

LOXYS Solutions Informatiques

SOLUTIONS INFORMATIQUES

DOCUMENTATION TECHNIQUE COMPLÈTE

Systeme de Monitoring IA & Portail Captif WiFi

Fourier Tiago — Stagiaire BTS SIO SISR

LOXYS Solutions Informatiques — Saint-Malo-du-Bois (Vendée)

Stage : 26 mai – 3 juillet 2026

1. Présentation de l'entreprise

LOXYS Solutions Informatiques est une SASU créée en 2018, dirigée par Thomas Hulin, spécialisée dans les services informatiques aux professionnels en Vendée.

Raison sociale	LOXYS Solutions Informatiques (SASU)
Dirigeant	Thomas Hulin (Président)
SIRET	835 073 750 00027
Adresse	1 Rue des Vendéens, 85590 Saint-Malo-du-Bois
Téléphone	02 22 44 44 34
Email	contact@loxys.fr
Création	1er février 2018 (8 ans d'activité)
Collaborateurs	5 collaborateurs
Clients	130 clients professionnels
Zone	Rayon de 30 km autour de Saint-Malo-du-Bois
Services	Infogérance, réseau WiFi, sécurité, téléphonie, sites web, matériel

2. Architecture réseau du projet

2.1 Équipements et adressage IP

Serveur Flowise	Ubuntu Server 26.04 LTS — 192.168.1.11
Serveur Database	Ubuntu Server 26.04 LTS — 192.168.1.111
Pare-feu / Routeur	OPNsense 26.1.10 — 192.168.1.254
Borne WiFi	UniFi UAP-AC-Mesh — 192.168.1.21
Switch LAN	Netgear (non manageable)
Zabbix Server	Docker sur 192.168.1.11 — port 8080
Flowise	Docker sur 192.168.1.11 — port 3000
API monitoring Flowise	Port 5000 (datacenter_monitor.py)
API monitoring Database	Port 5001 (server_monitor.py)
Relais mail Flask	Port 5002 (mail_relay.py)

2.2 Services systemd actifs

- datacenter_monitor.service — script Python port 5000 sur Flowise
- server_monitor.service — script Python port 5001 sur Database
- mail-relay.service — Flask SMTP Gmail port 5002 sur Flowise
- zabbix-agent2.service — agent Zabbix sur Flowise, Database et OPNsense

3. Projet Portail Captif WiFi

3.1 Objectif

Mise en place d'un portail captif WiFi branded aux couleurs LOXYS, permettant un accès Internet gratuit aux visiteurs avec collecte d'emails conforme au RGPD.

3.2 Stack technique

Pare-feu / portail	OPNsense 26.1.10 — portail captif intégré
Serveur web	Apache2 sur Ubuntu 26.04 — 192.168.1.111
Langage backend	PHP + HTML/CSS/JavaScript
Base de données	PostgreSQL 18 (emails + tokens)
Conformité	RGPD — consentement, token de suppression, nettoyage auto

3.3 Fichiers PHP créés

- save_email.php — enregistre l'email avec horodatage et token unique
- delete_email.php — supprime via token (droit à l'oubli RGPD)
- admin.php — interface d'administration protégée par mot de passe
- Cron job quotidien — nettoyage automatique des données > 1 an

3.4 Configuration OPNsense

- Portail captif configuré pour rediriger vers <http://192.168.1.111>
- Règle firewall : accès libre au serveur portail sur port 80
- DHCP distribue 192.168.1.0/24 aux clients WiFi

4. Scripts Python de monitoring

4.1 datacenter_monitor.py (port 5000 — Flowise)

Agrège les métriques des deux serveurs et les expose via une API REST Flask.

