ДИПЛОМ

Рекомендательный сервис для выбора активностей

СОДЕРЖАНИЕ

Основная часть:

Архитектура

Парсинг

Загрузка данных в базу

Или SDLC: анализ, планирование, проектирование, разработк

**ВВЕДЕНИЕ................................................................................................................6**

**1. ТЗ .................................................................................... 4 4**

**2 Теоретические аспекты веб-приложений и рекомендательных систем**

**- Общие понятие по веб-приложению**

**- Обзор существующих методов рекомендательных систем.........................7**

1.3 Обзор существующих платформ разработки сайтов..................................16

1.4 Обоснование выбора среды разработки и алгоритма

рекомендательной системы......................................................................................22

1.5 Выводы по разделу........................................................................................23

2 РАЗРАБОТКА БАЗЫ ДАННЫХ.........................................................................24

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»............................................................29

3.1.4 Use-case: «Фильтрация альбомов»........................................................29

3.1.5 Use-case: «Выбор альбома»....................................................................30

3.1.6 Use-case: «Получение информации об альбоме».................................30

3.1.7 Use-case: «Прослушивание альбома»...................................................30

3.1.8 Use-case: «Оценка альбома»..................................................................30

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 баллам, тогда это можно будет учитывать в модели.

**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).

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

# Общая информация

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

Информация, доступная пользователям интернета, хранится на серверах – специализированных компьютерах с большой памятью. Сервера находятся в специализированных зданиях - 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.

Гипертекстовые документы, размещаемые в интернете, называются веб-страницами. Набор веб-страниц с общей темой и оформлением, а также связанных между собой ссылками, называется веб-сайтом.

Современные веб-приложения — это …..

## 

## Веб-приложение

Первые веб-сайты были статическими, и соответственно обладали небольшой функциональностью. Веб-приложения появились как что-то среднее между статичными 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.

Таким образом, что веб-приложения имеют большое количество плюсов при отсутствии видимых минусов, самым большим и очевидным из которых является невозможность использования приложений при отсутствии доступа в интернет.

Как работают веб-приложения?

Веб-приложения имеют архитектуру клиент-сервер. Их код разделен на два компонента: скрипты на

стороне клиента и скрипты на стороне сервера.

Архитектура на стороне клиента

Скрипт на стороне клиента отвечает за функциональность пользовательского интерфейса, например,

кнопки и выпадающие поля. Когда конечный пользователь нажимает на ссылку веб-приложения, веб-

браузер загружает скрипт на стороне клиента и отображает графические элементы и текст для

взаимодействия с пользователем. Например, пользователь может читать контент, смотреть видео или

заполнять данные в форме контакта. Такие действия, как нажатие кнопки отправки, передаются на

сервер в виде клиентского запроса.

Архитектура на стороне сервера

Скрипт на стороне сервера обрабатывает данные. Сервер веб-приложений обрабатывает запросы клиентов и отправляет

Обычно запросы касаются получения дополнительных данных или

работуответ.

изменения или сохранения новых данных. Например, если пользователь нажимает на кнопку

Подробнее, сервер веб-приложений отправляет контент обратно пользователю. Если пользователь

нажмет кнопку Отправить, сервер приложений сохранит данные пользователя в базе данных. В

некоторых случаях сервер завершает запрос данных и отправляет полную HTML-страницу обратно

клиенту. Это называется рендерингом на стороне сервера.

# **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!

**Выбор среды 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.

Открытость

Минусы1

И почему выбран он

# 2. СРЕДСТВА РАЗРАБОТКИ WEB-ПРИЛОЖЕНИЙ

## 2.1 HTML

Язык разметки гипертекста (Hypertext Markup Language), или, как его чаще называют, HTML, — это компьютерный язык, лежащий в основе World Wide Web (Всемирной Паутины). Благодаря языку HTML любой текст можно разметить, преобразовав его в гипертекст с последующей публикацией в Web.

