Министерство образования Московской области

Государственное образовательное учреждение высшего образования Московской области

«Государственный гуманитарно-технологический университет»

Ликино-Дулевский политехнический колледж – филиал ГГТУ

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Дипломный проект

Разработка ООО «ЦА «Максималист» сайта для сетевого сообщества

Выполнил:

Шабалкин Данила Алексеевич\_\_\_\_\_\_\_\_\_\_\_

студент группы ИСП.18.1А\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

по специальности

09.02.07 Информационные системы и программирование

очной формы обучения

Руководитель:

Пронина Алла Юрьевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка:

\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ликино-Дулево

2022 год

**Содержание**

[Введение 3](#_Toc105155577)

[1. Разработка системного проекта 7](#_Toc105155578)

[1.1. Назначение разработки 7](#_Toc105155579)

[1.2. Требования к функциональным характеристикам 7](#_Toc105155580)

[1.2.1. Состав выполняемых функций 7](#_Toc105155581)

[1.3. Требования к надежности и безопасности 8](#_Toc105155582)

[1.4. Требования к составу и параметрам технических средств 8](#_Toc105155583)

[1.5. Требования к информационной и программной совместимости 9](#_Toc105155584)

[2. Разработка технического проекта 10](#_Toc105155585)

[2.1. Обоснование выбора CASE-средств 10](#_Toc105155586)

[2.2. Проектирование модели данных 11](#_Toc105155587)

[2.3. Детальное проектирование программного приложения 15](#_Toc105155588)

[2.4. Функциональная схема программного приложения 18](#_Toc105155589)

[3. Реализация 19](#_Toc105155590)

[3.1. Обоснование выбора средств разработки 19](#_Toc105155591)

[3.2. Руководство программиста 20](#_Toc105155592)

[3.3. Руководство пользователя 29](#_Toc105155593)

[4. Тестирование и отладка 34](#_Toc105155594)

[4.1. Виды тестирования 34](#_Toc105155595)

[4.2. Отладка программного приложения 38](#_Toc105155596)

[5. Расчет базовой себестоимости разрабатываемого продукта 39](#_Toc105155597)

[5.1. Исходные данные 40](#_Toc105155598)

[5.2. Расчеты затрат на выполнение программы 40](#_Toc105155599)

[5.3. Расчет отчислений на социальное страхование и обеспечение 41](#_Toc105155600)

[5.4. Расчет амортизационных отчислений по компьютерной 42](#_Toc105155601)

[Заключение 45](#_Toc105155602)

[Список литературы 49](#_Toc105155603)

[Приложения 51](#_Toc105155604)

# Введение

Веб — система доступа к связанным между собой документам на различных компьютерах, подключённых к Интернету. Веб – часть составных слов, относящихся к «всемирной паутине».

Веб-платформа представляет собой набор технологий, разработанных в качестве открытых стандартов.

Социальная сеть (сокр. соцсеть) — онлайн-платформа, которая используется для общения, знакомств, создания социальных отношений между людьми, которые имеют схожие интересы или офлайн-связи, а также для развлечения (музыка, фильмы) и работы.

Веб-форум — интернет-сервис (платформа) для общения между пользователями интернета (более двух участников) на одну тему или на несколько тем (зависит от специализации форума). Одна из форм социальных сетей.

Суть работы форума заключается в создании пользователями (посетителями форума) своих тем с их последующим обсуждением, путём размещения сообщений внутри этих тем. Отдельно взятая тема представляет собой тематическую гостевую книгу. Пользователи могут комментировать заявленную тему, задавать вопросы по ней и получать ответы, а также сами отвечать на вопросы других пользователей форума и давать им советы. Внутри темы также могут устраиваться опросы (голосования), если это позволяет движок. Вопросы и ответы сохраняются в базе данных форума, и в дальнейшем могут быть полезны как участникам форума, так и любым пользователям сети Интернет, которые могут зайти на форум, зная адрес сайта, или получив его от поисковых систем при поиске информации.

Тематика форумов может быть самой разнообразной, охватывая все сферы жизни, и определяется либо владельцами форума или его администрацией, либо зависит от контингента участников. При этом форумы могут быть как специализированные, например, посвящённый музыке или какому-либо музыкальному жанру, а также узкоспециализированные, посвящённые какой-либо музыкальной группе или исполнителю, так и без специализации, в этом случае пользователи сами выбирают темы интересные им. Форумы могут существовать как автономно, без привязки к какому-либо сайту, так и быть частью веб-порталов.

Процесс создания:

* создание информационного наполнения;
* создание системы пользователей, их регистрации и авторизации;
* система постов и комментариев к ним;
* система репутации;
* система личных сообщений.

Целью курсовой работы является изучение предметной области решения проблем в программировании и разработка веб-форума в соответствии с этапами ЖЦ ПП.

При создании проекта предположительно будет использоваться следующее ПО:

* фронт-энд технологии: HTML, CSS, JavaScript – стандарты в сфере фронт-энд разработки веб-сайтов;
* MS SQL Server Management Studio – СУБД для работы с реляционными базами данных, базируется на SQL Server.
* C# – как основной язык программирования. Это компилируемый язык программирования, разработанный внутри компанией Microsoft.
* Visual Studio – как среда разработки, отладки кода и запуска сервера под отладку и тестирования сайта
* любой веб-браузер для просмотра и отладки сайта

# Разработка системного проекта

## Назначение разработки

Важной частью жизни человека является поиск необходимой для него информации. Во времена до массового распространения интернета, который позволил быстро получать доступ к любой доступной для обывателя информации, люди получали ее с помощью бумажных носителей, радио, телевидения, передавали ее друг другу словесно. Эти способы сегодня никуда не делись. Но после появления интернета, который предоставил людям более быстрый и удобный способ поиска информации, ими стали реже пользоваться.

Форум Сетевого сообщестава предназначен для поиска решений проблем при разработке ПО. Если у пользователя возникает проблема во время решения задачи, он может обратиться к другим пользователям форума за ответом. После того, как пользователь обратился с вопросом, его пост отображается в так называемом «фиде», который видит каждый пользователь форума, и на который может перейти. После перехода на страницу вопроса, пользователь видит полностью вопрос, с возможными прикрепленными файлами и фрагментами кода. У пользователя также есть возможность отвечать на вопросы в таком же формате, с прикреплением файлов и фрагментов кода. Автор вопроса может отмечать ответы как решение проблемы, после чего вопрос закрывается и добавлять к нему ответы становится невозможно. Вопросы можно дополнять или удалять. Также на форуме присутствует система тегов, которая помогает систематизировать все вопросы. На форуме присутствует поиск и фильтрация, которая помогает найти уже существующие вопросы, ответы на которые уже были даны и отмечены как решение. Поиск осуществляется набором запроса в строке поиска и подбором возможных тегов. После загрузки поиска вопросы можно отфильтровать, для чего появляются поля со списками. Для того, чтобы задавать вопросы и отвечать на них пользователю необходимо войти в аккаунт, или, если он уже существует войти в него.

## Требования к функциональным характеристикам

## Состав выполняемых функций

Форум должен обеспечивать следующие функции:

* Обеспечение входа и регистрации пользователей;
* Опубликование вопросов пользователей;
* Возможность ответа на эти вопросы
* Редактирование вопросов и ответов
* Удаление вопросов и ответов
* Показ вопросов в ленте на главной
* Предоставлять права администратора соответствующей группе пользователей (изменять, удалять вопросы и ответы всех пользователей, назначать других людей администраторами)
* Отображать информацию о пользователе на его странице, и вопросы, которые он задал на форуме
* Поиск по названию и тексту вопроса
* Фильтрация вопросов по одной из категории

## Требования к надежности и безопасности

* Сайт должен предусматривать базовую защиту от основных видов атак: межсайтового скриптинга (XSS), SQL-инъекций, CSRF-уязвимостей.
* Рекомендуется хранить одну копию программного обеспечения на внешнем носителе. Также рекомендуется регулярно копировать файл базы данных на внешний носитель, если такую услугу не предоставляет хостинговая компания.
* Защита паролей путем хранения в базе не самого пароля, а его хеша.
* Шифрование файлов cookie, благодаря чему их невозможно будет подделать
* Размещение сайта на домене с SSL-сертификатом
* Обеспечение ведения журнала сайта и мониторинга его активности и событий безопасности
* Предоставление прав администратора ограниченной группе лиц

## Требования к составу и параметрам технических средств

* Со стороны сервера:

|  |  |
| --- | --- |
| Процессор | Многопоточный (от 12 и более) процессор с тактовой частотой не менее 2.5гГц |
| ОЗУ | 8Гб DDR4 |
| Накопители | Минимум 8гб SSD |
| Соединение | 1гБит Соединение |

* Со стороны клиента:

|  |  |
| --- | --- |
| Соединение | Широкополосное подключение к сети |

## Требования к информационной и программной совместимости

* Сервер должен содержать в составе ПО операционную систему Windows Server
* Клиентский ПК должен содержать браузер с поддержкой актуальных веб-технологий.
* База данных – Microsoft SQL Server 2019.
* Утилита для подключения и управления БД - SQL Server Management Studio 18.

# Разработка технического проекта

## Обоснование выбора CASE-средств

Одним из основных этапов жизненного цикла программного продукта является этап проектирования предметной области с точки зрения заказчика, непосредственного участника бизнес-процессов. Цель: выявить участников, определить их взаимосвязь, автоматизировать процессы, наладить электронный документооборот в организации.

Use Case (вариант использования, ВИ, Прецедент, юскейс) — это сценарная техника описания взаимодействия. С помощью Use Case может быть описано и пользовательское требование, и требование к взаимодействию систем, и описание взаимодействия людей и компаний в реальной жизни. Проектирование предметной области с использованием Case-средства Draw.io и Case-технологии UML.

В разработке ПО эту технику часто применяют для проектирования и описания взаимодействия пользователя и системы, поэтому название Use Case часто воспринимает как синоним требования человека-пользователя к решению определенной задачи в системе.

Описание требований к системе не в виде отдельных функций, а в виде описания контекста и последовательности действий пользователя помогает сформировать набор функциональных требований, который будет обеспечивать полноту и не избыточность требований.

Draw.io - это сервис, предназначенный для формирования диаграмм и схем. Сервис разделён на три части — меню, панель объектов и сам документ.

С помощью веб-сервиса Draw.io можно создавать:

* Диаграммы.
* UML-модели.
* Графики.
* Блок-схемы.
* Формы.

## Проектирование модели данных

Диаграмма прецедентов - диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Актёр («Эктор») — стилизованный человечек, обозначающий набор ролей пользователя, взаимодействующего с некоторой сущностью (системой, подсистемой, классом). Актёры не могут быть связаны друг с другом (за исключением отношений наследования).

Диаграмма прецедентов необходимо для:

* чёткого отделения системы от её окружения;
* определения действующих лиц (актёров), их взаимодействия с системой и ожидаемый функционал системы;
* определения в глоссарии предметной области понятия, относящиеся к детальному описанию функционала системы.

