Введение

Вы почти окончили курс и дошли до его завершающей части. Позади много пройденной теории и практики. Мы гордимся вашими трудолюбием и упорством! Осталось совсем немного — выполнить **финальную работу.**

Кто-то считает её формальностью, но давайте разберём, почему она нужна.

Во-первых, она станет первым кейсом для вашего портфолио, если вы только начинаете в профессии. Если продолжаете — выгодно его дополнит.

Во-вторых, работа наглядно показывает уровень вашего кода. Глядя на неё, опытный разработчик оценит ваши навыки построения архитектуры приложения, code style, в том числе нейминг, оформление, читаемость, гибкость. Это может стать поводом для первого собеседования. Всё в ваших руках.

В-третьих, Телеграм-боты набирают популярность и часто используются для автоматизации рутинных процессов. Они востребованы. Вы можете заложить базу для развития будущих коммерческих проектов, работая с ними уже сейчас. Возможно, вы оставите своего первого бота как pet-проект. Кто знает?

Поэтому давайте соберёмся и приступим. Будет интересно!

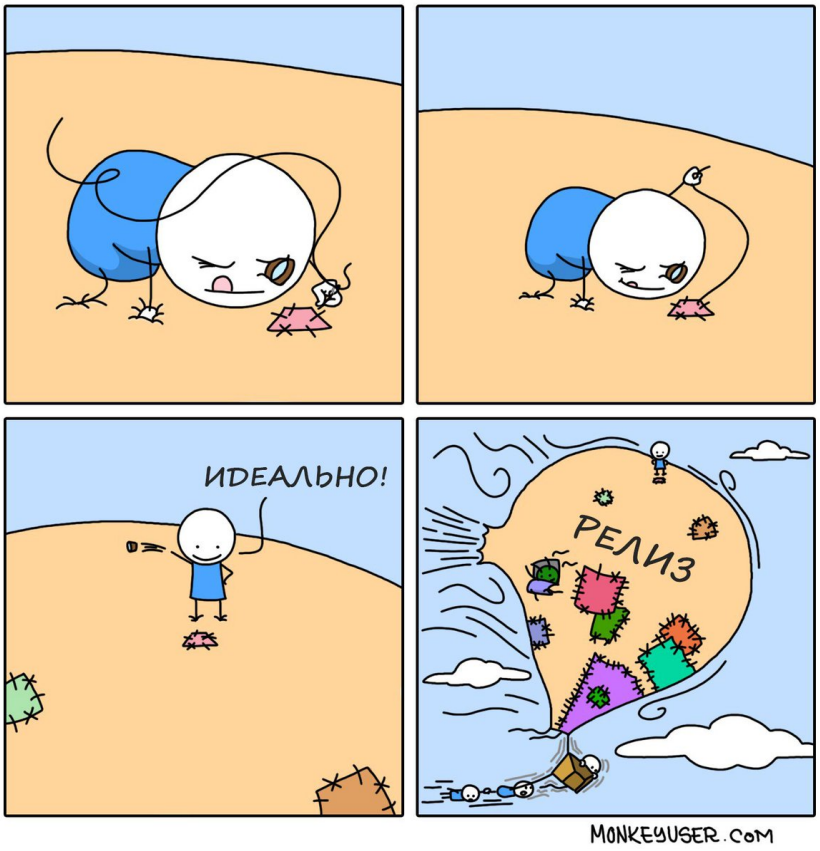
Описание проекта

Тема финальной работы

Telegram-бот для работы с API стороннего сайта

Мы собрали основную информации, чтобы вам было легче приступить к его выполнению.

Изучайте этот материал с карандашом в руках. Делайте пометки на полях, выписывайте возникающие во время чтения вопросы, рисуйте блок-схемы, если необходимо. Словом, адаптируйте под себя.



Тема финальной работы

Телеграм-бот для работы с API стороннего сайта.

Описание задачи

К вам обратился заказчик, который хочет вести бизнес в интернете. Для этого ему нужен первоклассный Телеграм-бот, но бюджет скромный и не соответствует его запросам.

После обсуждений вы предложили универсальное решение: написать бота, у которого можно менять модуль работы с внешним API и при этом не трогать остальные модули.

У некоторых сайтов есть открытый (бесплатный!) API, над которым уже работали другие программисты, а это значит, что на них можно запустить ваши разработки.

