**Telegram Travel BOT (Телеграмм бот для путешествий)**

Пояснительная записка

Наш проект – это телеграмм бот для путешествий, разработанный на языке программирования Python. Бот предоставляет ряд полезных функций, которые помогут путешественникам на любом этапе их путешествия: создание напоминаний, перевод иностранных слов, информация о курсах валют и погоде, а также другие возможные функции, направленные на улучшение опыта путешественников.

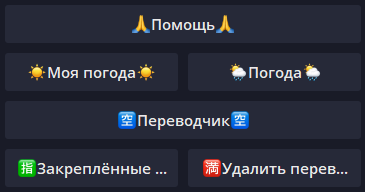
Реализованные функции бота:

1. Возможность перевода текста любого объёма (поддержка 10 наиболее популярных языков);
2. Возможность узнать погоду в городе, в котором находится пользователь в данный момент, при помощи геолокации (встроенной функции телеграмма);
3. Возможность узнать погоду в любом другом городе;
4. Сохранение важных (часто используемых) переводов в базу данных, чтобы после вывести все их одним сообщением;
5. …

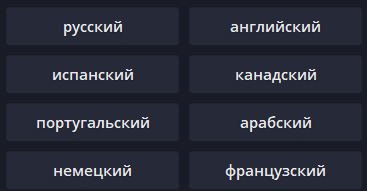
Используемые библиотеки и API:

* Python-telegram-bot (официальный API от телеграм)
* OpenWeather (API получения данных о погоде в конкретном городе)
* Geocoder (API для расшифровки координат)
* Googletrans (API гугл-переводчика)
* Math (для упрощения работы с математическими моделями)
* Datetime (для синхронизации со времением)
* Requests (для получения корректных данных)
* Sqlalchemy (работа с базой данных)
* Dotenv (для скрытия токенов API из публичного доступа)
* sys (для корректной работы приложения и работой с кодами возврата)
* os (для корректной работы приложения с операционной системой)

Карта интерфейса:











**Структура кода/исполнение функций**

Условно весь код можно разделить на 3 части:

1. Объявление констант (токенов API), создание рабочих словарей, основных функций;
2. Реакция бота на триггеры (команды, контекст сообщений и тд);
3. Главный цикл бота.

Объявление констант (токенов API), создание рабочих словарей, основных функций:

Самым первым действием с помощью библиотек dotenv, os и sys загружаются токены API (рис. 1.1) из текстового файла .env, чтобы при загрузке файлов проекта на GitHab, посторонние личности не могли их видеть.

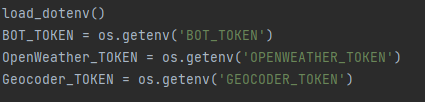


Рисунок 1.1. Загрузка токенов.

Далее задаются словари, а также инициализируется класс переводчика (рис. 1.2). В словаре code\_to\_smile закладываются ответы для расшифровки json с данными по погоде, а в словаре LANGUAGES – расшифровки языков для дальнейшего использования.

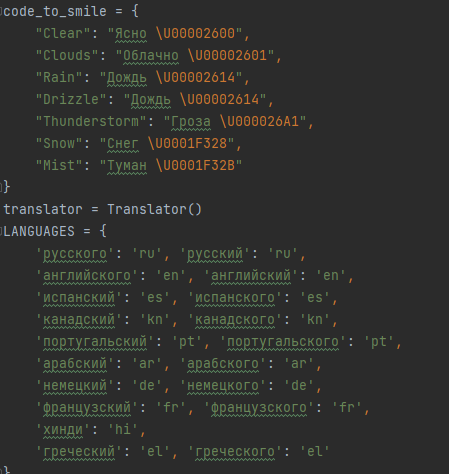


Рисунок 1.2. Инициализация словарей и переводчика.

Далее идёт создание клавиатур интерактивных с кнопками, упрощающих использование бота(рис. 1.3).

Первая используется для создания меню, в котором используется команда «Помощь», для предоставления всей необходимой информации по боту, а также команды вызова функций, среди которых выделяется отдельная кнопка wether\_button (или «Моя погода»), которая за счёт включённого параметра request\_location, отправляет боту геопозицию.

Вторая клавиатура используется для переводчика. На ней располагаются 10 поддерживаемых языков, а также кнопка отмены, прерывающий «диалог».

