Spécification des Conditions Requises pour l’Architecture

|  |  |
| --- | --- |
| *Nom du projet :* | Plateforme Geo-Aware Food Sourcing |
| *Préparé par :* | HEDI DHIB |
| *N° de version :* | 1.0 |
| *Date de version :* | 20/08/2025 |
| *Titre :* | Spécification des Conditions Requises pour l’Architecture |
| *Revu par :* | Vincent |
| *Date de révision :* | 21/08/2025 |
| *Liste de distribution :* | Architect, Dev team, CPO, CIO, CEO |
| De : | HEDI DHIB |
| Date : | 20/08/2025 |
| Email : | Hedi.dhib@gmail.com |
| Pour Action : | Révision |
| Date de rendu : | 21/8/2025 |
| Types d’action : | Approbation, Révision, Information, Classement, Action requise, Participation à une réunion, Autre (à spécifier) |
| Historique de versions : | Voir Github |

[1. Objectif du document 3](#_Toc206687867)

[2. Contexte général 3](#_Toc206687868)

[3. Sources des exigences 4](#_Toc206687869)

[4. Typologie des exigences 5](#_Toc206687870)

[5. Catalogue des exigences 6](#_Toc206687871)

[6. Priorisation 8](#_Toc206687872)

[7. Traçabilité et conformité 10](#_Toc206687873)

[8. Risques et hypothèses associés aux exigences 11](#_Toc206687874)

[Conclusion : 13](#_Toc206687875)

[Table de signatures : 14](#_Toc206687876)

## Objectif du document

Ce document constitue la **Spécification des Conditions Requises pour l’Architecture** du projet Foosus. Il a pour objectif de **formaliser l’ensemble des exigences** — métier, techniques, fonctionnelles, non-fonctionnelles, de sécurité, réglementaires ou liées à la transition — auxquelles l’architecture du système doit répondre.

Il sert de **référentiel partagé** entre les équipes produit, techniques et décisionnelles pour garantir que :

* l’architecture cible est **alignée sur les besoins métiers et utilisateurs**,
* les choix technologiques et structurants sont **justifiables et mesurables**,
* la transformation du SI respecte un **cadre de qualité, de conformité et d’évolutivité maîtrisé**.

**Ce document permet de :**

* **Supprimer toute ambiguïté** sur ce que l’architecture doit permettre ou garantir ;
* **Documenter formellement les exigences** liées à la refonte de l’architecture Foosus ;
* **Constituer une base de validation** des livrables techniques et fonctionnels ;
* **Assurer la traçabilité des décisions** prises en lien avec les besoins exprimés ;
* **Définir les critères de succès** en amont pour éviter les écarts d’interprétation lors du delivery.

## Contexte général

**Foosus** est une start-up engagée dans le domaine de l’**alimentation durable**, dont l’objectif est de rapprocher les consommateurs des **producteurs et artisans locaux** via une application mobile. Forte de trois années d’existence, l’entreprise entre dans une phase critique de **scalabilité, de structuration et d’industrialisation** de sa plateforme.

Cependant, les **choix historiques d’architecture**, orientés vers un développement rapide, ont généré une **dette technique importante**, un **monolithe difficile à faire évoluer** et une **manque de modularité** qui freinent aujourd’hui :

* l’expérimentation rapide de nouvelles fonctionnalités ;
* l’intégration de services tiers ;
* la robustesse face aux pics de charge induits par la croissance.

Dans ce contexte, l’entreprise a décidé d’engager un **chantier de refonte de son architecture** basé sur les principes suivants :

* **Modularité** : découpage en domaines métier, adoption de microservices.
* **Scalabilité et performance** : architecture cloud-native, tolérante aux pics.
* **Interopérabilité** : APIs standardisées, communication entre modules via événements.
* **Gouvernance** : traçabilité des décisions, documentation continue, standards clairs.
* **Conformité** : alignement avec le RGPD, l’OWASP, et les bonnes pratiques de sécurité.

Cette Spécification des Conditions Requises pour l’Architecture s’inscrit donc dans le cadre du **cycle ADM TOGAF**, en réponse à la phase **B (Architecture Business)** et comme **point d’entrée structurant des phases C et D** (Architecture des Systèmes d’Information et Architecture Technologique).

Elle a vocation à **servir de fondation** pour tous les livrables architecturaux, ainsi qu’à assurer la **cohérence entre les exigences exprimées et les solutions mises en œuvre**.

## Sources des exigences

Les exigences définies dans ce document proviennent de **plusieurs sources hétérogènes**, qui reflètent les attentes, contraintes et ambitions des différents acteurs impliqués dans la transformation de la plateforme Foosus.

