**Содержание**

[Введение 3](#_Toc29500936)

[1. Выбор технологии 5](#_Toc29500937)

[1.1. Язык программирования информационной системы 5](#_Toc29500938)

[1.2. Среда разработки информационной системы 8](#_Toc29500939)

[1.3. Современные технологии разработки веб-приложений. Особенности платформы ASP.NET Core 12](#_Toc29500940)

[Список использованных источников 23](#_Toc29500941)

# Введение

Актуальность темы. В связи с активным развитием технического и социального механизма общества, информация стала важнейшим ресурсом, который является частью всех сфер человеческой деятельности. Современные информационные технологии приравниваются к материальным и энергетическим ресурсам, создавая, храня, перерабатывая и обеспечивая эффективные способы представления информации потребителю. Они стали важным аспектом жизни общества и средством повышения эффективности управления всеми сферами деятельности общества.

В настоящее время информационные системы прочно вошли во все сферы человеческой деятельности: бизнес, производство, управление, образование и т. д. Это объясняется все возрастающим объемом и обрабатываемой информации, и задач управления, с которыми приходится сталкиваться человеку. Эффективность работы сотрудника, а следовательно, и всей организации зависит теперь от эффективности используемых информационных систем.

Поэтому современный подход к управлению опирается на использование информационных технологий, причем их количество тем больше, чем больше предприятие. С ростом сложности решаемых задач происходит рост сложности информационных систем, который отражается в увеличивающейся сложности архитектуры информационных систем. Развиваются подходы к ее реализации, средства и методологии ее описания.

Существует множество инструментов и способов разработки веб-приложений. Так, с помощью языка гипертекстовой разметки и каскадных таблиц стилей можно создавать простые статические веб-страницы, которые можно использовать в качестве сайтов-визиток. Подключая возможности Javascript, с простой статической веб-страницы можно сделать динамический сайт, который реагирует на действия пользователя. Значительное расширение функционала веб-приложения происходит при использовании серверных языков программирования, которые позволят выполнять серверные сценарии и соединятся с базой данных. Ключевым звеном в подобных системах является использованный серверный язык программирования и система управления базами данных. В среде веб-разработки используется множество технологий, при этом, наиболее используемыми являются PHP, ASP.NET, Perl, Phyton, которые позволяют создавать функциональные веб-приложения. Одной из новых и набирающих популярность технологий, с помощью которых можно создавать серверные приложения является технология .NET. Серверные приложения на платформе .NET можно разрабатывать в двух поддерживаемых реализациях: .NET Framework и .NET Core. В них используется множество одинаковых компонентов, а код можно использовать как в одной среде, так и в другой. При этом, между этими двумя средами существуют фундаментальные различия, и выбор платформы для реализации проекта зависит от поставленной задачи.

Таким образом, целью работы является разработка веб-сайта на основании технологии .NET Core.

Для достижения цели необходимо решить следующие задачи:

* обосновать выбор технологии разработки проекта;
* разработать структура проекта;
* выбрать СУБД для реализации проекта;
* реализовать сайт с использование технологии ASP.NET Core;
* проверить работоспособность созданного сайта.

Методологической основой исследования послужили обще-логические методы и приемы исследования (анализ, абстрагирование, обобщение, классификация, аналогия, системный подход, синтез), методы эмпирического исследования (сравнение, измерение, описание), методы проектирования и разработки программных средств.

# 1. Выбор технологии

## 1.1. Язык программирования информационной системы

Языком программирования для информационной системы выбран C#. Язык программирования C# — это универсальный объектно-ориентированный язык программирования. Первая общедоступная версия этого языка появилась в феврале 2002 года одновременно с выходом среды разработки Microsoft Visual Studio .NET. Начиная с того времени, язык C#, как и программная платформа Microsoft .NET, постоянно развиваются и приобретают всё более и более широкие дополнительные функциональные возможности. Для создания приложений для платформы .NET с использованием языка C# существуют различные средства разработки. Наиболее популярным из них является интегрированная среда разработки (ИСР) Microsoft Visual Studio.

В настоящее время язык программирования C# получает широкое распространение не только как универсальный язык профессиональной разработки программного обеспечения, но и как средство начального обучения программированию. В этом смысле язык C# составляет серьёзную конкуренцию таким языкам программирования, как BASIC и Pascal.

Отличительной особенностью языка программирования C# является то, что он построен на принципах объектно-ориентированного программирования (ООП). При этом любая, даже самая простая программа, является объектно-ориентированной. В языке C# удачно сочетаются испытанные средства программирования с самыми последними новшествами, и предоставляется возможность для эффективного написания программ, предназначенных для вычислительной среды современных предприятий.

