[¿TIENES UN TÍTULO PARA EL DOC?]

Haz doble clic o toca dos veces esto para editar

// Estructura completa del repo WebSite100

/* --- .env.example --- */

OWNER_EMAIL=antonioyorey@gmail.com

PAYPAL_CLIENT_ID=tu_paypal_client_id

PAYPAL_SECRET=tu_paypal_secret

PAYPAL_MODE=sandbox

PAYPAL_API=https://api-m.sandbox.paypal.com

DEFAULT_CREDITS_ON_REGISTER=100

BOT_CHECK_INTERVAL_MIN=360

DATABASE_URL=postgresql://user:password@localhost:5432/dbname

/* --- migrations/003_credits_promos.sql --- */

-- Agregar columna credits y subscription_id a users

ALTER TABLE IF EXISTS users

```
  ADD COLUMN IF NOT EXISTS
credits integer DEFAULT 0,
  ADD COLUMN IF NOT EXISTS
subscription_id text;


-- Crear tabla promos
CREATE TABLE IF NOT EXISTS
promos (
  id serial PRIMARY KEY,
  code text UNIQUE NOT NULL,
  discount_percent integer
DEFAULT 0,
  fixed_credits integer DEFAULT 0,
  uses_limit integer DEFAULT
NULL,
```

```
  uses_count integer DEFAULT 0,
  expires_at timestamptz DEFAULT NULL,
  active boolean DEFAULT true,
  created_at timestamptz DEFAULT now()
);

-- Inicializar promos precargadas
INSERT INTO promos (code, fixed_credits, discount_percent, uses_limit, expires_at)
VALUES
  ('FREE100', 100, 0, NULL, now() + interval '365 days'),
```

```
  ('TONY50', 0, 50, NULL, now() + interval '365 days'),
  ('BLACK25', 0, 25, NULL, now() + interval '365 days')
ON CONFLICT (code) DO NOTHING;

/* --- server/routes/auth.js --- */
const express = require('express');
const router = express.Router();
const { pool } = require('../db');
const bcrypt = require('bcrypt');
```

```
const DEFAULT_CREDITS =
parseInt(process.env.DEFAULT_CR
EDITS_ON_REGISTER || '100');


router.post('/register', async (req,
res) => {
  const { email, password, name } =
req.body;
  const ph = await
bcrypt.hash(password, 10);
  try {
    const r = await pool.query(
      'INSERT INTO users
(email,password_hash,name,role,c
```

```
redits) VALUES ($1,$2,$3,$4,$5)
RETURNING id, email',
      [email, ph, name, 'advertiser',
DEFAULT_CREDITS]
    );
    const userId = r.rows[0].id;
    res.json({ ok:true, userId });
  } catch(e){
    console.error(e);
    res.status(500).json({
error:'register failed' });
  }
});
```

```
module.exports = router;

/* --- server/routes/promos.js --- */
const express = require('express');
const router = express.Router();
const { pool } = require('../db');

router.post('/apply', async (req, res) => {
  const { code, userEmail } = req.body;
  if(!code) return res.status(400).json({ error:'no code' });
```

```
  try {
    const r = await
pool.query('SELECT * FROM
promos WHERE code=$1 AND
active=true', [code]);
    if(!r.rowCount) return res.json({
valid:false, error:'not found' });
    const promo = r.rows[0];
    if (promo.expires_at && new
Date(promo.expires_at) < new
Date()) return res.json({
valid:false, error:'expired' });
    if (promo.uses_limit &&
promo.uses_count >=
promo.uses_limit) return res.json({
```

```
valid:false, error:'uses exceeded'
});
    return res.json({ valid: true,
fixed_credits: promo.fixed_credits,
discount_percent:
promo.discount_percent });
  } catch(e){
    console.error(e);
    res.status(500).json({ error:
'server error' });
  }
});

router.post('/create', async (req,
res) => {
```

```
  const { code, fixed_credits=0,
discount_percent=0,
expires_at=null, uses_limit=null } =
req.body;
  try {
    const q = await
pool.query('INSERT INTO promos
(code,fixed_credits,discount_perc
ent,expires_at,uses_limit) VALUES
($1,$2,$3,$4,$5) RETURNING *',
[code, fixed_credits,
discount_percent, expires_at,
uses_limit]);
    res.json({ ok:true, promo:
q.rows[0] });
  } catch(e){
```

```
    console.error(e);
    res.status(500).json({ error:'create failed' });
  }
});

module.exports = router;

/* --- server/routes/payments.js --- */
const express = require('express');
const router = express.Router();
const { pool } = require('../db');
```

```
router.post('/webhook', async (req,res)=>{
  const event = req.body;
  console.log('paypal webhook', event.event_type);
  try {
    if(event.event_type === "BILLING.SUBSCRIPTION.CREATED" || event.event_type === "BILLING.SUBSCRIPTION.ACTIVATED"){
      const subscriber = event.resource?.subscriber;
      const email = subscriber?.email_address ||
```

```
(event.resource?.custom_id) ||
null;
    const subscriptionId =
event.resource?.id || null;
    if(email){
      const creditsToAdd = 400;
      await pool.query('UPDATE
users SET credits = credits + $1,
subscription_id = $2 WHERE email
= $3', [creditsToAdd,
subscriptionId, email]);
      console.log(`Acreditados
${creditsToAdd} créditos a
${email}`);
    }
```

```
    }

    if(event.event_type ===
"CHECKOUT.ORDER.APPROVED" ||
event.event_type ===
"PAYMENT.CAPTURE.COMPLETED
"){

        const custom =
event.resource?.custom_id;

