📘 CHAINSHIELD.MA - CAHIER DES CHARGES TECHNIQUE

Version : 1.0
Date : Novembre 2024
Événement : Huawei Cloud ICT Competition 2025
Équipe : Kaouthar Belkebir, Reda Ouzidan, Abdellatif Tazarni, Ikram Naoui, Mouad Omlil

---

📑 TABLE DES MATIÈRES

---

1. VISION & CONTEXTE

1.1 Problématique

Les entreprises marocaines et africaines subissent une augmentation de 34% des cyberattaques ciblant leurs chaînes d'approvisionnement. 61% de ces attaques passent par des fournisseurs compromis, avec un coût moyen de 4.5M$ par incident.

Cas d'usage réel :

- **NotPetya (2017)** : Fournisseur ukrainien compromis → 10 milliards $ de dégâts
- **SolarWinds (2020)** : Mise à jour logicielle malveillante → 18,000 organisations affectées
- **Kaseya (2021)** : Ransomware via fournisseur → 1,500 entreprises impactées

1.2 Solution ChainShield.MA

Plateforme SaaS d'évaluation et de surveillance continue des risques cyber dans les chaînes

d'approvisionnement, avec :

- Cartographie 3D interactive des relations fournisseurs
- Scoring IA en temps réel (0-10)
- OSINT (Open Source Intelligence) spécifique Afrique du Nord
- Monitoring 24/7 avec alertes automatiques
- Rapports PDF exécutifs

1.3 Objectifs du MVP (Hackathon)

Date limite : 8 novembre 2024

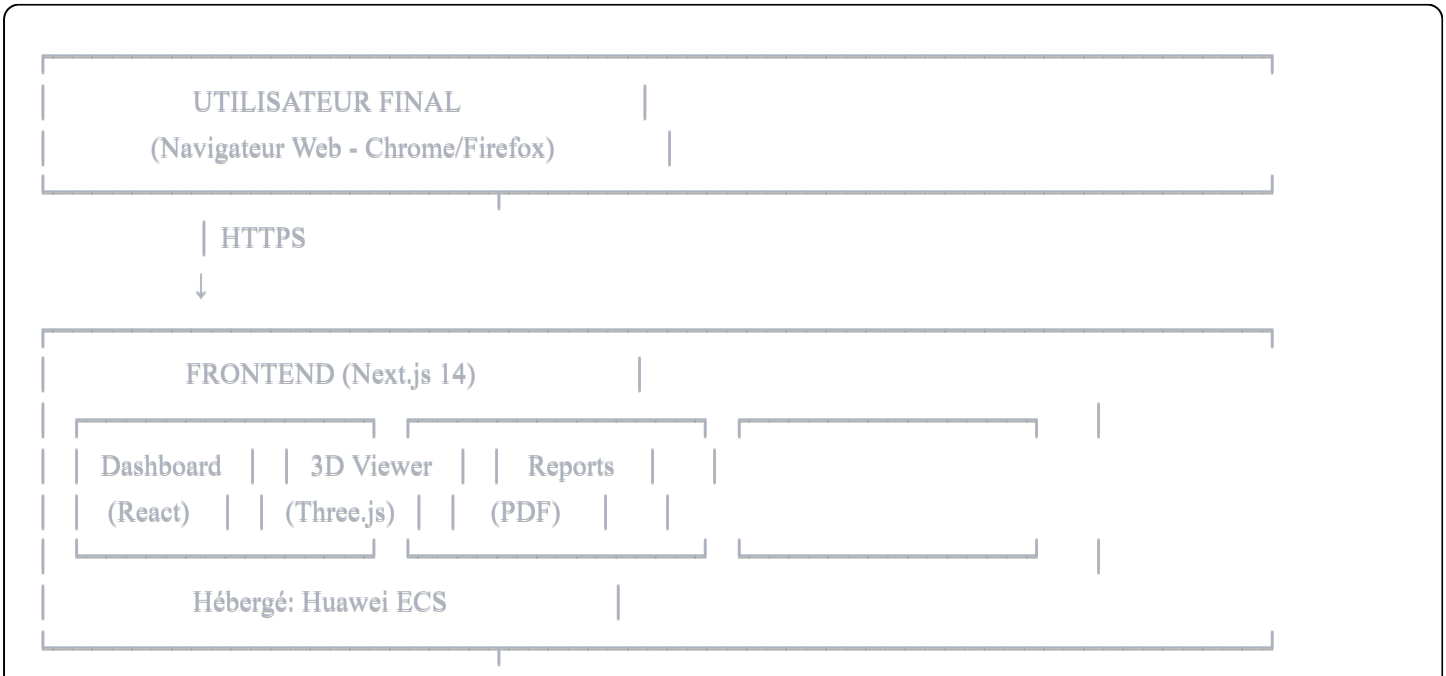
Fonctionnalités minimales :

