Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

Кафедра информационных систем и программной инженерии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

«Технологии программирования»

на тему: Разработка программной системы «База знаний отраслей»

Выполнила:

студентка группы ИСТ-116

Никонова М. А.

Принял:

доцент Вершинин В.В.

Владимир, 2018

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc534763029)

[1 ВВЕДЕНИЕ 2](#_Toc534763030)

[2 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc534763031)

[3 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc534763032)

[3.1 Словарь предметной области 5](#_Toc534763033)

[3.2 Сценарий взаимодействия пользователей с системой 5](#_Toc534763034)

[4 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 6](#_Toc534763035)

[4.1 Создание диаграмм UML 6](#_Toc534763036)

[5 РЕАЛИЗАЦИЯ СИСТЕМЫ 12](#_Toc534763037)

[5.1 Общие принципы организации системы 12](#_Toc534763038)

[5.2 Реализация модели 12](#_Toc534763039)

[5.3 Реализация механизма регистрации, аутентификации и авторизации 12](#_Toc534763040)

[5.4 Реализация поддержки различных справочников 12](#_Toc534763041)

[5.5 Реализация создания, отправки и публикации статей. 13](#_Toc534763042)

[5.6 Пользовательский интерфейс. 13](#_Toc534763043)

[6 ЗАКЛЮЧЕНИЕ 20](#_Toc534763044)

[7 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc534763045)

[ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММНОЙ СИСТЕМЫ 22](#_Toc534763046)

# 1 ВВЕДЕНИЕ

Бесчисленное множество новых технологий, вызванных бурным ростом информатизации общества, делает нашу жизнь невозможной без быстрого доступа к информации. В наше время очень легко получить информацию, одним из способов быстрого доступа к ней является веб-приложение.

База знаний — важный компонент интеллектуальной системы. Они предназначены для поиска способов решения проблем из некоторой предметной области, основываясь на записях БЗ и на пользовательском описании ситуации.

# 2 ПОСТАНОВКА ЗАДАЧИ

В процессе разработки прототипа программной системы необходимо:

1. выполнить исследование и анализ предметной области;
2. разработать прототип ПС;
3. выполнить моделирование работы ПС;
4. разработать схему базы данных;
5. реализовать ПС с использованием выбранных средств и технологий.

Исходные данные:

1. методология проектирования и разработки RUP;
2. язык моделирования UML;
3. платформа разработки ASP.NET;

Минимальный набор функций: поддержка различных типов пользователей, чтение, добавление, редактирование статей.

# 3 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Базой знаний может воспользоваться любой человек, которому требуется информация. Если пользователь обладает знаниями, которыми хочет поделиться с миром, он может добавить статью в базу, где ее опубликуют после проверки, или он может дополнить уже существующую статью комментариями.

**Функциональные требования:**

1. Зарегистрироваться;
2. Авторизоваться и аутентифицироваться;
3. Выйти из системы;
4. Посмотреть список статей;
5. Добавление статей;
6. Публикация статьи;
7. Редактирование статей;
8. Комментирование статей;
9. Просмотр списка авторов;
10. Просмотр категорий;
11. Редактирование информации о пользователе;

**Нефункциональные требования:**

1. Реализация на платформе .NET;
2. Использование технологии ASP.NET;
3. Использование шаблона проектирования MVC;
4. База данных MS SQL;
5. Использование Entity Framework для работы с данными.

Работа с приложением организована следующим образом: на сайт базы знаний обращается человек для получения нужной ему информации, просматривает интересующую его статью. Если он хочет оставить к ней комментарий, то сначала регистрируется. После регистрации ему доступны следующие функции: добавление, редактирование статей. Так же он может производить поиск статей по авторам или категориям.

## **3.1 Словарь предметной области**

База знаний – база данных, содержащая статьи с информацией, отсортированных по отраслям.

Статья – текст, в котором содержится информация по какой-либо теме.

Комментарии – дополнение к уже существующей статье, комментирование ее содержимого.

Категория – группа, содержащая статьи, которые объединены общей темой.

## **3.2 Сценарий взаимодействия пользователей с системой**

Каждый пользователь имеет определенную роль, которая определяет его возможности. Можно выделить следующие роли и их возможности:

1. Администратор

Администрирует сайт, следит за его корректной работой, контролирует учетные записи пользователей и менеджеров.

1. Модератор

Контролирует публикацию статей, оценивает предложенные статьи, просматривает комментарии к уже существующим.

1. Пользователь

Просматривает интересующие его статьи, комментирует их при желании, предлагает свои собственные работы для публикации.

1. Гость

Есть возможность просмотра статей, для получения доступа к остальным функциям базы знаний требуется авторизация.

Также, каждый зарегистрированный пользователь имеет возможность авторизоваться в системе и выйти из своего аккаунта.

# 4 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## **4.1 Создание диаграмм UML**

**4.1.1 Диаграмма прецедентов**

1. Публикация предложенной статьи

Предусловие: Есть предложенные статьи

Актёр: Модератор

Основной поток: Актёр просматривает предложенные статьи, оценивает содержание, одобряет и публикует их.

Альтернативный поток: Если предложенная статья не удовлетворяет требованиям или подобный материал уже есть в базе знаний, то статья возвращается на доработку.

Постусловие: Статья появляется в базе знаний в общем доступе.

Что здесь писать не знаю

2. Просмотр новостей

Предусловие: Новость должна быть в статусе «Не прочитано»

Актёр: Пользователь

Основной поток: Актер просматривает новость и подтверждает, что он ее прочитал.

Альтернативный поток: Актёр забыл подтвердить прочтение новости, она так и остается в статусе «не прочитано».

Постусловие: Статус новости меняется на «прочитано».

3. Создание/модерация новостей

Предусловие: Новость в статусе «создано» или «отмодерировано»

Актёр: Администратор, менеджер

Основной поток: Актер создает новость, редактирует ее, после публикует ее для ознакомления остальных пользователей базы знаний. Ее статус меняется на «опубликовано»

Альтернативный поток: Публикация новости не требуется в данный момент или она еще не готова к публикации, то ее статус остается «не опубликовано».

Постусловие: Статус новости изменяется на «опубликовано».

Диаграмма прецедентов представлена на рисунке 1.

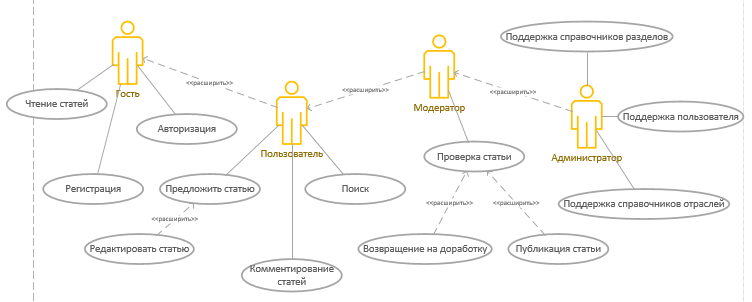


Рисунок 1. Диаграмма прецедентов

**4.1.2 Диаграмма последовательностей**

Диаграмма последовательностей для прецедента «предложить статью» представлена на рисунке 2.

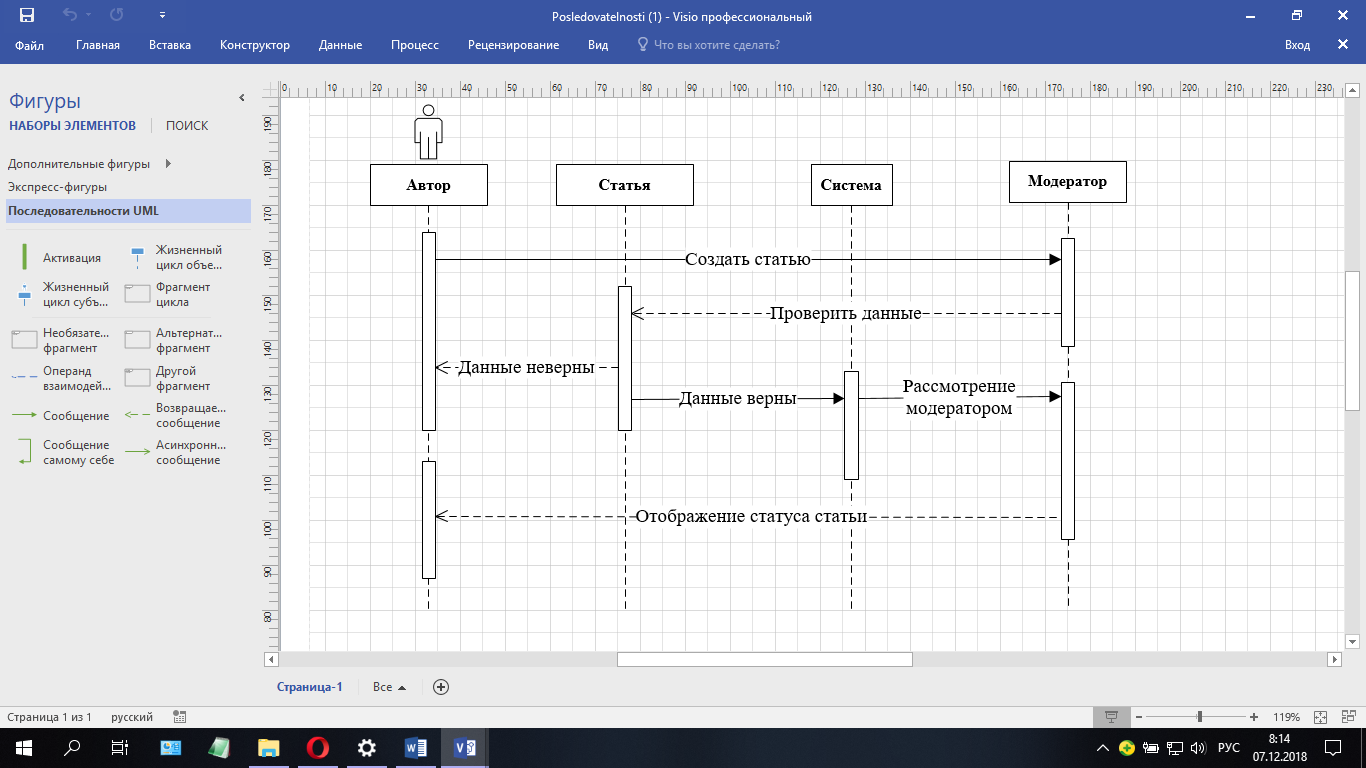


Рисунок 2. Диаграмма последовательностей для прецедента «Предложить статью»

**4.1.3 Диаграмма состояний**

Диаграмма состояний отображает жизненный цикл прецедента «предложение статьи».

Пользователь, нажимая вкладку «добавить статью», попадает на страницу создания статьи. Заполняет все необходимые поля, отправляет ее на проверку. После проверки статья будет либо опубликована, либо возвращена пользователю на доработку.

Диаграмма состояний объекта «Статья» представлена на рисунке 3.

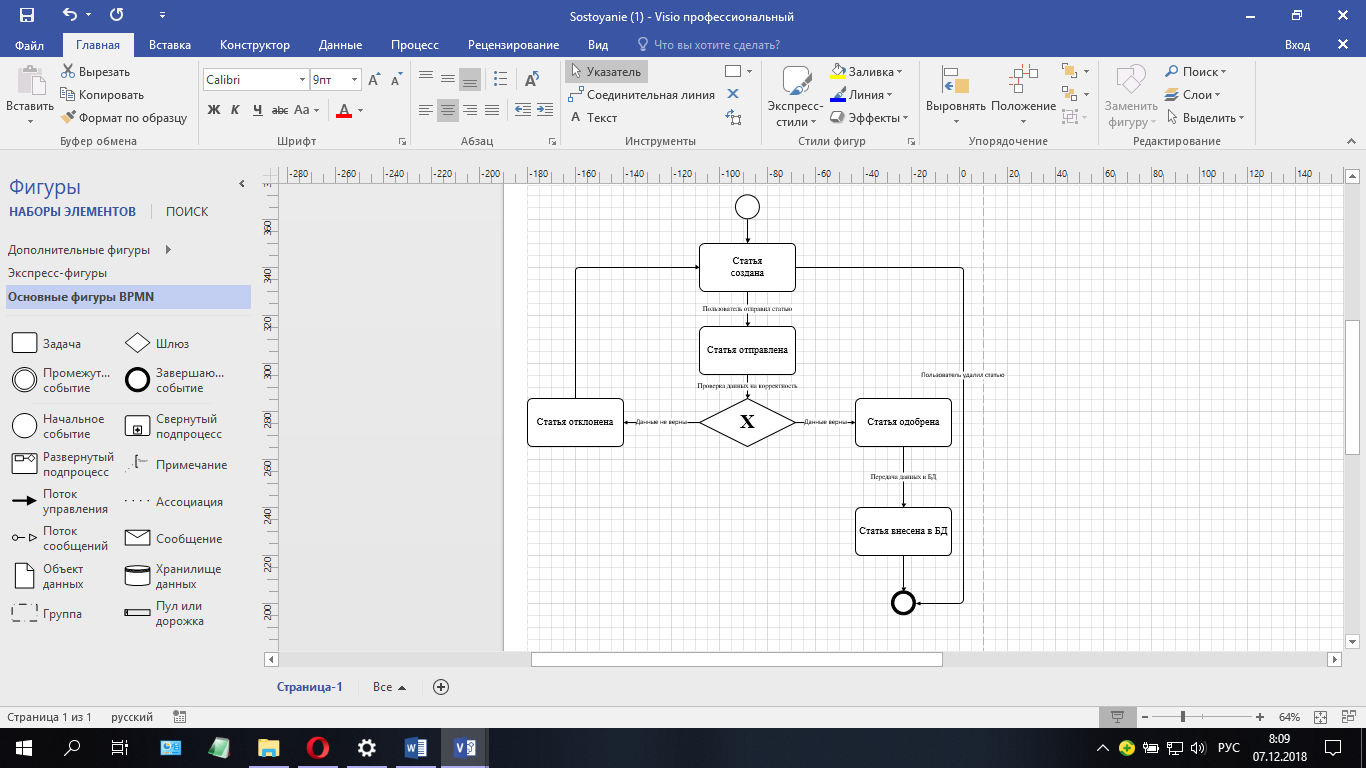


Рисунок 3. Диаграмма состояний предложения статьи.

