// /api/business-register.js — endpoint rejestracji firm bez weryfikacji NIP import { createClient } from ‘@supabase/supabase-js’;

const supabase = createClient( process.env.SUPABASE\_URL || ‘https://xdhlztmjktminrwmzcpl.supabase.co’, process.env.SUPABASE\_ANON\_KEY || ‘eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InhkaGx6dG1qa3RtaW5yd216Y3BsIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NTY3MjgwMTEsImV4cCI6MjA3MjMwNDAxMX0.EmvBqbygr4VLD3PXFaPjbChakRi5YtSrxp8e\_K7ZyGY’ );

export default async function handler(req, res) { // CORS headers res.setHeader(“Access-Control-Allow-Origin”, “\*“); res.setHeader(”Access-Control-Allow-Methods”, “POST,OPTIONS”); res.setHeader(“Access-Control-Allow-Headers”, “Content-Type, Authorization”);

if (req.method === “OPTIONS”) { return res.status(200).end(); }

if (req.method !== “POST”) { return res.status(405).json({ ok: false, error: “METHOD\_NOT\_ALLOWED”, message: “Tylko metoda POST jest obsługiwana” }); }

try { const { user\_id, name, category\_id, nip, address, city, phone, email, description } = req.body;

// Walidacja wymaganych pól  
if (!user\_id || !name || !category\_id) {  
 return res.status(400).json({  
 ok: false,  
 error: "MISSING\_REQUIRED\_FIELDS",  
 message: "Wymagane pola: user\_id, name, category\_id"  
 });  
}  
  
// Sprawdzenie czy użytkownik istnieje  
const { data: existingUser, error: userError } = await supabase  
 .from('profiles')  
 .select('id, email, user\_type, business\_id')  
 .eq('id', user\_id)  
 .single();  
  
if (userError || !existingUser) {  
 return res.status(404).json({  
 ok: false,  
 error: "USER\_NOT\_FOUND",  
 message: "Użytkownik nie został znaleziony"  
 });  
}  
  
// Sprawdzenie czy użytkownik już ma firmę  
if (existingUser.business\_id) {  
 return res.status(400).json({  
 ok: false,  
 error: "USER\_ALREADY\_HAS\_BUSINESS",  
 message: "Użytkownik już posiada zarejestrowaną firmę"  
 });  
}  
  
// Sprawdzenie czy kategoria istnieje  
const { data: category, error: categoryError } = await supabase  
 .from('business\_categories')  
 .select('id, name')  
 .eq('id', category\_id)  
 .single();  
  
if (categoryError || !category) {  
 return res.status(400).json({  
 ok: false,  
 error: "INVALID\_CATEGORY",  
 message: "Nieprawidłowa kategoria biznesu"  
 });  
}  
  
// Tworzenie nowej firmy (NIP jest opcjonalny)  
const businessData = {  
 name,  
 nip: nip || null, // NIP może być pusty  
 category\_id,  
 owner\_id: user\_id,  
 address: address || null,  
 city: city || null,  
 phone: phone || null,  
 email: email || existingUser.email,  
 description: description || null,  
 is\_verified: true, // Automatycznie weryfikowane (bez weryfikacji NIP)  
 is\_active: true // Automatycznie aktywne  
};  
  
const { data: newBusiness, error: businessError } = await supabase  
 .from('businesses')  
 .insert(businessData)  
 .select()  
 .single();  
  
if (businessError) {  
 console.error('Business creation error:', businessError);  
 return res.status(500).json({  
 ok: false,  
 error: "BUSINESS\_CREATION\_FAILED",  
 message: "Nie udało się utworzyć firmy",  
 details: businessError.message  
 });  
}  
  
// Aktualizacja profilu użytkownika - zmiana na konto biznesowe  
const { error: profileUpdateError } = await supabase  
 .from('profiles')  
 .update({  
 user\_type: 'business\_owner',  
 business\_id: newBusiness.id,  
 business\_role: 'owner',  
 business\_permissions: {  
 manage\_orders: true,  
 manage\_menu: true,  
 view\_analytics: true,  
 manage\_staff: true  
 },  
 updated\_at: new Date().toISOString()  
 })  
 .eq('id', user\_id);  
  
if (profileUpdateError) {  
 console.error('Profile update error:', profileUpdateError);  
 // Jeśli aktualizacja profilu się nie powiodła, usuń utworzoną firmę  
 await supabase.from('businesses').delete().eq('id', newBusiness.id);  
   
 return res.status(500).json({  
 ok: false,  
 error: "PROFILE\_UPDATE\_FAILED",  
 message: "Nie udało się zaktualizować profilu użytkownika",  
 details: profileUpdateError.message  
 });  
}  
  
// Pobieranie zaktualizowanych danych użytkownika  
const { data: updatedUser } = await supabase  
 .from('profiles')  
 .select('id, email, user\_type, business\_role, business\_id')  
 .eq('id', user\_id)  
 .single();  
  
return res.status(201).json({  
 ok: true,  
 message: "Firma została pomyślnie zarejestrowana i konto aktywowane",  
 data: {  
 business: newBusiness,  
 user: updatedUser,  
 activation\_info: {  
 account\_upgraded: true,  
 previous\_type: 'customer',  
 new\_type: 'business\_owner',  
 business\_id: newBusiness.id  
 }  
 }  
});

} catch (err) { console.error(“BUSINESS\_REGISTER error:”, err); return res.status(500).json({ ok: false, error: “BUSINESS\_REGISTER\_INTERNAL”, message: “Błąd wewnętrzny serwera”, detail: String(err?.message || err) }); } }