С# представляет собой систему на основе специального байт-кода, с обещаниями много-платформенности. В 2000 г. ЕСМА по запросу Microsoft одобрила С# как стандартный язык. В связи с этим, сейчас С# — это язык, определенный в стандарте, который Microsoft не сможет менять по своему усмотрению. В этом является преимущество С#, так как он получил явную независимость от Microsoft.

Теория, лежащая в основе С# состоит в том, что язык может быть использован для создания много платформенных программ. В Windows обычная программа состоит из двоичного исполняемого файла. То есть исходный код программы компилируется в машинный формат. .NET все еще предоставляет такой тип разработки программного обеспечения, хотя он специально называется неуправляемым кодом.

С# содержит большое количество повторно используемого кода. Программы на С# компилируются в специальный байт-код, который может понять и выполнить отдельная программа. Common Language Runtime (CLR) — это программа, которая работает с байт-кодом С#. Однако CLR пошла дальше подхода виртуальной машины, и на самом деле компилирует приложения «на лету» в машинный код, тем самым увеличивая производительность. В случае часто вызываемых приложений для веб-серверов, таких как ASP.NET, результаты компиляции кэшируются и повторно используются, опять же для увеличения производительности.

Таким образом, С# действует по-другому: программы, написанные на этом языке, сначала «компилируются» в CLR-совместимый формат (похоже на то, что делает Java), а затем компилируются в машинный код, когда CLR первый раз их выполняет.

С# предоставляет встроенную поддержку СОМ-компонентов и Windows API, а также ограниченное использование указателей. Это означает, что хотя программы на этом языке могут быть не такими быстрыми, как приложения, выполняемые напрямую (из-за интерпретации CLR), их производительность обычно не такая низкая, в сравнении с другими технологии.

С# реализует модель безопасности формате песочницы, но основной контроль устанавливается сборкой, которая позволяет определить, какие операции могут выполняться некоторой программой или классом. С# поддерживает указатели для прямых манипуляций с памятью, хотя их использование также может контролироваться настройками безопасности.

Теоретически С# переносим, но пока никакие операционные системы производства не Microsoft не могут использовать его. Это не означает, что он полностью непереносим. У Microsoft также есть операционная система для карманных устройств, которая называется Windows Mobile (ранее она называлась Windows СЕ, Windows PocketPC, PocketPC 2000 и т. д), которая распространяется с .NET Framework и CLR для выполнения программ на С#. Как и Java, С# можно использовать на веб-серверах, настольных компьютерах и карманных устройствах, если на них есть соответствующая версия Windows.

С# реализует автоматический сбор мусора. В С# можно объявлять деструкторы, важно отметить, что они вызываются, когда процесс сбора мусора определяет, что объект больше недоступен коду и что необходима память. Деструкторы в С# имеют формат: ~ClassName();

Процесс сбора мусора вызовет этот метод автоматически, когда возникнет необходимость в этом.

Язык С# был разработан, чтобы упростить создание компонентов или дополнительных модулей, которые могут быть привязаны к среде разработки программ. Это означает, что можно легко создавать новые элементы управления и легко использовать их как встроенную часть компонентов среды разработки.

С# не поддерживает множественное наследование, в отличие от C++. С# поддерживает интерфейсы похожим на Java образом. Как и в Java, интерфейсы определяют код, который должен быть написан, а не сам повторно используемый код. С # не поддерживает шаблоны или параметризованные типы.

## 1.2. Среда разработки информационной системы

При разработке информационной системы была использована интегрированная среда разработки Microsoft Visual Studio Community 2019. Microsoft Visual Studio Community 2019 – это среда разработки от Microsoft. В Microsoft существуют различные версии Visual Studio, начиная от бесплатных, которые предназначены для учащихся, разработчиков открытого программного обеспечения (ПО) и заканчивая платными, предназначенными для профессиональной командной разработки. Все версии предоставляют основные функции, такие как текстовый редактор, инструменты сборки и отладчик, а также инструменты визуального редактирования для пользовательских интерфейсов. NET-система, которую использует Visual Studio, доступна из командной строки, поэтому для работы можно использовать любой текстовый редактор. Visual Studio предназначена для разработки программ на базе платформы .NET.

ИСР Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Среда разработки Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов или инструментов для прочих аспектов процесса разработки программного обеспечения. [18]

При разработке приложения работа ведётся в контексте определённого проекта. Под проектом понимается совокупность всех файлов и конфигурационной информации, необходимых для создания конечного программного продукта: исполняемого файла (с расширением «exe») или библиотеки (с расширением «dll»).

