Проект_1

Общие данные

- Бриф
- Доска Trello

Основные соображения

- бот будет один на все три типа пользователя: клиенты, подрядчики, админы (на стороне заказчика). У каждого свое кнопочное меню. Пользователи заносятся в БД за рамками нашего проекта.
- пишем два модуля. Один отвечает за работу с ботами, другой - за работу с БД. Первый использует интерфейс второго, что даст возможность потом менять легко заменить БД если нужно.
- для модуля БД используем на текущем этапе SQLLite и стандартную библиотеку sqlite3. На уровне модуля бота исползуем словарь для кэширования оперативных данных общающихся с ботом пользователей
- для прогамммирования бота используем pyTelegramBotAPI

Модуль бота

Команды бота

- start - стандартная команда запуска бота с приветствием и указанием какие команды можно дать (можно дать /help и /menu).
- help - описывает функционал кнопок меню.Опять же зависит от типа пользователя, который отправил сообщение в бот.
- menu - выводит основное меню пользователя (в заивисимости от того кем является пользователь давший команду)
- step - заново показывает запрос к пользователю на ввод данных если он находится на каком то шаге ввода данных.

Кэш данных

- В модуле бота создаем словарь словарей (chats), где будем хранить оперативные данные о всех пользователях, общающихся с ботом. Словари в словаре отражают данные по конктреному чату, которые находим по ключу `chat_id`. Этот ключ получаем из `message` обработчиком сообщений Telebot.
    - user_id - идентификатор пользователя в Telegram

- type - указывает кто общается с ботом
  - 0 - админ
  - 1- клиент
  - 2 - подрядчик
- order_id - идентификатор заказа - контекст, который имеет отношение к последнему шагу пользователя для использования его на последующих шагах.
- text - любой текст, актуальный в рамках текущего контекста
- step - текущая команда пользователя при ожидании от него ввода данных - показывает на каком шаге находится данный пользователь. Это число, которое однозначно идентифицирует тип пользователя и его шаг
  - для любого типа
    - 0 - на старте (после команды `/start` или `/menu`)
  - клиент:
    - 10 - клиент запросил подачу заявки (нажата кнопка "подать заявку"). Бот ждет текст заявки
    - 11 - бот ждет от клиента учетные данные от админки сайта
    - 12 - бот ждет от клиента ответы на вопросы
    - 13 - бот ждет от клиента замечания, на основании которых он не принимает работу
  - подрядчик
    - 20 - бот ждет оценку времени по заявке, которую хочет взять подрядчика - ПЗ
    - 21 - бот ждет вопроса от Подрядчика по взятой на исполнение задаче

Алгоритм работы бота

- **П1. Пользователь зашел в бот и запустил его** (команда `/start`)
  - проверяем user_name в БД с целью определения его наличия в БД и принадлежности. Если он там есть, то помещаем в словарь `chats` данные (записываем по ключу chat_id тип пользователя и step=0, order_id=None, text=None). В БД в таблице users прописываем `chat_id`, который достаем из `message` и переходим к П1.1.
  - если пользователя в БД нет, пишем, что "У вас нет доступа к боту"
- **П1.1 Приветственное сообщение и меню** с кнопками действий в зависимости от типа пользователя. Для каждого свой набор кнопок и свое приветственное сообщение. Следующие кнопки выводятся на этом шаге, а также на последующих шагах (где нужно - указан ссылка на П1.1)
  - меню Админа #todo
    - пока вывести сообщение что меню амина еще находится в разработке
  - меню Клиента

        - сделать заявку - П2
        - мои заявки - П8
    - меню Подрядчика
        - условия оплаты - П4.
        - список заказов (при условии что нет активного заказа) - П5
        - что я делаю - П7
        - задать вопрос - П9.
        - сдать работу - П11
- **П1.2 Пользователь дал команду** `/menu`
    - step=0. Переход к П1.1
    - step <> 0, значит пользователь находится на каком то шаге ожидания ввода данных. Дейтсвия зависят от шага:
        - step=10. Бот ждет формулировку задачи от клиента
            - Переход к П1.1. step=0
        - step=11. Бот ждет от клиенты учетные данные к админке сайта. Формулировка задачи уже введена.
            - отправляем сообщение "отменить создание заявки?". Пришиваем кнопки "Да" и "Нет"
                - нажата кнопка "Да".
                    - вывод основного как в П1.1
                    - step=0
                - нажата кнопка "Нет"
                    - выводим "Пожалуйста предоставьте адрес сервера, логин и пароль от адинистративной части сайта"
        - step=20. Бот ждет от подрядчика оценку времени выполнения задачи, которую тот планирует взять на исполнение.
            - отправляем сообщение "Отменить взятие заявки". Пришиваем кнопки "Да" и "Нет"
                - нажата кнопка "Да"
                    - вывод основного как в П1.1
                    - step=0
                - нажата кнопка "Нет"
                    - "Пожалуйста введите оценку времени выполнения в свободной форме". Бот продолжит ждать оценку времени выполнения.
        - остальные значения step
            - step=0. Переход к П1.1
- **П2. Клиент нажал кнопку "сделать заявку"**.
    - проверяем наличие подписки. Если она есть, то идем дальше. Если нет, то

