Post.objects.filter(cat\_\_id=1) – это для поиска

Как сдшлеать взаимосвязанные списки при регистрации

И както быть с районом при рекомендациях

**Сделать слаги!**

**может считать общее количество баллов?**

**Как убрат в пиоске дублирование лейбла?**

**В каталоге сдлеатьм в грпаах убрать дублировние арйона**

ДИПЛОМ

Рекомендательный сервис для выбора активностей

СОДЕРЖАНИЕ

Или SDLC: анализ, планирование, проектирование, разработк

ВВЕДЕНИЕ стр.

1 Теоретические аспекты веб-приложений и рекомендательных систем

1.1. Основные понятия и терминология Web стр.

1.2 Обзор существующих методов рекомендательных систем стр.

1.3. Обзор выбранных технологий разработки веб-приложения стр.

1.3.1. Python стр.

1.3.2. Django стр.

1.3.3. Клиент-серверное взаимодействие стр.

1.3.4. IDE стр.

1.3.5. HTML стр.

1.3.6. CSS стр.

1.3.7. Bootstrap стр.

1.3.8. СУБД MySQL стр.

1.4 Обоснование выбора средств разработки и алгоритма рекомендательной системы стр.

2. Программная реализация

2.1. Техническое задание стр.

2.2. Моделирование базы данных стр.

2.2.1. Проектирование ER-диаграмм стр.

2.2.2. Формирование и заполнение базы данных стр.

2.1 Концептуальное проектирование.................................................................24

2.2 Даталогическое проектирование..................................................................24

2.3 Выводы по разделу........................................................................................27

3 РАЗРАБОТКА ПРИЛОЖЕНИЯ..........................................................................28

3.1 Диаграмма прецедентов................................................................................28

3.1.1 Use-case: «Регистрация».........................................................................28

3.4 Разработка фильтров.....................................................................................34

3.5 Разработка рекомендательной системы.......................................................35

3.6 Выводы по разделу........................................................................................36

4 ОПИСАНИЕ ПРИЛОЖЕНИЯ.............................................................................404.1 Главная страница сайта.................................................................................40

4.2 Страница альбома..........................................................................................41

4.3 Авторизация...................................................................................................42

4.4 Регистрация....................................................................................................42

4.5 Поиск по фильтрам и получение рекомендаций........................................43

4.6 Выводы по разделу........................................................................................44

**ЗАКЛЮЧЕНИЕ........................................................................................................45**

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК....................................................................46**

**ПРИЛОЖЕНИЯ**

БПМН

**ВВЕДЕНИЕ**

В последние годы интернет стал неотъемлемой частью жизни человека: мы ищем информацию, общаемся с близкими, смотрим фильмы и ролики. В России показатель распространенности интернета 88.2% на 2023 год. Интернет становится главным информационным источником. Чтобы не потеряться в море информации появляются рекомендательные системы. Они оптимизируют выдачу информации с учётом накопленного пользовательского опыта, таким образом помогая человеку сделать свой выбор.

Целью данной дипломной работы является разработка веб-приложения, включающего в себя рекомендательную систему.

Разработка приложения разделена на две части:

серверную

реализована с использованием СПИСОК

клиентская часть – с помощью СПИСОК, HTML и CSS.

Проектирование информационной системы было осуществлено с применением ER и UML-диаграмм.

В результате разработки реализовано веб - приложение, функциями которого являются:

- регистрация новых пользователей;

- тестирование пользователей с записью результатов в БД;

- рекомендации для новых пользователей на основе тестирования и локации;

- рекомендации для активных пользователей на основе их прошлого опыта

Ценность работы ПОМОГИ ПРОПИСАТЬ

Приложение обеспечивает структурирование учета массивов данных, что позволяет осуществлять контроль наполняемости групп(???). Осуществление статистики и оптимизацию групп(???). Дополнительно возможна корректировка набора групп(???), что позволяет экономить финансовое обеспечение социальных программ. Подготавливает ИД для аналитической работы.

Потом когда в сервисе будет инфо по баллам, тогда можно будет сделать рекомендации и на схожести пользователей.

**1 ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ВЕБ-ПРИЛОЖЕНИЙ И РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ – ПЕРЕИМЕНОВАТЬ**

**1.1. Основные понятия и терминология Web**

**Общая информация**

Интернет - глобальная система взаимосвязанных компьютерных сетей, которые общаются между собой с помощью интернет-протоколов для связи между сетями и устройствами.

Информация, доступная пользователям интернета, хранится на серверах – выделенных компьютерах с большой памятью. Сервера располагаются в специализированных объектах - дата-центрах. Веб-сервера хранят ресурсы сайта, принимают запросы по протоколу HTTP/HTTPS и отдают веб-страницы и дополнительные данные (рисунки, звуковые файлы, видеофайлы) на устройство пользователя.

Веб-страницы - это документы в формате HTML, размещаемые в интернете. Набор веб-страниц с общей темой и оформлением, а также связанных между собой ссылками и с одинаковым доменным именем, называется веб-сайтом.

Существуют следующие типы веб-сайтов:

* коммерческие

интернет-магазины, лендинги. Они создаются для привлекать клиентов и получать прибыль;

* информационные;
* социальные сети;
* веб-сервисы

они оказывают услугу (электронная почта, хранилище данных) или отвечает на определенный запрос пользователя (курс валют, погода).

**В****еб-приложения**

Первые веб-сайты были статическими, и соответственно обладали небольшой функциональностью. Задача веб-сайтов заключается в предоставлении информации пользователям. Веб-приложения появились как что-то среднее между статическими сайтами и программным обеспечением. По своему функционалу они не уступали программному обеспечению, а доступ к ним осуществлялся с помощью ввода URL в веб-браузере. Можно сказать, что веб-приложение — это программное обеспечение, которое запускается в веб-браузере. На сегодняшний день большинство современных сайтов по своей структуре являются веб-приложениями.

**Преимущества веб-приложений**

*Эффективная разработка*

Сам процесс разработки веб-приложений относительно простой и недорогой. Небольшие команды могут добиться коротких циклов разработки, что делает веб-приложения эффективным и доступным методом создания компьютерных программ. Кроме того, веб-приложение не зависит от того, какая операционная система установлена на компьютере пользователя, оно является кросс-платформенным, а значит не нужно создавать несколько разных вариантов для разных платформ. Также разработчикам не нужно заботиться о совместимости версий, поскольку пользователи единовременно получают доступ и работают с самой последней версией приложения.

*Доступность*

Доступ к веб-приложениям возможен из всех веб-браузеров и с различных личных и рабочих устройств. Единственная версия приложения расположена на сервере, а все пользователи имеют доступ к пользовательскому интерфейсу из любого места в мире. Сотрудники компании, находясь в разных местах, могут получить доступ к общим документам и сервисам через веб-приложение.

*Простота для пользователя*

Для работы с приложением от пользователя нужен компьютер и установленный браузер. Веб-приложения не требуют от пользователей загрузки, что делает их легкодоступными и избавляет от необходимости в обслуживании конечных пользователей и ограничении емкости жесткого диска. Веб-приложения автоматически получают обновления программного обеспечения и безопасности, что означает, что они всегда актуальны и меньше подвержены риску нарушения безопасности.

*Масштабируемость*

Компании, использующие веб-приложения, могут добавлять пользователей по мере необходимости, без дополнительной инфраструктуры и дорогостоящего оборудования. Кроме того, подавляющее большинство данных веб-приложений хранится в облаке, а значит не нужно содержать огромные команды специалистов технической поддержки. В любой момент можно добавить вычислительные мощности.

*Низкие затраты на установку и обслуживание*

Веб-приложения не требуют установки или обновления на отдельных устройствах, что снижает [затраты на](https://appmaster.io/ru/blog/otsutstvie-koda-snizhaet-stoimost-programmnogo-obespecheniia) установку и [обслуживание](https://appmaster.io/ru/blog/otsutstvie-koda-snizhaet-stoimost-programmnogo-obespecheniia).

**Недостатки веб-приложений**

*Интернет-зависимость*

Безусловно у веб-приложений есть минус: их невозможно использовать без доступа в интернет.

*Проблема совместимости*

Для корректной работы приложений необходимо их тестирование в разных веб-браузерах, что увеличивает затраты.

*Проблема безопасности*

Веб-приложения уязвимы к следующим угрозам:

* угрозы конфиденциальности – несанкционированный доступ;
* угрозы целостности – несанкционированное искажение или уничтожение данных;
* угрозы доступности – ограничение или блокирование доступа.

*Ограниченная функциональность*

Веб-приложения не имеют полного доступа к функциональности устройства пользователя, например, к камере или к геолокации.

**Принцип работы веб-приложений**

Веб-приложение представляет из себя не только изображения и текст в браузере, оно включает в себя данные, которые запрашивают или которые создают его пользователи. Таким образом, веб-приложения имеют клиент-серверную архитектуру.

Клиентская часть (frontend) представляет собой визуальную часть приложения, пользовательский интерфейс. Все, что пользователь видит в приложении, все, с чем он может взаимодействовать (кнопки, прокрутка страницы, ввод текста) – все это клиентская часть.

Серверную часть (backend) пользователь не может увидеть и доступа к ней не имеет. На сервере происходит хранение и обработка данных по определенным правилам, то есть серверная часть отвечает за логику работы приложения.

Клиент обращается с запросами к серверу через браузер. Это могут быть запросы на получение или отправку данных. Сервер получает запрос, обрабатывает его, при необходимости сохраняет данные в базу или достает данные из базы и отправляет ответ клиенту. Клиент оборачивает данные в удобный для пользователя дружественный интерфейс. На сервере могут запускаться фоновые задачи.

**Типы веб-приложений**

Первые веб-приложения были больше похожи на сайты, на любое вмешательство пользователя они перезагружали страницу. С усложнением приложений появилась потребность и в новых инструментах для их реализации. Таким образом, помимо многостраничных (MPA) веб-приложений появились одностраничные (SPA) и прогрессивные (PWA) приложения.

*MPA (Multi Page Application)*

В таких приложениях необходима полная перезагрузка страницы после изменений в данных. К таким приложениям пользователи привыкли больше всего, и они легко seo-оптимизируются. **MPA** требуют больше времени и финансовых вложений при разработке, но зато они лучше других подходят для работы с большим объемом и разнообразием данных. Вначале MPA были полностью статическими: работа приложения заключалась в отображении статического содержимого файлов, хранящихся на сервере с помощью технологий HTML и CSS. Статические до сих пор могут использоваться на информационных сайтах. Потом появилась динамическая генерация HTML. В результате переходов между страницами, отправки форм сервер генерирует новую HTML-страницу, которая и отобразится у пользователя. Такие приложения используют следующие технологии backend: PHP, Python и Ruby.

*SPA (Single page application)*

Такие приложения (например, почтовые сервисы) работают на технологии AJAX (Asynchronous JavaScript and XML), которая позволяет менять содержание HTML-страницы в ответ на действия пользователя без ее перезагрузки. Такие приложения быстрые в работе и в разработке, но создают нагрузку на браузер и могут иметь пробелы в безопасности. Отрисовка начальной страницы может осуществляться в браузере (client side rendering) или на сервере (server side rendering). Для разработки SPA используются фреймворки JavaScript, такие как React, Vue.js и Angular.