В состав проекта приложения, разрабатываемого на языке C# в среде Visual C#, входят:

* основной файл проекта (файл с расширением «csproj»);
* файлы с текстом программы на языке С# (файлы с расширением «cs»);
* другие файлы, необходимые для разработки приложения.

Совокупность всех файлов, образующих проект приложения, следует рассматривать как единое целое. При переносе проекта из одного каталога в другой должны быть перенесены все файлы, входящие в состав проекта. Для удобства работы все файлы, относящиеся к проекту, сохраняются в отдельном каталоге.

Кроме понятия «проект» при разработке приложений используется также и понятие «решение». Так же как проект представляет собой совокупность файлов, необходимых для работы над этим проектом, решение представляет собой совокупность нескольких проектов. В простых приложениях решение состоит только из одного проекта. В более сложных случаях решение может включать в себя несколько связанных друг с другом проектов.

При создании проекта одновременно с ним создаётся и решение, в состав которого включается этот проект. В общем случае имя решения может не совпадать с именем любого из проектов, входящего в его состав. Но для простых решений, состоящих только из одного проекта, рекомендуется выбирать имя, совпадающее с именем проекта (такое имя решения при создании нового проекта предлагается средой разработки по умолчанию).

Рабочее окно Visual Studio состоит из нескольких элементов: панель меню, стандартная панель инструментов, различные окна инструментов, закрепленные или автоматически скрытые слева, снизу и справа, а также пространство редактора. Доступные окна инструментов, меню и панели инструментов зависят от типа проекта или файла, в котором производится работа. Основной частью интерфейса среды разработки является окно редактора кода, которое предназначено для редактирования текста программы. Во время работы над приложением в редакторе кода может быть открыто любое количество файлов, которые располагаются на отдельных вкладках и между которыми можно переключаться произвольным образом [8].

Кроме вкладок редактора кода, в которых выполняется редактирование текста программы, в среде разработки могут отображаться вкладки, предназначенные для решения других задач. Например, после запуска среды разработки по умолчанию отображается вкладка Начальная страница, на которой представлена полезная информация, ссылки на разделы справочной системы и список недавно открывавшихся проектов и решений.

Первое, что нужно сделать при работе в Microsoft Visual Studio Community 2019, при необходимости создать приложение, это создать проект. Его можно создать как на начальной странице, так и с помощью меню.

В этом окне можно выбрать подходящий шаблон, в зависимости от того, какое приложение вы хотите создать, а также имя и местоположение для вашего проекта и решения.

Наиболее распространенными приложениями являются:

* Приложение Windows Form – приложение с графическим интерфейсом на основе платформы .NET.
* Консольное приложение – приложение, работающее в режиме текстового интерфейса и использующее при вводе-выводе и представлении информации исключительно набор буквенно-цифровых символов и символов псевдографики.
* Приложение WPF – приложение с графическим интерфейсом и векторной системой визуализации, не зависящей от разрешения устройства вывода, на основе платформы .NET, но в отличие от Windows Forms приложния WPF может иметь структуру навигации.
* Веб-приложение ASP.NET – приложение, которое основывается на протоколе HTTP и использует его правила взаимодействия между сервером и браузером.

Microsoft Visual Studio Community 2019 помогает работать с несколькими связанными проектами с помощью решения. Решение – это набор проектов, которые обычно связаны между собой, но необязательно. Любой нетривиальный проект будет иметь несколько файлов исходного кода, а в Microsoft Visual Studio Community 2019 они будут принадлежать проекту. Каждый проект создает один вывод или цель. Цель сборки может быть такой же простой, как и один файл: например, проект C# может создавать исполняемый файл или библиотеку, но некоторые проекты создают более сложные результаты. Некоторые типы проектов создают веб-сайты. Обычно веб-сайт содержит несколько файлов, но в совокупности эти файлы представляют собой единый объект: один веб-сайт. Каждый проект обычно развертывается как единое целое, даже если он состоит из нескольких файлов.

Панель инструментов содержит все необходимые элементы управления, которые мы можем использовать при разработке пользовательского интерфейса. Чтобы использовать их в пользовательском интерфейсе, необходимо открыть графический конструктор и переместить элементы на форму.

Интегрированная среда разработки Microsoft Visual Studio Community 2019 имеет разные редакторы и инструменты проектирования. Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода.

IntelliSense — технология автодополнения Microsoft, наиболее известная в Microsoft Visual Studio. Дописывает название функции при вводе начальных букв. Кроме прямого назначения, IntelliSense используется для доступа к документации и для устранения неоднозначности в именах переменных, функций и методов.

