# Document d’Architecture Technique (DAT)

**Projet** : Refonte SI BusinessCare
**Version** : 1.0
**Date** : 19/05/2025

---

## 1. Contexte et périmètre

### 1.1 Contexte

- Modernisation du SI BusinessCare (Web, mobile, desktop).
- Migration vers conteneurs Docker + orchestration.
- Haute disponibilité, sécurité renforcée, CI/CD, monitoring, sauvegarde, support 24×7.

### 1.2 Périmètre

- **Applications** : Next.js (Front), FastAPI (API), Android, Java standalone.
- **Intégrations** : Stripe, NFC, chatbot.
- **Infra réseau** : Proxmox/ESXi, OPNsense HA, VPN site-to-site & client-to-site.
- **Stockage** : TrueNAS/Synology pour backups & données partagées.
- **Base de données** : MySQL conteneurisé.
- **CI/CD** : GitLab CI/CD & runners Docker.
- **Supervision** : Prometheus, Grafana, ELK.

---

## 2. Architecture Applicative

### 2.1 Composants métier

| Couche | Technologie | Entrées / Sorties |
|---|---|---|
| **Front** | Next.js (Node.js) | Appels HTTP(s) vers `api.$DOMAIN` |
| **API** | FastAPI (Uvicorn/Gunicorn) | Routes : `/auth`, `/company`, `/checkout`, … |
| **DB** | MySQL | TCP 3306 (réseau Docker interne) |
| **Stockage** | Volume Docker ( `/app/uploads` ) | Fichiers, certificats, assets statiques |
| **Mail** | Postfix | SMTP 25/587 pour e-mails transactionnels |
| **CI/CD** | GitLab Runner | Build/push images, déploiement staging/prod |

### 2.2 Diagramme fonctionnel

```
Utilisateur <-> Nginx <-> Next.js <-> Nginx <-> FastAPI <-> MySQL
```

---

## 3. Architecture Infrastructure

### 3.1 Topologie réseau

| Segment / VLAN | Sous-réseau | Usage |
|---|---|---|
| **DMZ** | 10.10.10.0/24 | Nginx, frontend, certbot |
| **Infra conteneurs** | 10.10.20.0/24 | Services Docker |
| **DB** | 10.10.30.0/24 | MySQL, backups |
| **Management** | 10.10.40.0/24 | Proxmox / OPNsense GUI |
| **VPN S2S** | IPSec (AES-GCM) | Chiffrement site-to-site |
| **VPN C2S** | OpenVPN/SSL | Accès administratif |

### 3.2 Matériel & virtualisation

- 2× serveurs Proxmox/vSphere ESXi (16 cœurs CPU, 64 Go RAM, RAID10).
- Cluster OPNsense HA (deux interfaces réseau).
- NAS TrueNAS/Synology (iSCSI/NFS) pour sauvegardes.

### 3.3 Conteneurisation (docker-compose)

```yaml
version: '3.8'
services:
  nginx:
    image: nginx:latest
    ports: ['80:80','443:443']
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./nginx/certbot/conf:/etc/letsencrypt:ro
    networks:
      - pa-network

  db:
    image: mysql:latest
    ports: ['3306:3306']
    env_file: .env
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
      - db_data:/var/lib/mysql
```

```
    networks:
      - pa-network

  api:
    build:
      context: .
      dockerfile: ./docker/api.dockerfile
    depends_on: [db]
    env_file: .env
    volumes:
      - ./uploads:/app/uploads
    networks:
      - pa-network

  frontend:
    build:
      context: .
      dockerfile: ./docker/next.dockerfile
      args:
        - NEXT_PUBLIC_API_URL_ARG=https://api.${DOMAIN}
        - NEXT_PUBLIC_STRIPE_PUBLIC_KEY_ARG=${NEXT_PUBLIC_STRIPE_PUBLIC_KEY}
    env_file: .env
    networks:
      - pa-network

  postfix:
    image: boky/postfix:latest
    ports: ['25:25','587:587']
    env_file: .env
    volumes:
      - postfix_data:/var/spool/postfix
      - ./nginx/certbot/conf:/etc/letsencrypt:ro
    networks:
      - pa-network

volumes:
  db_data:
  postfix_data:

networks:
  pa-network:
    driver: bridge
```

## 3.4 Dockerfiles

`docker/api.dockerfile`

```
FROM python:3.11-slim
WORKDIR /app
COPY app/requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app/ .
```

```
RUN mkdir uploads
CMD ["uvicorn","main:app","--host","0.0.0.0","--port","8000"]
```

`docker/next.dockerfile`

```
FROM node:18-alpine
ARG NEXT_PUBLIC_API_URL_ARG
ARG NEXT_PUBLIC_STRIPE_PUBLIC_KEY_ARG
ENV NEXT_PUBLIC_API_URL=$NEXT_PUBLIC_API_URL_ARG
ENV NEXT_PUBLIC_STRIPE_PUBLIC_KEY=$NEXT_PUBLIC_STRIPE_PUBLIC_KEY_ARG
WORKDIR /app
COPY frontend/package*.json ./
RUN npm ci
COPY frontend/ .
RUN npm run build
CMD ["npm","run","start"]
```

---

# 4. Sécurité

## 4.1 Chiffrement

- TLS 1.2/1.3 via Let’s Encrypt (Certbot + Nginx).
- Certificats montés en volumes Docker.

## 4.2 Firewalling (OPNsense)

- Autoriser 80/443 → Nginx (DMZ).
- Autoriser 8000 (API) uniquement depuis Nginx.
- Bloquer 3306 (MySQL) depuis l’extérieur.
- SSH (22) restreint au VPN client-to-site.

## 4.3 VPN

- **Site-to-Site** : IPSec AES-256 GCM.
- **Client-to-Site** : OpenVPN (SSL).

---

# 5. CI/CD & Déploiement

## 5.1 Pipeline GitLab CI/CD

1. Build images Docker (api, frontend)
2. Lint & tests unitaires

3. Push vers registry interne
4. Déploiement staging ( `docker-compose -f docker-compose.staging.yml up -d` )
5. Tests automatisés
6. Déploiement production (manuelle)

### 5.2 Stratégie de branches

- `main` → production
- `develop` → staging
- `feature/*` , `bugfix/*` → develop

---

## 6. Monitoring & Logs

### 6.1 Monitoring

- **Prometheus** : collecte métriques API, Nginx, DB.
- **Grafana** : tableaux de bord (CPU, RAM, latence, disponibilité).

### 6.2 Centralisation des logs

- **Filebeat** sur chaque conteneur.
- **Logstash** → **Elasticsearch** → **Kibana** (recherche, alerting).

---

## 7. Sauvegarde & Restauration

### 7.1 MySQL

- Dump quotidien (cron hôte) → NAS (iSCSI).
- Rétention 30 jours.

### 7.2 Volumes uploads & certificats

- Snapshots NFS quotidiens.

---

## 8. Haute Disponibilité & PRA

- OPNsense HA (failover automatique).
- Politique `restart: always` sur services critiques.
- Plan de reprise d’activité (< 4 h) en cas de sinistre.
- Évolution future : réplication MySQL master-slave.

# Annexes

- **A.** Schémas réseau (fichiers `schema_reseau_initial.pdf`)
- **B.** Cahier des charges (`cahier-des-charges.pdf`)
- **C.** Docker-Compose complet (`docker-compose.yml`)
- **D.** Script d'installation (`install.sh`)
- **E.** MCD / MLD base de données (`db_mcd.pdf`, `db_mld.pdf`)