

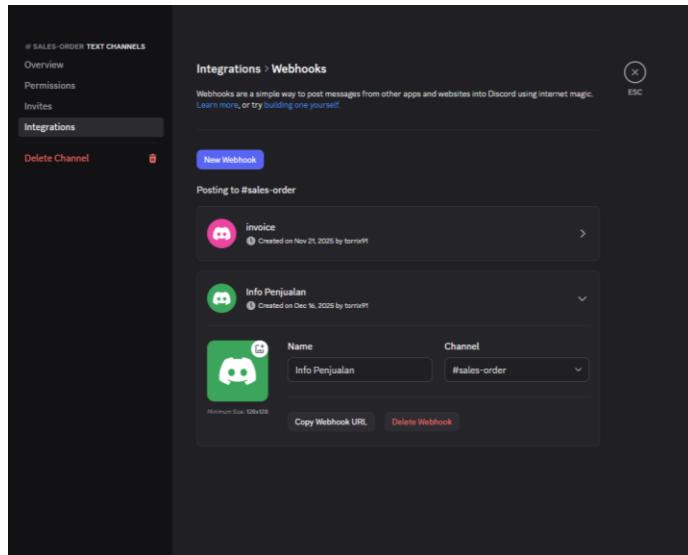
Dokumentasi

Sistem notifikasi otomatis ini dibangun menggunakan **n8n** sebagai middleware yang menghubungkan **ERPNext** (Sumber Data) dengan **Discord** (Penerima Notifikasi). Berikut adalah rincian fitur dan konfigurasi teknisnya.

1. Persiapan Lingkungan Sistem

Sebelum alur kerja berjalan, dilakukan konfigurasi pada kedua sisi platform:

1. **Integrasi Discord (Webhook Injection):** Fitur ini tidak memerlukan pembuatan *Bot Application* yang rumit, melainkan cukup menggunakan *Webhook Integration* pada level channel.
 - a. **Cara Konfigurasi:** Masuk ke *Channel Settings > Integrations > Webhooks > New Webhook*.



- b. **Output:** Diperoleh URL Endpoint (contoh: <https://discord.com/api/webhooks/...>) yang berfungsi sebagai alamat tujuan pengiriman pesan.
2. **Aktivasi Server n8n:** Server n8n dijalankan di lingkungan lokal (*localhost*) untuk memproses data.

- a. **Perintah:** Menjalankan `n8n start` dan `node index.js` pada terminal/CLI

```
PS C:\Users\N I T R O 5\discord-bot> node index.js
[dotenv@17.2.3] injecting env (2) from .env -- tip: ⚡ write to custom object with { processEnv: myObject }
✓ Bot sudah online! Login sebagai: ErpNext Assistant#3480
```

```

torix@LAPTOP-LK06L3TN: ~/fr > Windows PowerShell > + >
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\N I T R O S> n8n start
Initializing n8n process
n8n ready on ::1, port 5678

There are deprecations related to your environment variables. Please take the recommended actions to update your
ration:
- DB_SQLITE_POOL_SIZE -> Running SQLite without a pool of read connections is deprecated. Please set 'DB_SQLITE
ZE' to a value higher than zero. See: https://docs.n8n.io/hosting/configuration/environment-variables/database/#d
- N8N_RUNNERS_ENABLED -> Running n8n without task runners is deprecated. Task runners will be turned on by defa
future version. Please set 'N8N_RUNNERS_ENABLED=true' to enable task runners now and avoid potential issues in
re. Learn more: https://docs.n8n.io/hosting/configuration/task-runners/
- N8N_BLOCK_ENV_ACCESS_IN_NODE -> The default value of N8N_BLOCK_ENV_ACCESS_IN_NODE will be changed from false
in a future version. If you need to access environment variables from the Code Node or from expressions, please
BLOCK_ENV_ACCESS_IN_NODE=false. Learn more: https://docs.n8n.io/hosting/configuration/environment-variables/secu
- N8N_GIT_NODE_DISABLE_BARE_REPOS -> Support for bare repositories in the Git Node will be removed in a future
due to security concerns. If you are not using bare repositories in the Git Node, please set N8N_GIT_NODE_DISAB
EPOS=true. Learn more: https://docs.n8n.io/hosting/configuration/environment-variables/security/

[license SDK] Skipping renewal on init: license cert is not initialized
Version: 1.123.5
Start Active Workflows:
Activated workflow "My workflow" (ID: l4vRy8LTCplB3gV2)

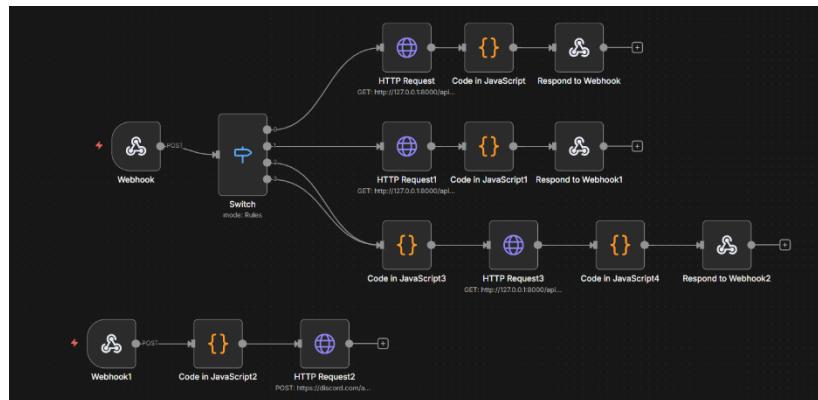
Editor is now accessible via:
http://localhost:5678

