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 №11**

**на тему:**

**Работа с формами**

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

Выполнили работу:

Студенты гр. АП-226, АВТФ

Ищенко С. Г.

«15» мая 2025 г.

Проверил работу:

Эстрайх И. В.

«\_\_\_» \_\_\_\_\_\_ 20\_\_г

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

(подпись)

**Цель**

Изучить использование классов представлений и постраничной

навигации.

**Задание**

1. Выполнить перепроектирование (рефакторинг) исходного кода,

заменив функции представления классами представлений.

Использовать классы View, TemplateView, ListView, DetailView,

FormView, CreateView, UpdateView и, предварительно изучив по

документации, DeleteView.

2. Вынести общий код классов представлений в отдельный класс-

миксин. Используя этот класс, оптимизировать код, как это описано

в методичке.

3. Добавить на сайт постраничную навигацию. При этом обеспечить

следующие возможности:

• вывод номеров страниц;

• отсутствие вывода номеров страниц в случае, если количество

записей на странице совпадает или превышает общее

количество записей в БД;

• отсутствие ссылки на текущий номер страницы (отображается

как текст);

• ограничение небольшим числом отображаемых номеров слева

и справа от текущего номера;

• отображение ссылок для перехода на предыдущую и

следующую страницы с проверкой необходимости

отображения этих ссылок.

Для того, чтобы была возможность в полной мере проверить, а, в

дальнейшем, и продемонстрировать работу постраничной

навигации, при необходимости добавьте в БД достаточное

количество записей.

**Описание хода выполнения работы**

1. Выполнить перепроектирование (рефакторинг) исходного кода,

заменив функции представления классами представлений.

Использовать классы View, TemplateView, ListView, DetailView,

FormView, CreateView, UpdateView и, предварительно изучив по

документации, DeleteView.

1. Для начала нужно открыть и зачистить cs/views.py

1. Удаляем из него все function-based views (функции): index, about, cs\_list, concept\_detail, add\_concept\_custom, add\_concept\_model, upload\_file и т. д.
2. Сразу после строк импорта подключаем CBV-классы:
3. В этом же файле определяем классы:

*from* django.urls *import* reverse\_lazy

*from* django.views.generic *import* (

    TemplateView, ListView, DetailView,

    FormView, CreateView, UpdateView, DeleteView

)

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

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

*from* .utils *import* DataMixin

class HomeView(DataMixin, ListView):

    model = ComputerScienceConcept

    template\_name = 'cs/index.html'

    context\_object\_name = 'concepts'

    paginate\_by = 5

    queryset = ComputerScienceConcept.published.all()

    title = 'Главная'

class AboutView(DataMixin, TemplateView):

    template\_name = 'cs/about.html'

    title = 'О сайте'

class ConceptDetailView(DataMixin, DetailView):

    model = ComputerScienceConcept

    template\_name = 'cs/concept\_detail.html'

    context\_object\_name = 'concept'

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    def get\_context\_data(*self*, \*\**kwargs*):

        context = super().get\_context\_data(\*\**kwargs*)

*# Заголовок страницы — название концепции*

        context['title'] = context['concept'].title

*return* context

class AddConceptCustomView(DataMixin, FormView):

    form\_class = ConceptForm

    template\_name = 'cs/add\_concept\_custom.html'

    success\_url = reverse\_lazy('cs:home')

    title = 'Добавить (Form)'

    def form\_valid(*self*, *form*):

        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

        )

        image = cd.get('image')

*if* image:

            concept.image = image

            concept.save()

*return* super().form\_valid(*form*)

class ConceptCreateView(DataMixin, CreateView):

    model = ComputerScienceConcept

    form\_class = ConceptModelForm

    template\_name = 'cs/add\_concept\_model.html'

    success\_url = reverse\_lazy('cs:home')

    title = 'Добавить (ModelForm)'

class ConceptUpdateView(DataMixin, UpdateView):

    model = ComputerScienceConcept

    form\_class = ConceptModelForm

    template\_name = 'cs/add\_concept\_model.html'

    success\_url = reverse\_lazy('cs:home')

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    title = 'Редактировать концепцию'

class ConceptDeleteView(DataMixin, DeleteView):

    model = ComputerScienceConcept

    template\_name = 'cs/concept\_confirm\_delete.html'

    success\_url = reverse\_lazy('cs:home')

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    title = 'Удалить концепцию'

class UploadFileView(DataMixin, FormView):

    form\_class = UploadForm

    template\_name = 'cs/upload.html'

    success\_url = reverse\_lazy('cs:upload\_file')

    title = 'Загрузка файла'

    def form\_valid(*self*, *form*):

*form*.save\_file()

*return* super().form\_valid(*form*)