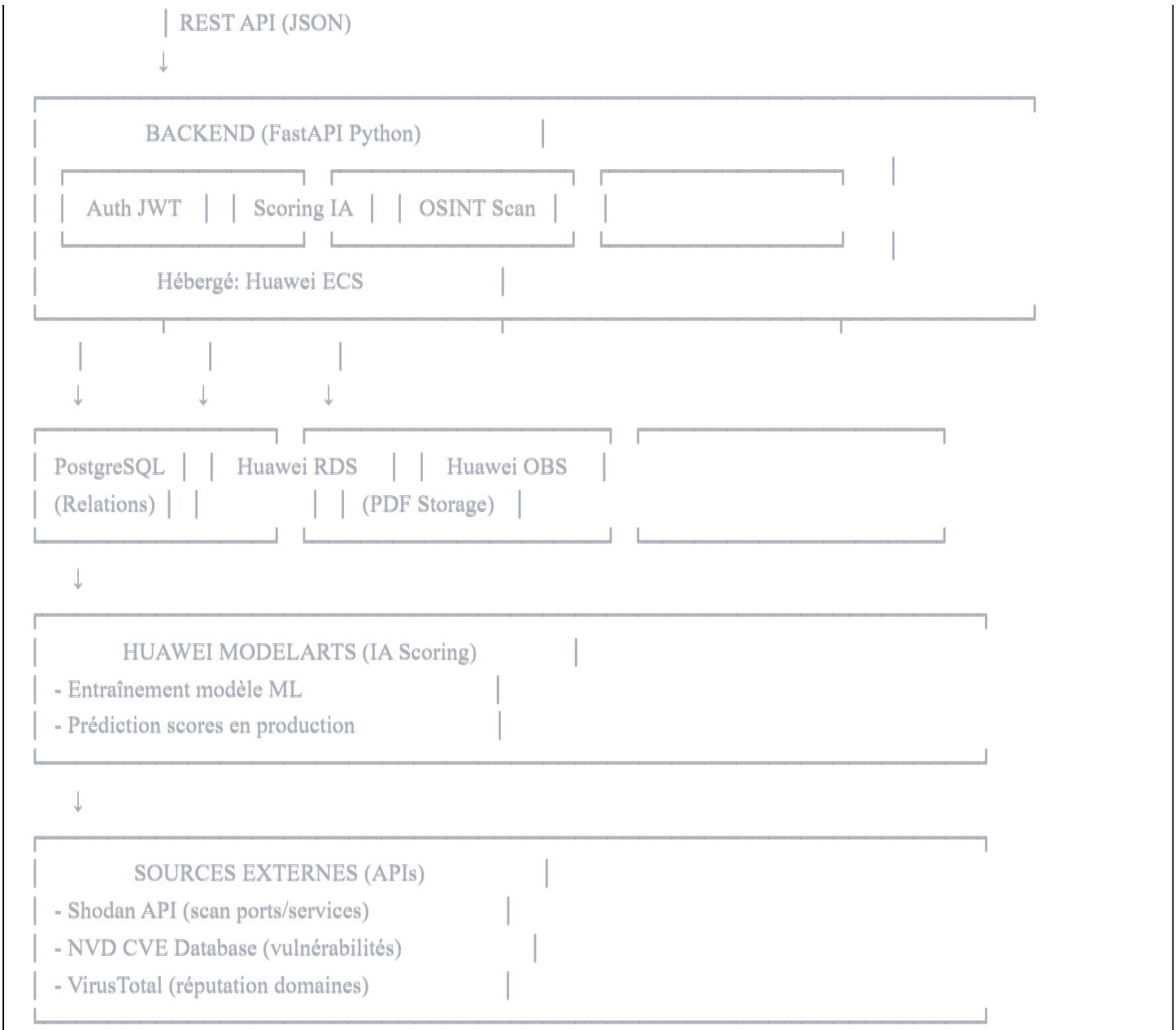
- ✅ Dashboard avec métriques clés
- ✅ Ajout manuel de 5-10 fournisseurs
- ✅ Scan automatique (Shodan + CVE)
- ✅ Calcul score IA
- ✅ Visualisation 3D basique (Three.js)
- ✅ Génération 1 rapport PDF
- ✅ Déploiement Huawei Cloud

---

2. ARCHITECTURE GLOBALE

2.1 Schéma Architecture (Vue d'ensemble)





2.2 Architecture en Couches

Couche	Technologie	Responsabilité
Présentation	Next.js 14, TailwindCSS, Three.js	Interface utilisateur, visualisation
Application	FastAPI (Python 3.12)	Logique métier, orchestration
Intelligence	Scikit-learn, Huawei ModelArts	Scoring IA, détection anomalies
Données	PostgreSQL (Huawei RDS)	Stockage structuré (clients, fournisseurs, scans)
Stockage	Huawei OBS	Fichiers PDF, logs
Sécurité	JWT, RBAC, WAF	Authentification, autorisations, protection

---

3. SPÉCIFICATIONS FONCTIONNELLES

3.1 Modules Principaux

MODULE 1: Authentification & Gestion Utilisateurs

User Stories :

- En tant qu'utilisateur, je veux créer un compte pour accéder à la plateforme
- En tant qu'admin, je veux gérer les rôles (Admin, Analyst, Viewer)

Fonctionnalités :

- Inscription (email + password)
- Connexion JWT (token 24h)
- Rôles RBAC :
  - **Admin** : accès complet
  - **Analyst** : lecture + création scans
  - **Viewer** : lecture seule

Endpoints API :

```
POST /api/auth/register
POST /api/auth/login
GET  /api/auth/me
POST /api/auth/logout
```

MODULE 2: Gestion Fournisseurs

User Stories :

- En tant qu'analyste, je veux ajouter un fournisseur pour le surveiller
- En tant qu'admin, je veux voir tous les fournisseurs de mon organisation

Fonctionnalités :

- Ajout manuel (nom, domaine, IP, secteur)
- Import CSV (bulk)
- Édition/Suppression
- Historique des modifications

Endpoints API :

```
GET    /api/suppliers
POST   /api/suppliers
GET    /api/suppliers/{id}
PUT    /api/suppliers/{id}
DELETE /api/suppliers/{id}
POST   /api/suppliers/import-csv
```

Champs Fournisseur :

json

```
{
  "id": "uuid",
  "name": "string",
  "domain": "example.com",
  "ip_addresses": ["1.2.3.4"],
  "sector": "banking|telecom|industry|other",
  "risk_score": 0-10,
  "status": "active|inactive|blocked",
  "last_scan_date": "ISO 8601",
  "created_at": "ISO 8601",
  "updated_at": "ISO 8601"
}
```

MODULE 3: Scanning & OSINT

User Stories :

- En tant qu'analyste, je veux scanner un fournisseur pour détecter les vulnérabilités
- En tant que système, je veux scanner automatiquement tous les fournisseurs chaque semaine

Sources de données :