## 1.3. Современные технологии разработки веб-приложений. Особенности платформы ASP.NET Core

В настоящее время веб-приложения получили большое развитие в различных сферах деятельности общества. Работа веб-приложения производится посредством клиент-серверной технологии, где клиентом является браузер, а в качестве сервера выступает веб-сервер. Для начала приведем общую схему работы веб-приложений, которая описана ниже на рис. 1.

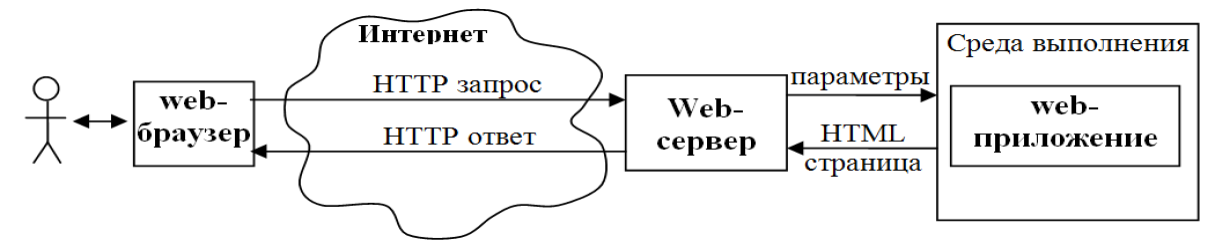


Рисунок 1 – Принцип работы веб-приложения

Из рис. 1 видно, что клиент, обращаясь к веб-браузеру, отправляет HTTP-запрос по определенному URL адресу, указывающему на некоторый динамический ресурс, а именно само веб-приложение. Далее сервер формирует на основе веб-приложения HTML-страницу, которая посредством браузера отображается клиенту. Из описания схему можно сделать вывод, что основная работа веб-приложения осуществляется на стороне сервера.

В настоящий момент существует множество технологий, реализующих логику веб-приложений на стороне сервера. Первой широко используемой технологией стала CGI (Common Gateway Interface), которая особенно применима для создания динамических веб-страниц и служит для обеспечения связи между клиентом (веб-браузером) и веб-сервером. Данная технология представляет собой набор правил, следуя которым, программа способна выполняться на различных серверах операционных системах. В соответствии с технологией CGI, НТТР запрос, содержащий ссылку на динамическую страницу, поступая на веб-сервер, генерирует новый процесс и запускает нужную прикладную программу. Технология CGI позволяет использовать любой язык программирования, способный работать с устройствами ввода/вывода. Также в разра-ботке веб-приложения можно использовать CGI скрипты, например Python, Perl, Tcl и т. д. Если в CGI программе содержатся скрипты, то при ее выполнении вызывается script engine (интерпретатор скриптов), которому передаются данные НТТР запроса и имя файла, содержащего запрашиваемый скрипт. После выполнения данного скрипта программой клиенту возвращается сформированная HTML страницу.

Несмотря на то, что технология CGI позволяет достаточно просто формировать динамическим образом информацию в веб-сети, она имеет значительные недостатки. Одним из главных недостатков является производительность. Причиной этого низкой производительности является сам процесс обработки НТТР запроса: для каждой обработки такого запроса веб-сервер генерирует новый процесс, который заканчивает свою работу только после завершения программы, что является достаточно трудоемким и при наличии множества таких процессов, начинается конкуренция за ресурсы оперативной памяти.

Следующей технологией, которая довольно широко распространена, стала Java Servlets или просто сервлеты. Данная технология позволяет решить проблему производительности путем выполнения всех запросов в одном процессе путем распределения их по потокам внутри процессов. Это означает, что программный код сервлета должен быть потоко-безопасным. Также плюсом использования сервелетов является их независимость от платформы, потому что они выполняются на виртуальной Java-машине. Java Servlets обладает широким функционалом, который достижим благодаря большому количеству библиотек. Сервлет-контейнер запускает сервлет, который является классом Java, при первом же обра-щении к нему или при специальном указании.

Java Servlet Pages (JSP) – это надстройка над технологией Java Servlets, разработанной ранее компанией Sun Microsystems, что означает, что их архитектуры взаимосвязаны. Данная технология обеспечивает быструю и упрощенную разработку web-приложений с применением шаблонного похода. Шаблоны страниц JSP представляют собой шаблоны HTML страниц и крайне схожи с шаблонами ASP и PHP. Отличием данной технологии от других похожих является то, что при обращении к странице код, находящийся внутри тегов, не интерпретируется, а предварительно компилируется в Java Servlet, то есть так, как если бы он располагалась внутри сервлета. Данная процедура производится один раз при первом запуске к странице или сервлет-контейнера, т. к. выполнение вышеописанной операции является трудоемкой. Так же, как и Java Servlets, JSP не привязана к конкретной программной или аппаратной платформе. JSP гармонично сочетает шаблонную реализацию страниц и все имеющиеся преимущества платформы Java.

