МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И

МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

СВЯЗИ И ИНФОРМАТИКИ»

Кафедра «Сети связи и системы коммутации»

Лабораторная работа №3

по дисциплине «Web-программирование»

Выполнил студент

группы БФИ1901

Кареева В.В.

Проверил:

Москва, 2021

# **1 Задание на лабораторную работу**

1. Перейдите во вкладку Articles и создайте 3 статьи, заполнив все поля.

2. Измените файл базы данных в соответствии с требованиями лабораторной работы.

3. Откройте файл базы данных, где хранятся экземпляры статей текущего проекта, с помощью программы управления базами данных sqlite3 и добавьте новую запись в блог через менеджера базы;

# **2 Ход лабораторной работы.**

## 2.1 Создание первой модели данных и ее регистрация в административном приложении Django

С помощью команды: django-admin.py startproject blog необходимо создать проект с названием blog в директории lab3. Затем, перейти в папку blog и выполнить команду: python manage.py startapp articles. Это представлено на рисунке 1.

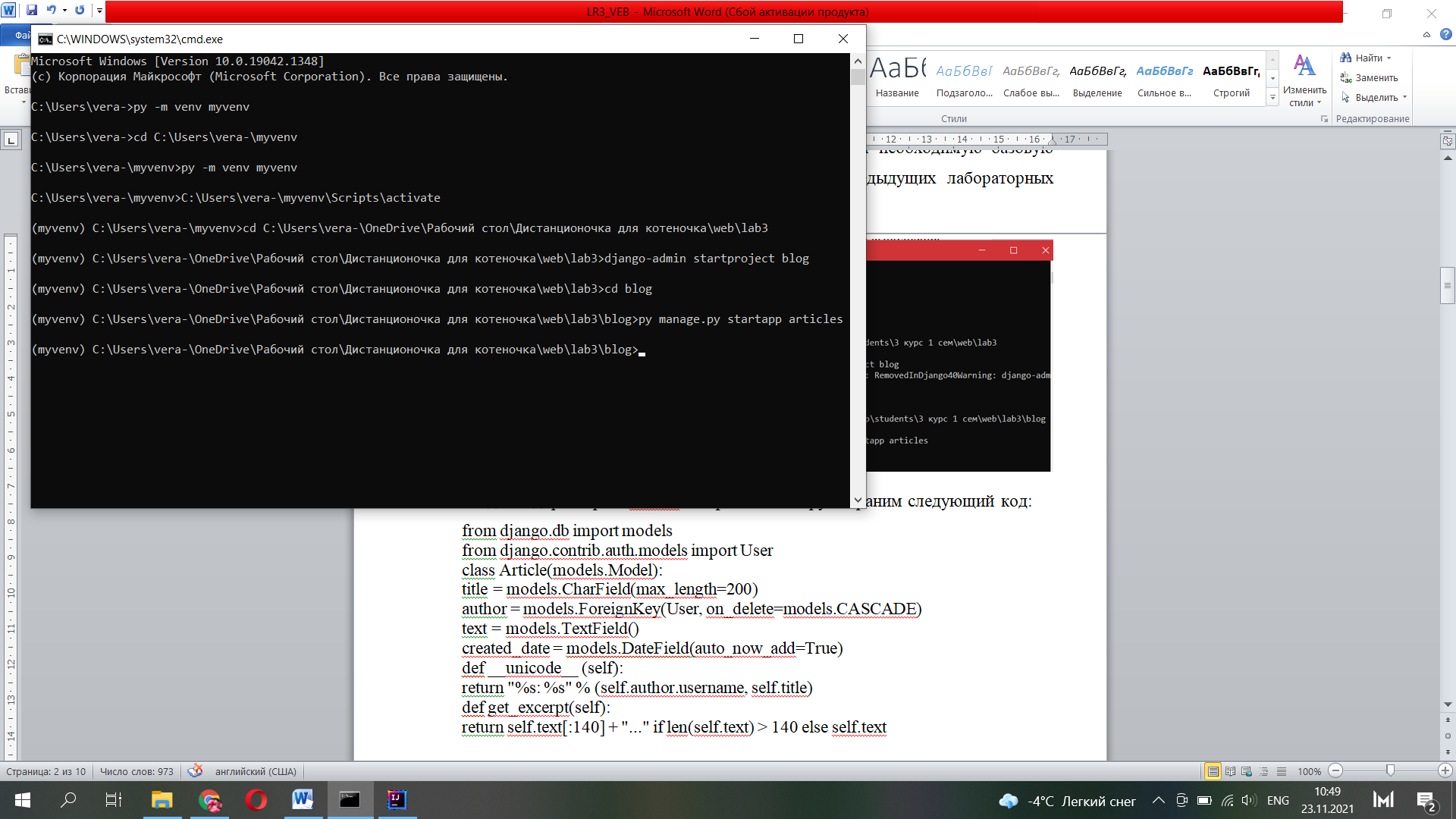


Рисунок 1 - Создание проекта

Эта команда создаст в вашем проекте blog новое приложение. Далее необходимо выполнить базовую настройку проекта, как это было описано в предыдущих лабораторных работах.

Для этого нужно зайти в директорию articles и в файл models.py сохранить код, представленный на листинге 1.

Листинг 1 - Models.py

from django.db import models

from django.contrib.auth.models import User

class Article(models.Model):

title = models.CharField(max\_length=200)

author = models.ForeignKey(User, on\_delete=models.CASCADE)

text = models.TextField()

created\_date = models.DateField(auto\_now\_add=True)

def \_\_unicode\_\_ (self):

return "%s: %s" % (self.author.username, self.title)

def get\_excerpt(self):

return self.text[:140] + "..." if len(self.text) > 140 else self.text

else self.text

Далее в этой же директории необходимо открыть файл admin.py и сохрать в нем код, представленный на листинге 2.

Листинг 2 – admin.py

from django.contrib import admin

from .models import Article

class ArticleAdmin(admin.ModelAdmin):

list\_display = ('title', 'author', 'get\_excerpt', 'created\_date')

admin.site.register(Article, ArticleAdmin)

Затем проверим корректность выпиленных действий, для этого перейдем по адресу: <http://127.0.0.1:8000/admin/>. На рисунке 2 представлена административная панель.