1. **Shodan API** :
   - Ports ouverts
   - Services exposés (HTTP, FTP, SSH, etc.)
   - Bannières serveurs
   - Certificats SSL expirés
2. **NVD CVE Database** :
   - Vulnérabilités connues (CVE-XXXX-XXXXX)

- Score CVSS
- Gravité (Low, Medium, High, Critical)

3. **VirusTotal API** :
   - Réputation domaine
   - Détection malware
   - Whois information

Endpoints API :

```
POST /api/scans/start/{supplier_id}
GET  /api/scans/{scan_id}
GET  /api/scans/supplier/{supplier_id}
```

Résultat Scan :

json

```
{
  "scan_id": "uuid",
  "supplier_id": "uuid",
  "status": "pending|running|completed|failed",
  "started_at": "ISO 8601",
  "completed_at": "ISO 8601",
  "findings": [
    {
      "type": "open_port|cve|ssl_issue|malware",
      "severity": "low|medium|high|critical",
      "title": "Port 22 (SSH) exposed",
      "description": "...",
      "remediation": "...",
      "cvss_score": 7.5
    }
  ],
  "raw_data": {
    "shodan": {...},
    "nvd": {...},
    "virustotal": {...}
  }
}
```

MODULE 4: Scoring IA

Algorithme de Scoring (Version MVP):

```python
def calculate_risk_score(scan_results):
    score = 10.0  # Score parfait de départ

    # Pénalités
    for finding in scan_results['findings']:
        if finding['severity'] == 'critical':
            score -= 2.0
        elif finding['severity'] == 'high':
            score -= 1.0
        elif finding['severity'] == 'medium':
            score -= 0.5
        elif finding['severity'] == 'low':
            score -= 0.2

    # Bonus
    if has_valid_ssl_certificate(scan_results):
        score += 0.5
    if no_critical_cves_last_30_days(scan_results):
        score += 0.5

    # Normalisation 0-10
    score = max(0.0, min(10.0, score))

    return round(score, 1)
```

Évolution post-MVP (avec Huawei ModelArts):

- Entraînement modèle ML sur dataset de 10,000+ fournisseurs
- Features : nombre CVE, ancienneté domaine, secteur, pays, historique incidents
- Algorithme : Random Forest ou XGBoost
- Prédiction : probabilité d'incident dans les 90 jours

Endpoints API :

```
GET /api/scoring/{supplier_id}
POST /api/scoring/recalculate/{supplier_id}
```

MODULE 5: Visualisation 3D

Technologie : Three.js

Représentation :

- **Nœud central** (bleu) = Votre entreprise
- **Nœuds satellites** = Fournisseurs
  - Vert (score > 7) = Faible risque
  - Jaune (score 4-7) = Risque moyen
  - Rouge (score < 4) = Risque élevé
- **Lignes** = Relations
  - Épaisseur = importance du fournisseur
  - Couleur = niveau de risque

Interactions :

- Rotation 3D (souris)
- Zoom (scroll)
- Clic sur nœud → affiche détails fournisseur
- Filtres par score/secteur

Endpoint API :

```
GET /api/graph/3d
Response: {
  "nodes": [
    {"id": "main", "type": "company", "x": 0, "y": 0, "z": 0},
    {"id": "sup1", "type": "supplier", "score": 8.5, "x": 10, "y": 5, "z": -3},
    ...
  ],
  "edges": [
    {"source": "main", "target": "sup1", "weight": 0.8},
    ...
  ]
}
```

MODULE 6: Rapports PDF

Technologie : ReportLab (Python)

Structure Rapport :

1. **Page de garde**
   - Logo ChainShield
   - Nom entreprise cliente
   - Date génération
   - Période analysée
2. **Executive Summary**
   - Score global supply chain (0-10)
   - Nombre total fournisseurs
   - Fournisseurs à risque critique
   - Tendance évolution (↗️ ↘️)
3. **Top 10 Risques**
   - Liste fournisseurs score < 5
   - CVE critiques non patchées
   - Recommandations prioritaires
4. **Détails par Fournisseur**
   - Fiche identité
   - Score détaillé
   - Findings (vulnérabilités)
   - Actions recommandées
5. **Méthodologie**
   - Sources de données
   - Calcul du score
   - Limites de l'analyse

Endpoint API :

```
POST /api/reports/generate/{supplier_id}
GET  /api/reports/{report_id}/download
```

MODULE 7: Dashboard

Métriques affichées :

- **Score global** : moyenne pondérée tous fournisseurs
- **Répartition risques** :
  - Nombre fournisseurs verts/jaunes/rouges
  - Graphique donut
- **Top 5 fournisseurs à risque**
- **Tendances** : évolution scores 30 derniers jours
- **Derniers scans** : tableau avec statuts
- **Alertes récentes** : CVE critiques découvertes

Technologies :

- Recharts (graphiques)
- React Table (tableaux)
- Lucide Icons (icônes)

---

4. STACK TECHNIQUE DÉTAILLÉE

4.1 Frontend

Next.js 14 (App Router)

```
chainshield-frontend/
├── app/
│   ├── (auth)/
│   │   ├── login/page.tsx
│   │   └── register/page.tsx
│   ├── dashboard/
│   │   ├── page.tsx
│   │   └── layout.tsx
│   ├── suppliers/
│   │   ├── page.tsx
│   │   ├── [id]/page.tsx
│   │   └── add/page.tsx
│   ├── 3d-view/page.tsx
│   ├── reports/page.tsx
│   └── layout.tsx
├── components/
│   ├── ui/ (shadcn/ui)
│   ├── Dashboard/
│   │   ├── MetricsCard.tsx
│   │   ├── RiskChart.tsx
│   │   └── SupplierTable.tsx
│   ├── 3D/
│   │   └── SupplyChainGraph.tsx
│   └── Suppliers/
│       └── SupplierForm.tsx
├── lib/
│   ├── api.ts (axios config)
│   ├── auth.ts (JWT helpers)
│   └── utils.ts
├── public/
└── package.json
```

