

Бот WhatsApp воздушных тревог в Украине через API на Node.js

Бот запускается и авторизуется через QR-код.

Каждые 30 секунд проверяет API тревог.

Если в Запорожской области тревога когда тревога отменяется: отправляется сообщение.

Всё логируется в консоль сообщениями.

1. Импорт библиотек

```
import fetch from "node-fetch";
import pkg from "whatsapp-web.js";
import qrcode from "qrcode-terminal";
import chalk from "chalk";
```

```
const { Client, LocalAuth } = pkg;
```

◆ `node-fetch` — делает HTTP-запросы к API (аналог `fetch` в браузере).

◆ `whatsapp-web.js` — библиотека для управления твоим WhatsApp через браузер Chrome.

◆ `qrcode-terminal` — выводит QR-код прямо в консоли (для авторизации в WhatsApp).

◆ `chalk` — для красивого цветного вывода в консоли.

◆ `Client, LocalAuth` — классы из `whatsapp-web.js`:

- `Client` — главный объект для работы с WhatsApp.
- `LocalAuth` — сохраняет сессию, чтобы не сканировать QR-код каждый раз.

2. Конфигурация

```
const API_URL = "https://alerts.com.ua/api/states";
```

```
const CHAT_ID = "120363380889369101@g.us";
```

◆ `API_URL` — ссылка на открытый API, где обновляется статус тревог.

◆ `CHAT_ID` — ID группы WhatsApp, куда бот будет отправлять уведомления.

(оканчивается на `@g.us` — значит, это **групповой чат**).

3. Настройка WhatsApp клиента

```
console.clear();
```

```
console.log(chalk.cyan("🚀 Запуск WhatsApp клиента..."));
```

```
const client = new Client({
  authStrategy: new LocalAuth(),
  puppeteer: {
    headless: false,
    executablePath: "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe",
    args: ["--no-sandbox", "--disable-setuid-sandbox"]
  }
});
```

◆ Очищает консоль и пишет сообщение о запуске.

◆ Создаёт клиента WhatsApp с такими настройками:

- `authStrategy: new LocalAuth()` — хранит авторизацию локально.
- `headless: false` — Chrome будет открыт (чтобы ты видел процесс).
- `executablePath` — путь к твоему `Chrome.exe`.
- `args` — опции для стабильности в некоторых системах.

4. События клиента

```
client.on("qr", (qr) => {
  console.log(chalk.yellow("📄 Отсканируй QR-код WhatsApp:"));
  qrcode.generate(qr, { small: true });
});
```

```
});
```

Когда клиент запрашивает вход, он выводит **QR-код** в консоль, который нужно отсканировать с телефона в WhatsApp → меню "Связанные устройства".

```
client.on("ready", () => {  
  console.log(chalk.green("✔ WhatsApp клиент готов!"));  
  startMonitoring();  
});
```

После входа в систему бот готов к работе и запускает функцию `startMonitoring()` для мониторинга тревог.

1. Получение тревог из API

```
async function getAlerts() {  
  try {  
    const response = await fetch(API_URL);  
    if (!response.ok) {  
      console.log(chalk.red(`⚠ Ошибка ${response.status}: ${await response.text()}`));  
      return null;  
    }  
    return await response.json();  
  } catch (e) {  
    console.error(chalk.red("✖ Ошибка при запросе:"), e.message);  
    return null;  
  }  
}
```

Эта функция:

1. Делает запрос к API `alerts.com.ua/api/states`.
2. Проверяет статус ответа:
 - Если всё хорошо — возвращает JSON (объект с данными).
 - Если нет — выводит ошибку.
3. Если запрос упал (например, нет интернета) — ловит исключение и пишет сообщение.

6. Переменная состояния тревоги

```
let wasAlert = false;
```

Она хранит текущее состояние:

- `false` — тревоги нет.
- `true` — тревога активна.

Это нужно, чтобы не спамить одинаковыми сообщениями.

7. Основной цикл мониторинга

```
async function startMonitoring() {  
  console.log(chalk.blue("🔄 Мониторинг тревог (обновление каждые 30 сек)..."));  
  
  while (true) {  
    const data = await getAlerts();
```

◆ Цикл `while (true)` — бесконечный мониторинг (пока работает скрипт).

◆ Каждые 30 секунд делает запрос к API.

8. Проверка тревоги по региону

```
if (data && Array.isArray(data.states)) {  
  const region = data.states.find(r =>  
    r.name === "Запорізька область" || r.name_en === "Zaporizhia oblast"
```

```
);
```

◆ Проверяет, что в данных есть список `states`.

