**Форма для добавления записей в базу данных**

В данной лабораторной работе мы рассмотрим создание шаблона для ввода данных, а также и сам процесс ввода и сохранения введённых данных в формах из этого шаблона в саму базу данных в уже созданной в предыдущих лабораторных модель.

Создаём в базовом шаблоне (*base.html*) кнопку “Добавить запись” в “Панели навигации”, при нажатии на которой будет совершаться переход на шаблон *create.html* для ввода данных.

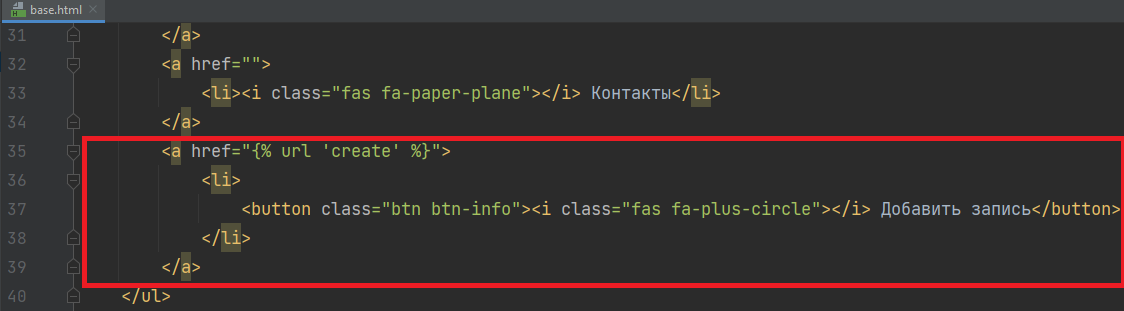


Рисунок 1 - Добавление кнопки “Добавить запись”

После этого нам необходимо создать URL-адрес *‘create’* для файла представления (*views.py*) непосредственно в Django-приложении *news*, а именнов файле *urls.py*.

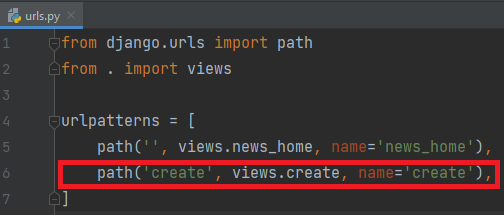


Рисунок 2 - URL-адрес views “create”

Создадим сам шаблон в папке *templates/news*.

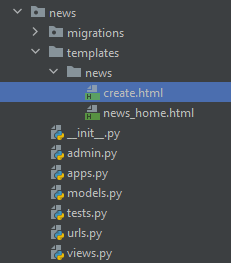


Рисунок 3 - Create.html в файловой структуре

Далее в файле представления *views.py* приложения *news* создадим функцию вывода необходимого нам шаблона.

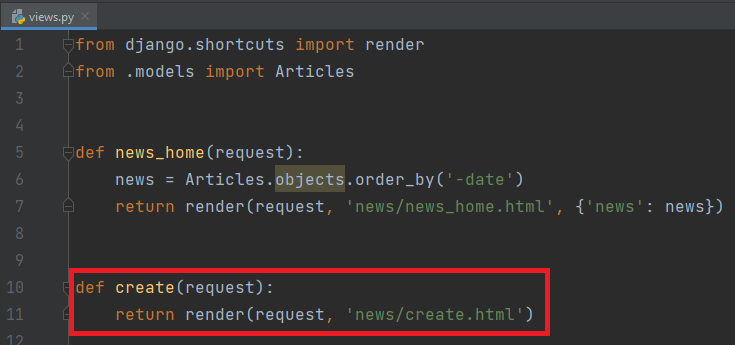


Рисунок 4 - Ссылка на шаблон news/create

После этого переходим в сам шаблон *create.html*. В нём мы должны реализовать поля (формы) для ввода данных. Для этого нам нужен метод запроса *POST*, который предназначен для направления запроса, при котором веб-сервер принимает данные, заключённые в тело сообщения, для хранения. Внутри формы мы пока что делаем заготовки форм, которые не связаны с полями модели базы данных. Для этого мы выбираем тег *<input>,* <*textarea*> и атрибут <*type*>, в которых уже задаём описание для будущих данных. В данном случае у нас 4 типа данных (вспоминаем свойства полей созданной модели).

Для форм ввода с небольшим текстом нам нужен тип HTML-атрибута *“text”*, для большого текста нужен тег <*textarea*>, для полей даты и времени - атрибуты “*date*” и “*time*” соответственно.

После списка из форм создадим кнопку, при нажатии на которой происходит подтверждение и сохранение введённых данных. Для этого мы для кнопки указываем тип “*submit*”.

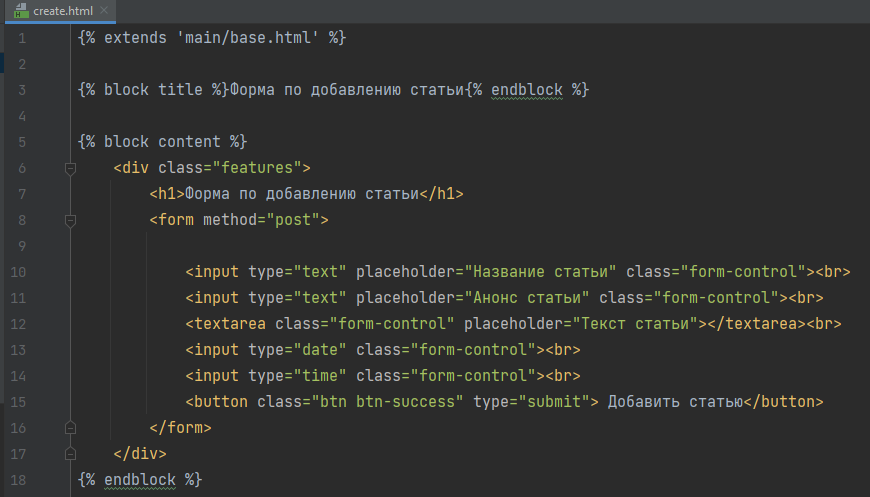


Рисунок 5 - Создание формы для добавления записей

Проверяем наш результат работы на запущенном локальном сервере.