**4.1.4 Диаграмма классов**

Описание сущностей:

Пользователь:

* идентификатор;
* фио;
* аккаунт;
* ник;

Аккаунт:

* идентификатор;
* электронная почта;
* пароль;
* роль;
* вход;
* выход;

Роль:

* идентификатор;
* наименование;

Категория:

* идентификатор;
* название;
* описание;
* добавление;
* удаление;

Статья:

* идентификатор;
* название;
* дата;
* автор;
* категория;
* содержание;
* статус;
* добавить статью;
* удалить статью;

Комментарий:

* идентификатор;
* пользователь;
* дата;
* содержание;
* статья;
* добавить комментарий;
* удалить комментарий;

Статус:

* идентификатор;
* тип статуса.

Диаграмма классов представлена на рисунке 4.

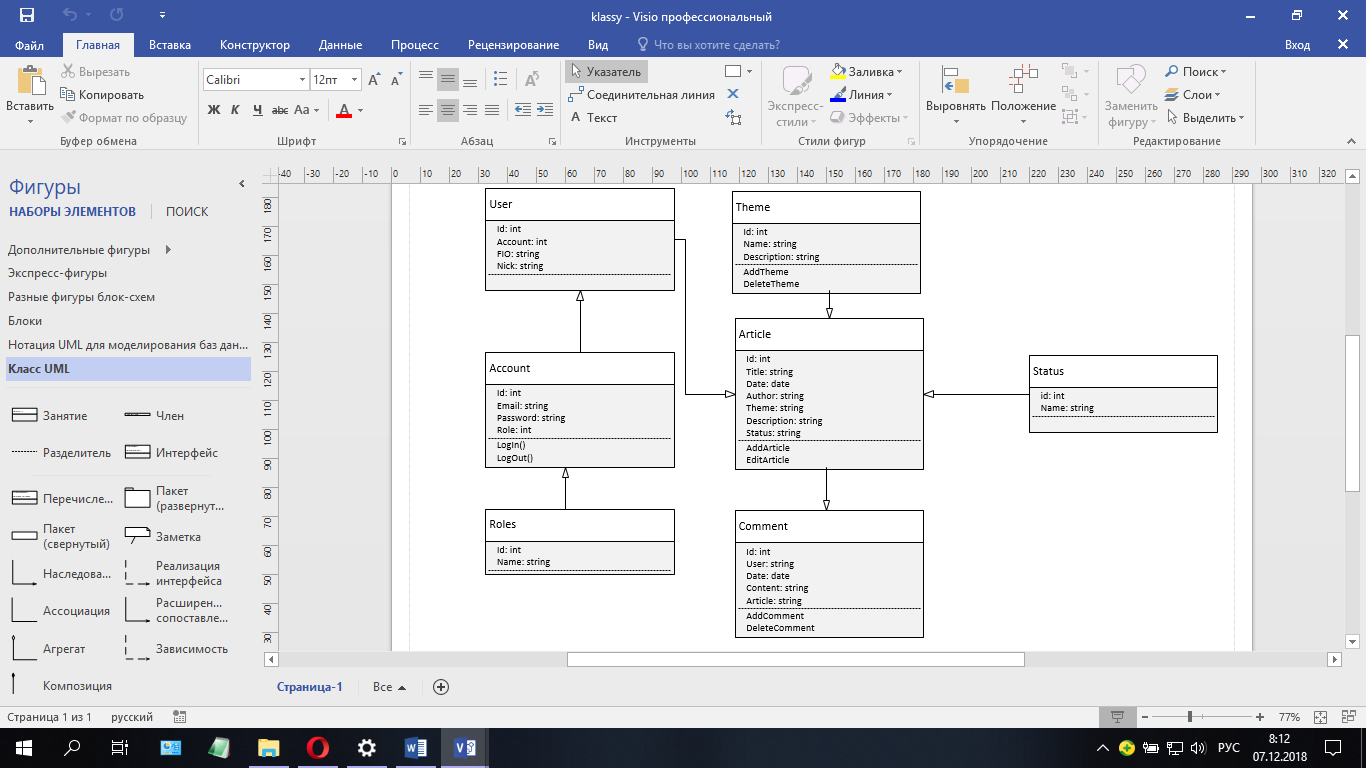


Рисунок 4. Диаграмма классов

**4.1.5 Физическая модель базы данных**

Физическая модель базы данных представлена на рисунке 5.

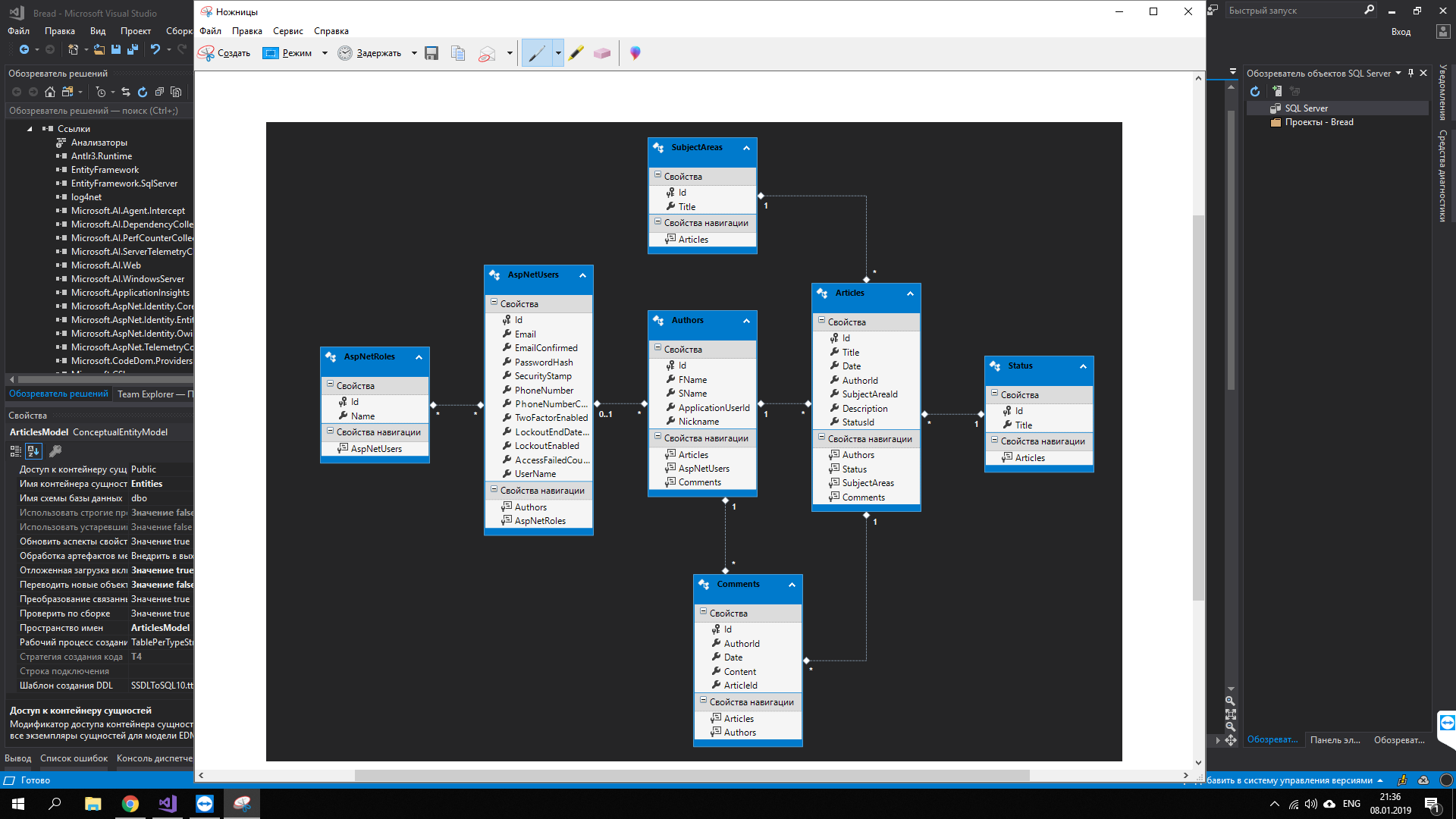


Рисунок 5. Физическая модель базы данных

# 5 РЕАЛИЗАЦИЯ СИСТЕМЫ

## **5.1 Общие принципы организации системы**

Разрабатываемая система представляет собой веб-приложение, которое построено на базе архитектуры ASP.NET MVC (код программы представлен в приложении А). [22]

В приложении организована реализация трех компонентов: контроллера, модели и представления.

Представления построены с использованием HtmlHelper.

Код программы представлен в приложении А. [29]

## **5.2 Реализация модели**

Для реализации данной системы был выбран подход Code-First и объектно-ориентированная технология доступа к данным Entity Framework. Сначала была создана база данных, состоящая из 7 таблиц. 3 из них являются встроенными в проект таблицами, используемых для работы с ролями и пользователями.

## **5.3 Реализация механизма регистрации, аутентификации и авторизации**

!!!!!!!Для реализации регистрации, аутентификации и авторизации пользователей были использованы стандартные методы шаблона Microsoft Identity. К методам контроллера имеют доступ, только пользователи, соответствующей роли. Это обеспечивает аннотация вида [Authorize(Roles="роль")]. Пользователю присвоена роль "User". Администратору присвоена роль "Admin". Незарегистрированные пользователи не могут оставлять комментарии, предлагать статьи, благодаря установке над методами [Authorize]. В личном кабинете имеется возможность сменить пароль.

## **5.4 Реализация поддержки различных справочников**

Пользователи системы имеют доступ к базе данных. Пользователь может предложить статью для публикации или отредактировать ее, если ему ее вернули на доработку, а также комментировать уже существующие статьи. Администратор может публиковать предложенные статьи после проверки, контролировать пользователей.

## **5.5 Реализация создания, отправки и публикации статей.**

Пользователь нажимает на кнопку «Добавить статью», заполняет все представленные поля, затем нажимает на кнопку создать или отправить. Система автоматически отправит данные в базу данных и присвоит первоначальный статус. Пользователь может отслеживать статус своих статей в личном кабинете.

Администратор может просмотреть конкретную статью, опубликовать, присвоив нужный статус, если статья прошла проверку.

## **5.6 Пользовательский интерфейс.**

Пользовательский интерфейс представлен на рисунках 6-7.

