Департамент образования и науки города Москвы

Государственное бюджетное профессиональное образовательное учреждение

города Москвы

«Колледж малого бизнеса № 4»

(ГБПОУ КМБ № 4)

**КУРСОВОЙ ПРОЕКТ**

по МДК05.02 Разработка кода информационных систем, МДК 05.03 Тестирование информационных систем

для специальности 09.02.07 Информационные системы и программирование

Базовая подготовка

Тема: Разработка информационной системы для поиска и взаимодействия с менторами

Выполнил(а) студент(ка)

3 курса группы № ИПО-31.21

Бормотов Вадим Дмитриевич

(Фамилия, имя, отчество студента)

Проверил

руководитель курсовой работы

Минеева Кристина Константиновна

Работа защищена с оценкой

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

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024

Москва, 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc167360382)

[ГЛАВА 1 Теоретическая часть программного обеспечения 5](#_Toc167360383)

[1.1 Понятие, актуальные виды и особенности программного обеспечения 5](#_Toc167360384)

[1.2 Особенности программных средств разработки, моделирования и программирования 7](#_Toc167360385)

[ГЛАВА 2 Практическая часть создания проекта 16](#_Toc167360386)

[2.1 Функции и особенности Django 16](#_Toc167360387)

[2.2 Обоснование выбора Django для разработки. 17](#_Toc167360388)

[2.3 Почему Django? Анализ приложений конкурентов 19](#_Toc167360389)

[2.4 Обоснование выбора метода разработки на Django 21](#_Toc167360390)

[2.5 Поэтапная разработка сайта 22](#_Toc167360391)

[ЗАКЛЮЧЕНИЕ 30](#_Toc167360392)

[ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА 31](#_Toc167360393)

[ПРИЛОЖЕНИЕ 33](#_Toc167360394)

# **ВВЕДЕНИЕ**

В современном мире технологий, поиск ментора для студентов и аспирантов может стать довольно сложной задачей. Несмотря на наличие множества ресурсов, где можно найти ментора, не все они гарантируют качественное взаимодействие и обучение.

В данной курсовой работе предлагается разработать информационную систему для поиска и взаимодействия с менторами на фреймворке Django и использованием sqlite в качестве базы данных. Целью данной разработки является создание удобного и надежного ресурса для студентов и аспирантов, которые ищут ментора в своей области интересов.

В рамках данной разработки будет проведена аналитическая часть, которая позволит определить характеристики области разработки системы веб-сайта и обосновать выбор способа разработки. Затем, пройдет процесс проектирования системы, разработки системы, тестирования системы и развития системы.

Курсовая работа будет включать в себя различные модули, такие как:

* Поиск ментора по специализации и интересам.
* Создание и управление профилем ментора.

По завершению разработки, будет проведено тестирование системы, которое позволит выявить и устранить все возможные недостатки и ошибки. После чего, система будет представлена в виде рабочего сайта, доступного для использования студентами и аспирантами.

Курсовая работа завершится заключением, где будут описаны основные результаты разработки, выводы и рекомендации по развитию системы в будущем.

Для достижения данной цели в работе решаются следующие задачи:

* Добавление базы данных в веб-приложение.
* Знание языков программирования HTML, Python, CSS, JAVASCRIPTS, SQL, Django.
* Описание программы для создания веб-приложения Visual Studio Code.
* Разработка информационной системы для поиска и взаимодействия с менторами.

# **ГЛАВА 1 Теоретическая часть программного обеспечения**

В данной главе представлено описание существующих и актуальных видов программного обеспечения, а также особенности программных средств разработки, моделирования и программирования.

# **1.1 Понятие, актуальные виды и особенности программного обеспечения**

Программное обеспечение (ПО) — это набор программ, используемых для управления компьютером. Согласно международным и российским стандартам, ПО также определяется как совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ.

Цель программного обеспечения — осуществление управления электронными и механическими частями компьютера. ПО — это гибкая система, имеющая возможность усовершенствования. Программный продукт, использующий информацию, превращает аппаратную часть ПК в комплекс, с помощью которого возможно решать самые разнообразные задачи.

Жизненный цикл любого компьютерного программного обеспечения состоит из трех основных стадий: написание, работа и поддержка. Сам процесс написания условно подразделяется на 6 этапов:

1. Постановка задачи.
2. Создание концепции.
3. Выработка алгоритма.
4. Формирование и компиляция.
5. Выявление ошибок и отладка.
6. Завершение процедур по оформлению документации.

Особенности ПО:

* Практически любым процессом можно управлять определённым набором инструкций.
* Затруднение разработки состоит в сложности его оцифровки.
* Создание нового ПО осуществляется на ПК с установленным программным обеспечением.
* Написание кода менее трудоёмкая операция, чем его отладка и настройка управления.
* Программное обеспечение пользователя является методом реализации поставленных задач.

Виды программного обеспечения: краткий обзор.

В мире программного обеспечения существует огромное разнообразие, но все программы можно разделить на три основные категории:

Системное программное обеспечение.

Это фундамент, на котором строится работа компьютера. Системное ПО управляет всеми аппаратными ресурсами, такими как процессор, память, устройства ввода-вывода и сеть. Оно обеспечивает взаимодействие между пользователем, аппаратными компонентами и прикладными программами.

**Примеры системного ПО -** операционные системы (Windows, macOS, Linux), драйверы устройств, утилиты для управления дисками и сетью.

Прикладное программное обеспечение.

Это программы, которые используем для выполнения конкретных задач. Они помогают нам работать, учиться, развлекаться и общаться.

**Примеры прикладного ПО -** текстовые редакторы, графические редакторы, веб-браузеры, игры, офисные приложения, программы для обработки видео и аудио.

Инструментальное программное обеспечение.

Это ПО, предназначенное для разработчиков. Оно помогает создавать новые программы и управлять процессом разработки.

**Примеры инструментального ПО -** компиляторы, интерпретаторы, отладчики, системы управления версиями, интегрированные среды разработки (IDE).

# **1.2 Особенности программных средств разработки, моделирования и программирования**

Классификация информационных систем по различным признакам.

ИС могут быть классифицированы по различным признакам, таким как:

* Уровень управления - оперативные, тактические, стратегические.
* Функциональная направленность - производственные, управленческие, научно-технические.
* Сфера применения - автоматизированные системы управления (АСУ), системы поддержки принятия решений (СППР), экспертные системы.
* Обеспечивающая подсистема: - техническое обеспечение, программное обеспечение, информационное обеспечение.

Модели жизненного цикла информационных систем.

Модель жизненного цикла ИС – это совокупность этапов, через которые проходит ИС в процессе своего создания, эксплуатации и развития.

Наиболее распространенными моделями жизненного цикла ИС являются:

* Каскадная модель - предполагает последовательное выполнение этапов разработки ИС.
* Итерационная модель - предполагает поэтапную разработку ИС с возможностью внесения изменений в проект на каждом этапе.
* Спиральная модель - сочетает в себе элементы каскадной и итерационной моделей.

Методологии разработки информационных систем.

Методология разработки ИС – это набор принципов, методов и средств, используемых для разработки ИС.

Наиболее распространенными методологиями разработки ИС являются:

* Структурированный подход - предполагает декомпозицию ИС на подсистемы и модули.
* Объектно-ориентированный подход - предполагает разработку ИС на основе объектов.
* Компонентный подход - предполагает разработку ИС из готовых компонентов.

Обеспечивающие подсистемы информационных систем.

Обеспечивающая подсистема ИС – это совокупность технических, программных и информационных средств, необходимых для функционирования ИС.

* Техническое обеспечение - включает в себя компьютеры, серверы, сети, периферийные устройства и т.д.
* Программное обеспечение - включает в себя операционные системы, системное программное обеспечение, прикладное программное обеспечение и т.д.
* Информационное обеспечение - включает в себя данные, базы данных, знания и т.д.

