

Санкт-Петербургский Национальный Исследовательский

Университет ИТМО

Факультет «Информационных технологий и программирования»

Направление подготовки «Инфокоммуникационные технологии и системы  
связи»

**Практическая работа №7**

**Создание проекта на Django**

Выполнила:

Бакланова А.Г.

Группа: К3322

Проверил:

Марченко Е.В.

Санкт-Петербург,

2024

## Содержание

Цель работы: .....	3
Ход работы.....	4
1. Практическое задание .....	4
1.1. Создание структуры.....	4
1.2. Frontend .....	8
Заключение .....	20

## Цель работы:

Создать свой проект на django с нуля. В проекте три страницы: главная, о разработке, страница обратной связи. У всех трех страниц одинаковые футеры и хидеры (сделать шаблоны в django), с использованием возможностей django разработать форму обратной связи (на третьей странице) и обработку данных формы.

## Ход работы

### 1. Практическое задание

#### 1.1. Создание структуры

Для создания проекта на Django с нуля выполняли следующие шаги:

1. Создание директории проекта:

```
mkdir mysite
```

```
cd mysite
```

2. Создание виртуального окружения

```
python -m venv venv
```

3. Активация виртуального окружения

```
venv\Scripts\activate
```

4. Установка Django

```
pip install django
```

5. Создаем Django – проект

```
django-admin startproject mysite .
```

6. Создание приложения

```
python manage.py startapp main
```

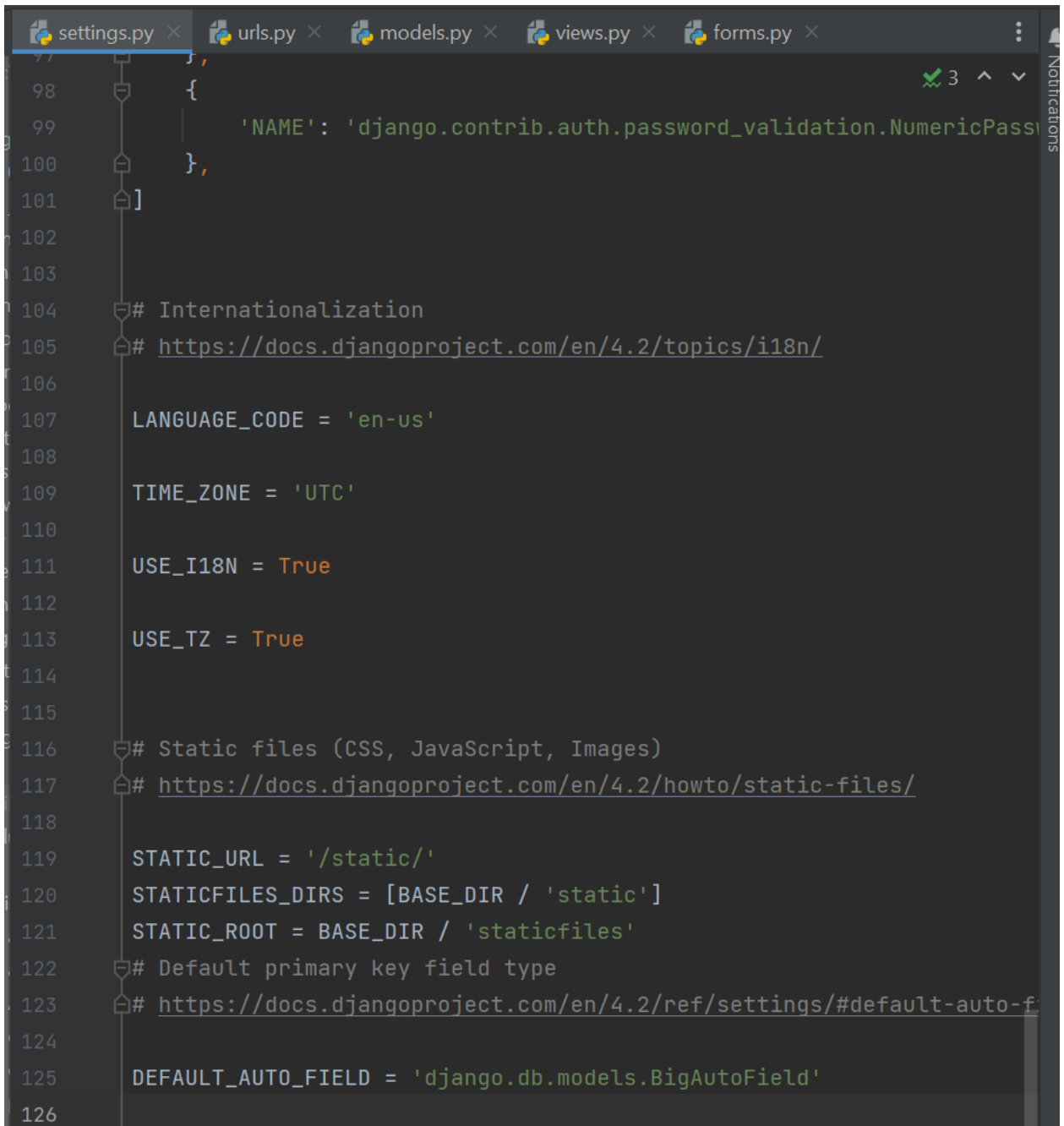
После создания проекта в файле settings.py были добавлены следующие строчки (Рисунок 1):

```
STATICFILES_DIRS = [BASE_DIR / 'static']
```

```
STATIC_ROOT = BASE_DIR / 'staticfiles'
```

STATICFILES\_DIRS нужен для указания путей к папкам со статическими файлами во время разработки, чтобы Django мог их находить без команды collectstatic.