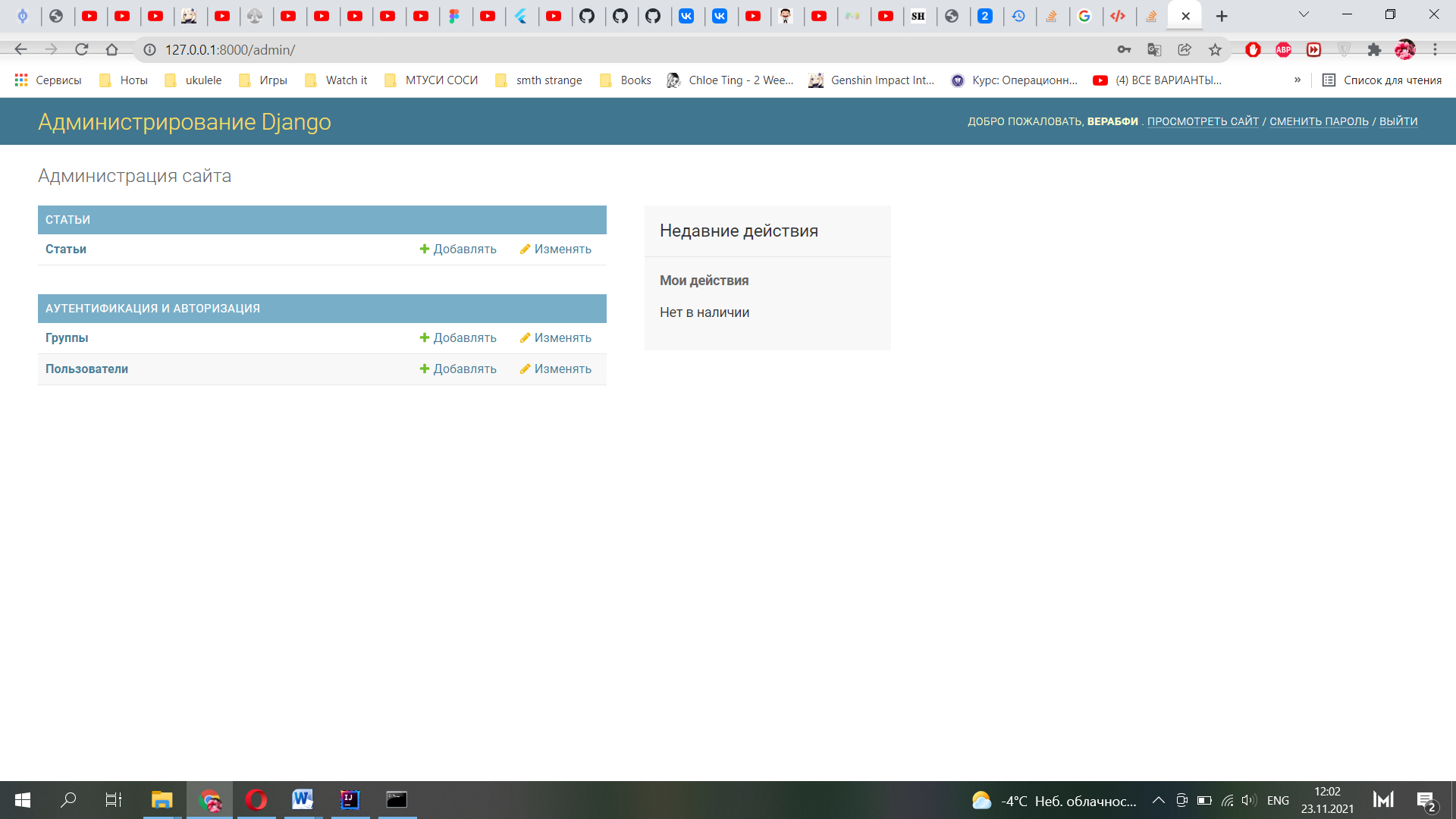


Рисунок 2 - Административная панель

Далее создадим 3 статьи, заполнив все поля. Создание статьи представлено на рисунке 3.

