Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(МОСКОВСКИЙ ПОЛИТЕХ)

Факультет информационных технологий

Кафедра Инфокогнитивные технологии

09.03.01 «Информатика и вычислительная техника»

Образовательная программа (профиль) «Веб-технологии»

**Курсовая работа**

**«**Разработка веб-сайта, где музыканты будут публиковать свою музыку и предоставлять разработчикам игр возможность использовать ее.»

Выполнили: Романов Александр Дмитриевич

Каматали Алина Аароновна  
Казаков Дмитрий Александрович   
Музыченко Сергей Кириллович

Группа: 241-321

Проверил: Красникова Ирина Николаевна

МОСКВА 2024

**Курсовая работа**

**«РАЗРАБОТКА ВЕБ-САЙТА, ПРЕДОСТАВЛЯЮЩЕГО БЕСПЛАТНЫЙ ДОСТУП К МУЗЫКЕ»**

**Цель работы:** получить практические навыки самостоятельной реализации программного продукта. Закрепление теоретических знаний по дисциплине «Основы программирования», а также практических навыков по программированию на языке Python.

**Постановка задачи:** тематика проектов разрабатывается, утверждается и ежегодно, актуализируется кафедрой ИКТ. В рамках предлагаемой тематики студентам предоставляется право выбора темы, но ее необходимо согласовать с преподавателем. Студенты должны распределиться на команды из 4 человек. 1 команда выбирает 1 тему. Внутри команды студенты разбиваются на 2 подкоманды и разрабатывают проект на конкурентной основе.

**АННОТАЦИЯ**

Работа состоит из: введения, двух глав, заключения и списка используемых источников.

Во введении обоснована актуальность темы исследования, цель и задачи.

В рамках первой главы курсовой работы было уделено внимание разработке серверной части проекта “OpenGameMusic”. Основным инструментом для реализации этой задачи был выбран веб-фреймворк Django, основанный на языке программирования Python.

Вторая глава курсовой работы была посвящена разработке пользовательского интерфейса (фронтенда) веб-сайта “OpenGameMusic” и интеграции его с серверной частью. Работа включала в себя изучение и применение HTML, CSS, а также понимание взаимодействия с сервером через API.

В заключении приведены основные выводы и результаты проделанной работы.

Ключевые слова: таблица базы данных, языки программирования, фронтенд, бэкенд.

Общий объём работы 23 страницы.

**1. Введение**

1.1. Наименование программы

Наименование программы: "OpenGameMusic"

1.2. Назначение работы и область применения

Веб-сайт, предоставляющий возможность музыкантам публиковать свою музыку и предоставлять разработчикам игр возможность использовать её. Сайт будет полезен музыкантам, разработчикам игр, студентам музыкальных учебных заведений и другим.

1.3. Основная цель работы – получить практические навыки самостоятельной реализации программного продукта. Закрепление теоретических знаний по дисциплине «Основы программирования», а также практических навыков по программированию на языке Python.

**2. Основные результаты выполнения работы**

Основные этапы выполнения работы, а также результаты представлены в главах 1 и 2.

**Глава первая**

В данной главе описан процесс создания серверной части веб-приложения “OpenGameMusic”. Основным инструментом для реализации был выбран веб-фреймворк Django, который основан на языке Python.

Этапы выполнения работы включали:

1. Проектирование базы данных – разработана структура базы MySQL, обеспечивающая хранение данных о пользователях (музыкантах и разработчиках), музыкальных произведениях, категориях, жанрах и других необходимых элементах. Определены таблицы, поля и взаимосвязи между ними.
2. Создание моделей Django – на основе структуры базы данных созданы Django-модели, представляющие собой Python-классы, обеспечивающие удобное взаимодействие с базой через объектно-ориентированный подход.
3. Настройка Django-проекта – произведена настройка проекта, включая установку необходимых пакетов, подключение к MySQL, создание приложений для управления пользователями и музыкальными произведениями.
4. Административный интерфейс – настроен встроенный административный интерфейс Django для удобного управления пользователями, музыкальными файлами и другими сущностями.
5. Разработка API – реализованы API-эндпоинты, позволяющие фронтенду взаимодействовать с сервером. Включены методы для работы с музыкальными произведениями, аутентификации пользователей и управления контентом.