Elles ont été recueillies via des **entretiens**, **ateliers d’architecture**, **analyses techniques**, ainsi que l’étude des **objectifs métier** et des **contraintes réglementaires**.

**Typologie des sources :**

| **Source** | **Origine** | **Description** |
| --- | --- | --- |
| **Exigences métier** | CPO, PO, Product Designers | Attentes fonctionnelles exprimées par les équipes produit pour répondre aux besoins des utilisateurs finaux (parcours utilisateur, géolocalisation, tunnel de commande, etc.) |
| **Exigences techniques** | Architecte, Lead Dev, DevOps | Nécessités liées à la modernisation du SI, à la résilience, à la scalabilité, au découplage technique, à l’intégration de composants réutilisables |
| **Exigences de sécurité** | Responsable Sécurité / DPO | Conformité RGPD, gestion des droits, chiffrement, auditabilité, principe de moindre privilège |
| **Contraintes réglementaires** | Juridique / DPO | Obligations légales (protection des données personnelles, consentement utilisateur, traçabilité) |
| **Exigences d’exploitation** | Équipe Ops / Support | Observabilité, logs, alerting, facilité de déploiement et de rollback, SLOs |
| **Exigences de transformation** | Direction, CTO, Architecte | Nécessité de migrer hors du legacy, assurer la cohabitation temporaire, cadrer la dette technique, fournir une architecture évolutive |
| **Retour utilisateurs / terrain** | Support / Tests / Feedback terrain | Frictions identifiées dans les parcours actuels, lenteurs, bugs récurrents, manque de fluidité UX |
| **Best practices sectorielles** | TOGAF, OWASP, OpenAPI, DDD, 12-Factor App | Intégration des standards de qualité, d’interopérabilité, de sécurité, de résilience et de documentation moderne |

Les exigences issues de ces sources sont **consolidées, normalisées et classifiées** dans les sections suivantes, afin de garantir leur traçabilité tout au long du cycle de vie du projet d’architecture.

## Typologie des exigences

Les exigences recueillies dans le cadre du projet Foosus sont **classifiées selon leur nature et leur rôle dans l’architecture cible**. Cette typologie facilite leur compréhension, leur priorisation, leur affectation aux livrables, ainsi que leur traçabilité dans les phases de conception, de mise en œuvre et de validation.

**Catégories d’exigences :**

| **Type** | **Description** | **Exemples pour Foosus** |
| --- | --- | --- |
| **Fonctionnelles** | **Décrivent les capacités métier attendues de la plateforme. Elles spécifient "ce que le système doit faire".** | **- Permettre aux utilisateurs de trouver des producteurs dans un rayon donné - Créer une commande multi-producteurs - Suivre les livraisons en temps réel** |
| **Non-fonctionnelles** | **Définissent les qualités attendues du système, indépendamment des fonctions spécifiques.** | **- Temps de réponse < 1,5 sec - Disponibilité 99.9 % - Tolerance aux pannes - Intégration CI/CD automatisée** |
| **Techniques** | **Imposent des contraintes de design ou d’implémentation propres à l’environnement technique.** | **- Utilisation d’une architecture microservices - Format standard des APIs (OpenAPI 3.x) - Logging centralisé** |
| **De sécurité et conformité** | **Spécifient les règles de confidentialité, d’intégrité, de traçabilité, ou de conformité réglementaire.** | **- RGPD : droit à l’oubli, gestion du consentement - SSO obligatoire pour l’admin - Chiffrement des données sensibles** |
| **D’exploitabilité** | **Exigences liées à la maintenance, l’observabilité, la supervision et l’automatisation.** | **- Alertes en cas de latence > 2s - Possibilité de rollback automatisé - Logs consultables en temps réel** |
| **De transition** | **Exigences liées à la cohabitation temporaire avec le legacy, à la migration, ou aux MVPs intermédiaires.** | **- Maintien du système existant pendant 6 mois - Interfaces de synchronisation temporaires - Priorisation de certains domaines métier** |
| **De gouvernance** | **Exigences sur la traçabilité des décisions, la documentation, la gestion des standards et de la qualité.** | **- Chaque composant structurant doit être associé à un ADR validé - Documentation minimale obligatoire avant mise en production** |

Cette typologie servira de base de classement pour le catalogue des exigences à venir (section 5), ainsi que pour leur traçabilité vers les artefacts d’architecture.

