ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОМУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

К защите допустить
Зав. кафедрой

_____
_____2014 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему: «Разработка подсистемы «Блог» WEB-портала «Software.uz»»

Выпускница _____ Сайдалиева Р.М.
подпись Ф.И.О.

Руководитель _____ Когай В.Н.
подпись Ф.И.О.

Консультант по БЖД _____ Абдуллаева С.М.
подпись Ф.И.О.

Рецензент _____ Гулямов Ш.М.
подпись Ф.И.О.

Ташкент – 2014

ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Факультет: «Компьютерный инжиниринг»

Кафедра: «Мультимедийные технологии»

Направление (специальность): 5521900 – «Информатика и информационные технологии»

У Т В Е Р Ж Д А Ю

Зав. кафедрой _____
« ___ »_____ 2014 г.

ЗАДАНИЕ

на выпускную квалификационную работу студентки
Сайдалиевой Роксаны Малимбаевной
(фамилия, имя, отчество)

1. Тема работы: *«Разработка подсистемы «Блог» WEB-портала «Software.uz»»*

2. Тема утверждена приказом по университету от 19.04.2014г. № 254-15

3. Срок сдачи законченной работы: *31.05.2014*

4. Исходные данные к работе: *Техническое задание на разработку и модернизацию «Национального каталога разработчиков и программных продуктов Software.uz».*

5. Содержание расчётно-пояснительной записки (перечень подлежащих к разработке вопросов): *анализ предметной области, проектирование подсистемы на основе UML, проектирование структуры БД, программная реализация*.

6. Перечень графического материала: *таблицы, пользовательские интерфейсы, диаграммы, презентация.*

7. Дата выдачи задания: *03.03.2014*

Руководитель _____ Задание приняла _____

(подпись) (подпись)

8. Консультанты по отдельным разделам выпускной работы

Раздел	Ф.И.О руководителя	Подпись дата	
		Задание выдал	Задание получил
Основная часть	Когай В.Н.	05.03.2014	05.03.2014
БЖД	Абдуллаева С.М.	20.03.2014	20.03.2014

9. График выполнения работы

№	Наименование раздела работы	Срок выполнения	Отметка руководителя о выполнении
1	Анализ предметной области	05.03.2014 -17.03.2014	
2	Проектирование архитектуры подсистемы и структуры БД	18.03.2014 -18.04.2014	
3	Программная реализация системы	19.04.2014 -19.05.2014	
4	БЖД	20.03.2014 -28.05.2014	
5	Оформление пояснительной записки	29.05.2014 -31.05.2014	

Выпускник _____ «_____» _____ 2014 г.
(подпись)

Руководитель _____ «_____» _____ 2014 г.
(подпись)

Данная выпускная квалификационная работа посвящена разработке подсистемы «Блог» WEB-портала «Software.uz», обеспечивающей размещение и обсуждение статей и ссылок, содержащих информацию о сфере ИКТ, а также сведения по использованию прикладного программного обеспечения. В данной работе были рассмотрены теоретические основы построения блогосферы, определены проблемы и выработаны требования для реализации, приведена постановка задачи и практическая реализация системы. При разработке были использованы платформа ASP.NET, язык программирования C#, а также СУБД Microsoft SQL Server 2008 R2.

This graduate work is devoted to development of the subsystem «Blog» of WEB-portal «Software.uz», providing accommodation and discussion of articles and links containing information of ICT, as well as information of using of application software. In this work were considered the theoretical base for the blog sphere, identified problems and developed requirements for implementation, given the problem and the practical implementation of the system. During development of subsystem were used platform ASP.NET, programming language C #, as well as the DBMS Microsoft SQL Server 2008 R2.

Ushbu bitiruv malakaviy ish «Blog» Web-portal «Software.uz»ga tizimosti ishlov berishga bag’ishlangan, joylashtirish va muhokama moddalarni va havolalar ta’minlovchi, IKT haqida ma’lumotni o’z ichiga oluvchi, hamda ma’lumotlar foydalanish bo’yicha amaliy dasturiy ta’minlash. Ushbu ishda blog yunalish tuzilishini nazariy assoslari kurib chiqildi, aniqlangan muammolar korib chiqildi va echimlar tuzildi. Ishlab chiqarish moboynida ASP.NET platformasi, C# dasturiy tili hamda MBBT Microsoft SQL Server 2008 R2 qo’llandi.

О Г Л А В Л Е Н И Е

ВВЕДЕНИЕ

В XXI веке овладение информационными технологиями и внедрение глобальных информационных систем является решающим фактором для прогресса страны и национального развития. Информационная технология формирует передний край научно-технического прогресса, создает информационный фундамент развития науки и всех остальных технологий. Главными, определяющими стимулами развития информационной технологии, являются социально-экономические потребности общества.

Развитие информационно-коммуникационных технологий (ИКТ), являющееся важнейшим фактором поднятия благосостояния и экономического роста, становится одним из основных приоритетов государственной политики Узбекистана.

Принципиальная позиция Президента Узбекистана по ключевым вопросам компьютеризации общества придала этой проблеме особую значимость, позволила коренным образом изменить подходы к информатизации, перейти к комплексному внедрению компьютерных и информационных технологий.

За последнее время были приняты постановления Президента и Кабинета Министров РУз, стимулирующие развитие рынка программных продуктов отечественного производства. К их числу относятся следующие документы:

1. Постановление Президента Республики Узбекистан от 21.03.2012г. № ПП-1730 «О мерах по дальнейшему внедрению и развитию информационно-коммуникационных технологий» [1].

2. Постановление Президента Республики Узбекистан № ПП-2042 от 20.09.13 «О мерах по дальнейшему усилению стимулирования отечественных разработчиков программного обеспечения» [2].

В результате по данным Государственного комитета статистики Республики Узбекистан, уже в 2013 году рынок услуг по компьютерному

программированию в Узбекистане вырос более чем в два раза, почти на 230%, по сравнению с прошлым годом.

В условиях бурного развития сферы ИКТ в Узбекистане возникла актуальная потребность в создании социально-профессиональной ВЕБ сети разработчиков программного обеспечения, ядром которой должен быть ВЕБ-портал национального каталога разработчиков и программных продуктов.

В соответствии с решениями Государственного комитета по связи и информационных технологий Республики Узбекистан в 2013 году группой специалистов и студентов Ташкентского университета информационных технологий был разработан и внедрен «Национальный каталог разработчиков и программных продуктов Software.uz» (далее Каталог). В данной выпускной квалификационной работе рассматриваются вопросы создания подсистемы «Блог» в соответствии с техническим заданием на разработку и модернизацию «Национального каталога разработчиков и программных продуктов Software.uz».

Целью данной работы является разработка подсистемы «Блог» WEB-портала «Software.uz», обеспечивающий размещение и обсуждение полезных статей и ссылок, содержащих информацию о сфере ИКТ, а также информацию по использованию прикладного программного обеспечения.

Выпускная квалификационная работа состоит из введения, четырёх глав и заключения.

Во введении обоснована актуальность работы, а также определены цели и задачи необходимые для реализации.

В первой главе приведены основные понятия блогосферы и проведен анализ функций и требований к разрабатываемой подсистеме «Блог».

Во второй главе сформулирована постановка задачи, описываются архитектура объектной модели подсистемы, структура базы данных, построены диаграммы проектирования при помощи языка UML.

В третьей главе объясняется выбор технологии разработки

ASP.NET, представлено описание программного продукта.

В четвертой главе приводятся сведения об пожарной безопасности и рациональной организации рабочего места.

В заключении сделаны выводы по выпускной квалификационной работе.

В приложении приводятся объектная модель подсистемы, описание таблиц базы данных и код программного продукта.

ГЛАВА 1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1. Подходы к определению блога

Блог (англ. *blog*, от *we**b log*** — интернет-журнал событий, интернет-дневник, онлайн-дневник) — веб-сайт, основное содержимое которого — регулярно добавляемые записи, содержащие текст, изображения или мультимедиа. Для блогов характерны недлинные записи вре́менной значимости, упорядоченные в обратном хронологическом порядке (последняя запись сверху). Отличия блога от традиционного дневника обусловливаются средой: блоги обычно публичны и предполагают сторонних читателей, которые могут вступить в публичную полемику с автором (в комментарии к блогозаписи или своих блогах).

Людей, ведущих блог, называют бло́ггерами. Совокупность всех блогов Сети принято называть блогосферой.

Для блогов характерна возможность публикации отзывов (комментариев, «комментов») посетителями. Она делает блоги средой сетевого общения, имеющей ряд преимуществ перед электронной почтой, группами новостей, веб-форумами и чатами.

Под блогами также понимаются персональные сайты, которые состоят в основном из личных записей владельца блога и комментариев пользователей к этим записям.

Первым блогом считается страница Тима Бернерса-Ли, где он, начиная с 1992 г., публиковал новости. Более широкое распространение блоги получили с 1996 г. В августе 1999 г. компьютерная компания Pyra Labs из Сан-Франциско открыла сайт Blogger.com, который стал первой бесплатной блоговой службой.

В настоящее время особенность блогов заключается не только в структуре записей, но и в простоте добавления новых записей. Пользователь просто обращается к веб-серверу, проходит процесс идентификации

пользователя, после чего он добавляет новую запись к своей коллекции. Сервер представляет информацию как последовательность сообщений, помещая в самом верху самые свежие сообщения. Структура коллекции напоминает привычную последовательную структуру дневника или журнала.

1.2. Мотивация участия и функции блогов

Следует учитывать, что чтение блогов и авторство — два разных по содержанию процесса. Люди, пользующиеся коммуникативными возможностями блогов вне контекста ведения собственного блога, отмечают возможности общения с людьми, с которыми они не имеют возможности общаться непосредственно, например, с друзьями, живущими в других городах. Альтернативная экономика сообщений, действующая в блогах, делает такую форму общения наиболее удобной, так как она не предполагает обязательной взаимности и других ограничений общения «один на один».

Выделяет, помимо поддержания контакта с близкими, следующие цели, которые может преследовать читатель блогов:

— получение информации;
— чтение-развлечение;
— отслеживание реакции публики на те или иные действия (в самом деле, блоги представляют собой готовую огромную фокус-группу);
— чтение ради социализации, ощущения себя причастным к жизни известных людей;

Функции блогов

Коммуникативная функция упоминается чаще всего. Большинство блогеров говорят, что ведут или читают блоги ради общения с интересными им людьми. В первую очередь это возможность сказать что-то один раз так, чтобы это услышали многие. Какой смысл рассказывать десяти, двадцати, тридцати знакомым о поездке в пригородный парк, если это можно описать в своём блоге, украсив запись фотографиями? Каждый прочтёт об этом в

удобное ему время или не будет читать вовсе, решив, что это ему неинтересно. («Иногда хочется ночами поговорить, а все спят. Тогда можно написать в Живой журнал и потом прочитают»). Впрочем, такая ситуация порождает встречную проблему, когда двум встретившимся «в реале» блогерам, если они не обладают должной фантазией, бывает не о чем поговорить.

Как для «читателей», так и для «писателей» можно выделить два направления коммуникативной мотивации в использовании блогов — общение со знакомыми и расширение круга общения. В то время как одни люди заводят блог для удобства коммуникации с имеющимися знакомыми, другие заводят блог для того, чтобы познакомиться с новыми людьми, для расширения своей аудитории. В этих двух формулировках — «познакомиться с новыми людьми» и «расширить аудиторию», которые действительно употреблялись участниками опроса,— находит своё отражение ещё одно различие: в то время как одним нужны друзья, другим — слушатели.

Функция самопрезентации. Несколько респондентов отмечали, что изначально задумывали создать персональную страничку (сайт в Интернете), но позже, узнав о том, насколько легко вести блог, предпочли эту форму изложения информации о себе. Существует класс блогов, предназначенных для публикации и обсуждения произведений автора (прозы, стихов, фотографий, рисунков), однако и обычный дневник несомненно несёт информацию о личности автора. «Веду дневник, чтобы меня читали»,— могут сказать многие блогеры.

Функция развлечения. Многие люди предпочитают ведение блога, чтение блогов и дискуссии в комментариях в качестве развлекательного времяпрепровождения, особенно, если они по каким-либо причинам ограничены в других средствах развлечения, кроме интернета, и имеют довольно много свободного времени, которое надо тратить, например, молодые матери составляют заметную часть сообщества блогеров — им

всегда есть о чём написать в свой блог, у них много вопросов, с которыми они могут обратиться к другим и т. д. Блоги представляют собой неисчерпаемый источник развлекательного чтения.

Некоторые пользователи используют блоги, чтобы тратить время, которого у них много, другие же используют его из-за нехватки свободного времени для полноценного общения. Таким образом, механизм сообщества блогеров позволяет вести общение в удобном для каждого пользователя режиме и с той интенсивностью, которая ему нужна (или которую он может себе позволить).

Функция сплочения и удержания социальных связей. Как уже упоминалось, сообщество Livejournal объединяет десятки тысяч русскоязычных пользователей, среди которых многие могут найти людей, с которыми они были когда-то знакомы. Блоги, выполняя функции социальных сетей, позволяют поддерживать прервавшиеся в реальной жизни социальные связи и лучше узнавать своих знакомых.

Благодаря особенности отложенной многопользовательской коммуникации некоторые из респондентов используют блоги с нетрадиционной целью — для организации взаимодействия рабочей группы, обсуждения рабочих вопросов и т. п., так как для многих задач подобный способ оказывается более удобным, чем электронная почта, службы мгновенных сообщений и т. п. Группы, организованные на основе списков рассылки электронной почты, также могут выполнять подобную задачу, однако в ситуации, когда все участники проекта ведут блоги на одном сервисе, добавлять ещё один канал коммуникации может показаться излишним.

Функция мемуаров. Как и традиционный бумажный дневник, блог, помимо новых функций, может осознаваться и как несущий функцию мемуаров, места для каких-то записей, которые могут пригодиться в будущем, способом не забыть о подробностях тех или иных событий своей жизни. Пользующиеся этой функцией респонденты полагают, что ведут

дневник для себя, для того чтобы потом читать, для того чтобы записывать что-то, что хочется помнить. И подумать об этом позже. Авторы создают нечто вроде отложенной коммуникации с самим собой.

Функция саморазвития, или рефлексии. Эта функция связана с тем, что блог предоставляет возможность участникам создать образ иного Я, возможно, такого, к которому автор стремится. («Я начинал журнал как упражнение в открытости и спонтанности»). Некоторые отмечают, что публичность дневника вынуждает их продолжать его вести, а также заставляет учиться более грамотно структурировать свои мысли, что помогает им и самим лучше понять переживаемые события («При изложении своей проблемы или идеи в письменном виде становится легче анализировать ситуацию»).

Психотерапевтическая функция. Встречаются также упоминания о психотерапевтической функции блога, которая либо предполагалась заранее, либо была осознана в процессе ведения записей — «выплеснуть эмоции», «изложить наболевшее», «для успокоения нервов, в конце концов». Данная функция традиционного дневника, ведущегося в укромной тетрадке, неоднократно упоминается различными авторами и, по всей видимости, приобрела новую форму и новые возможности, как способ пожаловаться на жизнь множеству людей сразу и получить в ответ успокоительные «поглаживания». Если рассмотреть содержание регулярных диалогов в сообществе пользователей livejournal «ru_psychology», и не только там, то такого рода транзакции являются одной из наиболее популярных форм общения: один «клиент» жалуется на жизнь, а другие «клиенты» его утешают.

Продвижение товаров и услуг. С помощью блогов можно отслеживать то, что говорит множество людей о вашем рынке, организации, товарах. Принимать непосредственное участие в обсуждении, размещая комментарии в блогах других людей. Можно сотрудничать с блогерами, пишущими на

схожие темы, и наконец, оказывать непосредственное влияние на обсуждение через собственный блог.

1.3. Разновидности блогов

По авторскому составу блоги могут быть личными, групповыми (корпоративными, клубными), общественными (открытыми).

По содержанию — тематическими.

По авторству:

- Личный (персональный, авторский, частный) блог — ведётся одним лицом (как правило, его владельцем).
- Коллективный или социальный блог — ведётся группой лиц по правилам, определяемым владельцем и модераторами.
- Корпоративный блог — ведётся сотрудниками одной организации.

По тематической направленности:

Обычно персональные блоги носят личный характер. Однако в среде персональных, так же, как и в среде коллективных и корпоративных блогов, существуют специализированные блоги, посвящённые определенным сферам жизни:

- Политика — блоги, посвященные политике. Обычно политические блоги ведут политические лидеры, представители политических партий и политических объединений, политологи.
- Быт — блоги, в которых затрагиваются обычно вопросы взаимоотношений между людьми, психологии, ведения домашнего хозяйства — всего того, что связано с понятиями «быт», «личная жизнь».
- Образование— блоги, посвященные теме образования. Часто это блоги определенных учебных заведений, в которых участники обсуждают процесс обучения и другие темы в образовании.

По наличию/виду мультимедиа:

- Текстовый блог — основное содержание составляют тексты.
- Фотоблог — основное содержание составляют фотографии.
- Артблог — основное содержание составляют рисунки автора блога.
- Музыкальный блог — основное содержание составляет музыка.
- Подкаст и блогкастинг — основное содержание блога надиктовывается и выкладывается в виде MP3-файлов.
- Видеоблог — основное содержание представлено в виде видеофайлов.
- Микроблог — основное содержание составляют размещенные небольшие кусочки цифрового контента, который может быть текстом, изображением, ссылкой, коротким видеороликом. Друзья используют его, чтобы быть в курсе деловых встреч, узнать больше о знаменитостях и политике, о датах концертов, лекций, книг-релизов.

По особенностям контента:

- Контентный блог — блог, публикующий первичный авторский контент.
- Мониторинговый (ссылочный) блог — блог, основным контентом которого являются откомментированные ссылки на другие сайты.
- Цитатный блог — блог, основным контентом которого являются цитаты из других блогов.
- Тамблелог, Тамбллог, Тлог — почти то же самое, что и обычный блог, с одним отличием: запись в блоге может быть только определённого формата. Например, цитата, видео, ссылка, песня, разговор и так далее). Тамблелог — скорее не система типа дневника, а черновик или записная книжка.
- Флоги и фэйковые блоги. Наряду с обычными блогами существуют так называемые флоги и фэйковые блоги. Во флогах оплаченные записи с рекламным содержанием публикуются маскируются под личные впечатления. Подобная деятельность нарушает европейский закон о защите прав потребителей, и в некоторых странах за неё

предусмотрено наказание. Например, в Великобритании с 2007 года за флоги предусмотрена уголовная ответственность.

Выводы по первой главе

Рассмотрев в данной главе особенности, историю и функции блогов, можно сделать вывод, что среди различных форм интернет-коммуникации и обсуждении информации, самой популярной являются блоги.

Существует несколько определений блога, но в данной работе целесообразнее взять за основу следующее определение: блог – это веб-сайт, содержащий датированные записи мультимедийного характера, расположенные в обратном хронологическом порядке, с возможностью оставления комментариев к записям и просмотра любой из них на отдельной веб-странице.

Разрабатываемый блог по авторству является коллективным блогом, тематическим направлением которого является «ИКТ».

Актуальная потребность в подобных блогах – эффективное обучение новым материалам. Как и в любом обучаемом процессе, выделяются два объекта:

- Продвинутый пользователь, кто обучает (далее Фрилансер).
- Начинающий пользователь, кто обучается (далее Гость).

Возникает вопрос создания удобной площадки для реализации процесса обучения начинающих пользователей в сфере ИКТ, в разрабатываемом проекте это реализуется в виде досок гостей, где фрилансеры могут вносить поправки к постам, выделив именно тот фрагмент, где гости неправильно рассуждает.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ ПОДСИСТЕМЫ

2.1. Постановка задачи

Целью данной работы является создание подсистемы «Блог» Web-портала «Software.uz».

Разрабатываемая подсистема должна удовлетворять следующим функциональными требованиями:

- Добавление, редактирование, удаление сфер и направлений модератором;
- Просмотр, добавление, удаление, комментирование постов.
- Подписки на направления;
- Уведомления о подписках и ответах;
- Отношение пользователей к посту (нравится/не нравится);
- Определение полезности поста (рейтинг);
- Создание персональных досок для постов;
- Вносить поправки на посты;
- Сортировка постов;
- Поиск по блогу.

2.2. UML – диаграммы функций подсистемы

UML (англ. Un*ified Modeling Language* – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном,

программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода [3].

Диаграмма использования (use case diagram) – это наиболее общее представление функционального назначения системы (см. рис. 2.1.)

Диаграммы последовательности – для моделирования взаимодействия объектов во времени. Взаимодействие – в виде двумерной схемы. По вертикали – временная ось сверху вниз, по горизонтали – роли.

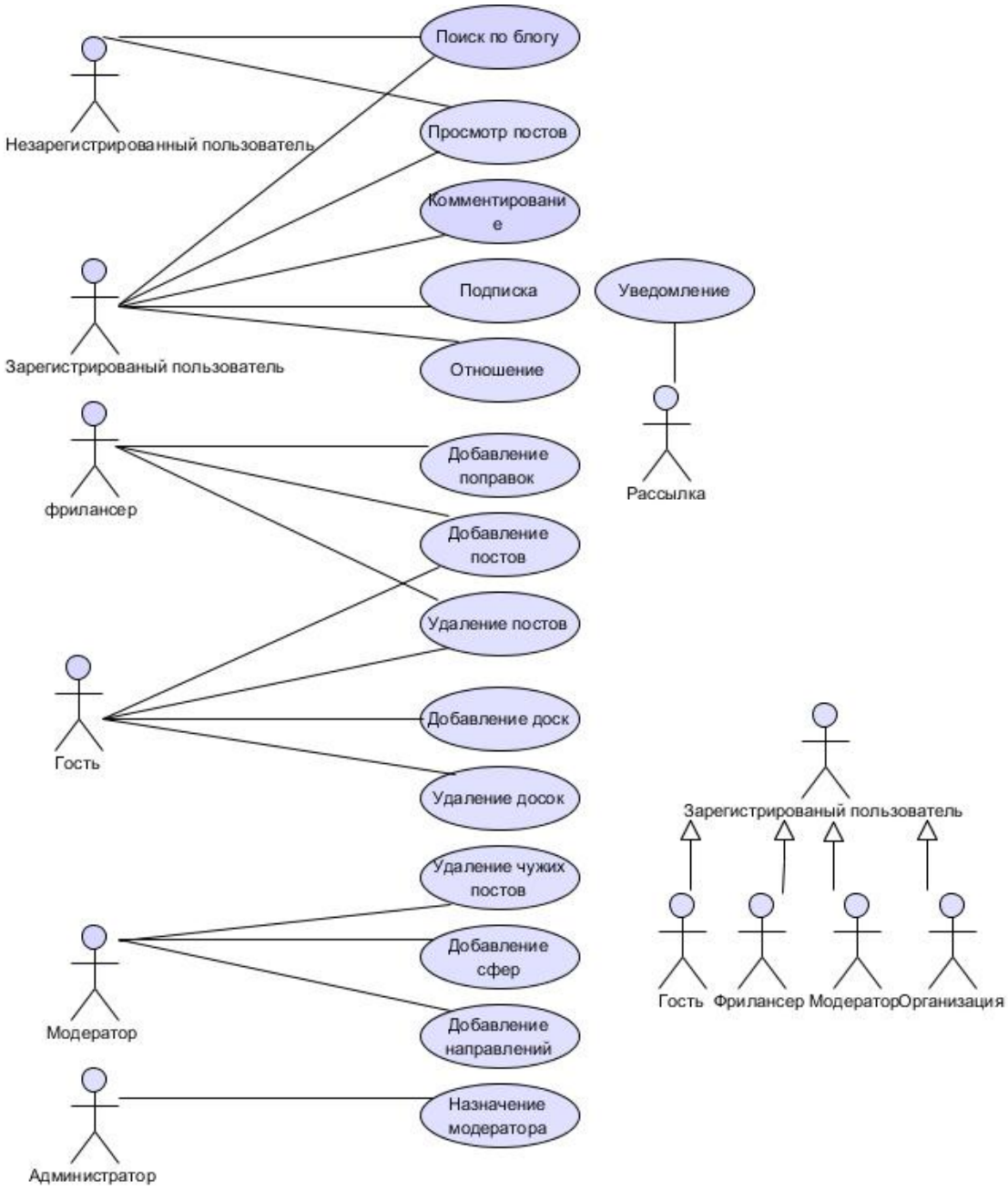


Рис. 2.1. Диаграмма вариантов использования.

Диаграмма последовательности процесса аутентификации.

Со стороны клиента приходит запрос на проверку логина и пароля в БД пользователей.Web приложение проверяет на существование пользователя в БД, в случае успешности система выдает клиенту главную страницу с сгенерированым списком постов. При этом создается объект клиента помещаемый в сессию, где хранится полная информация о клиенте и его правах (см. рис. 2.2.).

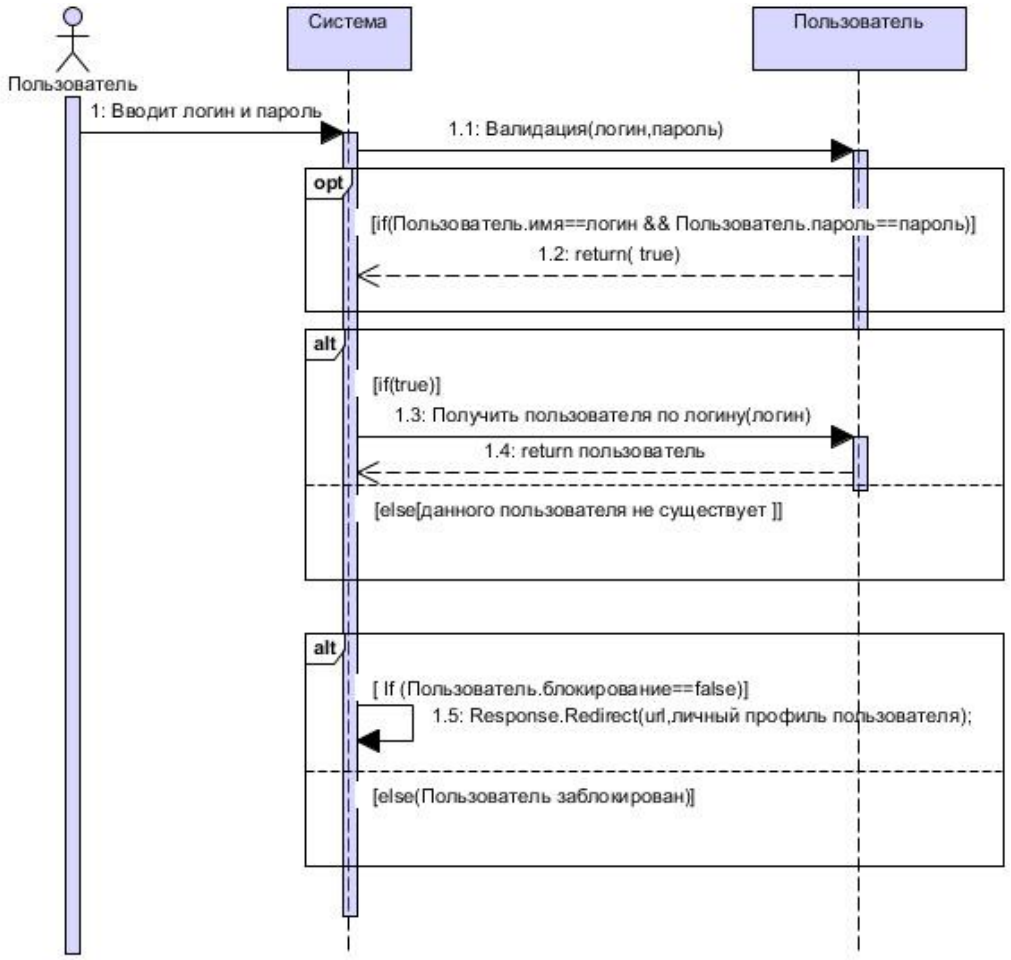


Рис. 2.2. Процесс аутентификации.