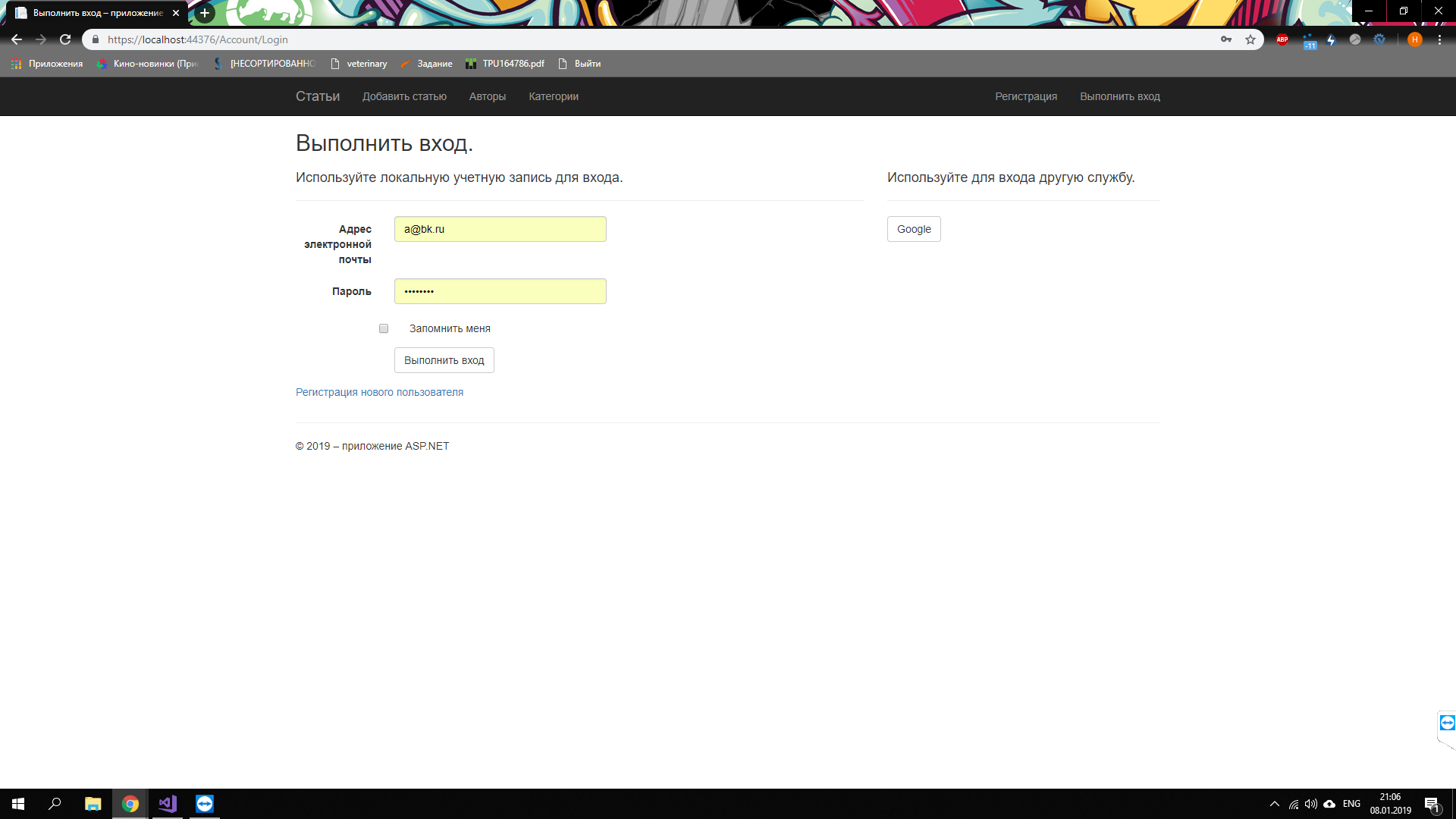


Рисунок 6. Страница авторизации.

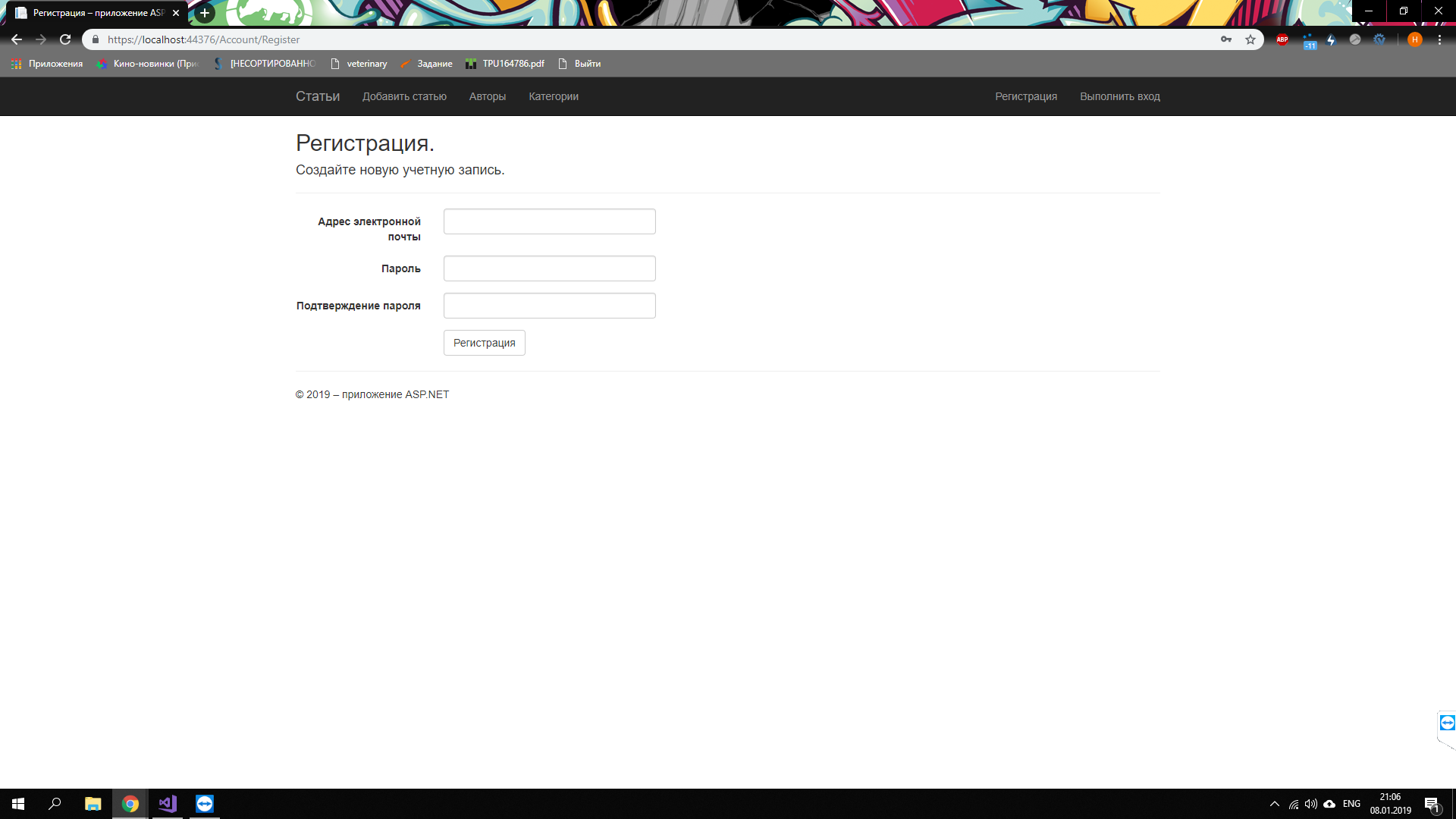


Рисунок 7. Страница регистрации.

Интерфейс зарегистрированного пользователя представлен на рисунках 8-15.

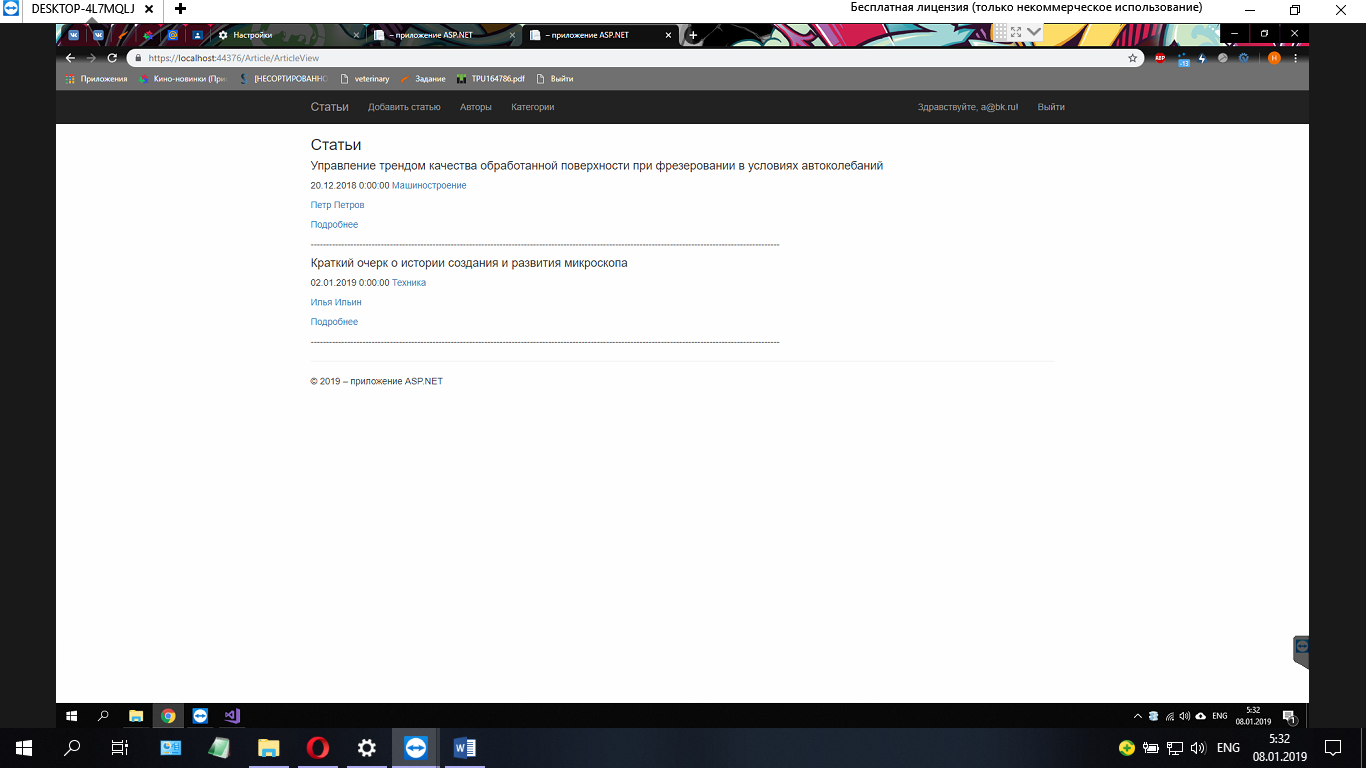


Рисунок 8. Список доступных статей.

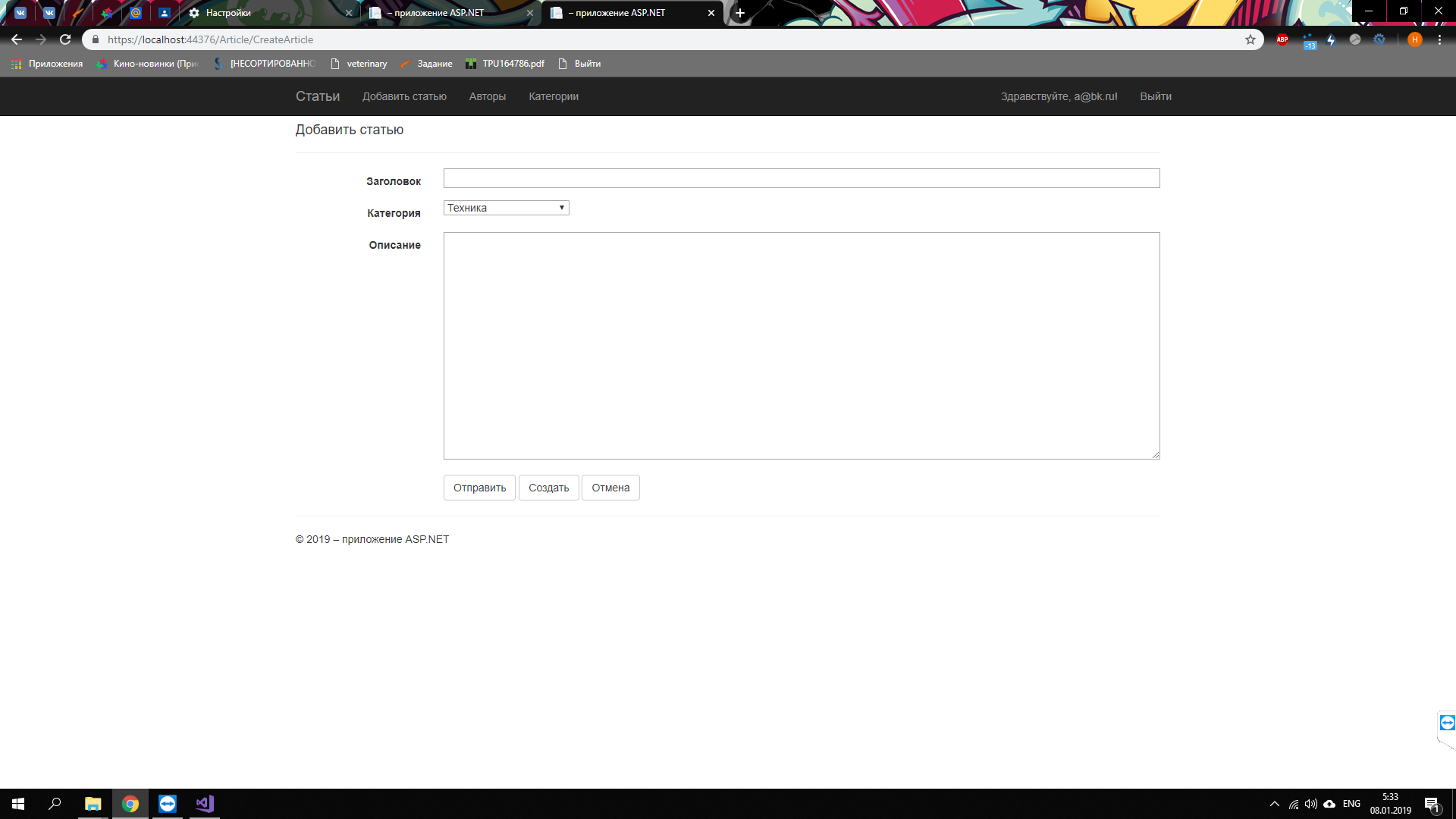


Рисунок 9. Окно добавления статьи.