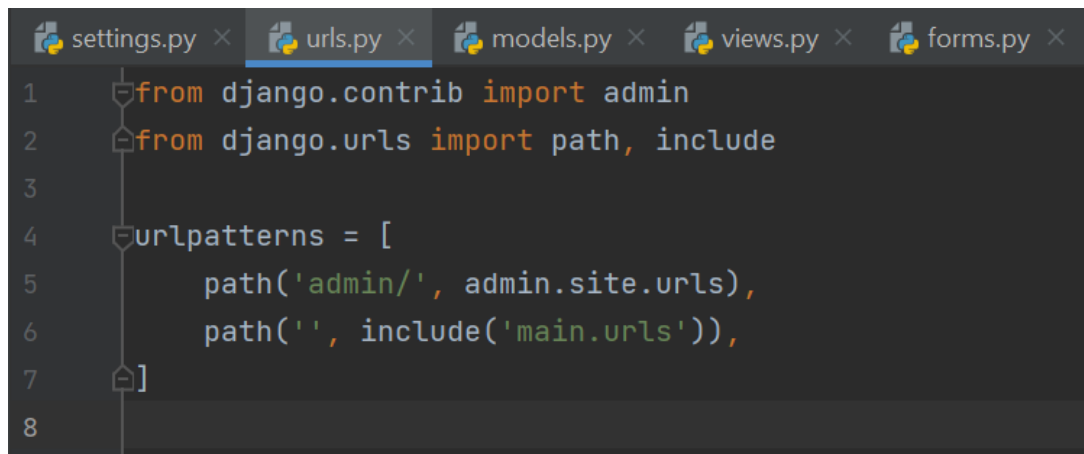
STATIC\_ROOT используется для указания папки, куда будут собираться все статические файлы при подготовке проекта к продакшену через команду python manage.py collectstatic.



```
97     },
98     {
99         'NAME': 'django.contrib.auth.password_validation.NumericPassw
100     },
101 ]
102
103
104 # Internationalization
105 # https://docs.djangoproject.com/en/4.2/topics/i18n/
106
107 LANGUAGE_CODE = 'en-us'
108
109 TIME_ZONE = 'UTC'
110
111 USE_I18N = True
112
113 USE_TZ = True
114
115
116 # Static files (CSS, JavaScript, Images)
117 # https://docs.djangoproject.com/en/4.2/howto/static-files/
118
119 STATIC_URL = '/static/'
120 STATICFILES_DIRS = [BASE_DIR / 'static']
121 STATIC_ROOT = BASE_DIR / 'staticfiles'
122
123 # Default primary key field type
124 # https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-f
125
126 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Рисунок 1 – Файл settings.py

В файле urls.py подключается административная панель по пути admin/ и маршруты приложения main, чтобы структура URL-адресов проекта была организованной и расширяемой. Функция include('main.urls') позволяет вынести маршрутизацию приложения main в отдельный файл, упрощая управление маршрутами при росте проекта (Рисунок 2).

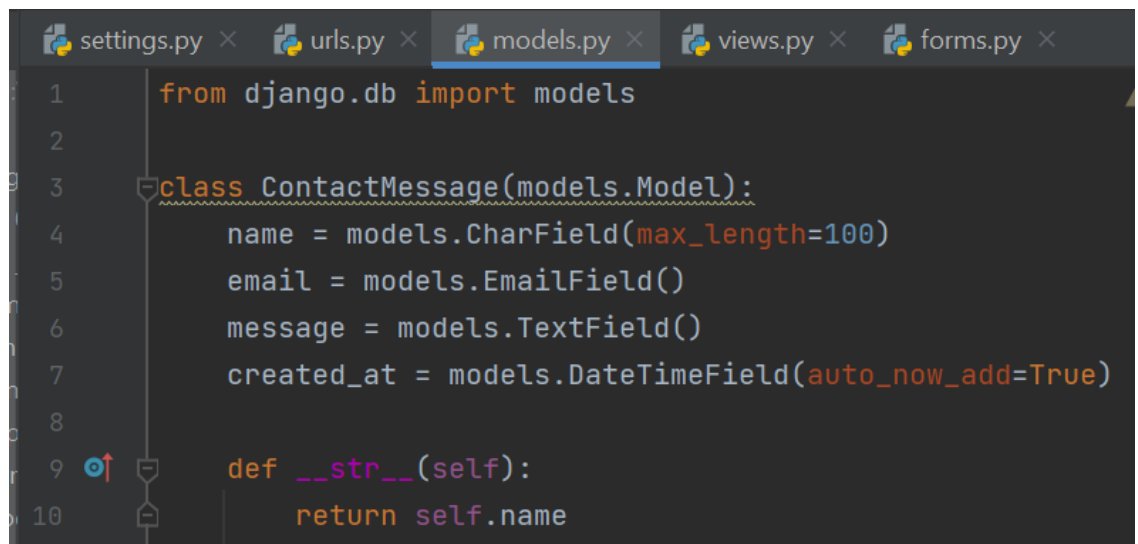


```
1 from django.contrib import admin
2 from django.urls import path, include
3
4 urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('', include('main.urls')),
7 ]
8
```

Рисунок 2 – Файл urls.py

В файле models.py модель ContactMessage описывает структуру данных для хранения сообщений от пользователей, включая их имя, email, текст сообщения и дату отправки. Определение модели позволяет Django автоматически создавать таблицу в базе данных и управлять данными через ORM (Рисунок 3).

В качестве приложения для работы с базой данных был скачен и установлен DB Browser (SQLite).

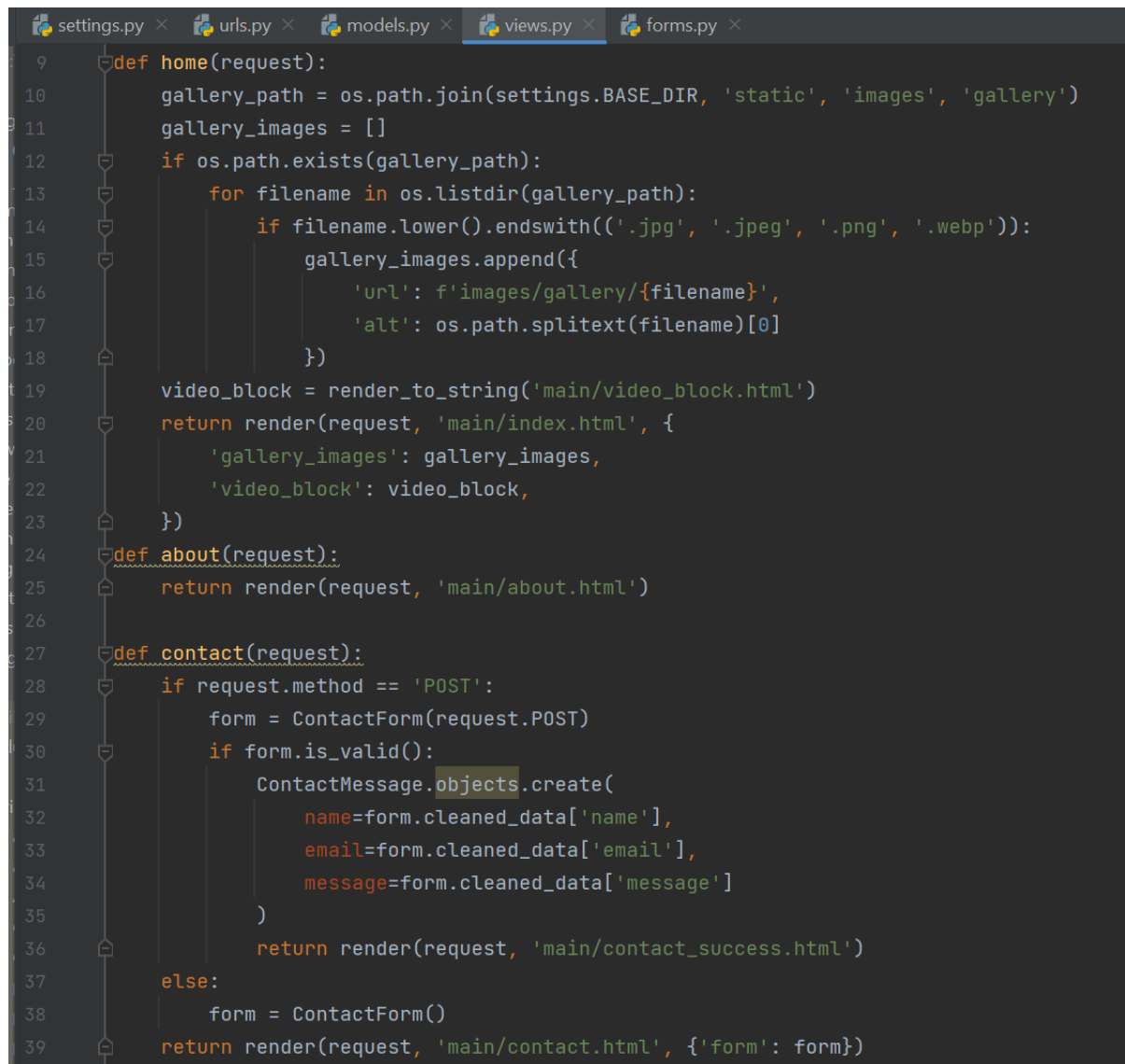


