Documentación del Entorno de Desarrollo API

Sistema completo configurado para desarrollo de APIs dinámicas con optimización de imágenes

Información del Servidor

Servidor VPS: 5.78.106.16

Dominio: dev.api.clicinmobiliaria.com

Sistema: Ubuntu 24.04.2 LTS

SSL: Certificado Let's Encrypt válido hasta 2025-12-24

Arquitectura del Sistema

Nginx (Reverse Proxy)

- **Puerto:** 443 (SSL) y 80 (redirección a HTTPS)
- **Configuración:** `/etc/nginx/sites-available/dev-dashboard`
- **Función:** Proxy reverso que redirige tráfico a servicios internos

Servicios Internos

1. **Dev API Server** - Puerto 8080
2. **Code Server (IDE)** - Puerto 8443
3. **File Browser** - Puerto 8081

Configuración Inicial

1. Instalación de Code Server

```bash
curl -fsSL https://code-server.dev/install.sh | sh
mkdir -p ~/.config/code-server
cat > ~/.config/code-server/config.yaml << EOF
bind-addr: 127.0.0.1:8443
auth: none
cert: false
EOF
```

2. Configuración SSL con Let's Encrypt

```bash
```

```
sudo apt update
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d dev.api.clicinmobiliaria.com
```

Resultado:

- Certificado: `/etc/letsencrypt/live/dev.api.clicinmobiliaria.com/fullchain.pem`
- Clave privada: `/etc/letsencrypt/live/dev.api.clicinmobiliaria.com/privkey.pem`
- Renovación automática configurada

3. Configuración de Nginx

Archivo: `/etc/nginx/sites-available/dev-dashboard`

```nginx
```

```
server {
    if ($host = dev.api.clicinmobiliaria.com) {
        return 301 https://$host$request_uri;
    }
    listen 80;
    server_name dev.api.clicinmobiliaria.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name dev.api.clicinmobiliaria.com;
    ssl_certificate /etc/letsencrypt/live/dev.api.clicinmobiliaria.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/dev.api.clicinmobiliaria.com/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;

    # Dashboard principal (Dev API Server)
    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Code-Server (IDE Web)
    location /ide/ {
        proxy_pass http://127.0.0.1:8443/;
        proxy_set_header Host $host;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection upgrade;
        proxy_set_header Accept-Encoding gzip;
    }

    # File Manager
    location /files/ {
        proxy_pass http://127.0.0.1:8081/;
        proxy_set_header Host $host;
    }
}
```

4. Estructura del Proyecto Dev API

Directorio: `/root/dev-api/`

```
/root/dev-api/
├── server.js             # Servidor principal Node.js
├── package.json          # Dependencias del proyecto
├── functions/            # Directorio de funciones API
│   ├── reduce-image.js   # Función de ejemplo para reducir imágenes
│   ├── optimize-image.js  # Optimizador individual de imágenes
│   └── bulk-optimize-properties.js  # Optimizador masivo
└── node_modules/         # Dependencias instaladas
```

Dependencias instaladas:

json

```
{
  "express": "^4.x",
  "cors": "^2.x",
  "helmet": "^7.x",
  "morgan": "^1.x",
  "multer": "^1.x",
  "sharp": "^0.33.x",
  "nodemailer": "^6.x",
  "@supabase/supabase-js": "^2.x"
}
```

Servidor Principal (server.js)

Funcionamiento del Sistema de APIs Dinámicas

El servidor principal carga automáticamente cualquier archivo `.js` en `/functions/` como un endpoint:

javascript

```
// Endpoint dinámico que carga funciones al vuelo
app.all('/:functionName', async (req, res) => {
    const { functionName } = req.params;
    const functionPath = path.join(FUNCTIONS_DIR, `${functionName}.js`);

    // Verificar si existe la función
    if (!fs.existsSync(functionPath)) {
        return res.status(404).json({
            error: `Function '${functionName}' not found`
        });
    }

    // Limpiar cache para recargar cambios
    delete require.cache[require.resolve(functionPath)];

    // Cargar y ejecutar la función
    const functionModule = require(functionPath);
    await functionModule(req, res);
});
```

Características Clave:

- **Hot Reload:** Los cambios en archivos se reflejan inmediatamente
- **Cache Clearing:** Elimina cache para recargar código modificado
- **Error Handling:** Manejo de errores robusto
- **CORS Habilitado:** Permite requests desde cualquier origen
- **Soporte Completo HTTP:** GET, POST, PUT, DELETE, etc.

