МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Кафедра 319 «Системы интеллектуального мониторинга»

КУРСОВАЯ РАБОТА

по дисциплине «Технология разработки программного обеспечения»

«Проектирование и разработка веб-приложения классификации новостных статей с помощью методов машинного обучения»

Студент _____ Новиков Р.А.

Группа МЗО – 221М – 19

Руководитель _____ Полицына Е. В.

Оценка _____ Дата защиты «____» февраля 2021 г.

Москва 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Кафедра 319 «Системы интеллектуального мониторинга»

З А Д А Н И Е

на курсовую работу по дисциплине
Технология разработки программного обеспечения

Студент МЗО – 221М – 19 Новиков Роман Андреевич
(№ группы, Ф. И. О.)

Тема Проектирование и разработка веб-приложения классификации новостей с помощью методов машинного обучения

Перечень вопросов, подлежащих разработке в курсовой работе:

1. Проектирование архитектуры системы и выбор средств разработки
2. Проектирование и реализация ядра системы – классификатора
3. Проектирование и реализация фронтенда
4. Проектирование и реализация бекенда

Рекомендуемая литература

1. Вьюгин В.В. «Математические основы теории машинного обучения и прогнозирования» М.: 2013. – 387 с.
2. Django [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Django - Заглавие с экрана. – (Дата обращения 20.01.21).

Задание выдано « 12 » сентября 2020 г.
Руководитель Полицына Екатерина Валерьевна, к.т.н., доцент кафедры 319 МАИ
(Ф. И. О., должность, подпись)
Студент Новиков Р.А.
(подпись)

СОДЕРЖАНИЕ

1 ОПИСАНИЕ ВОЗМОЖНОСТЕЙ ПРИЛОЖЕНИЯ

1.1 Общие сведения

Веб-приложение классификации новостных статей представляет веб-сервис, направленный на обработку и классификацию новостных статей из новостного портала «www.finam.ru».

1.2 Назначение и цель

С помощью веб-сервиса возможно определить, к какой предопределенной в системе категории относится новостная статья, введенная пользователем (алгоритм классификации основан на методе машинного обучения). Также сервис предоставляет возможность просмотреть новостные статьи, которые представляли собой входные данные для обучения классификатора (обучающую выборку) с возможностью их фильтрации по предопределенным категориям.

Функции веб-сервиса:

- Классификация пользовательских новостных статей по предопределенным категориям;
- вывод новостных статей, на которых был обучен классификатор;
- удаление новостной статьи.
-

2 ТРЕБОВАНИЯ К ПРИЛОЖЕНИЮ

2.1 Функциональные требования

Функциональные требования к системе:

- Добавление текста – система должна позволять пользователю добавить текст, который требуется классифицировать;
- удаление текста – система должна позволять пользователю удалять текст, который требуется классифицировать;
- классификация текста – система должна обработать введенный пользователем текст и вывести его предполагаемую категорию;
- просмотр новостных статей – система должна позволять пользователю просматривать коллекцию новостных статей, на которых был обучен классификатор;
- фильтрация новостных статей по категориям – система должна позволять просматривать новостные статьи по конкретным категориям;
- просмотр общего количества статей – система должна выводить информацию о общем количестве имеющихся в системе новостных статей;
- просмотр отображаемого количества статей – система должна выводить информацию о количестве статей, которые есть в конкретной категории.

2.2 Нефункциональные требования

Система должна быть реализована на базе клиент-серверной архитектуры и представлять из себя веб-сервис для классификации с графическим пользовательским интерфейсом.

Средства разработки:

- backend должен быть разработан на языке программирования Python с помощью фреймворка Django;
- СУБД SQlite;
- frontend должен быть реализован с помощью HTML и языка программирования Javascript;

- проект и все необходимые документация и данные по нему должны храниться в системе контроля версий на сайте сервиса GitHub.

Система должна быть отказоустойчивой при работе пользователя с системой.

3 АРХИТЕКТУРА СИСТЕМЫ

3.1 Схема архитектуры

Архитектура разработанного приложения изображена на рисунке 1.