Язык HTML имеет собственный набор символов, с помощью которых Web-браузеры отображают страницу. Эти символы, называемые дескрипторами, включают в себя элементы, необходимые для создания гиперссылок

Одной из отличительных особенностей HTML-документов является то, что сам документ содержит только текст, а все остальные объекты встраиваются в документ в момент его отображения Браузером с помощью специальных тэгов и хранятся отдельно. При сохранении HTML-файла в месте размещения документа создается папка, в которую помещаются сопутствующие ему графические элементы оформления

## 2.2 CSS

Каскадные таблицы стилей или CSS (от английского Cascading Style Sheets) - это довольно позднее нововведение. Если сам HTML появился в 1989 году, то таблицы стилей - только в 1997. Мало того, таблицы стилей не считаются частью HTML. Связано это с тем, что WWW была создана учеными как средство для обмена текстовыми документами, а HTML был языком, с помощью которого создавали эти документы. Для ученых главным было содержимое документа, а не его оформление. Поэтому первые версии HTML не включали даже средств для размещения на Web-страницах графических изображений, не то, что для сложного форматирования текста. Но время шло, и в Интернет пришел обыватель, тотчас потребовавший от Web-дизайнеров «сделать ему красиво». А Web-дизайнеры, в свою очередь, потребовали от разработчиков стандарта HTML средств, облегчающих им работу.

Каскадные таблицы стилей или просто таблицы стилей (CSS - Cascading Style Sheets) - это набор правил, описывающих форматирование разных фрагментов HTML-кода и хранящихся отдельно от него. Одно такое правило, отображающее форматирование какого-то одного фрагмента или однотипной группы фрагментов кода, называется стилем. Таблицы стилей описываются на особом языке CSS и хранятся в особых файлах с расширением сss, хотя могут быть внедрены в саму Web-страницу.

## 2.3 Ajax

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.

Поведение сервера тоже изменилось. Если раньше на каждый запрос сервер выдавал новую страницу, то теперь он отсылает лишь те данные, которые нужны клиенту, а HTML из них прямо в браузере формирует движок Ajax.

Асинхронность проявляется в том, что далеко не каждый клик пользователя доходит до сервера, причем обратное тоже справедливо - далеко не каждая реакция сервера обусловлена запросом пользователя. Большую часть запросов формирует движок Ajax, причем его можно написать так, что он будет загружать информацию, предугадывая действия пользователя.

Где стоит использовать Ajax:

· Формы.

· Навигация в виде "дерева".

· Поиск по файлам и папкам.

## 2.4 JavaScript

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

## 2.5 ДЖАНГО

Архитектурный шаблон MVC делит приложение на

три отдельные модульные части:

 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.

Представление–это пользовательский интерфейс, который пользователь видит и взаимодействует. Он динамический и генерируется на основе текущей модели приложения.

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.

## 2.5 JetBrains WebStorm

ОПИСАНИЕ

## 

## 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

# 3.1 Постановка задачи

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

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.

