**2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ**

Полностью изучив и проанализировав теоретическую часть разрабатываемой системы, получаем список требований необходимых для эффективного и стабильного функционирования разрабатываемого новостного портала. Исходя из полученных требований в целях обеспечения гибкой архитектуры разбиваем систему на функциональные блоки.

В разрабатываемом веб-приложении можно выделить следующие блоки:

* блок администрирования;
* блок взаимодействия с пользователями;
* блок работы сервера;
* блок классов взаимодействия с базой данных;
* блок пользовательского интерфейса;
* блок авторизации пользователя;
* блок реляционной базы данных;
* блок связи подсистем приложения.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.111 C1.

Рассмотрим каждый структурный блок подробнее.

**2.1 Блок администрирования**

Блок администрирования –это часть системы, отвечающая за управление всеми ресурсами сайта. В данном проекте за блок администрирования отвечает Microsoft Identity. Microsoft Identity предоставляет возможность назначать зарегистрированным пользователям роли, с помощью которых каждому отдельному пользователю предоставляются доступные для данной роли действия. Для роли admin в пользовательском интерфейсе открываются дополнительные опции для доступа ко всем ресурсам приложения, а также появляется возможность для проведения любых манипуляций с данными.

**2.2 Блок взаимодействия с пользователями**

Блок взаимодействия с пользователями – это блок, основной задачей которого является отвечать на запросы пользователя. Для реализации данного блока используется библиотека SignalR.

ASP.NET SignalR - это библиотека для ASP.NET разработчиков. Данная библиотека упрощает добавление в приложения компонентов, работающих в реальном времени. Функциональность, работающая в реальном времени – это способность сервера отдать свежие данные подключенным клиентам немедленно, вместо того, чтобы ждать пока клиенты запросят эти данные.

SignalR может быть использован для добавления в ASP.NET приложения любого вида веб-функциональности, работающей в реальном времени.

SignalR имеет простой API для вызовов удаленных процедур от сервера к клиенту (RPC server-to-client), которые вызывают Javascript функции в клиентских браузерах из кода .NET сервера. SignalR также имеет API для управления соединениями и группировкой соединений.

SignalR управляет соединениями автоматически, и отсылает сообщения всем подключенным клиентам одновременно, как в чате.

SignalR приложения могут масштабироваться на тысячах клиентах, используя Service Bus, SQL Server или Redis.

SignalR поставляется с открытым кодом, который доступен на GitHub.

**2.3 Блок работы сервера**

Блок веб-сервера отвечает за обрабатывание запросов, отправленных клиентом, в качестве серверной части будет использоваться ASP.NET.

ASP.NET — это веб-платформа с открытым исходным кодом для создания веб-приложений на платформе .NET. Она создана Microsoft, и в 2002 году была выпущена версия 1.0, позволяющая разработчикам создавать динамические веб-приложения, службы и сайты.

ASP.NET является преемником технологии ASP (Active Server Pages) и представляет собой значительное обновление с точки зрения гибкости и мощности. Это расширение платформы .NET с дополнительными инструментами и библиотеками, специально предназначенными для создания вещей в Интернете, включая веб-приложения и веб-сайты.

ASP.NET основывается на Common Language Runtime: разработчики могут писать код для ASP.NET, используя практически любые языки программирования, некоторые из которых входят в комплект .NET Framework (C#, Visual Basic.NET и JScript .NET), а другие могут быть установлены дополнительно (IronRuby, IronPython, PHP, Perl, Smalltalk, Haskell и др.).

Некоторые особенности ASP.NET:

Компилируемый код выполняется быстрее, а большинство ошибок отлавливается ещё на стадии разработки.

Расширяемый набор элементов управления и библиотек классов, ускоряющий разработку.

Возможность кэширования всей страницы, её частей или данных, используемых на странице.

Возможность разделения визуальной части и бизнес-логики по разным файлам, есть возможность выделять часто используемые шаблоны пользовательских элементов управления, таких как меню сайта, наличие master-страниц для задания шаблонов оформления, поддержка AJAX (расширение ASP.NET AJAX).

Расширяемые модели событий, обработки запросов и серверных элементов управления.

Поддержка CRUD-операций при работе с таблицами через GridView.

ASP.NET включает в себя базовые библиотеки из платформы .NET, а также библиотеки для распространенных веб-шаблонов. Одной из таких библиотек является Model View Controller (MVC), которая позволяет использовать шаблон проектирования MVC для разработки веб-приложений и сайтов. Шаблон MVC позволяет создать веб-приложение, состоящее из трех ролей — бизнес-уровня, уровня отображения и контроля ввода.

**2.4 Блок классов взаимодействия с базой данных**

Взаимодействие с базой данных производится через классы DataContext и Migrations.

Migrations — это удобный способ изменить базу данных структурированным и организованным образом. Можно редактировать фрагменты SQL вручную.  Миграции поддерживаются всеми бэкендами, если они запрограммированы на поддержку изменения схемы.

Класс Migration обрабатывает детали выполнения миграции за пользователя - перебор исходных записей, создание целевых объектов и отслеживание взаимосвязей между ними.

Класс DataContext обрабатывает подключение к базе данных. Он также обрабатывает запросы, обновления, вставки в базу данных, отслеживает идентичность, отслеживает изменения, обрабатывает их, обеспечивает целостность транзакций и даже создание базы данных. Класс DataContext транслирует запросы сущностных классов в операторы SQL, которые выполняются на подключенной базе данных.