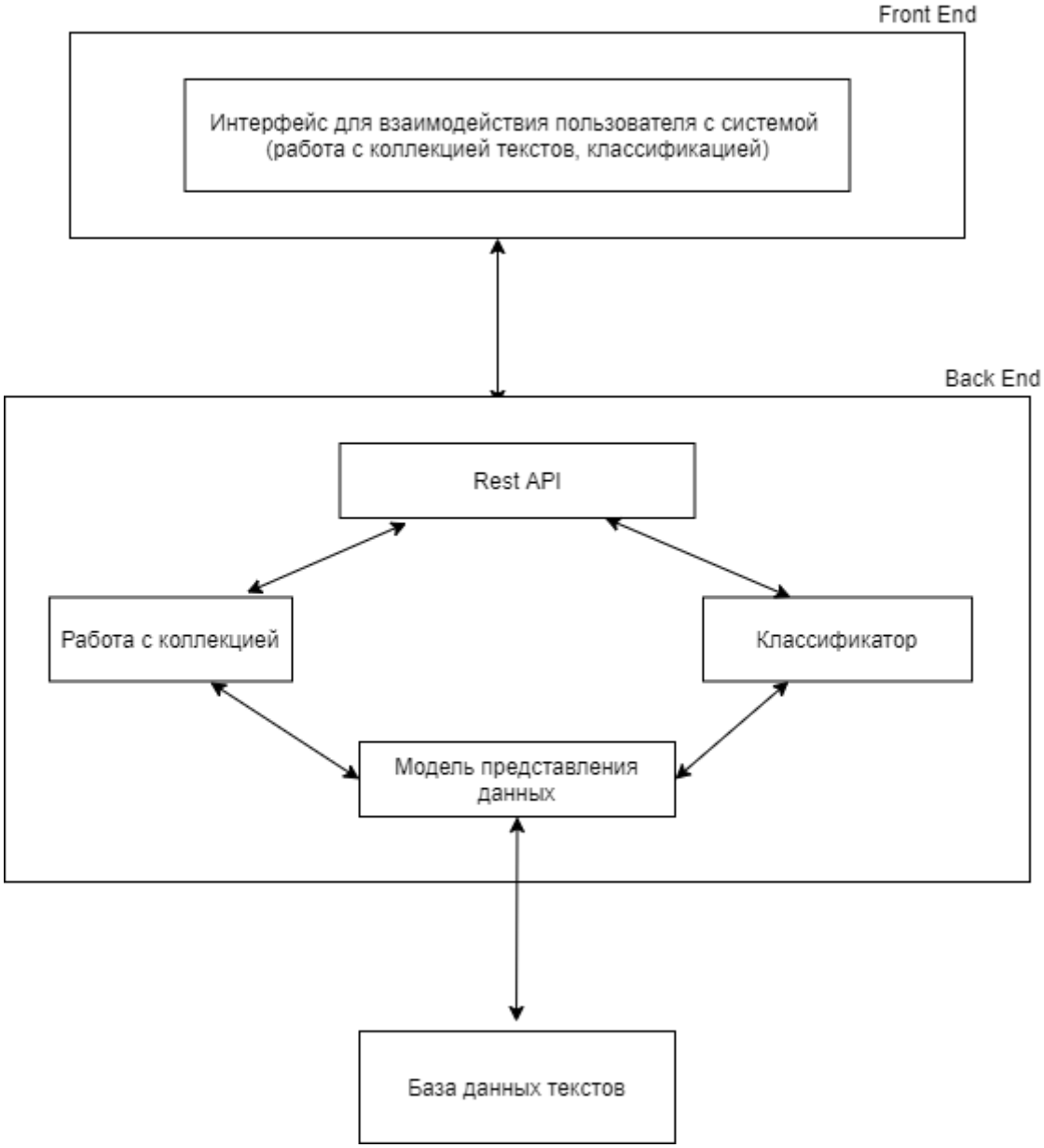

Рисунок 1 – Архитектура приложения

Блок Backend описывает серверную часть системы:

- Контроллер – часть системы, обозначающая совокупность контроллеров – компонентов, интерпретирующих действия пользователя, которые приходят по HTTP протоколу с фронтенда;
- Сервис классификации – классификатор;

- сервис по статьям – сервис для обработки статей.

Блок Frontend описывает клиентскую часть системы:

- Интерфейс для взаимодействия с пользователем – пользовательский интерфейс.