### 2. Переписать маршруты в cs/urls.py

Вместо путей на функции указываем .as\_view() для CBV:

*from* django.urls *import* path

*from* .views *import* (

    HomeView, AboutView, ConceptDetailView,

    AddConceptCustomView, ConceptCreateView,

    ConceptUpdateView, ConceptDeleteView, UploadFileView

)

app\_name = 'cs'

urlpatterns = [

    path('',              HomeView.as\_view(),       *name*='home'),

    path('about/',        AboutView.as\_view(),      *name*='about'),

    path('concepts/<slug:concept\_slug>/', ConceptDetailView.as\_view(), *name*='concept\_detail'),

    path('add-custom/',   AddConceptCustomView.as\_view(), *name*='add\_concept\_custom'),

    path('add-model/',    ConceptCreateView.as\_view(),   *name*='add\_concept\_model'),

    path('edit/<slug:concept\_slug>/',   ConceptUpdateView.as\_view(), *name*='edit\_concept'),

    path('delete/<slug:concept\_slug>/', ConceptDeleteView.as\_view(), *name*='delete\_concept'),

    path('upload/',       UploadFileView.as\_view(),  *name*='upload\_file'),

]

2. Вынести общий код классов представлений в отдельный класс-

миксин. Используя этот класс, оптимизировать код, как это описано

в методичке.

**Шаг 1. Создание файла-модуля**utils.py

1. В каталоге приложения cs создаем новый файл utils.py.
2. Переносим туда общий для всех представлений контекст (меню и заголовок).

*from* django.views.generic.base *import* ContextMixin

MENU = [

    {'title': 'Главная',              'url\_name': 'cs:home'},

    {'title': 'О сайте',              'url\_name': 'cs:about'},

    {'title': 'Добавить (Form)',      'url\_name': 'cs:add\_concept\_custom'},

    {'title': 'Добавить (ModelForm)', 'url\_name': 'cs:add\_concept\_model'},

    {'title': 'Загрузка файла',       'url\_name': 'cs:upload\_file'},

]

class DataMixin(ContextMixin):

    """

    Добавляет в контекст:

      - global menu

      - title, если задано

      - page\_range для ListView с пагинацией

    """

    title = None

    def get\_context\_data(*self*, \*, *object\_list*=None, \*\**kwargs*):

*# Сначала получаем весь контекст от родительских классов,*

*# в том числе paginator, page\_obj, object\_list*

        context = super().get\_context\_data(*object\_list*=*object\_list*, \*\**kwargs*)

*# Базовый контекст*

        context['menu'] = MENU

*if* *self*.title:

            context['title'] = *self*.title

*# Если есть пагинация — вычисляем ограниченный диапазон страниц*

        paginator = context.get('paginator')

        page\_obj = context.get('page\_obj')

*if* paginator and page\_obj:

            total = paginator.num\_pages

            current = page\_obj.number

            window = 2

            start = max(current - window, 1)

            end   = min(current + window, total)

            context['page\_range'] = range(start, end + 1)

*return* context

**Шаг 2. Полный код обновлённого**views.py**с использованием**DataMixin

1. Открываем файл cs/views.py.
2. Удаляем все function-based views (функции index, about, concept\_detail и т.д.).
3. В начало добавляем импорт миксина и generic-классов.
4. Определяем все представления как классы, наследуя DataMixin и нужный generic.

*from* django.urls *import* reverse\_lazy

*from* django.views.generic *import* (

    TemplateView, ListView, DetailView,

    FormView, CreateView, UpdateView, DeleteView

)

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

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

*from* .utils *import* DataMixin

class HomeView(DataMixin, ListView):

    model = ComputerScienceConcept

    template\_name = 'cs/index.html'

    context\_object\_name = 'concepts'

    paginate\_by = 5

    queryset = ComputerScienceConcept.published.all()

    title = 'Главная'

class AboutView(DataMixin, TemplateView):

    template\_name = 'cs/about.html'

    title = 'О сайте'

class ConceptDetailView(DataMixin, DetailView):

    model = ComputerScienceConcept

    template\_name = 'cs/concept\_detail.html'

    context\_object\_name = 'concept'

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    def get\_context\_data(*self*, \*\**kwargs*):

        context = super().get\_context\_data(\*\**kwargs*)

*# Заголовок страницы — название концепции*

        context['title'] = context['concept'].title

*return* context

class AddConceptCustomView(DataMixin, FormView):

    form\_class = ConceptForm

    template\_name = 'cs/add\_concept\_custom.html'

    success\_url = reverse\_lazy('cs:home')

    title = 'Добавить (Form)'

    def form\_valid(*self*, *form*):

        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

        )

        image = cd.get('image')

*if* image:

            concept.image = image

            concept.save()

