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 №6**

**на тему:**

**Использование шаблонов в Django**

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

Выполнили работу:

Студенты гр. АП-226, АВТФ

Ищенко С. Г.

«27» марта 2025 г.

Проверил работу:

Эстрайх И. В.

«\_\_\_» \_\_\_\_\_\_ 20\_\_г

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

(подпись)

**Цель**

Изучить принципы создания и использования шаблонов в Django,

разработать шаблоны для своего проекта.

**Задание**

Разработать шаблоны для своего сайта. Развить свой проект, используя

приёмы, описанные в руководстве: передачу данных в шаблоны, фильтры,

теги шаблонов, наследование шаблонов, пользовательские теги шаблонов.

Подключить статические файлы, в том числе css.

**Описание хода выполнения работы**

#### 1. Организация структуры шаблонов

**Папка шаблонов:**  
В своем приложении cs создаём каталог шаблонов по пути:  
**Computer\_Science/cs\_ty/cs/templates/cs/**  
Это необходимо для того, чтобы избежать конфликтов имён, если в проекте появятся шаблоны других приложений.

**Файлы:**  
Добавим новые HTML-файлы в эту папку. Такие как: **index.html**, **about.html**, **compare.html**, **cs\_list.html**

#### 2. Создание базового шаблона

Чтобы избежать дублирования кода (например, подключения Bootstrap, меню, контейнера) создаём общий базовый шаблон, который будет наследоваться дочерними шаблонами.

**Computer\_Science/cs\_ty/cs/templates/cs/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 'home' %}">Главная</a> |

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

        <a href="{% url 'compare' %}">Сравнить</a> |

        <a href="{% url 'cs' %}">Темы</a> |

        <a href="{% url 'go\_home' %}">На главную</a>

      </nav>

    </header>

    <div class="categories my-3">

      <h3>Категории:</h3>

      {% get\_categories as cats %}

      <ul>

        {% for cat in cats %}

          <li><a href="{% url 'category' cat.id %}">{{ cat.name }}</a></li>

        {% endfor %}

      </ul>

    </div>

    <main>

      {% block content %}{% endblock %}

    </main>

  </div>

</body>

</html>

В этом файле определены три основных блока:

* **title** – для тега <title>
* **heading** – для главного заголовка страницы (например, <h1>)
* **content** – для основного содержимого страницы

#### 3. Модификация дочерних шаблонов

Каждый из существующих шаблонов теперь будет наследоваться от базового. Это позволит оставить общий код (Bootstrap, меню, контейнер) в одном месте.

**Computer\_Science/cs\_ty/cs/templates/cs**/index.html:

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

{% block title %}Главная страница{% endblock %}

{% block heading %}Добро пожаловать на сайт о компьютерных науках!{% endblock %}

{% block content %}

  <p>Здесь вы найдёте информацию о различных областях компьютерных наук.</p>

{% endblock %}

**Computer\_Science/cs\_ty/cs/templates/cs**/about.html:

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

{% block title %}О нас - Компьютерные науки{% endblock %}

{% block heading %}О нас{% endblock %}

{% block content %}

  <p>Мы — команда энтузиастов, собравшая информацию о различных областях компьютерных наук для создания образовательных материалов и обзоров.</p>

{% endblock %}

**Computer\_Science/cs\_ty/cs/templates/cs**/compare.html:

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

{% block title %}Сравнение тем - Компьютерные науки{% endblock %}

{% block heading %}Сравнение тем{% endblock %}

{% block content %}

  <p>Выберите две темы для сравнения их характеристик и возможностей.</p>

*<!-- Здесь можно разместить форму или список для выбора тем -->*

{% endblock %}

**Computer\_Science/cs\_ty/cs/templates/cs**/cs\_list.html:

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

{% block title %}Темы - Компьютерные науки{% endblock %}

{% block heading %}Список тем{% endblock %}

{% block content %}

  <p>Ниже представлен список тем в области компьютерных наук.</p>

  {% if cs\_topics %}

    <ul>

      {% for topic in cs\_topics %}

        <li>

          <strong>{{ topic.name }}</strong> – {{ topic.description|truncatewords:20 }}

          <a href="{% url 'cs\_detail' topic.id %}">Подробнее</a>

        </li>

      {% endfor %}

    </ul>

  {% else %}

    <p>Темы не найдены.</p>

  {% endif %}

{% endblock %}

#### 4. Обновление вызовов render() в views.py

Так как шаблоны теперь находятся в папке **cs/templates/cs/**, в функциях render() нужно указывать путь с префиксом **cs/**.