Диаграмма последовательности процесса добавления постов.

В случае успешной аутентификации, фрилансеру предоставляется возможность размещения постов двумя методами (см. рис. 2.3.):

— Направления (Direction);

— Личные доски (Board);

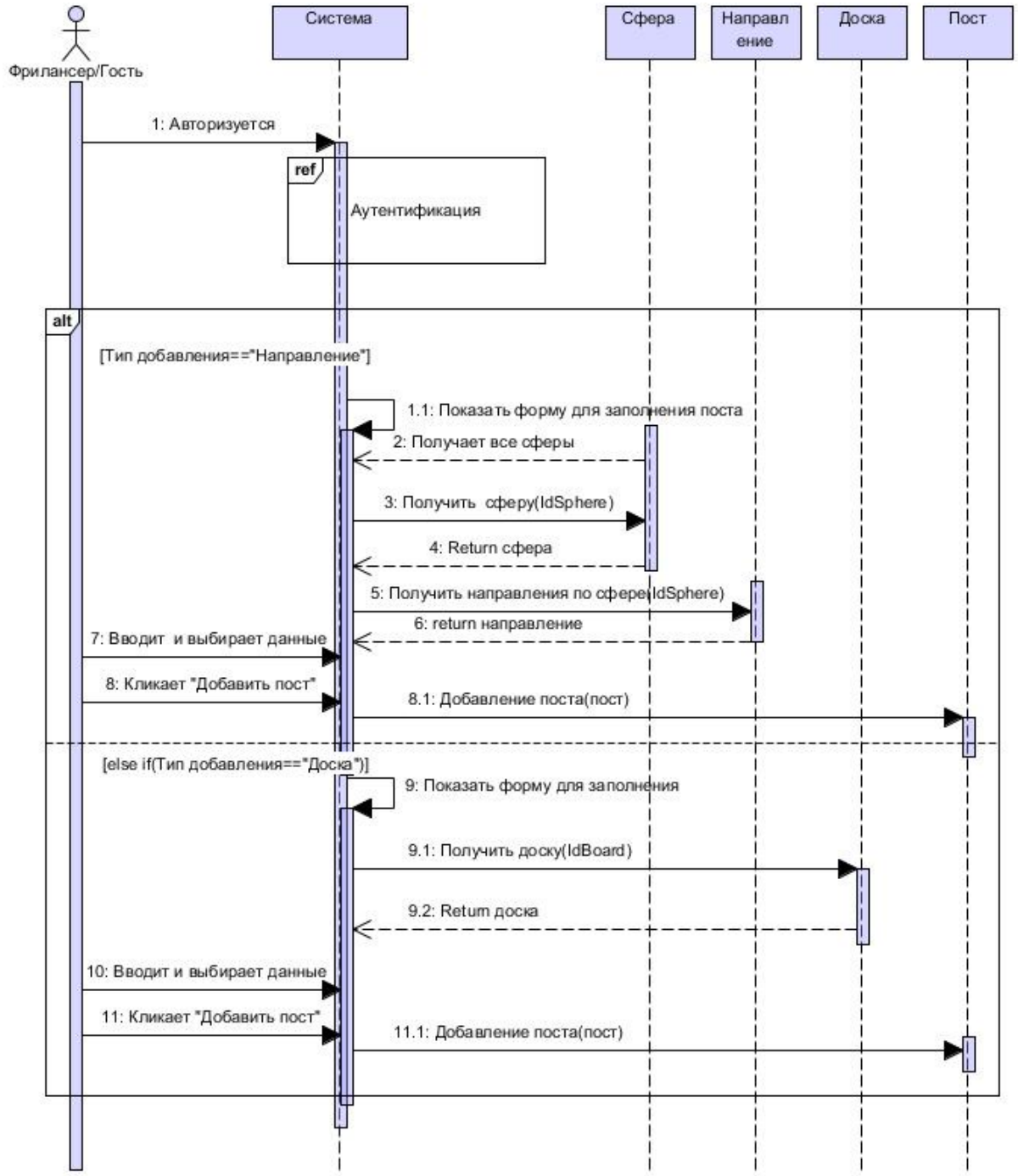


Рис. 2.3. Процесс добавления постов.

Диаграмма последовательности процесса просмотра постов.

Всем пользователям предоставляется возможность просмотра постов. Для зарегистрированных пользователей формируется личное собственное меню, где в виде ссылок формируются все направления, где пользователь оставил посты и личные доски, созданные пользователем. Пройдя по данным ссылкам, пользователь получает все посты данной категории (см. рис. 2.4.).

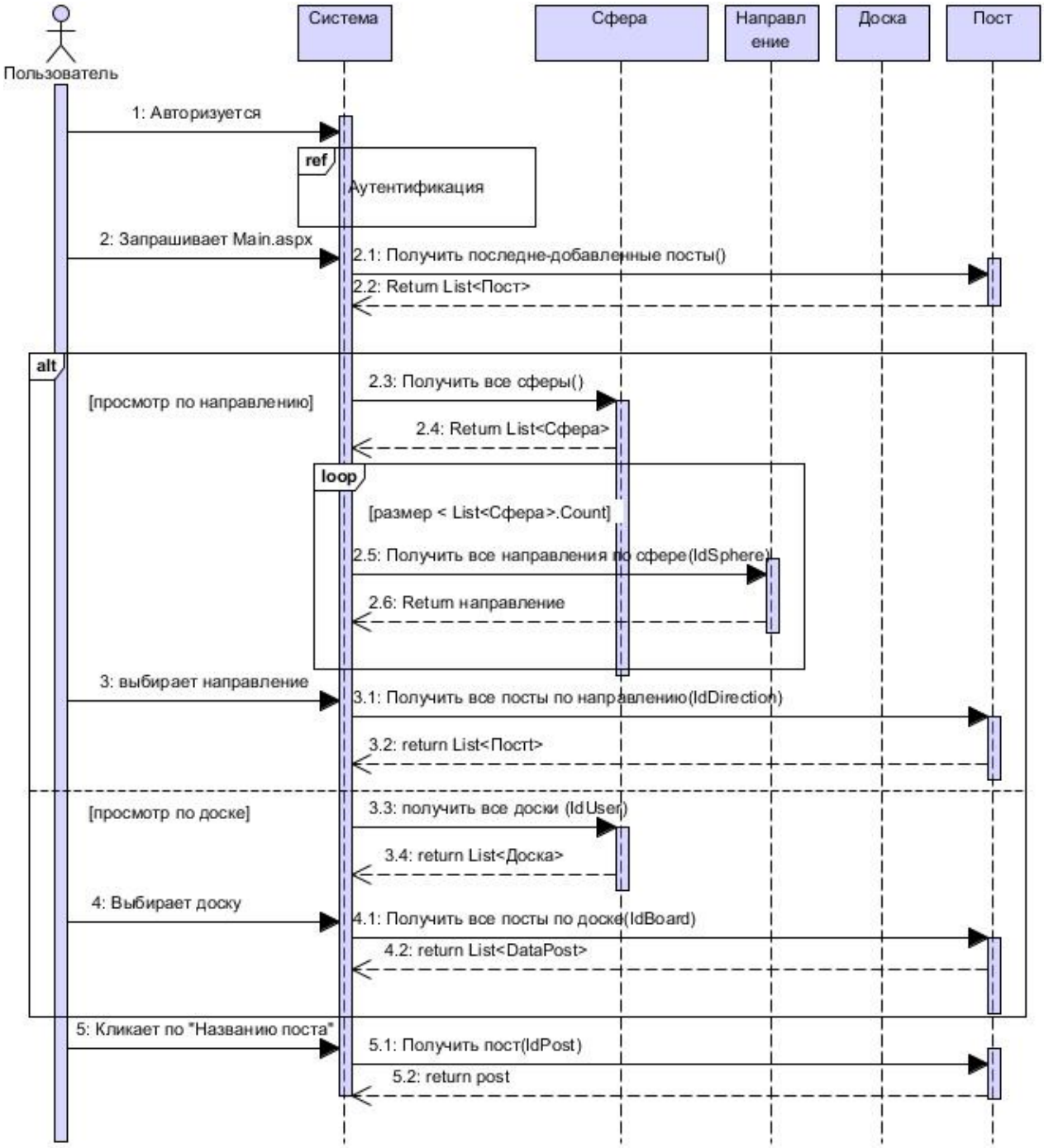


Рис. 2.4. Процесс просмотра постов.

Диаграмма последовательности процесса удаления поста.

Автор поста и модератор имеет право на удаление поста. Если уникальный ключ пользователя совпадает с номером пользователя поста, то появляется возможность удаления (см. рис.2.5.).

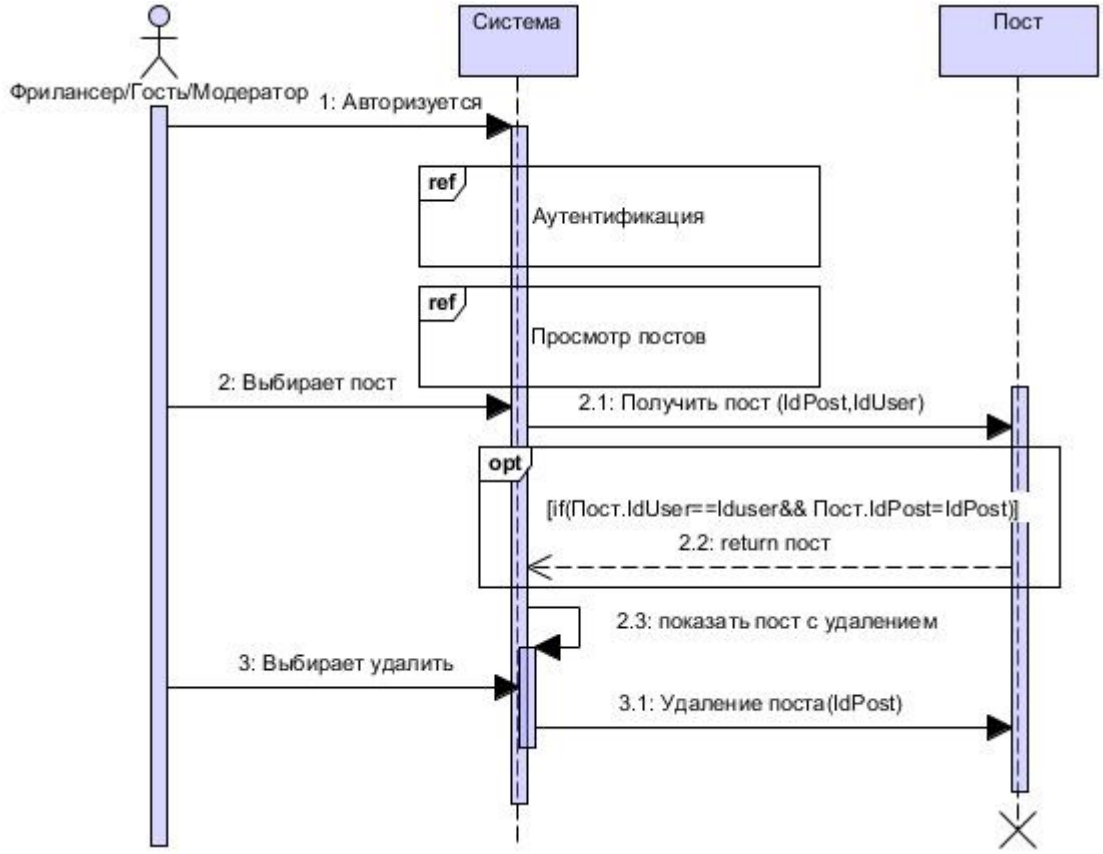


Рис. 2.5. Процесс удаления поста.

Диаграмма последовательности процесса добавления доски.

Гости могут добавлять доски, воспользовавшись ссылкой “Добавить доску” (см. рис. 2. 6.).

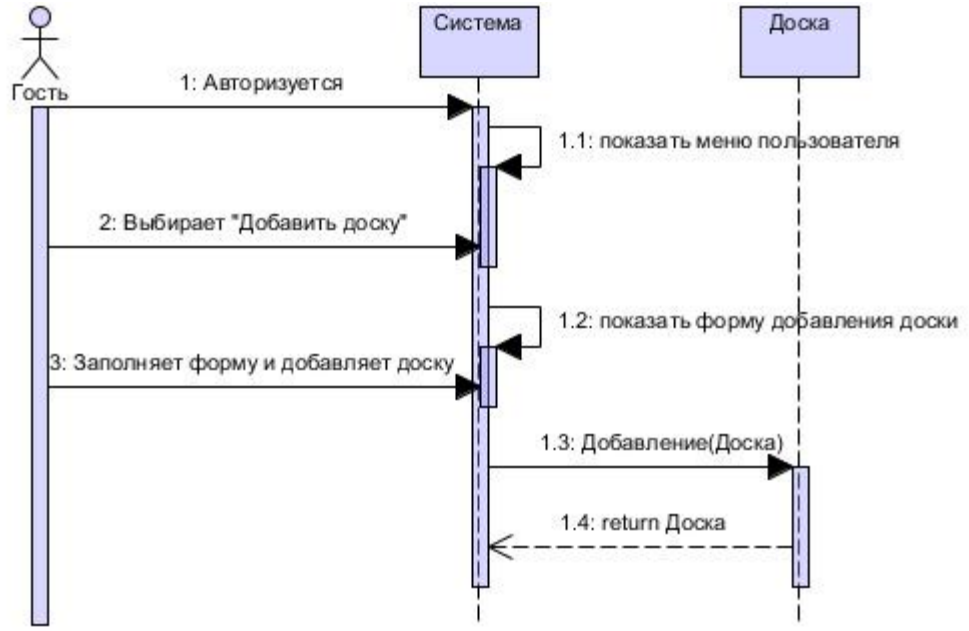


Рис. 2.6. Процесс добавления доски.

Диаграмма последовательности процесса поиска по блогу.

Все пользователи имеют доступ к поиску постов. По умолчанию поиск осуществляется по названию постов. Так же предоставляется возможность пользования расширенным поиском по следующим видам: определенная сфера, определенное направление, имя автора поста (см. рис. 2.7.).

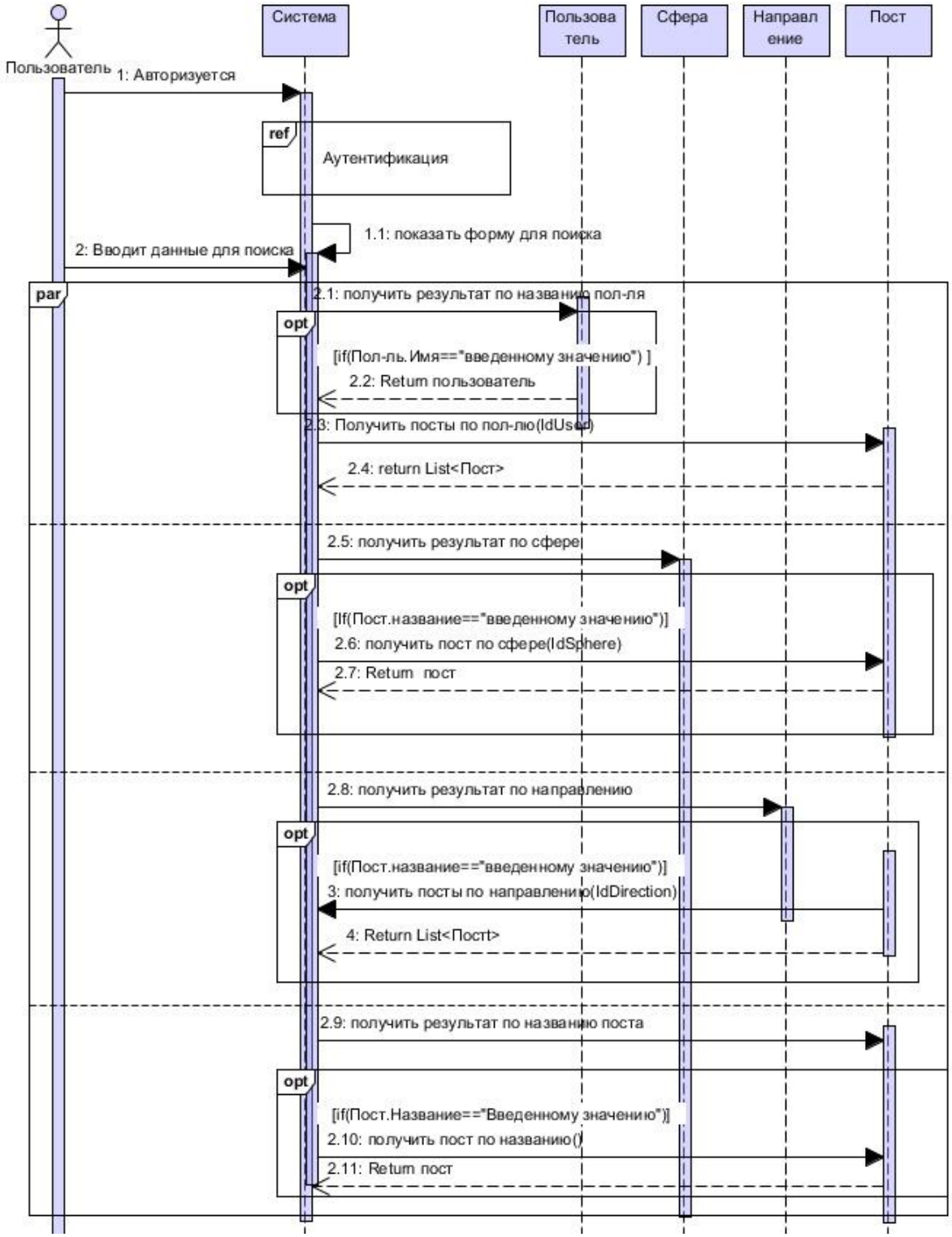


Рис. 2.7. Процесс поиска по блогу.

Диаграмма последовательности процесса подписывания.

В случае успешной аутентификации, каждый пользователь может подписаться на определенное направление и получать рассылки в виде постов. А так же отказаться от подписок, в следствии прекратиться процесс рассылки (см. рис. 2.8.).

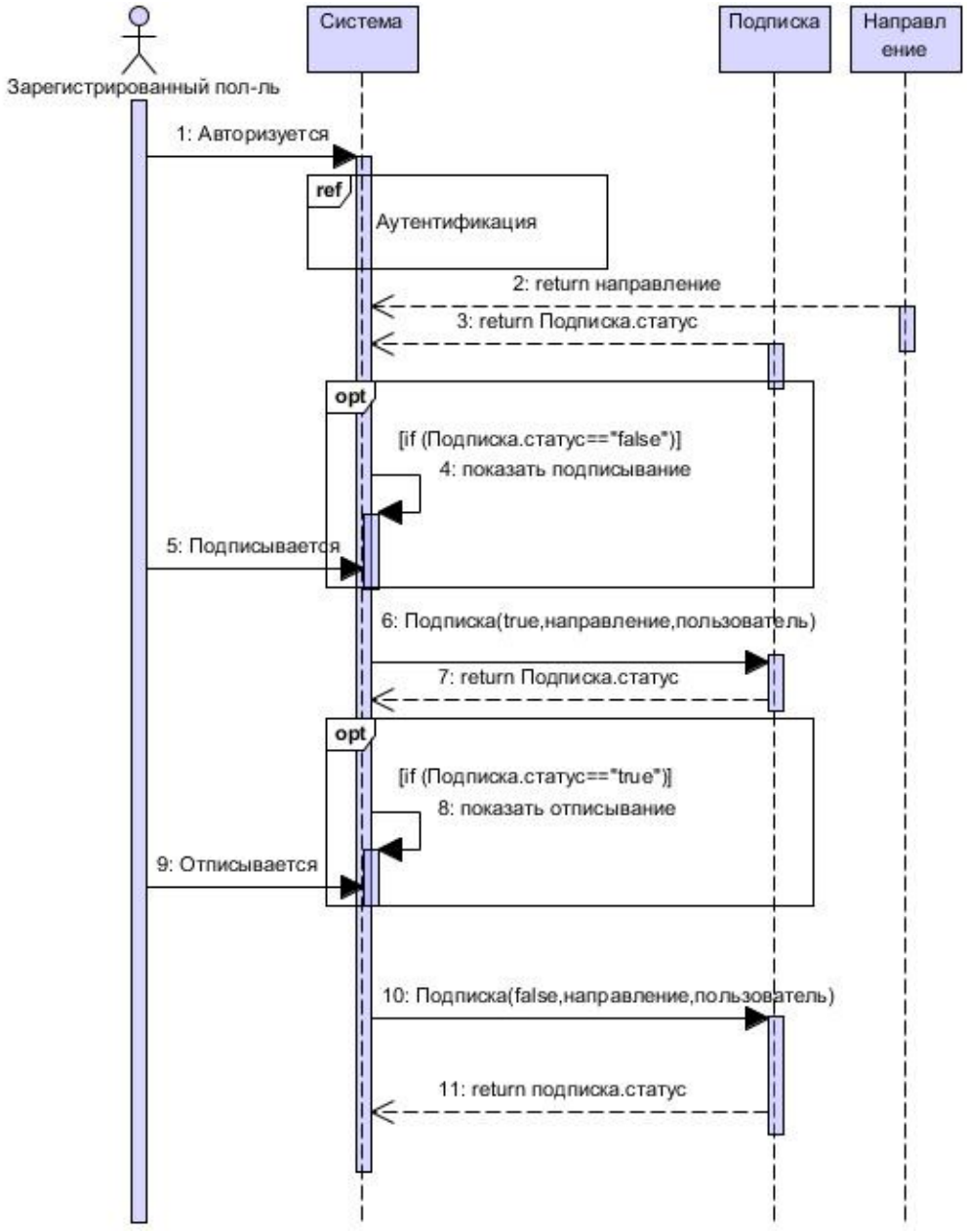


Рис. 2.8. Процесс подписывания.

*Диаграмма последовательности определения полезности поста.*Для определения полезности поста, критерием которой является рейтинг поста,

который формируется вследствии подсчета количества лайков и дислайков. (см. рис. 2.9.)

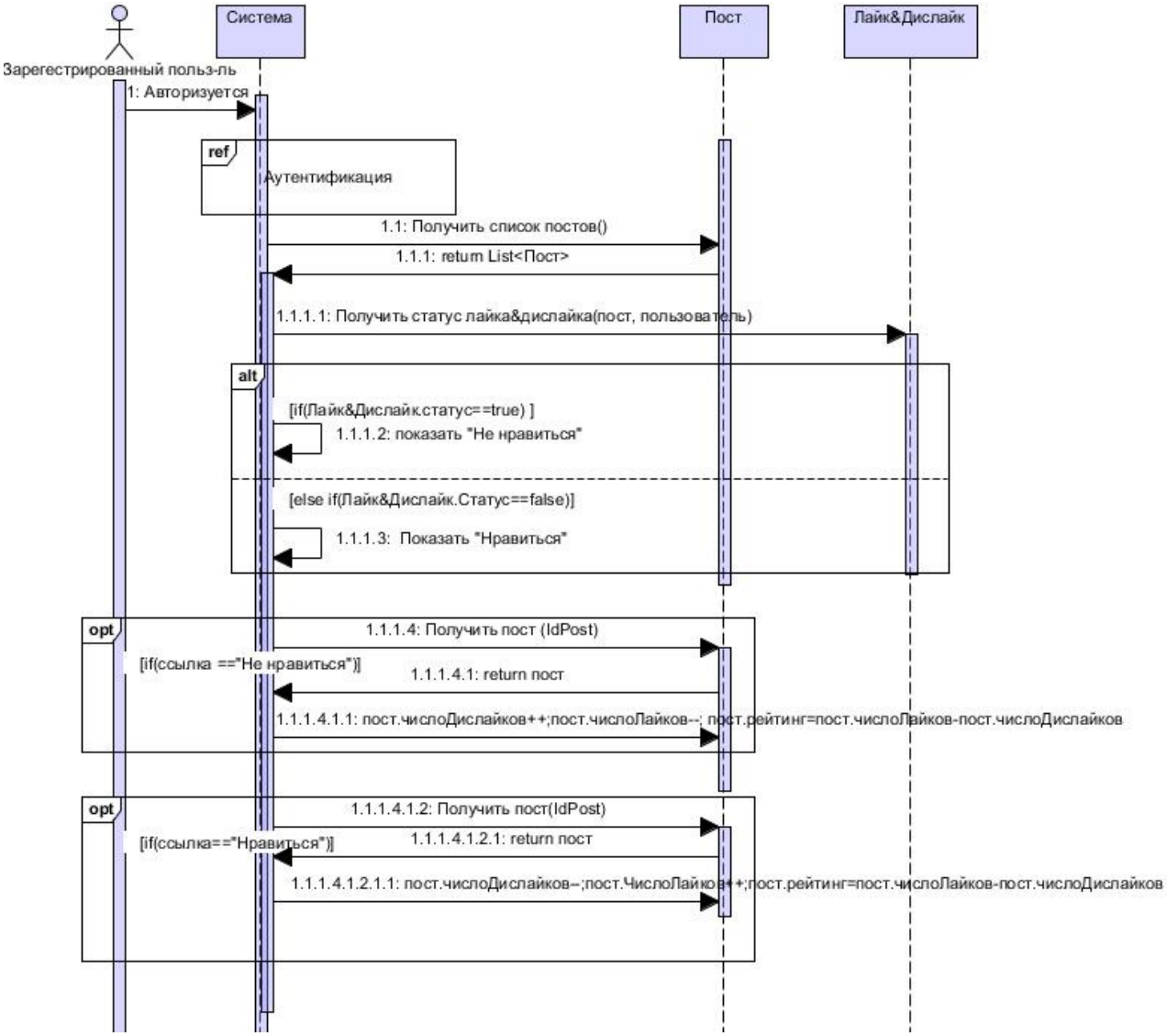


Рис. 2.9. Определение полезности поста.

Диаграмма последовательности процесса комментирования.

В случай успешной аутентификации зарегистрированный пользователь имеет возможность комментирования поста. Переходя по ссылке комментирования и заполнения формы и кликая по кнопке «Добавить» система создает два объекта:

- Объект класса DataComment(),проходит инициализация объекта
- Объект класса DataBlog() и вызывает метод объекта добавления, принимающий объект DataComment(). (см. рис. 2.10.)