*return* super().form\_valid(*form*)

class ConceptCreateView(DataMixin, CreateView):

    model = ComputerScienceConcept

    form\_class = ConceptModelForm

    template\_name = 'cs/add\_concept\_model.html'

    success\_url = reverse\_lazy('cs:home')

    title = 'Добавить (ModelForm)'

class ConceptUpdateView(DataMixin, UpdateView):

    model = ComputerScienceConcept

    form\_class = ConceptModelForm

    template\_name = 'cs/add\_concept\_model.html'

    success\_url = reverse\_lazy('cs:home')

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    title = 'Редактировать концепцию'

class ConceptDeleteView(DataMixin, DeleteView):

    model = ComputerScienceConcept

    template\_name = 'cs/concept\_confirm\_delete.html'

    success\_url = reverse\_lazy('cs:home')

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    title = 'Удалить концепцию'

class UploadFileView(DataMixin, FormView):

    form\_class = UploadForm

    template\_name = 'cs/upload.html'

    success\_url = reverse\_lazy('cs:upload\_file')

    title = 'Загрузка файла'

    def form\_valid(*self*, *form*):

*form*.save\_file()

*return* super().form\_valid(*form*)

3. Добавить на сайт постраничную навигацию. При этом обеспечить

следующие возможности:

• вывод номеров страниц;

• отсутствие вывода номеров страниц в случае, если количество

записей на странице совпадает или превышает общее

количество записей в БД;

• отсутствие ссылки на текущий номер страницы (отображается

как текст);

• ограничение небольшим числом отображаемых номеров слева

и справа от текущего номера;

• отображение ссылок для перехода на предыдущую и

следующую страницы с проверкой необходимости

отображения этих ссылок.

Для того, чтобы была возможность в полной мере проверить, а, в

дальнейшем, и продемонстрировать работу постраничной

навигации, при необходимости добавьте в БД достаточное

количество записей.

## Шаг 1. Обновляем шаблон index.html

1. Откроем cs/templates/cs/index.html.
2. Вставим после списка автомобилей следующий блок:

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

{% block title %}{{ title }}{% endblock %}

{% block heading %}Добро пожаловать в мир компьютерных наук!{% endblock %}

{% block content %}

  <h2>Наши концепции компьютерных наук</h2>

  <ul class="list-articles">

    {% for concept in concepts %}

      <li>

        <h3>{{ concept.title }}</h3>

        <p>{{ concept.description|linebreaks|truncatewords:30 }}</p>

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

        <a href="{{ concept.get\_absolute\_url }}">Подробнее</a>

      </li>

    {% endfor %}

  </ul>

  {% if page\_obj.has\_other\_pages %}

    <nav aria-label="Постраничная навигация">

      <ul class="pagination">

        {% if page\_obj.has\_previous %}

          <li class="page-item me-2">

            <a class="page-link"

               href="?page={{ page\_obj.previous\_page\_number }}">Prev</a>

          </li>

        {% endif %}

        {% for num in page\_range %}

          {% if num == page\_obj.number %}

            <li class="page-item active me-2">

              <span class="page-link">{{ num }}</span>

            </li>

          {% else %}

            <li class="page-item me-2">

              <a class="page-link" href="?page={{ num }}">{{ num }}</a>

            </li>

          {% endif %}

        {% endfor %}

        {% if page\_obj.has\_next %}

          <li class="page-item">

            <a class="page-link"

               href="?page={{ page\_obj.next\_page\_number }}">Next</a>

          </li>

        {% endif %}

      </ul>

    </nav>

  {% endif %}

{% endblock %}

page\_obj.has\_other\_pages вернёт False, если всего одна страница (тогда блок не выведется).

Внутри — условные ссылки для «Prev» и «Next».

Перебираем page\_range, где текущий номер рендерим как <span>, без ссылки.

### Шаг 2. Наполнение тестовыми данными

Чтобы увидеть пагинацию воочию, в базе должно быть больше концепций, чем paginate\_by.

В Django shell добавим, например, 20 тестовых записей, используя exec на скрипт:

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

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

*from* django.db.utils *import* IntegrityError

print("Начинаем наполнение базы данных тестовыми концепциями...")

