Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информационные системы и управление

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

Студенты Мазур Екатерина Алексеевна, Супрунова Екатерина Алексеевна, Лубянская Анастасия Алексеевна, Петрова Анна Алексеевна, Фролов Евгений Алексеевич

Группы ИУ7-26Б, ИУ7-25Б

Тип практики Проектно-технологическая

Название предприятия Московский государственный технический университет имени Н.Э. Баумана

Студент	_____ *подпись, дата*	**Мазур Е.А.** *фамилия, и.о.*
Студент	_____ *подпись, дата*	**Супрунова Е.А.** *фамилия, и.о.*
Студент	_____ *подпись, дата*	**Лубянская А.А.** *фамилия, и.о.*
Студент	_____ *подпись, дата*	**Петрова А.А.** *фамилия, и.о.*
Студент	_____ *подпись, дата*	**Фролов Е.А.** *фамилия, и.о.*
Ментор	_____ *подпись, дата*	**Хетагуров П.К.** *фамилия, и.о.*
Руководитель практики	_____ *подпись, дата*	**Оленев А.А.** *фамилия, и.о.*

Оценка _____

2020 г.

Содержание

Введение

Общее описание проблемы:
Самое популярное времяпрепровождение среди жителей России – это просмотр фильмов и сериалов. Во время самоизоляции свободное время на просмотр сериалов увеличилось. Существуют тысячи сериалов на любой вкус и ежегодно бесчисленная мировая база сериалов пополняется сотнями новых произведений. (только в США за 2019 год вышло 532 сериала). Необходим сервис для быстрого и удобного подбора сериала для просмотра.

Формулировка задачи:
Разработать веб-приложение для получения пользователем рекомендуемых к просмотру сериалов и расчета времени, которое пользователь потратит на просмотр сериала.
Приложение должно содержать следующие функции:
1) Предоставление информации о сериале.
2) Рекомендация сериала по критериям:

- жанр;
- год выхода сериала;
- страна;
- актеры, играющие главные роли;
- количество сезонов;
- общее количество серий;
- продолжительность одной серии;
- статус сериала;
- категория;
- количество дней/недель/месяцев, которое пользователь хочет потратить на просмотр сериала, учитывая количество серий, которое он собирается просматривать в день/неделю.

3) Расчет количества дней/недель/месяцев, которое пользователь потратит на просмотр сериала, учитывая количество серий, которое он собирается просматривать в день/неделю.

Состав команды с указанием написанных частей отчета:

- Мазур Екатерина - тимлид, бэкенд разработчик
  - Введение
  - Декомпозиция задачи
  - Структура создаваемого программного продукта
  - Ограничения, сделанные при решении задачи
  - Выбор и обоснование технических средств
  - Выбор и обоснование модели разработки
  - Развертывание разработанного программного продукта
  - Заключение
  - Список использованных источников
- Фролов Евгений - бэкенд разработчик

    - Аналитический раздел
    - Бэкенд разработка
- Петрова Анна – тестировщик
    - Описание способов тестирования
    - Подготовка модульных тестовых данных
    - Реализация тестирования
- Супрунова Екатерина - разработчик баз данных
    - Работа с базой данный
- Лубянская Анастасия - фронтенд разработчик, дизайнер
    - Фронтенд разработка

Аналитический раздел

После согласования идеи с ментором и формулирования технического задания мы разобрал уже готовые аналоги. Анализ проводился над 8 существующими проектами.

Цель разбора была: «Объединить достоинства готовых сайтов в один и добавить собственные функции».

В процессе анализа, мы выявили все преимущества и недостатки каждого сайта. При рассмотрении аналогов, учитывались такие параметры, как:

- Удобство интерфейса;
- Параметры поиска;
- «Фишки» сайта;

На **7** из **8** сайтах отсутствует поиск по «Категориям», «Статусу», «Количеству сезонов»;