Dépendances principales :

json

```
{
 "dependencies": {
  "next": "14.0.0",
  "react": "18.2.0",
  "tailwindcss": "3.3.0",
  "three": "0.158.0",
  "@react-three/fiber": "8.15.0",
  "@react-three/drei": "9.88.0",
  "recharts": "2.9.0",
  "axios": "1.6.0",
  "lucide-react": "0.292.0",
  "shadcn/ui": "latest"
 }
}
```

4.2 Backend

FastAPI (Python 3.12)

```
chainshield-backend/
├── app/
│   ├── main.py
│   ├── config.py
│   ├── database.py
│   ├── models/
│   │   ├── user.py
│   │   ├── supplier.py
│   │   ├── scan.py
│   │   └── report.py
│   ├── schemas/
│   │   ├── user.py
│   │   ├── supplier.py
│   │   └── scan.py
│   ├── routers/
│   │   ├── auth.py
│   │   ├── suppliers.py
│   │   ├── scans.py
│   │   ├── scoring.py
│   │   ├── graph.py
│   │   └── reports.py
│   ├── services/
│   │   ├── osint.py (Shodan, CVE, VT)
│   │   ├── scoring.py (ML algorithm)
│   │   ├── pdf_generator.py
│   │   └── huawei_modelarts.py
│   └── utils/
│       ├── auth.py (JWT)
│       ├── security.py
│       └── validators.py
├── tests/
├── requirements.txt
└── Dockerfile
```

requirements.txt :

```
fastapi==0.104.1
uvicorn[standard]==0.24.0
sqlalchemy==2.0.23
psycopg2-binary==2.9.9
pydantic==2.5.0
python-jose[cryptography]==3.3.0
passlib[bcrypt]==1.7.4
python-multipart==0.0.6
shodan==1.30.1
requests==2.31.0
scikit-learn==1.3.2
reportlab==4.0.7
boto3==1.29.7  # Pour Huawei OBS (compatible S3)
```

---

5. BASE DE DONNÉES

5.1 Schéma PostgreSQL

sql

```sql
-- Table: users
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    hashed_password VARCHAR(255) NOT NULL,
    full_name VARCHAR(255),
    role VARCHAR(50) DEFAULT 'viewer', -- admin, analyst, viewer
    organization_id UUID REFERENCES organizations(id),
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Table: organizations
CREATE TABLE organizations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    domain VARCHAR(255),
    sector VARCHAR(100),
    country VARCHAR(2) DEFAULT 'MA',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Table: suppliers
CREATE TABLE suppliers (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID REFERENCES organizations(id),
    name VARCHAR(255) NOT NULL,
    domain VARCHAR(255),
    ip_addresses JSONB, -- ["1.2.3.4", "5.6.7.8"]
    sector VARCHAR(100),
    risk_score DECIMAL(3,1) DEFAULT 0.0,
    status VARCHAR(50) DEFAULT 'active',
    last_scan_date TIMESTAMP,
    metadata JSONB, -- Champs personnalisés
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Table: scans
CREATE TABLE scans (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    supplier_id UUID REFERENCES suppliers(id),
    initiated_by UUID REFERENCES users(id),
    status VARCHAR(50) DEFAULT 'pending', -- pending, running, completed, failed
    started_at TIMESTAMP,
```

```
    completed_at TIMESTAMP,
    findings JSONB, -- Array of findings
    raw_data JSONB, -- Shodan, CVE, VT responses
    error_message TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Table: reports
CREATE TABLE reports (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID REFERENCES organizations(id),
    generated_by UUID REFERENCES users(id),
    type VARCHAR(50) DEFAULT 'full', -- full, supplier, summary
    supplier_id UUID REFERENCES suppliers(id), -- NULL si report global
    file_url VARCHAR(500), -- URL Huawei OBS
    metadata JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Table: alerts
CREATE TABLE alerts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    supplier_id UUID REFERENCES suppliers(id),
    severity VARCHAR(50), -- low, medium, high, critical
    title VARCHAR(255),
    description TEXT,
    is_read BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Indexes pour performance
CREATE INDEX idx_suppliers_org ON suppliers(organization_id);
CREATE INDEX idx_scans_supplier ON scans(supplier_id);
CREATE INDEX idx_scans_status ON scans(status);
CREATE INDEX idx_reports_org ON reports(organization_id);
CREATE INDEX idx_alerts_read ON alerts(is_read, created_at);
```

---

6. APIS & INTÉGRATIONS

6.1 Shodan API

Documentation : https://developer.shodan.io/api

Utilisation :

```python
import shodan

api = shodan.Shodan(os.getenv('SHODAN_API_KEY'))

# Scan par IP
result = api.host('8.8.8.8')
# Retourne: ports ouverts, services, vulnérabilités, certificats SSL

# Scan par domaine
result = api.search(f'hostname:{domain}')
```

Coût : 59$/mois (Small Business Plan) = 10,000 scans/mois

6.2 NVD CVE Database

Documentation : https://nvd.nist.gov/developers/vulnerabilities

Utilisation :

```python
import requests

url = f"https://services.nvd.nist.gov/rest/json/cves/2.0?keywordSearch={technology}"
response = requests.get(url)
cves = response.json()['vulnerabilities']

# Filtrer par criticité CVSS > 7.0
critical_cves = [cve for cve in cves if cve['cvss']['baseScore'] >= 7.0]
```

Coût : Gratuit (rate limit: 5 req/30s sans API key)

6.3 VirusTotal API

Documentation : https://developers.virustotal.com/reference/overview

Utilisation :

```python
```

```
import requests

headers = {'x-apikey': os.getenv('VT_API_KEY')}
url = f"https://www.virustotal.com/api/v3/domains/{domain}"
response = requests.get(url, headers=headers)
reputation = response.json()['data']['attributes']
```

Coût : Gratuit (500 req/jour) ou Premium (190$/mois)

6.4 Huawei ModelArts SDK

Documentation : https://support.huaweicloud.com/intl/en-us/sdkreference-modelarts/

