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 новостей с применением методов машинного обучения»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | | |  | Закупин Д.Е. |
| Группа | | | М3О – 222М – 19 |  |
| Руководитель | | |  | Полицына Е.В. |
| Оценка |  | Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_ 20 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 М3О – 222М – 19 Закупин Дмитрий Евгеньевич

(№ группы, Ф. И. О.)

Тема Проектирование и разработка веб-приложения классификации новостей с применением 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. Django [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/. – Заглавие с экрана. – (Дата обращения 25.12.20).
2. Scikit-learn [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/. – Заглавие с экрана. – (Дата обращения 25.12.20).

Задание выдано « 12 » сентября 2020 г.

Руководитель Полицына Екатерина Валерьевна, к.т.н., доцент кафедры 319 МАИ

(Ф. И. О., должность, подпись)

Студент Закупин Д.Е.

(подпись)

**СОДЕРЖАНИЕ**

[1 ОПИСАНИЕ ВОЗМОЖНОСТЕЙ ПРИЛОЖЕНИЯ 4](#_Toc65283885)

[1.1 Общие сведения 4](#_Toc65283886)

[1.2 Назначение и цель 4](#_Toc65283887)

[2 ТРЕБОВАНИЯ К ПРИЛОЖЕНИЮ 5](#_Toc65283888)

[2.1 Функциональные требования 5](#_Toc65283889)

[2.2 Нефункциональные требования 5](#_Toc65283890)

[3 АРХИТЕКТУРА СИСТЕМЫ 6](#_Toc65283891)

[3.1 Схема архитектуры 6](#_Toc65283892)

[3.2 Протоколы взаимодействия 7](#_Toc65283893)

[3.3 Используемые технологии 7](#_Toc65283894)

[3.4 Паттерны проектирования 8](#_Toc65283895)

[4 ОПИСАНИЕ КЛАССИФИКАТОРА 9](#_Toc65283896)

[4.1 Классы 9](#_Toc65283897)

[4.2 Вектор признаков 9](#_Toc65283898)

[4.3 Модель машинного обучения классификатора 9](#_Toc65283899)

[4.4 Обучающая и тестовая коллекция документов 9](#_Toc65283900)

[4.5 Оценка точности классификации 10](#_Toc65283901)

[5 ОПИСАНИЕ ИНФРАСТРУКТУРЫ РАЗРАБОТКИ 11](#_Toc65283902)

[6 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 12](#_Toc65283903)

[7 ИНТЕРФЕЙС И ВОЗМОЖНОСТИ СИСТЕМЫ 13](#_Toc65283904)

[8 АНАЛИЗ РЕЗУЛЬТАТОВ 16](#_Toc65283905)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc65283906)

# ОПИСАНИЕ ВОЗМОЖНОСТЕЙ ПРИЛОЖЕНИЯ

## Общие сведения

Веб-приложение классификации новостных статей – это веб-приложение которое позволяет классифицировать статью и работать с коллекцией статей с новостного сайта «meduza.io»

## Назначение и цель

Веб-приложение дает возможность изменять коллекцию статей, а именно добавлять новые статьи, редактировать уже имеющиеся, удалять и отображать. Причем в отображении статей имеется фильтрация по автору и категории, а также поиск по заголовкам статей. Так же имеется возможность определения категории с помощью технологий машинного обучения.

# ТРЕБОВАНИЯ К ПРИЛОЖЕНИЮ

## Функциональные требования

Функциональные требования к системе:

* добавление новостной статьи;
* удаление новостной статьи;
* изменение новостной статьи;
* вывод новостных статей;
* фильтрация новостных статей по категориям и автору;
* поиск по заголовкам новостных статей;
* классификация текста по категории новостной статьи с помощью методов машинного обучения;

## Нефункциональные требования

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

Веб-приложение должно быть разработано с помощью следующих технических средств:

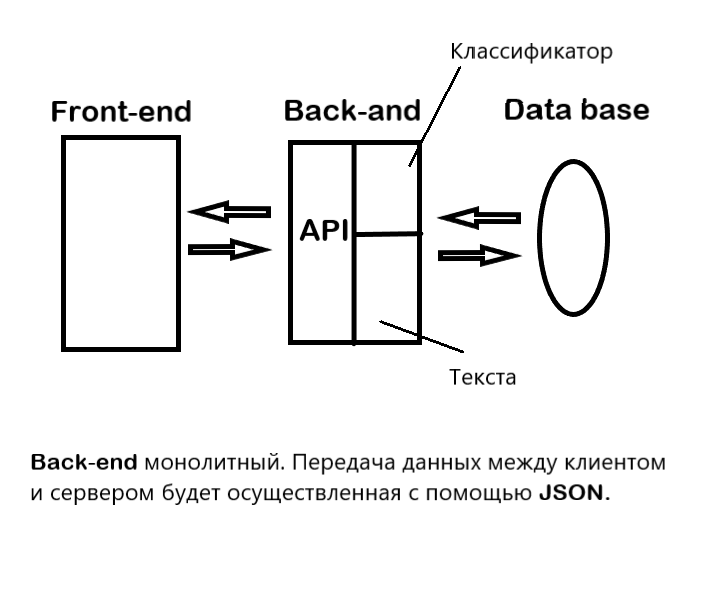
* backend должен быть реализован на языке программирования Python с помощью фреймворка Django;
* СУБД SQlite;
* frontend должен быть реализован с помощью javascript фреемвока Ract;
* проект и вся необходимая документация и данные по нему должны храниться в системе контроля версий git на сайте сервиса GitHub.

Система должна быть отказоустойчивой при работе пользователя с системой.

# АРХИТЕКТУРА СИСТЕМЫ

## Схема архитектуры

Архитектура разработанного приложения изображена на рисунке 1.

Рисунок 1 – Архитектура приложения

Веб-приложение состоит из Frontend и Backend частей. Frontend представляет из себя SPA приложение написанное на javascript фреемворке React. Backend – приложение написанное на фреемворке Django языка python. Состоит из 2 независимых частей: модуль для работы с коллекцией текстов и модуль классификатора. Так же Backend взаимодействует с базой данных SQLite.

Связь между Frontend и Backend осуществляется посредством http запросов.

## Протоколы взаимодействия

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

* Получение списка статей

GET /api/articles

* Добавление статьи

PUT /api/articles

* Редактирование статьи

POST /api/articles

* Удаление статьи

DELETE /api/articles/?id={id}

* Получение списка авторов

GET /api/articles/get-author

* Классификация текста

POST api/classifier

## Используемые технологии

Для реализации серверной части использовался фреймворк Django языка Python.

В качестве СУБД используется SQlite.

Для создания модели классификатора используется модуль языка Python scikit-learn.

В качестве средства реализации пользовательского интерфейса используется фреймворк языка javascript React.

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

## Паттерны проектирования

При разработке использовался паттерн MVC, который позволяет разрабатывать бизнес-логику и визуальное представление отдельно.

# ОПИСАНИЕ КЛАССИФИКАТОРА

## Классы

Классификатор определяет следующие классы:

* Истории
* Новости
* Шапито
* Разбор

## Вектор признаков

В качестве модели представления текста новостных статей использовался метод bag-of-words. При этом проводилась начальная подготовка текста: удалялись лишние неинформативные символы, а также различные стоп-слова. Вектором признаков является масштабированное отношение количества вхождений слова среди всех новостей на количество слов в тексте по методу tf-idf.

## Модель машинного обучения классификатора

В качестве модели машинного обучения был выбран метод Перцептрон.

## Обучающая и тестовая коллекция документов

Для обучения модели классификатора использовалось по 800 статей каждой категории.

В качестве тестовой выборки было взято по 200 статей каждой категории.

