МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления



**ОТЧЁТ**

**по лабораторной работе №10**

**на тему:**

**Работа с формами**

**по дисциплине:** Разработка программных приложений и WEB-программирование

Выполнили работу:

Студенты гр. АП-226, АВТФ

Ищенко С. Г.

«26» апреля 2025 г.

Проверил работу:

Эстрайх И. В.

«\_\_\_» \_\_\_\_\_\_ 20\_\_г

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Цель**

Изучить работу с формами.

**Задание**

1. Создать форму для добавления записи, несвязанную с моделью.

Проверить работу формы на корректных и некорректных данных с

использованием стандартных валидаторов. Написать собственный

валидатор, проверить работу формы на нем.

2. Создать форму для добавления записи, связанную с моделью.

Проверить работу формы на корректных и некорректных данных с

использованием стандартных валидаторов. Написать собственный

валидатор, проверить работу формы на нем.

3. Реализовать загрузку файлов на сервер. Использовать генерацию

случайных имен для обеспечения возможности загрузки файлов с

одинаковыми исходными именами.

4. Добавить изображения к записям в БД. Добавить соответствующее

поле в форму добавления записи. Проверить работу формы.

Обеспечить отображение изображений на сайте и в админ-панели.

**Описание хода выполнения работы**

1. Создать форму для добавления записи, несвязанную с моделью.

Проверить работу формы на корректных и некорректных данных с

использованием стандартных валидаторов. Написать собственный

валидатор, проверить работу формы на нем.

Для начала создадим файл формы cs/forms.py

*from* django *import* forms

*from* .models *import* ComputerScienceConcept, FieldOfStudy, Tag

*# 1. Собственный валидатор: запрет цифр в названии*

def validate\_title\_no\_digits(*value*):

*if* any(ch.isdigit() *for* ch *in* value):

*raise* forms.ValidationError(

            'Название не должно содержать цифр',

*code*='no\_digits'

        )

*# 2. Собственный валидатор: запрет слова "test"*

def validate\_no\_test(*value*):

*if* 'test' in value.lower():

*raise* forms.ValidationError(

            'Слово "test" запрещено в названии',

*code*='no\_test'

        )

*# Несвязанная с моделью форма*

class ConceptForm(forms.Form):

    title = forms.CharField(

*max\_length*=255,

*label*='Название концепции компьютерных наук',

*validators*=[validate\_title\_no\_digits],

*help\_text*='До 255 символов, без цифр'

    )

    description = forms.CharField(

*widget*=forms.Textarea,

*required*=False,

*label*='Описание',

*help\_text*='Дополнительная информация (необязательно)'

    )

    difficulty = forms.IntegerField(

*label*='Сложность (от 1 до 5)',

*min\_value*=1,

*max\_value*=5,

*help\_text*='Число от 1 до 5'

    )

    image = forms.ImageField(

*required*=False,

*label*='Изображение',

*help\_text*='JPG/PNG до 5 МБ'

    )

*# Форма, связанная с моделью*

class ConceptModelForm(forms.ModelForm):

*# Переопределяем title, чтобы применить оба валидатора*

    title = forms.CharField(

*max\_length*=255,

*label*='Название концепции компьютерных наук',

*validators*=[validate\_no\_test, validate\_title\_no\_digits],

*help\_text*='Без цифр и слова "test"'

    )

    class Meta:

        model = ComputerScienceConcept

        fields = [

            'title',

            'slug',

            'description',

            'difficulty',

            'field\_of\_study',

            'tags',

            'image',

        ]

        widgets = {

            'description': forms.Textarea,

        }

class UploadForm(forms.Form):

    file = forms.FileField(

*label*='Выберите файл',

*help\_text*='Максимум 10 МБ. Любой тип.'

    )

    def clean\_file(*self*):

        f = self.cleaned\_data['file']

*if* f.size > 10 \* 1024 \* 1024:

*raise* forms.ValidationError('Файл слишком большой (больше 10 МБ)')

*return* f

Здесь мы:

* Определили три поля: title, description, difficulty.
* Добавили стандартные валидаторы (max\_length, min\_value и т. д.).
* Написали собственный валидатор validate\_title\_no\_digits, который выбрасывает ValidationError, если в строке есть цифра.
* И валидатор validate\_no\_test, который запрещает использовать слово "test" в названии.

Далее нужно описать представление и добавить в файл cs/views.py

*from* .forms *import* ConceptForm, ConceptModelForm, UploadForm

def add\_concept\_custom(*request*):

    """

    Пункт 1: форма ConceptForm (несвязанная с моделью) с поддержкой загрузки изображения.

    """

*if* request.method == 'POST':

        form = ConceptForm(request.POST, request.FILES)

*if* form.is\_valid():

            cd = form.cleaned\_data

            concept = ComputerScienceConcept.objects.create(

*title*=cd['title'],

*slug*=cd['title'].lower().replace(' ', '-'),

*description*=cd['description'],

*difficulty*=cd['difficulty'],

*is\_published*=ComputerScienceConcept.Status.DRAFT

            )

*# Сохраняем изображение, если пользователь его загрузил*

            image = cd.get('image')

*if* image:

                concept.image = image

                concept.save()

*return* redirect(concept.get\_absolute\_url())

*else*:

        form = ConceptForm()

*return* render(request, 'cs/add\_concept\_custom.html', {'form': form})

Теперь нужно прописать маршрут в файле urls.py:

*from* .views *import* add\_concept\_custom, add\_concept\_model, upload\_file

path('add-custom/', add\_concept\_custom, *name*='add\_concept\_custom'),

После этого форма будет доступна в браузере.

Далее создадим шаблон.

**Файл:** cs/templates/cs/add\_concept\_custom.html

{% extends 'cs/base.html' %}

{% block content %}

  <h1>Добавить концепцию (каркасная форма)</h1>

  <form method="post" enctype="multipart/form-data">{% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.title.label\_tag }}<br>

      {{ form.title }}<br>

      {{ form.title.help\_text }}<br>

      {{ form.title.errors }}

    </p>

    <p>

      {{ form.description.label\_tag }}<br>

      {{ form.description }}<br>

      {{ form.description.help\_text }}<br>

      {{ form.description.errors }}

    </p>

    <p>

      {{ form.difficulty.label\_tag }}<br>

      {{ form.difficulty }}<br>

      {{ form.difficulty.help\_text }}<br>

      {{ form.difficulty.errors }}

    </p>

    <p>

      {{ form.image.label\_tag }}<br>

      {{ form.image }}<br>

      {{ form.image.help\_text }}<br>

      {{ form.image.errors }}

    </p>

    <button type="submit">Сохранить</button>

  </form>

{% endblock %}

Чтобы было видно, как и где выводятся ошибки:

* {{ form.non\_field\_errors }}

Теперь у нас есть рабочая форма по адресу /add-custom/

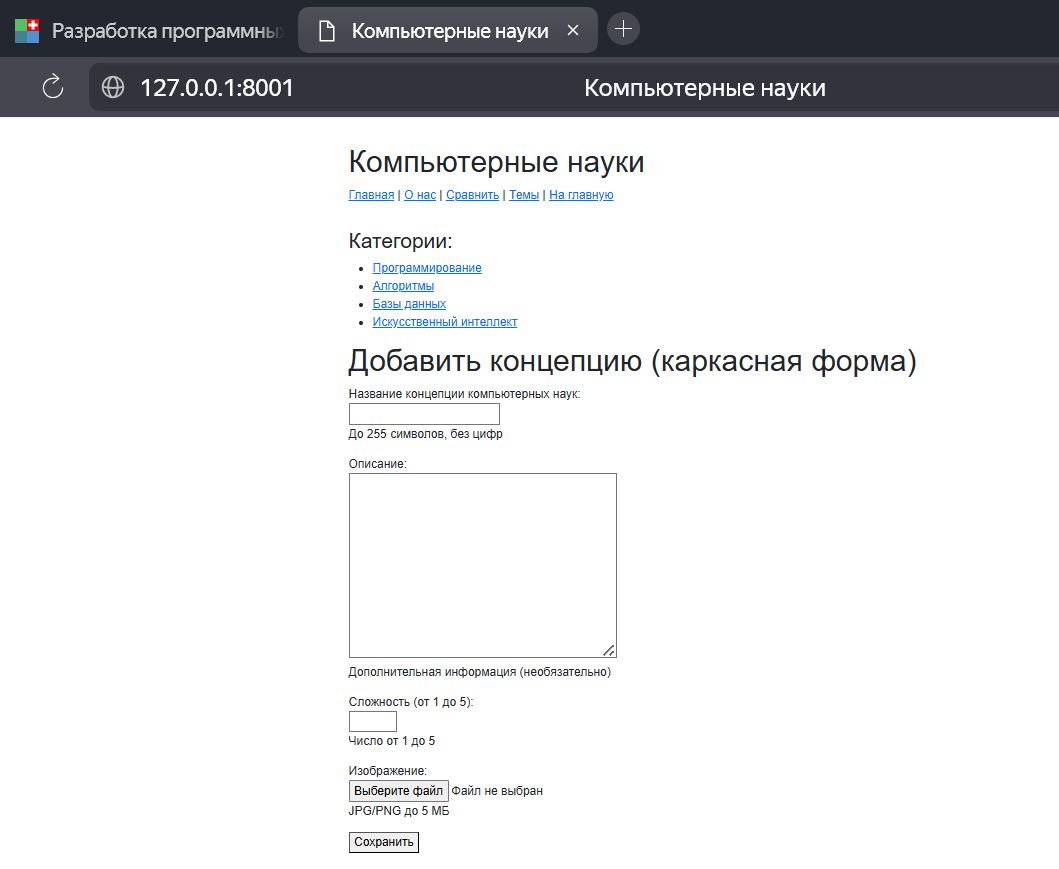


Рисунок 1 – Форма add-custom.

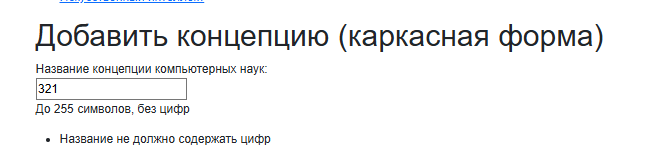
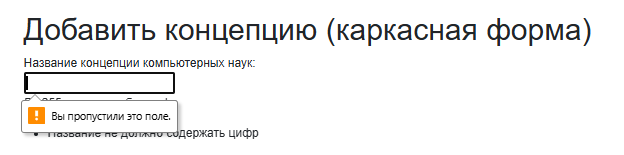


Рисунок 2 – Валидатор на наличие цифр.