Техническое и программное обеспечение информационных систем

* Масштабируемость - ИС должна быть масштабируемой, чтобы ее можно было легко расширить по мере необходимости.
* Безопасность - ИС должна быть защищена от несанкционированного доступа и утечки информации.
* Стоимость - Стоимость технического и программного обеспечения должна быть приемлемой.

Информационное обеспечение информационных систем.

Информационное обеспечение ИС является основой для принятия решений. Оно должно быть актуальным, достоверным, полным и структурированным.

Разработка информационного обеспечения ИС включает в себя:

* Определение состава и структуры данных.
* Разработку схем баз данных.
* Создание процедур ввода, обработки и хранения данных.

Правовые и организационные основы создания информационных систем

Создание ИС должно осуществляться в соответствии с действующим законодательством. Необходимо учитывать такие правовые аспекты, как:

* Авторское право.
* Право на неприкосновенность частной жизни.
* Защита персональных данных.

Организационные основы создания ИС включают в себя:

* Создание рабочей группы по разработке ИС.
* Разработку технического задания на ИС.
* Планирование работ по разработке ИС.
* Бюджетирование проекта.

Безопасность информационных систем.

Безопасность ИС является важнейшей задачей. Необходимо обеспечить защиту ИС от следующих угроз:

* Несанкционированный доступ.
* Вредоносное программное обеспечение (ПО).
* Атаки на серверы.
* Утечка информации.

Для обеспечения безопасности ИС необходимо использовать различные методы и средства, такие как:

* Авторизация и аутентификация пользователей.
* Шифрование данных.
* Системы обнаружения и предотвращения вторжений (IDS/IPS).
* Резервное копирование данных.

Django — это высокоуровневый веб-фреймворк Python, который используется для разработки веб-приложений с быстрой разработкой, административным интерфейсом и мощной функциональностью. На рисунке 1 изображён логотип django. Он основан на архитектуре "мнение против конфигурации", что означает, что он поощряет использование конвенций вместо явной конфигурации. Django включает в себя множество функций, которые упрощают разработку веб-приложений, таких как:

* Система маршрутизации - Django позволяет вам определять URL-адреса для своего приложения и сопоставлять их с представлениями, которые генерируют HTML-ответы.
* Система шаблонов - Django позволяет вам использовать шаблоны HTML для создания динамического контента.
* Система объектно-реляционного отображения (ORM) - Django позволяет вам взаимодействовать с базой данных с помощью Python-объектов.
* Система административного интерфейса - Django поставляется с административным интерфейсом, который позволяет вам управлять контентом своего сайта.

Django — это популярный выбор для разработки веб-приложений Python, потому что он прост в использовании, мощный и надежный.



Рисунок 1 - Логотип фреймворка Django

Python — это универсальный язык программирования высокого уровня, который используется для широкого спектра задач, включая веб-разработку, научные вычисления и машинное обучение. Он известен своим читаемым и лаконичным синтаксисом, что делает его легким для изучения и использования. Python также имеет большую и активную сообщество пользователей и разработчиков, что означает, что существует множество ресурсов, доступных для помощи вам в изучении языка. На рисунке 2 изображён логотип Python.

Некоторые из ключевых функций Python включают:

* Читаемый синтаксис - Python часто описывается как язык программирования, похожий на английский, что делает его легким для изучения и использования.
* Интерпретируемый язык - Python — это интерпретируемый язык, что означает, что код не компилируется в машинный код перед его выполнением. Это делает Python более гибким и простым в использовании, чем компилируемые языки.
* Широкая библиотека стандартных модулей - Python поставляется с большой библиотекой стандартных модулей, которые предоставляют функции для выполнения широкого спектра задач.
* Большое и активное сообщество - У Python большое и активное сообщество пользователей и разработчиков, что означает, что существует множество ресурсов, доступных для помощи вам в изучении языка.

Python — это популярный выбор для разработки программного обеспечения, потому что он прост в использовании, мощный и универсальный.



Рисунок 2 – Логотип Python

SQLite — это легкая, автономная база данных SQL с открытым исходным кодом. Она не требует отдельного сервера базы данных и хранит свои данные в одном файле. SQLite — это популярный выбор для встраиваемых приложений, таких как мобильные приложения и настольные приложения. На рисунке 3 изображён логотип SQLite.

Некоторые из ключевых функций SQLite включают:

* Легкая и автономная - SQLite не требует отдельного сервера базы данных и хранит свои данные в одном файле. Это делает его идеальным для встраиваемых приложений.
* Быстрая - SQLite — это быстрая база данных, которая может обрабатывать запросы с высокой скоростью.
* Надежная - SQLite — это надежная база данных, которая имеет низкую вероятность сбоя или повреждения данных.
* С открытым исходным кодом - SQLite - это база данных с открытым исходным кодом, что означает, что она бесплатна для использования и модификации.



Рисунок 3 - Логотип SQLite

HTML (HyperText Markup Language) — это язык разметки, который используется для создания веб-страниц. Он определяет структуру и содержание веб-страницы, используя теги. HTML — это базовый язык, который используется для создания всех веб-страниц. На рисунке 4 изображён логотип HTML.

Некоторые из ключевых элементов HTML включают:

* Теги HTML - Теги HTML используются для определения структуры и содержимого веб-страницы.
* Атрибуты тегов - Атрибуты тегов используются для предоставления дополнительной информации о теге.
* Элементы HTML - Элементы HTML — это строительные блоки веб-страниц.

HTML — это относительно простой язык, который легко изучить. Существует множество ресурсов, доступных для помощи вам в изучении HTML, включая онлайн-учебные пособия, книги и статьи.



Рисунок 4 - Логотип HTML

CSS (Cascading Style Sheets) — это язык стилей, который используется для форматирования визуального представления веб-страниц, написанных на HTML или XML. Он отделяет содержание веб-страницы (HTML) от ее представления (CSS), что делает код более управляемым и упрощает обслуживание веб-страниц. На рисунке 5 изображён логотип CSS.

Некоторые из ключевых возможностей CSS включают:

* Управление стилями элементов - CSS позволяет вам управлять стилями элементов HTML, такими как цвет, шрифт, размер, расположение и т.д.
* Селекторы - CSS использует селекторы для определения того, какие элементы HTML должны быть стилизованы.
* Свойства и значения - Свойства CSS используются для определения того, какой аспект элемента должен быть стилизован, а значения используются для определения значения этого свойства.
* Каскадирование - Каскадирование — это то, как браузеры решают, какие стили применять к элементу, когда несколько стилей могут применяться.

Использование CSS позволяет вам создавать более привлекательные и удобные веб-страницы. Существует множество ресурсов, доступных для помощи вам в изучении CSS, включая онлайн-учебные пособия, книги и статьи.



Рисунок 5 - Логотип HTML

JavaScript (JavaScript) — это высокоуровневый, интерпретируемый язык программирования, который используется для создания интерактивных элементов на веб-страницах. Он позволяет вам добавлять динамическое поведение к веб-страницам, что делает их более привлекательными и удобными для пользователей. На рисунке 6 изображён логотип JavaScript.

Некоторые из ключевых возможностей JavaScript включают:

* Манипуляция DOM - JavaScript позволяет вам манипулировать Document Object Model (DOM), который представляет собой древовидную структуру документа HTML. Это позволяет вам динамически добавлять, удалять и изменять содержимое веб-страницы.
* Обработка событий - JavaScript позволяет вам обрабатывать события, такие как клики мыши, нажатия клавиш и загрузки страницы. Это позволяет вам создавать интерактивные элементы на веб-страницах.
* Асинхронное программирование - JavaScript позволяет выполнять код асинхронно, что означает, что код может выполняться без блокировки браузера. Это делает веб-страницы более отзывчивыми.

JavaScript является одним из основных языков программирования, используемых для создания современных веб-приложений. Существует множество ресурсов, доступных для помощи вам в изучении JavaScript, включая онлайн-учебные пособия, книги, статьи и видеокурсы.