## Оценка точности классификации

Для оценки классификатора были выбраны следующие метрики: точность, полнота, F-мера.

Точность классификатора равна 82%. Оценки точности по категориям представлены в таблице 1.

Таблица 1 – Оценка точности обученного классификатора по категориям

|  |  |  |  |
| --- | --- | --- | --- |
| Категория | Точность (precision) | Полнота (recall) | F-мера |
| Истории | 0.75 | 0.84 | 0.79 |
| Новости | 0.89 | 0.78 | 0.83 |
| Разбор | 0.8 | 0.93 | 0.86 |
| Шапито | 0.87 | 0.75 | 0.8 |

# ОПИСАНИЕ ИНФРАСТРУКТУРЫ РАЗРАБОТКИ

Для разработки использовался текстовый редактор Visual Studio Code.

В качестве системы контроля версий использовался Github.

Для развертывания приложения необходим Python версии 3.9.1, NodeJS 12.6.1. Так же для работы серверной части необходимо установить библиотеки: django-rest-fraemwork, scikit-learn. При наличии данных библиотек достаточно в корне проекта запустить команду *py manage.py runserver.* Для запуска клиентской части необходимо выполнить команду *npm install* для установки всех необходимых библиотек, а затем команду *npm start.*

Для переобучения модели необходимо выполнить функцию trainingModel в файле classifierTraine.

# РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Для тестирования системы был выбран метод черного ящика.

Тестирование производилось в браузере Google Chrome. Основные тесты и результаты по ним представлены в таблице 2.

Таблица 2 – Проведенные тест

|  |  |
| --- | --- |
| Сценарий тестирования | Результат тестирования |
| Ввод текста для классификации и нажатие на кнопку Классифицировать | Был выведен результат классификации |
| Нажатии на кнопку Классифицировать с пустым полем для ввода | Было выведено сообщение «Введите текст» |
| Ввод данных в форму фильтрации и нажатие на кнопку найти | Были выведены статьи в соответствии с заданными параметрами |
| Ввод некорректного запроса в поле Поиск | Было выведено сообщение «Статьи не найдены» |
| Нажатие на кнопку Удалить на статье | Страница со статьями обновилась, статья была удалена |
| Нажатие на кнопку Отправить с пустыми полями в форме добавления статьи | Выведено сообщение что поля обязательны для заполнения |
| Первоначальная загрузка страницы | Отображается весь список статей |
| Поиск по запросу, в котором количество статей меньше 10 | Пагинатор не отображается |

# ИНТЕРФЕЙС И ВОЗМОЖНОСТИ СИСТЕМЫ

Веб-приложение позволяет пользователю классифицировать статью, а также производить CRUD операции над коллекцией имеющихся статей. Для этого было реализовано два модуля, переключение между которыми производится в шапке приложения.

Классификация текста происходит в разделе классификатор. Для того что бы классифицировать статью необходимо добавить текст в поле ввода и нажать кнопку Классифицировать. После чего под формой для классификации появится название класса статьи. В случае, если пользователь после этого захочет добавить статью в коллекцию, необходимо нажать на кнопку Добавить. В этом случае откроется окно, в котором уже будет подставлен текст статьи и класс.

Интерфейс раздела Классификатор представлен на рисунке 2.