*PWA (Progressive Web Application)*

**PWA основаны на современных веб-технологиях, они максимально приближены к обычным десктопным приложениям. PWA больше используются на мобильных устройствах, занимают они мало места в памяти и быстро загружаются. Они работают автономно, могут отправлять пользователю push-уведомления, а также имеют доступ к таким средствам самого устройства, как геолокация или камера. Но они поддерживаются не всеми платформами.**

**1.2 Обзор существующих методов рекомендательных систем**

***Рекомендательная система***

Рекомендательная система (РС) нужна для того, чтобы предложить пользователю продукты, о которых он ранее не знал, но которые могут оказаться полезными или интересными для него. В любой РС используется сбор данных. При явном сборе пользователи отвечают на вопросы анкеты, при неявном протоколируются его действия. (например, что пользователь просматривает в интернет-магазинах). Данные сохраняются в БД, чтобы затем использоваться для вычисленения рекомендаций для пользователя.

Существует четыре основных типа систем выдачи рекомендаций:

#### *Фильтрация по популярности (****popularity-based recommender systems****).*

#### Это самая простая система, основанная на рейтингах. Но такая система не учитывает предпочтения отдельного пользователя.

#### *Фильтрация на основании контента (****content-based filtering****)*

Такая система будет рекомендовать пользователю продукты, исходя из его прошлых предпочтений. Но такая система никогда не предложит пользователю попробовать что-то новое.

*Коллаборативная фильтрация (collaborative filtering)*

Такая система будет рекомендовать пользователю продукты в категориях, близких к уже выбранным данным пользователем и пользователями с похожим поведением. Суть алгоритма - нахождение ближайших соседей. Близость двух пользователей или предметов определяется метриками схожести.

*Фильтрация на основе знаний*

Алгоритмически это самая сложная система. Для получения рекомендаций используются полученные каким-либо образом знания о предпочтениях пользователя. Но такая система должна каким-то образом получить от пользователя информацию по всем запрашиваемым параметрам.

*Гибридные системы*

У каждой системы есть свои плюсы и минусы. Поэтому чаще используются гибридные системы, которые объединяют алгоритмы в сбалансированный набор. Гибридные системы наиболее точные, количество используемых рекомендательных алгоритмов ограничено только разработчиком системы и бюджетом.

Для данного приложения разработанного для данной дипломной работы используется гибридная система. По результатам тестирования пользователей осуществляется два типа фильтрации:

* на основе знаний;

тестирование пользователей для разделения их на три группы по типам активности «Для души», «Для тела» и «Для разума»

* по популярности.

На основании посещения групп пользователями выбирается топ-5 популярных видов занятий он-лайн и топ-5 офф-лайн. Затем пользователям предлагаются группы по их типу активности, которые принадлежат топ-5.

**1.3. Обзор выбранных технологий разработки веб-приложения**

Разработка веб-приложения состоит из следующих частей:

* frondend

пользовательский интерфейс, то есть та часть, с которой взаимодействует пользователь;

* backend

внутренняя часть приложения, которая обрабатывает запросы пользователя;

* система управления базами данных (СУБД)

комплекс программного обеспечения для создания и работы с базами данных.

Следовательно, перед разработкой веб-приложения нужно определиться с:

* языком программирования и фреймворком для backend;
* языком программирования и фреймворком для frondend;
* как будет осуществляться клиент-серверное взаимодействие;
* СУБД

**1.3.1. Python**

Перед началом разработки приложения необходимо выбрать язык программирования. Язык программирования должен обеспечивать возможность разработки всех функций приложения, заложенных в ТЗ (техническом задании на разработку). Изменение языка программирования в процессе разработки повлечет за собой дополнительные трудозатраты, а применение дополнительных языков, используя их для частичной разработки потребует выполнить интерфейсы между частями приложения, увеличивая человеческий ресурс на разработку, дополнительную отладку, и тестирование.

Ведущими языками для разработки серверной части являются: JavaScript (безусловный лидер), Python, Ruby, PHP, Java, C# и Golang.

Для данного веб-приложения предполагается дополнительное написание отдельной модели для РС. РС можно написать на разных языках: Python, Matlab, Java, C++ и других, но чаще всего для этой задачи используется Python. На сегодняшний день Python является надежным вариантом для приложений, использующих машинное обучение.

Язык Python был задуман как потомок языка программирования ABC еще в 1980- ых годах. Официальный год выхода – 1991 год. Гвидо ван Россум назвал язык в честь популярного телешоу «Monty Python’s Flying Circus», поскольку автор был его поклонником. Python является высокоуровневым языком программирования (например, низкоуровневый – это язык машинного кода). Все в Python сделано для повышения производительности: автоматический сборщик мусора, читаемость кода, компактность, отсутствие «{ }», автоматическое выделение памяти в устройстве. Python платформенно независимый язык, его можно адаптировать практически к любой операционной системе. Узкое место языка – динамическая типизация. Тип данных переменной в нем определяется исходя из присвоенного ей значения.

Для разработки приложения в рамках данной дипломной работы в качестве backend языка программирования был выбран Python, так как:

* обладает большой коллекцией разнообразных библиотек, фреймворков и плагинов, обеспечивающих высокий уровень удобства разработки;
* разработка веб-приложения проводилась на двух рабочих станциях с различными ОС, а код на Python можно использовать без изменений на различных ОС.
* Python является одним из лучших языков программирования для веб-скрапинга (сбор информации с сайтов, её анализ и преобразование в нужный формат.

**1.3.2. Django**

Фреймворк - это набор готовых библиотек и инструментов. Для написания приложений на Python существует большое количество фреймворков, самые известные из них: Flask. FastAPI, CherryPy, Django, Pyramid.

Для написания веб-приложения «Еще не бабушка» был выбран Django:

* монолитная артхитектура;

Такая архитектура позволяет разбивать разработку на части, сразу получая обратную связь. Она подходит для веб-приложений с небольшим функционалом и небольшой командой разработки (в данном случае один человек).

* наличие сильного сообщества;

Сообщество Django дружественное для новичков. Ответы на вопросы по работе фреймворка легко найти в интернете, ему посвящены тысячи статей и обучающих видео. Официальная документация Django является одной из лучших.

* работа из коробки;

Django можно представить, как конструктор, где есть необходимые типовые блоки для работы приложения, например, такие как авторизация, сессии.

* свой [веб-сервер](https://en.wikipedia.org/wiki/Web_server) для разработки и тестирования.

Django существует почти 20 лет. Он был создан веб-программистами из газеты Lawrence-Journal World Эдрианом Головати и Саймоном Виллисоном во время разработки новостного сайта. Назвали фреймфорк в честь джазового гитариста Джанго Рейнхардта, который виртуозно брал аккорды тремя пальцами.

Преимущества Django:

* большой функционал

В Django есть необходимое все для создания приложения в рамках дипломной работы, он способен решить все задачи при разработке. Помимо ORM, системы аутентификации пользователей, панели администратора и форм, для работы с API Django предоставляет REST Framework, для управления контентом - Django CMS. Соответственно сокращаются трудозатраты на поиск дополнительных библиотек.

* SEO-ориентированность

Django позволяет динамически генерировать Sitemaps, использование которого делает приложение более заметным в рейтинге поисковых систем. Для моделей предусмотрено Slugfield – это короткие этикетки, которые позволяют создавать более чистые, читаемые и удобные для поисковых систем URL-адреса.

* универсальность

Django создает структуру для проекта, которая разделяет бизнес-логику и визуальную части. Благодаря стандартной структуре проще искать готовые решения. Проект может состоять из одного или нескольких приложений. У Django есть готовые модули, которые легко добавить в проект.

* безопасность

В Django встроена защита от CSRF-атак (использование учетных данных другого пользователя). Она работает в каждом POST-запросе. Из коробки Django защищен от SQL-инъекций (выполнение произвольного запроса SQL в базе данных), так как автоматически обрабатывает пользовательский ввод и делает его безопасным для использования в SQL-запросах.

Использование шаблонов Django защищает от XSS-атак (внедрение клиентских скриптов в браузеры других пользователей). Помимо этого, в Django можно настроить защиту от Clickjacking («захват клика» - позволяет вредоносной странице кликнуть по сайту-жертве от имени пользователя). Но в Django нет надежной проверки загружаемого пользователем контента.

* административная панель

Django поставляется с панелью администратора, она генерируется автоматически при создании приложения. С ее помощью можно работать с базой данных через веб-интерфейс. Панель можно настроить под свои нужды как технически, так и визуально.

* расширяемость

Этот гибкий фреймворк позволяет создавать собственные компоненты, работать с собственными моделями, расширяя функциональность ORM. Для Django существует большое количество плагинов, и есть возможность создавать собственные.

Недостатки Django ORM:

Он основан на паттерне Active Record, который прост в использования. Данный паттерн нарушает принцип единственной ответственности (SRP), из-за чего его часто называют антипаттерном. В Active Record один класс управляет и данными, и поведением. К сожалению, эта простота уместна только в простых приложениях.

Django является большим фреймворком, поэтому он развивается и совершенствуется относительно медленно.

Также Django ORM уступает в производительности, например, SQLAlchemy, так как у него есть тенденция генерировать сложные запросы, когда в этом нет необходимости.

Однако, данными недостатками можно пренебречь в силу простоты задач, поставленных в рамках разрабатываемого дипломного проекта. Позитивный эффект от преимуществ Django полностью нивелирует все вышеизложенные недостатки. Он является достаточным фреймворком для решения поставленных задач.

**1.3.3. Клиент-серверное взаимодействие**

Все взаимодействия между фронтендом и бэкендом обычно происходят через API. Но для написания данного приложения выбран фреймворк Django. Мнения о нем у разработчиков разделяются, многие не считают его полноценным full-stack фреймворком. Однако, Django предлагает для frontend удобный и понятный Django Templates, который позволяет создавать динамические страницы с использованием циклов, условий и куда можно передавать переменные. Также можно использовать шаблонизатор Jinja2. Шаблоны имеют расширение .html, в них содержится HTML-код со специальными вставками, в которых определяются динамические элементы.

