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

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Проектно-исследовательская работа

на тему «Сайт для экотуризма в России»

Обучающиеся 10Б ГБОУ Школы №1748:

Смирнов Александр Денисович

Андрияшкин Владимир Сергеевич

Мамонтов Ростислав Валерьевич

Руководитель от Университета:

Гришина Арина Александровна

Москва 2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ……....3

Актуальность работы……...3

Цель…….……....4

Задачи…….....4

ОСНОВНАЯ ЧАСТЬ…….5

1. Разработка структуры и дизайна…….……5

1.1 Составляющие…….……...5

1.2 Структура сайта и проектирование диаграмм……..…5

1.3 Выбор графического редактора…….…7

1.4 Создание интерфейса……...…….……..8

2. Создание программного обеспечения и верстки……11

2.1 Среда разработки Visual Studio Code с использованием фреймворка Django…….…….…….11

2.2 Фреймворк Django …….……11

2.3 Верстка сайта и принципы строения каждой страницы..11

3. Работа с базами данных……13

4. Тестирование и комментарии……..18

ЗАКЛЮЧЕНИЕ…….…....20

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ…….....21

ВВЕДЕНИЕ

Актуальность работы:

В современном мире все больше людей стремятся проводить время на природе, наслаждаться её красотой и гармонией. Экологический туризм становится популярным направлением отдыха, особенно среди тех, кто устал от шума мегаполисов и хочет найти уединение с природой. Проект «Сайт для экотуризма в России» направлен на создание удобной онлайн-платформы, которая поможет пользователям находить информацию о природных достопримечательностях нашей страны, получать полезные советы по организации экологических туров.

По данным Росзаповедцентра, Ростуризма, Минприроды РФ на рисунках 1 и 2 изображена статистика развития экотуризма.

Рисунок 1 — Поток экотуристов (млн. чел) в России

Рисунок 2 — Поток экотуристов в России (соотношение в %)

Цель:

Создать сайт, который будет предоставлять информацию об уникальных местах природы России и знакомить с новостями в сфере экотуризма.

Задачи работы:

1) Разработать дизайн сайта, подобрать подходящую цветовую гамму, умеренно расположить объекты на странице для понятного пользования;

2) Создать верстку веб-страниц и обеспечить навигацией или передвижением по всем разделам сайта, то есть подготовить Front-end;

3) Разработать Back-end, то есть провести работу с базами данных, добавлением, изменением или удалением динамичной информации, например, новостей или объявлений об обновлениях на сайте;

4) Провести функциональное тестирование всех функций сайта, проверить его верную работоспособность;

5) Собрать информацию об обратной связи для подведения итогов, то есть провести оценочное тестирование.

ОСНОВНАЯ ЧАСТЬ

Процесс создания сайта можно разделить на 3 подпроекта: первый — разработка дизайна сайта, второй — создание программного обеспечения сайта или верстки, третий — создание базы данных и диаграмм.

1. Разработка структуры и дизайна

1.1 Составляющие

- Популярные точки экотуризма (краткое и подробное описание) — этот раздел служит для изучения краткой информации об точках экотуризма в России. Это, к примеру, фотографии разных мест точки экотуризма, краткая информация об месте и его особенностях.

- Новости — на сайте будут показываться интересные статьи об точках экотуризма.

- Обновления — представляет собой страницу с обновлениями и обращениями к пользователям о будущих изменениях.

- Регистрация — на сайте для удобства пользователей предусмотрена функция регистрации, для доступа к сайту с разных устройств.

- Поддержка — на сайте есть поддержка, к которой может обратиться пользователь, чтобы сообщить об ошибке в работе сайта, предложить идею.

1.2 Структура сайта и проектирование диаграмм:

Существует несколько видов диаграмм:

- ER-диаграмма (схема «сущность-связь») — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса.

- Диаграмма состояний — это тип диаграммы, используемый в языке UML для описания поведения систем. Она отображает разрешённые состояния и переходы, а также события, которые влияют на эти переходы. Диаграммы состояний используются в дизайне интерфейсов, чтобы показать, как интерфейс меняется в ответ на действия пользователя. Это помогает разработчикам создать более удобный пользовательский опыт.

ER-диаграммы делятся на концептуальные и физические. В отличие от физических, в концептуальных ER-диаграммах не учитываются особенности конкретной базы данных.

Для понимания структуры сайта на рисунках 3 и 4 изображены схемы, включающие в себя функции обоих диаграмм.