Servicios Systemd (Permanentes)

1. Code Server Service

Archivo: `/etc/systemd/system/code-server.service`

```ini
```

```
[Unit]
Description=Code Server
After=network.target

[Service]
Type=simple
User=root
WorkingDirectory=/root
ExecStart=/usr/bin/code-server --config /root/.config/code-server/config.yaml
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

2. Dev API Service

Archivo: `/etc/systemd/system/dev-api.service`

```ini
[Unit]
Description=Dev API Server
After=network.target

[Service]
Type=simple
User=root
WorkingDirectory=/root/dev-api
ExecStart=/usr/bin/node server.js
Restart=always
RestartSec=5
Environment=NODE_ENV=production

[Install]
WantedBy=multi-user.target
```

3. File Browser Service

Archivo: `/etc/systemd/system/filebrowser.service`

```ini
```

```
[Unit]
Description=File Browser
After=network.target

[Service]
Type=simple
User=root
ExecStart=/usr/local/bin/filebrowser -a 127.0.0.1 -p 8081 -r /root
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Comandos de Gestión:

```bash
# Habilitar servicios para inicio automático
sudo systemctl enable code-server dev-api filebrowser

# Iniciar servicios
sudo systemctl start code-server dev-api filebrowser

# Ver estado
sudo systemctl status code-server dev-api filebrowser

# Reiniciar un servicio
sudo systemctl restart dev-api

# Ver logs
sudo journalctl -u dev-api -f
```

Configuración de Supabase

Credenciales

- **URL:** https://pacewqgypevfgjmdsorz.supabase.co
- **Anon Key:** eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

Storage - Bucket "images"

Políticas configuradas:

```sql
-- Permitir lectura pública
CREATE POLICY "Allow public reads" ON storage.objects
FOR SELECT USING (bucket_id = 'images');

-- Permitir subida pública
CREATE POLICY "Allow public uploads" ON storage.objects
FOR INSERT WITH CHECK (bucket_id = 'images');

-- Permitir actualizaciones públicas
CREATE POLICY "Allow public updates" ON storage.objects
FOR UPDATE USING (bucket_id = 'images');
```

Sistema de Optimización de Imágenes

Configuración de Tamaños

```javascript
const SIZES = {
    thumbnail: { width: 300, height: 200, quality: 85 },    // Previews/cards
    medium: { width: 800, height: 600, quality: 90 },       // Modales/detalles
    large: { width: 1200, height: 900, quality: 95 }        // Zoom/alta calidad
};

// Para bulk optimization
const SIZES = {
    small: { width: 400, height: 300, quality: 85 },       // BD (main_image_url)
    large: { width: 1200, height: 900, quality: 95 }       // Zoom/detalles
};
```

Formatos Generados

- **WebP:** Prioridad principal (mejor compresión)
- **JPG:** Respaldo para compatibilidad

Procesamiento de Base de Datos

Tabla: `properties`

- **Campo:** `main_image_url` - Se actualiza con versión small WebP
- **Campo:** `gallery_images_url` - Array JSON actualizado con versiones small WebP

URLs de Acceso

Servicios Web

- **Dashboard Principal:** https://dev.api.clicinmobiliaria.com/
- **IDE (Code Server):** https://dev.api.clicinmobiliaria.com/ide/
- **File Manager:** https://dev.api.clicinmobiliaria.com/files/
  - Usuario: admin
  - Contraseña: Y9GvaWpWjtQisn1-