```
1 from django.db import models
2
3 class ContactMessage(models.Model):
4     name = models.CharField(max_length=100)
5     email = models.EmailField()
6     message = models.TextField()
7     created_at = models.DateTimeField(auto_now_add=True)
8
9     def __str__(self):
10         return self.name
```

Рисунок 3 – Файл models.py

В файле views.py описаны представления, которые обрабатывают запросы к разным страницам сайта. Функция home формирует список изображений из папки /static/images/gallery, загружает отдельный HTML-блок с видео и передаёт их в шаблон index.html для отображения на главной странице. Функция about просто рендерит страницу "О нас" без передачи

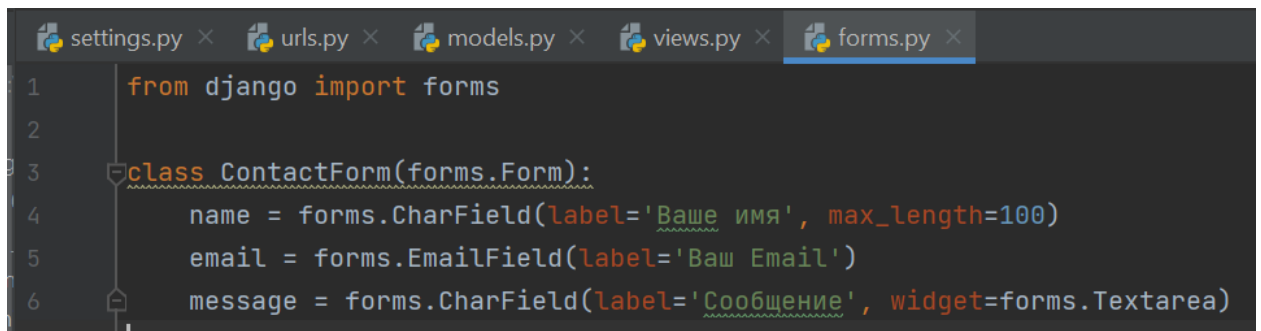
дополнительных данных. Функция `contact` обрабатывает отправку формы: если запрос методом `POST` и данные формы валидны, то они сохраняются в базу через модель `ContactMessage`, а затем пользователю показывается страница успешной отправки сообщения. Если форма не была отправлена или заполнена некорректно, пользователю снова показывается страница с формой обратной связи. Код представлен на рисунках 4.



```
9 def home(request):
10     gallery_path = os.path.join(settings.BASE_DIR, 'static', 'images', 'gallery')
11     gallery_images = []
12     if os.path.exists(gallery_path):
13         for filename in os.listdir(gallery_path):
14             if filename.lower().endswith(('.jpg', '.jpeg', '.png', '.webp')):
15                 gallery_images.append({
16                     'url': f'images/gallery/{filename}',
17                     'alt': os.path.splitext(filename)[0]
18                 })
19     video_block = render_to_string('main/video_block.html')
20     return render(request, 'main/index.html', {
21         'gallery_images': gallery_images,
22         'video_block': video_block,
23     })
24 def about(request):
25     return render(request, 'main/about.html')
26
27 def contact(request):
28     if request.method == 'POST':
29         form = ContactForm(request.POST)
30         if form.is_valid():
31             ContactMessage.objects.create(
32                 name=form.cleaned_data['name'],
33                 email=form.cleaned_data['email'],
34                 message=form.cleaned_data['message']
35             )
36             return render(request, 'main/contact_success.html')
37         else:
38             form = ContactForm()
39     return render(request, 'main/contact.html', {'form': form})
```

Рисунок 4 – Файл `views.py`

Файл `forms.py` содержит форму `ContactForm`, которая описывает поля для ввода имени, email и сообщения, а также настраивает отображение формы в шаблоне. Форма используется для валидации данных, отправленных пользователем через страницу обратной связи (Рисунок 5).



```
1 from django import forms
2
3 class ContactForm(forms.Form):
4     name = forms.CharField(label='Ваше имя', max_length=100)
5     email = forms.EmailField(label='Ваш Email')
6     message = forms.CharField(label='Сообщение', widget=forms.Textarea)
```

Рисунок 5 – Файл form.py

## 1.2. Frontend

Далее было написание frontend части и были созданы файлы:

index.html, который является основным и является главной страницей сайта;

about.html, содержащий информацию о сайте и разработчике;

contact.html представляет собой страницу с формой обратной связи;

contact\_succeses.html представляет собой код для страницы с сообщением об успешном заполнении формы;

collapsible\_block.html содержит структуру для сворачивающегося блока с линиями, стрелкой и областью для отображения видеоконтента внутри сетки.

press\_hint.html содержит подсказку для пользователя, которая многократно вставляется в файле index.html с помощью шаблонного тега {% include %} для упрощения кода и повторного использования элемента.

styles.css отвечает за стилизацию внешнего вида сайта, включая оформление элементов, цветовую схему и макет страницы;

about.css файл стилизации страницы about.html;

contact.css файл стилизации формой обратной связи на странице contact.html;

script.js отвечает за добавление интерактивности на сайт, включая обработку событий, манипуляцию DOM-элементами и выполнение различных функций с использованием JavaScript;



video\_loader.js отвечает за динамическую загрузку и отображение видеороликов на странице, создавая раскрывающиеся блоки с видеогалереями на основе данных, содержащихся в объекте videoData.

Сайт представляет с собой сборник ныне популярных итальянских мемов с картинками, звуком и видео. На рисунке 6 файла index.html представлено создание структуры сайта, создание хедера, галереи фото и левой панели с информацией.