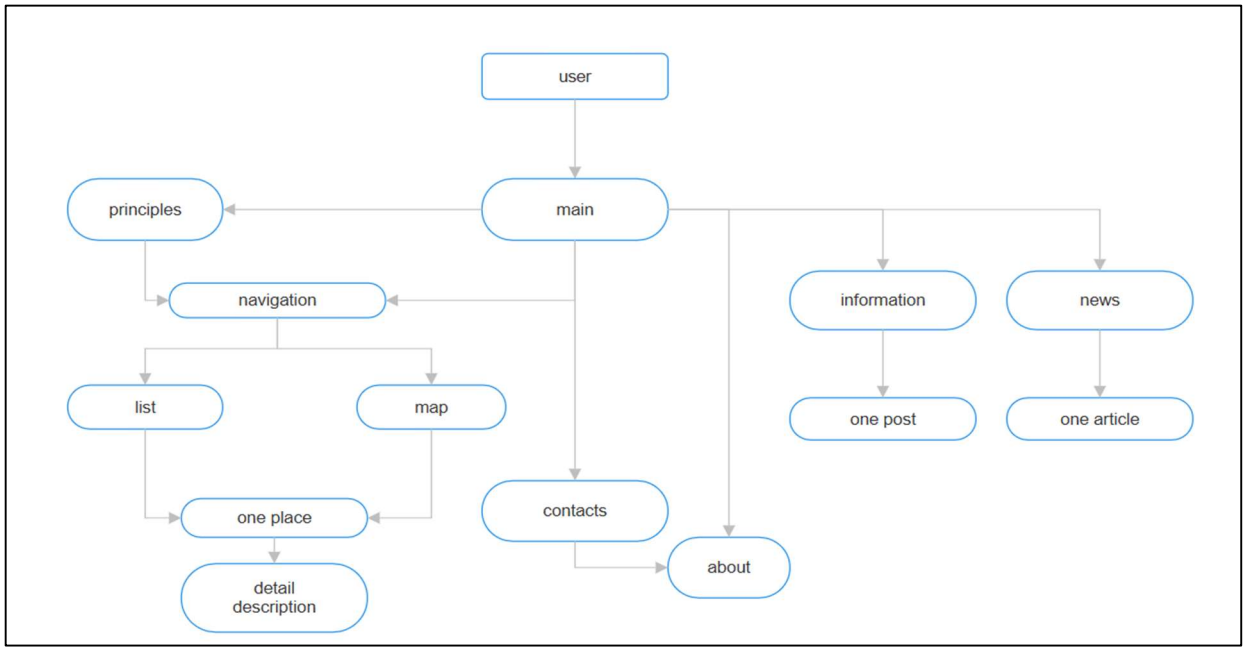

Рисунок 3 — Концептуальная диаграмма компоновки страницами

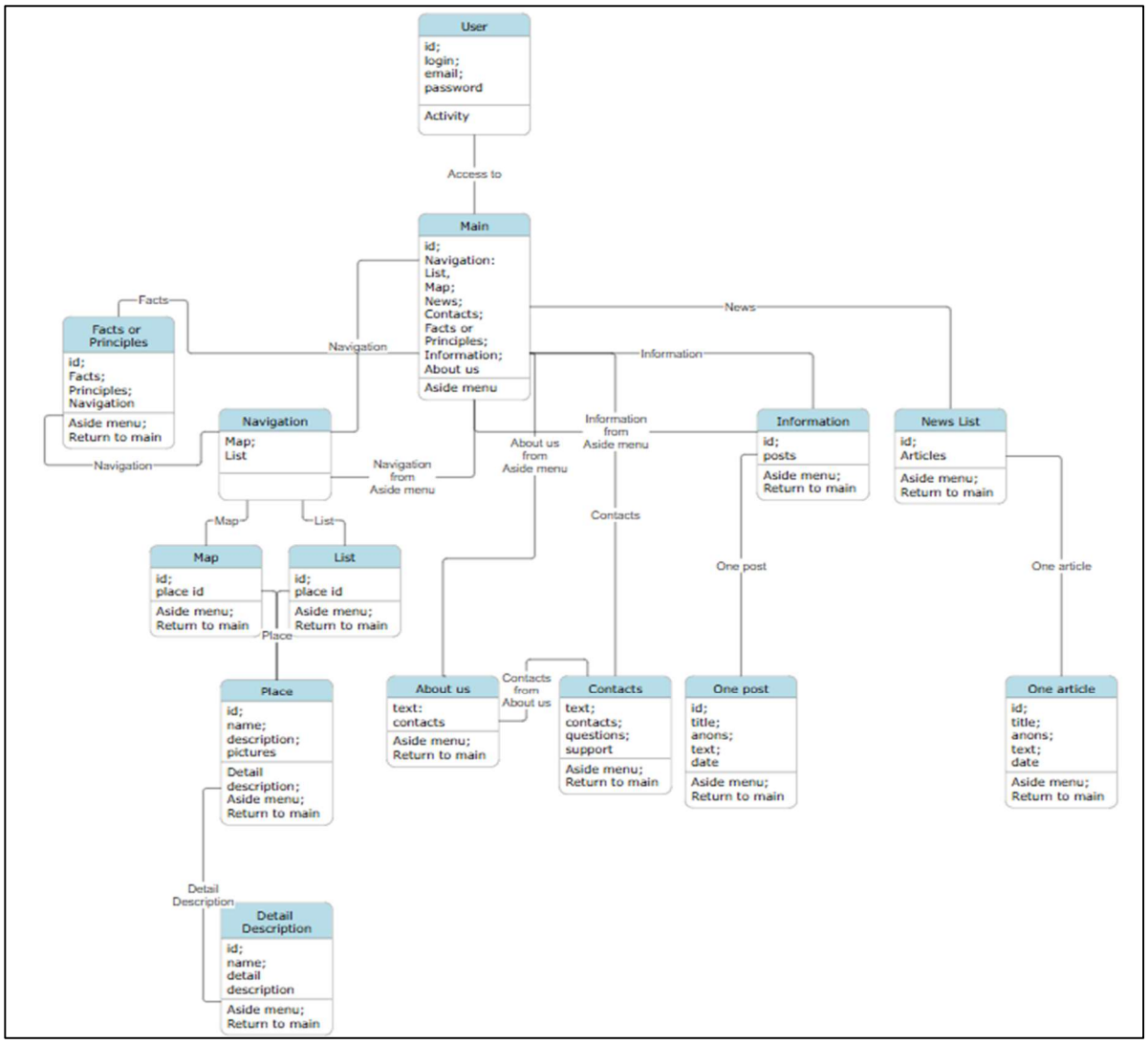

Рисунок 4 — Подробная схема с учетом типа содержания страницы или информации вывода из базы данных.

Упрощенная версия диаграммы показывает все возможные перемещения пользователя по страницам и их связь между собой, а подробная учитывает тип информации или содержание, а также информацию вывода из базы данных.

1.3 Выбор графического редактора Figma

Этот графический редактор был выбран в силу следующих основных характеристик:

- Онлайн-доступ — возможность легко получить доступ к программе и использовать бесплатную версию с широким функционалом.

- Совместная работа — позволяет объединяться в команды и

работать над общим проектом, делиться наработками и комментировать работы других людей.

- Интеграции — объединение разнородных частей и систем в единую среду на базе веб; возможность подключить инструмент для дизайна к другим инструментам и сервисам. Это позволяет упростить процесс проектирования.

- Библиотеки компонентов — это хранилище всех готовых элементов интерфейса, которые можно использовать в разных проектах. Использование библиотек компонентов позволяет: ускорить процесс разработки — не нужно создавать каждый элемент заново, а использовать уже готовый компонент; обеспечить единообразие дизайна — все элементы, созданные на основе одного компонента, будут выглядеть одинаково; упростить процесс редактирования — изменения в одном месте обновляют все экземпляры компонента; повысить эффективность работы — сотрудничество между дизайнерами и разработчиками становится более слаженным.

1.4 Создание интерфейса

Так как тема проекта связана с экотуризмом, которая ассоциируется с природой и экологией, то страницы сайта должны сочетать в себе основные цвета зеленых оттенков, гармонирующих с природой, и картинки, изображающие пейзаж местности, о которой идет речь. Это можно увидеть на примере главной страницы, которая изображена на рисунке 5.

Рисунок 5 — Главная страница

Более того, для простого и понятного использования сайта важно создать интерфейс, удовлетворяющий требованиям UX-дизайна (User Experience — «пользовательский опыт») и UI-дизайна (User Interface — «пользовательский интерфейс»).

UX-дизайн отвечает за то, как посетитель сайта или приложения взаимодействует с интерфейсом и какие эмоции получает в ходе этого взаимодействия. Его главная цель — помогать пользователям максимально быстро и комфортно решать задачи.