ТЗ на проект

Проект должен быть разработан на языке Python, быть портируемым. Это значит, что он запускается с помощью клонирования репозитория и установки необходимых

библиотек (pip install -r requirements.txt). Подразумевается, что Телеграм у заказчика уже есть.

Для разработки проекта используется любой открытый API. Обратите внимание, что некоторые API имеют ограничения на количество запросов, поэтому при выборе тематики бота обязательно изучите нюансы используемого API.

Обязательно согласуйте с куратором тематику и функционал до написания кода.

Например, у rapidapi.com при базовом пакете есть ограничение по количеству запросов в месяц — 500. Проблема нехватки запросов в данном случае решается дополнительной регистрацией одного или нескольких аккаунтов.

Проект состоит из нескольких функциональных слоёв: скрипта main.py, модуля, отвечающего за работу с Telegram, модуля, отвечающего за работу с API стороннего сайта, и модуля, работающего с БД или лог-файлами.

Пользователь с помощью специальных команд бота может сделать следующее:
1. Запросить минимальные значения (команда /low).
2. Запросить максимальные значения (команда /high).
3. Запросить диапазон значений (команда /custom).
4. Узнать историю запросов (команда /history).

Без запущенного скрипта бот на команды (и на что-либо ещё) не реагирует.

Описание работы команд

Команда /low

После ввода команды у пользователя запрашивается:
1. Услуга/товар, по которым будет проводиться поиск (самая низкая стоимость, самые доступные авто, самое близкое местоположение и так далее).
2. Количество единиц категории (товаров/услуг), которое необходимо вывести (не больше программно определённого максимума).

Команда /high

После ввода команды у пользователя запрашивается:
1. Услуга/товар, по которым будет проводиться поиск (самая высокая стоимость, самые дорогие авто, самое удалённое местоположение и так далее).
2. Количество единиц категории (товаров/услуг), которое необходимо вывести (не больше программно определённого максимума).

Команда /custom

После ввода команды у пользователя запрашивается:
1. Услуга/товар, по которым будет проводиться поиск.

2. Диапазон значений выборки (цена от и до, расстояние от и до, срок от и до и так далее).
3. Количество единиц категории (товаров/услуг), которые необходимо вывести (не больше заранее определённого программно максимума).

Команда /history

После ввода команды выводится краткая история запросов пользователя (последние десять запросов).

Описание внешнего вида и UI

Окно Телеграм-бота при запущенном Python-скрипте должно воспринимать команды:

- /help — помощь по командам бота;
- /low — вывод минимальных показателей (с изображением товара/услуги/и так далее);
- /high — вывод максимальных (с изображением товара/услуги/и так далее);
- /custom — вывод показателей пользовательского диапазона (с изображением товара/услуги/и так далее);
- /history — вывод истории запросов пользователей.

Сообщение с результатом команды должно быть содержательным.

Технические требования

- Скрипт для Телеграм-бота должен быть написан с использованием библиотек для работы с различными API и запросами.
- Бот должен поддерживать многопользовательский режим.
- Запуск бота должен выполняться командой *python main.py* из терминала — из папки с проектом. Реализация main.py и остальных файлов проекта остаётся за вами. Файлы не должны содержать ошибок, работа должна быть корректной.
- Команды бота должны работать в соответствии с их описанием в ТЗ. Выдача результата производится по соответствующей команде Телеграм-бота.
- Интерфейс должен быть отзывчивым — при возникновении пользовательских ошибок (ввод несуществующих команд, ввод данных неверного типа и так далее) выводится соответствующее уведомление.
- К скрипту должен быть приложен файл readme.md, который содержит инструкцию (документацию) для работы со скриптом и пользователем.

Формат сдачи материалов и оценивание

Используйте для проекта отдельный Git-репозиторий в gitlab.skillbox.ru.

Этапы разработки проекта:

1. Создание бота, который реагирует на команду /hello-world, а также на текст «Привет» (необходимо сообщить проверяющему куратору имя вашего бота для тестирования).

2. Реализация команды /low.
3. Реализация команды /high.
4. Реализация команды /custom.
5. Реализация команды /history.
6. Сдача проекта.

Создайте на каждом этапе свой Merge Request (MR) на ветку master, пришлите его куратору для код-ревью. Куратор оставит замечания или одобрит этот MR. В случае одобрения запрос сольётся в ветку master.