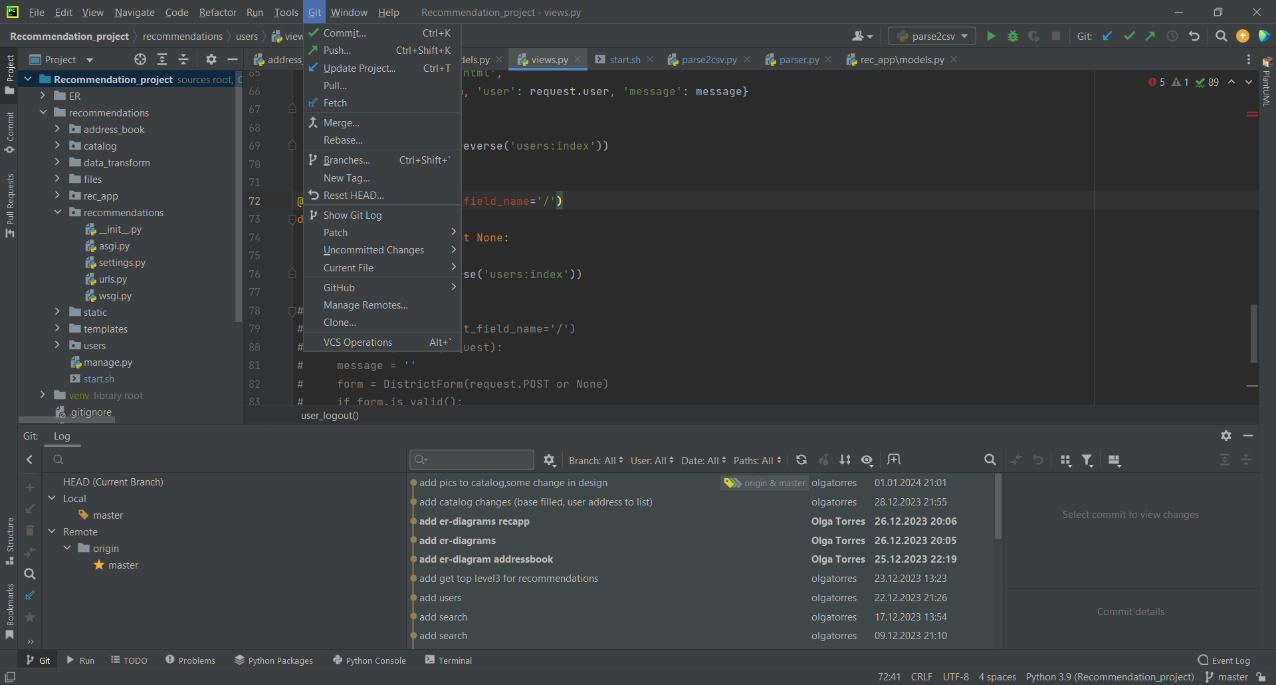
Для full-stack разработки безусловно помогает и *Django ORM.* Основная цель ORM – обеспечение связи между базой данных и моделями в приложении. Django предоставляет объектно-ориентированный интерфейс для взаимодействия с базой данных на Python вместо SQL-запросов. Это повышает скорость и удобство разработки, так как упрощает управление базой данных и обеспечивает согласованность между внутренними данными и их представлением в пользовательском интерфейсе.

Преимущества Django ORM:

* таблицы в базе данных на основе моделей создаются автоматически;
* работает с основными реляционными СУБД – PostgreSQL, MySQL, SQLite;
* можно легко поменять СУБД;
* простой синтаксис;
* удобный доступ к связанным объектам.

В случае совместной работы с frontend-разработчиком над веб-приложением «Еще не бабушка», можно было использовать Django REST (Representational State Transfer) API. Он бы преобразовывал проект во внутренний API, и уже этот API взаимодействовал с фреймворком JavaScript, выбранным для интерфейса.

**1.3.4. IDE (Integrated development environment)**

Использование среды IDE обосновано тем, что в данной среде предусмотрена подсветка корректности записи синтаксиса, также существует встроенный компилятор и возможность тестирования кода. IDE обеспечивает простую установку необходимых для работы библиотек, управление конфигурацией и итеративность разработки.

Интерфейс IDE PyCharm.

Для языка программирования Python наиболее популярными IDE являются PyCharm и Visual Studio. PyCharm - это среда разработки на Python от компании JetBrains, существующая для основных операционных систем: Windows, [Linux](https://blog.skillfactory.ru/glossary/linux/) и macOS. Visual Studio - это проект Microsoft, поэтому данная IDE больше подходит для работы в Windows и С#. Visual Studio более неповоротливый и потребляет больше системных ресурсов.

Для разработки приложения в рамках дипломной работы выбран IDE PyCharm, так эта среда разработки больше подходит для небольших проектов.

Преимущества PyCharm:

* удобный просмотр документации в IDE и браузере;
* встроенное Unit тестирование;
* работа в консоли;
* поддержка систем контроля версий;
* коллекция плагинов;
* рефакторинг;
* удобный умный редактор кода, включая шаблоны для типовых задач

Что немаловажно, HTML и CSS, которые понадобятся для frontend, хорошо поддерживаются в редакторе PyCharm.

**1.3.5. HTML**

HTML - язык гипертекстовой разметки (Hypertext Markup Language) лежит в основе любого сайта, в нем информация о структуре сайта. Благодаря HTML можно разметить любой текст, преобразовав его в гипертекст, в котором будет указано, где заголовки и где обычный текст, откуда загружать изображения и видео.

HTML не является языком программирования. Он не умеет обрабатывать данные или выполнять вычисления. Задача гипертекста структурировать контент по определенным правилам. При этом документ содержит только текст, а все остальные объекты встраиваются в момент отображения браузером с помощью специальных тэгов и хранятся отдельно в специальной папке.

HTML позволяет:

* разметить текст с помощью иерархической структуры (заголовки, абзацы, списки)
* размещать аудио-, видео- и фотоматериалы
* встраивать гиперссылки и меню
* создавать таблицы и формы

В веб-приложении «Еще не бабушка» HTML используется в шаблонах страниц для их верстки. Для того, чтобы добавить дизайн и стиль приложению используются CSS и Bootstrap.

**1.3.6. CSS**

CSS (Cascading Style Sheets) тоже не является языком программирования, это язык таблицы стилей. С его помощью можно применять различные стили форматирования элементам HTML. CSS появился благодаря желанию пользователей видеть более дружественный и красивый интерфейс.

Таблицы стилей могут быть прописаны как в самом файле HTML, так и в отдельном файле с расширением «.css». Файл представляет из себя список селекторов (названий объектов) и стилей, применяемых к ним. За счет разделения оформления от содержания время на разработку приложения уменьшается. Ведь для внесения изменений в дизайн приложения, нужно лишь изменить параметры форматирования в CSS-файле.

Помимо этого, CSS позволяет сократить время на загрузку страницы, так как стили загружаются один раз для всего приложения, а при переходе со страницы на страницу визуальное оформление берется из кэша.

**1.3.7. Bootstrap**

Изначально Bootstrap создавался как внутренний продукт компании Twitter. Bootstrap – это бесплатный фреймворк с открытым исходным кодом, разработанный на базе HTML, CSS и JavaScript. Он применяется во frontend разработке, и на сегодня является самым популярным среди подобных. Bootstrap по сути работает как конструктор, при подключении к нему становятся доступными готовые компоненты и классы. К компонентам относятся готовые блоки с часто используемыми элементами страницы: кнопки, формы, навигация, выпадающие списки, всплывающие панели. Классы Bootstrap основаны на CSS. Они используются для создания макета страницы и для стилизации контента.

Преимущества Bootstrap:

* экономия времени на разработку за счет готовых решений;
* современный и единый стиль;
* кроссплатформенность;
* адаптивность к устройствам разных размеров;
* возможность создать собственную сборку.

Для веб-приложения, созданного в рамках данной дипломной работы, был выбран Bootstrap за его относительную простоту и возможность быстро сделать дизайн в едином стиле.

**1.3.8. СУБД MySQL**

База данных (БД) в данном веб-приложении необходима для хранения данных о пользователях, а также технических данных для работы веб-приложения.

Первостепенно необходимо определится с типом базы данных. В данном случае ИД предоставлены в табличном виде. Для хранения табличной структуры подходят реляционные базы данных. Реляционные БД бывают с хранением данных по строкам и по колонкам. Колоночные используются для аналитики, а строковые подходят для транзакций. Соответственно, для данного проекта подходит реляционная строковая БД.

Следующим этапом нужно определить систему управления базами данных, которая будет использоваться для разработки веб-приложения. Самые популярные в веб-разработке: PostgreSQL и MySQL. Для веб-приложения «Еще не бабушка» была выбрана MySQL, так как в приоритете была скорость работы, а не расширенные функции как в PostgreSQL (например, вложенные подзапросы).

Преимущества СУБД MySQL:

* безопасность;

В MySQL управление доступом для пользователей осуществляется через Access Control Lists, что позволяет наделять каждую учетную запись сервера правами на определенные действия с данными даже на уровне отдельных таблиц. Также в MySQL встроена система хэширования паролей. В платной версии Enterprise Edition реализована защита от кибератак, плагины аутентификации и шифрование с помощью SSL (Secure Sockets Layer).

* производительность;

У MySQL хорошая производительность и быстродействие благодаря оптимизированным алгоритмам выполнения запросов. Изначально эта СУБД была создана для обработки огромных массивов информации.

* масштабируемость

MySQL позволяет расширять БД по необходимости. Для вертикального масштабирования (покупка более мощного сервера) MySQL ограничена своей архитектурой. Для горизонтального (разделение на несколько серверов) существует несколько вариантов решений. Таким образом MySQL может обрабатывать большое количество одновременных запросов.

* открытый исходный код;

У компании Oracle есть платная СУБД, а MySQL они представляют как проект с открытым исходным кодом, что позволяет бесплатно использовать базовую версию, а также адаптировать исходный код под свои задачи. При этом Oracle предлагает клиентам платные продвинутые версии с дополнительными возможностями.

* phpMyAdmin;

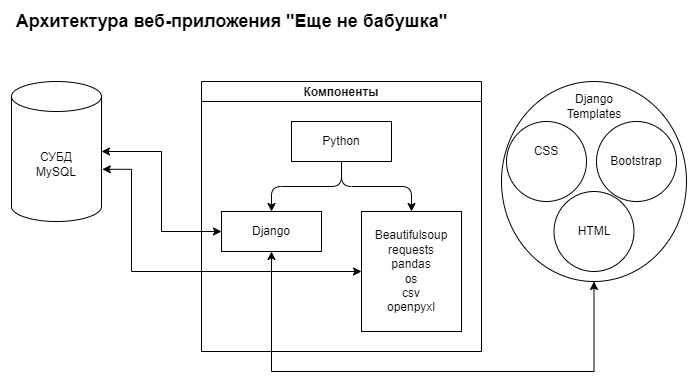
Веб-приложение phpMyAdmin позволяет удобно и легко взаимодействовать с БД через интерфейс браузера.

Недостатки СУБД MySQL:

MySQL не подходит для работы с большими данными (big data), она теряет скорость ответа. Также MySQL не соответствует стандарту SQL: она не поддерживает ряд функций (например, поиск в этой СУБД реализован без учета регистра) некоторые функции и имеет расширения, не относящиеся к стандартному SQL. Это может стать проблемой при переходе проекта на другую СУБД.

**1.4. Выводы по главе 1**

В данной главе были рассмотрены выбранные средства разработки для веб-приложения «Еще не бабушка», а также проведено обоснование данного выбора.

****

**2. Программная реализация веб-приложения**

**2.1. Техническое задание**

*Описание проекта*

В Москве работает проект «Еще не бабушка» для граждан старшего возраста (55+ для женщин и 60+ для мужчин), который предоставляет им возможность принимать участие в широком диапазоне активностей.

Активности проходят офлайн и онлайн формате. Для удобства работы руководство проекта планирует внедрение рекомендательной механики.

*Постановка задачи*

Создание рекомендательной системы на основании имеющихся данных о пользователях и их активности в проекте.

Предоставлены данные о зарегистрированных пользователях, на их основании должна проводится идентификация пользователя в системе. Такому пользователю система должна предлагать активности на основании истории его посещений. Для нового пользователя система должна предлагать активности на основании поведения зарегистрированных пользователей.

Предусмотрены следующие пути пользователя:

а) поиск группы через поисковую строку;

б) поиск через каталог групп;

в) прохождение тестирования, которое определяет предпочтения пользователя. Приоритет при подборе активности - офлайн занятия в районе проживания пользователя.