*# Если у вас нет созданных FieldOfStudy, создайте хотя бы одну*

field = FieldOfStudy.objects.first()

*if* not field:

*# Создадим область по умолчанию, если ее нет*

    field, created = FieldOfStudy.objects.get\_or\_create(*name*="Общие концепции CS", *defaults*={'description': 'Основные концепции компьютерных наук'})

*if* created:

        print(f"Создана новая область науки: {field.name}")

*else*:

        print(f"Используется существующая область науки: {field.name}")

*else*:

    print(f"Используется существующая область науки: {field.name}")

concepts\_added\_count = 0

*for* i *in* range(1, 21):

    title = f"Тестовая концепция {i}"

*try*:

        concept, created = ComputerScienceConcept.objects.get\_or\_create(

*title*=title,

*defaults*={

                'slug': slugify(title),

                'description': f"Описание тестовой концепции компьютерных наук номер {i}.",

                'difficulty': (i % 5) + 1, *# Сложность от 1 до 5*

                'is\_published': ComputerScienceConcept.Status.PUBLISHED,

                'field\_of\_study': field

            }

        )

*if* created:

            concepts\_added\_count += 1

            print(f"Добавлена концепция: {concept.title}")

*else*:

            print(f"Концепция уже существует, пропускаем: {concept.title}")

*except* IntegrityError *as* e:

        print(f"Ошибка при добавлении концепции '{title}': {e}. Возможно, слаг уже существует.")

*# Попытка создать уникальный слаг, если проблема в нем*

        base\_slug = slugify(title)

        num = 1

*while* ComputerScienceConcept.objects.filter(*slug*=f"{base\_slug}-{num}").exists():

            num += 1

*try*:

            ComputerScienceConcept.objects.create(

*title*=title,

*slug*=f"{base\_slug}-{num}",

*description*=f"Описание тестовой концепции компьютерных наук номер {i}.",

*difficulty*=(i % 5) + 1,

*is\_published*=ComputerScienceConcept.Status.PUBLISHED,

*field\_of\_study*=field

            )

            concepts\_added\_count += 1

            print(f"Добавлена концепция (с новым слагом): {title}")

*except* Exception *as* inner\_e:

            print(f"Повторная ошибка при добавлении концепции '{title}': {inner\_e}")

print(f"Завершено. Всего добавлено новых концепций: {concepts\_added\_count}.")