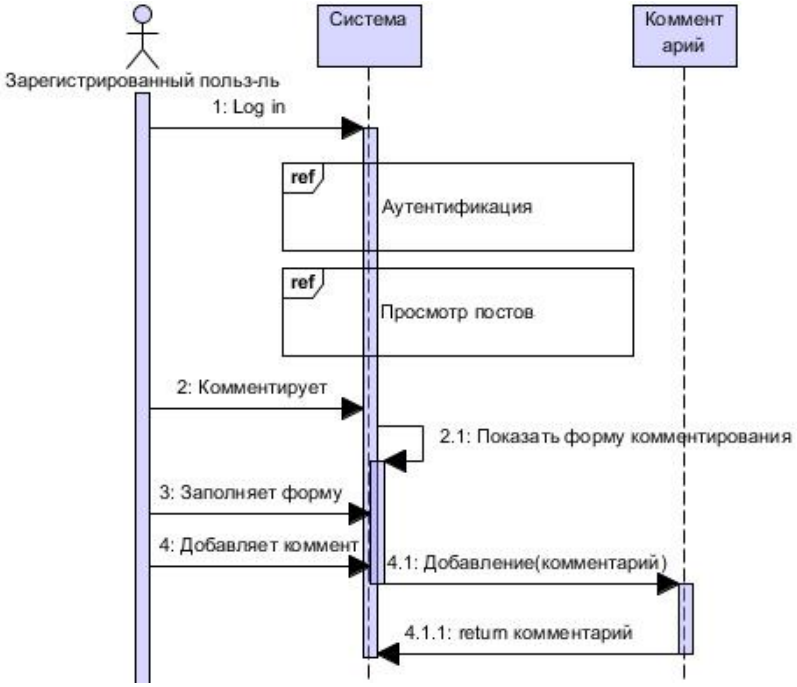


Рис. 2.10. Процесс комментирования.

Диаграмма последовательности процесса добавления поправки на пост. Фрилансеры имеют возможность отсавлять поправки на посты, принадлежащих доскам гостей (см. рис. 2.11.).

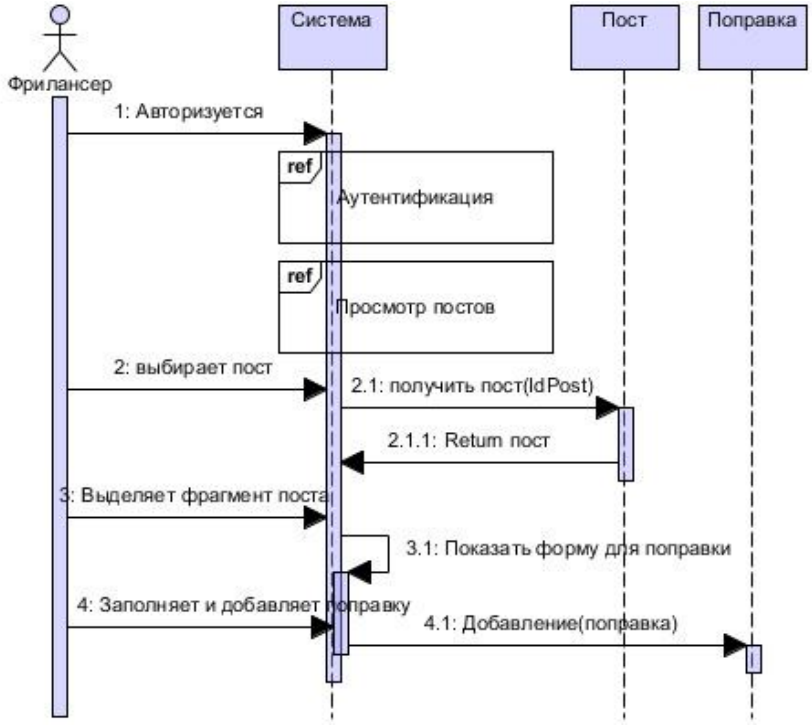


Рис. 2.11. Процесса добавления поправки на пост.

Детализация методов получения постов.

Детализация методов получения поста по уникальному ключу и получение по определенному направлению (см. рис. 2.12.).

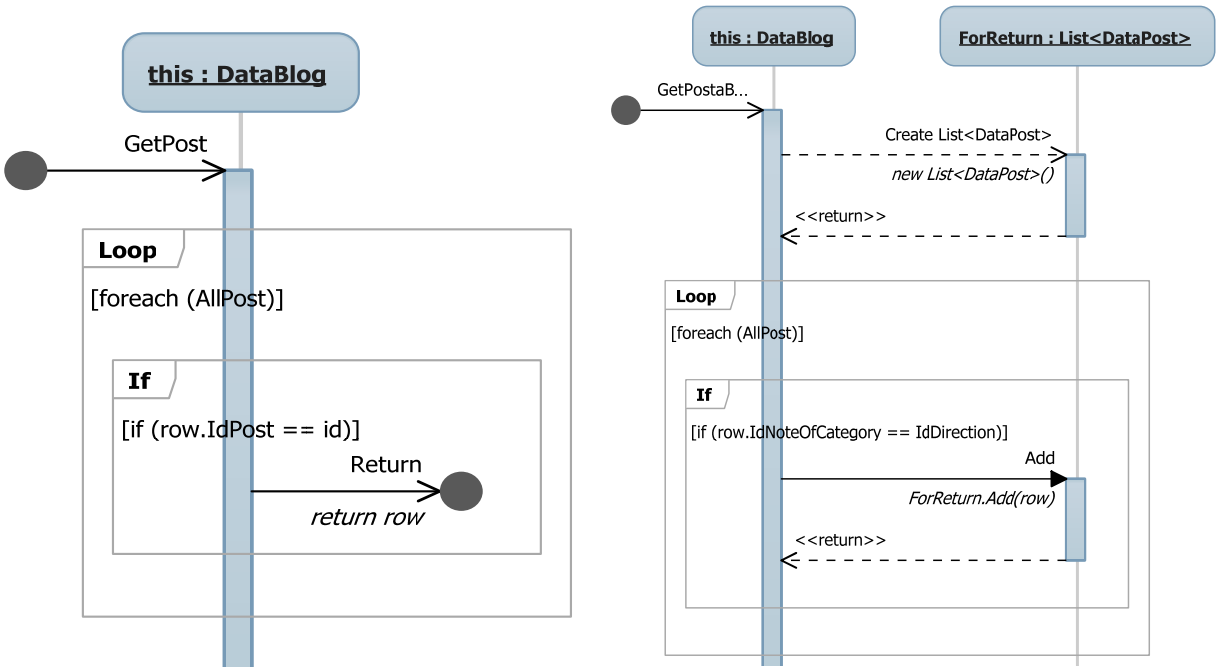


Рис. 2.12. Детализация методов получения постов.

2.3. Объектная модель подсистемы

Объектная модель разрабатываемой подсистемы «Блог» Веб-портала «Software.uz» представляет собой совокупность классов, разделяемых на классы обработчики страниц (в том числе и пользовательских контроллов), классы бизнес логики и объектно-реляционного представления.

Классы объектно-реляционного представления, начинаются с приставки Provider и представляют собой строго структурированные классы, где общедоступными методами могут быть лишь четыре типа методов – это методы по выборки, добавлению, удалению, обновлению данных БД. Остальные методы, вспомогательные которые являются закрытыми, они не должны напрямую взаимодействовать с БД, и могут лишь включают в себя логику которая неоднократно используются в основных методах класса объектно-реляционного представления [4].

Классы бизнес логики, начинаются с приставки Data и представляют собой отражение объектов используемых подсистемой для выдачи контента пользователю, обработки данных.

Классы начинающиеся со слова Provider – это классы предназначеные для реализации запросов к базе данных в виде логически сгруппированных методов(таких, как выборка данных, редактирование, добавление записей и удаление).

Классы бизнес-логики блога поделены на 3 категории:

1) Провайдеры
2) Классы объектов
3) Классы страниц

Подробная схема приведена в Приложении 1.

Подводя итоги можно сказать что Web-приложение базируется на архитектуре классов разделенных на классы, начинающиеся со слова Provider (классы объектно-реляционного представления) которые работают на прямую с БД, и никакой другой класс не имеет право на обращение к БД кроме этих классов. И классы, начинающиеся со слова Data (Бизнес логика) отвечающие за обработку данных и дополнительную манипуляцию над ними. Структурно это можно отобразить следующим образом (см. рис. 2.13.):

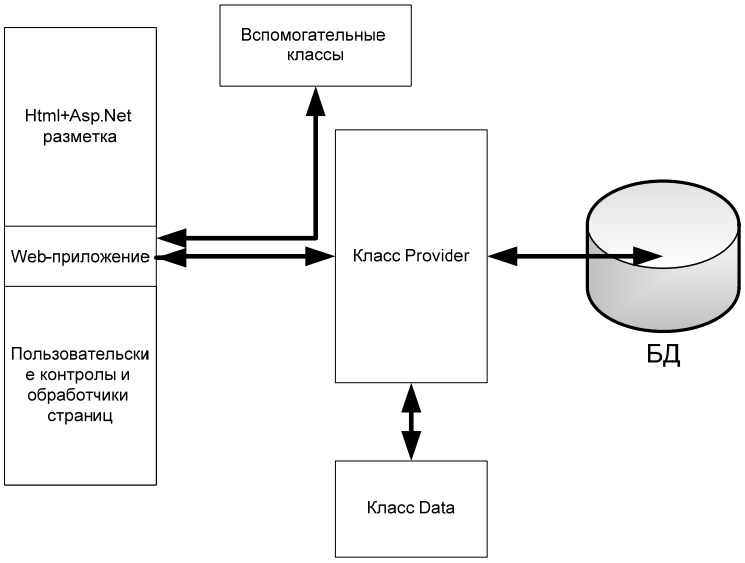


Рис. 2.13. Архитектура приложения.

Один из главных плюсов данной архитектуры это появление у Web-приложения высокой степени расширяемости, так как:

1. В случае изменения БД, нет необходимости искать места использования таблиц во множестве классов, обработчиках страниц и пользовательских контролах, достаточно только подкорректировать соответствующий класс Provider.
2. В случае создания нового функционала для Web-приложения нет необходимости писать код для работы с БД, так как он уже написан, и достаточно просто обратиться к соответствующему Provider классу.
3. Намного легче выявлять ошибки. Так как программист чаще всего допускает ошибки в логике программы. Учитывая факт, что такая архитектура позволяет облегчить объемы кода, и сделать его более читабельным то соответственно поиск ошибок будет занимать меньше времени.

2.4. Структура базы данных

Для работы подсистемы необходима база данных, в которой хранятся сведения касающиеся постов и пользователей.

Одним из основных преимуществ реляционного подхода к организации баз данных является то, что пользователи реляционных БД получают возможность эффективной работы в терминах простых и наглядных понятий таблиц, их строк и столбцов без потребности знания реальной организации данных во внешней памяти [5].

Базовым требованием к реляционным СУБД является наличие мощного и в тоже время простого языка, позволяющего выполнять все необходимые пользователям операции. В последние годы таким повсеместно принятым языком стал язык реляционных БД SQL - Structured Query Language.

В соответствии с поставленной задачей была дополнена БД портала (см. рис. 2.14.) с использованием Microsoft SQL Server 2008 R2.

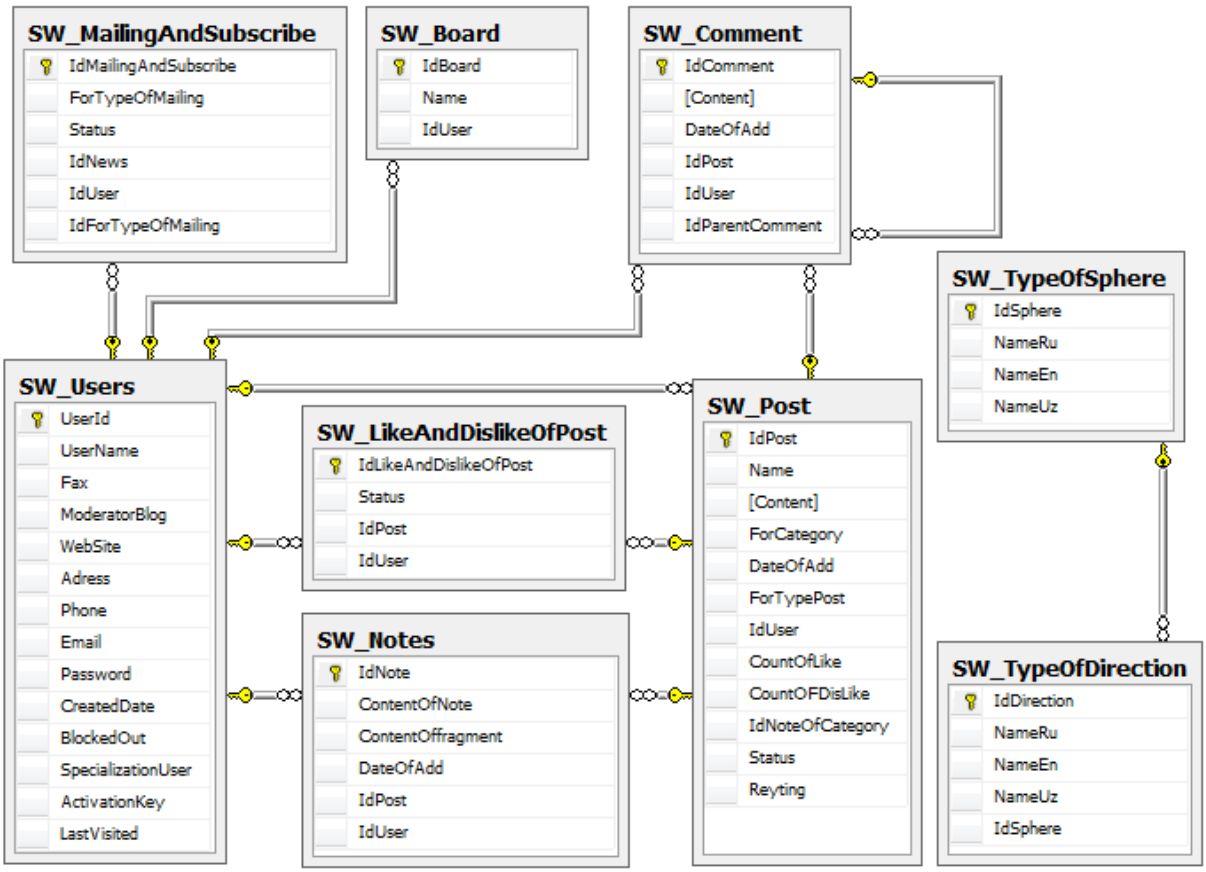


Рис. 2.14. Реляционная модель БД.

В приложении 2. описаны структуры каждой таблицы.

Вывод по второй главе

Во второй главе были определены постановка задач и требования к функционалу, с помощью унифицированного языка моделирования (UML) были спроектированы диаграмма использования и диаграмма последовательностей каждого функционала. Разработана структура БД.

Разработана объектная модель подсистемы, базирующая на архитектуре классов разделенных на классы, начинающиеся со слова Provider (классы объектно-реляционного представления) которые работают на

прямую с БД, и никакой другой класс не имеет право на обращение к БД кроме этих классов. И классы, начинающиеся со слова Data (Бизнес логика) отвечающие за обработку данных и дополнительную манипуляцию над ними (например конвертация).

Основной особенностью разработанной модели является, появление у подсистемы высокой степени расширяемости

Она позволяет:

- В случае изменения БД, нет необходимости искать места использования таблиц во множестве классов, обработчиках страниц и пользовательских контролах, достаточно только подкорректировать соответствующий класс Provider.
- В случае создания нового функционала для Web-приложения нет необходимости писать код для работы с БД, так как он уже написан, и достаточно просто обратиться к соответствующему Provider классу.
- Намного легче выявлять ошибки. Так как программист чаще всего допускает ошибки в логике программы. Учитывая факт, что такая архитектура позволяет облегчить объемы кода, и сделать его более читабельным то соответственно поиск ошибок будет занимать меньше времени.

ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПОДСИСТЕМЫ.

3.1. Выбор инструмента для реализации

Технология .NET является разработкой компании Microsoft и заявлена как новый этап в развитии средств взаимодействия между приложениями. В настоящий момент она доступна в качестве дополнения .NET Framework к семейству операционных систем MicrosoftWindows, а также в новом продукте WindowsServer 2008. Также ведутся работы по созданию .NET Framework на других операционных системах. Платформа .NET упрощает разработку приложений и повышает надежность кода. В частности, она обеспечивает автоматическое управление временем жизни объектов, нейтральные к языкам библиотеки классов и пересекающие границы языков наследование, обработку исключений и отладку. [6]

Основа .NET - CommonLanguageRuntime (общая среда исполнения языков) опирается на системные службы операционной системы и управляет выполнением кода, написанного на любом современном языке программирования. Набор базовых классов дает доступ к сервисам платформы, которые разработчики могут использовать из любого языка программирования. CommonLanguageRuntime и базовые классы вместе составляют основу .NET платформы. NET предлагает также высокоуровневые сервисы:

- ADO .NET - новое поколение ADO, которое использует XML и SOAP для обмена данными;
- ASP .NET - новая версия ASP, позволяющая использовать любой (.NET совместимый) язык для программирования Web страниц;
- WindowsForms и WebForms - набор классов для построения пользовательского интерфейса локальных и Web-ориентированны приложений. [7]

Развертывание систем на платформе .NET осуществляется особым образом. Исходные коды компилируются не в команды процессора x86 или другие машинные коды. Вместо этого компилятор создает код на Промежуточном Языке Microsoft (Microsoftintermediatelanguage - MSIL). Файл, содержащий MSIL, может выполняться на платформе любого процессора, если операционная система, предоставляет .NET CLR.

Важной составляющей частью платформы .NET является новая среда ASP.NET (ранее использовалось название ASP+). Возможности ASP.NET настолько велики, что ее сложно назвать следующей версией ASP. В ее основе лежит другая платформа, и основными языками программирования для нее выбраны C# и VisualBasic, вместо бывших скриптинг языков. В то же время, новая технология позволяет писать ASP страницы на любом подходящем языке. [8]

В ASP.NET заложено все, для того, чтобы сделать весь цикл разработки web -приложения более быстрым, а поддержку проще. Ниже приведены основные возможности и принципы работы ASP.NET. [9].

- Компилирование кода при первом обращении.
- Широкий выбор библиотек компонентов, поставляемых с .NET.
- Поддержка мощного средства разработки - VisualStudio. NET.
- Языковая независимость в пределах платформ для которых реализована общая
- языковая среда исполнения CLR.
- Возможности расширения с помощью мультипроцессорных и кластерных решений.
- Новые возможности по обработке ошибок.
- Объектно-ориентированные языки разработки (C#).
- Расширенные возможности повторного использования компонент.

Очевидно, что платформа .NET и ASP.NET предоставили новые возможности по разработке Web - систем. Они отвечают всем современным требованиям и позволяют значительно ускорить и упростить разработку сложных приложений.

Рассмотрим платформы по требованиям, основные оценочные характеристики платформ сравним в сводной таблице, где "-" – полное отсутствие поддержки, "-/+" – недостаточная поддержка, "+/-" – поддержка не в полном объеме, и "+" – полная поддержка (см. таблицу 3.1.). Для сравнительных характеристик, таких как язык реализации или производительность, оценки соответствуют степени превосходства технологии.

Таблица 3.1.

Таблица сравнений технологий.

Критерии требований	PHP	JavaServlets	JSP	ASP.NET
Многоплатформенность	+/-	+	+	+/-
Производительность	-/+	+/-	+/-	+
Масштабируемость	-	+	+	+
Язык реализации	+/-	+	+	+
Возможности расширения и интеграции	-	+	+/-	+
Простота использования, наличие средств разработки	+/-	+/-	+	+
Наличие необходимых программных библиотек	+	+	+	+
Разделение дизайна и логики	+/-	-/+	+/-	+

Исходя из вышеперечисленного, актуально в качестве инструмента разработки использовать технологию ASP.NET, за высокие показатели большинства критериев.

Из рассмотренного можно выделить следующие основные подходы к архитектуре серверных приложений:

1. Отдельное выполнение запросов. При каждом запросе динамического содержимого, запускается отдельная программа для обработки запросов. Программа генерирует содержимое, передаваемое клиенту. Этот подход используется в классических CGI-скриптах.
2. Накопление исполняемых процессов. Подход аналогичен предыдущему, но при этом если запрос выполняется повторно, нового запуска программы не происходит, а обработка передается существующему процессу. Данный подход применяется в технологиях JavaServlets, Fast CGI.
3. Шаблоны страниц. При запросе шаблоны заполняются динамическим содержимым, обычно, но необязательно, создаваемым интерпретируемым языком сценариев. Подход применяется в технологиях ASP, JSP, PHP.
4. Расширения Web - сервера. Web - сервер обращается к особым расширениям для обработки динамического содержания. Расширения специфичны для Web - сервера. Этот подход используется в IS API, NSAPI, mod_perl.

3.2. Описание работы блога

Подсистема предоставляет функциональные возможности в зависимотсти от роли пользователя (см. рис. 3.1.), среди них Фрилансеры и Гости обладают наиболее расширенными возможностями.

Формирование правого меню в зависимости от роли пользователя

При переходе в подсистему «Блог», формируется главная страница блога, где формируется правое меню и список последних добавленных постов. Правое меню формируется в зависимости от роли пользователя:

— Незарегистрированный пользователь – рубрика постов, поиск;

— Организации – мои новости, рубрика постов, поиск;

— Фрилансер – мои направления, личный кабинет, мои новости, рубрика постов, поиск;

— Гость – мои доски, личный кабинет, добавить доску, мои новости, рубрика постов, поиск;

— Модератор – кабинет модератора;

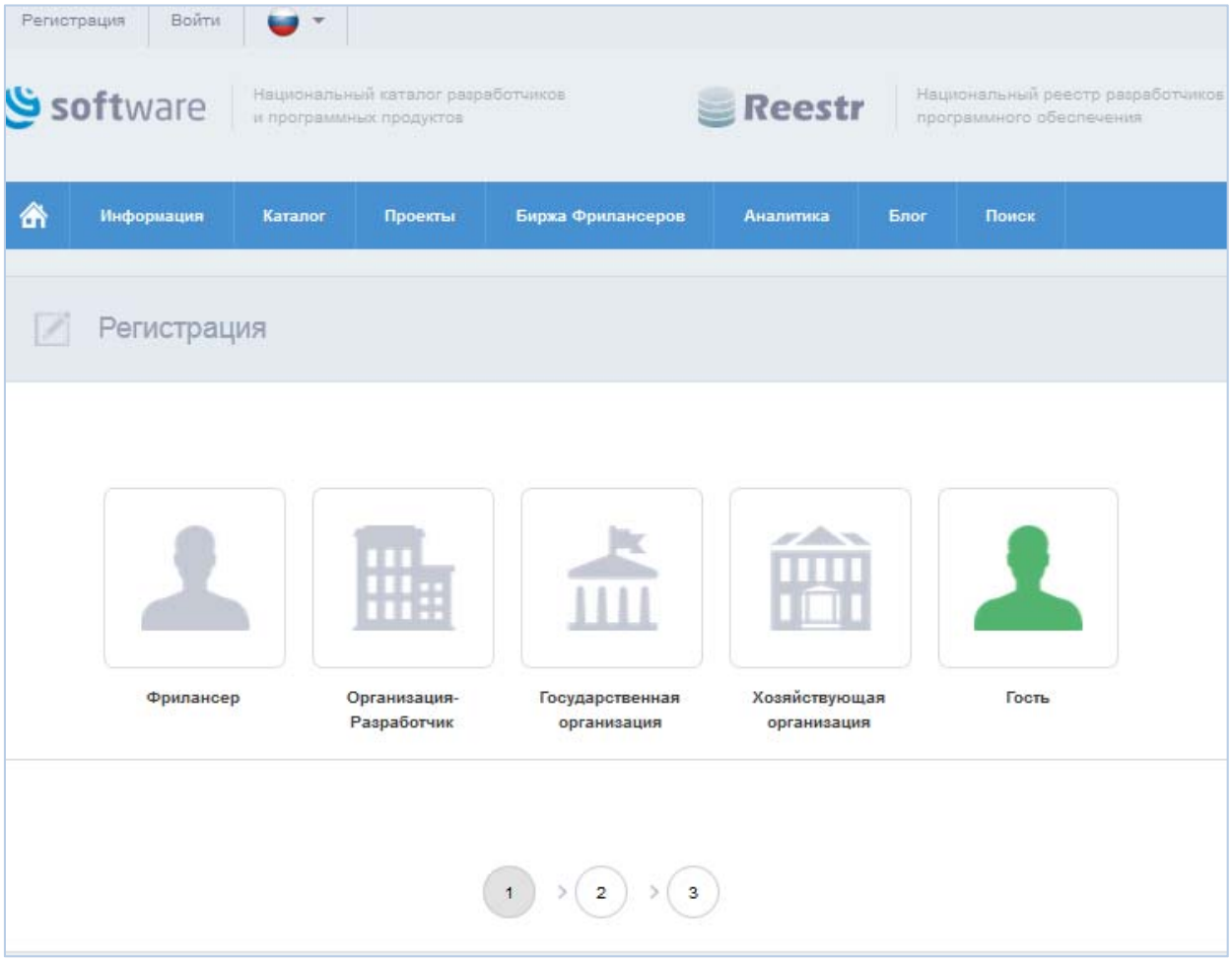


Рис. 3.1. Категории зарегистрированных пользователей подсистемы.

Формирование списка постов при просмотре

Существует четыре условий, в зависимости от которых меняется содержимое списка постов.

— Список всех постов (Главная);

— Список постов по определенному направлению и различным пользователям;

— Список постов по определенной доске;

— Список постов по определенному направлению и определенному пользователю;

— Список постов, на которые пользователь был подписан.

Список всех постов. Если пользователь находится на «Главной» странице, то отображаются все имеющийся посты в блоге в обратном хронологическом порядке.

Список постов по определенному направлению и различным пользователям (рубрика). Пользователь переходит по определенному направлению в «Рубрике постов» или в «Моих направлениях», затем на текущей странице формируется список всех постов, принадлежащих текущему направлению (см. рис. 3.2).

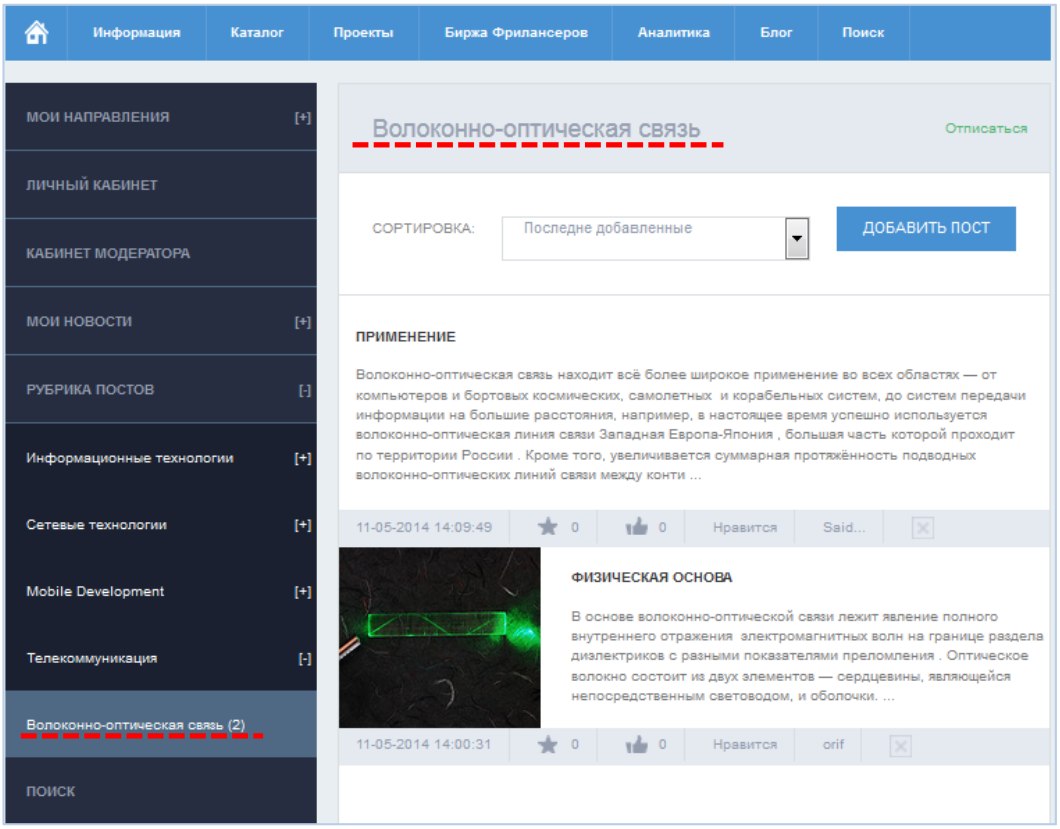


Рис. 3.2. Список постов по определенному направлению и различным пользователям.