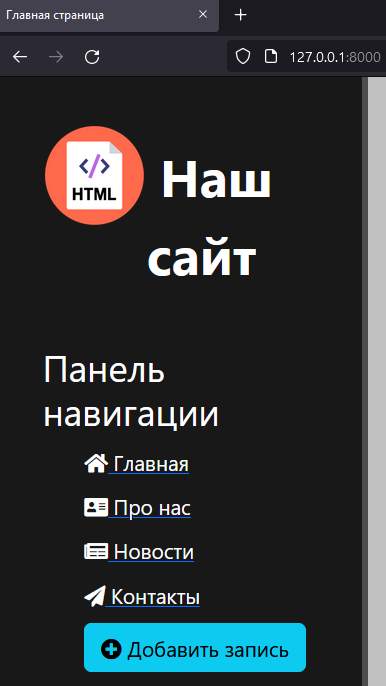


Рисунок 6 - Кнопка “Добавить запись” в панеле навигации

После нажатия на кнопку “Добавить запись” нас направляет на URL-адрес созданного шаблона *create.html*.

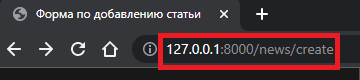


Рисунок 7 - Адрес страницы добавления записей

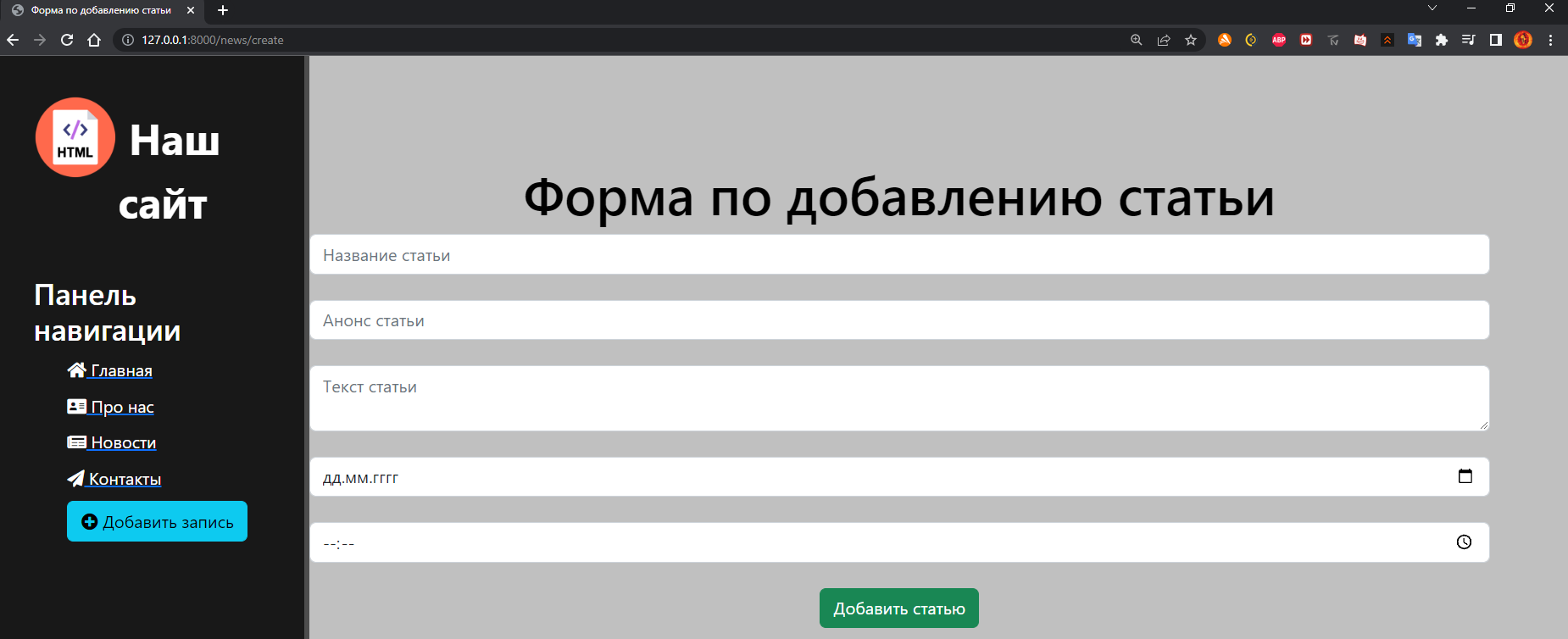


Рисунок 8 - Форма по добавлению статьи

Примечание: на некоторых браузерах могут не отображаться значки для выбора даты и времени в полях.

Зададим CSS-стиль для форм ввода данных в файле *main.css*.

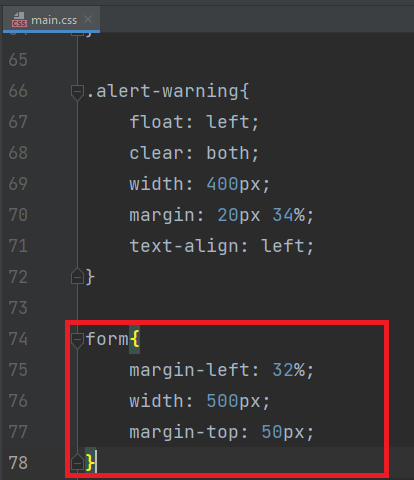


Рисунок 9 - Параметры форм

Обновим страницу для проверки.