Рисунок 6 - Логотип JavaScript

# **ГЛАВА 2 Практическая часть создания проекта**

Данная глава представляет собой основную информацию по созданию сайта, а также программирование и его особенности.

# **2.1 Функции и особенности Django**

Django — это высокоуровневый веб-фреймворк Python, который используется для разработки веб-приложений. Он обладает множеством функций, которые делают его мощным и универсальным инструментом для создания различных типов веб-сайтов и приложений.

Основные функции Django:

* URL-маршрутизация - Django позволяет легко определять URL-адреса для вашего приложения и связывать их с соответствующими представлениями.
* Система шаблонов - Django имеет встроенную систему шаблонов, которая позволяет создавать динамические HTML-страницы с использованием Python-кода.
* Объектно-реляционная отображение (ORM) - Django упрощает работу с базами данных, позволяя вам работать с моделями Python, которые отображаются на таблицы базы данных.
* Административная панель - Django включает в себя встроенную административную панель, которая позволяет вам управлять контентом вашего сайта через веб-интерфейс.
* Система аутентификации и авторизации - Django предоставляет функции для аутентификации пользователей и управления их правами доступа.
* Поддержка AJAX - Django упрощает разработку AJAX-приложений, позволяя вам легко отправлять и получать асинхронные запросы с сервера.
* Безопасность - Django включает в себя ряд функций безопасности, которые помогут защитить ваше приложение от распространенных уязвимостей.
* Расширяемость - Django — это модульная платформа, которую можно расширять с помощью сторонних пакетов и приложений.

Особенности Django:

* Легко изучить и использовать - Django имеет относительно простую кривую обучения, что делает его доступным для начинающих разработчиков.
* Быстрое развитие - Django — это активно развивающийся проект с большим сообществом пользователей и разработчиков.
* Масштабируемость - Django может использоваться для создания приложений любого масштаба, от простых веб-сайтов до сложных веб-приложений с высокой нагрузкой.
* Безопасность - Django уделяет большое внимание безопасности, что делает его надежным выбором для разработки корпоративных приложений.
* Гибкость - Django можно использовать для создания различных типов веб-приложений, от блогов и форумов до интернет-магазинов и социальных сетей.

Django является отличным выбором для разработки:

* Веб-сайтов - Блоги, форумы, новостные сайты, интернет-магазины, лендинги.
* Веб-приложений - CRM-системы, ERP-системы, системы управления проектами, системы управления контентом (CMS).
* RESTful API - Django REST framework — это популярная библиотека для создания REST API на Django.

# **2.2 Обоснование выбора Django для разработки.**

1. Простота изучения и использования:

* Django обладает относительно простой кривой обучения, что делает его доступным для начинающих разработчиков.
* Это особенно важно, так как в проекте могут быть задействованы студенты, которые не имеют большого опыта разработки веб-приложений.

2. Быстрое развитие:

* Django — это активно развивающийся проект с большим сообществом пользователей и разработчиков.
* Это означает, что вы всегда будете иметь доступ к актуальной документации, примерам кода и сторонним библиотекам.
* Кроме того, вы можете быть уверены, что в случае возникновения проблем вы сможете найти помощь на форумах или в сообществах Django.

3. Масштабируемость:

* Django может использоваться для создания приложений любого масштаба, от простых веб-сайтов до сложных веб-приложений с высокой нагрузкой.
* Это означает, что вы можете быть уверены, что ваша информационная система сможет расти и развиваться вместе с вашими потребностями.

4. Безопасность:

* Django уделяет большое внимание безопасности, что делает его надежным выбором для разработки корпоративных приложений.
* В информационной системе поиска менторов будут храниться личные данные пользователей, поэтому важно, чтобы система была защищена от несанкционированного доступа.

5. Гибкость:

* Django можно использовать для создания различных типов веб-приложений, от блогов и форумов до интернет-магазинов и социальных сетей.
* Это означает, что вы можете использовать Django для реализации всех функций, которые вам необходимы в вашей информационной системе.

6. Наличие готовых решений:

* Существуют уже готовые решения для создания некоторых функций информационной системы, таких как система аутентификации и авторизации, система управления контентом (CMS) и система управления файлами.
* Использование этих решений может значительно ускорить процесс разработки.

7. Соответствие требованиям:

* Django обладает всеми необходимыми функциями для разработки информационной системы поиска менторов, такими как URL-маршрутизация, система шаблонов, ORM, система аутентификации и авторизации, поддержка AJAX и многое другое.

8. Поддержка сообщества:

* Django имеет большое и активное сообщество пользователей и разработчиков, которые всегда готовы помочь и ответить на ваши вопросы.
* Он прост в изучении, обладает богатым функционалом, масштабируем, безопасен, гибок и имеет большое сообщество пользователей.

Использование Django позволит вам:

* Быстро и эффективно разработать информационную систему.
* Создать систему, которая будет соответствовать вашим требованиям.
* Обеспечить безопасность и надежность системы.

# **2.3 Почему Django? Анализ приложений конкурентов**

Выбор Django для разработки информационной системы поиска менторов обосновывается не только его собственными преимуществами, но и сравнительным анализом с фреймворками, используемыми в конкурирующих приложениях.

Рассмотрим несколько популярных приложений для поиска менторов:

* MentorMe - использует Python и Flask.
* Mentee - использует Node.js и Express.js.
* Protégé - использует Ruby on Rails.
* Gigster - использует Go.

Сравнительный анализ этих приложений по используемым фреймворкам:

Django обладает рядом преимуществ по сравнению с фреймворками, используемыми в конкурирующих приложениях:

* Простота изучения и использования - Django обладает относительно простой кривой обучения, что делает его доступным для начинающих разработчиков.
* Быстрая разработка - Django позволяет быстро создавать прототипы и MVP приложения.
* Масштабируемость - Django может использоваться для создания приложений любого масштаба, от простых веб-сайтов до сложных веб-приложений с высокой нагрузкой.
* Безопасность - Django уделяет большое внимание безопасности, что делает его надежным выбором для разработки корпоративных приложений.
* Гибкость - Django можно использовать для создания различных типов веб-приложений, от блогов и форумов до интернет-магазинов и социальных сетей.
* Наличие готовых решений - Существуют уже готовые решения для создания некоторых функций информационной системы, таких как система аутентификации и авторизации, система управления контентом (CMS) и система управления файлами.
* Соответствие требованиям - Django обладает всеми необходимыми функциями для разработки информационной системы поиска менторов, такими как URL-маршрутизация, система шаблонов, ORM, система аутентификации и авторизации, поддержка AJAX и многое другое.

Помимо этого, Django имеет большое и активное сообщество пользователей и разработчиков, которые всегда готовы помочь и ответить на ваши вопросы.

Таким образом, Django является не только простым и удобным в использовании фреймворком, но и обладает всеми необходимыми функциями и преимуществами для разработки информационной системы поиска менторов, которая будет соответствовать всем вашим требованиям.

# **2.4 Обоснование выбора метода разработки на Django**

Выбор метода разработки на Django для создания информационной системы поиска менторов обуславливается следующими факторами:

1. Подход, ориентированный на пользователя:

* При разработке системы необходимо ориентироваться на потребности пользователей.
* Это означает, что необходимо провести анализ требований пользователей, создать прототипы системы и получить обратную связь от пользователей на протяжении всего процесса разработки.

2. Итеративный подход:

* Разработка системы должна осуществляться итеративно, разбивая ее на небольшие этапы.
* Это позволит вам гибко реагировать на изменения требований и вносить необходимые коррективы в систему.

3. Agile-методологии:

* Для разработки системы можно использовать Agile-методологии, такие как Scrum или Kanban.
* Эти методологии позволяют эффективно управлять работой команды, повышать вовлеченность пользователей и быстро реагировать на изменения.