Основная задача сервиса - максимальное сокращение пути пользователя до кнопки “Записаться” (в группу).

*Описание данных*

Исходные данные (далее – ИД) содержат информацию о пользователях проекта «Еще не бабушка» и их активностях в 2023 году. ИД представлены 3 файлами в формате csv и одним файлом в формате Excel. В каждом файле находится одна таблица.

В ИД содержится информация:

groups.csv - “Группы”

users.csv - “Пользователи”

attend.csv - “Посещаемость”

dict.xlsx - “Справочник направлений”

Наименования столбцов перечислены в Приложении 1.

**2.2. Моделирование базы данных**

Разработка базы данных начинается с ее проектирования. Для этого проект «Еще не бабушка» разделен на несколько логических приложений, для каждого приложения выделены все сущности и связи, а также связи между сущностями разных приложений.

Пользователи сайта могут искать группы с различными занятиями по разделам каталога или с помощью поиска, авторизованные пользователи могут пройти тестирование и получить рекомендации в виде списка групп. Поскольку в техническом задании обозначена первостепенность выдачи групп по месту жительства пользователя, то необходимо систематизировать информацию по адресам пользователей и места проведения занятий. Таким образом, можно выделить следующие модули (приложения) внутри проекта:

* пользователи
* каталог групп
* адресная книга
* тестирование

**2.2.1. Проектировка ER-диаграмм**

Так как для данного веб-приложения выбрана реляционная БД MySQL, то для проектирования объектов используется ER (Entity Relationship)-диаграмма (ERD), которая отображает отношения между сущностями, хранящимися в базе данных. ERD создаются на основе сущностей, их атрибутов и отношений. ERD визуализирует структуру БД, помогая разработчику.

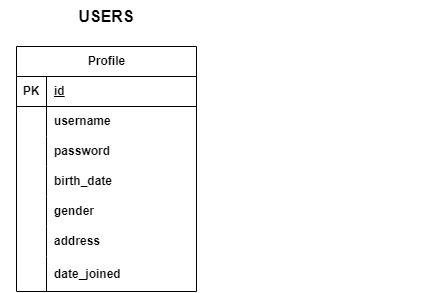
*Users*

В Django есть своя встроенная система аутентификации, но в модели пользователя не хватает атрибутов, которые присутствуют в ИД:

* дата рождения;
* пол;
* адрес.

Поэтому модель пользователя «Profile» создана как расширение от AbstractUser.

ЕСТЬ ВАРИАНТ ПРОПИСАТЬ ОТДЕЛЬНО ПРОФАЙЛ И ЮЗЕР ДЛЯ ХРАНЕНИЯ ЛОГИН ПАРОЛЬ

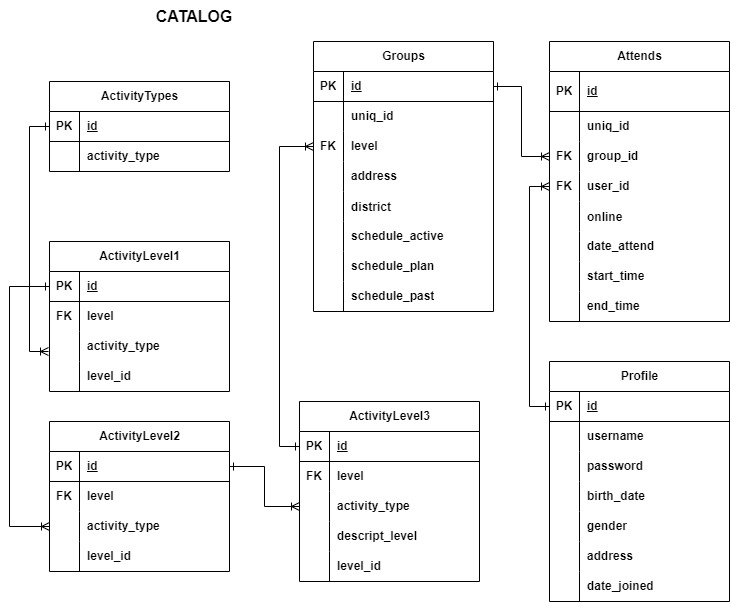


ERD Users

*Catalog*

Для создания каталога занятий сделаем нормализацию отношений в файле «Dict», содержащим информацию о типах и уровнях активности. Название колонок изменены для удобства работы с ними. В файле «Groups» находятся данные о группах с занятиями, группы могут быть в трех состояниях: прошедшие, актуальные и будущие.

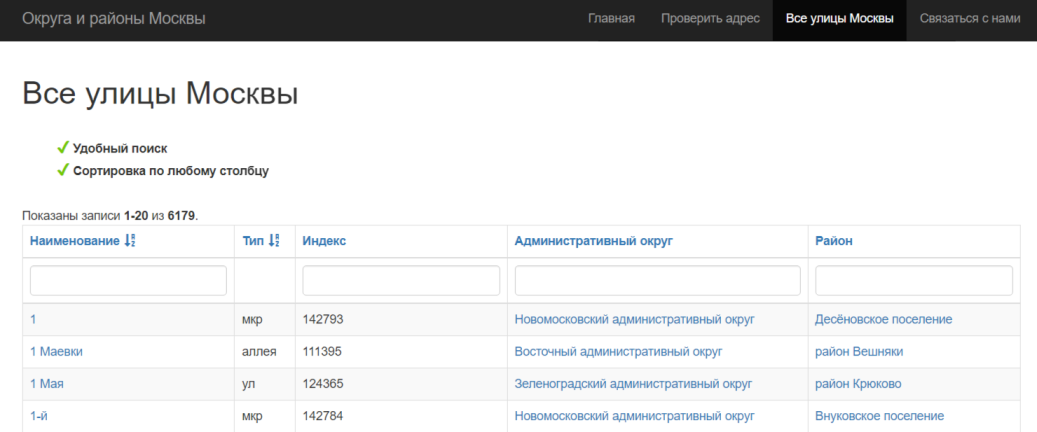
Для работы проведены изменения в колонках с адресом и районом группы, так как в списках в них содержались дубли. Сделать нормализацию первой формы и разделить таким образом списки из нескольких адресов) невозможно, так как файл с посещениями групп «Attends» соединяется с «Groups» только по идентификатору группы из ИД.



ERD Catalog

*AddressBook*

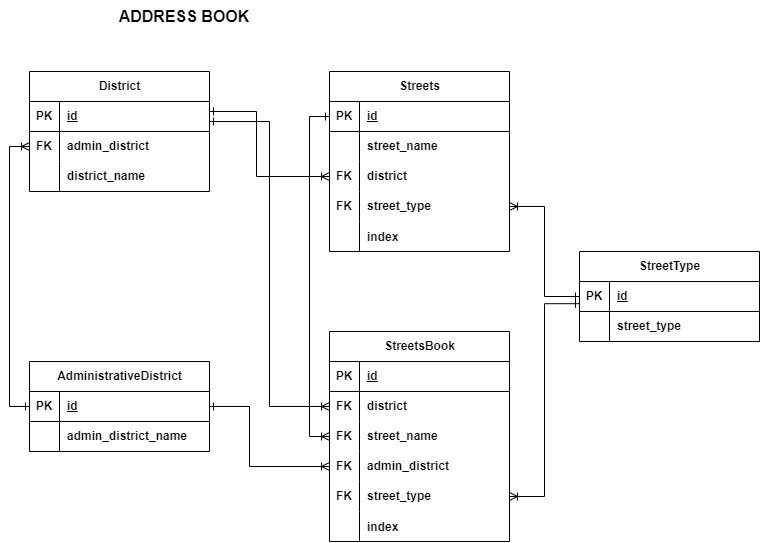
Одна из поставленных задач состоит в подборе занятий в районе пользователя. В ИД в таблице Users представлены адреса без районов, в таблице Groups есть привязка адресов групп к районам и округам. Для определения округа проживания пользователя спарсим сайт по улицам Москвы: <http://округа-районы.москва/все_улицы_москвы>.



Скрин с сайта «округа-районы.москва»

Для веб-скрапинга были использованы библиотеки: BeautifulSoup и Pandas. С помощью инструментов разработчика на сайте получено название таблицы <table class="table table-striped table-bordered”>, и из нее в цикле выгружена информация из каждой строки. Обработка данных не использовалась. Заголовки созданы вручную, так как таблица одна. Результат загружен в таблицу "moscow\_streets".

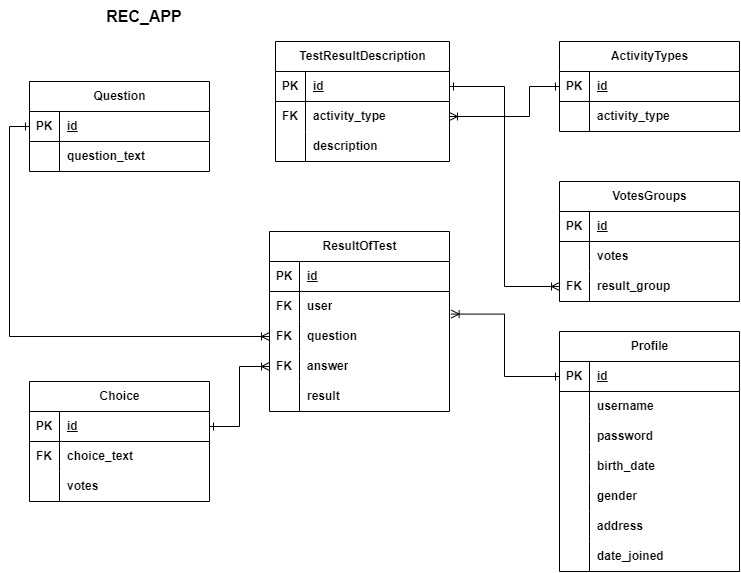
Данные этой таблицы также используются при регистрации пользователей для исключения ошибок с вводом адреса. Поэтому помимо основных сущностей, создана сущность, связывающая другие сущности.

 ERD Address\_Book

*Rec\_app*

Для построения РС от заказчика не были предоставлены данные. Так как одно из условий было тестирование для пользователей, то выделены следующие сущности:

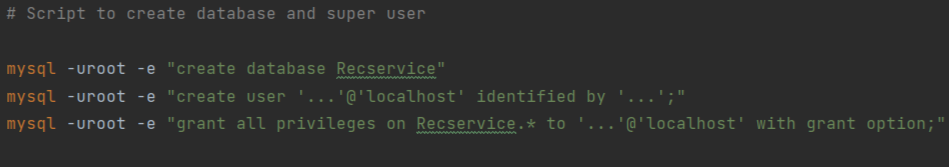
* вопрос;
* ответ (не связанный с вопросом);
* описание результатов тестирования (связанные с типом активности);
* группы по баллам тестирования;
* результаты тестирования.

