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

**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

## Проектирование схемы данных блога

Начнем с определения исходных моделей для нашего блога. Модели отображают информацию о данных, с которыми вы работаете. Они содержат поля и поведение ваших данных. Обычно одна модель представляет одну таблицу в базе данных. Джанго создаст таблицу для каждой модели, определенной в файле Models.py. При создании модели Джанго предлагает API для простого запроса к базе данных.

## готовка проекта и настройка среды разработки

Python - дитя Unix. Поэтому, выбор unix-подобной операционной системы (Linux или macOS) будет совершенно логичным решением. Тем более, что переходить на компьютер с другой OC совсем не обязательно: можно просто установить на своём компьютере с любой операционной системой виртуальную машину и установить Linux уже на неё. А также установить PyCharm - удобную и бесплатную среду разработки.

О том, как это сделать подробно рассказано и показано в этих видео:

* [Как установить VirtualBox на Windows](https://youtu.be/DB66HfnV1q0)
* [Установка Ubuntu 20.04 на VirualBox](https://youtu.be/Xo-E6GMl6ck)
* [Установка PyCharm и создание виртуального окружения](https://youtu.be/nfD-A0S8oKg)

После успешной установки всего вышеперечисленного можно переходить к созданию проекта. Для этого надо создать каталог с названием проекта **django-dropdown-admin**, войти в этот каталог и запустить процесс создания виртуального окружения и тут же его запустить:

$ virtualenv venv - p python3

$ source venv/bin/activate

Виртуальное окружение запущено, о чём говорит имя папки в скобках, появившееся в начале командной строки: **(venv)**.

Далее устанавливаем Django:

(venv) $ pip install django

Создаём проект (обратите внимание на точку в конце!):

(venv) $ django-admin startproject main .

И приложение **dropdown**:

(venv) $ django-admin startapp dropdown

В результате должно получиться что-то вроде этого:

.

├── dropdown

│   ├── admin.py

│   ├── apps.py

│   ├── \_\_init\_\_.py

│   ├── migrations

│   │   └── \_\_init\_\_.py

│   ├── models.py

│   ├── tests.py

│   └── views.py

├── main

│   ├── asgi.py

│   ├── \_\_init\_\_.py

│   ├── settings.py

│   ├── urls.py

│   └── wsgi.py

├── manage.py

└── venv

С этого момента все дальнейшие действия удобнее будет выполнять в IDE PyCharm. (см видео в конце этой статьи)

## 1. Изменяем файл settings.py

Прежде всего внесём изменения в настройки - файл **main/settings.py**:

***1.) Добавим вновь созданное приложение в INSTALLED\_APPS***

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

# === my apps =======

'dropdown',

]

СОДЕРЖАНИЕ

Основная часть:

Архитектура

Парсинг

Загрузка данных в базу

Или SDLC: анализ, планирование, проектирование, разработк

ВВЕДЕНИЕ стр.

1 Теоретические аспекты веб-приложений и рекомендательных систем

1.1. Основные понятия и терминология Web

1.2 Обзор существующих методов рекомендательных систем стр.

1.3 Обзор средств разработки веб-приложений стр.

1.3.1. Django

1.3.2. HTML

1.3.3. CSS

1.3.4. Bootstrap стр.

1.3.5. IDE

1.4 Обоснование выбора средств разработки и алгоритма рекомендательной системы стр.

2. Программная реализация

2.1. Техническое задание стр.

2.2. Разработка 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

3.1.2 Use-case: «Ввод логина и пароля».........................................................29

3.1.3 Use-case: «Просмотр альбомов».3.1.4 Use-case: «Фильтрация альбомов»........Выбор альбома»..3.1.6 Use-case: «Получение информации об альбоме».................................30

3.1.7 Use-case: «Прослушивание альбома».3.1.8 Use-case: «Оценка альбома».

3.1.9 Use-case: «Получение рекомендаций»..................................................30

3.3 Настройка шаблона JXTC Hardcore Shop....................................................31

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

- названия столбцов в начальных таблицах

**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 разделилась на две части: серверную и клиентскую. Серверная часть 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

Потом когда в сервисе будет инфо по баллаём, тогда это можно будет учитывать в модели.

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

**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-сайты:

- коммерческие

Это 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-приложение**

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 клиентских и серверных веб-приложений появились одностраничные (SPA), многостраничные (MPA) и прогрессивные (PWA) приложения.

