

**Содержание**

[Введение 7](#_Toc74772403)

[1 Нормативные ссылки 9](#_Toc74772404)

[2. Термины, определения и сокращения 10](#_Toc74772405)

[3. Исследование предметной области 11](#_Toc74772406)

[3.1 Анализ функциональной структуры проекта 11](#_Toc74772407)

[3.2 Анализ востребованности проекта 12](#_Toc74772408)

[4. Спецификация требований к приложению 15](#_Toc74772409)

[4.1 Постановка задачи 15](#_Toc74772410)

[4.2 Требования к программному обеспечению 15](#_Toc74772411)

[5. Проектирование приложения 17](#_Toc74772412)

[5.1 Пользователи 17](#_Toc74772413)

[5.2 Функции приложения 18](#_Toc74772414)

[5.3 Расположение страниц и переход между ними 19](#_Toc74772415)

[5.4 Структура взаимодействия клиент-сервер 20](#_Toc74772416)

[6. Пример реализации информационной системы – сборника студенческих работ 21](#_Toc74772417)

[6.1 Выбор среды разработки и языка программирования 21](#_Toc74772418)

[6.2 Модель MTV в Django 22](#_Toc74772419)

[6.3 Описание разработанного приложения 34](#_Toc74772420)

[7. Безопасность и экологичность проекта 39](#_Toc74772421)

[7.1 Значение и задачи безопасности жизнедеятельности 39](#_Toc74772422)

[7.2 Анализ условий труда и мероприятия по защите от воздействия вредных производственных факторов 40](#_Toc74772423)

[7.3 Обеспечение электробезопасности 43](#_Toc74772424)

[7.4 Пожарная безопасность 43](#_Toc74772425)

[Заключение 45](#_Toc74772426)

[Список использованных источников 46](#_Toc74772427)

[Приложение А](#_Toc74772428) – [Листинг 47](#_Toc74772429)

# Введение

В качестве задания на выпускную квалификационную работу была поставлена задача - проанализировать актуальность разработки сайта – сборника курсовых работ, рефератов, дипломных. Также перед самой реализацией проекта необходимо исследовать предметную область и составить техническое задание, чтобы отчётливо представлять, как спроектировать приложение, а главное – для чего, будет ли оно востребованным потребителем, нужно ли оно вообще.

С началом ХХI века интернет стал доступен широкому кругу пользователей в России. Это значительно упростило доступ к информации. Теперь, чтобы, например, узнать новости, человеку не обязательно ждать выпуска свежей газеты или нового новостного блока по телевизору. Достаточно просто открыть сайт. Также, не нужно идти в библиотеку и копаться в куче книг и журналов, чтобы получить какую-либо информацию. You’re a click away! Кто-то скажет, что в этом есть свои минусы, люди стали ленивее, но ведь есть поговорка: «Лень – двигатель прогресса». В общем и целом – будущее за автоматизацией процессов. В том числе и за автоматизацией получения информации, её сортировки. С этой точки зрения создание склада готовых работ ­– очень полезная и актуальная идея. Для чего перебирать горы литературы, если ты можешь получить и проанализировать уже отсортированную, сконцентрированную информацию? Прочитать и понять, как человек уже до тебя реализовал ту или иную идею, взять его разработки в пример. Тем более, что сам автор это не запрещает, так как сам предоставил эту информацию в общий доступ. Так мы дошли до основной идеи реализации проекта – по сути файлообменника. Каждый зарегистрированный пользователь может загружать свои работы, тем самым расширяя базу сайта. И чем больше таких людей, тем больше работ, тем больше вероятность найти что-то для себя.

Также можно услышать уместный аргумент: «Такие проекты уже есть и не один». Есть, но, во-первых, этот факт породит конкуренцию, которая, в свою очередь станет стимулом для дальнейшего совершенствования и развития проекта. Во-вторых, каждый проект по своей сути уникален и не может в точности быть похожим на какой-либо другой (если он, конечно, просто не скопирован). Начав свой путь с простого приложения – обменника материалами, проект может перерасти во что-то ещё, обзавестись новыми возможностями и функциями. Такими, например, как общение пользователей, а это уже во многом повысит его популярность.

Вместе со всем вышесказанным приложение будет подкреплено чатом в очень популярном мессенджере – Telegram. Почему именно Telegram. Так как на наш взгляд это набирающий популярность мессенджер, с очень широким функционалом, в котором можно реализовать практически любую задачу в рамках нашего проекта. Это в перспективе новостной канал и чат-бот.

Что касается способа раскрутки: на начальном этапе планируется разослать ссылку на сайт в группы и паблики ВУЗа и знакомых студенческих групп, с просьбой помочь наполнить проект контентом. В качестве стимула к этому будут подобраны несколько конкурсов с призами самым активным пользователям.

# 1 Нормативные ссылки

В выпускной квалификационной работе использованы ссылки на следующие нормативные документы:

1. ГОСТ Р 1.5-2004 Стандартизация в Российской Федерации. Стандарты национальные Российской Федерации. Правила построения, изложения, оформления и обозначения.
2. ГОСТ Р 1.12-2004 Стандартизация в Российской Федерации. Термины и определения.
3. ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления.
4. ГОСТ Р ИСО 9000-2008 Системы менеджмента качества. Основные положения и словарь.
5. ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы жизненного цикла программных средств.
6. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных продуктов.
7. ГОСТ 19.105-78 ЕСПД. Обозначение программ и программных продуктов.
8. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.
9. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.

1. **Термины, определения и сокращения**

В описании выпускной квалификационной работы применялись термины в соответствие с определениями и сокращениями, установленными нормативным документом ГОСТ Р 1.12-2004. Стандартизация в Российской Федерации. Термины и определения:

**1 Web-сайт –** обычно, массив связанных между собой html-страниц, воспринимаемых пользователем как единое целое.

**2 Информационная система (ИС) –** система, которая служит для поиска, хранения и обработки информации. Предназначена для своевременного доступа конкретных людей к конкретной информации.

**3 База данных –** массив данных, хранящихся в соответствие с определённой схемой данных, управление которыми выполняется в соответствие с правилами средств моделирования данных.

**4 Python –** высокоуровневый язык программирования, имеющий строгую динамическую типизацию и применяемый в различных областях программирования.

**5 ER диаграмма –** диаграмма, отражающая «сущность-связь». Отображает связи между объектами и их атрибутами в интуитивно-понятном виде.

**6 Фреймворк –** некий набор модулей, созданный для определённого языка программирования, облегчающий программисту разработки проекта.

**7 ИС** – информационная система.

**8 MTV – “**Models-Templates-Views” (Модели, шаблоны, представления).

**9 СУБД –** система управления базой данных.

**10 DRY – “**Don’t repeat yourself” (Не повторяйся).

**11 ОС –** операционные системы.

1. **Исследование предметной области**
   1. **Анализ функциональной структуры проекта**

Итак, главной функцией приложения является хранение сконцентрированных данных в виде готовых работ – рефератов, контрольных, курсовых проектов. При этом официальное оформление работ согласно ГОСТам или ещё каким-либо стандартам не имеет особого значения, так как данная информация преподносится окончательному пользователю именно для изучения, ознакомления, не для «копирования». Главная задача проекта – облегчить поиск информации для человека, студента, создать площадку для общения, «соединить» людей.

Функциональная структура сайта довольно проста. Необходимо реализовать возможность загружать работы на сервер со стороны пользователя, создать сортировку работ по категориям и, конечно же, нужен поиск работ по ключевым словам. Эта последняя функция имеет ключевое значение, так как с накоплением материала найти что-то без поиска станет просто нереальной задачей. Ну и, естественно, нужна возможность загрузки файла на компьютер пользователя, иначе, проект потерял бы всякий смысл.