В результате выполнения данной главы был создан серверный фундамент для веб-платформы, обеспечивающий хранение и обработку данных, а также взаимодействие с пользователями через API.

**Глава вторая**

**HTML, CSS, работа с сервером**

Вторая глава посвящена созданию фронтенда веб-приложения “OpenGameMusic” и его интеграции с серверной частью. Работа включала в себя изучение HTML, CSS, а также взаимодействие с сервером через API.

Основные этапы:

1. **Верстка HTML-страниц** – разработана структура страниц, включая просмотр музыкальных композиций, личные кабинеты пользователей, формы загрузки и редактирования музыки, а также главную страницу сайта.
2. **Стилизация с помощью CSS** – оформлены элементы интерфейса, обеспечена адаптивная верстка для корректного отображения на различных устройствах (ПК, планшетах, смартфонах).
3. **Интеграция с сервером** – реализовано взаимодействие фронтенда с API, разработанным в первой главе. Настроены запросы к серверу для загрузки данных, отправки пользовательских действий и динамического обновления информации.

В результате выполнения второй главы был создан полноценный пользовательский интерфейс веб-сайта “OpenGameMusic”, обеспечивающий удобное взаимодействие пользователей с системой.

**3. Заключение**

В ходе выполнения работы были изучены и применены на практике следующие технологии: Django (Python), MySQL, HTML, CSS, а также принципы организации базы данных и работы с веб-фреймворком.

Была проведена систематизация и оформление полученных знаний в виде отчета, описывающего все этапы разработки. Итогом работы стал веб-сайт "OpenGameMusic", предоставляющий возможность музыкантам загружать свою музыку, а разработчикам игр – находить и использовать её.

**4. Список литературы**

**I. Основы программирования и Python:**

1. Книга Эрика Мэтиза “Python Crash Course: A Hands-On, Project-Based Introduction to Programming”

2. Книга Марка Саммерфилд “Программирование на Python” «Изучаем Python. Программирование игр, визуализация данных, веб-приложения»

3. [Официальная документация Python](https://docs.python.org/3/)

4. [Официальная документация Django](https://docs.djangoproject.com/)

**II. Базы данных и MySQL:**

5. “SQL for Dummies” by Allen G. Taylor.

6. [Официальная документация MySQL](https://dev.mysql.com/doc/)

**III. Фронтенд-разработка (HTML, CSS)**

7. [Официальная документация HTML](https://developer.mozilla.org/en-US/docs/Web/HTML)

8. [Официальная документация CSS](https://developer.mozilla.org/en-US/docs/Web/CSS)

**IV. Дополнительные ресурсы:**

9. Stack Overflow: <https://stackoverflow.com/>

10. GitHub: <https://github.com/>

11. Medium: <https://medium.com/>

**5. Листинг программы**

Далее (весь оставшийся документ) будет приведён полный листинг веб-сайта, а именно файлов с разрешением .py, .html и .css.

manage.py

import os

import sys

def main():

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

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you "

"forget to activate a virtual environment?"

) from exc

execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

main()

statup.py

import os

import subprocess

import platform

def run\_django\_server():

try:

# Construct the command to run the Django server

command = "python manage.py runserver"

# Check the operating system and modify command if needed

if platform.system() == "Windows":

command = "cmd /c " + command

#Execute the command in terminal

os.system(command)

except Exception as e:

print(f"An unexpected error occurred: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

run\_django\_server()

myapp/apps.py

from django.apps import AppConfig

class MyappConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'myapp'

myapp/forms.py

from django import forms

from .models import Song, Author, Genre

class UploadSongForm(forms.ModelForm):

username = forms.CharField(max\_length=255, required=True, initial="")

class Meta:

model = Song

fields = ['file', 'title', 'username']

myapp/models.py

from django.db import models

from django.utils import timezone

class Author(models.Model):

name = models.CharField(max\_length=255, unique=True)

def \_\_str\_\_(self):

return self.name

class Genre(models.Model):

name = models.CharField(max\_length=255, unique=True)

def \_\_str\_\_(self):

return self.name

class Song(models.Model):

file = models.FileField(upload\_to='uploads/')

