Техническое задание

Персональный блогхост на платформе ASP.NET Core

Список ответственных исполнителей

Организация, должность	**Фамилия, имя, отчество**
УО «БГУИР», Студент	Баранович П. Л.

СОДЕРЖАНИЕ

Введение 4
1 Описание предметной области 5
2 Термины и определения 7
2.1 Архитектурные паттерны 8
2.2 Инструменты разработки 6
2.3 Языки программирования 9
2.4 База данных 10
3 Общие положения 12
3.1 Назначение документа 12
3.2 Цели создания системы 12
3.3 Основные функциональные возможности системы 12
3.4 Плановые сроки выполнения работ 12
3.5 Использование технического задания 12
4 Функциональные требования 13
4.1 Архитектура приложения 13
4.2 Описание вариантов использования 14
4.2.1 ВИ «Найти блог по названию» 14
4.2.2 ВИ «Найти статью по названию» 14
4.2.3 ВИ «Открыть страницу статьи для просмотра» 15
4.2.4 ВИ «Открыть страницу блогов для просмотра» 15
4.2.5 ВИ «Создать статью» 15
4.2.6 ВИ «Редактировать статью» 16
4.2.7 ВИ «Удалить статью» 16
4.2.8 ВИ «Создать блог» 16
4.2.9 ВИ «Редактировать блог» 17
4.2.10 ВИ «Удалить блог» 17
4.2.11 ВИ «Поставить лайк» 17
4.2.12 ВИ «Написать комментарий» 18
4.3 Система ролей 18
4.4 Диаграмма базы данных 19
5 Нефункциональные требования 20
5.1 Интерфейс пользователя 20
5.2 Отображение на различных ОС 20
5.3 Документация 20
5.4 Порядок разработки 20
6 Перспективы развития 21

ВВЕДЕНИЕ

Рост востребованности веб-приложений неоспорим, и изменяется пропорционально количеству пользователей сети интернет. Это уже не обычные статические приложения – современный набор JavaScript подобных фреймворков позволяет создавать приложения, почти не уступающие нативным.

Интернет превратился из набора сайтов для развлечений в полноценную платформу для развёртки и ведения бизнеса. Он позволяет создавать приложения с общим доступом, создавать программные продукты любой сложности без необходимости установки на устройстве, кроме, непосредственно, инструмента для просмотра веб страниц, так как популярные браузеры Chrome, Firefox, Edge и другие.

Так же это сейчас лидирующая образовательная платформа, которая давно потеснила традиционные библиотеки, а в чём-то начинает обгонять лекционные занятие в университете.

Некоторые полагают, что образовательные порталы мешают получать детям знания, так как они просто скачивают нужный материал, не изучая его. Но это мнение неверно. Благодаря порталам ученик меньше времени тратит на поиск нужной информации и у него остается много времени на изучение и усвоение материала. Всемирная паутина создает идеальные условия для обучения. Поэтому необходимо объединение процесса образования с сетью.

В рамках данного проекта предполагается реализация приложения на тему «Персональный блогхост на платформе ASP.NET Core».

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Веб-приложение — клиент-серверное приложение, в котором клиентом выступает браузер, а сервером — веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами. Веб-приложения стали широко популярными в конце 1990-х — начале 2000-х годов.

Существенное преимущество построения веб-приложений для поддержки стандартных функций браузера заключается в том, что функции должны выполняться независимо от операционной системы данного клиента. Вместо того, чтобы писать различные версии для Microsoft Windows, Mac OS X, GNU/Linux и других операционных систем, приложение создаётся один раз для произвольно выбранной платформы и на ней разворачивается. Однако различная реализация HTML, CSS, DOM и других спецификаций в браузерах может вызвать проблемы при разработке веб-приложений и последующей поддержке. Кроме того, возможность пользователя настраивать многие параметры браузера (например, размер шрифта, цвета, отключение поддержки сценариев) может препятствовать корректной работе приложения.

Но стоит учесть, что современные инструменты для создания веб-приложений, а также библиотеки позволяют минимизировать кросс-браузерные различия в написании приложения. (Babel, Webpack и другие).