```
indexhtml x collapsible_blockhtml x styles.css x scriptjs x video_loaderjs x press_hint.html x about.html x contact.html x about.css x c
1 <!DOCTYPE html>
2 <html lang="ru">
3 <head>
4   <meta charset="UTF-8">
5   <title>{% block title %}Мой сайт{% endblock %}</title>
6   {% load static %}
7   <link rel="stylesheet" href="{% static 'css/styles.css' %}">
8   {% block extra_css %}{% endblock %}
9 </head>
10 <body>
11   <header>
12     <h1>Memes</h1>
13     <nav>
14       <a href="/" class="{% if request.path == '/' %}active{% endif %}">Главная</a>
15       <a href="/about/" class="{% if request.path == '/about/' %}active{% endif %}">О разработке</a>
16       <a href="/contact/" class="{% if request.path == '/contact/' %}active{% endif %}">Сотрудничество</a>
17     </nav>
18   </header>
19
20   <main>
21     {% block main_content %}
22
23     <div class="container_1">
24       <div class="gallery-container">
25         <!-- Галерея с горизонтальной прокруткой -->
26         <h2 class="gallery-title">Галерея мемов</h2>
27
28         <div class="gallery-scroll">
29           {% for img in gallery_images %}
30             <div class="gallery-item">
31               
32             </div>
33           {% empty %}
34             <!-- Изображения не найдены. Добавьте файлы в static/images/gallery/ -->
35           {% endfor %}
36         </div>
37       </div>
38     </div>
39
40     <div class="container_2">
41       <div class="left_panel_container">
42
43         <div class="photo_n_sound_container">
44
45           <h3 class="meme-title">Bombardino Crocodilo</h3>
46           {% include 'main/press_hint.html' %}
47
48           <div class="meme-image-container">
49             
50
51             <audio id="memeAudio_bombardino_crocodilo" src="{% static 'sound/bombardino-crocodilo.mp3' %}" preload="auto"></audio>
52           </div>
53           <div id="container-bombardino_crocodilo"></div>
54         </div>
55       </div>
56     </div>
57   </main>
58 </body>
```

Рисунок 6 – Файл index.html. Структура, хедер, галерея и левая панель

В блоке left\_panel\_container создаются отдельные блоки с каждым из мемов. В этих блоках HTML отображаются заголовок с названием мема, затем подключается подсказка из файла press\_hint.html с помощью {% include 'main/press\_hint.html' %}. После этого создается контейнер для изображения и аудиофайла мема, где картинка и звуковой файл подгружаются с использованием тега {% static %} для указания путей к статическим файлам.

После создается пустой контейнер для динамического контента, который будет заполняться видеороликами с помощью JavaScript.

На рисунке 7 представлен код с созданием правой панели информации, футера и подключение скриптов, написанных на языке JavaScript.

```
130         <!-- Правая панель -->
131         <div class="side-panel">
132             <h3>Все звуки с сайта</h3>
133             <p>Здесь будут все звуки с сайта</p>
134         </div>
135
136     </div>
137
138     {% endblock %}
139 </main>
140
141 <footer>
142     <hr>
143     <p>&copy; 2025. Все права защищены.</p>
144 </footer>
145 <script src="{% static 'js/script.js' %}"></script>
146 <script src="{% static 'js/video_loader.js' %}"></script>
147
148
149
150 </body>
151 </html>
152
```

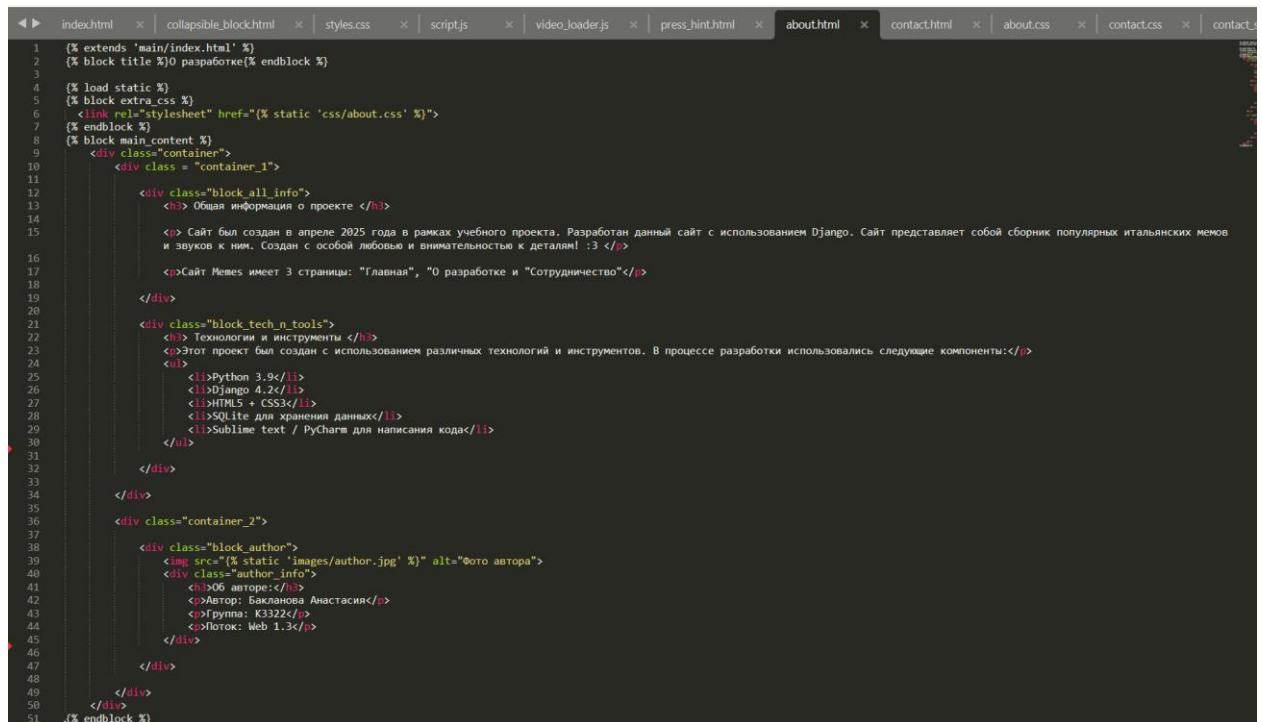
Рисунок 7 – Завершение файла index.html

Файл about.html (Рисунок 8) отвечает за отображение страницы "О разработке" и построен на основе базового шаблона index.html с помощью конструкции `{% extends %}`. Внутри блока `{% block title %}` задается заголовок страницы, а в блоке `{% block extra_css %}` подключается дополнительный файл стилей about.css для оформления именно этой страницы.

Основной контент страницы размещается внутри блока `{% block main_content %}`. Здесь создается разметка с двумя основными контейнерами:

- В первом контейнере (container\_1) размещены блоки с общей информацией о проекте и списком используемых технологий и инструментов (см. фото кода блока информации).
- Во втором контейнере (container\_2) расположен блок с фотографией автора и его кратким описанием (см. фото кода блока автора).

Также используются теги `{% load static %}` и `{% static %}` для правильной подгрузки изображений и CSS-файлов через Django.