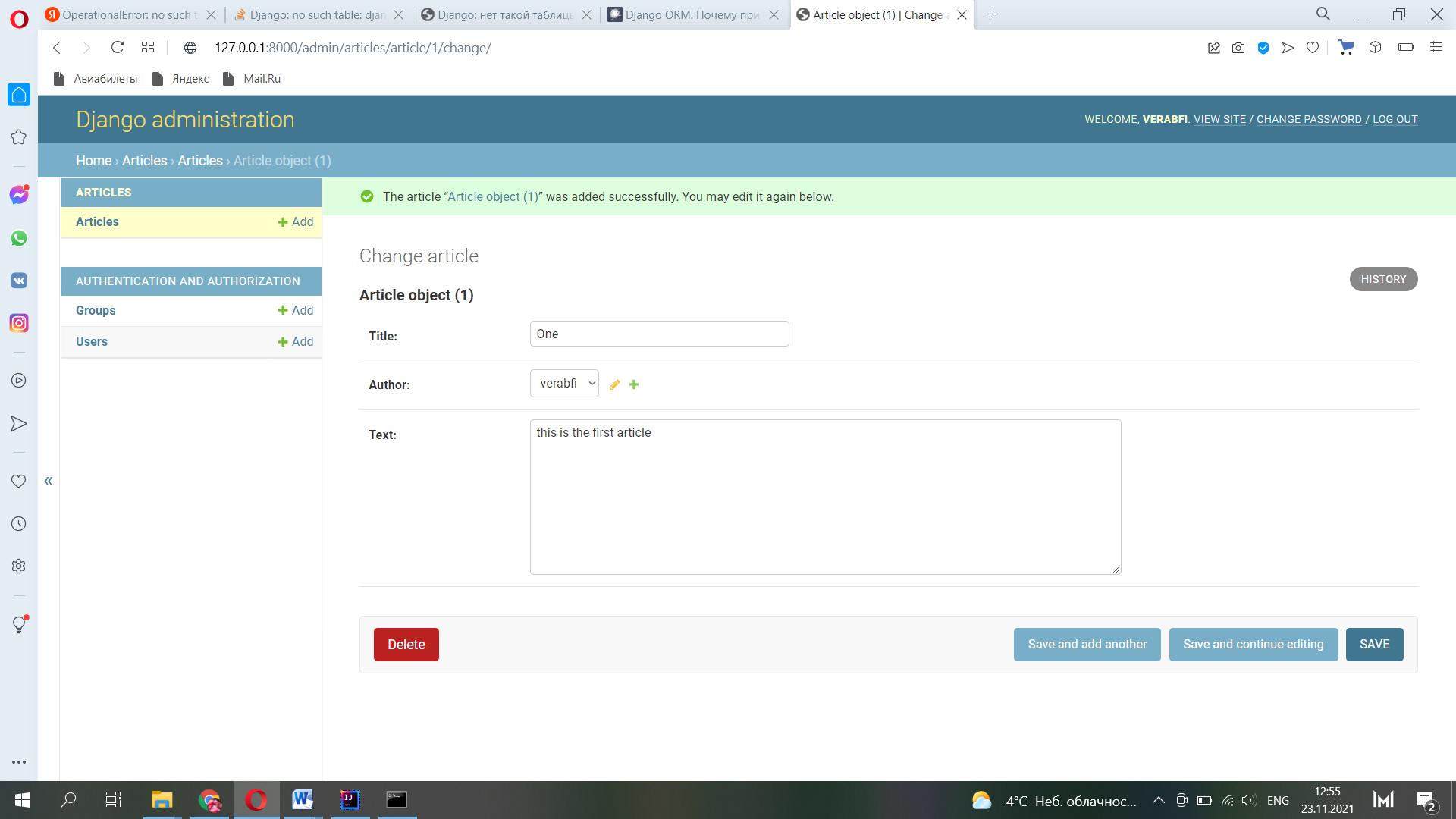


Рисунок 3 – Создание статьи.

Результат добавления статей представлен ан рисунке 4.

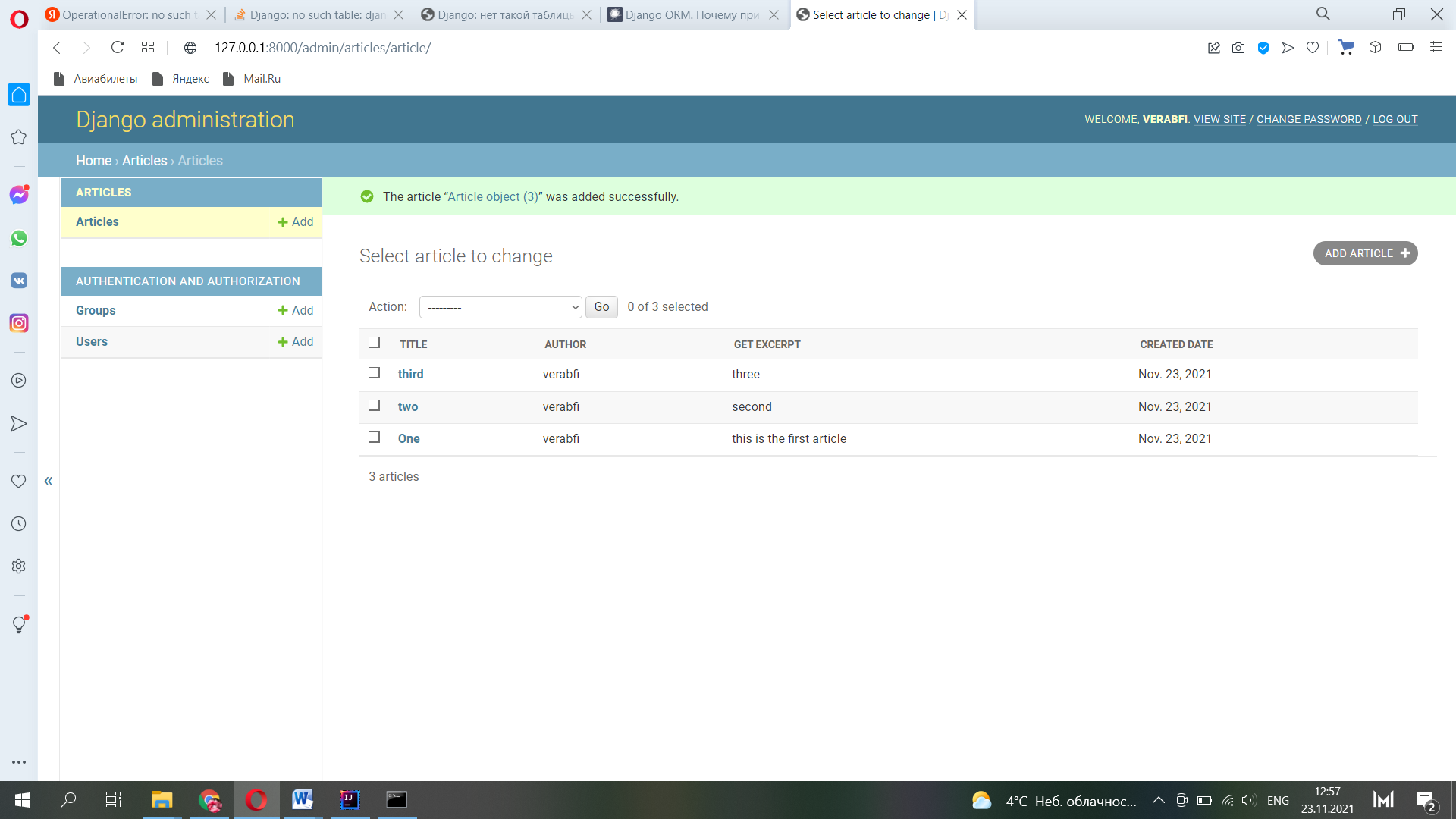


Рисунок 4 – Созданные статьи

Далее, необходимо открыть файл вашей базы данных текущего проекта, который хранится в папке проекта с именем, объявленным в настройках проектах в переменной «DATABASES.NAME», найти созданные в предыдущем пункте задания экземпляры записей, а затем изменить текст одной записи и название статьи для другой, а также создать еще одну статью. На рисунке 5 представлена программа SQLite Maneger.