UI-дизайн — это создание визуальной части приложения или сайта: экранов, кнопок, иконок. UI-дизайнер придумывает, как будут выглядеть иконки и кнопки, какие цвета и шрифты будут использованы в приложении. Цель — создать привлекательный и понятный визуал. Для этого он разрабатывает визуальный стиль — выбирает цветовую палитру и шрифты, создаёт графические элементы интерфейса — иконки, иллюстрации, дополнительную визуальную графику.

На рисунках 6 и 7 можно заметить элементы UX и UI дизайна: например, требованиям UX соответствует точное расположение объектов на странице, то есть по единому шаблону каждая страница имеет порядок или структуру, например, текст расположен в нужном поле слева, а картинки справа, боковая

панель сохраняет свое положение на любой странице для лучшего ориентирования пользователя на сайте. Требованиям UI соответствует наличие зеленых оттенков в интерфейсе, ассоциирующиеся с экологией, природой, а также все картинки являются пейзажами той местности, о которой идет речь.

Рисунок 6 — Пример страницы с информацией о месте

Рисунок 7 — Карта

UI тесно связан с UX-дизайном. UX отвечает за функциональность и архитектуру интерфейса, а UI — за его внешний вид и визуальную эстетику.

2. Создание программного обеспечения и верстки

2.1 Среда разработки Visual Studio Code с использованием фреймворка Django

Эти инструменты разработки были выбраны благодаря таким особенностям:

- Разнообразие языков программирования — позволяет программировать на разных языках, например, Python, Paskal, Java, JavaScript, C++, C Sharp, HTML, CSS.

- Управление версиями — для управления версиями в Visual Studio Code (VS Code) можно использовать панель Source Control. Она находится в нижней части редактора и активируется с помощью иконки Git (выглядит как ветка). С ее помощью можно просматривать и отслеживать изменения в коде, сравнивать версии файлов, работать с ветками.

- Расширяемость — позволяет добавлять языки, отладчики и инструменты для поддержки рабочего процесса разработки.

- Интеллектуальная помощь — включает несколько инструментов для интеллектуальной помощи в Visual Studio Code: GitHub Copilot, Blackbox, ChatGPT, Tabnine, Mintlify Doc Writer, Kodezi.

2.2 Фреймворк Django

Фреймворк (с англ. framework — «каркас») — заготовка, готовая модель в программировании для быстрой разработки, на основе которой можно дописать собственный код.

Django — свободный фреймворк для веб-приложений на языке Python. С его помощью можно быстрее и проще реализовывать на Python сайты и приложения, которые работают в браузере.

2.3 Верстка сайта и принципы строения каждой страницы

Верстка — это описание визуальной части сайта с помощью

гипертекстового документа на основе HTML-разметки. Простыми словами, это соединение и расположение на странице документа разных элементов веб-сайта: текстовых блоков, изображений, таблиц, видео и т.д.

HTML (HyperText Markup Language) — это язык гипертекстовой разметки, который используется для создания и структурирования веб-страниц.

В создании любой страницы используются тэги (<…> — знаки, которые обозначают границы блока или объекта), внутри которых прописываются тип объекта, стили и другие атрибуты.

Ниже представлен шаблон страницы с комментариями к каждому элементу. Код представлен на листинге 1.

Листинг 1 — Шаблон страницы

```
{% load static%} # загрузка стилей #
<!DOCTIPE html> # вводная, обязательная строка #
<html lang = "ru"> # открывающий главный тэг «html» и настройка языка #
<head> # открывающий тэг «head» #
   <meta charset="utf-8"> # настройка кодировки #
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-eqiv="X-UA--Compatible" content ="IE-edge"> # исп. браузер #
   <title> {%block title%} {% endblock %} </title>
   # название вкладки в тэгах «title» и «/title» #
   <link rel = 'stylesheet' href = "[]">
   <link rel="stylesheet" href="[]" integrity="[]" crossorigin="anonymous">
   {%block css%} {% endblock %}
   <style>
      {%block style%}
      {% endblock %}
   </style> # подключение ссылок стилей Bootstrap и статичных файлов #
</head> # закрывающий тэг «/head»
<body> # открывающий тэг «body» #
```

Продолжение листинга 1

```
   {%block body%} {% endblock %} # главное содержание страницы #
<script src="[]" integrity="[]" crossorigin="anonymous"></script>
   <script src="[]" integrity="[]" crossorigin="anonymous"></script>
  # настройка и подключение скриптов #
</body> # закрывающий тег «/body» #
</html> # закрывающий главный тэг «/html» #
# в данном случае также использовался шаблонизатор Jinja # #
```

Примечание: вместо знаков [] расположены предполагаемые ссылки или ключи для использования нужных ресурсов или функций CSS или JavaScript.