3.2 Протоколы взаимодействия

Разработанная система является RESTful веб-сервисом. Используемые в системе API:

- Добавление новостной статьи:

POST api/articles

- возвращение страницы списка новостных статей и общее количество новостных статей:

POST api/articles/all

- возвращение страницы списка новостных статей, отфильтрованных по категориям и общее количество отфильтрованных новостных статей по категориям:

POST api/articles/filter-by-category

- возвращение новостной статье по идентификатору:

GET api/articles/(?P<id>[0-9]+)

- удаление новостной статьи:

DEETE api/articles/(?P<id>[0-9]+)

- возвращение страницы для классификации новостных статей:

GET /

- возвращение html страницы для фильтрации коллекции новостных статей:

GET /articles-collection

- возвращение html страницы для ввода новостной статьи

GET /add-article

- возвращение html страницы с полной информацией о статье

GET /articles/[0-9]+

- выполнение классификации новостной статьи и возвращение категории:

POST api/classify

- обучение модели для классификации новостных статей:

POST api/model-learn

3.3 Используемые технологии

Для реализации серверной части использовался язык программирования Python с помощью фреймворка Django - фреймворка, обеспечивающего комплексную модель разработки и конфигурации для современных бизнес-приложений на любых платформах.

В качестве СУБД использовалось SQlite.

Для создания обученной модели классификатора использовался модуль языка Python scikit-learn – Scikit-learn построена поверх SciPy (Scientific Python), который должен быть установлен перед использованием scikit-learn. Данный стек включает в себя: NumPy: расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц, вместе с

большой библиотекой высокоуровневых математических функций для операций с этими массивами.

Для поддержки контроля версий использовалась использовался сервис GitHub, представляющий из себя графический интерфейс для технологии git – распределенной системы управления версиями.

3.4 Паттерны проектирования

При разработке использовался паттерн MVC (Model – View – Controller) – способ разделения данных системы, пользовательского интерфейса и управляющей логики на три отдельных компонента, таким образом, чтобы модификация каждого компонента могла осуществляться независимо.

4 ОПИСАНИЕ КЛАССИФИКАТОРА

4.1 Классы

Классификатор разрешает текст по следующим категориям:

- Общество;
- финансы и рынки;
- политика;
- экономика;

4.2 Вектор признаков

В качестве модели представления текста использовалась модель bag-of-words. Вектор признаков – это количество вхождений того или иного слова среди всех слов из всех новостей. При этом из общего перечня слов убирались предлоги, союзы и частицы.

4.3 Обучающая и тестовая коллекция документов

В качестве обучающей коллекции были взяты 210 новостных статей по каждой из категории из общей коллекции текстов. Таким образом, классификатор обучался на 840 новостных статьях.

В качестве тестовой коллекции были взяты 360 новостных статей по каждой из категории из общей коллекции текстов. Таким образом классификатор тестировался на 360 новостных статьях.

4.4 Оценка точности классификации

Классификатор обучался при помощи классификатора дерева решений, который обеспечивает наилучшую классификацию моделей представления вида bag-of-words.

Оценка точности разработанного классификатора оценивалась по трем критериям: точности, полноте и F-мере.

Точность – это доля статей, действительно относящихся к данной категории относительно всех статей, которые классификатор отнес к этой категории.

Полнота – это доля найденных классификатором статей, принадлежащих классу, относительно всех статей этой категории в тестовой выборке.

F-мера – это гармоническое среднее между точностью и полнотой. Она стремится к 0, если точность и полнота стремятся к нулю.

Все необходимые статистические данные, а также точность классификации, которая составляет 78%, были автоматически рассчитаны модулем языка Python scikit-learn и данные представлены в таблице 1.

Таблица 1 – Оценка точности разработанного классификатора

Категория	**Точность**	**Полнота**	**F-мера**
Общество	0.64	0.57	0.60
Политика	0.75	0.71	0.73
Финансы и рынки	0.97	1.00	0.98
Экономика	0.74	0.83	0.78

5 ОПИСАНИЕ ИНФРАСТРУКТУРЫ РАЗРАБОТКИ

5.1 Система контроля версий

Проектирование и реализация приложения велась локально на компьютере разработчика при помощи Microsoft Visual Studio Code. При разработке системы использовалась система контроля версий git. Исходный код хранится локально на компьютере разработчика, и удаленно, на сайте сервиса GitHub:

https://github.com/rain-krast/rain_krast

5.2 Сборка и развертывание приложения

Для успешного запуска веб-приложения необходимо наличие интерпретатора Python 3.0 (64 бит) или выше, а также операционной системы Windows 7 и выше. Необходимо скачать архив с проектом, разархивировать папку FinamCollectionService. Далее через командную строку активировать виртуальное окружение python с помощь команд:

1) cd {pathTo}\
   FinamCollectionService;
2) env\Scripts\activate.bat

После активации окружения, достаточно запустить web-приложение по данной команде: manage.py runserver.