Список постов по определенному направлению и определенному пользователю. Пользователь переходит по определенному направлению (см. рис. 3.3.) в личном профиле, как по своему, так и по определенному другому пользователю, затем на текущей странице формируется список постов, принадлежащих текущему направлению и текущему пользователю (см. рис.3.4.).

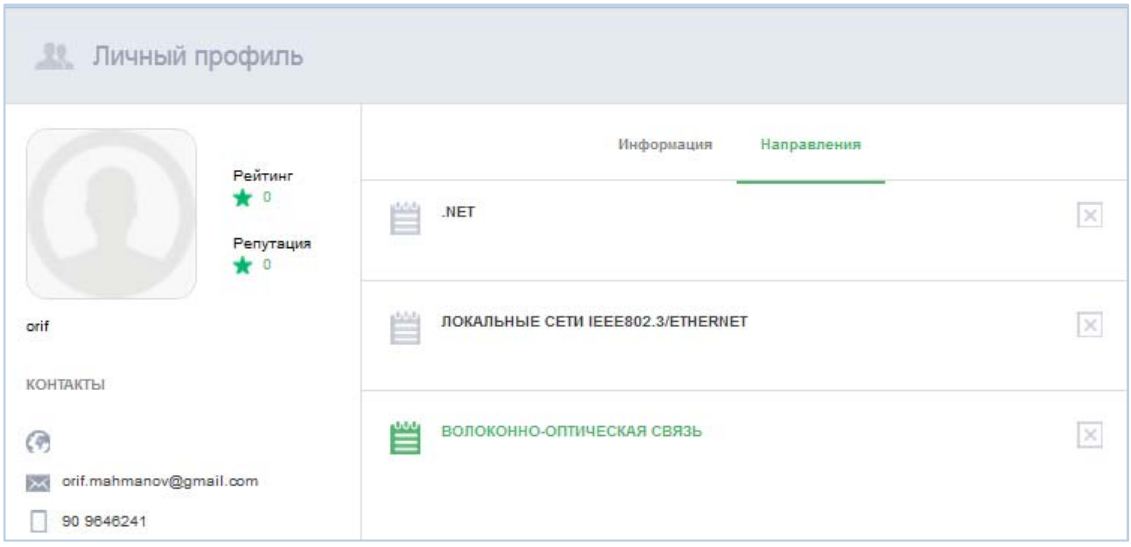


Рис. 3.3. Список направлений пользователя в личном профиле.

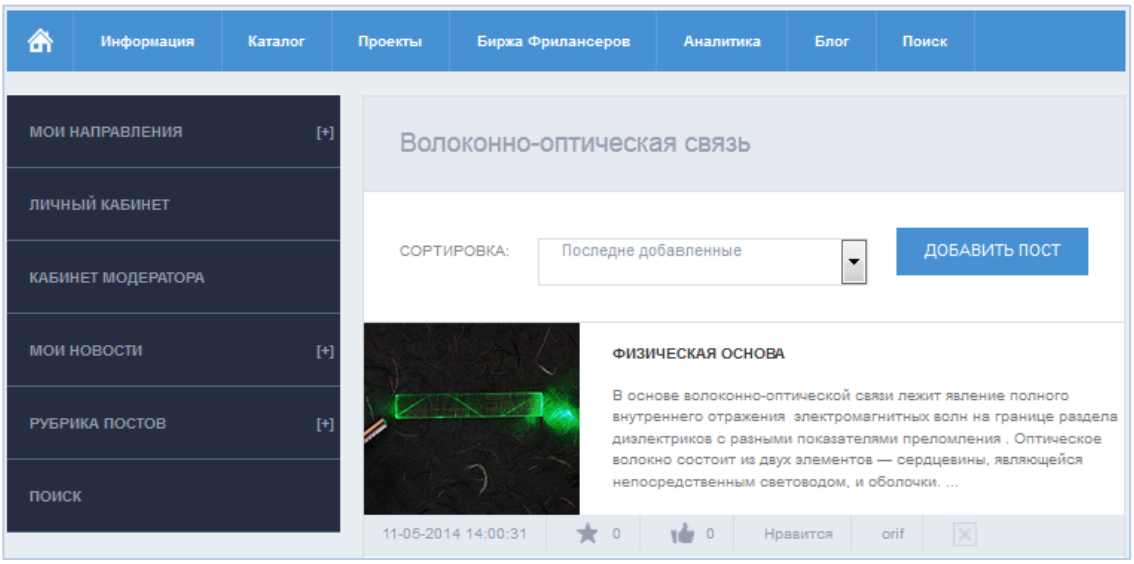


Рис. 3.4. Список постов по определенному направлению и определенному пользователю.

Список постов по определенной доске. Если пользователь является гостем, у него есть возможность сознания досок для размешения постов.

Переходя по определенной доске в правом меню или в личном кабинете (см. рис. 3.5, 3.6) формируется список всех постов, принадлежащих текущей доске (см. рис. 3.7.).

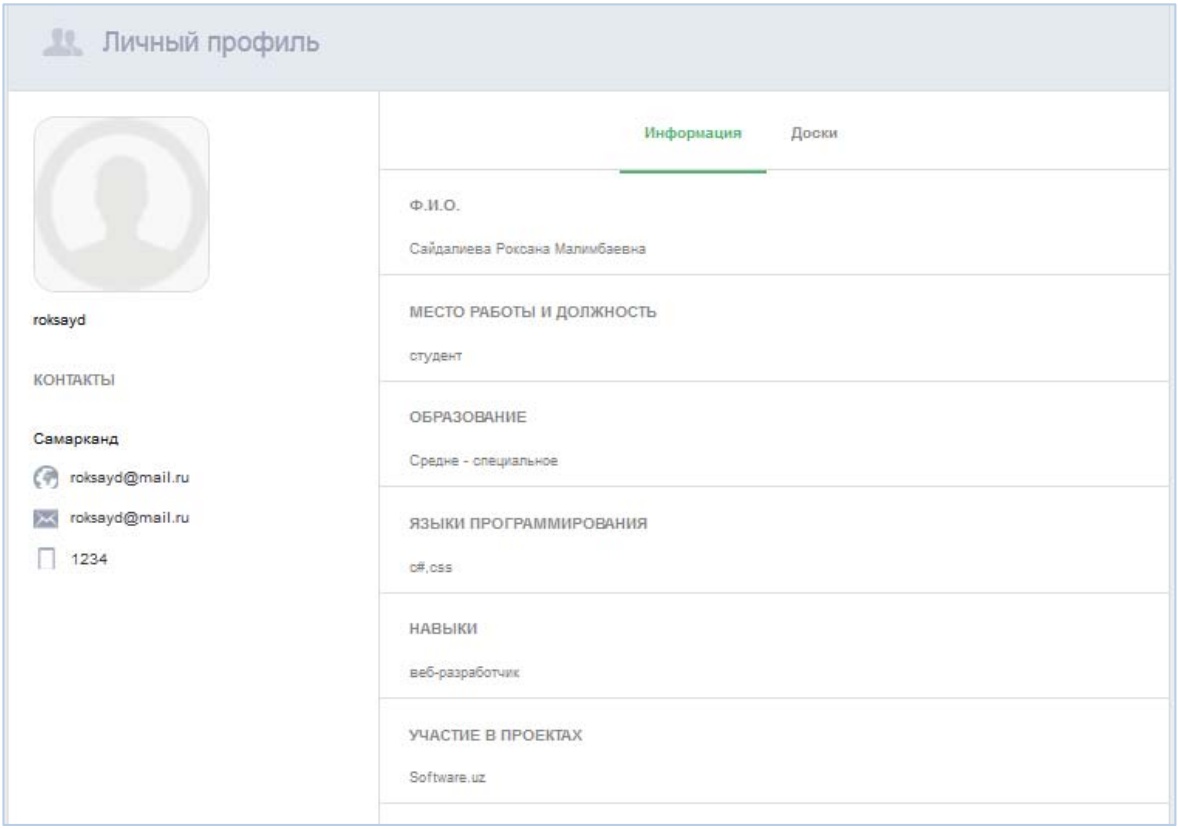


Рис. 3.5. Личный кабинет пользователя в подсистеме «Блог». Информация о пользователе.

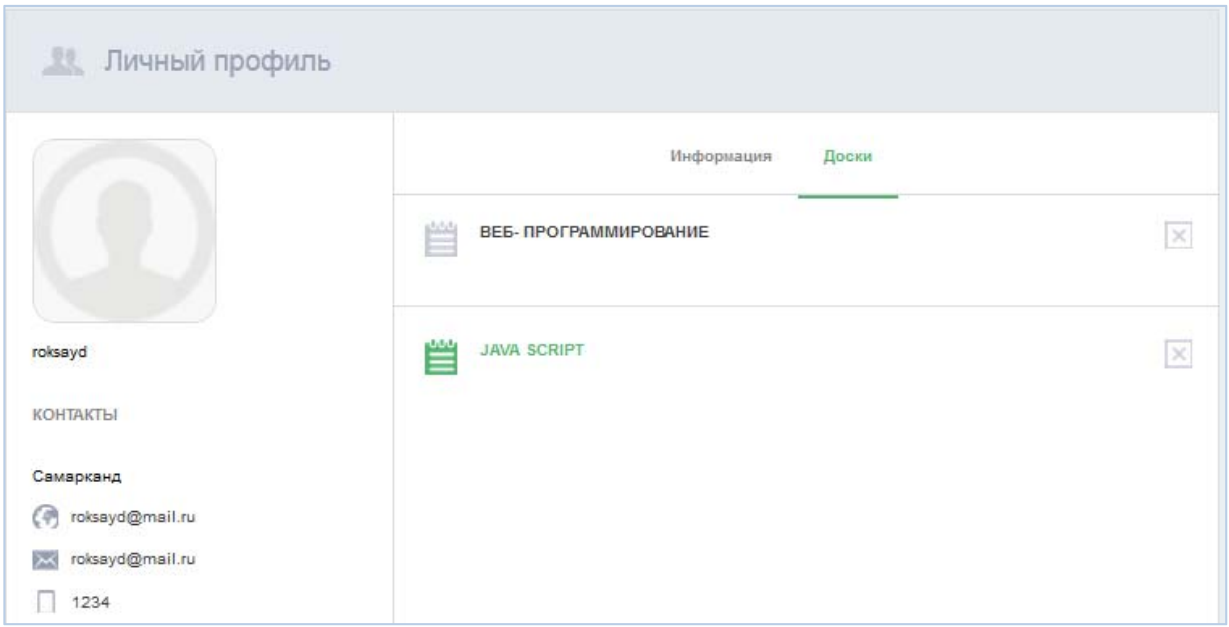


Рис. 3.6. Список досок пользователя.

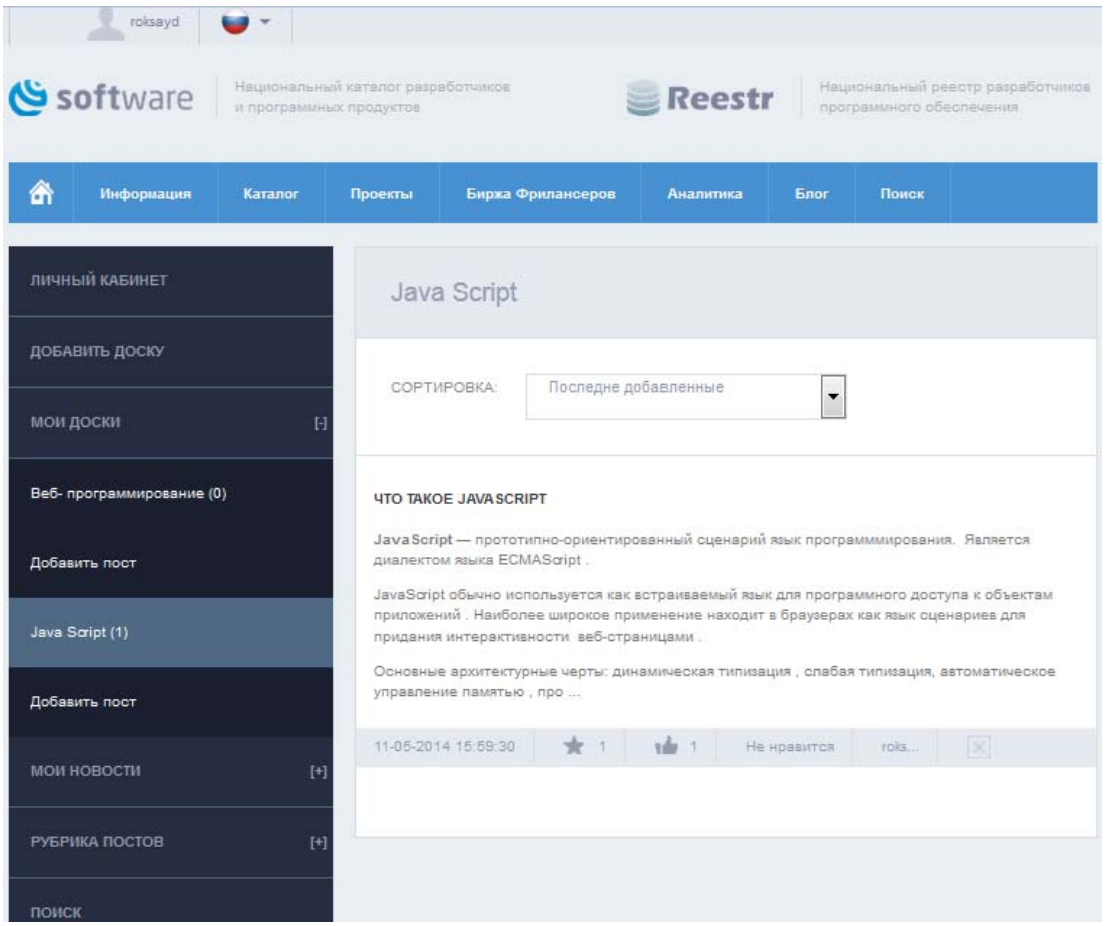


Рис. 3.7. Список постов по доске пользователя.

Добавление поста

Было реализовано три способа добавления постов:

— Первый способ – «Общее добавление»;
— Второй способ – по «Моим направлениям»;
— Третий способ – по «Моим доскам»;

Общее добавление. В блоге существуют двухуровневое разделение тематик: первый уровень-сферы, второй уровень-направления.Модератором добавляется сферы и направления, необходимые для фиксации места размещения поста. При первом добавлении поста фрилансеру необходимо нажать на кнопку «Добавить пост» (см. рис. 3.8.) затем необходимо указать соответстующее направление на странице добавления. (см. рис. 3.9.) Как сформируется страница добавления поста, пользователю необходимо заполнить все поля.

По "моим направлениям". Предоставляется только в том случае, если фрилансер ранее размещал посты в определенные направления. Для более быстрого поиска интересующих направлений в правом меню формируются все направления пользователя, где были размещены посты. Фрилансер

кликает по «Мои направления» повляется выпадающий список сфер, внутри которых имеются направления, принадлежащих данной сфере. Фрилансер выбирает определенное направление в меню и переходит по ссылке «Добавить пост». (см. рис. 3.10.) Далее формируется страница «Добавления поста», где уже автоматически пост добавляется в текущее направление (см. рис. 3.11.)

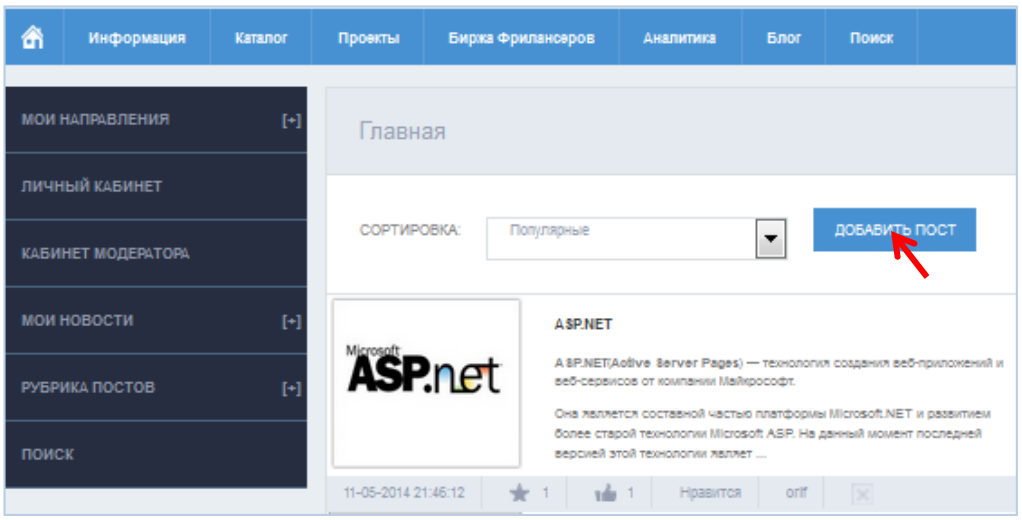


Рис. 3.8. Переход на добавление поста.

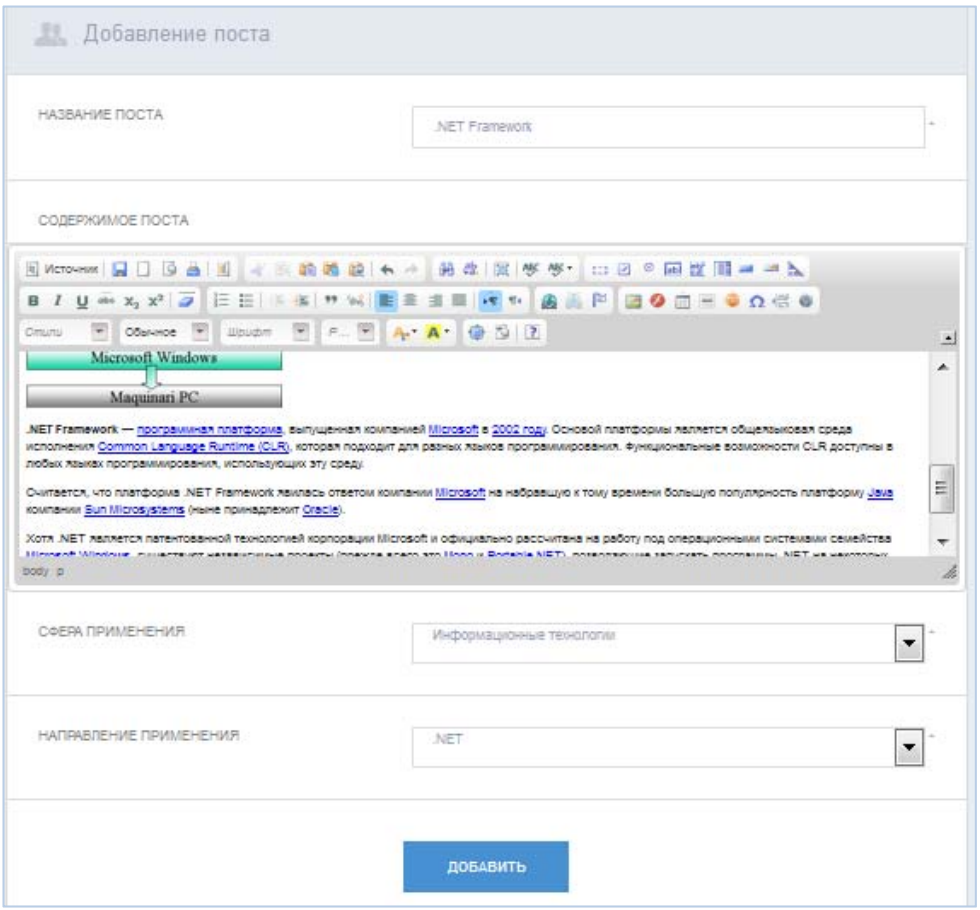


Рис. 3.9. Размещение поста по направлению.

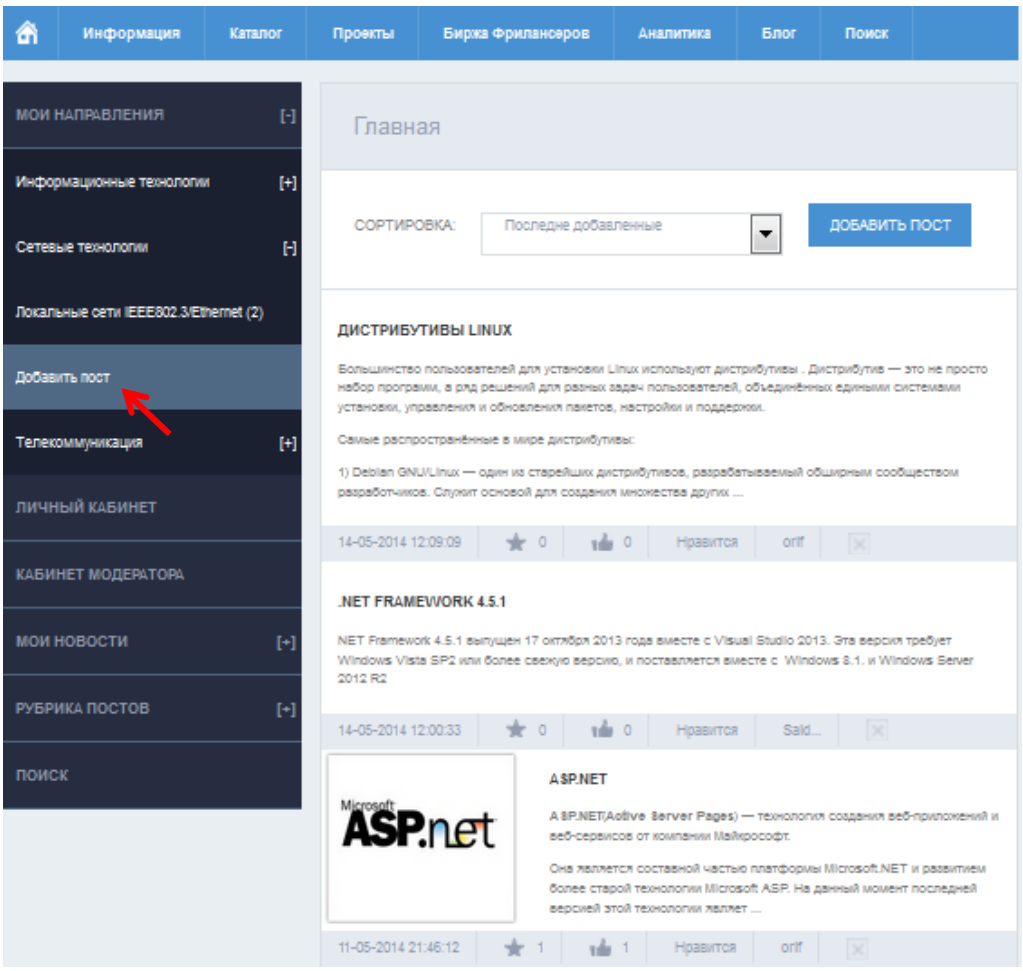


Рис. 3.10. Переход на добавлению по направлению фрилансера.

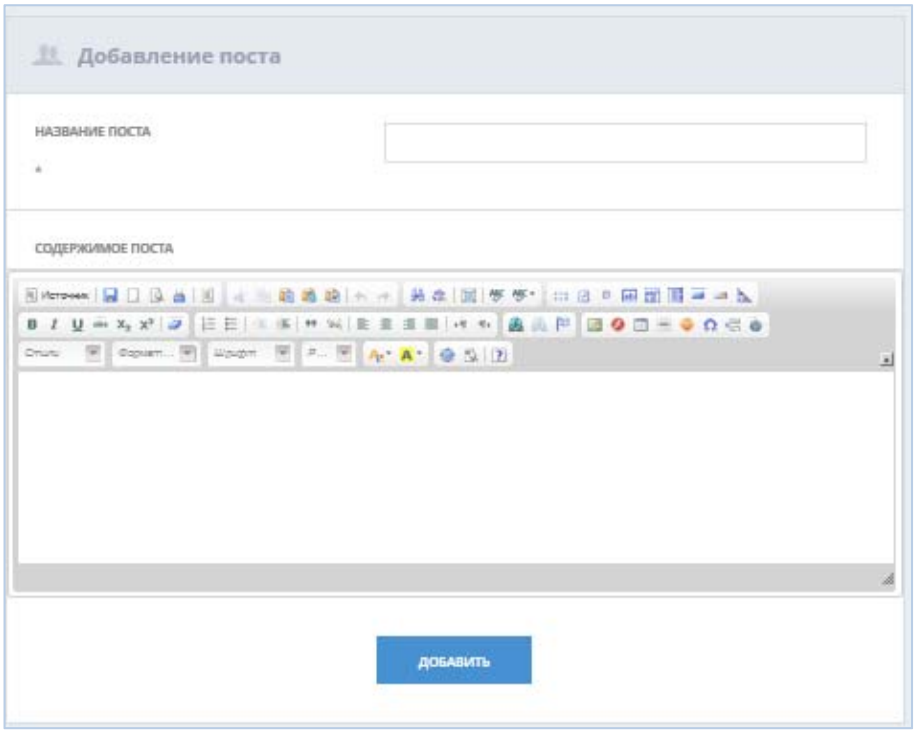


Рис. 3.11. Автоматическое добавление поста по текущему направлению.

По “Доскам”. Предоставляется только в том случае, если гость ранее добавил определенную доску. В правом меню формируется список всех добавленных досок гостя. Гость выбирает определенную доску в меню и переходит по ссылке «Добавить пост». (см. рис. 3.12.)