```
1 {% extends 'main/index.html' %}
2 {% block title %}О разработке{% endblock %}
3
4 {% load static %}
5 {% block extra_css %}
6 <link rel="stylesheet" href="{% static 'css/about.css' %}">
7 {% endblock %}
8 {% block main_content %}
9 <div class="container">
10 <div class="container_1">
11
12 <div class="block_all_info">
13 <h2>Общая информация о проекте </h2>
14
15 <p>Сайт был создан в апреле 2025 года в рамках учебного проекта. Разработан данный сайт с использованием Django. Сайт представляет собой сборник популярных итальянских мемов и звуков к ним. Создан с особой любовью и внимательностью к деталям! :3 </p>
16
17 <p>Сайт Мемес имеет 3 страницы: "Главная", "О разработке" и "Сотрудничество"</p>
18
19 </div>
20
21 <div class="block_tech_n_tools">
22 <h2>Технологии и инструменты </h2>
23 <p>Этот проект был создан с использованием различных технологий и инструментов. В процессе разработки использовались следующие компоненты:</p>
24 <ul>
25 <li>Python 3.9</li>
26 <li>Django 4.2</li>
27 <li>HTML5 + CSS3</li>
28 <li>SQLite для хранения данных</li>
29 <li>Sublime text / PyCharm для написания кода</li>
30 </ul>
31 </div>
32
33 </div>
34
35 <div class="container_2">
36
37 <div class="block_author">
38 
39 <div class="author_info">
40 <h3>Об авторе:</h3>
41 <p>Автор: Бакланова Анастасия</p>
42 <p>Группа: K3322</p>
43 <p>Поток: Web 1.3</p>
44 </div>
45 </div>
46
47 </div>
48
49 </div>
50 </div>
51 {% endblock %}
```

Рисунок 8 – Файл about.html

Файл `contact.html` (Рисунок 9) формирует страницу "Обратная связь" и использует базовый шаблон `index.html`, расширяя его через `{% extends %}`. В блоке `{% block title %}` задается заголовок страницы, а в `{% block extra_css %}` подключается дополнительный CSS-файл `contact.css` для стилизации формы и текстов.

Основное содержимое страницы помещено в `{% block main_content %}`. На странице размещены два приветственных абзаца с описанием цели формы, а затем создаётся форма обратной связи, которая отправляет данные методом POST на адрес, связанный с именем маршрута `contact` (см. фото кода формы).

Форма генерируется автоматически через шаблонную переменную `{{ form.as_p }}`, которая выводит поля формы в формате абзацев (`<p>`). Также используется тег `{% csrf_token %}` для защиты от CSRF-атак при отправке формы.

```
press_hint | about.html x | contact.html x | about.css x | contact.css x | contact_succes

1  {% extends 'main/index.html' %}
2  {% block title %}Обратная связь{% endblock %}
3
4  {% load static %}
5  {% block extra_css %}
6  <link rel="stylesheet" href="{% static 'css/contact.css' %}">
7  {% endblock %}
8
9  {% block main_content %}
10
11     <p class="p_contact">Понравился сайт, также, как и мы любишь мемы, и хочешь
12     принимать участие в пополнении коллекции популярных мемов?</h2>
13     <p class="p_contact">Заполни форму и мы с тобой свяжемся в ближайшее время!</h3>
14
15     <form method="post" action="{% url 'contact' %}">
16         {% csrf_token %}
17         {{ form.as_p }}
18         <button type="submit">Отправить</button>
19     </form>
20 {% endblock %}
21
```

Рисунок 9 – Файл contact.html

Файл collapsible\_block.html (Рисунок 10) содержит разметку для создания интерактивного, сворачиваемого блока с видео-контентом на сайте. Он состоит из двух основных частей: верхняя часть — это строка с линиями и стрелкой (collapsible-line-container), которая по нажатию вызывает функцию toggleCollapsible(this) для открытия или закрытия содержимого (см. фото кода).

Ниже находится контейнер collapsible-content, куда затем через JavaScript динамически подгружаются видеоролики в сетку video-grid.

```
index.html x | collapsible_block.html x | styles.css x | script.js x | video_loader.js

1  <div class="collapsible-wrapper">
2      <div class="collapsible-line-container" onclick="toggleCollapsible(this)">
3          <div class="collapsible-line"></div>
4          <div class="collapsible-circle">
5              <svg class="arrow-down" width="16" height="16" viewBox="0 0 24 24">
6                  <path fill="currentColor" d="M7 10 5 5 3 10" />
7              </svg>
8          </div>
9          <div class="collapsible-line"></div>
10     </div>
11
12     <div class="collapsible-content">
13         <div class="video-grid">
14         </div>
15     </div>
16
```

Рисунок 10 – Файл collapsible\_block.html

Файл `scripts.js` (Рисунок 11-12) отвечает за добавление интерактивного поведения на сайт.

- Автопрокрутка галереи: в начале скрипта автоматически запускается плавная прокрутка блока с изображениями (`gallery-scroll`). Содержимое галереи клонируется, чтобы создать эффект бесконечной ленты. При наведении мыши или касании прокрутка приостанавливается, а при уходе — возобновляется (см. фрагмент с функцией `animateScroll` на фото).
- Аудио-проигрывание: каждому изображению мемов привязано аудио. При клике на картинку начинается воспроизведение соответствующего звука или его остановка (см. обработку события клика на картинке).
- Плавный переход к мемам: при нажатии на миниатюру из галереи страница плавно прокручивается к основному изображению мема, а сам мем визуально подсвечивается (см. обработчик кликов по элементам `gallery-item img`).
- Открытие и закрытие контента: добавляется функция `toggleCollapsible`, позволяющая скрывать или показывать видео-блоки при нажатии на стрелку.

```
index.html x collapsible_block.html x styles.css x script.js x video_loader.js
1 document.addEventListener("DOMContentLoaded", function () {
2
3     // === АВТО-ПРОКРУТКА ГАЛЕРЕИ ===
4     const galleryScroll = document.querySelector(".gallery-scroll");
5
6     // Клонировем всё содержимое галереи
7     const galleryContent = galleryScroll.innerHTML;
8     galleryScroll.innerHTML += galleryContent;
9
10    let scrollSpeed = 0.5; // скорость прокрутки
11    let scrollPosition = 0;
12    let isPaused = false;
13
14    function animateScroll() {
15        if (!isPaused) {
16            scrollPosition += scrollSpeed;
17            if (scrollPosition >= galleryScroll.scrollWidth / 2) {
18                scrollPosition = 0;
19            }
20            galleryScroll.scrollLeft = scrollPosition;
21        }
22        requestAnimationFrame(animateScroll);
23    }
24
25    animateScroll();
26
27    // Остановить при взаимодействии
28    ["mouseenter", "touchstart", "mousedown"].forEach(event => {
29        galleryScroll.addEventListener(event, () => {
30            isPaused = true;
31        });
32    });
33
34    ["mouseleave", "touchend", "mouseup"].forEach(event => {
35        galleryScroll.addEventListener(event, () => {
36            isPaused = false;
37        });
38    });
39
40
41
42    // === АУДИО ===
43    const imageElements = document.querySelectorAll("img[id^='image-']");
44    imageElements.forEach(img => {
45        const idSuffix = img.id.replace('image-', '');
46        const audio = document.getElementById(`memeAudio_${idSuffix}`);
47        if (audio) {
48            img.addEventListener("click", () => {
49                if (audio.paused) {
50                    audio.play();
51                } else {
52                    audio.pause();
53                    audio.currentTime = 0;
54                }
55            });
56        }
57    });
58
```

Рисунок 11 – Файл scripts.js

