МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Новосибирский государственный технический университет»

Кафедра Автоматизированных систем управления



**Отчет**

**по лабораторной работе №3**

**Тема**: Подключение базы данных к проекту**Дисциплина**: Разработка программных приложений

|  |  |  |
| --- | --- | --- |
| **Бригада:** |  | **Проверил:** |
|  | *Багателия К. А.* | Старший преподаватель кафедры АСУ  Эстрайх И.В |
| **Факультет** | *АВТФ* |
| **Направление** | *09.03.03 - Прикладная информатика* |
| **Группа** | *АП-227* |

Новосибирск

2025

# Цель работы

# Изучить принципы подключения и использования в проекте базы

# данных.

# Задание

1. Создать таблицу БД, заполнить несколькими записями с помощью

фреймворка.

2. Зайти в БД с помощью менеджера SQLiteStudio и убедиться, что

данные в таблице действительно появились

3. Обеспечить наполнение страницы информацией из базы данных.

4. Реализовать работу с записями БД из своей программы: создание,

изменение, удаление, выборку, фильтрацию, сортировку. Проверить,

что все эти операции работают корректно.

5. Добавить в БД слаги и обеспечить отображение записей по их слагу.

6. Создать пользовательский менеджер модели.

7. Использовать в программе класс перечисления

# Ход выполнения задания

1. Создать таблицу БД, заполнить несколькими записями с помощью

фреймворка.

Добавим таблицу Новостей

class News(models.Model):

    name = models.CharField(verbose\_name="Название", max\_length=127)

    slug = models.CharField(verbose\_name="Слаг", max\_length=127)

    description = models.TextField(verbose\_name="Название")

    def \_\_str\_\_(self):

        return f"{self.name}"

    class Meta:

        verbose\_name = "Новость"

        verbose\_name\_plural = "Новости"

Проведём миграции  
python manage.py makemigrations  
python manage.py migrate

В нашем проекте появился файл – db.sqlite3

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Для заполнения несколькими записями, создадим отдельный скрипт который вызовем из консоли.

import os

import django

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'healthcare.settings')

django.setup()

from main.models import News

news\_mock = {

    1: {

        "name": "Тестовая новость 1",

        "slug": "test\_news\_1",

        "desc": "<p style='color:red'>Описание тестовой новости 1</p>"

    },

    2: {

        "name": "Тестовая новость 2",

        "slug": "test\_news\_2",

        "desc": "Описание тестовой новости 2"

    },

    3: {

        "name": "Тестовая новость 3",

        "slug": "test\_news\_3",

        "desc": "Описание тестовой новости 3"

    }

}

for item in news\_mock.values():

    name = item["name"]

    desc = item["desc"]

    slug = item["slug"]

    news = News.objects.create(

        name=name,

        slug=slug,

        description=desc

    )

2. Зайти в БД с помощью менеджера SQLiteStudio и убедиться, что данные в таблице действительно появились

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

3. Обеспечить наполнение страницы информацией из базы данных

Изменим код view который выводит новости

from django.shortcuts import render, redirect

from django.urls import reverse

from django.http import Http404

from main.models import News

from django.shortcuts import get\_object\_or\_404

def homepage(request):

    return render(request, "main.html", {})

def send\_form(request):

    if (request.POST.get("name") and request.POST.get("request")):

        return render(request, "success\_from.html")

    else:

        url\_redirect = f"{reverse('homepage')}?error=true"

        return redirect(url\_redirect)

def news(request):

    news = News.objects.all()

    return render(request, "news.html", {"news" : news})

def news\_detail(request, news\_slug):

    news\_el = get\_object\_or\_404(News, slug=news\_slug)

    return render(request, "news-detail.html", {"elem": news\_el })

def not\_found(request):

    return render(request, "404.html")