Третья клавиатура используется в тех случаях, когда «диалог» можно только остановить.



Рисунок 1.3. Функции клавиатур.

Также в этом блоке находится функция wether, в которую передаётся аргумент city – название города (рис. 1.4). Данная функция, используя API OpenWether, отравляет специальный запрос и получает документ json, содержащий сведения о погоде в городе, который являлся аргументом. Затем некоторые данные этого документа, а именно температура, давление, погода, влажность, скорость ветра и длительность светового дня (разница между началом светового дня и его концом) возвращаются в качестве текста.

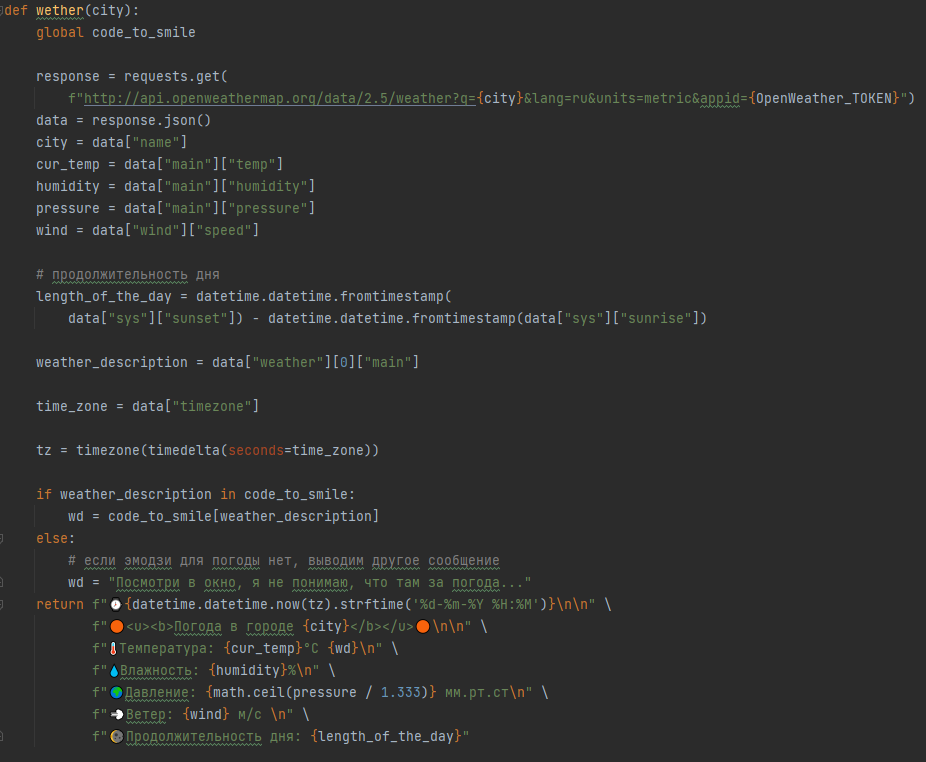


Рисунок 1.4. Функция wether.



Рисунок 1.5. Пример ответа от бота, с помощью функции wether.

Реакция бота на триггеры

В этом блоке располагаются исключительно асинхронные функции, используемые непосредственно ботом для ответных действий на команды, контекст сообщений пользователя и прочие триггеры.

Первые две команды являются уже стандартными для каждого бота в телеграмме: /start (функция start) и /help (функция help\_command) (рис. 2.1). Они просто рассказывают пользователю о функционале, иногда даже с примерами (рис. 2.2 и рис. 2.3).

