Сустов С.В., Подколзин В.В.

Создание серверной части системы управления контентом в сфере профессионального образования

*Кубанский государственный университет
(Россия, Краснодар)*

doi: 10.18411/lj-04-2019-261

idsp: ljjournal-04-2019-261

Аннотация

В статье раскрывается подход к созданию системы управления контентом (CMS) в сфере профессионального образования. Актуальность обусловлена необходимостью обеспечить возможность простого создания новых образовательных платформ с целью предоставить возможность получать и совершенствовать профессиональные навыки с помощью современных технологий. Освещается архитектура предлагаемой системы.

Ключевые слова: профессиональное образование, управление контентом, профессиональные навыки, CMS

Abstract

The article describes the approach to creating the content management system (CMS) for the professional education. The need for designing a simple way to create new educational platforms targeted to provide an ability to acquire and improve professional skills sets the relevance of the article. The suggested system architecture is being highlighted.

Keywords: professional education, content management, professional skills, CMS

Современные сайты растут в объемах содержащейся информации и в сложности представления этой информации. Стоимость труда программистов слишком высока, чтобы нанимать их для добавления страниц или изменения информации. Особенно невыгодным это становится при учете того, что именно типов страниц у каждого сайта, как правило, не очень много, много самих экземпляров страниц, использующих одну и ту же разметку, но содержащих разные данные. Для решения этой проблемы существует класс систем, служащих для управления контентом (CMS). Современные CMS стремятся охватить максимально широкий спектр потребностей, жертвуя при этом удобством управления и даже функциональными возможностями в узких предметных областях. Например, не представляется возможным на большинстве подобных систем построить одностраничное приложение (SPA), хотя сами технология и методология их разработки уже достаточно не новы.

Цель работы заключалась в создании серверной части CMS, позволяющей создавать порталы в сфере профессионального образования, достаточно узкой предметной области. Созданные на базе этой системы решения должны обеспечивать возможность обучения и повышения квалификации для сотрудников различных учреждений.

В ходе анализа существующих решений стало ясно, что CMS должна быть исполнена с реализацией концепции «headless». Это означает, что сама система в данном случае выступает лишь как хранилище «сырых» данных. Интерпретация этих данных и их визуализация должны быть возложены на конечного клиента CMS. Подобный подход позволяет выступать в роли клиента любому приложению — будь то сайт или мобильное приложение. Это особенно актуально с учетом масштабного распространения мобильного интернета и позволяет снять ограничения стандартных систем, привязывающих содержимое данных к их визуальному представлению.

В качестве языка программирования был выбран C# и фреймворк ASP.NET Core MVC. Именно Core-версия фреймворка выбрана для того, чтобы позволить системе запускаться на Unix-системах в силу специфичности и сложности грамотной настройки

рабочего приложения на серверах под управлением ОС Windows. Однако, использование данной версии фреймворка также не отменяет возможность запустить приложение на Windows при использовании IIS в качестве проксирующего сервера.

Сам же C# идеально подходит под задачу в силу своей объектно-ориентированности и встроенной поддержки динамических типов [1]. Тип `dynamic` позволяет строить структуры данных неограниченной сложности, сериализовать/десериализовать информацию, считывать и записывать данные, избегая повторной компиляции и перезапуска приложения.

Сами данные, формируемые администратором площадки хранятся в базе данных под управление СУБД PostgreSQL. Эта СУБД поставляется с исходным программным кодом под собственной лицензией, сходной с BSD и MIT, что позволяет неограниченно ее использовать и изменять. PostgreSQL представляет из себя довольно стандартную реляционную СУБД, полностью реализующую ACID, но при этом имеющую дополнительные возможности [2]. Например, есть встроенная поддержка массивов и JSON-объектов как типов данных. При этом PostgreSQL во многочисленных испытаниях показала себя лучше зачастую выбираемого MySQL в плане надежности и скорости работы.