4. Контроль версий:

* Для контроля версий кода системы необходимо использовать систему контроля версий, такую как Git.
* Это позволит вам отслеживать изменения в коде, откатываться к предыдущим версиям и работать над системой совместно с другими разработчиками.

5. Автоматизированное тестирование:

* Для обеспечения качества системы необходимо использовать методы автоматизированного тестирования.
* Это позволит вам автоматизировать рутинные задачи тестирования и повысить надежность системы.

6. Непрерывная интеграция и доставка (CI/CD):

* Для автоматизации процесса разработки и развертывания системы можно использовать CI/CD-подход.
* Это позволит вам автоматизировать такие задачи, как сборка кода, тестирование и развертывание системы на сервере.

7. Документация:

* Важно вести подробную документацию системы, которая будет включать в себя описание архитектуры системы, API, руководства пользователя и т.д.
* Это позволит вам поддерживать систему в дальнейшем и облегчит работу другим разработчикам.

Использование этих методов позволит вам:

* Разработать информационную систему поиска менторов, которая будет соответствовать потребностям пользователей.
* Обеспечить высокое качество системы.
* Эффективно управлять процессом разработки.
* Снизить риски проекта.

# **2.5 Поэтапная разработка сайта.**

**1. Подготовка:**

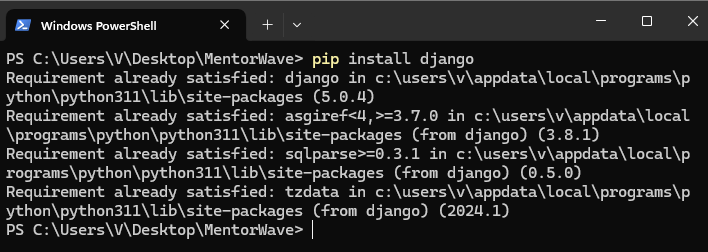
* **Создайте папку MentorWave**
  + Выберите место на вашем компьютере, где будете хранить файлы проекта.
  + Создайте в этом месте новую папку с названием "MentorWave".
* **Откройте терминал**
  + В системах Windows, скорее всего, вам понадобится использовать командную строку.
  + Перейдите в созданную вами папку MentorWave с помощью команды cd (например, cd MentorWave).
* **Установите Python и Django:**
  + **Python**
    - Скачайте установщик Python с официального сайта.
    - Запустите установщик и следуйте инструкциям.
  + **Django:**
    - В терминале выполните команду: pip install django. На рисунке 7 изображён терминал после установки библиотеки. 

Рисунок 7 – Установка библиотеки

**2. Инициализация проекта:**

* **Создайте проект Django:**
  + В терминале выполните команду: django-admin startproject App, где App - название вашего приложения. На рисунке 8 изображено строение проекта
  + Это создаст структуру папок и файлов для вашего проекта Django.
* **Запустите сервер разработки:**

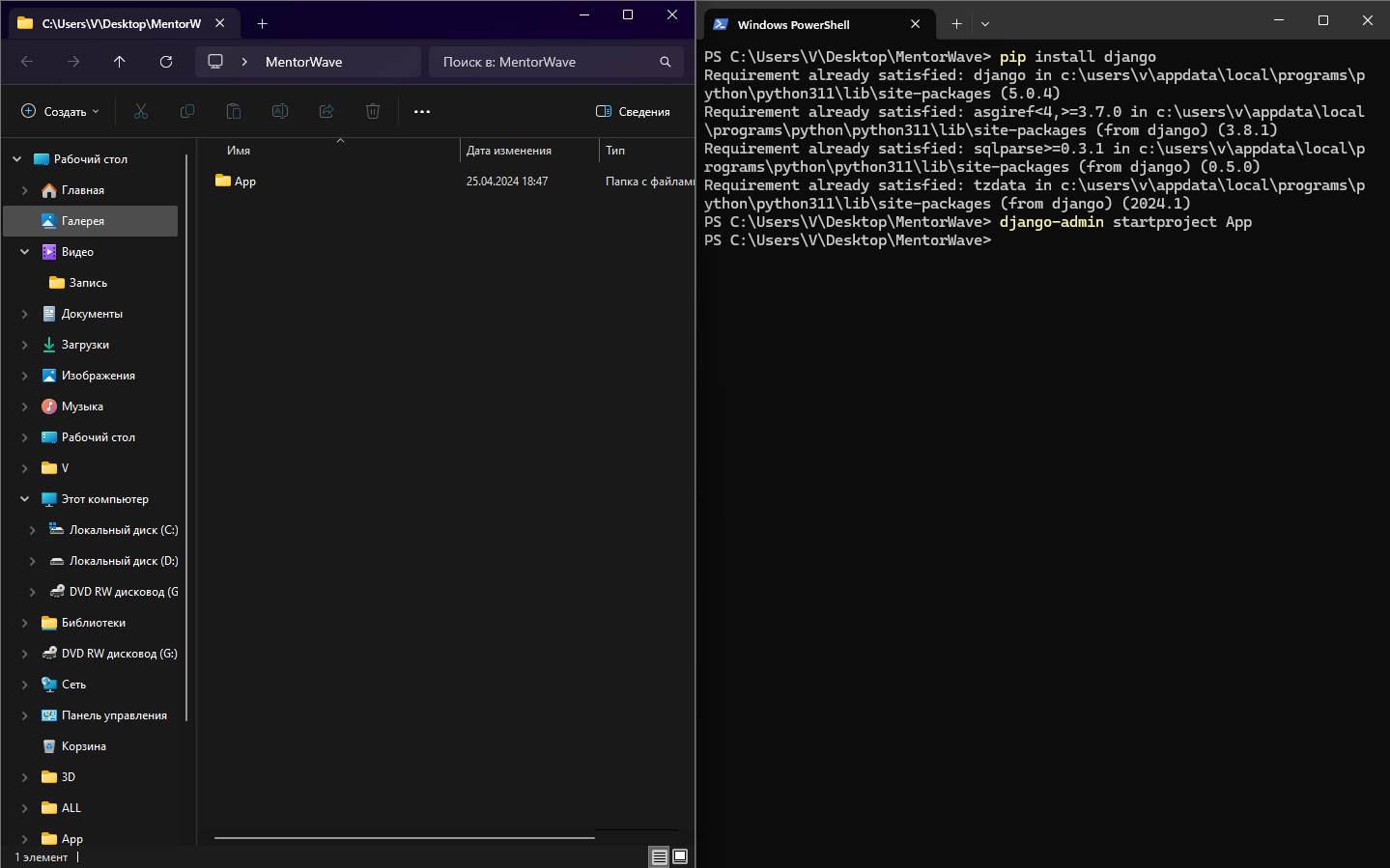


Рисунок 8 – Строение проекта

Перейдите в папку проекта (cd App).

