|  |  |
| --- | --- |
|  |  |
| 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 №8

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

# Практическая работа №8

**Цель работы**

Знакомство обучающихся с ещё одним архитектурным стилем, который может использоваться для создания программных интерфейсов приложений широкого назначения. Данный стиль основывается на использовании технологии удалённого вызова процедур (англ. remote procedure call) и её дальнейшей модернизации, осуществлённой в рамках технологии gRPC.

**Постановка задачи**

Используя теоретические сведения из данной практической работы, открытые интернет-источники, официальную документацию по gRPC необходимо реализовать минимально жизнеспособный продукт (MVP) клиент-серверное приложение с API и базовыми операциями create, reade, update, delete (CRUD) для следующей бизнес-задачи: cервис для формирования списка продуктов в магазине. Приложение должно иметь возможность ввода названия продукта и его количества, редактирования введённой информации, удаления продукта из списка, вывода всего списка покупок, а также установления отметок того, что продукт уже куплен.

**Ход работы**

Инициализируем проект и устанавливаем библиотеки Express, gRPC, hbs и uuid. (Рисунок 1-2)

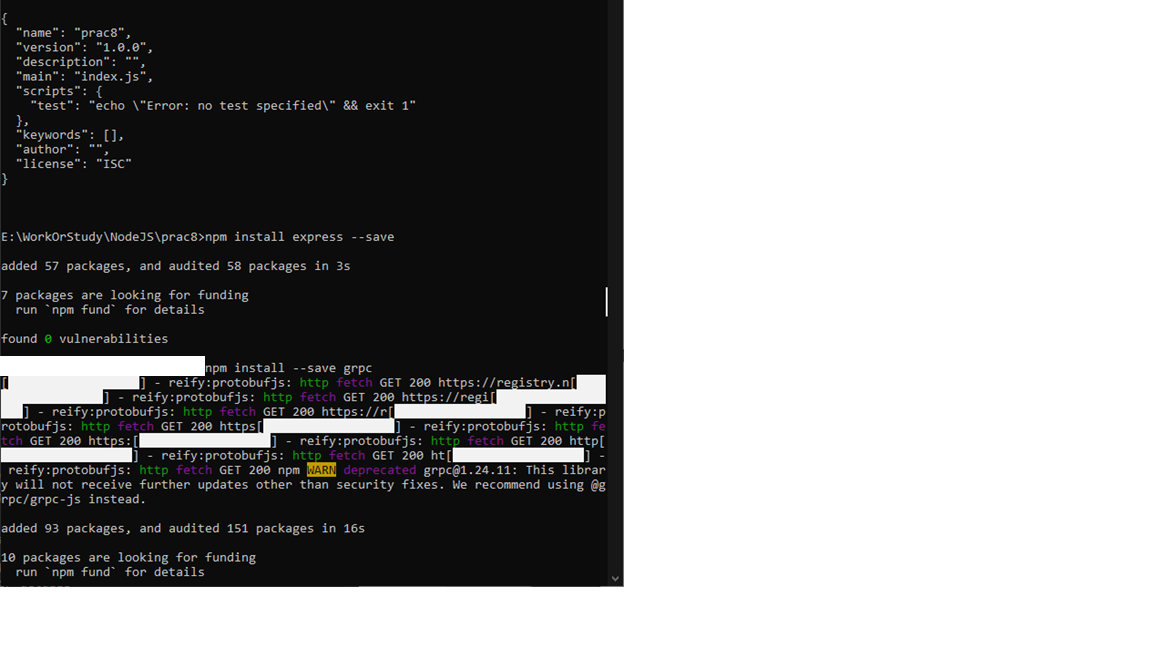


Рисунок 1 – Инициализация проекта

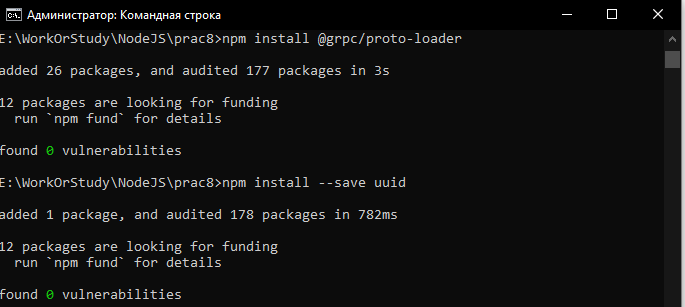


Рисунок 2 – Установка зависимостей

Далее необходимо сконфигурировать \*.proto файл (Рисунок 3) и подстроить server.js (Листинг 5) и index.js (Листинг 6) файлы для корректного удаленного вызова функций.