```

2. Detail Fitur Workflow (n8n)

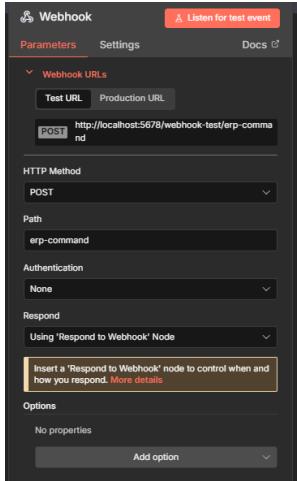
Alur kerja sistem terdiri dari tiga node utama yang berjalan secara sekuensial.

1. Gambaran Global Workflow



2. Fitur 1: Real-time Data Trigger (Webhook Node)

Fitur ini memungkinkan sistem menerima data detik itu juga saat tombol "Save" ditekan di ERPNext.

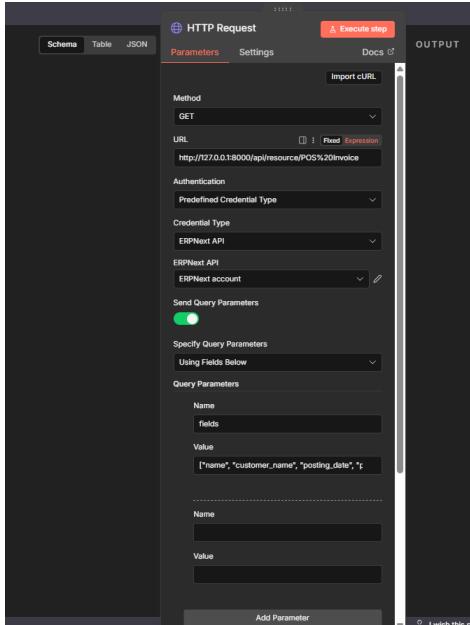


Method : POST.

Endpoint Path: /info-penjualan.

Integrasi ERPNext: URL Webhook ini ditanamkan pada "Webhook DocType" di ERPNext.

Fitur 2: Cek Pos invoice yang mana fitur akan menampilkan semua pos invoice yang ada di ErpNext (!tes)



Dari gambar diatas Konfigurasi HTTP Request untuk Mengambil Data Invoice

Deskripsi: "Langkah pertama dalam fitur pelaporan ini adalah mengambil data transaksi dari database ERPNext. Node ini dikonfigurasi menggunakan metode **GET** ke endpoint API POS Invoice. Untuk menjaga efisiensi data, request ini menggunakan parameter filter (fields) sehingga sistem hanya mengambil data yang diperlukan saja, yaitu: Nomor Invoice, Nama Customer, Tanggal Posting, Total Tagihan, dan Status Pembayaran. Hal ini mencegah pengambilan data berlebih yang dapat membebani server."

```

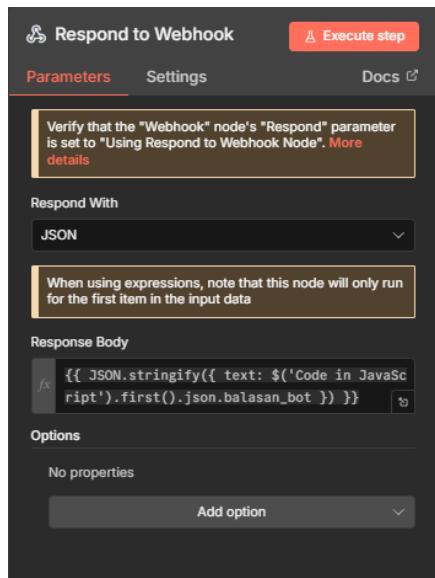
1 // 1. Ambil data
2 // Pastikan node sebelumnya bernama "HTTP Request"
3 const invoices = $`HTTP Request`.first().json.data;
4
5 const formatRupiah = (angka) => {
6   return new Intl.NumberFormat('id-ID').format(angka);
7 };
8
9 // 2. Buat Pesan
10 let pesan = `| ${`+LAPORAN TRANSAKSI TERAKHIR`}|\n`;
11 pesan += "-----\n";
12
13 // 3. Batas Aman Discord (sisakan ruang untuk footer)
14 const BATAS_MAX = 1900;
15
16 if (invoices && invoices.length > 0) {
17   for (let i = 0; i < invoices.length; i++) {
18     const inv = invoices[i];
19
20     // Siapkan teks per item
21     let itemText = `| ${`+${inv.item} - ${inv.quantity} ${inv.item}`}`;
22     pesan += itemText + "\n";
23   }
24 }