Другой (менее универсальный) подход заключается в использовании Adobe Flash, Silverlight или Java-апплетов для полной или частичной реализации пользовательского интерфейса. Поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью плагинов), Flash- или Java-приложения могут выполняться с легкостью. Так как они предоставляют программисту больший контроль над интерфейсом, они способны обходить многие несовместимости в конфигурациях браузеров, хотя несовместимость между Java- или Flash-реализациями на стороне клиента может приводить к различным осложнениям. И стоит заметить, что данный тренд пришёл в состояние упадка. Многие компании с мировым именем (Apple, Google, и другие), стали ограничивать использования данных технологий в своих браузерах, так как они признаны небезопасными и плохо оптимизированными.

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер».

Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него.

Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP или иного протокола.

Само веб-приложение может выступать в качестве клиента других служб, например, базы данных или другого веб-приложения, расположенного на другом сервере. Ярким примером веб-приложения является система управления содержимым статей Википедии: множество её участников могут принимать участие в создании сетевой энциклопедии, используя для этого браузеры своих операционных систем (будь то Microsoft Windows, GNU/Linux/MacOS или любая другая операционная система) и не загружая дополнительных исполняемых модулей для работы с базой данных статей.

В настоящее время набирает популярность новый подход к разработке веб-приложений, называемый Ajax. При использовании Ajax страницы веб-приложения не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными.

Так же стоит упомянуть про принцип работы технологии WebSocket – которая не требует постоянных запросов от клиента к серверу, а создает двунаправленное соединение, при котором сервер может отправлять данные клиенту, без запроса от последнего. Таким образом появляется возможность динамически управлять контентом в режиме реального времени.

Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые языки программирования, но большинство их них реализует одну из двух стандартных паттернов проектирования – WebAPI и MVC.

Как уже обговаривалось ранее, современные возможности всемирной паутины позволяют разрабатывать универсальные, кроссплатформенные полноценные рабочие инструменты для ведения бизнеса. Приложения, написанные с использованием современных паттернов и Фреймворков, могут вовсе не уступать нативным, а общий доступ и параллельная работа с контентом делают их незаменимым инструментом для ведения бизнеса.

2 ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

2.1 Архитектурные паттерны

Паттерн проектирования Repository

«Repository» — один из наиболее часто используемых паттернов проектирования при работе с базами данных (БД). Он позволяет отделить программную логику, работающую непосредственно с БД, от всей остальной программы выступая посредником меду ними с помощью интерфейса во многом схожего с коллекциями.

Преимущество использования паттерна «Repository»:

- Отделение программной логики, работающей непосредственно с БД от всей остальной программы. Если требуется изменить логику работы с БД или даже тип БД (например, перенос БД с MS SQL Server на MySQL), то внесение изменений осуществляется локально и централизованно. Нет необходимости вносить многочисленные правки по всей программе с вероятностью что-либо пропустить и тем самым спровоцировать ошибку в работе программы.
- С помощью «Repository» можно значительно упростить и алгоритмы по работе с БД в остальной программе. Чаще всего для выполнения какой-либо операции с БД достаточно вызвать один из методов класса, реализующего этот паттерн.

В силу своих особенностей паттерн «Репозиторий» удобнее всего реализовать на основе какой-либо ORM библиотеки. Например, входящих в состав .NET Framework, LINQ to SQL и Entity Framework.

Паттерн проектирования Service Layer

Паттерн определяет границу между приложением и слоем сервисов, который образует набор доступных операций и управляет ответом приложения в каждой операции.

Бизнес-приложения обычно нуждаются в различных интерфейсах к данным, которые они хранят и логике, которую реализуют: загрузчики данных, пользовательские интерфейсы, интеграционные шлюзы и другое. Вопреки различным целям, эти интерфейсы часто нуждаются во взаимодействии с приложением для доступа и управления его данными и исполнения логики. Эти взаимодействия могут быть сложными, использующими транзакции на нескольких ресурсах и управление несколькими ответами на действие. Программирование логики взаимодействия для каждого интерфейса вызовет больше количество дублирования.

Паттерн Service Layer определяет для приложения границу и набор допустимых операций с точки зрения взаимодействующих с ним клиентских компонентов. Он инкапсулирует бизнес-логику приложения, управляя транзакциями и управляя ответами в реализации этих операций.