Что касается пользователей, то все они делятся на два вида: авторизованные и неавторизованные. Неавторизованным разрешено пользоваться всем функционалом сайта, кроме загрузки работ на сервер, что разрешено авторизованным. Также на сайте предусмотрена регистрация, но она необязательна.

Как было уже сказано выше, в рамках проекта была поставлена задача: дать людям площадку для коммуникаций. Чтобы пользователи могли спросить совет, задать вопрос, найти решение своей проблемы. Для этого на начальном этапе проекта было принято решение использовать набирающий популярность в кругах не только молодёжи, но и взрослого поколения мессенджер «Telegram». Почему именно Telegram? Да потому что это приложение обладает просто огромнейшим функционалом. Это и общие/приватные чаты, каналы с возможностью делать голосования и не только, а также чат-боты, в которых можно реализовать практически что угодно, в частности прикрепить базу сайта и реализовать поиск работ именно через чат-бот, что и планируется в дальнейшем.

* 1. **Анализ востребованности проекта**

Существует несколько способов поиска работ в интернете:

­– поиск на бесплатных сервисах с открытой библиотекой (будь то различные сайты, группы VK и пр.);

– поиск в закрытых платных базах;

– поиск по работам в базах ВУЗов.

В данной работе мы обходим стороной разбор всех платных ресурсов, так как целью нашего проекта является разработка открытой свободной библиотеки.

С некоторых пор для ВУЗов стало обязательным условием хранить электронные архивы всех выполненных студентами работ, будь то курсовые, либо дипломные работы. Но очень небольшое кол-во заведений хранят их на сайте в открытом доступе. Среди этих немногих можно отметить кафедру Системного программирования Математико-механического факультета СПбГУ [1] или электронную базу Московского государственного психолого-педагогического университета [2]. В общем и целом это очень неплохая практика хранить работы учащихся ВУЗов в открытом доступе, ведь в таком случае можно взять работу в пример не опасаясь, что данные будут не верны, ведь источником работ является само учебное заведение.

Теперь перейдём к свободным сервисам с открытыми библиотеками. Разберём на примере несколько ресурсов:

– [bestreferat.ru](http://bestreferat.ru);

– [bibliofond.ru](http://bibliofond.ru).

Bestreferat.ru – пожалуй наибольший в рунете банк рефератов (около 364 тыс. работ). Приветственная страница показана на рисунке 3.1. На ней мы видим яркий отлично свёрстанный контент. На странице есть место для рекламы (реклама, конечно, необходима, чтобы покрыть расходы на администрирование). Также на главной странице есть форма для поиска, что очень удобно. Из минусов данного сайта стоит отметить отсутствие поиска по категориям (рефераты, курсовые, дипломные работы). Также проштудировав несколько скачанных работ было замечено отсутствие в работах конкретных вычислений, формул, отсутствие картинок. То есть в работах опускается практическая часть, что является неотъемлемой составляющей любого проекта. Таким образом, можно сделать вывод, что данный банк работ подходит по большей части пользователям, которые ищут материал гуманитарной направленности, хотя и в таком случае в большинстве таких работ должны содержаться различные таблицы, схемы и т.д.



Рисунок 3.1 – Приветственная страница Bestreferat.ru

Bibliofond.ru – ещё один немалоизвестный портал с широчайшим выбором готовых работ. На главной странице нас встречает простая и удобная форма поиска (рисунок 3.2).

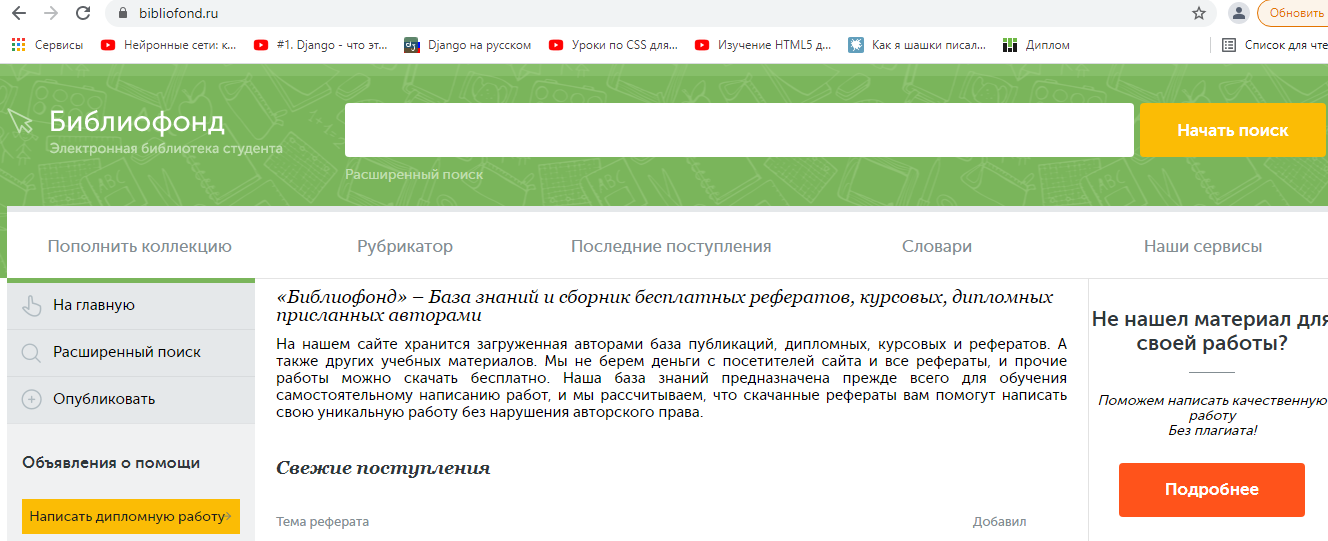


Рисунок 3.2 – Главная страница сайта bibliofond.ru

Тут, в отличие от первого объекта анализа, более интуитивно-понятный интерфейс, присутствует поиск по категориям, а также по направлениям (математическое, иностранные языки и т.д.). Помимо работ, база также включает сборники различной литературы, словарей, книг и проч. Из недостатков можно отметить отсутствие коммуникации, возможности обсудить ту или иную тему, есть только возможность заказать работу (что не есть лучший вариант для обучающегося).

Проанализировав подобные проекты, были сделаны соответствующие выводы о том, как сделать свой собственный ресурс полезным, востребованным и имеющим право на существование и дальнейшее развитие.

**4. Спецификация требований к приложению**

**4.1 Постановка задачи**

В основу создаваемого приложения ляжет интуитивно-понятный интерфейс. Не должно быть ничего лишнего. Также на начальном этапе не планируется никакой рекламы. На главной странице сайта будет располагаться поисковая форма. Также весь контент будет поделён на категории, с добавлениями новых категорий этот список будет расширяться. Все поступающие работы будут проверяться на соответствие, и после этого поступать в публикацию. Разрешены все текстовые форматы. Также должна быть предусмотрена активная обратная связь, как через электронную почту, так и в чате, который будет посвящён данному ресурсу. Загрузить работу сможет только зарегистрированный пользователь, а скачать – любой желающий.

## 4.2 Требования к программному обеспечению

Для выполнения поставленной задачи необходим ПК с предустановленной ОС Windows 8.1 и выше. Также предполагается наличие установленного интерпретатора Python версии не ниже v3 и фреймворка Django.