```

Type \$ for a list of special vars/methods. Debug by using console.log() statements and viewing their output in the browser console.

Logika Pemformatan Laporan & Proteksi Limitasi Karakter Deskripsi: "Data mentah yang diambil dari ERPNext kemudian diproses menggunakan node JavaScript. Kode ini memiliki tiga fungsi krusial:

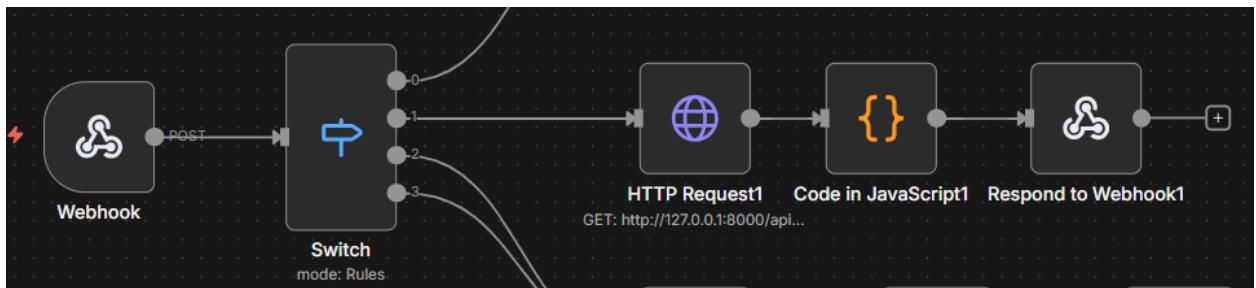
- **Formatting Mata Uang & Ikon:** Mengubah angka nominal (misal: 100000) menjadi format Rupiah (Rp 100.000) dan memberikan ikon status (✓ untuk *Paid*, ⏳ untuk *Unpaid*) agar laporan visual dan mudah dibaca.
- **Looping Data:** Melakukan perulangan (*loop*) untuk menyusun daftar transaksi satu per satu menjadi sebuah daftar panjang.
- **Proteksi Karakter (Safety Limit):** Discord memiliki batasan 2000 karakter per pesan. Kode ini secara cerdas menghitung panjang pesan (pesan.length). Jika pesan sudah mendekati batas aman (1900 karakter), sistem akan otomatis berhenti menambahkan data dan memberikan penutup '*...data terpotong*' untuk mencegah *error* pengiriman akibat pesan yang terlalu panjang."



Pengiriman Respon Balik ke User (Respond to Webhook) Deskripsi: "Setelah pesan tersusun rapi, langkah terakhir adalah mengirimkannya kembali ke platform Discord

untuk ditampilkan kepada user. Node Respond to Webhook ini berfungsi mengemas variabel balasan_bot (hasil olahan JavaScript tadi) menjadi format JSON final. Node ini memastikan bahwa pesan laporan transaksi muncul sebagai balasan langsung dari perintah yang diketikkan oleh user sebelumnya."

Fitur 3 : cek stok maka mengirimkan semua stok di semua gudang (!stok)



Fitur ini berperan layaknya "**Kepala Gudang Digital**" yang memiliki akses kunci ke seluruh lokasi penyimpanan. Jika biasanya sales harus menelpon orang gudang satu per satu untuk bertanya "*Barang X ada di Jakarta atau Surabaya?*", sistem ini bisa menjawabnya dalam hitungan detik untuk seluruh cabang sekaligus.

