**Prototype Telegram Bot**

**Окружение**

* IDE: Microsoft Visual Studio 2019
* DB: локальная база данных MySql на Open Server; СУБД: phpMyAdmin

**Библиотеки**

* JSON (Newtonsoft.Json)
* Telegram API (Telegram.Bot)
* MySql (MySql от Oracle)

**Краткое описание внутренней структуры программы**

*Классы:*

1. Program.cs - класс, откуда запускается программа. Инициализация клиента, подключение по API, инициализации события ввода команд, обработка локального файла .json.
2. CommandService.cs - Обработка и выполнение команд.
3. Miscellaneous.cs - Вспомогательный статичный класс.
4. DataBase.cs - Статический класс для работы с базой данных.

**Краткое описание внешней структуры программы**

Файл config.json служит конфигурационным файлом, который редактируется индивидуально.

Поле “token” является ключом API Telegram бота.

Поле “commandPrefix” является префиксом всех команд.

Поля “host” “port” “database” “uid” “password” - информация для подключения к БД.

*Таблица в БД:*

**id** - *int*, *автоинкремент*; **datetime** - *bigint* (для хранения в unix timestamp); **name** - *varchar(100)*; **phone** - *varchar(40)*;

**Описание работы**

**Program.cs:**

После запуска программы, происходит чтение конфигурационного файла .json, присвоение значений в локальные статические переменные, инициализация и подключение клиента к серверу, подключение к базе данных, запуск обработки событий (\_client.OnUpdate), которое предназначено для последующей обработки команд:

Метод **\_client\_OnUpdate** вызывается при каждом отправлении любого сообщения в телеграмм канал (либо при другом событии с ботом). Если тип аргумента события (“e” класса **UpdateEventArgs)** является **ChannelPost** перечисления **UpdateType,** происходит проверка текста на командный префикс, удаление префикса и передача аргументов в обработчик команд (**CommandService**). Все вышеперечисленные действия заключены в инструкцию *try-catch*, которая обрабатывает любые исключения, вызванные в обработчике команд и пишет причину ошибки, в случае ее возникновения.

**CommandService:**

В пространстве имен проекта расположено перечисление **commandType.** Класс **CommandService** содержит метод обработки и вызова получаемых из класса **Program** команд (*Execute*).

*Add, Delete, Sel, Help -* методы, выполняющиеся после вызова команд, вывод информации в канал. (в аргументах функции содержится значение чата, куда нужно отправить сообщение)

**Miscellaneous:**

Вспомогательный класс, который содержит в себе методы:

* *returnEnumCommandType -* получает строку и возвращает значение типа *commandType* (enum)
* *separateRows* - разделяет входные аргументы команды и возвращает массив типа *row*, для последующего использования (запись в БД, удаление из БД, вывод данных в канал.
* *getId* - вырезает ID из строки.
* *Структура row* - визуализация данных из БД, для упрощенной работы.

**DataBase:**

* *Init* - вызывается из **Program** при запуске программы. Инициализирует подключение к БД.
* *getUserInfo* - возвращает значение типа *row* по аргументу ID. Поиск происходит по SQL запросу, полученный запрос преобразуется в reader и обрабатывается в цикле.
* *addRow* - добавляет запись с заданными параметрами в БД, возвращает ID записанной роли.
* *deleteRow* - удаляет запись с заданным ID.
* *startTimer -* запускает событие бесконечного таймера, который вызывает метод *OnTimerTicked*
* *OnTimerTicked* - проверка на новые записи в БД, которые были добавлены не через команды. Сравниваются размеры полученных записей предыдущей итерации и текущей. Если количество текущих записей больше, то информация о всех добавленных записях отправляется в Telegram канал.