        - отправляем сообщение, "Ваша подписка истекла. Вы не можете создавать новые заявки, однако можете посмотреть статус активных"
    - step =10.
    - сообщение "Пожалуйста опишите постановку задачи. Примеры заявок ..... (показыват примеры заявок). Далее бот ждет текстовое сообщение с описанием задачи. Обработка см. П3
- **П3. Пользователь написал в бот сообщение**. Обработка сообщения ведется в зависимости от текущего значения шага на котором находится автор сообщения. Шаг смотрим в словаре `chats[chat_id]`. Далее ветвление:
    - если нет такого chat_id в словаре, проверяем в БД по аналогии с П1 и переходим к п. П1.1
    - если же chat_id есть, то
        - step=10 значит пришел текст заявки от клиента
            - сохраняеми текст заявки в `chats[chat_id][text]` для последующего использования в других шагах
            - посылаем сообщение "Пожалуйста предоставьте адрес сервера, логин и пароль от адинистративной части сайта".
            - step=11
        - step=11 (клиент направил сообщение с учетными данными от админки сайта)
            - пишем в БД данные по заявке - текст, которые достаем из `chats[chat_id]['text']` , учетные данные, chat_id
            - посылаем сообщение Ваша заявка принята, исполнитель будет назначен в течение 24 часов.
            - step=0
        - step=12 (клиент написал ответы на вопросы)
            - ответы на вопросы пишем в БД. `order_id` достаем из кэша клиента.
            - step клиента=0
            - получаем из БД данные по задаче и вопросам
            - поучаем chat_id подрядчика по order_id, который достаем из словаря клиента
            - отправляем в чат подрядчика сообщение "Клиент ответил на ваши вопросы по задаче ID..., текст задачи, вопросы, ответы на вопросы ". Разделям все это с новой строки. К сообщению пришиваем кнопки "принять ответ", "остались вопросы".
                - нажата кнопка "остались вопросы"
                    - step подрядчика = 21
                    - сообщение подрядчику "Введите содераженые вопросов. Каждый вопрос - с новой строки"
                    - бот ждет сообщения от подрядчика. Обработка - П3
                - нажата кнопка "принять ответ"

            - в БД прописываем флаг о снятии вопроса
            - в чат клиенту сообщаем, что его ответ принят и работа по заявке ID такой то продолжается.
    - step=13 (клиент написал замечания из за которых работа не принята)
        - записываем замечания в БД
            - put_client_comments(order_id, comments) `order_id` берем из словаря клиента
        - get_exec_chat_by_order(order_id) - получаем исполнителя работы
        - исполнителю сообщаем "Ваш работа по заявке ID такой то не принята. Есть замечания клиента." И ниже выводим замечания клиента.
        - step киента = 0
    - step=20 (подрядчик ввел оценку времени выполнения)
        - пишем в БД факт взятия заявки подрядчиком и оценку времени выполнения.
            - set_executor_for_oder(oder_id, chat_id, estimation)
        - Дату взятия заказа. ID заказа берем из кэша подрядчика
        - клиенту в чат отправляем сообщение "Ваш заказ номер такой то взят на исполнение" и оценку времени исполнения. `chat_id` клиента получаем по запросу к БД.
            - get_client_chat_by_order(order_id)
        - подрядчику в чат отправляем "Заказ такой то (указаываем ID) закреплен за Вами. Успешной работы!" . order_id заказа хранится в `chats[chat_id]['order_id']` для последующего использования.
        - подрядчику сообщаем "Учетные данные для входа в административную часть сайта клиента....."
            - потребуется get_order_by_id(order_id) - возращает словарь с данными конкретного заказа.
    - step=21 (подрядчик задал вопрос по текущей задаче)
        - записываем вопросы в БД. `order_id` достаем из кэша подрядчика.
            - set_question_for_order(order_id, question)
        - из БД получаем chat_id клиента по order_id
            - get_client_chat_by_order(order_id)
        - сообщение клиенту: "Возникли вопросы по задаче ID ... , описание задачи, вопросы". Приклеиваем кнопку "ответить на вопросы". В callback_data сохраняем `order_id`.
            - нажата кнопка "ответить на вопросы" - П10
        - пишем сообщение в чат подрядчику - ваши вопросы отправлены клиенту.
- **П4. Подрядчик нажал кнопку "условия оплаты".**