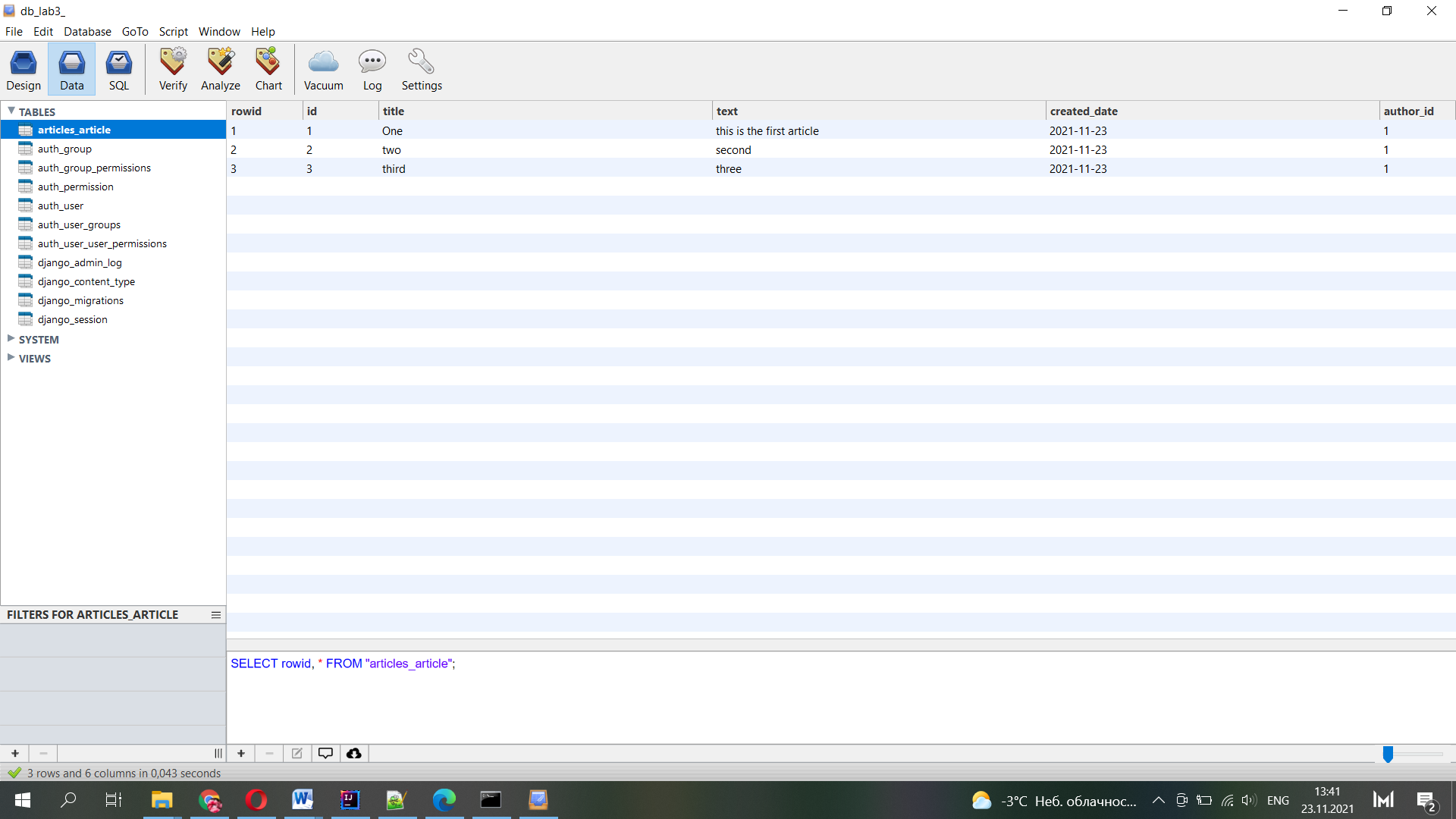


Рисунок 5 - SQLiteManager

Для того что бы изменить запись, необходимо выбрать любую запись, а затем изменить её. На рисунке 6 представлен процесс изменения записи.