  
Рисунок 3 – Валидатор на заполненность поля.

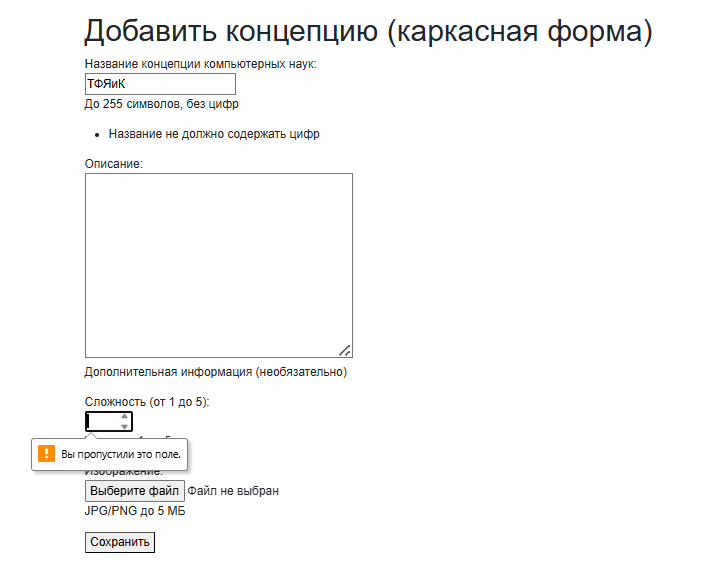


Рисунок 4 – Валидатор на заполненность поля.

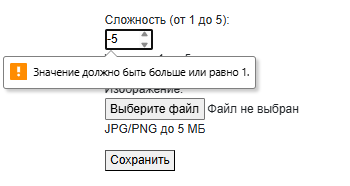


Рисунок 5 – Валидатор отрицательной сложности

2.Создать форму для добавления записи, связанную с моделью. Проверить работу формы на корректных и некорректных данных с использованием стандартных валидаторов. Написать собственный валидатор, проверить работу формы на нем.

Для этого добавим новое представление add\_concept\_model в файл views.py

def add\_concept\_model(*request*):

    """

    Пункт 2: форма ConceptModelForm (ModelForm) с поддержкой загрузки изображения.

    """

*if* request.method == 'POST':

        form = ConceptModelForm(request.POST, request.FILES)

*if* form.is\_valid():

            concept = form.save(*commit*=False)

            concept.is\_published = ComputerScienceConcept.Status.DRAFT

            concept.save()

            form.save\_m2m() *# сохраняем M2M-поля (tags) и image*

*return* redirect(concept.get\_absolute\_url())

*else*:

        form = ConceptModelForm()

*return* render(request, 'cs/add\_concept\_model.html', {'form': form})

Добавим новый url в файл urls.py

path('add-model/',  add\_concept\_model,  *name*='add\_concept\_model'),

И создадим новый шаблон add\_concept\_model.html

{% extends "cs/base.html" %}

{% block content %}

  <h1>Добавить концепцию (ModelForm)</h1>

  <form method="post" enctype="multipart/form-data">{% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.title.label\_tag }}<br>

      {{ form.title }}<br>

      <small>{{ form.title.help\_text }}</small><br>

      {{ form.title.errors }}

    </p>

    <p>

      {{ form.slug.label\_tag }}<br>

      {{ form.slug }}<br>

      {{ form.slug.errors }}

    </p>

    <p>

      {{ form.description.label\_tag }}<br>

      {{ form.description }}<br>

      {{ form.description.errors }}

    </p>

    <p>

      {{ form.difficulty.label\_tag }}<br>

      {{ form.difficulty }}<br>

      {{ form.difficulty.errors }}

    </p>

    <p>

      {{ form.field\_of\_study.label\_tag }}<br>

      {{ form.field\_of\_study }}<br>

      {{ form.field\_of\_study.errors }}

    </p>

    <p>

      {{ form.tags.label\_tag }}<br>

      {{ form.tags }}<br>

      {{ form.tags.errors }}

    </p>

    <p>

      {{ form.image.label\_tag }}<br>

      {{ form.image }}<br>

      {{ form.image.help\_text }}<br>

      {{ form.image.errors }}

    </p>

    <button type="submit">Сохранить</button>

  </form>

{% endblock %}

После этого получим готовую рабочую форму.

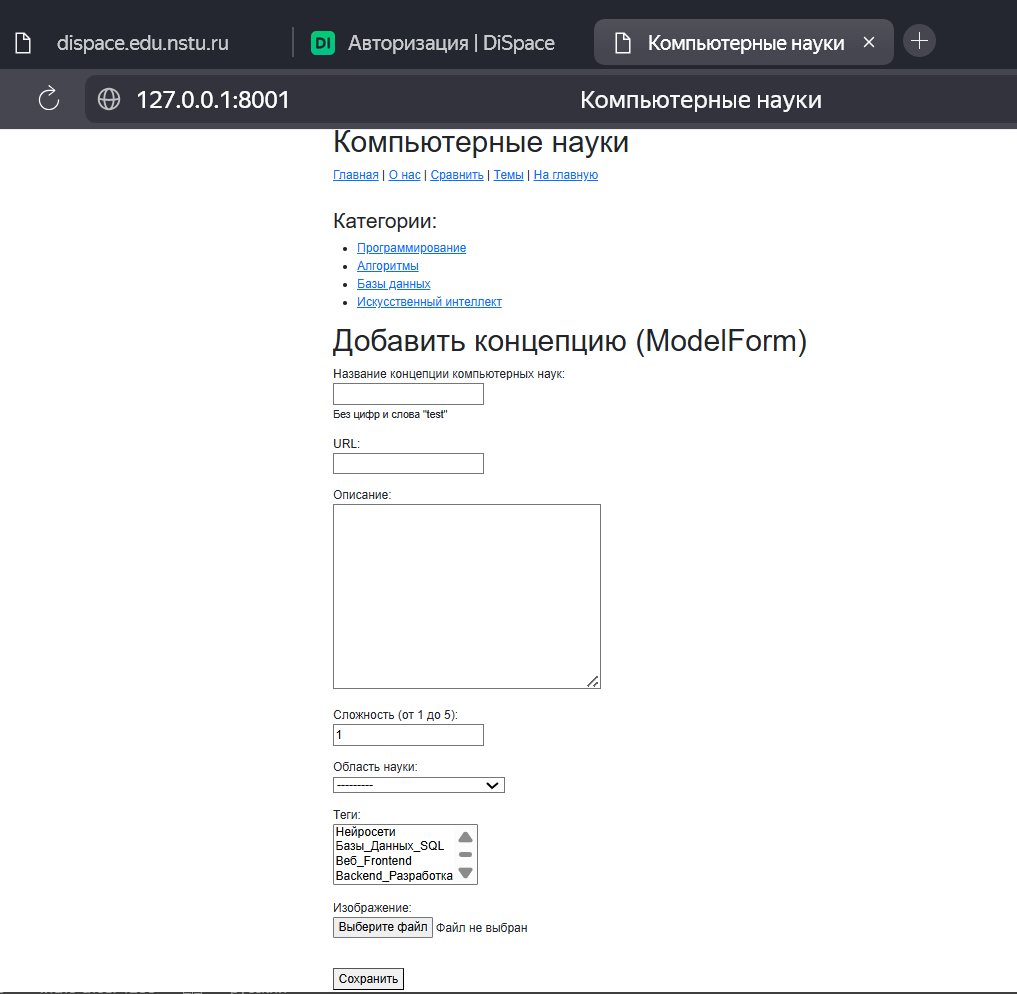


Рисунок 6 – Форма add-model.

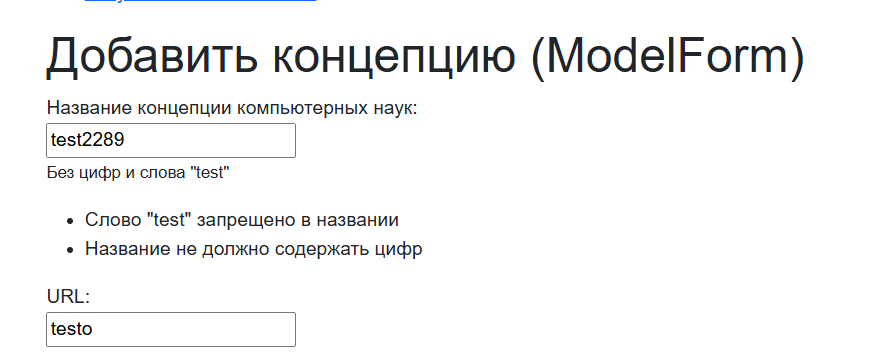
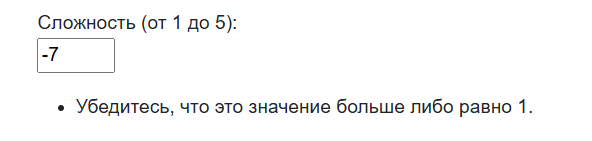
****

Рисунок 7 – Проверка собственных валидаторов.

  
Рисунок 8 – Проверка стандартного валидатора на не отрицательность сложности.

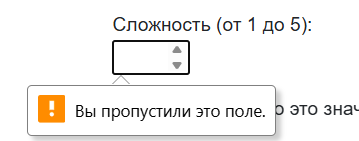


Рисунок 9 – Проверка на заполненность поля.

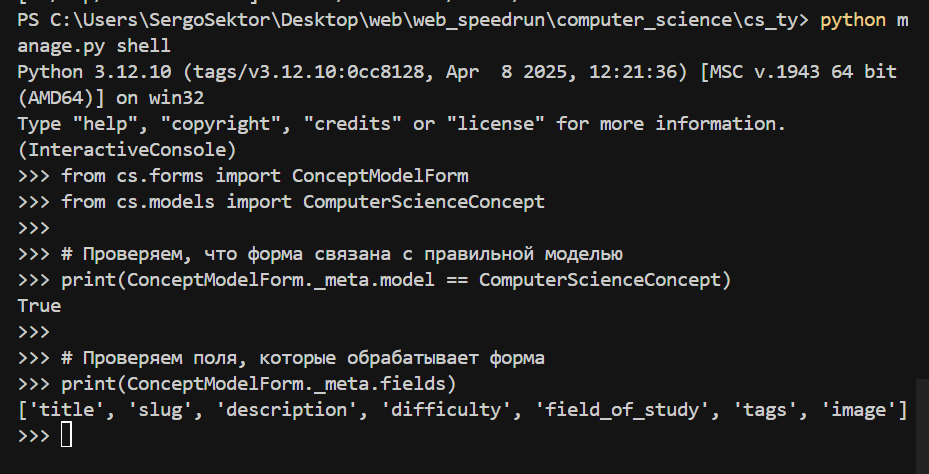


Рисунок 10 – Проверка того, что форма связана с моделью.

