УДК 004.42
А. В. Стрелков, А. С. Истомин
ФГАОУ ВО «Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина», г. Екатеринбург, Россия

ИССЛЕДОВАНИЕ АЛЬТЕРНАТИВНЫХ СПОСОБОВ ПОСТРОЕНИЯ API НА ПРИМЕРЕ СОЗДАНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ НОМЕНКЛАТУРОЙ ТОВАРОВ С ПОМОЩЬЮ GRAPHQL

Аннотация. *В результате работы изучены способы взаимодействия программ между собой на примере использования технологии GraphQL при программировании на C# и её применимости при построении современных информационных систем. Спроектирована база данных с использованием объектно-ориентированной технологии Entity Framework и подхода Code First. Изучена open-source библиотека «Hot Chocolate» и особенности работы с ней. Описаны основы работы с языком запросов, представлены результаты обращения к базе данных при формировании номенклатурных позиций. Объединение нескольких подзапросов в одну конечную точку. Особое отличие данной технологии -использование всех полей в модели данных в качестве аргументов при фильтрации. Проанализирована структура данных ответа и её влияние на коммутационную сеть.*
Ключевые слова: *программное обеспечение, взаимодействие программ, API, микросервисы, GRAPHQL, схема, архитектура, запрос, hot chocolate.*
Abstract. *As a result of the work, there are studied the ways of interaction of programs with each other by the example of using GraphQL technology in programming on C# and its application in the construction of modern information systems. The database was designed, using the object-oriented EntityFramework technology and the CodeFirst approach. There are studied the open-source library «Hot Chocolate» and the features of working with it. Also are described the basics of working with the query language and the results of accessing the database during the formation of nomenclature positions. There is combining multiple subqueries into one endpoint. The differentiating feature of this technology is using all fields in the data model as arguments during filtering. Analyzed the response data structure and its impact on the switching network.*
Key words: *software, program integration, API, microservices, GRAPHQL, scheme, architecture, request, hot chocolate.*

При построении кроссплатформенных информационных систем необходимо проектировать архитектуру будущего приложения таким образом, чтобы интеграция с новыми платформами осуществлялась с минимальными затратами. И чем более разнородными являются модули между собой, тем больше вероятность ошибки. Решением этой проблемы является создание программного обеспечения в виде набора сервисов, каждый из которых является независимым, но в то же время имеет возможность обмениваться информацией с другими звеньями системы. Отдельные компоненты приложений становятся абстракциями: создателям нового ПО не приходится лезть в логику низкоуровневых функций и разбираться в их реализации [1].

Посредником при общении нескольких приложений являются программные интерфейсы приложений – API. Основным подходом является REST API. Данных подход предполагает следующие методы взаимодействия с данными:

GET, POST, PUT, DELETE. Однако данный архитектурный стиль предполагает наличие нескольких «конечных» точек при взаимодействии с разными сервисами, так как каждое обращение к разнородным данным является отдельным URL -запросом. Это накладывает сложности при создании и сопровождении, а также увеличивает время разработки, и, следовательно, стоимость [2].

Одним из альтернативных способов построения микросервисной архитектуры является использование GraphQL — языка запросов и манипулирования данными для API, позволяющий разработчикам оперировать только теми данными, которые нужны в конкретный момент. Для реализации схемы реализации API используется синтаксис языка определения схем (SDL) Данная технология имеет позволяет комбинировать запросы, объединяя несколько точек входа в одни состоит из двух основных блоков: схемы и запросов. Схема является базой для создания типов данных и их полей, а запросы, предназначенные для оперирования данными, разделяются на два типа: Queries (запросы, реализуют операцию чтения) и Mutations (мутации, реализуют операции создания, обновления и удаления) [3].

Целью данной работы является исследование применимости использования технологии GraphQL при создании информационной системы формирования заказов и номенклатурных позиций. Прародителем разрабатываемой системы является модуль Web-продаж предназначенное для платежно-пропускных систем [4]. Приложение представляет собой API, реализующий функции создания и редактирования заказов, а также наполнения номенклатурных позиций.

Разработка базы данных осуществлялась с использованием фреймворка EntityFramework, позволяющий абстрагироваться от базы данных и работать с данными независимо от типа хранилища. При этом оперировать приходится сущностями – которые являются объектами и наборами их полей, а не таблицами. При этом создание модели данных реализовано с использованием подхода «Code First», предполагающий автоматическую генерацию таблиц и связей по имеющимся классам моделей данных. В результате была разработана следующая архитектура базы данных, представленная на рисунке 1 [5].