Все используемые программой ресурсы при их задействовании должны быть отверифицированы (критичные непосредственно при старте). Например: наличие свободной оперативной памяти, наличие достаточного свободного дискового пространства, наличие исполняемых модулей и компонент самой программы, а также допустимость их версий, наличие переменных окружения, существование и доступность для открытия используемых программой каталогов и файлов данных, допустимость и непротиворечивость установок вообще, так и установок, зависящих от пользователя, и т.д. В случае обнаружения несоответствия условиям нормальной работы, должно быть выдано осмысленное и развернутое сообщение. Обязательным условием является ведение журнала протоколирования работы программы (как для ошибок, так и для обычной работы).

# 5. Проектирование приложения

Разработку любого продукта нужно начинать с его проектирования. Здесь нам на помощь приходит язык UML – система обозначений, применяемая для объектно-ориентированного анализа и проектирования. В общем и целом при планировании продукта необходимо разобраться в следующих аспектах:

– пользователи и их роли;

– функции приложения;

– расположение страниц и переход между ними.

­

## 5.1 Пользователи

Все пользователи сайта делятся на гостей и зарегистрированных пользователей. Зарегистрированные пользователи в свою очередь делятся на простых пользователей и администраторов (суперЮзер). Диаграмма пользователей показана на рисунке 5.1.

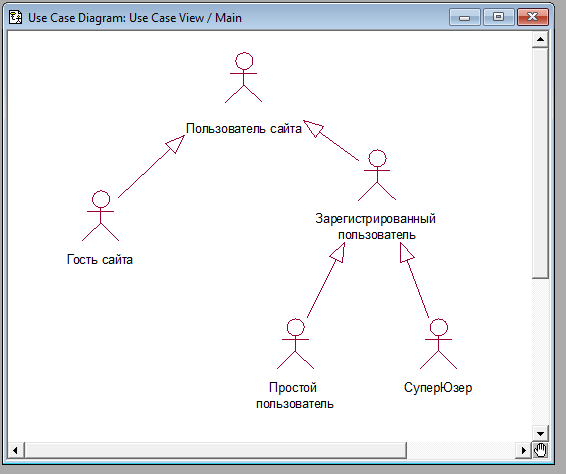


Рисунок 5.1 – Диаграмма пользователей

В зависимости от класса пользователям доступны различные роли. Чем специфичнее класс пользователя, тем больше в совокупности у него возможностей. Простой посетитель (гость) может пользоваться всеми доступными ресурсами сайта: скачивать готовые работы, переходить в новостную ленту. Зарегистрированный простой пользователь обладает всеми теми же правами, но с доступом к загрузке на сайт своих работ. Администратор сайта, может корректировать (разрешать/запрещать) публикацию работ, добавлять новые категории.

## 5.2 Функции приложения

Итак, переходим к функционалу сайта. Основные функции отображены в диаграмме на рисунке 5.2.

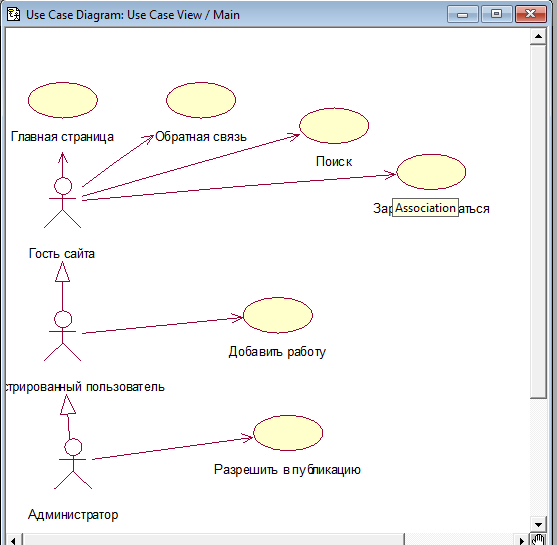


Рисунок 5.2 – Диаграмма функционала.

Гостю сайта доступны такие функции как переход на главную страницу, поиск, доступ к обратной связи, регистрация. Зарегистрированному пользователю кроме всего прочего доступна функция авторизации, добавления работ на портал. А администратору помимо всех вышеупомянутых доступна возможность снимать и ставить в публикацию работы других пользователей.

## 5.3 Расположение страниц и переход между ними

Каждому отображению web-страницы соответствует свой класс представления и html-документ. В таблице 5.1 показана схема каждой из них.

Таблица 5.1 – Структура сайта

|  |  |  |
| --- | --- | --- |
| Класс-представление | Html-документ | Отображаемая страница |
| ReflistHome | index.html | Главная страница |
| SearchList | search\_results.html | Поиск |
| AddWork | addwork.html | Добавление работы |
| ContactFormView | contact.html | Обратная связь |
| ReflistCategory | index.html | Категории |
| RegisterUser | register.html | Регистрация |
| LoginUser | login.html | Авторизация |

Структура сайта должна быть организована таким образом, что с каждого представления можно перейти по любому маршруту.

## 5.4 Структура взаимодействия клиент-сервер

Для разработки проекта было принято решение использовать фреймворк Django. Схема работы данного фреймворка отображена в диаграмме на рисунке 5.3

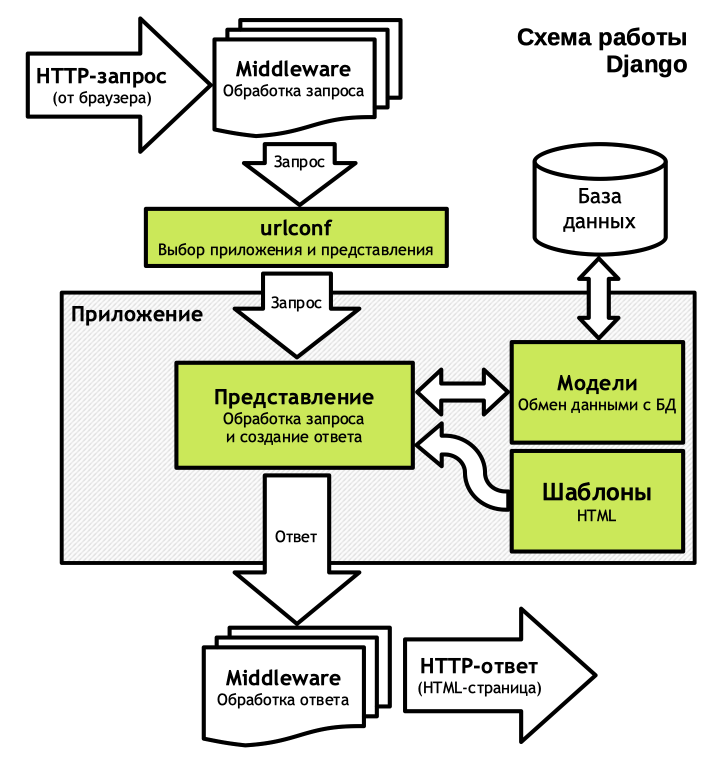


Рисунок 5.3 – диаграмма взаимодействия клиент-сервер

Django предоставляет удобный инструмент для разработки web-приложений, а также обладает удобной встроенной ORM для работы с базой данных.

1. **Пример реализации информационной системы – сборника студенческих работ**
   1. **Выбор среды разработки и языка программирования**

Для разработки поставленной задачи было принято решение выбрать язык программирования Python. Сегодня python стал очень популярным языком программирования из-за простоты реализации задач любого уровня. Этот язык подходит буквально для любого проекта, будь то реализация нейронных сетей, их обучение, машинное программирование, deep learning, а также разработка web-сайтов. Основное отличие языка python от остальных (JAVA, C#, PHP) – предоставление очень удобных конструкций для написания наглядного и удобного кода программ. Ещё один огромный плюс языка – огромное количество написанных библиотек для решения общих задач. В частности, для разработки Web-приложений создан фреймворк Django.