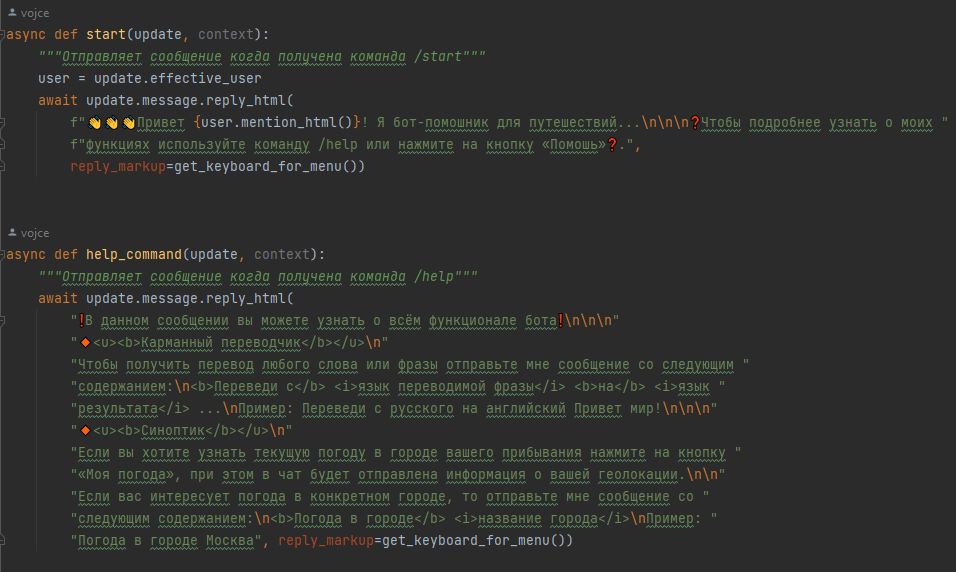
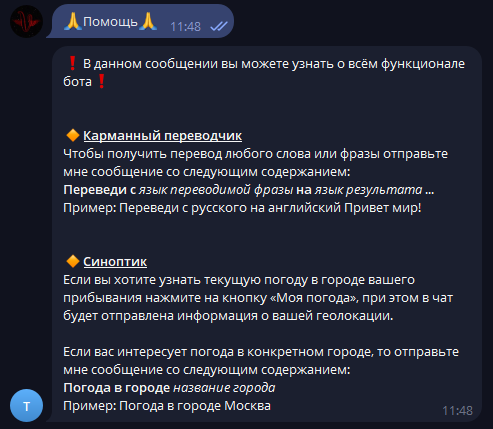


Рисунок 2.1. Исполнение команд /start и /help.



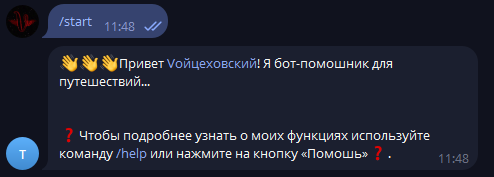


Рисунок 2.2. Ответ на команду start Рисунок 2.3. Ответ на команду help 88888888888888888888888888888888888888888888 8(«Помощь»)

Далее идёт функция my\_wether (рис. 2.4), которая вызывается отправлением боту геопозиции (по спец. кнопке). С помощью API Geocoder, она узнаёт город, в котором находится пользователь и передаёт это значение в функцию wether, а после выдаёт получившийся результат.



Рисунок 2.4. Функция my\_wether.

Далее идёт «диалог» с переводчиком (рис. 2.5), который включает в себя 4 основные функции:

1. translation – просит пользователя ввести текст для перевода;
2. start\_language – принимает текст для перевода и записывает его в словарь с временными данными пользователя, просит выбрать на клавиатуре языков язык, с которого будет производиться перевод;
3. end\_language – принимает начальный язык и записывает выбор в словарь, просит выбрать тем же способом и конечный язык;
4. end\_translation – выводит результат с помощью библиотеки googletrans, спрашивает пользователя о намерении сохранить перевод;
5. seve\_translation – сохраняет перевод в базу данных, с помощью ORM модели (рис. 2.6), завершает «диалог».