3. Реализовать загрузку файлов на сервер. Использовать генерацию

случайных имен для обеспечения возможности загрузки файлов с

одинаковыми исходными именами.

Сперва нужно добавить секцию в файл settings.py:

*# URL, по которому будут доступны загруженные файлы*

MEDIA\_URL = '/media/'

*# Директория на диске, в которую Django будет сохранять загруженные файлы*

MEDIA\_ROOT = BASE\_DIR / 'media'

Затем в файл forms.py добавим новый класс UploadForm

* FileField даёт доступ к request.FILES['file'].
* В clean\_file мы проверяем размер .

class UploadForm(forms.Form):

    file = forms.FileField(

*label*='Выберите файл',

*help\_text*='Максимум 10 МБ. Любой тип.'

    )

    def clean\_file(*self*):

        f = self.cleaned\_data['file']

*if* f.size > 10 \* 1024 \* 1024:

*raise* forms.ValidationError('Файл слишком большой (больше 10 МБ)')

*return* f

В файл views.py добавляем представление:

def upload\_file(*request*):

    """

    Обрабатывает UploadForm, сохраняет файл с случайным именем

    и выводит пользователю ссылку на загруженный файл.

    """

    link = None

*if* request.method == 'POST':

        form = UploadForm(request.POST, request.FILES)

*if* form.is\_valid():

            f = form.cleaned\_data['file']

*# Генерируем новое имя: uuid4 + оригинальное расширение*

            ext = os.path.splitext(f.name)[1]  *# включая точку, например ".png"*

            new\_name = f"{uuid.uuid4().hex}{ext}"

*# Сохраняем файл в MEDIA\_ROOT*

            path = os.path.join('uploads', new\_name)  *# в папке media/uploads/*

            saved\_path = default\_storage.save(path, f)

*# Формируем URL, по которому файл будет доступен*

            link = settings.MEDIA\_URL + saved\_path

*else*:

        form = UploadForm()

*return* render(request, 'cs/upload.html', {

        'form': form,

        'link': link,

    })

Мы сохраняем файлы в подпапку media/uploads/.

default\_storage.save автоматически пишет файл и не перезаписывает, если имя новое.

В urls.py нужно добавить новый маршрут и подключить раздачу медиа-файлов:

*from* django.urls *import* path

*from* . *import* views

*from* . *import* converters

*from* django.urls *import* register\_converter

*from* django.contrib *import* admin

*from* .views *import* add\_concept\_custom, add\_concept\_model, upload\_file

*from* django.conf *import* settings

*from* django.conf.urls.static *import* static

admin.site.site\_header = "Панель Администрирования"

admin.site.index\_title = "Управление сайтом"

admin.site.site\_title = "Администрирование MyCSProject"

register\_converter(converters.FourDigitYearConverter, 'year4')

urlpatterns = [

    path('', views.index, *name*='home'),

    path('about/', views.about, *name*='about'),

    path('compare/', views.compare, *name*='compare'),

    path('concepts/', views.concepts\_list, *name*='concepts'),

    path('concepts/<int:concept\_id>/', views.concept\_detail, *name*='concept\_detail'),

    path('archive/<year4:year>/', views.archive, *name*='archive'),

    path('go-home/', views.redirect\_example, *name*='go\_home'),

    path('hello/', views.get\_params\_example, *name*='hello'),

    path('category/<int:cat\_id>/', views.category, *name*='category'),

    path('concept/<slug:concept\_slug>/', views.concept\_detail, *name*='concept\_detail'),

    path('tags/', views.tags\_list, *name*='tags\_list'),

    path('admin/', admin.site.urls),

    path('add-custom/', add\_concept\_custom, *name*='add\_concept\_custom'),

    path('add-model/',  add\_concept\_model,  *name*='add\_concept\_model'),

    path('upload/', upload\_file, *name*='upload\_file'),

]

*if* settings.DEBUG:

    urlpatterns += static(settings.MEDIA\_URL, *document\_root*=settings.MEDIA\_ROOT)

*# handler404 = 'cs.views.custom\_404'*

Затем нужно создать шаблон upload.html

{% extends "cs/base.html" %}

{% block content %}

  <h1>Загрузка файла</h1>

  <form method="post" enctype="multipart/form-data">

    {% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.file.label\_tag }}<br>

      {{ form.file }}<br>

      <small>{{ form.file.help\_text }}</small><br>

      {{ form.file.errors }}

    </p>

    <button type="submit">Загрузить</button>

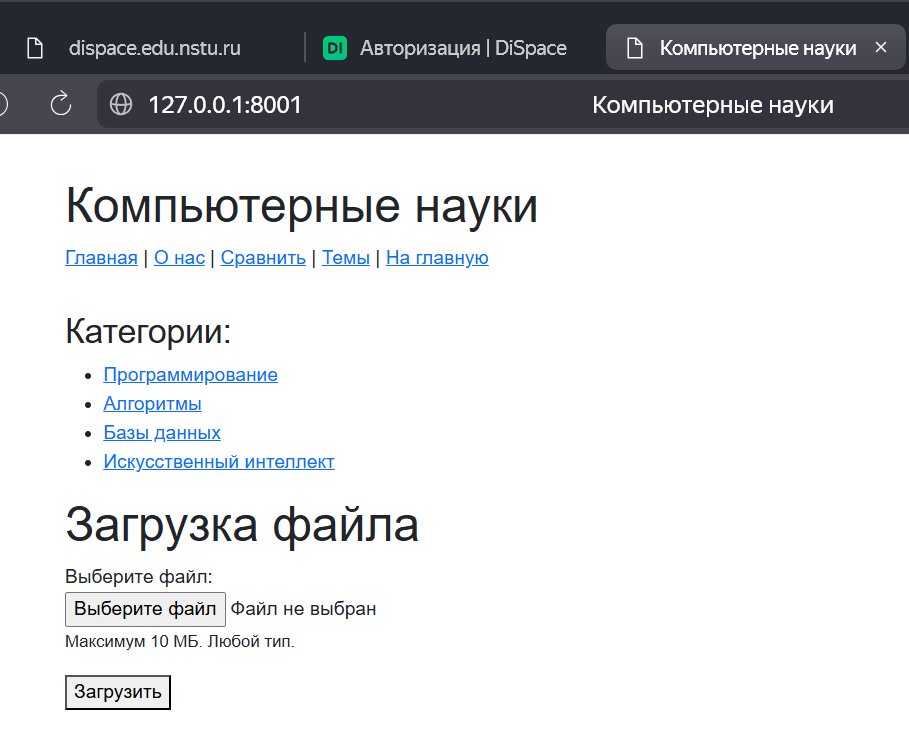
  </form>

  {% if link %}

    <p>Файл успешно загружен: <a href="{{ link }}" target="\_blank">{{ link }}</a></p>

  {% endif %}

{% endblock %}

  
Рисунок 11 – Форма Upload.

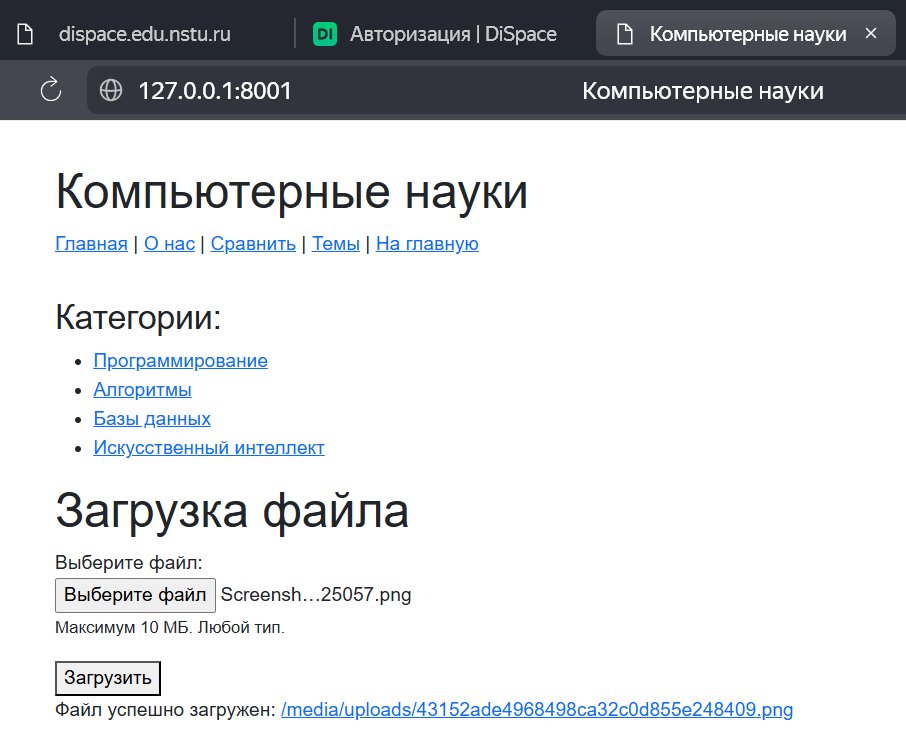


Рисунок 12 – Загрузка файла.

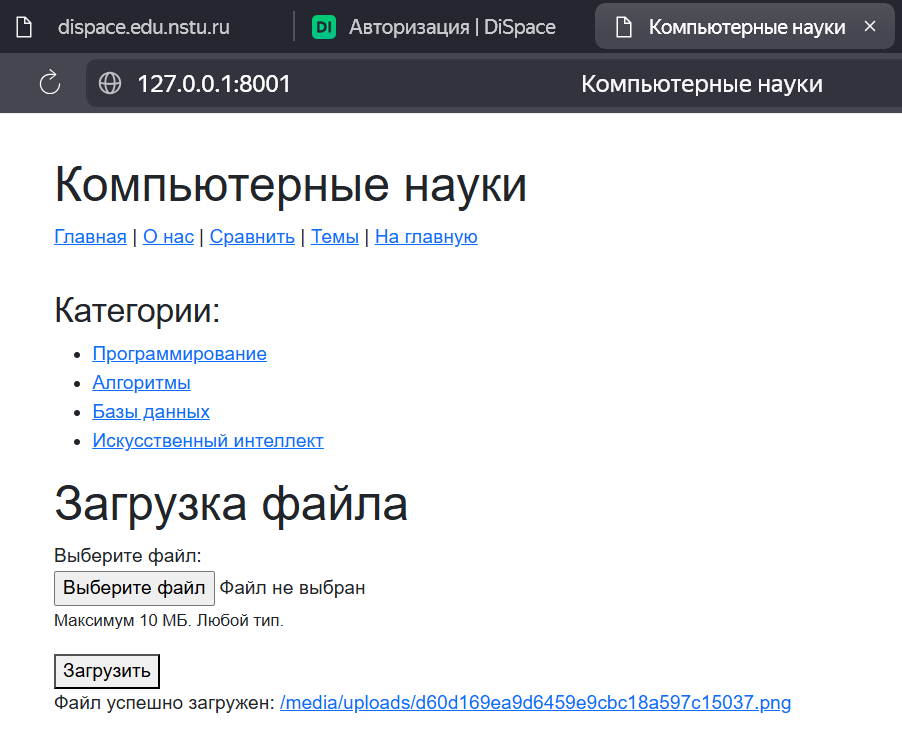


Рисунок 13 – Загрузка того же файла.