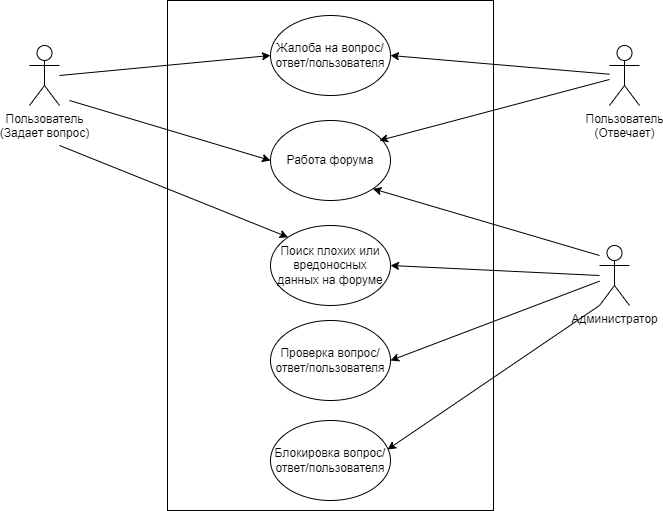


Рис. 1 «Диаграмма прецедентов»

Для выделения бизнес-процессов необходимо сначала выделить действия, совершаемые компанией. В данном случае все зарегистрированные пользователи форума могут посещать его с целью просмотра уже существующих вопросов, а при регистрации задавать уже новые вопросы, и также отвечать на существующие. Порядок действий таков:

* + 1. Пользователь заходит на сайт, авторизуется и оставляет вопрос по своей проблеме;
    2. Затем этот пользователь ожидает ответов на его вопрос
    3. Другие пользователи просматривают вопрос, и, если у них есть возможные варианты решения, они их описывают.

1. «Бизнес-процессы»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2-Раб\_форума | Операция | Исполнитель | Как часто | Входящие документы | Исходящий документ |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 2-Раб\_Форума1 | 1. Задание вопроса | Пользователь (задающий вопрос) | По мере необходимости | Текст вопроса | Нет |
| 2-Раб\_форума2 | 2. Ожидание ответа на вопрос | Пользователь (задающий вопрос) | По мере необходимости | Нет | Нет |
| 2-Раб\_Форума3 | 3. Ответ на вопрос | Пользователь (отвечающий на вопрос) | По мере необходимости | Текст вопроса | Текст ответа |
| 2-Раб\_Форума4 | 4. Пометка ответа как решения | Пользователь (задающий вопрос) | По мере необходимости | Текст ответа | Решение вопроса |

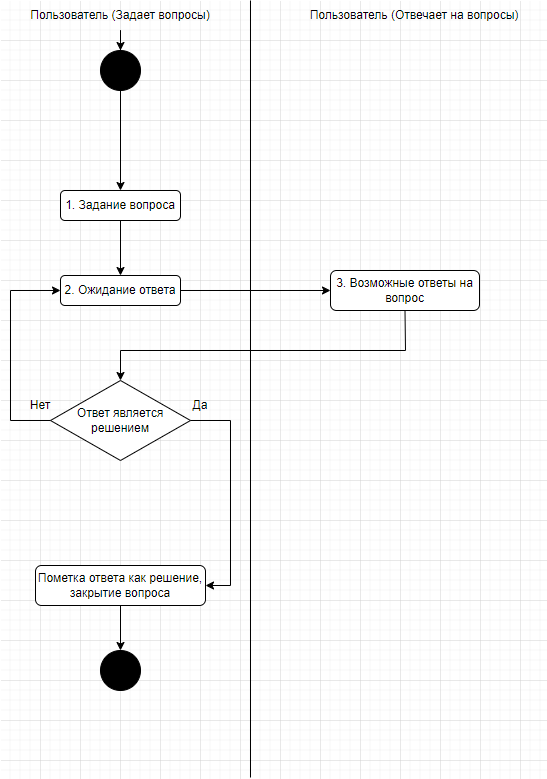


Рис. 2 «Диаграмма действий»

## Детальное проектирование программного приложения

Интерфейс — это вся видимая пользователю часть сервиса, с которой он взаимодействует, решая свои задачи.

Далее представлены макеты пользовательского интерфейса на основе которого реализуется интерфейс в программном приложении.



Рис. 3 «примерный макет главной страницы»

Проектирование — процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части. Результатом проектирования является проект — целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для реализации системы.

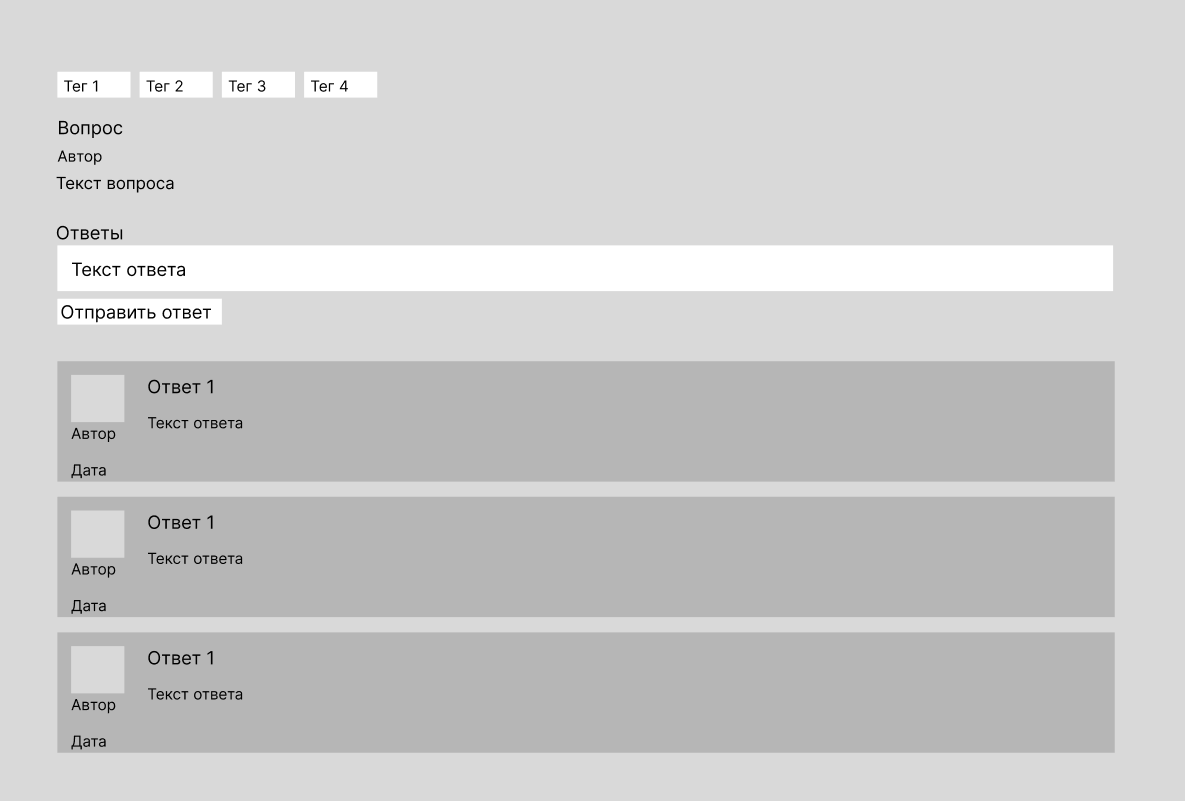


Рис. 4 «Примерный макет страницы вопроса»

Проектирование включает в себя:

* Разработку документации, удовлетворяющего принятым в данной области критериям проекто-сообразности;
* Научно-техническое и социокультурное обоснование, оценку и согласование проекта среди заинтересованных сторон (официальных или общественных организаций, потребительских групп или отдельных лиц), заканчивающееся решением о принятии проекта;
* Реализацию его доступными для изготовителя технологическими, организационными, инвестиционными и прочими средствами.



Рис. 5 «Примерный макет формы добавления вопроса»

В проектировании интерфейсов если следующие принципы:

1. Опираться на опыт пользователей

Каким бы красивым или впечатляющим не был дизайн интерфейса, он должен быть удобен в использовании и быть основан на пользовательском опыте.

1. Проектируемый интерфейс должен быть удобен всем

Необходимо брать во внимание не только среднестатистического пользователя, но также учитывать возраст, инвалидность, культурные различия пользователей.

1. Не нужно заставлять запоминать много информации

Необходимая информация для работы с приложением должна быть на виду и легко-находимой, предлагайте помощь прямо в контексте

1. Пользователю необходимо дать ощущение контроля над ситуацией

Пользователи – это люди, а людям свойственно совершать ошибки, поэтому на этот случай необходимо предусмотреть «Аварийный выход». Позвольте пользователю отменить своё действие, кнопка отмены дает ощущение уверенности и избавляет от страха ошибиться.

## Функциональная схема программного приложения

Блок-схема — это диаграмма, на которой представлен процесс, система или компьютерный алгоритм и которая используется для документирования, планирования, уточнения или визуализации многоэтапного рабочего процесса. В блок-схемах в том виде, в каком они известны сегодня, используют символы различной формы для обозначения специфических элементов рабочих процессов, а стрелки и линии указывают на направление перехода от этапа к этапу.

Как правило, блок-схемы используются в следующих целях:

* Документирование, описание и стандартизация процессов для создания общего видения
* Упрощение и визуализация сложных систем и идей
* Организация рабочих групп, определение избыточных элементов и сотрудничество для усовершенствования процессов
* Мгновенное отслеживание прогресса

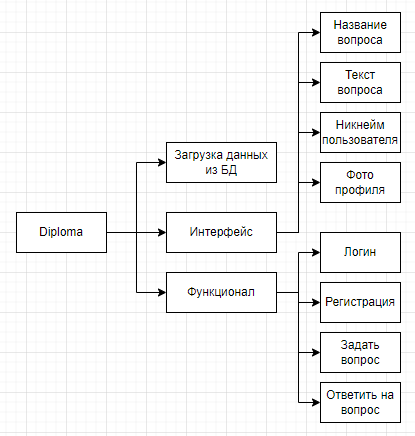
****

Рис. 6 «Блок-схема сайта»

# Реализация

## Обоснование выбора средств разработки

Интегрированная среда разработки (IDE) — это многофункциональная программа, которая поддерживает многие аспекты разработки программного обеспечения. Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки. Данная среда идеально подходит для разработки оконных приложений, она содержит в себе средства разработки приложений под Windows, Android, приложения Azure, веб-приложения, Office, хорошо подходит для разработки игр и работ с базами данных. Имеет в своем арсенале технологии для разработки, отладки, тестирования программных продуктов, дает возможность установить множество всевозможных расширений и предоставляет гибкую систему управления версиями.

Для разработки и проектирования баз данных была использована программа SQL Server Management Studio (SSMS) - это интегрированная среда для управления любой инфраструктурой SQL. Используйте SSMS для доступа, настройки, администрирования и разработки всех компонентов SQL Server, Базы данных SQL Azure, Управляемого экземпляра SQL Azure, SQL Server на виртуальной машине Azure и Azure Synapse Analytics. Среда SSMS предоставляет единую комплексную служебную программу, которая сочетает в себе обширную группу графических инструментов с рядом многофункциональных редакторов скриптов для доступа к SQL Server для разработчиков и администраторов баз данных всех профессиональных уровней.

Для верстки и дальнейшей работы с интерфейсом были выбраны стандартные средства веб-разработки: HTML, CSS, JavaScript. Эти средства давно зарекомендовали себя в сфере web и являются стандартными для абсолютно всех сайтов, так как все браузеры поддерживают их.