## Catalogue des exigences

Le tableau ci-dessous recense les exigences **prioritaires, critiques ou structurantes** identifiées pour la conception, la mise en œuvre et la validation de l’architecture cible.  
Chaque exigence est **classée, priorisée et liée à sa source et à un livrable vérifiable**.

| **ID** | **Type** | **Exigence** | **Priorité (MoSCoW)** | **Source** | **Livrable de vérification** |
| --- | --- | --- | --- | --- | --- |
| REQ-01 | Fonctionnelle | Permettre à un utilisateur de localiser des producteurs dans un rayon de 10 km | MUST | Product Owner / UX | Modèle de processus + maquette fonctionnelle |
| REQ-02 | Non-fonctionnelle | Le temps de réponse sur mobile ne doit pas dépasser 1,5 seconde sous charge nominale | MUST | Dev Team / Performance | Tests de charge + métriques de performance |
| REQ-03 | Sécurité / Conformité | Le système doit respecter le RGPD (droit à l’oubli, consentement explicite, accès aux données) | MUST | DPO / Juridique | Modèle de gouvernance des données + test de conformité |
| REQ-04 | Technique | L’architecture applicative doit être basée sur des microservices découplés par domaine métier | MUST | Architecte / CTO | Diagrammes C4 + découpage DDD |
| REQ-05 | Transition | Le legacy doit coexister avec la nouvelle architecture pendant 6 mois minimum | SHOULD | CTO | Plan de migration + interfaces d’interopérabilité |
| REQ-06 | Exploitabilité | Les logs doivent être centralisés, accessibles en temps réel et liés aux IDs utilisateur | MUST | Support / Ops | Stack ELK / Grafana + dashboard live |
| REQ-07 | Gouvernance | Chaque décision structurante doit faire l’objet d’un ADR approuvé et historisé | MUST | Architecte | Registre ADR dans GitHub (docs/adr) |
| REQ-08 | Fonctionnelle | Le système doit gérer des commandes multi-producteurs et un panier partagé | MUST | Métier / UX | Modèle fonctionnel + cas d’usage + test de recette |
| REQ-09 | Non-fonctionnelle | Le système doit garantir une disponibilité de 99,9 % (hors maintenance planifiée) | SHOULD | SLA produit | Architecture redondante + monitoring uptime |
| REQ-10 | API / Interopérabilité | Toutes les APIs doivent être documentées en OpenAPI 3.x et versionnées | MUST | Tech Lead | Documentation auto via Swagger + CI build check |

**Remarques :**

* Le **niveau de priorité** suit le modèle **MoSCoW** (*Must / Should / Could / Won’t*).
* Chaque exigence est **liée à un artefact vérifiable**, ce qui permet de **contrôler la couverture réelle** des exigences.
* Ce tableau pourra être enrichi dans une **feuille de calcul ou outil de gestion des exigences (Notion, Excel, GitHub Projects)**.

## Priorisation

La priorisation des exigences repose sur une méthode structurée permettant de :

* **Aligner les efforts d’architecture avec la valeur métier**,
* **Garantir la faisabilité technique dans un contexte contraint** (temps, ressources, dette legacy),
* **Assurer l’acceptabilité des livrables par les parties prenantes**.

**Méthodologie de priorisation utilisée :** **MoSCoW**

| **Code** | **Signification** | **Règle d’application** |
| --- | --- | --- |
| **MUST** | Indispensable pour la viabilité du système | Exigence critique — toute livraison sans cette exigence serait considérée comme incomplète ou invalide |
| **SHOULD** | Importante, mais contournable temporairement | Peut être livrée dans une itération suivante si justifiée (retard, dépendance, budget) |
| **COULD** | Facultative / opportunité | Apporte une valeur ajoutée mais non critique — à considérer si les ressources le permettent |
| **WON’T (this time)** | Explicitement exclue | Décision validée de ne pas inclure dans ce cycle — à reconsidérer dans le futur |

**Critères de priorisation appliqués à Foosus**

| **Critère** | **Question posée** | **Impact sur la priorité** |
| --- | --- | --- |
| **Impact métier** | **Est-ce essentiel à l’utilisateur final ou à la promesse produit ?** | **Influence fort le MUST** |
| **Risque technique** | **Le non-traitement bloque-t-il l’évolution ou crée-t-il de la dette ?** | **Peut faire basculer en MUST ou SHOULD** |
| **Effort estimé** | **Peut-on le livrer dans un MVP réaliste ?** | **Oriente vers SHOULD ou COULD** |
| **Valeur produit** | **Est-ce différenciant ou secondaire ?** | **Oriente entre SHOULD et COULD** |
| **Contraintes légales ou de sécurité** | **Imposée par la loi ou des standards ?** | **Est forcément un MUST** |

**Suivi et évolution :**

* **Le niveau de priorité est versionné dans le catalogue des exigences.**
* **Tout changement de priorité (ex : SHOULD → MUST) doit être :**
  + **documenté dans l’ADR ou le backlog,**
  + **justifié par une évolution métier, légale ou technique,**
  + **validé en comité d’architecture.**

## Traçabilité et conformité

La traçabilité des exigences est un pilier de la gouvernance architecturale. Elle permet de **relier chaque exigence à un artefact, une décision, un composant ou un test**, assurant ainsi une couverture complète et vérifiable de ce qui a été défini dans cette spécification.