5.3 Обновление модели классификатора

Для обновления модели классификатора, достаточно послать POST запрос по ссылке /api/FinamCollectionService/fit.

5.4 Обработка ошибок

При возникновении ошибок выводит в интерфейсе приложения информацию о произошедших ошибках.

6 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Для тестирования системы применялось ручное тестирование.

Тестирование проводилось компьютере разработчика со следующей конфигурацией:

- процессор: AMD A8-6410 APU with AMD Radeon R5 Graphics 2.00 GHz;
- оперативная память: 8ГБ;
- операционная система: 64-разрядная.

Для выполнения тестирования на компьютере разработчика развертывалось приложение. В процессе тестирования использовался веб-браузер Google Chrome версии 87.0.4280.141. При тестировании использовался метод «чёрный ящик».

Тестирование чёрного ящика или поведенческое тестирование — стратегия (метод) тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве (коде) тестируемого объекта. Иначе говоря, тестированием чёрного ящика занимаются тестировщики, не имеющие доступ к исходному коду приложения. Под стратегией понимаются систематические методы отбора и создания тестов для тестового набора. Стратегия поведенческого теста исходит из технических требований и их спецификаций.

7 ИНТЕРФЕЙС И ВОЗМОЖНОСТИ СИСТЕМЫ

Система предоставляет пользователю классифицировать текст и просматривать новостные статьи, на которых был обучен классификатор.

Классификация текста происходит во вкладке «Классификация статей». Также присутствует кнопка «переобучить модель классификатора», которая отвечает за переобучение модели классификатора заново. Интерфейс страницы классификации текстов представлен на рисунке 2 .

