Пояснительная записка asenshik\_bot

Реализация электронного дневника с помощью Telegram API и библиотеки Telebot. Суть заключается в создании пользователей в виде учителей и учеников. Учителя могут писать домашнее задание и составлять расписание. Ученики в свою очередь могут просматривать домашние задания на определенные дни и узнавать расписания по дням недели.

* База данных, подключение к ней, методами execute вносим информацию и получаем информацию

# глобальный класс с функциями, отвечающими за работу с БД  
class SQLighter:  
 global ACTIVE\_CLASS, SHEDULE\_ID  
  
 # подключение к БД  
 def \_\_init\_\_(self, user\_id):  
 self.database = 'db.db'  
 self.con = sqlite3.connect(self.database)  
 self.cursor = self.con.cursor()  
 self.user\_id = user\_id

* Создание кнопок в боте, реакция на их нажатия, получение сообщений методом message

# команда /start  
@bot.message\_handler(commands=['start', 'help'])  
def start\_message(message):  
 # получаем имя user и здороваемся с ним  
 user\_first\_name = str(message.chat.first\_name)  
 # создаём список с нужными для нас кнопками  
 buttons = ['👩‍🏫Создать класс', '👨‍🎓Найти класс', '❓Связаться с разработчиками', '🎓Ваши классы',  
 '🆔Получить id']  
 markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 # добавление кнопок из списка на главный экран  
 for button in buttons:  
 markup.add(button)  
 # отправляем сообщения пользователю, который написал команду /start  
 bot.send\_message(message.chat.id, f'Привет, {user\_first\_name}!\nТебя приветствует Asenshik\_bot,'  
 f' пользуйся ботом с помощью команд снизу👇.', reply\_markup=markup)

* Вызов функций в зависимости от того, какую опцию использовал пользователь

# создание класса по вызову от нажатой кнопки  
def create\_class(message):  
 global ACTIVE\_CLASS  
 try:  
 # Выход из функции при необходимости вернуться на главную  
 if message.text == '✅Назад в главную':  
 return start\_message(message)  
 elif message.text == '/start':  
 return start\_message(message)  
 # создание уникального ключа  
 creating\_key = True  
 # начальное значение ключа  
 key = ''  
 # подключение к БД  
 sqlighter = SQLighter(message.from\_user.id)  
 # генерация случайного ключа  
 while creating\_key:  
 # добавление случайных значений в ключ  
 key = ''.join(random.choice(string.digits) for \_ in range(6))  
 # проверка уникальности ключа  
 if key[0] != '0':  
 if not sqlighter.add\_class(key, message.text):  
 creating\_key = False  
 # присваеваем постоянной переменной ACTIVE\_CLASS значение ключа  
 ACTIVE\_CLASS = key  
 # добавление админа в класс по его ключу  
 sqlighter.add\_admin(key)  
 # добавление пользователя в класс по его ключу  
 sqlighter.add\_user\_to\_class(key)  
 markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 # создание списка необходимых кнопок  
 buttons = ['📓Расписание',  
 '📒ДЗ',  
 '✅Назад в главную']  
 # добавление кнопок на экран  
 markup.add(\*buttons)  
 bot.send\_message(message.chat.id, f'Класс был успешно создан. Ваш ключ:\n{key}',  
 reply\_markup=markup)  
 # вывод сообщения об ошибке в случае некоррекного ввода данных / неверного вызова  
 except Exception as e:  
 bot.send\_message(message.chat.id, '❌Ошибка! Не удалось создать класс.')