**Computer\_Science/cs\_ty/cs**/views.py:

*from* django.shortcuts *import* render, redirect

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

def index(*request*):

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

def about(*request*):

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

def compare(*request*):

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

def cs\_list(*request*):

    all\_cs\_topics = [

        {'id': 1, 'name': 'Искусственный интеллект', 'description': 'Область компьютерных наук, изучающая методы создания интеллектуальных агентов.'},

        {'id': 2, 'name': 'Машинное обучение', 'description': 'Подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться на данных.'},

        {'id': 3, 'name': 'Веб-разработка', 'description': 'Процесс создания веб-сайтов и веб-приложений.'},

        {'id': 4, 'name': 'Базы данных', 'description': 'Организованная коллекция данных, предназначенная для эффективного хранения и извлечения.'},

    ]

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

def cs\_detail(*request*, *topic\_id*):

*if* *topic\_id* > 10:

*raise* Http404("Тема с таким ID не найдена")

*return* HttpResponse(f"<h1>Информация о теме</h1><p>ID: {*topic\_id*}</p>")

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>")

#### 5. Обновление URL-конфигурации

Обновим маршруты, чтобы они указывали на новые пути шаблонов:

**Computer\_Science/cs\_ty/cs/urls.py:**

*from* django.urls *import* path

*from* . *import* views

*from* . *import* converters

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

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

urlpatterns = [

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

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

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

    path('cs/', views.cs\_list, *name*='cs'),      *# Список тем*

    path('cs/<int:topic\_id>/', views.cs\_detail, *name*='cs\_detail'),  *# Детальная страница темы*

    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'),

]

handler404 = 'cs.views.custom\_404'

#### 6. Подключение статических файлов (CSS)

Создаем в приложении папку для статики:

**Computer\_Science/cs\_ty/cs /static/cs/css/styles.css:**

body {

    font-family: Arial, sans-serif;

    margin: 20px;

}

header nav ul {

    list-style-type: none;

    padding: 0;

}

header nav ul li {

    display: inline;

    margin-right: 15px;

}

.container {

    margin-top: 20px;

}

footer {

    margin-top: 20px;

    text-align: center;

    color: #777;

}

#### 7. Пользовательские шаблонные теги

Для демонстрации создадим простой пользовательский тег для вывода категорий:

**Computer\_Science/cs\_ty/cs /templatetags/cs\_tags.py:**

*from* django *import* template

*# Пример фиктивного списка категорий*

categories\_db = [

    {'id': 1, 'name': 'Программирование'},

    {'id': 2, 'name': 'Алгоритмы'},

    {'id': 3, 'name': 'Базы данных'},

    {'id': 4, 'name': 'Искусственный интеллект'},

]

register = template.Library()

@register.simple\_tag()

def get\_categories():

*return* categories\_db

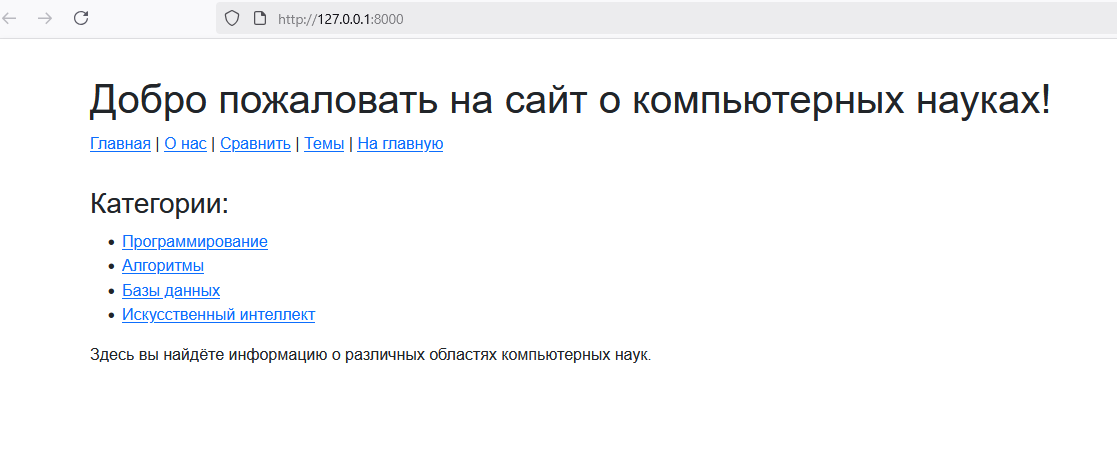
Так же добавим view функцию для обработки маршрута категории в cs/views/py