2.2 Инструменты разработки

Visual Studio

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

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. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Git

Git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. На сегодняшний день его поддерживает Джунио Хамано.

Примерами проектов, использующих Git, являются Ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt и некоторые дистрибутивы Linux.

Программа является свободной и выпущена под лицензией GNU GPL 2.

Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone, Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-daemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

На проекте используется в связке с GitHub (GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки).

2.3 Языки программирования

C#

C# — объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота[4] как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ и некоторых других языков, не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

JavaScript

JavaScript — прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

2.4 База данных

MySQL

На сегодняшний день СУБД MySQL является одной из самых известных, надежных и быстрых из всего семейства существующих СУБД. Одной из причин являются правила ее распространения — за нее не надо платить деньги и распространяется она вместе со своими исходными текстами, другая причина – это то, что MySQL относительно быстрая СУБД.

MySQL написан под десятки видов операционных систем. Это и FreeBSD, OpenBSD, MacOS, OS/2, SunOS, Win9x/00/NT и Linux. Сегодня MySQL особенно распространена на платформах Linux и Windows. Причем на последней встречается гораздо реже.

Принцип работы СУБД MySQL аналогичен принципу работы любой СУБД, использующей SQL (Structured Query Language, язык структурированных запросов) в качестве командного языка для создания/удаления баз данных, таблиц, для пополнения таблиц данными, для осуществления выборки данных.

Приведем очень краткий список возможностей MySQL:

- ACID-совместимые транзакции;
- кроссплатформенную поддержку;
- репликации;
- поддержку огромных таблиц и баз данных;
- полнотекстовый поиск;
- поддержку подзапросов;
- поддержку большинства требований синтаксиса SQL 92.
- представления;
- использование сохраненных процедур;
- триггеры.

Entity Framework

ADO.NET Entity Framework (EF) — объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений

используется как ADO.NET Data Services (Astoria), так и связка из Windows Communication Foundation и Windows Presentation Foundation, позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования MVC, MVP или MVVM.

LINQ to Entities - это альтернативный интерфейс LINQ API, используемый для обращения к базе данных. Он отделяет сущностную объектную модель данных от физической базы данных, вводя логическое отображение между ними. Так, например, схемы реляционных баз данных не всегда подходят для построения объектно-ориентированных приложений и в результате мы имеем объектную модель приложения, существенно отличающуюся от логической модели данных, в этом случае используется LINQ to Entities, который использует модель EDM (Entity Data Model). То есть, если вам нужно ослабить связь между вашей сущностной объектной моделью данных и физической моделью данных, например, если ваши сущностные объекты конструируются из нескольких таблиц или вам нужна большая гибкость в моделировании ваших сущностных объектов используйте LINQ to Entities.

Изначально с самой первой версии Entity Framework поддерживал подход Database First, который позволял по готовой базе данных сгенерировать модель edmx. Затем эта модель использовалась для подключения к базе данных. Позже был добавлен подход Model First. Он позволял создать вручную с помощью визуального редактора модель edmx, и по ней создать базу данных. Начиная с 5.0 предпочтительным подходом становится Code First. Его суть - сначала пишется код модели на C#, а затем по нему генерируется база данных. При этом модель edmx уже не используется.

3. ОБЩИЕ ПОЛОЖЕНИЯ

3.1. Назначение документа

В настоящем документе приводится полный набор требований системе, необходимых для реализации.

3.2. Цели создания системы

Создать информационное приложение, представляющее собой блогхост и помогающее в распространении различного рода информации в виде блогов, статей и комментариев к ним.

3.3. Основные функциональные возможности системы

Приложение должно соответствовать следующим требованиям:

- создание блогов, статей блога и их последующее редактирование;
- возможность регистрации и авторизации пользователя, при этом нельзя зарегистрировать несколько аккаунтов на один email;
- иметь дружественный, интуитивно понятный интерфейс;
- редактирование личных пользовательских данных;
- управление списком пользователей через интерфейс администратора;
- модерирование статей и комментариев;
- поиск по тэгам, тексту;
- обмен сообщениями между пользователями;
- возможность оставить оценку «мне нравится» для статей;
- комментирование статей (добавление новых комментариев и их обновление должно происходить без перезагрузки страницы);
- ограничение действий для не аутентифицированных пользователей.