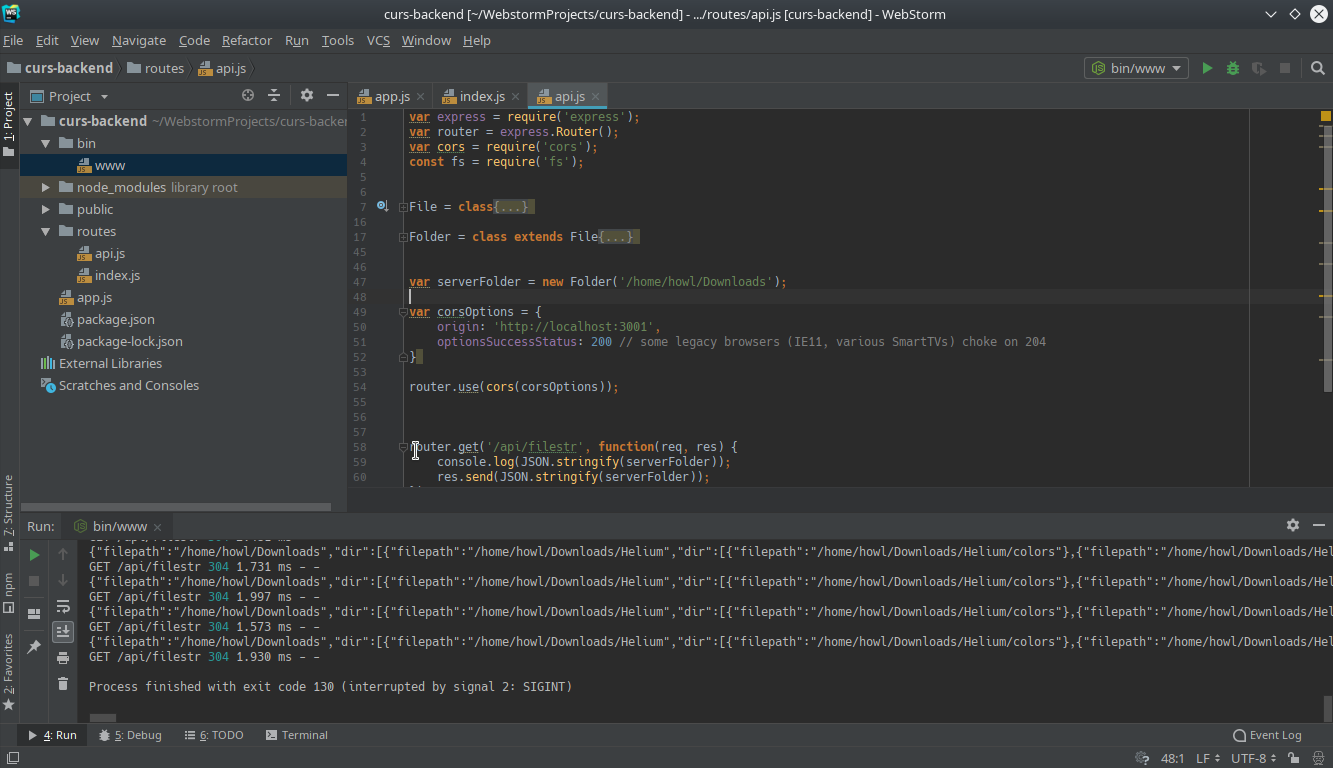


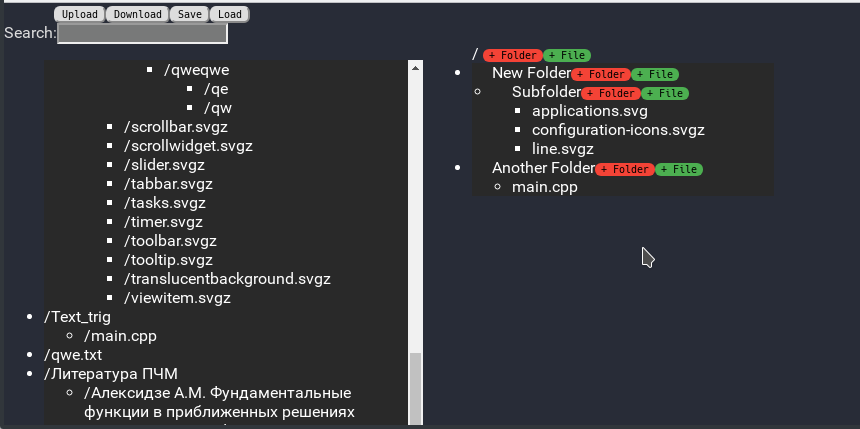
Рисунок 1 — Интерфейс IDE WebStorm.

# 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.

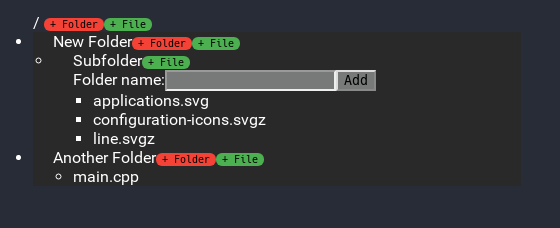


Рисунок 4 — создание папки.

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.

# ПРИЛОЖЕНИЕ А

# ПРОЦЕДУРЫ ПОИСКА И ДОБАВЛЕНИЯ ЭЛЕМЕНТОВ

ОСНОВНАЯ ЧАСТЬ

Django описывают как «веб-фреймворк для перфекционистов с дедлайнами». Его создали, чтобы переходить от прототипов к готовым сервисам как можно быстрее.