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

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

И маршрут в cs/urls.py

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

****

8. Передача данных в шаблоны

В моем проекте данные для динамического формирования страниц передаются через контекст функций представления в файле **Computer\_Science/cs\_ty/cs/**views.py. Например, функция cs\_list формирует список автомобилей и передаёт его в шаблон следующим образом:

def cs\_list(*request*):

    all\_cs\_topics = [

        {'id': 1, 'name': 'Искусственный интеллект', 'description': 'Область компьютерных наук, изучающая методы создания интеллектуальных агентов.'},

        {'id': 2, 'name': 'Машинное обучение', 'description': 'Подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться на данных.'},

        {'id': 3, 'name': 'Веб-разработка', 'description': 'Процесс создания веб-сайтов и веб-приложений.'},

        {'id': 4, 'name': 'Базы данных', 'description': 'Организованная коллекция данных, предназначенная для эффективного хранения и извлечения.'},

    ]

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

В этом примере словарь {'cs\_topics': all\_cs\_topics} передаётся в шаблон **Computer\_Science/cs\_ty/cs/**templates/cs/cs\_list.html, где данные можно использовать для динамического вывода информации.

9. Использование фильтров шаблонов

В шаблонах проекта используются стандартные встроенные фильтры Django для форматирования и обработки данных. Например:

**Фильтр truncatewords:**  
В шаблоне **Computer\_Science/cs\_ty/cs/**templates/cs/cs\_list.html для описания автомобилей применяется фильтр truncatewords для ограничения количества выводимых слов:

{{ topic.description|truncatewords:20 }}

Это позволяет сократить длинное описание до 20 слов, что улучшает читаемость страницы.