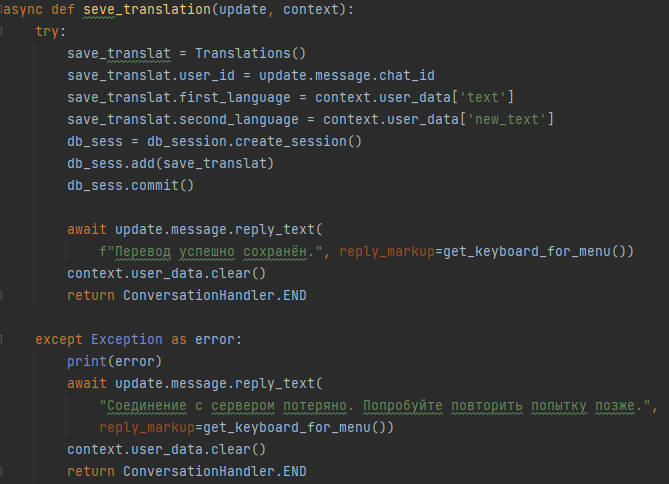
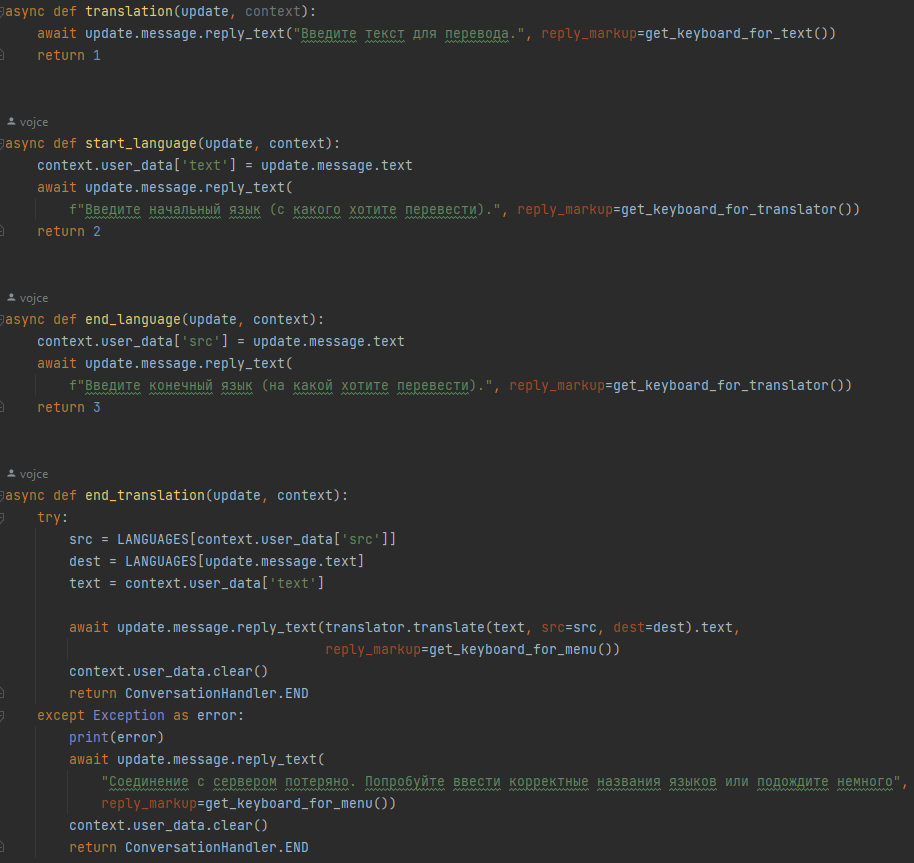


Рисунок 2.5. «Диалог» с переводчиком. Рисунок 2.6. Взаимодействие с ORM 999999999999999999999999999999999999999999999999999моделью

После идут функции для взаимодействия с вышеупомянутой базой данных с сохранёнными переводами при помощи ORM модели (рис. 2.7):

1. fixed\_translations – перебирает все строки базы данных, где есть id пользователя и выводит результат в сообщении;
2. clian\_translations – перебирает все строки базы данных, где есть id пользователя и удаляет их.