```

59 // === ПЕРЕХОД К ИЗОБРАЖЕНИЮ ===
60 const galleryImages = document.querySelectorAll(".gallery-item img");
61 galleryImages.forEach(item => {
62     item.addEventListener("click", () => {
63         const name = item.getAttribute("data-name");
64         const target = document.getElementById(`image-${name}`);
65         if (target) {
66             target.scrollIntoView({ behavior: 'smooth', block: 'center' });
67             target.classList.add('highlight');
68             setTimeout(() => target.classList.remove('highlight'), 1500);
69         }
70     });
71 });
72
73
74
75 // === КОЛЛАПС ===
76 window.toggleCollapsible = function (elem) {
77     const content = elem.nextElementSibling;
78     content.style.display = (content.style.display === "block") ? "none" : "block";
79 };
80
81
82
83
84 });
85

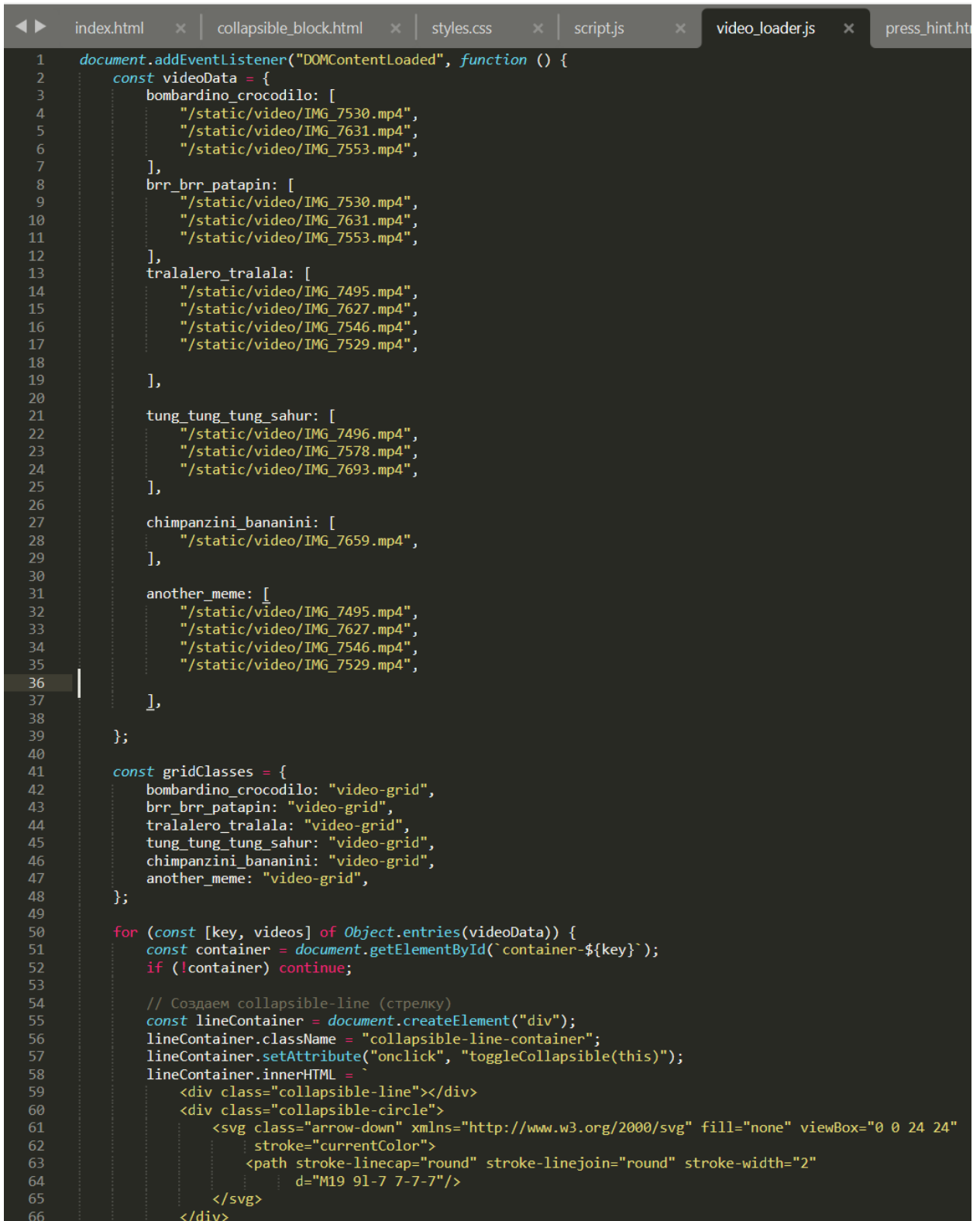
```

Рисунок 12 – Файл scripts.js. Продолжение

Файл video\_loader.js (Рисунок 13-14) автоматически создаёт и выводит на страницу списки видео для каждого мема, а также добавляет возможность их скрывать и показывать.

- Определение данных: в начале файла создаются два объекта — videoData, который хранит пути к видеофайлам для каждого мема, и gridClasses, который задаёт CSS-класс для сетки размещения видео (см. блок объявления переменных на фото).
- Генерация блоков с видео: далее скрипт проходит по всем записям в videoData. Для каждого мема:
  - находит соответствующий контейнер (container-имя\_мема);
  - создаёт элемент стрелки (collapsible-line-container) для сворачивания/разворачивания видео-блока;
  - создаёт скрытый блок с сеткой (collapsible-content), куда вставляет все видеофайлы, привязанные к этому мему (см. фрагмент с for (const [key, videos] of Object.entries(videoData))).

- Функциональность сворачивания: каждому созданному блоку с видео автоматически добавляется обработчик события onclick, чтобы при нажатии стрелка раскрывала или скрывала содержимое блока.



