Telegram Bot

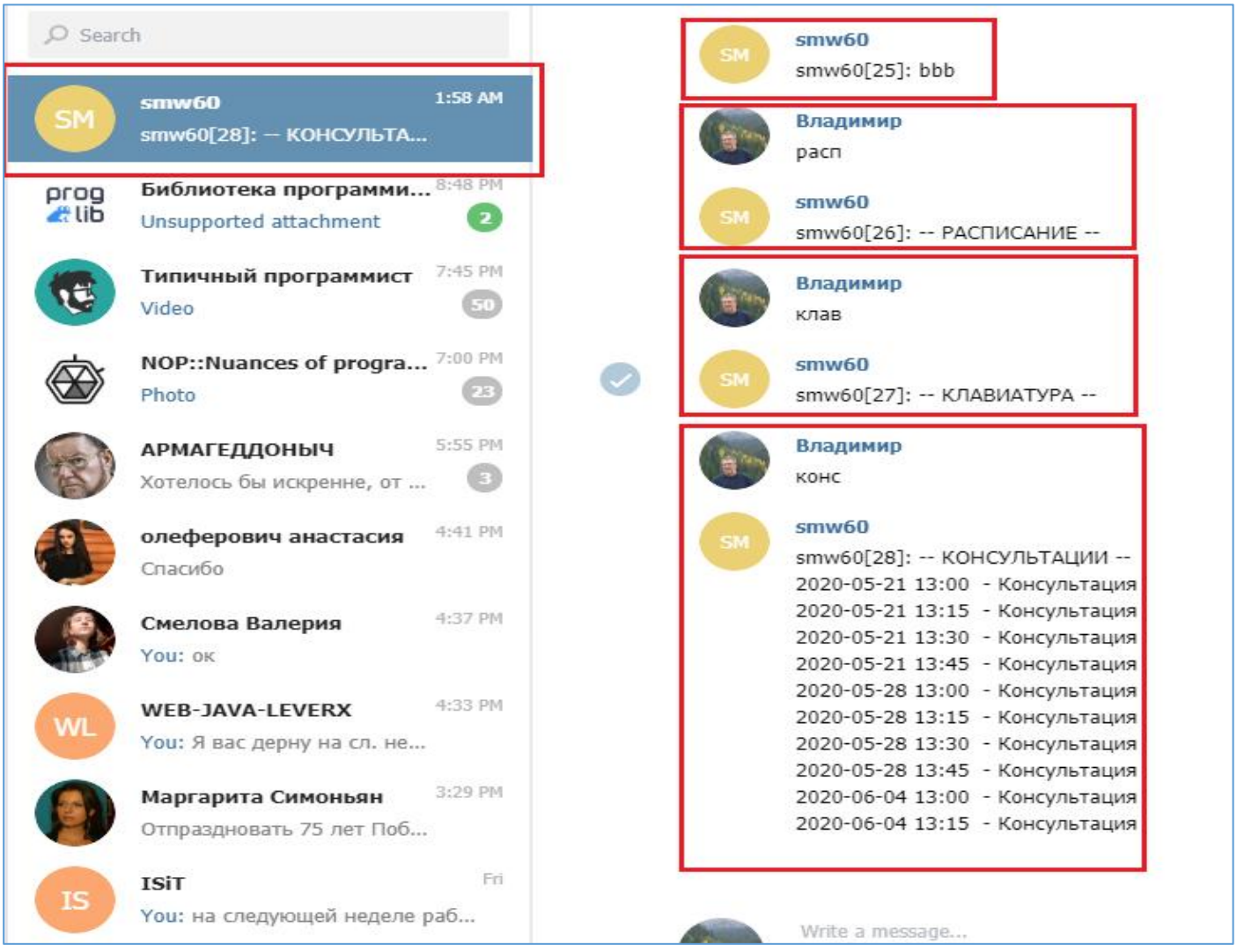
1. Telegram Bot

https://khashtamov.com/ru/create-telegram-bot-in-python/
https://vc.ru/selectel/22593-howto-bot-selectel
https://blog.ringostat.com/ru/sozdaem-chat-bot-v-telegram/
https://www.youtube.com/watch?v=eQrTCHW2UqY
https://netpeak.net/ru/blog/kak-sozdat-chat-bot-dlya-telegram-instruktsiya-dlya-administratorov-kanalov/
https://core.telegram.org/bots
https://core.telegram.org/bots/api