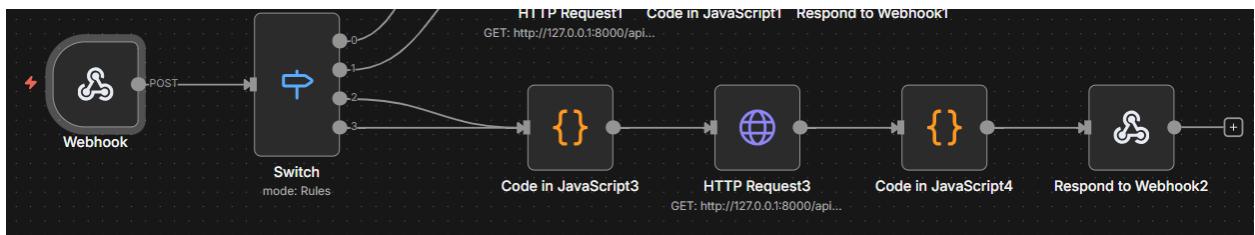
Proses kerjanya dibagi menjadi 3 tahap:

1. Mengintip ke Dalam Gudang (Input)
Deskripsi: "Ketika ada permintaan cek stok, sistem ini langsung 'membuka' catatan gudang pusat (tabel Bin di ERPNext). Ia tidak hanya mengecek satu lokasi, tapi langsung memindai data ketersediaan barang di **seluruh gudang** yang terdaftar di sistem. Jadi, data yang diambil adalah data *real* yang ada di rak penyimpanan saat itu juga."

2. Menyusun Daftar Stok (Proses)
Deskripsi: "Data stok mentah dari berbagai gudang biasanya membingungkan jika dilihat langsung. Di tahap ini, sistem bekerja merapikan data tersebut:
 - Ia mengelompokkan barang berdasarkan nama gudangnya (Misal: *Gudang Jakarta, Gudang Surabaya, dsb*).
 - Ia menghitung sisa stok (*Actual Qty*) dan memastikannya tertulis dengan jelas (Misal: **Sisa Stok: 100 Unit**).
 - Jika stok kosong, ia akan memberikan tanda peringatan agar tim sales tahu."

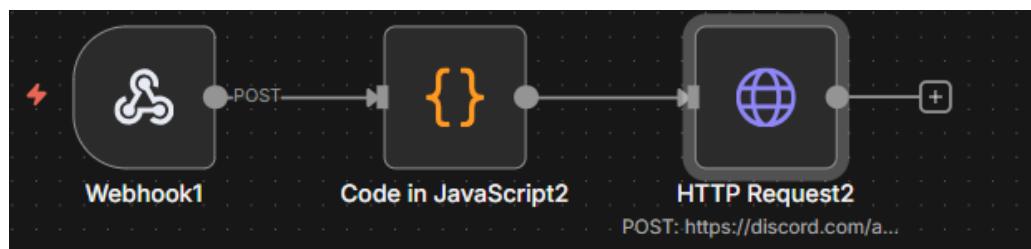
3. Melaporkan ke Grup (Output)
Deskripsi: "Setelah daftar stok tersusun rapi per gudang, sistem langsung menyerahkan laporan tersebut ke Discord. Sales bisa langsung melihat daftar lengkap: barang apa ada di gudang mana dan berapa sisanya, tanpa perlu membuka aplikasi ERP sama sekali."

Fitur 4 : pengecekan harga barang. (!harga, !harga ac, !harga Dummy)



- b. **Logika Pemrosesan & Navigasi Halaman (Code Node) Judul:** Logika Pencarian, Pemformatan, dan Paginasi Data **Deskripsi:** "Node JavaScript ini berfungsi sebagai otak dari fitur pencarian harga. Kode program dirancang untuk menangani tiga fungsi utama:
 1. **Pengambilan & Validasi Data:** Sistem mengambil data mentah dari respon API ERPNext (HTTP Request3). Jika barang tidak ditemukan, sistem langsung mengembalikan pesan error yang informatif.
 2. **Sistem Paginasi (Pagination Logic):** Untuk menangani jumlah barang yang banyak, kode membatasi tampilan hanya **5 item per halaman** (limit = 5). Sistem membaca parameter halaman saat ini dari input tombol user (page_1, page_2, dst.) dan memotong data array (slice) sesuai indeks halaman tersebut.
 3. **Generator Tombol Navigasi:** Kode secara otomatis menentukan apakah tombol 'Prev' atau 'Next' perlu dimunculkan. Contohnya, tombol 'Next' hanya akan aktif jika sisa barang masih ada (hasNext = listBarang.length > endIndex). Sistem juga menyimpan 'memori' pencarian (currentQuery) agar saat pindah halaman, bot tidak lupa barang apa yang sedang dicari user."