Endpoint principal :

```
curl http://192.168.1.11:5000/status
```

Réponse JSON :

```
{
  "serveur_flowise": {
    "cpu_percent": 4.0, "ram_percent": 16.2,
    "disque_percent": 17.1, "cpu_temperature": 37.0,
    "ram_disponible_go": 5.93, "disque_libre_go": 77.01
  },
  "serveur_database": {
    "cpu_percent": 0.3, "ram_percent": 9.1,
    "disque_percent": 8.9, "cpu_temperature": 46.0,
    "ip": "192.168.1.111"
  },
  "reseau": {
    "flowise": "UP", "database": "UP",
    "borne_wifi": "DOWN", "opnsense": "UP"
  }
}
```

4.2 Commandes systemd

```
# Démarrer et activer les services
sudo systemctl enable datacenter_monitor --now
sudo systemctl enable server_monitor --now

# Vérifier le statut
sudo systemctl status datacenter_monitor
sudo systemctl status server_monitor

# Redémarrer après modification
sudo systemctl restart datacenter_monitor
```

5. Zabbix 7.0 — Supervision réseau

5.1 Installation via Docker (192.168.1.11)

```
# Créer le dossier et le compose
mkdir -p /opt/zabbix && cd /opt/zabbix
nano docker-compose.yml

# Lancer les conteneurs
docker compose up -d

# Vérifier
docker ps
```

Contenu du docker-compose.yml :

```
services:
  zabbix-db:
    image: postgres:16
    environment:
      POSTGRES_DB: zabbix
      POSTGRES_USER: zabbix
      POSTGRES_PASSWORD: .....

  zabbix-server:
    image: zabbix/zabbix-server-pgsql:ubuntu-7.0-latest
    ports:
      - "10051:10051"
    environment:
      DB_SERVER_HOST: zabbix-db
      POSTGRES_DB: zabbix
      POSTGRES_USER: zabbix
      POSTGRES_PASSWORD: .....

  zabbix-web:
    image: zabbix/zabbix-web-nginx-pgsql:ubuntu-7.0-latest
    ports:
      - "8080:8080"
    environment:
      ZBX_SERVER_HOST: zabbix-server
      PHP_TZ: Europe/Paris
```

5.2 Installation Zabbix Agent2 — Flowise et Database

```
# Ajouter le dépôt Zabbix 8.0 (Ubuntu 26.04)
wget https://repo.zabbix.com/zabbix/8.0/release/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest_8.0+ubuntu26.04_all.deb
sudo dpkg -i zabbix-release_latest_8.0+ubuntu26.04_all.deb
sudo apt update

# Installer l'agent
sudo apt install zabbix-agent2 -y
```

```
# Configurer /etc/zabbix/zabbix_agent2.conf
Server=192.168.1.11,172.19.0.0/16
ServerActive=192.168.1.11
Hostname=Flowise # ou Database selon le serveur

# Démarrer
sudo systemctl enable zabbix-agent2 --now
```

5.3 Installation plugin Zabbix sur OPNsense

```
# Via l'interface OPNsense :
# System → Firmware → Plugins → Cocher 'Show community plugins'
# Installer : os-zabbix7-agent

# Configuration : Services → Zabbix Agent → Settings
# Server: 192.168.1.11
# Hostname: OPNsense
# Active Check Servers: 192.168.1.11
```

5.4 Test de connexion depuis le conteneur Zabbix

```
# Tester la connexion vers Flowise
docker exec zabbix-server zabbix_get -s 192.168.1.11 -p 10050 -k system.uptime

# Tester la connexion vers Database
docker exec zabbix-server zabbix_get -s 192.168.1.111 -p 10050 -k system.uptime

# Tester la connexion vers OPNsense
docker exec zabbix-server zabbix_get -s 192.168.1.254 -p 10050 -k system.uptime
```

5.5 Hôtes configurés dans Zabbix

Flowise	IP 192.168.1.11:10050 — Template: Linux by Zabbix agent
Database	IP 192.168.1.111:10050 — Template: Linux by Zabbix agent
OPNsense	IP 192.168.1.254:10050 — Template: FreeBSD by Zabbix agent
Zabbix server	127.0.0.1:10050 — Templates: Linux + Zabbix server health