Фреймворк поможет разработать CRUD приложение под ключ. С Django не придется изобретать велосипед. Он работает из коробки и позволит сосредоточиться на бизнес-логике и продуктах для обычных людей.

## Плюсы Джанго

### Принцип «Всё включено» («Batteries included»)

Фраза «всё включено» означает, что большинство инструментов для создания приложения — часть фреймворка, а не поставляются в виде отдельных библиотек.

Django содержит огромное количество функциональности для решения большинства задач веб-разработки. Вот некоторые из высокоуровневых возможностей Django, которые вам придётся искать отдельно, если вы предпочтёте микро-фреймворк:

* ORM
* Миграции базы данных
* Аутентификация пользователя
* Панель администратора
* Формы

### Стандартизированная структура

Django как фреймворк задаёт структуру проекта. Она помогает разработчикам понимать, где и как добавлять новую функциональность.

Благодаря одинаковой для всех проектов структуре гораздо проще найти уже готовые решения или получить помощь от сообщества. Огромное количество увлеченных разработчиков поможет справиться с любой задачей гораздо быстрее.

### Приложения Django

Приложения в Django позволяют разделить проект на несколько частей. Приложения устанавливаются путём добавления в settings.INSTALLED\_APPS. Этот подход позволяет легко интегрировать готовые решения.

Сотни универсальных модулей и приложений очень сильно ускорят разработку. Взгляните на их список на сайте [djangopackages.org](https://djangopackages.org/).

### Безопасный по умолчанию

Django безопасен из коробки и включает механизмы предотвращения распространенных атак вроде SQL-инъекций (XSS) и подделки межсайтовых запросов (CSRF). Подробнее об этом можно почитать в [официальном руководстве по безопасности](https://docs.djangoproject.com/en/2.2/topics/security/).

### REST Framework для создания API

Django REST Framework, который часто сокращают до «DRF», является библиотекой для построения API. Он имеет модульную и настраиваемую архитектуру, которая хорошо работает для создания как простых, так и сложных API.

В DRF политики аутентификации и разрешений доступны из коробки. Он поставляется с базовыми классами для CRUD операций и встроенной утилитой для тестирования разрабатываемого API.

### GraphQL фреймворк для создания API

Большие REST API часто требуют большого количества запросов для получения всех необходимых данных. GraphQL — это язык запросов, который позволяет обмениваться связанными данными гораздо проще. Подробнее почитать про основные концепции GraphQL можно в [официальной документации](https://graphql.org/learn/).

[Graphene-Django](https://docs.graphene-python.org/projects/django/en/latest/) позволит легко добавить соответствующую функциональность в ваш проект. Модели, формы, аутентификация, политики разрешений и другие функциональные возможности Django можно использовать для создания GraphQL API. Библиотека так же поставляется с утилитой для тестирования результата.

## Недостатки Джанго

### Django ORM

Django ORM сегодня значительно уступает последней SQLAlchemy.

Django ORM основан на шаблоне Active Record, который хуже, чем шаблон Unit of Work, используемый в SQLAlchemy. На практике это выражается в том, что в Django модели могут «сохранять» себя по желанию, а транзакции отключены по умолчанию. Подробнее об этом можно почитать в статье [«Вещи, которые мне не нравятся в Django»](http://dev.nando.audio/2014/04/04/why_i_sort_of_dislike_django.html).

### Django развивается медленно

Django является большим и монолитным фреймворком. Это позволяет сообществу разрабатывать сотни универсальных модулей и приложений, но снижает скорость разработки самого Django. Кроме того, фреймворк должен поддерживать обратную совместимость, поэтому он развивается относительно медленно.

### В итоге: должен ли я выбрать Django?

Хотя Django ORM не так гибок, как SQLAlchemy, а большая экосистема многократно используемых модулей и приложений замедляет развитие инфраструктуры, очевидно, Django должен быть первым кандидатом на роль фреймворка для питониста.