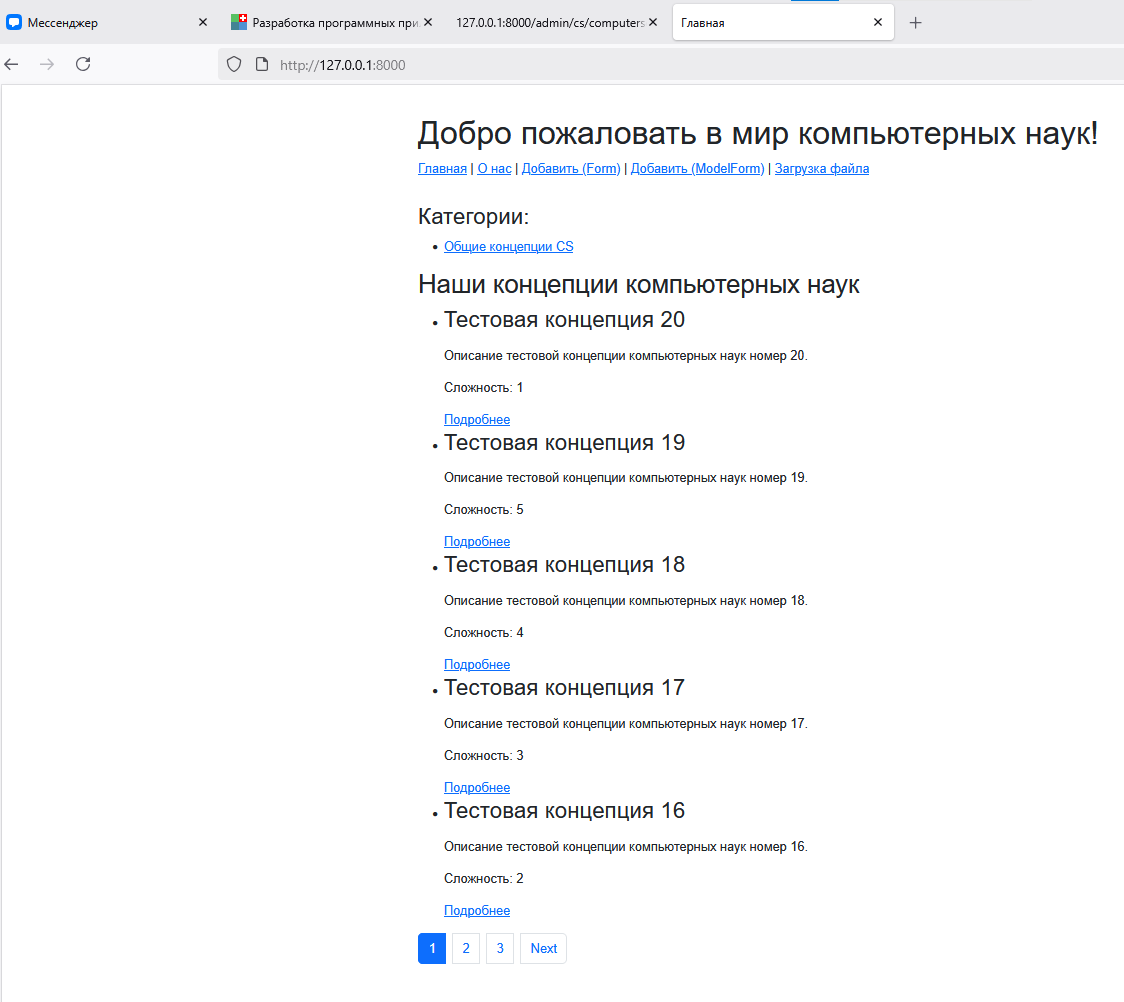
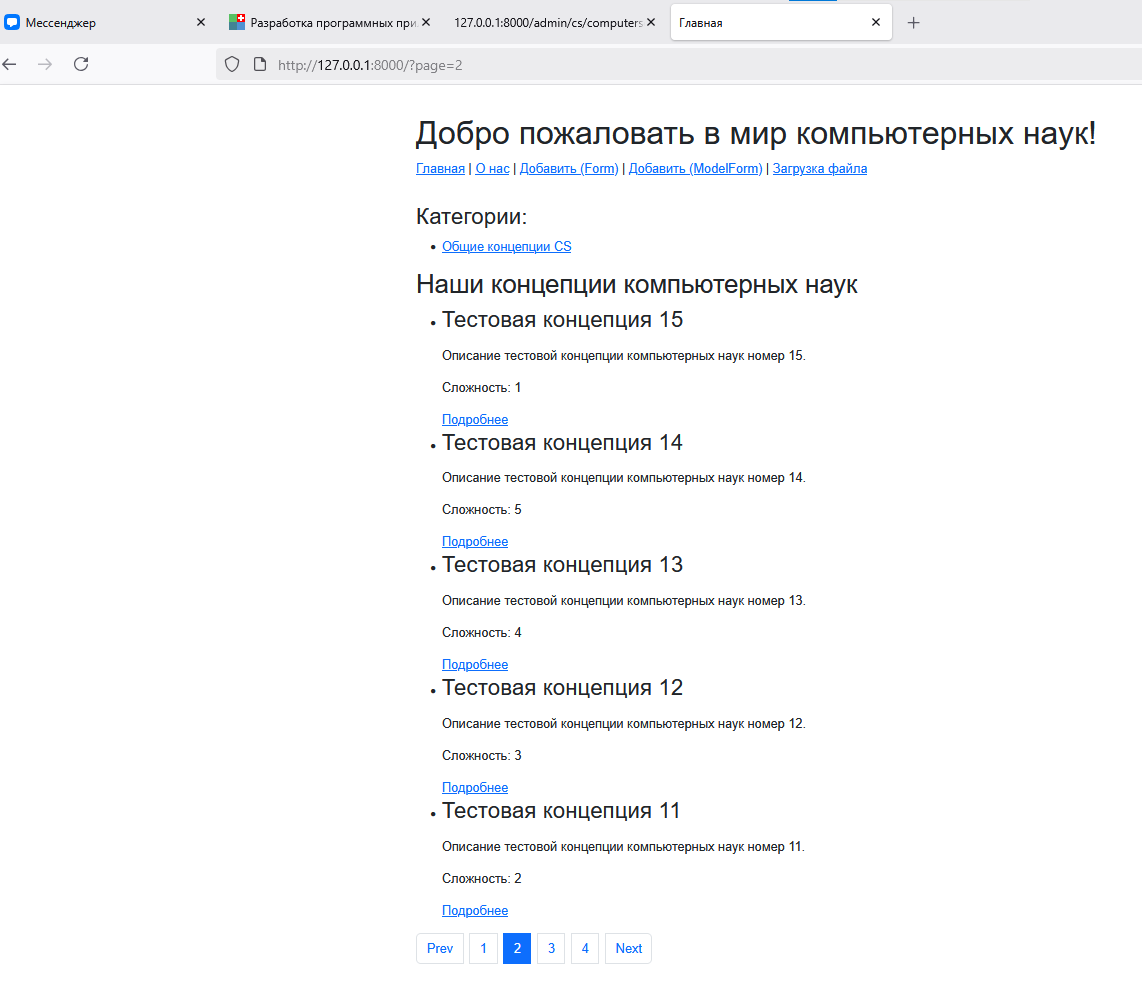
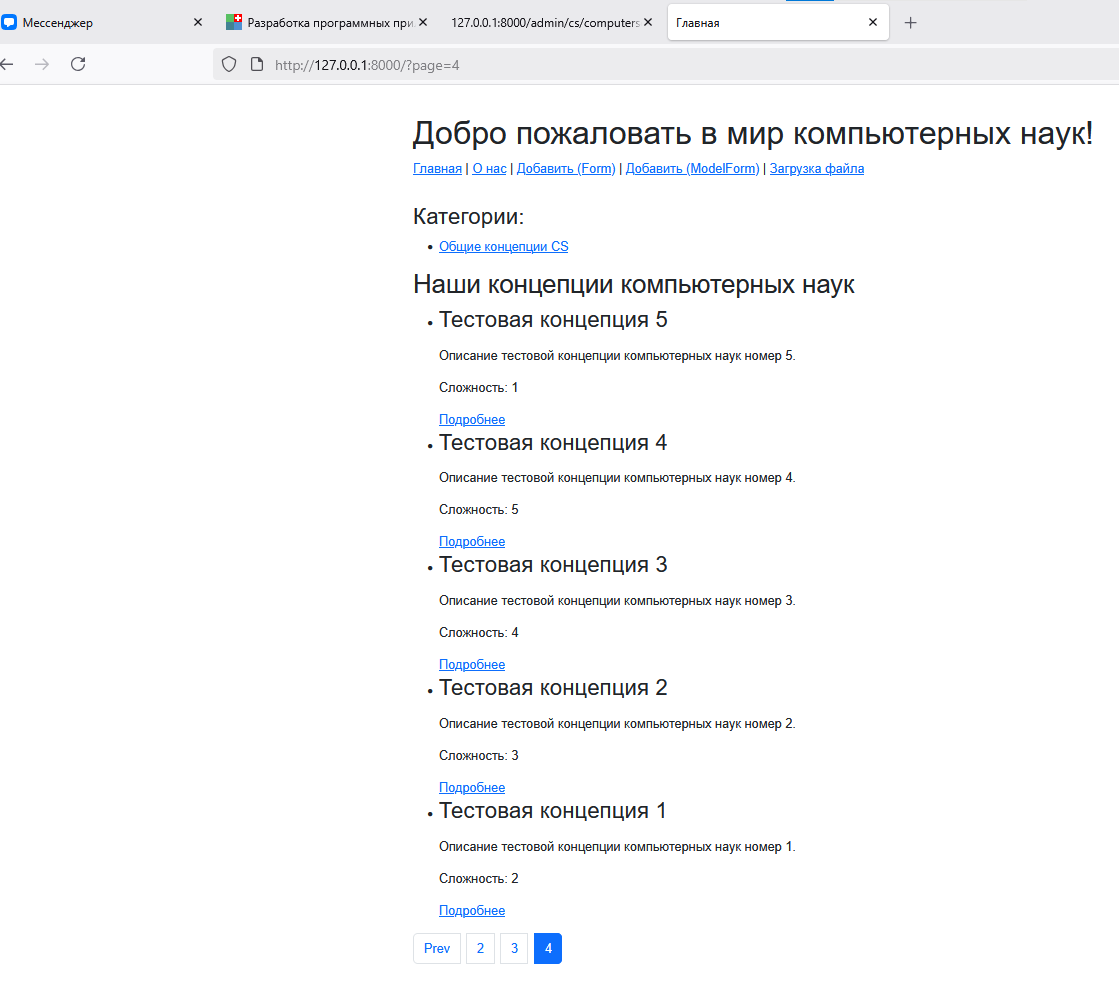
****

