**Пояснительная записка к проекту «Телеграмм чат-бот для подготовки к экзаменам ОГЭ и ЕГЭ».**

Авторы проекта: Андрей Лубянников и Даниил Дрокин.

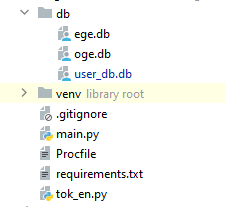
**Описание идеи:**

Каждому школьнику в 9 и 11 классах приходится писать государственный экзамен, и для этого мы решили написать телеграмм чат бота. Данный бот поможет вам подготовиться к экзаменам по выбранному Вами предмету, присылая подходящие для вас задачи.

**Описание реализации:**

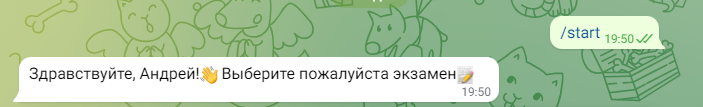
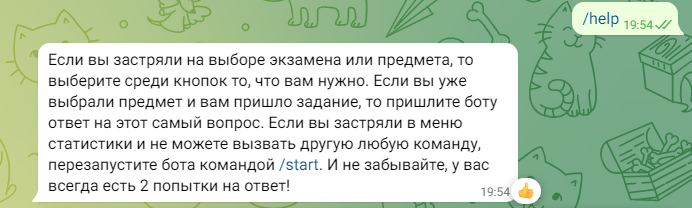
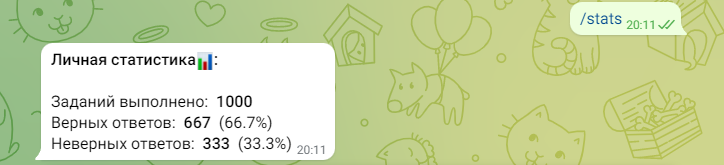
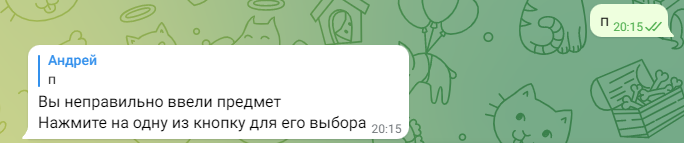
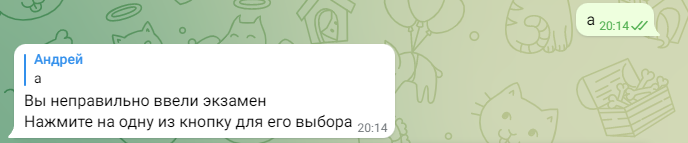
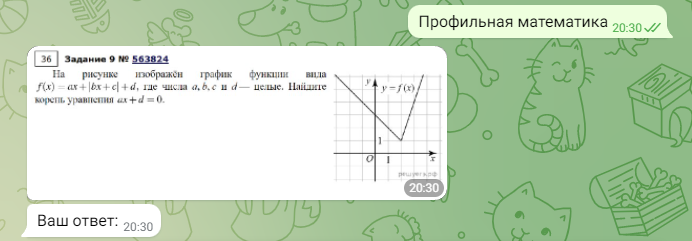
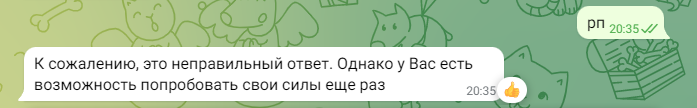
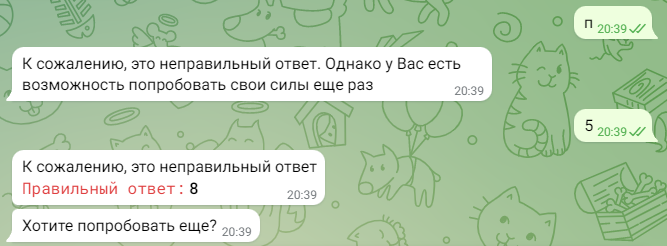
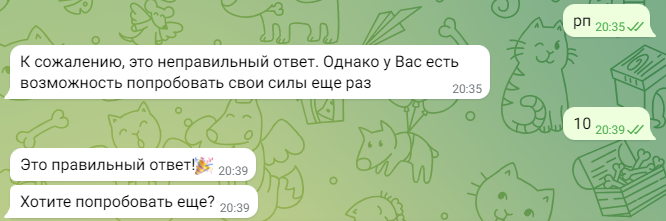
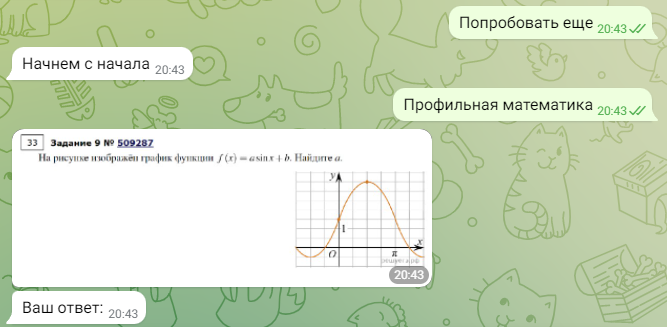
В коде присутствует один класс, в котором будут находиться все состояния, и 11 функций: 5 функций отвечают за обработку команд (таких как start, help, stats, oge и ege), 2 функции за обработку неправильно введенных экзаменов и предметов, и остальные 4 за обработку ответа пользователя к отправленной ему задаче. Наш проект состоит из двух python файлов: **main.py**, в котором хранится весь код, и **tok\_en**.py, в котором хранится токен нашего телеграмм бота. В папке **bd** хранятся 3 базы данных: oge.bd (задания к предметам ОГЭ), ege.bd (задания к предметам ЕГЭ) и user\_db.db (данные о пользователях, нужные для составления статистики).