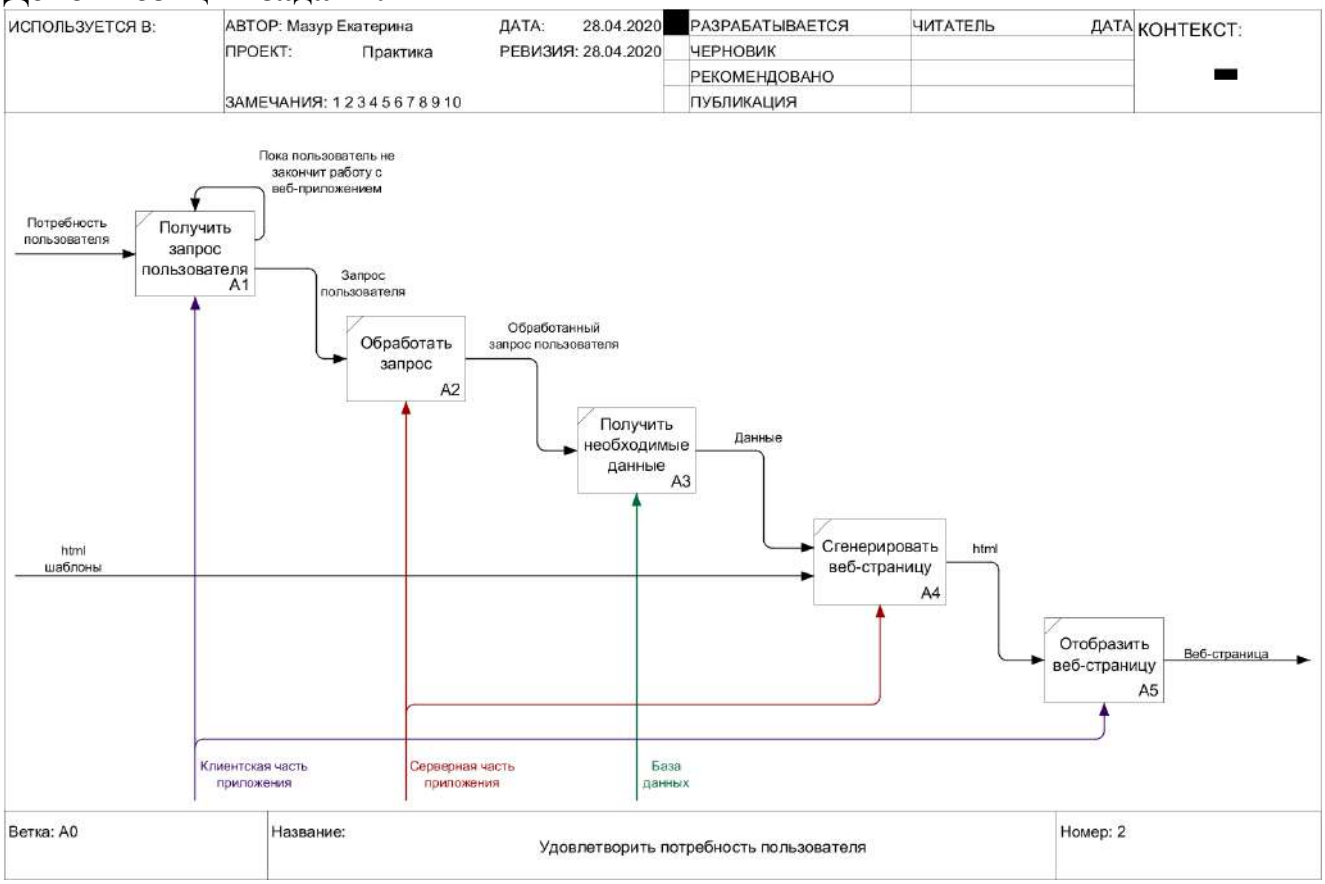
На **6** из **8** сайтах отсутствует поиск по «Актерам»;

Стоит отметить, что мы не нашли сервис, который подсчитывал бы количество времени, которое потребуется пользователю на просмотр сериала.

После анализа, мы объединили функционал других сервисов, дополнили его своими параметрами поиска, реализовали удобный интерфейс, и добавили собственные, уникальные функции.

Конструкторский раздел

Декомпозиция задачи:



Структура создаваемого программного продукта:

Разрабатываемое веб-приложение состоит из следующих компонент:

1. Клиентская часть (взаимодействие пользователя с приложением);
2. Серверная часть (обработка запросов и обращение к базе данных);
3. База данных (хранение информации).

Клиентская часть представляет собой графический интерфейс, который отображается у пользователя в браузере. Пользователь взаимодействует с приложением через браузер, отмечая необходимые параметры выбора сериала, представленные в виде чекбоксов, ползунков и форм ввода. Главное требование к клиентской части заключается в том, чтобы интерфейс приложения был удобным и интуитивно понятным.