В общем и целом фреймворк Django разработан для обеспечения работы ресурса на стороне сервера – это инструмент backend-разработчика. Весь процесс взаимодействия пользователя с сайтом показан на рисунке 6.1.

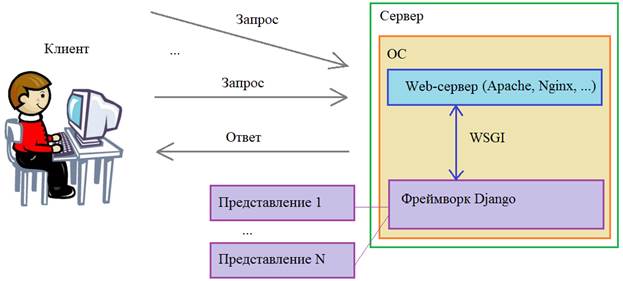


Рисунок 6.1 – Процесс взаимодействия пользователя с сайтом.

В частности фреймворк Django – это обёртка языка Python, при помощи которого сервер по интерфейсу WSGI передаёт этому фреймворку обработку запроса и на выходе используется определённое представление, отвечающее за текущий запрос, результатом которого является HTML –страница которая вначале передаётся Web-серверу по тому же WSGI-интерфейсу, а затем уже пользователю.

WSGI – это стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим Web-сервером.

* 1. **Модель MTV в Django**

Давайте подробнее разберемся как в Django работает механизм обработки запросов от пользователя. Нам это понимание пригодится в дальнейшем, описывая отдельные элементы всей этой процедуры.

Смотрите, вначале, когда запрос приходит на фреймворк, он пропускается через блок маршрутизации (рисунок 6.2).

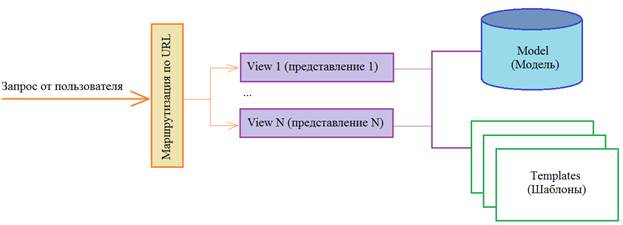


Рисунок 6.2 – Процесс взаимодействия приложения с пользователем

Здесь фиксируется тип URL-адреса и в списке шаблонов предопределенных адресов ищется первое совпадение. Например, пользователь вводит запрос:

http://127.0.0.1:8000/

Это есть не что иное, как главная страница сайта, фактически пустой запрос без параметров. Следующий запрос:

http://127.0.0.1:8000/category/1/

соответствует шаблону:

category/<число>

Это уже другой маршрут. Или такой запрос:

http://127.0.0.1:8000/refs/referati/

с шаблоном:

refs/<слаг (строка)>

И так далее. У нас может быть прописано множество типов URL-адресов и каждый адрес связан со своим обработчиком – представлением (иногда его еще называют контроллером). Если текущий запрос от пользователя не совпал ни с одним предопределенным URL, то возвращается код ошибки 404 – страница не найдена.

Предположим, что маршрутизатор нашел совпадение. Далее, активизируется представление, связанное с найденным типом URL-адреса. Представление (иногда его еще называют контроллером) – это или функция или класс, который отвечает за формирование ответа на соответствующий запрос. Как правило, ответом является HTML-страница. Эта страница, затем, возвращается клиенту, и он ее видит в браузере. Так вот, чтобы контроллер мог сформировать страницу, в общем случае, требуются данные (информация), плюс  шаблоны, в которые эти данные упаковываются. Например, приходит запрос на вывод страницы о рефератах:

http://127.0.0.1:8000/refs/referati/

Активизируется соответствующее представление, которое берет шаблон информационной страницы и наполняет ее данными об этой певице, хранящиеся в БД (рисунок 6.3).

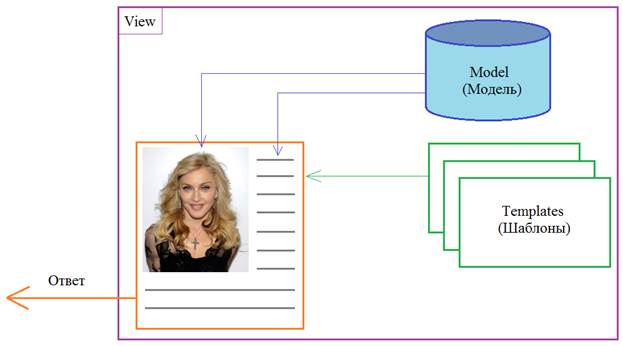


Рисунок 6.3 – Активизация представления

На выходе получаем сформированную HTML-страницу, которая и возвращается пользователю. Вот такое разделение на данные (model), шаблоны (templates) и представления (views) представляет собой общеизвестный паттерн MTV, то есть, разделение данных и HTML-шаблонов. Техника довольно эффективна и удобна, так как позволяет независимо наполнять БД информацией и параллельно разрабатывать или изменять функционал сайта. Кроме того здесь легче находить ошибки, в отличие от подхода, когда в одном скрипте присутствует и подключение к БД и оперирование шаблонами. Методика «разделяй и властвуй» очень хорошо себя зарекомендовала в мире программирования и довольно часто используется в том или ином виде.

Отлично, я думаю общие принципы работы фреймворка Django вам понятны. Но пока наш сайт пустой, он не будет реагировать ни на какие запросы пользователей, кроме тестовой главной страницы. Какой должен быть наш следующий шаг? Согласно философии Django мы должны создать новое приложение в рамках нашего сайта. Что это за приложение и зачем оно вообще нужно? Разработчики фреймворка решили, что каждая самостоятельная часть сайта должна представляться в виде своего отдельного приложения. Например, создавая информационный сайт, мы должны будем определить приложение для отображения страниц этого сайта по определенным запросам. Далее, к нам приходит руководитель проекта и сообщает, что еще нужно реализовать форум на сайте. И, так как это функционально независимая часть сайта, то мы создаем еще одно приложение для форума. Затем, руководитель вспомнил, что еще нужно сделать раздел с опросом пользователей по разным тематикам. И на сайте появляется еще одно приложение – для опроса. И так далее. Каждая логически и функционально независимая часть сайта предполагает его реализацию в виде отдельного приложения (рисунок 5.4).

Приложения в Django следует реализовывать максимально независимыми, в идеале – полностью независимыми, чтобы в дальнейшем мы могли их просто скопировать в другой сайт и там оно сразу же начинало бы работать. Это не всегда удается, но нужно к этому стремиться.



Рисунок 5.4 – Пример отдельных приложений.

Итак, давайте создадим в нашем сайте первое приложение, которое возьмет на себя базовый функционал, то есть, оно и будет являться ядром нашего сайта. Для этого я открою терминал и, находясь в каталоге django/djsite/mysite, выполню команду:

python manage.py startapp refs

Здесь startapp – команда для создания нового приложения;   
women – название приложения. Название может быть любым (мы его придумываем сами), но так, чтобы оно отражало суть своего функционала. В данном случае приложение refs (работы) будет формировать станицы сайта со списком работ.

После выполнения команды у нас в проекте появилась еще одна папка – refs, которая уже содержит несколько файлов и файл \_\_init\_\_.py, следовательно, приложение в Django реализуется как пакет языка Python. Также здесь присутствует одна вложенная папка migrations для хранения миграций БД нашего приложения. Подробнее о ней мы поговорим позже. Остальные файлы имеют следующее назначение:

– admin.py – для настройки админ-панели сайта (админ-панель поставляется совместно с Django и каждый сайт может сразу ее использовать);

– apps.py – для настройки (конфигурирования) текущего приложения;

– models.py – для хранения ORM-моделей для представления данных из базы данных;

– tests.py – модуль с тестирующими процедурами;

– views.py – для хранения представлений (контроллеров) текущего приложения.