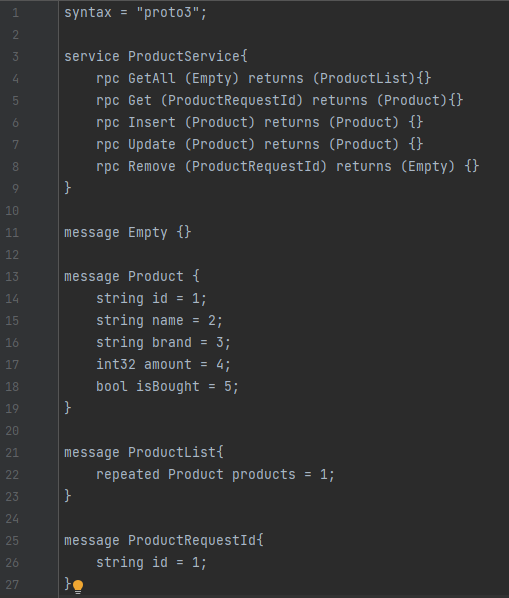


Рисунок 3 – Файл product.proto

Листинг 1– Код server.js

const PROTO\_PATH = "./product.proto";  
var ***grpc*** = ***require***("grpc");  
var protoLoader = ***require***("@grpc/proto-loader");  
var packageDefinition = protoLoader.loadSync(PROTO\_PATH, {  
 keepCase: true,  
 longs: String,  
 enums: String,  
 arrays: true  
});  
var productProto = ***grpc***.loadPackageDefinition(packageDefinition);  
const {v4: uuidv4} = ***require***("uuid");  
const server = new ***grpc***.Server();

Продолжение листинга 1.

const products = [  
 {  
 id: "1",  
 name: "Apex legends",  
 brand: "EA",  
 amount: 1,  
 isBought: false  
 }  
];  
server.addService(productProto.ProductService.service, {  
 getAll: (\_, callback) => {  
 callback(null, {products: products});  
 },  
 get: (call, callback) => {  
 let product = products.find(n => n.id == call.request.id);  
 if (product) {  
 callback(null, product);  
 } else {  
 callback({  
 code: ***grpc***.status.*NOT\_FOUND*,  
 details: "Не найдено"  
 });  
 }  
 },  
 insert: (call, callback) => {  
 let product = call.request;  
 product.id = uuidv4();  
 products.push(product);  
 callback(null, product);  
 },  
 update: (call, callback) => {  
 let existingProduct = products.find(n => n.id == call.request.id);  
 if (existingProduct) {  
 existingProduct.name = call.request.name;  
 existingProduct.brand = call.request.brand;  
 existingProduct.amount = call.request.amount;  
 existingProduct.isBought = call.request.isBought;  
 callback(null, existingProduct);  
 } else {  
 callback({  
 code: ***grpc***.status.*NOT\_FOUND*,  
 details: "Не найдено"  
 });  
 }  
 },  
 remove: (call, callback) => {  
 let existingProductIndex = products.findIndex(  
 n => n.id == call.request.id  
 );  
 if (existingProductIndex != -1) {  
 products.splice(existingProductIndex, 1);  
 callback(null, {});  
 }

Продолжение листинга 1.

else {  
 callback({  
 code: ***grpc***.status.*NOT\_FOUND*,  
 details: "Не найдено"  
 });  
 }  
 }  
});  
server.bind("127.0.0.1:50051", ***grpc***.ServerCredentials.*createInsecure*());  
***console***.log("Сервер запущен по адресу http://127.0.0.1:50051");  
server.start();