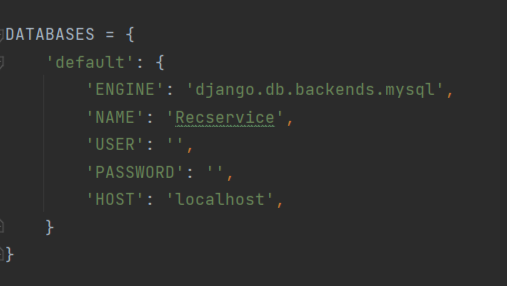


ERD Rec\_app

**2.2.2. Формирование и заполнение БД**

Для работы с MySQL необходимо установить mysqlclient - поточно-совместимый интерфейс к серверу базы данных MySQL, который предоставляет API баз данных Python. Затем создается база данных и пользователь, который наделяется максимальными правами доступа к ней. В данном проекте это реализовано следующим скриптом:

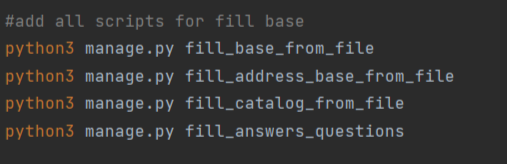


Формирование таблиц в БД в Django происходит через создание моделей. В настройках проекта settings.py указывается СУБД и информация для подключения к ней. Затем в каждом приложении проекта в файле models.py описываются сущности и отношения между ними. Для формирования таблиц в БД выполняется команда makemigrations, а затем migrate. Данные обо всех изменениях структуры БД записываются в папку migrations каждого приложения.

Следующим этапом выполняется скрипт, запускающий файл с парсингом адресов Москвы:

C:\Users\вяаы\PycharmProjects\Recommendation_project\parsing_script.png

Заполнение БД осуществляется путем создания собственных команд в каждом приложении. Команды запускаются скриптом в определенном порядке.



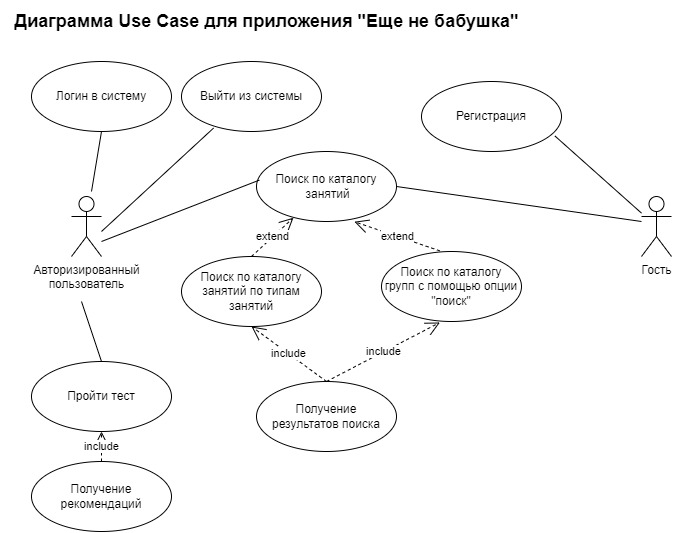
2.3 Выводы по разделу В данном разделе была разработана база данных для системы. Сначала было выполнено концептуальное проектирование, в ходе которого сначала были выделены сущности и связи в системе, а потом на основе них была построена ER-диаграмма. После этого был осуществлен переход к даталогическому проектированию – построена схема реляционной базы данных, приведено описание всех таблиц и атрибутов, находящихся в ней.

**2.3. Описание функционала веб-приложения**

Технические требования к веб-приложению необходимо прописать перед началом разработки. Они могут быть функциональными (как должно работать приложение) и нефункциональными (ОС, оборудование). Для их описания существуют разные подходы.

Для разработчика функциональные требования к проекту удобнее и проще описать через сценарии использования (Use Case), чем, например, в BPMN 2.0, который больше подходит для описания бизнес-процессов в более широком смысле.

Use Case - это диаграмма, которая описывает действующие лица (акторы) и прецеденты. К диаграмме прилагается текстовое описание конкретных действий (главная и альтернативная последовательности), которые может осуществлять пользователь при работе с приложением.



Use-case: «Регистрация»

В приложении «Еще не бабушка» есть фунционал только для зарегистрированных пользвателей: это прохождение теста и получение персональных рекомендаций. Для регистрации пользователю необходимо указать следующие данные о себе:

* логин;
* пароль;
* дата рождения;
* пол;
* округ;
* район;
* адрес (улица, проспект).

Главная последовательность: пользователь осуществляет ввод данных в форму, подтвержает ввод кнопкой «Подтвердите». Приложение переходит на главную страницу с расширенным фунционалом.

Альтернативные последовательности:

1. Пользователь вводит уже зарегистрированный логина. Система выдает сообщение о том, что логин или почта заняты, и предлагает повторить ввод.

Возраст меньше пенсионного 59 и больше 120 лет

Имя содержит пробелы

Макс пароля 20

Макс имя 12

2. Ввод некорректного адреса почты. Система выдает сообщение о том, что адрес почты введен ошибочно, и предлагает повторить ввод.

Альтернативная последовательность 3: Ввод короткого пароля. Система выдает сообщение о том, что пароль менее четырех символов, и предлагает повторить ввод.

Альтернативная последовательность 4: Неверное подтверждение пароля. Система выдает сообщение о том, что почта или пароль неверно подтверждены, и предлагает повторить ввод.

Use-case: «Ввод логина и пароля» Если у посетителя уже есть профиль на сайте, то он может авторизоваться на нем, указав свои логин и пароль.

Главная последовательность: Пользователь вводит логин и пароль в форме на странице авторизации и подтверждает ввод. Система переводит пользователя на страницу с информацией о профиле. Альтернативная последовательность: Неверный ввод логина или пароля. Система выдает сообщение о том, что логин или пароль введены неверно, и предлагает повторить ввод.

Use-case: «Просмотр каталога занятий

На главной странице сайта представлен каталог всех альбомов, существующих в базе данных. Пользователь может просматривать их, прокручивая колесико мышки или используя полосу прокрутки.

Use-case: «Фильтрация альбомов» Если у посетителя сайта нет желания просматривать все существующие альбомы, он может выставить необходимые фильтры и просмотреть только те альбомы, которые удовлетворяют им. Главная последовательность: На главной странице сайта в специальной форме пользователь по желанию выбирает новинки и один или несколько жанров и годов выпуска и подтверждает ввод. После этого страница перезагружается и выводятся альбомы, соответствующие выставленным фильтрам. Альтернативная последовательность: В базе данных отсутствуют альбомы, удовлетворяющие выставленным фильтрам. Система выводит сообщение о том, что искомые альбомы не существуют. 30

Use-case: «Выбор альбома» При просмотре альбомов посетитель сайта может выбрать заинтересовавший его альбом, кликнув на него левой кнопкой мыши, и тем самым перейти на посвященную ему страницу.

Use-case: «Получение информации об альбоме» На странице об альбоме может присутствовать информация о его исполнителе. Главная последовательность: Пользователь выбирает альбом и автоматически переходит на страницу альбома, содержащую более подробную информацию. Альтернативная последовательность: В базе данных отсутствует информация. Посетитель сайта видит только обложку альбома, его название, исполнителя и год выпуска.

Use-case: «Прослушивание альбома» Страница, посвященная альбому, может содержать плейлист с входящими в него песнями. Главная последовательность: Пользователь выбирает альбом и автоматически переходит на страницу альбома, содержащую аудиоплеер. Далее он кликает на какую-либо песню левой кнопкой мыши и начинается воспроизведение.

Use-case: «Оценка альбома» Авторизованным посетителям сайта предоставляется возможность оценивать альбомы на соответствующих им страницах. Главная последовательность: Пользователь выставляет оценку альбому от одного до пяти посредством выбора количества звездочек, расположенных внизу страницы об альбоме. Альтернативная последовательность: Пользователь не авторизован. Система выводит сообщение о невозможности голосования и предлагает посетителю пройти авторизацию.

Use-case: «Получение рекомендаций» Авторизованный на сайте пользователь может получить каталог альбомов, которые по предположению рекомендательной системы сайта могут быть ему интересны.

Главная последовательность: Пользователь по желанию выставляет необходимые ему фильтры и нажимает кнопку «Рекомендации» на главной странице сайта. Страница перезагружается и выводятся рекомендуемые ему альбомы.

Альтернативная последовательность

1.Пользователь не авторизован. Система выводит сообщение о невозможности получения рекомендаций и предлагает пройти авторизацию.

2: Пользователь еще не оценил ни один альбом. Система выводит сообщение о невозможности получения рекомендаций и предлагает оценить альбомы.

3: Пользователь оценил все существующие альбомы. Система выводит сообщение о невозможности получения рекомендаций и предлагает подождать пополнения библиотеки альбомов. \

4: Отсутствуют пользователи, с которыми можно сравнить текущего (то есть текущий пользователь и остальные пользователи выставили оценки разным альбомам или остальные пользователи не выставили оценки вообще). Система выводит сообщение о невозможности получения рекомендаций и предлагает повторить попытку позже.

5: В базе данных отсутствуют альбомы, удовлетворяющие выставленным фильтрам. Система выводит сообщение о том, что искомые альбомы не существую

**Архитект шаблон для приложения и какой паттерн из СПА и прочих**

*Архитектурный шаблон в Django*

Архитектурный шаблон - это основа для веб-приложения. Шаблон содержит набор решений о том, какие роли отводятся модулям приложения, по каким правилам эти модули будут общаться между собой и с внешними системами.

Архитектурный шаблон MVC (Model-View-Controller) был представлен в конце 1970-ых годов норвежцем Т. Реенскаугом, на сегодня он является одним из самых распространенных.

MVC делит приложение на следующие компоненты:

View (представление) представляет собой интерфейс, который пользователь видит и с которым взаимодействует (нажимает кнопку, например). Представление получает данные из модели через контроллер. Функция представления — отображение данных, которые хранятся в моделях.

Model (модель) представляет собой данные и бизнес-логику. Она отвечает за обработку, хранение и управление данными.

Controller (контроллер) осуществляет связь между моделями и представлениями.

это уровень бизнес-логики и представления, который выполняет такие действия, как выборка данных, и принимает решения, например, как представить модель, какие части ее отображать и т. д. Контроллер получает пользовательский ввод из представления, обрабатывает его и обновляет модель. После обновления модели она уведомляет контроллер, который затем обновляет представление новыми данными. Основная ответственность контроллера — управлять потоком приложений и поддерживать синхронизацию модели и представления. Функции в этом компоненте обычно запускаются событием (пользователь нажимает кнопку, состояние Модели изменилось и т.д.) И выполняют соответствующие действия (ищут элементы в базе данных и возвращают результат для просмотра или отправляют в View сообщение, уведомляющее пользователя об изменении некоторых состояний Модели и т.д.). В веб-системах этот уровень обрабатывает http-запрос и возвращает http-ответ (мы рассмотрим это позже).

**Примечание:**

Терминология, которую мы здесь используем, на самом деле отличается от соглашения об именовании Django. В документации Django уровень обработки http-запроса и возвращаемого ответа называется **View**, а html-страницы обычно называются **Template**. Несмотря на то, что этот курс основан на Django, мы решили не следовать их правилам по двум основным причинам:

1. Это сбивает с толку. В официальных документах Django то, что они называют *View*, представляет собой своего рода комбинацию серверной части и внешнего интерфейса. Но мы хотим дать **Controller** и **View** четкие и несвязанные определения. **Контроллер** - это серверная часть, **представление** - это интерфейс.