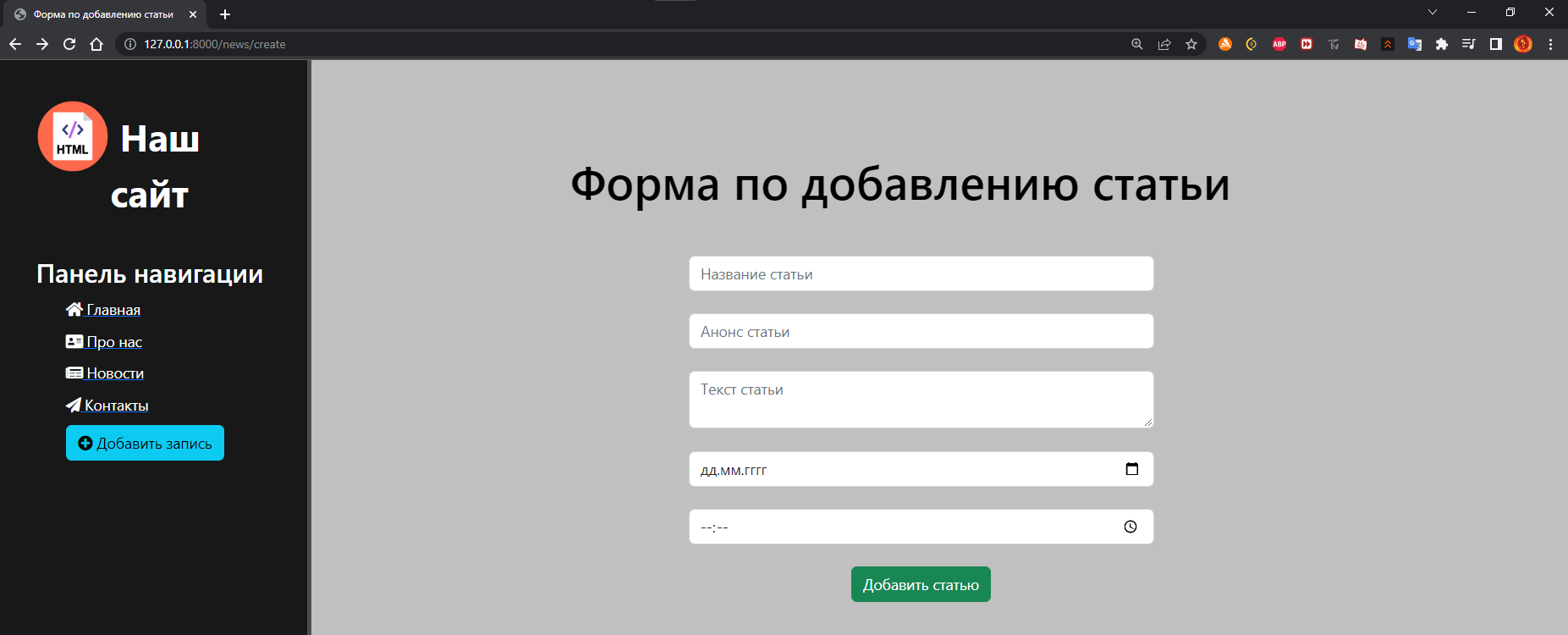


Рисунок 10 - Обновлённая форма по добавлению статьи

Если мы попытаемся нажать на *кнопку “Добавить статью”*, то у нас произойдёт *ошибка доступа (403)*.

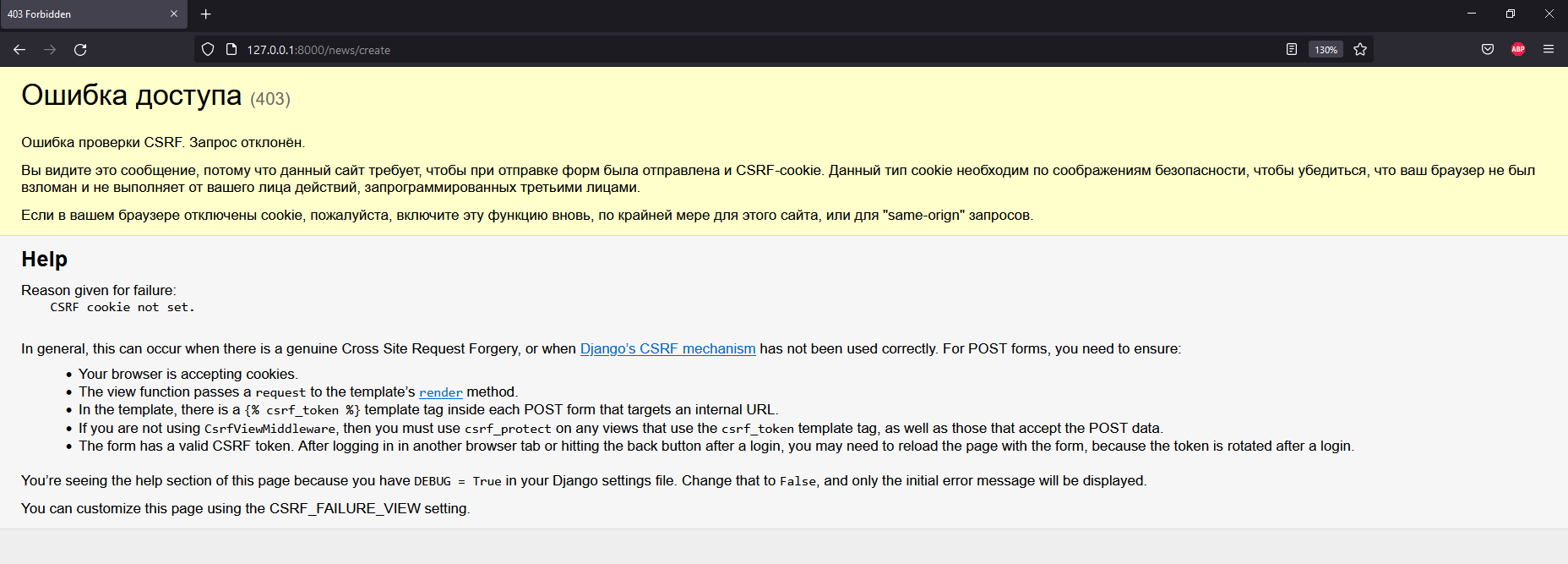


Рисунок 11 - Ошибка доступа

Для того, чтобы её исправить, нам необходимо прописать в шаблоне *create* “*CSRF-token”*. Это максимально простой и результативный способ защиты сайта. Он работает так: сервер создаёт случайный ключ (он же токен) и отправляет его браузеру клиента. Когда браузер запрашивает у сервера информацию, сервер, прежде чем дать ответ, требует показать ключ и проверяет его достоверность. Переходим снова в файл *create.html* и указываем Jinja-шаблон для *CSRF-токена*. После этого, если мы ничего не ввели в формы на веб-странице, то у нас при нажатии на ту же кнопку просто обновляется страница без каких-то изменений.