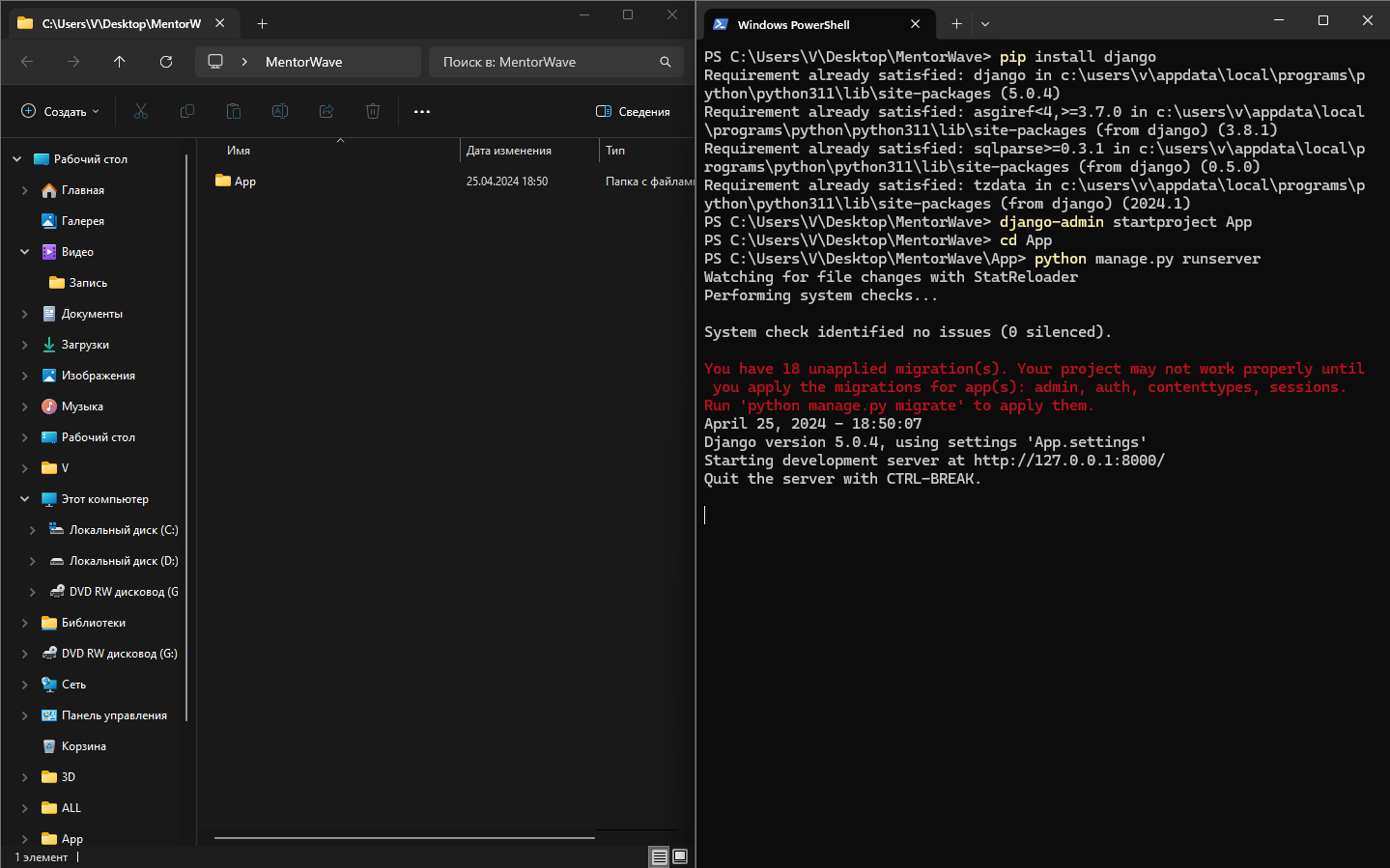
* + Выполните команду: python manage.py runserver
  + Откройте браузер и перейдите по адресу http://localhost:8000.
  + Вы должны увидеть стартовую страницу Django.
  + На рисунке 9 изображён запуск сервера.
  + На рисунке 10 изображена начальная страница. 

Рисунок 9 - Запуск сервера

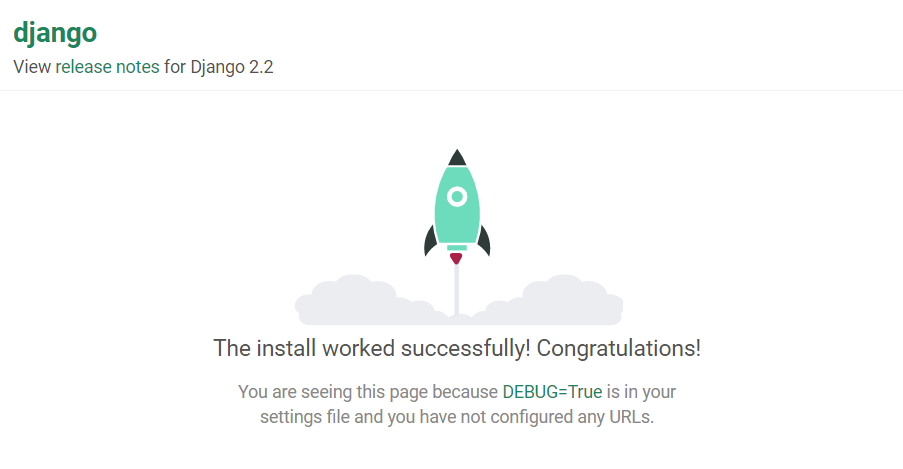


Рисунок 10 - Начальная страница

**3. Разработка приложений:**

* **Создайте приложения:**
  + В терминале выполните команды:
    - python manage.py startapp main для создания приложения main.
    - python manage.py startapp authenticate для создания приложения authenticate.
  + Это создаст структуру папок и файлов для ваших приложений.
* **Добавьте приложения в settings.py:**
  + Откройте файл App/settings.py.
  + В списке INSTALLED\_APPS добавьте строки:
    - 'main',
    - 'authenticate'.
* **Создайте папки для статических файлов и шаблонов:**
  + В папке App создайте папки static и templates.
  + В папку static будете помещать CSS, JavaScript и другие статические файлы.
  + В папку templates будете помещать HTML-шаблоны для ваших приложений.

**Ваш settings.py**

**4. Разработка интерфейса:**

* **Создайте HTML-шаблоны:**
  + В папках templates/main и templates/authenticate создайте HTML-файлы для разных страниц вашего приложения (например, index.html, profile.html, search.html).
  + В HTML-шаблонах используйте переменные Django для динамического вывода контента.
  + Создать templates/base.html
* **Напишите CSS.**
  + В папках static/main и static/authenticate создайте CSS-файлы для стилизации вашего сайта.
  + Используйте CSS-селекторы для изменения оформления HTML-элементов.
* **Напишите JavaScript.**
  + (Необязательно) В папках static/main и static/authenticate создайте JavaScript-файлы для добавления интерактивности на ваш сайт.
  + Используйте JavaScript для обработки событий, создания динамических элементов и т.д.

**5. Разработка функциональности:**

* **Создайте модели:**
  + В папке authenticate/models.py опиши модели данных для ваших приложений, используя классы Django models.
* **Напишите представления:**
  + В папках main/views.py и authenticate/views.py создайте функции представления для обработки запросов пользователей.
  + Используйте модели Django для работы с данными.
  + Возвращайте из функций представления HTTP-ответы с HTML-содержимым.
  + Вот модели:
* **Создайте URL-адреса:**
  + В файле App/urls.py добавьте URL-адреса для ваших приложений.
  + Сопоставьте URL-адреса с функциями представления.