Новейшей технологией разработки web-приложений является .NET технологий, разработанная компанией Microsoft. Платформа .NET значительно упростила процесс разработки приложений и повысила надежность кода. Стали доступными функции автоматического управления временем жизни объектов, обработка исключений и их отладка, в наличии появились библиотеки, нейтральные к языкам программирования. Набор стандартных базовых классов обеспечивают разработчику доступ к сервисам платформы при использовании любых языков программирования, совместимых с .NET. Common Language Runtime совместно с базовыми классами составляют основу платформы .NET и предлагает разработчикам высокоуровневые сервисы, такие как ADO.NET (усовершенствованный ADO, используемой SOAP и XML с целью обмена данными), ASP.NET (новое поколение ASP, дающий возможность использовать любой язык программирования, совместимый с .NET). Компилирование исходного кода происходит по следующей схеме: создается код на промежуточном языке (Microsoft Intemediate Language). В отличие от старой версии, где компилятор создавал машинный код, данный вид компиляции позволяет скомпилированному файлу выполняться на платформе любого процесса. Новые возможности ASP.NET соответствуют современным требованиям. Вот лишь некоторые из них: большой набор библиотек, языковая независимость платформы, новые пути обработки ошибок и т. п. Каждая из вышеперечисленных платформ имеет возможности и ограничения в индивидуальном порядке, а также свою собственную область применения, что предоставляет разработчику широкий выбор инструментов разработки.

Поскольку технологией разработки проекта выбрана технология .NET Core, проведем обзор сравнительный обзор этой технологии. Платформа ASP.NET Core представляет технологию от компании Microsoft, предназначенную для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов.

С одной стороны, ASP.NET Core является продолжением развития платформы ASP.NET. Но, с другой стороны, это не просто очередной релиз. Выход ASP.NET Core фактически означает революцию всей платформы, ее качественное изменение.

Разработка над платформой началась еще в 2014 году. Тогда платформа условно называлась ASP.NET vNext. В июне 2016 года вышел первый релиз платформы. А в декабре 2019 года вышла версия ASP.NET Core 3.1, которая, собственно, и будет охвачена в текущем руководстве.

ASP.NET Core теперь полностью является opensource-фреймворком. Все исходные файлы фреймворка доступны на GitHub.

ASP.NET Core может работать поверх кроссплатформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS, Linux. И таким образом, с помощью ASP.NET можно создавать кроссплатформенные приложения. И хотя Windows в качестве среды для разработки и развертывания приложения до сих пор превалирует, но теперь уже разработчики не ограничены только этой операционной системой. Это значит, что можно запускать веб-приложения не только на ОС Windows, но и на Linux и Mac OS. А для развертывания веб-приложения можно использовать традиционный IIS, либо кроссплатформенный веб-сервер Kestrel.

Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget. Кроме того, в отличие от предыдущих версий платформы нет необходимости использовать библиотеку System.Web.dll.

ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages. В предыдущих версиях платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности. Сейчас же они объединены в одну программную модель ASP.NET Core MVC.

Кроме объединения вышеупомянутых технологий в одну модель в MVC был добавлен ряд дополнительных функций. Одной из таких функций являются тэг-хелперы (tag helper), которые позволяют более органично соединять синтаксис html с кодом С#.

ASP.NET Core характеризуется расширяемостью. Фреймворк построен из набора относительно независимых компонентов. И разработчик может либо использовать встроенную реализацию этих компонентов, либо расширить их с помощью механизма наследования, либо вовсе создать и применять свои компоненты со своим функционалом.

Также было упрощено управление зависимостями и конфигурирование проекта. Фреймворк теперь имеет свой легковесный контейнер для внедрения зависимостей, и больше нет необходимости применять сторонние контейнеры, такие как Autofac, Ninject. Хотя при желании их также можно продолжать использовать.

В качестве инструментария разработки мы можем использовать последние выпуски Visual Studio, начиная с версии Visual Studio 2015. Кроме того, можно создавать приложения в среде Visual Studio Code, которая является кроссплатформенной и может работать как на Windows, так и на Mac OS X и Linux.

Для обработки запросов теперь используется новый конвейер HTTP, который основан на компонентах Katana и спецификации OWIN. А его модульность позволяет легко добавить свои собственные компоненты.

