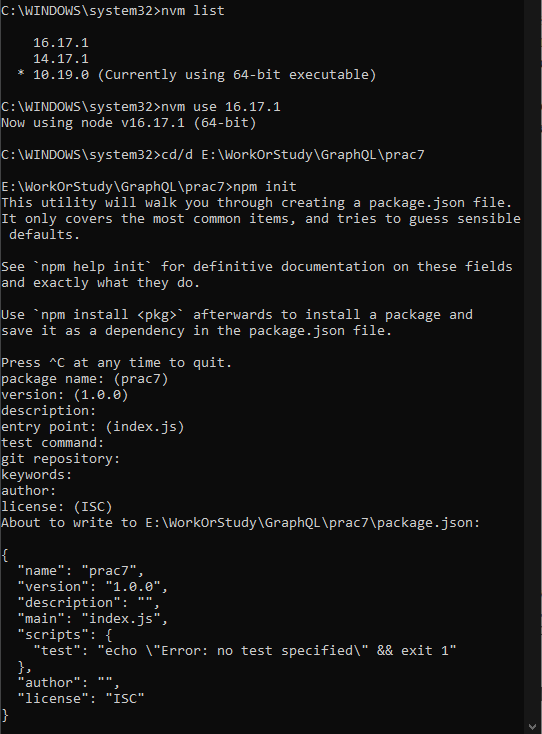


Рисунок 1– Инициализация node.js проекта

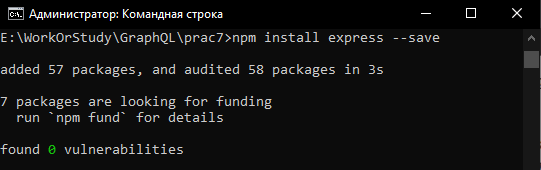


Рисунок 2– Установка express

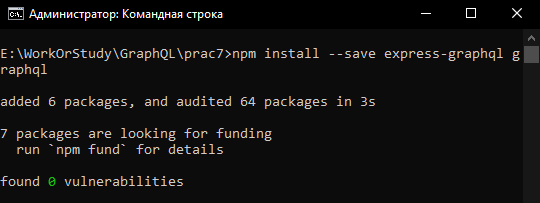


Рисунок 3 – Установка GrapQL

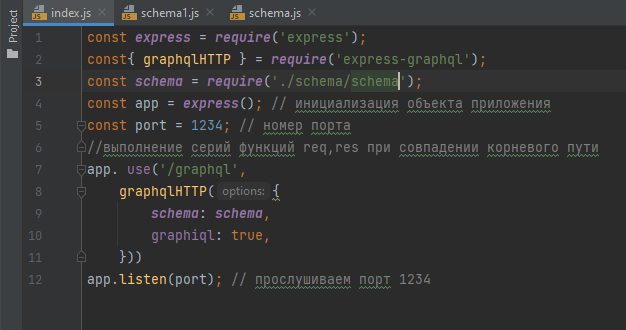


Рисунок 4– Конфигурация index.js

Листинг 1– Код schema.js

const graphql = ***require***('graphql');  
const ***\_*** = ***require***("lodash");  
const {  
 GraphQLID,  
 GraphQLString,  
 GraphQLList,  
 GraphQLInt,  
 GraphQLNonNull,  
 GraphQLObjectType,  
 GraphQLSchema  
} = graphql;  
  
const cars = [  
 {  
 id: "1",  
 title: "Mark 2 НЕ бит не крашен",  
 brand: "Toyota",  
 price: 666666,  
 age: 1  
 },

Продолжение листинга 1.

{  
 id: "2",  
 title: "Вишневая семерка",  
 brand: "ВАЗ",  
 price: 50000,  
 age: 5  
 },  
 {  
 id: "3",  
 title: "Гелик гусейна гасанова",  
 brand: "Mercedes-Benz",  
 price: 10000000,  
 age: 7  
 }];  
  
const CarInfoType = new GraphQLObjectType({  
 name:'CarInfo',  
 fields: () =>({  
 id: {type: GraphQLID},  
 title: {type: GraphQLString},  
 brand: {type: GraphQLString},  
 price: {type: GraphQLInt},  
 age: {type: GraphQLInt}  
 })  
})  
  
const RootQuery = new GraphQLObjectType({  
 name: 'RootQueryType',  
 fields: {  
 info:{  
 type: GraphQLString,  
 resolve(parent, args){  
 return "Сервер запущен"  
 }  
 },  
 car:{  
 type: CarInfoType,  
 args: {id: {type: GraphQLID}},  
 resolve(parent, args) {  
 return ***\_***.find(cars, {id: args.id});  
 }  
 },  
 cars:{  
 type: new GraphQLList(CarInfoType),  
 resolve(parent, args) {  
 return cars;  
 }  
 }  
 }  
});  
  
const Mutations = new GraphQLObjectType({  
 name: 'Mutations',  
 fields:{  
 addcar: {  
 type: CarInfoType,