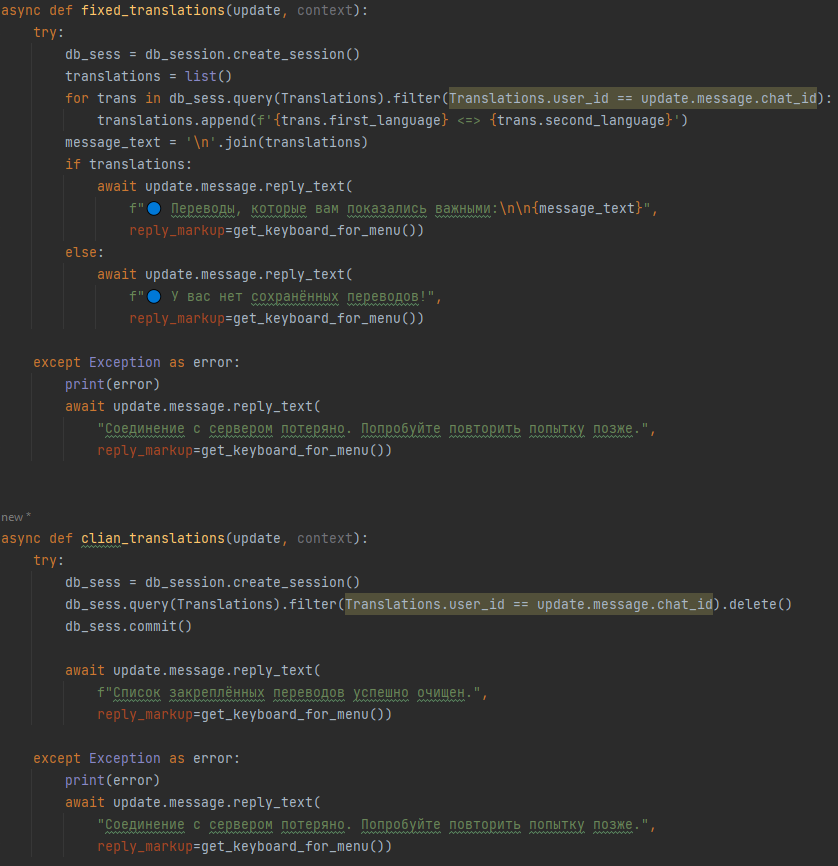


Рисунок 2.7. fixed\_translations и clian\_translations

Далее идёт «диалог» с синоптиком (рис. 2.8), состоящий из двух функций:

1. city\_wether – запрашивает у пользователя название города;
2. end\_wether – выводит результат функции wether по аргументу, введёным пользователем.

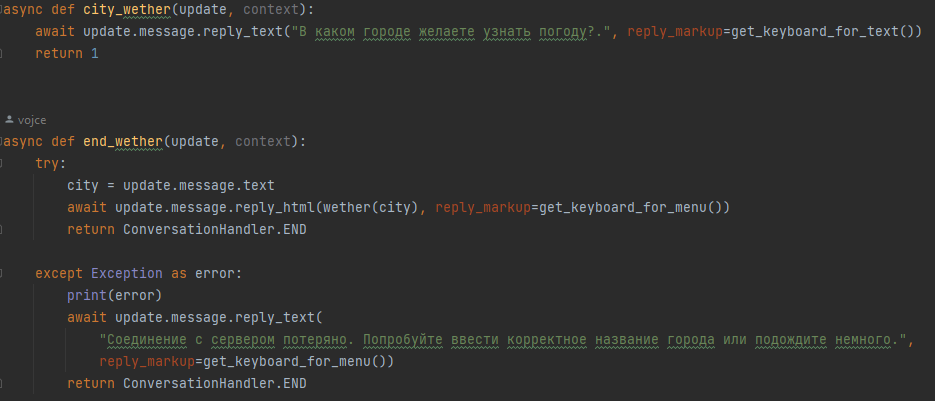


Рисунок 2.8. «Диалог» с синоптиком.

Подытоживает два «диалога» функция stop (рис. 2.9), которая при использовании позволяет выйти из диалога.

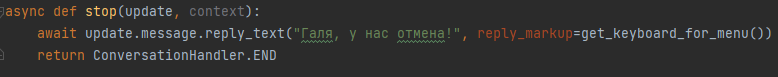


Рисунок 2.9. Отмена «диалогов».

Завершает данный блок функция, работающая с триггерами в сообщениях (рис. 2.9), то есть вызывает функцию перевода или функцию вывода погоды, если видит в сообщении пользователя фразы “Переведи” или “Погода в городе” соответственно.