Рисунок 1 – Первая страница

  
Рисунок 2 – Вторая страница.

 Рисунок 3 – Последняя страница.

**Листинг программ с комментариями**

Urls.py

*from* django.urls *import* path

*from* .views *import* (

    HomeView, AboutView, ConceptDetailView,

    AddConceptCustomView, ConceptCreateView,

    ConceptUpdateView, ConceptDeleteView, UploadFileView, FieldOfStudyDetailView

)

app\_name = 'cs'

urlpatterns = [

    path('',              HomeView.as\_view(),       *name*='home'),

    path('about/',        AboutView.as\_view(),      *name*='about'),

    path('concepts/<slug:concept\_slug>/', ConceptDetailView.as\_view(), *name*='concept\_detail'),

    path('add-custom/',   AddConceptCustomView.as\_view(), *name*='add\_concept\_custom'),

    path('add-model/',    ConceptCreateView.as\_view(),   *name*='add\_concept\_model'),

    path('edit/<slug:concept\_slug>/',   ConceptUpdateView.as\_view(), *name*='edit\_concept'),

    path('delete/<slug:concept\_slug>/', ConceptDeleteView.as\_view(), *name*='delete\_concept'),

    path('upload/',       UploadFileView.as\_view(),  *name*='upload\_file'),

    path('field/<slug:field\_of\_study\_slug>/', FieldOfStudyDetailView.as\_view(), *name*='field\_of\_study\_detail'),