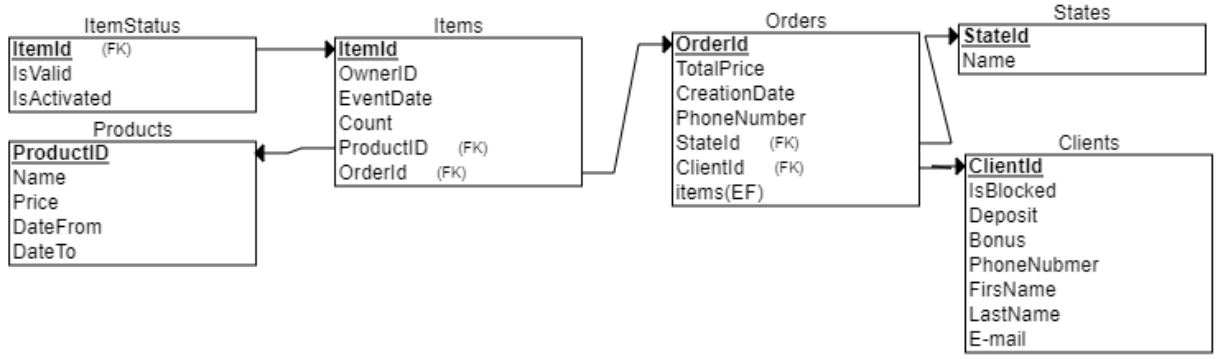

Рис. 1. Архитектура базы данных

Так как GraphQL является лишь спецификацией, то необходимо выбрать библиотеку для разработки системы. Выбор пал на библиотеку «Hot Chocolate», имеющую в составе инструменты, позволяющие упростить написание как backend, так и тестирование полученных результатов [6].

В ходе разработки приложения были реализованы методы чтения из базы данных, а также функции создания, изменения и удаления заказов и номенклатурных позиций в них. Отличием от прародительской системы заключается в том, что в разработанном приложении выполнение этих действий можно объединить в одно, убрав несколько точек входа. Сравнение структуры запроса на создание нового заказа и заполнения позиций представлено на рисунке 2.

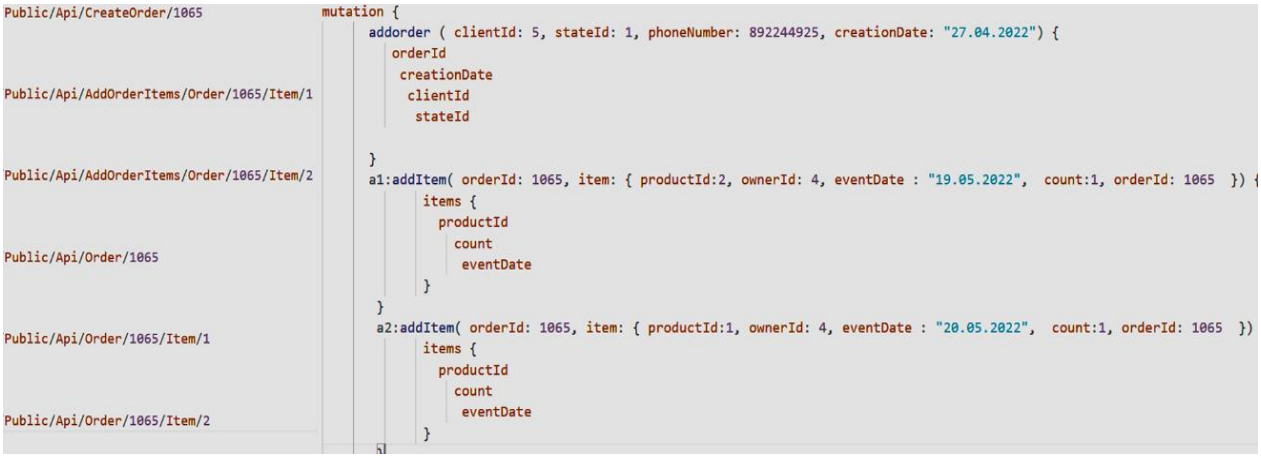

Рис. 2. Сравнение запросов на создание заказа

Использование GraphQL позволяет объединять практически неограниченное количество запросов в один, что существенно ускоряет и прощает разработку приложения. Также, в отличие от традиционного подхода, результатом вышеописанных действий является вывод информации о созданной сущности. В результате возвращаются ожидаемые данные, так как они полностью соответствует структуре запроса. Это позволяет уменьшить количество данных, передаваемых клиенту, по сравнению с традиционным подходом, где для вывода каждой вложенности модели необходимо было бы посылать отдельные подзапросы (рис. 3).