4. Реализовать работу с записями БД из своей программы: создание, изменение, удаление, выборку, фильтрацию, сортировку. Проверить, что все эти операции работают корректно  
Добавим возможность нашим партнерам размещать свои новости, напишем CRUD для новостей

def api\_news\_list\_create(request):

    if request.method == 'GET':

        name\_filter = request.GET.get('name')

        ordering = request.GET.get('ordering')

        news = News.objects.all()

        if name\_filter:

            news = news.filter(name\_\_icontains=name\_filter)

        if ordering:

            news = news.order\_by(ordering)

        data = [

            {

                'id': n.id,

                'name': n.name,

                'slug': n.slug,

                'description': n.description

            } for n in news

        ]

        return JsonResponse(data, safe=False)

    elif request.method == 'POST':

        try:

            body = json.loads(request.body)

            name = body.get('name')

            slug = body.get('slug')

            description = body.get('description')

            if not all([name, slug, description]):

                return HttpResponseBadRequest('Нет всех полей')

            news = News.objects.create(name=name, slug=slug, description=description)

            return JsonResponse({

                'id': news.id,

                'name': news.name,

                'slug': news.slug,

                'description': news.description

            }, status=201)

        except json.JSONDecodeError:

            return HttpResponseBadRequest('Invalid JSON')

    else:

        return HttpResponseNotAllowed(['GET', 'POST'])

def api\_news\_detail\_update\_delete(request, pk):

    try:

        news = News.objects.get(pk=pk)

    except News.DoesNotExist:

        return HttpResponseNotFound('News not found')

    if request.method == 'GET':

        return JsonResponse({

            'id': news.id,

            'name': news.name,

            'slug': news.slug,

            'description': news.description

        })

    elif request.method == 'PUT':

        try:

            body = json.loads(request.body)

            news.name = body.get('name', news.name)

            news.slug = body.get('slug', news.slug)

            news.description = body.get('description', news.description)

            news.save()

            return JsonResponse({

                'id': news.id,

                'name': news.name,

                'slug': news.slug,

                'description': news.description

            })

        except json.JSONDecodeError:

            return HttpResponseBadRequest('Invalid JSON')

    elif request.method == 'DELETE':

        news.delete()

        return JsonResponse({'status': 'deleted'}, status=204)

    else:

        return HttpResponseNotAllowed(['GET', 'PUT', 'DELETE'])

Протестируем все виды запросов:

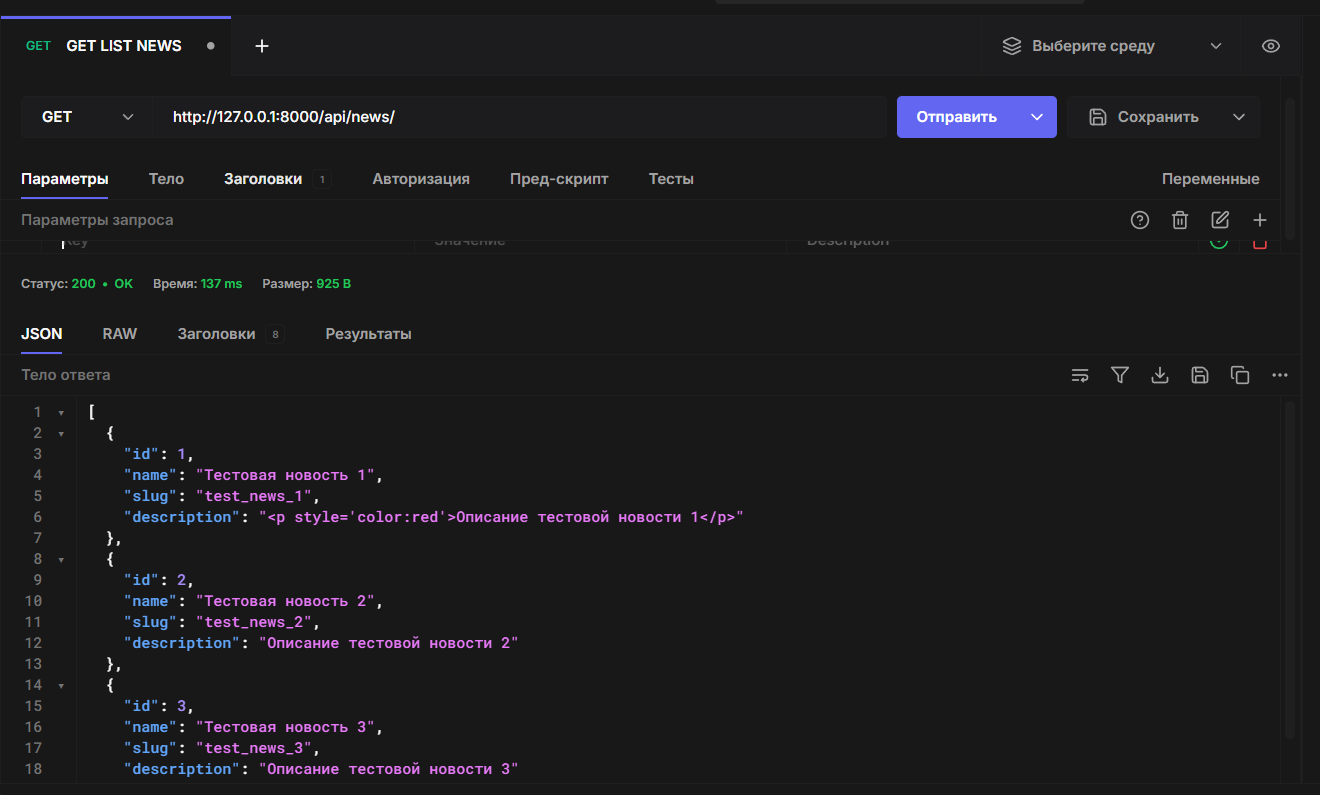
GET /api/news/ - все обьекты с возможностью фильтрации

GET /api/news/<id>/ - один обьект по id

POST /api/news/ - добавляет новую новость

PUT /api/news/<id>/ - Изменяет новость по ID

DELETE /api/news/<id>/ - Удаляет новость по ID