3.4. Плановые сроки окончания работ

Проект планируется сдать до 20 декабря 2019 года.

3.5. Использование технического задания

Для составления общей картины проекта для лабораторных работ.

4 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

4.1 Архитектура приложения

Весь проект должен реализовывать трехуровневую модель архитектуры. Схематично она представлена на рисунке 1.

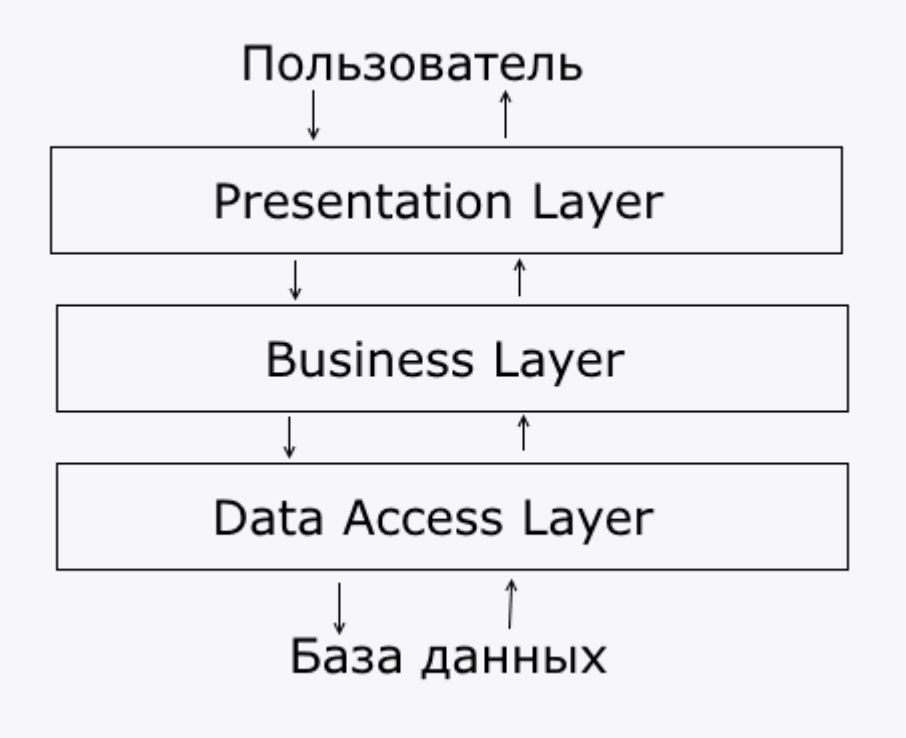

Рисунок 1. Модель трнхуровневой архитектуры.

Опишем более подробно каждый из уровней:

- *Presentation layer* (уровень представления): это тот уровень, с которым непосредственно взаимодействует пользователь. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя. Применительно к рассматриваемому проекту, на данном уровне расположены представления и все те компоненты, который составляют пользовательский интерфейс (стили, статичные страницы html, javascript), а также модели представлений, контроллеры, объекты контекста запроса.
- *Business layer* (уровень бизнес-логики): содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.
- *Data Access layer* (уровень доступа к данным): хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным,

например, класс контекста данных Entity Framework Core. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных.

При этом надо отметить, что крайние уровни не могут взаимодействовать между собой, то есть уровень представления (контроллеры) не могут напрямую обращаться к базе данных и даже к уровню доступа к данным, а только через уровень бизнес-логики.

Уровень доступа к данным не зависит от других уровней, уровень бизнес-логики зависит от уровня доступа к данным, а уровень представления - от уровня бизнес-логики.

Компоненты являются слабосвязанными, поэтому неотъемлемым звеном приложения является внедрение зависимостей.

4.2 Описание вариантов использования

4.2.1 ВИ «Найти блог по названию»

Описание ВИ:

Пользователь должен иметь возможность поиска блогов по набору слов, входящих в название.

Предусловия:

- Пользователь зашел на сайт.

Основной поток действий:

- Пользователь кликает на пункт «Search», находящийся в шапке сайта;
- В появившемся списке пользователь выбирает пункт «Blogs»;
- На открывшейся странице сайта пользователь вводит поисковый запрос в соответствующую форму ввода.
- Пользователь нажимает кнопку «Search»;
- После отображения результатов поиска, пользователь находит интересующий его блог.