Чтобы было понятнее, давайте воспользуемся простым правилом перебора для различения контроллера и представления, в нашей терминологии: весь код, написанный на Python, принадлежит **Controller**, в то время как коды, написанные на html, javascript, css, принадлежат **View**

**URLs:** Хотя можно обрабатывать запросы с каждого URL-адреса с помощью одной функции, гораздо удобнее писать отдельную функцию для обработки каждого ресурса. URL-маршрутизатор используется для перенаправления HTTP-запросов в соответствующее представление на основе URL-адреса запроса. Кроме того, URL-маршрутизатор может извлекать данные из URL-адреса в соответствии с заданным шаблоном и передавать их в соответствующую функцию отображения (view) в виде аргументов.

**View:** это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы. Функция view имеет доступ к данным, необходимым для удовлетворения запросов, и делегирует ответы в шаблоны через модели.

**Models:** Модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для управления (добавления, изменения, удаления) и выполнения запросов в базу данных.

**Templates:** это текстовый файл, определяющий структуру или разметку страницы (например HTML-страницы), с полями для подстановки, которые используются для вывода актуального содержимого. *View* может динамически создавать HTML-страницы, используя HTML-шаблоны и заполняя их данными из модели (*model).* Шаблон может быть использован для определения структуры файлов любых типов, не обязательно HTML.

## [Как выглядит код Django?](https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction#как_выглядит_код_django)

На традиционном информационном веб-сайте веб-приложение ожидает HTTP-запросы от веб-браузера (или другого клиента). Когда запрос получен, приложение разрабатывает то, что необходимо на основе URL-адреса и, возможно, данных в POST или GET запросах. В зависимости от того, что требуется, далее он может читать или записывать информацию из базы данных или выполнять другие задачи, необходимые для удовлетворения запроса. Затем приложение вернёт ответ веб-браузеру, часто динамически создавая HTML-страницу для отображения в браузере, вставляя полученные данные в HTML-шаблон.

### Краткое введение в http-запрос и http-ответ

Как мы упоминали ранее, базовой моделью того, как работает веб-система, является:

1. Клиент отправляет запрос на сервер и ожидает ответа сервера
2. Сервер обрабатывает запрос и выполняет некоторые действия. Это та часть, которая больше всего волнует контроллер.
3. Сервер возвращает результат клиенту

Прежде чем рассматривать контроллер, давайте сначала получим базовое представление о том, что представляют собой запросы и ответы, чтобы при реализации контроллеров вам было более понятно, как идентифицировать и считывать данные из запросов и как возвращать правильные ответы. Основным стандартным протоколом здесь является HTTP

**Http-запрос** содержит URL-адрес, в котором указывается адрес ресурса; заголовок, который может содержать файл cookie, а также другую информацию; и, вероятно, параметр запроса или закодированный объект (полезная нагрузка данных).

Существует множество методов запроса, включая, но не ограничиваясь ими:

* ПОЛУЧИТЬ
* ГЛАВА
* Публикация
* ПОСТАВИТЬ
* Удалить
* …

Более подробную информацию об этих методах также можно найти [здесь](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods). Среди них, вероятно, больше всего используются **GET** и **POST**.

**Http-ответ** содержит:

* Заголовок, который содержит такую информацию, как формат возвращаемых данных и т. Д
* Код состояния. Это то, что вам действительно нужно знать. Каждый код имеет свое явное значение и не должен использоваться неправильно. Вот несколько примеров (из [официального документа](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html))
  + 200: Успешный ответ
  + 302: Найдено. Но запрошенный ресурс временно находится под другим URI.
  + 400: Неверный запрос. Запрос не может быть понят сервером из-за неправильного синтаксиса. Клиент НЕ ДОЛЖЕН повторять запрос без изменений.
  + 401: неавторизованный. Запрос требует аутентификации пользователя.
  + 404: Не найдено. Сервер не нашел ничего, соответствующего URI запроса. Вы, вероятно, хорошо знакомы с этим :)
  + …

При реализации веб-сервера очень важно отправлять правильный ответ клиентам в различных ситуациях. Поэтому имейте это в виду и при необходимости обращайтесь к официальной документации.

* Полезная нагрузка данных. Полезной нагрузкой данных может быть текст, html, двоичные данные и т.д.

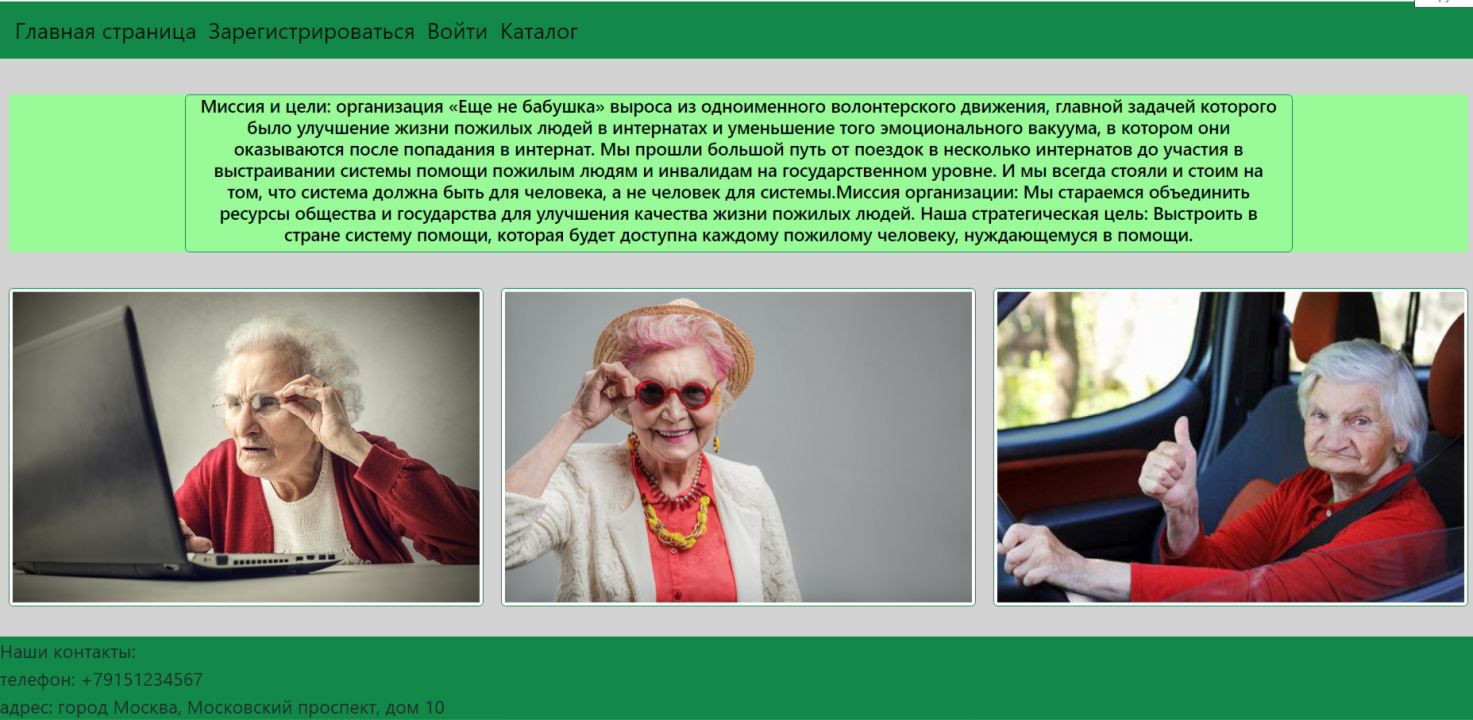
**4. Описание разработанного веб-приложения «Еще не бабушка»**

**4.1. Описание пользовательского интерфейса**

Пользовательский интерфейс (UI) это способ передачи информации между пользователем и веб-приложением. UI включает в себя элементы, которые позволяют пользователям взаимодействовать с веб-приложением. К ним относятся,например, навигация, кнопки, выпадающие списки и уведомления.

Веб-приложение «Еще не бабушка» создавалось для пожилых людей, и это необходимо было учесть в интерфейсе. Для оформления навигации, кнопок и подвала был выбран зеленый цвет, так как в психологии он считается цветом жизни и роста. Для всего приложения использовался единый читабельный шрифт ТАКОЙТО, размер шрифта определялся в корреляции со значимостью теста. Оформление веб-приложения минималистично.

Так выглядит главная страница веб-приложения, разработанного для данной дипломной работы - «Еще не бабушка» для незарегистрированного пользователя.



Навигация выполнена в виде меню, расположенного вверху каждой страницы приложения. Для зарегистрированного пользователя меню расширено и выглядит так:

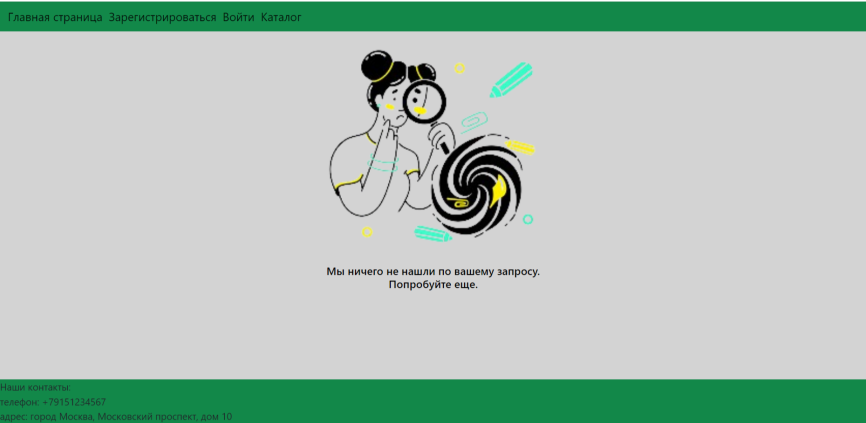
ФОТО МЕНЮ ЗАРЕГАННОГО ЮЗЕРА

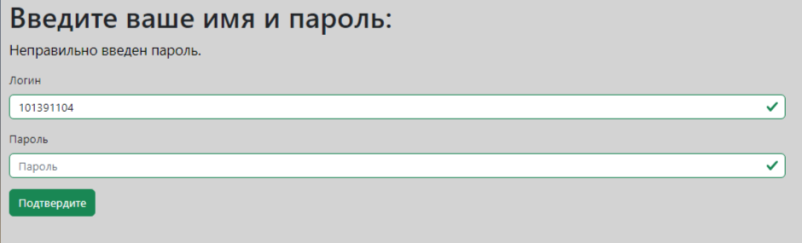
Внизу каждой страницы расположен подвал с контактными данными

, поиск, ссылки, хлебные крошки позволяют перемещаться пользователю по различным разделам, легко ориентироваться в структуре.

Обратная связь c пользователем реализована следующим образом:

* сообщения об ошибочном вводе данных
* уведомления о неудачном поиске с помощью отдельной страницы
* уведомления о невозможности записаться в группу



****

**Обратная связь с пользователем**

. После выполнения какого либо действия пользователи должны получать обратную связь. Это могут быть уведомления, анимации, лоадеры, подсказки, которые сообщают пользователям о выполнении какого-либо действия, его выполнении или об ошибке.