    - в БД запрашивается ставка оплаты и выводится в чат подрячдику сообщение "Текущая ставка оплаты за один заказ - ...." и кнопки меню подрядчика.
- **П5. Подрядчик нажал кнопку "список заказов"**.
    - В БД запрашиваются актуальные заказы и в бот отправляется их список отдельными сообщениями к каждому из которых пришита кнопка "взять заказ". callback*data кнопки = `какойто*текст_ID завки`. ID заявки будет использоваться для обработки нажатия этой кнопки.
        - П6. Подрядчик нажал на кнопку "Взять заказ" на каком либо из выведенных заказов.
            - #todo - непонятно как запросить оценку времени исполнения и подтвердить взятие заказа кнопкой как это описано в брифе. Сообщение боту в моем понимание это или нажатая кнопка или сообщение набранное пользователем. Поэтому в качестве решения пока предлагаю так: бот отправляет сообщение "Пожалуйста введите оценку времени выполнения в свободной форме". step меняем на 20. order_id = ID заказа, который вытаскиваем из callback_data кнопки. Ждем ответ Подрядчика. Обработка см. П3
- **П7. Подрядчик нажад кнопку "что я делаю"**.
    - запрашиваем в БД необходимые данные об активной заявке на которую назначен исполнитель. Есле они есть, то:
        - в чат подрядчика сообщаем описание заказа и оценку его времени выполнения , а также дату взятия заказа в работу. Если есть замечания клиента (была неудачная попытка сдать работу), то показываем и их тоже.
        - обновляем в словаре подрядчика `order_id`
- **П8. Клиент нажал кнопку "Мои заявки"**.
    - В БД запрашиваются все заказы этого подрядчика и выводятся в чат отдельными сообщениями. По каждому заказу указываем ID заявки, Содержание заявки, Статус (в работе/поиск исполнителя/есть вопросы).
    - К сообщениям, где есть неотвеченные вопросы подрядчика пришиваем кнопку 'ответить на вопросы'. К сообщениям где работа выполнена и ждет приемки клеим кнопку "принять" и "отклонить"
        - клиента нажал кнопку "ответить на вопросы" - П10
        - клиент нажал кнопку "принять" - П12
        - клиент нажал кнопку "отклонить" - П13
- **П9. Подрядчик нажал на кнопку "Задать вопрос"**.
    - проверяем в БД наличие активной задачи у подрядчика. Если нет, то
        - сообщение "У вас не тактивной задачи"
    - иначе
        - обновляем `order_id` в словаре клиента.

        - сообщение "Введите содержание вопросов. Каждый вопрос - с новой строки"
        - step=21. Бот переходит в ожидание ввода вопроса от подрядчика. Обработка в П3
- **П10. Клиент нажал кнопку "Ответить на вопросы"**.
    - в словарь клиента сохраняем order_id, который берем из callback_data кнопки
    - сообщение клиенту "Вопросы по задаче ID...."."Введите ваши ответы"
    - step=12. Следующее сообщение от этого клиента будет рассматриваться как ответы на вопросы. Обработка см. П3
- **П11. Подрядчик нажал кнопку "сдать работу"**
    - достаем chat_id клиента по order_id заказа (который в кэше у подрядчика)
        - get_client_chat_by_order(order_id)
    - достает данные заказа
        - get_order_by_id(order_id)
    - клиенту пишем "Ваш заказ ID такой то выполнен". Приклеиваем кнопки "принять" (П12) и "отклонить" (П13). В callback_data кнопок засовываем `order_id`
- **П12. Клиент нажал кнопку "Принять"**
    - в БД отмечаем работу как принятую.
    - запрашиваем `chat_id` подрядчика и сообщаем ему, что такая то работа принята клиентом.
    - клиенту пишем что "Вы приняли работу по заявке ID такой то
    - в словаре подрядчика делаем None `order_id`, `text`
- **П13. Клиент нажал кнопку "отклонить"**
    - в БД делаем статус заказва `в работе`.
    - step клиента = 13
    - `order_id` в словаре клиента = `order_id`, который мы достали из callback_data кнопки.
    - бот ждет замечаний клиента. Обработка см. П3

