Основные цели:
- Бот должен быть **отказоустойчивым**.
- Бот должен считывать инфу с **json конфиг**а.
- Бот должен сообщать своё **состояние** каждые **30 минут** всем «status_checkers_ids».
- При ошибке создания сервера бот должен попытаться снова через 5 раз **5 секунд**, 5 раз **1 минуту**, остальное - **10 минут**.
- Бот должен вести **логирование**.
- Бот должен отслеживать **message_new event**.
- Бот должен **запускаться при запуске системы**.
- Бот должен **считывать** токен и source-target беседы при запуске.
- Бот должен уметь работать **в двух режимах**:
  1. **Режим настройки**. **Можно** использовать все **команды**.
  2. **Режим работы**. **Нельзя** использовать все **команды**.

Используемые библиотеки:
- "jsoncpp" чтобы работать с json.
- "Boost.Log" ну тут понятно да.
- "libhttpserver" простая и вроде как приятная библиотека для поднятия сервера.
- "cpr" для легкого использования http запросов.

Механизм работы:
1. Бот считывает файл config.json и находит там:
   - «**mode**» «work» or «config» **(обязательное)**
   - «**token**» токен сообщества бота **(обязательное)**
   - «**target_chats**» с парами (номер блока : peer_id беседы) — map (обязательное **для work**)
   - «**source_chat**» peer_id чата (обязательное **для work**)
   - «**status_checkers_ids**» кому отправлять каждые **30 минут** инфу о состоянии бота (необязательное)
   - «**secret_string**» строка для подтверждения связи с вк

2.1 Режим настройки. **Mode = «config»**
  С помощью команд можно настроить бота. При вызове команд - файл будет перезаписываться.

2.2 Режим работы. **Mode = «work»**
  Бот просто анализирует сообщения в студсовете и действует в соответствии с тегами.

Описание команд:
Режим настройки:
- «**reg_target** номер_блока» прописывается во всех блоках во время первоначальной настройки. Надо чтобы и номер_блока и id беседы были свободны.
  Если id беседы является source, то сообщение об ошибке.
  Если id беседы используется с другим номером блока, то сообщение об ошибке.
  Если номер блока уже привязан к какой-то беседе, то сообщение об ошибке.
- «**reg_source**» прописывается в беседе откуда будут пересылаться сообщения. Надо чтобы source был не занят и id не лежал в targets.
  Если source уже существует, то сообщение об ошибке.
  Если id находится в targets, то сообщение об ошибке.
- «**reg_checker**» отправляется боту в личные сообщения.
- «**del_target_by_id»** В зарегестрированной беседе.
- «**del_target_by_block** номер_блока**»** В личке с ботом.
- «**del_source»** В личке с ботом.
- «**del_this_source**» В source беседе. Беседа перестает быть source. Если беседа и так не сурс, то сообщение.
- «**del_checker»** в личке с ботом. Если айди не является checker, то сообщение об ошибке.

Для всех режимов:
- «**print_map**» отправляется боту в личные сообщения. Доступно только определенным лицам. Отправляет map в личном сообщении. Вместо peer_id помещаются названия бесед.

Теги сообщений:
- «**_всем_блокам_** title» такое сообщение будет переслано всем блокам. Title будет помещен в тексте сообщения бота.
- «**_всем_блокам_важно_** title» такое сообщение будет переслано всем блокам с пометкой @all. Title будет помещен в тексте сообщения бота.

Исключительные ситуации:
- Код ошибки 5. Неверный токен. Может возникнуть при работе с API. Нужно попытаться снова через **1, 2, 5 секунд** — если не получилось то прервать выполнение операции и ~~надеяться на лучшее~~. Редко такое бывает в вк, что он ебланит жестко с токенами и не узнает их. Как правило быстро проходит (~5 минут).
- Код ошибки 1 или 10. Неизвестная ошибка. Может возникнуть при работе с API. Нужно попытаться снова через **1, 2, 5 секунд** — если не получилось то прервать выполнение операции. И такая хуйня периодически случается, но тоже быстро проходит. (одноразовая ошибка)
- Код ошибки 6. Слишком много запросов в секунду. Может возникнуть при работе с API. Нужно выдержать паузу в 2 секунды и повторить снова. И вообще таких случаев лучше не допускать. Будут тормозить программу. Использовать метод **messages.send peer_ids** чтобы избежать проблемы.
- При остальных кодах нужно попытаться снова через **1, 2, 5 секунд** — если не получилось то прервать выполнение операции и ~~надеяться на лучшее.~~
- Если пришёл json неадекватного содержания, то просто проигнорировать его с соотвествующим WARNING.
- Если выскочило исключение связанное с http сервером, то необходимо его перезапустить через **1, 2, 5, 10, 30, 30... секунд.** Также необходимо попробовать отправить соответствующее сообщение status_checker’ам.
- Если произошло какое то исключение во время обработки очередного сообщения, то сообщение пропускается с WARNING.