* **Предоставление информации**. Ваши интерфейсы должны эффективно предоставлять информацию пользователям. Это может быть обычный текст, видео, различные графики, прикрепленные документы.
* **Интерактивные элементы**. О них мы выше упоминали, это кнопки, выпадающие меню, формы, прогрессбары. Они позволяют пользователям взаимодействовать с вашим приложением.
* **Доступность**. Доступность интерфейса для пользователей с ограниченными возможностями. Например, вы можете сделать возможность изменять размер шрифта для слабозрячих или делать текст более контрастным. Или предоставить возможность навигировать с помощью одной лишь клавиатуры.
* **Согласованность**. Единообразные элементы дизайна, прописанные стандарты взаимодействия в вашем приложении помогают пользователям понять как им пользоваться.
* **Пользовательский опыт**. Удобство использования и положительный опыт — вот две вещи, которые должны сформироваться у пользователя после использования интерфейсов.

Пользовательский интерфейс разработанного Web-приложения состоит из ====

4.2. Описание основного функционала Web-приложения

**Описание реализации каждого приложения: как устроено, какие урлы и вью.**

Каталог пользователь может смотреть без регистрации.

Для тестирования работы приложения необходимо зарегистироватся.

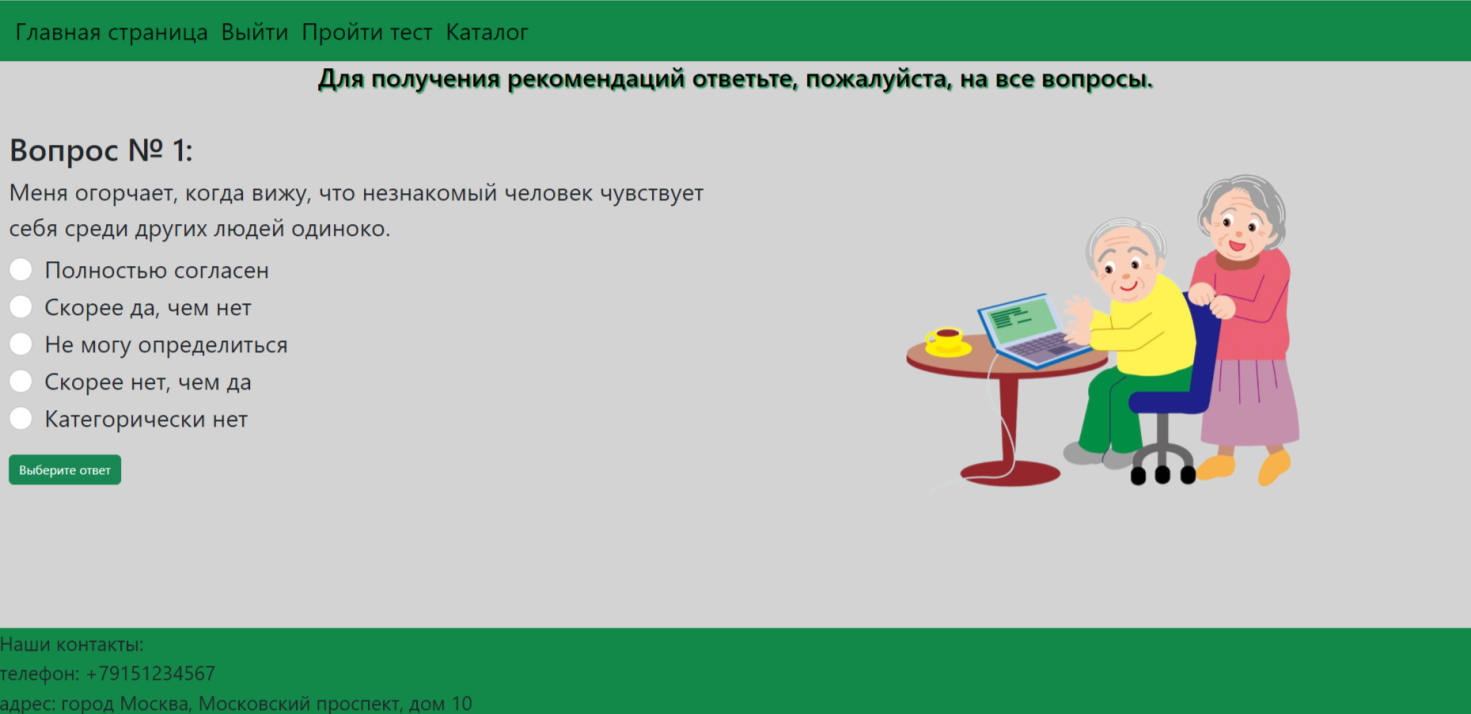
Рисунок 3 — демонстрация тестовой структуры.

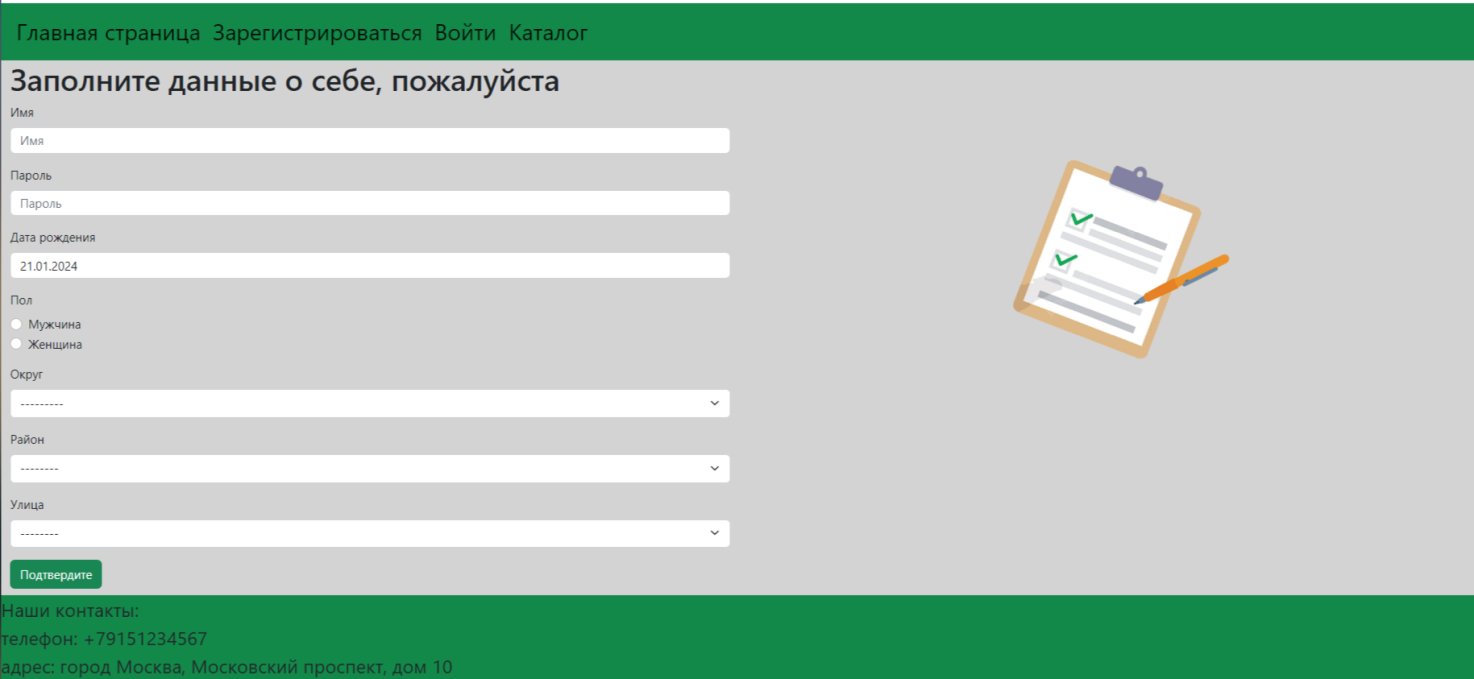
Кнопка (+Folder) в правой части приложения позволяет добавить папку в структуру, представленную таблицей. При нажатии она вызывает поле ввода названия папки и дополнительную кнопку для подтверждения. Это показано на рисунке 4.

Рисунок 4 — создание папки.

Добавление файла в необходимую происходит по нажатию кнопки

(+File) . Предварительно надо выбрать файл в левой таблице, просто кликнув по нему.



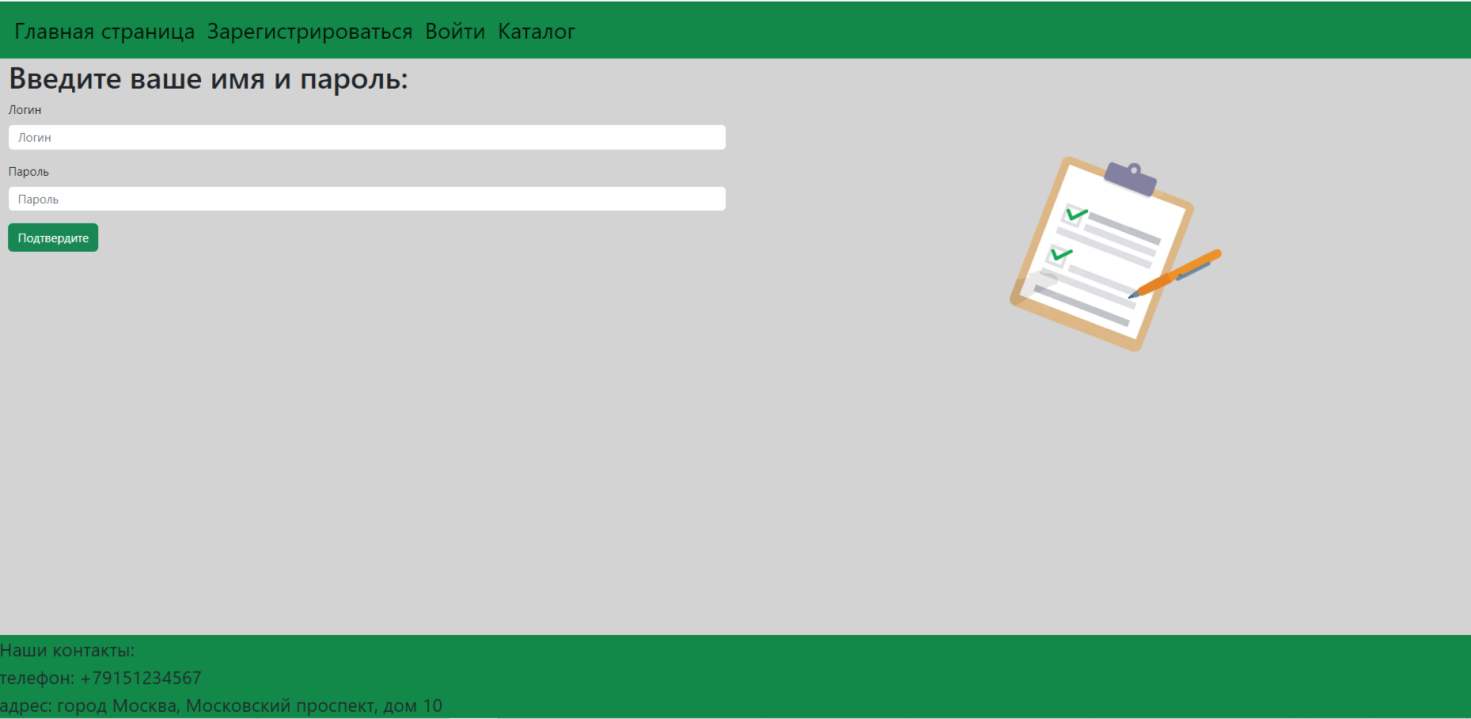
Добавление файлов и папок реализовано с помощью рекурсивных функций и соответствуют стандартам React, то есть, они не изменяют состояние структуры напрямую, а предварительно создается копия состояния (Приложение А). 