Альтернативный поток действий:

- Пользователь ищет блог путем ввода его полного названия в строку поиска браузера (после доменного имени).

4.2.2 ВИ «Найти статью по названию»

Описание ВИ:

Пользователь должен иметь возможность поиска статей по набору слов, входящих в название.

Предусловия:

- Пользователь зашел на сайт.

Основной поток действий:

- Пользователь кликает на пункт «Search», находящийся в шапке сайта;
- В появившемся списке пользователь выбирает пункт «Posts»;

• На открывшейся странице сайта пользователь вводит поисковый запрос в соответствующую форму ввода.

• Пользователь нажимает кнопку «Search»;

• После отображения результатов поиска, пользователь находит интересующую его статью.

4.2.3 ВИ «Открыть страницу статьи для просмотра»

Описание ВИ:

Пользователь должен иметь возможность просмотра как собственных статей, так и статей других пользователей.

Предусловия:

• Пользователь зашел на сайт;

• Пользователь выполнил поиск статьи, см. п. 4.2.2. ВИ «Найти статью по названию».

Основной поток действий:

• Пользователь кликает на заголовок найденной ранее статьи;

• Система отображает статью с необходимой информацией поста;

4.2.4 ВИ «Открыть страницу блогов для просмотра»

Описание ВИ:

Пользователь должен иметь возможность просмотра как собственных блогов, так и блогов других пользователей.

Предусловия:

• Пользователь зашел на сайт;

• Пользователь выполнил поиск блога, см. п. 4.2.1. ВИ «Найти статью по названию».

Основной поток действий:

• Система отображает список блогов вместе с их описанием;

• Пользователь кликает на заголовок найденного ранее блога;

• Система отображает список статей блога;

4.2.5 ВИ «Создать статью»

Описание ВИ:

Пользователь должен иметь возможность создавать новые статьи, для их последующего просмотра другими пользователями.

Предусловия:

• Пользователь зашел на сайт.

• Пользователь открыл список блогов для просмотра, см. п. 4.2.4. ВИ «Открыть страницу блогов для просмотра».

Основной поток действий:

• Пользователь кликает на кнопку «Add new post» у соответствующего блога;
• На появившейся форме пользователь заполняет данные нового поста;
• Пользователь нажимает кнопку «Save»;
• Система сохраняет всю введенную информацию в базе данных.

4.2.6 ВИ «Редактировать статью»

Описание ВИ:
Пользователь должен иметь возможность редактировать основную информацию статей.
Предусловия:
• Пользователь зашел на сайт.
• Пользователь открыл статью для просмотра, см. п. 4.2.3. ВИ «Открыть страницу статьи для просмотра».
Основной поток действий:
• Пользователь кликает на кнопку «Edit» у соответствующей статьи;
• На появившейся форме пользователь редактирует данные существующего поста;
• Пользователь нажимает кнопку «Save»;
• Система сохраняет всю введенную информацию в базе данных.

4.2.7 ВИ «Удалить статью»

Описание ВИ:
Пользователь должен иметь возможность удаления статей.
Предусловия:
• Пользователь зашел на сайт.
• Пользователь открыл статью для просмотра, см. п. 4.2.3. ВИ «Открыть страницу статьи для просмотра».
Основной поток действий:
• Пользователь кликает на кнопку «Delete» у соответствующей статьи;
• Система удаляет информацию о статье из базы данных.

4.2.8 ВИ «Создать блог»

Описание ВИ:
Пользователь должен иметь возможность создавать новые блоги, для их последующего просмотра другими пользователями.
Предусловия:
• Пользователь зашел на сайт.
Основной поток действий:
• Пользователь кликает на пункт «My blogs», находящийся в шапке сайта;

- В появившемся списке пользователь выбирает пункт «Add new»;
- На появившейся форме пользователь заполняет данные нового блога;
- Пользователь нажимает кнопку «Save»;
- Система сохраняет всю введенную информацию в базе данных.

4.2.9 ВИ «Редактировать блог»

Описание ВИ:

Пользователь должен иметь возможность редактировать основную информацию блогов.