Пример списка новостей без фильтрации  


Пример списка новостей с сортировкой

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Пример списка новостей с фильтрацией

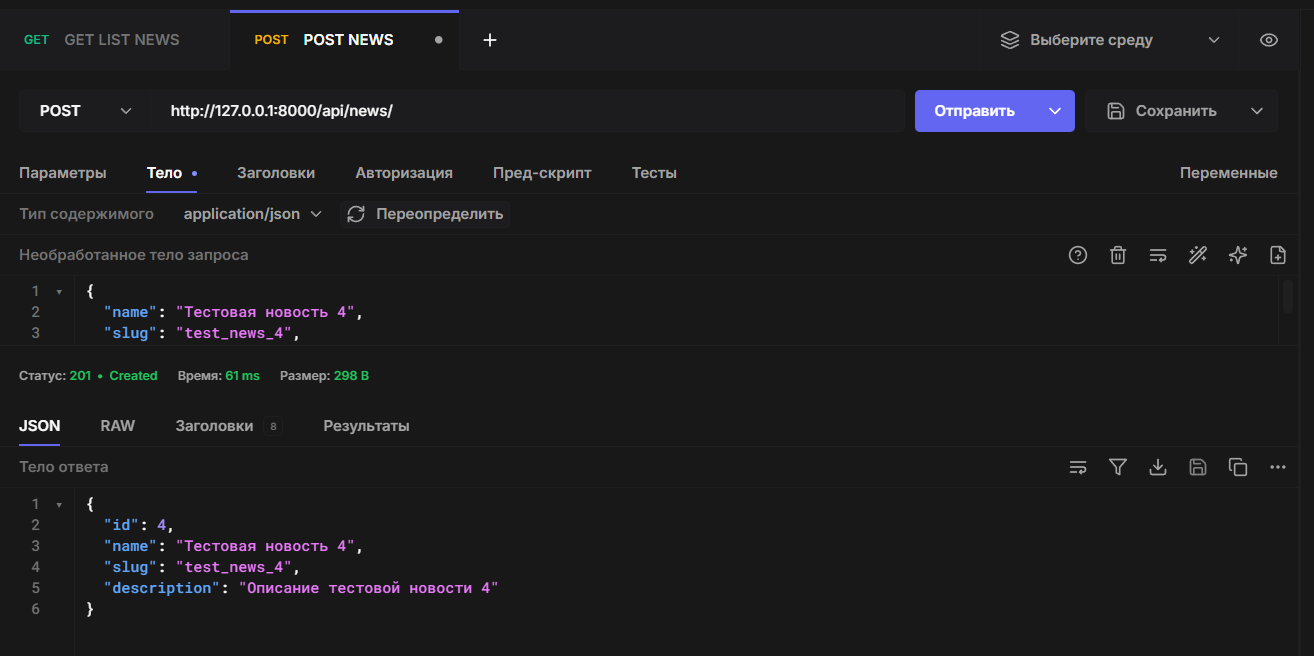
Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.  
Пример получения новости по id