6. Accès SSH sécurisé — compte flowise-diag

6.1 Création du compte dédié (sur Database)

```
# Créer l'utilisateur sans droits sudo
sudo useradd -m -s /bin/bash flowise-diag
sudo usermod -aG adm flowise-diag # accès aux logs système

# Créer le dossier SSH
sudo mkdir -p /home/flowise-diag/.ssh
sudo chmod 700 /home/flowise-diag/.ssh
```

6.2 Génération de la clé Ed25519 (sur Flowise)

```
# Générer la clé SSH dédiée
ssh-keygen -t ed25519 -f ~/.ssh/flowise_diag_key -C 'flowise-diag'

# Copier la clé publique vers Database
ssh-copy-id -i ~/.ssh/flowise_diag_key.pub flowise-diag@192.168.1.111
```

6.3 Script de diagnostic (sur Database)

```
# Créer le script restreint
sudo nano /usr/local/bin/flowise-diag.sh

#!/bin/bash
echo '--- CPU ---' && top -bn1 | head -20
echo '--- MEM ---' && free -h
echo '--- DISK ---' && df -h
echo '--- TOP PROCESSES (CPU) ---' && ps aux --sort=-%cpu | head -10
echo '--- TOP PROCESSES (MEM) ---' && ps aux --sort=-%mem | head -10
echo '--- CONNEXIONS RESEAU ---' && ss -tnp | head -20
echo '--- LOGS ERREURS ---' && journalctl -p err -n 30 --no-pager

sudo chmod +x /usr/local/bin/flowise-diag.sh
```

6.4 Forcer la commande via authorized_keys

```
# Éditer authorized_keys sur Database
sudo nano /home/flowise-diag/.ssh/authorized_keys

# Ajouter la commande forcée avant la clé :
command="/usr/local/bin/flowise-diag.sh",no-pty,no-port-forwarding ssh-ed25519
AAAA...
```

6.5 Test de la connexion SSH

```
# Test depuis Flowise
```

```
ssh flowise-diag@192.168.1.111 -i ~/.ssh/flowise_diag_key
```

```
# Copier la clé dans le conteneur Flowise
```

```
docker cp ~/.ssh/flowise_diag_key docker-flowise-1:/root/flowise_diag_key
```

7. Agent IA Flowise

7.1 Chatflow configuré

Chatflow ID	[identifiant masqué]
LLM	Groq — llama-3.3-70b-versatile
Temperature	0.9
Mémoire	Buffer Memory
Max Iterations	5
Type	Tool Agent

7.2 Outil 1 — datacenter_monitor

Appelle l'API Python pour obtenir les métriques live de tous les serveurs.

Tool Name : datacenter_monitor

Tool Description : Get the current status of all datacenter servers and network.

Input Schema : []

Code JavaScript :

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

try {
  const response = await axios.get(
    'http://192.168.1.11:5000/status',
    { timeout: 5000 }
  );
  const d = response.data;
  const f = d.serveur_flowise;
  const db = d.serveur_database;
  const r = d.reseau;

  return `Statut datacenter:
Serveur Flowise: CPU ${f.cpu_percent}%, RAM ${f.ram_percent}%
Serveur Database: CPU ${db.cpu_percent}%, RAM ${db.ram_percent}%
Réseau: Flowise=${r.flowise}, Database=${r.database}`;
} catch (error) {
  return 'Erreur: ' + error.message;
}
```

7.3 Outil 2 — diagnostic_ssh_database

Se connecte en SSH à Database et retourne un diagnostic complet en temps réel.

Code JavaScript (clé SSH embarquée dans le code) :

