# 1. Introduction

1.1 Objectif du DocumentExplique l’intention et la portée du document (ex. : informer les parties prenantes, servir de base à la maintenance, etc.).

1.2 Contexte et Portée du SystèmeDécris brièvement le système visé, son périmètre fonctionnel et son lien avec l’écosystème (clients, partenaires, réglementations, etc.).

1.3 Parties Prenantes et Audiences CiblesIdentifie qui lira le document et pourquoi (analystes, architectes, équipes de support, etc.).

1.4 Définitions, Acronymes et AbréviationsListe les termes spécifiques au domaine, les abréviations techniques, etc.

1.5 RéférencesDocuments externes ou internes sur lesquels s’appuie cette architecture (spécifications, manuels, normes, etc.).

1.6 Structure du DocumentExplique brièvement la logique d’organisation : comment lire et naviguer dans ce document.

# 2. Vue Logique

Audience principale : Analystes fonctionnels et architectes

## 2.1 Vue d’Ensemble du Domaine Métier

Brève présentation du domaine métier couvert par l’application.

Contexte métier, objectifs généraux.

## 2.2 Modèle Conceptuel

Entités Principales du Domaine : description des “objets” majeurs (Clients, Commandes, etc.).

Relations et Règles Métier : diagrammes de classes ou diagrammes entité-relation pour montrer comment les entités interagissent.

## 2.3 Décomposition en Modules/Packages Fonctionnels

Regroupements logiques (sous-systèmes, services, “bounded contexts” si DDD, etc.).

Responsabilités et interactions principales de chaque module.

## 2.4 Abstractions Clés et Leurs Interactions

Zoom sur les éléments clés de l’architecture fonctionnelle (ex. “contrôleurs”, “services métier”).

Diagrammes de séquence ou diagrammes de collaboration si besoin.

## 2.5 Patterns Architecturaux Appliqués

MVC, Hexagonal, Microservices, Event Sourcing, etc.

Raison d’être du choix de ces patterns.

## 2.6 Justification des Choix de Conception

Lien entre les exigences fonctionnelles et les éléments de conception.

Avantages, inconvénients et compromis retenus.

# 3. Vue de Processus

Audience principale : Intégrateurs système et architectes performance

## 3.1 Architecture de Processus

Comment le système s’exécute (processus, threads, conteneurs, microservices…).

Diagrammes d’activité ou d’interaction s’il y a lieu.

## 3.2 Communication Inter-Processus

Mécanismes de messaging, protocoles réseau, file d’attente, etc.

Gestion des flux asynchrones (si pertinent).

## 3.3 Gestion de la Concurrence

Stratégies de threading, pools de connexions, orchestration.

Sémaphores, verrous ou autres mécanismes de synchronisation.

## 3.4 Mécanismes de Synchronisation

Détail des techniques pour éviter la corruption de données ou les conflits (si nécessaire).

Coordination entre processus/services (transactions distribuées, compensations, etc.).

## 3.5 Considérations de Performance et Scalabilité

Stratégies de montée en charge (horizontal/vertical).

Goulots d’étranglement potentiels, mise en cache, load balancing.

# 4. Vue de Développement

Audience principale : Développeurs et équipe de build

## 4.1 Organisation du Code Source

Structure par projets, modules ou repositories.

Alignement entre structure de code et décomposition fonctionnelle (si pertinent).

## 4.2 Gestion des Dépendances

Outils (Maven, Gradle, NPM, etc.) et configuration.

Politique de versioning, librairies tierces critiques, convention de nommage ...

## 4.3 Structure des Builds et Intégration Continue

Processus de compilation, tests automatiques, pipelines CI/CD, etc.

Environnements de déploiement (test, staging, production).

## 4.4 Environnements de Développement

Principaux environnements requis : IDE, conteneurs Docker, bases de données locales, etc.

Configuration et contraintes spécifiques.

## 4.5 Standards et Conventions de Codage

Styles de code, patterns d’implémentation recommandés, linters.

Documentation du code et guidelines sur la qualité.

# 5. Vue Physique

Audience principale : Équipes d’exploitation et administrateurs système

## 5.1 Topologie du Système

Diagrammes d’infrastructure : serveurs, conteneurs, réseau, CDN, etc.

Différenciation des environnements (dev, test, prod).

## 5.2 Infrastructure Matérielle

Spécifications serveurs, virtualisation, cluster, région Cloud, etc.

Contraintes de localisation des données (RGPD, etc.) si pertinent.

## 5.3 Configuration Réseau

Segmentation, pare-feu, VPN, DMZ, etc.

Protocoles et ports utilisés, règles de routage.

## 5.4 Exigences Système

Ressources minimales (RAM, CPU, stockage), compatibilités OS, etc.

Dépendances externes (LDAP, Active Directory, etc.).

## 5.5 Stratégie de Déploiement

Processus de release, rollback etc.

Outils et automatisation (Ansible, Terraform, Kubernetes, etc.).

# 6. Scénarios (+1)

Audience principale : Toutes les parties prenantes, y compris utilisateurs finaux

## 6.1 Cas d’Utilisation Principaux

Description des user stories ou use cases majeurs.

Diagrammes d’activité ou de séquence pour illustrer les workflows critiques.

## 6.2 Scénarios de Qualité

Performance et Charge : scénarios de pic de trafic, batchs, etc.

Disponibilité et Résilience : cas de panne, tolérance aux fautes.

Sécurité : authentification, autorisation, protection des données, etc.

Maintenabilité : scénarios de mise à jour, migrations de schéma…

## 6.3 Validation de l’Architecture

Adéquation aux Besoins Métier : comment chaque besoin fonctionnel est couvert.

Conformité aux Exigences Non-Fonctionnelles : mesures de performance, SLA, etc.

# 7. Considérations Transversales

(Optionnel si déjà couvert dans les sections précédentes, mais souvent utile pour consolider)

## 7.1 Sécurité

Approches globales (IAM, encryption, RGPD, gestion des secrets, etc.).

Politique de patching, analyse de vulnérabilités.

## 7.2 Persistance des Données

Bases de données, choix de SGBD, stratégies de partitionnement ou réplication.

Considérations de backup/restore et archivage.

## 7.3 Observabilité (Logging et Monitoring)

Stratégie de logs (format, niveau, centralisation), métriques, dashboards.

Mécanismes d’alerting, gestion d’événements d’exploitation.

## 7.4 Gestion des Erreurs et Résilience

Politique de retries, circuit breaker, timeout, fallback.

Gestion des erreurs côté client, cas d’erreurs critiques.

# 8. Annexes

## 8.1 Journal des Décisions d’Architecture (ADR)

Liste chronologique ou thématique des décisions majeures.

Justification, alternatives écartées, date et approbateurs.

## 8.2 Alternatives Étudiées

Présentation succincte des solutions envisagées et pourquoi elles ont été retenues ou écartées.

## 8.3 Contraintes Techniques

Contraintes liées aux licences, réglementations, budgets, compatibilité legacy, etc.

## 8.4 Documentation Complémentaire

Diagrammes plus détaillés, documents contractuels, modes opératoires, etc.