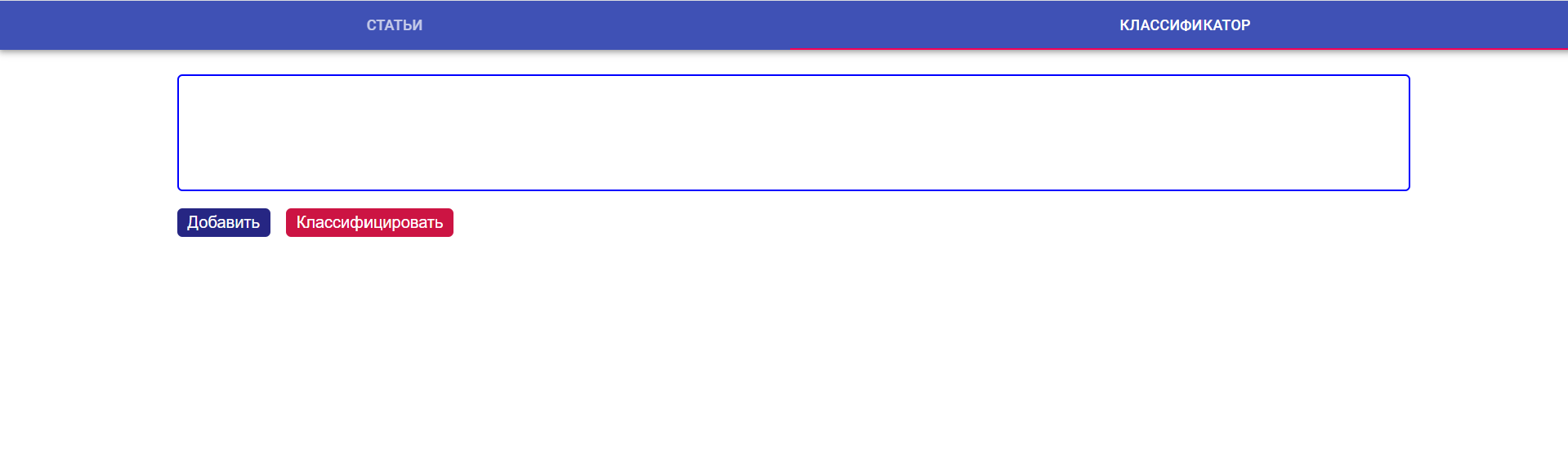


Рисунок 2 – Интерфейс классификации новостной статьи

Для фильтрации текстов необходимо задать параметры и нажать кнопку найти. Так же в этой форме присутствует кнопка Добавить для добавления новой статьи в коллекцию. Интерфейс для фильтрации статей представлен на рисунке 3.

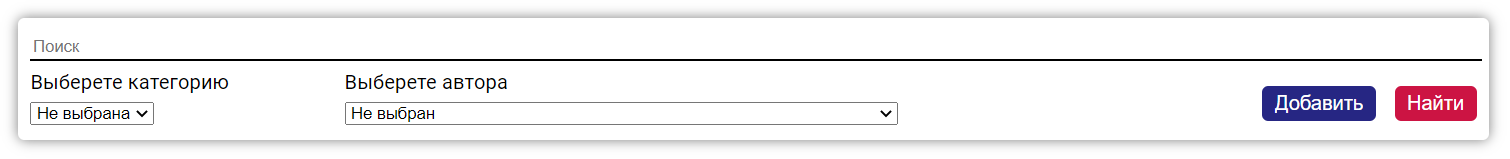


Рисунок 3 – Интерфейс для фильтрации статей

Статьи отображаются списком. В каждой статье пользователь может развернуть текст целиком, а также удалить или редактировать статью.