Если суммировать, то можно выделить следующие ключевые отличия ASP.NET Core от предыдущих версий ASP.NET:

* Новый легковесный и модульный конвейер HTTP-запросов
* Возможность развертывать приложение как на IIS, так и в рамках своего собственного процесса
* Использование платформы .NET Core и ее функциональности
* Распространение пакетов платформы через NuGet
* Интегрированная поддержка для создания и использования пакетов NuGet
* Единый стек веб-разработки, сочетающий Web UI и Web API
* Конфигурация для упрощенного использования в облаке
* Встроенная поддержка для внедрения зависимостей
* Расширяемость
* Кроссплатформенность: возможность разработки и развертывания приложений ASP.NET на Windows, Mac и Linux
* Развитие как open source, открытость к изменениям

Эти и другие особенности и возможности стали основой для новой модели программирования.

Также следует выделить цели, которые достигнуты путем внедрения .NET Core:

* Межплатформенная поддержка. Платформа .NET Core способна функционировать в средах Windows, Linux и macOS. Приложения .NET Core могут строиться на разных платформах с помощью Visual Studio Code или Visual Studio для Mac. Кроме того, включение Xamarin добавляет iOS и Android к числу платформ, поддерживаемых для развертывания.
* Производительность. Производительность .NET Core постоянно приближается к вершине всех важных графиков производительности, и в каждый выпуск вносятся дополнительные усовершенствования.
* Переносимые библиотеки классов, пригодные к потреблению всеми исполняющими средами .NET. В .NET Core введен стандарт .NET Standard — формальная спецификация, устанавливающая согласованное поведение исполняющей среды .NET.
* Переносимое или автономное развертывание. Приложения .NET Core могут развертываться параллельно с инфраструктурой либо использовать установленную копию .NET Core на уровне машины.
* Полная поддержка командной строки. Платформа .NET Core сосредоточена на полной поддержке командной строки в качестве основной цели.
* Открытый код. Как упоминалось, платформа .NET Core и ее документация являются системами с открытым кодом, укомплектованными запросами на изменение от мирового сообщества разработчиков.
* Возможность взаимодействия с полной платформой .NET Framework. Выпуск 2.0 платформы .NET Core разрешает ссылаться на библиотеки .NET Framework.

Примеры рабочих окон создания проекта .NET Core в среде Visual Studio 2019 показаны на рис. 2 - 4.