Все необходимые файлы для работы бота:



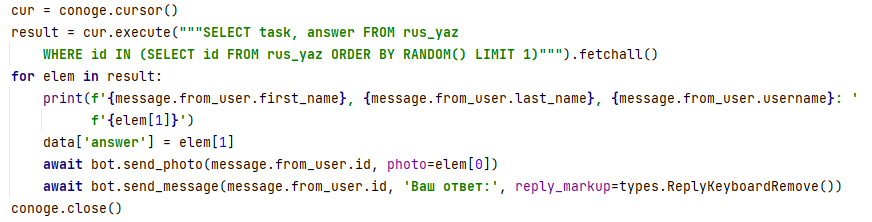
Procfile необходим лишь для запуска самого проекта на хостинг Heroku.

**Функции:**

1. start\_handler – отвечает за запуск бота и предложение пользователю экзамена. 
2. help\_handler – при вызове выводит информацию пользователю, объясняющая ему как выйти из определенной ситуации. 
3. stats\_handler – выводит личную статистику пользователя. 
4. failed\_process\_examen и failed\_process\_predmet – сообщают пользователю о неправильно введенном экзамене или предмете и просят снова вести экзамен или предмет. 
5. process\_oge и process\_ege – при вызове команд oge или ege с помощью кнопок предлагает выбрать предмет для соответствующего экзамена. 
6. process\_predmet – после выбора предмета программа берет из базы данных (для каждого экзамена своя база данных) случайное задание и ответ к нему (для каждого предмета своя таблица). 
7. first\_answer – проверяет ответ, введенный пользователем. Если ответ правильный, мы поздравляем пользователя и предлагаем попробовать решить еще одну задачу из того же предмета (происходит переход к функции process\_end\_ans). Иначе бот сообщает, что ответ неверный, но у пользователя есть еще шанс на решение этой задачи (происходит переход к функции second\_answer). 
8. second\_answer – проверяет второй ответ пользователя. Если ответ правильный, мы поздравляем пользователя и предлагаем попробовать решить еще одну задачу из того же предмета (происходит переход к функции process\_end\_ans). Иначе бот сообщает, что ответ неверный, и присылает ему правильный ответ, и предлагает попробовать решить еще одну задачу из того же предмета (происходит переход к функции process\_end\_ans). 
9. process\_end\_ans – если пользователь нажал на кнопку 'Попробовать еще', бот предлагает нажать на кнопку предмета, который он решал, и бот снова кидает ему задачу (происходит переход к функции process\_predmet). Иначе бот прощается с пользователем, и для повторного использования, предлагается перезапустить его кнопкой start. 