После создания приложения его необходимо зарегистрировать в проекте нашего сайта, чтобы фреймворк Django «знал» о его существовании и корректно с ним работал. Для этого нужно перейти в пакет конфигурации сайта (myreflist), открыть файл settings.py и в списке INSTALLED\_APPS прописать новое приложение. В нем уже прописаны несколько стандартных приложений самого фреймворка и к ним мы просто добавим свое (листинг 6.1):

Листинг 6.1 – Добавление приложения

INSTALLED\_APPS = [

...

    'refs',

]

По идее этого вполне достаточно и все будет работать, но в действительности Django обращаясь к этому пакету находит файл apps.py, откуда и берет настройки приложения из класса RefsConfig. Чтобы в дальнейшем каждый раз не конкретизировать этот путь, я пропишу его сразу в списке приложений (листинг 6.2):

Листинг 6.2 – Другой вариант добавления приложения.

INSTALLED\_APPS = [

...

    'refs.apps.RefsConfig',

]

Все, приложение создано и зарегистрировано. Давайте теперь создадим обработчик главной страницы сайта. Для этого нужно определить представление этой страницы. Как я уже отмечал, представления в Django можно реализовывать или в виде функций или в виде классов. Давайте для начала воспользуемся функцией, как наиболее простой реализацией для понимания. Эта функция будет отвечать за формирование ответа для главной страницы приложения refs (листинг 6.3) и называться index (название может быть любым, мы его определяем сами):

Листинг 6.3 – Функция обработчик главной страницы

**def** index(request):

**return** HttpResponse("Страница приложения refs.")

Здесь указывается первый обязательный параметр request – это ссылка на экземпляр класса HttpRequest, который содержит информацию о запросе, о сессии, о куках и так далее. То есть, через переменную request нам доступна вся возможная информация в рамках текущего запроса.

На выходе эта функция должна возвращать экземпляр объекта HttpResponse, который будет автоматически формировать нужный заголовок ответа, а содержимое HTML-страницы будет определяться указанной строкой.

Теперь нам нужно связать эту функцию представления главной страницы с соответствующим URL-адресом. Для этого в пакете конфигурации mysite откроем файл urls.py и в список адресов urlpatterns добавим новый путь с помощью специальной функции path (листинг 6.4):

Листинг 6.4 – Добавление маршрута.

urlpatterns = [

    path('admin/', admin.site.urls),

    path('refs/', index),

]

Здесь первый параметр – это суффикс URL-адреса, то есть, часть URL, которая добавляется после доменного имени (при этом в конце ставится слеш). Например, если наш сайт располагается по адресу

http://127.0.0.1:8000

то первый аргумент ‘refs/' добавляется в конце к этому пути:

http://127.0.0.1:8000/refs/

Именно такой адрес мы сейчас и определяем. Второй аргумент – это ссылка на функцию представления, которая должна возвращать ответ на данный запрос. Как мы уже знаем, ответ формируется в виде экземпляра класса HttpResponse.

Теперь нам нужно импортировать функцию index, чтобы она была доступна в пакете конфигурации (листинг 6.5):

Листинг 6.5 – Импортирование функции index.

**from** refs.views **import** index

Если интегрированная среда вам здесь указывает ошибку, то это потому, что рабочим каталогом следует указать проект mysite.

Проверим работоспособность нашего нового приложения и маршрута. Запустим тестовый веб-сервер:

python manage.py runserver

И откроем страницу:

http://127.0.0.1:8000/refs/

В рамках этого приложения мы можем определять сколько угодно таких функций, связанных с разными URL-адресами. Например, пропишем еще одну функцию отображения списка статей по рубрикам (листинг 6.6):

Листинг 6.6 – Добавление функции – обработчика categories

**def** categories(request):

**return** HttpResponse("<h1>Статьи по категориям</h1>")

Мы здесь используем тег h1, чтобы браузер отобразил эту строку как заголовок первого уровня. Затем, добавляем еще один путь в список urlpatterns (листинг 6.7):

Листинг 6.7 – Добавление пути для обработки categories.

urlpatterns = [

    path('admin/', admin.site.urls),

    path('refs/', index),

    path('cats/', categories),

]

И делаем импорт всех функций из модуля views нашего приложения (листинг 6.8).

Листинг 6.8 – Добавление всех функций из модуля views.

**from** refs.views **import** \*

Все, у нас появился новый URL-адрес:

http://127.0.0.1:8000/cats/

по которому отображается заголовок первого уровня. По аналогии мы можем добавлять самые разные URL в наш сайт.

Обратите внимание, как только мы добавили дополнительные маршруты, тестовая главная страница перестала выдаваться, вместо этого мы видим исключение 404 – страница не найдена. Чтобы задать маршрут для главной страницы, нужно вместо «refs/» записать пустую строку (листинг 6.9).

Листинг 6.9 – Изменеие маршрута для index.

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', index),

    path('cats/', categories),

]

Здесь пустая строка как раз и соответствует маршруту главной страницы и теперь при обращении к ней будет вызываться функция представления index.

Однако такой подход, когда мы маршруты приложения прописываем в пакете конфигурации, нарушает принцип независимости приложений. Действительно, если мы захотим перенести приложение refs на другой сайт, то нам придется дополнительно копировать и его маршруты, что не очень удобно и хорошо. Как это можно разрешить? Очень просто. Django позволяет вторым параметром вместо функции представления передавать список URL-адресов приложения и связанные с ними функции. Для этого мы сначала импортируем специальную функцию include (листинг 6.10).

Листинг 6.10 – Импортирование функции include.

**from** django.urls **import** path, include

А, затем, в списке маршрутов с ее помощью подключим список URL уже из нашего приложения refs (листинг 6.11).

Листинг 6.11 – Подключение списка маршрутов из refs

urlpatterns = [

    path('admin/', admin.site.urls),

    path('refs/', include('women.urls')),

]

Мы здесь в качестве параметра указываем строку, в которой сначала записываем имя приложения и через точку файл urls, где будут прописаны маршруты приложения refs. Ну а дальше все просто. Мы добавляем в приложение новый файл urls.py и в нем формируем список urlpatterns (листинг 6.12).

Листинг 6.12 – Добавление файла urls.py в приложение.

**from** django.urls **import** path

**from** .views **import** \*

urlpatterns = [

    path('', index),

]

Здесь мы, во-первых, импортируем функцию path, которая и связывает URL c функциями представления и, во-вторых, импортируем функции из модуля views текущего пакета. Далее, в списке urlpatterns вызываем функцию path,  первым параметром указываем пустую строку, а вторым функцию index. Как вы думаете, какому URL-адресу будет соответствовать эта пустая строка? Смотрите, в основном пакете конфигурации у нас указано, что адреса в 'refs.urls' следует добавлять как суффикс к адресу 'refs/', то есть, к адресу:

http://127.0.0.1:8000/refs/

Поэтому пустая строка в нашем приложении будет ссылаться именно на этот URL-адрес. Давайте проверим. Запустим веб-сервер и откроем эту страницу. Да, мы видим, что отработала именно функция index.

Если же мы добавим еще один маршрут в наш список приложения (листинг 6.13).

Листинг 6.13 – Добавление маршрута cats.

urlpatterns = [

    path('', index),

    path('cats/', categories),

]

То у нас появится еще один маршрут:

http://127.0.0.1:8000/refs/cats/

Как видите все достаточно просто и при этом мы получили относительную независимость нашего приложения от основного проекта сайта.