Utilisation :

```
python

from modelarts.session import Session
from modelarts.model import Model

session = Session(
    access_key=os.getenv('HUAWEI_AK'),
    secret_key=os.getenv('HUAWEI_SK'),
    region_name='af-south-1'  # Région Maroc
)

# Charger modèle entraîné
model = Model(session, model_id='chainshield-scoring-v1')

# Prédiction
prediction = model.predict(data={
    'cve_count': 5,
    'open_ports': [22, 80, 443],
    'ssl_valid': True,
    'sector': 'banking'
})

risk_score = prediction['score']  # 0-10
```

---

7. SÉCURITÉ

7.1 Authentification JWT

Flux :

1. User POST `/api/auth/login` avec email/password
2. Backend vérifie credentials → génère JWT token (expire 24h)
3. Frontend stocke token dans httpOnly cookie
4. Toutes les requêtes incluent header `Authorization: Bearer {token}`
5. Backend valide token sur chaque requête protégée

Implémentation :

```python
from jose import JWTError, jwt
from datetime import datetime, timedelta

SECRET_KEY = os.getenv('JWT_SECRET_KEY')
ALGORITHM = "HS256"

def create_access_token(data: dict):
    to_encode = data.copy()
    expire = datetime.utcnow() + timedelta(hours=24)
    to_encode.update({"exp": expire})
    return jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)

def verify_token(token: str):
    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
        return payload
    except JWTError:
        return None
```

7.2 RBAC (Role-Based Access Control)

Rôle	Permissions
Admin	Tout + gestion utilisateurs
Analyst	Lecture, création scans, rapports
Viewer	Lecture seule (dashboard, rapports)

Implémentation :

```python
```

```python
from fastapi import Depends, HTTPException

def require_role(allowed_roles: list):
    def decorator(current_user: User = Depends(get_current_user)):
        if current_user.role not in allowed_roles:
            raise HTTPException(status_code=403, detail="Insufficient permissions")
        return current_user
    return decorator

# Usage
@router.post("/suppliers")
def create_supplier(
    supplier: SupplierCreate,
    user: User = Depends(require_role(['admin', 'analyst']))
):
    ...
```

7.3 Protection Données Sensibles

- **Passwords** : Hachés avec bcrypt (12 rounds)
- **API Keys externes** : Stockées chiffrées AES-256 dans DB
- **Secrets** : Variables d'environnement (jamais dans code)
- **HTTPS** : Obligatoire (Huawei WAF + Let's Encrypt)
- **Rate Limiting** : 100 req/min par IP
- **CORS** : Whitelist domaines autorisés

7.4 Conformité RGPD

- Consentement explicite collecte données
- Droit accès/rectification/suppression (endpoints API)
- Pseudonymisation logs (pas d'IP en clair)
- Hébergement données EU/Maroc (Huawei Cloud)
- DPO désigné : Kaouthar Belkebir

---

8. INFRASTRUCTURE HUAWEI CLOUD

8.1 Services Utilisés

ECS (Elastic Cloud Server)

Usage : Hébergement frontend + backend

Configuration MVP :

- **Type** : c6.large.2 (2 vCPU, 4GB RAM)
- **OS** : Ubuntu 22.04 LTS
- **Disque** : 40GB SSD
- **Région** : af-south-1 (Johannesburg, proche Maroc)
- **Prix** : ~50$/mois

Setup :

```bash
# Installation Docker
sudo apt update && sudo apt install docker.io docker-compose -y

# Déploiement
git clone https://github.com/chainshield/backend
cd backend
docker-compose up -d
```

RDS (Relational Database Service)

Usage : PostgreSQL 15

Configuration MVP :

- **Type** : Single instance (dev/test)
- **Specs** : 2 vCPU, 4GB RAM, 100GB storage
- **Backup** : Automatique quotidien (7 jours rétention)
- **Prix** : ~60$/mois

OBS (Object Storage Service)

Usage : Stockage rapports PDF + logs

Configuration :

- **Bucket** : chainshield-reports-prod
- **ACL** : Private (accès via signed URLs)
- **Lifecycle** : Suppression auto après 1 an
- **Prix** : ~5$/mois (50GB)

ModelArts

Usage : Entraînement + déploiement modèle IA

Configuration :

- **Notebook** : Development (Jupyter, 2 vCPU, 8GB RAM)
- **Training** : 4 vCPU, 16GB RAM (1-2h)
- **Inference** : 2 vCPU, 4GB RAM (real-time API)
- **Prix** : ~80$/mois

WAF (Web Application Firewall)

Usage : Protection contre DDoS, injection SQL, XSS

Configuration :

- **Mode** : Detection + Prevention
- **Rules** : OWASP Top 10
- **Rate Limiting** : 1000 req/min par IP
- **Prix** : ~100$/mois

