**Dokumentasi API Billing System V2.0.1**

Selamat datang di dokumentasi resmi untuk API Billing System Artacomindo Jejaring Nusa. API ini dirancang untuk mengelola seluruh alur kerja ISP, mulai dari manajemen pengguna, pelanggan, hingga penagihan.

**Base URL**: http://127.0.0.1:8000

**1. Otentikasi**

Saat ini, API belum menerapkan sistem otentikasi. Semua endpoint dapat diakses secara publik. Implementasi token (JWT) akan menjadi langkah selanjutnya.

**2. Manajemen Pengguna & Peran**

**Roles**

Mengelola peran pengguna (misalnya, admin, sales, finance).

**POST /roles/**

Membuat peran baru.

* **Request Body**:
* {
* "name": "finance"
* }
* **Success Response (201 Created)**:
* {
* "name": "finance",
* "id": 3
* }
* **Error Response (409 Conflict)**: Terjadi jika nama role sudah ada.
* {
* "detail": "Role dengan nama 'finance' sudah ada."
* }

**GET /roles/**

Mendapatkan semua peran yang ada.

* **Success Response (200 OK)**:
* [
* { "name": "administrator", "id": 1 },
* { "name": "noc", "id": 2 },
* { "name": "finance", "id": 3 }
* ]

**PATCH /roles/{role\_id}**

Memperbarui nama peran.

* **Request Body**:
* {
* "name": "finance\_staff"
* }
* **Success Response (200 OK)**:
* {
* "name": "finance\_staff",
* "id": 3
* }

**DELETE /roles/{role\_id}**

Menghapus sebuah peran.

* **Success Response**: 204 No Content (Tidak ada body respons).
* **Catatan**: Jika ada user yang masih menggunakan role ini, role\_id pada user tersebut akan menjadi NULL karena kita menggunakan ON DELETE SET NULL.

**Users**

Mengelola akun pengguna yang dapat mengakses sistem.

**POST /users/**

Membuat pengguna baru.

* **Request Body**:
* {
* "name": "Budi Finance",
* "email": "budi.finance@example.com",
* "password": "password\_aman\_123",
* "role\_id": 3
* }
* **Success Response (201 Created)**: (Password tidak pernah dikembalikan)
* {
* "name": "Budi Finance",
* "email": "budi.finance@example.com",
* "id": 2,
* "role\_id": 3,
* "created\_at": "2025-07-17T10:30:00.123Z",
* "updated\_at": "2025-07-17T10:30:00.123Z"
* }

**GET /users/**

Mendapatkan semua pengguna.

* **Success Response (200 OK)**: Array berisi objek pengguna.

**GET /users/{user\_id}**

Mendapatkan detail satu pengguna.

* **Success Response (200 OK)**: Objek pengguna tunggal.

**PATCH /users/{user\_id}**

Memperbarui data pengguna.

* **Request Body**:
* {
* "name": "Budi Hartono",
* "role\_id": 2
* }
* **Success Response (200 OK)**: Objek pengguna yang telah diperbarui.

**DELETE /users/{user\_id}**

Menghapus pengguna.

* **Success Response**: 204 No Content.

**3. Manajemen Pelanggan & Layanan**

Ini adalah alur kerja utama untuk operasional ISP.

**Harga Layanan (Brand)**

Mengelola brand/area layanan dan pajaknya.

* POST /harga\_layanan/: Membuat brand baru.
* GET /harga\_layanan/: Mendapatkan semua brand.
* GET /harga\_layanan/{id\_brand}: Mendapatkan detail satu brand.
* PATCH /harga\_layanan/{id\_brand}: Memperbarui nama atau pajak brand.
* DELETE /harga\_layanan/{id\_brand}: Menghapus brand. **Peringatan**: Ini akan menghapus semua PaketLayanan yang terhubung.

**Paket Layanan**

Mengelola katalog paket internet yang ditawarkan untuk setiap brand.

* POST /paket\_layanan/: Membuat paket baru.
  + **Request Body**:
  + {
  + "id\_brand": "ajn-01",
  + "nama\_paket": "Home Premium 75Mbps",
  + "kecepatan": 75,
  + "harga": 650000
  + }
* GET /paket\_layanan/: Mendapatkan semua paket.
* GET /paket\_layanan/{paket\_id}: Mendapatkan detail satu paket.
* PATCH /paket\_layanan/{paket\_id}: Memperbarui detail paket.
* DELETE /paket\_layanan/{paket\_id}: Menghapus paket.

**Pelanggan**

Mengelola data pelanggan.

* POST /pelanggan/: Membuat pelanggan baru.
* GET /pelanggan/: Mendapatkan semua pelanggan.
* GET /pelanggan/{pelanggan\_id}: Mendapatkan detail satu pelanggan.
* PATCH /pelanggan/{pelanggan\_id}: Memperbarui data pelanggan.
* DELETE /pelanggan/{pelanggan\_id}: Menghapus pelanggan. **Peringatan**: Ini akan menghapus DataTeknis dan Langganan yang terhubung.