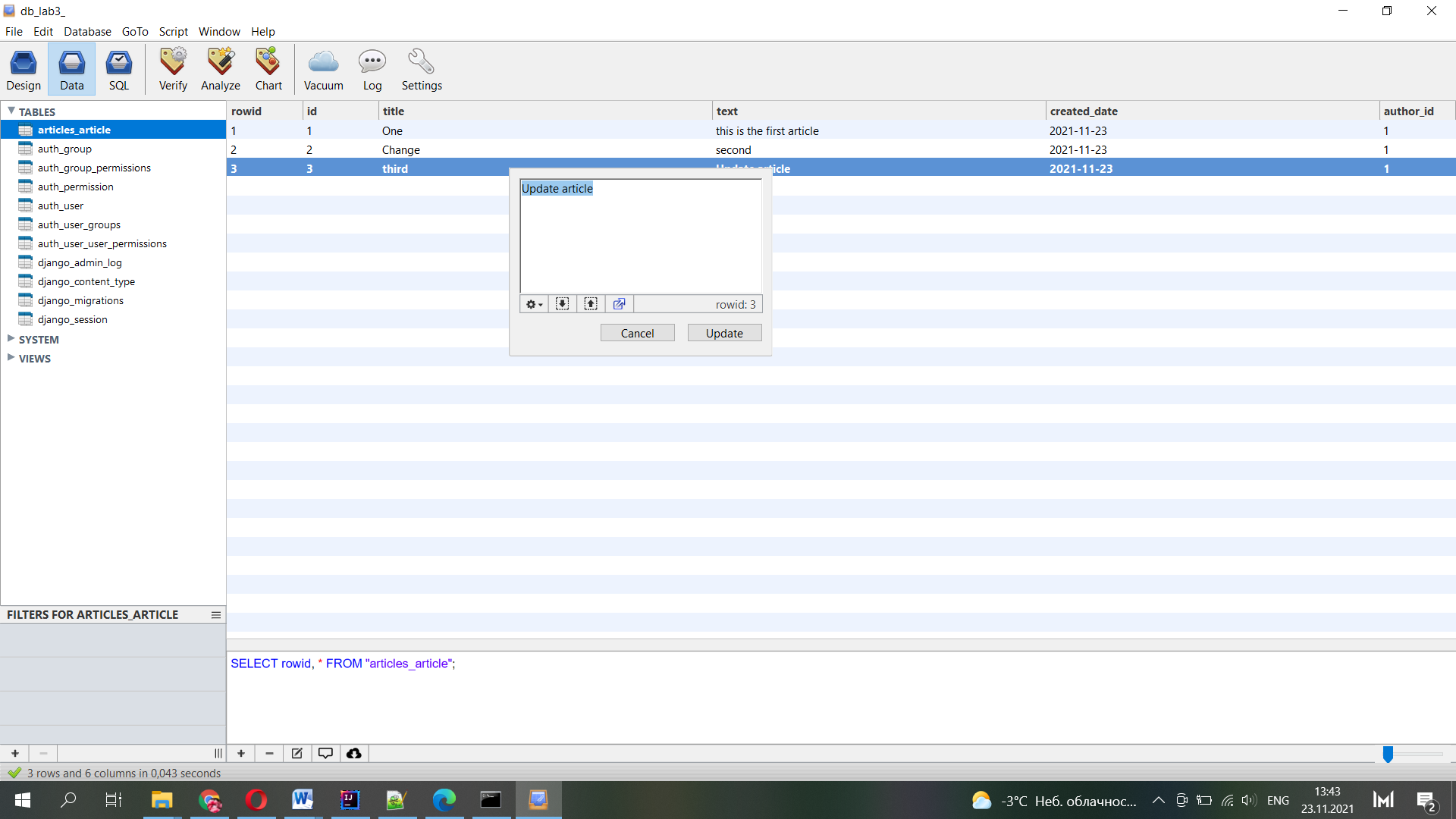


Рисунок 6 – Изменение записи.

Затем необходимо добавить новую запись, для этого нажмем на кнопку «+» снизу, а затем введем данные. На рисунках 7-8 представлен этого процесс. На рисунке 8 представлен результат добавление статьи.

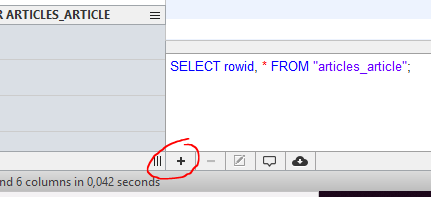


Рисунок 7 – Добавление записи

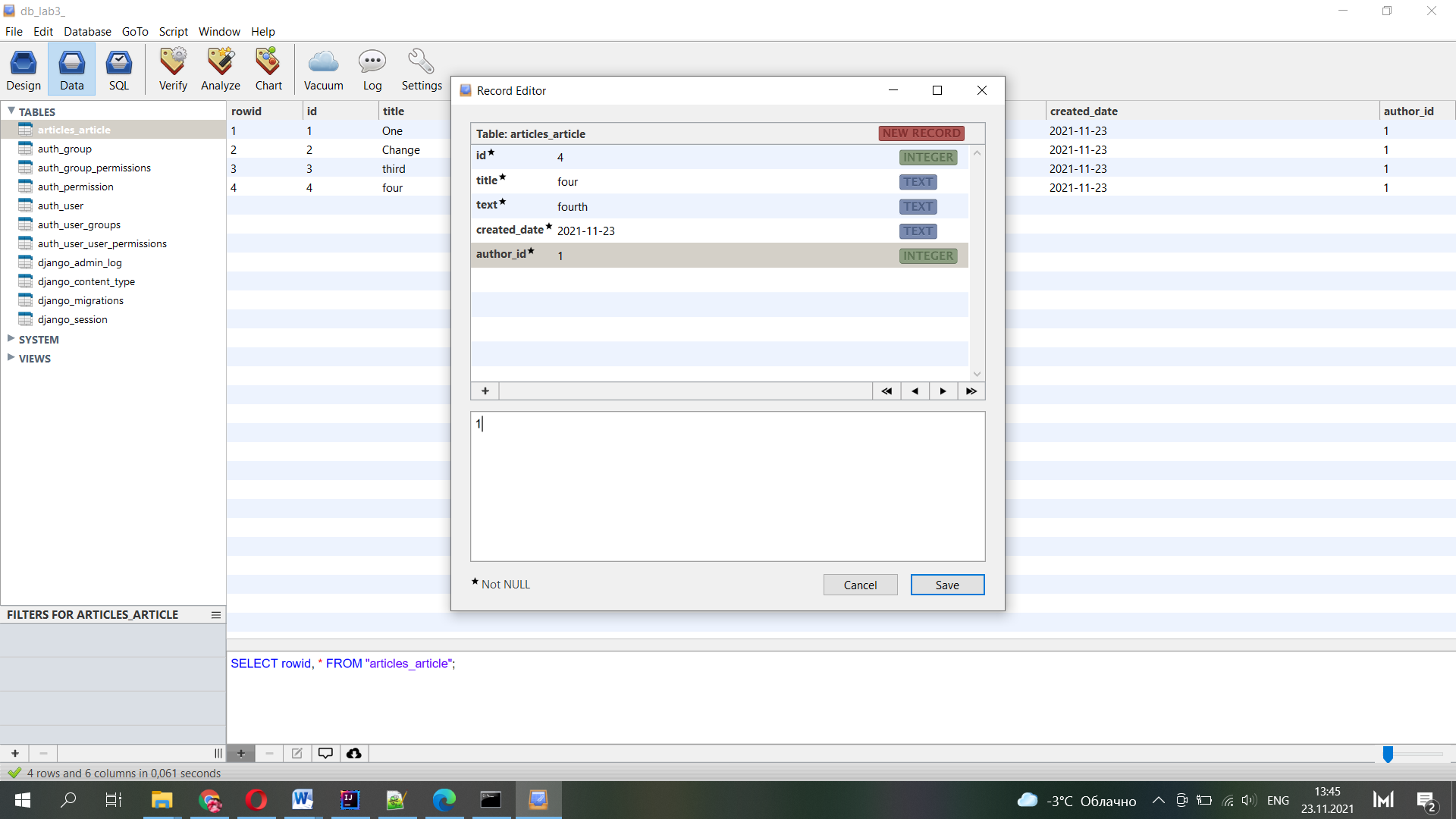


Рисунок 8 - Внесение данных в таблицу.