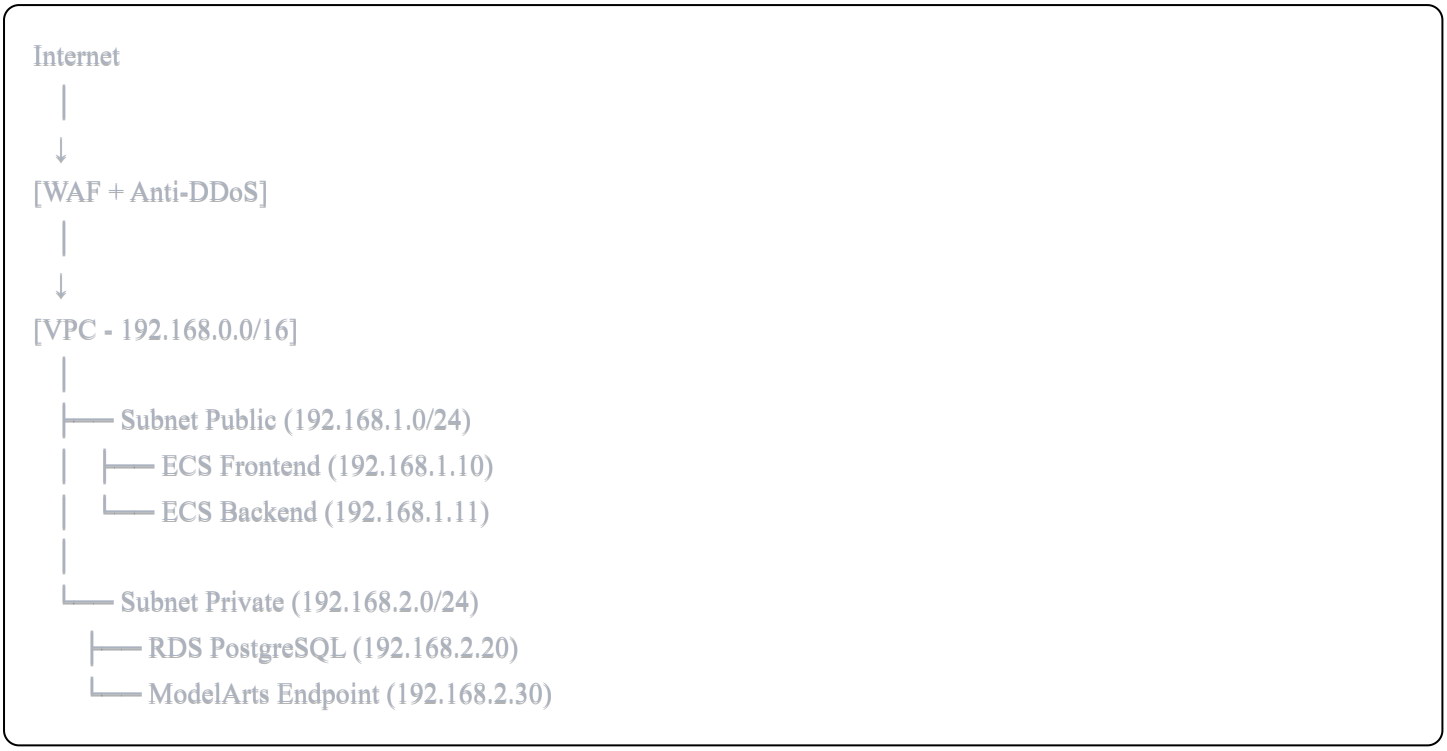
IAM (Identity & Access Management)

Usage : Gestion accès services Huawei

Configuration :

- User chainshield-admin (accès complet)
- User chainshield-backend (accès RDS, OBS, ModelArts)
- User chainshield-frontend (accès OBS lecture seule)

8.2 Architecture Réseau



Règles Firewall :

- Port 443 (HTTPS) : Public → Frontend
- Port 8000 (API) : Frontend → Backend
- Port 5432 (PostgreSQL) : Backend → RDS
- Tout autre port : DENY

8.3 CI/CD avec DevCloud

Pipeline Automatique :

yaml

```yaml
# .github/workflows/deploy.yml
name: Deploy to Huawei Cloud

on:
  push:
    branches: [main]

jobs:
  deploy-backend:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Build Docker Image
        run: |
          docker build -t chainshield-backend:latest ./backend

      - name: Push to Huawei SWR
        run: |
          docker tag chainshield-backend:latest \
            swr.af-south-1.myhuaweicloud.com/chainshield/backend:latest
          docker push swr.af-south-1.myhuaweicloud.com/chainshield/backend:latest

      - name: Deploy to ECS
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.HUAWEI_ECS_IP }}
          username: ubuntu
          key: ${{ secrets.SSH_PRIVATE_KEY }}
          script: |
            docker pull swr.af-south-1.myhuaweicloud.com/chainshield/backend:latest
            docker-compose down && docker-compose up -d

  deploy-frontend:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Build Next.js
        run: |
          cd frontend
          npm install
          npm run build

      - name: Deploy to ECS
        run: |
```

```
        rsync -avz --delete ./frontend/out/ \
          ubuntu@${{ secrets.HUAWEI_ECS_IP }}:/var/www/chainshield/
```

8.4 Monitoring avec Cloud Eye

Métriques surveillées :

- CPU/RAM utilisation ECS
- Latence API (P50, P95, P99)
- Requêtes/seconde
- Erreurs 5xx
- Connexions DB actives
- Temps réponse ModelArts

Alertes configurées :

- CPU > 80% pendant 5 min → Email + SMS
- Erreurs 5xx > 10/min → Email équipe
- DB connexions > 90% → Email admin
- Disque > 85% → Email + Slack

---

9. PLAN DE DÉVELOPPEMENT

9.1 Sprint 1 (Nov 1-3) - Foundation

Responsables : Abdellatif (Backend), Mouad (Frontend)

Tâches Backend :

- [ ] Setup projet FastAPI
- [ ] Configuration Huawei RDS PostgreSQL
- [ ] Modèles SQLAlchemy (User, Organization, Supplier)
- [ ] Endpoints Auth (register, login, logout)
- [ ] Middleware JWT
- [ ] Tests unitaires auth

Tâches Frontend :

- [ ] Setup Next.js 14 + TailwindCSS
- [ ] Pages auth (login, register)

- [ ] Layout dashboard
- [ ] Intégration API auth
- [ ] Protected routes

Livrables :

- ✅ Utilisateur peut créer compte et se connecter
- ✅ Dashboard vide accessible après login

9.2 Sprint 2 (Nov 4-5) - Core Features

Responsables : Reda (OSINT), Ikram (IA), Mouad (Dashboard)

Tâches Backend :

- [ ] Service OSINT (Shodan API integration)
- [ ] Endpoints Suppliers CRUD
- [ ] Endpoint POST /scans/start
- [ ] Algorithme scoring V1 (règles simples)
- [ ] Background tasks (Celery ou simple async)

Tâches Frontend :