* 1. **Описание разработанного приложения**

Главная страница сайта показана на рисунке 6.5. С главной страницы можно перейти в любую вкладку.

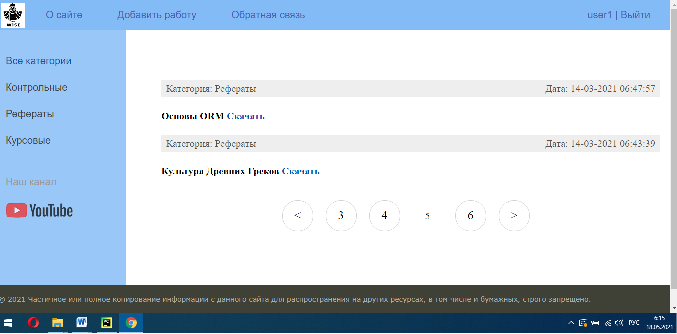


Рисунок 6.5 – Главная страница сайта

На сайте предусмотрена регистрация пользователей. Форма регистрации отображена на рисунке 6.6. Зарегистрироваться может любой пользователь. Для этого нужно придумать себе логин. Также указать электронную почту, создать надёжный пароль, который для проверки на правильность ввода пользователю предлагается ввести несколько раз. После удачной регистрации пользователь входит на сайт уже как зарегистрированный.

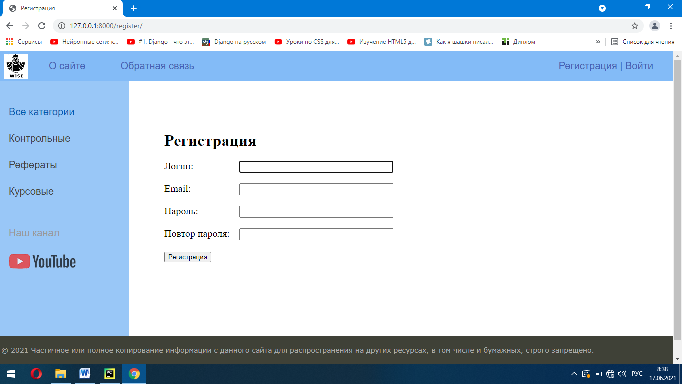


Рисунок 6.6 – Форма регистрации пользователей

Работы могут добавлять только зарегистрированные и авторизованные пользователи. После успешного прохождения авторизации (рисунок 6.7) в меню появляется вкладка «Добавить работу» (рисунок 6.8).

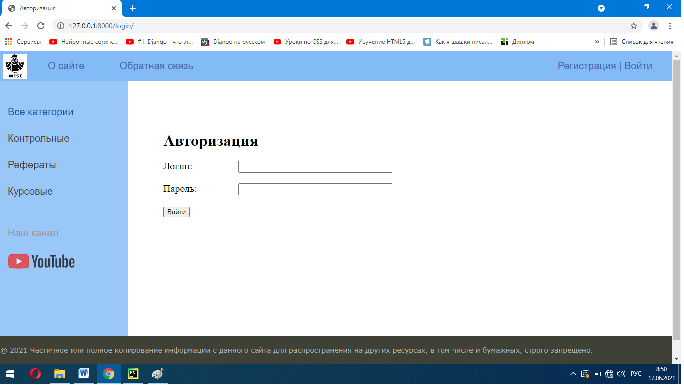


Рисунок 6.7 – Форма авторизации

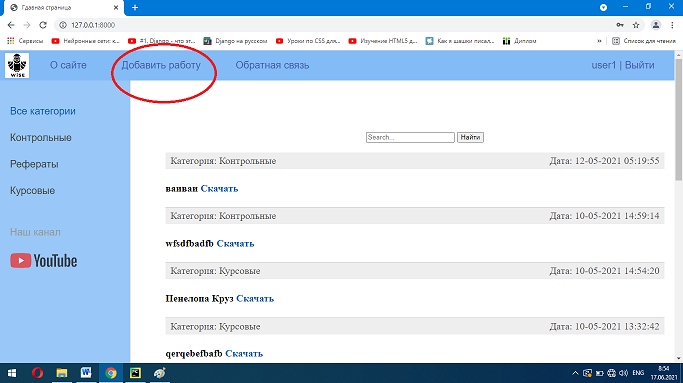


Рисунок 6.8 – Вкладка «Добавить работу»

Чтобы добавить работу пользователь должен заполнить специальную форму: должен ввести название работы, прикрепить файл и выбрать нужную категорию (рисунок 6.9).

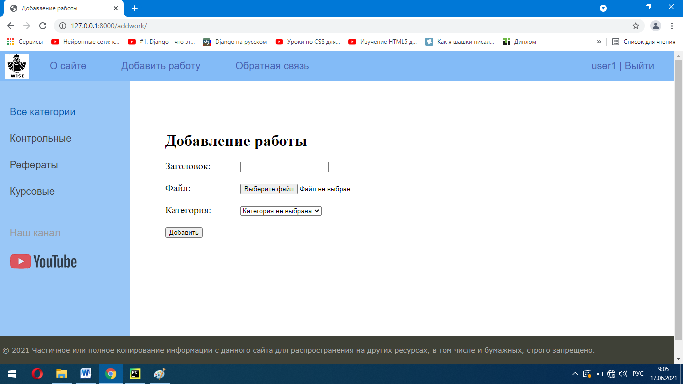


Рисунок 6.9 – Форма добавления новой работы

В меню есть обратная связь по электронной почте. Для этого необходимо перейти во вкладку «Обратная связь» главного меню сайта. Появится интуитивно понятная форма обратной связи (рисунок 6.10)

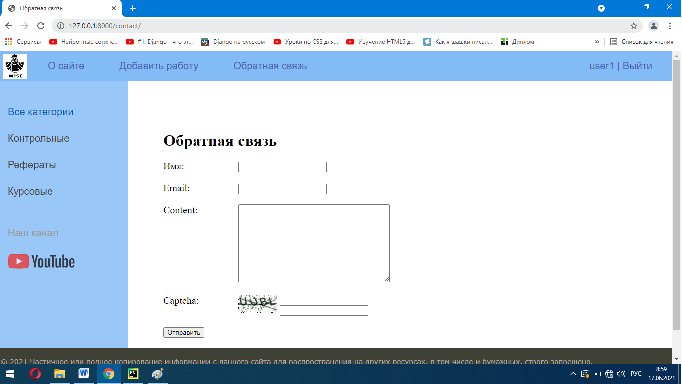


Рисунок 6.10 – Форма обратной связи

Также предусмотрена градация работ по категориям (контрольные, рефераты, дипломные). Для обеспечения бесперебойной работы администрации сайта с пользователями предусмотрен специальный паблик в мессенждере «Telegram». Такая форма обратной связи является более предпочтительной, так как пользователи могут общаться не только непосредственно с администрацией, но и друг с другом, причём в общем чате.

Ещё одной обязательной функцией сайта является система поиска (рисунок 6.11). Система поиска происходит по ключевым словам. Программа найдёт любое название работы, которое содержит поисковый запрос. Регистр неважен.

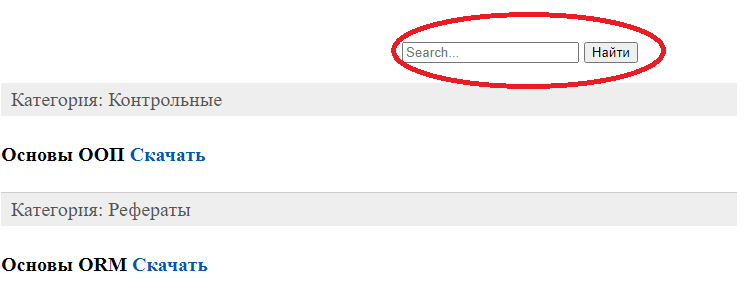
****

Рисунок 6.11 – Форма поиска

В целом поставленная задача выполнена, разработан полноценный ресурс, отвечающий всем требованиям заданной цели.