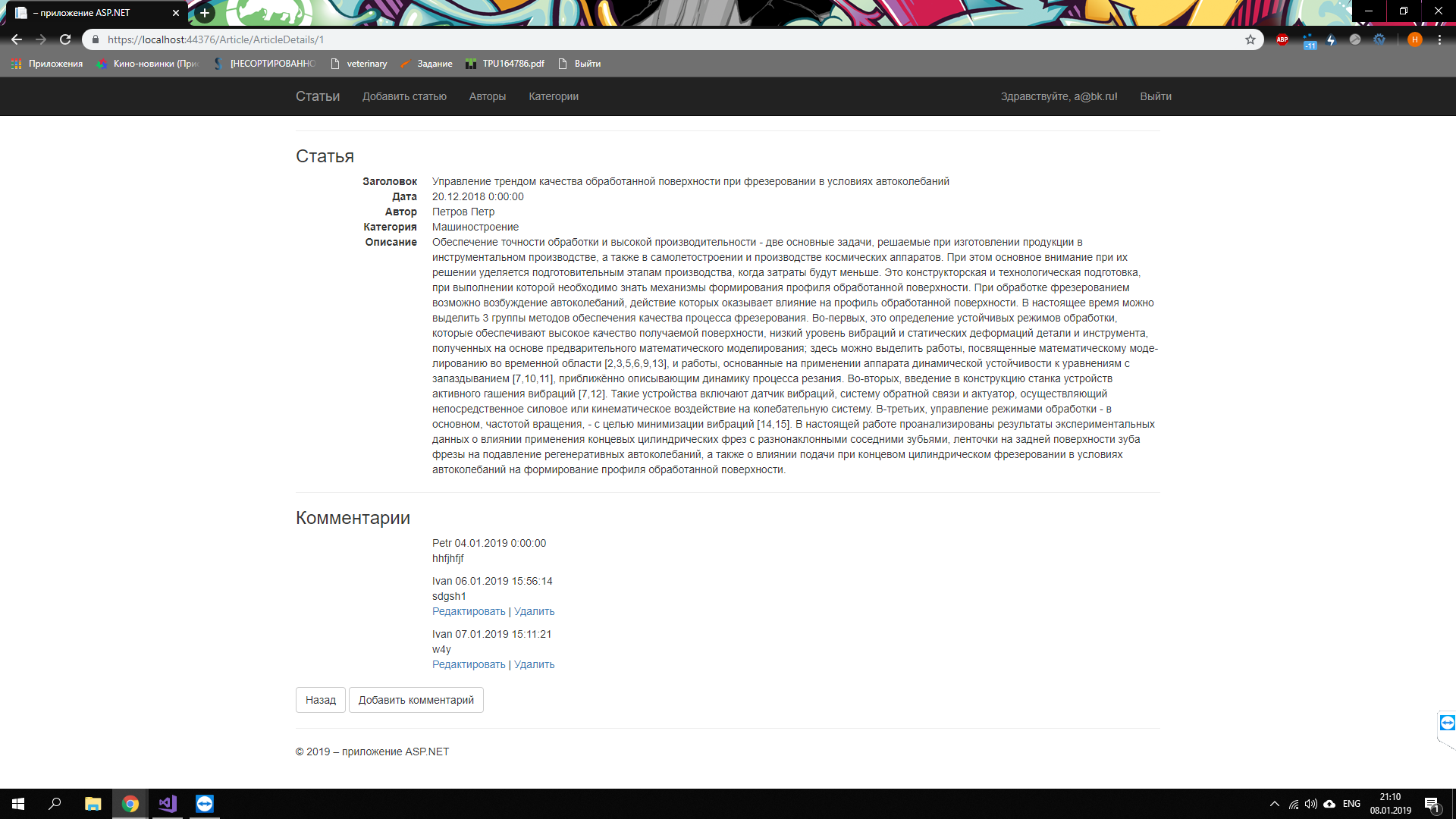


Рисунок 10. Чтение статьи.

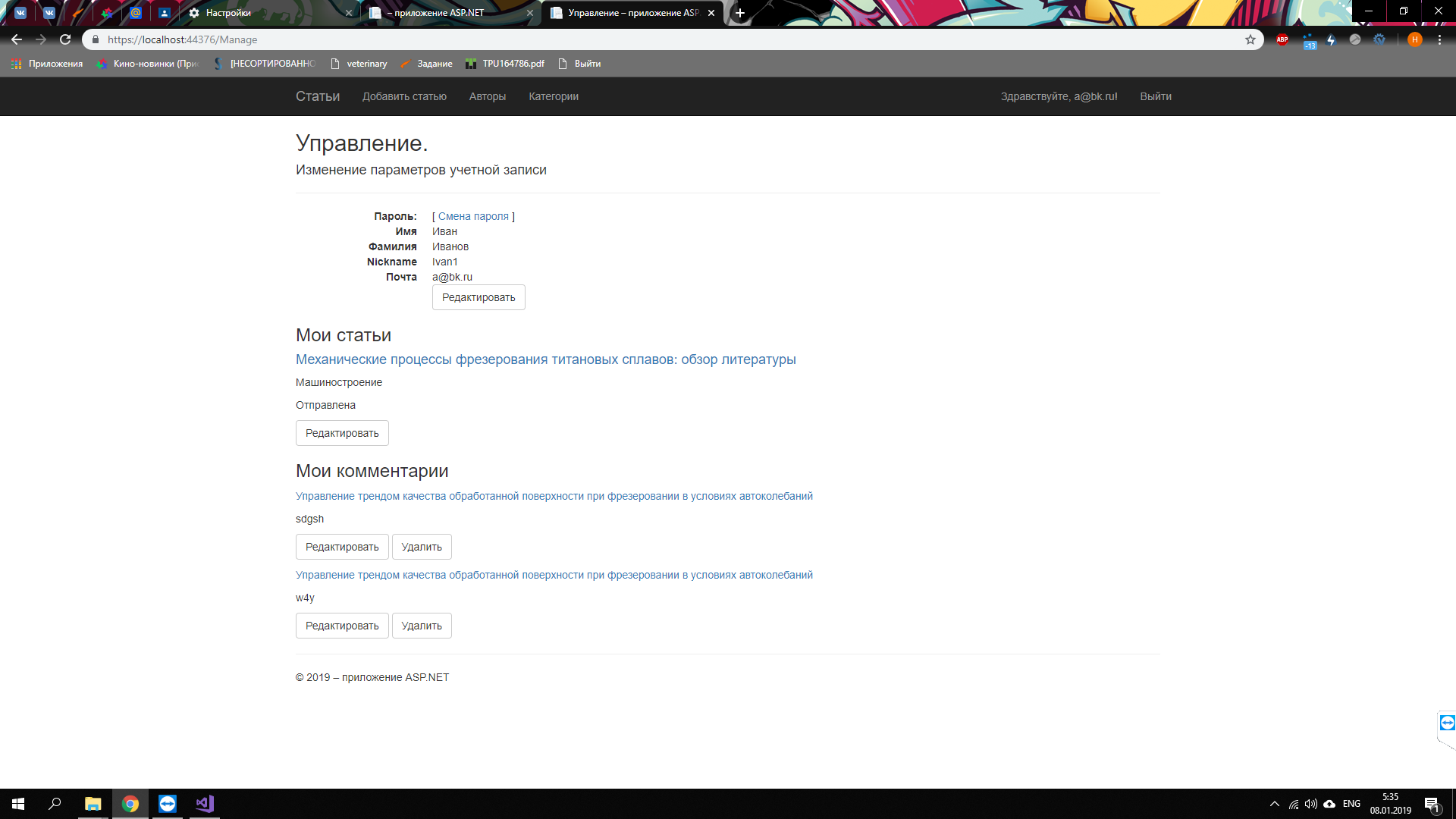


Рисунок 11. Окно личного кабинета.

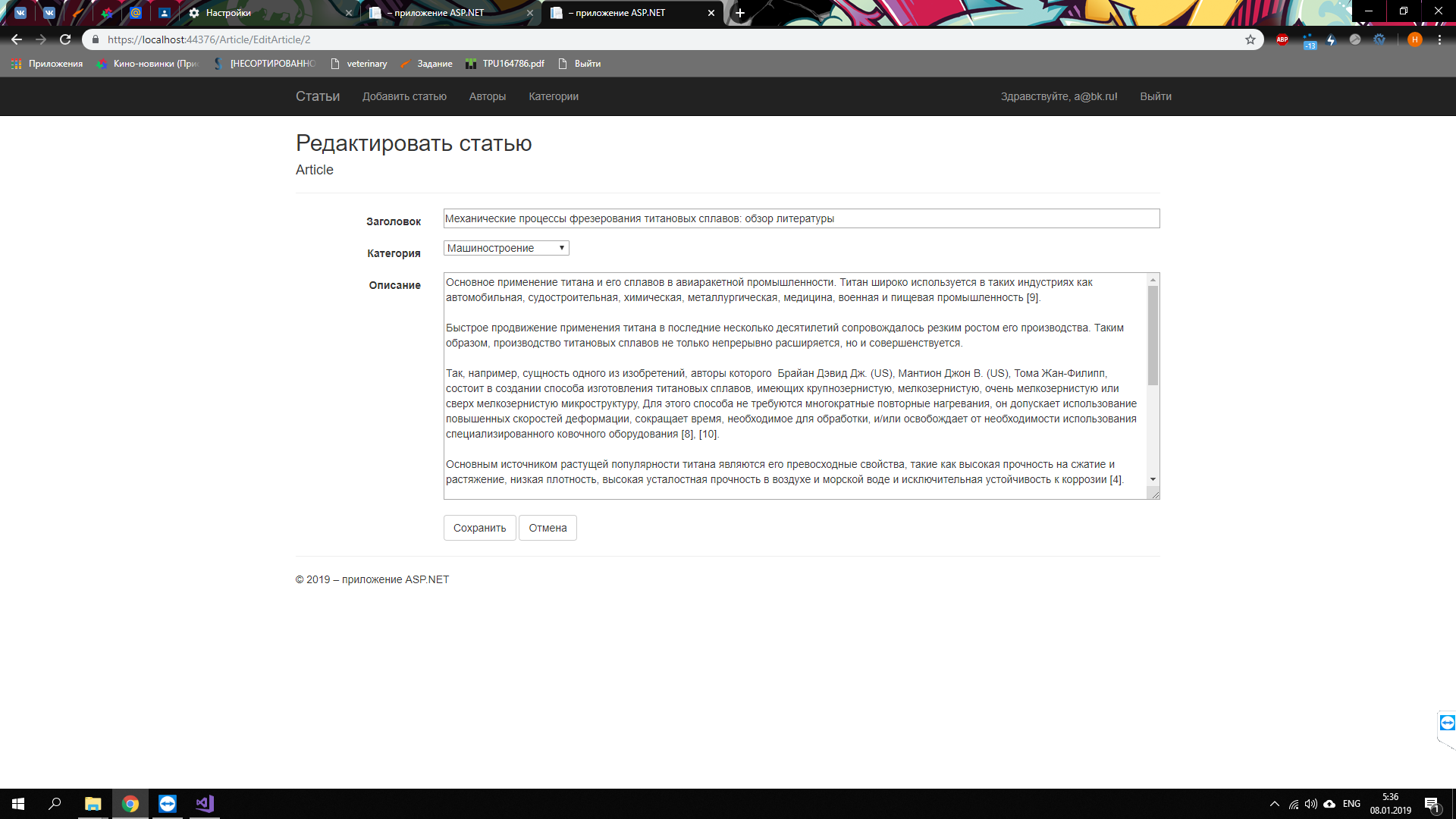


Рисунок 12. Окно редактирования статьи.