*Клиентские веб-приложения*

Таким приложениям (например, информационный сайт) не нужна обработка на стороне сервера, они используют браузер пользователя в качестве оболочки. Данные пользователя сохраняются только на время сессии. Их работа заключается в отображении статического содержимого с помощью технологий HTML, CSS и JavaScript.

*Серверные веб-приложения*

Такие приложения работают на удаленных компьютерах. Для их работы практически ну нужно вмешательство пользователя (например, push-уведомления). 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, Ruby и Node.js.

*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. Для их разработки используются фреймворки JavaScript, такие как React, Vue.js и Angular.

*MPA (Multi Page Application)*

(**MPA**, многостраничные web-приложения) применяются для построения сложных систем. В данном случае любые изменения в данных приводят к полной перезагрузке страницы. Когда имеется большой массив данных и контента, разнообразие представляемых сведений, **MPA**подходят лучше всего.  
  
Несмотря на то, что они требуют больших объемов ресурсов для реализации и существенно дороже, другие виды web-приложений их заменить не могут. Однако тенденции показывают, что общая доля **MPA**постепенно снижается.  
Стандартный пример такого приложения – интернет магазины с большим массивом товаров (**Amazon**, **Citilink**и т.п.).

|  |  |
| --- | --- |
| Привычны посетителям, легкая seo-оптимизация | Сложная разработка и поддержка, существенные финансовые вливания |

### PWA (****Progressive Web Application**** )

1. **Прогрессивные веб-приложения (PWA)**: PWA - это веб-приложения, использующие современные веб-технологии для предоставления пользователям возможностей, аналогичных нативным приложениям, включая автономную функциональность, push-уведомления и доступ к аппаратным средствам устройства.

(**PWA**, прогрессивные web-приложения) – новый способ «подачи» web-сервисов, который максимально сближает их с обычным, привычным десктопным приложением, но на качественно более высоком уровне.  
  
Представим ситуацию: человек посещает некоторый сайт, который предлагает ему установить его на другие устройства. Теперь и на ПК, и в телефоне вы сможете получать уведомления, работать оффлайн, независимо от модели устройств и их мощностей.  
  
Главная область применения таких приложений – мобильные устройства. Пользователю больше не нужно входить на **AppStore** или **PlayMarket**, чтобы скачать программу – все сделает браузер автоматически (а еще создаст ярлык на рабочем столе, позволит работать с собой без доступа к сети и т.д.). Фактически, мы получаем аналог обычного приложения с тем же функционалом и множеством плюсов (для **PWA**не требуется лишняя память смартфона, ради которой приходится удалять фото и видео, они весят менее 1 Мб).  
  
При появлении нового контента **PWA**отправляет пользователю push-уведомление. Следует признать, что в скором времени эти приложения смогут заменить практически все мобильные аналоги. Хоть технология и появилась в 2007 году в компании **Apple**, изначально она «не зашла» в силу слабой «прогретости» публики, зато ко второй половине 2016 года **Google**развил ее и сделал популярной.  
  
**PWA-версии** приложений встречаются у многих компаний (тот же **Aliexpress**почти в 2 раза повысил конверсию от новых посетителей благодаря этому).

[Прогрессивные веб-приложения](https://appmaster.io/ru/blog/progressivnye-veb-prilozheniia-pwa-protiv-nativnykh-prilozhenii-kakoi-tip-budet-luchshe-v-2022-godu) (PWA) - это веб-приложения, которые используют современные веб-технологии для предоставления пользователям возможности работы с родными приложениями, включая автономную функциональность, [push-уведомления](https://appmaster.io/ru/blog/push-uvedomleniia) и доступ к аппаратным средствам устройства, без необходимости загрузки и установки [родного приложения](https://appmaster.io/ru/blog/otlichie-nativnyh-mobilnyh-prilozhenij-ot-vseh-ostalnyh) из магазина приложений.

PWA разработаны для обеспечения бесперебойной работы на всех устройствах и платформах, независимо от операционной системы или размера экрана. Они используют принципы отзывчивого дизайна для адаптации к различным размерам экрана и типам ввода, и доступ к ним возможен через любой современный веб-браузер.