Продолжение листинга 1.

args: {  
 id: {type: new GraphQLNonNull(GraphQLID)},  
 title: {type: new GraphQLNonNull(GraphQLString)},  
 brand: {type: GraphQLString},  
 price: {type: new GraphQLNonNull(GraphQLInt)},  
 age: {type: GraphQLInt}  
 },  
 resolve(parent, args) {  
 const arrLength = cars.push(args);  
 return cars [arrLength - 1];  
 }  
 }  
 }  
});  
  
***module***.exports = new GraphQLSchema({  
 query: RootQuery,  
 mutation: Mutations  
});

Далее проверка работы сервера с помощью POST-запроса или mutation (Рисунок 40) и GET-запроса (Рисунок 41), который выводит список всех машин.

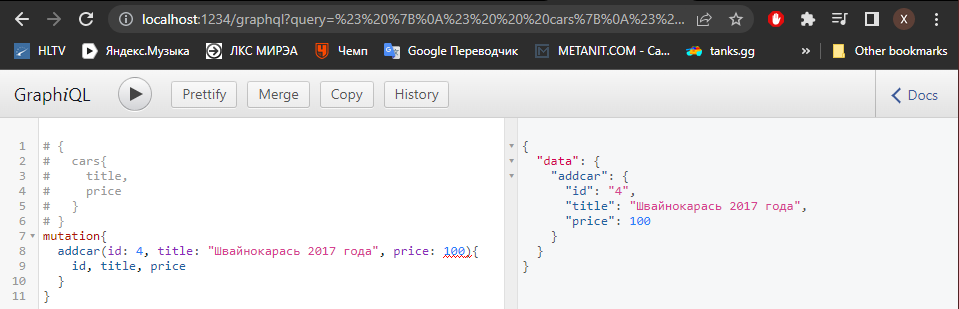


Рисунок 4 – Выполнение mutation-query

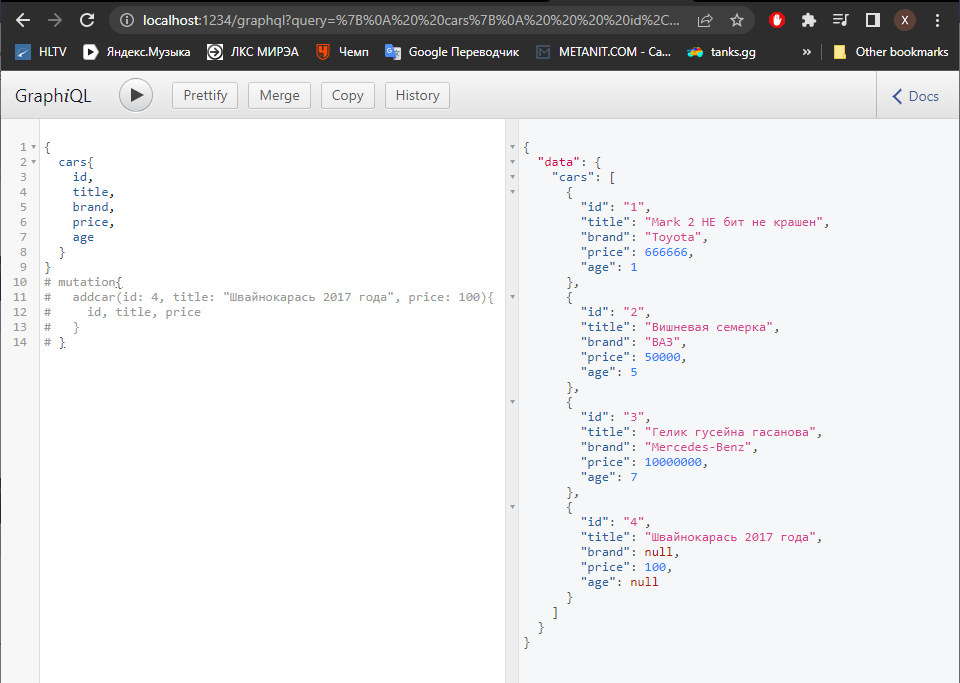


Рисунок 5– Выполнение cars-query

**Вывод**

Используя теоретические сведения из данной практической работы, открытые интернет-источники, официальную документацию по GraphQL с использованием SDL создал схему, реализовал сервер и клиента GraphQL для приложения для хранения информации об автомобилях.