```

const { Client } = require('ssh2');

return new Promise((resolve, reject) => {
  const conn = new Client();
  let output = '';

  const privateKey = `-----BEGIN OPENSSH PRIVATE KEY-----
[clé privée masquée]
-----END OPENSSH PRIVATE KEY-----`;

  const commands = [
    'echo "--- CPU ---" && top -bn1 | head -20',
    'echo "--- MEM ---" && free -h',
    'echo "--- DISK ---" && df -h',
    'echo "--- TOP PROCESSES (CPU) ---" && ps aux --sort=-%cpu | head -10',
    'echo "--- LOGS ERREURS ---" && journalctl -p err -n 30 --no-pager'
  ].join(' && ');

  conn.on('ready', () => {
    conn.exec(commands, (err, stream) => {
      stream.on('data', d => output += d.toString());
      stream.on('close', () => { conn.end(); resolve(output); });
    });
  });
  conn.connect({ host: '192.168.1.111', port: 22,
    username: 'flowise-diag', privateKey });
});

```

7.4 Correction du schéma JSON — problème connu

Le schéma des Custom Tools doit être [] (tableau vide) et non un objet JSON Schema.

Si les outils retournent toolInput: null, corriger directement en base SQLite :

```

# Sur le serveur Flowise
sudo sqlite3 /home/tiago/flowise-data/database.sqlite \
  "UPDATE tool SET schema='[]' WHERE name='datacenter_monitor';"

sudo sqlite3 /home/tiago/flowise-data/database.sqlite \
  "UPDATE tool SET schema='[]' WHERE name='diagnostic_ssh_database';"

# Vérifier
sudo sqlite3 /home/tiago/flowise-data/database.sqlite \
  "SELECT name, schema FROM tool;"

# Redémarrer Flowise
cd /home/tiago/Flowise/docker && docker compose restart

```

7.5 System Prompt du Tool Agent

Tu es un expert en administration système pour LOXYS Solutions Informatiques.

Quand tu reçois une alerte :

1. Appelle datacenter_monitor pour voir les métriques globales
2. Appelle diagnostic_ssh_database pour obtenir les données réelles du serveur
3. Analyse les données RÉELLES obtenues
4. Structure ta réponse :

DIAGNOSTIC PRINCIPAL (basé sur les données réelles)
Cause probable avec preuves (PID, processus, valeurs réelles)

CAUSES ALTERNATIVES MOINS PROBABLES

MARCHE À SUIVRE CONCRÈTE
Commandes shell précises basées sur les vrais processus trouvés

Réponds toujours en français.
Base ton analyse sur les données réelles des outils uniquement.

7.6 Test de l'API Flowise

```
# Test direct
curl -X POST http://192.168.1.11:3000/api/v1/prediction/[identifiant masqué] \
  -H 'Content-Type: application/json' \
  -d '{"question": "Appelle datacenter_monitor et dis-moi le résultat."}'

# Réponse attendue : toolInput doit être {} (pas null)
# usedTools[0].toolInput: {} → outil appelé correctement
# usedTools[0].toolInput: null → schéma mal configuré
```

8. Webhook Zabbix → Flowise → Email

8.1 Configuration dans Zabbix

Alerts → Media types → Create media type :

Name	Flowise AI Alert
Type	Webhook
Script timeout	60s (Administration → General → Timeouts)

Paramètres du webhook :

event_name	{EVENT.NAME}
host_name	{HOST.NAME}
severity	{EVENT.SEVERITY}
item_value	{ITEM.VALUE1}

8.2 Script JavaScript du webhook

```
try {
  var params = JSON.parse(value);
  var req = new HttpRequest();
  req.addHeader('Content-Type: application/json');

  var question = '[ALERTE ZABBIX] Hôte: ' + params.host_name +
    '. Problème: ' + params.event_name +
    '. Sévérité: ' + params.severity +
    '. Valeur: ' + params.item_value +
    '. Appelle datacenter_monitor PUIS diagnostic_ssh_database.',
    ' Base ton analyse UNIQUEMENT sur les données réelles.';

  var payload = JSON.stringify({ question: question });
  var resp = req.post(
    'http://192.168.1.11:3000/api/v1/prediction/[identifiant masqué]',
    payload
  );

  var flowiseResponse = JSON.parse(resp);
  var aiAnalysis = flowiseResponse.text;

  var emailReq = new HttpRequest();
  emailReq.addHeader('Content-Type: application/json');
  var emailPayload = JSON.stringify({
    subject: 'Alerte: ' + params.event_name + ' (' + params.host_name + ')',
    host: params.host_name,
    problem: params.event_name,
    severity: params.severity,
    value: params.item_value,
    ai_analysis: aiAnalysis
  });
  emailReq.post('http://192.168.1.11:5002/send-alert', emailPayload);
}
```