PWA создаются с использованием комбинации веб-технологий, включая HTML, CSS, JavaScript и Service Workers, что позволяет им работать в автономном режиме и быстро загружаться даже в медленных или ненадежных сетях. Они также используют файл Web App Manifest, в котором содержится информация о метаданных приложения, таких как его название, иконка и цвет темы.

PWA имеют ряд преимуществ перед традиционными "родными" приложениями, включая более быстрое время загрузки, меньшее потребление данных и более высокую вовлеченность. Они могут быть добавлены на главный экран, как нативные приложения, и предлагают бесшовный опыт, стирающий грань между веб-приложениями и нативными приложениями.

PWA представляют собой значительный сдвиг в том, как мы создаем и используем веб-приложения, и они быстро завоевывают популярность как среди разработчиков, так и среди пользователей

|  |  |
| --- | --- |
| Быстрая разработка, высокая скорость загрузки, оффлайн-работа | Поддерживается не всеми браузерами и платформами |

## Развитие веб-приложений

На клиент-серверной архитектуре построена большая часть приложений. Сейчас большая часть приложений работает так:

1. Клиент делает первичный запрос на сервер.
2. Сервер отвечает HTML-страницей, иногда с набором каких-то данных внедрённых в виде JS-объекта в конце страницы.
   1. Страница может быть либо отрисована на сервере и тогда клиенту придёт готовый HTML.
   2. Либо отрисовкой будет заниматься сам клиент с помощью какой-нибудь библиотеки, например, React. В этом случае от сервера приходит просто набор необходимых данных.
3. Пользователь совершает какое-то действие, например, просит отсортировать таблицу.
4. Клиент в ответ на это действие решает, какой запрос отправить на сервер, строит этот запрос и отправляет его.
5. Сервер принимает этот запрос, обрабатывает его и отправляет на клиент порцию новых данных.
6. Клиент принимает данные и перерисовывает часть страницы по ним сам. То есть он уже не заменяет один кусок разметки другим готовым, а рисует разметку сам.

Плюсов такого общения (когда передаются только данные) несколько:

* Сервер и клиент становятся независимыми друг от друга. Сервер может ничего не знать об устройстве страниц, ему достаточно лишь уметь работать с БД и обрабатывать данные (первичная отрисовка может быть сделана самим сервером с помощью SSR).
* Количество информации, которое приходится передавать и принимать, меньше — а это уменьшает объём трафика.
* Логика приложения на сервере может быть проще, потому он и клиент становятся менее зависимы друг от друга в плане формата данных.

**1.2 Обзор существующих методов рекомендательных систем**

**Рекомендательный сервис**

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.

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 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.

У меня взята на основе знаний за счет полученного теста!

**1.3. Обзор средств разработки веб-приложений**

**ЧТОТО ОБЩЕЕ НАПИСАТЬ**

**1.3.1. Django**

Фреймворк — это набор готовых библиотек и инструментов. Фреймворк Django в первую очередь удобен работой «из коробки». Django можно представить как конструктор, где есть необходимые типовые блоки, например, такие как авторизация, для работы приложения.

Django существует почти 20 лет. Это большой фреймворк, поэтому он развивается и совершенствуется относительно медленно.

Преимущества Django:

* наличие сообщества

Его безусловное преимущество в дружественном сообществе Django. Ответы на вопросы по работе фреймворка легко найти в интернете, ему посвящены тысячи статей и обучающих видео. Официальная документация Django является одной из лучших.

* большой функционал

В Django есть почти все для создания приложения, он способен решить большинство задач при разработке. Помимо ORM, системы аутентификации пользователей, панели администратора и форм, для работы с API Django предоставляет REST Framework, для управления контентом - Django CMS. А значит разработчик не будет тратить время на поиск дополнительных библиотек.

* SEO-ориентированность

В Django можно 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.

*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.

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.

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 случаях. В более-менее серьезном приложении от этого больше проблем, чем достоинств.

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

*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. Представление получает данные из модели через контроллер. Функция представления — отображение данных, которые хранятся в моделях.

 Модель является движущей силой приложения. Это, как правило, данные приложения, обычно полученные с сервера. Любой пользовательский интерфейс с данными, которые видит пользователь, получен из модели или подмножества модели. **Модель** : представляет данные и бизнес-логику приложения. Он отвечает за обработку, хранение и управление данными, а также за реализацию всех необходимых бизнес-правил. Модель не зависит от пользовательского интерфейса и не взаимодействует напрямую с представлением или контроллером.