DataContext является источником всех сущностей, отображаемых через соединение с базой данных. Он отслеживает изменения, внесенные вами во все извлеченные сущности, и поддерживает «кэш удостоверений», который гарантирует, что сущности, извлеченные более одного раза, представлены с использованием одного и того же экземпляра объекта.

**2.5 Блок пользовательского интерфейса**

Блок пользовательского интерфейса является клиентской частью веб-приложения. Для реализации данного блока используется фрэймворк Angular.

Angular — это не просто фреймворк, а скорее платформа, которая позволяет разработчикам создавать приложения.

Фреймворк Angular содержит множество библиотек, некоторые из которых являются основными.

Angular активно поддерживается и имеет большое сообщество и экосистему.  Можно найти множество материалов поэтому фреймворку, а также множество полезных сторонних инструментов.

Angular предоставляет не только инструменты, но и шаблоны проектирования для создания проекта в удобном для сопровождения виде. Angular построен на TypeScript, который, в свою очередь, опирается на JS ES6.

Angular стремился устранить тесную связь между различными компонентами приложения. Внедрение происходит в стиле NodeJS.

Angular предназначен для тщательного тестирования и поддерживает как модульное, так и сквозное тестирование с помощью таких инструментов, как Jasmine и Protractor.

**2.6 Блок авторизации пользователя**

Блок авторизации пользователяявляется частью системы, которая отвечает за проверку существования пользователя и в случае его существования в системе, генерирует авторизационный токен для этого пользователя. Для данных целей будет использоваться Microsoft Identity

Платформа Microsoft Identity состоит из нескольких компонентов:

1. Служба аутентификации, совместимая со стандартами OAuth 2.0 и OpenID Connect, позволяющая разработчикам аутентифицировать несколько типов удостоверений, в том числе:

* личная учетная запись Microsoft, такая как Skype, Xbox и Outlook.com;
* рабочие или учебные учетные записи, подготовленные через Azure AD;
* социальные или локальные учетные записи с помощью Azure AD B2C.

1. Библиотеки с открытым исходным кодом: библиотеки проверки подлинности Microsoft (MSAL) и поддержка других библиотек, соответствующих стандартам.
2. Портал управления приложениями: возможность регистрации и настройки на портале Azure, а также другие возможности управления Azure.
3. API конфигурации приложений и PowerShell: программная настройка приложений с помощью API Microsoft Graph и PowerShell, позволяющая автоматизировать задачи DevOps.
4. Контент для разработчиков: техническая документация, включая краткие руководства, учебные пособия, практические руководства и примеры кода.

Для разработчиков платформа Microsoft Identity предлагает интеграцию современных инноваций в области идентификации и безопасности, таких как проверка подлинности без пароля, поэтапная проверка подлинности и условный доступ.

С помощью платформы Microsoft Identity можно написать код один раз и получить доступ к любому пользователю. Можно создать приложение один раз, и оно будет работать на многих платформах, или создать приложение, которое будет функционировать как клиент, а также ресурсное приложение (API).

**2.7 Блок реляционной базы данных**

Блок базы данныхвключает данные, используемые веб-приложением. При реализации используется реляционная база данных PostgreSQL.

PostgreSQL поддерживает транзакции со свойствами ACID. Это означает, что транзакции должны поддерживать четыре атрибута:

Атомарность - транзакции считаются завершенными единицами; транзакция может либо полностью завершиться успешно, либо полностью провалиться - в случае неудачи состояние базы данных остается неизменным.

Непротиворечивость - база данных между транзакциями может существовать только в допустимом состоянии; все данные, записываемые в базу данных, должны соответствовать существующим ограничениям, триггерам, каскадам и связанным с ними комбинациям.

Изоляция - функция контроля параллелизма — гарантирует, что данные не будут повреждены незаконными или параллельными транзакциями, поскольку транзакции обрабатываются так, как если бы они происходили последовательно.

Долговечность - гарантирует, что транзакция остается зафиксированной даже в случае сбоя системы - обычно завершенные транзакции записываются, например, в [журнал упреждающей записи](https://en.wikipedia.org/wiki/Write-ahead_logging) .

Основными достоинствами PostgreSQL являются:

* надежность;
* производительность (основывается на использовании индексов, интеллектуальном планировщике запросов, тонкой системы блокировок, системе управления буферами памяти и кэширования, превосходной масштабируемости при конкурентной работе);
* расширяемость (означает, что пользователь может настраивать систему путем определения новых функций, агрегатов, типов, языков, индексов и операторов);
* поддержка SQL;
* поддержка JSON;
* богатый набор типов данных;
* простота использования.

**2.8 Блок связи подсистем приложения**

Блок связи подсистем реализован с помощью шаблона Data Transfer Object (DTO).

Data Transfer Object (DTO) - один из шаблонов проектирования, используется для передачи данных между подсистемами приложения.

Data Transfer Object, в отличие от business object или data access object не должен содержать какого-либо поведения.

DTO — это класс, представляющий некоторые данные без какой-либо логики. DTO обычно используются для передачи данных между разными приложениями или разными уровнями в одном приложении. Можно смотреть на них как на пакеты с информацией, единственной целью которых является просто передать эту информацию получателю.  
 Классы DTO используются для сериализации / десериализации данных из разных источников. Сериализатор – это технология для загрузки объектов DTO.