Модуль БД

Таблицы БД

- **users** - пользователи бота. Предполагается что занесение данных о пользователях за рамками нашего проекта.
    - **name** - имя пользователя
    - **tg_name** - ник в телеграмме
    - **user_id** - id пользователя в телеграмме
    - **chat_id** - id чата пользователя в телеграмме

    - **group** - группа пользователей (0 - администратор, 1 - клиент, 2 - подрядчик)
    - **access** - доступ к сервису для клиентов. Указывает есть подписка или нет (True/False)
- **order** - заказы клиентов
    - **order_id**
    - **client_id** - chat_id клиента, создавшего заказ
    - **description** - описание задачи, которое вводит клиент
    - **comments** - доработки, которые указывает клиент если не принимает работу.
    - **credentials** - данные дла входя в админку сайта, предоставляемые клиентом
    - **executor** - chat_id подрядчика, взявшегося за выполнение задачи. На текущем этапе подрядчик может брать только один заказ.
    - **question** - вопросы подрядчика клиенту
    - **answer** - ответы клиента на вопросы подрядчика
    - **status** - статус задачи (0-нет исполнителя, 1- в работе, 2 - есть вопросы подрядчика, 3 - выполнена, 4 - принята заказчиком)
    - **date_reg** - дата регистрации задачи
    - **date_appoint** - дата назначения исполнителя
    - **date_accepted** - дата приемки выполненной работы Клиентом

Функции модуля БД для использования в модуле бота

- set_executor_for_oder(oder_id, chat_id, estimation)
    - **Описание:** задает исполнителя для заказа и устанавливает статус заказа=1.
    - **Аргументы**
        - `order_id` - ID заказа
        - `chat_id` - ID чата подрядчика, который взялся за заказ
        - `estimation` - текст. Оценка сроков, изложенная в свободной форме.
    - **Возвращает**
        - `True` если все прошло без ошибок
- get_active_orders_by_executor(chat_id, status=())
    - **Аргументы:**
        - `chat_id` - id чата исполнителя
        - `status` - кортеж с набором статусов
    - **Возвращает**
        - данные о заказах исполнителя , представленные в виде списка словарей, где каждый заказ это словарь. Ключи словаря - имена полей данных таблицы `orders` Если аргумент `status` пропущен или пуст то возвращаются все заказы независимо от статуса. В противном случае возвращаются заказы только с указанными в кортеже статусами.

- get_order_by_id(order_id, fields=())
  - **Аргументы:**
    - `order_id` - ID заказа
    - `fields` - кортеж с полями которые хотим получить. Если он пуст (по умолчанию), то запрашиваются все поля.
  - **Возвращает:**
    - словарь с данными по заказу #todo
- get_client_chat_by_order(order_id)
  - **Аргументы**
    - `order_id` - ID заказа
  - **Возвращает**
    - `chat_id` клиента, который создал заявку.
- get_exec_chat_by_order(order_id)
  - **Аргументы:** `order_id` - id заказа
  - **Возвращает:** `chat_id` исполнителя. Он может быть `None` в том случае если исполитель не назначен на заказ.
- get_salary_rate()
  - **Возвращает** словарь со ставками в зависимости от типа работ. На перспективу, если ставки будут разные в зависимости от чего либо. На текущем этапе возращает словарь с единственным ключем `rate_per_task` и значением ставки за одну задачу.
- chek_access_by_id(chat_id)
  - **Описание:** проверяет наличие доступа к системе у пользователя с указанным `chat_id`.
  - **Возвращает:**
    - кортеж из двух элементов, где первый - указывает доступ (-1 - пользователя нет в БД, 0 - есть в БД но нет доступа. 1 - есть в БД и есть доступ), а второй - группу пользователя если он есть в БД (в противном случае None)
- chek_access_by_tgname(chat_id)
  - **Описание:** То же, что `chek_access_by_id`, но проверка идет по имени пользователя в Telegram
- Продолжение следует ....