2. Ngrok

https://pavelpage.ru/koderstvo/nastroyka-ngrok-dlya-otladki-telegram-bota.html
https://pavelpage.ru/koderstvo/nastroyka-ngrok-dlya-otladki-telegram-bota.html
https://dashboard.ngrok.com/get-started/setup
https://dashboard.ngrok.com/get-started/setup
https://ngrok.com/docs

3. Telegram Bot



4. **Telegram Bot:** основной цикл запросов обновлений

```
(async (msg)=>{
   let parms  =  {limit:10, timeout:60, offset:0};            // макс. сообщ.,  long pool timeout, смещение в запросах
   try{
           let clock = (new Date()).getTime();
           for (let i = 0; i < 1500; i++){                     // цикл long pool requests
               let rc1    = await TlgGet(parms);               // получаем изменения
               let rc2    = await TlgOut(rc1, msg);            // отправляем ответ
               let rc3    = await TlgWait(clock, 5000);        // задержка между запросами на обновления
               console.log(i, 'clock =', rc3);
           }
    }
    catch(err){console.log( err);}
})(msg);
```

```
22 2020-05-15T22:56:39.545Z timeout tick { limit: 10, timeout: 60, offset: 916712421 }
21 clock = 1019
23 2020-05-15T22:57:34.733Z timeout tick { limit: 10, timeout: 60, offset: 916712421 }
22 clock = 1074
24 2020-05-15T22:57:40.122Z response tick { limit: 10, timeout: 60, offset: 916712422 } aaa
23 clock = 1079
25 2020-05-15T22:58:01.256Z response tick { limit: 10, timeout: 60, offset: 916712423 } bbb
24 clock = 1101
26 2020-05-15T22:58:23.561Z response tick { limit: 10, timeout: 60, offset: 916712424 } расп
25 clock = 1123
27 2020-05-15T22:58:37.805Z response tick { limit: 10, timeout: 60, offset: 916712425 } клав
26 clock = 1137
28 2020-05-15T22:59:03.596Z response tick { limit: 10, timeout: 60, offset: 916712426 } конс
27 clock = 1163
29 2020-05-15T22:59:58.381Z timeout tick { limit: 10, timeout: 60, offset: 916712426 }
28 clock = 1218
30 2020-05-15T23:00:53.569Z timeout tick { limit: 10, timeout: 60, offset: 916712426 }
29 clock = 1273
31 2020-05-15T23:01:48.796Z timeout tick { limit: 10, timeout: 60, offset: 916712426 }
30 clock = 1328
```

5. **Telegram Bot:** запрос обновлений

```
let TICK = 0;
let https = require('https');
let  getoptions = (method)=>{
                    return {
                        host:'api.telegram.org',
                        path: `/bot1198153330:AAErhmwgwSSh1oo-B2mokUCss5jl6X6eu5I/${method}`,
                        port: 443,
                        method:'POST',
                        headers:{'content-type':'application/json', 'accept':'application/json'}
                    };
};
```

```
let reqX = (parms, resolve, reject)=>{
    let rc = {next_parms:parms, tick:TICK,  result:[]};
    let req = https.request(getoptions('getUpdates'), (res) => {
        let data = '';
        TICK++;
        res.on('data', (chunk) => {data += chunk.toString('utf8');});
        res.on('end', () => {
                        let teleg = JSON.parse(data);
                        if (teleg && teleg.ok){
                            if (teleg.result.length > 0) {rc.next_parms.offset = teleg.result[teleg.result.length-1].update_id+1;}
                            rc.result = teleg.result;
                            resolve(rc);
                        }
                        else reject('error 1');
        });
        req.on('error', (e) => {console.log('http.request: error:', e.message); reject('error 2');});
    })
    req.write(JSON.stringify(parms));
    req.end();
}
module.exports.TlgGet = (parms)=>{
                        let rcc = new Promise((resolve, reject)=>{reqX(parms, resolve, reject);});
                        rcc.catch((err)=>{return err;});
                        return rcc;
};
```