Листинг 2– Код index.js

const ***client*** = ***require***("./client");  
const path = ***require***("path");  
const ***express*** = ***require***("express");  
const bodyParser = ***require***("body-parser");  
const app = ***express***();  
app.set("views", path.join(***\_\_dirname***, "views"));  
app.set("view engine", "hbs");  
app.use(bodyParser.json());  
app.use(bodyParser.***urlencoded***({extended: false}));  
app.get("/", (req, res) => {  
 ***client***.getAll(null, (err, data) => {  
 if (!err) {  
 res.render("products", {  
 results: data.products  
 });  
 }  
 });  
});  
app.post("/save", (req, res) => {  
 let newProduct = {  
 name: req.body.name,  
 brand: req.body.brand,  
 amount: req.body.amount,  
 isBought: req.body.isBought  
 };  
 newProduct.isBought = newProduct.isBought == null ? false : true;  
 ***client***.insert(newProduct, (err, data) => {  
 if (err) throw err;  
 ***console***.log("Товар добавлен", data);  
 res.redirect("/");  
 });  
});  
app.post("/update", (req, res) => {  
 const updateProduct = {  
 id: req.body.id,  
 name: req.body.name,  
 brand: req.body.brand,  
 amount: req.body.amount,  
 isBought: req.body.isBought  
 };  
 updateProduct.isBought = updateProduct.isBought == null ? false : true;  
 ***client***.update(updateProduct, (err, data) => {  
 if (err) throw err;  
 ***console***.log("Товар успешно обновлён", data);  
 res.redirect("/");  
 });  
});

Продолжение листинга 2.

app.post("/remove", (req, res) => {  
 ***client***.remove({id: req.body.product\_id}, (err, \_) => {  
 if (err) throw err;  
 ***console***.log("Товар удалён");  
 res.redirect("/");  
 });  
});  
const PORT = ***process***.env.PORT || 3000;  
app.listen(PORT, () => {  
 ***console***.log("Сервер запущен на порту %d", PORT);  
});

После необходимо создать фронтэнд-часть с помощью шаблонизатора hbs, (фрагмент представлен на рисунке 4).