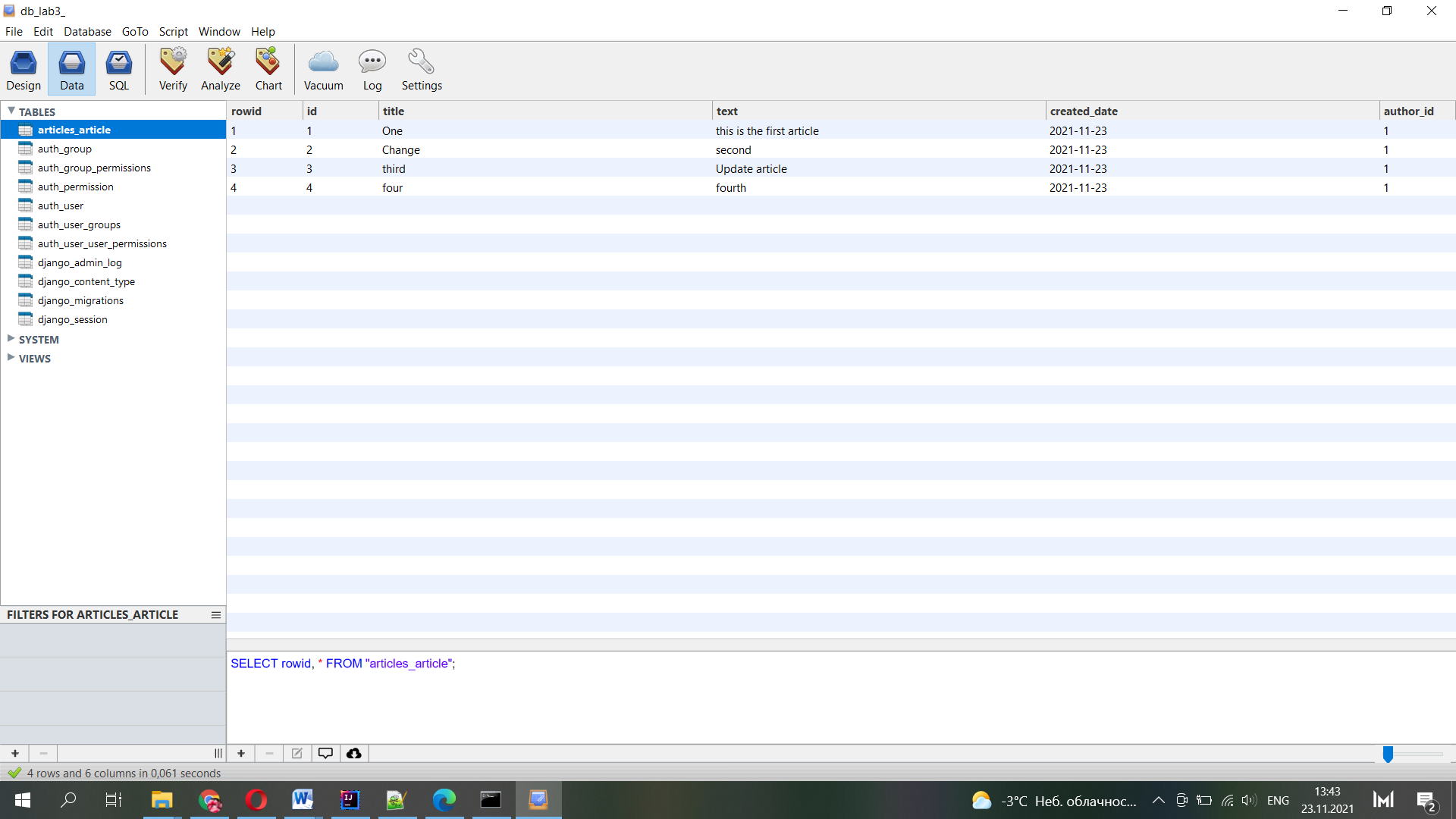


Рисунок 9 – Таблица после добавления новой статьи.

2.2 Динамическое генерирование шаблона для вывода всех экземпляров этой модели

Далее необходимо в директории articles создать папку templates, внутри которой создать файл archive.html, программный код которого представлен на листинге 3.

Листинг 3 – Файл archive.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Архив статей</title>  
</head>  
<body>  
 <div class="header">  
 <img src="{{ STATIC\_URL }}/static/img/cat.png"/>  
 </div>  
 <div class="archive">  
 {% for post in posts %}  
 <div class="one-post">  
 <h2 class="post-title">{{ post.title }}</h2>  
 <div class="article-info">  
 <div class="article-author">{{ post.author.username }}</div>  
 <div class="article-created-date">{{ post.created\_date }}</div>  
 </div>  
 <p class="article-text">{{ post.get\_excerpt }}</p>  
 </div>  
 {% endfor %}  
 </div>  
</body>  
</html>

Затем, в файле views.py в директории articles создать представление archive, которое будет возвращать html-страницу со всеми созданными постами в текущем проекте. Программный код views.py представлен на листинге 4.

Листинг 4 – Файл views.py

from models import Article

from django.shortcuts import render

def archive(request):

return render(request, 'archive.html', {"posts": Article.objects.all()})

Далее, необходимо настроить url, по которому будут отображаться все статьи проекта. Для этого в файле urls.py в urlpatterns вставим «path('', views.archive, name='home')». После этого запустим сервер и перейдем по адресу «127.0.0.1:8000». На рисунке 10 представлен сайт со статьями.