Предусловия:

- Пользователь зашел на сайт.
- Пользователь открыл список блогов для просмотра, см. п. 4.2.4 ВИ «Открыть страницу блогов для просмотра».

Основной поток действий:

- Пользователь кликает на кнопку «Edit» у соответствующего блога;
- На появившейся форме пользователь редактирует данные существующего блога;
- Пользователь нажимает кнопку «Save»;
- Система сохраняет всю введенную информацию в базе данных.

4.2.10 ВИ «Удалить блог»

Описание ВИ:

Пользователь должен иметь возможность удаления статей.

Предусловия:

- Пользователь зашел на сайт.
- Пользователь открыл статью для просмотра, см. п. 4.2.4. ВИ «Открыть страницу блогов для просмотра».

Основной поток действий:

- Пользователь кликает на кнопку «Delete» у соответствующего блога;
- Система удаляет информацию о блоге из базы данных.

4.2.11 ВИ «Поставить лайк»

Описание ВИ:

Пользователь должен иметь возможность оценить понравившуюся статью с помощью «лайка».

Предусловия:

- Пользователь зашел на сайт.
- Пользователь открыл статью для просмотра, см. п. 4.2.3. ВИ «Открыть страницу статьи для просмотра».

Основной поток действий:

- Пользователь кликает на значок в форме сердца;

• Система отображает поставленный лайк путем изменения цвета значка с серого на красный и сохраняет информацию в базе данных (сохранение и отображение происходит без перезагрузки страницы).

4.2.12 ВИ «Написать комментарий»

Описание ВИ:

Пользователь должен иметь возможность прокомментировать любую из статей.

Предусловия:

• Пользователь зашел на сайт.

• Пользователь открыл статью для просмотра, см. п. 4.2.3. ВИ «Открыть страницу статьи для просмотра».

Основной поток действий:

• Пользователь вводит текст комментария в форму ввода находящуюся под основной информацией статьи;

• Пользователь нажимает на кнопку «Comment»;

• Система отображает новый комментарий и сохраняет информацию о нем в базу данных (сохранение и отображение происходит без перезагрузки страницы).

4.3 Система ролей

Рассмотрим подробнее возможности пользователей веб-приложения. Всех пользователей можно разделить на 4 категории, дополняющие друг друга:

• Неавторизованный пользователь – пользователь с наименьшим количеством возможностей, однако имеющий возможность пользоваться достаточным набором информации. Доступные операции: просмотр всех блогов, постов и комментариев к ним.

• Обычный аторизованный пользователь – пользователь, имеющий доступ ко всем операциям неавторизованного пользователя, плюс следующие операции: возможность создания, редактирования и удаления своих постов, блогов и комментариев, возможность поставить и убрать оценку «Мне нравится» для поста.

• Модератор – пользователь, имеющий доступ ко всем операциям обычного авторизованного пользователя, плюс следующие операции: редактирование и удаление своих постов, блогов и комментариев любых пользователей. Стоит также отметить, что модератор не может снять с себя предоставленные ему «особые» полномочия. Он может сделать это лишь обратившись к администротору.

• Администратор – пользователь, имеющий доступ ко всем операциям обычного модератора, плюс следующие операции: просмотр, редактирование

и удаление личных данных любых пользователей, назначение и удаление прав «модератор» и «администратор».

4.4 Диаграмма базы данных

Результат предполагаемого каркаса базы данных, в виде диаграммы представлен на рисунке 2.