Контроллер – 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** : действует как посредник между моделью и представлением. 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. Архитектура MVC обеспечивает слабосвязанные компоненты, улучшая удобство обслуживания и тестирования приложений.

*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.

**1.3.2. 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 HTML можно создать простой сайт, но для более сложных с красивым дизайном и динамическими элементами без CSS и Java Script не обойтись.

**1.3.3. 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.

**1.3.4. 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 фронтенд 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.

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

Использовать среду разработки намного удобнее, чем обычный текстовый файл. В средах разработки предусмотрена подсветка синтаксиса. Код можно сразу запустить и увидеть результат работы. В IDE легко устанавливаются необходимые для работы библиотеки, а также в них есть встроенная поддержка систем контроля версий.

Для 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 заточена под работу в Windows и С#. Visual Studio более неповоротливый и потребляет больше системных ресурсов, однако, лучше показывает себя на крупных проектах.

*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

## 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

**2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

**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) поиск через прохождение тестирования, которое определяет предпочтение 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. Разработка базы данных**

**Цуль: разработка бд для?**

**Задачи:**

* **выбор БД**
* Проектировка ER-диаграммы
* Создание описания к диаграмме
* Формирование базы данных

**2.2.1. Выбор СУБД**

Для хранения табличной структуры выбрана реляционная база данных ////.

Производительность

Для веб-приложений важна скорость работы. Пользователь не будет ждать загрузки страницы — он просто уйдет с неё, если приложение не будет работать достаточно быстро. В плане производительности приложения «болезненным» местом является база данных, и MySQL не подведет вас.

Кроссплатформенность

MySQL предоставляет кроссплатформенную базу данных: она работает на Linux, FreeBSD и конечно на Windows. Этот критерий нужно учитывать при выборе СУБД для проектов, нацеленных на несколько платформ, в частности веб-приложений. К слову, MySQL входит в состав популярных стеков для разработки веб-приложений — LAMP (Linux, Apache, MySQL, PHP) и WAMP (Windows, Apache, MySQL, PHP).

Бесплатная СУБД

Oracle развивает MySQL как проект с открытым исходным кодом, что позволяет сообществу вносить в него свой вклад и, при необходимости, адаптировать его под свои задачи. СУБД предоставляется для конечных пользователей бесплатно. Многие компании выбирают в качестве СУБД MySQL из-за того, что она бесплатна.

Oracle предлагает клиентам некоторые проприетарные модули с закрытым кодом, для использования которых потребуется денежное вложение. Но эти модули нужны не всем и не всегда.

Выбор движков

Движок базы данных (database engine) — это программный модуль, с помощью которого СУБД создает, читает и обновляет данные в базе. В MySQL есть два типа движков: транзакционные и нетранзакционные. Транзакционные движки хранения данных подразумевают, что операции в этих БД могут быть отменены, если они не завершены.

Всего MySQL поддерживает 8 движков:

**InnoDB**— это распространенный движок, который используется в MySQL по умолчанию.  Он поддерживает ACID, блокировку на уровне строк, восстановление после сбоев и многопоточность. Также это единственный движок, который обеспечивает ограничение ссылочной целостности внешнего ключа. Oracle советует использовать именно InnoDB, за исключением ряда случаев.

Безопасность

MySQL предлагает пользователям высокий уровень безопасности. В MySQL есть встроенные инструменты безопасности, которые поддерживают управление пользователями и их привилегиями. При недостатке стандартных инструментов пользователь всегда может установить дополнительные плагины.

Сообщество

Большое и развитое сообщество MySQL не оставит пользователя одного с проблемой. Кроме того, благодаря сообществу на базе MySQL появились новые системы управления базами данных — Drizzle, OurDelta, Percona Server и, самый популярный, MariaDB.

Минусы MySQL

Работа с большими базами

При работе с малыми и средними приложениями MySQL радует пользователей, но при увеличении объемов информации у MySQL возникают проблемы. Речь идет о базах данных с миллионами строк. Такие БД тяжело масштабировать, а сложные запросы к ним начинают буксовать и иногда не успевают выполняться в допустимое время ожидания. Для решения проблемы с запросами придется прилагать усилия для их оптимизации.

Большие объемы информации— это сфера применения для Oracle Database или SQL Server, а не MySQL.