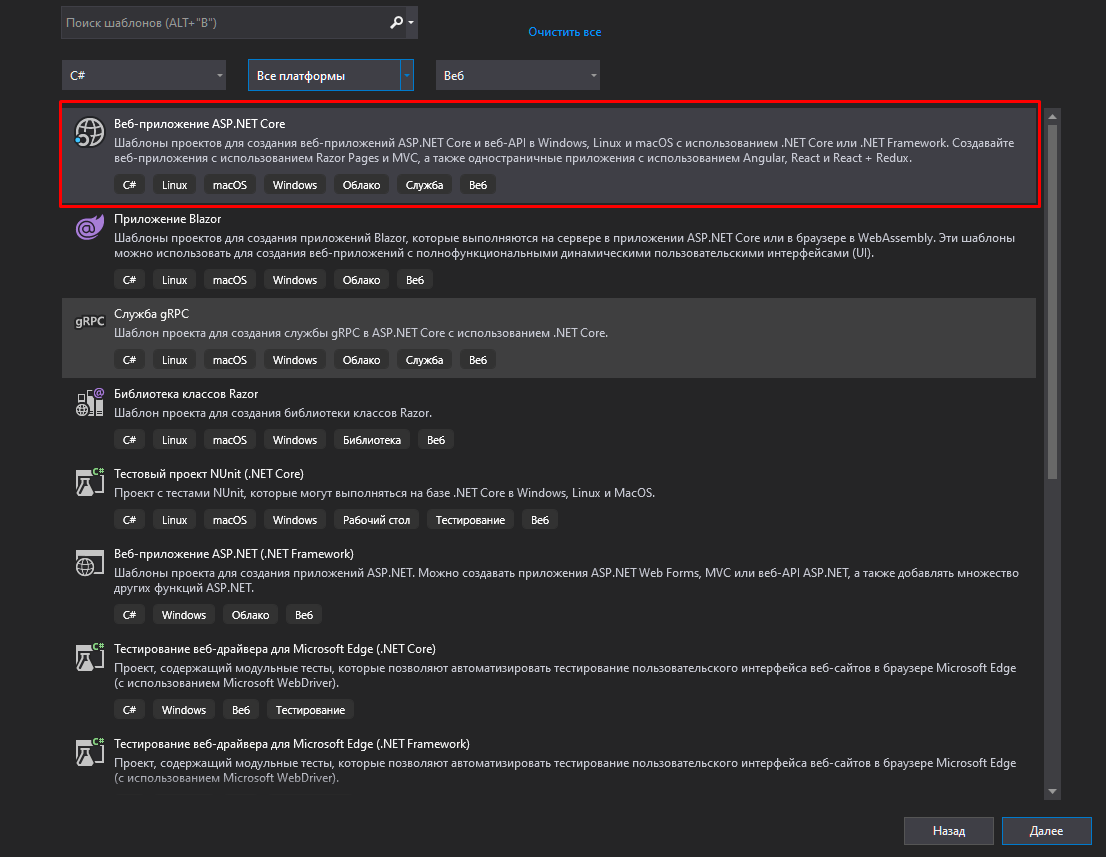


Рисунок 2 – Выбор вида проекта

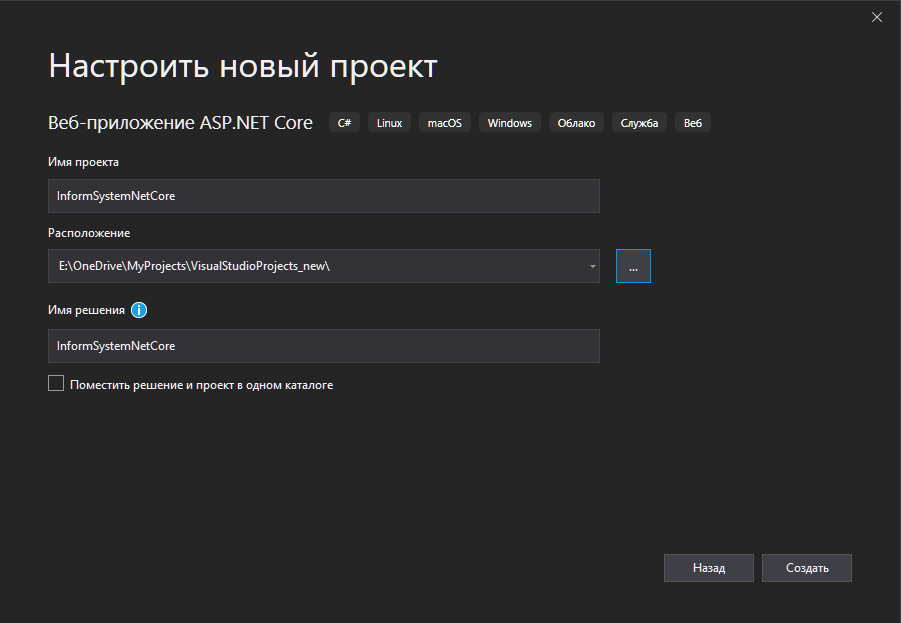


Рисунок 3 – Настройка проекта

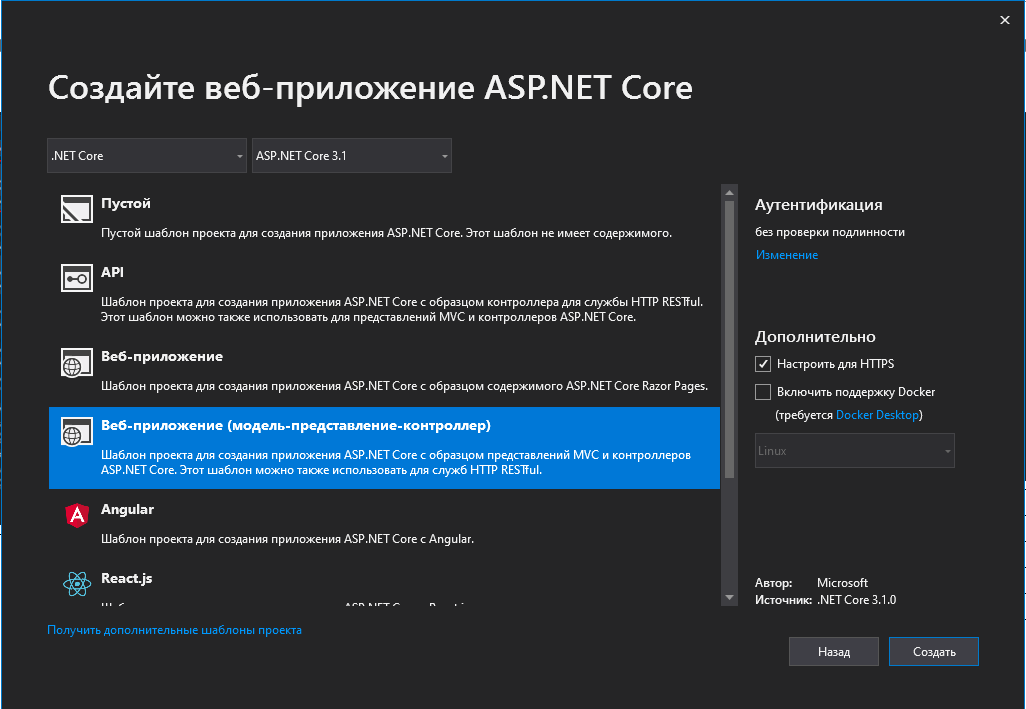


Рисунок 4 – Создание веб-приложения ASP.NET Core

Далее рассмотрим состав .NET Core. В общих терминах платформа образована из четырех основных частей:

* исполняющая среда .NET Core;
* набор библиотек инфраструктуры;
* инструменты SDK и хост приложений dotnet;
* компиляторы языков.

Исполняющая среда .NET Core (CoreCLR). Это базовая библиотека для .NET Core. Она включает сборщик мусора, компилятор JIT, базовые типы .NET и множество низкоуровневых классов. Исполняющая среда предоставляет мост между библиотеками инфраструктуры .NET Core (CoreFX) и лежащими в основе ОС. В ее состав включены только типы, которые имеют строгую зависимость от внутренней работы исполняющей среды. Большая часть библиотеки классов реали-зована в виде независимых пакетов NuGet.

При проектировании CoreCLR разработчики пытались свести к минимуму объем реализованного кода, оставляя специфические реализации многих классов инфраструктуры на CoreFX. В результате получилась небольшая гибкая кодовая база, которую можно модифицировать и быстро развертывать для исправления дефектов или добавления функциональных средств. Сама по себе среда CoreCLR делает не особо много работы. Любой определенный в ней библиотечный код скомпилирован в сборку System.Private.CoreLib.dll, которая не предназначена для потребления за преде-лами CoreCLR или CoreFX.

В состав дополнительного инструментария, предлагаемого CoreCLR, входит ILDASM и ILASM (версии .NET Core программ, которые вы применяли ранее в книге), а также хост тестирования — небольшая оболочка для запуска DLL-библиотек IL из командной строки.

