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 **Информатика и системы управления**

КАФЕДРА **Программное обеспечение ЭВМ и информационные технологии**

ОТЧЕТ ПО ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

Команда **Pythlings**
Группа **ИУ7-26Б**

Студент	_____	Казаева Т.А.
	подпись, дата	*фамилия, и.о.*
Студент	_____	Андрич К.
	подпись, дата	*фамилия, и.о.*
Студент	_____	Леонов В.В.
	подпись, дата	*фамилия, и.о.*
Студент	_____	Хрюкин А.А.
	подпись, дата	*фамилия, и.о.*
Студент	_____	Руденко И.А.
	подпись, дата	*фамилия, и.о.*
Ментор	_____	Романов А.В.
	подпись, дата	*фамилия, и.о.*
Руководитель практики	_____	Оленев А.А.
	подпись, дата	*фамилия, и.о.*

2020 г.

Оглавление

Введение

Отмечаемая в течение последних десятилетий в мире тенденция к увеличению интереса к фильмам и сериалам и к росту времени их просмотра говорит о необходимости оптимизации процесса поиска контента для потребления. О существовании подобной проблемы упоминается в статистических исследованиях о процентном соотношении различных активностей человека в течение всей его жизни.

Таким образом, наша команда разработки ставит перед собой цель создать приложение (Telegram – бота), позволяющего сократить время поиска фильма или сериала.

Распределение ролей в команде разработки:

1) Казаева Татьяна – тимлид, разработчик, дизайнер

2) Катарина Андрич – разработчик.

3) Леонов Владислав – разработчик, подготовка презентационных выступлений, дизайнер.

4) Хрюкин Арсений – разработчик.

5) Руденко Илья – тестировщик.

Структура отчета:

1) Казаева Татьяна(стр.11-12)

2) Катарина Андрич (стр.8-9)

3) Леонов Владислав (стр.3-6)

4) Хрюкин Арсений (стр.7-8)

5) Руденко Илья (стр. 1-2, 13)

Аналитический раздел

Безусловно, наша команда разработки не первая пытается решить поставленную проблему, однако следует отметить, что существующие на рынке игроки не могут в полной мере помочь рядовому потребителю с быстрым и качественным поиском фильма для развлечений.

Кинопоиск, IMDb и Rotten Tomatoes предоставляют слишком широкий выбор кинофильмов и сериалов. Согласно исследованиям Барри Шварца, неожиданно потребители сталкиваются с ситуацией, когда большой выбор требует напряжения при принятии решения, причем время на совершение конкретной покупки значительно увеличивается, в результате у человека остается неудовлетворенность в своем приобретении. Люди сомневаются в оптимальности сделанного выбора и часто разочаровываются в своих действиях. Более того, последние два сервиса доступны только на английском языке, что может стать камнем преткновения для довольно широкой прослойки общества.

Следует отметить, что данные сервисы предоставляют большое количество дополнительной информации о кинолентах, что в большом количестве случаев не нужно обычному потребителю.

На поприще Telegram-ботов можно выделить бота Киноман, но он предоставляет пользователю лишь подборки кинофильмов и ничего более.

Итак, мы считаем, что идеальный сервис должен включать следующие аспекты:

1) круглосуточное функционирование без перебоев;

2) предоставлять потребителю наиболее качественные фильмы и сериалы;

3) иметь простой и удобный пользовательский интерфейс;

4) предоставлять списки рекомендаций от экспертов;

5) быть доступным широкой аудитории.

Конструкторский раздел

Декомпозиция задачи

Для оптимизации процесса реализации сервиса были выделены следующие подзадачи, требующие последовательного выполнения:

1. Создание интерфейса взаимодействия пользователя с Telegram – ботом.
2. Создание алгоритма поиска по ключевым словам.
3. Создание базы данных пользователя.
4. Обеспечение взаимодействия базы данных и Telegram – бота.
5. Обеспечение взаимодействия алгоритма поиска и Telegram – бота, основываясь на информации из базы данных.
6. Процесс тестирования и отладки ПО.

Разработка структуры создаваемого программного продукта

Для реализации данного программного продукта, решающего обозначенные задачи, следует выделить следующие компоненты:

1. **Создание алгоритма функционирования** бота Telegram.
2. **Создание базы данных пользователей** продукта.
3. **Создание алгоритмов поиска** по магазинам цифровой дистрибуции**.**

Описание назначения, требований к выделенным компонентам и их интерфейса.