6. **Telegram Bot:** отправка сообщений

```
let tlgout = (p, pmsg, resolve, reject)=>{
        if (p.result.length > 0){
            p.result.map(el =>{
                        pmsg(el.message.text, TICK).then((text)=> {
                            let req = https.request(getoptions('sendMessage'), (res) => {
                                console.log(TICK,(new Date()).toISOString(), 'response tick', p.next_parms, el.message.text);
                            })
                            req.on('error', (err) => {console.log('error 3', err); reject('error 3');});
                            req.write(JSON.stringify({chat_id:el.message.chat.id, parse_mode:'HTML', text:text}));
                            req.end();
                        });
            });
            resolve('response');
        }
        else {
            console.log(TICK, (new Date()).toISOString(),'timeout tick', p.next_parms);
            resolve('timeout tick');
        }
}

module.exports.TlgOut = (p, msg)=>{
                        let rcc = new Promise((resolve, reject)=>{tlgout(p, msg, resolve, reject);});
                        rcc.catch((err)=>{return err;});
                        return rcc;
}
```

7. **Telegram Bot:** задержка

```
module.exports.TlgWait = (clock, pt)=>{
                        let t = pt||5000;
                        return new Promise((resolve, reject)=>{
                                        setTimeout((c)=>{
                                            resolve(parseInt(((new Date()).getTime() - c)/1000).toFixed(0));
                                        }, t, clock);
                                    });
};
```

8. **Telegram Bot:** формирования ответного сообщения

```
let code = (txt)=>{
    let rc ='uncn';
    if       (txt == 'расписание' || txt == 'расп' || txt == 'распис' )  rc = 'shed';
    else if  (txt =='конс')                                              rc = 'cons';
    else if  (txt =='клав')                                              rc = 'ordr';
    return rc;
};
```

```
let  msg = async (txt, tick)=>{
   let rc = `smw60[${tick}]: ${txt}`;
   switch(code(txt)){
       case     'cons':    rc = `smw60[${tick}]: ${ cons(await ConsGet())}`; break;
       case     'shed':    rc = `smw60[${tick}]: -- РАСПИСАНИЕ --`; break;
       case     'ordr':    rc = `smw60[${tick}]: ${ordr()}`; break;
       default:            rc = `smw60[${tick}]: ${txt}`;
   }
   return rc;
};

(async (msg)=>{
   let parms  =  {limit:10, timeout:60, offset:0};            // макс. сообщ.,  long pool timeout, смещение в запросах
   try{
           let clock = (new Date()).getTime();
           for (let i = 0; i < 1500; i++){                    // цикл long pool requests
               let rc1    = await TlgGet(parms);              // получаем изменения
               let rc2    = await TlgOut(rc1, msg);           // отправляем ответ
               let rc3    = await TlgWait(clock, 5000);       // задержка между запросами на обновления
               console.log(i, 'clock =', rc3);
           }
    }
    catch(err){console.log( err);}
})(msg);
```