**6. Настройка Django:**

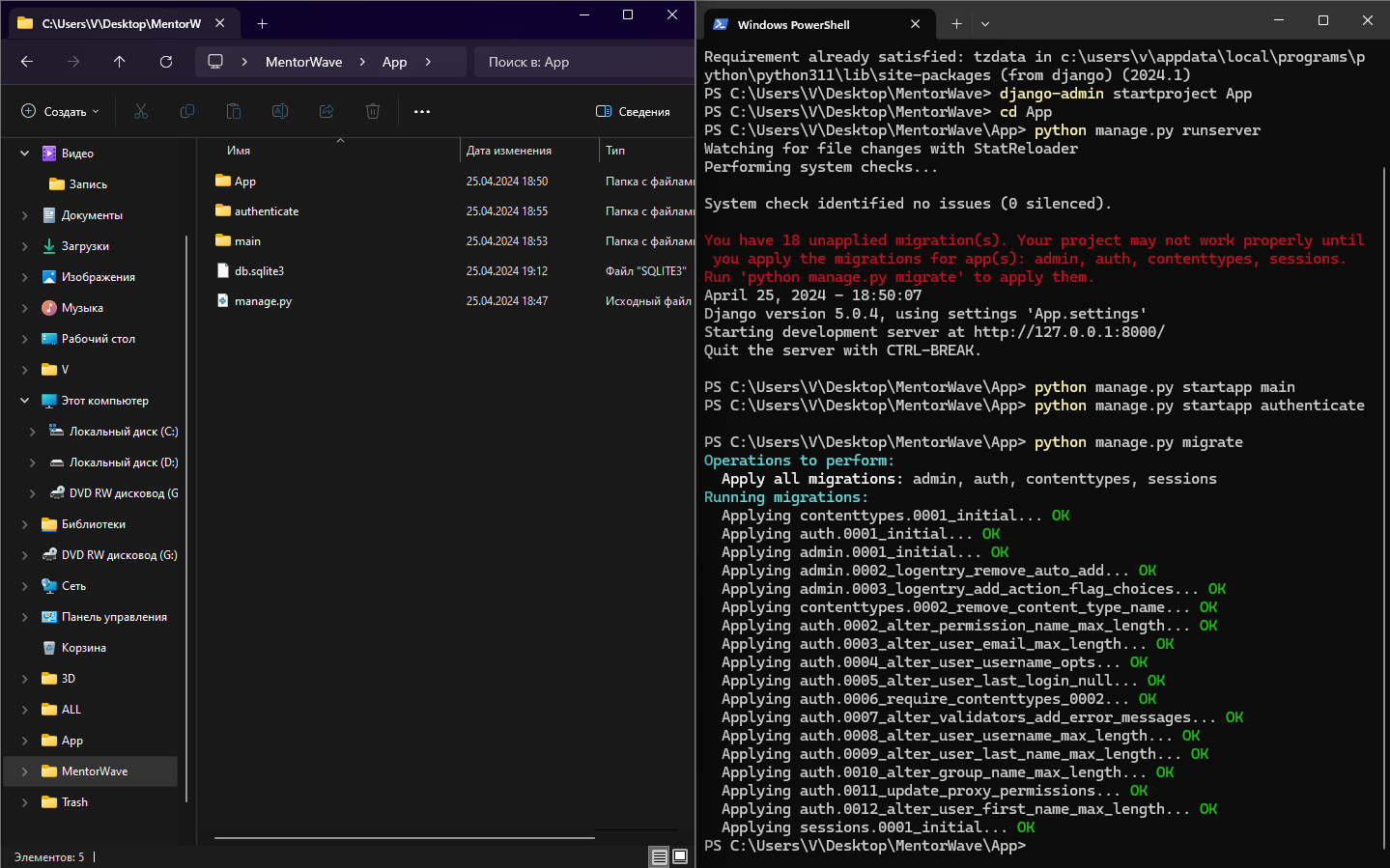
* **Настройте авторизацию:**
  + Определите модели пользователей и группы в settings.py.
  + Используйте систему аутентификации Django для защиты страниц вашего сайта.
* Настройте базу данных:
  + создайте таблицы базы данных с помощью команды python manage.py makemigrations и python manage.py migrate.
  + На рисунке 11 изображены миграции.
* Настройте систему сообщений:
  + В settings.py добавьте 'django.contrib.messages' в список INSTALLED\_APPS.
  + Настройте теги сообщений в ваших шаблонах для вывода сообщений пользователям.
* Настройте систему управления файлами (необязательно):
  + Если вашему сайту требуется хранение загруженных файлов (изображений, документов и т.д.), то вам следует рассмотреть использование стороннего приложения, такого как django-storages. Это позволит хранить файлы в облачном хранилище или другой службе.
* Настройте формы Django:
  + Используйте формы Django для создания форм регистрации, входа, поиска и редактирования данных.
  + Формы Django помогут вам обрабатывать ввод данных пользователей и выполнять соответствующие действия.
* Настройте систему кеширования (необязательно):
  + Для улучшения производительности сайта вы можете настроить систему кеширования, такую как django-redis. Это позволит кэшировать часто запрашиваемые данные и снизить нагрузку на базу данных.

7. Разработка и тестирование:

* Разработайте и протестируйте каждую функцию вашего сайта.
  + Пишите код поэтапно, тестируя его работоспособность на каждом этапе.
  + Используйте инструменты для автоматического тестирования, такие как Django test framework.
* Обеспечьте безопасность сайта.
  + Валидируйте пользовательский ввод, чтобы предотвратить внедрение SQL-кода (SQL injection) и другие уязвимости.
  + Используйте безопасные методы для работы с паролями пользователей.

8. Документирование:

* Задокументируйте ваш код.
* На рисунке 12 главная страница сайта.
* На рисунке 13 страница логина.