Алгоритм функционирования бота

Служит для обеспечения взаимодействия сервиса и пользователя, а также должен удовлетворять технологическим стандартам, чтобы сделать возможными продажу продукта сторонним студиям разработки и возможность быстрой модификации и оптимизации продукта, должен быть надежным и не иметь непредвиденного поведения.

База данных пользователей

Служит для хранения информации и реализации программных особенностей сервиса и возможности дальнейшего расширения функционала приложения, путем внедрения алгоритмов индивидуальной подборки контента, учитывая историю запросов и просмотренных фильмов. Информация в базе данных должна быть оптимизированной и защищенной от сторонних пользователей.

Алгоритмы поиска

Пользователь выбирает критерий и вводит ключевое слово(а), алгоритм поиска в свою очередь служит для предоставления пользователю наиболее выгодных предложений для покупки. Алгоритмы поиска должны обеспечивать максимальное быстродействие и удовлетворять запросы пользователя.

Разработка алгоритмов и структур

Чтобы обеспечить возможность поиска по разным критериям было необходимо создать следующие функциональные модули:

- MostPopular – несколько наиболее популярных фильмов.
- RandomMovie – подборка случайного фильма для просмотра.
- SearchByActor – поиск фильмов с участием заданного актера.
- SearchByDirector – поиск фильмов с участием заданного режиссера.
- SearchByGenre – поиск фильмов в заданном жанре.

- SearchByName – поиск фильмов с заданном названием.
- SearchByYear – поиск фильмов по заданному году выпуска.

Для взаимодействия алгоритма Telegram-бота с алгоритмами поиска была выделена следующая структура данных State, содержащая следующие поля:

- user_id – уникальный идентификатор пользователя Telegram.
- state – текущее состояние пользователя.
- mode – информация о выбранном режиме поиска (фильм/сериал)

Проектирование пользовательского интерфейса

В данном случае интерфейс представляет собой диалоговое окно Telegram-бота, включающего в себя различные команды и графические элементы (кнопки управления). Telegram был выбран за счет своей кроссплатформенности и доступности. Кроме того, у Telegram реализован удобный интерфейс для интеграции в любую систему с доступом к сети интернет.

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

Так как самой важной частью являются алгоритмы поиска, следственно на тестирование данного компонента был сделан основной упор. Учитывались разные варианты написания текста (разные регистры). Было проведено ручное тестирование под руководством Хрюкина Арсения.

В дальнейшем планируется создание собственных алгоритмов тестирования для автоматизации.

Технологический раздел

Выбор и обоснование технических средств

Для реализации нашего сервиса был использован следующий технологический стек:

1. **Язык программирования Python** был использован в виду опыта разработки на нем у всех членов команды, легкого и понятного синтаксиса языка, большого количества библиотек и широкой доступности качественных Интернет-ресурсов о документации и внедрении технологий.
2. **Библиотека PyTelegramBotAPI** ЯП Python – официальная библиотека Telegram, дающая возможность легкого создания Telegram ботов. Регулярные обновления и большой функционал позволяют обеспечить все потребности разработчиков программного обеспечения.
3. **MongoDB** — документо ориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++. Была выбрана ввиду простого интерфейса взаимодействия, наличия качественной документации, быстрой работы, регулярных обновлений и широкого спектра технологических возможностей для разработчиков.
4. **Библиотека mongoengine** ЯП Python — для предоставления разработчикам инструмента взаимодействия с базой данных, используя язык программирования.
5. **Библиотека Selenium WebDriver** ЯП Python — инструмент для автоматизации действий веб-браузера. использован для решения задач регулярного получения данных из различных источников (сайтов).
6. **СУВ GitLab** — веб-инструмент жизненного цикла DevOps с открытым исходным кодом. Ввиду гибких возможностей управления репозиториями кода для Git с собственной вики, системой отслеживания ошибок, простым

интерфейсом и опыта взаимодействия всех членов команды с ним был сделан выбор в его пользу.

Выбор и обоснование модели разработки

Каждому члену команды разработки отводилась определенная роль - разработчик, тестировщик, дизайнер и так далее. Разработчикам было поручено сделать определенную часть проекта, после завершения работы над ней данные отправлялись в репозиторий. Ментор оценивал проделанную работу, обсуждал неточности и советовал как более грамотно оформить ту или иную часть подзадачи. Тестировщик проверял работоспособность выполненной части и докладывал об ошибках разработчику.