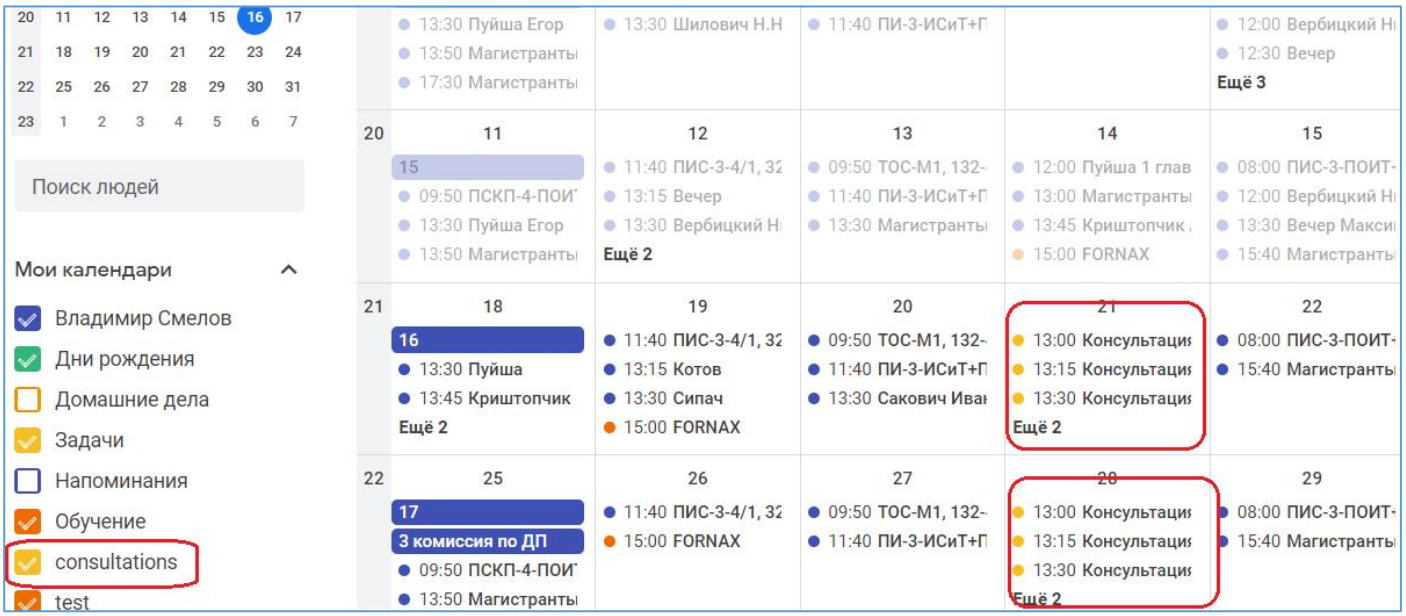
```
let code = (txt)=>{
   let rc ='uncn';
   if       (txt == 'расписание' || txt == 'расп' || txt == 'распис' )  rc = 'shed';
   else if  (txt =='конс')                                              rc = 'cons';
   else if  (txt =='клав')                                              rc = 'ordr';
   return rc;
};

let cons = (list)=>{
   let rc = '-- КОНСУЛЬТАЦИИ --\n';
   let f =  new Intl.DateTimeFormat('ru-RU', {  year:'numeric', month:'2-digit',day:'2-digit', hour:'2-digit',minute:'2-digit'})
   list.forEach(el =>{
      rc+= `${f.format(new Date(el.datetime))}  - ${el.event}\n`;
   });
   return rc;
};

let ordr = ()=>{
   let rc = '-- КЛАВИАТУРА --\n';

   return rc;
};
```

9. **Googles Calendar:**




```
let  msg = async (txt, tick)=>{
   let rc = `smw60[${tick}]: ${txt}`;
   switch(code(txt)){
        case     'cons':    rc = `smw60[${tick}]: ${ cons(await ConsGet())}`; break;
        case     'shed':    rc = `smw60[${tick}]: -- РАСПИСАНИЕ -- ; break;
        case     'ordr':    rc = `smw60[${tick}]: ${ordr()}`; break;
        default:            rc = `smw60[${tick}]: ${txt}`;
   }
   return rc;
};
```