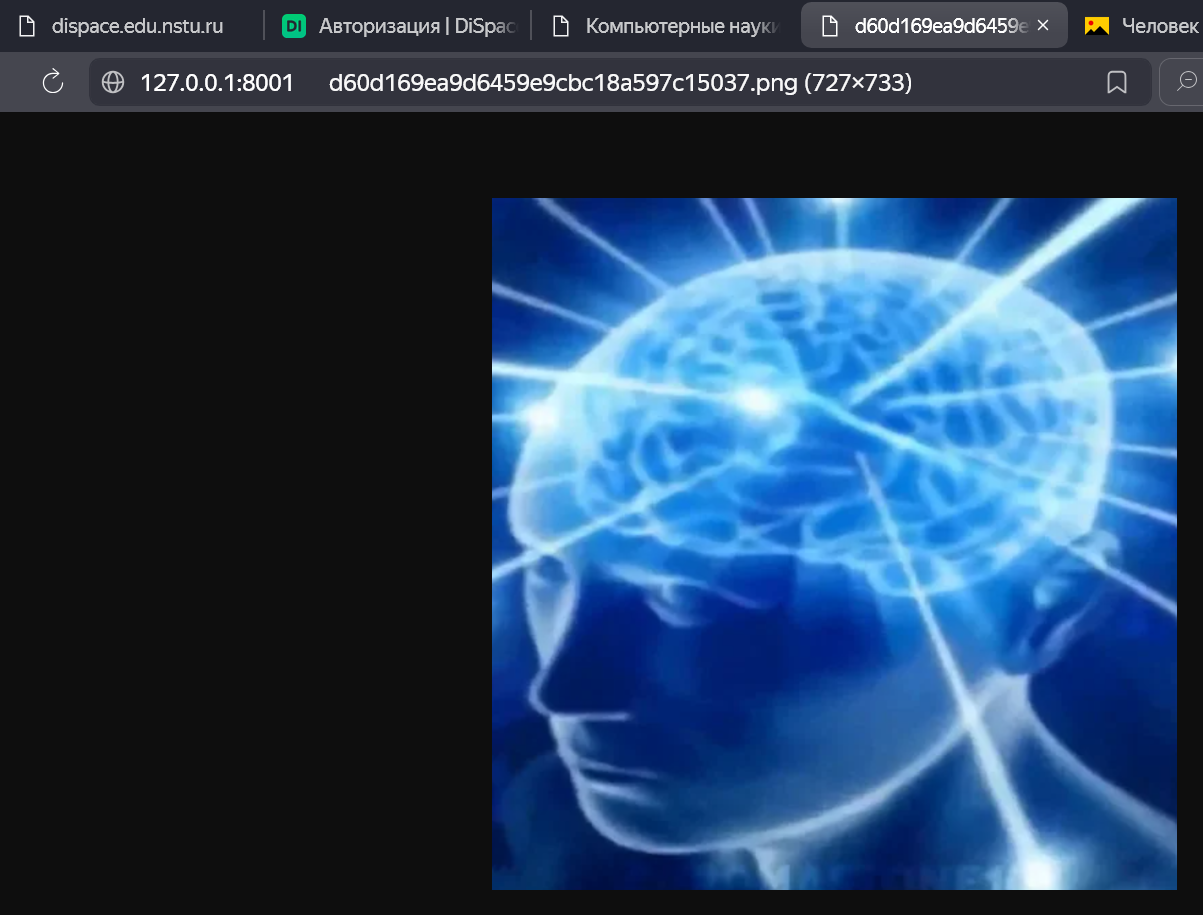


Рисунок 14 – Проверка ссылки для скачивания.

4. Добавить изображения к записям в БД. Добавить соответствующее

поле в форму добавления записи. Проверить работу формы.

Обеспечить отображение изображений на сайте и в админ-панели.

В файле forms.py в класс ConceptForm и ConceptModelForm нужно добавить поле image

image = forms.ImageField(

*required*=False,

*label*='Изображение',

*help\_text*='JPG/PNG до 5 МБ'

    )

Теперь нужно пропустить request.FILES в представлениях

*from* django.shortcuts *import* render, redirect, get\_object\_or\_404

*from* django.http *import* Http404, HttpResponse

*from* django.conf *import* settings

*from* django.core.files.storage *import* default\_storage

*import* os

*import* uuid

*from* .models *import* ComputerScienceConcept, Tag, FieldOfStudy *# Updated model import*

*from* .forms *import* ConceptForm, ConceptModelForm, UploadForm *# Updated import*

def index(*request*):

    concepts = ComputerScienceConcept.published.all()

*return* render(request, 'cs/index.html', {

        'title': 'Главная страница',

        'concepts': concepts,

    })

def about(*request*):

*return* render(request, 'cs/about.html')

def compare(*request*):

*return* render(request, 'cs/compare.html')

def concepts\_list(*request*):

    all\_concepts = [

        {'id': 1, 'name': 'Data Structures', 'description': 'Fundamental ways to organize data.'},

        {'id': 2, 'name': 'Algorithms', 'description': 'Step-by-step procedures for calculations.'},

    ]

*return* render(request, 'cs/cs\_list.html', {'concepts': all\_concepts})

def concept\_detail(*request*, *concept\_slug*):

    concept = get\_object\_or\_404(ComputerScienceConcept, *slug*=concept\_slug)

*return* render(request, 'cs/concept\_detail.html', {

        'title': concept.title,

        'concept': concept,

    })

def archive(*request*, *year*):

*if* year > 2025:

*raise* Http404("Нет данных за этот год")

*return* HttpResponse(f"<h1>Архив за {year} год</h1>")

def redirect\_example(*request*):

*return* redirect('home')

def get\_params\_example(*request*):

    name = request.GET.get('name', 'Гость')

*return* HttpResponse(f"<h1>Привет, {name}!</h1>")

def category(*request*, *cat\_id*):

*return* HttpResponse(f"<h1>Страница категории</h1><p>ID категории: {cat\_id}</p>")

def tags\_list(*request*):

    tags = Tag.objects.all()

*return* render(request, 'cs/tags\_list.html', {

        'title': 'Теги',

        'tags': tags,

    })

def add\_concept\_custom(*request*):

    """

    Пункт 1: форма ConceptForm (несвязанная с моделью) с поддержкой загрузки изображения.

    """

*if* request.method == 'POST':

        form = ConceptForm(request.POST, request.FILES)

*if* form.is\_valid():

            cd = form.cleaned\_data

            concept = ComputerScienceConcept.objects.create(

*title*=cd['title'],

*slug*=cd['title'].lower().replace(' ', '-'),

*description*=cd['description'],

*difficulty*=cd['difficulty'],

*is\_published*=ComputerScienceConcept.Status.DRAFT

            )

*# Сохраняем изображение, если пользователь его загрузил*

            image = cd.get('image')

*if* image:

                concept.image = image

                concept.save()

*return* redirect(concept.get\_absolute\_url())

*else*:

        form = ConceptForm()

*return* render(request, 'cs/add\_concept\_custom.html', {'form': form})

    def add\_concept\_model(*request*):

        """

        Пункт 2: форма ConceptModelForm (ModelForm) с поддержкой загрузки изображения.

        """

*if* request.method == 'POST':

            form = ConceptModelForm(request.POST, request.FILES)

*if* form.is\_valid():

                concept = form.save(*commit*=False)

                concept.is\_published = ComputerScienceConcept.Status.DRAFT

                concept.save()

                form.save\_m2m() *# сохраняем M2M-поля (tags) и image*

*return* redirect(concept.get\_absolute\_url())

*else*:

            form = ConceptModelForm()

*return* render(request, 'cs/add\_concept\_model.html', {'form': form})

def upload\_file(*request*):

    """

    Обрабатывает UploadForm, сохраняет файл с случайным именем

    и выводит пользователю ссылку на загруженный файл.

    """

    link = None

*if* request.method == 'POST':

        form = UploadForm(request.POST, request.FILES)

*if* form.is\_valid():

            f = form.cleaned\_data['file']

*# Генерируем новое имя: uuid4 + оригинальное расширение*

            ext = os.path.splitext(f.name)[1]  *# включая точку, например ".png"*

            new\_name = f"{uuid.uuid4().hex}{ext}"

*# Сохраняем файл в MEDIA\_ROOT*

            path = os.path.join('uploads', new\_name)  *# в папке media/uploads/*

            saved\_path = default\_storage.save(path, f)

*# Формируем URL, по которому файл будет доступен*

            link = settings.MEDIA\_URL + saved\_path

*else*:

        form = UploadForm()

*return* render(request, 'cs/upload.html', {

        'form': form,

        'link': link,

    })

* В add\_concept\_custom и add\_concept\_model к конструктору форм добавлен второй аргумент request.FILES, чтобы файлы передавались в форму.
* В add\_concept\_custom после создания Concept сохраняется загруженное изображение из form.cleaned\_data['image'].
* В add\_concept\_model при form.save() теперь автоматически сохраняется поле image.