SQL

MySQL не соответствует стандарту SQL: СУБД не поддерживает некоторые функции и имеет расширения, не относящиеся к стандартному SQL. Для большинства проектов это не станет серьезной проблемой, но при миграции на другие СУБД этот момент может всплыть.

Развитие

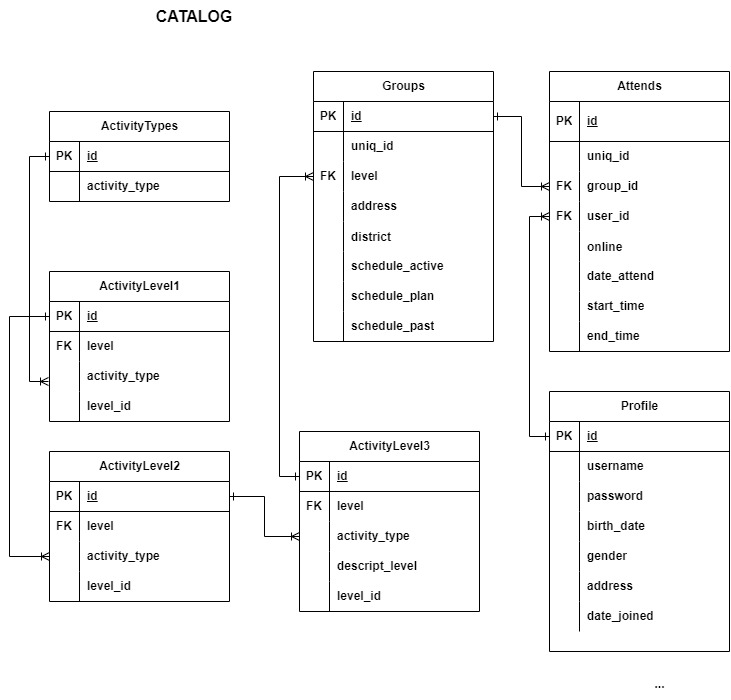
MySQL — это проект с открытым исходным кодом, что негативно сказывается на скорости его развития. Сообщество разработчиков сосредоточено в первую очередь на поддержку уже существующих функций, а Oracle, с учетом наличия проприетарных модулей для MySQL с закрытым кодом, не сильно заинтересованы в развитии проекта.

**2.2.2. Проектировка ER-диаграмм**

*Users*

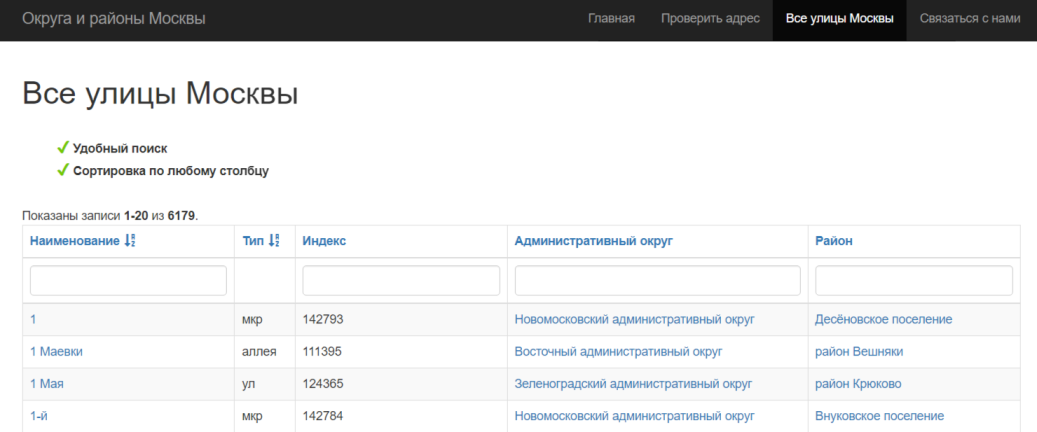
*Catalog*

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.