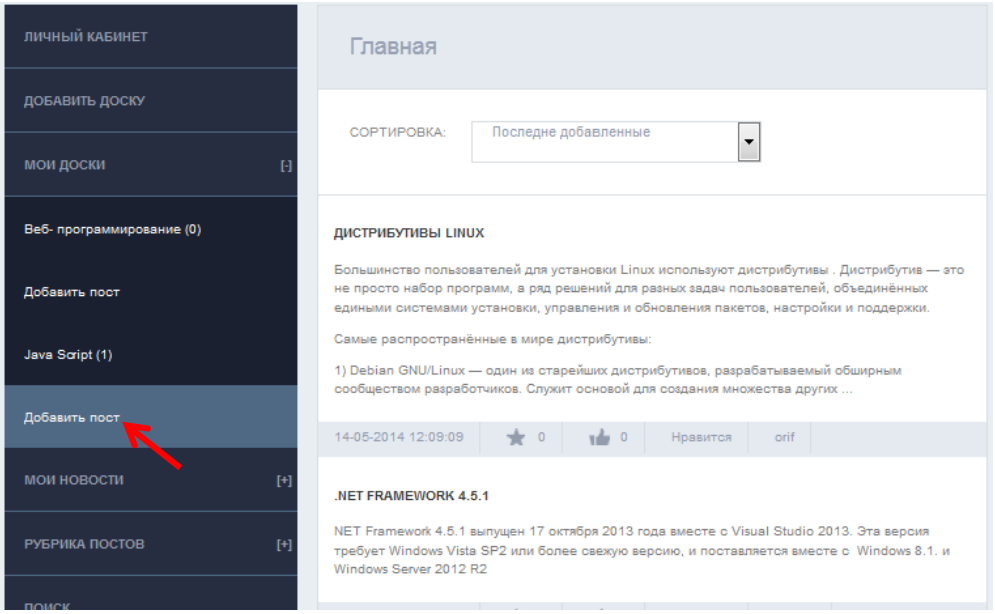


Рис. 3.12. Переход на добавление поста по доске гостя.

Далее формируется страница «Добавления поста», где уже автоматически пост добавляется в текущую доску (см. рис. 3.13.).

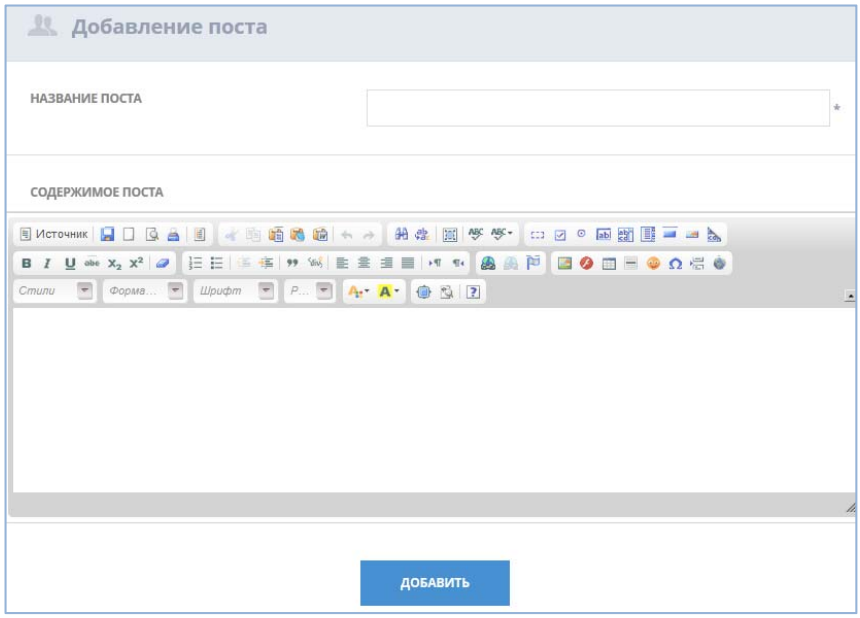


Рис. 3.13. Автоматическое добавление поста по текущей доске.

После процесса добавления поста, формируется новая страница отображения ранее указанного направления или доски при добавлении поста, где содержаться все посты, относящиеся данному направлению или доске.

Детальный просмотр

Переходя по ссылке “Название поста” (см. рис. 3.14.) формируется новая страница “Детального просмотра поста” (см. рис. 3.15.).

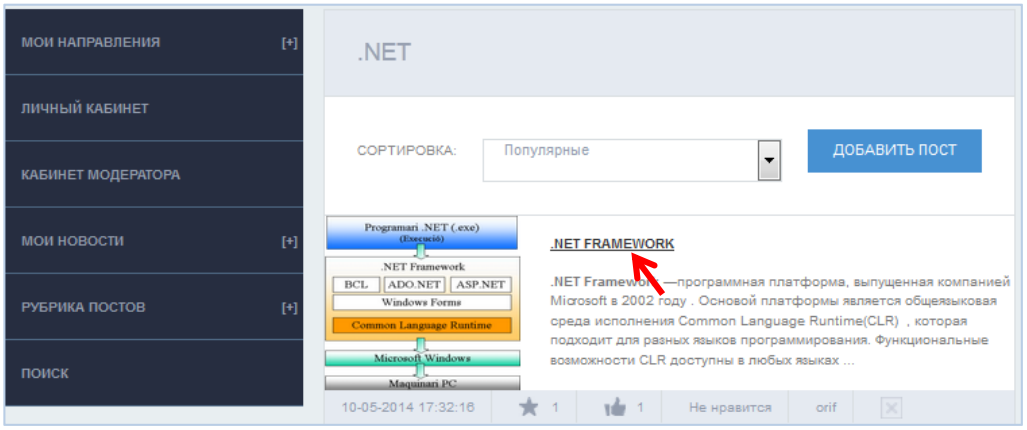


Рис. 3.14. Переход по название поста для детального просмотра.

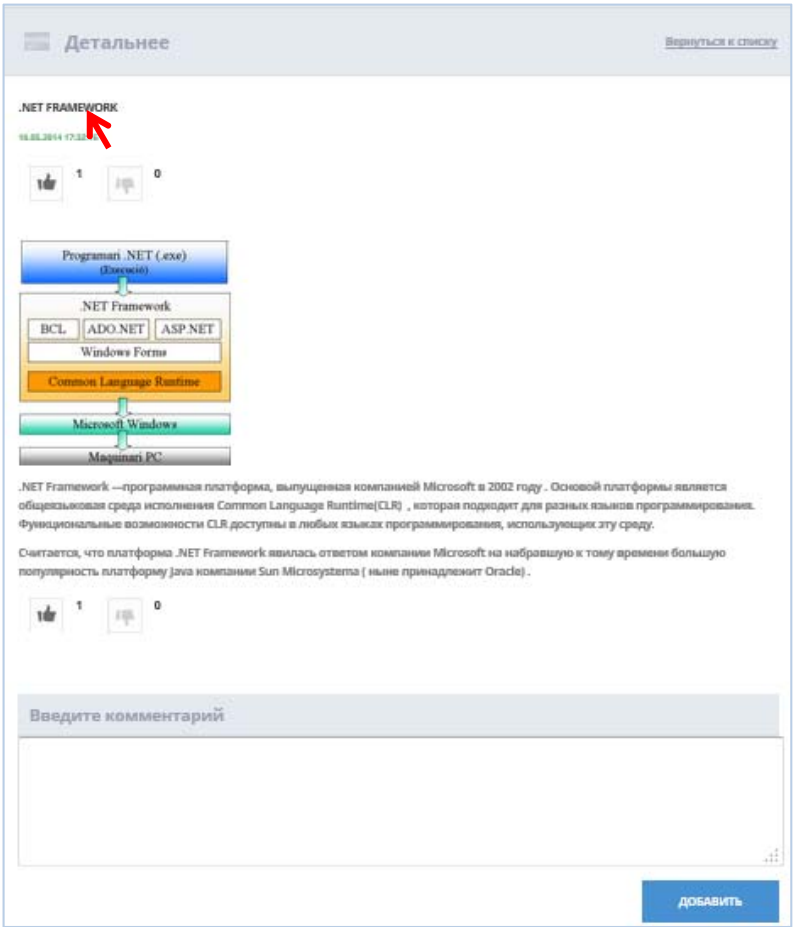


Рис. 3.15. Детальный просмотр.

Удаление поста

Кликая на знак “Крестик” пользователь удаляет свои посты. (см. рис. 3.16.)

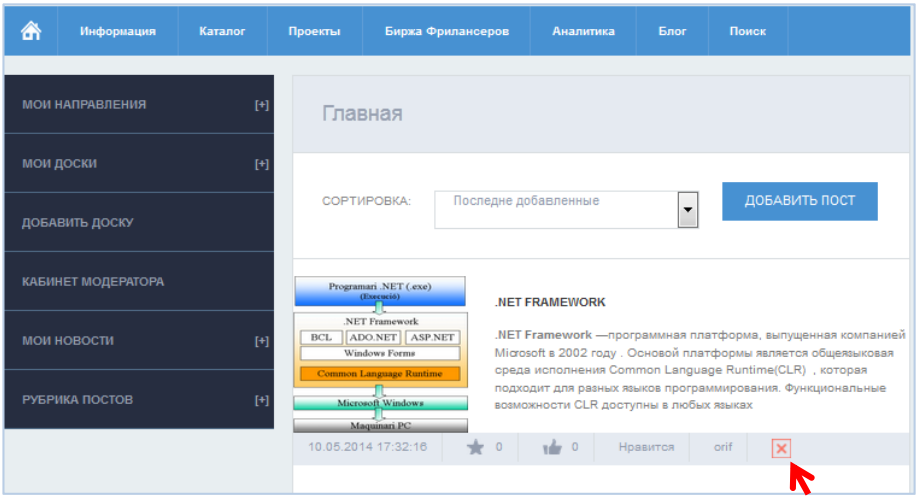


Рис. 3.16. Удаление поста.

Комментирование

При добавлении комментария необходимо заполнить поле под заголовком “Введите комментарий” на странице детальнее. Далее необходимо кликнуть на кнопку “Добавить”. Далее та же страница перезагружается с новым добавленным комментарием. Переходя по ссылке ответить пользователь отвечает на текущий комментарий. Переходя по ссылке «Удалить», комментарий удаляется (см. рис. 3.17.).

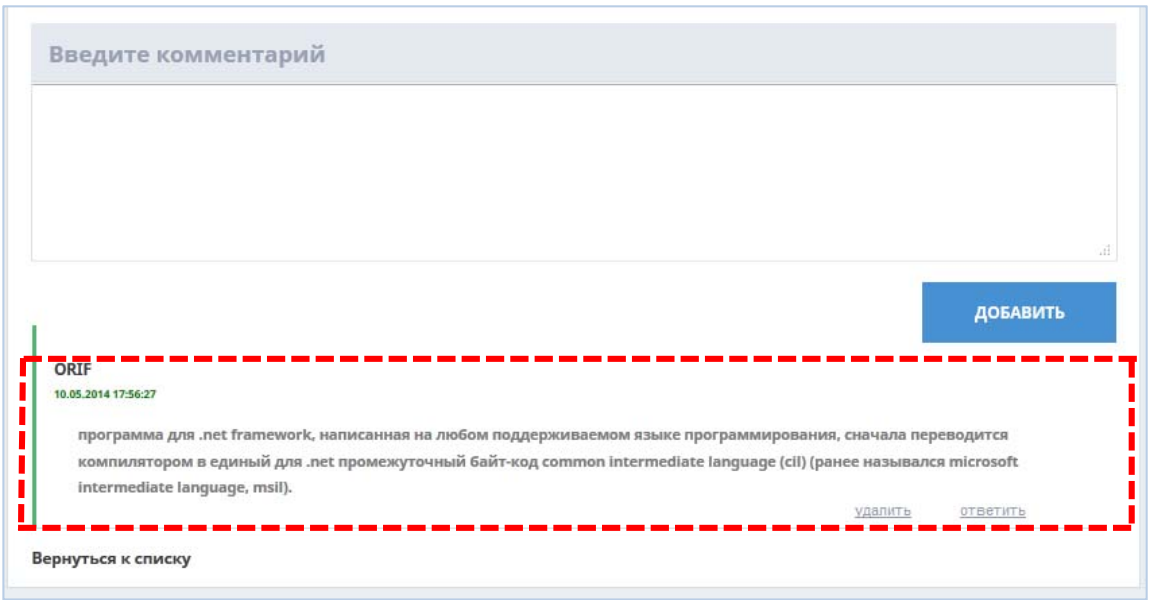


Рис.3.17. Отображение добавленного комментария.

Сортировка

Пользователю необходимо выбрать один из элементов в выпадающем списке в основной части страницы (см. рис. 3.18.).

Если выбрать элемент «Популярные», то список постов формируется по рейтингу в возрастающем порядке (см. рис. 3.19.).

Если выбрать элемент «Последне добавленные», то список постов формируется в обратном хронологическом порядке (см. рис. 3.20.).

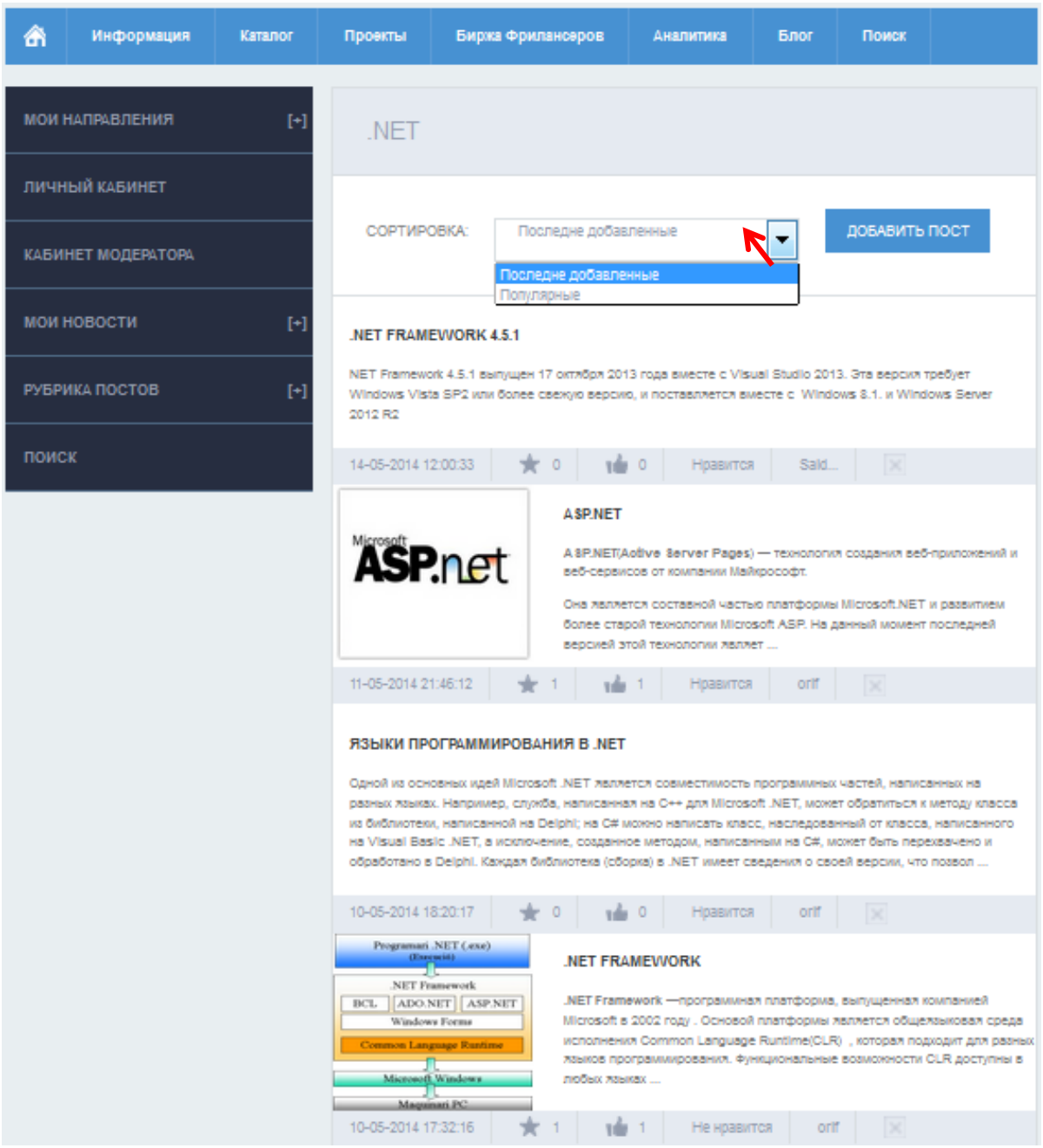


Рис. 3.18. Выбор типа сортировки.

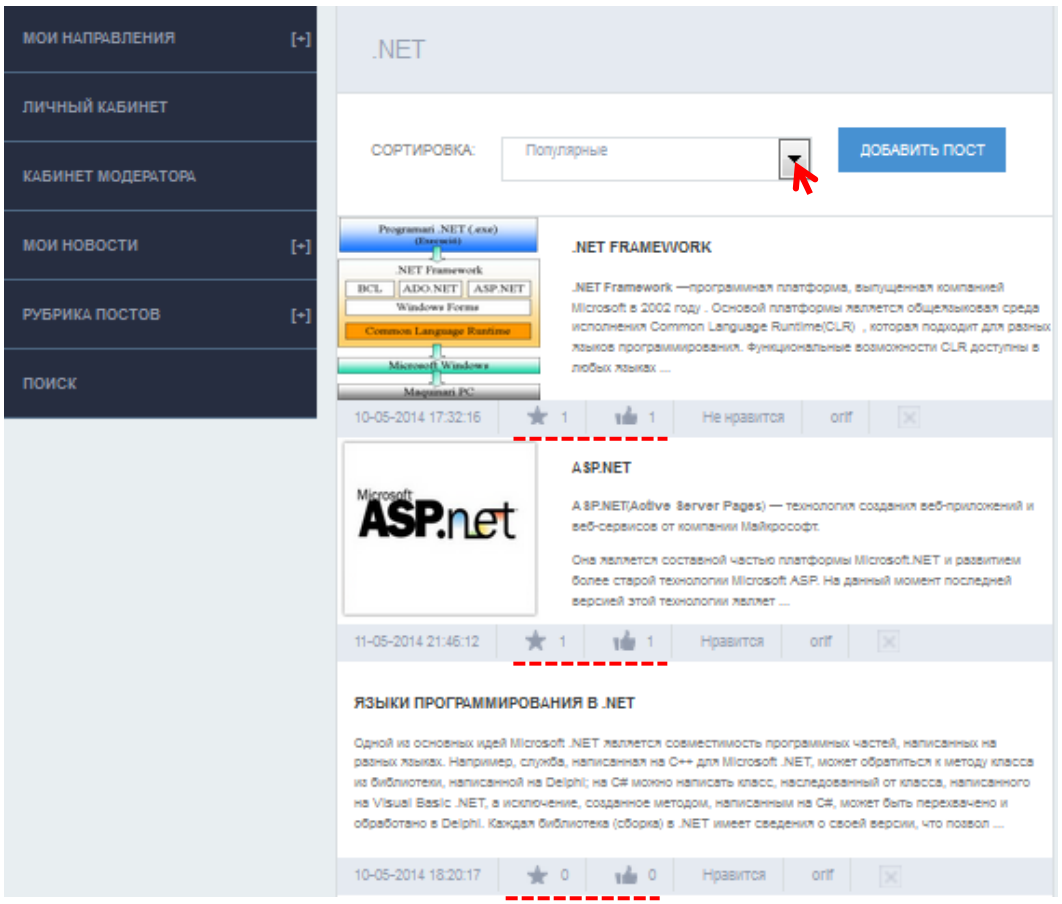


Рис. 3.19. Сортировка по популярности.

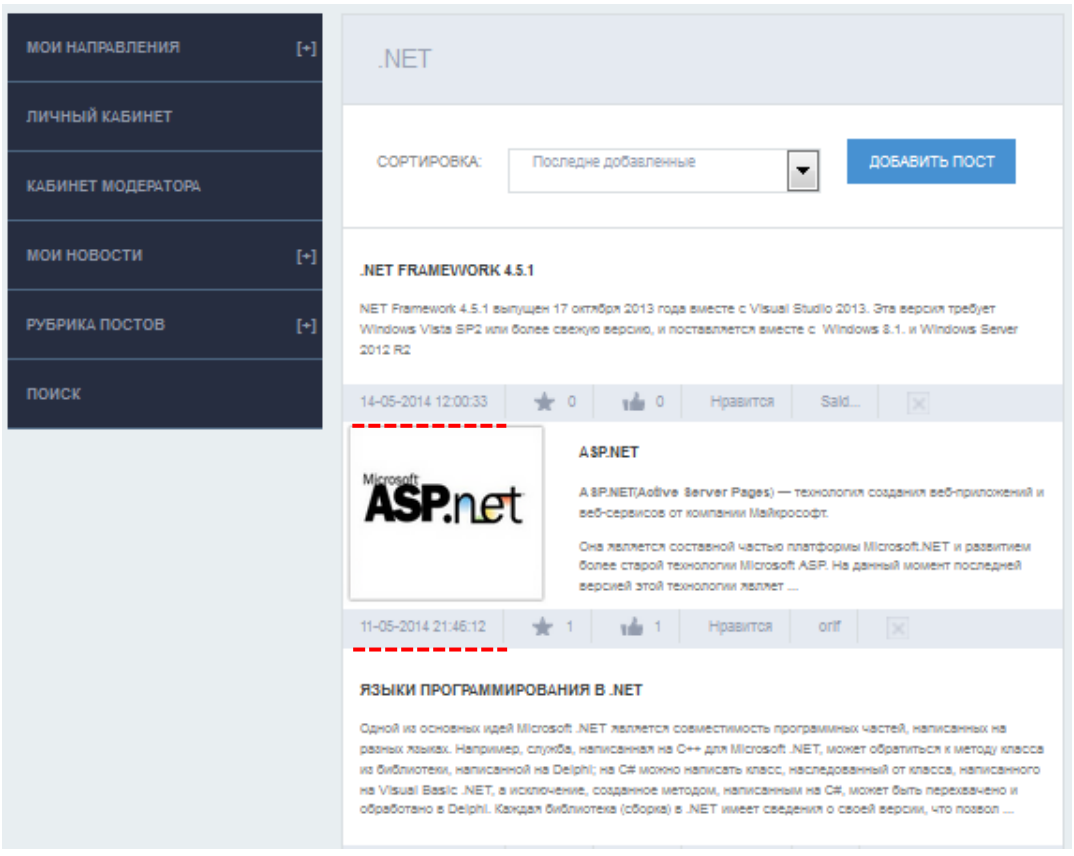


Рис. 3.20. Сортировка по последне добавленным.

Получение уведомлений

Если пользователи комментируют чей-либо пост или же отвечает на чей либо комментарий, в результате владельцу поста владельцу комментария приходит эти ответы в виде уведомлении в правом меню и идет подсчет на количество новых ответов. Как только пользователь просматривает данные ответы, то счетчик уменьшает количество ответов соотвественно, а прочитанные ответы приобретают зеленый цвет. Так же пользователь может полностью удалить определенное уведомление, нажав на кнопку «Крестик». (см. рис. 3.21.)

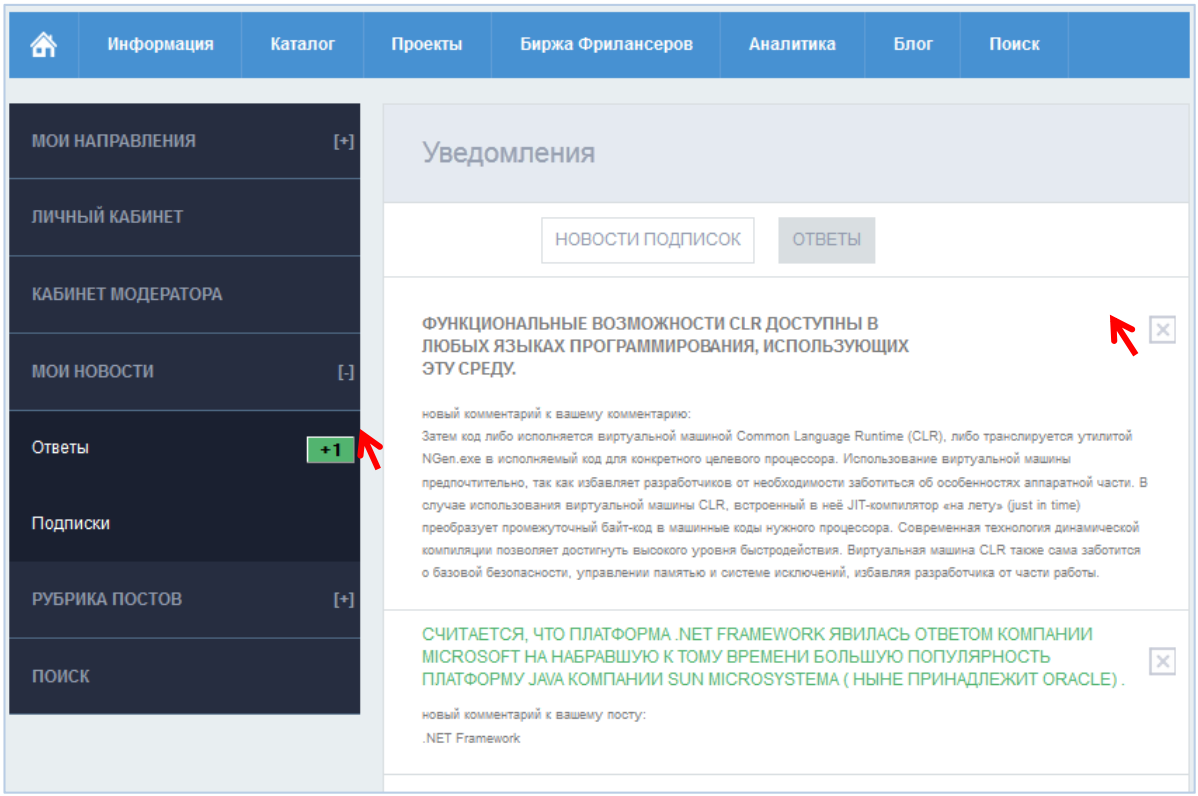


Рис. 3.20. Уведомление ответов.

Процесс подписок

Зарегистрированным пользователям предоставляется возможность подписаться на определенное направление для получения новой информации. Пользователь переходит по определенному направлению в рубрике постов, затем на новой сгенерированной странице кликает на ссылку «Подписаться». (см. рис. 3.21.)