Теперь в шаблоны нужно добавить поле и enctype

add\_concept\_custom.html:

{% extends 'cs/base.html' %}

{% block content %}

  <h1>Добавить концепцию (каркасная форма)</h1>

  <form method="post" enctype="multipart/form-data">{% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.title.label\_tag }}<br>

      {{ form.title }}<br>

      {{ form.title.help\_text }}<br>

      {{ form.title.errors }}

    </p>

    <p>

      {{ form.description.label\_tag }}<br>

      {{ form.description }}<br>

      {{ form.description.help\_text }}<br>

      {{ form.description.errors }}

    </p>

    <p>

      {{ form.difficulty.label\_tag }}<br>

      {{ form.difficulty }}<br>

      {{ form.difficulty.help\_text }}<br>

      {{ form.difficulty.errors }}

    </p>

    <p>

      {{ form.image.label\_tag }}<br>

      {{ form.image }}<br>

      {{ form.image.help\_text }}<br>

      {{ form.image.errors }}

    </p>

    <button type="submit">Сохранить</button>

  </form>

{% endblock %}

add\_concept\_model.html:

{% extends "cs/base.html" %}

{% block content %}

  <h1>Добавить концепцию (ModelForm)</h1>

  <form method="post" enctype="multipart/form-data">{% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.title.label\_tag }}<br>

      {{ form.title }}<br>

      <small>{{ form.title.help\_text }}</small><br>

      {{ form.title.errors }}

    </p>

    <p>

      {{ form.slug.label\_tag }}<br>

      {{ form.slug }}<br>

      {{ form.slug.errors }}

    </p>

    <p>

      {{ form.description.label\_tag }}<br>

      {{ form.description }}<br>

      {{ form.description.errors }}

    </p>

    <p>

      {{ form.difficulty.label\_tag }}<br>

      {{ form.difficulty }}<br>

      {{ form.difficulty.errors }}

    </p>

    <p>

      {{ form.field\_of\_study.label\_tag }}<br>

      {{ form.field\_of\_study }}<br>

      {{ form.field\_of\_study.errors }}

    </p>

    <p>

      {{ form.tags.label\_tag }}<br>

      {{ form.tags }}<br>

      {{ form.tags.errors }}

    </p>

    <p>

      {{ form.image.label\_tag }}<br>

      {{ form.image }}<br>

      {{ form.image.help\_text }}<br>

      {{ form.image.errors }}

    </p>

    <button type="submit">Сохранить</button>

  </form>

{% endblock %}

В concept\_detail.html нужно добавить вывод изображения:

{% extends 'cs/base.html' %}

{% block title %}{{ concept.title }}{% endblock %}

{% block content %}

  <h2>{{ concept.title }}</h2>

  {# Если для этой концепции загружено изображение, покажем его #}

  {% if concept.image %}

    <div style="margin-bottom: 1em;">

      <img src="{{ concept.image.url }}" alt="{{ concept.title }}" style="max-width:100%; height:auto;">

    </div>

  {% endif %}

  <p>Сложность: {{ concept.difficulty }}</p>

  {% if concept.field\_of\_study %}

    <p>Область науки: {{ concept.field\_of\_study.name }}</p>

  {% endif %}

  <div>

    {{ concept.description|linebreaks }}

  </div>

  {% if concept.tags.all %}

    <p>Теги:

      {% for tag in concept.tags.all %}

        <a href="{{ tag.get\_absolute\_url }}">{{ tag.name }}</a>{% if not forloop.last %}, {% endif %}

      {% endfor %}

    </p>

  {% endif %}

  {% if concept.detail %}

    <h3>Дополнительная информация:</h3>

    <p>Ключевые технологии: {{ concept.detail.core\_technologies }}</p>

    <p>Предварительные условия: {{ concept.detail.prerequisites }}</p>

    <p>Примерное время изучения: {{ concept.detail.estimated\_learning\_time }} часов</p>

  {% endif %}

{% endblock %}

* Добавлена проверка {% if concept.image %} — сработает только если в модели есть файл.
* Тег <img src="{{ concept.image.url }}"> берёт URL из ImageField.
* Через inline-стилика гарантируем, что картинка не выйдет за ширину контейнера.

Затем нужно обновить файл admin.py добавив

* Поле image и поле-превью image\_preview в секцию формы (fields и readonly\_fields).
* Импорт mark\_safe и реализацию метода image\_preview для отображения миниатюр.
* Подключение image для сохранённых изображений.

*from* django.contrib *import* admin, messages

*from* django.contrib.admin *import* SimpleListFilter

*from* django.utils.html *import* mark\_safe *# New import*

*from* django.db.models *import* ExpressionWrapper, F, DecimalField

*from* .models *import* ComputerScienceConcept, FieldOfStudy, ConceptDetail, Tag *# Updated model imports*

admin.site.site\_header = "Панель администрирования"

admin.site.index\_title = "Управление сайтом"

admin.site.site\_title = "Администрирование Computer Science Project"

*# Пользовательское вычисляемое поле: краткая информация (п.7)*

@admin.display(*description*="Краткая информация")

def brief\_info(*obj*):

*return* f"Описание: {len(obj.description)} символов" *if* obj.description *else* "Нет описания"

*# Пользовательское вычисляемое поле: цена с налогом (п.7)*

@admin.display(*description*="Сложность") *# Changed description*

def display\_difficulty(*obj*):

*if* obj.difficulty is not None:

*return* f"{obj.difficulty}/5"

*return* "N/A"

*# Кастомный фильтр для статуса публикации (п.9)*

class PublishedFilter(SimpleListFilter):

    title = "Статус публикации"

    parameter\_name = "pub\_status"

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ("published", "Опубликовано"),

            ("draft", "Черновик"),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* self.value() == "published":

*return* queryset.filter(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

*elif* self.value() == "draft":

*return* queryset.filter(*is\_published*=ComputerScienceConcept.Status.DRAFT)

*return* queryset

*# Дополнительный кастомный фильтр по диапазону сложности (п.9)*

class DifficultyRangeFilter(SimpleListFilter):

    title = "Диапазон сложности"

    parameter\_name = "difficulty\_range"

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ('low', 'Низкая (1-2)'),

            ('medium', 'Средняя (3)'),

            ('high', 'Высокая (4-5)'),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* self.value() == 'low':

*return* queryset.filter(*difficulty\_\_lte*=2)

*elif* self.value() == 'medium':

*return* queryset.filter(*difficulty*=3)

*elif* self.value() == 'high':

*return* queryset.filter(*difficulty\_\_gte*=4)

*return* queryset

@admin.register(ComputerScienceConcept) *# Updated model*

class ComputerScienceConceptAdmin(admin.ModelAdmin):

*# Поля для формы добавления/редактирования*

    fields = [

        'title', 'slug', 'description', 'difficulty',

        'field\_of\_study', 'tags', 'image', 'image\_preview' *# Updated fields*

    ]

    prepopulated\_fields = {"slug": ("title",)}

    readonly\_fields = ['time\_create', 'time\_update', 'image\_preview'] *# Added image\_preview*

*# Список записей*

    list\_display = (

        'id', 'title', 'field\_of\_study', 'time\_create',

        'is\_published', brief\_info, display\_difficulty, *# Updated display\_difficulty*

    )

    list\_display\_links = ('id', 'title')

    list\_editable = ('is\_published',)

    ordering = ['-time\_create', 'title']

    list\_per\_page = 5

    search\_fields = ['title', 'field\_of\_study\_\_name'] *# Updated search fields*

    list\_filter = [PublishedFilter, 'field\_of\_study', DifficultyRangeFilter] *# Updated list\_filter*

    actions = ['set\_published', 'set\_draft']

    @admin.action(*description*="Опубликовать выбранные записи")

    def set\_published(*self*, *request*, *queryset*):

        count = queryset.update(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

        self.message\_user(request, f"Статус 'Опубликовано' обновлён для {count} записей.", messages.SUCCESS)

    @admin.action(*description*="Снять с публикации выбранные записи")

    def set\_draft(*self*, *request*, *queryset*):

        count = queryset.update(*is\_published*=ComputerScienceConcept.Status.DRAFT)

        self.message\_user(request, f"{count} записей сняты с публикации.", messages.WARNING)

    @admin.display(*description*='Превью изображения') *# New method*

    def image\_preview(*self*, *obj*):

*if* obj.image:

*return* mark\_safe(f"<img src='{obj.image.url}' style='max-height:200px;' />")

*return* "(нет изображения)"

@admin.register(FieldOfStudy) *# Updated model*

class FieldOfStudyAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name', 'description') *# Updated fields*

    list\_display\_links = ('id', 'name')

    search\_fields = ('name', 'description') *# Updated search fields*

@admin.register(ConceptDetail) *# Updated model*

class ConceptDetailAdmin(admin.ModelAdmin):

    list\_display = ('id', 'concept', 'core\_technologies', 'prerequisites', 'estimated\_learning\_time') *# Updated fields*

    list\_display\_links = ('id', 'concept') *# Updated link*

    search\_fields = ('concept\_\_title', 'core\_technologies', 'prerequisites') *# Updated search fields*

@admin.register(Tag)

class TagAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name')

    list\_display\_links = ('id', 'name')

    search\_fields = ('name',)

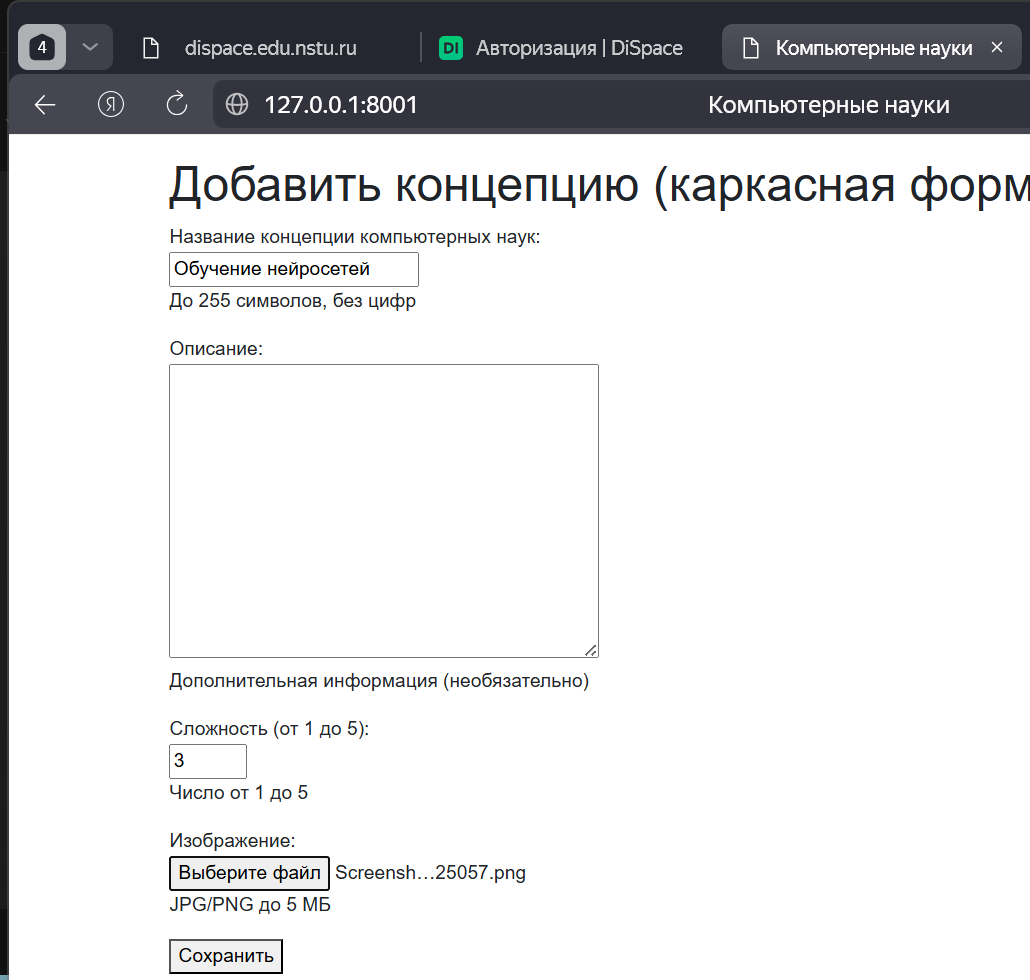


Рисунок 15 – Добавление концепции в форму на сайте.