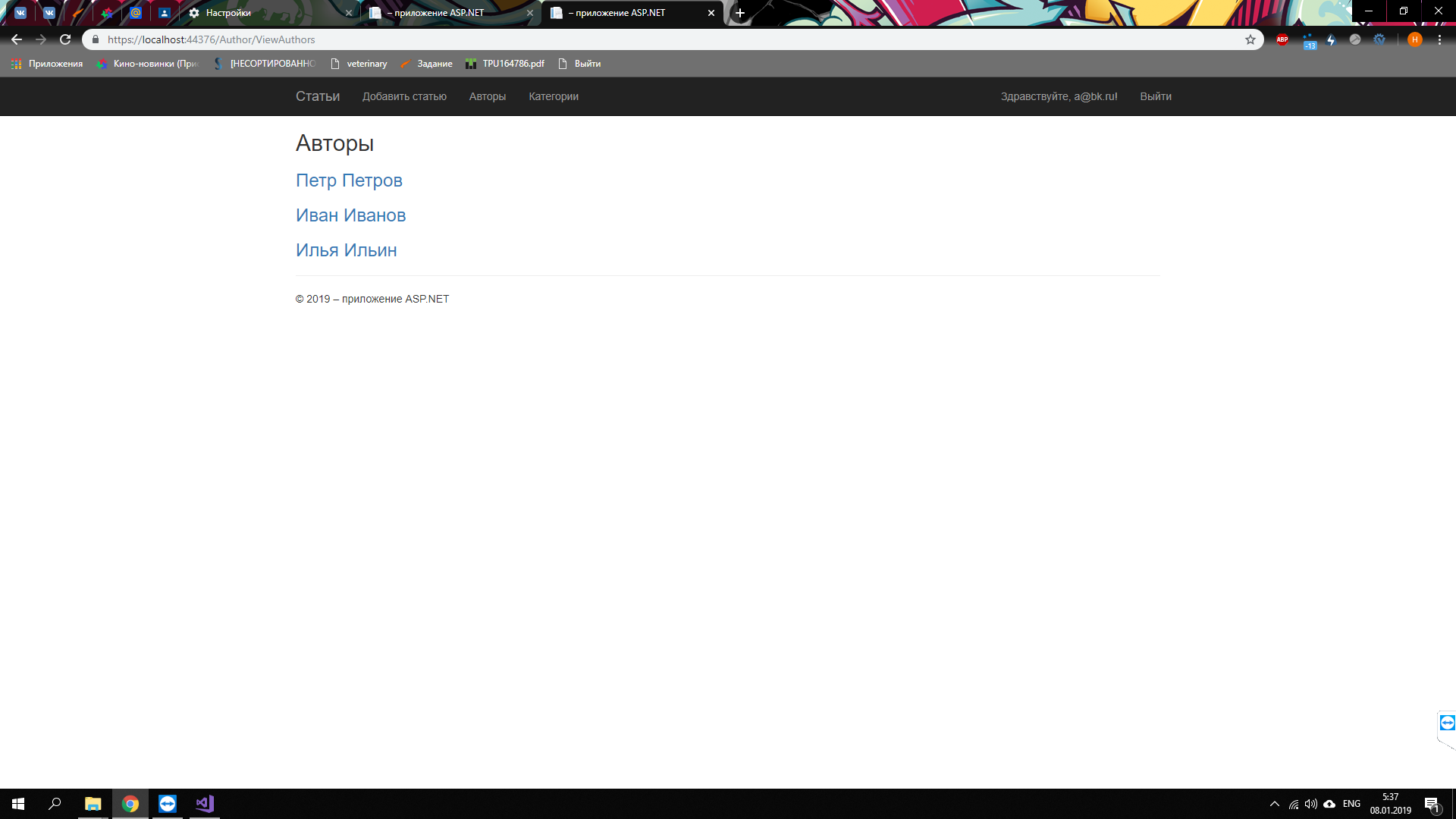


Рисунок 13. Список авторов.

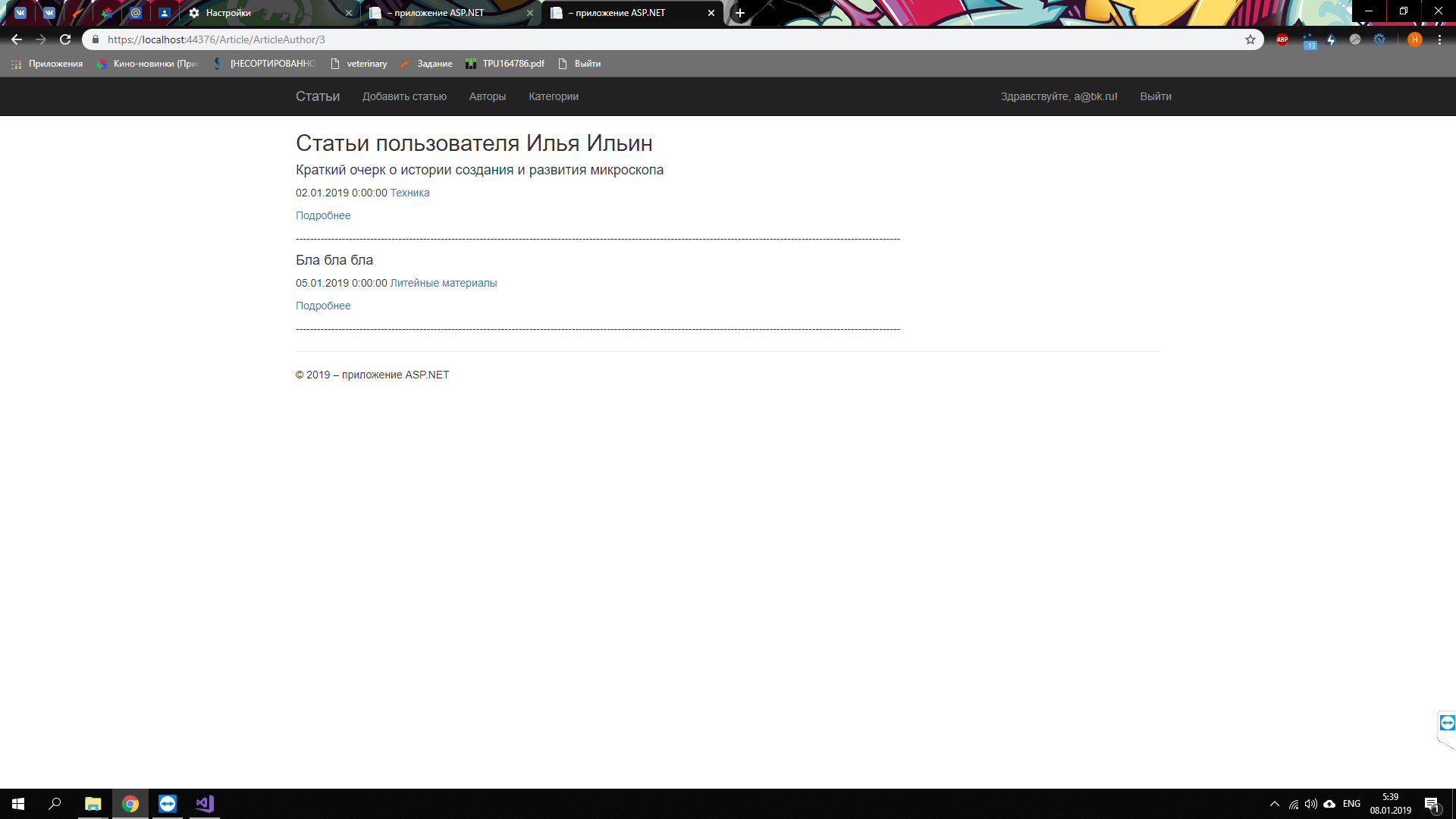


Рисунок 14. Информация об авторе.

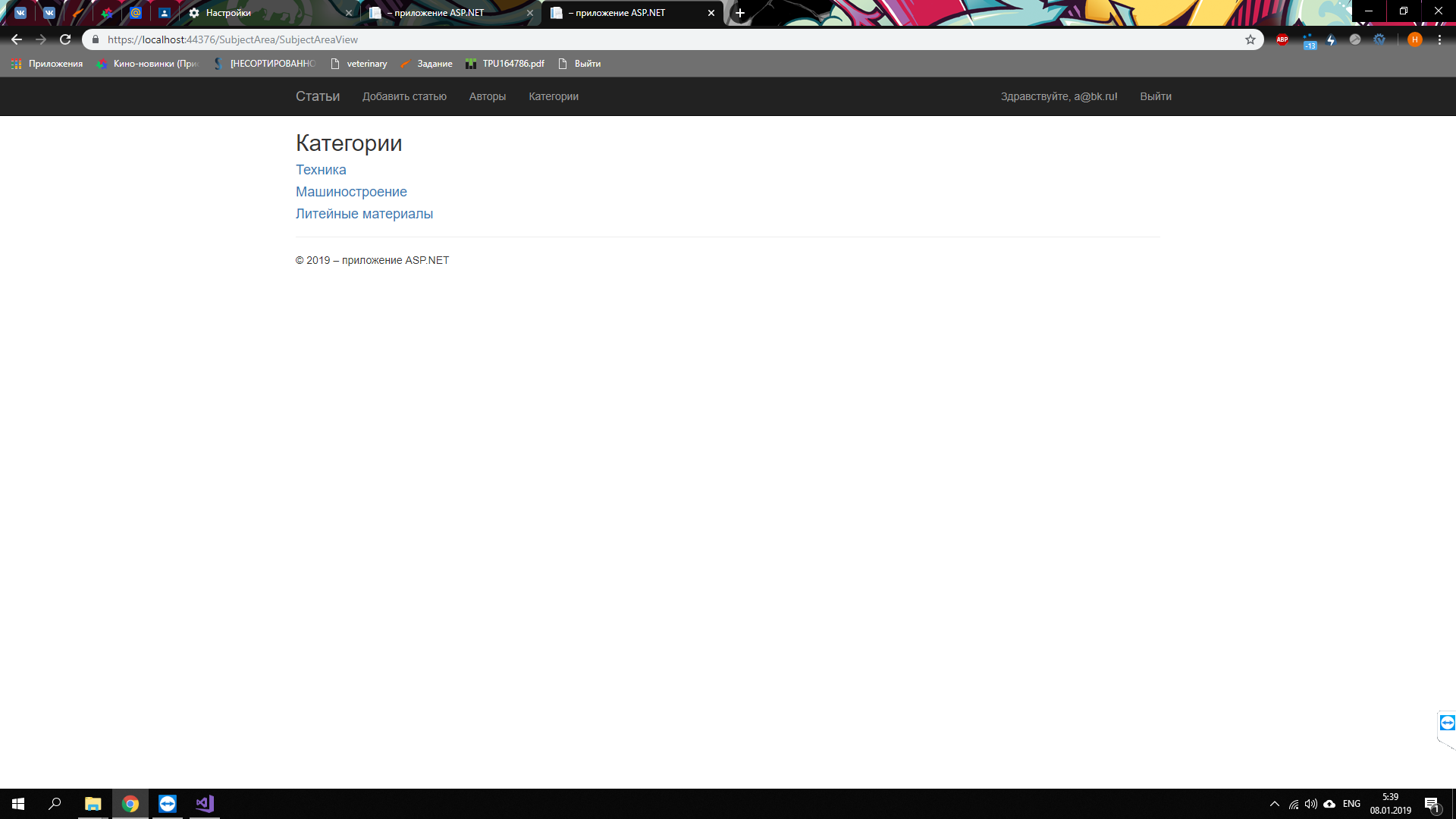


Рисунок 15. Категории.

Интерфейс администратора представлен на рисунках 16 - 21.

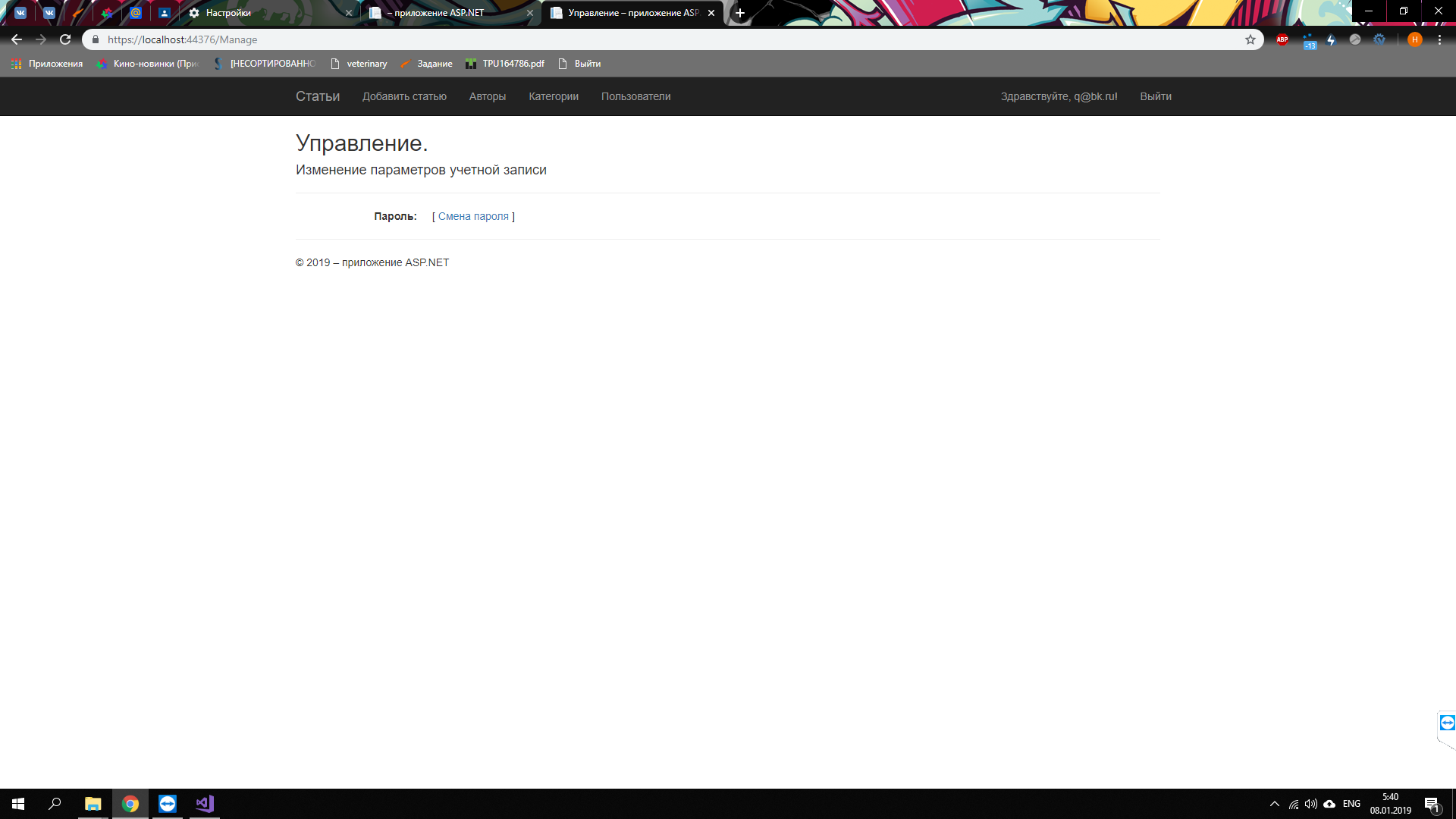


Рисунок 16. Личный кабинет администратора.