Для перехода от функции к функции используется машина состояний (FSM). Всего их 5, хранятся они в классе User. Пример работы машины состояния:

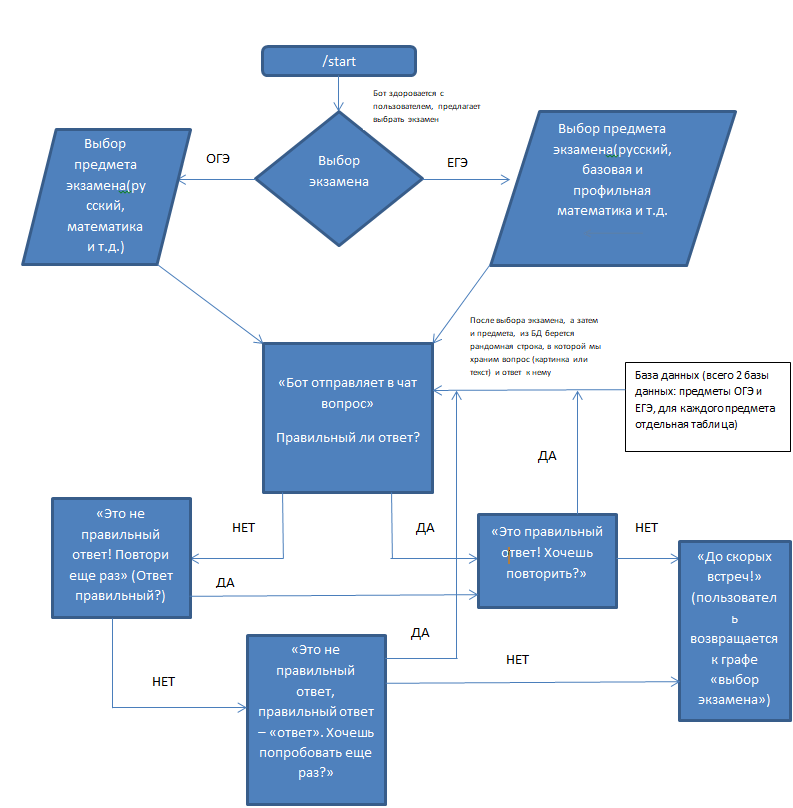
Запуск бота командой /start (включаем состояние examen) -> функции process\_oge и process\_ege принимают состояние examen (включаем состояние predmet) -> функция process\_predmet принимает состояние predmet (включаем состояние answer) -> функция first\_answer принимает состояние answer (при правильном ответе включаем состояние end\_ans) -> функция process\_end\_ans принимает состояние end\_ans (при выборе повторного выполнения задания, включается состояние predmet, иначе бот сбрасывает все состояния и прощается с пользователем).