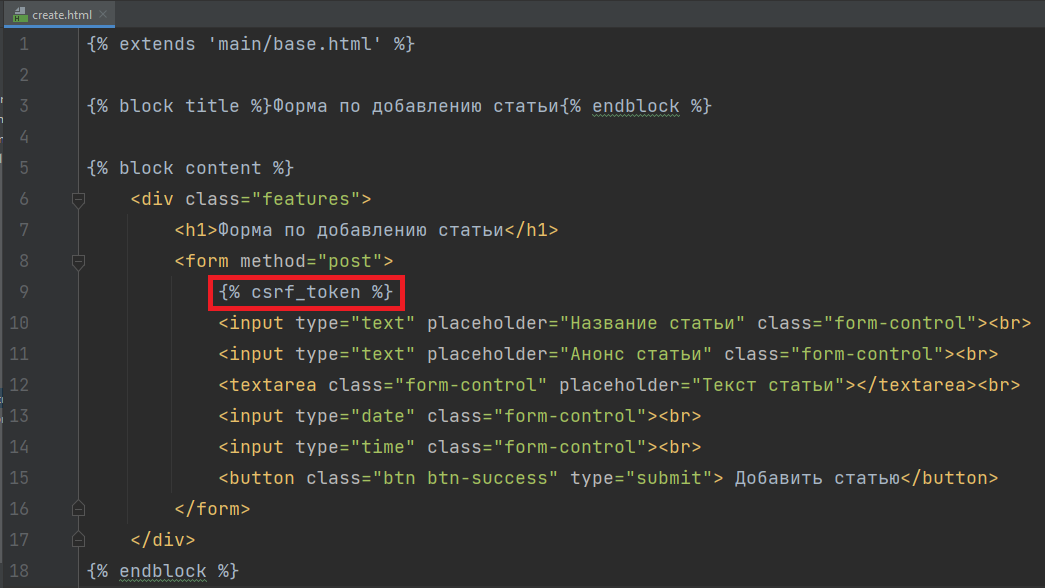


Рисунок 12 - Jinja-шаблон CSRF-token

Теперь перейдём уже к формированию полей (форм) ввода для полей модели *Articles* базы данных. Создадим файл *forms.py* внутри директории *news*, в котором для каждого поля сделаем собственную ячейку, к которой потом будем обращаться в HTML-файле.

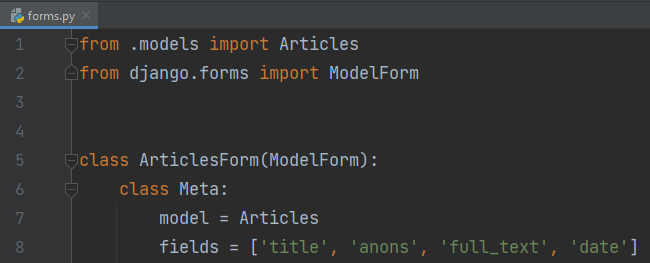


Рисунок 13 - Файл form.py

Создадим переменную *form* для хранения данных класса *ArticlesForm*.

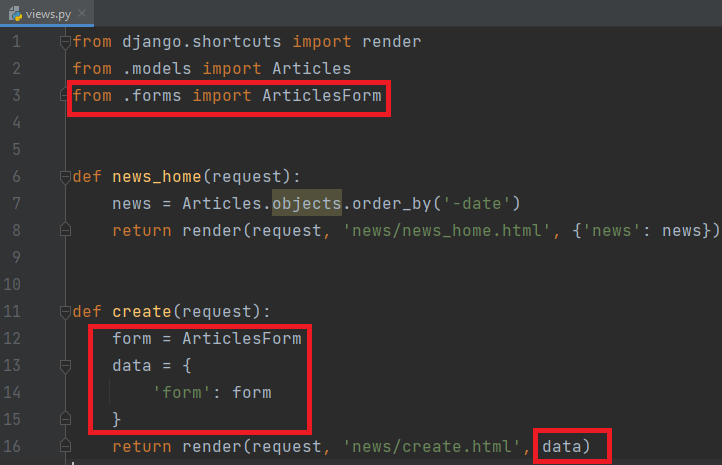


Рисунок 14 - Передача данных формы

Для вывода полей формы в шаблоне нам нужно выводить переменную *form*.

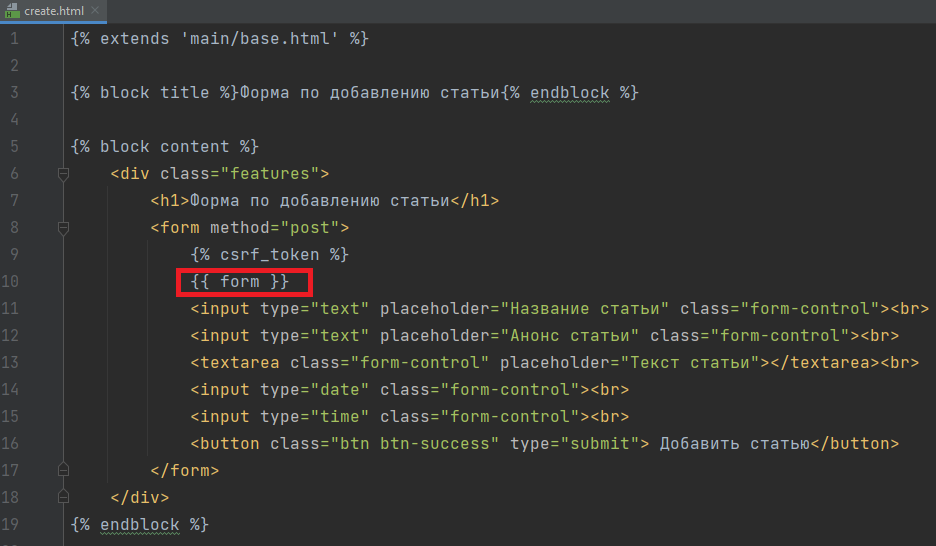


Рисунок 15 - Вывод формы на страницу news/create

Проверим, как выводится веб-страница с новыми формами. Если обновить её, то должно отобразиться две формы: новая и первоначальная.