Для работы с репозиторием была использована модель git-flow. Она используется во многих крупных компаниях. Модель ветвления git-flow формирует понимание процессов ветвления и слияния, достаточно несложна в эксплуатации и не требует большого количества времени на изучение.

Реализация программного продукта

Изначально бот был запущен на локальной рабочей станции, без участия базы данных. Были разработаны диалог с пользователем и система поиска. Далее была встроена база данных, система поиска была внедрена в основную структуру бота.

Реализация тестирования

Тестирование было разделено на следующие блоки:

1. **Модульное тестирование (unit testing).**

   Проверялись отдельные объекты, функции и модули, такие как, например, алгоритм поиска.

2. **Функциональное тестирование.**

   Данный тип тестирования проводится для всей системы в целом. В основном, воспроизводились различные модели диалога с

пользователем для проверки корректного перехода программы с одного декоратора на другой.

Тестирование проводилось в ручном режиме. специалист по тестированию самостоятельно проходил весь контрольный сценарий тестирования.

Некоторые моменты были указаны разработчиками в отчетах об ошибках в GitLab. Тестировщик целенаправленно искал ошибки, полученные из информации в отчетах.

Срок тестовой эксплуатации не превысил одной недели.

По результатам тестирования был составлен перечень замечаний. Тестировщик его предоставил каждому из разработчиков.

Развертывание разработанного программного продукта

Развертывания данных продукт не требует. Для начала работы с ним необходимо запустить бота и начать с ним общение.

Заключение

Выполненная командой практическая работа, безусловно, позволила нам приобрести некоторые навыки, которые возможно поделить на два блока.

Hard skills (досл. “жесткие” навыки) - профессиональные навыки, которым можно научить и которые можно измерить. Команда впервые столкнулась со следующими задачами:

- Управление интернет - браузером через ПО;
- Программирование диалога с телеграм - ботом;
- Использование базы данных;
- Использование системы ветвления git-flow.

Soft skills (досл. “мягкие” навыки) - универсальные компетенции, которые гораздо труднее измерить количественными показателями. Иногда их называют личными качествами. Для нашей команды работа в таком формате была осуществлена впервые. Работа в команде позволила приобрести многие навыки, которые при индивидуальной разработке не используются.

Для лидера группы важно:

- Грамотно распределить обязанности, оценивая способности и желания каждого участника;
- Разрешать конфликты, происходящие в команде;
- Информировать участников об изменениях календарного плана и некоторых особенностей разработки своевременно;
- Осуществлять контроль над проделанной работой.

Для разработчика важно:

- Уважать труд другого разработчика;
- Сфокусироваться на читаемости своей части проекта, сделать ее удобной для интеграции;
- Уметь подстраиваться под ситуацию и оказать помощь другим участникам команды.

Для тестировщика важно:

- Конструктивно составлять отчеты об ошибках;
- Обеспечить результат своей работы в понятном для всех участников формате;
- Обеспечить подробность результатов работы таким образом, чтобы разработчик точно знал, какая часть проекта нуждается в доработке.

Команда не сталкивалась с трудностями в плане коммуникации. Атмосфера была дружелюбной и уважительной, в ней было приятно работать.

С техническими трудностями наша команда, столкнулась при реализации возврата в предыдущий пункт меню.

Проблема была решена участником команды, заменив inline - кнопку на команду /back. Такая модель поведения служит примером приобретения “мягких” и “жестких” навыков - умение подстраиваться под ситуацию и решать проблемы эффективно и знание возможностей модулей, использованных при разработке.

В общем, команда не только приобрела полезные для дальнейшей работы в этой области знания, но и получила удовольствие от проделанной работы и полученного результата.

Список использованной литературы

1. Документация Telegram [Электронный ресурс]. – Режим доступа: https://tlgrm.ru/docs.
2. Документация Telegram: Роботы [Электронный ресурс]. – Режим доступа: https://tlgrm.ru/docs/bots.
3. Каталог ботов для Telegram [Электронный ресурс]. – Режим доступа: http://www.telegrambots.info/.
4. Документация mongoengine [Электронный ресурс]. – Режим доступа: http://docs.mongoengine.org/
5. Удачная модель ветвления для Git [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/106912/
6. Шпаргалка по git-flow [Электронный ресурс]. – Режим доступа: https://danielkummer.github.io/git-flow-cheatsheet/index.ru_RU.html
7. Документация Selenium [Электронный ресурс]. – Режим доступа: https://selenium-python.readthedocs.io/