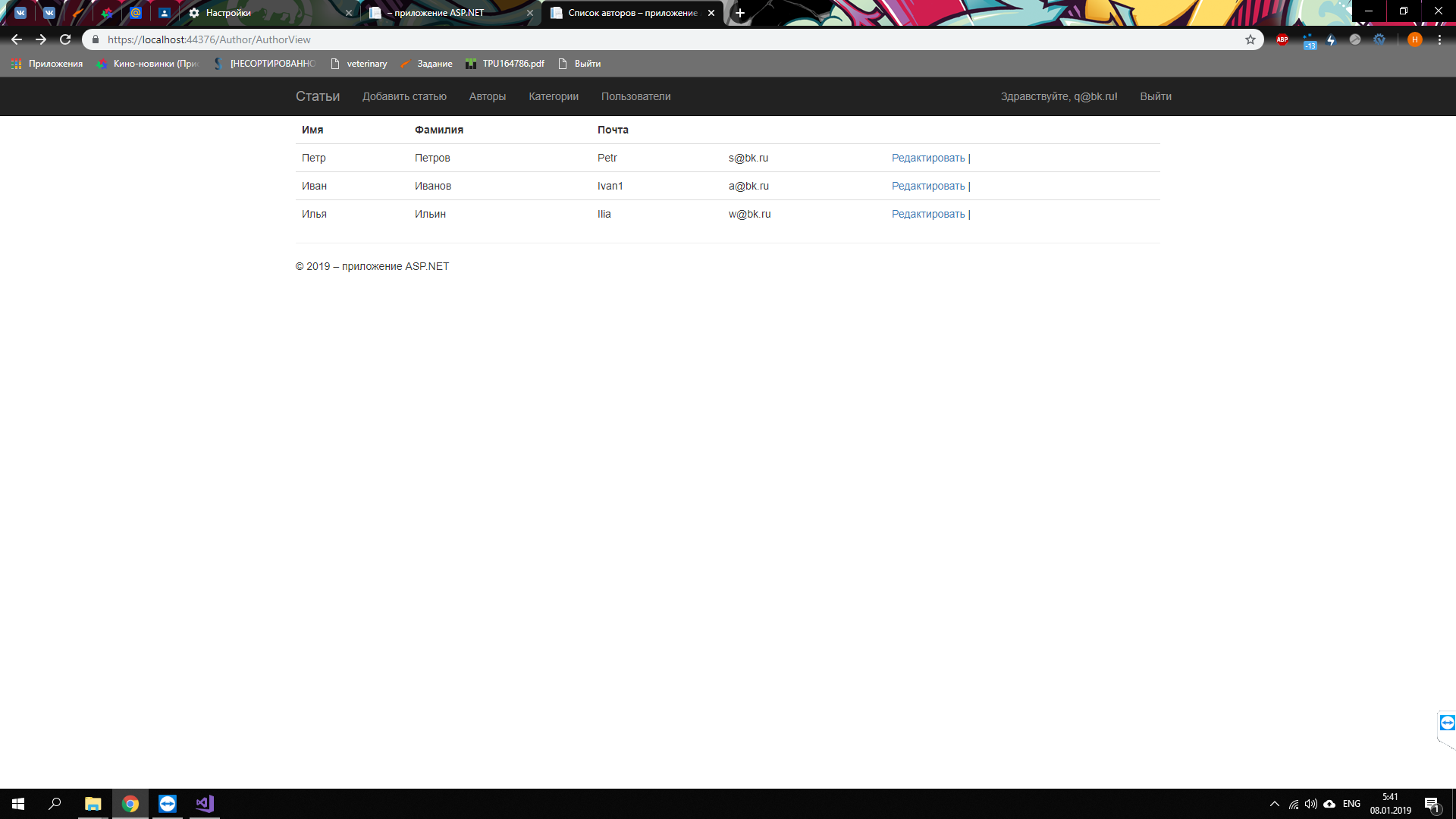


Рисунок 17. Список пользователей у администратора.

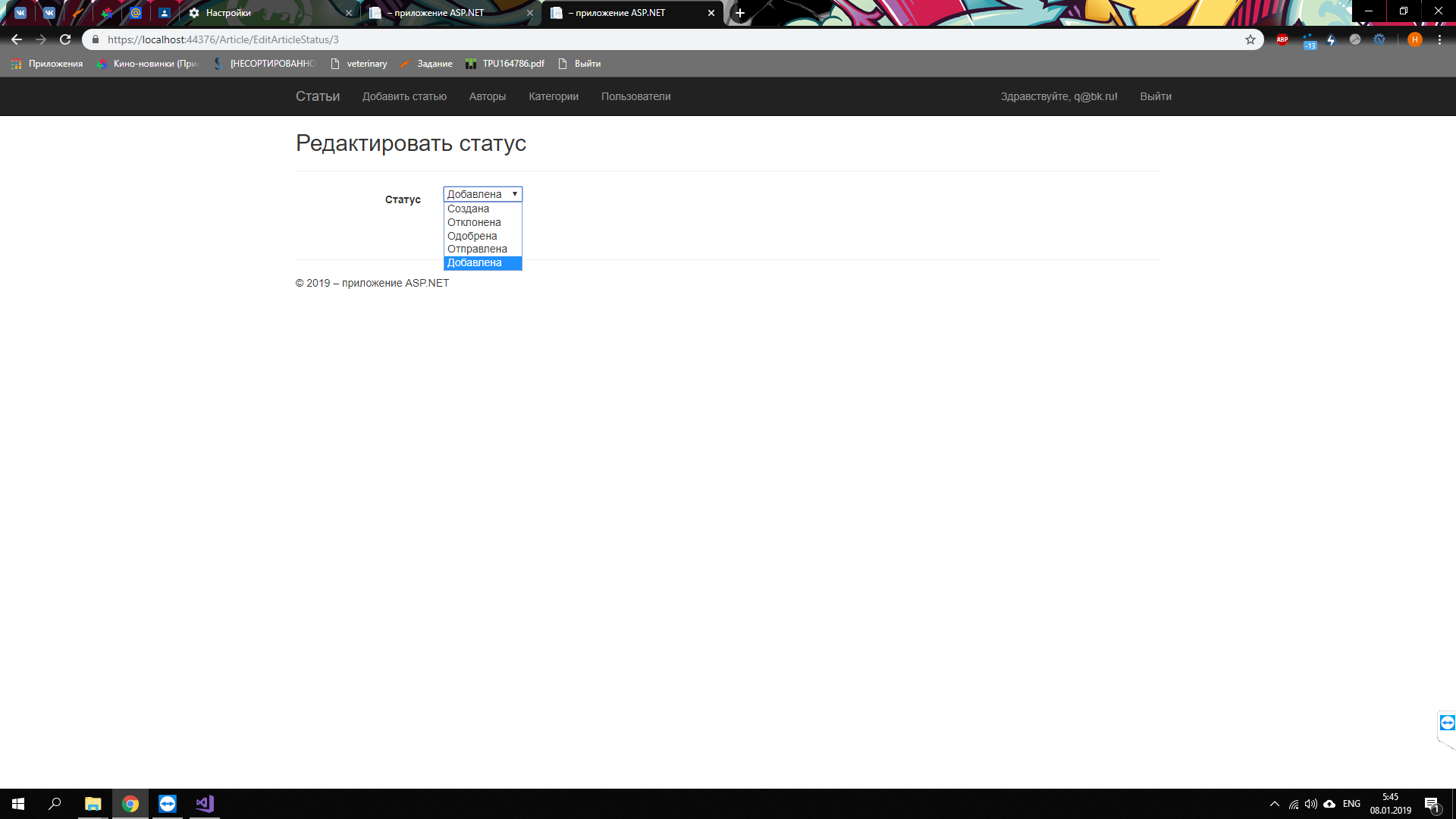


Рисунок 18. Редактирование статуса статьи.

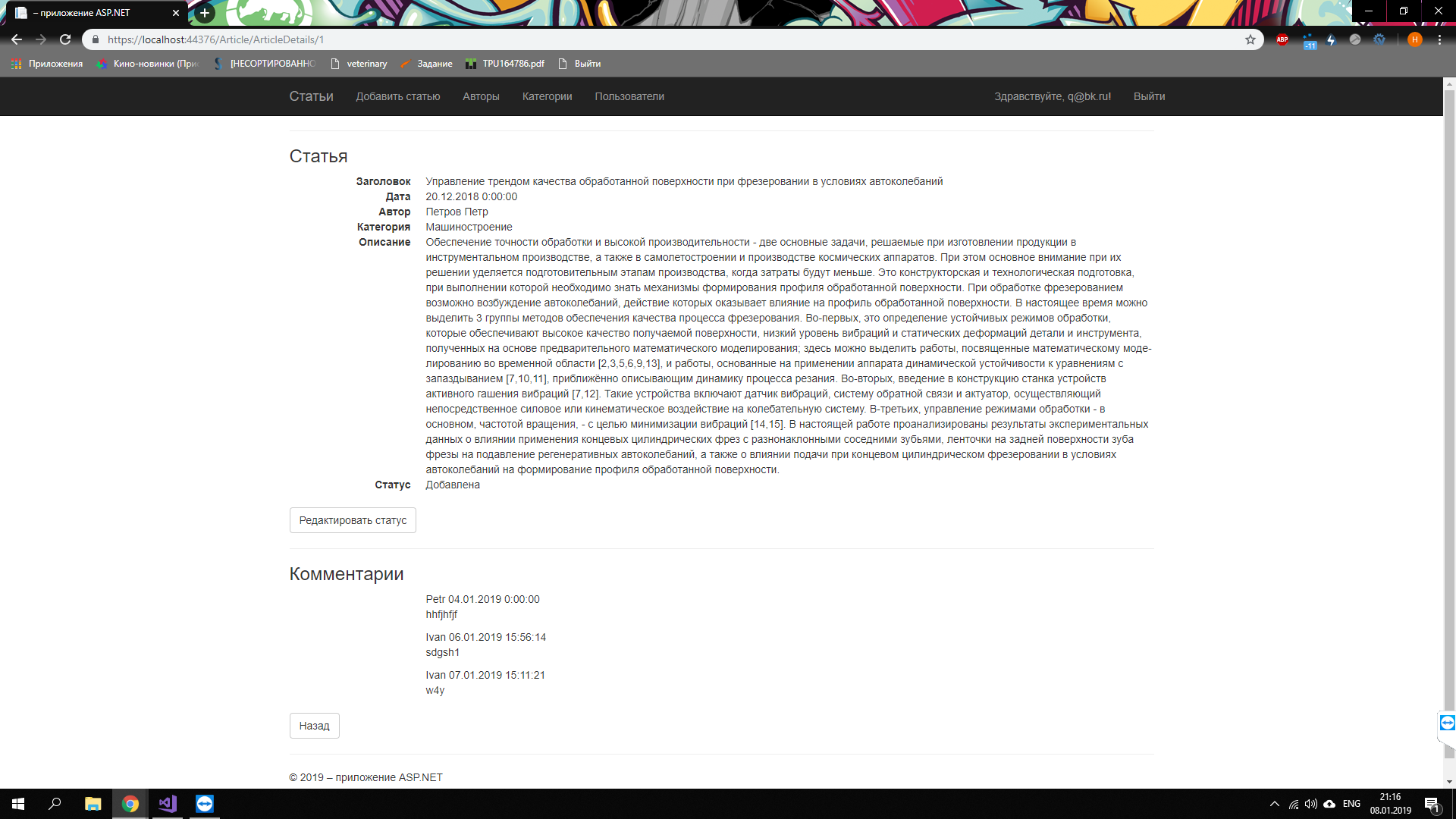


Рисунок 19. Статья глазами админа.