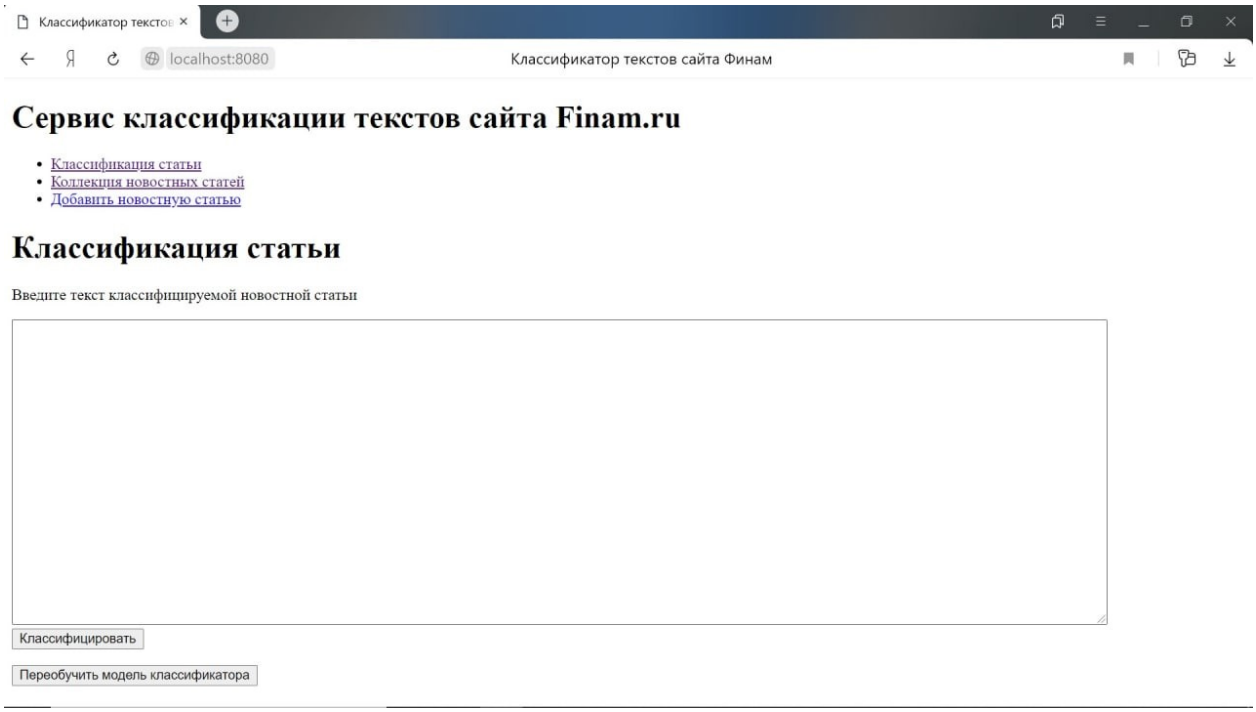

Рисунок 2 – Интерфейс вкладки классификации текста

Для фильтрации коллекции текстов необходимо задать желаемые параметры и нажать кнопку «Найти». Если нажать на значок «Крестик», то можно удалить статью. Интерфейс для работы с коллекцией новостных статей представлен на рисунке 3.

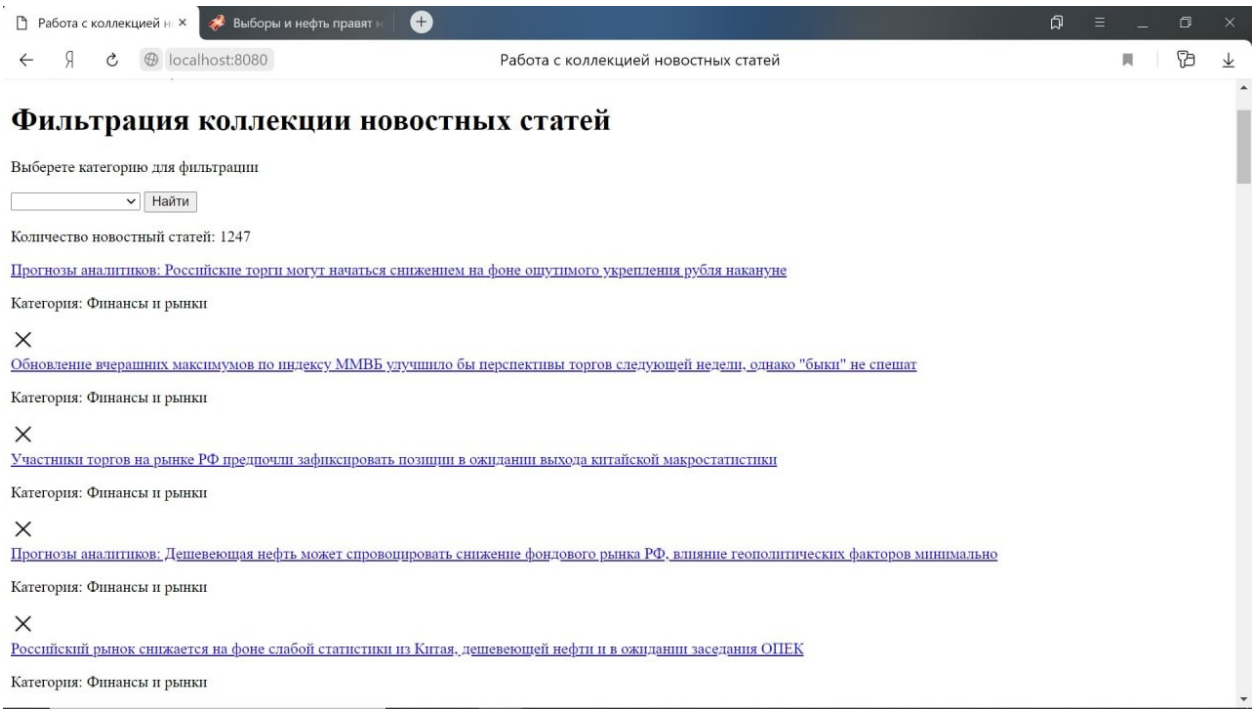

Рисунок 3 – Интерфейс работы с коллекцией новостных статей.

Для добавления новой статьи необходимо ввести данные о ней в форму, затем нажать на кнопку «Добавить статью». Интерфейс для добавления новой статьи представлен на рисунке 4.