**Objectifs de la traçabilité :**

* **S’assurer qu’aucune exigence critique ne soit oubliée** lors de la conception ou du développement.
* **Permettre un audit clair** de la conformité des livrables (architecture, code, déploiement).
* **Faciliter la validation et les tests** fonctionnels et techniques.
* **Justifier les choix réalisés** (ex. : pourquoi un découpage en microservices, pourquoi tel outil ou tel protocole).
* **Adapter rapidement l’architecture** en cas de changement d’exigence ou de contexte.

**Traçabilité descendante (exigence → livrable) :**

Chaque exigence du catalogue (section 5) est associée à au moins un des éléments suivants :

* Un **artefact architectural** (diagramme C4, schéma DDD, modélisation de processus, etc.)
* Un **ADR (Architecture Decision Record)** justifiant un choix technique
* Un **livrable technique** (documentation OpenAPI, script CI/CD, log d’audit…)
* Un **test d’acceptation** ou un critère de validation fonctionnelle

🡺 Ces liens sont **versionnés dans Git** et/ou référencés dans **Notion ou Confluence** via des IDs croisés (ex : REQ-04 ←→ ADR-03).

**Traçabilité ascendante (livrable → exigences) :**

Chaque composant ou décision peut être relié à une ou plusieurs exigences initiales, ce qui permet :

* De **justifier l’existence d’un composant ou d’un standard**,
* De **vérifier sa cohérence avec les objectifs métier ou réglementaires**,
* D’**éviter l’introduction de solutions non justifiées ou non prioritaires**.

**Outils utilisés pour la traçabilité**

| **Outil** | **Usage** |
| --- | --- |
| **GitHub / GitLab** | Stockage des ADRs, des livrables versionnés, tagging des issues / commits |
| **Notion / Confluence** | Référentiel centralisé des exigences et des liens vers les artefacts |
| **CI/CD (GitHub Actions, GitLab CI)** | Vérification automatique de la couverture documentaire (lint OpenAPI, présence README/ADR) |
| **Trello / Jira / Backlog agile** | Suivi des tickets liés aux exigences et mise en production |

**Conformité :**

* Les **audits de conformité** sont menés par le Comité d’Architecture à chaque jalon (fin de vague ou MVP).
* Les exigences légales et de sécurité (RGPD, logs, access control…) sont **revues par le DPO ou l’équipe sécurité**.
* Tout écart identifié (non-conformité, exigence non couverte) fait l’objet d’un **ticket correctif** ou d’une **demande de modification du périmètre**.

*"Chaque exigence priorisée dans ce document fait l’objet d’un ticket ou d’un epic dans le backlog agile du projet (Jira/Trello). Les tickets sont étiquetés par leur ID (ex. : REQ-01) afin d’assurer une traçabilité entre les spécifications d’architecture et les user stories de mise en œuvre."*  
*"Ce lien permet un pilotage conjoint architecture / produit, aligné sur les itérations de delivery."*

## Risques et hypothèses associés aux exigences

Certaines exigences peuvent comporter des **zones d’incertitude**, soit en raison de dépendances externes, soit en lien avec des choix techniques non encore validés. Il est donc essentiel de :

* **formuler explicitement les hypothèses de travail** liées à ces exigences,
* **identifier les risques** associés à leur mauvaise interprétation, non-couverture ou retard de mise en œuvre,
* **prévoir des actions de réduction ou de mitigation**.

**Hypothèses critiques :**

| **ID** | **Hypothèse** | **Impact si fausse** | **Action recommandée / Validation** |
| --- | --- | --- | --- |
| HYP-01 | Les utilisateurs finaux accepteront de partager leur géolocalisation | Risque de perte de fonctionnalité clé (REQ-01) | Tester via un A/B test, inclure des options de désactivation |
| HYP-02 | Les équipes DevOps sont disponibles pour mettre en place la stack CI/CD attendue | Délai de mise en œuvre des exigences REQ-06 et REQ-09 | Réserver des ressources dès le sprint 0 |
| HYP-03 | Le legacy est maintenable pendant toute la phase de transition (6 mois) | Risque si obsolescence technique non anticipée (REQ-05) | Audit technique du legacy et plan de sauvegarde |
| HYP-04 | Le format OpenAPI est accepté et maîtrisé par toutes les équipes | Risque de retard ou de mauvaise documentation (REQ-10) | Organiser un tech talk de montée en compétence |