**7. Безопасность и экологичность проекта**

**7.1 Значение и задачи безопасности жизнедеятельности**

Сегодня интернет стал неотъемлемой частью жизни общества. Трудно себе представить жизнь современного человека без смартфона и ноутбука. Причём без возможности выхода в интернет эти гаджеты становятся бессмысленной техникой. Будущее за автоматизацией процессов. В том числе и за автоматизацией получения и сортировки информации. Человек значительно экономит своё время, пользуясь различными сервисами для получения той или иной информации. Именно поэтому целью данной ВКР является разработка web – сайта «Информационный портал – сборник студенческих работ».

Работа с предполагаемым программным обеспечением предполагает постоянное взаимодействие администратора портала с компьютером, что может негативно сказаться на его здоровье и работоспособности и безопасности. Работая с компьютером администратор может подвергнуть себя воздействию нескольких неблагоприятных и даже вредных факторов: электрический ток, электромагнитное излучение, различные шумы, неправильная организация рабочего места, низкое качество аппаратуры. Все эти факторы могут нанести вред зрению и опорно-двигательному аппарату работника.

Основной задачей безопасности жизнедеятельности является создание комфортных и благоприятных условий для использования портала и привлечения, обработки и проверки поступающих на портал данных.

Чтобы предотвратить подобные негативные последствия в разработанном ПО внедрены все возможные меры защиты от воздействия вредных и опасных факторов производства. Необходимые мероприятия и допустимые уровни влияния факторов при работе за компьютером регламентируются следующим пакетом документов:

- «Трудовой кодекс Российской Федерации» от 30.12.2001 № 197-ФЗ (ред. От 10.01.2016);

- СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»;

- ТОИ Р-45-084-01 «Типовая инструкция по охране труда при работе на персональном компьютере»;

- ГОСТ 12.0.003-2015 «ССБТ. Опасные и вредные производственные факторы. Классификация».

**7.2 Анализ условий труда и мероприятия по защите от воздействия вредных производственных факторов**

«Информационный портал – сборник студенческих работ» – это web-сайт, основной задачей которого является сортировка поступающей информации, в связи с этим постоянная работа с базой данных может привести к проблемам с нервной системой и болезни глаз.

Чтобы исключить все неблагоприятные факторы, влияющие на сотрудников фирмы необходимы хорошие условия труда. В связи с этим соблюдаются нормы тяжести и напряженности трудового процесса. По напряженности различают три класса условий труда: 1 класс – оптимальный (напряженность труда легкой степени); 2 класс – допустимый (напряженность труда средней степени); 3 класс (3.1, 3.2) – вредный (напряженный) труд.

Необходимо оценить напряжённость работы администратора портала с постоянно пополняющейся базой данных с помощью документа – Р 2.2.2006-05 «Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда».

Т а б л и ц а 7.1 – Оценка напряженности трудового процесса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Показатели | Класс условий труда | | | |
| 1 | 2 | 3.1 | 3.2 |
| 1. Интеллектуальные нагрузки | | | | | |
| 1.1 | Содержание работы | + |  |  |  |
| 1.2 | Восприятие информации и их оценка |  |  | + |  |
| 1.3 | Распределение функций по степени сложности задания | + |  |  |  |
| 1.4 | Характер выполняемой работы | + |  |  |  |
| 1. Сенсорные нагрузки | | | | | |
| 2.1 | Длительность сосредоточенного наблюдения (% времени смены) |  |  | + |  |
| 2.2 | Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы | + |  |  |  |
| 2.3 | Число производственных объектов одновременного наблюдения | + |  |  |  |
| 2.4 | Размер объекта различения (при расстоянии от глаз работающего до объекта различения не более 0,5 м) в мм при длительности сосредоточенного наблюдениия (% времени смены) |  | + |  |  |
| 2.5 | Работа с оптическимии приборами при длительности сосредоточения наблюдения | + |  |  |  |
| 2.6 | Наблюдение за экранами видеотерминалов (часов в смену) |  |  |  | + |
| 2.7 | Нагрузка на слуховой анализатор | + |  |  |  |

*Продолжение таблицы 6.1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2.8 | Нагрузка на голосовой аппарат | + |  |  |  |
| 1. Эмоционльные нагрузки | | | | | |
| 3.1 | Степень ответственности за результат собственной деятельности |  |  | + |  |
| 3.2 | Степень риска для собственной жизни | + |  |  |  |
| 3.3 | Ответственность за безопасность других лиц | + |  |  |  |
| 3.4 | Кол-во конфликтных ситуаций в течение смены – от 1 до 3 | + |  |  |  |
| 1. Монотонность нагрузок | | | | | |
| 4.1 | Число элементов, необходимых для реализации простого задания или многократно повторяющихся операций |  | + |  |  |
| 4.2 | Продолжительность выполнения простых заданий или повторяющихся операций |  | + |  |  |
| 4.3 | Время активных действий |  | + |  |  |
| 4.4 | Монотонность производственной обстановки | + |  |  |  |
| 1. Режим работы | | | | | |
| 5.1 | Фактическая продолжительность работчего дня |  | + |  |  |
| 5.2 | Сменность работы | + |  |  |  |
| 5.3 | Наличие регламентированных перерывов и их продолжительность | + |  |  |  |
| Кол-во показателей в каждом классе | | 14 | 5 | 3 | 1 |
| Общая оценка напряжённости | | 3,1 | | | |

Итого, получены такие результаты: к классу №1 относятся 14 показателей, к №2 – 5, №3 – 3, №4 – 1. После анализа можно сделать вывод, что работа администратора web-сайта имеет достаточно вредный характер и может спровоцировать болезнь глаз и опорно-двигательного аппарата. Чтобы избежать этих последствий, разработана система кофе-брейк и несколько небольших по длительности перерывов. Также для снижения сенсорных нагрузок разработана система звукового оповещения о поступлении новой информации.

**7.3 Обеспечение электробезопасности**

Электробезопасность в офисе работы администратора сайта обеспечивается комплексом мероприятий: безопасным устройством электро-розеток, стабилизаторами напряжения и источниками бесперебойного питания. Также проводятся плановые мероприятия по ремонту, проверке и обслуживанию источников электрического тока, проводки и всех электроприборов. В сроки проводятся специальные инструктажи по технике безопасности с сотрудниками. Рабочее место сотрудника ежедневно подвергается уборке, что позволяет исключить наличие токопроводящей пыли.

Рабочий офис администратора сайта имеет умеренный уровень опасности. Поддерживается оптимальная влажность (50 -55%), а также комнатная температура воздуха (21-23ОС). Также на электроприборах предусмотрено заземление для устранения опасности поражения электрическим током при перенапряжении.

**7.4 Пожарная безопасность**

Пожарная безопасность в офисном помещении регламентируется и соответствует нормам Федерального закона от 22.07.2008 №123-ФЗ (ред. от 29.07.2017) «Технический регламент о требованиях пожарной безопасности». Согласно п.2 Статьи 27 (в ред. Федерального закона от 10.07.2012 N 117-ФЗ) «Определение категории зданий, сооружений и помещений по пожарной и взрывопожарной опасности» офис не является производственным или складским помещением, вследствие чего не подлежит разделению на категории. Для обеспечения пожаробезопасности на предприятии выполнены следующие мероприятия:

– установлена пожарная сигнализация;

– разработан план эвакуации сотрудников из офиса согласно 53 статье «Пути эвакуации людей при пожаре» «Технического регламента о требованиях пожарной безопасности»;

– вывешены наглядные материалы в доступных местах, которые показывают путь к выходу и предоставлена возможность беспрепятственно дойти до него;

– во всех помещениях вывешены таблички с указанием номера телефона вызова пожарной охраны и инструкция по определению порядка обесточивания электрооборудования в случае пожара и по окончании рабочего дня;

– определен порядок и сроки прохождения противопожарного инструктажа и занятий по пожарно-техническому минимуму, а также назначены ответственные за их проведение.