**Data Teknis**

Mengelola data teknis instalasi untuk setiap pelanggan.

* POST /data\_teknis/: Membuat data teknis baru.
* GET /data\_teknis/: Mendapatkan semua data teknis.
* GET /data\_teknis/{data\_teknis\_id}: Mendapatkan detail satu data teknis.
* PATCH /data\_teknis/{data\_teknis\_id}: Memperbarui data teknis.
* DELETE /data\_teknis/{data\_teknis\_id}: Menghapus data teknis.

**Langganan**

Mengelola langganan aktif pelanggan terhadap sebuah paket layanan.

* POST /langganan/: Membuat (mengaktifkan) langganan baru.
  + **Request Body**:
  + {
  + "pelanggan\_id": 1,
  + "paket\_layanan\_id": 2
  + }
* GET /langganan/: Mendapatkan semua data langganan.
* GET /langganan/{langganan\_id}: Mendapatkan detail satu langganan.
* PATCH /langganan/{langganan\_id}: Memperbarui langganan (misalnya mengubah status).
* DELETE /langganan/{langganan\_id}: Menghapus (menonaktifkan) langganan.

**Dokumentasi API Sistem Billing Xendit**

**Deskripsi**

API ini merupakan bagian dari sistem billing terintegrasi dengan Xendit untuk mengelola invoice, pelanggan, langganan, dan layanan terkait. API ini dibangun menggunakan FastAPI dan menggunakan database relasional dengan SQLAlchemy.

**Base URL**

http://localhost:8000

**Endpoint**

**1. Root Endpoint**

* **URL**: /
* **Method**: GET
* **Deskripsi**: Menampilkan pesan selamat datang ke API Billing System.
* **Response**:
* {
* "message": "Selamat datang di API Billing System"
* }
* **Status Code**: 200 OK

**2. Generate Invoice**

* **URL**: /invoices/generate
* **Method**: POST
* **Deskripsi**: Membuat invoice baru untuk langganan tertentu dan mendaftarkannya ke Xendit untuk mendapatkan payment link.
* **Request Body**:
* {
* "langganan\_id": 2
* }
  + langganan\_id (integer, required): ID langganan yang akan dibuatkan invoice.
* **Response**:
* {
* "invoice\_number": "INV-20250717-0EEFF6",
* "pelanggan\_id": 5,
* "total\_harga": 388500.0,
* "tgl\_invoice": "2025-07-17",
* "tgl\_jatuh\_tempo": "2025-07-31",
* "status\_invoice": "Belum Dibayar",
* "id": 8,
* "id\_pelanggan": "CUST-5",
* "brand": "Jakinet",
* "no\_telp": "08986937819",
* "email": "budi@test.com",
* "payment\_link": "https://checkout-staging.xendit.co/web/6878b3d4f72950f656a0e46c",
* "expiry\_date": "2025-07-18 15:27:00",
* "xendit\_id": "6878b3d4f72950f656a0e46c",
* "xendit\_external\_id": "INV-20250717-0EEFF6",
* "paid\_amount": null,
* "paid\_at": null,
* "created\_at": "2025-07-17T15:26:59",
* "updated\_at": "2025-07-17T15:27:00"
* }
* **Status Code**:
  + 201 Created: Invoice berhasil dibuat.
  + 404 Not Found: Langganan tidak ditemukan.
  + 500 Internal Server Error: Gagal mengintegrasikan dengan Xendit.
* **Catatan**: Pastikan langganan\_id valid dan data pelanggan serta paket layanan sudah ada di database.

**3. Handle Xendit Callback**

* **URL**: /xendit-callback
* **Method**: POST
* **Deskripsi**: Endpoint untuk menangani callback dari Xendit ketika invoice dibayar atau kedaluwarsa.
* **Request Body** (Contoh Payload dari Xendit):
* {
* "external\_id": "INV-20250717-0EEFF6",
* "paid\_amount": 388500,
* "paid\_at": "2025-07-17T15:30:00Z",
* "status": "PAID"
* }
  + external\_id (string, required): Nomor invoice yang sama dengan invoice\_number.
  + paid\_amount (float, optional): Jumlah yang dibayar.
  + paid\_at (string, optional): Waktu pembayaran dalam format ISO 8601.
  + status (string, required): Status pembayaran (misalnya "PAID" atau "EXPIRED").
* **Response**:
* {
* "message": "Callback processed successfully",
* "invoice\_number": "INV-20250717-0EEFF6",
* "status": "PAID"
* }
* **Status Code**:
  + 200 OK: Callback diproses dengan sukses.
  + 400 Bad Request: External ID tidak ditemukan dalam payload.
  + 404 Not Found: Invoice dengan external\_id tidak ditemukan.
  + 500 Internal Server Error: Error saat memproses callback.
* **Catatan**: Endpoint ini harus dikonfigurasi di dashboard Xendit untuk mengirim callback.