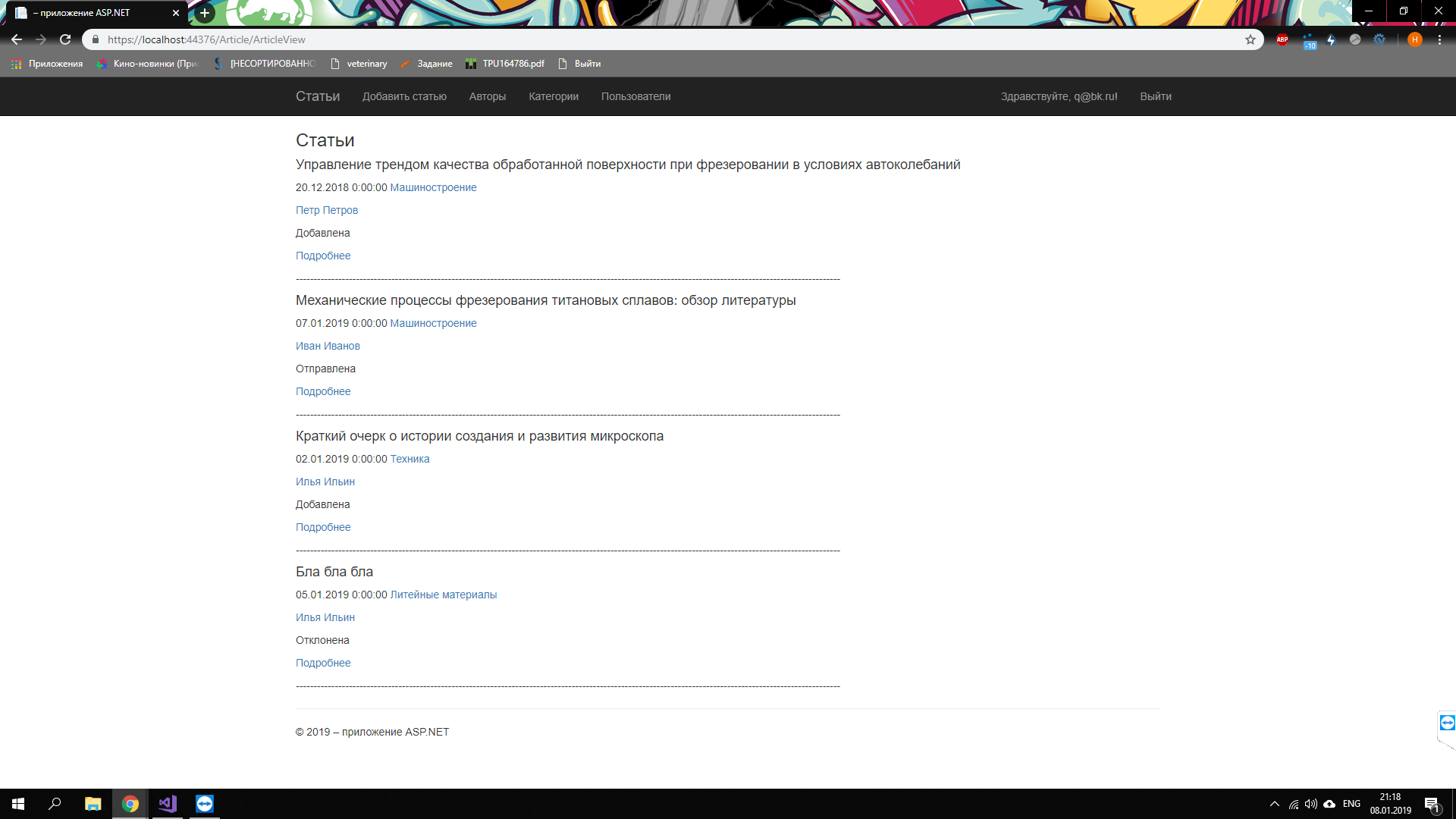


Рисунок 20. Статьи, доступные администратору.

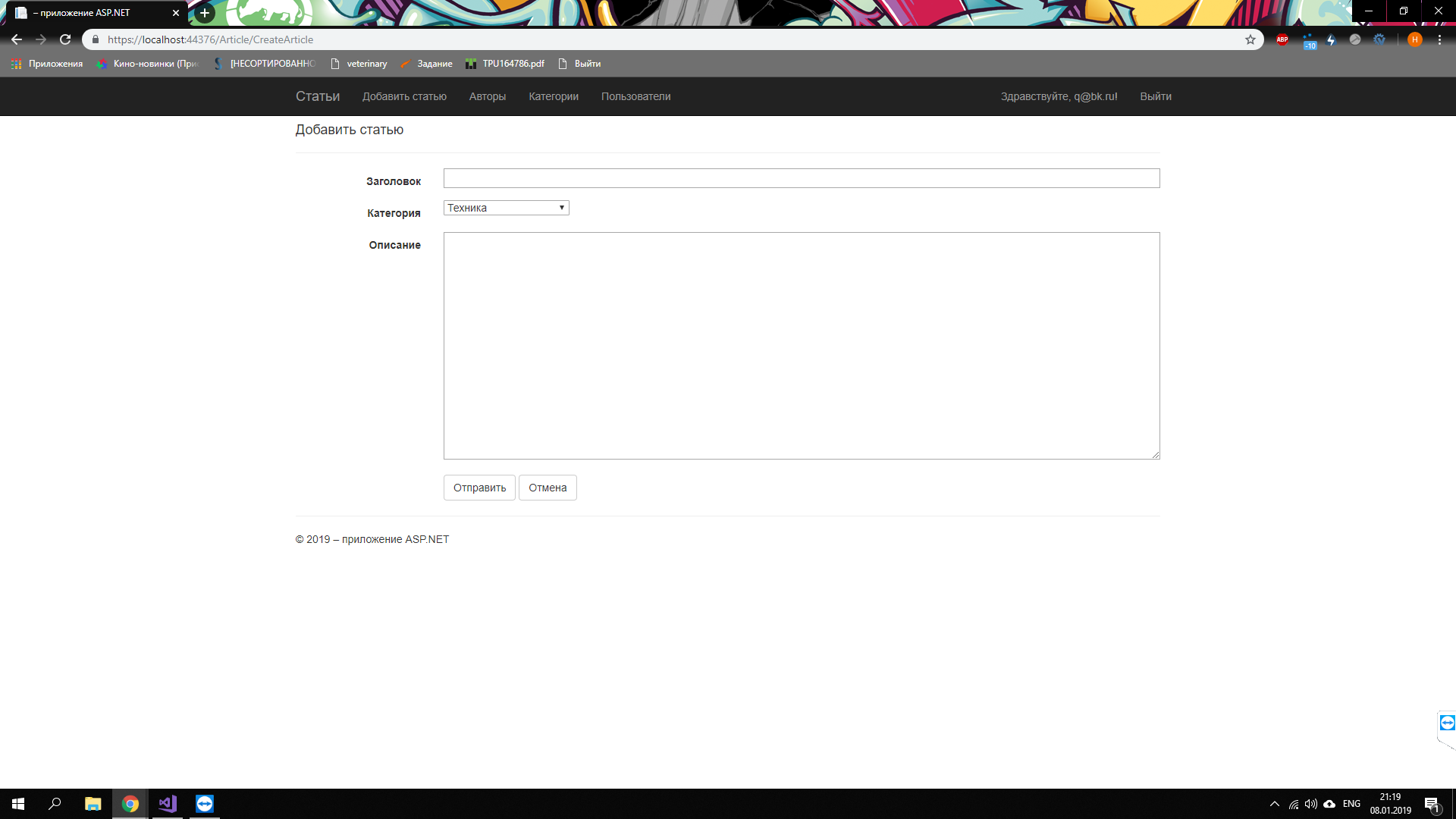


Рисунок 21. Добавление статьи администратором.

# 6 ЗАКЛЮЧЕНИЕ

В ходе работы была разработана программная система по информационной области «База знаний отраслей». Была использована система контроля версий. Для удобства работы с системой были созданы различные методы для взаимодействия разных пользователей с системой.

Разработанная база данных является актуальной на сегодняшний день и имеет практическую значимость.

# 7 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Липаев В.В. Проектирование программных систем. М.: Высш. шк, 1990.
2. Буч Г. Объективно-ориентированное проектирование / Пер. с анг. Канкорд, 1996.
3. Майерс Г. Надежность программного обеспечения. М.:Мир, 1980
4. Калянов Г.Н. CASE структурный системный анализ (автоматизация и применение.) – М.: «ЛОРИ», 1996.
5. Шмуллер Дж. Освой самостоятельно UML. 2-е изд. М.: изд. дом «Вильямс», 2002
6. Калберстон Р., Браун К., Кобб Г. Быстрое тестирование М.: изд. дом «Вильямс», 2002
7. Кармайкл Э., Хейвуд Д. Быстрая и качественная разработка программного обеспечения. М.: изд. дом «Вильямс», 2003
8. Руководство по ASP.NET MVC 5 [Электронный ресурс]: Metantit.com. – Режим доступа: http://metanit.com/sharp/mvc5/

# ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММНОЙ СИСТЕМЫ

Модель:

Класс ArticleViewModel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Bread.Models

{

public class ArticleViewModel

{

public IEnumerable<Article> Articles { get; set; }

public Article Article { get; set; }

public IEnumerable<Comment> Comments { get; set; }

public Comment Comment { get; set; }

public SubjectArea SubjectArea { get; set; }

public Author Author { get; set; }

public int Id { get; set; }

}

}

Класс Author

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Bread.Models

{

public class Author

{

public int Id { get; set; }

public string FName { get; set; }

public string SName { get; set; }

public string Nickname { get; set; }

public string ApplicationUserId { get; set; }

public virtual ApplicationUser ApplicationUser { get; set; }

public virtual ICollection<Comment> Comment { get; set; }

}

}

Класс Comment

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Bread.Models

{

public class Comment

{

public int Id { get; set; }

public int AuthorId { get; set; }

public System.DateTime Date { get; set; }

public string Content { get; set; }

public int ArticleId { get; set; }

public virtual Article Article { get; set; }

public virtual Author Author { get; set; }

}

}

Класс Logger

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using log4net;

using log4net.Config;

namespace Bread.Models

{

public class Logger

{

static Logger()

{

XmlConfigurator.Configure();

}

public static ILog For(object LoggedObject)

{

if (LoggedObject != null)

return For(LoggedObject.GetType());

return For(null);

}

public static ILog For(Type ObjectType)

{

if (ObjectType != null)

return LogManager.GetLogger(ObjectType.Name);

return LogManager.GetLogger(string.Empty);

}

}

}

Класс Status

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Bread.Models

{

public class Status

{

public int Id { get; set; }

public string Title { get; set; }

public virtual ICollection<Article> Article { get; set; }

}

}

Класс SubjectArea

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Bread.Models

{

public class SubjectArea

{

public int Id { get; set; }

public string Title { get; set; }

public virtual ICollection<Article> Article { get; set; }

}

}

Контроллеры:

Класс ArticleController

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Bread.Models;

using Microsoft.AspNet.Identity;

namespace Bread.Controllers

{

public class ArticleController : Controller

{

private static readonly ApplicationDbContext AppContext = new ApplicationDbContext();

// GET: Article

public ActionResult ArticleView()

{

if (User.IsInRole("Author"))

{

return View(AppContext.Article.Select(x => x).Where(s => s.StatusId == 5));

}

return View(AppContext.Article.Select(x => x));

}

public ActionResult ArticleDetails(int id)

{

var model = new ArticleViewModel

{

Article = AppContext.Article.First(s => s.Id == id),

Comments = AppContext.Comment.Select(s => s).Where(s => s.ArticleId == id)

};

return View(model);

}

[HttpGet]

[Authorize(Roles = "Admin, Manager")]

public ActionResult EditArticleStatus(int id)

{

var article = AppContext.Article.First(s => s.Id == id);

if (article != null)

{

var status = new SelectList(AppContext.Status.Select(s => s), "Id", "Title");

ViewBag.Status = status;

return View(article);

}

return RedirectToAction("ArticleView");

}

[HttpPost]

[Authorize(Roles = "Admin, Manager")]

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult EditArticleStatus(int id, Article article)

{

var status = new SelectList(AppContext.Status.Select(s => s), "Id", "Title");

ViewBag.Status = status;

try

{

if (id > 0 && article != null && ModelState.IsValid)

{

var art = AppContext.Article.FirstOrDefault(s => s.Id == id);

art.StatusId = article.StatusId;

AppContext.SaveChanges();

Logger.For(article).Info("Запись успешно изменена: " + article);

return RedirectToAction("ArticleView");

}

}

catch (Exception ex)

{

Logger.For(article).Error("Ошибка: ", ex);

throw;

}

return View("EditArticleStatus", AppContext.Article.FirstOrDefault(s => s.Id == id));

}

[HttpGet]

[Authorize(Roles = "Admin, Manager, Author")]

public ActionResult CreateArticle()

{

var subjectArea = new SelectList(AppContext.SubjectArea.Select(s => s), "Id", "Title");

ViewData["SubjectArea"] = subjectArea;

return View("CreateArticle");

}

[HttpPost]

[Authorize(Roles = "Admin, Manager, Author")]

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult CreateArticle(Article article)

{

var subjectArea = new SelectList(AppContext.SubjectArea.Select(s => s), "Id", "Title");

ViewData["SubjectArea"] = subjectArea;

if (User.IsInRole("Author"))

{

var id = User.Identity.GetUserId();

var author = AppContext.Author.FirstOrDefault(s => s.ApplicationUserId == id);

article.AuthorId = author.Id;

}

article.StatusId = 2;

article.Date = DateTime.Now;

try

{

if (AppContext.Article.Add(article) != null)

{

AppContext.SaveChanges();

Logger.For(article).Info("Запись успешно добавлена: " + article);

return RedirectToAction("ArticleView", "Article");

}

}

catch (Exception ex)

{

Logger.For(article).Error("Ошибка: ", ex);

throw;

}

return View("CreateArticle");

}

[HttpPost]

[Authorize(Roles = "Author")]

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult SaveArticle(Article article)

{

var subjectArea = new SelectList(AppContext.SubjectArea.Select(s => s), "Id", "Title");