Fitur 5 : Sistem Notifikasi Otomatis sales invoice



Sistem ini bekerja layaknya "**Kurir Digital**" yang tidak pernah tidur. Tugas utamanya adalah memantau transaksi dan langsung melaporkannya ke grup Discord setiap kali ada penjualan baru, tanpa perlu ada orang yang melapor secara manual.

Proses kerjanya terdiri dari 3 tahapan sederhana:

1. Menerima Laporan (Input)

Deskripsi: "Bayangkan sistem ini memiliki 'telinga' yang selalu siaga. Setiap kali admin menekan tombol **Save** pada Sales Invoice di ERPNext, sistem ini langsung 'mendengar' dan menangkap data penjualan tersebut detik itu juga. Hal ini memastikan tidak ada satu pun transaksi yang terlewat atau terlambat dilaporkan."

2. Merapikan Tulisan (Proses)

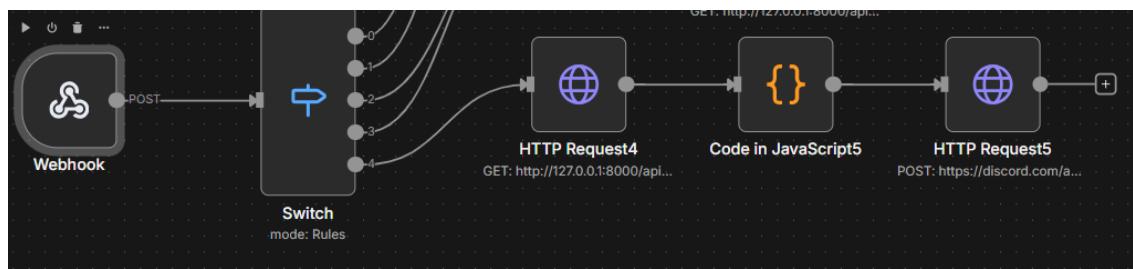
Deskripsi: "Data mentah yang keluar dari komputer biasanya berantakan (hanya berupa angka-angka polos). Di tahap ini, sistem bertugas layaknya seorang sekretaris yang merapikan laporan:

- Mengubah angka mentah (misal: 100000) menjadi format Rupiah (**Rp 100.000**) agar enak dibaca.
- Menyusun kalimat menjadi rapi, misalnya: '*Ada penjualan baru nih, dari Customer A!*'.
- Memastikan pesan siap dibaca oleh manusia."

3. Mengirim ke Grup (Output)

Deskripsi: "Setelah pesan tersusun rapi, tugas terakhir adalah mengantarkannya. Sistem langsung mengirimkan pesan tersebut ke saluran (channel) Discord tim Sales. Hasilnya, notifikasi *pop-up* muncul di HP seluruh anggota tim secara bersamaan, menciptakan transparansi dan kecepatan informasi yang *real-time*."

Fitur 6 : Monitoring Top 5 Item



Sistem ini bekerja layaknya "Asisten Data Pribadi" yang siap diperintah kapan saja. Tugas utamanya adalah mengambil data produk dari gudang digital (ERPNext) dan menyajikannya menjadi daftar peringkat secara instan, tanpa perlu user membuka aplikasi ERP yang rumit.

1. Menerima Perintah (Input)

Deskripsi: "Bayangkan sistem ini memiliki 'mata' yang selalu mengawasi obrolan di Discord. Ia diam saja saat obrolan biasa terjadi, tapi begitu ada yang mengetik perintah khusus **!top**, sistem langsung 'terbangun'. Ia paham bahwa user sedang membutuhkan data produk saat itu juga, lalu ia

langsung menghubungi server database ERPNext untuk meminta data barang terbaru."

2. Meracik Data (Proses)

Deskripsi: "Data mentah yang diambil dari database awalnya sangat kaku dan membingungkan (hanya deretan kode dan angka polos). Di tahap ini, sistem bertugas layaknya seorang editor yang merapikan naskah:

- Memilih dan memilah data untuk mencari 5 produk prioritas.
- Menyusun tata letak teks agar terlihat rapi

3. Menyajikan Informasi (Output)

Deskripsi: "Setelah data tersusun rapi, tugas terakhir adalah menyajikannya. Sistem langsung membalas chat tersebut di Discord dengan daftar '**Top 5 Produk**' yang sudah diformat. Hasilnya, user mendapatkan informasi harga dan kode barang yang akurat dalam hitungan detik, tanpa harus repot login atau mencari manual di komputer."