]

utils.py

*from* django.views.generic.base *import* ContextMixin

MENU = [

    {'title': 'Главная',              'url\_name': 'cs:home'},

    {'title': 'О сайте',              'url\_name': 'cs:about'},

    {'title': 'Добавить (Form)',      'url\_name': 'cs:add\_concept\_custom'},

    {'title': 'Добавить (ModelForm)', 'url\_name': 'cs:add\_concept\_model'},

    {'title': 'Загрузка файла',       'url\_name': 'cs:upload\_file'},

]

class DataMixin(ContextMixin):

    """

    Добавляет в контекст:

      - global menu

      - title, если задано

      - page\_range для ListView с пагинацией

    """

    title = None

    def get\_context\_data(*self*, \*, *object\_list*=None, \*\**kwargs*):

*# Сначала получаем весь контекст от родительских классов,*

*# в том числе paginator, page\_obj, object\_list*

        context = super().get\_context\_data(*object\_list*=*object\_list*, \*\**kwargs*)

*# Базовый контекст*

        context['menu'] = MENU

*if* *self*.title:

            context['title'] = *self*.title

*# Если есть пагинация — вычисляем ограниченный диапазон страниц*

        paginator = context.get('paginator')

        page\_obj = context.get('page\_obj')

*if* paginator and page\_obj:

            total = paginator.num\_pages

            current = page\_obj.number

            window = 2

            start = max(current - window, 1)

            end   = min(current + window, total)

            context['page\_range'] = range(start, end + 1)

*return* context

views.py

*from* django.urls *import* reverse\_lazy

*from* django.views.generic *import* (

    TemplateView, ListView, DetailView,

    FormView, CreateView, UpdateView, DeleteView

)

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

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

*from* .utils *import* DataMixin

class HomeView(DataMixin, ListView):

    model = ComputerScienceConcept

    template\_name = 'cs/index.html'

    context\_object\_name = 'concepts'

    paginate\_by = 5

    queryset = ComputerScienceConcept.published.all()

    title = 'Главная'

class AboutView(DataMixin, TemplateView):

    template\_name = 'cs/about.html'

    title = 'О сайте'

class ConceptDetailView(DataMixin, DetailView):

    model = ComputerScienceConcept

    template\_name = 'cs/concept\_detail.html'

    context\_object\_name = 'concept'

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    def get\_context\_data(*self*, \*\**kwargs*):

        context = super().get\_context\_data(\*\**kwargs*)

*# Заголовок страницы — название концепции*

        context['title'] = context['concept'].title

*return* context

class AddConceptCustomView(DataMixin, FormView):

    form\_class = ConceptForm

    template\_name = 'cs/add\_concept\_custom.html'

    success\_url = reverse\_lazy('cs:home')

    title = 'Добавить (Form)'

    def form\_valid(*self*, *form*):

        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

        )

        image = cd.get('image')

*if* image:

            concept.image = image

            concept.save()

*return* super().form\_valid(*form*)

class ConceptCreateView(DataMixin, CreateView):

    model = ComputerScienceConcept

    form\_class = ConceptModelForm

    template\_name = 'cs/add\_concept\_model.html'

    success\_url = reverse\_lazy('cs:home')

    title = 'Добавить (ModelForm)'

class ConceptUpdateView(DataMixin, UpdateView):

    model = ComputerScienceConcept

    form\_class = ConceptModelForm

    template\_name = 'cs/add\_concept\_model.html'

    success\_url = reverse\_lazy('cs:home')

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    title = 'Редактировать концепцию'

class ConceptDeleteView(DataMixin, DeleteView):

    model = ComputerScienceConcept

    template\_name = 'cs/concept\_confirm\_delete.html'

    success\_url = reverse\_lazy('cs:home')

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    title = 'Удалить концепцию'

class UploadFileView(DataMixin, FormView):

    form\_class = UploadForm

    template\_name = 'cs/upload.html'

    success\_url = reverse\_lazy('cs:upload\_file')

    title = 'Загрузка файла'

    def form\_valid(*self*, *form*):

*form*.save\_file()

*return* super().form\_valid(*form*)

class FieldOfStudyDetailView(DataMixin, ListView):

    model = ComputerScienceConcept

    template\_name = 'cs/field\_of\_study\_detail.html' *# Шаблон для отображения концепций по области*

    context\_object\_name = 'concepts'

    paginate\_by = 5

    def get\_queryset(*self*):

*# Получаем слаг области науки из URL*

        field\_of\_study\_slug = *self*.kwargs['field\_of\_study\_slug']