Серверная часть веб-приложения - это программа на сервере, которая обрабатывает запросы пользователя. После получения отмеченных пользователем параметров, программа отбирает из базы данных подходящие сериалы, генерирует html страницу и отсылает ее назад клиенту. Если это необходимо пользователю, на сервере так же вычисляется количество времени, которое пользователь потратит на просмотр сериалов. Серверная часть должна точно и быстро обрабатывать запросы пользователя.

База данных используется для хранения данных о сериалах и списков параметров поиска. Она состоит из нескольких таблиц:

- Serials: информация о сериалах (столбцы: id, название, оригинальное название, количество сезонов, количество серий, продолжительность одной серии, год выхода, жанры, описание, год окончания, страны, статус, категория).
- Actors: список всех актеров, игравших главные роли в сериалах из таблицы Serials.
- Genres: список всех жанров.
- Countries: список всех стран.

Ограничения, сделанные при решении задачи:
Приложение функционирует только при доступе пользователя к интернету. Мобильная версия веб-приложения не предусмотрена.

Описание способов тестирования как выделенных компонент, так и программного продукта в целом:

Виды используемого тестирования:

- Модульное
- Функциональное

Юнит-тестирование проводится для трёх отдельных модулей:

1. получение времени, необходимого для просмотра сериала
    a. преобразование слова («день», «неделя», «месяц») в количество дней (целое число)
    b. создание строки из числа дней, недель, месяцев, лет
2. проверка на то, что сериал подходит под заданные пользователем критерии
3. сохранение позиции ползунка

Функциональное тестирование проводится непосредственно в веб-приложении. Тестируется работоспособность всех ползунков, выборов из списка и других видов критериев, соответствие предложенных сериалов выбранным критериям, корректность записи времени, необходимого для просмотра сериала.

Подготовка модульных тестовых данных:

Первый модуль:

Преобразование слова в количество дней:

Входные данные	Результат
День	1
Дней	1
Неделю	7
Недель	7
Месяц	30
Месяцев	30

Создание строки из числа дней, недель, месяцев:

Входные данные	Результат
1, 1, 1, 1	1 год 1 месяц 1 неделю 1 день
0, 11, 3, 5	11 месяцев 3 недели 5 дней
4, 2, 6, 3	4 года 2 месяца 6 недель 3 дня
5, 2, 2, 0	5 лет 2 месяца 2 недели

1, 0, 0, 0	1 год

Второй модуль:

Входные данные	Результат
[“фантастика”, “ужасы”], [“Time”, “фантастика”]	False
[“1234”, “qwerty”], [“qwerty”, “1234”, 2]	True
[“1234”, “qwerty”], [“Qwerty”, “12345”]	False

Третий модуль:

На вход поступают положение ползунка, крайняя левая граница и крайняя правая граница. Результатом должно являться положение ползунка относительно данных границ, представленное в виде десятичной дроби, меньшей единицы.

Входные данные	Результат
3, 1, 10	0.2
3, 3, 10	0.0
10, 1, 10	0.9

Технологический раздел

Выбор и обоснование технических средств

Клиентская часть. Для ее реализации наша команда использует фреймворк Bootstrap. Он включает в себя все, что нам могло понадобится для создания презентабельного и удобного интерфейса: HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения. Мы остановили свой выбор именно на этом фреймворке, так как фронтенд-разработчик нашей команды уже имел опыт работы с ним.
Для разработки активных элементов интерфейса, таких как выпадающие списки или ползунки, был использован JavaScript фреймворк jQuery.

Серверная часть веб-приложения реализована на языке программирования Python 3, так как все члены нашей команды хорошо владеют этим языком программирования и при этом не имеют опыта разработки на других языках, широко используемых для написания веб-приложений.

В разработке мы использовали веб-фреймворк Flask. Flask является микрофреймворком - он очень прост и минималистичен, не включает в себя ничего лишнего и предоставляет лишь базовые возможности. Таким образом, мы можем сами выбирать расширения и пакеты под конкретные задачи и устанавливать их по мере необходимости. Это делает Flask довольно гибким и удобным.

В работе мы использовали:

- Расширение **flask_sqlalchemy** (интегрирует в приложение библиотеку SQLAlchemy) - работа с базой данных.
- Библиотека **psycopg2** - адаптация базы данных PostgreSQL под Python.