- [ ] Page liste fournisseurs
- [ ] Formulaire ajout fournisseur
- [ ] Page détails fournisseur
- [ ] Dashboard métriques (cards)
- [ ] Bouton "Lancer scan"

Tâches IA :

- [ ] Dataset test (50 fournisseurs fictifs)
- [ ] Notebook Jupyter algorithme scoring
- [ ] Export modèle pickle
- [ ] Intégration backend

Livrables :

- ✅ Ajout/modification fournisseurs
- ✅ Lancement scan manuel
- ✅ Calcul score affiché

9.3 Sprint 3 (Nov 6-7) - Advanced Features

Responsables : Mouad (3D), Kaouthar (PDF), Abdellatif (Huawei)

Tâches :

- [ ] Visualisation 3D avec Three.js
- [ ] Endpoint GET /graph/3d
- [ ] Génération PDF avec ReportLab
- [ ] Upload PDF vers Huawei OBS
- [ ] Déploiement complet Huawei Cloud
- [ ] Tests E2E

Livrables :

- ✅ Graphe 3D interactif fonctionne
- ✅ Génération rapport PDF
- ✅ Application hébergée Huawei Cloud
- ✅ Démo ready pour le 8 novembre

9.4 Sprint 4 (Nov 8) - Finalisation & Pitch

Responsables : Toute l'équipe

Tâches :

- [ ] Tests finaux
- [ ] Fixes bugs critiques
- [ ] Optimisation performances
- [ ] Préparation démo (scénario scripté)
- [ ] Répétition pitch
- [ ] Soumission projet

Scénario Démo (3 minutes) :

1. **Login** (10s) : "Je me connecte en tant qu'analyste cyber chez AttijariWafa Bank"
2. **Dashboard** (30s) : "Voici notre supply chain : 156 fournisseurs, dont 7 à risque critique"
3. **Détail fournisseur** (45s) : "Prenons TechSupply.ma, score 3.2/10. Pourquoi ? CVE critique non patchée, port SSH exposé"
4. **Scan temps réel** (30s) : "Je lance un nouveau scan... Shodan détecte 3 nouveaux ports ouverts"
5. **3D View** (45s) : "Visualisation 3D de notre chaîne. En rouge, les fournisseurs à traiter en priorité"
6. **Rapport PDF** (20s) : "Génération rapport exécutif pour le RSSI en 1 clic"

---

10. TESTS & QUALITÉ

10.1 Tests Unitaires

Backend (pytest) :

```python
# tests/test_scoring.py
def test_calculate_risk_score_critical_cve():
    scan_results = {
        'findings': [
            {'severity': 'critical', 'type': 'cve'}
        ]
    }
    score = calculate_risk_score(scan_results)
    assert score == 8.0  # 10 - 2
```

Couverture cible : 80% du code backend

10.2 Tests d'Intégration

Endpoints API :

```python
# tests/test_api_suppliers.py
def test_create_supplier_success(client, auth_headers):
    response = client.post(
        "/api/suppliers",
        json={"name": "Test Corp", "domain": "test.com"},
        headers=auth_headers
    )
    assert response.status_code == 201
    assert response.json()['name'] == "Test Corp"
```

10.3 Tests E2E

Frontend (Playwright) :

```typescript
```

```
// e2e/supplier-workflow.spec.ts
test('complete supplier scan workflow', async ({ page }) => {
  await page.goto('/login');
  await page.fill('input[name="email"]', 'test@chainshield.ma');
  await page.fill('input[name="password"]', 'password123');
  await page.click('button[type="submit"]');

  await page.goto('/suppliers/add');
  await page.fill('input[name="name"]', 'ACME Corp');
  await page.fill('input[name="domain"]', 'acme.com');
  await page.click('button:has-text("Ajouter")');

  await page.click('button:has-text("Lancer scan")');
  await page.waitForSelector('text=Scan terminé');

  const score = await page.textContent('.risk-score');
  expect(parseFloat(score)).toBeGreaterThan(0);
});
```

10.4 Tests de Performance

Load Testing (Locust) :

```
python
```

```python
# locustfile.py
from locust import HttpUser, task, between

class ChainShieldUser(HttpUser):
    wait_time = between(1, 3)

    def on_start(self):
        # Login
        response = self.client.post("/api/auth/login", json={
            "email": "test@chainshield.ma",
            "password": "password123"
        })
        self.token = response.json()['access_token']
        self.headers = {"Authorization": f"Bearer {self.token}"}

    @task(3)
    def get_suppliers(self):
        self.client.get("/api/suppliers", headers=self.headers)

    @task(1)
    def start_scan(self):
        self.client.post("/api/scans/start/supplier-uuid", headers=self.headers)
```

Objectifs Performance :

- Temps réponse API < 200ms (P95)
- Support 100 utilisateurs concurrents
- Dashboard load < 2s
- Scan complet < 30s

10.5 Tests de Sécurité

Checklist :

- [ ] Injection SQL (SQLMap)
- [ ] XSS (Burp Suite)
- [ ] CSRF tokens
- [ ] Rate limiting
- [ ] JWT expiration
- [ ] HTTPS obligatoire
- [ ] Secrets non exposés
- [ ] CORS correctement configuré

☐ Headers sécurité (CSP, HSTS)

Outils :

- OWASP ZAP (scan automatique)
- Bandit (analyse code Python)
- npm audit (dépendances Node.js)

---

11. DOCUMENTATION

11.1 Documentation API (OpenAPI/Swagger)

FastAPI génère automatiquement la doc interactive : `http://localhost:8000/docs`

Exemple endpoint documenté :

```python
@router.post("/suppliers", response_model=SupplierResponse, status_code=201)
async def create_supplier(
    supplier: SupplierCreate,
    current_user: User = Depends(get_current_user)
):
    """
    Créer un nouveau fournisseur à surveiller.

    - **name**: Nom du fournisseur (requis)
    - **domain**: Nom de domaine (ex: example.com)
    - **sector**: Secteur d'activité (banking, telecom, industry, other)

    Retourne le fournisseur créé avec son ID unique.
    """
    ...
```