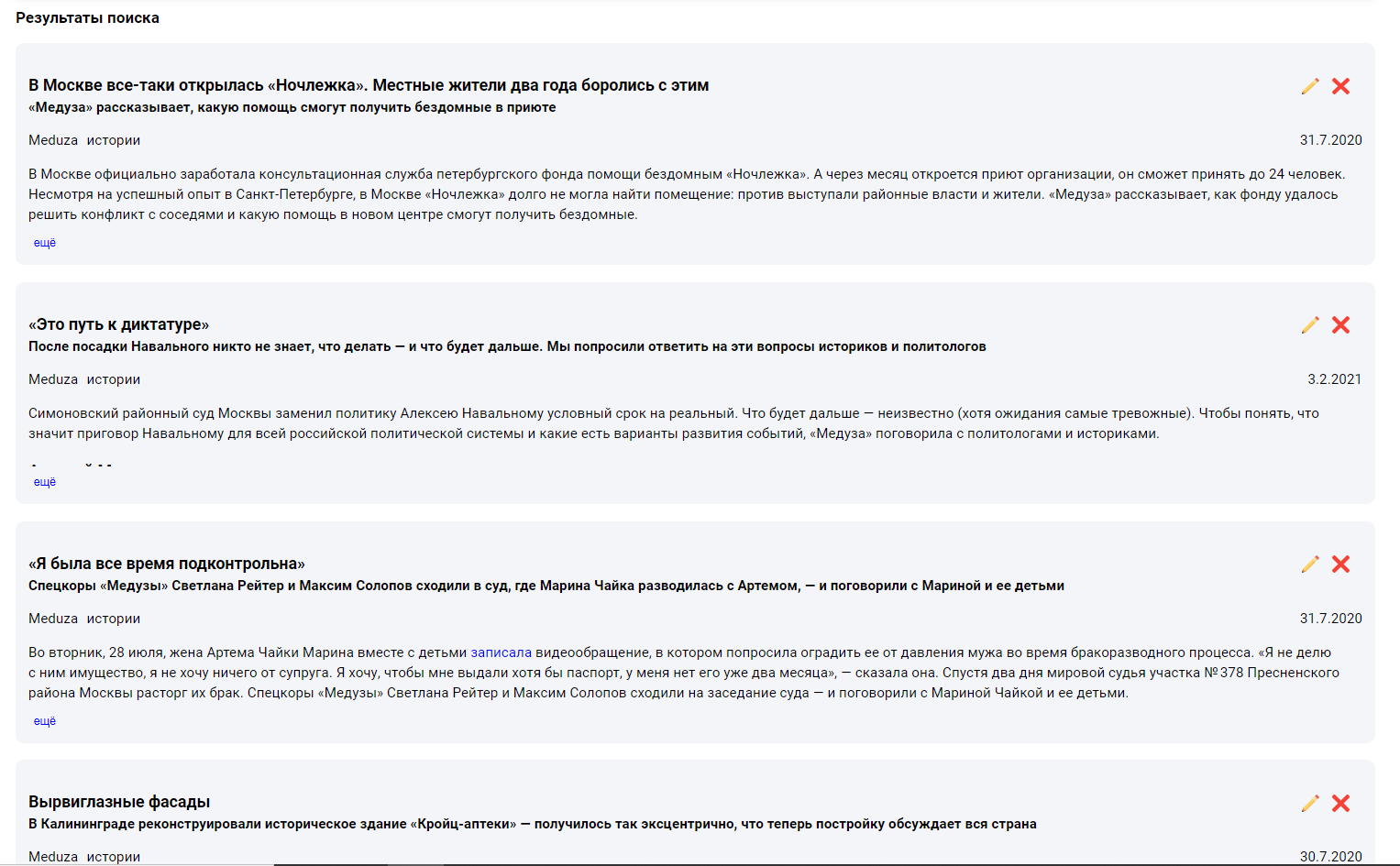


Рисунок 4 – Интерфейс для отображения коллекции новостей

Для удобства просмотра статей список был разделен на страницы. По 10 статей на каждой странице. Интерфейс для переключения между страницами представлен на рисунке 5.



Рисунок 5 – Интерфейс для переключения страниц

Для редактирования или добавления статьи необходимо ввести или изменить данные в форме и нажать на кнопку отправить. Интерфейс для редактирования или добавления статьи представлен на рисунке 6.