**Заключение**

В результате проделанной работы был разработан функциональный ресурс по поиску и скачиванию готовых работ. Были реализованы все поставленные задачи, углублённо изучен фреймворк Django. Сайт разработан с учётом его дальнейшего развития и усовершенствования. Был реализован такой функционал как регистрация и авторизация пользователей. Также была проведена работа по анализу и исследованию подобных ресурсов, выявлены их плюсы и недостатки, и на этой основе спроектирован и реализован данный сервис.

**Список использованных источников**

1. Курсовые и выпускные квалифмкационные работы студентов кафедры системного программирования с 2009 по 2021 год. СПбГУ [Электронный ресурс] URL: <https://se.math.spbu.ru/theses.html> (дата обращения 01.05.2021).
2. МГППУ. Электронная бииблиотека [Электронный ресурс] URL: <http://psychlib.ru/index.php?s=tip&cat=2.5#titleWithAll> (дата обращения 01.05.2021).
3. Дронов В.А. - Django 3.0. Практика создания веб-сайтов на Python (Профессиональное программирование) Москва, 2021 – 415 с.
4. Илюшечкин В.М. - Основы использования и проектирования баз данных Москва, 2018 – 245 с.
5. Test-Driven Development with Python: Obey the Testing Goat: Using Django, Selenium, and JavaScript 2nd Edition, Kindle Edition 2021 – 312 с.

6. Маслова В.М. Безопасность жизнедеятельности [Электронный ресурс]: учеб. пособие / В.М. Маслова, И.В. Кохова, В.Г. Ляшко; Под ред. В.М. Масловой - 3 изд., перераб. и доп. - М.: Вузовский учебник: НИЦ ИНФРА-М, 2015. - 240 с. - Режим доступа: <http://znanium.com/catalog.php?bookinfo=513995> (дата обращения 05.05.2021 )

7. Никифоров Л.Л. Безопасность жизнедеятельности [Электронный ресурс]: учеб. пособие для бакалавров / Л.Л. Никифоров, В.В. Персиянов - М.: Дашков и К, 2013. - 496 с. Режим доступа: <http://znanium.com/bookread2.php?book=415279> (дата обращения 05.05.2021 )

Приложение А

**Листинг**

**А.1 Файл views.py**

from django.contrib.auth import logout, login

from django.contrib.auth.views import LoginView

from django.http import HttpResponse, HttpResponseNotFound

from django.shortcuts import render,redirect

from django.urls import reverse\_lazy

from django.views.generic import ListView, CreateView, FormView

from django.contrib.auth.mixins import LoginRequiredMixin

from .forms import \*

from .models import \*

from .utils import \*

class ReflistHome(DataMixin, ListView):

model=Reflist

template\_name = 'refs/index.html'

context\_object\_name = 'posts'

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

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

c\_def=self.get\_user\_context(title='Гдавная страница')

return dict(list(context.items())+list(c\_def.items()))

class SearchList(DataMixin,ListView):

form\_class=SearchForm

template\_name = 'refs/search\_results.html'

model = Reflist

context\_object\_name = 'posts'

def get\_queryset(self):

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

object\_list=Reflist.objects.filter(title\_\_icontains=query)

return object\_list

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

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

c\_def=self.get\_user\_context(title='Поиск')

return dict(list(context.items())+list(c\_def.items()))

def about(request):

return render(request,'refs/about.html',{'menu':menu,'title':'О сайте'})

class AddWork(LoginRequiredMixin,DataMixin, CreateView):

form\_class = AddWorkForm

template\_name = 'refs/addwork.html'

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

login\_url = reverse\_lazy('home')

raise\_exception = True

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

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

c\_def=self.get\_user\_context(title='Добавление работы')

return dict(list(context.items())+list(c\_def.items()))

class ContactFormView(DataMixin,FormView):

form\_class = ContactForm

template\_name = 'refs/contact.html'

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

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

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

c\_def=self.get\_user\_context(title='Обратная связь')

return dict(list(context.items())+list(c\_def.items()))

def form\_valid(self,form):

print(form.cleaned\_data['name'])

return redirect('home')

def pageNotFound(request, exception):

return HttpResponseNotFound('Страница не найдена')

def down\_work(request,work\_id):

return HttpResponse(f'Скачивание работы с id={work\_id}')

class ReflistCategory(DataMixin,ListView):

model=Reflist

template\_name = 'refs/index.html'

context\_object\_name = 'posts'

allow\_empty = False

def get\_queryset(self):

return Reflist.objects.filter(cat\_\_id=self.kwargs['cat\_id'])

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

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

c\_def=self.get\_user\_context(title='Категория - '+str(context['posts'][0].cat),

cat\_selected=context['posts'][0].cat\_id)

return dict(list(context.items())+list(c\_def.items()))

class RegisterUser(DataMixin, CreateView):

form\_class = RegisterUserForm

template\_name = 'refs/register.html'

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

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

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

c\_def=self.get\_user\_context(title='Регистрация')

return dict(list(context.items())+list(c\_def.items()))

def form\_valid(self, form):

user=form.save()

login(self.request,user)

return redirect('home')

class LoginUser(DataMixin, LoginView):

form\_class=LoginUserForm

template\_name='refs/login.html'

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

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

c\_def=self.get\_user\_context(title='Авторизация')

return dict(list(context.items())+list(c\_def.items()))

def get\_success\_url(self):

return reverse\_lazy('home')

def logout\_user(request):

logout(request)

return redirect('home')

**А.2 Файл models.py**

from django.db import models

from django.urls import reverse

class Reflist(models.Model):

title=models.CharField(max\_length=255, verbose\_name='Заголовок')

file=models.FileField(upload\_to='ref\_files/%Y/%m/%d', verbose\_name='Файл')

time\_create=models.DateTimeField(auto\_now\_add=True)

cat=models.ForeignKey('Category', on\_delete=models.PROTECT,verbose\_name='Категория')

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

return self.title

def get\_absolute\_url(self):

return reverse ('work', kwargs={'work\_id':self.pk})

class Meta:

verbose\_name='Список работ'

verbose\_name\_plural='Список работ'

ordering=['-time\_create']

class Category(models.Model):

name=models.CharField(max\_length=100,db\_index=True, verbose\_name='Категория')

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

return self.name

def get\_absolute\_url(self):

return reverse('category', kwargs={'cat\_id': self.pk})

class Meta:

verbose\_name='Категории'

verbose\_name\_plural='Категории'

**А.3 Файл urls.py**

from django.urls import path

from django.views.decorators.cache import cache\_page

from .views import \*

urlpatterns=[

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

path('about/', about, name='about'),

path('addwork/',AddWork.as\_view(), name='add\_work'),

path('contact/', ContactFormView.as\_view(),name='contact'),

path('login/',LoginUser.as\_view(), name='login'),

path('logout/', logout\_user, name='logout'),

path('register/',RegisterUser.as\_view(), name='register'),

path('work/<int:work\_id>',down\_work, name='work'),

path('category/<int:cat\_id>',ReflistCategory.as\_view(), name='category'),

path('search', SearchList.as\_view(), name='search'),

]