title = models.CharField(max\_length=255)

username = models.CharField(max\_length=255, default='Anonymous') # Replaced ForeignKey with CharField

authors = models.ManyToManyField(Author, through='SongAuthor')

genres = models.ManyToManyField(Genre, through='SongGenre')

uploaded\_at = models.DateTimeField(default=timezone.now)

def \_\_str\_\_(self):

return f"{self.title} by {', '.join([author.name for author in self.authors.all()])} ({', '.join([genre.name for genre in self.genres.all()])})"

class SongGenre(models.Model):

song = models.ForeignKey(Song, on\_delete=models.CASCADE)

genre = models.ForeignKey(Genre, on\_delete=models.CASCADE)

class Meta:

unique\_together = ('song', 'genre')

class SongAuthor(models.Model):

song = models.ForeignKey(Song, on\_delete=models.CASCADE)

author = models.ForeignKey(Author, on\_delete=models.CASCADE)

class Meta:

unique\_together = ('song', 'author')

myapp/urls.py

from django.urls import path

from .views import upload\_and\_list\_files

urlpatterns = [

path('upload\_and\_list.html/', upload\_and\_list\_files, name='upload\_and\_list\_files'),

]

myapp/views.py

from django.shortcuts import render, redirect

from django.core.paginator import Paginator

from django.db import models

from .forms import UploadSongForm

from .models import Song, Author, Genre

import os

from django.conf import settings

def upload\_and\_list\_files(request):

if request.method == 'POST':

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

if form.is\_valid():

title = form.cleaned\_data['title']

username = form.cleaned\_data['username']

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

authors\_data = request.POST.getlist('authors[]')

genres\_data = request.POST.getlist('genres[]')

song = Song.objects.create(title=title, username=username, file=file)

authors = [Author.objects.get\_or\_create(name=author)[0] for author in authors\_data if author]

genres = [Genre.objects.get\_or\_create(name=genre)[0] for genre in genres\_data if genre]

song.authors.set(authors)

song.genres.set(genres)

return redirect('upload\_and\_list\_files')

else:

form = UploadSongForm()

for song in Song.objects.all():

file\_path = os.path.join(settings.MEDIA\_ROOT, str(song.file))

if not os.path.exists(file\_path):

song.delete()

query = request.GET.get('q', '')

songs = Song.objects.filter(

models.Q(title\_\_icontains=query) |

models.Q(authors\_\_name\_\_icontains=query) |

models.Q(genres\_\_name\_\_icontains=query) |

models.Q(username\_\_icontains=query)

).distinct()

per\_page = request.GET.get('per\_page', 5)

try:

per\_page = int(per\_page) if per\_page else 5

except ValueError:

per\_page = 5

paginator = Paginator(songs, per\_page)

page\_number = request.GET.get('page')

page\_obj = paginator.get\_page(page\_number)

return render(request, 'upload\_and\_list.html', {'form': form, 'page\_obj': page\_obj, 'query': query})

myproject/asgi.py

import os

from django.core.asgi import get\_asgi\_application

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

application = get\_asgi\_application()

myproject/settings.py

import os

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, 'static'),]

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-l$@h50\_#1g-dm\_r\_m\*tl#y%j$no0!2@p6=wm6!u678akw9sds@'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = False

ALLOWED\_HOSTS = []

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'myapp',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'myproject.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'myproject.wsgi.application'

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

STATIC\_URL = 'static/'

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

MEDIA\_URL = '/media/'

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media/')

myproject/urls.py

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('myapp.urls')),

]

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

myproject/wsgi.py

import os

from django.core.wsgi import get\_wsgi\_application

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

application = get\_wsgi\_application()

myapp/templates/upload\_and\_list.html

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

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

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

<link rel="icon" type="image/x-icon" href="{% static 'images/noti-cropped.svg' %}">