ViewData["SubjectArea"] = subjectArea;

var id = User.Identity.GetUserId();

var author = AppContext.Author.FirstOrDefault(s => s.ApplicationUserId == id);

article.AuthorId = author.Id;

article.StatusId = 1;

article.Date = DateTime.Now;

try

{

if (AppContext.Article.Add(article) != null)

{

AppContext.SaveChanges();

Logger.For(article).Info("Запись успешно добавлена: " + article);

return RedirectToAction("ArticleView", "Article");

}

}

catch (Exception ex)

{

Logger.For(article).Error("Ошибка: ", ex);

throw;

}

return View("CreateArticle");

}

[Authorize(Roles = "Author")]

public ActionResult SentArticle(int id)

{

var article = AppContext.Article.First(s => s.Id == id);

article.StatusId = 4;

var art = AppContext.Article.FirstOrDefault(s => s.Id == id);

art.StatusId = article.StatusId;

AppContext.SaveChanges();

return RedirectToAction("Index", "Manage");

}

[HttpGet]

public ActionResult EditArticle(int id)

{

var subjectArea = new SelectList(AppContext.SubjectArea.Select(s => s), "Id", "Title");

ViewData["SubjectArea"] = subjectArea;

return View(AppContext.Article.FirstOrDefault(s => s.Id == id));

}

[HttpPost]

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult EditArticle(int id, Article article)

{

var subjectArea = new SelectList(AppContext.SubjectArea.Select(s => s), "Id", "Title");

ViewData["SubjectArea"] = subjectArea;

try

{

if (id > 0 && article != null && ModelState.IsValid)

{

article.StatusId = 4;

var art = AppContext.Article.FirstOrDefault(s => s.Id == id);

art.StatusId = article.StatusId;

art.Date = DateTime.Now;

art.Description = article.Description;

art.Title = article.Title;

art.SubjectAreaId = article.SubjectAreaId;

AppContext.SaveChanges();

Logger.For(article).Info("Запись успешно изменена: " + article);

return RedirectToAction("ArticleDetails", "Article", new { id = art.Id });

}

}

catch (Exception ex)

{

Logger.For(article).Error("Ошибка: ", ex);

throw;

}

return View("EditArticle", AppContext.Article.FirstOrDefault(s => s.Id == id));

}

public ActionResult ArticleSubjectArea(int id)

{

var model = new ArticleViewModel

{

Articles = AppContext.Article.Select(s => s).Where(s => s.SubjectAreaId == id),

SubjectArea = AppContext.SubjectArea.First(s => s.Id == id)

};

return View(model);

}

public ActionResult ArticleAuthor(int id)

{

var model = new ArticleViewModel

{

Articles = AppContext.Article.Select(s => s).Where(s => s.AuthorId == id),

Author = AppContext.Author.First(s => s.Id == id)

};

return View(model);

}

public ActionResult EditComment(int id)

{

return RedirectToAction("EditComment", "Comment", new { id });

}

}

}

Класс AuthorController

using System;

using System.Collections.Generic;

using System.Data.Entity.Migrations;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Bread.Models;

using Microsoft.AspNet.Identity;

namespace Bread.Controllers

{

public class AuthorController : Controller

{

private static readonly ApplicationDbContext AppContext = new ApplicationDbContext();

// GET: Author

public ActionResult AuthorView()

{

return View(AppContext.Author.Select(x => x));

}

public ActionResult ViewAuthors()

{

return View(AppContext.Author.Select(x => x));

}

[HttpGet]

[Authorize(Roles = "Author")]

public ActionResult CreateAuthor(string id)

{

var author = new Author { ApplicationUserId = id };

return View(author);

}

[HttpPost]

[Authorize(Roles = "Author")]

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult CreateAuthor(Author author)

{

var id = User.Identity.GetUserId();

author.ApplicationUserId = id;

try

{

if (AppContext.Author.Add(author) != null)

{

AppContext.SaveChanges();

Logger.For(author).Info("Запись успешно добавлена: " + author);

return RedirectToAction("Index", "Home");

}

}

catch (Exception ex)

{

Logger.For(author).Error("Ошибка: ", ex);

throw;

}

return View("CreateAuthor");

}

[HttpGet]

[Authorize(Roles = "Author")]

public ActionResult EditAuthor(int id)

{

var author = AppContext.Author.First(s => s.Id == id);

if (author != null)

{

return View(author);

}

return RedirectToAction("Index", "Manage");

}

[HttpPost]

[Authorize(Roles = "Author")]

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult EditAuthor(int id, Author author)

{

try

{

if (id > 0 && author != null && ModelState.IsValid)

{

var auth = AppContext.Author.First(s => s.Id == id);

auth.FName = author.FName;

auth.SName = author.SName;

auth.Nickname = author.Nickname;

AppContext.SaveChanges();

Logger.For(author).Info("Запись успешно изменена: " + author);

return RedirectToAction("Index", "Manage");

}

}

catch (Exception ex)

{

Logger.For(author).Error("Ошибка: ", ex);

throw;

}

return View("EditAuthor", AppContext.Author.FirstOrDefault(s => s.Id == id));

}

public ActionResult ArticleAuthor(int id)

{

return RedirectToAction("ArticleAuthor", "Article", new { id });

}

}

}

Класс CommentController

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Bread.Models;

using Microsoft.AspNet.Identity;

namespace Bread.Controllers

{

public class CommentController : Controller

{

private static readonly ApplicationDbContext AppContext = new ApplicationDbContext();

// GET: Comment

public ActionResult ViewComment()

{

return View(AppContext.Comment.Select(s => s));

}

public ActionResult ArticleAuthor(int id)

{

return RedirectToAction("ArticleAuthor", "Article", new { id });

}

public ActionResult ArticleDetails(int id)

{

return RedirectToAction("ArticleDetails", "Article", new { id });

}

[HttpGet]

[Authorize(Roles = "Author")]

public ActionResult CreateComment(int id)

{

var model = new ArticleViewModel

{

Id = id

};

return View(model);

}

[HttpPost]

[Authorize(Roles = "Author")]

public ActionResult CreateComment(ArticleViewModel model)

{

var uId = User.Identity.GetUserId();

var author = AppContext.Author.FirstOrDefault(s => s.ApplicationUserId == uId);

model.Comment.AuthorId = author.Id;

model.Comment.Date = DateTime.Now;

model.Comment.ArticleId = model.Id;

try

{

if (AppContext.Comment.Add(model.Comment) != null)

{

AppContext.SaveChanges();

Logger.For(author).Info("Запись успешно добавлена: " + author);

return RedirectToAction("ArticleDetails", "Article", new { id = model.Comment.ArticleId });

}

}

catch (Exception ex)

{

Logger.For(model).Error("Ошибка: ", ex);

throw;

}

return View("CreateComment");

}

[HttpGet]

[Authorize(Roles = "Admin, Manager, Author")]

public ActionResult EditComment(int id)

{

return View(AppContext.Comment.FirstOrDefault(s => s.Id == id));

}

[HttpPost]

[Authorize(Roles = "Admin, Manager, Author")]

public ActionResult EditComment(int id, Comment comment)

{

try

{

if (id > 0 && comment != null && ModelState.IsValid)

{

var comm = AppContext.Comment.FirstOrDefault(s => s.Id == id);

comm.Content = comment.Content;

AppContext.SaveChanges();

Logger.For(comment).Info("Запись успешно изменена: " + comment);

return RedirectToAction("ArticleDetails", "Article", new { id = comm.ArticleId });

}

}

catch (Exception ex)

{

Logger.For(comment).Error("Ошибка: ", ex);

throw;

}

return View("EditComment", AppContext.Comment.FirstOrDefault(s => s.Id == id));

}

public ActionResult DeleteComment()

{

return View();

}

}

}

Класс SubjectAreaController

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Bread.Models;

namespace Bread.Controllers

{

public class SubjectAreaController : Controller

{

private static readonly ApplicationDbContext AppContext = new ApplicationDbContext();

// GET: SubjectArea

public ActionResult SubjectAreaView()

{

return View(AppContext.SubjectArea.Select(s => s));

}

[HttpGet]

[Authorize(Roles = "Admin, Manager")]

public ActionResult CreateSubjectArea()

{

return View("CreateSubjectArea");

}

[HttpPost]

[Authorize(Roles = "Admin, Manager")]

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult CreateSubjectArea(SubjectArea subjectArea)

{

try

{

if (AppContext.SubjectArea.Add(subjectArea) != null)

{

AppContext.SaveChanges();

Logger.For(subjectArea).Info("Запись успешно добавлена: " + subjectArea);

return RedirectToAction("SubjectAreaView", "SubjectArea");

}

}

catch (Exception ex)

{

Logger.For(subjectArea).Error("Ошибка: ", ex);

throw;

}

return View("CreateSubjectArea");

}

public ActionResult ArticleSubjectArea(int id)

{

return RedirectToAction("ArticleSubjectArea", "Article", new {id});

}

}

}

Файл конфигурации:

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

<section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler, log4net" />

</configSections>

<log4net>

<appender name="LogFileAppender" type="log4net.Appender.RollingFileAppender">

<param name="File" value="Log.log" />

<appendToFile value="true" />

<rollingStyle value="Size" />

<maximumFileSize value="10MB" />

<staticLogFileName value="true" />

<layout type="log4net.Layout.PatternLayout">

<param name="ConversionPattern" value="%d [%t] %-5p %c %m%n" />

</layout>

</appender>

<root>

<level value="ALL" />

<appender-ref ref="LogFileAppender" />

</root>

</log4net>

<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\aspnet-Bread-20181105095727.mdf;Initial Catalog=aspnet-Bread-20181105095727;Integrated Security=True" providerName="System.Data.SqlClient" />

</connectionStrings>

<appSettings>

<add key="webpages:Version" value="3.0.0.0" />

<add key="webpages:Enabled" value="false" />

<add key="ClientValidationEnabled" value="true" />

<add key="UnobtrusiveJavaScriptEnabled" value="true" />

</appSettings>

<system.web>

<authentication mode="None" />

<compilation debug="true" targetFramework="4.6.1" />

<httpRuntime targetFramework="4.6.1" />

<httpModules>

<add name="ApplicationInsightsWebTracking" type="Microsoft.ApplicationInsights.Web.ApplicationInsightsHttpModule, Microsoft.AI.Web" />

</httpModules>

</system.web>

<system.webServer>

<modules>

<remove name="FormsAuthentication" />

<remove name="TelemetryCorrelationHttpModule" />

<add name="TelemetryCorrelationHttpModule" type="Microsoft.AspNet.TelemetryCorrelation.TelemetryCorrelationHttpModule, Microsoft.AspNet.TelemetryCorrelation" preCondition="integratedMode,managedHandler" />

<remove name="ApplicationInsightsWebTracking" />

<add name="ApplicationInsightsWebTracking" type="Microsoft.ApplicationInsights.Web.ApplicationInsightsHttpModule, Microsoft.AI.Web" preCondition="managedHandler" />

</modules>

<validation validateIntegratedModeConfiguration="false" />

</system.webServer>

<runtime>

<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

<dependentAssembly>

<assemblyIdentity name="Microsoft.Owin.Security" publicKeyToken="31bf3856ad364e35" />

<bindingRedirect oldVersion="0.0.0.0-4.0.0.0" newVersion="4.0.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="Microsoft.Owin.Security.OAuth" publicKeyToken="31bf3856ad364e35" />

<bindingRedirect oldVersion="0.0.0.0-4.0.0.0" newVersion="4.0.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="Microsoft.Owin.Security.Cookies" publicKeyToken="31bf3856ad364e35" />

<bindingRedirect oldVersion="0.0.0.0-4.0.0.0" newVersion="4.0.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="Microsoft.Owin" publicKeyToken="31bf3856ad364e35" />

<bindingRedirect oldVersion="0.0.0.0-4.0.0.0" newVersion="4.0.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="Antlr3.Runtime" publicKeyToken="eb42632606e9261f" />

<bindingRedirect oldVersion="0.0.0.0-3.5.0.2" newVersion="3.5.0.2" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Diagnostics.DiagnosticSource" publicKeyToken="cc7b13ffcd2ddd51" />

<bindingRedirect oldVersion="0.0.0.0-4.0.2.1" newVersion="4.0.2.1" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Web.Optimization" publicKeyToken="31bf3856ad364e35" />

<bindingRedirect oldVersion="0.0.0.0-1.1.0.0" newVersion="1.1.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />

<bindingRedirect oldVersion="0.0.0.0-1.6.5135.21930" newVersion="1.6.5135.21930" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" />

<bindingRedirect oldVersion="0.0.0.0-11.0.0.0" newVersion="11.0.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />

<bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />

<bindingRedirect oldVersion="0.0.0.0-5.2.4.0" newVersion="5.2.4.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />

<bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />

</dependentAssembly>

</assemblyBinding>

</runtime>

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />

<providers>

<provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />

</providers>

</entityFramework>

<system.codedom>

<compilers>

<compiler language="c#;cs;csharp" extension=".cs" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4" compilerOptions="/langversion:default /nowarn:1659;1699;1701" />

<compiler language="vb;vbs;visualbasic;vbscript" extension=".vb" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4" compilerOptions="/langversion:default /nowarn:41008 /define:\_MYTYPE=\&quot;Web\&quot; /optionInfer+" />

</compilers>

</system.codedom>

</configuration>