**Пояснительная записка**

**к проекту команды «СССР»**

**CBbot.**

1. Папка «contants».

1.Файл «exceptions.py».

Содержит в себе классы исключений, которые были написаны нами для более комфортной классификации и отлова ошибок.

2. Файл «keyboards.py».

Представляет собой собрание всех клавиатур и кнопок для них. В файле они классифицированы по операциям, для которых они используются. Было принято решение писать клавиатуры именно в python-файле, а не json, т.к. это повысило скорость импорта и понимание кода.

3. Файл «phrases.py».

Это полноценные функции для генерации текста сообщений или фразы, которые из-за частого использования было неудобно помещать в файл static/json/phrases.json. Исходя из названия, в файле находятся фразы, а именно функции для генерации таких фраз(например, требующих имени пользователя) и т.п.

4. Файл «states.py».

Собрание так называемых States - специальных объектов от модуля AioGram, созданных для ведения диалога с пользователем.

1. Папка «data» - просто объекты для работы с БД. Далее будет описание используемой СУБД и моделей.
2. Файл «db\_session.py».

Представляет собой простую процедуру подключения к базе данных. В качестве СУБД мы использовали PostgreSQL, а не SQLite, т.к. на сервере, на котором в данный момент находится наш бот, данные диска стираются каждые 24 часа, что было бы критично для базы данных SQLite. Наша же база данных лежит на специальном сервере от Amazon и доступна по ссылке, что делает хранение данных безопаснее.

1. Файл «doing\_diagramm.py».

Модель для хранения информации о том, какую диаграмму хочет сделать пользователь. Хранилище данных в оперативной памяти от aiogram(MemoryStorage) не подходило нам, т.к. хранило данные только для одного пользователя, поэтому для нормальной работы было принято решение сделать модель.

1. Файл «user.py».

Обычная модель пользователя. Хранит в себе электронную почту, id Telegram, статус подписки на новости и номера привязанных к аккаунту криптовалютных адресов(хранится только публичная часть адреса, что делает хранение данных безопасным).

1. Файл «verification.py».

Модель для хранения кодов для подтверждения email. Было решено использовать БД, а не MemoryStorage, в силу большей надежности.

1. Файл «waiting\_for\_money.py».

Модель для хранения кодов покупки криптовалют(созданы для верификации пользователя, который покупает). Хранит в себе сам код, id пользователя Telegram и название покупаемой криптовалюты для проверки всех данных при совершении покупки.

1. Папка «modules» - это 5 классов, написанных для обеспечения бота его заявленными и дополнительными функциями.

1. Файл «crypto\_operations.py».

Содержит в себе класс CryptoOperating. Класс позволяет: генерировать криптовалютные кошельки с помощью seed-фраз и библиотеки PyWallet, проверять на существование криптовалютные кошельки, проверять баланс кошельков, проверять статус транзакций в сетях различных криптовалют и отправлять транзакции с помощью API сервиса BlockCypher.

2. Файл «email\_operations.py».

Класс EmailOperations. Позволяет проводить различные операции с протоколом электронной почты SMTP, а именно:

Метод «send\_authorization\_message» - отправляет сообщение с кодом авторизации на необходимый email-адрес.

Метод «send\_buy\_info» - отправляет на необходимый адрес сообщение с чеком и ID транзакции в сети криптовалюты, чтобы можно было проверить её статус.

Метод «verify\_email» - проверяет электронную почту на существование.

1. Файл «math\_operations.py».

Класс MathOperations. Позволяет создавать графики по поддерживаемым криптовалютам и периодам, таким как: Неделя, Месяц, Год, Пять лет.

Подробнее о двух самых непонятных методах.

Первая – str\_period\_to\_machine. Ниже в методах вы можете увидеть функцию для обработки каждого периода(названия начинаются на process\_...). Для того, чтобы получить цену для определённой даты, эту дату надо представить в формате ДДММГГ. Метод str\_period\_to\_machine это и делает – с помощью несложных циклов он в список self.date\_lst добавляет необходимое количество дат в необходимом в формате, причём учтены все нюансы(например, что часть недели была в прошлом году, или в месяце 28 дней и т.д.)

Вторая – асинхронная функция create\_process. Необходима для того, чтобы добавлять в очередь задач задачу «построение диаграммы»(без этой особенности диаграмма строилась только для последнего написавшего и отправлялась всем). Также, эта функция ускоряет работу бота, т.к. отправка сообщения с графиком происходит непосредственно в функции, что позволяет при завершении процесса сразу же отправить диаграмму и стереть её из памяти.

1. Файл «news.py».

Класс News. Необходимен для ежедневной рассылки новостей подписавшимся пользователям. Делает он это с помощью парсинга самой последней новости с сайта investing.com. В классе также реализована функция send\_news, которая отправляет сообщение непосредственно при завершении своей работы, что позволяет немного ускорить время выполнения.

1. Файл «payment\_operations.py».

Класс PaymentOperations. Создан для операций с платёжной системой Qiwi(в боте используется как основная для всех покупок). Позволяет генерировать код для оплаты и ещё множество функций. (Там понятный код с комментариями, ничего вроде бы не надо объяснять).

1. Папка «static».

Состоит из двух подкатологов – json и img.

img – необходима для хранения графиков, которые оттуда почти сразу же удаляются.

json – это, исходя из названия хранилище json-файлов. Файл crypto\_fees.json – это комиссии, применяемые для покупки каждой криптовалюты. Комиссия включает в себя часть, которую забирает блокчейн для совершения транзакции и часть – для того, чтобы мы тоже были в маленьком плюсе(серьёзно маленьком, комиссии максимально малы).

Файл month\_names.json – необходим для построения диаграмм. Содержит в себе правильные склонения названий месяцев.

Phrases.json – файл, который хранит в себе почти все фразы, в которых не надо никаких особых деталей(вроде имени пользователя).

General\_bot\_info.json(нет в github) – файл, нужный для хранения всех данных бота, в том числе и компрометирующих.

1. Ну, и последнее. Файл «main.py». В нём вся основная логика бота.

Не будем описывать каждую функцию, в основном там понятный код. Опишем самые нижние функции, а именно названия которых представляют собой что-то вроде register…handlers. Это функции для связи самих функций с их ролью в диалоге.