Одной из главных особенностей применения GraphQL является встроенная фильтрация данных. Это обусловлено тем, что каждое описанное поле или свойство в структуре модели одновременно является фильтром для вывода информации. В результате работы были реализованы методы чтения из БД, позволяющие производить фильтрацию по нескольким параметрам одновременно. На рисунке 4 представлен результат запроса с фильтрацией: вывод всех заказов, где у клиента в имени имеется буква «А» и баланс на счету более 100.

Рис. 3. Вывод информации о созданном заказе

Рис. 4. Запрос с фильтрацией

Именно эта особенность данной технологии позволяет уменьшить нагрузку на коммутационную сеть и получать информацию именно в том количестве, в котором она запрашивается, в результате снижения количества запросов. в REST каждый запрос обычно вызывает ровно одну функцию-обработчик маршрута. В GraphQL один запрос может вызвать множество функций для построения сложного ответа с множеством вложенных ресурсов.

В результате работы была спроектирована и реализована структура базы данных для системы формирования заказов и номенклатурных позиций, а также ее серверная составляющая. Созданный функционал позволяет производить выборку данных и осуществлять операции создания, изменения и удаления позиций товаров в системе. Использование технологии GraphQL позволяет объединять несколько запросов в один единый, тем самым лишая разработчика проблемы нескольких точек входа. Помимо этого, созданное приложение уменьшает нагрузку на сеть благодаря уменьшению количества данных, предо-

ставляемых клиенту. Таким образом, данный метод построения API является перспективным для использования для подобного рода приложений. В дальнейшем программа будет расширяться, а её функциональные возможности совершенствоваться.

Список использованных источников

1. REST API Design Rulebook / Mark Masse, O REILLY 2011. – 210 с.
2. Сравнение архитектурных стилей API: SOAP vs REST vs GraphQL vs RPC [Электронный ресурс] – URL: https://nuancesprog.ru/p/11310/ (дата обращения: 23.04.2022 г.).
3. HotChocolate | ChilliCream GraphQL Platform [Электронный ресурс] – Режим доступа: https://chillicream.com/docs/hotchocolate (дата обращения: 23.04.2022 г.).
4. Программное обеспечение для платежно-пропускных и билетно-пропускных систем БАРС [Электронный ресурс] – URL: https://www.datakrat.ru/software/bars (дата обращения: 23.04.2022 г.).
5. Beginning Entity Framework Core 5. From Novice to Professional / Eric Vogel, Apress, 2021. – 336 с.
6. Banana Cake Pop| GUI for any GraphQL API [Электронный ресурс] – URL: https://chillicream.com/docs/bananacakepop (дата обращения: 24.04.2022 г.).

УДК 004.428.4:669.162.263

И. С. Уланов, В. В. Лавров, И. А. Гурин, Н. А. Спирин
ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина», г. Екатеринбург, Россия

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ РАСЧЕТА ПОКАЗАТЕЛЕЙ ГАЗОДИНАМИЧЕСКОГО РЕЖИМА ДОМЕННОЙ ПЕЧИ

Аннотация. *Рассмотрены основные предпосылки, принципы перехода от использования локальных систем к веб-приложениям на предприятиях на примере разработки программного обеспечения «Расчёт газодинамического режима доменной печи» для металлургического предприятия. Программное обеспечение предназначено для расчёта показателей газодинамического режима доменной печи в нескольких периодах: базовый, сравнительный и прогнозный. При переходе к веб-технологии выполнена разработка веб-интерфейсов, программирование клиентской и серверной частей приложения, осуществлено подключение к базе данных. Представлена архитектура новой системы, описаны основные функциональные возможности.*

Ключевые слова: *разработка, доменный цех, web-приложение, газодинамический режим, база данных, ASP.NET MVC.*

Abstract. *The main prerequisites, principles of transition from the use of local systems to web applications at enterprises are considered on the example of developing software "Calculation of the gas-dynamic regime of a blast furnace" for a metallurgical enterprise. The software is designed*