Альтернативные легкие фреймворки типа Flask, хотя и позволяют быть свободнее Django в экосистеме и конфигурации, могут потребовать лишнего времени на поиск/создание дополнительных библиотек и функциональных возможностей в долгосрочной перспективе.

Стабильность Django и сообщество вокруг него выросли до невообразимых размеров с момента первого релиза. Официальная документация и учебные пособия по фреймворку являются одними из лучших в своём роде. А с каждой новой версией Django продолжает обрастать возможностями.

Т

## Почему Django — отличный фреймворк для веб-разработки: экосистема, SEO, библиотеки

Если вы спросите у нескольких разработчиков, почему они выбрали Django, ответы будут преимущественно одинаковыми. Ниже описаны основные преимущества фреймворка, благодаря которым он стал популярным.

### Развитая экосистема

Опытные разработчики рекомендуют воспринимать Django как систему. Это значит, что фреймворк обычно используется с большим количеством сторонних приложений. Их можно выбирать в зависимости от потребностей конкретного проекта.

Чтобы лучше понять этот принцип, представьте конструктор Lego. В нём есть много типовых блоков. В Django тоже есть типовые блоки. Например, блок авторизации или блок подписки на рассылку применяется практически в каждом проекте. Созданные с помощью фреймворка веб-приложения состоят из таких независимых блоков.

### Зрелость

Django был представлен в 2005 году. За 14 лет существования он сильно изменился и усовершенствовался. В фреймворке постоянно появляются новые возможности, а старые совершенствуются.

Важный момент: когда вы разбираетесь с Django и ищете ответ на конкретный вопрос, чаще всего это не вызывает затруднений. Тысячи специалистов уже решали такие же проблемы до вас и делились своим опытом в интернете. Так работает сообщество Django.

### Административная панель

Административная панель Django автоматически генерируется при создании приложения. Это избавляет разработчика от необходимости создавать админку вручную.

С помощью сторонних приложений дефолтную консоль управления Django можно усовершенствовать и адаптировать под нужды своего проекта. Кроме того, фреймворк позволяет настраивать интерфейс дефолтной административной панели.

### SEO-дружественность

Написанный на Python код получается читабельным и понятным даже неподготовленным людям. Это один из факторов, благодаря которым веб-приложения на Python считаются SEO-дружественными. Django генерирует семантические URL. Их также называют человеко-понятными URL или ЧПУ. В приложениях на Django легко реализуются другие функции, необходимые для поисковой оптимизации.

### Расширяемость

Функциональность Django расширяется с помощью плагинов. Это программные модули, которые позволяют быстро добавить на сайт нужную функцию. В официальном каталоге есть сотни плагинов, которые позволяют легко реализовать на сайте sitemap.xml, управлять доступами, подключить платёжную систему Stripe и так далее. При необходимости вы можете отключать или заменять плагины, чтобы адаптировать приложение к текущим нуждам проекта.

### Библиотеки

В популярных языках программирования есть библиотеки, с помощью которых удобно решать специальные задачи. В библиотеках можно найти готовые решения: функции, классы, конфигурации и так далее. Благодаря таким решениям расширяются возможности языка, а также упрощается создание приложений.

Django поддерживает использование библиотек при разработке веб-приложений. В число популярных библиотек входят:

* Django REST Framework, который упрощает работу с API.
* Django CMS — удобный инструмент для управления контентом.
* Django-allauth — с его помощью реализуются функции регистрации, авторизации, управления учётными записями.

### ORM

В Django реализовано объектно-реляционное отображение (ORM), которое обеспечивает взаимодействие приложения с базами данных (БД). ORM автоматически передаёт данные из БД, например, PostgreeSQL или MySQL, в объекты, которые используются в коде приложения.

Разработчики выбирают Django благодаря таким характеристикам:

* Разделение бизнес-логики и визуальной части на уровне архитектуры.
* SEO-дружественность.
* Расширяемость.
* Развитая инфраструктура: большое количество библиотек и плагинов.
* Многочисленное и дружественное сообщество, благодаря которому легко искать ответы на сложные вопросы.

## а

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.

**Описание данных**

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.