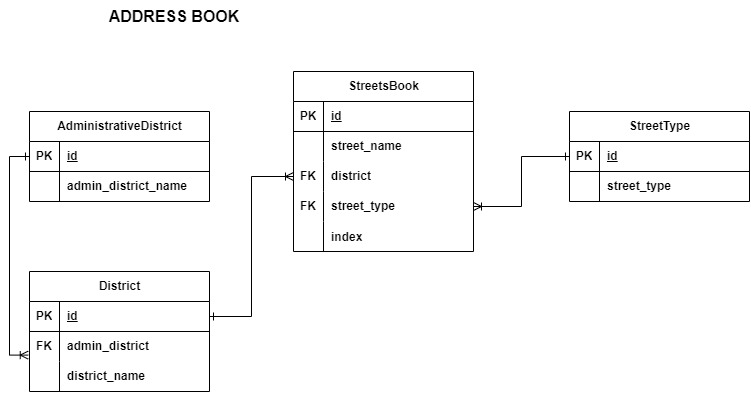


*AddressBook*

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>.



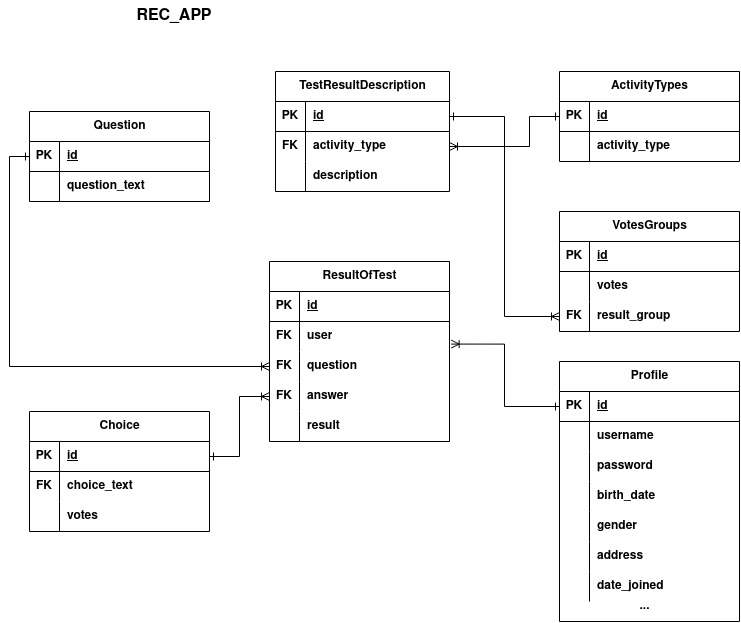
Для парсинга были использованы библиотеки: BeautifulSoup и Pandas. С помощью инструментов разработчика на сайте получено название таблицы <table class="table table-striped table-bordered”>, и из нее в цикле выгружена информация из каждой строки. Обработка данных не использовалась. Заголовки созданы вручную,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



*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);
* ////
* результаты тестирования.



Программной реализацией будет являться веб-клиент и веб-сервер, в связке образующее Full-Stack Web-приложение, выполняющее необходимые нам функции.

При GET-запросе на \*/ адрес сервера, он должен отдавать веб-клиент в виде одной страницы и нескольких JS-скриптов, в которых будет основной функционал. При подключении клиента, сервер должен отдавать содержание выделеной директории.

Далее, на клиенте после создания некоторой структуры файлов из переданного содержания, клиент посредством POST запроса может передать структуру на сервер или сохранить ее на локальном хранилище в виде XML-файла, чтобы в будущем можно было использовать эту структуру файлов внезависимости от доступности сервера.

Должна поддерживаться загрузка структуры с клиента на сервер для редактирования или сохранения этой структуры на сервере.

Сама структура представляет из себя ничто иной, как дерево, начинающееся с корня папки. Каждый узел дерева предсталяет из себя массив из трех элементов: имени или пути (filepath), поддерева (dir), и технических элементов (realloc или active).

Для передачи запросов с клиента будет использоваться недавно разработанный метод fetch, а формат передачи через сеть — JSON.

# 3.2 Выбранные технологии и платформа

**Выбор языка разработки рекомендательной системы**

РС можно написать на разных языках: Python, Matlab, Java, C++ и других, но чаще всего для этой задачи используется Python. Python является высокоуровневым языком. Помимо веб-разработки он используется при работе с большими данными. Python платформенно независимый язык, его можно адаптировать практически к любой операционной системе.

* Python – это очень популярный, широко используемый язык программирования общего назначения. Он имеет обширный набор специально разработанных модулей и широко используется разработчиками. Многие онлайн-сервисы предоставляют API для Python.
* Python очень прост в изучении. Низкий порог вхождения делает его идеальным первым языком для тех, кто занимается программированием.
* Такие программные пакеты как pandas, scikit-learn и Tensorflow, делают Python надежным вариантом для современных приложений в области машинного обучения.