Библиотеки инфраструктуры (CoreFX). Это набор фундаментальных библиотек, включающий классы для коллекций, файловых систем, консоли, разметки XML, асинхронной работы и многих других элементов. Библиотеки инфраструктуры построены поверх CoreCLR и предоставляют для других инфраструктур интерфейсные точки в исполняющую среду. Помимо специфических реализаций CoreCLR, содержащихся в CoreFX, остальные библиотеки имеют независимую от исполняющей среды и платформы природу. В CoreCLR находится mscorlib.dll — открытый фасад CoreCLR. Вместе CoreCLR и CoreFX образуют .NET Core.

Инструменты SDK и хост приложений dotnet. Включенные в SDK инструменты представляют собой интерфейс командной строки (command-line interface — CLI) платформы .NET, используемый для построения приложений и библиотек .NET Core. Хост приложений dotnet является универсальным драйвером для выполнения команд CLI, включая приложения .NET Core.

Команды CLI и приложения .NET Core запускаются с применением хоста приложе-ний dotnet. В дополнение к встроенным командам CLI инфраструктуры могут добавлять собственные команды.

Компиляторы языков. Платформа .NET Compiler (“Roslyn”) предоставляет компиляторы с открытым кодом языков C# и Visual Basic с развитыми API-интерфейсами для анализа кода. Язык C# считается основным и поддерживается во всех инфраструктурах .NET Core.

# Список использованных источников

1. Зашеловский А. Е. Среды разработки программного обеспечения, ориентированные на процессы [Электронный ресурс] / А. Е. Зашеловский, О. Ф. Абрамова. – Режим доступа: <http://www.scienceforum.ru/2015/pdf/15128.pdf>. (дат обращения: 02.12.2019).
2. Зиборов В. Visual C# 2012 на примерах / В. Зиборов. – СПб.: БХВПетербург, 2013. ‒ 473 с.
3. Косенко В. В. Популярные языки программирования с позиций системного программирования / В. В. Косенко // Информационные технологии и вычислительные системы. – 2013. - № 1. – С. 54-59
4. Либерти Д. Программирование на C#. Создание .NET приложений. Программирование на C# / Д. Либерти. – М.: Бином 2010. ‒ 684 с.
5. Рихтер Джеффри. CLR via C# программирование на платформе Microsoft .Net Framework 4.0 на языке C# / Д. Рихтер – Издательство Питер, 3-е издание, 2012. – 928 с.