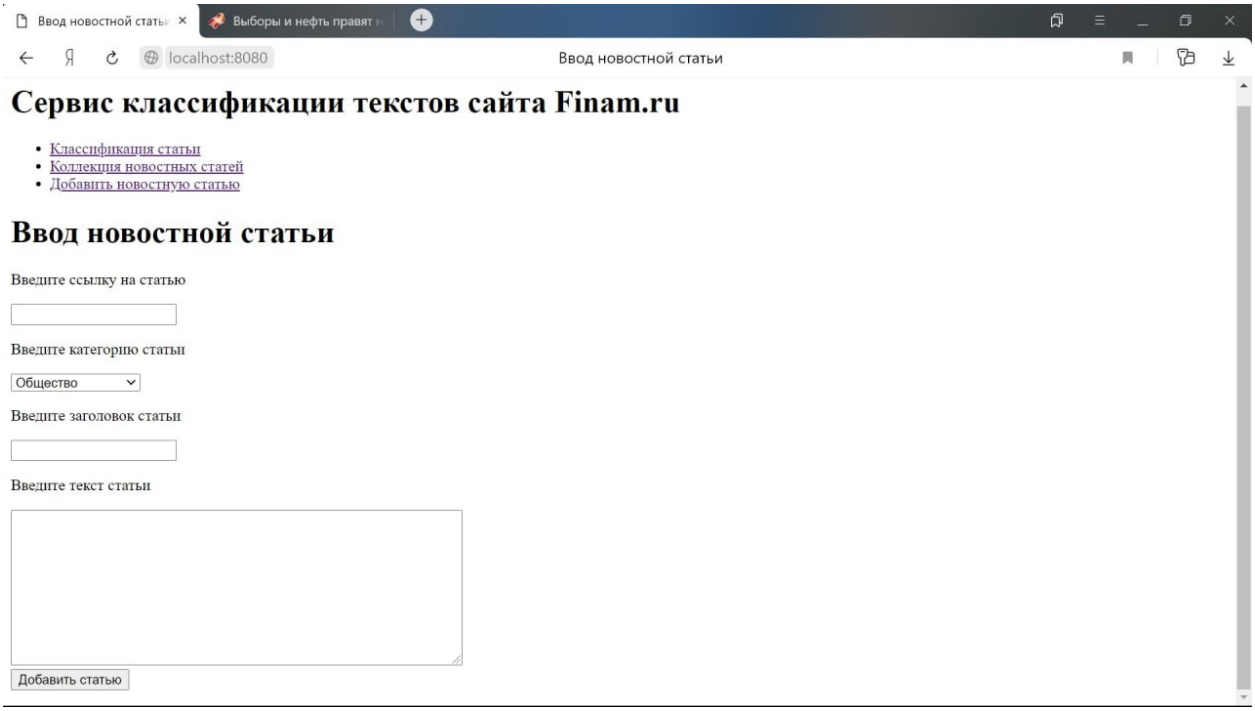

Рисунок 4 – Интерфейс добавления новой статьи.

8 АНАЛИЗ РЕЗУЛЬТАТОВ

В процессе работы были выполнены следующие задачи:

1. Спроектирована архитектура системы, выбраны средства разработки;
2. Спроектировано и реализовано ядро системы – классификатор: определены категории классификации, исходя из предметной области, выбрана и подготовлена модель представления вектора признаков, подготовлены обучающая и тестовые выборки, обучена модель классификации с помощью классификатора дерева решений;
3. Спроектирован и реализован фронтенд;
4. Спроектирован и реализован бэкенд.

Разработанная система позволяет пользователям загружать и классифицировать новостные статьи, а также просматривать новостные статьи, на которых был обучен классификатор с выводом объема коллекции статей.

Разработанное веб-приложение не требует чрезмерных затрат в вопросах поддержки, сопровождения и обновления. При повышении нагрузки на сервера с развернутым приложением достаточно выделить больше оперативной памяти и ядер.

Архитектура приложения разработана таким образом, что добавление новых контроллеров и сервисов не зависит и не влияет на работу других частей системы. Этого удалось добиться благодаря принципу разработки многослойных систем, где каждый слой представляет из себя отдельный независимый логический компонент.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. www.finam.ru [Электронный ресурс]. – Режим доступа: https://www.finam.ru. – Заглавие с экрана. – (Дата обращения: 20.01.21).

2. Python [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Python – Заглавие с экрана. – (Дата обращения: 20.01.21).

3. Django [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Django. – Заглавие с экрана. – (Дата обращения: 20.01.21).

4. GitHub [Электронный ресурс]. – Режим доступа: https://github.com. – Заглавие с экрана. – (Дата обращения: 20.01.21).

5. Microsoft Visual Studio Code [Электронный ресурс]. – Режим доступа: https://code.visualstudio.com – Заглавие с экрана. – (Дата обращения: 20.01.21).