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>OpenGameMusic</title>

</head>

<body>

<div class="header">

<img src="{% static 'images/noti.svg' %}" alt="" class="logo">

</div>

<h1>Upload Song</h1>

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

{% csrf\_token %}

{{ form.file }}

{{ form.title.as\_widget }}

{{ form.username.as\_widget }}

<div id="authors">

<div id="author-fields">

<input type="text" name="authors[]" placeholder="Author">

</div>

<div>

<button type="button" onclick="addAuthorField()">Add Author</button>

<button type="button" onclick="removeLastAuthorField()">Delete Author</button>

</div>

</div>

<div id="genres">

<div id="genre-fields">

<input type="text" name="genres[]" placeholder="Genre">

</div>

<div>

<button type="button" onclick="addGenreField()">Add Genre</button>

<button type="button" onclick="removeLastGenreField()">Delete Genre</button>

</div>

</div>

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

<button type="reset">Reset</button>

</form>

<h1>Downloads</h1>

<form method="get">

<input type="text" name="q" value="{{ query }}" placeholder="Search...">

<select name="per\_page">

<option value="5" {% if request.GET.per\_page == "5" %}selected{% endif %}>5</option>

<option value="10" {% if request.GET.per\_page == "10" %}selected{% endif %}>10</option>

<option value="20" {% if request.GET.per\_page == "20" %}selected{% endif %}>20</option>

</select>

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

</form>

<div>

{% for song in page\_obj %}

<div>

<strong>File:</strong> <a href="{{ song.file.url }}">{{ song.file.name }}</a><br>

<strong>Title:</strong> {{ song.title }}<br>

<strong>Authors:</strong> {{ song.authors.all|join:", " }}<br>

<strong>Genres:</strong> {{ song.genres.all|join:", " }}<br>

<strong>User:</strong> {{ song.username }}<br>

<strong>Uploaded at:</strong> {{ song.uploaded\_at }}<br>

</div>

<hr>

{% empty %}

<p>No songs found.</p>

{% endfor %}

</div>

<div class="pagination">

<span>Page {{ page\_obj.number }} of {{ page\_obj.paginator.num\_pages }}</span>

<div>

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

<a href="?page=1&per\_page={{ request.GET.per\_page }}&q={{ query }}">First</a>

<a href="?page={{ page\_obj.previous\_page\_number }}&per\_page={{ request.GET.per\_page }}&q={{ query }}">Previous</a>

{% endif %}

{% for num in page\_obj.paginator.page\_range %}

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

<strong>{{ num }}</strong>

{% elif num > page\_obj.number|add:-2 and num < page\_obj.number|add:2 %}

<a href="?page={{ num }}&per\_page={{ request.GET.per\_page }}&q={{ query }}">{{ num }}</a>

{% elif num == 1 or num == page\_obj.paginator.num\_pages %}

<a href="?page={{ num }}&per\_page={{ request.GET.per\_page }}&q={{ query }}">{{ num }}</a>

{% elif num == page\_obj.number|add:-3 or num == page\_obj.number|add:3 %}

<span>...</span>

{% endif %}

{% endfor %}

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

<a href="?page={{ page\_obj.next\_page\_number }}&per\_page={{ request.GET.per\_page }}&q={{ query }}">Next</a>

<a href="?page={{ page\_obj.paginator.num\_pages }}&per\_page={{ request.GET.per\_page }}&q={{ query }}">Last</a>

{% endif %}

</div>

</div>

<script>

function addAuthorField() {

const authorsDiv = document.getElementById('author-fields');

const newInput = document.createElement('input');

newInput.type = 'text';

newInput.name = 'authors[]';

newInput.placeholder = 'Author';

authorsDiv.appendChild(newInput);

}

function removeLastAuthorField() {

const authorsDiv = document.getElementById('author-fields');

const inputFields = authorsDiv.querySelectorAll('input[name="authors[]"]');

if (inputFields.length > 1) {

authorsDiv.removeChild(inputFields[inputFields.length - 1]);

}

}

function addGenreField() {

const genresDiv = document.getElementById('genre-fields');

const newInput = document.createElement('input');

newInput.type = 'text';

newInput.name = 'genres[]';

newInput.placeholder = 'Genre';

genresDiv.appendChild(newInput);

}

function removeLastGenreField() {

const genresDiv = document.getElementById('genre-fields');

const inputFields = genresDiv.querySelectorAll('input[name="genres[]"]');

if (inputFields.length > 1) {

genresDiv.removeChild(inputFields[inputFields.length - 1]);