Нахождение файла по клику в левой таблице также было реализовано посредством подхвата события нажатия и вызовом рекурсивной функции поиска в дереве.

Далее загрузим сохраним структуру на сервере в виде тестового XML-файла. Нажмем на кнопку Upload и на сервере получим соедующий XML-файл, представленный на рисунке 5.

Рисунок 5 — полученная в результате XML структура.

то в файле settings.py нужно проверить определение параметра AUTH\_USER\_MODEL, которое должно быть таким:



**Цель работы**

Разработка технического проекта для возможного решения рекомендательного сервиса для проекта «Еще не бабушка» на основе предоставленных данных (далее по тексту ИД).

В практической части необходимо реализовать сервис на Python.

Список параметров фильтрации групп в сервисе записи:

●направления занятий (1,2,3 уровней)

●формат занятий (очно / онлайн)

●район занятий (для очных занятий)

●расписание (дни недели / время)

*Принцип работы Django*

Первым этапом запрос от пользователя попадает в роутер (URL dispatcher), который решает какую функцию для обработки запроса надо вызвать. Решение принимается на основе списка правил, состоящих из регулярного выражения и названия функции: если такой-то урл, то вот такая функция.  
  
Функция, которая вызывается роутером, называется вью (view). Внутри может содержаться любая бизнес-логика, но чаще всего это одно из двух: либо из базы берутся данные, подготавливаются и возвращаются на фронт; либо пришел запрос с данными из какой-то формы, эти данные проверяются и сохраняются в базу.  
  
Данные приложения хранятся в базе данных (БД). Чаще всего используются реляционные БД. Это когда есть таблицы с заранее заданными колонками и эти таблицы связаны между собой через одну из колонок.  
  
Данные в БД можно создавать, читать, изменять и удалять. Иногда для обозначения этих действий можно встретить аббревиатуру CRUD (Create Read Update Delete). Для запроса к данным в БД используется специальный язык SQL (structured query language).  
  
В Джанго для работы с БД используются модели (model). Они позволяют описывать таблицы и делать запросы на привычном разработчику питоне, что гораздо удобнее. За это удобство приходится платить: такие запросы медленнее и ограничены в возможностях по сравнению с использованием чистого SQL.  
  
Полученные из БД данные подготавливаются во вью к отправке на фронт. Они могут быть подставлены в шаблон (template) и отправлены в виде HTML-файла. Но в случае одностраничного приложения это происходит всего один раз, когда генерируется HTML-страница, на который подключаются все JS-скрипты. В остальных случаях данные сериализуются и отправляются в JSON-формате.

Когда запрос от браузера доходит до сервера, он не сразу попадает в Джанго. Сначала его обрабатывает веб-сервер Nginx. Если запрашивается статический файл (например, картинка), то сам Nginx его отправляет в ответ клиенту. Если запрос не к статике, то Nginx должен проксировать (передать) его в Джанго.  
  
К сожалению, он этого не умеет. Поэтому используется еще одна программа-прослойка — сервер приложений. Например для приложений на питоне, это могут быть uWSGI или Gunicorn. И вот уже они передают запрос в Джанго.  
  
После того как Джанго обработал запрос, он возвращает ответ c HTML-страницей или данными, и код ответа. Если все хорошо, то код ответа — 200; если страница не найдена, то — 404; если произошла ошибка и сервер не смог обработать запрос, то — 500. Это самые часто встречающиеся коды.

## Дополнительные инструменты

Ajax расшифровывается как Asynchronous Javascript And XML (Асинхронные Javascript и XML) и технологией в строгом смысле слова не является. Если в стандартном web-приложении обработкой всей информации занимается сервер, тогда как браузер отвечает только за взаимодействие с пользователем, передачу запросов и вывод поступившего HTML, то в Ajax-приложении между пользователем и сервером появляется еще один посредник - движок Ajax. Он определяет, какие запросы можно обработать "на месте", а за какими необходимо обращаться на сервер.

Язык программирования JavaScript разработан фирмой Netscape для создания интерактивных HTML-документов. Это объектно-ориентированный язык разработки встраиваемых приложений, выполняющихся как на стороне клиента, так и на стороне сервера. Синтаксис языка очень похож на синтаксис Java – поэтому его называют – Java-подобным.

Основные области применения JavaScript делятся на следующие категории: · динамическое создание документа с помощью сценария; · оперативная проверка достоверности заполняемых пользователем полей форм HTML до передачи их на сервер; · создание динамических HTML-страниц совместно с каскадными таблицами стилей и объектной моделью документа; · взаимодействие с пользователем при решении "локальных" задач, решаемых приложением JavaScript, встроенном в HTML-страницу

Основа разработки — язык программирования JavaScript, являющийся на сегодняшний день одним из самых распространненых языков программирования и самым распространненым в веб-разработке. Этот язык дает понятный синтаксис, универсальную структуру хранения данных и возможность писать асинхронный код, который повысит скорость работы приложения в разы.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта цель достигнута, задачи выполнены, реализовано веб-приложение.

Были проанализированы принципы работы систем, структура и функциональность, а также взаимодействие компонентов. Изучены теоретические основы разработки web– приложений. Рассмотрены новейшие web–технологии и этапы создания web– приложения, а также некоторые вопросы по оптимизации и выбору средств разработки. Разработана структура работы web– приложений и технически реализован данный алгоритм работы.

Для разработки данного модуля были изучены различное программное обеспечение и среды разработки. В частности HTML, CSS, JavaScript, React.JS, Node.js, Express.

Приложение было максимально адаптировано для различных разрешений экрана.

*Web* – приложение удовлетворяет всем запросам и требованиям, которые я поставил при постановке целей. Все плагины и модули которые я использовал в процессе разработки сайта были доработаны с учетом спецификации *web*–сайта и внедрены в его структуру. Также стоит отметить большие возможности для дальнейшей модернизации и улучшения сайта в зависимости от дальнейшего позиционирования.

В данном дипломном проекте разработано Web-приложение ---- в частности решены следующие задачи: проведен обзор существующих инструментов и информационных технологий для создания Web-приложений; определена структура и дизайн Web-приложения, реализован основной функционал клиентской части Web-приложения. Разработанное Web-приложение содержит =====. Обладает гибкой системой поиска объектов, которая позволяет ===

В Web-приложении также реализована возможность ===

СПИСОК ЛИТЕРАТУРЫ

<https://timeweb.cloud/blog/mysql-preimushchestva-i-nedostatki>

https://ru.hexlet.io/blog/posts/chto-takoe-html-i-zachem-on-nuzhen

https://habr.com/ru/articles/473042/

<https://ru.hexlet.io/blog/posts/pochemu-django-luchshiy-freymvork-dlya-razrabotki-saytov>

<https://aws.amazon.com/ru/amplify/>

https://doka.guide/js/web-app-types/

<http://lib.broadcasting.ru/articles2/Oborandteh/rekomendatelnyy-servis>

**https://ninavasilok.ru/kak-rabotat-s-frontendom-v-django**

https://www.sravni.ru/kursy/info/rekomendatelnye-sistemy/

https://www.web-canape.ru/business/statistika-interneta-i-socsetej-na-2023-god-cifry-i-trendy-v-mire-i-v-rossii/

https://znanierussia.ru/articles/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F\_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0

https://habr.com/ru/articles/176549/

https://ru.wikipedia.org/wiki/Bootstrap\_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)

<http://web.spt42.ru/index.php/chto-takoe-bootstrap>

<https://excitinggames.ru/preimushhestva-django-vsyo-chto-nuzhno-znat-dlya-veb-razrabotki-na-python/>

https://nasmorku-net.ru/rol-orm-v-django-i-eyo-znacenie-dlya-razrabotki-veb-prilozenii

https://smartiqa.ru/courses/web/lesson-1

<https://pythobyte.com/seo-for-django-5-methods-to-improve-seo-625-daefb33a/>

https://skysmart.ru/articles/programming/chto-takoe-css

https://windowsreport.com/visual-studio-vs-pycharm/

<https://maksyutin.ru/Архитектура%20ПО/Архитектурные%20шаблоны/Шаблоны%20архитектуры%20/>

<https://cyberleninka.ru/article/n/vybor-instrumentalnyh-sredstv-dlya-razrabotki-obrazovatelnogo-veb-prilozheniya/viewer>

<https://www.microsoft.com/ru-ru/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling>

https://otus.ru/journal/python-vse-o-yazyke-programmirovaniya/

https://vegibit.com/what-is-pythons-django-framework-and-how-does-it-facilitate-full-stack-development/

1. Дронов, Владимир JavaScript. / Владимир Дронов. - М.: БХВ-Петербург, 2014. - 819 c.

2. Дунаев, Вадим HTML, скрипты и стили / Вадим Дунаев. - М.: БХВ-Петербург, 2015. - 816 c.

3. Никсон, Робин Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript / Робин Никсон. - М.: Питер, 2013. - 496 c.

4. Роббинс, Дженнифер HTML5, CSS3 и JavaScript. Исчерпывающее руководство (+ DVD-ROM) / Дженнифер Роббинс. - М.: Эксмо, 2014. - 528 c.

5. Симпсон, Кайл ES6 и не только / Кайл Симпсон. - М.: Питер, 2017. - 336 c.

. Заяц А.М. Основы WEB технологий. Разработка WEB-приложений современными инстру-

ментальными средствами: учебно-методическое пособие. Санкт-Петербург: СПбГЛТУ, 2021. 116 с.

2. Гвоздева Т.В., Баллод Б.А. Проектирование информационных систем: технология автомати-

зированного проектирования. Лабораторный практикум: учебное пособие. Санкт-Петербург: Лань, 2020.

156 с.

3. Садыков А.М. Методы разработки веб-приложений: учебно-методическое пособие. Ивано-

во: ИГЭУ, 2019. 72 с.

4. Войт Н.Н. Разработка программного обеспечения оценки действий обучаемых. Ульяновск:

2019. С. 151-159.

5. Вагин Д.В., Петров Р.В. Современные технологии разработки веб-приложений: учебное по-

собие. Новосибирск: НГТУ, 2019. 52 с.

6. Сычев А.В. Теория и практика разработки современных клиентских веб-приложений: учеб-

ное пособие. Москва: ИНТУИТ, 2016. 483

**Приложение 1. Содержание таблиц в ИД.**

Attend:

уникальный номер занятия;

уникальный номер группы;

уникальный номер участника;

направление 2;

направление 3;

онлайн/офлайн;

дата занятия;

время начала занятия;

время окончания занятия.

Groups:

уникальный номер;

направление 1;

направление 2;

направление 3;

адрес площадки;

округ площадки;

район площадки;

расписание в активных периодах;

расписание в закрытых периодах;

расписание в плановом периоде.

Dict:

Разметка: Для ума/ Для души / Для тела;

id\_level1;

level1;

id\_level2;

level2;

id\_level3;

leve3;

d\_level1;

d\_level2;

d\_level3.

Users!

уникальный номер;

дата создание личного дела;

пол;

дата рождения;

адрес проживания.