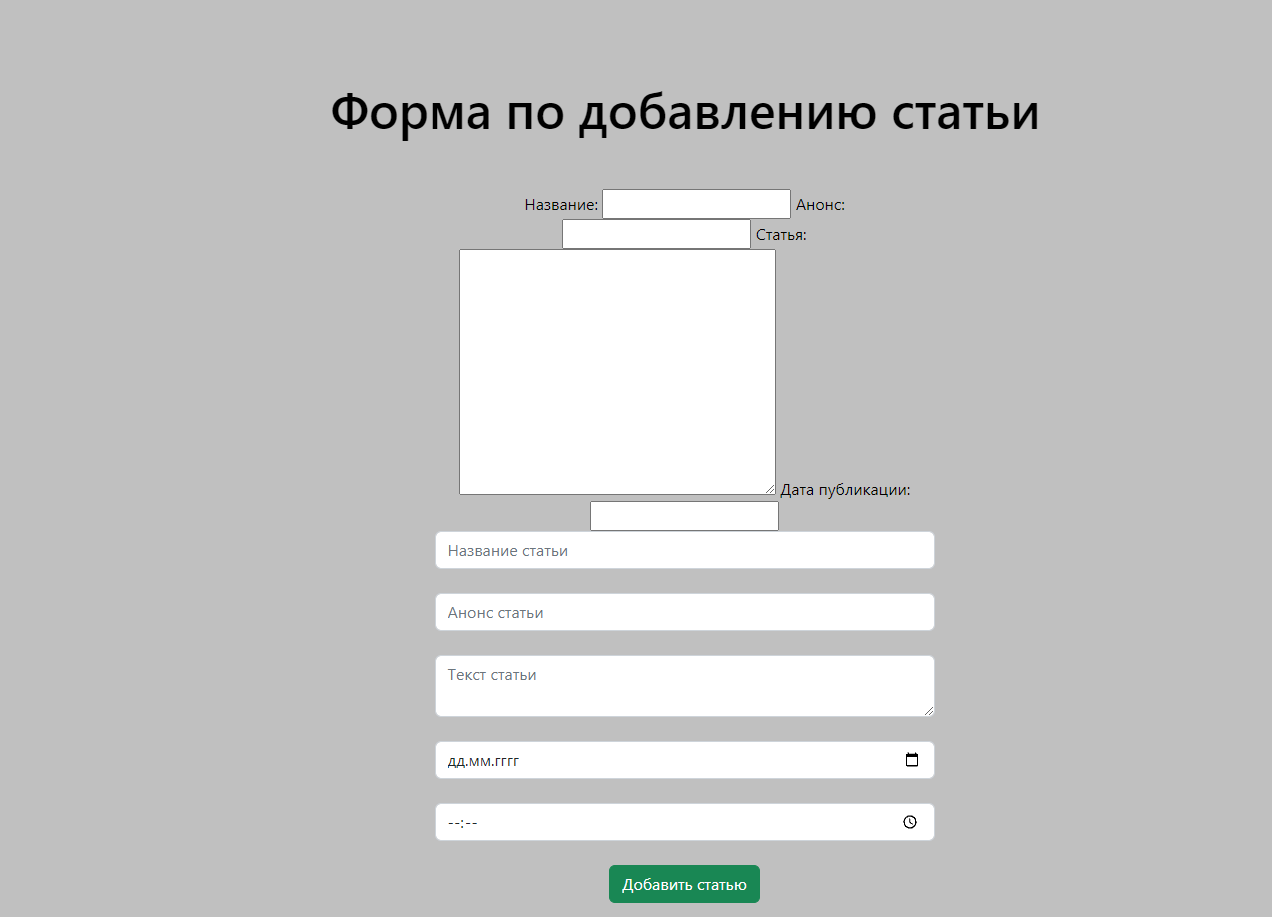


Рисунок 16 - Два вида форм по добавлению статьи

Теперь в *create.html* выведем каждое из полей формы по отдельности.

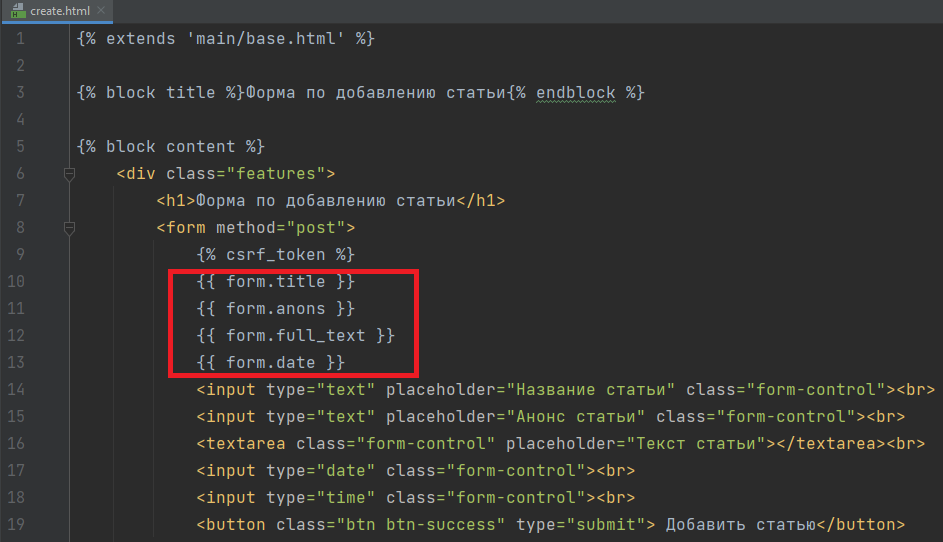


Рисунок 17 - Вывод полей формы по-отдельности

Проверив результат, можно заметить, что названия новых полей теперь не отображаются.