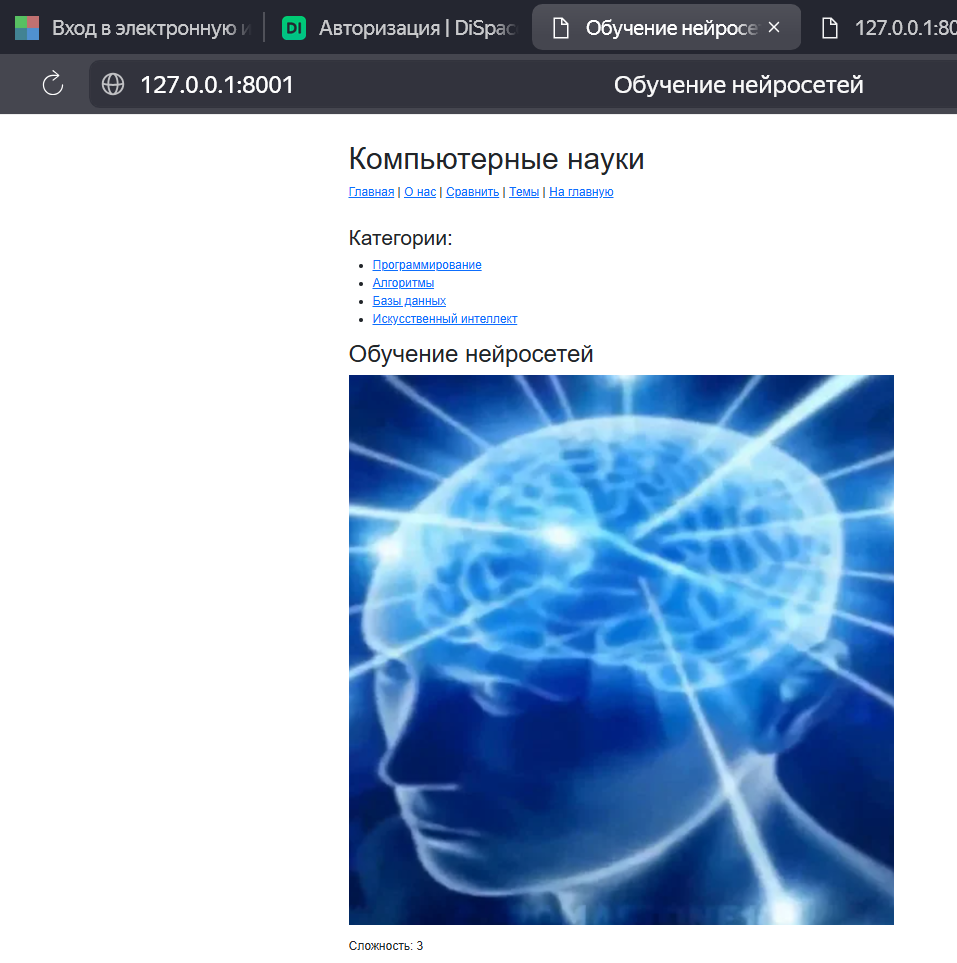


Рисунок 16 – Отображение концепции на сайте.

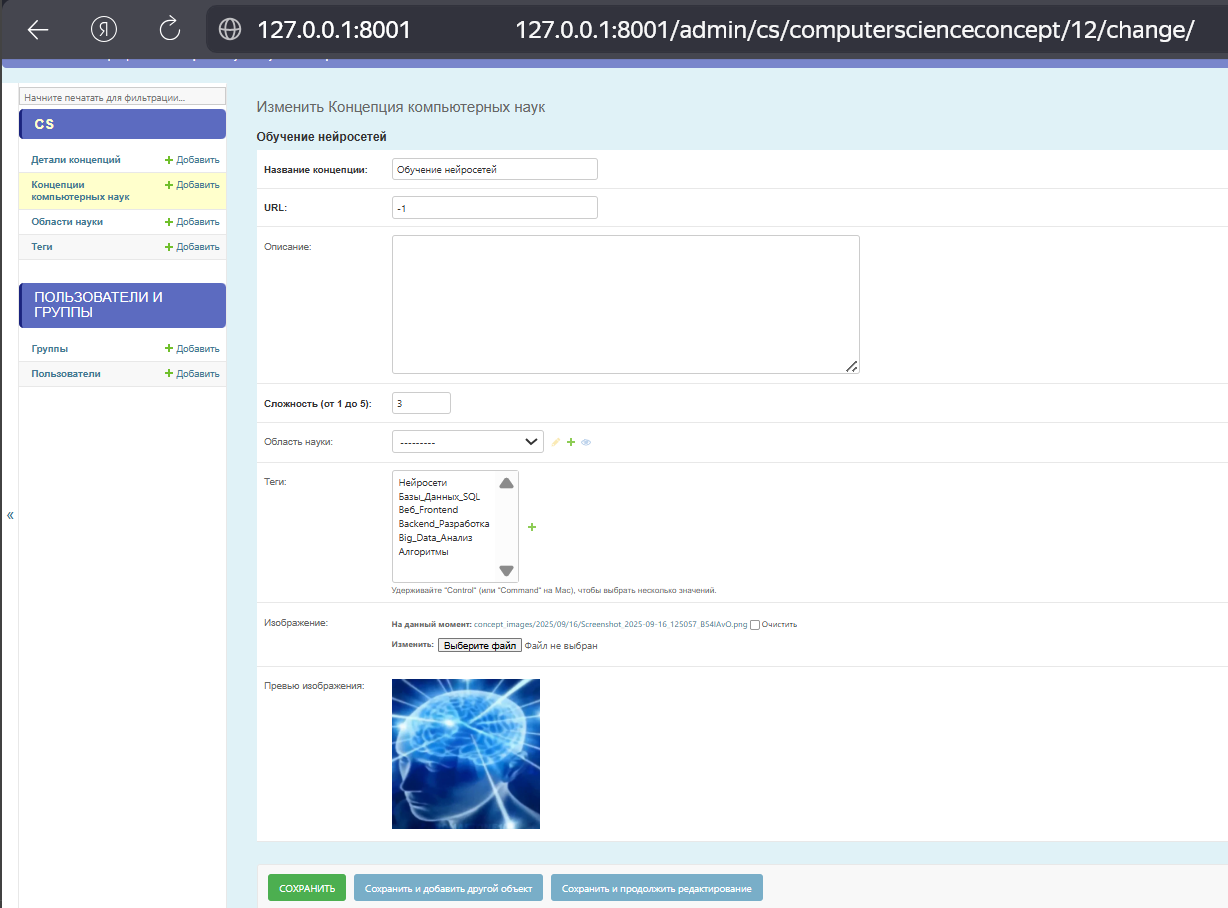


Рисунок 17 – Отображение концепции в admin панели.

**Листинг программ**

Views.py:

*from* django.shortcuts *import* render, redirect, get\_object\_or\_404

*from* django.http *import* Http404, HttpResponse

*from* django.conf *import* settings

*from* django.core.files.storage *import* default\_storage

*import* os

*import* uuid

*from* .models *import* ComputerScienceConcept, Tag, FieldOfStudy *# Updated model import*

*from* .forms *import* ConceptForm, ConceptModelForm, UploadForm *# Updated import*

def index(*request*):

    concepts = ComputerScienceConcept.published.all()

*return* render(request, 'cs/index.html', {

        'title': 'Главная страница',

        'concepts': concepts,

    })

def about(*request*):

*return* render(request, 'cs/about.html')

def compare(*request*):

*return* render(request, 'cs/compare.html')

def concepts\_list(*request*):

    all\_concepts = [

        {'id': 1, 'name': 'Data Structures', 'description': 'Fundamental ways to organize data.'},

        {'id': 2, 'name': 'Algorithms', 'description': 'Step-by-step procedures for calculations.'},

    ]

*return* render(request, 'cs/cs\_list.html', {'concepts': all\_concepts})

def concept\_detail(*request*, *concept\_slug*):

    concept = get\_object\_or\_404(ComputerScienceConcept, *slug*=concept\_slug)

*return* render(request, 'cs/concept\_detail.html', {

        'title': concept.title,

        'concept': concept,

    })

def archive(*request*, *year*):

*if* year > 2025:

*raise* Http404("Нет данных за этот год")

*return* HttpResponse(f"<h1>Архив за {year} год</h1>")

def redirect\_example(*request*):

*return* redirect('home')

def get\_params\_example(*request*):

    name = request.GET.get('name', 'Гость')

*return* HttpResponse(f"<h1>Привет, {name}!</h1>")

def category(*request*, *cat\_id*):

*return* HttpResponse(f"<h1>Страница категории</h1><p>ID категории: {cat\_id}</p>")

def tags\_list(*request*):

    tags = Tag.objects.all()

*return* render(request, 'cs/tags\_list.html', {

        'title': 'Теги',

        'tags': tags,

    })

def add\_concept\_custom(*request*):

    """

    Пункт 1: форма ConceptForm (несвязанная с моделью) с поддержкой загрузки изображения.