**ERD**

Выделим следующие сущности:

пользователь

вопрос

ответ

тип активности

уровень 1 активности

уровень 2 активности

уровень 3 активности

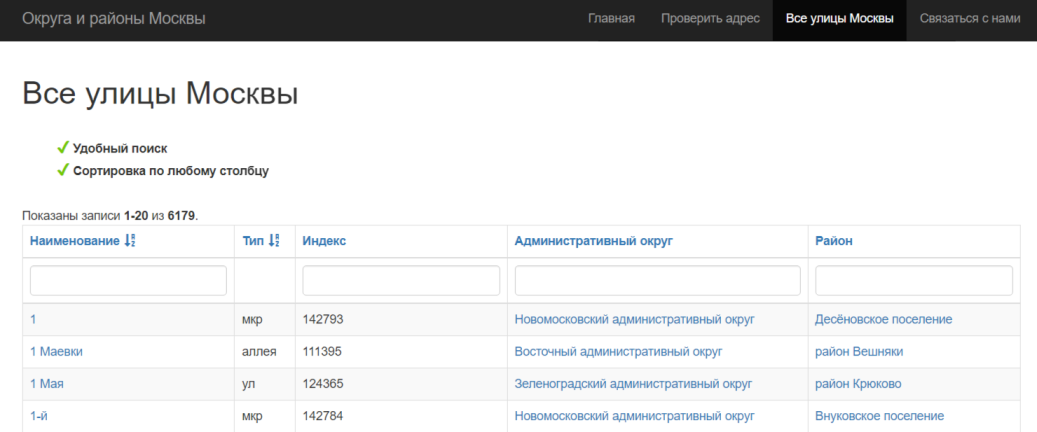
Для пользовтелей понадобится парсинг!

таблице юзерс представлены только адреса пользовтаелей.

В таблице групс есть привязка адресов к районам и округам.

Для удобства работы по местоположению пользователя добавлены район и округ проживания.

Для парсинга взят сайт <http://округа-районы.москва/все_улицы_москвы>.



Использовались библиотеки: BeautifulSoup и pandas. С помощью инструментов разрабочика 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".

Итого получена сущность "москоу стритс".

КАРТИНКА

СДЕЛАТЬ ТАК ЧТОБ парсер останавлився не от нолмера страницы

КАРТИНКИ

=

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

5. Выбор СУБД

## Преимущества MySQL

### 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.

**MyISAM**— этот движок использовался по умолчанию до появления InnoDB. И в отличие от него, MyISAM не поддерживает ACID. В этом движке реализована блокировка только на уровне таблиц, поэтому он небезопасен для транзакций. В тоже время MyISAM оптимизирован для чтения. Сфера применения MyISAM — это системы хранения с минимальными транзакционными данными.

**Memory**— этот движок реализуется хранение данных в памяти в виде кучи. Такая система обеспечивает очень быстрый доступ к данным. Memory поддерживает блокировку на уровне таблиц и хранит данные в оперативной памяти — при перезагрузке сервера они теряются. Этот движок подходит для временных таблиц.

**CSV**— этот движок хранит данные в файлах формата .csv. Не поддерживает индексирование и разделение. CSV используется для обмена данными между различными программами или приложениями.

**Merge**— движок работает с таблицами MyISAM. Он логически группирует ряд идентичных таблиц MyISAM и ссылается на них, как на один объект. Merge упрощает управление большими объемами данных и используется в системах хранения данных.

**Archive**— этот движок оптимизирован для высокоскоростной вставки. При вставке он сжимает данные. Archive не поддерживает транзакции, операции DELELE и UPDATE, но зато поддерживает блокировку на уровне строк. Отлично подходит для хранения больших объемов архивных данных, на которые редко ссылаются.

**Federated**используется создания одной БД на нескольких физических серверах.  Подходит для распределенных систем.

Движок **Blackhole**принимает данные, но не сохраняет их. На первый взгляд этот движок бесполезен, но на самом деле он находит применение при репликации данных и тестировании. Blackhole можно использовать в качестве фильтра между серверами.

### 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.

Для хранения табличной структуры выбрана реляционная база данных SQLite.

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

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://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/

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.