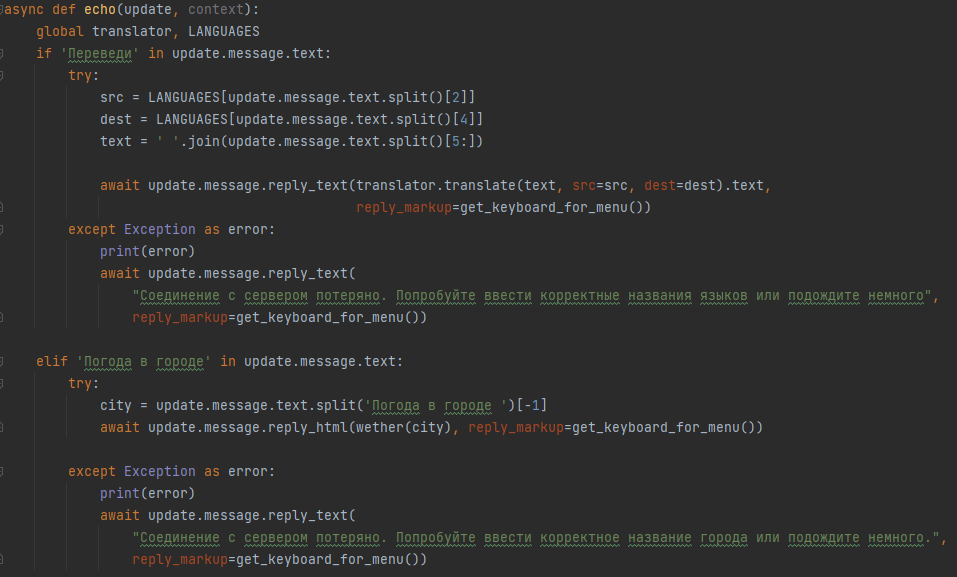


Рисунок 2.9. Работа с триггерами в сообщениях пользователя

Главный цикл:

В главном цикле бота (рис. 3) происходит инициализация самого бота по токену, а также все ранее описанные функции получают все свои триггеры:

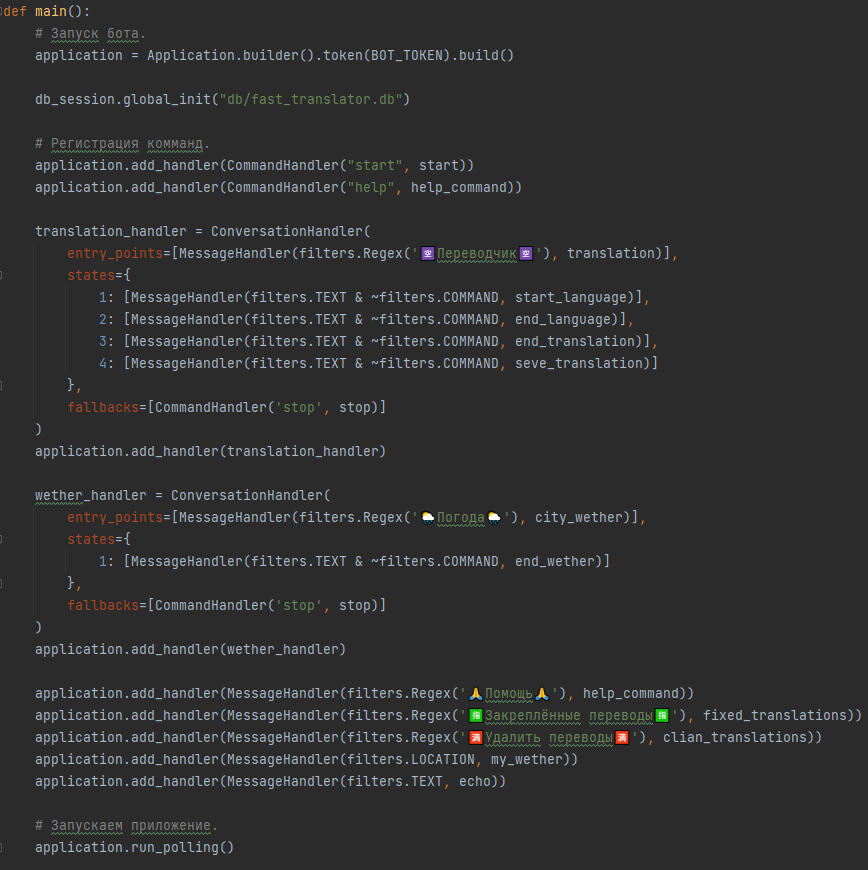


Рисунок 3. Главный цикл бота.

ORM модель

Как уже было сказано выше, бот использует базу данных с помощью ORM модели, для корректной работы которой используется ещё 3 файла с кодом: \_\_all\_models.py (перечень моделей), db\_session.py (создание функций, для работы с моделью), translations.py (сама ORM модель/её класс Translations), а также сама база данных fast\_translator.db.