        const payerEmail =
event.resource?.payer?.email_add
ress || custom;

        const amount =
parseFloat((event.resource?.purc
hase_units?.[0]?.amount?.value) ||
0);
```

```
    const creditsMap = { '5':100, '15':400, '25':800, '50':2000 };
    const credits = creditsMap[amount] || Math.floor(amount * 100);
    if(payerEmail){
      await pool.query('UPDATE users SET credits = credits + $1 WHERE email = $2', [credits, payerEmail]);
      console.log(`Pago confirmado: ${payerEmail} +${credits} créditos`);
    }
  }
```

```
    if(event.event_type === "BILLING.SUBSCRIPTION.CANCELLED" || event.event_type === "BILLING.SUBSCRIPTION.SUSPENDED"){
        const email = event.resource?.subscriber?.email_address;
        if(email){
          await pool.query('UPDATE users SET subscription_id = NULL WHERE email = $1', [email]);
          await pool.query('UPDATE users SET credits = GREATEST(credits - 0, 0) WHERE email = $1', [email]);
```

```
      console.log(`Subscription canceled for ${email}`);
    }
  }
 } catch(e){ console.error('webhook handler error', e); }
  res.status(200).send('ok');
});

module.exports = router;

/* --- server/bot.js --- */
require('dotenv').config();
```

```
const { Pool } = require('pg');
const fetch = require('node-fetch');
const pool = new Pool({ connectionString: process.env.DATABASE_URL });

const OWNER_EMAIL = process.env.OWNER_EMAIL || 'antonioyorey@gmail.com';
const PAYPAL_API = process.env.PAYPAL_API || 'https://api-m.sandbox.paypal.com';
```

```
const CLIENT = process.env.PAYPAL_CLIENT_ID;
const SECRET = process.env.PAYPAL_SECRET;


async function getPayPalToken(){
  const resp = await fetch(`${PAYPAL_API}/v1/oauth2/token`, {
    method: 'POST',
    headers: { Authorization: 'Basic ' + Buffer.from(CLIENT + ':' + SECRET).toString('base64'), 'Content-Type': 'application/x-www-form-urlencoded' },
```

```
    body:
'grant_type=client_credentials'
  });
  const data = await resp.json();
  return data.access_token;
}

async function
checkSubscriptionsAndPayments(
){
  console.log('Bot: checking
subscriptions/payments...');
  try {
    const token = await
getPayPalToken();
```

```
    const res = await
pool.query('SELECT id, email,
subscription_id FROM users
WHERE subscription_id IS NOT
NULL');
    for(const u of res.rows){
      if(!u.subscription_id) continue;
      try {
        const r = await
fetch(`${PAYPAL_API}/v1/billing/s
ubscriptions/${u.subscription_id}`
, { headers: { Authorization:
`Bearer ${token}`, 'Content-Type':
'application/json' }});
        const data = await r.json();
```

```
        if(data.status !== 'ACTIVE'){
          console.log(`Subscription not active for ${u.email}: ${data.status}`);
          await pool.query('UPDATE users SET subscription_id = NULL WHERE id=$1', [u.id]);
        }
      } catch(e){ console.error('check sub err', e); }
    }
  } catch(e){ console.error('bot main err', e); }
}
```

```
async function mainLoop(){
  console.log('Bot started', new Date().toISOString());
  await checkSubscriptionsAndPayments();
}

mainLoop();
setInterval(mainLoop, (parseInt(process.env.BOT_CHECK_INTERVAL_MIN || '360') ) * 60 * 1000);

/* --- server/server.js --- */
```

```
const express = require('express');

const app = express();

const authRoutes =
require('./routes/auth');

const promos =
require('./routes/promos');

const payments =
require('./routes/payments');


app.use(express.json());

app.use('/api/auth', authRoutes);

app.use('/api/promos', promos);

app.use('/api/payments',
payments);
```

```
app.listen(3000, () =>
console.log('Server running on
port 3000'));


/* ---
frontend/components/BuyCredits
.jsx --- */
"use client";
import { useState } from 'react';


export default function
BuyCredits({ plans }) {
  const [promo, setPromo] =
useState('');
```

```
  async function applyPromo(){
    const r = await
fetch('/api/promos/apply', {
method:'POST', body:
JSON.stringify({ code: promo }),
headers:{'Content-
Type':'application/json'}});
    const j = await r.json();
    return j;
  }

  async function
createOrder(amount, credits){
```

```
    const r = await
fetch('/api/payments/create-
order', { method:'POST', headers:
{'Content-
Type':'application/json'}, body:
JSON.stringify({ amount }) });
    const j = await r.json();
    return j;
  }

  return (
    <div>
      {plans.map(p=>(
        <div key={p.amount}>
```

```
        <div>{p.credits} créditos — ${p.amount} USD</div>
        <button onClick={()=> createOrder(p.amount, p.credits)}>Comprar</button>
      </div>
    ))}
    <div>
      <input value={promo} onChange={e=>setPromo(e.target.value)} placeholder="Código promoción"/>
      <button onClick={applyPromo}>Aplicar</button>
```

```
      </div>
    </div>
  );
}
```

Ahora puedes copiar cada archivo y estructura en tu máquina y crear el ZIP

.