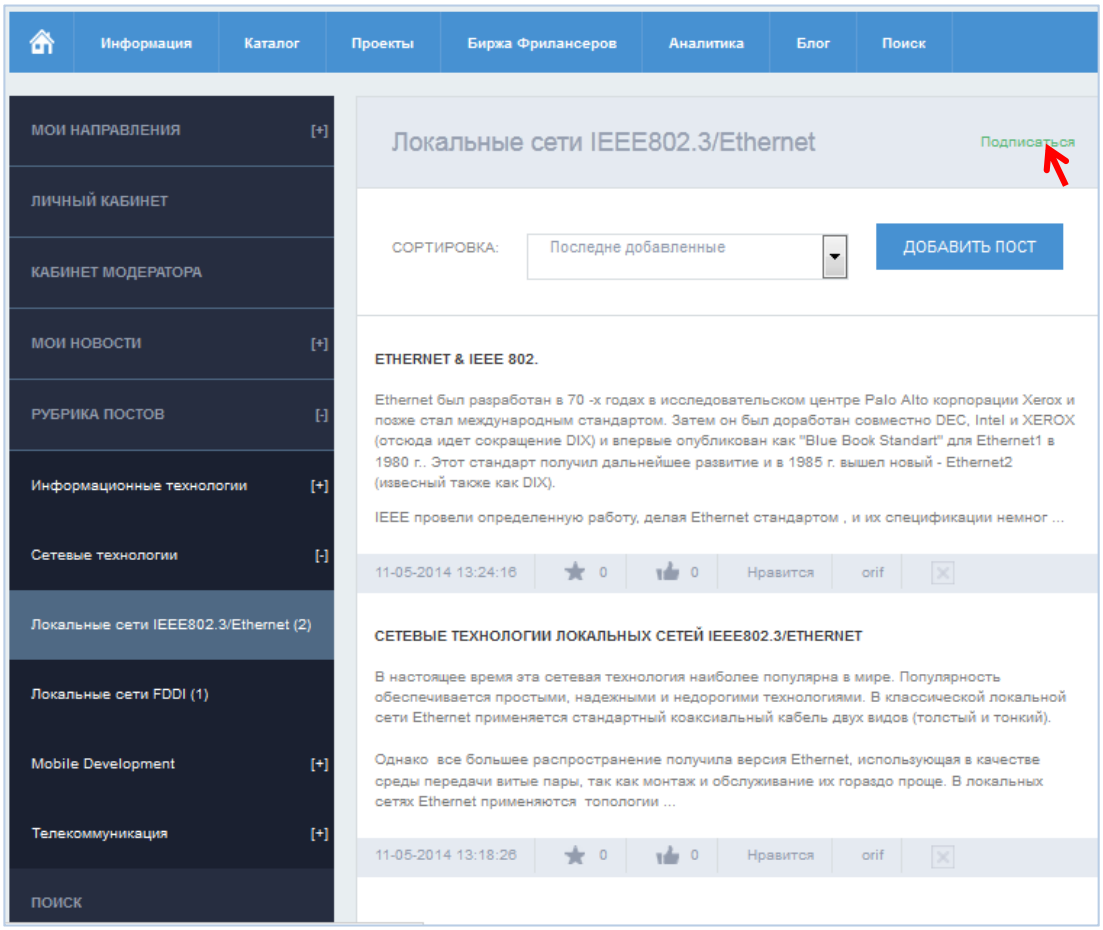


Рис. 3.21. Подписка на направление.

Если пользователь подписан на данное направление, ему предоставляется отписка от текущего направления (см. рис. 3.22.).

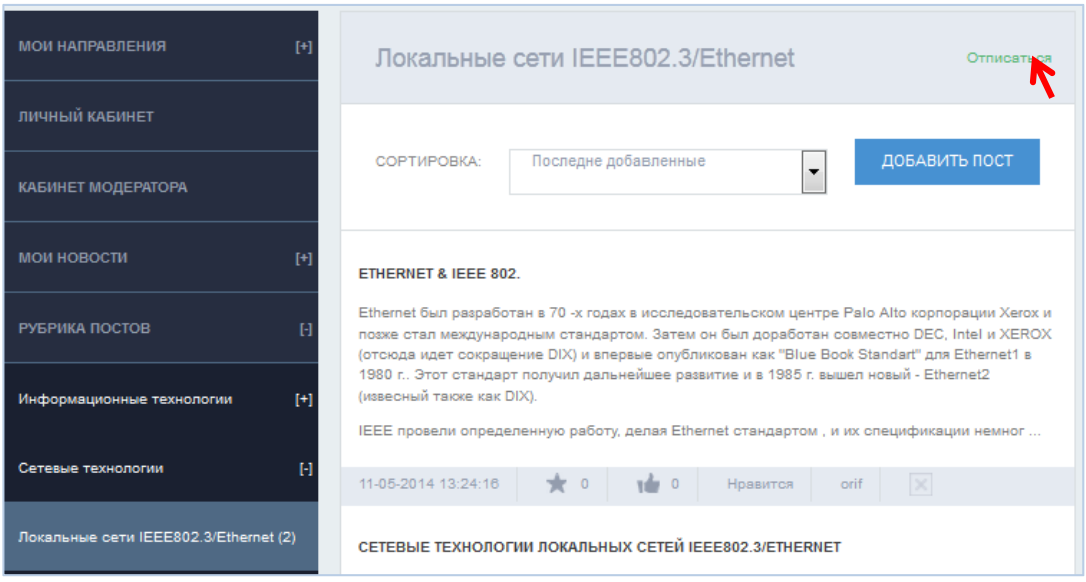


Рис.3.22. Отписка от направления.

Перейдя по ссылке «Подписки» в правом меню, пользователь получает все новые посты, принадлежащих к подписанным направлениям. (см. рис. 3.23.)

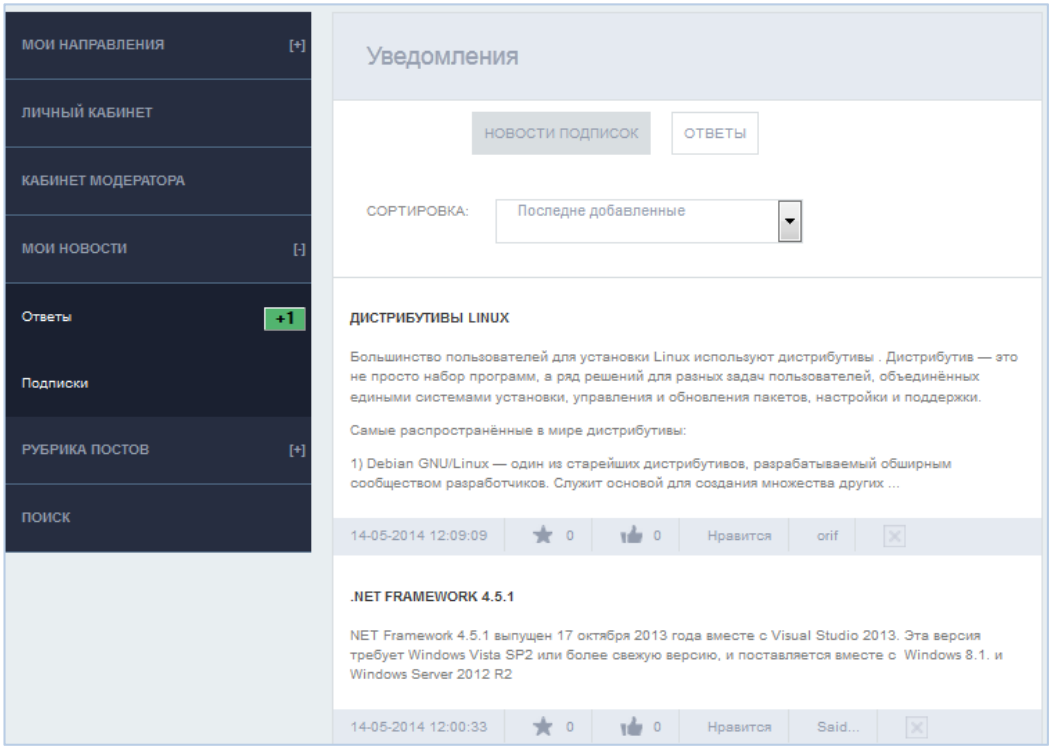


Рис. 3.23. Просмотр новостей подписок.

Функционал модератора

Доступен, только если пользователь является модератором. Служит для отслеживания контента постов, направлений, сфер. Обладает следующими возможностями:

— Редактирование, удаление, добавление чужих и собственных постов;

— Редактирование, удаление, добавление тематик направлений;

На данной странице имеются закладки:

— Посты;

— Направления;

— Сферы;

Переходя по данным закладкам, модератор получает возможность работы либо с постами, либо с направлениями, либо со сферами.

Что бы добавить направление, необходимо кликнуть на кнопку «Добавить направление» и заполнить форму добавления (см. рис. 3.25.).

Редактирование направлений: Клик на ссылку «Редактировать». Если кликнуть «Обновить», то текущая запись будет изменена, если кликнуть «Отмена», то текущая запись останется прежней.

Если кликнуть на «Удалить», запись будет удалена (см. рис. 3.24.).

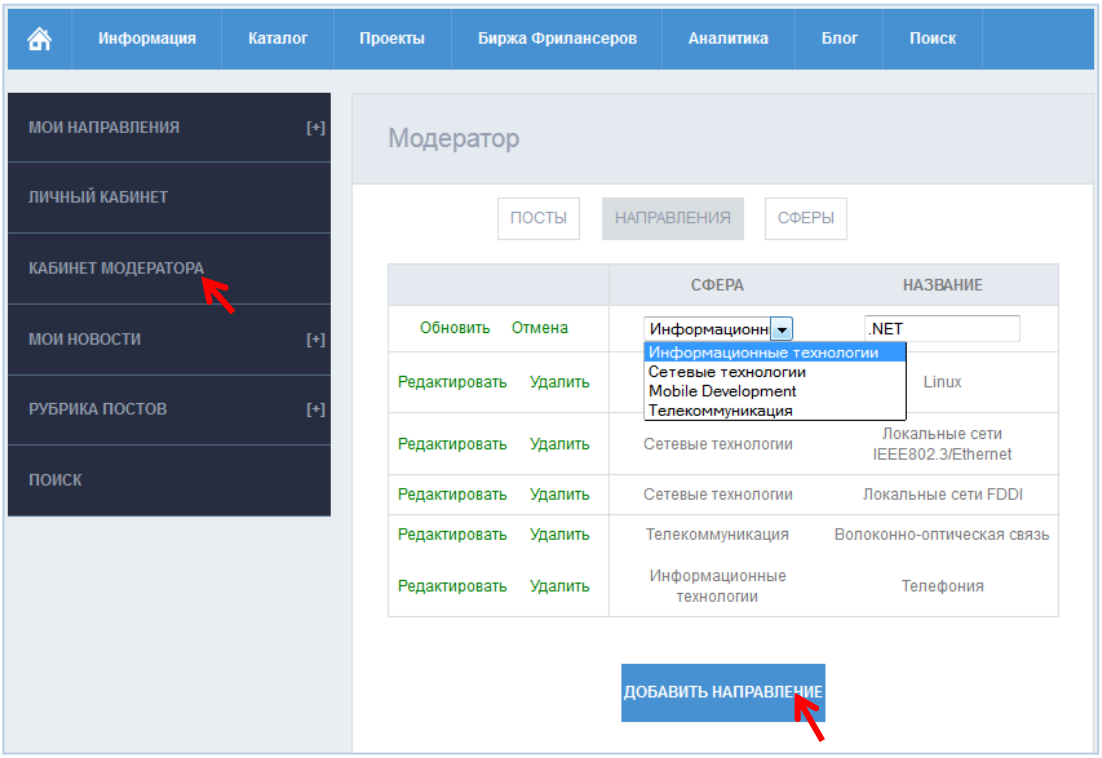


Рис. 3.24. Кабинет модератора.

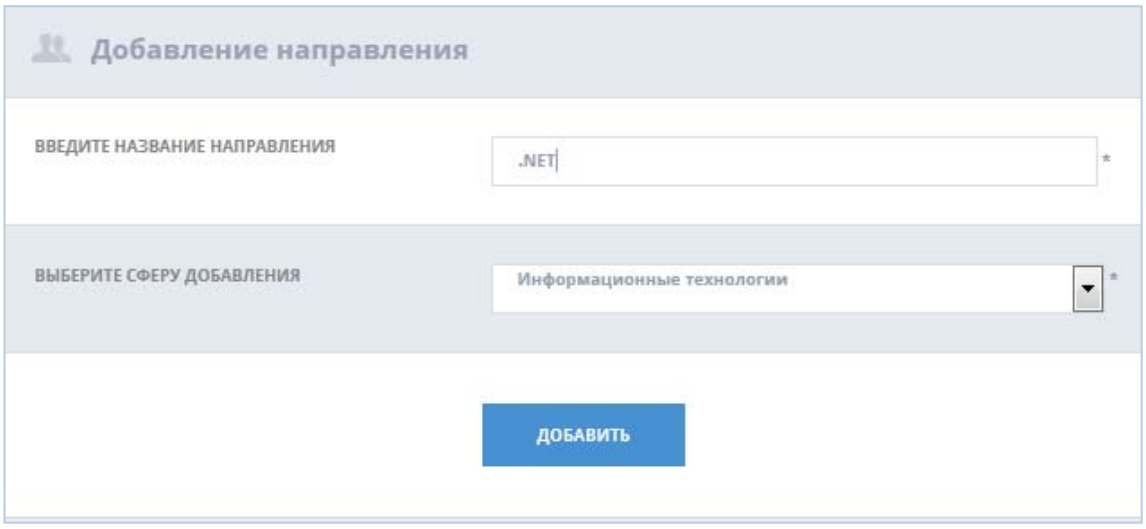


Рис. 3.25. Форма добавления направления.

Аналогичные действия для функционала со сферами.

Если перейти по постам, модератор имеет возможность удаления постов, если пост не соответствует установленным требованиям. (см. рис. 3.26.)

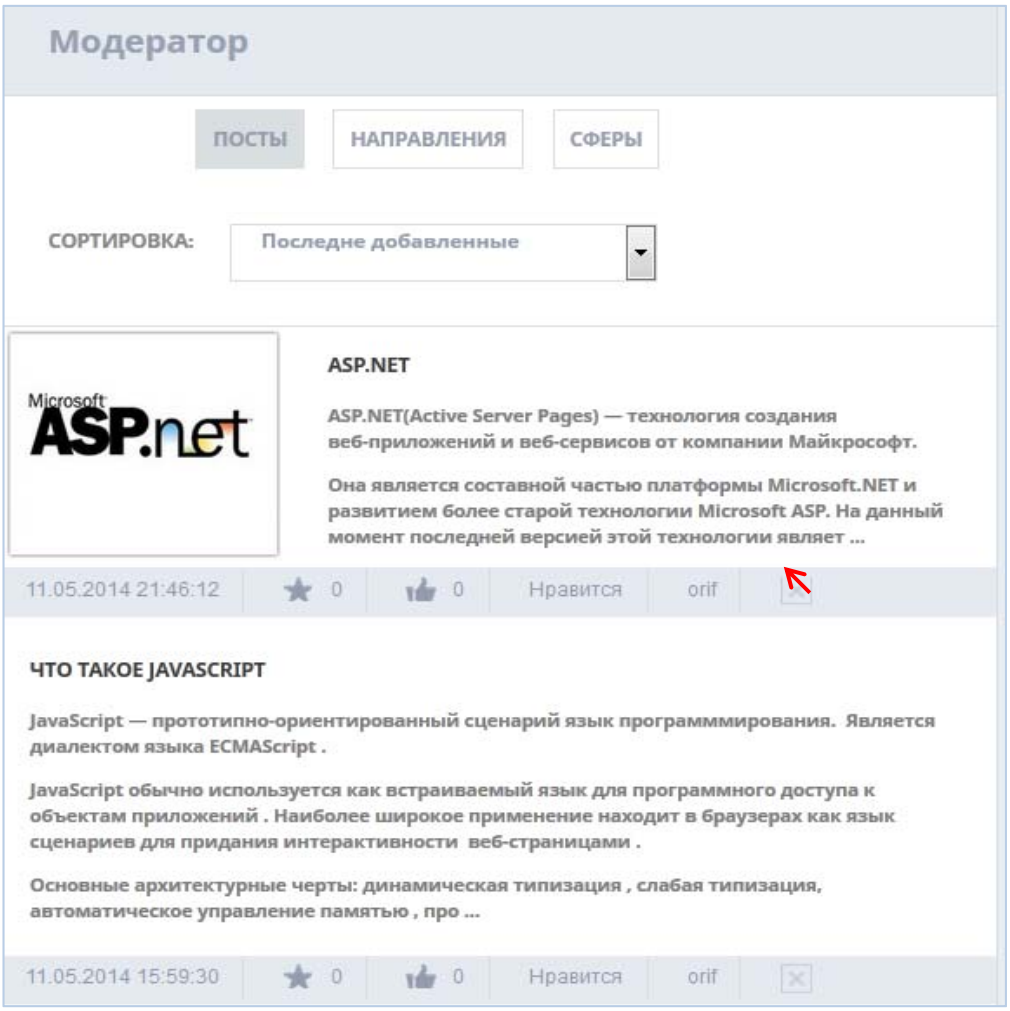


Рис. 3.26. Удаление постов модератором.

Добавление поправок

Фрилансеры могут поправлять посты, которые размещены на досках гостей, если гости неправильно рассуждают в своих постах. Фрилансер переходит в личный профиль гостя, далее выбирает доску, выбирает пост, выбирает фрагмент поста (см. рис. 3.27.), в результате появляется форма добавления поправки с выбранным фрагментом (см. рис. 3.28.). Добавив свою поправку, возвращается на детальный просмотр текущего поста со списком поправок (см. рис. 3.29.).

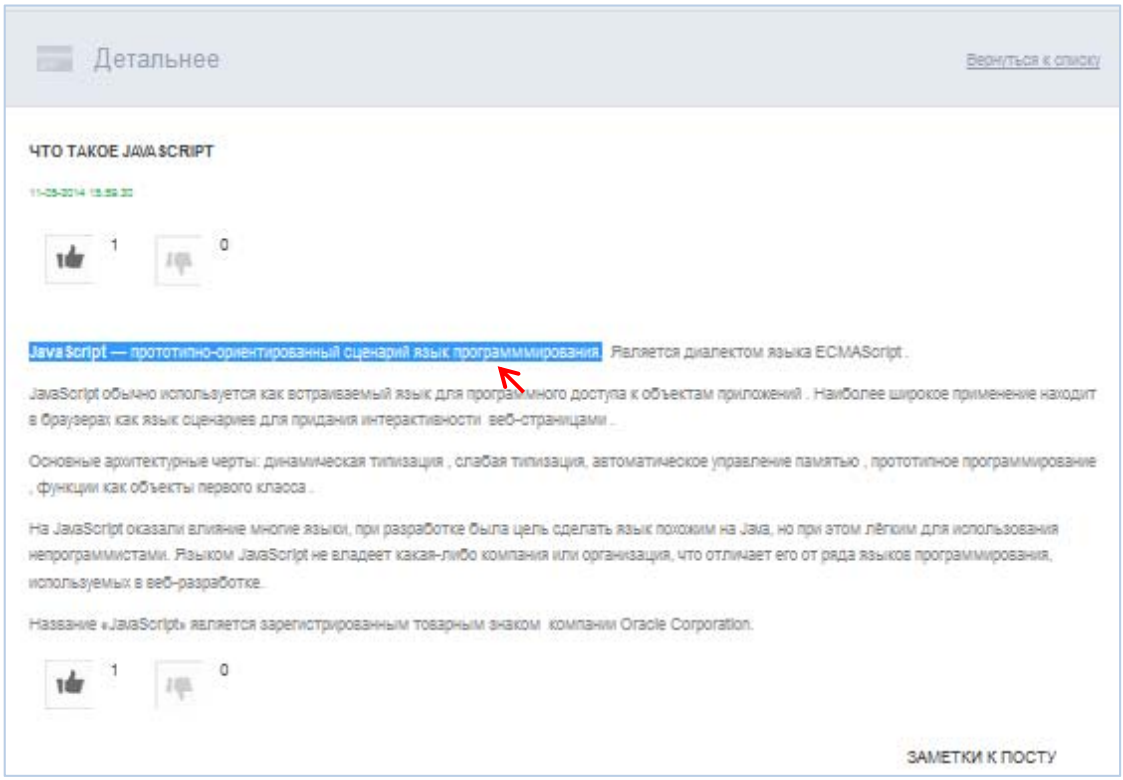


Рис. 3.27. Выбор фрагмента поста.

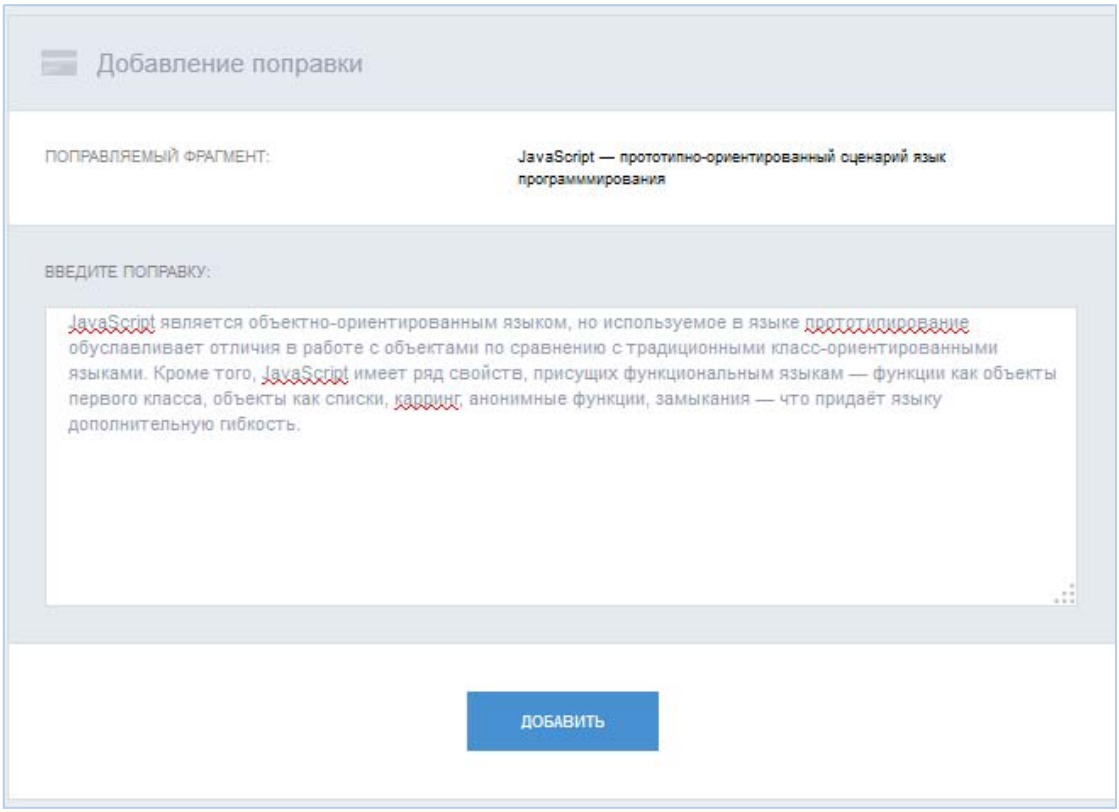


Рис. 3.28. Форма добавления поправки.

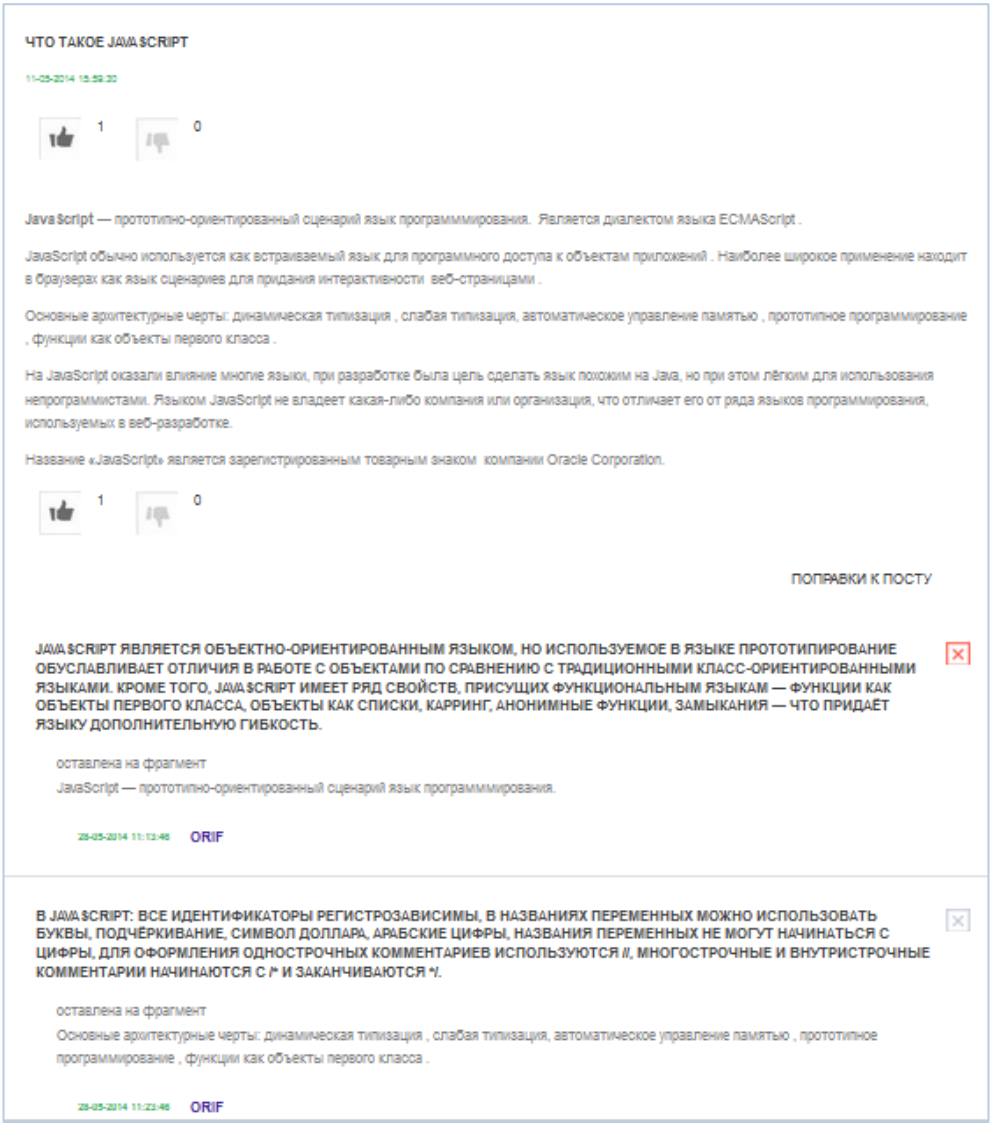


Рис. 3.29. Детальный просмотр текущего поста со списком поправок.

Поиск по блогу

Предоставляется всем пользователям. Результатом поиска являются посты, название которых, удовлетворяют введённому значению в текстовом поле. По умолчанию поиск осуществляется среди всех постов.

Так же предоставляется расширенный поиск по таким критериям как:

—Имя поста;

— Имя автора;

— Сфера;

— Направление;

В зависимости от указанных критериев результатом будут отфильтрованные посты. Результат формируется в виде ссылок на детальный просмотр выбранного поста.

Если отметить по имени автора, результатом будут все посты пользователя, указанного в текстовой поле (см. рис. 3.30.).

