```
// /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.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InhkaGx6dG1qa3Rt
);

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)
});
}
}
```