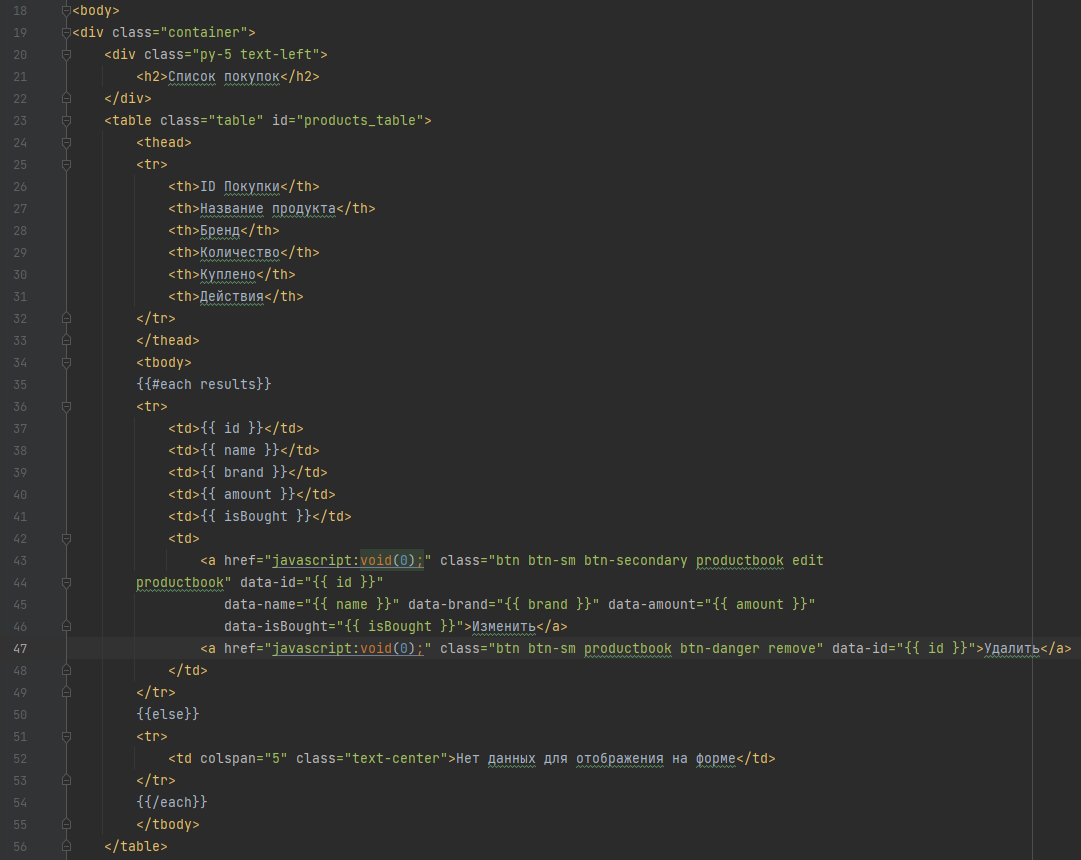


Рисунок 4 – Файл products.hbs

Переходим к тестированию, запускаем раздельно серверную и клиентскую часть (Рисунки 45-46).



Рисунок 5 – Серверная часть



Рисунок 6 – Клиентская часть

Проведем тестирование всех методов (Create, Read, Update, Delete). Результаты представлены в виде логов в консоли клиентской части, а также в графическом интерфейсе пользователю Рисунок 7.

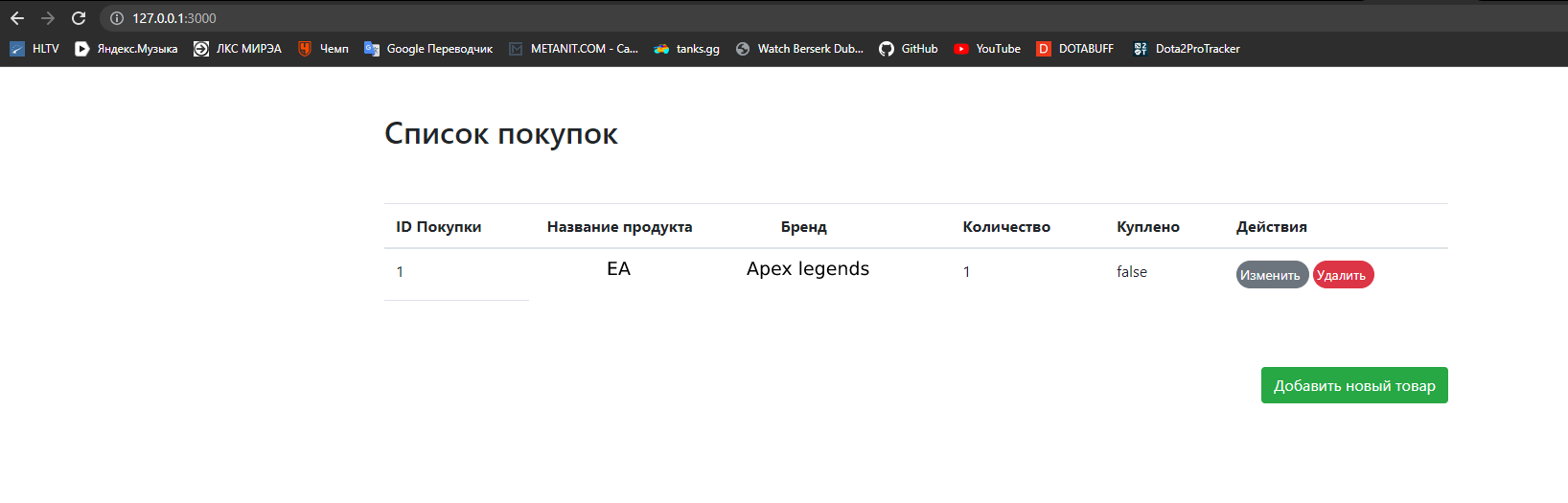


Рисунок 7 – Конечный вид фронт-части API

**Вывод**

Используя теоретические сведения из данной практической работы, открытые интернет-источники, официальную документацию по gRPC реализовал минимально жизнеспособный продукт (MVP) клиент-серверное приложение с API и базовыми операциями create, read, update, delete (CRUD) для cервиса.