**Недостатки:**  
Типобезопасность. Python – это динамически типизированный язык, а это значит, что вы должны быть осторожными при работе с ним. Ошибки несоответствия типов (например, передача строки (string) в качестве аргумента методу, который ожидает целое число (integer)) могут время от времени случаться.

**Наш вердикт – удобен во всех отношениях**  
  
Python является хорошим вариантом для целей науки о данных (data science), и это утверждение справедливо как для начального, так и для продвинутого уровней работы в данной области. Большая часть науки о данных сосредоточена вокруг процесса ETL (извлечение-преобразование-загрузка). Эта особенность делает Python идеально подходящим для таких целей языком программирования. Библиотеки, такие как Tensorflow от Google, делают Python очень интересным языком для работы в области машинного обучения.

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.

В качестве платформы был выбран Node.js. Платформа позволяет использовать JavaScript не только для разработки клиента в веб-браузере, но и в Backend разработке для написания скриптов сервера. Для упрощения разработки для Node.js существует множество фреймворков.

Используемый в данном приложении Express предлагает баланс между гибкостью низкоуровневой разработки сервера и удобностью использования готовых движков.

Клиент был реализован на паттерне SPA (Single Page Application) с фреймворком React.JS, который позволяет полноценно использовать приложение не загружая постоянно новые страницы, а также работу в оффлайн режиме. React.JS — один из самых перспективных современных фреймворков, который поддерживает много разработчиков со всего мира, поэтому можно будет не бспокоиться о расширении функционала приложения в будущем.

Средой разработки был выбран JetBrains WebStorm. Она предлагает единовременную поддержку как React.JS, так и Node.JS Express, быструю работу и встроенную поддержку git, хорошую подстветку синтаксиса JS и JSX. Интерфейс программы показан на рисунке 1.

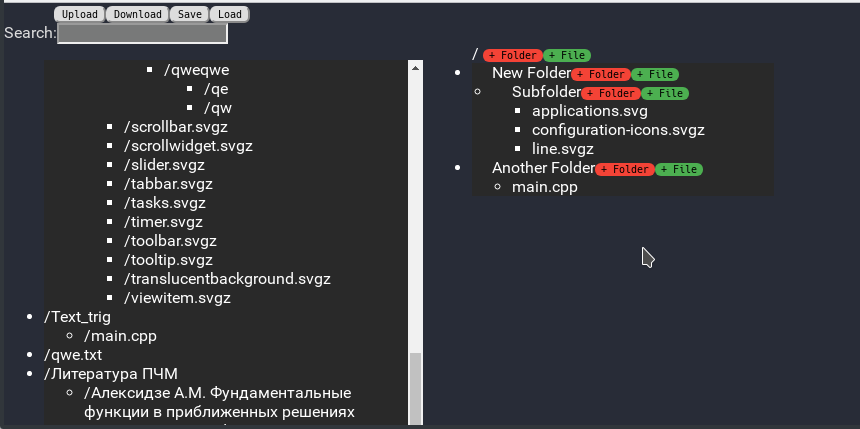
Рисунок 1 — Интерфейс IDE WebStorm.

# 3.3 Результат работы

Веб-приложение, выполненное в результате, представлено на рисунке 2.

 Рисунок 2 — общий вид веб-приложения

Дабы протестировать приложение, создадим тестовую структуру файлов, подемонстрированную на рисунке 3.

 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.

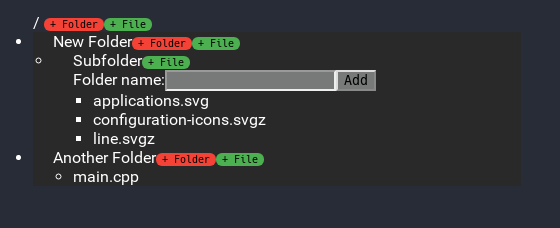


Рисунок 4 — создание папки.

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.