Для работы с базой данных в нашем проекте используется система управления базами данных PostgreSQL. Это стабильная и надежная СУБД со множеством возможностей. Мы выбрали ее, так как возможно и после окончания учебной практики будем развивать свой проект.

В качестве хостинга выбран Heroku. Heroku — это облачная PaaS-платформа (Platform as a Service — платформа как услуга), поддерживающая ряд языков программирования, включая Python. Мы выбрали эту платформу, так как она популярна, проста в освоении и использовании и имеет бесплатный уровень обслуживания.

Выбор и обоснование модели разработки

Для реализации нашего проекта мы выбрали каскадную модель разработки (Проектирование -> дизайн -> кодирование -> тестирование -> развертывание -> поддержка). Мы выбрали эту модель, так как она хорошо подходит для

небольших проектов и все наши задачи были четко сформулированы до начала работы над приложением.

Реализация программного продукта

Работа с базой данных
При создании проекта использовалась система управления базами данных PostgreSQL, как стабильная СУБД с множеством возможностей. Для администрирования и разработки использовался pgAdmin 4. Изначально база данных создавалась локально, после чего была перенесена на Heroku.

Для предоставления пользователям достаточного объема информации о конкретном произведении кинематографа была взята информация с разных похожих сайтов, но в основном с «Кинопоиска», т.к. на данный момент на этом сервисе можно найти наиболее полную коллекцию сериалов/фильмов и др. на русском языке.

Про каждое произведение известно его оригинальное название, русское название, количество сезонов, общее количество эпизодов, продолжительность каждой серии, год выхода, год окончания (если статус «Завершен»), жанры, страна производства, полное и краткое описания, список актеров, игравших главные роли, рейтинг, статус (Завершен/Выходит), категория (сериал/мультсериал/аниме), а также постер.

Была составлена общая база данных с полной информацией о конкретном сериале, далее с помощью Python 3 были сформированы базы данных по актерам, игравших главные роли, жанрам и странам.

Бэкенд разработка
Проект состоит из нескольких модулей: routes.py, models.py, search.py и time.py.

Модуль routes.py отвечает за генерацию html страниц. Он содержит две функции main и info для генерации главной страницы с поиском сериалов и страницы с подробным описанием сериала соответственно. Также в этом модуле содержится функция pos, которая определяет положение ползунка в зависимости от его значения в виде числа от 0 до 1. Она необходима для сохранения положения ползунка при обновлении страницы. В функции main программа присваивает всем параметрам значения по умолчанию. Далее, если пользователь выполнил поиск, программа получает указанные пользователем параметры. Затем с помощью функции found_serials
создается список подходящих сериалов и словарь с рассчитанным временем просмотра. Функция заканчивает свою работу генерацией html страницы на основе шаблона main.html. Функция info получает на вход id сериала, находит его в базе данных и генерирует страницу на основе шаблона info_page.html.

Модуль models.py содержит классы Serials, Actors, Countries и Genres. Они позволяют обращаться к таблицам базы данных.

Модуль search.py содержит функцию found_serials, которая производит поиск сериалов в базе данных по переданным ей параметрам. Также в этой функции производится расчет времени, которое пользователь потратит на просмотр сериала. Она возвращает список подходящих сериалов и словарь с рассчитанным временем просмотра. В модуле также находятся функции matches и includes, для определения соответствия сериала некоторым параметрам.

Наконец, модуль time.py содержит две функции. Функция get_days преобразует название периода (день, неделя, месяц) в количество дней. А функция make_time преобразует количество лет, месяцев, недель, дней в строку. Например, если функции переданы значения (1, 3, 2, 4), она вернет строку “1 год 3 месяца 2 недели 4 дня”.
Модуль parameters.py не используется в основной программе. Он использовался для автоматического наполнения таблиц баз данных с имеющимися параметрами.

Фронтенд разработка
Для разработки интерфейса было выделено три файла:

- **main.html** отвечает за наполнение главной страницы сайта
- **info_page.html** отвечает за наполнение страницы с информацией о конкретном фильме
- **style.css** отвечает за внешнее представление страниц сайта.