```
1 document.addEventListener("DOMContentLoaded", function () {
2     const videoData = {
3         bombardino_crocodilo: [
4             "/static/video/IMG_7530.mp4",
5             "/static/video/IMG_7631.mp4",
6             "/static/video/IMG_7553.mp4",
7         ],
8         brr_brr_patapin: [
9             "/static/video/IMG_7530.mp4",
10            "/static/video/IMG_7631.mp4",
11            "/static/video/IMG_7553.mp4",
12        ],
13        tralalero_tralala: [
14            "/static/video/IMG_7495.mp4",
15            "/static/video/IMG_7627.mp4",
16            "/static/video/IMG_7546.mp4",
17            "/static/video/IMG_7529.mp4",
18        ],
19        tung_tung_tung_sahur: [
20            "/static/video/IMG_7496.mp4",
21            "/static/video/IMG_7578.mp4",
22            "/static/video/IMG_7693.mp4",
23        ],
24        chimpanzini_bananini: [
25            "/static/video/IMG_7659.mp4",
26        ],
27        another_meme: [
28            "/static/video/IMG_7495.mp4",
29            "/static/video/IMG_7627.mp4",
30            "/static/video/IMG_7546.mp4",
31            "/static/video/IMG_7529.mp4",
32        ],
33    },
34    gridClasses = {
35        bombardino_crocodilo: "video-grid",
36        brr_brr_patapin: "video-grid",
37        tralalero_tralala: "video-grid",
38        tung_tung_tung_sahur: "video-grid",
39        chimpanzini_bananini: "video-grid",
40        another_meme: "video-grid",
41    },
42    for (const [key, videos] of Object.entries(videoData)) {
43        const container = document.getElementById(`container-${key}`);
44        if (!container) continue;
45
46        // Создаем collapsible-line (стрелку)
47        const lineContainer = document.createElement("div");
48        lineContainer.className = "collapsible-line-container";
49        lineContainer.setAttribute("onclick", "toggleCollapsible(this)");
50        lineContainer.innerHTML = `
51            <div class="collapsible-line"></div>
52            <div class="collapsible-circle">
53                <svg class="arrow-down" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
54                    stroke="currentColor">
55                    <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2"
56                        d="M19 9l-7 7-7-7"/>
57                </svg>
58            </div>`
59    }
60 }
```

Рисунок 13 – Файл video\_loader.js