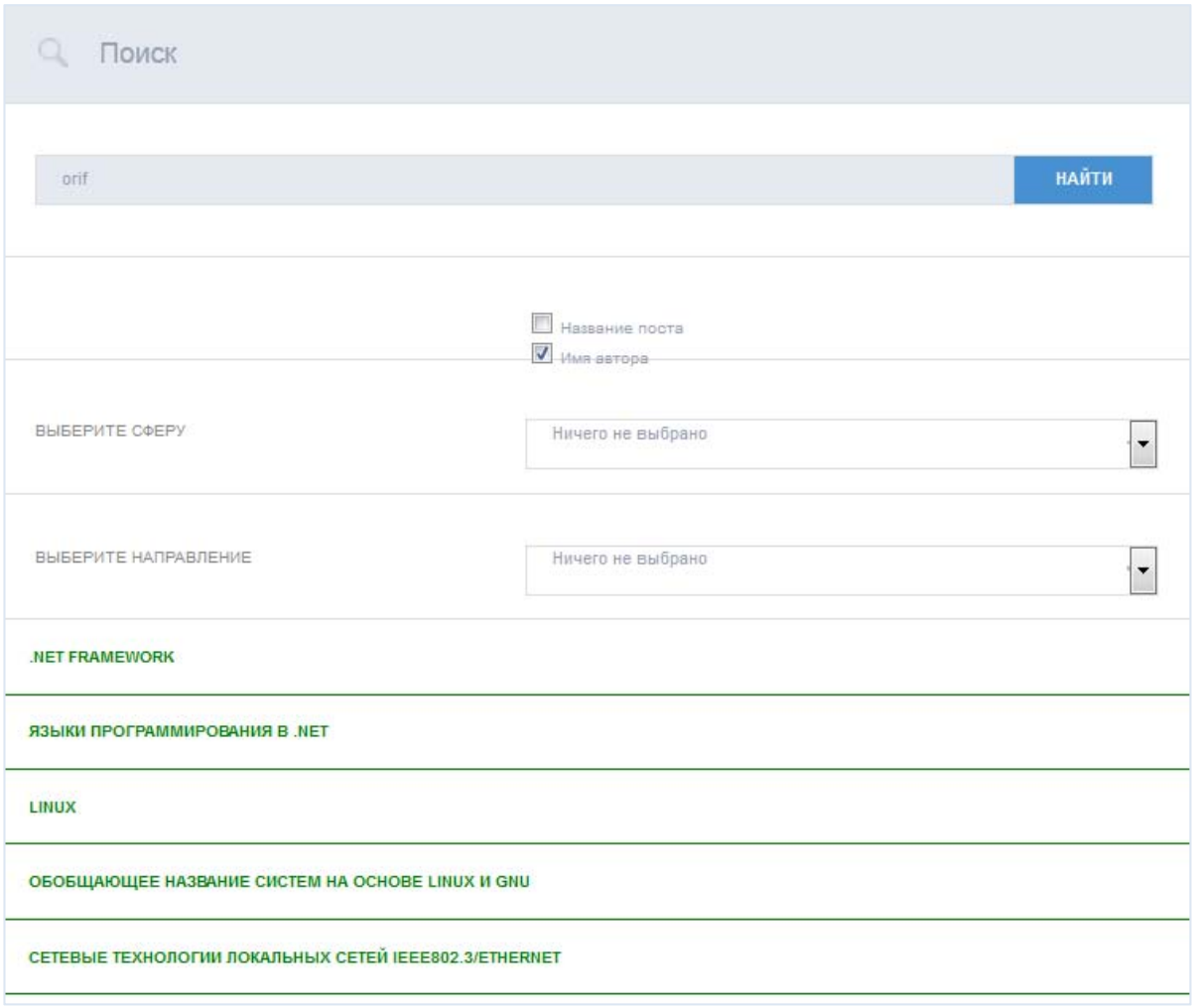


Рис. 3.30. Поиск по автору постов.

Выводы по третьей главе

В третьей главе был произведен выбор средств и анализ технологии ASP.NET для разработки подсистемы. Подсистема была разработана на языке программирования C# с использованием технологии разработки WEB-приложений ASP.NET и СУБД SQL Server 2008 R2. Приведено подробное описание подсистемы по работе блога.

ГЛАВА 4. БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

4.1. Пожарная безопасность

Пожары наносят громадный материальный ущерб и в ряде случаев сопровождаются гибелью людей. Поэтому защита от пожаров является важнейшей обязанностью каждого члена общества и проводится в общегосударственном масштабе.

Противопожарная защита имеет своей целью изыскание наиболее эффективных, экономически целесообразных и технически обоснованных способов и средств предупреждения пожаров и их ликвидации с минимальным ущербом при наиболее рациональном использовании сил и технических средств тушения.

Пожарная безопасность – это состояние объекта, при котором исключается возможность пожара, а в случае его возникновения используются необходимые меры по устранению негативного влияния опасных факторов пожара на людей, сооружения и материальных ценностей. Пожарная безопасность может быть обеспечена мерами пожарной профилактики и активной пожарной защиты. Пожарная профилактика включает комплекс мероприятий, направленных на предупреждение пожара или уменьшение его последствий. Активная пожарная защита – меры, обеспечивающие успешную борьбу с пожарами или взрывоопасной ситуацией.

Совокупность сил и средств, а также мер правового, организационного, экономического, социального и научно-технического характера образуют систему обеспечения пожарной безопасности.

Основными элементами системы обеспечения пожарной безопасности являются органы государственной власти, органы местного самоуправления, предприятия и граждане, принимающие участие в обеспечении пожарной безопасности в соответствии с законодательством Республики Узбекистан.

Причины пожаров на производственных объектах

Производственные объекты отличаются повышенной пожарной опасностью, так как характеризуется сложностью производственных процессов; наличием значительных количеств ЛВЖ и ГЖ, сжиженных горючих газов, твердых сгораемых материалов; большой оснащенностью электрическими установками и другое.

Причины:

1. Нарушение технологического режима – 33%.
2. Неисправность электрооборудования – 16 %.
3. Плохая подготовка к ремонту оборудования – 13%.
4. Самовозгорание промасленной ветоши и других материалов – 10%

Источниками воспламенения могут быть открытый огонь технологических установок, раскаленные или нагретые стенки аппаратов и оборудования, искры электрооборудования, статическое электричество, искры удара и трения деталей машин и оборудования и др.

А также нарушение норм и правил хранения пожароопасных материалов, неосторожное обращение с огнем, использование открытого огня факелов, паяльных ламп, курение в запрещенных местах, невыполнение противопожарных мероприятий по оборудованию пожарного водоснабжение, пожарной сигнализации, обеспечение первичными средствами пожаротушения и др.

Как показывает практика, авария даже одного крупного агрегата, сопровождающаяся пожаром и взрывом, например, в химической промышленности они часто сопутствуют один другому, может привести к весьма тяжким последствиям не только для самого производства и людей его обслуживающих, но и для окружающей среды. В этой связи чрезвычайно важно правильно оценить уже на стадии проектирования пожаро- и взрывоопасность технологического процесса, выявить возможные причины

аварий, определить опасные факторы и научно обосновать выбор способов и средств пожаро- и взрывопредупреждения и защиты.

Немаловажным фактором в проведении этих работ является знание процессов и условий горения и взрыва, свойств веществ и материалов, применяемых в технологическом процессе, способов и средств защиты от пожара и взрыва.

Мероприятия по пожарной профилактике разделяются на организационные, технические, режимные и эксплуатационные.

Организационные мероприятия: предусматривают правильную эксплуатацию машин и внутризаводского транспорта, правильное содержание зданий, территории, противопожарный инструктаж.

Технические мероприятия: соблюдение противопожарных правил и норм при проектировании зданий, при устройстве электропроводов и оборудования, отопления, вентиляции, освещения, правильное размещение оборудования.

Режимные мероприятия – запрещение курения в неустановленных местах, запрещение сварочных и других огневых работ в пожароопасных помещениях и тому подобное.

Эксплуатационные мероприятия – своевременная профилактика, осмотры, ремонты и испытание технологического оборудования.

Пожарная профилактика. Противопожарные преграды.

Для предупреждения распространения пожара с одного здания на другое между ними устраивают противопожарные разрывы. При определении противопожарных разрывов исходят из того, что наибольшую опасность в отношении возможного воспламенения соседних зданий и сооружений представляет тепловое излучение от очага пожара. Количеством принимаемой теплоты соседним с горящим объектом зданием зависит от свойств горючих материалов и температуры пламени, величины излучающей поверхности, площади световых проемов, группы возгораемости

ограждающих конструкций, наличия противопожарных преград, взаимного расположения зданий, метеорологических условий и т.д.

К ним относят стены, перегородки, перекрытия, двери, ворота, люки, тамбур-шлюзы и окна. Противопожарные стены должны быть выполнены из несгораемых материалов, иметь предел огнестойкости не менее 2,5 часов и опираться на фундаменты. Противопожарные стены рассчитывают на устойчивость с учетом возможности одностороннего обрушения перекрытий и других конструкций при пожаре.

Противопожарные двери, окна и ворота в противопожарных стенах должны иметь предел огнестойкости не менее 1,2 часа, а противопожарные перекрытия не менее 1 часа. Такие перекрытия не должны иметь проемов и отверстий, через которые могут проникать продукты горения при пожаре.

Пути эвакуации

При проектировании зданий необходимо предусмотреть безопасную эвакуацию людей на случай возникновения пожара. При возникновении пожара люди должны покинуть здание в течение минимального времени, которое определяется кратчайшим расстоянием от места их нахождения до выхода наружу.

Число эвакуационных выходов из зданий, помещений и с каждого этажа зданий определяется расчетом, но должно составлять не менее двух. Эвакуационные выходы должны располагаться рассредоточено. При этом лифты и другие механические средства транспортирования людей при расчетах не учитывают. Ширина участков путей эвакуации должна быть не менее 1 м, а дверей на путях эвакуации не менее 0.8м. Ширина наружных дверей лестничных клеток должна быть не менее ширины марша лестницы, высота прохода на путях эвакуации – не менее 2 м.

При проектировании зданий и сооружений для эвакуации людей должны предусматриваться следующие виды лестничных клеток и лестниц:

1. незадымляемые лестничные клетки (сообщающиеся с наружной воздушной зоной или оборудованные техническими устройствами для подпора воздуха);
2. закрытые клетки с естественным освещением через окна в наружных стенах; закрытые лестничные клетки без естественного освещения; внутренние открытые лестницы (без ограждающих внутренних стен);
3. наружные открытые лестницы.

Для зданий с перепадами высот следует предусматривать пожарные лестницы.

Способы тушения пожаров

Прекращение горения при пожарах может быть достигнуто путем:

- прекращения поступления в зону горения кислорода воздуха и горючих веществ или снижения их поступления до значений, при которых горение не происходит;
- охлаждения зоны горения ниже температуры самовоспламенения или понижения температуры горючего вещества ниже температуры воспламенения;
- разбавления реагирующих веществ (горючей смеси) негорючими веществами;
- механического срыва пламени в результате воздействия на него сильной струи воды или газа.

4.2. Рациональная организация рабочего места

Научно-технический прогресс внес серьезные изменения в условия производственной деятельности работников умственного труда. Их труд стал более интенсивным, напряженным, требующим значительных затрат умственной, эмоциональной и физической энергии. Это потребовало

комплексного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

В настоящее время компьютерная техника широко применяется во всех областях деятельности человека. При работе с компьютером человек подвергается воздействию ряда опасных и вредных производственных факторов: электромагнитных полей (диапазон радиочастот: ВЧ, УВЧ и СВЧ), инфракрасного и ионизирующего излучений, шума и вибрации, статического электричества и др.

Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой операторов, высокой напряженностью зрительной работы и достаточно большой нагрузкой на мышцы рук при работе с клавиатурой компьютера. Большое значение имеет рациональная конструкция и расположение элементов рабочего места, что важно для поддержания оптимальной рабочей позы человека-оператора.

В процессе работы с компьютером необходимо соблюдать правильный режим труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

Эргономические требования к рабочему месту. Проектирование рабочих мест, снабженных видеотерминалами, относится к числу важных проблем эргономического проектирования в области вычислительной техники.

Рабочее место и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия: оптимальное размещение оборудования,

входящего в состав рабочего места и достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения.

Эргономическими аспектами проектирования видеотерминальных рабочих мест, в частности, являются: высота рабочей поверхности, размеры пространства для ног, требования к расположению документов на рабочем месте (наличие и размеры подставки для документов, возможность различного размещения документов, расстояние от глаз пользователя до экрана, документа, клавиатуры и т.д.), характеристики рабочего кресла, требования к поверхности рабочего стола, регулируемость элементов рабочего места.

Главными элементами рабочего места программиста являются стол и кресло. Основным рабочим положением является положение сидя.

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Моторное поле - пространство рабочего места, в котором могут осуществляться двигательные действия человека.

Максимальная зона досягаемости рук - это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

Оптимальная зона - часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом.

Для комфортной работы стол должен удовлетворять следующим условиям:

- высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;

- нижняя часть стола должна быть сконструирована так, чтобы программист мог удобно сидеть, не был вынужден поджимать ноги;

- поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения программиста;

- конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, листингов, канцелярских принадлежностей).

- высота рабочей поверхности рекомендуется в пределах 680-760мм. Высот поверхности, на которую устанавливается клавиатура, должна быть около 650мм.

Большое значение придается характеристикам рабочего кресла. Так, рекомендуемая высота сиденья над уровнем пола находится в пределах 420-550мм. Поверхность сиденья мягкая, передний край закругленный, а угол наклона спинки - регулируемый.

Необходимо предусматривать при проектировании возможность различного размещения документов: сбоку от видеотерминала, между монитором и клавиатурой и т.п. Кроме того, в случаях, когда видеотерминал имеет низкое качество изображения, например заметны мелькания, расстояние от глаз до экрана делают больше (около 700мм), чем расстояние от глаза до документа (300-450мм). Вообще при высоком качестве изображения на видеотерминале расстояние от глаз пользователя до экрана, документа и клавиатуры может быть равным.

Положение экрана определяется:

- расстоянием считывания (0,6…0,7м);

- углом считывания, направлением взгляда на 20(ниже горизонтали к центру

экрана, причем экран перпендикулярен этому направлению.

Должна также предусматриваться возможность регулирования экрана:

- по высоте +3 см;
- по наклону от -10(до +20(относительно вертикали;
- в левом и правом направлениях.

Большое значение также придается правильной рабочей позе пользователя.При неудобной рабочей позе могут появиться боли в мышцах, суставах и сухожилиях. Требования к рабочей позе пользователя видеотерминала следующие:

- голова не должна быть наклонена более чем на 20 см
- плечи должны быть расслаблены,
- локти - под углом 80 100,
- предплечья и кисти рук - в горизонтальном положении.

Причина неправильной позы пользователей обусловлена следующими факторами: нет хорошей подставки для документов, клавиатура находится слишком высоко, а документы - низко, некуда положить руки и кисти, недостаточно пространство для ног.

В целях преодоления указанных недостатков даются общие рекомендации: лучше передвижная клавиатура; должны быть предусмотрены специальные приспособления для регулирования высоты стола, клавиатуры и экрана, а также подставка для рук .

Существенное значение для производительной и качественной работы на компьютере имеют размеры знаков, плотность их размещения, контраст и соотношение яркостей символов и фона экрана. Если расстояние от глаз оператора до экрана дисплея составляет 60…80 см, то высота знака должна быть не менее 3мм, оптимальное соотношение ширины и высоты знака составляет 3:4, а расстояние между знаками – 15…20% их высоты. Соотношение яркости фона экрана и символов - от 1:2 до 1:15 .

Во время пользования компьютером медики советуют устанавливать монитор на расстоянии 50-60 см от глаз. Специалисты также считают, что верхняя часть видеодисплея должна быть на уровне глаз или чуть ниже.

Когда человек смотрит прямо перед собой, его глаза открываются шире, чем когда он смотрит вниз. За счет этого площадь обзора значительно увеличивается, вызывая обезвоживание глаз. К тому же если экран установлен высоко, а глаза широко открыты, нарушается функция моргания. Это значит, что глаза не закрываются полностью, не омываются слезной жидкостью, не получают достаточного увлажнения, что приводит к их быстрой утомляемости.

Создание благоприятных условий труда и правильное эстетическое оформление рабочих мест на производстве имеет большое значение как для облегчения труда, так и для повышения его привлекательности, положительно влияющей на производительность труда.

Режим труда. Как уже было неоднократно отмечено, при работе с персональным компьютером очень важную роль играет соблюдение правильного режима труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли,раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

Примечание. Время перерывов дано при соблюдении указанных Санитарных правил и норм. При несоответствии фактических условий труда требованиям Санитарных правил и норм время регламентированных перерывов следует увеличить на 30%.

В соответствии с нормами все виды трудовой деятельности,

связанные с использованием компьютера, разделяются на три группы:

группа А: работа по считыванию информации с экрана ВДТ предварительным запросом;

группа Б: работа по вводу информации;

группа В: творческая работа в режиме диалога с компьютером.

Эффективность перерывов повышается при сочетании с производственной гимнастикой или организации специального помещения для отдыха персонала с удобной мягкой мебелью, аквариумом, зеленой зоной и т.п.

Параметры микроклимата. Параметры микроклимата могут меняться в широких пределах, в то время как необходимым условием жизнедеятельности человека является поддержание постоянства температуры тела благодаря терморегуляции, т.е. способности организма регулировать отдачу тепла в окружающую среду. Принцип нормирования микроклимата – создание оптимальных условий для теплообмена тела человека с окружающей средой.

Вычислительная техника является источником существенных тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении. В помещениях, где установлены компьютеры, должны соблюдаться определенные параметры микроклимата.

Объем помещений, в которых размещены работники вычислительных центров, не должен быть меньше 19,5м3/человека с учетом максимального числа одновременно работающих в смену.

Для обеспечения комфортных условий используются как организационные методы (рациональная организация проведения работ в зависимости от времени года и суток, чередование труда и отдыха), так и технические средства (вентиляция, кондиционирование воздуха, отопительная система).

Выводы по четвертой главе

В четвертой главе были рассмотрены характеристики пожарной безопасности, причины возникновения пожаров на производственных объектах, противопожарные преграды, пути эвакуации и способы тушения, рациональная организация рабочего места, влияющая на работоспособность и здоровье человека при выполнении работ за рабочим местом.

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа посвящена разработке подсистемы «Блог» WEB-портала «Software.uz».

В рамках данной работы были получены следующие результаты:

- Проведен анализ требований к разрабатываемой системе;
- Проведено проектирование функциональных требований с использованием языка UML;
- Разработана архитектура объектной модели подсистемы и схема базы данных;
- Проведен анализ технологии ASP.NET для реализации подсистемы.
- Разработана подсистема «Блог»;
- Приведено описание работы блога.

В процессе анализа предметной области была рассмотрена классификация блогов и проведен анализ требований к подсистеме «Блог». На основании этих требований определена новизна разрабатываемой подсистемы – создание условий эффективного обучения новым материалам начинающих пользователей более продвинутыми пользователями.

На этапе проектирования была определены постановка задач и основные функции подсистемы.

С помощью унифицированного языка моделирования (UML) были спроектированы диаграмма использования и диаграмма последовательностей каждого функционала. Была разработана структура базы данных подсистемы.

Разработана объектная модель подсистемы (Приложение 1), базирующаяся на архитектуре классов разделенных на классы, начинающиеся со слова Provider (классы объектно-реляционного представления), которые работают на прямую с БД, и никакой другой класс не имеет право на обращение к БД кроме этих классов. И классы, начинающиеся со слова Data (Бизнес логика) отвечающие за обработку

данных и дополнительную манипуляцию над ними. Основной особенностью разработанной модели является, появление у подсистемы высокой степени расширяемости.

На этапе программной реализации были выбраны средства и технология для реализации подсистемы. Блог реализован на языке программирования C# с использованием технологии разработки WEB-приложений ASP.NET и СУБД MySQL, приведено подробное описание работы блога.

Были рассмотрены характеристики пожарной безопасности, причины возникновения пожаров на производственных объектах, противопожарные преграды, пути эвакуации и способы тушения, рациональная организация рабочего места, влияющая на работоспособность и здоровье человека при выполнении работ за рабочим местом.

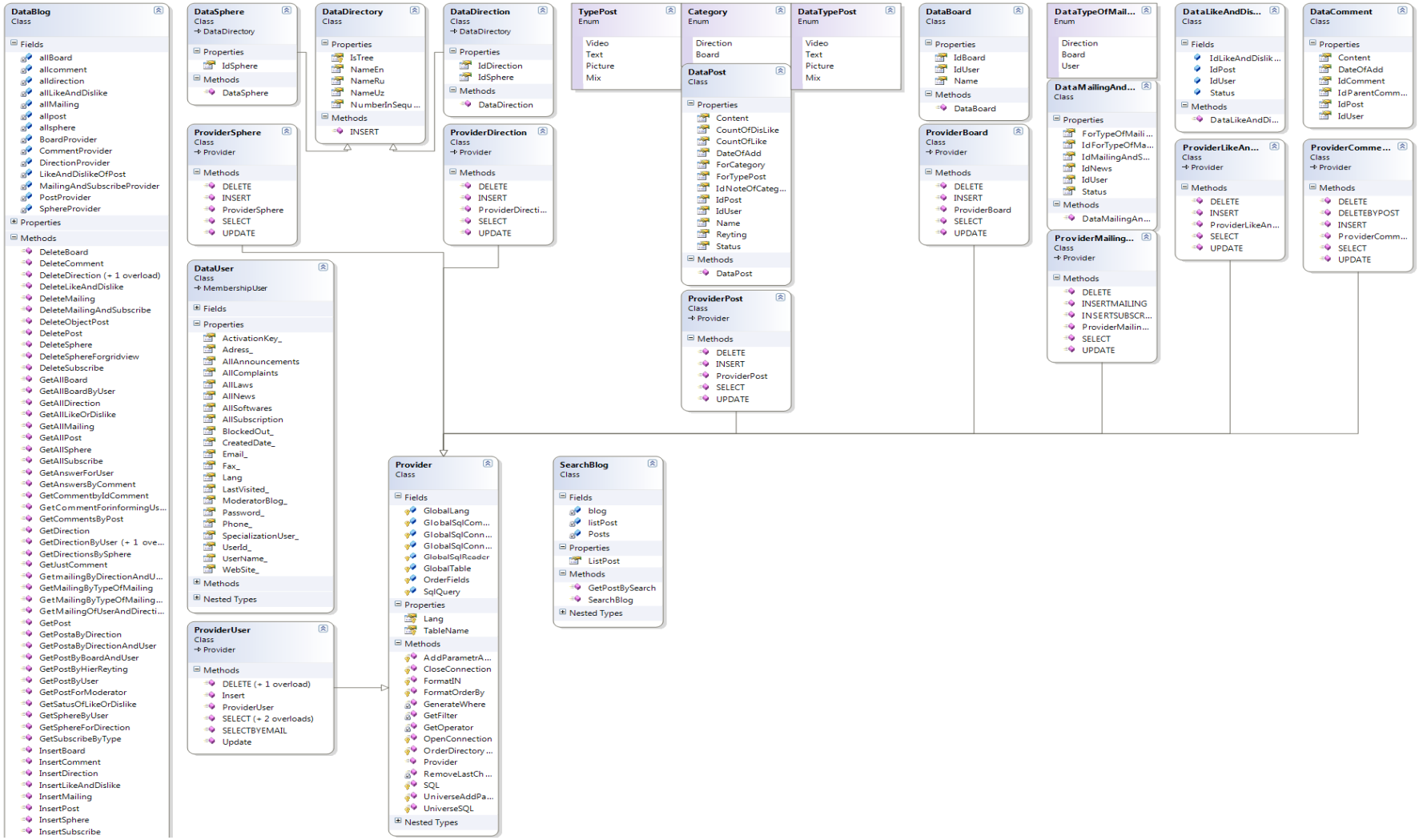
Итогом всей работы является интерактивная подсистема «Блог» WEB – портала «Software.uz».

Разработанная подсистема успешно функционирует в составе WEB – портала «Национальный каталог разработчиков и программных продуктов», который доступен в сети Интернет по адресу http://catalog.sofware.uz.

СПИСОК ЛИТЕРАТУРЫ

1. Постановление Президента Республики Узбекистан от 21.03.2012г. № ПП-1730 «О мерах по дальнейшему внедрению и развитию информационно-коммуникационных технологий» - URL: http://www.lex.uz/Pages/GetAct.aspx?lact_id=1986811
2. Постановление Президента Республики Узбекистан № ПП-2042 от 20.09.13 «О мерах по дальнейшему усилению стимулирования отечественных разработчиков программного обеспечения» – URL: http://lex.uz//pages/getpage.aspx?lact_id=2239892
3. Wikipedia:UML – URL: http://ru.wikipedia.org/wiki/UML
4. Определение сервис-ориентированной архитектуры – URL: http://opengroup.org/projects/soa/doc.tpl?gdid=10632,
5. Райордан Р. «Основы реляционных баз данных» изд. «Русская редакция» 2001г.
6. Мартин Фаулер «Архитектура корпоративных программных приложений» изд. «Вильямс» 2006г.
7. .NET Web Service Description Language Tool – URL: http://msdn2.microsoft.com/en- us/library/7h3ystb6(VS.80).aspx
8. Мэтью Мак-Дональд, Адам Фримен, Марио Шпушта -Microsoft ASP .NET 4 с примерами на C # 2010 для профессионалов.
9. Эндрю Троелсен - Язык программирования C# 2010 и платформа .NET 4 5-е издание.
10. Буч Г., Рамбо Дж. UML. Руководство пользователя. – М.: ДМК Пресс, 2007. – 496 с.
11. Техническое задание на разработку и модернизацию «Национального каталога разработчиков и программных продуктов Software.uz».
12. Абдуллаева С.М. Конспект лекций по курсу БЖД – Ташкент: ТУИТ,2011.

ПРИЛОЖЕНИЕ 1. ОБЪЕКТНАЯ МОДЕЛЬ ПОДСИСТЕМЫ



ПРИЛОЖЕНИЕ 2. ОПИСАНИЕ ТАБЛИЦ БД

1) **SW_Users**, данная таблица содержит пользователей.

Таблица 1.
Описание таблицы SW_Users.

Название поля	**Тип данных**	**Ограничение на поле**	**Описание поля**
IdUser	Int	PK	Уникальный ключ записи в таблице.
UserName	Text		Имя пользователя
Fax	Text		факс
ModeratorBlog	Bit		Является ли пользователь модератором
WebSite	Text		вебсайт
Adress	Text		Адресс
Phone	Text		Номер телефона
Email	Text		Электронная почта
Password	Text		Пароль
CreatedDate	date		Дата создания/рождения
BlockedOut	Bit		Блокирование пользователя
SpecializationUser	Text		Специализация пользователя
AktivationKey	Text		Ключ активации
LastVisited	Date		Последний визит

2) **SW_Post**, данная таблица содержит посты.

Таблица 2.

Описание таблицы SW_Post.

Название поля	**Тип данных**	**Ограничение на поле**	**Описание поля**
Idpost	Int	PK	Уникальный ключ записи в таблице.
Name	Text		Название поста
Content	Text		Содержимое поста
ForCategory	Text		Перечисление категорий(направление,доска), которым относится пост
DateOfAdd	Date		Дата добавления поста
IdUser	Int	FK	Ссылка на пользователя, который является автором

			поста
CountOfLike	Int		Количество лайков
CountOfDislike	Int		Количество дислайков
IdNoteOfCategory	Int		Номер определенной категории поста
Status	Bit		Определение просмотра поста
Reyting	Float		Рейтинг поста

3) **SW_TypeOfSphere**, данная таблица содержит сферы.

Таблица 3.
Описание таблицы SW_TypeOfSphere.

Название поля	**Тип данных**	**Ограничение на поле**	**Описание поля**
IdSphere	Int	PK	Уникальный ключ записи в таблице.
NameUz	Text		Название сферы на узбекском
NameRu	Text		Название сферы по-русски
NameEn	Text		Название сферы по-английски

4) **SW_TypeOfDirection**, данная таблица содержит направления.

Таблица 4.

Описание таблицы SW_TypeOfDirection.