Рисунок 11 - Делаем миграции

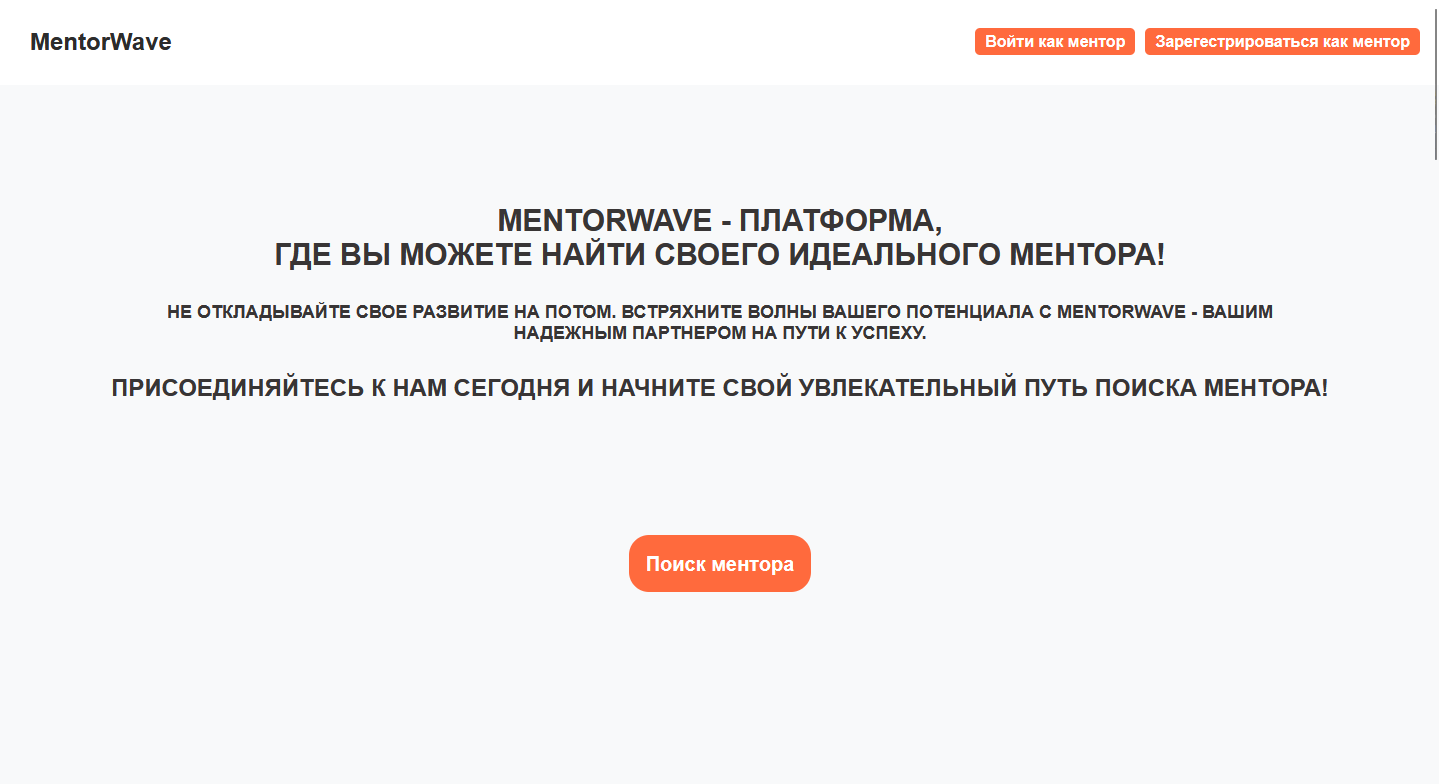
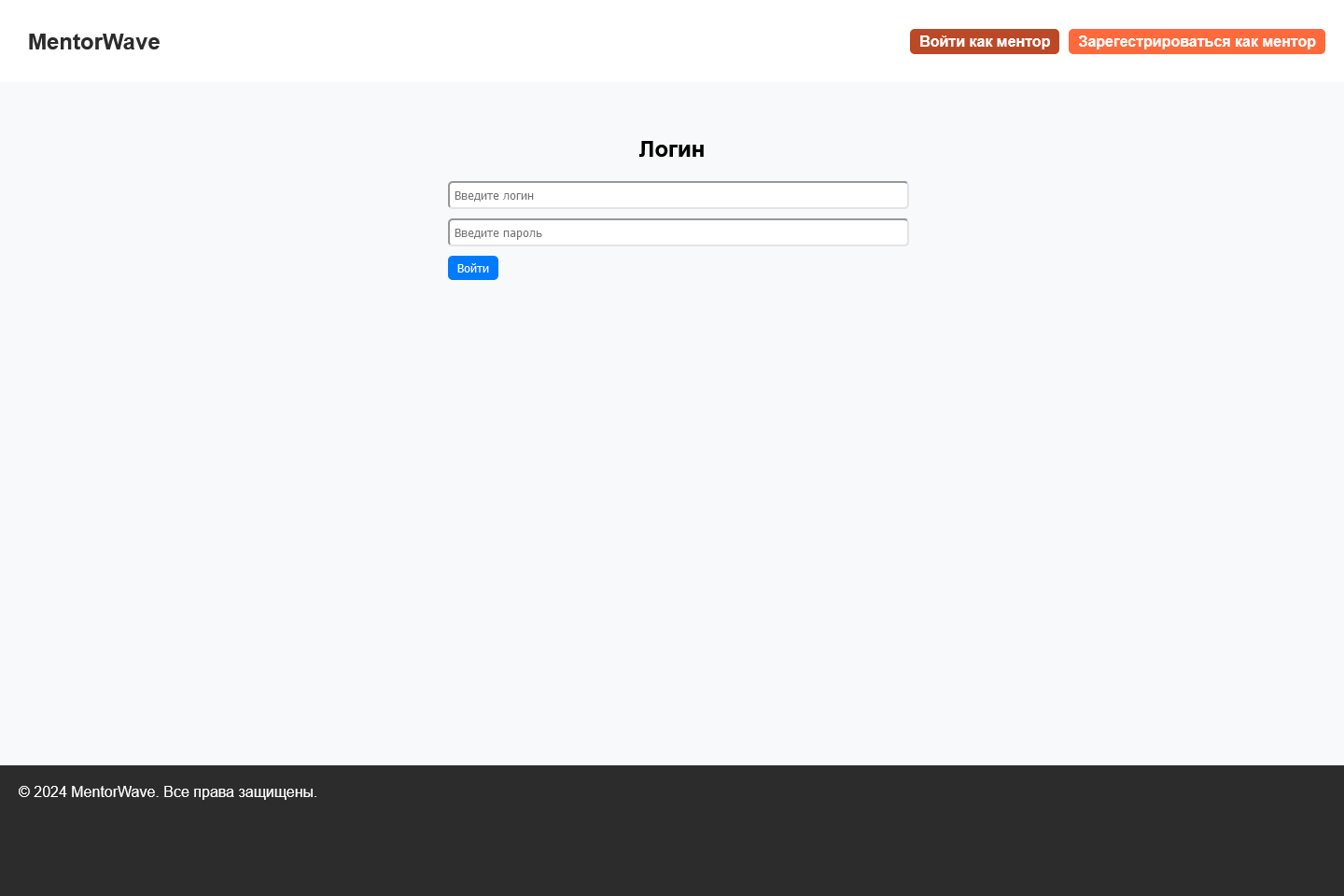
Рисунок 12 – Главная страница

Рисунок 13 – Страница логина

# **ЗАКЛЮЧЕНИЕ**

**Создание сайта MentorWave стало актуальным и необходимым шагом в современном мире.**

С каждым годом все больше людей ищут возможности для саморазвития и получения новых знаний. MentorWave — это платформа, которая помогает людям найти менторов, получить ценные советы и поддержку, а также поделиться своим опытом с другими.

**В рамках данного проекта:**

* **Была проведена разработка сайта MentorWave на базе системы Django.**
* **Описаны основные этапы разработки, включая подготовку, инициализацию проекта, разработку приложений, интерфейса, функциональности, настройку Django и тестирование.**
* **Даны рекомендации по дальнейшему развитию сайта.**

**Создание сайта MentorWave является значимым шагом в развитии системы поиска менторов.**

Данный сайт станет ценным инструментом для людей, которые хотят развиваться и совершенствоваться, а также для тех, кто хочет поделиться своими знаниями и опытом с другими.

**Дальнейшее развитие сайта может включать:**

* **Добавление новых функций, таких как система оплаты, система отзывов, система рейтингов.**
* **Расширение аудитории сайта за счет привлечения пользователей из разных стран.**
* **Создание мобильного приложения MentorWave.**

**Верю, что сайт MentorWave станет успешным проектом, который поможет множеству людей реализовать свой потенциал.**

# **ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА**

**Нормативные документы:**

Конституция Российской Федерации (с учетом поправок, внесен­ных за­конами Российской Федерации «О поправках к Конституции Россий­ской Федерации» от 30 декабря 2008 г. № 6-ФКЗ и от 30 декабря 2008г. № 7-ФКЗ) // Российская газета № 4831 от 21 января 2009 г.

Федеральный закон Российской Федерации от 7 февраля 2011 № 3-ФЗ «О полиции»// Россий­ская газета от 8 февраля 2011 № 5401.

**Основные источники:**

1. Django documentation. URL: <https://docs.djangoproject.com/en/4.1/>
2. Python documentation. URL: <https://docs.python.org/3/>
3. HTML documentation. URL: <https://html.spec.whatwg.org/>
4. CSS documentation. URL: <https://www.w3.org/Style/CSS/>
5. JavaScript documentation. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
6. SQLite documentation. URL: <https://www.sqlite.org/docs.html>
7. Хабр. URL: <https://habr.com/ru/feed/>
8. Stack Overflow. URL: <https://stackoverflow.com/>
9. Туториал по jinja2. URL: <https://dvmn.org/encyclopedia/modules/jinja2/>
10. Github. URL: [https://github.com](https://github.com/)

Ахо А. Построение и анализ вычислительных алгоритмов. системе / Ахо А., Хопкрофт Дж.. – СПб.: Знание, 2020. – 232 с.

Ахо А. Построение и анализ вычислительных алгоритмов. системе / Ахо А., Хопкрофт Дж., Ульман Дж. – М. : Экзамен, 2020. – 575 с.

**Дополнительные источники**:

Управленческая деятельность: структура, функции, навыки персонала / К. Д. Скрипник [и др.]. – М. : Приор, 2022. – 189 с.

Основы программирования: учеб. пособие / С. И. Самыгин [и др.]; под ред. С. И. Самыгина. – Ростов-на-Дону : Феникс, 2007. – 511 с.

**Словари и энциклопедии**

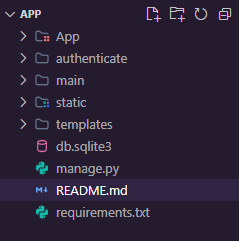
Социальная философия: словарь / под общ. ред. В. Е. Кемерова, Т. Х. Керимова. – М. : Академический Проект, 2011. – 588 с.