В силу выбранного архитектурного подхода данные потенциально имеют разнородную и сложную структуру. Стандартный Restful API, обеспечивающий доступ к информации, в конечном итоге рискует стать сильно усложненным. Для избежания такой ситуации был применен опыт компании Facebook и их разработка GraphQL. Полная структурная схема системы представлена на рисунке 1.

Рисунок 1. Полная структурная схема разработанной CMS

Как видно из рисунка 1, клиентом системы может выступать любое приложение. Администраторская панель также является одним из клиентов. Сервер GraphQL выступает посредником между источником запроса и сервером приложения, пропуская все запросы через себя. Сам же сервер приложения забирает данные из СУБД.

GraphQL представляет из себя язык запросов к данным. С его помощью клиент формирует запросы декларативно, т. е. описывает, какие данные он хочет получить, а система по заданным правилам предоставляет ему желаемые данные [3]. Правила задаются в виде разрешающих функций (resolvers), а запросы в backend передаются через сервер GraphQL, который обрабатывает эти запросы и вызывает разрешающие функции. Подобный подход помимо гибкости в запросах позволяет также фильтровать передаваемые данные. Имея большой объект, зачастую необходимо получить из него лишь несколько полей. GraphQL позволяет указывать конкретные поля, необходимые

клиенту. Сервер GraphQL фильтрует выдаваемые данные, тем самым позволяя экономить интернет трафик. Это в свою очередь очень актуально для мобильных клиентов, где интернет в силу своей природы далеко не всегда бывает стабильным.

Общая схема взаимодействия клиент-серверного взаимодействия представлена на рисунке 2.

Рисунок 2. Схема взаимодействия компонентов системы

Согласно предоставленной схеме (рисунок 2), клиент отправляет запрос на получение данных в API в единственный маршрут, предоставляемый сервером GraphQL. Сервер GraphQL обрабатывает запрос клиента и вызывает соответствующие разрешающие функции, обращаясь за данными по HTTP в рамках локальной сети в сервер приложения. Сервер приложения получает запрошенные данные из PostgreSQL и отдает их обратно. GraphQL фильтрует полученные данные согласно запросу пользователя и передает результат клиенту.

Созданное программное решение позволяет управлять содержимым порталов профессионального образования. Оно предоставляет возможность создания обучающих программ, проведения тестирования, выставления оценок и присваивания ступеней ученикам. Для полной реализации концепции «headless» в данных условиях администратору сайта предоставляется возможность создавать сложные структуры данных из заранее определенных примитивов. Например, html-текст, булевый тип, список, целое число.

Многие из этих примитивов транслируются в один и тот же тип внутри базы данных. Однако же, разграничение необходимо как для более точной интерпретации данных на стороне клиента, так и для обработки данных и их валидации. Текст и html-текст транслируются в тип text, не имеющий ограничений по длине. Целое число, число с плавающей точкой и булевый тип хранятся в соответствующих типах. Дата и дата со временем имеют тип timestamp without time zone. Списки формируются из созданных объектов данных. По сути, элементы списка просто ссылаются на уже сохраненные данные. Приведенный набор примитивных данных позволяет создавать любой сложности вопросы и тесты с различным количеством вариантов ответа.

Таким образом, было создано решение, занимающее свою нишу и предоставляющее возможности, которыми не обладают многие существующие аналоги. Сосредоточение на узкой предметной области позволило сконцентрировать усилия на необходимых для решения задачи характеристиках, отказавшись от не имеющих смысла в ее рамках. Созданная система позволяет эффективно управлять содержимым порталов в сфере профессионального образования.

1. Фримен, А. ASP.NET Core MVC с примерами на C# для профессионалов // Вильямс. – 2017. – 992с.
2. Smith, G. PostgreSQL 9.0 High Performance // Packt Publishing. – 2010. – 990с.
3. Бэнкс А., Порселло Е. GraphQL: язык запросов для современных веб-приложений // Питер. – 2019. – 240с.