Название поля	**Тип данных**	**Ограничение на поле**	**Описание поля**
IdDirection	Int	PK	Уникальный ключ записи в таблице.
NameUz	Text		Название направления на узбекском
NameRu	Text		Название направления по-русски
NameEn	Text		Название направления по-английски
IdSphere	Int	FK	Ссылка на сферу, которой принадлежит направление

5) **SW_Board**, данная таблица содержит доски.

Таблица 5.
Описание таблицы SW_Board.

Название поля	Тип данных	Ограничение на поле	Описание поля
IdBoard	Int	PK	Уникальный ключ записи в таблице.
Name	Text		Название доски
IdUser	Int	FK	Ссылка на пользователя, которому принадлежит доска

6) **SW_LikeAndDislike**, данная таблица содержит отношение пользователей к постам.

Таблица 6.
Описание таблицы SW_LikeAndDislike.

Название поля	Тип данных	Ограничение на поле	Описание поля
IdLikeAndDislike	Int	PK	Уникальный ключ записи в таблице.
Status	Bit		Определение статуса(да /нет)
IdPost	Int	FK	Номер поста
IdUser	Int	FK	Номер пользователя

7) **SW_MailingAndSubscribe**, данная таблица содержит подписки и уведомления пользователей.

Таблица 7.

Описание таблицы SW_MailingAndSubscribe.

Название поля	Тип данных	Ограничение на поле	Описание поля
IdMailingAndSubscribe	Int	PK	Уникальный ключ записи в таблице.
ForTypeOfMailing	Text		Перечисление подписки(направление, доска, автор)
IdNews	Int		Номер новости
Status	Bit		Прочитано/непрочитано
IdUser	Int	FK	Номер пользователя, кому отправлять новости и уведомления
IdForTypeOfMailing	Int		Номер записи определенного перечисления

8) **SW_Comment,** данная таблица содержит комментарии постов.

Таблица 8.
Описание таблицы SW_Comment.

Название поля	**Тип данных**	**Ограничение на поле**	**Описание поля**
IdComment	Int	PK	Уникальный ключ записи в таблице.
Content	Text		Содержимое
DateofAdd	DateTime		Дата добавления
IdPost	Int	FK	Прокомментированный пост
IdUser	Int	FK	Номер пользователя,кто прокомментировал
IdParentComment	Int	FK	Номер номер родительского комментария

9) **SW_Notes**, данная таблица содержит поправки постов.

Таблица 9.
Описание таблицы SW_Notes.

Название поля	**Тип данных**	**Ограничение на поле**	**Описание поля**
IdNote	Int	PK	Уникальный ключ записи в таблице.
ContentOfNote	Text		Содержимое поправки
ContentOfFragment	Text		Поправляемый фрагмент поста
IdPost	Int	FK	Поправляемый пост
IdUser	Int	FK	Номер пользователя,кто поправил
DateOfAdd	Date		Дата добавления поста

ПРИЛОЖЕНИЕ 3. КОД ПРОГРАММЫ

DataPost(){}- представление объкта в БД.

```
namespace SoftwareCore
{
  public enum TypePost
  {
    Video,
    Text,
    Picture,
    Mix,

  }

  public enum Category
  {
    Direction,
    Board,

  }

      public class DataPost {
            public string Name {get;set;}
    public string Content { get; set; }
    public string ForCategory { get; set; }
    public DateTime DateOfAdd { get; set; }
    public int IdUser { get; set; }
    public string ForTypePost { get; set; }
    public int IdPost { get; set; }
    public int CountOfLike { get; set; }
    public int CountOfDisLike { get; set; }
    public int IdNoteOfCategory { get; set; }
    public float Reyting { get; set; }
    public bool Status { get; set; }
     public DataPost() { }
 }
}
```

ProviderPost(){}-формирование SQL запросов

```
namespace SoftwareCore
{
  public class ProviderPost:Provider
  {

    public ProviderPost()
    {
      this.TableName = "SW_Post";
      //считывать из сессии или из объекта пользователя
      this.Lang = "ru";
    }

    public List<DataPost> SELECT()
    {
      this.SQL(TypeOfRequest.Select);
      this.AddParametrAndRun(TypeOfRequest.Select);
      List<DataPost> ReturnList = new List<DataPost>();
      while (GlobalSqlReader.Read())
      {
```

```
        DataPost Temp = new DataPost ();
        Temp.IdPost = Int32.Parse(GlobalSqlReader["IdPost"].ToString());
        Temp.ForCategory = (GlobalSqlReader["ForCategory"].ToString());
        Temp.DateOfAdd = Convert.ToDateTime
(GlobalSqlReader["DateOfAdd"].ToString());
        Temp.IdUser = Int32.Parse(GlobalSqlReader["IdUser"].ToString());
        Temp.ForTypePost = (GlobalSqlReader["ForTypePost"].ToString());
        Temp.Name = (GlobalSqlReader["Name"].ToString());
        Temp.Content = (GlobalSqlReader["Content"].ToString());
        Temp.CountOfDisLike =
Int32.Parse(GlobalSqlReader["CountOfDisLike"].ToString());
        Temp.CountOfLike = Int32.Parse(GlobalSqlReader["CountOfLike"].ToString());
        Temp.IdNoteOfCategory =
Int32.Parse(GlobalSqlReader["IdNoteOfCategory"].ToString());
        if (!GlobalSqlReader.IsDBNull(11))
          Temp.Reyting = float.Parse(GlobalSqlReader["Reyting"].ToString());
        else Temp.Reyting = 0;
        Temp.Status = bool.Parse(GlobalSqlReader["Status"].ToString());
        ReturnList.Add(Temp);
      }

      CloseConnection();
      return ReturnList;
    }

    public void INSERT(DataPost InsertPost)
    {
      List<string> fields = new List<string>();
      List<object> values = new List<object>();
      string ID = null; string valueID = null;


      fields.Add("Name"); values.Add(InsertPost.Name);
      fields.Add("ForCategory"); values.Add(InsertPost.ForCategory);
      fields.Add("DateOfAdd"); values.Add(InsertPost.DateOfAdd);
      fields.Add("IdUser"); values.Add(InsertPost.IdUser);
      fields.Add("ForTypePost"); values.Add(InsertPost.ForTypePost);
      fields.Add("Content"); values.Add(InsertPost.Content);
      fields.Add("CountOFDisLike"); values.Add(InsertPost.CountOfDisLike);
      fields.Add("CountOfLike"); values.Add(InsertPost.CountOfLike);
      fields.Add("Reyting"); values.Add(InsertPost.Reyting);
      fields.Add("IdNoteOfCategory"); values.Add(InsertPost.IdNoteOfCategory);
      fields.Add("Status"); values.Add(InsertPost.Status);

      this.SQL(TypeOfRequest.Insert, fields, values, ID, valueID);
      this.AddParametrAndRun(TypeOfRequest.Insert, fields, values, ID, valueID);
      CloseConnection();
    }

    public void UPDATE(DataPost UpdatePost)
    {
      List<string> fields = new List<string>();
      List<object> values = new List<object>();
      string ID = "IdPost"; string valueID = UpdatePost.IdPost.ToString();


      fields.Add("Name"); values.Add(UpdatePost.Name);
      fields.Add("ForCategory"); values.Add(UpdatePost.ForCategory);
      fields.Add("DateOfAdd"); values.Add(UpdatePost.DateOfAdd);
      fields.Add("IdUser"); values.Add(UpdatePost.IdUser);
      fields.Add("ForTypePost"); values.Add(UpdatePost.ForTypePost);
```

```
      fields.Add("Content"); values.Add(UpdatePost.Content);
      fields.Add("CountOFDisLike"); values.Add(UpdatePost.CountOfDisLike);
      fields.Add("CountOfLike"); values.Add(UpdatePost.CountOfLike);
      fields.Add("Reyting"); values.Add(UpdatePost.Reyting);
      fields.Add("IdNoteOfCategory"); values.Add(UpdatePost.IdNoteOfCategory);
      fields.Add("Status"); values.Add(UpdatePost.Status);
      this.SQL(TypeOfRequest.Update, fields, values, ID, valueID);
      this.AddParametrAndRun(TypeOfRequest.Update, fields, values, ID, valueID);
      CloseConnection();
    }

    public void DELETE(string IdPost)
    {
      string ID = "IdPost"; string valueID = IdPost;
      this.SQL(TypeOfRequest.Delete, null, null, ID, valueID);
      this.AddParametrAndRun(TypeOfRequest.Delete, null, null, ID, valueID);
      CloseConnection();
    }
  }
}
DataBlog(){}- шлюз класс
public class DataBlog
  {
    #region Получение всех постов
    ProviderPost PostProvider;
    private List<DataPost> allpost;
    public List<DataPost> AllPost
    {
      get
      {
        PostProvider = new ProviderPost();
        this.allpost = PostProvider.SELECT();
        return this.allpost;
      }
    }
    #endregion

    #region Получение всех доск
    ProviderBoard BoardProvider;
    private List<DataBoard> allBoard;
    public List<DataBoard> AllBoard
    {
      get
      {
        BoardProvider = new ProviderBoard();
        this.allBoard = BoardProvider.SELECT();
        return this.allBoard;
      }

    }
    #endregion

    #region Подучение всех комментарий
    ProviderComment CommentProvider = new ProviderComment();
    private List<DataComment> allcomment;
    public List<DataComment> AllComment
    {
      get
      {
        CommentProvider = new ProviderComment();
        this.allcomment = CommentProvider.SELECT();
```

```
    return this.allcomment;
  }
}

#endregion

#region Получение всех Направлений
ProviderDirection DirectionProvider;
private List<DataDirection> alldirection;
public List<DataDirection> AllDirection
{
  get
  {
    DirectionProvider = new ProviderDirection();
    this.alldirection = DirectionProvider.SELECT();
    return this.alldirection;
  }
}
#endregion

#region Получение всех рассылок
ProviderMailingAndSubscribe MailingAndSubscribeProvider;
private List<DataMailingAndSubscribe> allMailing;
public List<DataMailingAndSubscribe> AllMailing
{
  get
  {

    MailingAndSubscribeProvider = new ProviderMailingAndSubscribe();
    this.allMailing = MailingAndSubscribeProvider.SELECT();
    return this.allMailing;
  }
}
#endregion

#region Получение всех сфер
ProviderSphere SphereProvider;
private List<DataSphere> allsphere;
public List<DataSphere> AllSphere
{
  get
  {
    SphereProvider = new ProviderSphere();
    this.allsphere = SphereProvider.SELECT();
    return this.allsphere;
  }
}
#endregion
#region Получение всех лайков
ProviderLikeAndDislikeOfPost LikeAndDislikeOfPost;
private List<DataLikeAndDislikeOfPost> allLikeAndDislike;
public List<DataLikeAndDislikeOfPost> AllLikeAndDislike
 {
  get{
   LikeAndDislikeOfPost=new ProviderLikeAndDislikeOfPost();
   this.allLikeAndDislike = LikeAndDislikeOfPost.SELECT();
   return this.allLikeAndDislike;


   }
```

```
     }
    #endregion
    #region CRUD operation with the Like&Dislike
    //получения записи лайков и дислайков для пределения статуса(LIKE OR DISLIKE)
поста  по айдишнику поста и айдишника пользователя находящегося в системе
    public DataLikeAndDislikeOfPost GetSatusOfLikeOrDislike(int IdPost, int IdUser) {
      List<DataLikeAndDislikeOfPost> LikeAndDislike = new
List<DataLikeAndDislikeOfPost>();
      foreach (DataLikeAndDislikeOfPost row in AllLikeAndDislike) {
        if (row.IdPost == IdPost && row.IdUser == IdUser )
        {
          return row;
        }
      }
      return null;
    }
    //полученгие всех лайков и дислайков для удв\аления по айдишнику поста
    public List<DataLikeAndDislikeOfPost> GetAllLikeOrDislike(int IdPost) {
      List<DataLikeAndDislikeOfPost> LD = new List<DataLikeAndDislikeOfPost>();
      foreach(DataLikeAndDislikeOfPost row in AllLikeAndDislike ){

        if (row.IdPost == IdPost) {

          LD.Add(row);
        }
      }
      return LD;
    }

    //добавление лайка и дислайка
    public void InsertLikeAndDislike(DataLikeAndDislikeOfPost LikeAndDislikeForInsert
) {
      ProviderLikeAndDislikeOfPost LD = new ProviderLikeAndDislikeOfPost();
      LD.INSERT(LikeAndDislikeForInsert);
    }
    //обновление лайка и дислайка
    public void UpdateLikeAndDislike(DataLikeAndDislikeOfPost
LikeAndDislikeforUpdate) {
      ProviderLikeAndDislikeOfPost LD = new ProviderLikeAndDislikeOfPost();
      LD.UPDATE(LikeAndDislikeforUpdate);
    }

    public void DeleteLikeAndDislike(int IdLikeAndDislike)
    {
      ProviderLikeAndDislikeOfPost LD = new ProviderLikeAndDislikeOfPost();
      LD.DELETE(IdLikeAndDislike.ToString());

    }

    #endregion

    #region CRUD операции с постами
    //Выборка всех постов
    public List<DataPost> GetAllPost()
    {
      return AllPost;
    }

    //Выборка определенного поста(правильно)
    public DataPost GetPost(int id)
    {
```

```
  foreach (DataPost row in AllPost)
  {
    if (row.IdPost == id)
    {
      return row;
    }
  }
  return null;
}

//выборка постов для модератора
public List<DataPost> GetPostForModerator() {

  List<DataPost> forreturn = new List<DataPost>();
  foreach (DataPost row in AllPost) {
      forreturn.Add(row);
  }
  return forreturn;
}

//Выборка поста по рейтенгу
public List<DataPost> GetPostByHierReyting() {

  List<DataPost> ForReyting = new List<DataPost>();
  ForReyting = AllPost;
  List<DataPost> ForMaxReyting =new List<DataPost>();


  for(int i=0; i<ForReyting.Count; i++) {

    for (int j = 0; j<ForReyting.Count; j++) {
      if (ForReyting[i].Reyting < ForReyting[j].Reyting)
      {
      DataPost temp = ForReyting[i];
      ForReyting[i] = ForReyting[j];
      ForReyting[j] = temp;

      }
    }
  }
  ForReyting.Reverse();
  return ForReyting;
}


//выборка поста по пользователю()
public List<DataPost> GetPostByUser(int IdUser)
{
  List<DataPost> ForReturn = new List<DataPost>();
  foreach(DataPost row in AllPost ){
    if (row.IdUser == IdUser) {

      ForReturn.Add(row);
    }
  }
  return ForReturn;
}

//Выборка постов по направления(правильно)
public List<DataPost> GetPostaByDirection(int IdDirection)
{
```

```
      List<DataPost> ForReturn = new List<DataPost>();
      foreach (DataPost row in AllPost)
      {
        if (row.IdNoteOfCategory == IdDirection)
        {
          ForReturn.Add(row);
        }

      }
      return ForReturn;
    }


    //Выборка постов по направления и пользователю(правильно)
    public List<DataPost> GetPostaByDirectionAndUser(int IdDirection, int IdUser)
    {
       DataBlog ForGetPost = new DataBlog();
       List<DataPost> ForReturn = new List<DataPost>();
      List<DataPost> Posts = new List<DataPost>();
      Posts=ForGetPost.GetAllPost();

       Posts.Reverse();
       foreach (DataPost rowpost in Posts) {
         if (rowpost.IdUser == IdUser &&
rowpost.ForCategory==Convert.ToString(Category.Direction)) {

           foreach (DataDirection rowdirection in ForGetPost.GetAllDirection()) {

             if (rowpost.IdNoteOfCategory == rowdirection.IdDirection &&
rowdirection.IdDirection == IdDirection && rowpost.IdUser==IdUser )
             {
               ForReturn.Add(rowpost);

             }
           }
         }
       }
       return ForReturn;
    }
    //получение всех постов по пользователю и определеной доске(правильно)
    public List<DataPost> GetPostByBoardAndUser(int IdUser, int IdBoard) {
      DataBlog ForGetPost = new DataBlog();
      List<DataPost> ForReturn = new List<DataPost>();
      List<DataPost> Posts = new List<DataPost>();
      Posts = ForGetPost.GetAllPost();

      Posts.Reverse();
      foreach (DataPost rowpost in Posts)
      {
        if (rowpost.IdUser == IdUser && rowpost.ForCategory ==
Convert.ToString(Category.Board))
        {
          foreach (DataBoard rowboard in ForGetPost.GetAllBoard())
          {
            if (rowpost.IdNoteOfCategory == rowboard.IdBoard && rowboard.IdBoard ==
IdBoard)
            {
              ForReturn.Add(rowpost);
            }
          }
```

```
    }
  }
  return ForReturn;
 }

// Вставка поста(добавление)
public void InsertPost(DataPost PostToInsert)
{
  PostProvider = new ProviderPost();
  PostProvider.INSERT(PostToInsert);
}


public void UpdatePost(DataPost PostToUpdate)
{
  PostProvider = new ProviderPost();
  PostProvider.UPDATE(PostToUpdate);
}

public void DeleteObjectPost(DataPost PostToDelete) {

  PostProvider = new ProviderPost();
  PostProvider.DELETE(PostToDelete.IdPost.ToString());
}

public void DeletePost(string IdPost)
{
  PostProvider = new ProviderPost();
  PostProvider.DELETE(IdPost);
}
#endregion

#region CRUD операции с досками
// Получение всех доск по пользователю(правильно)
public List<DataBoard> GetAllBoardByUser(int IdUser)
{
  List<DataBoard> ForReturn = new List<DataBoard>();
  foreach (DataBoard rowboard in AllBoard)
  {
    if (rowboard.IdUser == IdUser)
      {
        ForReturn.Add(rowboard);
      }

  }
  if (ForReturn.Count!=0)
  {
    return ForReturn;
  }
  else { return null; }
}

//получение всех доск по пользователю взависимости категории(board)

public List<DataBoard> GetDirectionByUser(int IdUser)
{

  List<DataBoard> ForReturn = new List<DataBoard>();
  DataUser user = new DataUser();
  user = user.GetUserByUserId(IdUser);
  foreach (DataPost rowpost in AllPost)
```

```
      {

        if (rowpost.IdUser == user.UserId_ && rowpost.ForCategory ==
Convert.ToString(Category.Direction))
        {
          foreach (DataBoard rowboard in AllBoard)
          {

            if (rowpost.IdNoteOfCategory == rowboard.IdBoard)
            {
              ForReturn.Add(rowboard);

            }
          }
        }
      }
      return ForReturn;
    }

    //Получение всех доск
    public List<DataBoard> GetAllBoard()
    {
      return AllBoard;
    }

    //вставка доски
    public void InsertBoard(DataBoard BoardToInserrt)
    {

      BoardProvider = new ProviderBoard();
      BoardProvider.INSERT(BoardToInserrt);
    }
    //удаление доски
    public void DeleteBoard(string IdBoard)
    {

      BoardProvider = new ProviderBoard();
      BoardProvider.DELETE(IdBoard);

    }

    #endregion

    #region CRUD операции с комментариями

    //получение комментарий определенного поста
    public List<DataComment> GetCommentsByPost( int IdPost )
    {
      List<DataPost> Posts = new List<DataPost>();
      List<DataComment> ForReturn = new List<DataComment>();
      DataBlog Post = new DataBlog();
      Posts = Post.GetAllPost();
      foreach (DataComment rowcomment in AllComment)
      {
          if (rowcomment.IdPost == IdPost)
          {
            ForReturn.Add(rowcomment);
          }
        }
      return ForReturn;
      }
```

```
// получение комментария по айдишнику комментария
public DataComment GetCommentbyIdComment( int IdComment) {

  foreach (DataComment row in AllComment) {
  if(row.IdComment==IdComment ){

    return row;
  }
  }
  return null;
}

//получение просто комментарий по посту
public List<DataComment> GetJustComment(int IdPost)
{

  List<DataComment> ForReturn = new List<DataComment>();

  foreach (DataComment rowcomment in AllComment)
    if (rowcomment.IdParentComment == 0 && rowcomment.IdPost==IdPost)
    {
      ForReturn.Add(rowcomment);
    }
  return ForReturn;
}

/// <summary>
/// получение ответов на определеннвй комментарий
/// </summary>
/// <param name="IdComment"></param>
/// <returns></returns>
public List<DataComment> GetAnswersByComment(int IdComment)
{
  List<DataComment> ForReturn = new List<DataComment>();
  foreach (DataComment rowcommentForJustComment in AllComment)
  {
    if (rowcommentForJustComment.IdParentComment==IdComment)
    {
      ForReturn.Add(rowcommentForJustComment);
    }
  }
  return ForReturn;
}
public void InsertComment(DataComment CommentToInserrt)
{

  CommentProvider.INSERT(CommentToInserrt);
}

public void UpdateComment(DataComment CommentToUpdate)
{

  CommentProvider.UPDATE(CommentToUpdate);
}

public void DeleteComment(string IdComment)
{
  ProviderComment PLD = new ProviderComment();
  PLD.DELETE(IdComment);
}
```

```
    #endregion

    #region CRUD операции с направлениями
    //выборка всех направлений
    public List<DataDirection> GetAllDirection()
    {
      return AllDirection;
    }

    // получение определенного направления

    public DataDirection GetDirection(int IdDirection)
    {

      foreach (DataDirection row in AllDirection)
      {
        if (row.IdDirection == IdDirection)
        {
          return row;
        }
      }
      return null;
    }

    //получение всех направлений по определенной сфере
    public List<DataDirection> GetDirectionsBySphere(int IdSphere)
    {

      List<DataDirection> ForReturn = new List<DataDirection>();
      if (IdSphere == -1)
        return ForReturn;
      else
      {
        foreach (DataDirection rowdirection in AllDirection)
        {
          foreach (DataSphere rowsphere in AllSphere)
          {
            if ((rowdirection.IdSphere == rowsphere.IdSphere) &&
(rowdirection.IdSphere == IdSphere))
            {
              ForReturn.Add(rowdirection);
            }
          }
        }
        return ForReturn;
      }
    }

    //получение всех направлений по пользователю и сфере взависимости
категории(direction)

    public List<DataDirection> GetDirectionByUser(int IdUser, int IdSphere)
    {

      List<DataDirection> ForReturn = new List<DataDirection>();
      DataUser user = new DataUser();
      user = user.GetUserByUserId(IdUser);
      foreach (DataDirection rowdirection in AllDirection)
      {

        if (rowdirection.IdSphere == IdSphere)
```

```
        {

          foreach (DataPost rowpost in AllPost)
          {
            if (rowpost.IdUser == user.UserId_ && rowpost.ForCategory ==
Convert.ToString(Category.Direction))
            {

              if (rowpost.IdNoteOfCategory == rowdirection.IdDirection)
              {

                ForReturn.Add(rowdirection);
              }
            }
          }
        }
      }

      bool ToAdd = true;
      DataDirection temp = new DataDirection();
      List<DataDirection> forReturnNonRepeat = new List<DataDirection>();
      forReturnNonRepeat.Add(ForReturn[0]);
      for (int i = 1; i < ForReturn.Count; i++)
      {
        ToAdd = true;
        for (int j = 0; j < forReturnNonRepeat.Count; j++)
        {
          if (ForReturn[i].IdSphere == forReturnNonRepeat[j].IdSphere)
          {
            ToAdd = false;
            break;
          }
        }
        if (ToAdd)
        {
          forReturnNonRepeat.Add(ForReturn[i]);
        }

      }
      return forReturnNonRepeat;
    }

    //получение комментарий для уведомлении для пользователя

    public List<DataComment> GetCommentForinformingUser( int IdUser )
    {
      List<DataComment> forreturn = new List<DataComment>();
      List<DataMailingAndSubscribe> Mailing = new List<DataMailingAndSubscribe>();
      Mailing =GetAnswerForUser(IdUser);
      DataComment comments = new DataComment();

      foreach (DataMailingAndSubscribe rowinforming in Mailing)
      {
        foreach (DataComment rowcomment in AllComment) {
          if (rowinforming.IdNews == rowcomment.IdComment) {

            forreturn.Add(rowcomment);
          }
        }

      }
```

```
        return forreturn;
    }

    //вставка направления по сфере
    public void InsertDirection(DataDirection directionToInsert)
    {

        ProviderDirection DirectionProvider = new ProviderDirection();
        DirectionProvider.INSERT(directionToInsert);

    }

    public void UpdateDirection(DataDirection DirectioToUpdate)
    {
        ProviderDirection DirectionProvider = new ProviderDirection();
        DirectionProvider.UPDATE(DirectioToUpdate);

    }
    public void DeleteDirection(int IdDirection)
    {
        ProviderDirection DirectionProvider = new ProviderDirection();
        DirectionProvider.DELETE(IdDirection.ToString());
    }
    public void DeleteDirection(DataDirection DirectioToDelete)
    {
        ProviderDirection DirectionProvider = new ProviderDirection();
        DirectionProvider.DELETE(DirectioToDelete.IdDirection.ToString());
    }
    #endregion

    #region CRUD с рассылками&подписками
    //получение всех рассылок
    public List<DataMailingAndSubscribe> GetAllMailing()
    {
        List<DataMailingAndSubscribe> ForReturn = new List<DataMailingAndSubscribe>();
        foreach (DataMailingAndSubscribe rowMailing in AllMailing)
        {
            if (rowMailing.IdNews != 0)
            {
                ForReturn.Add(rowMailing);
            }
        }
        return ForReturn;
    }

    //получение рассылок по типу рассылок для комментариий
    public List<DataMailingAndSubscribe> GetMailingByTypeOfMailing(string
TypeOfMailing)
    {

        List<DataMailingAndSubscribe> ForReturn = new List<DataMailingAndSubscribe>();
        foreach (DataMailingAndSubscribe row in AllMailing)
        {
            if (row.IdNews != 0)
            {
                if (row.ForTypeOfMailing == TypeOfMailing )
                {

                    bool temp = false;
                    foreach(DataComment rowcomment in AllComment){
```

```
            if(row.IdNews==rowcomment.IdComment){

              temp = true;
            //ForReturn.Add(row);
            }

          }
          if (temp == true)
          {

            ForReturn.Add(row);

          }
          else { DataBlog blog=new DataBlog();
          blog.DeleteMailingAndSubscribe(row.IdNews);
          }

        }


      }
    }
    return ForReturn;
  }

  // получение подписок по пользователю и направлению для определения отображения
ссылок(подписаться, отписаться)
  public List<DataMailingAndSubscribe> GetMailingOfUserAndDirection(int IdUser, int
IdDirectionOrBoard,string TYpeOfMailing)
  {

    List<DataMailingAndSubscribe> ForReturn = new List<DataMailingAndSubscribe>();
    foreach (DataMailingAndSubscribe row in AllMailing)
    {
      if (row.ForTypeOfMailing == TYpeOfMailing && row.IdUser == IdUser &&
row.IdForTypeOfMailing == IdDirectionOrBoard)
      {
        ForReturn.Add(row);
      }

    }
    return ForReturn;

  }
  //получение подписок для отправки новости всем пользователям на данное
направление
  public List<DataMailingAndSubscribe> GetMailingByTypeOfMailingWhereIdNewsNull (
string TypeOfMailing, int IdDirection) {

  List<DataMailingAndSubscribe> ForReturn = new List<DataMailingAndSubscribe>();
    foreach (DataMailingAndSubscribe row in AllMailing){
      if(row.ForTypeOfMailing=="Direction" ){
        if (row.IdNews == 0)
        {
          if( row.IdForTypeOfMailing==IdDirection){
          ForReturn.Add(row);
          }
        }
     }
    return ForReturn;
}
```