**Листинг**

|  |
| --- |
| Содержимое “urls.py” |
| *from* django.urls *import* path  *from* . *import* views  *from* . *import* converters  *from* django.urls *import* register\_converter  register\_converter(converters.FourDigitYearConverter, 'year4')  urlpatterns = [      path('', views.index, *name*='home'),           *# Главная страница*      path('about/', views.about, *name*='about'),      *# О нас*      path('compare/', views.compare, *name*='compare'), *# Сравнение*      path('cs/', views.cs\_list, *name*='cs'),      *# Список тем*      path('cs/<int:topic\_id>/', views.cs\_detail, *name*='cs\_detail'),  *# Детальная страница темы*      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'),  ]  handler404 = 'cs.views.custom\_404' |
| Содержимое “views.py” |
| *from* django.shortcuts *import* render, redirect  *from* django.http *import* Http404, HttpResponse  def index(*request*):  *return* render(*request*, 'cs/index.html')  def about(*request*):  *return* render(*request*, 'cs/about.html')  def compare(*request*):  *return* render(*request*, 'cs/compare.html')  def cs\_list(*request*):      all\_cs\_topics = [          {'id': 1, 'name': 'Искусственный интеллект', 'description': 'Область компьютерных наук, изучающая методы создания интеллектуальных агентов.'},          {'id': 2, 'name': 'Машинное обучение', 'description': 'Подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться на данных.'},          {'id': 3, 'name': 'Веб-разработка', 'description': 'Процесс создания веб-сайтов и веб-приложений.'},          {'id': 4, 'name': 'Базы данных', 'description': 'Организованная коллекция данных, предназначенная для эффективного хранения и извлечения.'},      ]  *return* render(*request*, 'cs/cs\_list.html', {'cs\_topics': all\_cs\_topics})  def cs\_detail(*request*, *topic\_id*):  *if* *topic\_id* > 10:  *raise* Http404("Тема с таким ID не найдена")  *return* HttpResponse(f"<h1>Информация о теме</h1><p>ID: {*topic\_id*}</p>")  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>") |
| Содержимое “about.html” |
| {% extends 'cs/base.html' %}  {% block title %}О нас - Компьютерные науки{% endblock %}  {% block heading %}О нас{% endblock %}  {% block content %}    <p>Мы — команда энтузиастов, собравшая информацию о различных областях компьютерных наук для создания образовательных материалов и обзоров.</p>  {% endblock %} |
| Содержимое “compare.html” |
| {% extends 'cs/base.html' %}  {% block title %}Сравнение тем - Компьютерные науки{% endblock %}  {% block heading %}Сравнение тем{% endblock %}  {% block content %}    <p>Выберите две темы для сравнения их характеристик и возможностей.</p>  *<!-- Здесь можно разместить форму или список для выбора тем -->*  {% endblock %} |
| Содержимое “index.html’’ |
| {% extends 'cs/base.html' %}  {% block title %}Главная страница{% endblock %}  {% block heading %}Добро пожаловать на сайт о компьютерных науках!{% endblock %}  {% block content %}    <p>Здесь вы найдёте информацию о различных областях компьютерных наук.</p>  {% endblock %} |
| Содержимое cs\_list.html |
| {% extends 'cs/base.html' %}  {% block title %}Темы - Компьютерные науки{% endblock %}  {% block heading %}Список тем{% endblock %}  {% block content %}    <p>Ниже представлен список тем в области компьютерных наук.</p>    {% if cs\_topics %}      <ul>        {% for topic in cs\_topics %}          <li>            <strong>{{ topic.name }}</strong> – {{ topic.description|truncatewords:20 }}            <a href="{% url 'cs\_detail' topic.id %}">Подробнее</a>          </li>        {% endfor %}      </ul>    {% else %}      <p>Темы не найдены.</p>    {% endif %}  {% endblock %} |
| Содержимое 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 'home' %}">Главная</a> |          <a href="{% url 'about' %}">О нас</a> |          <a href="{% url 'compare' %}">Сравнить</a> |          <a href="{% url 'cs' %}">Темы</a> |          <a href="{% url 'go\_home' %}">На главную</a>        </nav>      </header>      <div class="categories my-3">        <h3>Категории:</h3>        {% get\_categories as cats %}        <ul>          {% for cat in cats %}            <li><a href="{% url 'category' cat.id %}">{{ cat.name }}</a></li>          {% endfor %}        </ul>      </div>      <main>        {% block content %}{% endblock %}      </main>    </div>  </body>  </html> |
| Содержимое ‘’styles.css’’ |
| body {      font-family: Arial, sans-serif;      margin: 20px;  }  header nav ul {      list-style-type: none;      padding: 0;  }  header nav ul li {      display: inline;      margin-right: 15px;  }  .container {      margin-top: 20px;  }  footer {      margin-top: 20px;      text-align: center;      color: #777;  } |
| Содержимое ‘cs\_tags.py’’ |
| *from* django *import* template  *# Пример фиктивного списка категорий*  categories\_db = [      {'id': 1, 'name': 'Программирование'},      {'id': 2, 'name': 'Алгоритмы'},      {'id': 3, 'name': 'Базы данных'},      {'id': 4, 'name': 'Искусственный интеллект'},  ]  register = template.Library()  @register.simple\_tag()  def get\_categories():  *return* categories\_db |

Вывод

В ходе выполнения лабораторной работы №6 «Использование шаблонов в Django» были достигнуты следующие результаты:

* Организация шаблонов и наследование:  
  Создан базовый шаблон, содержащий общие элементы сайта (подключение статики, меню, шапку, футер). Дочерние шаблоны наследуют этот базовый, что позволяет централизованно управлять общим оформлением и существенно сокращает дублирование кода.
* Передача данных в шаблоны:  
  Данные передаются через контекст функций представления, что обеспечивает динамическое наполнение страниц. Это позволяет легко интегрировать информацию, полученную из базы или иных источников, непосредственно в HTML‑шаблоны.
* Использование стандартных фильтров и тегов шаблонов:  
  Применение фильтров (например, capfirst, truncatewords, linebreaks) и шаблонных тегов (if, for, url) обеспечивает гибкое форматирование и условный вывод данных. Это улучшает читаемость шаблонов и упрощает обработку информации.
* Разработка пользовательских шаблонных тегов:  
  Создан и интегрирован пользовательский тег для вывода списка категорий, что демонстрирует возможность расширения функционала шаблонизатора Django. Пользовательские теги позволяют вынести повторяющуюся логику в отдельный модуль и использовать её во всех шаблонах.
* Подключение статических файлов:  
  Настроено подключение статических файлов (CSS, JavaScript, изображения) с помощью тега {% load static %} и соответствующих настроек в файле settings.py. Это обеспечивает единообразное оформление сайта и облегчает дальнейшее управление внешним видом проекта.

В итоге, лабораторная работа позволила освоить принципы работы шаблонов Django, улучшить структуру проекта, обеспечить гибкость и масштабируемость веб-приложения, а также подготовить базу для дальнейшего расширения функционала сайта.