```

67         <div class="collapsible-line"></div>
68     ~
69     ;
70     const collapsibleContent = document.createElement("div");
71     collapsibleContent.className = "collapsible-content";
72
73     const gridClass = gridClasses[key] || "video-grid-default";
74     const grid = document.createElement("div");
75     grid.className = gridClass;
76
77     videos.forEach(src => {
78         const videoItem = document.createElement("div");
79         videoItem.className = "video-item";
80
81         const video = document.createElement("video");
82         video.controls = true;
83         video.src = src;
84
85         videoItem.appendChild(video);
86         grid.appendChild(videoItem);
87     });
88
89     collapsibleContent.appendChild(grid);
90
91     container.appendChild(lineContainer);
92     container.appendChild(collapsibleContent);
93 }
94 });
95

```

Рисунок 14 – Файл video\_loader.js. Продолжение

Главная страница представлена на рисунке 15.

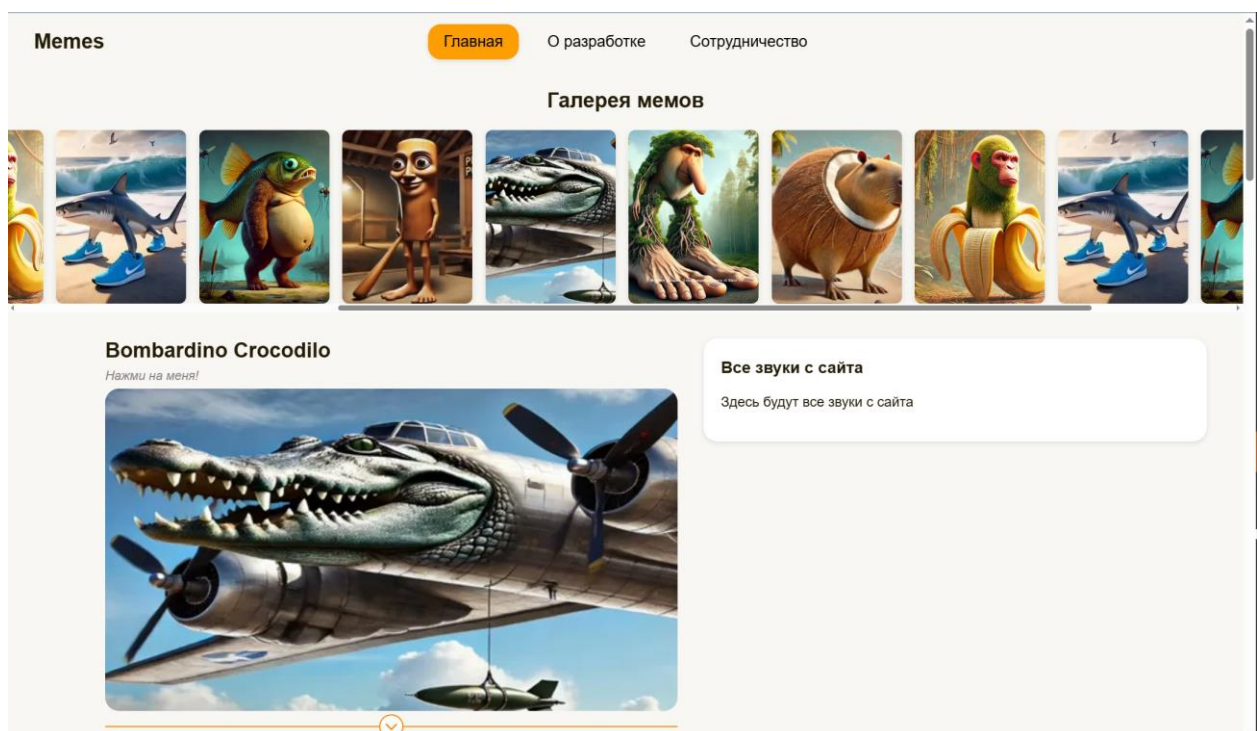


Рисунок 15 – Главная страница

Выпадающий блок с видео представлен на рисунке 16.

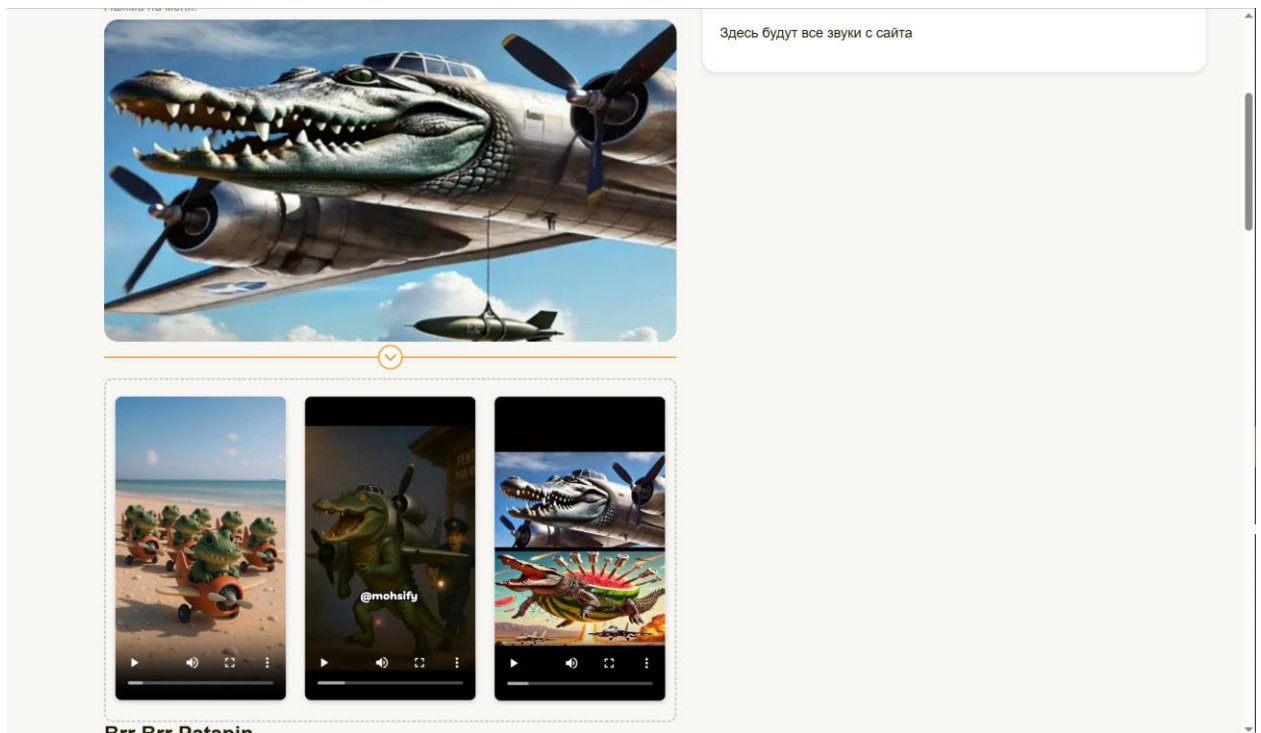


Рисунок 16 – Выпадающий блок с видео

Странице с информацией о проекте и разработчике представлена на рисунке 17.

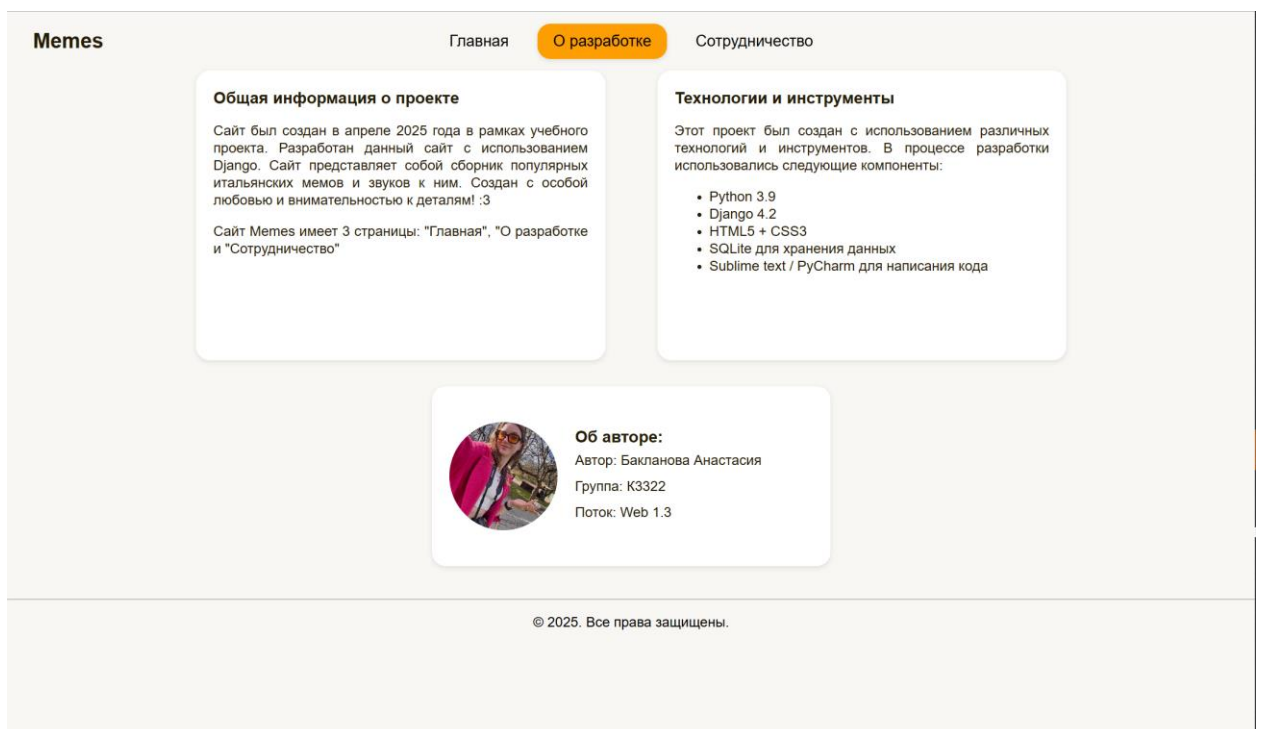


Рисунок 17 – Страница «О разработке»

Форма обратной связи располагается на странице «Сотрудничество» (Рисунок 18).

Memes

Главная
О разработке
Сотрудничество

Понравился сайт, также, как и мы любим мемы, и хочешь принимать участие в пополнении коллекции популярных мемов?

Заполни форму и мы с тобой свяжемся в ближайшее время!

Ваше имя:

Ваш Email:

Сообщение:

Отправить

© 2025. Все права защищены.

Рисунок 18 – Страница «Сотрудничество»

После заполнения и отправки формы информация отображается в таблице БД main\_contactmessage (Рисунок 19).

DB Browser for SQLite - D:\Users\User\PycharmProjects\mysite\db.sqlite3

Файл Редактирование Вид Инструменты Справка

New Database Open Database Записать изменения Отменить изменения Undo Open Project Save Project Прикрепить БД Закрывать базу данных

Database Structure Browse Data Edit Pragas Execute SQL

Таблица: main\_contactmessage

id	name	email	message
1	Анастасия	nastya.baklanova004@gmail.com	Хочу работать с вами!
8	Anastasia Baklanova	nastya.baklanova.04@mail.com	jhvjcjcg

Редактирование ячейки БД

Режим: Текст

Editing row=1, column=1  
Type: Text / Numeric; Size: 1 character(s)

Удаленный сервер

ID Select an identity to connect

DBHub.io Local Current Database

Имя Изменен

Журнал SQL График Схема БД Удаленный сервер

UTF-8

Рисунок 19 – База данных

## **Заключение**

В седьмой практической работе создан проект на django с нуля. Проект содержит три страницы: главная, о разработке, сотрудничество. У всех трех страниц одинаковые футеры и хидеры, реализованные через шаблоны, с использованием возможностей django разработана форма обратной связи на третьей странице, и реализована обработка данных формы.