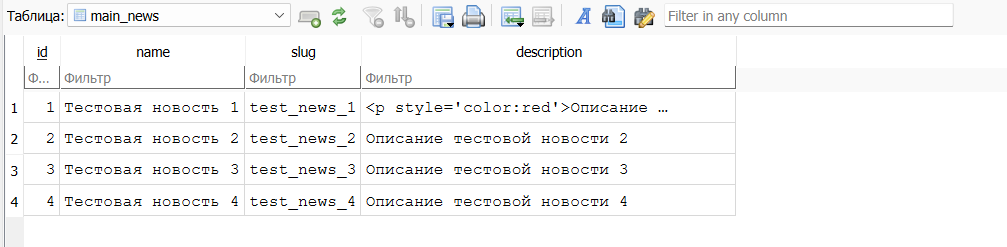
Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

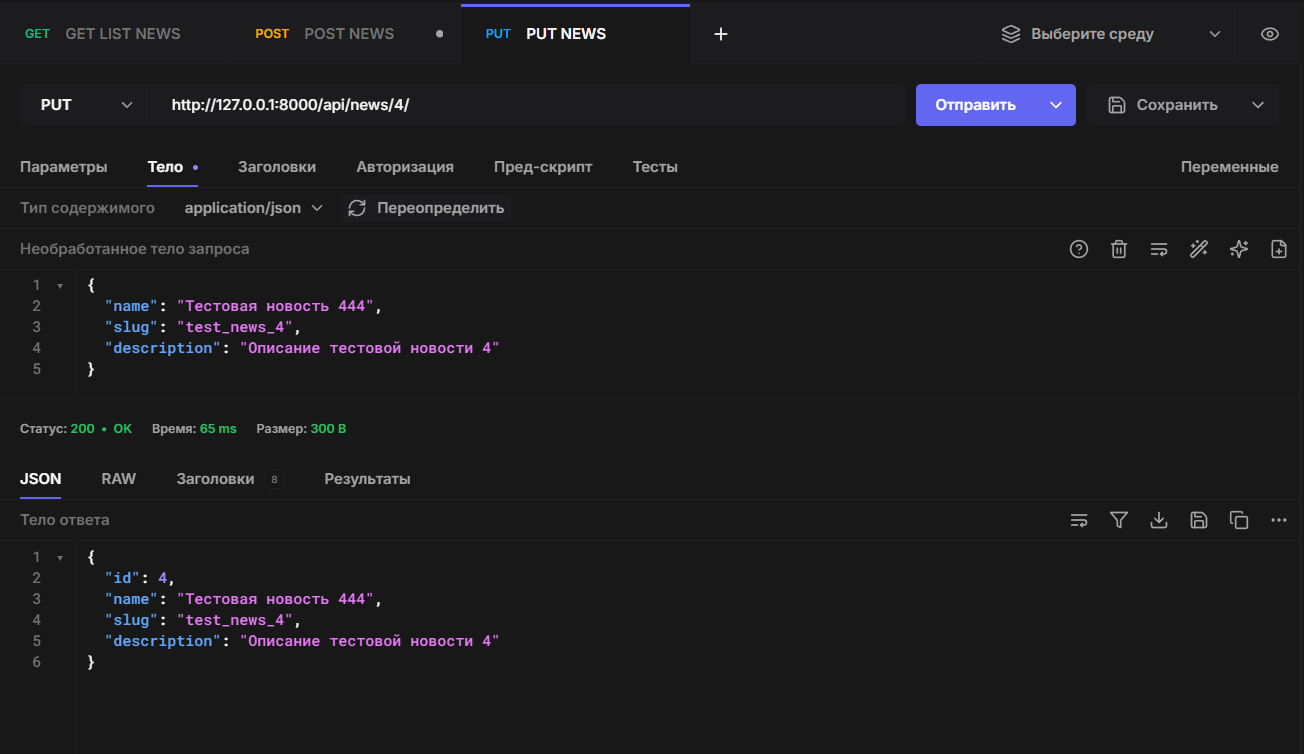
Пример добавления новости



Пример новых данных в таблице



Изменяем объект по id



Пример данных в таблице

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Удаляем объект по id

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

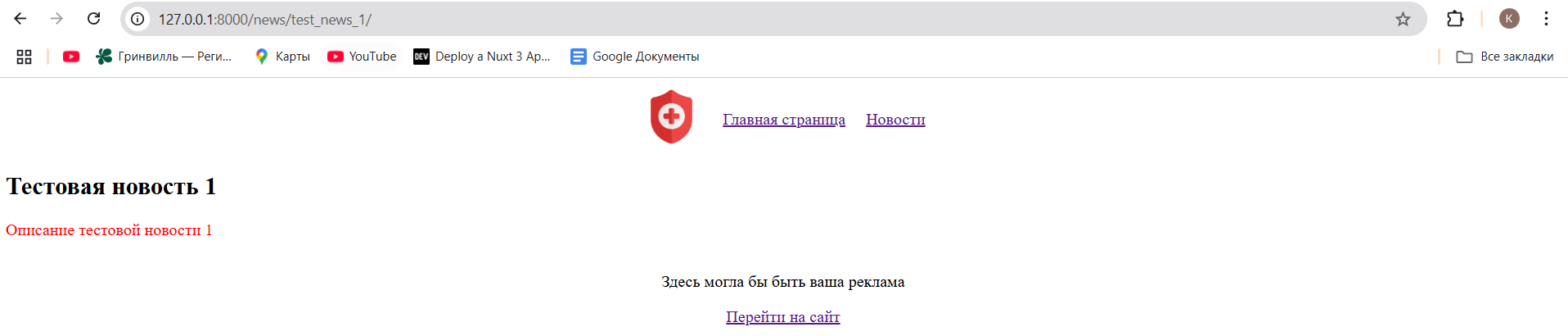
Пример данных в таблице

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

5. Добавить в БД слаги и обеспечить отображение записей по их слагу.

Поля модели были представлены выше, пример обращения по адресу со слагом



6-7. Создать пользовательский менеджер модели и использовать в программе класс перечисления.