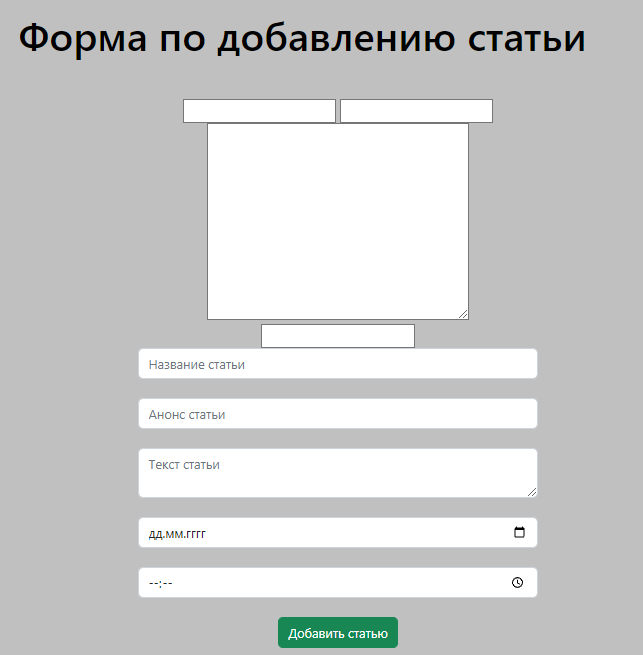


Рисунок 18 - Форма по добавлению статьи

Добавим к полям новых форм ввода названия, создавая в файле *forms.py* словарь, в котором мы создаём ключи с их значениями и с названиями полей модели *Articles*, а также атрибуты для типов данных.

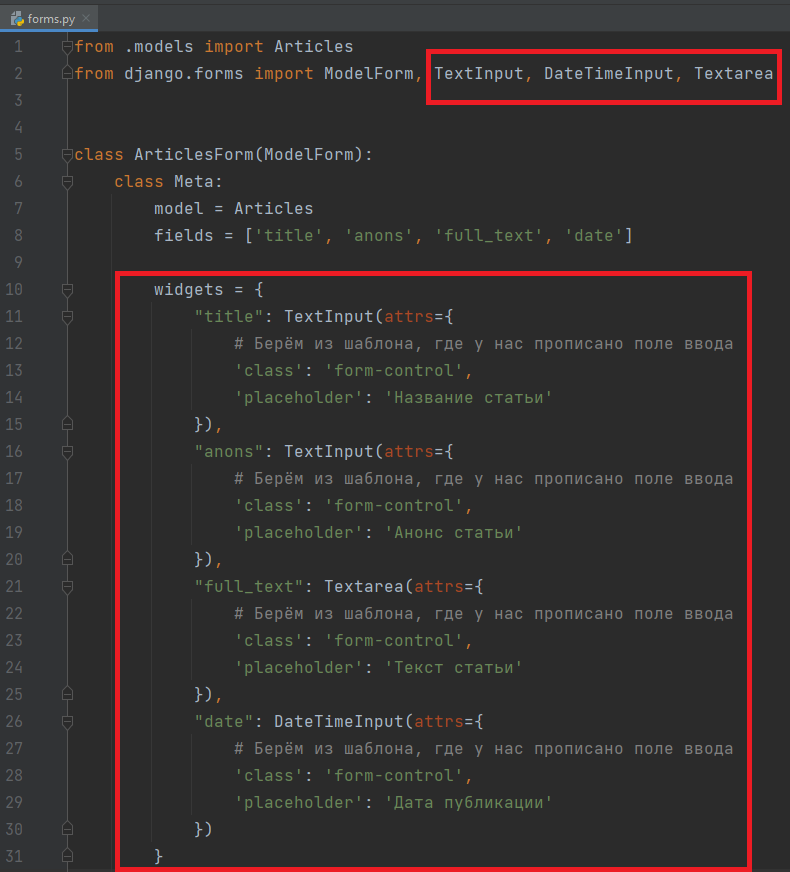


Рисунок 19 - Добавление атрибутов

Проверим результат на локальном сервере.

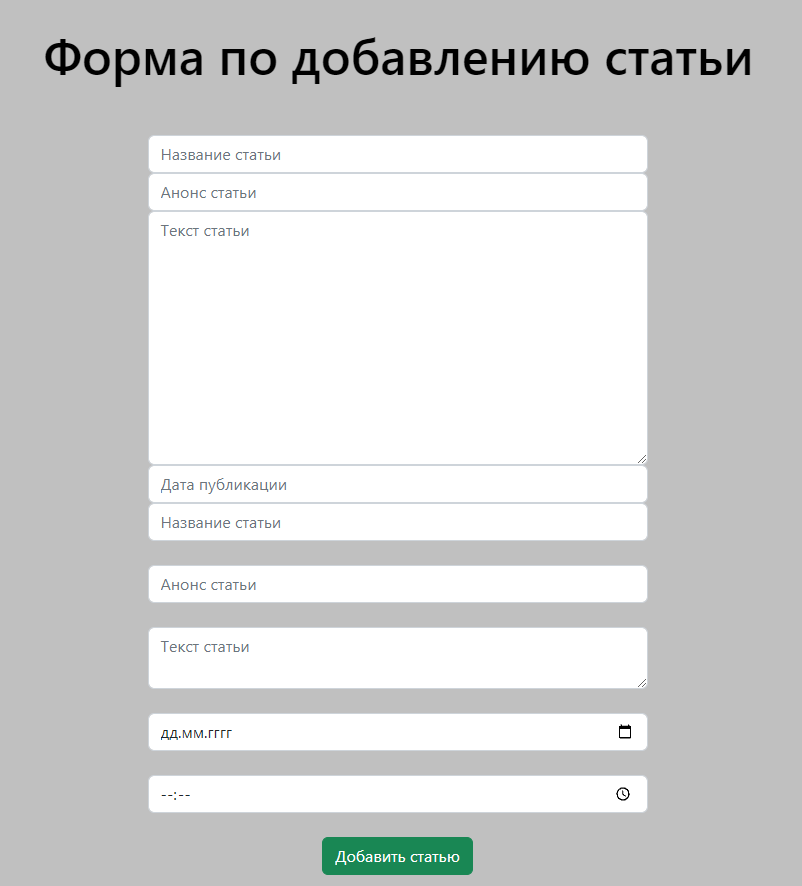


Рисунок 20 - Форма по добавлению статьи

Удалим ненужные поля ввода от первоначальной формы и добавим отступы между полями новой формы при помощи тега <*br*>.