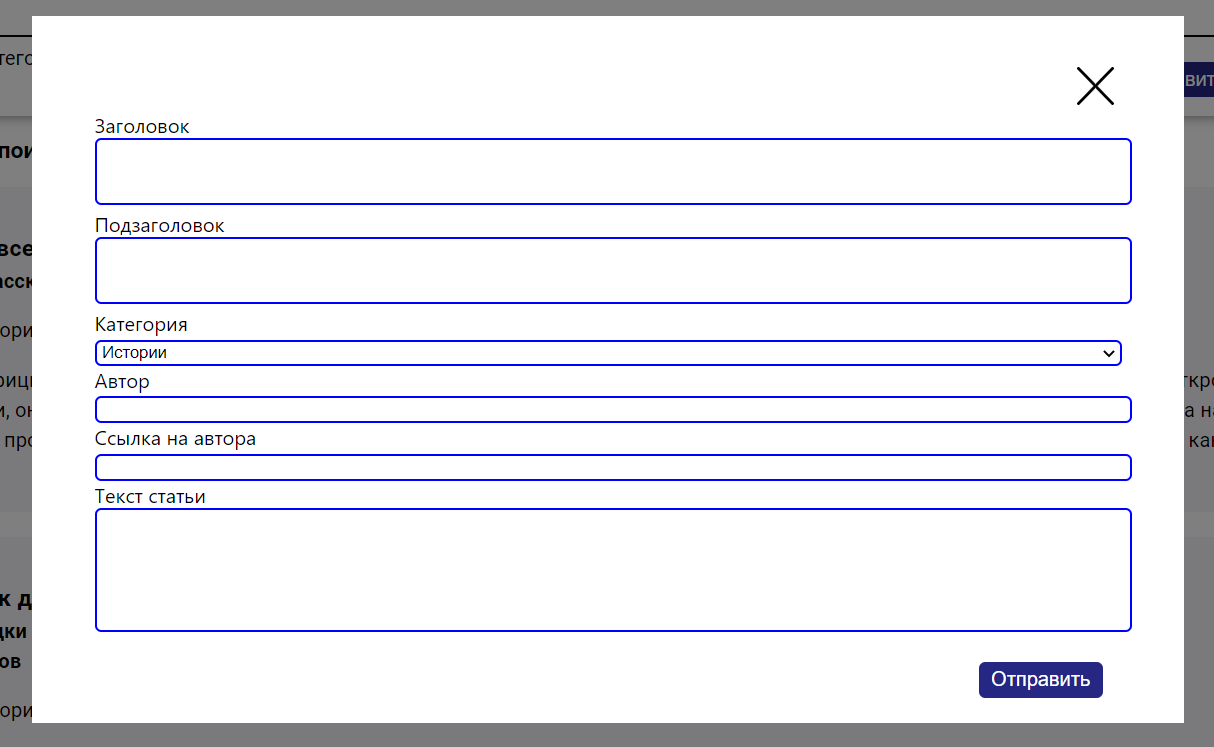


Рисунок 6 – Интерфейс для добавления/редактирования статьи

# АНАЛИЗ РЕЗУЛЬТАТОВ

В ходе выполнения курсовой работы было реализовано веб-приложение, которое позволяет классифицировать текста с новостного портала meduza.io, а также выполнять CRUD операции над уже имеющимися статьями.

В силу своей архитектуры приложение легко масштабируется. Для этого необходимо создать новое Django приложение на стороне сервера, а также добавить новые страницы на стороне клиента.

Переобучение модели возможно только при условии доступа к коду на сервере.

Классификатор показал неплохой результат, а именно 82%.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Meduza [Электронный ресурс]. – Режим доступа: https://www.meduza.io/. – Заглавие с экрана. – (Дата обращения: 20.02.21).
2. Python [Электронный ресурс]. – Режим доступа: https://www.python.org/. – Заглавие с экрана. – (Дата обращения: 20.02.21).
3. Django [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/. – Заглавие с экрана. – (Дата обращения 20.02.21).
4. React [Электронный ресурс] – Режим доступа: <https://reactjs.org/> - Заглавие с экрана. – (Дата обращения 20.02.21)
5. Scikit-learn [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/. – Заглавие с экрана. – (Дата обращения 20.02.21).
6. Современный учебник javascript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/. – Заглавие с экрана. – (Дата обращения 20.02.21).