}

}

document.addEventListener('DOMContentLoaded', function() {

document.getElementById('id\_title').placeholder = 'Song Title';

document.getElementById('id\_username').placeholder = 'Username';

});

</script>

</body>

</html>

Static/normalize.css

html {

line-height: 1.15;

-webkit-text-size-adjust: 100%;

}

main {

display: block;

}

h1 {

font-size: 2em;

margin: 0.67em 0;

}

hr {

box-sizing: content-box;

height: 0;

overflow: visible;

}

pre {

font-family: monospace, monospace;

font-size: 1em;

}

a {

background-color: transparent;

}

abbr[title] {

border-bottom: none;

text-decoration: underline;

text-decoration: underline dotted;

}

b,

strong {

font-weight: bolder;

}

code,

kbd,

samp {

font-family: monospace, monospace;

font-size: 1em;

}

small {

font-size: 80%;

}

sub,

sup {

font-size: 75%;

line-height: 0;

position: relative;

vertical-align: baseline;

}

sub {

bottom: -0.25em;

}

sup {

top: -0.5em;

}

img {

border-style: none;

}

button,

input,

optgroup,

select,

textarea {

font-family: inherit;

font-size: 100%;

line-height: 1.15;

margin: 0;

}

button,

input {

overflow: visible;

}

button,

select {

text-transform: none;

}

button,

[type="button"],

[type="reset"],

[type="submit"] {

-webkit-appearance: button;

}

button::-moz-focus-inner,

[type="button"]::-moz-focus-inner,

[type="reset"]::-moz-focus-inner,

[type="submit"]::-moz-focus-inner {

border-style: none;

padding: 0;

}

button:-moz-focusring,

[type="button"]:-moz-focusring,

[type="reset"]:-moz-focusring,

[type="submit"]:-moz-focusring {

outline: 1px dotted ButtonText;

}

fieldset {

padding: 0.35em 0.75em 0.625em;

}

legend {

box-sizing: border-box;

color: inherit;

display: table;

max-width: 100%;

padding: 0;

white-space: normal;

}

progress {

vertical-align: baseline;

}

textarea {

overflow: auto;

}

[type="checkbox"],

[type="radio"] {

box-sizing: border-box;

padding: 0;

}

[type="number"]::-webkit-inner-spin-button,

[type="number"]::-webkit-outer-spin-button {

height: auto;

}

[type="search"] {

-webkit-appearance: textfield;

outline-offset: -2px;

}

[type="search"]::-webkit-search-decoration {

-webkit-appearance: none;

}

::-webkit-file-upload-button {

-webkit-appearance: button;

font: inherit;

}

details {

display: block;

}

summary {

display: list-item;

}

template {

display: none;

}

[hidden] {

display: none;

}

p,

h1,

h2,

h3 {

margin: 0;

padding: 0;

}

ul,

ol {

padding: 0;

margin: 0;

}

static/style.css

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

line-height: 1.6;

background-color: #f4f4f4;

color: #333;

}

.header {

background-color: #007bff;

color: white;

padding: 15px;

text-align: center;

}

.logo {

max-width: 400px;

filter: invert(100%);

}

.logo\_caption {

font-size: 2rem;

font-weight: bold;

}

h1 {

text-align: center;

margin: 20px 0;

font-size: 1.5rem;

}

form {

max-width: 600px;

margin: 0 auto 20px;

padding: 20px;

background: white;

border-radius: 8px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

}

form input[type="text"],

form select {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 4px;

}

form button {

padding: 10px 15px;

background-color: #007bff;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

transition: background-color 0.3s;

}

form button:hover {

background-color: #0056b3;

}

div {

max-width: 600px;

margin: 0 auto;

padding: 10px;

background: white;

border-radius: 8px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

margin-bottom: 20px;

}

strong {

display: inline-block;

margin-top: 10px;

}

.pagination {

text-align: center;

margin: 0 auto;

}

.pagination a {

margin: 0 5px;

padding: 5px 10px;

background-color: #007bff;

color: white;

text-decoration: none;

border-radius: 4px;

transition: background-color 0.3s;

}

.pagination a:hover {

background-color: #0056b3;

}

.pagination strong {

margin: 0 5px;

}