Также были подключены библиотеки, необходимые для работы фреймворка:

- **jQuery**
- **Popper.js**
- **Bootstrap.js**

Страница состоит из двух блоков с классами search-bar и search-rez, что позволяет логически и визуально отделить панель поиска от панели вывода результатов поиска.
Весь раздел параметров поиска включает в себя блок form. Для каждого параметра поиска выделен отдельный блок с классом form-row и группа с классом group-row. Для каждого параметра поиска также есть тег <label>, в котором указывается его название.
Выпадающие списки для расчета времени просмотра были разработаны с помощью кнопок. Также была написана функция, которая обновляет внешний вид кнопки в соответствии с выбранным пунктом и обновляет ее значение.
Для корректной работы ползунков, определяющих значения года выхода, продолжительности серии и количества сезонов, была написана функция twoBombSlider, которая при передвижении ползунка обновляет его внешний вид и значение.

В блоке результатов поиска показываются сериалы, удовлетворяющие параметрам поиска. Каждый сериал оформлен с использованием компонента Bootstrap “card”(в переводе на русский - карта, карточка). В свою очередь карта сериала разделена на 2 составляющие: обложку сериала и краткую информацию о нем.

Реализация тестирования:

Для реализации модульного тестирования фреймворки не использовались. Юнит-тесты писались вручную. Покрытие тестами узнавалось с помощью coverage.

Итог тестирования первого модуля:

Тестирование функции get_days:

Тест 1 прошёл

Тест 2 прошёл

Тест 3 прошёл

Тест 4 прошёл

Тест 5 прошёл

Тест 6 прошёл

Тестирование функции make_time:

Тест 1 прошёл

Тест 2 прошёл

Тест 3 прошёл

Тест 4 прошёл

Тест 5 прошёл

Покрытие:

Name	*Miss*	*Cover*	*Missing*
app\time.py	*2*	*96%*	*15, 37*

Итог тестирования второго модуля:

Тестирование функции matches:

Тест 1 прошёл

Тест 2 прошёл

Тест 3 прошёл

Покрытие:

Name Miss Cover Missing

app\search.py 35 20% 16-61

Итог тестирования третьего модуля:

Тестирование функции pos:

Тест 1 прошёл

Тест 2 прошёл

Тест 3 прошёл

Покрытие:

Name Miss Cover Missing

app\routes.py 79 10% 15-122, 131-132

Такое маленькое покрытие в тестировании второго и третьего модулей обусловлено наличием в них не тестируемых функций (эти функции тестируются потом функциональными тестами). Во втором модуле это функция поиска сериалов, в третьем – основная подпрограмма.

Развертывание разработанного программного продукта

Развертывание производилось на Heroku. Для этого в корневом каталоге был созданы файлы Procfile и runtime.txt. Первый содержит имя процесса и команду, которая запускает процесс. Во втором записана используемая версия языка Python. Также для развертывания потребовалась установка Heroku CLI. Далее мы зарегистрировали наше приложение с помощью команды “heroku apps: create flask-microblogtime-for-series” и загрузили его на сервер командой “ git push heroku develop:master”. Аргумент develop:master означает, что мы переместили код из ветви develop нашего репозитория в главную ветвь master репозитория Heroku. После этого мы убедились, что веб-приложение функционирует корректно, и слили ветвь develop с master.

Заключение

В результате работы над практикой мы выполнили все поставленные задачи. Выбранные технические средства позволили нам полностью реализовать задуманное и разработать веб-приложение для получения пользователем рекомендуемых к просмотру сериалов и расчета времени, которое пользователь потратит на просмотр сериала.
Мы считаем, что разработанное веб-приложение полностью удовлетворяет поставленным задачам. Тем не менее, интерфейс приложения можно доработать с помощью разбивки на страницы и формы ввода для поиска актера.

Список использованных источников

1. Перевод Мега-Учебника Flask Мигеля Гринберга - https://habr.com/ru/post/346306/
2. Овладей Python, создавая реальные приложения - https://medium.com/nuances-of-programming/%D0%BE%D0%B2%D0%BB%D0%B0%D0%B4%D0%B5%D0%B9-python-%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B2%D0%B0%D1%8F-%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F-%D1%87%D0%B0%D1%81%D1%82%D1%8C-4-60e016f18422
3. Документация Flask - https://flask-russian-docs.readthedocs.io/ru/latest/
4. Документация Flask-SQLAlchemy - https://flask-sqlalchemy.palletsprojects.com/en/2.x/