– Возможность введения зависимостей;

– Расширяемость, основывающаяся на возможности добавления в проект дополнительных компонентов.

Таким образом, сравнивая ASP.NET с другими широко используемыми средствами веб-разработки, такими как PHP, Node.js или Ruby on Rails, можно выделить одно ключевое достоинство. Перечисленные платформы являются интерпретируемыми, в то время как C# – компилируемый язык. Это дает системе .NET серьезное преимущество в скорости работы. Все компоненты, не требуя интерпретатора, работают с фреймворком, который, в свою очередь, также скомпилирован и вызывает напрямую функции операционной системы, а большинство ошибок отлавливаются разработчиком в момент компиляции. Также наличие паттерна MVC позволяет разграничить написание кода на 3 основные части, что в свою очередь значительно добавляет удобство пользования.

Список литературы

1. ASP.NET Core. [Электронный ресурс], 2018. Режим доступа: https://www.asp.net/core/overview/aspnet-vnext/ (дата обращения: 02.07.2018).
2. ASP.NET Core. [Электронный ресурс], 2018. Режим доступа: https://metanit.com/sharp/aspnet5/ (дата обращения: 04.07.2018).

АРХИТЕКТУРА ТЕХНОЛОГИИ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ ASP.NET CORE MVC

Шарапов Н.Р.

Шарапов Николай Романович – бакалавр,
направление: информационные системы и технологии,
кафедра геоинформационных систем, факультет информатики и робототехники,
Уфимский государственный авиационный технический университет, г. Уфа

Аннотация: *технология разработки веб-приложений ASP.NET Core MVC является актуальной. Соответственно имеется необходимость в анализе архитектуры данной технологии.*

Ключевые слова: *ASP.NET Core MVC, архитектура, веб-приложение.*

ASP.NET Core MVC

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету. Приложения ASP.Net Core, разработанные с помощью паттерна MVC, имеют соответствующий архитектурный шаблон: модель – представление – контроллер [1].

Модель - описывает используемые в приложении данные, а также логику, которая связана непосредственно с данными. Как правило, объекты моделей хранятся в базе данных.

Представление - отвечают за визуальную часть или пользовательский интерфейс, также может содержать логику, связанную с отображением данных.

Контроллер - представляет центральный компонент, который обеспечивает связь между пользователем и приложением, представлением и хранилищем данных. Он содержит логику обработки запроса пользователя. Контроллер получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод.

ASP.NET Core MVC предоставляет функции, которые позволяют эффективно создавать веб-интерфейсы API и веб-приложения:

– Шаблон Model-View-Controller (MVC) помогает сделать веб-API и веб-приложения тестируемыми.

– Страницы Razor - это основанная на страницах модель программирования, которая упрощает создание пользовательского веб-интерфейса и повышает его эффективность.

– Разметка Razor предоставляет эффективный синтаксис для страниц Razor и представлений MVC.

– Вспомогательные функции тегов позволяют серверному коду участвовать в создании и отображении HTML-элементов в файлах Razor.

– Благодаря встроенной поддержке нескольких форматов данных и согласованию содержимого, веб-API становятся доступными для множества клиентов, включая браузеры и мобильные устройства.

– Привязка модели автоматически сопоставляет данные из HTTP-запросов с параметрами методов действия.

– Проверка модели автоматически выполняется на стороне сервера и клиента.

Архитектура ASP.NET Core MVC показана на рисунке 1.

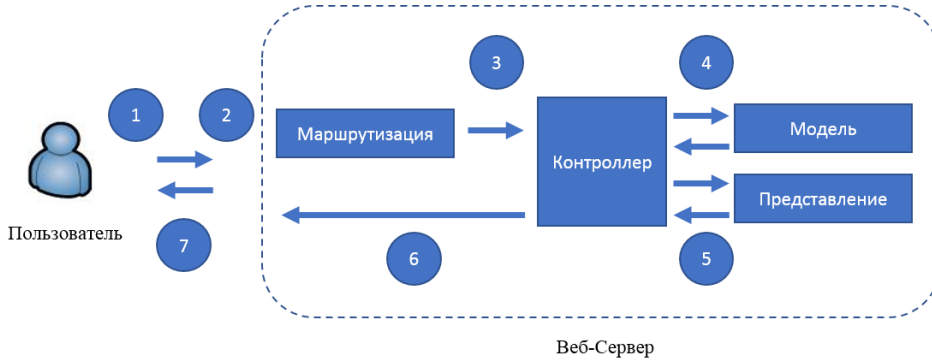

Рис. 1. Архитектура ASP.NET Core MVC

Ниже приведена последовательность шагов взаимодействия пользователя с веб-сайтом, разработанным с помощью технологии ASP.NET Core MVC:

1. Пользователь вводит URL-адрес в браузере и осуществляет запрос.
2. Запрос доходит до веб-сервера и перенаправляется на механизм маршрутизации.
3. На основе URL, механизм маршрутизации выбирает соответствующий контроллер.
4. Контроллер обращается к базе данных, используя модель, чтобы получить соответствующие данные для введенного запроса.
5. Контроллер вызывает механизм просмотра и возвращает представление страниц.
6. Контроллер возвращает полученное представление.
7. Запрошенный ресурс отправляется обратно в браузер.

Список литературы

1. ASP.NET Core. [Электронный ресурс], 2018. Режим доступа: https://www.asp.net/core/overview/aspnet-vnext/ (дата обращения: 07.07.2018).