Все новости перед публикацией на сайт должны проходить контроль администратора сайта. Добавим поле статус для фиксирования статуса «Черновик» или «Опубликовано», так же добавим менеджер модели который будет сразу фильтровать нам опубликованные новости.

class PublishedManager(models.Manager):

    def get\_queryset(self):

        return super().get\_queryset().filter(status=News.Status.PUBLISHED)

class News(models.Model):

    class Status(models.TextChoices):

        DRAFT = 'draft', 'Черновик'

        PUBLISHED = 'published', 'Опубликовано'

    name = models.CharField(verbose\_name="Название", max\_length=127)

    slug = models.CharField(verbose\_name="Слаг", max\_length=127)

    description = models.TextField(verbose\_name="Название")

    status = models.CharField(

        max\_length=127,

        choices=Status.choices,

        default=Status.DRAFT,

        verbose\_name="Статус"

    )

    objects = models.Manager()

    published = PublishedManager()

    def \_\_str\_\_(self):

        return f"{self.name}"

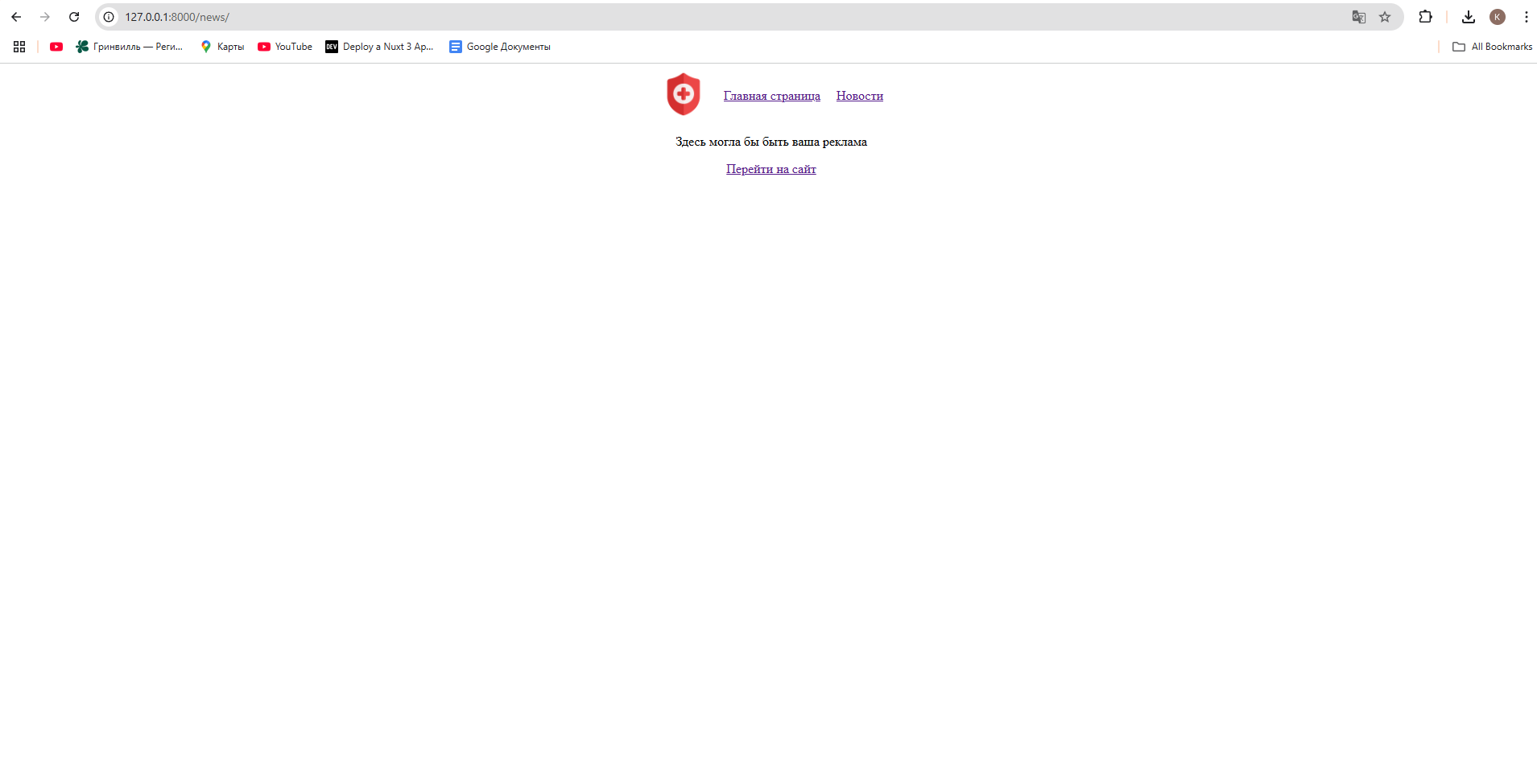
    class Meta:

        verbose\_name = "Новость"

        verbose\_name\_plural = "Новости"

Также во view изменим получение записей  
news = News.published.all()

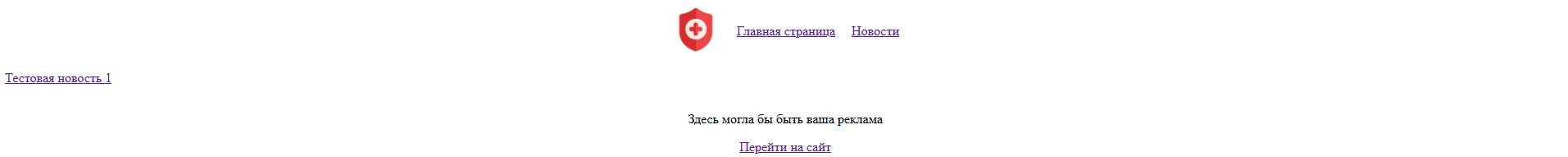
Сейчас страница новостей пуста



Поменяем статус с draft на published в базе данных

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Новость появилась на странице новостей  


# Вывод

# В результате выполнения работы была достигнута цель — изучены основы подключения и работы с базой данных в Django. С помощью фреймворка создана таблица, добавлены записи, которые успешно отобразились в DBBrowser.

# На веб-странице реализован вывод данных из БД, а также выполнены основные операции: добавление, редактирование, удаление, фильтрация и сортировка. Добавлены слаги и обеспечен их вывод в URL.

# Дополнительно реализован пользовательский менеджер для фильтрации опубликованных записей и использован перечисляемый тип для статуса.

# Листинг программы

|  |
| --- |
| main\views.py |
| from django.shortcuts import render, redirect  from django.urls import reverse  from django.http import Http404  from main.models import News  from django.shortcuts import get\_object\_or\_404  def homepage(request):      return render(request, "main.html", {})  def send\_form(request):      if (request.POST.get("name") and request.POST.get("request")):          return render(request, "success\_from.html")      else:          url\_redirect = f"{reverse('homepage')}?error=true"          return redirect(url\_redirect)    def news(request):      news = News.published.all()      return render(request, "news.html", {"news" : news})  def news\_detail(request, news\_slug):      news\_el = get\_object\_or\_404(News, slug=news\_slug)      return render(request, "news-detail.html", {"elem": news\_el })  def not\_found(request):      return render(request, "404.html") |

|  |
| --- |
| main\models.py |
| from django.db import models  class PublishedManager(models.Manager):      def get\_queryset(self):          return super().get\_queryset().filter(status=News.Status.PUBLISHED)    class News(models.Model):      class Status(models.TextChoices):          DRAFT = 'draft', 'Черновик'          PUBLISHED = 'published', 'Опубликовано'      name = models.CharField(verbose\_name="Название", max\_length=127)      slug = models.CharField(verbose\_name="Слаг", max\_length=127)      description = models.TextField(verbose\_name="Название")      status = models.CharField(          max\_length=127,          choices=Status.choices,          default=Status.DRAFT,          verbose\_name="Статус"      )      objects = models.Manager()      published = PublishedManager()      def \_\_str\_\_(self):          return f"{self.name}"      class Meta:          verbose\_name = "Новость"          verbose\_name\_plural = "Новости" |