◆ Ищет объект, где название региона соответствует **Запорожской области** (по-украински или по-английски).

9. Логика начала тревоги

```
if (region) {
  if (region.alert && !wasAlert) {
    const msg = ` ${new Date().toLocaleTimeString()} Повітряна тривога в Запорізькій області.`;
    await client.sendMessage(CHAT_ID, msg);
    console.log(chalk.green(`✉ Відправлено у ${CHAT_ID}: ${msg}`));
    wasAlert = true;
  }
}
```

◆ Если тревога началась (`region.alert === true`)

и **до этого тревоги не было** (`wasAlert === false`),

→ бот отправляет сообщение о начале тревоги.

◆ После этого флаг `wasAlert` меняется на `true`.

10. Логика окончания тревоги

```
if (!region.alert && wasAlert) {
  const msg = ` ${new Date().toLocaleTimeString()} Відбій повітряної тривоги в Запорізькій області.`;
  await client.sendMessage(CHAT_ID, msg);
  console.log(chalk.yellow(`✉ Відправлено у ${CHAT_ID}: ${msg}`));
  wasAlert = false;
}
```

◆ Если тревога закончилась (`region.alert === false`)

и **до этого тревога была** (`wasAlert === true`),

→ бот отправляет сообщение о **відбої** и сбрасывает флаг.

11. Задержка между запросами

```
await new Promise(r => setTimeout(r, 30000)); // пауза 30 секунд
}
```

◆ После каждого цикла бот **ждёт 30 секунд** перед следующим запросом, чтобы не перегружать API и не спамить.

12. Запуск клиента

```
client.initialize();
```

◆ Инициализация клиента — запускает всё вышеописанное.

После этого:

- появится QR-код;
- произойдёт вход;
- запустится мониторинг тревог.

Код бота

```
import fetch from "node-fetch";
import pkg from "whatsapp-web.js";
import qrcode from "qrcode-terminal";
import chalk from "chalk";

const { Client, LocalAuth } = pkg;

const API_URL = "https://alerts.com.ua/api/states";
const CHAT_ID = "120363380889369101@g.us";

console.clear();
console.log(chalk.cyan("🚀 Запуск WhatsApp клиента..."));

const client = new Client({
  authStrategy: new LocalAuth(),
  puppeteer: {
    headless: false,
    executablePath: "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe",
    args: ["--no-sandbox", "--disable-setuid-sandbox"]
  }
});

client.on("qr", (qr) => {
  console.log(chalk.yellow("📄 Отсканируй QR-код WhatsApp:"));
  qrcode.generate(qr, { small: true });
});

client.on("ready", () => {
  console.log(chalk.green("✅ WhatsApp клиент готов!"));
  startMonitoring();
});

async function getAlerts() {
  try {
    const response = await fetch(API_URL);
    if (!response.ok) {
      console.log(chalk.red(`⚠️ Ошибка ${response.status}: ${await response.text()}`));
      return null;
    }
    return await response.json();
  } catch (e) {
    console.error(chalk.red("❌ Ошибка при запросе:"), e.message);
    return null;
  }
}

let wasAlert = false;

async function startMonitoring() {
  console.log(chalk.blue("🔄 Мониторинг тревог (обновление каждые 30 сек)..."));

  while (true) {
    const data = await getAlerts();

    if (data && Array.isArray(data.states)) {
      const region = data.states.find(r =>
```

```
    r.name === "Запорізька область" || r.name_en === "Zaporizhia oblast"
  );

  if (region) {
    if (region.alert && !wasAlert) {
      const msg = `🚨 ${new Date().toLocaleTimeString()} Повітряна тривога в
Запорізькій області.`;
      await client.sendMessage(CHAT_ID, msg);
      console.log(chalk.green(`✉ Відправлено у ${CHAT_ID}: ${msg}`));
      wasAlert = true;
    }

    if (!region.alert && wasAlert) {
      const msg = `🚨 ${new Date().toLocaleTimeString()} Відбій повітряної
тривоги в Запорізькій області.`;
      await client.sendMessage(CHAT_ID, msg);
      console.log(chalk.yellow(`✉ Відправлено у ${CHAT_ID}: ${msg}`));
      wasAlert = false;
    }
  } else {
    console.log(chalk.gray("❗ Область не найдена в ответе API."));
  }
}

await new Promise(r => setTimeout(r, 30000));
}

client.initialize();
```