**Электронные ресурсы:**

[http://zhurnal.mipt.rssi.ru](http://zhurnal.mipt.rssi.ru/).

# **ПРИЛОЖЕНИЕ**

Приложение 1. Репозиторий проекта.



Приложение 2. Листинг файлов

Листинг 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
* *SECRET\_KEY* = 'django-insecure-\_f&0rq!a*%a*1)i!=(#4\_5db&xf92pc\*au2mkuntdk+n%)u!&f%0'
* *# SECURITY WARNING: don't run with debug turned on in production!*
* *DEBUG* = *True*
* *ALLOWED\_HOSTS* = ['127.0.0.1']
* *# Application definition*
* *INSTALLED\_APPS* = [
* 'django.contrib.admin',
* 'django.contrib.auth',
* 'django.contrib.contenttypes',
* 'django.contrib.sessions',
* 'django.contrib.messages',
* 'django.contrib.staticfiles',
* 'main',
* 'authenticate'
* ]
* *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* = 'App.urls'
* *TEMPLATES* = [
* {
* 'BACKEND': 'django.template.backends.django.DjangoTemplates',
* 'DIRS': [os.path.join(*BASE\_DIR*, 'templates')],
* '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* = 'App.wsgi.application'
* *# Database*
* *# https://docs.djangoproject.com/en/5.0/ref/settings/#databases*
* *DATABASES* = {
* 'default': {
* 'ENGINE': 'django.db.backends.sqlite3',
* 'NAME': *BASE\_DIR* / 'db.sqlite3',
* }
* }
* *# Password validation*
* *# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators*
* *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',
* },
* ]
* *# Internationalization*
* *# https://docs.djangoproject.com/en/5.0/topics/i18n/*
* *LANGUAGE\_CODE* = 'en-us'
* *TIME\_ZONE* = 'UTC'
* *USE\_I18N* = *True*
* *USE\_TZ* = *True*
* *# Static files (CSS, JavaScript, Images)*
* *# https://docs.djangoproject.com/en/5.0/howto/static-files/*
* *STATIC\_URL* = 'static/'
* *STATICFILES\_DIRS* = [
* os.path.join(*BASE\_DIR*, 'static'),
* os.path.join(*BASE\_DIR*, 'static/img'),
* os.path.join(*BASE\_DIR*, 'static/css'),
* os.path.join(*BASE\_DIR*, 'static/js'),
* ]
* *# Default primary key field type*
* *# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field*
* *DEFAULT\_AUTO\_FIELD* = 'django.db.models.BigAutoField'

Листинг base.html

* <!DOCTYPE html>
* <html *lang*="en">
* <head>
* *<!-- Required meta tags -->*
* <meta *charset*="utf-8" />
* <meta *name*="viewport"
* *content*="width=device-width, initial-scale=1, shrink-to-fit=no"
* />
* *<!-- Django-specific template tags -->*
* {% *load* *static* %}
* *<!-- General HTML head elements -->*
* <title>MentorWave</title>
* <link *rel*="stylesheet" *href*="{% *static* 'main.css' %}" />
* *{% comment %} <link rel="icon" href="{% static '' %}" type="image/x-icon" /> {% endcomment %}*
* </head>
* <body>
* <nav *class*="navbar navbar-expand-lg navbar-dark bg-dark">
* <a *class*="finLink" *href*="/">
* <div *class*="ImgNtTitle">
* <span id="titleFin">MentorWave</span>
* </div>
* </a>
* <ul *class*="navbar-nav ml-auto">
* {% *if* user.is\_authenticated %}
* <li *class*="nav-item">
* <a *class*="nav-link" *href*="{% *url* 'logout' %}">Выйти</a>
* </li>
* <li *class*="nav-item">
* <a *class*="nav-link" id="avatar" *href*="{% *url* 'home' %}"
* ><img *src*="{% *static* 'avatar.png' %}" /><span id="mar"
* >Аккаунт</span
* ></a
* >
* </li>
* {% *else* %}
* <li *class*="nav-item">
* <a *class*="nav-link" *href*="{% *url* 'login' %}">Войти как ментор</a>
* </li>
* <li *class*="nav-item">
* <a *class*="nav-link" *href*="{% *url* 'register' %}">Зарегестрироваться как ментор</a>
* </li>
* {% *endif* %}
* </ul>
* </nav>
* <br />
* {% *if* messages %} {% *for* message in messages %}
* <div *class*="alert alert-warning alert-dismissable" *role*="alert">
* {{ message }}
* </div>
* {% *endfor* %} {% *endif* %} {% *block* content %} {% *endblock* %}
* </div>
* <div *style*="height: 500px"></div>
* <footer>
* <div *class*="footer-content">
* <p>*&copy;* 2024 MentorWave. Все права защищены.</p>
* <ul>
* <li><a *href*="#">Про нас</a></li>
* <li><a *href*="#">Связаться с нами</a></li>
* <li><a *href*="#">Политика конфиденциальности</a></li>
* <li><a *href*="#">Условия обслуживания</a></li>
* </ul>
* </div>
* </footer>
* <script *src*="{% *static* 'main.js' %}"></script>
* </body>
* </html>

Листинг models.py

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

*from* django.contrib.auth *import* authenticate, login, logout, update\_session\_auth\_hash

*from* django.contrib.auth.forms *import* *UserCreationForm*, *UserChangeForm*, *PasswordChangeForm*

*from* django.contrib *import* messages

*from* .forms *import* *SignUpForm*, *EditProfileForm*, *CustomPasswordChangeForm*

def home(*request*):

*return* render(*request*, 'authenticate/home.html', {})

def login\_user(*request*):

*if* *request*.method == 'POST':

    username = *request*.*POST*['username']

    password = *request*.*POST*['password']

    user = authenticate(*request*, *username*=username, *password*=password)

*if* user is not *None*:

      login(*request*, user)

      messages.success(*request*, ('Вы вошли!'))

*return* redirect('home')

*else*:

      messages.success(*request*, ('Ошибка при входе в Систему - Пожалуйста, повторите попытку...'))

*return* redirect('login')

*else*:

*return* render(*request*, 'authenticate/login.html', {})

def logout\_user(*request*):

  logout(*request*)

  messages.success(*request*, ('Вы Вышли Из Системы...'))

*return* redirect('home')

def register\_user(*request*):

*if* *request*.method == 'POST':

    form = *SignUpForm*(*request*.*POST*)

*if* form.is\_valid():

      form.save()

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

      password = form.cleaned\_data['password1']

      user = authenticate(*username*=username, *password*=password)

      login(*request*, user)

      messages.success(*request*, ('Ваша учётная запись  должна пройти проверку администратора, прежде чем она будет опубликована.'))

*return* redirect('home')

*else*:

    form = *SignUpForm*()

  context = {'form': form}

*return* render(*request*, 'authenticate/register.html', context)

def edit\_profile(*request*):

*if* *request*.method == 'POST':

    form = *EditProfileForm*(*request*.*POST*, *instance*=*request*.user)

*if* form.is\_valid():

      form.save()

      messages.success(*request*, ('Вы Отредактировали Свой Профиль...'))

*return* redirect('home')

*else*:

    form = *EditProfileForm*(*instance*=*request*.user)

  context = {'form': form}

*return* render(*request*, 'authenticate/edit\_profile.html', context)

def change\_password(*request*):

*if* *request*.method == 'POST':

    form = *CustomPasswordChangeForm*(*data*=*request*.*POST*, *user*=*request*.user)

*if* form.is\_valid():

      form.save()

      update\_session\_auth\_hash(*request*, form.user)

      messages.success(*request*, ('Вы Отредактировали Свой Пароль...'))

*return* redirect('home')

*else*:

    form = *CustomPasswordChangeForm*(*user*=*request*.user)

  context = {'form': form}

*return* render(*request*, 'authenticate/change\_password.html', context)