```
        return 'OK: Analyse envoyée par email';
    } catch (error) {
        throw 'Échec: ' + error;
    }
}
```

8.3 Utilisateur Zabbix dédié

Username	flowise-bot
Groupe	Zabbix administrators
Rôle	User role
Media	Flowise AI Alert — Send to: flowise

8.4 Action Zabbix

Nom	Envoyer alertes à Flowise AI
Opération	Send message to users: flowise-bot via Flowise AI Alert
Déclenchement	Immediately
Conditions	Aucune (toutes les alertes)

9. Service relais mail (mail_relay.py)

9.1 Installation

```
# Sur le serveur Flowise
mkdir -p ~/mail-relay && cd ~/mail-relay
pip install flask --break-system-packages
nano mail_relay.py
```

9.2 Code Python (extrait)

```
from flask import Flask, request, jsonify
import smtplib, re
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

app = Flask(__name__)
SMTP_USER = '[adresse masquée]'
SMTP_PASSWORD = 'xxxx xxxx xxxx xxxx' # mot de passe app Gmail
DEST_EMAIL = '[adresse masquée]'

@app.route('/send-alert', methods=['POST'])
def send_alert():
    data = request.json
    # ... génération HTML email aux couleurs LOXYS ...
    # Envoi via SMTP Gmail port 587 (STARTTLS)
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(SMTP_USER, SMTP_PASSWORD)
    server.send_message(msg)
    return jsonify({'status': 'ok'}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5002)
```

9.3 Service systemd

```
sudo nano /etc/systemd/system/mail-relay.service

[Unit]
Description=Mail Relay for Zabbix Alerts
After=network.target

[Service]
Type=simple
User=tiago
WorkingDirectory=/home/tiago/mail-relay
ExecStart=/usr/bin/python3 /home/tiago/mail-relay/mail_relay.py
Restart=always

[Install]
```

```
WantedBy=multi-user.target
```

```
# Activer
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl enable mail-relay --now
```

9.4 Test du relais mail