|  |
| --- |
| main\urls.py |
| from django.urls import path, register\_converter  from main.views import (      homepage,      send\_form,      news,      news\_detail,      not\_found  )  from main.converters import (      OneDigitYearConverter,  )  handler404 = not\_found  register\_converter(OneDigitYearConverter,"onedigit")  urlpatterns = [      path('', homepage, name='homepage'),      path('send\_form', send\_form, name='send\_form'),      path('news/', news, name='news'),      path('news/<slug:news\_slug>/', news\_detail, name='news\_detail'),  ] |

|  |
| --- |
| news\_api\views.py |
| import json  from django.http import JsonResponse, HttpResponseBadRequest, HttpResponseNotAllowed, HttpResponseNotFound  from django.views.decorators.csrf import csrf\_exempt  from main.models import News  @csrf\_exempt  def api\_news\_list\_create(request):      if request.method == 'GET':          name\_filter = request.GET.get('name')          ordering = request.GET.get('ordering')          news = News.objects.all()          if name\_filter:              news = news.filter(name\_\_icontains=name\_filter)          if ordering:              news = news.order\_by(ordering)          data = [              {                  'id': n.id,                  'name': n.name,                  'slug': n.slug,                  'description': n.description              } for n in news          ]          return JsonResponse(data, safe=False)      elif request.method == 'POST':          try:              body = json.loads(request.body)              name = body.get('name')              slug = body.get('slug')              description = body.get('description')              if not all([name, slug, description]):                  return HttpResponseBadRequest('Нет всех полей')              news = News.objects.create(name=name, slug=slug, description=description)              return JsonResponse({                  'id': news.id,                  'name': news.name,                  'slug': news.slug,                  'description': news.description              }, status=201)          except json.JSONDecodeError:              return HttpResponseBadRequest('Invalid JSON')      else:          return HttpResponseNotAllowed(['GET', 'POST'])  @csrf\_exempt  def api\_news\_detail\_update\_delete(request, pk):      try:          news = News.objects.get(pk=pk)      except News.DoesNotExist:          return HttpResponseNotFound('News not found')      if request.method == 'GET':          return JsonResponse({              'id': news.id,              'name': news.name,              'slug': news.slug,              'description': news.description          })      elif request.method == 'PUT':          try:              body = json.loads(request.body)              news.name = body.get('name', news.name)              news.slug = body.get('slug', news.slug)              news.description = body.get('description', news.description)              news.save()              return JsonResponse({                  'id': news.id,                  'name': news.name,                  'slug': news.slug,                  'description': news.description              })          except json.JSONDecodeError:              return HttpResponseBadRequest('Invalid JSON')      elif request.method == 'DELETE':          news.delete()          return JsonResponse({'status': 'deleted'}, status=204)      else:          return HttpResponseNotAllowed(['GET', 'PUT', 'DELETE']) |

|  |
| --- |
| news\_api\urls.py |
| from django.urls import path  from news\_api import views  urlpatterns = [      path('api/news/', views.api\_news\_list\_create, name='api\_news\_list\_create'),      path('api/news/<int:pk>/', views.api\_news\_detail\_update\_delete, name='api\_news\_detail\_update\_delete'),  ] |

|  |
| --- |
| add\_data.py |
| import os  import django  os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'healthcare.settings')  django.setup()  from main.models import News  news\_mock = {      1: {          "name": "Тестовая новость 1",          "slug": "test\_news\_1",          "desc": "<p style='color:red'>Описание тестовой новости 1</p>"      },      2: {          "name": "Тестовая новость 2",          "slug": "test\_news\_2",          "desc": "Описание тестовой новости 2"      },      3: {          "name": "Тестовая новость 3",          "slug": "test\_news\_3",          "desc": "Описание тестовой новости 3"      }  }  for item in news\_mock.values():      name = item["name"]      desc = item["desc"]      slug = item["slug"]      news = News.objects.create(          name=name,          slug=slug,          description=desc      ) |