10. **Googles Calendar:** аутентификация для *запрос к* ***google calendar***

```
// https://googleapis.dev/nodejs/googleapis/latest/index.html
const {google}      = require('googleapis');
const credentials   = require('./credentials.json');
const tocken        = require('./token.json');
const consultations = {calendarId: 'h32usjd65upr2kllfh4eapfug0@group.calendar.google.com',
                       timeMin:     (new Date()).toISOString(),
                       maxResults:  10,
                       singleEvents: true,
                       orderBy: 'startTime'};

    console.log('tocken.expiry_date =', new Date(tocken.expiry_date));

    const {client_secret, client_id, redirect_uris} =  credentials.installed;
    const oAuth2Client = new google.auth.OAuth2(client_id, client_secret, redirect_uris[0]);
    oAuth2Client.setCredentials(tocken);
    oAuth2Client.on('tokens', (tokens) => {
        if (tokens.refresh_token) console.log('tokens.refresh_token = ', tokens.refresh_token);
        console.log('tokens.access_token = ', tokens.access_token);
    });
```

11. **Googles Calendar:**запрос к google

```
const calendar = google.calendar({version: 'v3', auth:oAuth2Client});

let getlist = (cb)=>{
        let rc = [];
        calendar.events.list(consultations, (err, res) => {

                                if (err) cb('Calendar API error: ', null);
                                else
                                {
                                    rc = res.data.items.map((event, i) => {
                                        const start = event.start.dateTime || event.start.date;
                                        return  {datetime:start, event:event.summary};
                                    });
                                    cb(null, rc);
                                }
        });

};

module.exports = ()=>{
    let rcc = new Promise((res, rej)=>{getlist((err, list)=>{(err)?rej(err):res(list);})});
    rcc.catch(err=>{return err;});
    return rcc;
}
```

12. **Telegram Bot: npm node-telegram-bot-api**

```
// https://github.com/yagop/node-telegram-bot-api/blob/master/examples/polling.js
// https://archakov.im/post/nodejs-make-buttons-on-telegram-api.html
// https://github.com/yagop/node-telegram-bot-api/issues/319
```

```
process.env.NTBA_FIX_319 = 1;
const TOKEN       ='1198153330:AAErhmwgwSSh1oo-B2mokUCss5jl6X6eu5I';
const TelegramBot = require('node-telegram-bot-api');
const request     = require('request');
const ConsGet     = require('./33-09m');
const fdate       = new Intl.DateTimeFormat('ru-RU', { year:'numeric', month:'2-digit',day:'2-digit', hour:'2-digit',minute:'2-digit'});
const bot         = new TelegramBot(TOKEN, {polling: true});

bot.onText(/конс/, (msg)=>{
    ConsGet().then((list)=>{
        let txt = '-- КОНСУЛЬТАЦИИ --\n';
        list.forEach(el => {txt+= `${fdate.format(new Date(el.datetime))} - ${el.event}\n`;});
        bot.sendMessage(msg.chat.id, txt);
    })
});
bot.onText(/расп/, (msg)=>{bot.sendMessage(msg.chat.id, '-- РАСПИСАНИЕ --');});
bot.onText(/клав/, (msg)=>{
   ConsGet().then((list)=>{
        let txt = '-- КОНСУЛЬТАЦИИ --\n';
        let opts = {reply_markup:{inline_keyboard:[]}};
        list.forEach(el => {
            let d = fdate.format(new Date(el.datetime));
            let e = el.event;
            opts.reply_markup.inline_keyboard.unshift([{text:`${fdate.format(new Date(el.datetime))} - ${el.event}`, callback_data:d}])
        });
        bot.sendMessage(msg.chat.id, '-- КОНСУЛЬТАЦИИ --\n', opts);
    });
});
bot.onText(/xxx/, (msg)=>{console.log('xxx = ', msg);});
bot.on('callback_query', (callbackQuery)=>{const action = callbackQuery.data; console.log(callbackQuery.message);});
bot.on("polling_error", (err) => console.log(err));
```