    """

*if* request.method == 'POST':

        form = ConceptForm(request.POST, request.FILES)

*if* form.is\_valid():

            cd = form.cleaned\_data

            concept = ComputerScienceConcept.objects.create(

*title*=cd['title'],

*# slug=cd['title'].lower().replace(' ', '-'), # Слаг теперь генерируется в модели*

*description*=cd['description'],

*difficulty*=cd['difficulty'],

*is\_published*=ComputerScienceConcept.Status.DRAFT

            )

*# Сохраняем изображение, если пользователь его загрузил*

            image = cd.get('image')

*if* image:

                concept.image = image

                concept.save()

*return* redirect(concept.get\_absolute\_url())

*else*:

        form = ConceptForm()

*return* render(request, 'cs/add\_concept\_custom.html', {'form': form})

def add\_concept\_model(*request*):

    """

    Пункт 2: форма ConceptModelForm (ModelForm) с поддержкой загрузки изображения.

    """

*if* request.method == 'POST':

        form = ConceptModelForm(request.POST, request.FILES)

*if* form.is\_valid():

            concept = form.save(*commit*=False)

            concept.is\_published = ComputerScienceConcept.Status.DRAFT

            concept.save()

            form.save\_m2m()  *# сохраняем M2M-поля (tags) и image*

*return* redirect(concept.get\_absolute\_url())

*else*:

        form = ConceptModelForm()

*return* render(request, 'cs/add\_concept\_model.html', {'form': form})

def upload\_file(*request*):

    """

    Обрабатывает UploadForm, сохраняет файл с случайным именем

    и выводит пользователю ссылку на загруженный файл.

    """

    link = None

*if* request.method == 'POST':

        form = UploadForm(request.POST, request.FILES)

*if* form.is\_valid():

            f = form.cleaned\_data['file']

            ext = os.path.splitext(f.name)[1]

            new\_name = f"{uuid.uuid4().hex}{ext}"

            path = os.path.join('uploads', new\_name)

            saved\_path = default\_storage.save(path, f)

            link = settings.MEDIA\_URL + saved\_path

*else*:

        form = UploadForm()

*return* render(request, 'cs/upload.html', {

        'form': form,

        'link': link,

    })

urls.py:

*from* django.urls *import* path

*from* . *import* views

*from* . *import* converters

*from* django.urls *import* register\_converter

*from* django.contrib *import* admin

*from* .views *import* add\_concept\_custom, add\_concept\_model, upload\_file

*from* django.conf *import* settings

*from* django.conf.urls.static *import* static

admin.site.site\_header = "Панель Администрирования"

admin.site.index\_title = "Управление сайтом"

admin.site.site\_title = "Администрирование MyCSProject"

register\_converter(converters.FourDigitYearConverter, 'year4')

urlpatterns = [

    path('', views.index, *name*='home'),

    path('about/', views.about, *name*='about'),

    path('compare/', views.compare, *name*='compare'),

    path('concepts/', views.concepts\_list, *name*='concepts'),

    path('concepts/<int:concept\_id>/', views.concept\_detail, *name*='concept\_detail'),

    path('archive/<year4:year>/', views.archive, *name*='archive'),

    path('go-home/', views.redirect\_example, *name*='go\_home'),

    path('hello/', views.get\_params\_example, *name*='hello'),

    path('category/<int:cat\_id>/', views.category, *name*='category'),

    path('concept/<slug:concept\_slug>/', views.concept\_detail, *name*='concept\_detail'),

    path('tags/', views.tags\_list, *name*='tags\_list'),

    path('admin/', admin.site.urls),

    path('add-custom/', add\_concept\_custom, *name*='add\_concept\_custom'),

    path('add-model/',  add\_concept\_model,  *name*='add\_concept\_model'),

    path('upload/', upload\_file, *name*='upload\_file'),

]

*if* settings.DEBUG:

    urlpatterns += static(settings.MEDIA\_URL, *document\_root*=settings.MEDIA\_ROOT)

*# handler404 = 'cs.views.custom\_404'*

models.py

*from* django.db *import* models

*from* django.urls *import* reverse

*from* django.utils.text *import* slugify *# Импортируем slugify*

*# Модель для областей компьютерных наук*

class FieldOfStudy(models.Model):

    name = models.CharField(*max\_length*=255, *verbose\_name*="Область науки")

    description = models.TextField(*blank*=True, *verbose\_name*="Описание области")

    class Meta:

        verbose\_name = "Область науки"

        verbose\_name\_plural = "Области науки"

    def \_\_str\_\_(*self*):

*return* self.name

*# Пользовательский менеджер для выборки только опубликованных записей*

class PublishedManager(models.Manager):

    def get\_queryset(*self*):

*# Фильтруем записи по полю публикации с использованием перечисления*

*return* super().get\_queryset().filter(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

class ComputerScienceConcept(models.Model):

*# Класс-перечисление для статуса публикации*

    class Status(models.IntegerChoices):

        DRAFT = 0, 'Черновик'

        PUBLISHED = 1, 'Опубликовано'

    title = models.CharField(*max\_length*=255, *verbose\_name*="Название концепции")

    slug = models.SlugField(*max\_length*=255, *unique*=True, *db\_index*=True, *verbose\_name*="URL")

    description = models.TextField(*blank*=True, *verbose\_name*="Описание")

    difficulty = models.IntegerField(*default*=1, *verbose\_name*="Сложность (от 1 до 5)") *# Новое поле для сложности*

    time\_create = models.DateTimeField(*auto\_now\_add*=True, *verbose\_name*="Время создания")

    time\_update = models.DateTimeField(*auto\_now*=True, *verbose\_name*="Время обновления")

    is\_published = models.BooleanField(*choices*=Status.choices, *default*=Status.DRAFT, *verbose\_name*="Публикация")

    image = models.ImageField(*upload\_to*='concept\_images/%Y/%m/%d/', *blank*=True, *null*=True, *verbose\_name*='Изображение') *# New field*

*# Связь один-ко-многим с FieldOfStudy*

    field\_of\_study = models.ForeignKey(

        FieldOfStudy,

*on\_delete*=models.CASCADE,

*related\_name*='concepts',

*verbose\_name*="Область науки",

*null*=True,

*blank*=True

    )

    objects = models.Manager()         *# Стандартный менеджер*

    published = PublishedManager()       *# Пользовательский менеджер для опубликованных записей*

    class Meta:

        ordering = ['-time\_create']

        indexes = [models.Index(*fields*=['-time\_create'])]

        verbose\_name = "Концепция компьютерных наук"

        verbose\_name\_plural = "Концепции компьютерных наук"

    def \_\_str\_\_(*self*):

*return* self.title

    def save(*self*, \**args*, \*\**kwargs*):

*if* not self.slug:

            self.slug = slugify(self.title)

*# Проверяем уникальность слага*

            base\_slug = self.slug

            num = 1

*while* ComputerScienceConcept.objects.filter(*slug*=self.slug).exists():

                self.slug = f"{base\_slug}-{num}"

                num += 1

        super().save(\*args, \*\*kwargs)

    def get\_absolute\_url(*self*):

*return* reverse('concept\_detail', *kwargs*={'concept\_slug': self.slug})

*# Модель для расширенной информации о концепции (OneToOne)*

class ConceptDetail(models.Model):

    concept = models.OneToOneField(

        ComputerScienceConcept,

*on\_delete*=models.CASCADE,

*related\_name*='detail',

*verbose\_name*="Концепция"

    )

    core\_technologies = models.TextField(*blank*=True, *verbose\_name*="Ключевые технологии")

    prerequisites = models.TextField(*blank*=True, *verbose\_name*="Предварительные условия")

    estimated\_learning\_time = models.PositiveIntegerField(*verbose\_name*="Примерное время изучения (часы)", *null*=True, *blank*=True)

    class Meta:

        verbose\_name = "Детали концепции"

        verbose\_name\_plural = "Детали концепций"

    def \_\_str\_\_(*self*):

*return* f"Детали {self.concept.title}"

*# Модель для тегов (Many-to-Many)*

class Tag(models.Model):

    name = models.CharField(*max\_length*=50, *unique*=True, *verbose\_name*="Тег")

    slug = models.SlugField(*max\_length*=50, *unique*=True, *db\_index*=True, *verbose\_name*="URL тега") *# Убрал default='default-slug'*

    class Meta:

        verbose\_name = "Тег"

        verbose\_name\_plural = "Теги"

    def \_\_str\_\_(*self*):

*return* self.name

    def save(*self*, \**args*, \*\**kwargs*):

*if* not self.slug:

            self.slug = slugify(self.name)

        super().save(\*args, \*\*kwargs)

    def get\_absolute\_url(*self*):

*return* reverse('concepts\_by\_tag', *kwargs*={'tag\_slug': self.slug})

*# Добавляем ManyToManyField в ComputerScienceConcept после определения Tag*

ComputerScienceConcept.add\_to\_class('tags', models.ManyToManyField(

    Tag,

*related\_name*='concepts',

*verbose\_name*="Теги",

*blank*=True

))

Admin.py

*from* django.contrib *import* admin, messages

*from* django.contrib.admin *import* SimpleListFilter

*from* django.utils.html *import* mark\_safe *# New import*

*from* django.db.models *import* ExpressionWrapper, F, DecimalField

*from* .models *import* ComputerScienceConcept, FieldOfStudy, ConceptDetail, Tag *# Updated model imports*

admin.site.site\_header = "Панель администрирования"

admin.site.index\_title = "Управление сайтом"

admin.site.site\_title = "Администрирование Computer Science Project"

*# Пользовательское вычисляемое поле: краткая информация (п.7)*

@admin.display(*description*="Краткая информация")

def brief\_info(*obj*):

*return* f"Описание: {len(obj.description)} символов" *if* obj.description *else* "Нет описания"

*# Пользовательское вычисляемое поле: цена с налогом (п.7)*

@admin.display(*description*="Сложность") *# Changed description*

def display\_difficulty(*obj*):

*if* obj.difficulty is not None:

*return* f"{obj.difficulty}/5"

*return* "N/A"

*# Кастомный фильтр для статуса публикации (п.9)*

class PublishedFilter(SimpleListFilter):

    title = "Статус публикации"

    parameter\_name = "pub\_status"

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ("published", "Опубликовано"),

            ("draft", "Черновик"),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* self.value() == "published":

*return* queryset.filter(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

*elif* self.value() == "draft":

*return* queryset.filter(*is\_published*=ComputerScienceConcept.Status.DRAFT)

*return* queryset

*# Дополнительный кастомный фильтр по диапазону сложности (п.9)*

class DifficultyRangeFilter(SimpleListFilter):

    title = "Диапазон сложности"

    parameter\_name = "difficulty\_range"

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ('low', 'Низкая (1-2)'),

            ('medium', 'Средняя (3)'),

            ('high', 'Высокая (4-5)'),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* self.value() == 'low':

*return* queryset.filter(*difficulty\_\_lte*=2)

*elif* self.value() == 'medium':