Советуем сделать первый MR с общей структурой проекта, в которой каждая команда Телеграм-бота вынесена в отдельный файл, чтобы в дальнейшем можно было создавать несколько параллельных MR под разные функционалы из ТЗ. Это позволит вам, не ожидая одобрения куратора, приступать к выполнению следующей задачи.

Таким образом, на этапе «Сдача проекта» вы сообщаете куратору о завершении проекта, в котором уже реализован весь необходимый функционал программы в соответствии с техническими требованиями.

Проект оценивается по следующим критериям:

- Следование техническому заданию. Элементы должны работать в соответствии с описанием.
- Скрипт запускается без ошибок, Телеграм-бот работает корректно.
- Отсутствие ошибок в результатах каждой реализованной команды.
- Usability приложения — отзывчивость интерфейса. Если есть ошибки при вводе данных, то пользователь получает соответствующее уведомление.
- Чистый Python-код: соблюдение правил PEP8, описанная документация для классов и функций, наличие аннотаций типов, совместимость с Python 3.9.
- Наличие файла readme.md, в котором есть инструкция по эксплуатации скрипта и бота для пользователя.

Проект возвращается на доработку, если:

- Отсутствует или не работает базовая функциональность, описанная в ТЗ..
- Есть грубые ошибки в организации или структуре проекта.
- Куратор не может запустить проект локально.

Дополнительные материалы (подсказки)

- «telebot быстро и понятно. Телеграмм-бот» — статья о библиотеке telebot.

- «Пишем ботов для Telegram на языке Python» — статья о создании ботов и о написании для них программ.

Инструкции

Референсы

Примеры чат-ботов для бизнеса.

API и документация к ним

- Выбирайте API на свой вкус.
- Документация API Telegram bots.
- Популярные API Telegram: aiogram, python-telegram-bot, PyTelegramBot.

Рекомендации по выполнению

- Внимательно изучите документацию используемых API (раздел endpoints, tutorials, pricing выбранного API). Протестируйте API на простых данных и проверьте корректность результата вручную на сайте.

- Сделайте самого простого бота, реагирующего на одну-две команды, и добавьте короткий текст (например, «Привет»).

- Чтобы было легче выполнять проект, рекомендуем разбить работу над ним на части. Например, сначала — реализация первой команды и получение результата в самом скрипте. Затем — вывод этого результата по команде бота и так далее. Часть может считаться готовой, если вы чувствуете, что готовы показать её заказчику. В вашем случае роль заказчика исполняет куратор.

- Составьте план работы по датам. Исходя из нашего опыта, продуктивно работать над финальным проектом по два-три часа несколько дней в неделю, чем делать этот же объём работы за один подход. Рекомендуем придерживаться этого графика и выделять время для отдыха.

- Отмечайте прогресс по мере выполнения плана. Это полезно по нескольким причинам: во-первых, вы будете держать ритм, во-вторых — контролировать ситуацию. И самое главное: каждый выполненный этап — это ваша маленькая победа. Чем больше таких побед вы будете отмечать, тем больше удовольствия вы получите от выполненного проекта.

- Прежде чем отправлять каждую часть проекта куратору (а в реальной работе — тестировщику), уделите время на проверку себя: посмотрите ещё раз кодом и результат, который он даёт. Идеально, если между завершением работы и проверкой себя пройдёт некоторое время — оно позволит вам переключиться и посмотреть на работу свежим незамыленным взглядом.

- Качество продукта — это не метрика, а личное отношение разработчика. Пригласите нескольких друзей или коллег протестировать проект вместе с вами. Обращайте внимание на их поведение. Попросите проговаривать мысли, действия и ощущения. Задавайте наводящие вопросы. Вы можете заранее

подготовить чек-лист, отталкиваясь от функциональных требований. Всё это поможет вам понять, в чём и на каких этапах у пользователя могут возникнуть проблемы, и сделать действительно удобный интерфейс.

Будет плюсом, если вы используете:

- БД (SQLite3 или Redis) + ORM (например, peewee, pony).
- Логирования (БД или лог-файл json; например, посмотрите в сторону loguru).
- Стратегии backoff.
- Парадигмы ООП + паттерны проектирования.
- Интерактивный функционал (игру, викторину, тесты или любой на ваш выбор).
- Линтеры (например, flake8 + wemake-python-styleguide или https://google.github.io/styleguide/pyguide.html).
- Проверку аннотации с помощью mypy.