ОСНОВНАЯ ЧАСТЬ

РХИТЕКТУРА ПРОЕКТА

**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.

**ERD**

Выделим следующие сущности:

пользователь

вопрос

ответ

тип активности

уровень 1 активности

уровень 2 активности

уровень 3 активности

=

4. Описание архитектуры

База данных, построенная с использованием MySQL

Компоненты:

- Python, включая библиотеки:

-os,

-shutil,

-logging,

-pandas, numpy

-sklearn,

-Beautifulsoup

-

Архитектура проекта заключается в:

- последовательной обработке поступающего файла в формате excel;

- парсинге данных с интернет-ресурса;

- последующей записи полученных данных в базу данных (SQLite) с префиксом

dwh;

- подготовке данных для витрины;

- визуализации данных из витрины;

- создании модели классификации на основе ИД.

- последующей записи полученных результатов в базу данных (SQLite);

Весь код обработки данных написан на Python и SQL. В ИД не указано, как часто

поступают данные для обработки. Для целей данной работы предположим, что файл

поступает раз в сутки, также раз в сутки запускается скрипт. Задача скрипта -

добавление в базу новых данных из полученного нового файла и его последующая

архивация.

База данных состоит из двух уровней:

- стейджинг, на который подгружаются приходящие данные

- основной, где хранятся данные

6. Решение по используемой СХД

СХД - устройство для хранения и управления данными и их резервного

копирования. ИД приходят в файлах. За год всего 20000 транзакций, следовательно

промежуточные файлы удобно хранить в файловой СХД. Преобразованные данные

хранятся в базе данных, размещенной на блочной СХД.7. Перечисление алгоритмов и методов анализа

предлагаемых к использованию в решении кейса.

и

обработки,

Парсинг, обработка поступающих данных, запись и обновление данных в БД

происходит с помощью DAG в Airflow. Модель классификации и метрики

сохраняются в Model Registry в MLFlow. Результат записывается в базу данных.

Airflow и MLFlow разворачиваются из контейнера Docker.

8. Описание модели угроз

Перечень возможных угроз:

1. Неограниченные привилегии доступа у сотрудников

2. Внедрение SQL – кода

3. Непрофессионализм сотрудников при работе с информацией

4. Открытие копии БД

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)

Список данных, содержащихся в датасете

## Создание вирутального окружения

Виртуальная среда позволяет вам изолировать ваши текущие зависимости проекта от остальных пакетов, установленных глобально в вашей системе или в других виртуальных средах. Вы можете использовать virtualenv, который необходимо установить в вашей системе, или модуль venv, доступный в последних версиях Python 3.

Перейдите к своему командному терминалу и запустите:

$ python -m venv env

Затем активируйте созданное окружение:

$ source env/bin/activate

Примечание: обратите внимание, что в Windows вам нужно использовать source env/Scripts/activate для активации вашей виртуальной среды.

После активации окружения установите Django с помощью pip:

$ pip install django

Если инфраструктура успешно установлена, то теперь вы можете использовать команды управления Django для создания и работы с вашим проектом.

Нам также необходимо установить mysql-client с помощью:

$ pip install mysqlclient

## Создание базы данных MySQL

Мы будем использовать базу данных MySQL. В терминале вызовите клиент mysql, используя следующую команду:

$ mysql -u root -p

Введите свой пароль MySQL и нажмите Enter.

Затем выполните следующую инструкцию SQL для создания базы данных:

mysql> create database mydb;

## Создание проекта Django

Давайте теперь создадим проект, используя django-admin.py. В терминале выполните следующую команду:

$ django-admin.py startproject demoproject

В Django есть средство для работы с БД — ORM, который абстрагирует операции с базой данных напрямую и поддерживает SQLite, который настроен по умолчанию в проекте.

Если вам нужно использовать в своем проекте Django [PostgreSQL](https://django.fun/tutorials/perehod-na-postgresql-s-mysql/), MySQL или любую другую систему управления базами данных, вам нужно сначала установить ее, а затем открыть файл settings.py вашего проекта и добавить адрес базы данных и учетные данные в объект DATABASES.

Вот конфигурация для MySQL:

DATABASES = {

'default': {

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

'NAME': 'mydb',

'USER': 'root',

'PASSWORD': 'YOUR\_DB\_PASSWORD',

'HOST': 'localhost',

'PORT': '3306',

}

}

Обязательно замените YOUR\_DB\_PASSWORD своим собственным паролем MySQL.

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.

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/

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

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://maksyutin.ru/Архитектура ПО/Архитектурные шаблоны/Шаблоны архитектуры /)

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.

**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.