11.2 README Technique

```markdown
```

ChainShield.MA - Setup Guide

Prérequis
- Python 3.12+
- Node.js 18+
- PostgreSQL 15+
- Docker & Docker Compose

Installation Backend
```bash
cd backend
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate
pip install -r requirements.txt

# Configuration
cp .env.example .env
# Éditer .env avec vos clés API

# Migration DB
alembic upgrade head

# Lancer serveur
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

Installation Frontend
```bash
cd frontend
npm install
cp .env.local.example .env.local
# Éditer .env.local

# Mode dev
npm run dev

# Build production
npm run build
npm start
```

Variables d'Environnement

Backend (.env)

DATABASE_URL=postgresql://user:pass@localhost:5432/chainshield JWT_SECRET_KEY=your-secret-key-change-in-production SHODAN_API_KEY=your-shodan-key VIRUSTOTAL_API_KEY=your-vt-key HUAWEI_ACCESS_KEY=your-huawei-ak HUAWEI_SECRET_KEY=your-huawei-sk

```
### Frontend (.env.local)
```

NEXT_PUBLIC_API_URL=http://localhost:8000

```
## Tests
```bash
# Backend
cd backend
pytest tests/ -v --cov=app

# Frontend
cd frontend
npm run test
npm run test:e2e
```
```

---

12. MÉTRIQUES DE SUCCÈS

12.1 Objectifs Hackathon (8 Nov 2024)

Critères techniques :

- ✅ Application fonctionnelle déployée Huawei Cloud
- ✅ 3+ services Huawei utilisés (ECS, RDS, OBS, ModelArts)
- ✅ Démo live sans bugs
- ✅ Code sur GitHub (public)

Critères business :

- ✅ Pitch 7 minutes respecté
- ✅ Problème + Solution + Démo + Business model
- ✅ Q&A jury préparé
- ✅ Documentation complète

12.2 KPIs Post-Hackathon

Mois 1-3 :

- 5 entreprises bêta-testers recrutées
- 100+ fournisseurs scannés
- Feedback NPS > 8/10
- 0 incident sécurité

Mois 4-6 :

- 3 clients payants (1 Basic, 2 Pro)
- 10,000€ MRR
- Précision scoring IA > 85%
- Temps réponse API < 150ms

Année 1 :

- 25 clients payants
- 137K€ ARR
- Expansion Tunisie + Algérie
- Levée seed 500K€

---

13. RISQUES & MITIGATION

Risque	Impact	Probabilité	Mitigation
API Shodan down	Haut	Faible	Cache résultats + fallback CVE only
Dépassement budget Huawei	Moyen	Moyen	Monitoring costs + alertes
Faux positifs scoring IA	Haut	Moyen	Validation manuelle + amélioration continue
RGPD non-conformité	Très haut	Faible	Audit juridique + DPO
Concurrence (SecurityScorecard)	Moyen	Élevé	Focus différenciation (3D, Afrique, prix)
Manque adoption marché	Haut	Moyen	Pilotes gratuits + marketing contenu

---

14. PROCHAINES ÉTAPES (POST-MVP)

Phase 2 (Mois 2-3)

Fonctionnalités avancées :

- Import automatique fournisseurs (API ERP)
- Historique évolution scores (graphiques temps)
- Alertes email/SMS automatiques
- Intégration Slack/Teams
- API webhooks pour SIEM
- Module conformité (ISO 27001, NIST)

Améliorations IA :

- Dataset 10,000+ fournisseurs réels
- Features engineering avancé
- A/B testing algorithmes
- Explainability (SHAP values)

Phase 3 (Mois 4-6)

Expansion :

- Support multi-langues (AR, EN, FR)
- Version mobile (React Native)
- Marketplace intégrations (Splunk, QRadar)
- API publique pour partenaires
- Programme affiliation

---

15. ANNEXES

Annexe A : Glossaire

- **OSINT** : Open Source Intelligence (collecte info publique)
- **CVE** : Common Vulnerabilities and Exposures
- **CVSS** : Common Vulnerability Scoring System (0-10)
- **Supply Chain** : Chaîne d'approvisionnement
- **SaaS** : Software as a Service
- **RBAC** : Role-Based Access Control

- **JWT** : JSON Web Token
- **SIEM** : Security Information and Event Management

Annexe B : Contacts Équipe

Membre	Email	Téléphone	Rôle urgence
Kaouthar Belkebir	kaouthar@chainshield.ma	+212 6XX XXX XXX	CEO - Décisions stratégiques
Reda Ouzidan	reda@chainshield.ma	+212 6XX XXX XXX	Tech Lead - Bugs critiques
Abdellatif Tazarni	abdellatif@chainshield.ma	+212 6XX XXX XXX	DevOps - Infrastructure
Ikram Naoui	ikram@chainshield.ma	+212 6XX XXX XXX	Data Science - IA
Mouad Omlil	mouad@chainshield.ma	+212 6XX XXX XXX	Frontend - UI/UX

Annexe C : Ressources

Documentation :

- Huawei Cloud : https://support.huaweicloud.com/intl/en-us/
- FastAPI : https://fastapi.tiangolo.com/
- Next.js : https://nextjs.org/docs
- Three.js : https://threejs.org/docs/

Outils :

- Design : Figma (https://figma.com)
- Gestion projet : Notion (https://notion.so)
- Code : GitHub (https://github.com/chainshield)
- Communication : Discord

🎯 CONCLUSION

Ce cahier des charges définit l'architecture complète de **ChainShield.MA** pour la Huawei Cloud ICT Competition 2025.

Objectif immédiat : MVP fonctionnel le 8 novembre 2024
Vision long-terme : Leader cybersécurité supply chain en Afrique

Prochaine action : Sprint 1 démarrage immédiat !

Document rédigé par : Équipe ChainShield.MA

Version : 1.0

Date : 3 novembre 2024

Statut : CONFIDENTIEL - Ne pas diffuser