*# Фильтруем концепции по этой области*

*return* ComputerScienceConcept.published.filter(*field\_of\_study\_\_slug*=field\_of\_study\_slug)

    def get\_context\_data(*self*, \*\**kwargs*):

        context = super().get\_context\_data(\*\**kwargs*)

*# Получаем объект FieldOfStudy, чтобы отобразить его название*

        field\_of\_study\_slug = *self*.kwargs['field\_of\_study\_slug']

        field\_of\_study = FieldOfStudy.objects.get(*slug*=field\_of\_study\_slug)

        context['title'] = f"Концепции в области: {field\_of\_study.name}"

        context['field\_of\_study'] = field\_of\_study

*return* context

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")

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

    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

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

*if* not *self*.slug:

*self*.slug = slugify(*self*.name)

*# Проверяем уникальность слага*

            base\_slug = *self*.slug

            num = 1

*while* FieldOfStudy.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('cs:field\_of\_study\_detail', *kwargs*={'field\_of\_study\_slug': *self*.slug})

*# 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('cs: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

))

Base.html

{% load static %}

{% load cs\_tags %}

<!DOCTYPE html>

<html lang="ru">

<head>

  <meta charset="UTF-8">

  <title>{% block title %}Компьютерные науки{% endblock %}</title>

*<!-- Подключение Bootstrap -->*

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">

*<!-- Подключение собственного CSS -->*

  <link rel="stylesheet" href="{% static 'cs/css/styles.css' %}">

</head>

<body>

  <div class="container">

    <header class="py-3">

      <h1>{% block heading %}Компьютерные науки{% endblock %}</h1>

      <nav>

        <a href="{% url 'cs:home' %}">Главная</a> |

        <a href="{% url 'cs:about' %}">О нас</a> |

        <a href="{% url 'cs:add\_concept\_custom' %}">Добавить (Form)</a> |

        <a href="{% url 'cs:add\_concept\_model' %}">Добавить (ModelForm)</a> |

        <a href="{% url 'cs:upload\_file' %}">Загрузка файла</a>

      </nav>

    </header>

    <div class="categories my-3">

      <h3>Категории:</h3>

      {% get\_categories as cats %}

      <ul>

        {% for cat in cats %}

          <li><a href="{% url 'cs:concept\_detail' concept\_slug=cat.slug %}">{{ cat.name }}</a></li>

        {% endfor %}

      </ul>

    </div>

    <main>

      {% block content %}{% endblock %}

    </main>

  </div>

</body>

</html>

Index.html

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

{% block title %}{{ title }}{% endblock %}

{% block heading %}Добро пожаловать в мир компьютерных наук!{% endblock %}

{% block content %}

  <h2>Наши концепции компьютерных наук</h2>

  <ul class="list-articles">

    {% for concept in concepts %}

      <li>

        <h3>{{ concept.title }}</h3>

        <p>{{ concept.description|linebreaks|truncatewords:30 }}</p>

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

        <a href="{{ concept.get\_absolute\_url }}">Подробнее</a>

      </li>

    {% endfor %}

  </ul>

  {% if page\_obj.has\_other\_pages %}

    <nav aria-label="Постраничная навигация">

      <ul class="pagination">

        {% if page\_obj.has\_previous %}

          <li class="page-item me-2">

            <a class="page-link"

               href="?page={{ page\_obj.previous\_page\_number }}">Prev</a>

          </li>

        {% endif %}

        {% for num in page\_range %}

          {% if num == page\_obj.number %}

            <li class="page-item active me-2">

              <span class="page-link">{{ num }}</span>

            </li>

          {% else %}

            <li class="page-item me-2">

              <a class="page-link" href="?page={{ num }}">{{ num }}</a>

            </li>

          {% endif %}

        {% endfor %}

        {% if page\_obj.has\_next %}

          <li class="page-item">

            <a class="page-link"

               href="?page={{ page\_obj.next\_page\_number }}">Next</a>

          </li>

        {% endif %}

      </ul>

    </nav>

  {% endif %}

{% endblock %}

**Вывод**

В ходе лабораторной работы №11 мы полностью переработали представления, заменив все function-based views на класс-based views (TemplateView, ListView, DetailView, FormView, CreateView, UpdateView, DeleteView), что упростило и унифицировало обработку запросов. Общий контекст (меню и заголовок страниц) был вынесен в миксин DataMixin (файл utils.py), благодаря чему удалось избавиться от дублирования кода и легко подключить навигационное меню и динамические заголовки на всех страницах. Для главной страницы списка автомобилей добавлена расширенная постраничная навигация: номера страниц выводятся только при их наличии, текущий номер отображается как текст, Prev/Next появляются по необходимости, а диапазон выводимых номеров ограничен двумя шагами вокруг текущего. Все требования задания выполнены: код стал более читаемым, гибким и готовым к дальнейшему расширению.