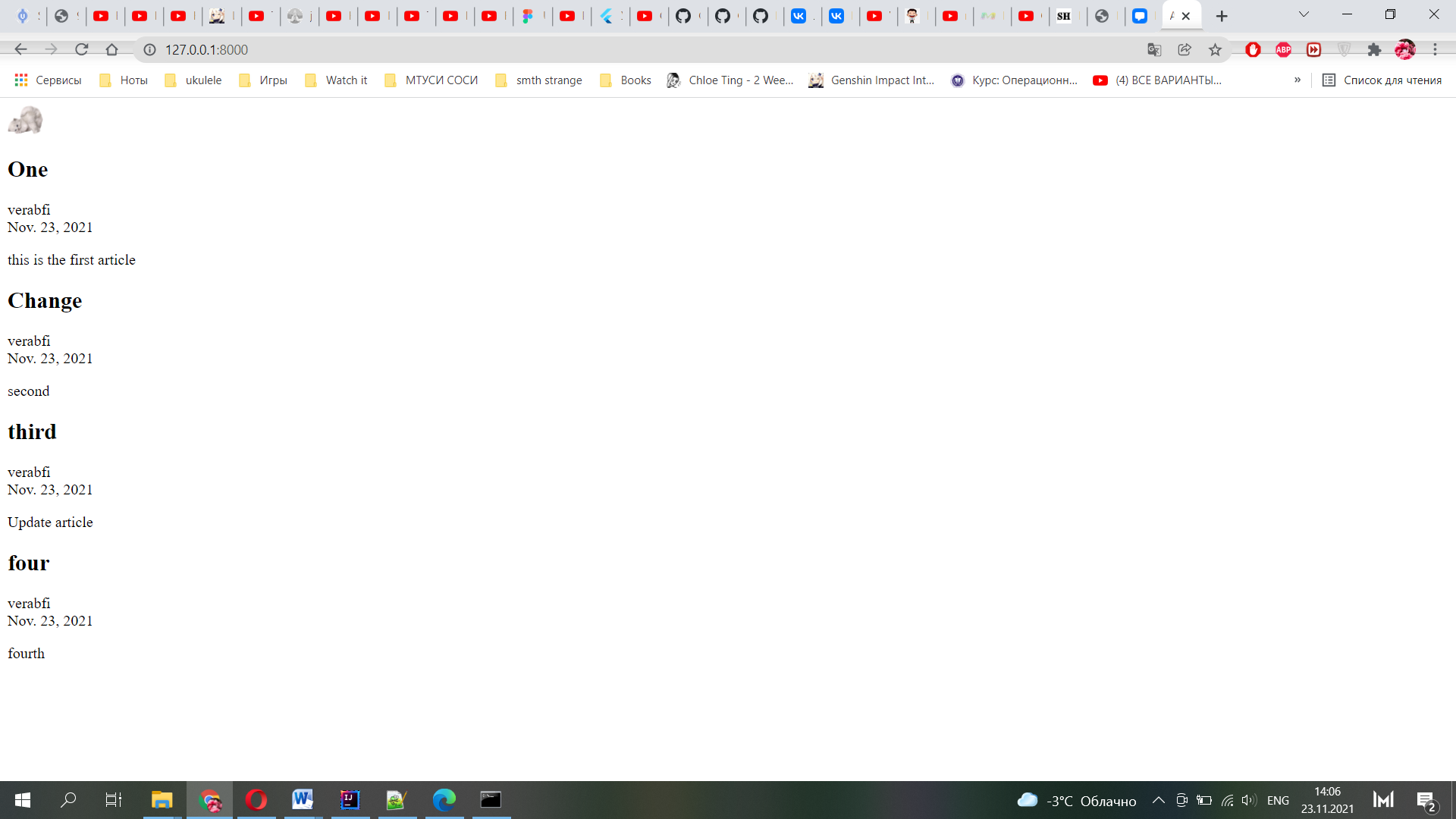


Рисунок 10 – Сайт со статьями.

Далее откроем файл базы данных и добавим ещё одну запись. Это представлено на рисунке 11.