## Руководство программиста

Для работы с данными необходимо настроить подключение к базе данных. Для этого была создана модель с таблицами базы данных, строка подключения и контекст, благодаря которому можно обратиться напрямую к базе в коде:

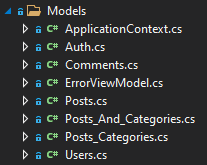


Рис. 7 «Модель данных в программном проекте»

Пример кода модели данных:

using System;

namespace kurs2.Models

{

public class Users

{

public int id { get; set; }

public string Name { get; set; }

public DateTime RegDate { get; set; }

public DateTime BirthDate { get; set; }

public string Bio { get; set; }

public string Photo { get; set; }

public string Password { get; set; }

public string Email { get; set; }

public int CatID { get; set; }

}

}

Создание контекста базы данных:

using Microsoft.EntityFrameworkCore;

using kurs2.Models;

namespace kurs2.Models

{

public class ApplicationContext : DbContext

{

public DbSet<Users> Users { get; set; }

public DbSet<Posts> Posts { get; set; }

public DbSet<Comments> Comments { get; set; }

public ApplicationContext(DbContextOptions<ApplicationContext> options)

: base(options)

{

Database.EnsureCreated();

}

}

}

Метод, инициализирующий контекст при старте сервера:

public void ConfigureServices(IServiceCollection services)

{

string connection = Configuration.GetConnectionString("DefaultConnection");

// добавляем контекст MobileContext в качестве сервиса в приложение

services.AddDbContext<ApplicationContext>(options =>

options.UseSqlServer(connection));

services.AddControllersWithViews();

}

Строка подключения в конфигурации приложения:

"ConnectionStrings": {

"DefaultConnection": "Server=.\\SQLEXPRESS;Database=KursForum;Trusted\_Connection=True;"

},

Для приложения спроектирована и создана база данных, в которой хранятся все динамические данные, выводящиеся в последствии на страницу пользователю:

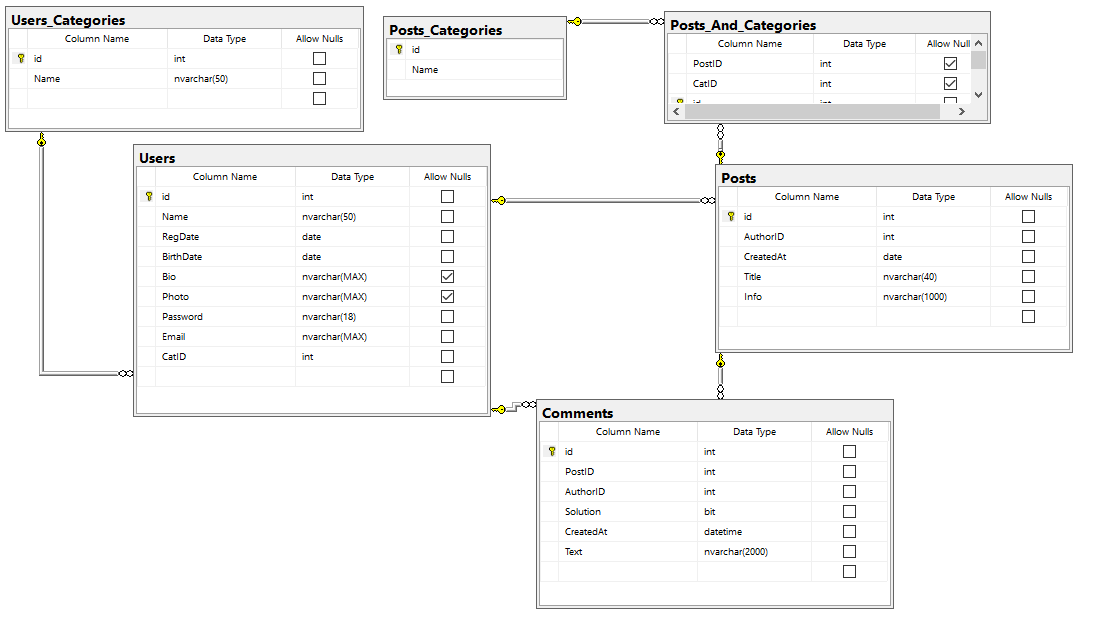


Рис. 8 «Схема данных в СУБД»

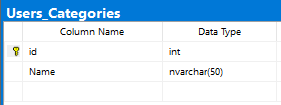


Рис. 9 «Таблица Users\_Categories»

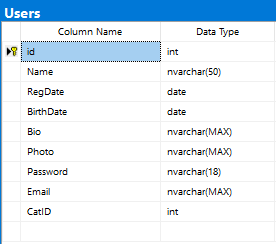


Рис. 10 «Таблица Users»

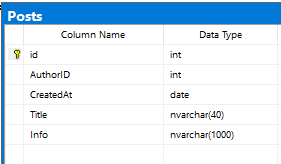


Рис. 11 «Таблица Posts»

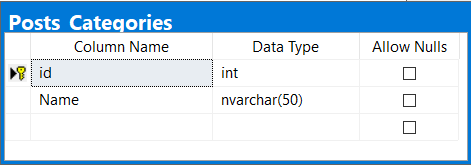


Рис. 12 «Таблица Posts\_Categories»

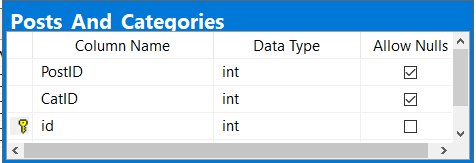


Рис. 13 «Таблица Posts\_And\_Categories»

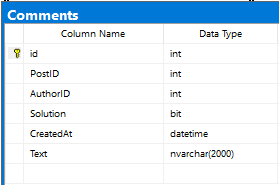


Рис. 14 «Таблица Comments»

После того, как пользователь нажмет на «Задать вопрос», на форуме появится соответствующая запись.

**Файловая структура**:

Проект сайта выполнен по паттерну программирования MVC. Этот паттерн предполагает собой, что backend структура проекта будет содержать в себе следующие директории: models, views, controllers:

* + - директория models содержит в себе модели из базы данных, а также всю необходимую логику проекта (методы, статические классы и т.д.);
    - директория views содержит в себе представления проекта, то есть файлы, содержащие в себе интерфейс страниц;
    - директория controllers содержит в себе классы для определения роутов страниц, а также их логики. Также, в них определяются данные, которые отправятся на представления.

Так как проект создавался по шаблону ASP.NET CORE MVC, также в корневой папке проекта содержится директория wwwroot, где хранятся статические файлы (css-таблицы стилей, js-скрипты, файлы типа картинок)

Также в корневой папке проекта хранятся классы Program.cs и Startup.cs. Они отвечают за запуск, корректную работу сайта, подключаемые к нему сервисы, подключения к базам данных, объявление роутов страниц.

**Описание классов:**

Класс HomeController.cs содержит методы для определения логики определенных страниц, и отвечает за отправку данных на эти страницы.

using kurs2.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Dynamic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace kurs2.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public static ApplicationContext db;

public HomeController(ApplicationContext context, ILogger<HomeController> logger)

{

\_logger = logger;

db = context;

}

public IActionResult Index()

{

dynamic mymodel = new ExpandoObject();

mymodel.Users = db.Users.ToList();

mymodel.Categories = db.Posts\_Categories.ToList();

mymodel.Posts = db.Posts.ToList();

return View(mymodel);

}

[Route("SearchPosts")]

public IActionResult SearchPosts(string q)

{

if (q != "")

{

Response.Redirect($"Searchq?q={q}");

}

else

{

Response.Redirect("/");

}

return View();

}

[Route("Searchq")]

public IActionResult Searchq(string q)

{

dynamic model = new ExpandoObject();

ViewBag.q = q;

model.Posts = db.Posts;

return View(model);

}

[Route("Index")]

public IActionResult Search(string q)

{

if (q != "")

{

Response.Redirect($"search?q={q}");

}

return View(q);

}

public IActionResult Privacy()

{

return View();

}

[Route("Post")]

public ActionResult Post(int id, int? usrId)

{

ViewBag.ID = id;

ViewBag.Controller = "Post";

ViewBag.Action = "Index";

dynamic postsmodel = new ExpandoObject();

postsmodel.Users = db.Users.ToList();

postsmodel.Comments = db.Comments.ToList();

postsmodel.Posts = db.Posts.ToList();

postsmodel.Categories = db.Posts\_Categories.ToList();

postsmodel.PostsCategories = db.Posts\_And\_Categories.ToList();

return View(postsmodel);

}

[Route("User")]

public ActionResult User(int id)

{

ViewBag.ID = id;

dynamic userModel = new ExpandoObject();

userModel.Users = db.Users.ToList();

userModel.Posts = db.Posts.ToList();

return View(userModel);

}

[Route("Login")]

public ActionResult Login(Users user)

{

var usr = db.Users.FirstOrDefault(x => x.Name == user.Name && x.Password == user.Password);

if (usr != null)

{

Response.Cookies.Append("auth", usr.id.ToString(), new Microsoft.AspNetCore.Http.CookieOptions { Expires = DateTimeOffset.Now.AddMonths(1) });

Response.Cookies.Append("pass", usr.Password.ToString(), new Microsoft.AspNetCore.Http.CookieOptions { Expires = DateTimeOffset.Now.AddMonths(1) });

Response.Cookies.Append("name", usr.Name.ToString(), new Microsoft.AspNetCore.Http.CookieOptions { Expires = DateTimeOffset.Now.AddMonths(1) });

Response.Cookies.Append("r", usr.CatID.ToString(), new Microsoft.AspNetCore.Http.CookieOptions { Expires = DateTimeOffset.Now.AddMonths(1) });

Response.Redirect("/");

}

else

{

Response.Cookies.Append("Test", "Texxt");

}

return View();

}

[Route("Exit")]

public ActionResult Exit()

{

Response.Cookies.Delete("auth");

Response.Cookies.Delete("pass");

Response.Cookies.Delete("r");

Response.Cookies.Delete("name");

Response.Redirect("/");

return View();

}

[Route("AddPost")]

public ActionResult AddPost(string title, string info, int authorID, DateTime creadetAt, string strId, int[] cats)

{

Posts post = new Posts();

post.AuthorID = authorID;

post.Info = info;

post.Title = title;

post.CreatedAt = creadetAt;

dynamic model = new ExpandoObject();

model.Posts = db.Posts.ToList();

model.Categories = db.Posts\_Categories.ToList();

strId = Request.Cookies["auth"];

int id = Convert.ToInt32(strId);

ViewBag.ID = id;

if (title != "" && info != null)

{

db.Posts.Add(post);

db.SaveChanges();

}

if (db.Posts.FirstOrDefault(x => x.Title == post.Title) != null)

{

foreach (var i in cats)

{

Diplom.Models.Posts\_And\_Categories pc = new Diplom.Models.Posts\_And\_Categories();

pc.CatID = i;

pc.PostID = post.id;

db.Posts\_And\_Categories.Add(pc);

db.SaveChanges();

}

Response.Redirect($"post?id={post.id}");

}

return View(model);

}

[Route("Registration")]

public ActionResult Registration(Users usr)

{

if (usr.Name != "" && usr.Bio != null && usr.Password != null)

{

db.Users.Add(usr);

db.SaveChanges();

Response.Redirect("/");

}

return View();

}

[Route("AddCommentary")]

public ActionResult AddCommentary(Comments cm, int id)

{

ViewBag.id = id;

if (cm.Text != null)

{

db.Comments.Add(cm);

db.SaveChanges();

Response.Redirect($"post?id={cm.PostID}");

}

return View();

}

public void AddCommentary\_(string text, int id)

{

if (text == null)

{

Response.Redirect($"/post?id={id}");

}

else

{

Comments comm = new Comments();

comm.Text = text;

comm.PostID = id;

comm.AuthorID = Convert.ToInt32(Request.Cookies["auth"]);

comm.CreatedAt = DateTime.Now;

db.Comments.Add(comm);

db.SaveChanges();

}

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

[Route("ChangePost")]

public ActionResult ChangePost(int id, Posts post)

{

ViewBag.id = id;

ViewBag.post = db.Posts.Find(id);

if (post.Title != null && post.Info != null)

{

db.Posts.Load();

db.Posts.Find(id).Title = post.Title;

db.Posts.Find(id).Info = post.Info;

db.SaveChanges();