**Risques associés aux exigences**

| **ID** | **Risque** | **Exigences concernées** | **Gravité / Probabilité** | **Plan d’action** |
| --- | --- | --- | --- | --- |
| RISK-01 | RGPD mal implémenté (consentement, suppression, anonymisation) | REQ-03 | Haute / Moyenne | Intégrer le DPO dans la validation des modèles de données |
| RISK-02 | Logs mal centralisés ou insuffisamment détaillés | REQ-06 | Moyenne / Élevée | Vérifier les logs sur 2 sprints consécutifs avec l'équipe support |
| RISK-03 | Écart entre architecture documentée et implémentée | REQ-04, REQ-07 | Haute / Moyenne | Contrôle qualité via revue d’ADR + relecture des schémas à chaque sprint |
| RISK-04 | Multiplication des microservices non justifiée | REQ-04 | Moyenne / Moyenne | Créer un ADR pour chaque composant structurant, validé en ARB |

**Suivi et gestion :**

* Ces risques et hypothèses sont **revus à chaque révision du catalogue des exigences**.
* Ils sont suivis dans le **registre des risques du projet**.
* Toute hypothèse invalidée doit donner lieu à **une mise à jour du périmètre, des priorités ou des tests de validation**.

## Conclusion :

*Ce document correspond aux livrables attendus dans les phases TOGAF suivantes :*  
– **Phase B (Architecture Métier)** : formalisation des besoins fonctionnels et réglementaires ;  
– **Phase C (Architecture des SI)** : exigences techniques, de découplage, interopérabilité ;  
– **Phase D (Architecture Technologique)** : exigences de performance, observabilité, CI/CD ;  
– **Phase E (Opportunités et Solutions)** : cadrage de la migration, MVPs, cohabitation legacy.

## Table de signatures :

| **Nom** | **Fonction** | **Organisation** | **Signature** | **Date** |
| --- | --- | --- | --- | --- |
| Hedi Dhib | Architecte Logiciel | Foosus |  | 19/08/2025 |
| Ash Callum | Chief Executive Officer (CEO) | Foosus |  |  |
| Natasha Jarson | Chief Information Officer (CIO) | Foosus |  |  |
| Daniel Anthony | Chief Product Officer (CPO) | Foosus |  |  |
| Christina Orgega | Chief Marketing Officer (CMO) | Foosus |  |  |
| Jo Kumar | Chief Financial Officer (CFO) | Foosus |  |  |
| Pete Parker | Engineering Owner | Foosus |  |  |
| Jack Harkner | Operations Lead | Foosus |  |  |

**Annexe A – Couverture des exigences par artefacts :**

| **ID Exigence** | **Type** | **Artefact associé** | **Emplacement / Lien (exemple)** |
| --- | --- | --- | --- |
| **REQ-01** | **Fonctionnelle** | **Maquette UX + diagramme de processus** | **/docs/UX/prod-localisation.png** |
| **REQ-02** | **Non-fonctionnelle** | **Rapport JMeter + KPI de charge** | **/tests/perf/loadtest-report-Q2.md** |
| **REQ-03** | **Sécurité / conformité** | **Politique RGPD + logs anonymisés** | **/compliance/RGPD-foosus-v1.pdf** |
| **REQ-04** | **Technique** | **Diagramme C4 + ADR-04 (architecture microservices)** | **/archi/C4-foosus.png, /docs/adr/adr-004.md** |
| **REQ-05** | **Transition** | **Roadmap de migration + plan de coexistence** | **/docs/legacy-plan.md** |
| **REQ-06** | **Exploitabilité** | **Stack ELK déployée + dashboard live** | **/infra/observability/kibana.json** |
| **REQ-07** | **Gouvernance** | **Registre des décisions (ADRs GitHub)** | **/docs/adr/index.md** |

**Glossaire des acronymes utilisés :**

| **Acronyme** | **Définition** |
| --- | --- |
| **REQ** | **Requirement (Exigence)** |
| **ADR** | **Architecture Decision Record** |
| **CI/CD** | **Continuous Integration / Continuous Delivery** |
| **RGPD** | **Règlement Général sur la Protection des Données** |
| **C4** | **Modèle de modélisation logicielle en 4 niveaux** |
| **DDD** | **Domain-Driven Design** |
| **SLA** | **Service Level Agreement** |
| **SLO** | **Service Level Objective** |