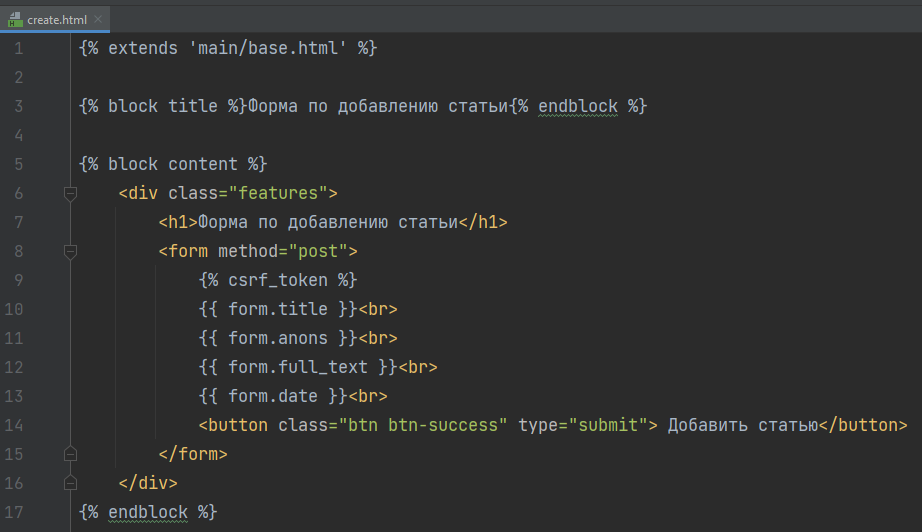


Рисунок 21 - Удаление первоначальной формы

Проверив результат, можно увидеть, что новая форма теперь отображается так же, как и предыдущая.

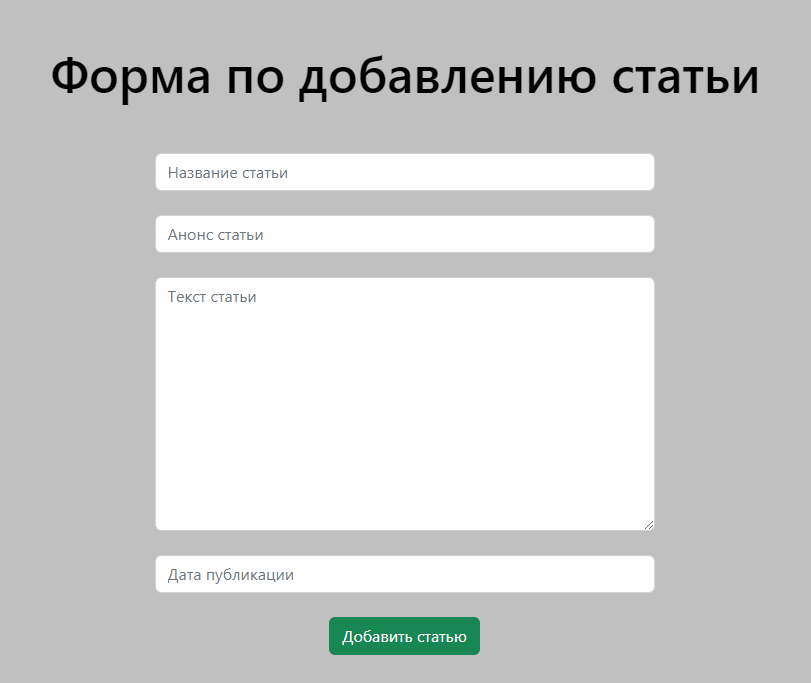


Рисунок 22 - Финальная форма по добавлению статьи

Теперь необходимо реализовать именно сам процесс передачи и сохранения введённых данных в формах на веб-странице в базу данных. Этот процесс будет производиться после нажатия на кнопку.

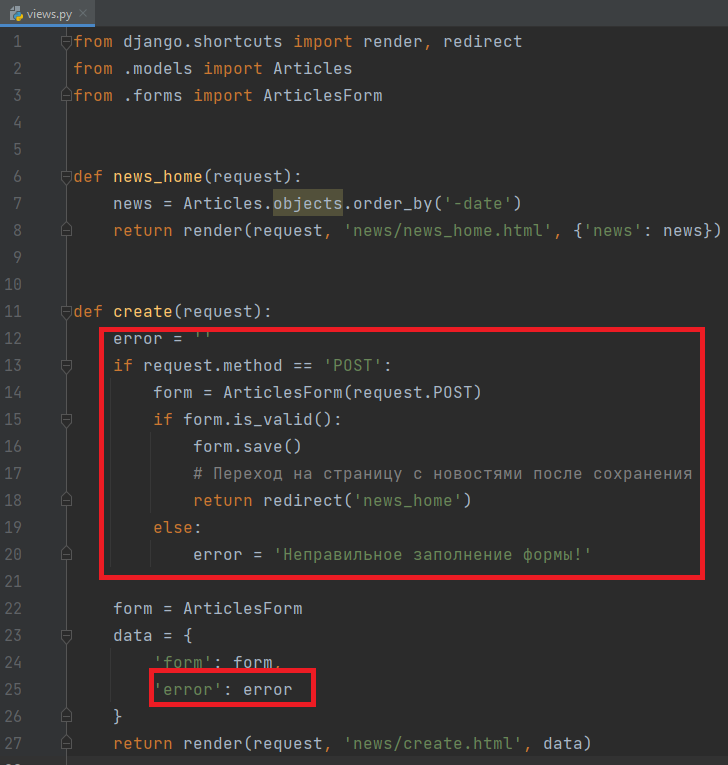


Рисунок 23 - Проверка на наличие ошибок и сохранение записи

Добавим запись в базу данных с помощью созданной формы, введя название, анонс и текст статьи, а также дату публикации.