Страница содержит разделы, которые характерны для любой другой, из-за чего выгоднее использовать шаблонизаторы. В случае с Django можно использовать шаблонизатор Jinja и его синтаксис {%...%}. В таком случае можно создать какой-либо шаблон и использовать его для внесения различной информации. Те разделы страницы, которые нуждаются в изменении заключаются в {%%}.

3. Работа с базами данных

База данных — набор информации, которая хранится упорядоченно в электронном виде; набор данных, который как-то структурирован. Базу данных нельзя назвать программой в полном смысле этого слова.

По умолчанию Django в качестве базы данных использует SQLite. Она проста в использовании и не требует запущенного сервера. Все файлы базы данных могут легко переноситься с одного компьютера на другой.

Для создания базы данных в Django нужно:

- Создать аккаунт суперпользователя (админа) командой «createsuperuser» для доступа к админ панели.
- Написать модели, определяющие нужную структуру БД.
- Подготовить и выполнить миграции — «makemigrations» и

«migrate»; Миграция — это механизм, который позволяет управлять изменениями в структуре базы данных приложения, она обеспечивают удобный способ создания, изменения и удаления таблиц и полей в базе данных, синхронизируя их с моделями Django.

Работу создания записей в базу данных и их вывод на страницу описан поэтапно ниже, на примере новостей.

Листинг 2 — Создание модели

```
from django.db import models
# импортируем функцию создавания модели в базе данных #
class Articles1(models.Model): # вводим класс на основе модели #
    title = models.CharField('Название', max_length=50)
    anons = models.CharField('Анонс', max_length=250)
    text = models.TextField('Статья', max_length=8000)
    date = models.DateTimeField('Дата публикации')
# создаем поля ввода с своими характеристиками #
    def __str__ (self):
        return self.title # вводим функцию вывода объектов по названию#
class Meta: # даем имя модели в базе данных #
        verbose_name = 'Новость1'
        verbose_name_plural = 'Новости1'
```

Листинг 3 — Регистрация модели

```
from .models import Articles1
# импортируем созданную модель из models.py #
admin.site.register(Articles1)
# регистрируем модель в приложении #
```

После всех шагов можно редактировать данную модель, записывать

новую информацию или удалять старую в админ панели фреймворка Django.

Чтобы обозначить использование модели на странице или вывод ее объектов, записываем в views.py вывод объектов, но при обращении по ключу.

Для вывода записей из базы данных нужно использовать цикл «for» с использованием шаблонизатора Jinja.

Это показано ниже на листингах 4 и 5.

Листинг 4 — Запись модели в views.py

```
from .models import Articles1 # импортируем модель #


def news1 (request):
    news = Articles1.objects.all
# переменная news содержит все объекты модели #
    return render (request, 'news/news.html', {'news': news})
# при обращении по ключу 'news' выводим переменную news #
```

Листинг 5 — Вывод записей из базы данных

```
{% for el in news%} # обращаемся по ключу 'news' #
<h3> {{ el.title }} </h3>
<p> {{ el.anons }} </p>
<p> {{ el.text }} </p>
<p> {{ el.date }} </p> # выводим каждый элемент  #
{% endfor %} # завершаем цикл #
```

По такому плану реализуется работа с базами данных от лица программиста или админа, но существует способы, при которых базы данных обновляются не только через админ панель, но и через заполнение «форм» пользователем.

Рассмотрим создание и заполнение «формы», на примере оставления

отзывов. Создадим модель по прежнему алгоритму. Это показано на листинге 6.

Листинг 6 — Создание модели для формы

```
from django.db import models
# импортируем функцию создавания модели в базе данных #
class Que(models.Model): # вводим класс на основе модели #
text = models.TextField('Отзыв', max_length=300)
# создаем поля ввода с своими характеристиками #
    def __str__ (self):
        return self.text # вводим функцию вывода объектов по тексту#


    class Meta: # даем имя модели в базе данных #
        verbose_name = 'Отзыв'
        verbose_name_plural = 'Отзывы'
```

Регистрация модели происходит по тому же принципу, ознакомится с кодом можно по ссылке https://disk.yandex.ru/d/13YiL1T8Ny8thw

Создаем файл forms.py и форму в ней как показано на листингах 7 и 8.