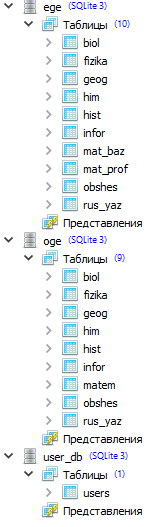
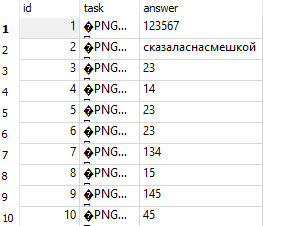
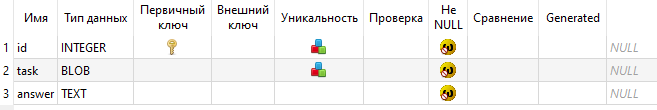
Также присутствует состояние wast\_ans, которое включается при неправильном первом введенном ответе, принимает это состояние функция second\_answer. Такие данные как правильный ответ и последний выбранный предмет пользователя хранятся в той же самой FSM в виде словаря: ключ и значение к нему. Также все задания выбираются случайно из базы данных с помощью ускоренного запроса, помогающего за секунду выбирать случайное задание. Пример (выбор задания из базы данных oge таблицы rus\_yaz): 

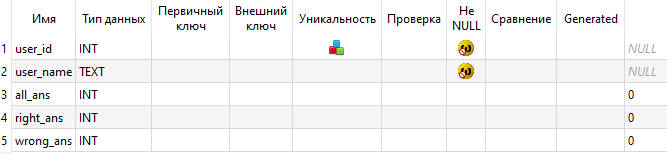
**Описание технологий:**

В данном проекте мы использовали библиотеку для разработки телеграмм бота aiogram. Она является асинхронной, что помогло сделать его многопользовательским (то есть им могут пользоваться одновременно несколько пользователей) и быстрым в работе (изначально планировалось писать бота на более простой библиотеке pyTelegramBotAPI (сокращенно telebot). На этой библиотеке мы смогли написать основную логику бота за 3 дня. Но поэкспериментировав, мы выяснили, что она путает между пользователями запоминающиеся данные (верный ответ и последний предмет), то есть сохранялись последними данные того человека, который последним запустил бота и выбрал предмет. Поэтому решением данной проблемы стал переход с синхронной библиотеки telebot на асинхронную aiogram. В aiogram все хранящиеся данные для пользователей были свои и никак не перемешивались между собой, плюс эта библиотека работала быстрее и содержала в себе удобные и интересные функции, что помогло нам сделать бота лучше). Поэтому перед каждой функцией будет стоять **async**, и перед каждой отправкой сообщения и подключению к состоянию будет стоять **await.** Также для удобства пользователя и правильной работе бота были добавлены кнопки (поэтому желательно при работе с ботом нажимать кнопки, кроме печати ответа на задание, но если пользователь неправильно напишет экзамен или предмет, бот ему об этом сообщит). Также использовалась ранее описанная машина состояний (FSM – Finite state machine).

Также до начала разработки бота был создан алгоритм работы бота, на который мы опирались при создании основной логики бота (автор алгоритма – Андрей Лубянников):



Для создания баз данных использовалось приложение SQLite Studio. Как выглядят все базы данных, необходимые для работы бота, а также названия полей для таблицы с предметами: 

А также база данных с хранящейся статистикой для пользователей:  

Столбец user\_id: id пользователя в телеграмме, у каждого оно уникально

Столбец user\_name: имя пользователя

Столбец all\_ans: сколько всего было сделано заданий

Столбец right\_ans: сколько было сделано правильно заданий

Столбец wrong\_ans: сколько было сделано неправильно заданий

Все задания были взяты с сайта <https://oge.sdamgia.ru/>.

Адрес бота на хостинге (при открытии вы ничего не увидите, но если бы это был сайт, то страница не была бы пуста): <https://aiogramegeogebot2.herokuapp.com/>

**Библиотеки, необходимые для запуска:**

-SQLite3 (стандартная библиотека, не нуждается в установке)

-asyncio (стандартная библиотека, не нуждается в установке)

-aiogram (установка: pip install aiogram)

-папка db с 3 базами данных: oge.db, ege.db и user\_db.db.