```
curl -X POST http://localhost:5002/send-alert \  
-H 'Content-Type: application/json' \  
-d '{  
  "subject": "Test LOXYS Monitoring",  
  "host": "Database",  
  "problem": "High CPU utilization",  
  "severity": "Warning",  
  "value": "100 %",  
  "ai_analysis": "Test de fonctionnement du relais mail."  
}'
```

10. Mission Festival de Poupet

10.1 Contexte

Déploiement réseau complet pour le Festival de Poupet — événement musical en plein air en Vendée. LOXYS était responsable de toute l'infrastructure réseau.

10.2 Réalisations

- Installation physique des bornes WiFi UniFi selon plan de déploiement
- Brassage des baies informatiques (switches, patch panels, câblage RJ45)
- Création et gestion des VLANs : Public / Artistes / Régie / Stock
- Segmentation réseau pour isolation des flux et sécurité
- Vérification de la connectivité fibre optique
- Installation d'écrans de gestion pour le stock en temps réel

10.3 Adressage VLANs

VLAN Public	Accès Internet filtré pour les visiteurs
VLAN Artistes	Réseau dédié aux artistes et backstage
VLAN Régie	Réseau de pilotage scène et technique
VLAN Gestion	Réseau stocks et administration

11. Missions tickets clients

11.1 Client piraté — scareware

PC client infecté par un faux message de sécurité Windows Defender (scareware). L'arnaque affichait un faux code d'erreur et un numéro de téléphone frauduleux.

- Identification du scareware (fausse alerte Windows Defender / Centre de Sécurité)
- Diagnostic via PowerShell — liste des processus et logiciels récemment installés
- Suppression des logiciels malveillants (ConnectWise RAT / WebCompanion / PremierOpinion)
- Nettoyage des entrées de registre suspectes
- Vérification RAM (configuration single channel identifiée)
- Remise en état et conseils à l'utilisateur

11.2 Installation caméras de surveillance

- Câblage des caméras IP (PoE)
- Configuration de l'enregistreur vidéo NVR
- Paramétrage de la détection de mouvement
- Configuration de l'accès à distance (port forwarding, application mobile)
- Tests de fonctionnement et formation du client

11.3 Autres interventions

- Installation Pack Office 365 sur postes clients
- Gestion des mises à jour Windows et logiciels
- Changements de composants matériels (RAM, SSD, alimentation)
- Configuration réseau et imprimantes

12. Bilan — État des composants

Composant	Statut	Détail
Portail captif WiFi (RGPD)	✓ OK	Apache2 + PHP — 192.168.1.111:80
OPNsense 26.1.10	✓ OK	Pare-feu, DHCP, portail captif — 192.168.1.254
Zabbix Server + Web + DB	✓ OK	Docker — 192.168.1.11:8080
Zabbix Agent — Flowise	✓ OK	zabbix-agent2 systemd — 192.168.1.11:10050
Zabbix Agent — Database	✓ OK	zabbix-agent2 systemd — 192.168.1.111:10050
Zabbix Agent — OPNsense	✓ OK	Plugin os-zabbix7-agent — 192.168.1.254:10050
Script datacenter_monitor	✓ OK	Flask systemd — port 5000
Script server_monitor	✓ OK	Flask systemd — port 5001
Flowise + Groq LLM	✓ OK	Docker — port 3000
Webhook Zabbix → Flowise	✓ OK	Media type JS, timeout 60s
Service mail-relay Gmail	✓ OK	Flask systemd — port 5002
Email HTML LOXYS automatique	✓ OK	Diagnostic IA réel inclus
Borne UniFi supervisée Zabbix	✗	Contrôleur abandonné (CPU sans AVX)

Flux d'alerte complet

1. Zabbix Agent2 détecte CPU > 90% pendant 5 min
2. Action Zabbix déclenche le webhook 'Flowise AI Alert'
3. Script JS POST vers `http://192.168.1.11:3000/api/v1/prediction/{id}`
4. Agent appelle datacenter_monitor → métriques live
5. Agent appelle diagnostic_ssh_database → processus réels, logs
6. LLM Groq génère diagnostic principal + causes alternatives + marche à suivre
7. Script POST vers `http://192.168.1.11:5002/send-alert`
8. Email HTML LOXYS envoyé sur [adresse masquée] via SMTP Gmail

13. Problèmes rencontrés et solutions

toolInput: null sur les outils	Mettre schema='[]' directement via sqlite3 + restart Flowise
NodeVM: Cannot find module 'fs'	Embedder la clé SSH directement dans le code JS (pas de fs.readFileSync)
process.env non accessible	Variable d'env non transmise au sandbox — clé en dur dans le code
MongoDB sans AVX	Downgrader vers mongo:4.4 (CPU sans AVX sur Database)
Conflit port 80/8080 Apache	Séparer Zabbix Web sur 8080, portail captif sur 80
Agent Zabbix rejeté (IP Docker)	Ajouter 172.19.0.0/16 dans Server= de zabbix_agent2.conf
Webhook timeout trop court	Passer le timeout Script à 60s dans Administration → Timeouts
Ubuntu 26.04 non supporté	Utiliser le dépôt Zabbix 8.0 spécifique Ubuntu 26.04
Agent IA invente des données	Forcer [] comme schéma + system prompt strict + max iterations 5
Chatflow ID invalide (404)	Récupérer le nouvel ID via Flowise → API → CURL après recreation