Листинг 7 — Создание типов полей для формы

```
from .models import Que # импортируем модель #
from django.forms import ModelForm
# используем функцию создания формы #
from django.forms import TextInput
# используем тип вводимых данных «текст» #
```

Структура формы показана на листинге 8.

Листинг 8 — Создание структуры для формы

```
class QueForm(ModelForm): # создаем класс на основе формы #
    class Meta:
        model = Que # наследуем модель #
        fields = ['text'] # записываем поля для ввода #
    widgets = {
            "text": TextInput(attrs={ # назначаем атрибуты #
                'class': 'form-control', # наследуем класс для стилей #
                'placeholder': 'Отзыв'
            } ),
        } # обозначаем характеристики для каждого поля #
```

Прописываем функцию вывода формы в views.py как показано на листинге 9.

Листинг 9 — Запись формы в views.py

```
from .models import Que # импортируем модель #
def otz(request):
    if request.method == 'POST': # если поступает заполнение #
        form = QueForm(request.POST)
        if form.is_valid(): # проверка правильности #
            form.save() # сохранение #
            return HttpResponseRedirect(reverse('questions')) # перенаправл. #
    form = QueForm()
    data = {'form': form, 'error': error} # установка ключей #
    return render (request, "main/otz.html", data)
```

Обязательно нужно использовать {% csrf_token %} для верной работы формы, и на любой странице записанный {{form.text}} будет обозначать поле для ввода. С использованием этих алгоритмов сайт поддерживает возможность регистрироваться, оставлять отзывы, а также редактировать, добавлять или удалять новости.

4. Тестирование

Таблица 1. Функциональное тестирование

№	Назначение теста	Значения исходных данных	Ожидаемый результат	Реакция программы	Вывод
1	Проверка работы передвижения пользователя по страницам с использованием ссылок; пример: использование карты и перемещение в любую точку на ней	Нажатие на текст в виде ссылки; пример: нажав на указанное место, пользователь перейдет на страницу описания локации	Перемещение на нужную страницу сайта; пример: откроется страница с информацией о выбранном месте	Открытие ожидаемой страницы; пример: пользователь перешел на выбранную страницу	Работа верная
2	Проверка вывода (создания и добавления) новостей из базы данных	Создание, заполнение модели и выведение из базы данных	Действия с моделью будут работать исправно	Модель отображает нужную информацию на сайте	Работа верная

3	Правильная работа заполнения формы, внешнее добавление записей в базу данных; пример: оставление отзыва	Создание, заполнение модели и выведение из базы данных. Создание формы для заполнения.	Действия с моделью будут работать исправно. При заполнении формы запись отправится в базу данных и затем отобразится на другой странице сайта	Форма заполнилась верно и отправилась в базу данных; Созданная модель отображает форму на сайте и отзыв человека	Работа верная

Таблица 2. Оценочное тестирование

№ Пользователя	Удобство использования	Визуальный стиль	Эксплуатация
«1»	8	9	7
«2»	8	9	7
«3»	7	8	6
«4»	8	8	7
«5»	9	7	8
«6»	8	9	7
«7»	9	9	8
«8»	7	10	9
«9»	9	10	8
«10»	8	10	7

Средняя оценка:	8,1	8,9	7,4

Итоговая оценка по всем баллам: 8,13

Комментарии и отзывы: «Интересный, познавательный сайт»; «Выглядит красиво и лаконично, вызывает хороший эффект»; «Только о половине всех мест что-то знаю, найду для себя что-нибудь новое».

ЗАКЛЮЧЕНИЕ

В результате работы был создан сайт, поддерживающий функции перемещения между страницами, возможностью оставить отзыв или подробно ознакомится с достопримечательностями нашей страны и быть в курсе последних новостей связанных с сферой экотуризма. Итоговый функционал доступный для пользователя: ознакомление с информацией о достопримечательностях России, их описанием с визуальными элементами; чтение новостей о развитии сферы экотуризма; возможность оставлять отзывы на сайте.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Силин, П. А. "Проектирование и разработка веб-приложений." М.: Издательство, 2020.
2. Баранов, С. В. "Основы работы с Django." М.: Издательство, 2021.
3. Степанов, И. А. "Методы тестирования программного обеспечения." М.: Издательство, 2017.
4. Шабанов, Д. Ю. "Дизайн пользовательского интерфейса." М.: Издательство, 2020.
5. Нейросеть GigaChat.