*return* queryset.filter(*difficulty*=3)

*elif* self.value() == 'high':

*return* queryset.filter(*difficulty\_\_gte*=4)

*return* queryset

@admin.register(ComputerScienceConcept) *# Updated model*

class ComputerScienceConceptAdmin(admin.ModelAdmin):

*# Поля для формы добавления/редактирования*

    fields = [

        'title', 'slug', 'description', 'difficulty',

        'field\_of\_study', 'tags', 'image', 'image\_preview' *# Updated fields*

    ]

    prepopulated\_fields = {"slug": ("title",)}

    readonly\_fields = ['time\_create', 'time\_update', 'image\_preview'] *# Added image\_preview*

*# Список записей*

    list\_display = (

        'id', 'title', 'field\_of\_study', 'time\_create',

        'is\_published', brief\_info, display\_difficulty, *# Updated display\_difficulty*

    )

    list\_display\_links = ('id', 'title')

    list\_editable = ('is\_published',)

    ordering = ['-time\_create', 'title']

    list\_per\_page = 5

    search\_fields = ['title', 'field\_of\_study\_\_name'] *# Updated search fields*

    list\_filter = [PublishedFilter, 'field\_of\_study', DifficultyRangeFilter] *# Updated list\_filter*

    actions = ['set\_published', 'set\_draft']

    @admin.action(*description*="Опубликовать выбранные записи")

    def set\_published(*self*, *request*, *queryset*):

        count = queryset.update(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

        self.message\_user(request, f"Статус 'Опубликовано' обновлён для {count} записей.", messages.SUCCESS)

    @admin.action(*description*="Снять с публикации выбранные записи")

    def set\_draft(*self*, *request*, *queryset*):

        count = queryset.update(*is\_published*=ComputerScienceConcept.Status.DRAFT)

        self.message\_user(request, f"{count} записей сняты с публикации.", messages.WARNING)

    @admin.display(*description*='Превью изображения') *# New method*

    def image\_preview(*self*, *obj*):

*if* obj.image:

*return* mark\_safe(f"<img src='{obj.image.url}' style='max-height:200px;' />")

*return* "(нет изображения)"

@admin.register(FieldOfStudy) *# Updated model*

class FieldOfStudyAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name', 'description') *# Updated fields*

    list\_display\_links = ('id', 'name')

    search\_fields = ('name', 'description') *# Updated search fields*

@admin.register(ConceptDetail) *# Updated model*

class ConceptDetailAdmin(admin.ModelAdmin):

    list\_display = ('id', 'concept', 'core\_technologies', 'prerequisites', 'estimated\_learning\_time') *# Updated fields*

    list\_display\_links = ('id', 'concept') *# Updated link*

    search\_fields = ('concept\_\_title', 'core\_technologies', 'prerequisites') *# Updated search fields*

@admin.register(Tag)

class TagAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name')

    list\_display\_links = ('id', 'name')

    search\_fields = ('name',)

upload.html

{% extends "cs/base.html" %}

{% block content %}

  <h1>Загрузка файла</h1>

  <form method="post" enctype="multipart/form-data">

    {% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.file.label\_tag }}<br>

      {{ form.file }}<br>

      <small>{{ form.file.help\_text }}</small><br>

      {{ form.file.errors }}

    </p>

    <button type="submit">Загрузить</button>

  </form>

  {% if link %}

    <p>Файл успешно загружен: <a href="{{ link }}" target="\_blank">{{ link }}</a></p>

  {% endif %}

{% endblock %}

Add\_concept\_model.html  
{% extends "cs/base.html" %}

{% block content %}

  <h1>Добавить концепцию (ModelForm)</h1>

  <form method="post" enctype="multipart/form-data">{% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.title.label\_tag }}<br>

      {{ form.title }}<br>

      <small>{{ form.title.help\_text }}</small><br>

      {{ form.title.errors }}

    </p>

    <p>

      {{ form.slug.label\_tag }}<br>

      {{ form.slug }}<br>

      {{ form.slug.errors }}

    </p>

    <p>

      {{ form.description.label\_tag }}<br>

      {{ form.description }}<br>

      {{ form.description.errors }}

    </p>

    <p>

      {{ form.difficulty.label\_tag }}<br>

      {{ form.difficulty }}<br>

      {{ form.difficulty.errors }}

    </p>

    <p>

      {{ form.field\_of\_study.label\_tag }}<br>

      {{ form.field\_of\_study }}<br>

      {{ form.field\_of\_study.errors }}

    </p>

    <p>

      {{ form.tags.label\_tag }}<br>

      {{ form.tags }}<br>

      {{ form.tags.errors }}

    </p>

    <p>

      {{ form.image.label\_tag }}<br>

      {{ form.image }}<br>

      {{ form.image.help\_text }}<br>

      {{ form.image.errors }}

    </p>

    <button type="submit">Сохранить</button>

  </form>

{% endblock %}

Add\_concept\_custom.html

{% extends 'cs/base.html' %}

{% block content %}

  <h1>Добавить концепцию (каркасная форма)</h1>

  <form method="post" enctype="multipart/form-data">{% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.title.label\_tag }}<br>

      {{ form.title }}<br>

      {{ form.title.help\_text }}<br>

      {{ form.title.errors }}

    </p>

    <p>

      {{ form.description.label\_tag }}<br>

      {{ form.description }}<br>

      {{ form.description.help\_text }}<br>

      {{ form.description.errors }}

    </p>

    <p>

      {{ form.difficulty.label\_tag }}<br>

      {{ form.difficulty }}<br>

      {{ form.difficulty.help\_text }}<br>

      {{ form.difficulty.errors }}

    </p>

    <p>

      {{ form.image.label\_tag }}<br>

      {{ form.image }}<br>

      {{ form.image.help\_text }}<br>

      {{ form.image.errors }}

    </p>

    <button type="submit">Сохранить</button>

  </form>

{% endblock %}

Concept\_detail.html

{% extends 'cs/base.html' %}

{% block title %}{{ concept.title }}{% endblock %}

{% block content %}

  <h2>{{ concept.title }}</h2>

  {# Если для этой концепции загружено изображение, покажем его #}

  {% if concept.image %}

    <div style="margin-bottom: 1em;">

      <img src="{{ concept.image.url }}" alt="{{ concept.title }}" style="max-width:100%; height:auto;">

    </div>

  {% endif %}

  <p>Сложность: {{ concept.difficulty }}</p>

  {% if concept.field\_of\_study %}

    <p>Область науки: {{ concept.field\_of\_study.name }}</p>

  {% endif %}

  <div>

    {{ concept.description|linebreaks }}

  </div>

  {% if concept.tags.all %}

    <p>Теги:

      {% for tag in concept.tags.all %}

        <a href="{{ tag.get\_absolute\_url }}">{{ tag.name }}</a>{% if not forloop.last %}, {% endif %}

      {% endfor %}

    </p>

  {% endif %}

  {% if concept.detail %}

    <h3>Дополнительная информация:</h3>

    <p>Ключевые технологии: {{ concept.detail.core\_technologies }}</p>

    <p>Предварительные условия: {{ concept.detail.prerequisites }}</p>

    <p>Примерное время изучения: {{ concept.detail.estimated\_learning\_time }} часов</p>

  {% endif %}

{% endblock %}

**Вывод**

В этой лабораторной работе мы поэтапно освоили приёмы работы с формами и файлами в Django. Сначала была создана несвязанная с моделью форма ConceptForm, в которой полям title, description и difficulty были назначены стандартные валидаторы, а также собственный валидатор, запрещающий цифры в названии концепции. Представление add\_concept\_custom принимало данные из request.POST и файлы из request.FILES, вручную создавало объект ComputerScienceConcept через ComputerScienceConcept.objects.create и, при наличии загруженного изображения, сохраняло его в поле image нашего экземпляра. Затем мы перешли к созданию ModelForm – класса ConceptModelForm, автоматически подтягивающего все необходимые поля из модели ComputerScienceConcept и дополнительно блокирующего слово «test» в заголовке. В методе add\_concept\_model с помощью form.save(commit=False) мы устанавливали статус «Черновик», сохраняли объект и вызывали save\_m2m для обработки связей Many-to-Many с тегами.

Для хранения изображений в модель был добавлен ImageField, а в представлении upload\_file реализована генерация случайного имени файла с использованием UUID для предотвращения конфликтов. В настройках проекта прописаны MEDIA\_ROOT и MEDIA\_URL и подключена отдача медиа-файлов через static() в режиме отладки. Шаблоны каждой формы получили атрибут enctype="multipart/form-data", что обеспечило корректную передачу бинарных данных. На странице детализации концепции в concept\_detail.html мы вставили тег <img src="{{ concept.image.url }}">, позволяющий видеть загруженную картинку. Наконец, в admin.py для модели ComputerScienceConcept добавлено поле-превью image\_preview, отображающее миниатюру через mark\_safe, и поле image включено в fields и readonly\_fields, что позволило красиво интегрировать изображения в административный интерфейс. Таким образом, лабораторная работа демонстрирует полный цикл: от валидации пользовательского ввода до загрузки и отображения медиа-контента в веб-приложении на Django.

Далее в работе настраивались панели поиска и фильтрации в админ-панели. Добавлены стандартные фильтры (например, по внешнему ключу «Область науки» – field\_of\_study) и собственный фильтр (PublishedFilter) для статуса публикации, а также дополнительный пользовательский фильтр (DifficultyRangeFilter) для диапазонов сложности. Эти фильтры позволяют быстро отсеивать записи по нужным критериям, а панель поиска даёт возможность фильтровать записи по заголовкам концепций и названиям связанных областей науки. Таким образом, мы продемонстрировали возможность комбинирования методов фильтрации в Django.

Наконец, для удобства работы с формами добавления и редактирования записей в админке мы настроили поля, отображаемые в форме, их порядок и автозаполнение слага на основе названия концепции. При этом поля для автоматической регистрации времени создания и обновления, а также превью изображения были объявлены только для чтения, что предотвращает их случайное изменение.

В результате лабораторная работа продемонстрировала, как с помощью встроенных инструментов Django можно не только управлять базовыми данными, но и глубоко настраивать административный интерфейс для предметной области компьютерных наук. Все изменения – от регистрации моделей до добавления кастомных действий, фильтров по сложности и статусу, а также настройки форм – позволяют администратору эффективно управлять образовательным контентом, проводить массовые операции и адаптировать внешний вид панели. Эти изменения подтверждаются визуально через обновлённые заголовки, новый интерфейс списка записей, удобные элементы поиска и фильтрации, а также через корректное отображение форм и превью изображений в админ-панели.