**4. Get All Invoices**

* **URL**: /invoices/
* **Method**: GET
* **Deskripsi**: Mengambil daftar semua invoice.
* **Response**:
* [
* {
* "invoice\_number": "INV-20250717-0EEFF6",
* "pelanggan\_id": 5,
* "total\_harga": 388500.0,
* "tgl\_invoice": "2025-07-17",
* "tgl\_jatuh\_tempo": "2025-07-31",
* "status\_invoice": "Belum Dibayar",
* "id": 8,
* "id\_pelanggan": "CUST-5",
* "brand": "Jakinet",
* "no\_telp": "08986937819",
* "email": "budi@test.com",
* "payment\_link": "https://checkout-staging.xendit.co/web/6878b3d4f72950f656a0e46c",
* "expiry\_date": "2025-07-18 15:27:00",
* "xendit\_id": "6878b3d4f72950f656a0e46c",
* "xendit\_external\_id": "INV-20250717-0EEFF6",
* "paid\_amount": null,
* "paid\_at": null,
* "created\_at": "2025-07-17T15:26:59",
* "updated\_at": "2025-07-17T15:27:00"
* }
* ]
* **Status Code**:
  + 200 OK: Daftar invoice berhasil diambil.

**5. Get Invoice by ID**

* **URL**: /invoices/{invoice\_id}
* **Method**: GET
* **Deskripsi**: Mengambil detail satu invoice berdasarkan ID.
* **Parameter**:
  + invoice\_id (integer, path): ID invoice yang ingin diambil.
* **Response**: Sama dengan struktur di endpoint "Get All Invoices" untuk satu item.
* **Status Code**:
  + 200 OK: Invoice ditemukan.
  + 404 Not Found: Invoice tidak ditemukan.

**6. Update Invoice**

* **URL**: /invoices/{invoice\_id}
* **Method**: PUT
* **Deskripsi**: Memperbarui data invoice secara manual (untuk tujuan testing atau admin).
* **Parameter**:
  + invoice\_id (integer, path): ID invoice yang akan diperbarui.
* **Request Body**:
* {
* "status\_invoice": "Lunas",
* "paid\_amount": 388500.0
* }
  + Field yang bisa diupdate tergantung pada skema InvoiceUpdate.
* **Response**: Sama dengan struktur invoice yang diperbarui.
* **Status Code**:
  + 200 OK: Invoice berhasil diperbarui.
  + 404 Not Found: Invoice tidak ditemukan.

**7. Delete Invoice**

* **URL**: /invoices/{invoice\_id}
* **Method**: DELETE
* **Deskripsi**: Menghapus invoice dari database (gunakan dengan hati-hati).
* **Parameter**:
  + invoice\_id (integer, path): ID invoice yang akan dihapus.
* **Response**: Tidak ada konten.
* **Status Code**:
  + 204 No Content: Invoice berhasil dihapus.
  + 404 Not Found: Invoice tidak ditemukan.

**8. Test Xendit Callback**

* **URL**: /invoices/test-callback
* **Method**: POST
* **Deskripsi**: Endpoint untuk menguji callback Xendit secara manual.
* **Request Body**: Sama dengan struktur callback Xendit (lihat endpoint "Handle Xendit Callback").
* **Response**: Sama dengan endpoint "Handle Xendit Callback".
* **Status Code**: Sama dengan endpoint "Handle Xendit Callback".

**Persyaratan**

* **Autentikasi**: Belum diimplementasikan (opsional untuk ditambahkan).
* **Database**: MySQL dengan tabel yang sesuai (misalnya invoices, langganan, dll.).
* **Lingkungan**: Python 3.x dengan dependensi FastAPI, SQLAlchemy, httpx, dan pytz.

**Catatan Penting**

* Pastikan konfigurasi Xendit (API key dan callback URL) sudah benar di file config.py.
* Endpoint callback harus dapat diakses dari luar jika digunakan di produksi (gunakan ngrok untuk pengujian lokal).
* Semua waktu disimpan dalam zona WIB (Asia/Jakarta).

**Dokumentasi API Mikrotik Billing System V2.0.1**

## Manajemen Server Mikrotik

### Mikrotik Servers

Mengelola data koneksi ke router Mikrotik Anda.

#### POST /mikrotik\_servers/

Mendaftarkan server Mikrotik baru.

* **Request Body**:
* {
* "name": "OLT-Gedung-A",
* "host\_ip": "192.168.88.1",
* "username": "admin",
* "password": "password\_router\_anda",
* "port": 8728
* }

#### POST /mikrotik\_servers/{server\_id}/test\_connection

Menguji koneksi ke server Mikrotik.

* **Success Response (200 OK)**:
* {
* "status": "success",
* "message": "Berhasil terhubung ke Mikrotik 'OLT-Gedung-A'.",
* "router\_os\_version": "7.1.1"
* }

#### Endpoint CRUD Lainnya

* GET /mikrotik\_servers/: Mendapatkan semua server.
* GET /mikrotik\_servers/{server\_id}: Mendapatkan detail satu server.
* PATCH /mikrotik\_servers/{server\_id}: Memperbarui data server.
* DELETE /mikrotik\_servers/{server\_id}: Menghapus data server.