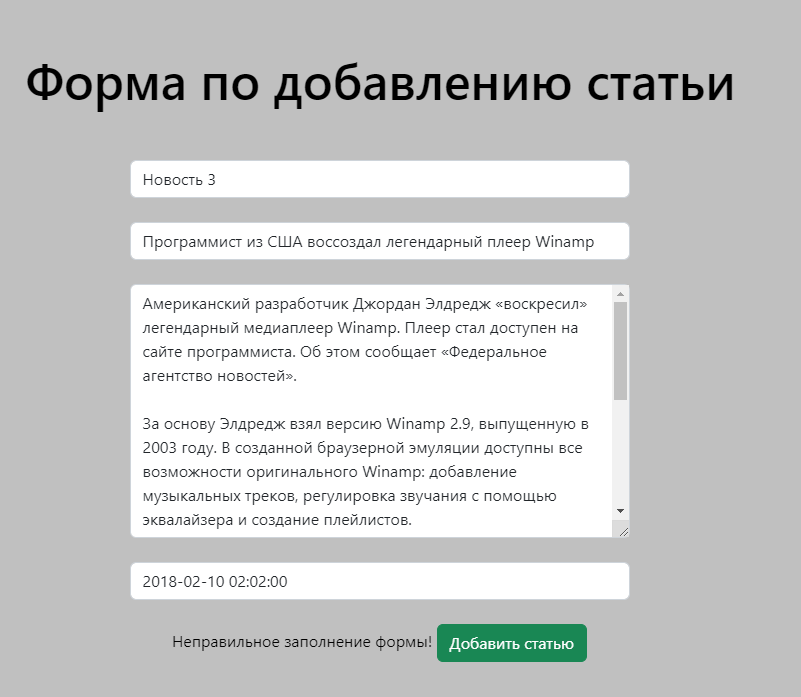


Рисунок 24 - Добавление записи в базу данных

После нажатия на кнопку “Добавить статью” произойдет перенаправление на страницу “Новости на сайте”, и добавленная запись будет отображаться в списке новостей.

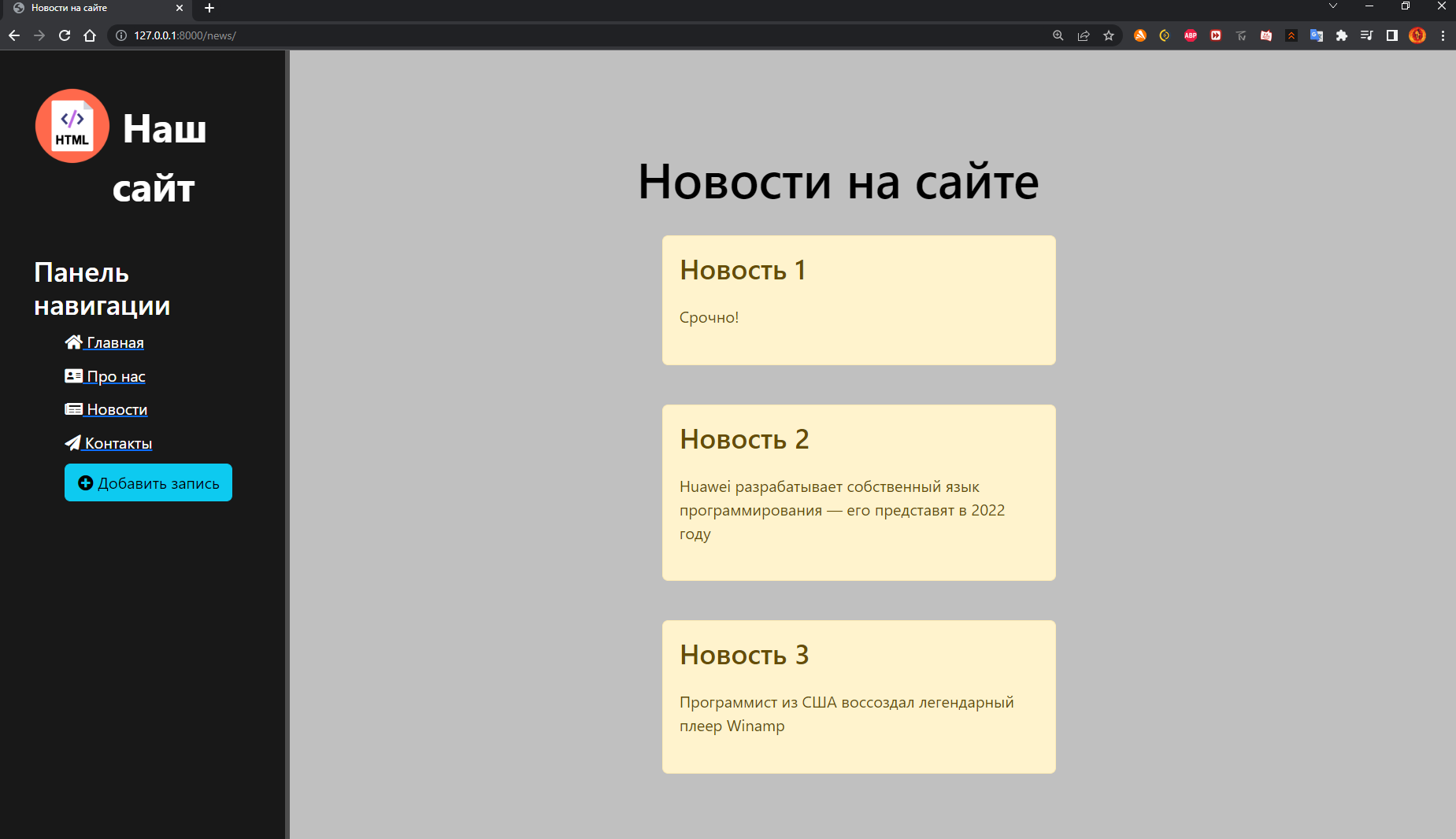


Рисунок 25 - Отображение добавленной записи