Response.Redirect($"/post?id={id}");

}

return View();

}

[Route("ChangeComm")]

public ActionResult ChangeComm(int id, Comments comm)

{

ViewBag.id = id;

ViewBag.comm = db.Comments.Find(id);

if (comm.Text != null)

{

db.Posts.Load();

db.Comments.Find(id).Text = comm.Text;

db.SaveChanges();

Response.Redirect($"/post?id={db.Comments.Find(id).PostID}");

}

return View();

}

[Route("DeletePost")]

public ActionResult DeletePost(int id)

{

ViewBag.id = id;

foreach(var a in db.Posts\_And\_Categories)

{

if(a.PostID == id)

{

db.Posts\_And\_Categories.Remove(a);

}

}

db.SaveChanges();

foreach (var a in db.Comments)

{

if (a.PostID == id)

{

db.Comments.Remove(a);

}

}

db.SaveChanges();

var post = db.Posts.Find(id);

db.Posts.Remove(post);

db.SaveChanges();

Response.Redirect("/");

return View();

}

[Route("DeleteComm")]

public ActionResult DeleteComm(int id)

{

ViewBag.id = id;

var comm = db.Comments.Find(id);

db.Comments.Remove(comm);

db.SaveChanges();

Response.Redirect($"Post?id={comm.PostID}");

return View();

}

[Route("Category")]

public ActionResult Category(int id)

{

dynamic model = new ExpandoObject();

model.Categories = db.Posts\_Categories.ToList();

model.PostsCategories = db.Posts\_And\_Categories.ToList();

model.Posts = db.Posts.ToList();

ViewBag.id = id;

return View(model);

}

[Route("AddCategory")]

public IActionResult AddCategory(string name)

{

var cat = new Diplom.Models.Posts\_Categories();

cat.Name = name;

db.Posts\_Categories.Add(cat);

db.SaveChanges();

Response.Redirect("/");

return View();

}

}

}

**Описание функций и методов:**

public IActionResult SearchPosts(string q) - принимает в качестве аргумента данные из поисковой строки, передает их на страницу поиска

[Route("SearchPosts")]

public IActionResult SearchPosts(string q)

{

if (q != "")

{

Response.Redirect($"Searchq?q={q}");

}

else

{

Response.Redirect("/");

}

return View();

}

public IActionResult Searchq(string q) – принимает в качестве параметра данные из предыдущего метода и отправляет на страницу поиска.

[Route("Searchq")]

public IActionResult Searchq(string q)

{

dynamic model = new ExpandoObject();

ViewBag.q = q;

model.Posts = db.Posts;

return View(model);

}

## Руководство пользователя

При переходе на сайт пользователь попадает на главную страницу:

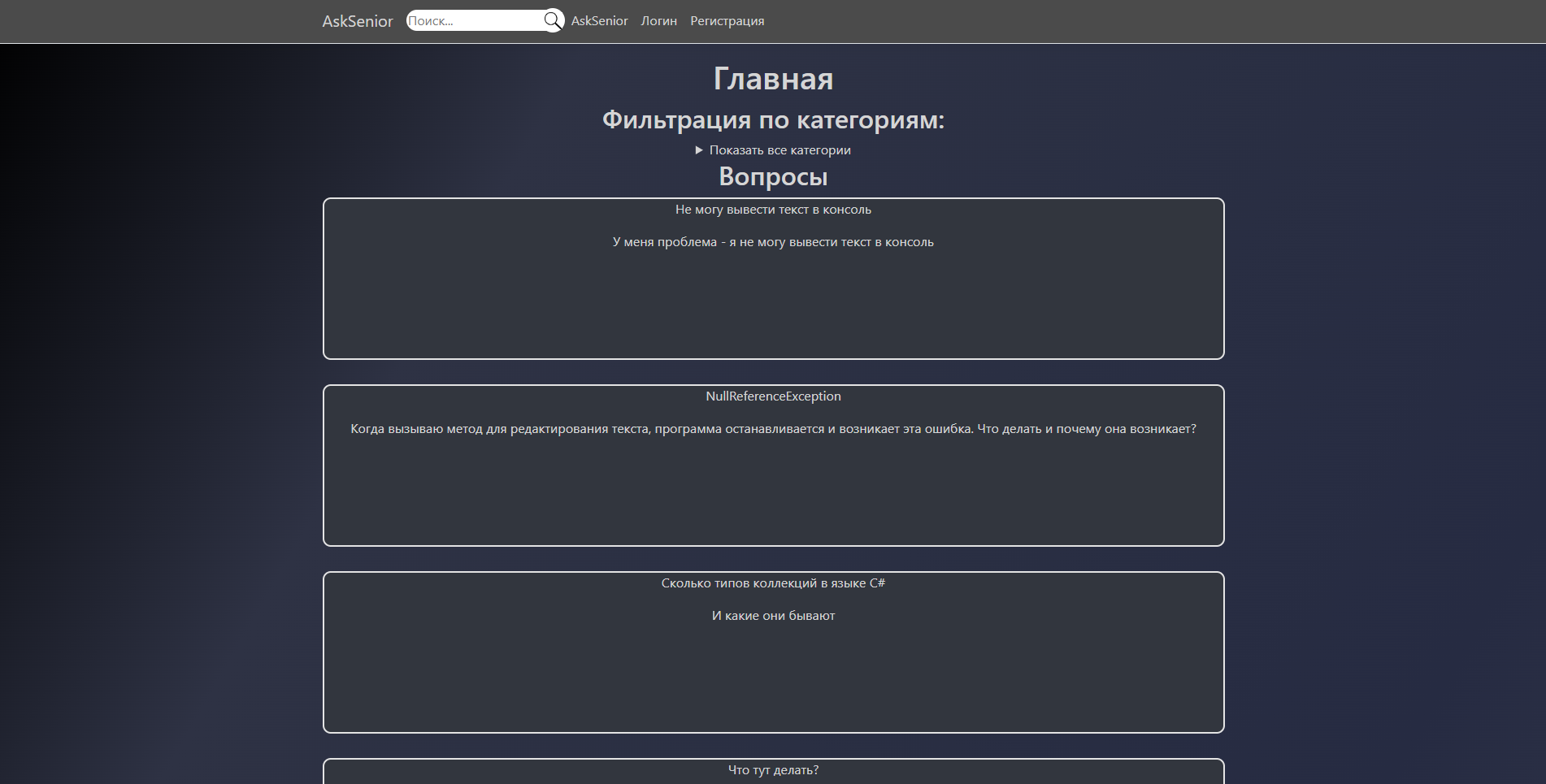


Рис. 15 «Главная страница сайта»

На ней находятся:

* + - Категории для фильтрации вопросов по ним
    - Сами вопросы

Также можно увидеть шапку сайта, которая отображается на каждой странице. В ней находятся:

* + - Ссылка на главную страницу сайта
    - поиск
    - ссылки на авторизацию и регистрацию

Для начала работы на сайте рекомендуется зарегистрироваться или войти.

Для этого нужно перейти на соответствующую страницу:

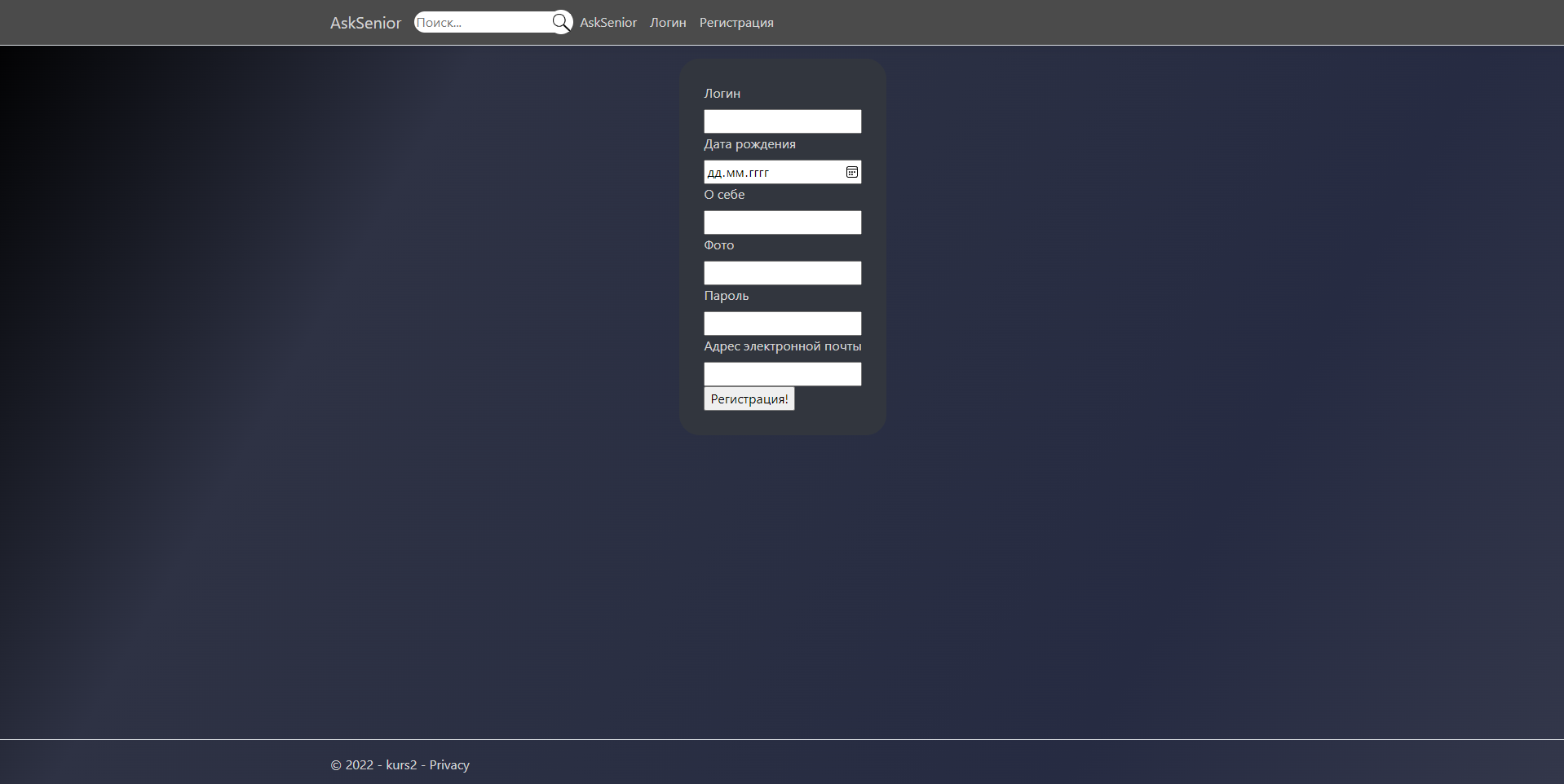


Рис. 16 «Страница регистрации»

После заполнения всех данных, пользователь будет зарегистрирован, после чего нужно будет войти:

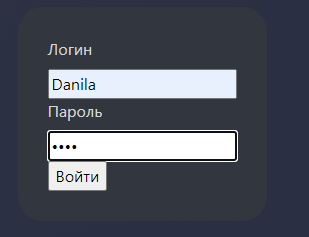


Рис. 17 «Страница авторизации»

После этого на главной странице появится ссылка на страницу добавления вопроса:

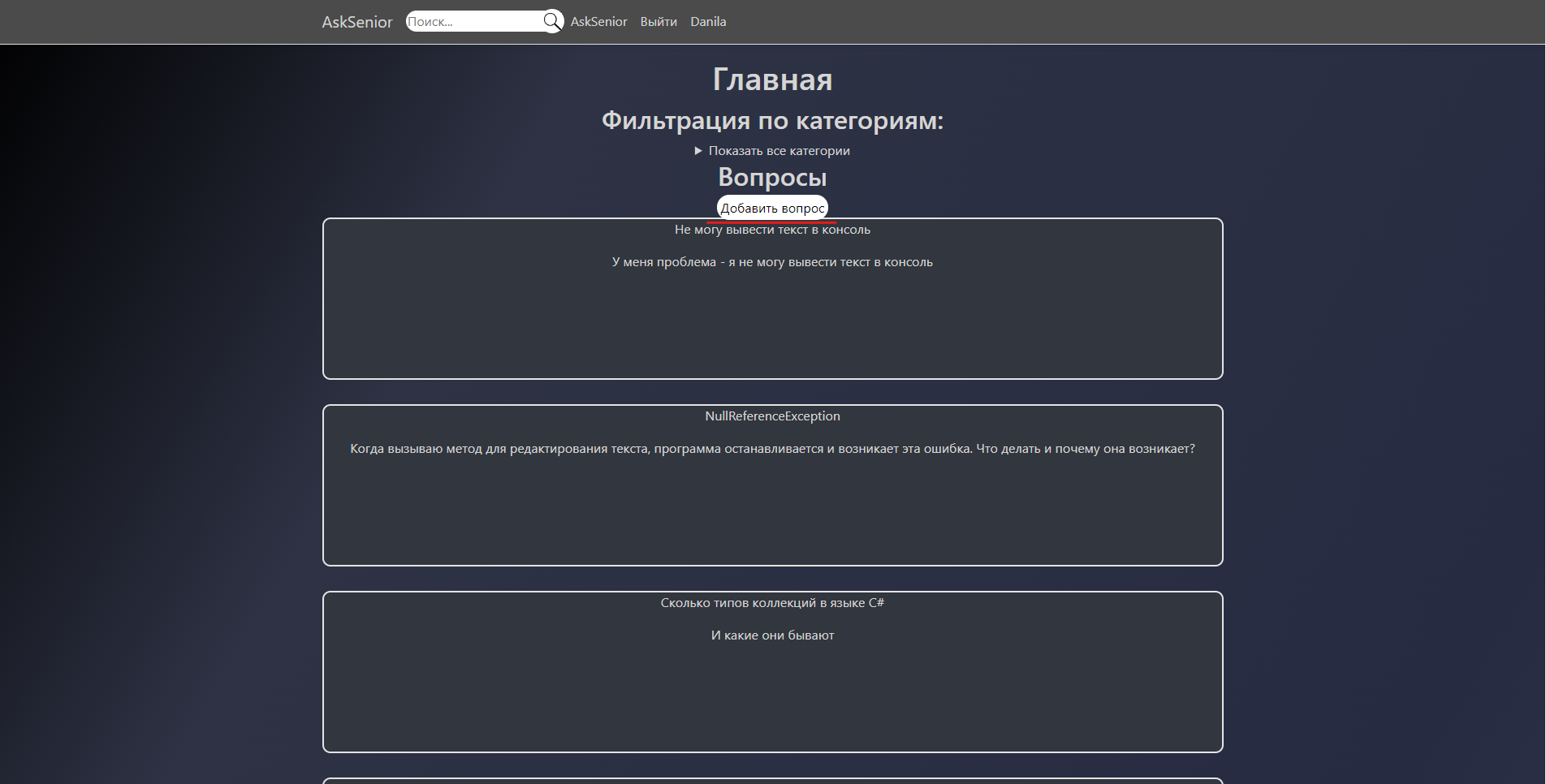


Рис. 18 «Ссылка добавления вопроса»

Если перейти по этой ссылке, то откроется следующая страница:

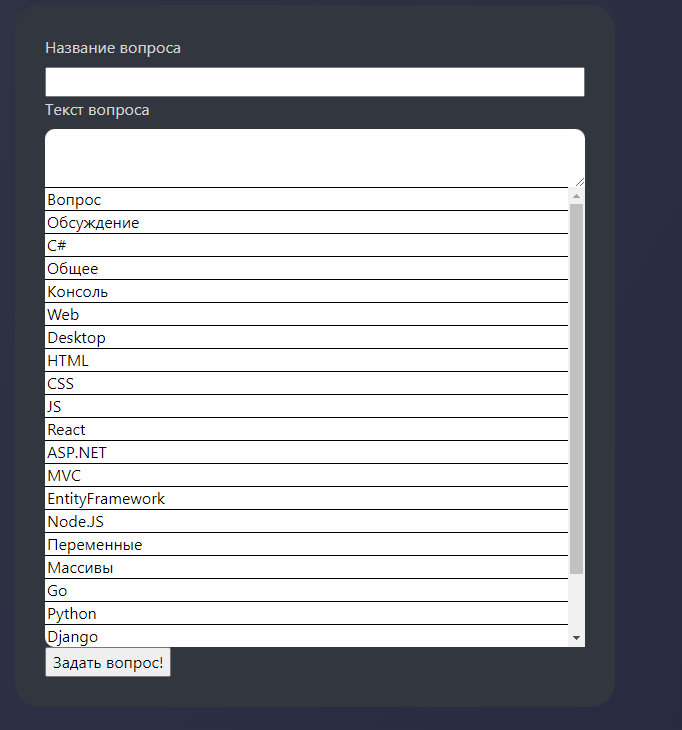


Рис. 19 «Страница добавления вопроса»

Здесь можно увидеть три поля для ввода: название вопроса, текст вопроса и поле выбора категорий. После заполнения всех полей и отправки формы на сайт, пользователя перекинет на страницу с его вопросом, где его можно будет отредактировать или удалить.

Также пользователь может отвечать на вопросы других пользователей. Для этого нужно перейти на соответствующую страницу, кликнув на любой вопрос из главной:

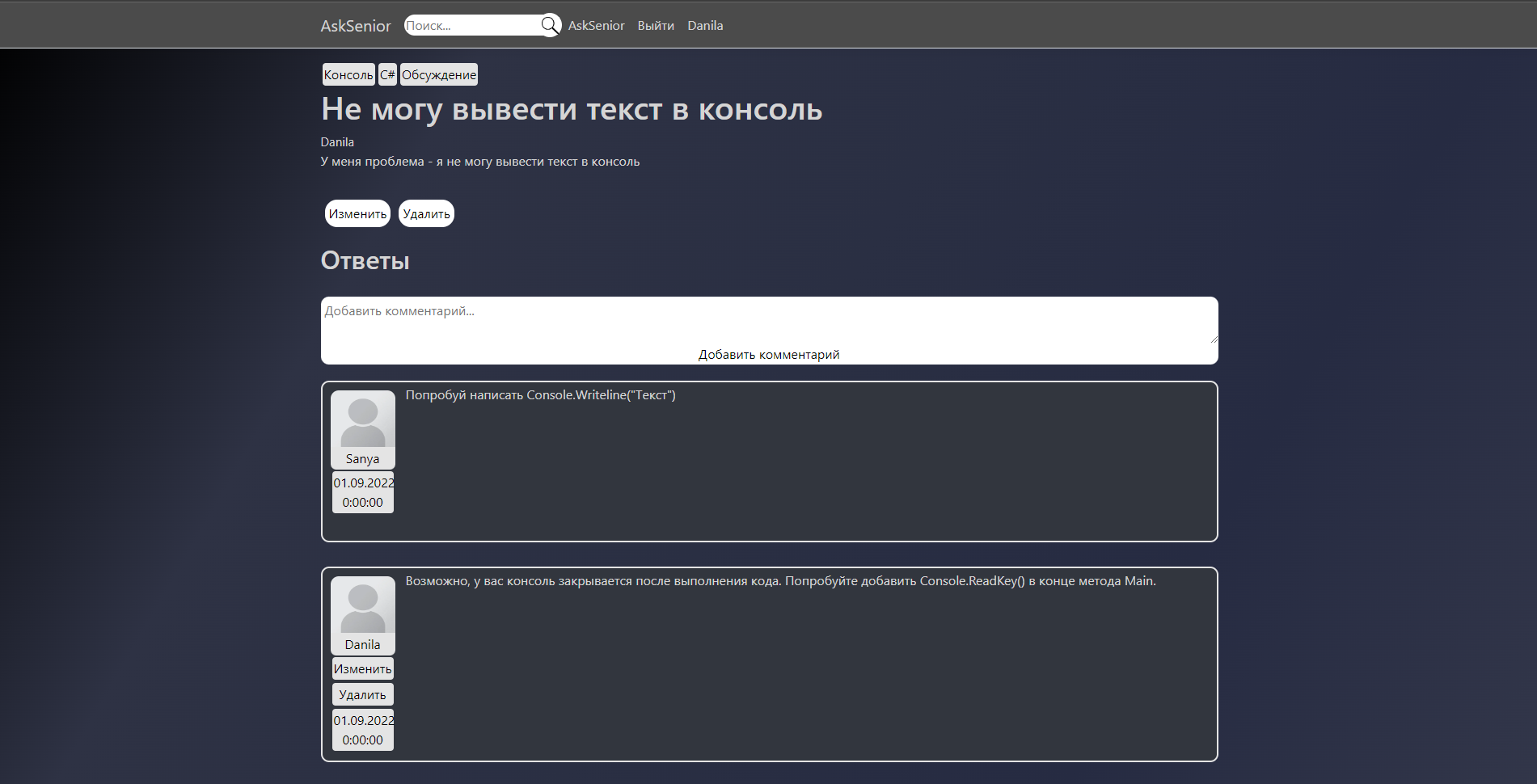


Рис. 20 «Страница вопроса»

Здесь можно увидеть заголовок вопроса, ссылки на изменение или удаления, поле для ответа на вопрос и ответы других пользователей. Чтобы ответить на вопрос, достаточно заполнить соответствующее поле и кликнуть по ссылке внизу.

Вернувшись на главную, можно раскрыть секцию с категориями вопросов:

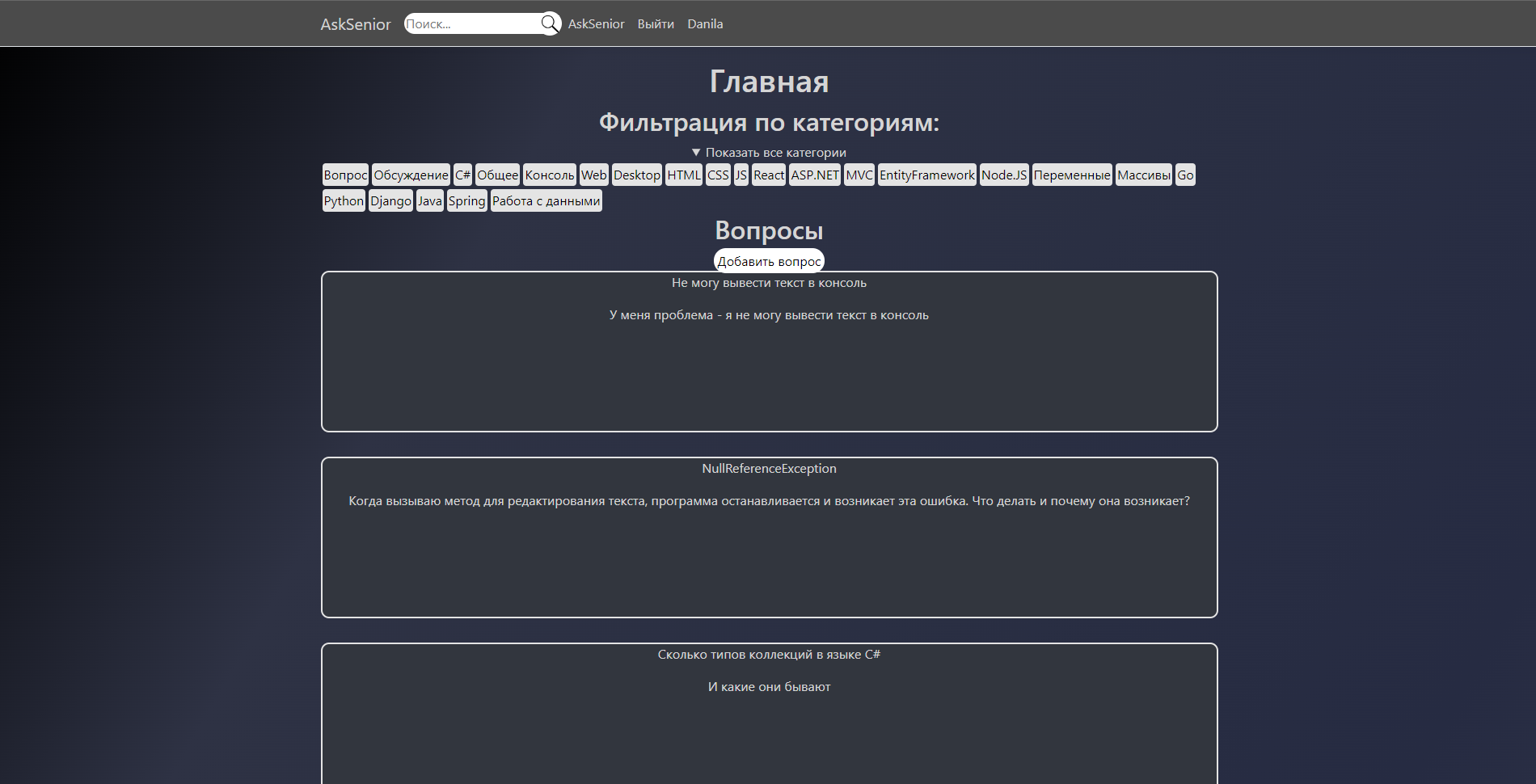


Рис. 21 «Категории для фильтрации»

Нажав на одну из них, можно отфильтровать вопросы по выбранной:

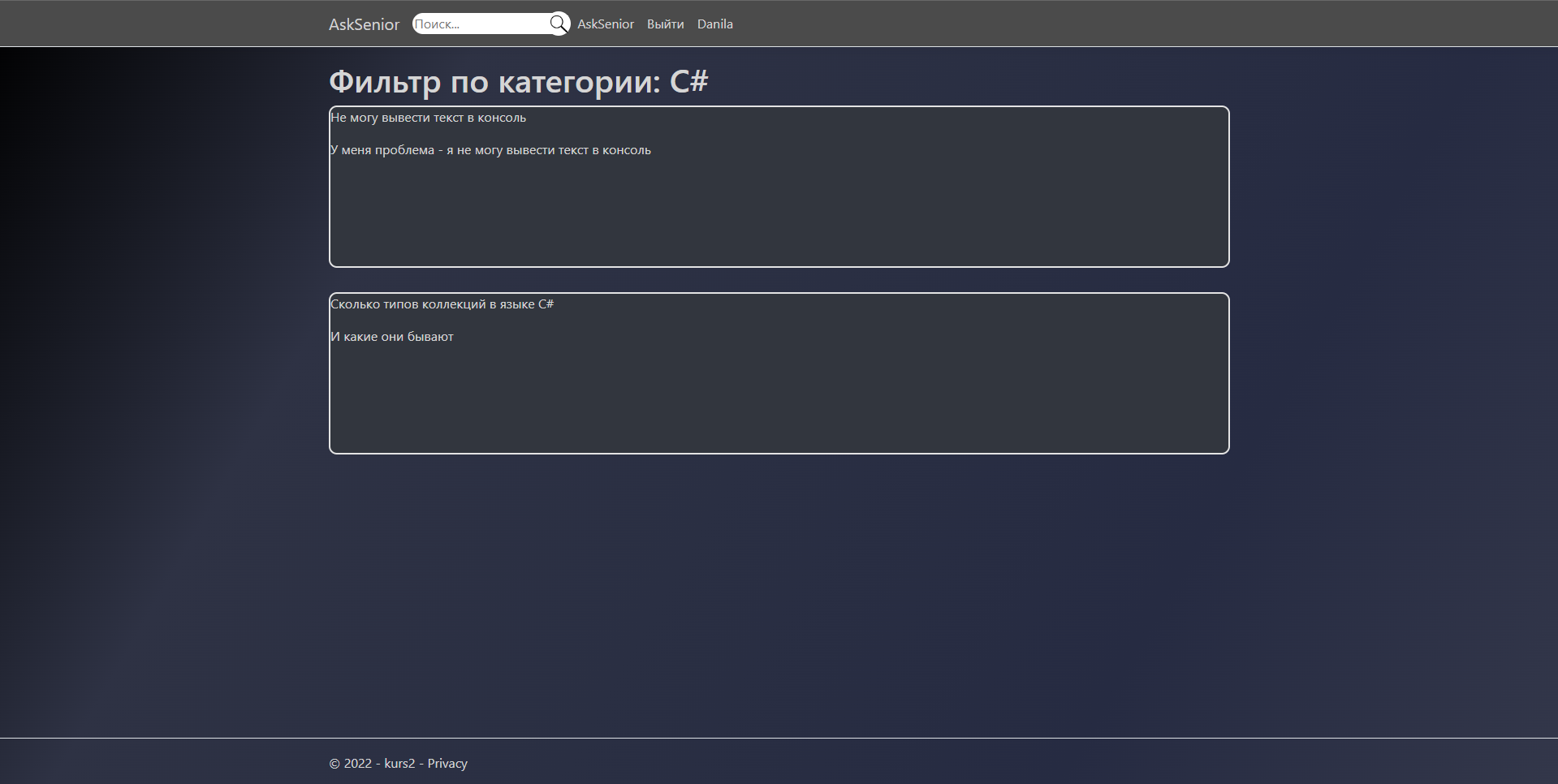


Рис. 22 «Вопросы, отфильтрованные по категории»

Для поиска определенного вопроса, достаточно ввести запрос в поисковую строку и нажать Enter или кликнуть по значку поиска слева. Результатом будет список вопросов:

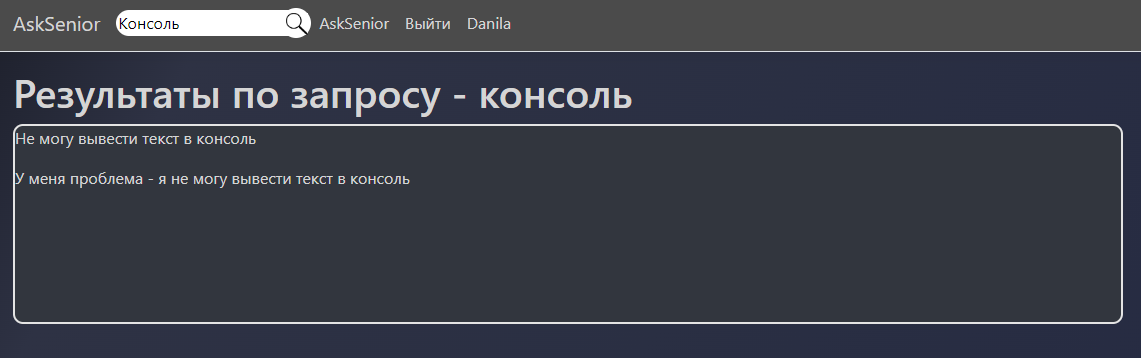


Рис. 23 «Результаты поиска»

Перейдя на страницу пользователя, можно увидеть его данные и заданные им вопросы:

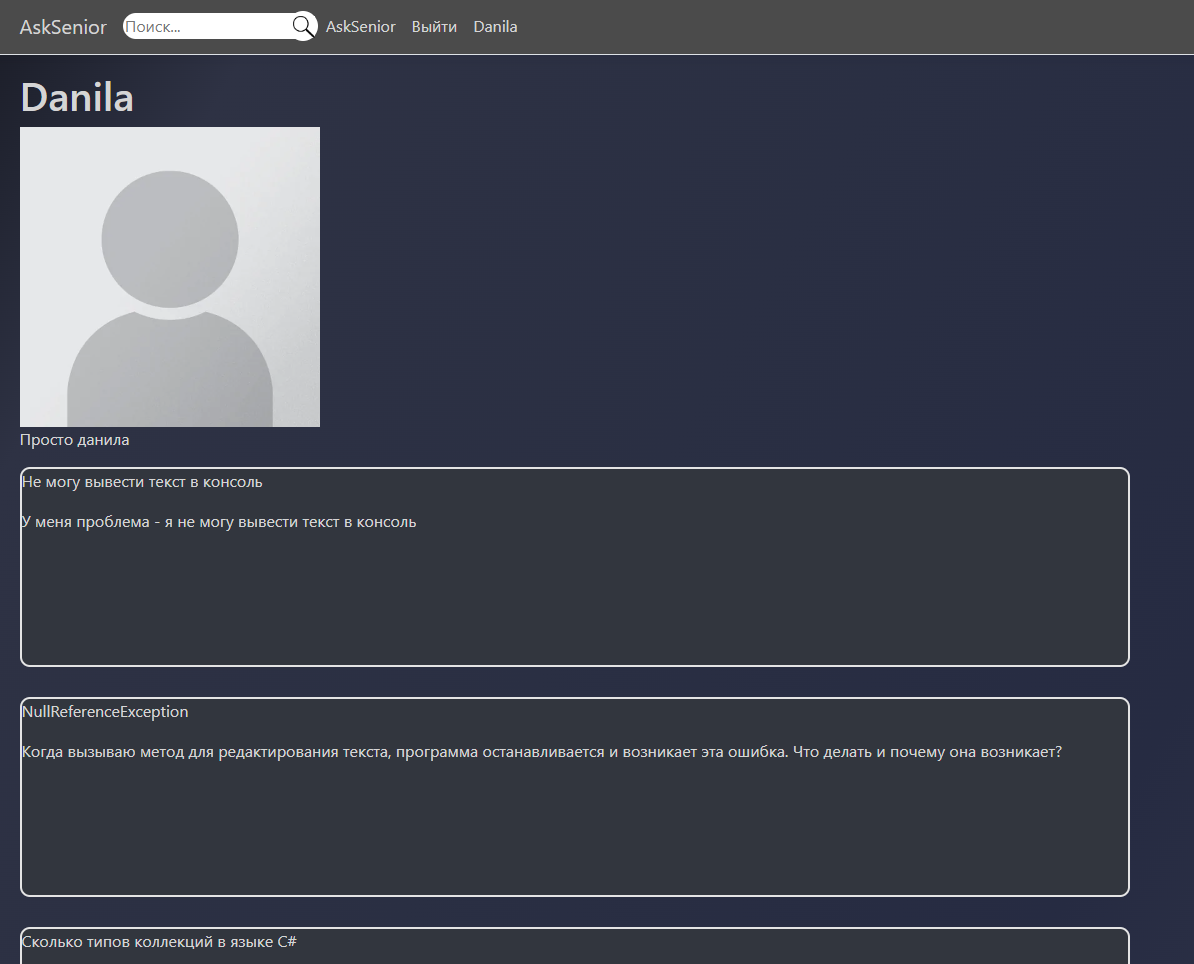


Рис. 24 «Страница пользователя»

Для администраторов доступны следующие функции:

Добавление категорий

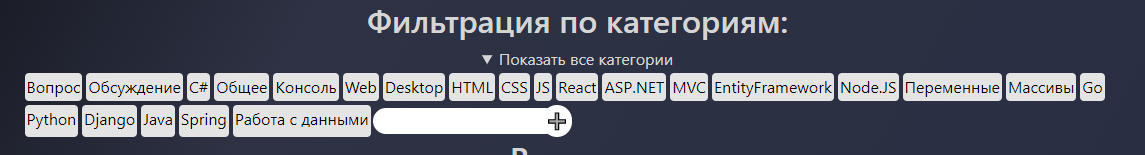


Рис. 25 «Добавление категории»

Изменение и удаление вопросов и ответов любых пользователей

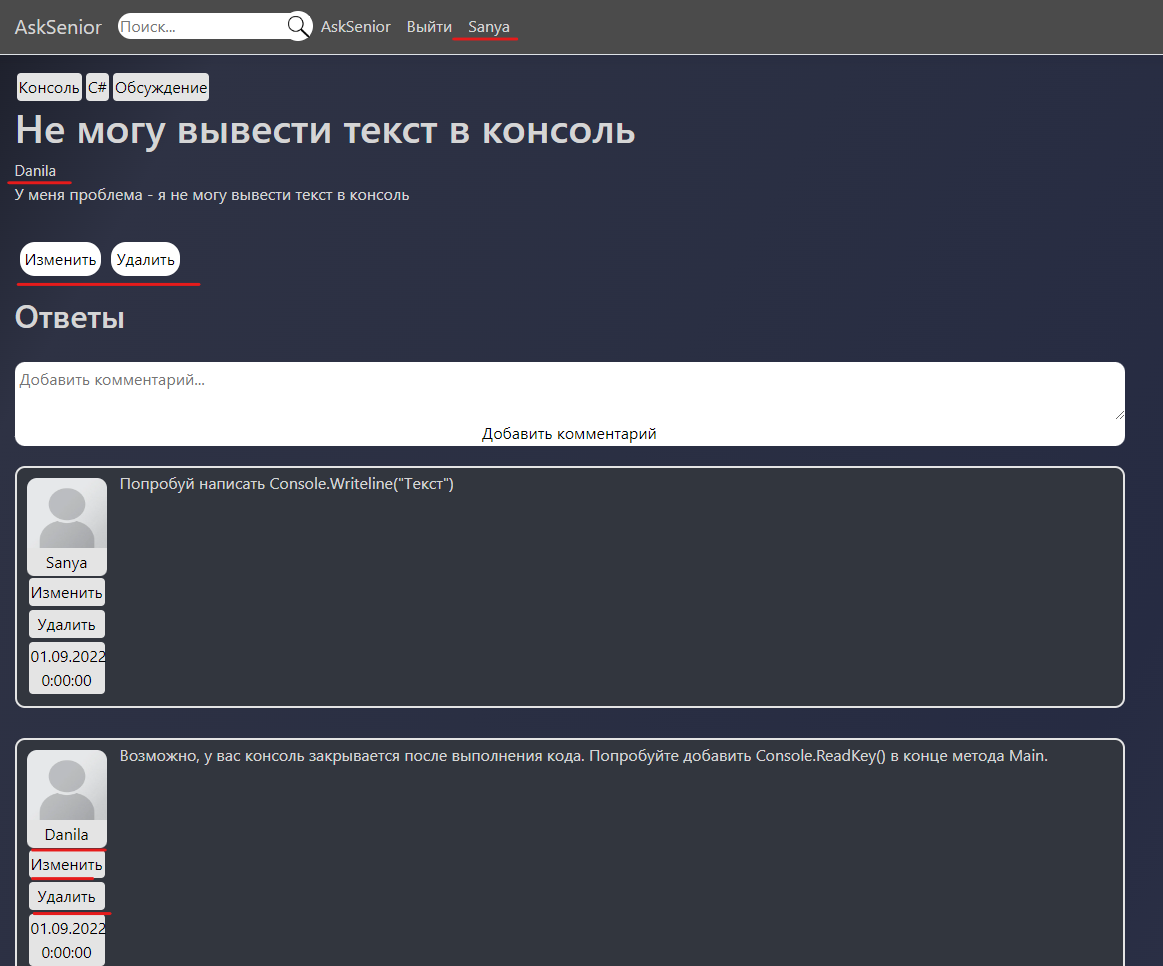


Рис. 26 «Демонстрация прав администратора»

# Тестирование и отладка

## Виды тестирования

Тестирование сайта – это последний и обязательный этап технической разработки сайта. Он играет важнейшую роль в процессе создания ресурса, так как именно от качества тестирования зависит дальнейшая жизнь ресурса. К сожалению, очень часто разработчики не уделяют должного внимания этому этапу, полагаясь на свой опыт. В результате существующие ошибки приводят к колоссальным затратам времени и денег.

Ведь ресурс, который имеет ошибки, вызывает негатив у посетителей и, как следствие, их потерю. В итоге владелец ресурса вынужден платить за доработку (а иногда за повторную разработку ресурса), а сотрудничество с бывшими разработчиками некачественного сайта оставляет только неприятный осадок.

1. **Цели и основные методы тестирования веб-сайта**

Целью тестирования является общая проверка реального функционирования веб-сайта на соответствие предъявленным требованиям. Весь этап представляет собой кропотливый труд специалистов, которые для выявления ошибок создают искусственные ситуации, которые могут возникнуть в период работы ресурса и анализируют «поведение» ресурса на предложенных условиях. После выявления багов (ошибок), тестировщик составляет отчет и передает его project-менеджеру, который распределяет работу по их устранению среди участников проекта. После доработки ресурс повторно тестируется. Такой цикл работ повторяется до тех пор, пока веб-сайт не достигнет поставленных результатов в проверке.

1. **Тестирование сайтов в разных браузерах**

Тестирование сайта на ошибки выполняется разными методами в разных браузерах. Чтобы добиться безупречного функционирования сайта, специалисты-тестировщики создают план действий, в который входит:

Тестирование функциональности;

Проверка usability (удобства пользования);

Тест производительности;

Проверка на безопасность;

UI Testing (тестинг интерфейса).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Шаги | Ожидаемый результат | Фактический результат |
| 1 | Регистрация пользователя | В шапке сайта нажать на ссылку «регистрация», на появившейся странице ввести все данные, нажать на кнопку «Зарегистрироваться» | Пользователь будет зарегистрирован и перенаправлен на главную страницу | Соответствует ожидаемому |
| 2 | Добавление Вопроса | На главной странице нажать на кнопку «Добавить вопрос», на появившейся странице ввести все данные, нажать на кнопку «Задать вопрос» | Пользователь будет перенаправлен на страницу с его вопросом, соответственно вопрос будет добавлен в базу данных и будет виден на главной странице сайта | Соответствует ожидаемому |
| 3 | Выход из аккаунта | Предварительно нужно войти на сайт, затем в шапке нажать кнопку «Выйти» в шапке | В шапке сайта снова отобразятся кнопки «Войти» и «Регистрация» вместо «Выйти», а на главной странице и странице с вопросом пропадут ссылки на добавление данных | Соответствует ожидаемому |

## 4.2. Отладка программного приложения

Таблица № 3 «Отладка Сайта»

|  |  |  |  |
| --- | --- | --- | --- |
| Название Ошибки | Скрин ошибки | Метод исправления | Результат |
| Не отображаются вопросы на главной |  | Исправить цикл для отображения вопросов |  |
| Черный текст на темном фоне |  | Исправить стиль для текста |  |

**Ошибки, возникшие в время разработки программы:**

**Ошибка №1**: Данные формы не отправляются после ввода данных и их отправки на сервер

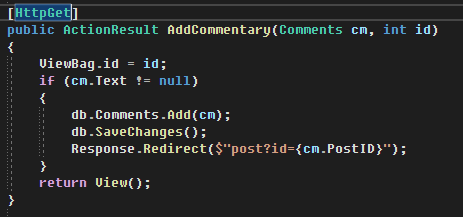


Рис. 27 «Ошибка 1»

**Ошибка №2**: При загрузке страницы не работает модель данных, соответственно, записи из базы данных не загружаются

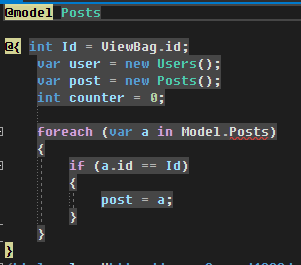


Рис. 28 «Ошибка 2»

**Ошибка №3:** При переходе на страницу, она не открывается

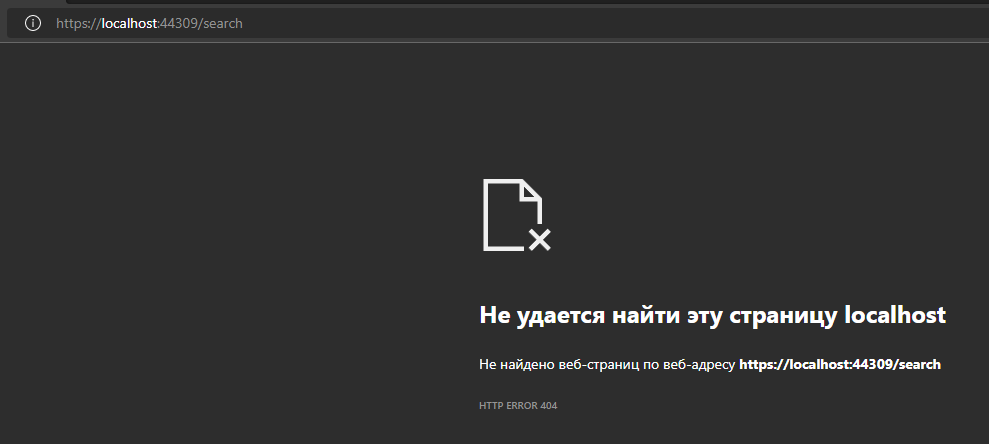


Рис. 29 «Ошибка 3»

**Ошибка №4**: При создании модели данных, не указан первичный ключ

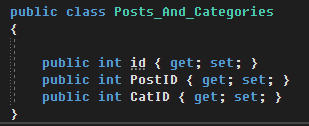


Рис. 30 «Ошибка 4»

# Расчет базовой себестоимости разрабатываемого продукта

## Исходные данные

Затраты времени и занятость специалистов соответствуют трудоемкости и сложности этапов создания приложения представлены в таблице № 5.

Таблица №5 «Исходные данные»

|  |  |  |  |
| --- | --- | --- | --- |
| **Этапы разработки программного продукта** | **Наименее возможная величина затрат, дни** | **Наиболее вероятная**  **величина затрат, дни** | **Наиболее возможная величина затрат, дни** |
| Анализ поставленной задачи | 1 | 2 | 3 |
| Разработка и проектирование базы данных | 2 | 3 | 4 |
| Дизайн сетевого сообщества | 4 | 5 | 6 |
| Программирование | 24 | 25 | 26 |
| Тестирование и отладка | 9 | 10 | 11 |
| Итого | 40 | 45 | 50 |

|  |  |
| --- | --- |
|  |  |

## 5.2 Расчёты затрат на выполнение программы

Создание компьютерной программы связанно со значительными затратами времени и соответственно большим удельным весом в себестоимости таких затратах, как оплата труда.

В работе по проектированию и разработке программы участвовал 1 специалист - разработчик.

Расчет оплаты специалистов производиться исходя из дневной тарифной ставки.

Тарифная ставка — в трудовом праве фиксированный размер оплаты: труда работника за выполнение нормы труда определённой сложности за единицу времени.