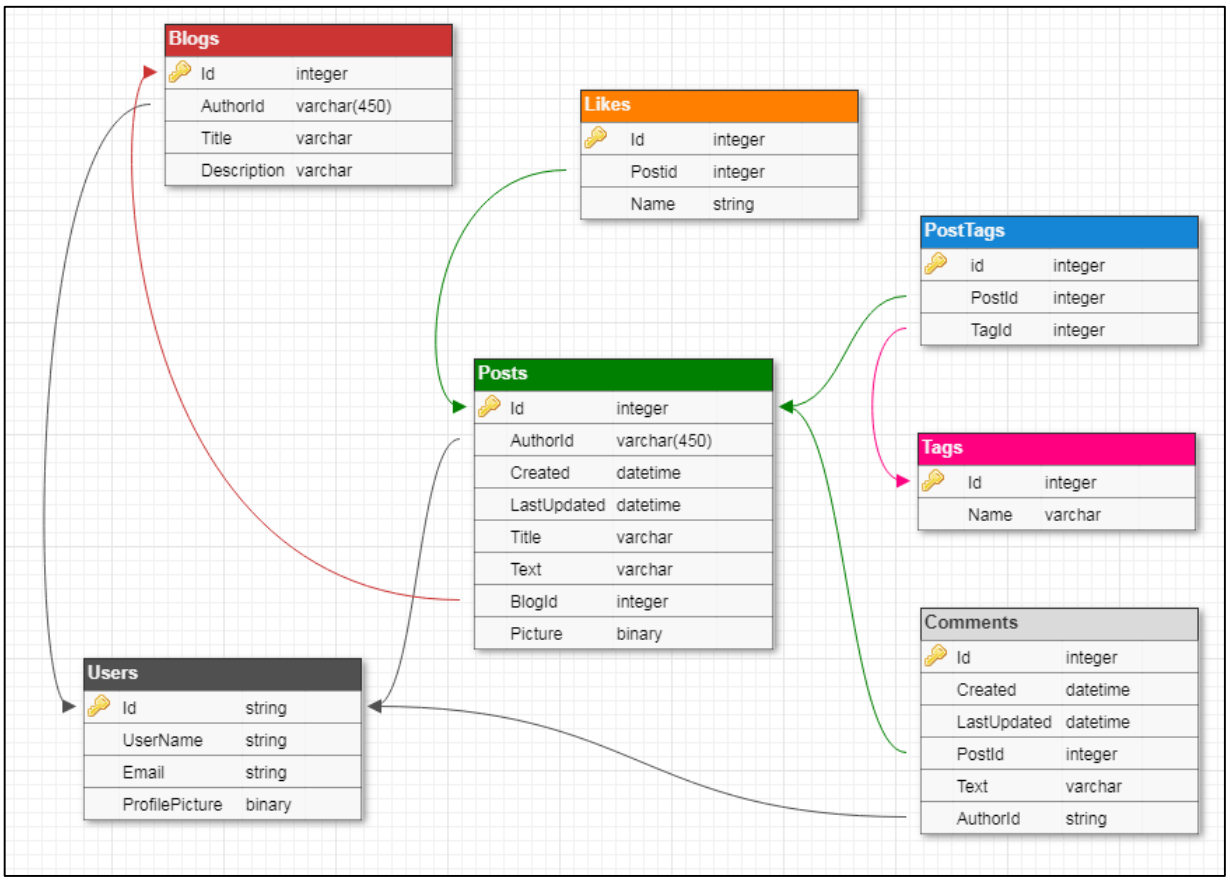

Рисунок 2. Схема базы данных.

5 НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

5.1. Интерфейс пользователя

Требования к пользовательскому интерфейсу:

- Интерфейс пользователя должен быть мультиязычным, в начале два языка: русский и английский.
- Система должна отображать корректно интерфейс пользователя с разрешением от 1024x600 пикселей.

5.2. Отображение на различных ОС

Приложение должно быть кроссплатформенным, в том числе, с возможностью запуска серверной части на различных ОС.

5.3. Документация

Документация должна соответствовать следующим требованиям:

- Разрабатываемые программные модули должны быть самодокументированы, т. е. тексты программ должны содержать все необходимые комментарии;
- В состав сопровождающей документации должна входить UML-диаграммы.

5.4. Порядок разработки

Требования к процессу разработки программного продукта:

- Вся разработка должна сопровождаться гранулярными коммитами (на английском языке) в системе GIT. Коммиты должны быть понятны и содержать лишь основную суть изменений кода программы. При разработке должны быть активно использованы соответствующие ветки (весь процесс разработки не может производиться только в ветке master);
- Разрабатываемая программа должна соответствовать руководству по стилю кода.

6. ПЕРСПЕКТИВЫ РАЗВИТИЯ

В дальнейшем планируется улучшение интерфейса web-приложения, увеличение скорости передачи данных на серверную часть, повышение удобства использования, улучшение системы ролей, повышение вовлеченности пользователей и привлекательности сайта.