מדריך: אינטגרציית מערכת Tranzila עם Lovable - סגירת הזמנה באמצעות דף Notify

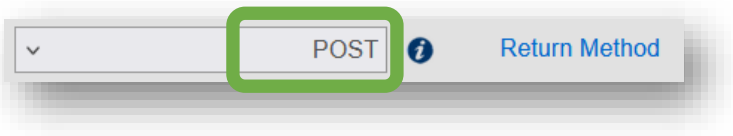
ה-Prompt להלן יוצר דף Notify עם Webhook (כתובת URL) עבור האתר או האפליקציה שלך ומאפשר לו לקבל תשובה חזרה עם כל פרטי העסקה כפי שבוצעו, בנוסף לפרמטרים נוספים, ותשובה אם העסקה עברה או לא.

דרישות לעבודה עם האינטגרציה של טרנזילה ל-Lovable לסגירת הזמנה באמצעות דף Notify

- האתר או האפליקציה שלך צריכים להיות מחוברים לדף **התשלום iFrame של טרנזילה** באמצעות האינטגרציה שלנו. במידה וטרם ביצעת את האינטגרציה – יש להריץ קודם את ה-Prompt של האינטגרציה.
- באתר או באפליקציה שלך צריך להגדיר **Back Order** ב-Lovable (לדוגמה: מערכת לניהול הזמנות, שם ניתן לראות את סטטוס ההזמנות שבוצעו – "ממתין לתשלום" או "שולם").

יצירת דף ה-Notify

לפני שמתחילים, ניכנס ל**הגדרות דף הסליקה** של המסוף, נבחר ב**הגדרות מתקדמות** ונוודא שה-Return Method מוגדר על **Post**:



לסיום, נעתיק את ה-**Prompt** הארוך שצירפנו להלן, נדביק אותו ב**חלון השאילתה** של האפליקציה שלך ונלחץ על **החץ השחור**:

```
Instructions for Creating a Tranzila Notify URL on Supabase:

The Tranzila notify URL is a webhook endpoint that Tranzila calls after processing a payment to inform your system of the transaction result. This allows you to automatically update order status based on actual payment outcomes.

Implementation Steps:
1. Create the Edge Function
Create a new Supabase Edge Function at supabase/functions/tranzila-notify/index.ts:
Key Components:
•  CORS Headers: Required for web requests
•  Form Data Parser: Tranzila sends data as application/x-www-form-urlencoded
•  Order Matching Logic: Find the correct order to update
•  Status Update Logic: Mark orders as 'paid' or 'failed'
•  Response: Return 'OK' to acknowledge receipt
Critical Elements:
// CORS headers for web requests
const corsHeaders = {
  'Access-Control-Allow-Origin': '*',
  'Access-Control-Allow-Headers': 'authorization, x-client-info, apikey, content-type',
}

// Tranzila data structure
```

```
interface TranzilaNotification {
  Response: string;      // '000' or '0' = success, other = failure
  supplier: string;      // Your Tranzila supplier name
  sum: string;          // Transaction amount
  currency: string;     // Currency code
  myid?: string;        // Custom ID you can pass
  ConfirmationCode?: string; // Success confirmation
  TranzilaTK?: string;  // Transaction token
  [key: string]: any;   // Additional Tranzila fields
}

// Order matching strategy:
// 1. First try by order ID (if passed in myid)
// 2. Fallback to amount + recent timestamp matching
```

2. Configure Public Access

Update supabase/config.toml to make the function publicly accessible:

```
project_id = "your-project-id"

[functions.tranzila-notify]
verify_jwt = false
```

Why this is needed:

- Tranzila calls your webhook directly (no JWT authentication)
- Without this, the function returns 401 Unauthorized
- This makes the endpoint public but only accepts POST requests

3. Add Notify URL to Payment Form

In your payment component, add the notify URL parameter to the Tranzila form:

```
<input
  type="hidden"
  name="notify_url_address"
  value="https://your-project-id.supabase.co/functions/v1/tranzila-notify"
/>
```

URL Format:

- https://[PROJECT_ID].supabase.co/functions/v1/[FUNCTION_NAME]
- Replace [PROJECT_ID] with your actual Supabase project ID
- Replace [FUNCTION_NAME] with the function name (tranzila-notify)

4. Order Creation Strategy

Before Payment:

- Create orders with status 'pending' immediately when payment process starts
- Store order ID for potential matching (optional)
- Ensure orders have the exact amount that will be charged

Order Matching Logic:

```
// Primary: Match by order ID
if (orderId) {
```

```
  const { data: orderById } = await supabase
    .from('orders')
    .select('*')
    .eq('id', orderId)
    .eq('status', 'pending')
    .single();
}

// Fallback: Match by amount and recent timestamp
const { data: pendingOrders } = await supabase
  .from('orders')
  .select('*')
  .eq('status', 'pending')
  .eq('total_amount_nis', Math.round(sum))
  .gte('created_at', new Date(Date.now() - 30 * 60 * 1000).toISOString())
  .order('created_at', { ascending: false })
  .limit(1);
```

5. Response Codes

Tranzila Response Codes:

- '000' or '0' = Successful payment
- Any other value = Failed payment

Your Function Response:

- Return 'OK' with status 200 to acknowledge receipt
- Return appropriate error codes (404, 500) for debugging
- Always include CORS headers in responses

6. Security Considerations

What's Safe:

- The function is public but only accepts POST requests
- Tranzila data includes transaction tokens for verification
- Order matching prevents updating wrong orders

What to Watch:

- Don't expose sensitive order data in responses
- Log all notifications for debugging
- Validate amounts match expected values
- Use time-window for order matching (e.g., last 30 minutes)

7. Testing and Debugging

Monitoring:

- Check Edge Function logs: https://supabase.com/dashboard/project/[PROJECT_ID]/functions/tranzila-notify/logs
- Look for successful notifications and order updates
- Monitor for 401 errors (JWT issues) or 404 errors (order not found)

Common Issues:

- 401 Unauthorized: Missing verify_jwt = false in config.toml
- No orders found: Orders not created as 'pending' or amount mismatch
- CORS errors: Missing CORS headers or OPTIONS handler

8. Complete Flow Summary

1. User initiates payment → Create order with status 'pending'
2. Redirect to Tranzila with notify_url_address parameter
3. User completes payment on Tranzila
4. Tranzila posts transaction result to your notify URL
5. Your function receives notification, finds matching order, updates status

Order is now marked as 'paid' or 'failed' based on actual payment result.