Фонд оплаты труда- суммарные денежные средства, израсходованные в течение определённого периода времени на заработную плату, включая: стимулирующие оплаты и дополнительную оплату.

Расчёт чистых затрат на выполнение программы рассчитывается из затрат на заработную плату разработчику умноженную на наиболее возможное количество дней разработки и суммирование с прочими затратами на разработку, такие как электричество.

Затраты на заработную плату разработчику – 22000/21\*50= 52 380,95 руб.

## 5.3 Расчет отчислений на социальное страхование и обеспечение

Обычный размер ставки - для наемного работника - составляет 30 %. Пенсионный фонд Российской Федерации - 22 %

* Фонд социального страхования - 2,9 %
* Фонды обязательного медицинского страхования - 5,1 %

Итого: 30 %

Таблица №6 «Расчет отчислений на социальное страхование и обеспечение»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Должность работника | Пенсионный фонд РФ – ПФР (22%), руб. | Фонд социального страхования РФ – ФССРФ (2.9%), руб. | ФОМС (5.1%), руб. | Итого отчислений на социальное страхование и обеспечение (ОСО), руб. |
| 1 | Разработчик | 11 523,80 | 1 519,04 | 2 671,42 | 15 714,26 |

Формулы расчета для таблицы №6:

ПФР = ОТ \* 22% = 52 380,95 руб. \* 22% = 11 523,80 руб.

ФСС РФ = ОТ \* 2,9% = 52 380,95 руб. \* 2,9% = 1 519,04 руб.

ФОМС = ОТ \* 5,1% = 52 380,95 руб. \* 5,1% = 2 671,42 руб.

ОСО = ПФР + ФССРФ + ФОМС 11 523,80‬ руб. + 1 519,04руб. + 2 671,42руб. = 15 714,26‬ руб.

* 1. **Расчет базовой себестоимости компьютерного продукта.**

Себестоимость - это текущие затраты организации на производство и реализацию продукции, выраженные в денежной форме.

Себестоимость продукции является качественным показателем, так как она характеризует уровень использования всех ресурсов, находящихся в распоряжении организации.

Косвенные расходы - это затраты, связанные с производством и реализацией продукции (работ, услуг), которые нельзя напрямую отнести к изготовлению продукции (работ, услуг) и можно учесть в расходах в том периоде, когда они понесены.

К косвенным расходам можно отнести, например, затраты на услуги связи, аренду офиса и т.д.

Самое главное отличие прямых расходов от косвенных расходов в том, что сумма косвенных расходов в полном объеме относится к расходам текущего отчетного (налогового) периода, а прямых — к расходам текущего периода по мере реализации товаров, работ, то есть с учетом остатков незавершенного производства.

Исключением являются случаи, когда деятельность организации связана с оказанием услуг. Данные налогоплательщики вправе относить сумму прямых расходов отчетного (налогового) периода в полном объеме на уменьшение доходов от производства и реализации данного отчетного (налогового) периода без распределения на остатки незавершенного производства.

Относить ли расходы к прямым или косвенным расходам компания определяет отдельно для каждого производственного цикла.

Если те или иные ресурсы согласно технологическим регламентам не включены в производственный цикл, не являются его неотъемлемой частью, то затраты на них можно учитывать в составе косвенных расходов.

**Перечень косвенных расходов**

Косвенные расходы - это расходы, которые нельзя прямо отнести на конкретные виды продукции (работ, услуг).

К косвенным расходам относятся:

1. административно - управленческие расходы;
2. расходы на отопление и освещение помещений;
3. расходы на страхование;
4. расходы на содержание общехозяйственного персонала;
5. амортизационные отчисления и расходы на ремонт основных средств управленческого и общехозяйственного назначения;
6. арендная плата за помещения общехозяйственного назначения;
7. расходы по оплате информационных, аудиторских, консультационных и т. п. услуг;
8. расходы, связанные со сбытом продукции:
9. другие аналогичные по назначению управленческие расходы.

Таблица №7 «Расчёт базовой себестоимости»

|  |  |  |
| --- | --- | --- |
| **№** | **Статьи затрат** | **Сумма в рублях** |
| 1 | Фонд оплаты труда (ФОТ) | 68 095,21 |
| 2 | Косвенные расходы (КР) | 6 809,52 |
| 3 | Расходы на продажу (РП) | 7 490,47 |
| 4 | Полная себестоимость (ПС) | 82 395,2 |

Формулы расчета для таблицы №6:

ФОТ = ОТ + ОСО = 52 380,95 руб. + 15 714,26‬ руб. = 68 095,21‬‬ руб.

В данной организации косвенные расходы составляют 10% от ФОТ.

КР = ФОТ \*10% = 68 095,21‬‬ руб. \* 10% = 6 809,52 руб.

РП = (ФОТ + КР) \* 10% = (68 095,21 руб. + 6 809,52руб.) \* 10% = ‬7 490,47‬ руб.

ПС = ФОТ + КР + РП = 68 095,21 руб. + 6 809,52руб. + 7 490,47‬ руб. = 82 395,2‬ руб.

Выгода = стоимость аналогичного программного продукта на рынке – себестоимость.

Аналоги данного программного продукта начинаются с 60 000 рублей, в зависимости от цены часа работы специалиста при доработки нужной конфигурации.

Итоговая выгода с разработки = 104 000 руб. – 82 395,2руб.

Выгода = 21 604,8 рублей.

# По итогам экономического расчёта полная себестоимость составила 82 395,2руб. и можно сделать вывод, что продукт полностью окупает затраты на его разработку, а также предоставляет выгоду от разработки в размере 21 604,8 рублей

# Заключение

Жизненный цикл программного обеспечения включает в себя все этапы его развития: от возникновения потребности в нем до полного прекращения его использования вследствие морального устаревания или потери необходимости решения соответствующих задач.

Жизненный цикл — это этапы, через которые программное приложение проходит с начала создания до конца разработки и внедрения. Чаще всего это подготовка, проектирование, создание и поддержка. Этапы могут называться по-разному и дробиться на более мелкие стадии.

По длительности жизненного цикла программные изделия можно разделить на два класса: с малым и большим временем жизни. Этим классам программ соответствуют гибкий (мягкий) подход к их созданию и использованию и жесткий промышленный подход регламентированного проектирования и эксплуатации программных изделий.

* Подготовка.

Разработчик решил запустить книжный интернет-магазин и начал анализировать, какие подобные сайты уже представлены в сети. Собрал информацию об их трафике, функциональности.

* Проектирование.

Разработчик выбрал компанию-подрядчика и обсудил с её специалистами архитектуру и дизайн будущего интернет-магазина.

* Создание.

Разработчик заключил с разработчиками договор. Они начали писать код, проектировать дизайн, составлять документацию.

* Поддержка.

Разработчик подписал акт сдачи-приёмки, и подрядчик разместил интернет-магазин на «боевых» серверах. Пользователи начали его посещать и сообщать о замеченных ошибках в поддержку, а программисты — оперативно всё исправлять.

Стандарты жизненного цикла представлены в ГОСТ 34.601-90

В ходе разработки дипломного проекта сайт для сетевого сообщества

В результате исследования предметной области спроектировано приложение, где учтены все требования к функциональным характеристикам. Для требований надежности и безопасности было учтено: каскадное удаление и выбор данных. Сайт запускается в соответствии с параметрами технических средств. Перегруженный код на сайте будет слишком сильно нагружать компьютер пользователя, в следствии чего, пользователь будет испытывать дискомфорт при использовании программного продукта, была произведена оптимизация, в результате которой нагрузка на конечный клиент должна снизиться. При разработке были учтены различные нюансы совместимости с разными браузерами, что должно сказаться на отображении стилизованных элементов интерфейса и привести к единообразию на всех доступных браузерах.

В приложении выявлены различные нюансы, которые учтены в разработке приложения, где пользователь программы легко может разобраться в интерфейсе, ведь он интуитивно-понятен.

Проведены тестирование и отладка на этапах его разработки. Проведены тесты чёрного и белого ящика, показаны ошибки, возникшие при отладке.

Экономическая часть. Рассчитана базовая стоимость разрабатываемого продукта, определены исходные данные. Выполнен расчёт затрат на реализацию, отчислений на социальное страхование, также рассчитана себестоимость.

Разработан прототип сайта со следующими функциональными характеристиками:

* Ввод личных данных пользователя (имя пользователя, пароль, фото, дата рождения и т.д.)
* Вывод данных хранящихся в базе данных (заголовок вопроса, текст вопроса, имя пользователя, время ответа и т.д.)
* Фильтрация и сортировка выводимых данных таблиц по заданному параметру (категории)
* Выбор и удаление данных таблиц
* Хранение данных и защита от искажения содержимого таблиц баз данных
* Реализация корректировки данных пользователей
* Поиск данных по заголовку или тексту вопроса
* Реализация разграничения прав доступа с заранее определенными ролями и преимуществом доступа к функциям этих ролей
* Возможность регистрации нового пользователя

Для хранения информации используется база данных, для программирования отдельных процедур и разработки интерфейса пользователя использовались следующие средства:

1. Visual Studio 2019 (C# ASP.NET)
2. Microsoft SQL Server 18
3. Draw.io
4. Figma

В результате проведенной работы выполнены все поставленные задачи в соответствии с выбранными методами, цель дипломного проекта достигнута.

# Список литературы

1. ГОСТ Р 56922-2016 Системная и программная инженерия. Тестирование программного обеспечения.
2. ГОСТ 19.201-78 Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.
3. ГОСТ 19.106-78 Единая система программной документации Требования к программным документам, выполненным печатным способом.
4. ГОСТ 19.504-79 ЕСПД. Руководство программиста. Требования к содержанию и оформлению
5. Распоряжение Правительства РФ от 17.11.2008 N 1662–р (ред. от 08.08.2009) «О Концепции долгосрочного социально–экономического развития Российской Федерации на период до 2020 года»: [Электронный ресурс] // Справочная правовая система «Консультант Плюс». – Режим доступа: http://www.consultant.ru.
6. Федорова Г.Н. Разработка модулей программного обеспечения для компьютерных систем: учебник для студентов СПО /Г.Н. Федорова. 2-е изд., стер. - М.: Издательский центр «Академия»,2018. - 384с.
7. Федорова Г.Н. Осуществление интеграции программных модулей: учебник для студентов СПО / Г.Н. Федорова. 2-е изд., стер. - М.: Издательский центр «Академия»,2018. - 285 с.
8. Царёв, Р.Ю. Алгоритмы и структуры данных (CDIO): учебник /Р.Ю. Царёв, А.В. Прокопенко; Министерство образования и науки Российской Федерации, Сибирский Федеральный университет. - Красноярск: СФУ, 2016. - 204 с.
9. Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва: Издательство Юрайт, 2020. — 235 с.
10. C# 7.0. Справочник. Полное описание языка.: Пер. с англ. – СпБ.: ООО “Альфакнига”, 2018. – 1024 с.: ил. – Парал. тит. англ.
11. Microsoft Visual C#. Подробное руководство. 8-е изд. – СПб.: Питер, 2017. – 848 с.: ил. – (Серия «Библиотека программиста»).
12. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд.: Пер. с англ. – СПб. : ООО “Диалектика”, 2018 – 1328 с. : ил. – Парал. тит. англ.
13. Оппель, Эндрю Дж. SQL. Полное руководство / Оппель Эндрю Дж.- М.: Диалектика / Вильямс, 2016. - 902 c.
14. Грибов, В.Д. Экономика организации (предприятия): учебник /Грибов В.Д., Грузинов В.П., Кузьменко В.А. — Москва: КноРус, 2016. — 407 с. — (СПО). — ISBN 978-5-406-05026-2. — URL: <https://book.ru/book/918920>

# Приложения