APIs Disponibles

- **Optimizador Individual:** https://dev.api.clicinmobiliaria.com/optimize-image
- **Optimizador Masivo:** https://dev.api.clicinmobiliaria.com/bulk-optimize-properties
- **Reductor Simple:** https://dev.api.clicinmobiliaria.com/reduce-image

Flujo de Desarrollo

Crear Nueva Función API

1. Acceder al IDE: https://dev.api.clicinmobiliaria.com/ide/
2. Navegar a `/root/dev-api/functions/`
3. Crear archivo `mi-nueva-funcion.js`:

```javascript
module.exports = async (req, res) => {
    res.json({
        message: "Nueva función funcionando",
        method: req.method,
        timestamp: new Date().toISOString()
    });
};
```

4. Guardar (Ctrl+S)
5. Acceder inmediatamente: https://dev.api.clicinmobiliaria.com/mi-nueva-funcion

Estructura de Función

```javascript
```

```
module.exports = async (req, res) => {
    try {
        // Manejar diferentes métodos HTTP
        if (req.method === 'GET') {
            // Lógica para GET
            return res.json({ data: "GET response" });
        }

        if (req.method === 'POST') {
            const { param1, param2 } = req.body;
            // Lógica para POST
            return res.json({ success: true, data: result });
        }

        res.status(405).json({ error: 'Método no permitido' });

    } catch (error) {
        res.status(500).json({
            error: error.message,
            timestamp: new Date().toISOString()
        });
    }
};
```

Mantenimiento y Monitoreo

Logs del Sistema

bash

```
# Logs de nginx
sudo tail -f /var/log/nginx/error.log
sudo tail -f /var/log/nginx/access.log

# Logs de servicios systemd
sudo journalctl -u dev-api -f
sudo journalctl -u code-server -f
sudo journalctl -u filebrowser -f
```

Comandos Útiles

bash

```
# Ver puertos en uso
sudo netstat -tlnp | grep -E "8080|8443|8081"

# Verificar estado SSL
sudo certbot certificates

# Renovar SSL manualmente
sudo certbot renew

# Reiniciar nginx
sudo systemctl reload nginx

# Ver procesos Node.js
ps aux | grep node
```

Respaldos Recomendados

- `/root/dev-api/` - Todo el proyecto
- `/etc/nginx/sites-available/` - Configuraciones nginx
- `/etc/systemd/system/` - Servicios systemd
- Base de datos Supabase (automático en su plataforma)

Escalabilidad y Mejoras Futuras

Automatización

- **Cron Jobs:** Para ejecutar optimización masiva programada
- **Webhooks:** Para procesar imágenes cuando se agregan nuevas propiedades
- **Queue System:** Para procesar lotes grandes de imágenes

Seguridad

- **API Keys:** Implementar autenticación para funciones sensibles
- **Rate Limiting:** Limitar requests por IP
- **Input Validation:** Validar datos de entrada más estrictamente

Performance

- **Caching:** Implementar Redis para cache
- **CDN:** Usar Cloudflare para servir imágenes optimizadas
- **Load Balancer:** Para múltiples instancias del servidor

Solución de Problemas Comunes

502 Bad Gateway

```bash
# Verificar servicios
sudo systemctl status code-server dev-api filebrowser

# Reiniciar servicios si están caídos
sudo systemctl restart dev-api
```

SSL "No seguro"

- Limpiar cache del navegador
- Usar ventana incógnita
- Verificar certificado: `sudo certbot certificates`

Función no encontrada

- Verificar que el archivo existe en `/root/dev-api/functions/`
- Nombre del archivo debe coincidir exactamente con la URL
- Reiniciar dev-api si es necesario: `sudo systemctl restart dev-api`

Error de permisos Supabase

- Verificar políticas RLS en Supabase Dashboard
- Confirmar que bucket 'images' existe
- Verificar credenciales en el código

Última actualización: 25 de Septiembre, 2025
Versión del sistema: 1.0
Mantenido por: Entorno de desarrollo API dinámico