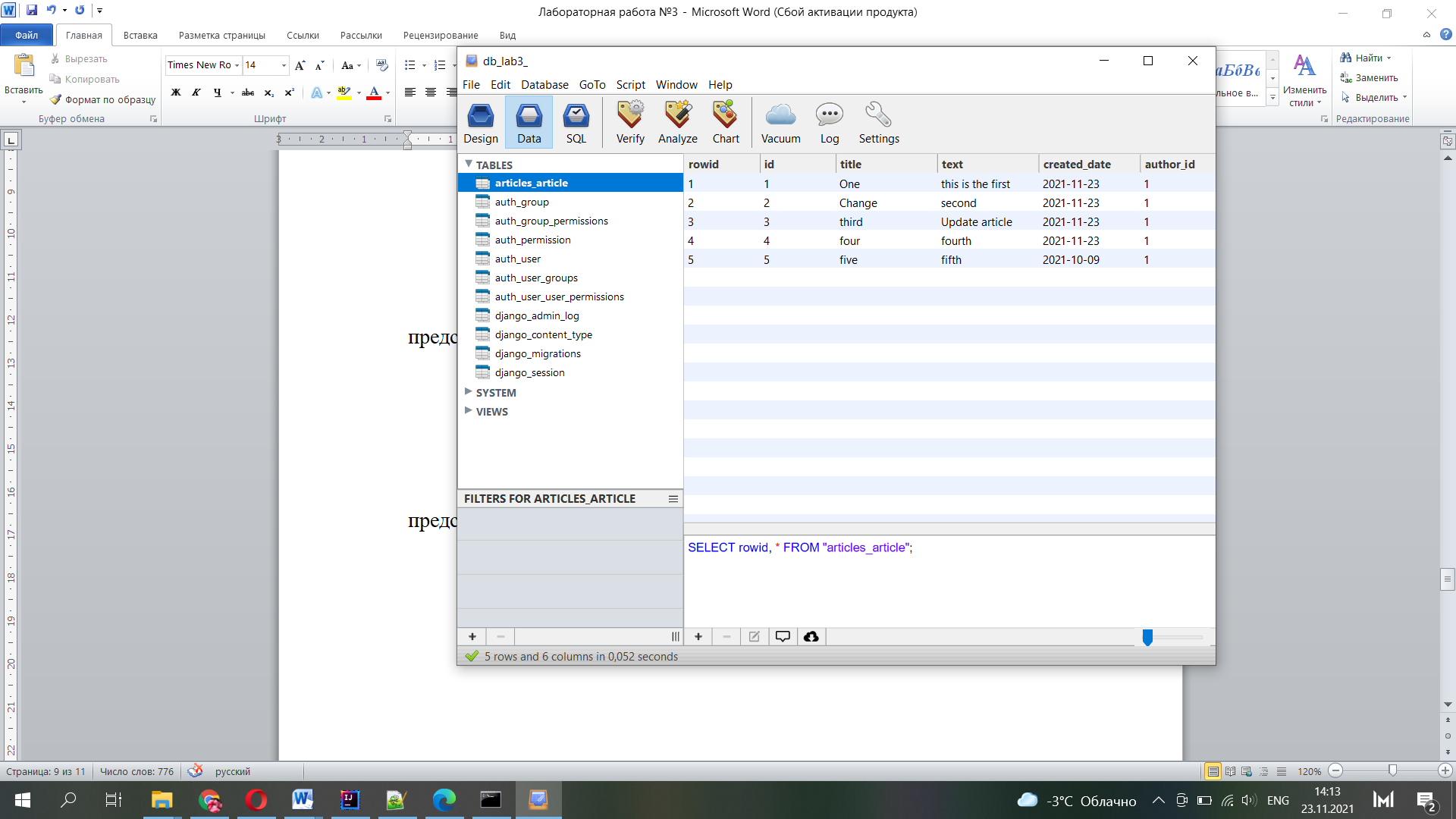


Рисунок 11 – Добавление новой статьи

Затем перезапустим сервер и посмотрим на результат. На рисунке 12 представлен сайт после добавления новой записи.

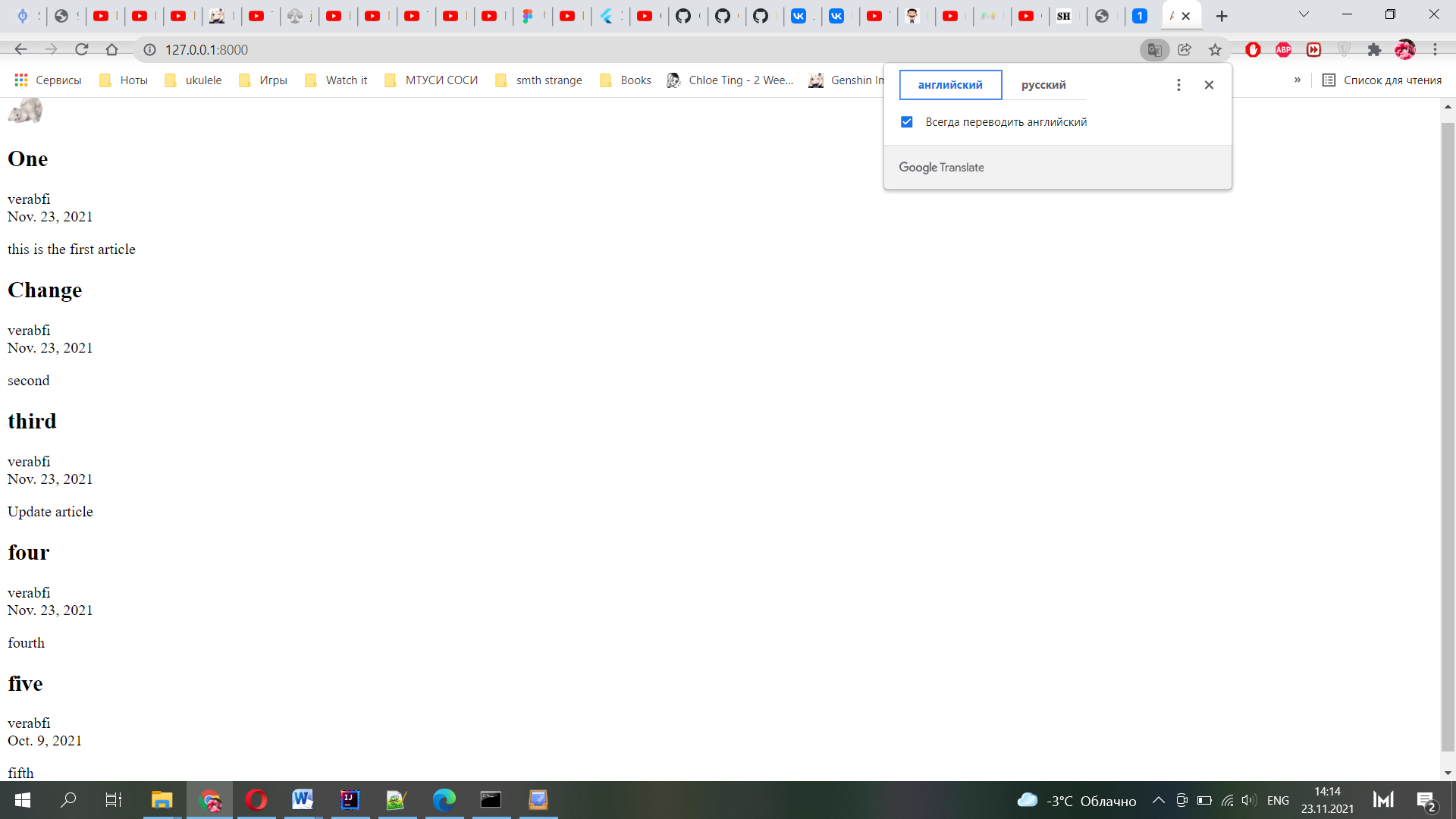


Рисунок 12 – Сайт после добавления статьи.

.

**Список используемых источников**

1 ГОСТ 7.32-2017 СИБИД. Отчет о научно-исследовательской работе. Структура и правила оформления.

2 ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание.