**Contrat d’Architecture avec les Fonctions Développement et Design**

|  |  |
| --- | --- |
| *Nom du projet :* | Plateforme Geo-Aware Food Sourcing |
| *Préparé par :* | HEDI DHIB |
| *N° de version :* | 1.0 |
| *Date de version :* | 20/08/2025 |
| *Titre :* | Contrat d’Architecture avec les Fonctions Développement et Design |
| *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. Objet de ce document 2](#_Toc1205615528)

[2. Contexte et portée de l’accord 4](#_Toc14479542)

[3. Nature de l’engagement mutuel 5](#_Toc1473208791)

[4. Objectifs de l’architecture et parties prenantes 7](#_Toc1165452809)

[a) Objectifs stratégiques 7](#_Toc797223880)

[b) Objectifs métier (regroupés) 8](#_Toc69126896)

[c) Parties prenantes, visions et responsabilités 8](#_Toc1288732294)

[d) Tableau récapitulatif des responsabilités par rôle 9](#_Toc588401846)

[5. Description de l’architecture, principes stratégiques et conditions requises 10](#_Toc495282256)

[a) Description de l’architecture cible 10](#_Toc1417518565)

[b) Principes stratégiques d’implémentation 11](#_Toc1282485058)

[c) Conditions de conformité et contraintes réglementaires 12](#_Toc1806232395)

[6. Livrables d’architecture et indicateurs associés 13](#_Toc1597142615)

[a) Liste des artefacts attendus (C4, ADR, APIs, doc, etc.) 13](#_Toc1809726960)

[b) Liens vers les exigences & métriques (business + tech) 13](#_Toc1370309208)

[c) Modalités de validation et d’acceptation des livrables 14](#_Toc2056449008)

[d) Cycle de vie des artefacts 14](#_Toc1020569768)

[7. Plan de travail commun priorisé 15](#_Toc902491060)

[8. Plan de communication et de synchronisation 16](#_Toc1884602355)

[a) Évènements de coordination (points techniques, démos, revues) 17](#_Toc2038237127)

[b) Canaux et formats (Slack, Git, Notion, Miro, etc.) 17](#_Toc259406103)

[c) Fréquence et rythme (hebdo, sprint, jalons…) 18](#_Toc721166598)

[9. Gouvernance, risques et arbitrage 18](#_Toc1870951045)

[a) Structure de gouvernance (comités, règles, outils) 19](#_Toc1270642437)

[b) Analyse des risques techniques & organisationnels 19](#_Toc125804923)

[c) Procédures d’escalade 20](#_Toc1373131474)

[10. Hypothèses 21](#_Toc1542408547)

[11. Critères d’acceptation et validation 23](#_Toc1874395970)

[a) KPIs de l’architecture cible 23](#_Toc1745695178)

[b) KPIs de livraison / valeur métier 23](#_Toc717610395)

[c) Processus de validation mutuelle 24](#_Toc290628239)

[12. Phasage, jalons et calendrier 24](#_Toc10819475)

[a) Conditions requises pour la conformité 24](#_Toc1006292735)

[b) Développement et propriété de l’architecture 25](#_Toc2002097083)

[c) Jalonisation prévisionnelle 25](#_Toc1780860296)

[13. Signataires et approbation 26](#_Toc184813967)

## Objet de ce document

Le présent document constitue le **Contrat d’Architecture entre l’équipe d’architecture de Foosus et les fonctions techniques** impliquées dans la conception, le développement, la qualité et l’opérationnalisation de l’architecture cible.

Il s’agit d’un **accord formel et mutuel** entre les **équipes de développement, de design, de DevOps** et les **sponsors de l’architecture**, portant sur :

* les **livrables architecturaux** attendus,
* la **qualité des implémentations**,
* et la **conformité aux objectifs stratégiques et techniques** de l’architecture cible.

Ce contrat vise à garantir, à travers une gouvernance rigoureuse :

* Un **système de contrôle continu** assurant l’intégrité, la traçabilité, l’évaluation des changements, et l’audit des décisions architecturales (cf. TOGAF Partie VII – Gouvernance) ;
* L’**adhésion explicite aux principes d’architecture**, aux standards définis (OpenAPI, ADR, CI/CD…) et aux conditions requises (scalabilité, sécurité, conformité) ;
* L’**identification et la maîtrise des risques** techniques, organisationnels et opérationnels tout au long du cycle de développement ;
* La mise en œuvre de **pratiques de collaboration transparentes, responsables et disciplinées**, notamment à travers le partage d’artefacts, de processus et d’outils ;
* Un **accord sur l’organe de gouvernance responsable** de ce contrat, son niveau d’autorité et le périmètre couvert.

Ce document constitue une **déclaration d’intention opérationnelle**, signée par les parties prenantes internes ou externes (intégrateurs, partenaires techniques, fournisseurs) impliquées dans la livraison de l’architecture cible — ou d’un de ses domaines (applicatif, technologique, données…).

Il garantit que **chaque partenaire contribue activement, durablement et en cohérence** avec la vision globale de l’architecture d’entreprise de Foosus.

Ce document s’appuie sur les normes TOGAF (Architecture Governance, Content Framework) et les bonnes pratiques du Software Craftsmanship. Un glossaire des acronymes est disponible en annexe.

## Contexte et portée de l’accord

**Contexte :**

Foosus est en cours de transformation technique et produit, visant à moderniser sa plateforme pour mieux répondre :

* aux **exigences métier d’accessibilité, de performance et de croissance**,
* aux **objectifs stratégiques d’innovation, de scalabilité et de modularité**,
* ainsi qu’à ses **engagements réglementaires** en matière de sécurité et de données (ex. RGPD).

Dans ce cadre, une **architecture cible** a été définie (phases B → D du cycle TOGAF ADM), incluant des principes directeurs, des exigences structurantes, et une trajectoire de mise en œuvre.

Le présent contrat d’architecture se situe dans la phase de **réalisation (Phase E/F)**, avec pour but de **formaliser l’alignement entre l’architecture définie et sa déclinaison technique concrète**, en coopération étroite avec les fonctions **développement, design, DevOps, qualité, et intégration**.

**Portée de l’accord :**

Ce contrat couvre l’ensemble des **activités techniques contribuant à la réalisation et à l’implémentation des composants architecturaux**, notamment :

* Le **développement des composants logiciels** alignés sur l’architecture de référence (microservices, APIs, découplage, etc.) ;
* La **mise en œuvre des décisions d’architecture** (ADRs), leur traçabilité et leur respect dans les projets ;
* L’**adoption des principes d’architecture** dans les choix de conception technique, d’UI/UX, de sécurité et d’exploitabilité ;
* L’**intégration dans les chaînes CI/CD**, la documentation technique et la stratégie de déploiement ;
* La **conformité avec les conditions requises** définies dans la Spécification d’Architecture (performance, testabilité, sécurité, observabilité).

Ce contrat s’applique aux **domaines suivants de l’Architecture d’Entreprise**, selon le cadre TOGAF :

| **Domaine d’architecture** | **Inclus dans ce contrat** |
| --- | --- |
| Architecture Métier | Non concernée |
| Architecture Applicative | Oui |
| Architecture des Données | Partiellement (modèles d’échange, contrats d’API) |
| Architecture Technologique | Oui (infra cible, CI/CD, observabilité) |

**Limites de la portée :**

Ce contrat **n’inclut pas** :

* la gouvernance budgétaire ou contractuelle avec les partenaires externes (hors périmètre),
* la transformation des processus métier (gouvernée ailleurs),
* les domaines réglementaires ne relevant pas directement du technique (ex. audit DPO, juridique).

## Nature de l’engagement mutuel

Ce contrat d’architecture formalise un **engagement réciproque** entre :

* les équipes **d’architecture d’entreprise** de Foosus (pilotage stratégique, gouvernance, conformité),
* et les fonctions **développement, design, DevOps et QA**, en charge de l’implémentation opérationnelle de l’architecture cible.

Cet engagement porte à la fois sur :

* la **réalisation technique conforme** à l’architecture définie,
* le **respect des principes directeurs, des décisions documentées**, et des exigences validées,
* une **coopération continue** tout au long du cycle de delivery (Build / Run),
* l’**intégration des artefacts d’architecture dans les produits livrés**,
* et la **traçabilité et auditabilité** des décisions et écarts justifiés.

**Engagements de l’équipe d’architecture :**

* Fournir une **vision claire et documentée** de l’architecture cible (diagrammes, standards, ADRs) ;
* Participer aux **revues techniques et validations** en continu (ex : Sprint Reviews, PR, merge requests) ;
* Assurer un **soutien continu** aux fonctions techniques pour interpréter, adapter ou ajuster les principes d’architecture ;
* Garantir la **mise à jour des artefacts** (référentiel de modèles, décisions, schémas, API, documentation) ;
* **Valider formellement** la conformité des livrables aux attentes d’architecture.

**Engagements des fonctions Développement & Design :**

* Implémenter les **composants techniques, interfaces, services et architectures logicielles** dans le respect des principes et contraintes définis ;
* Participer activement aux **revues d’architecture, relectures d’ADRs, et comités de gouvernance** ;
* Documenter les **décisions locales ou adaptations nécessaires** (justification, compatibilité, impact) ;
* Intégrer les **métriques d’évaluation** (performance, testabilité, sécurité, observabilité) dès la conception ;
* **Collaborer proactivement** en cas de blocage, ambiguïté ou besoin d’arbitrage.

**Cadre de gouvernance appliqué :**

Ce contrat est exécuté sous la supervision d’un **organe de gouvernance dédié à l’architecture d’exécution**, chargé de :

* suivre l’**avancement des livrables**,
* valider les **écarts justifiés ou évolutions**,
* arbitrer les **conflits d’interprétation ou d’implémentation**,
* et **actualiser les documents contractuels** en cas d’évolution du contexte ou des priorités.

è Le respect de ce contrat conditionne la **validation des versions majeures** des livrables produits et leur passage en production.

## Objectifs de l’architecture et parties prenantes

Cette section précise les **objectifs principaux** que l’architecture doit atteindre, ainsi que les **acteurs clés** qui contribuent à leur réalisation, directement ou indirectement. L’objectif est de garantir un **alignement stratégique, technique et opérationnel**, à tous les niveaux de delivery.

### Objectifs stratégiques

L’architecture cible vise à soutenir la vision produit de Foosus à travers les objectifs suivants :

| **Objectif** | **Description** |
| --- | --- |
| **Modularité** | Permettre le découplage des domaines métier pour un développement et un déploiement indépendants. |
| **Scalabilité** | Permettre la montée en charge et la gestion des pics d’activité via des composants autonomes. |
| **Observabilité** | Offrir une visibilité sur l’état de la plateforme pour faciliter l’exploitation, le support, et la résolution d’incidents. |
| **Conformité & Sécurité** | Intégrer les exigences RGPD, OWASP et les principes de “privacy by design” et “security by default”. |
| **Time-to-market** | Réduire les cycles de livraison par une architecture pensée pour le delivery continu (CI/CD). |

### Objectifs métier (regroupés)

| **Objectif métier** | **Traduction architecturale** |
| --- | --- |
| Mise en relation locale | APIs géociblées, recherche par proximité |
| Expérimentation produit | Feature toggles, dark releases, architecture découplée |
| Personnalisation de l’offre | APIs frontales adaptables, logique métier configurable |
| Résilience de la plateforme | Tolérance aux pannes, découplage synchro / asynchro, retry logic |
| Exploitabilité et pilotage | Centralisation des logs, métriques, alertes, dashboards |

### Parties prenantes, visions et responsabilités

| **Rôle** | **Vision / Préoccupation** | **Responsabilité vis-à-vis de l’architecture** |
| --- | --- | --- |
| **Tech Lead / Dev Lead** | Clarté des composants, réutilisabilité, simplicité | Traduction des concepts d’architecture en tâches techniques et en bonnes pratiques |
| **UX/UI Designer** | Cohérence interface / service, réactivité, accessibilité | Adaptation de la conception visuelle aux contraintes de l’architecture (latence, découplage, etc.) |
| **Développeurs** | Débogage, performance, documentation | Respect des normes, implémentation conforme, documentation du code |
| **QA / Test** | Automatisation, testabilité, robustesse | Intégration des exigences d’architecture dans les pipelines de test |
| **DevOps / SRE** | Disponibilité, logs, monitoring, rollback | Mise en œuvre des SLOs, des dashboards, des mécanismes de déploiement sécurisé |
| **Architecte** | Alignement global, gouvernance, vision long terme | Animation du contrat, validation des livrables, documentation des évolutions |

L’identification de ces parties prenantes permet de garantir que **chaque objectif a un acteur responsable**, impliqué dans son exécution et son évaluation.

### Tableau récapitulatif des responsabilités par rôle

Ce tableau permet de **visualiser les attentes, les responsabilités et les points de contribution** de chaque fonction impliquée dans l'exécution du contrat d’architecture. Il s’inscrit dans une logique de **gouvernance partagée et opérationnelle**, comme recommandée par TOGAF.

| **Fonction** | **Responsabilités clés** | **Livrables attendus** | **Devoirs vis-à-vis de l’architecture** |
| --- | --- | --- | --- |
| **Développeur Backend** | Implémenter les microservices, APIs, logique métier | Code, Specs OpenAPI, Tests, ADRs | Respect des specs, tests, documentation, sécurité |
| **Développeur Frontend** | Consommer les APIs, intégrer UI, respect du design system | Interfaces connectées, documentation d’usage | Suivi des contrats API, collaboration avec UI/UX |
| **Tech Lead / Lead Dev** | Encadrer les devs, relire les PRs, assurer la qualité technique | Revue de code, validation des specs, mise à jour ADR | Garant de la conformité technique locale |
| **Architecte** | Définir, diffuser et faire respecter les principes d’architecture | ADRs, modèles C4, guides, validations | Animation, gouvernance, arbitrage, documentation |
| **PO / CPO** | Prioriser les besoins, valider la valeur métier | Backlog, critères d’acceptation, décisions stratégiques | Collaboration sur périmètre, arbitrage sur valeur |
| **QA / Test Lead** | Vérifier la conformité des livrables aux specs et aux tests | Scénarios de test, rapports qualité | Tester les composants selon les exigences architecture |
| **DevOps / SRE** | Gérer l’infra, CI/CD, observabilité, sécurité en prod | Pipelines, dashboards, logs, alertes | Monitoring des KPIs, conformité aux SLOs/SLA |
| **UX / UI Designer** | Concevoir les parcours utilisateurs et maquettes alignées | Design system, maquettes, specs UI | Coordination avec contraintes techniques et performance |

Ce tableau peut être converti en **matrice RACI** ou affiché lors des **onboardings projet** pour rappeler à chacun son rôle vis-à-vis de l’architecture.

## Description de l’architecture, principes stratégiques et conditions requises

Cette section décrit l’**architecture cible**, les **principes stratégiques qui guident son exécution**, ainsi que les **contraintes réglementaires et techniques** auxquelles toutes les fonctions doivent se conformer.

### Description de l’architecture cible

L’architecture cible de Foosus repose sur une approche **modulaire, découplée et scalable**, reposant sur les composants suivants :

* **Architecture microservices** par domaine fonctionnel, exposés via des **APIs RESTful documentées en OpenAPI 3.0**.
* **Découplage front-end / back-end**, avec intégration via **API Gateway** et contrat explicite (contract-first).
* **Système d’événementiel asynchrone** via une **architecture orientée événements (EDA)**, pour les flux non critiques.
* **Infrastructure cloud managée**, orchestrée via des outils de type Kubernetes/Docker (selon contexte).
* **Chaîne CI/CD complète**, avec linting, tests automatiques, validation d’ADR, versioning sémantique et déploiement sans interruption.

è L’architecture est conforme au **modèle C4 (niveau context → container → component)**, versionnée dans un référentiel Git centralisé.

### Principes stratégiques d’implémentation

Les principes ci-dessous sont **obligatoires** pour toutes les équipes de développement et design :

| **ID** | **Principe** | **Impact attendu** |
| --- | --- | --- |
| PA-01 | **KISS** (Keep It Simple & Stupid) | Code et architecture lisibles, faciles à faire évoluer |
| PA-02 | **Modularité & réutilisabilité** | Composants autonomes et adaptables, réduisant les duplications |
| PA-03 | **API contractuelles** | Spécifications validées avant développement, intégration fluide |
| PA-04 | **Documentation systématique (ADR, OpenAPI, C4)** | Traçabilité, onboarding facilité, auditabilité |
| PA-05 | **Observabilité dès la conception** | Détection proactive des incidents, pilotage produit |
| PA-06 | **Privacy & Security by Design** | RGPD intégré, sécurité au plus tôt dans les décisions techniques |
| PA-07 | **Automatisation des processus** | CI/CD, tests, validations = gain de temps, réduction du risque humain |

Toutes les équipes sont invitées à se référer aux standards formels listés en annexe 1 : OWASP Top 10, spécification OpenAPI v3, Clean Architecture, modèle C4, et pratiques CI/CD issues de GitLab/GitHub Actions.

### Conditions de conformité et contraintes réglementaires

Toutes les livraisons techniques doivent respecter les normes suivantes :

| **Type de contrainte** | **Référence** | **Exigence** |
| --- | --- | --- |
| **Réglementaire** | RGPD / CNIL | Gestion des consentements, anonymisation, droit à l’oubli, exportabilité |
| **Sécurité** | OWASP Top 10 / SSO / OAuth2 | Prévention XSS/CSRF, sécurisation des accès, chiffrement TLS |
| **API & intégration** | OpenAPI 3.x | Spécifications valides, versionnées, avec tests contractuels |
| **Interopérabilité** | REST + JSON / Webhooks | Respect des formats normalisés, gestion des erreurs, fallback |
| **Documentation technique** | ADRs / Modèles C4 | Mise à jour à chaque release majeure ou changement structurant |
| **Testing & qualité** | TDD / Lint / Coverage | Seuils de couverture définis, pipelines bloquants si non conformes |

è **La conformité est vérifiée à chaque jalon technique** via des revues croisées, des tableaux de validation, et des outils d’analyse automatique (lint, test coverage, scan sécurité…).

## Livrables d’architecture et indicateurs associés

Cette section recense l’ensemble des **artefacts d’architecture** attendus de la part des équipes techniques (développement, design, DevOps), en lien avec :

* les exigences métier et techniques définies dans les documents précédents (déclaration, spécification),
* les **standards TOGAF** sur la production d’artefacts formels,
* et les **métriques d’acceptation** pour mesurer leur pertinence et leur conformité.

### Liste des artefacts attendus (C4, ADR, APIs, doc, etc.)

| **Nom du livrable** | **Responsable** | **Format attendu** | **Périodicité / Jalon** |
| --- | --- | --- | --- |
| Diagrammes C4 (contexte, container, component) | Architecte / Dev Lead | PNG + fichier source (ex : draw.io) | Initial + MAJ à chaque version majeure |
| ADRs (Architecture Decision Records) | Architecte / Devs | Markdown (adr/001-title.md) | 1 par décision structurante |
| Contrats d’API (OpenAPI 3.x) | Développeur Back / Architecte | YAML + documentation lisible | Avant dev + à jour en production |
| Cas de tests automatisés | QA / Dev | Code + rapports CI/CD | À chaque merge ou release |
| Dashboards d’observabilité | DevOps / SRE | Grafana, Kibana, etc. | Obligatoire avant mise en production |
| Livret de déploiement | DevOps | Documentation technique versionnée | Obligatoire en production |

### Liens vers les exigences & métriques (business + tech)

| **Indicateur** | **Valeur cible** | **Mesuré comment ?** |
| --- | --- | --- |
| Couverture de tests automatisés | ≥ 80 % (unitaires) | Outils CI/CD + badge Git |
| Respect des règles de lint | 100 % OK | Linter dans pipeline |
| Taux de documentation ADR par décision majeure | 100 % | Validation par relecture |
| Délai de livraison d’un contrat API après validation fonctionnelle | ≤ 3 jours ouvrés | Suivi Jira / Notion |
| Complétude des modèles C4 | 100 % composants critiques représentés | Revue d’architecture trimestrielle |

### Modalités de validation et d’acceptation des livrables

Tous les livrables sont :

* versionnés dans un **dépôt Git dédié** (architecture/, docs/, adr/, etc.),
* soumis à **relecture (pull request ou comité architecture)**,
* validés via des **checklists de conformité** automatisées ou manuelles,
* mis à jour à chaque **release majeure**, ou en cas d’**évolution d’architecture**.

è Aucun livrable n’est considéré comme **“accepté”** s’il ne respecte pas les formats, les standards et les règles de validation définies dans ce contrat.

### Cycle de vie des artefacts

**Cycle de vie et gouvernance des artefacts :**

| **Artefact** | **Responsable de la mise à jour** | **Versioning** | **Stockage** |
| --- | --- | --- | --- |
| ADRs | Architecte / Tech Lead | Sémantique (ex : v1.0, v1.1) | adr/ (GitHub/Notion) |
| C4 Diagrams | Dev Lead / Architecte | Selon jalon / release | architecture/c4/ |
| API Specs | Dev Backend / QA | Par endpoint (ex: v1.2.yaml) | api/openapi/ |
| Dashboards | DevOps / SRE | Continu (GitOps / UI) | Grafana / Kibana |
| Documentation utilisateur | QA / PO | À chaque fonctionnalité | Notion / Confluence |

Une revue de gouvernance des artefacts est prévue à chaque jalon d’architecture ou sprint majeur.

## Plan de travail commun priorisé

Cette section détaille les **actions concrètes à mener**, organisées en **lots de travail (Work Packages)**. Chaque item contient :

* les **activités associées**,
* les **livrables attendus**,
* les **responsables identifiés**,
* et les **jalons temporels** (si définis).

**Exemple de format (à suivre pour chaque item) :**

| **Élément de travail** | **Activités clés** | **Livrables associés** | **Responsable** | **Échéance** |
| --- | --- | --- | --- | --- |
| WP-1 : APIs Produits | Définir le contrat OpenAPI | Spéc OpenAPI | Lead Tech | Semaine 38 |

* Implémenter endpoint GET /produits
* Tests automatisés
* Documentation front / Spéc OpenAPI validée
* Endpoint déployé
* Tests automatisés
* Page de doc API
* WP-2 : Logging centralisé / Choix des événements clés
* Configuration Stack ELK
* Intégration dans microservices
* Dashboard initial
* Logs vérifiables
* Guide de loggage / DevOps

**Itération typique :**

Chaque élément de travail suit une **séquence standardisée** :

1. **Design technique initial** : alignement sur principes d’architecture
2. **Implémentation** : développement + test unitaire
3. **Validation architecture** : conformité OpenAPI, ADR, C4
4. **Documentation et versioning**
5. **Déploiement dans environnement cible**

**Gouvernance du plan de travail :**

* Chaque livrable est **tracké dans Jira / Notion / GitHub** avec des étiquettes spécifiques (#archi, #adr, #openapi) ;
* Les **pull requests critiques sont relues par un architecte** ou référent technique ;
* Une **vue consolidée des livrables d’architecture** est tenue à jour dans un espace commun (/architecture-deliverables) ;
* Les blocages sont **escaladés lors des comités de gouvernance architecture** hebdomadaires ou mensuels.

## Plan de communication et de synchronisation

La réussite de ce contrat repose autant sur la qualité des livrables que sur une **communication régulière, structurée et bidirectionnelle** entre les équipes concernées.

Ce plan vise à :

* Assurer un **suivi proactif des actions liées à l’architecture**,
* Garantir la **visibilité mutuelle des dépendances, arbitrages, décisions**,
* Faciliter la **résolution rapide des blocages**,
* Maintenir un **alignement constant** entre architecture cible, implémentation et besoins métier.

### Évènements de coordination (points techniques, démos, revues)

| **Type d’évènement** | **Fréquence** | **Participants** | **Objectif** |
| --- | --- | --- | --- |
| Revue d’architecture technique | 1× / sprint | Architecte, Dev Leads, QA, DevOps | Suivi des ADR, livrables C4, APIs, logs, etc. |
| Points de synchronisation Design–Dev | 1× / semaine | UX/UI, Front, Back, PO | Alignement interface ↔ APIs ↔ UX ↔ périmètre |
| Comité de gouvernance architecture | 1× / mois | Sponsors, Architecte, Tech Leads | Arbitrages, décisions majeures, roadmap, risques |
| Démos internes d’implémentation | 1× / sprint | Toutes fonctions | Présentation des features + vérification conformité architecture |
| Retrospectives spécifiques architecture | Chaque jalon | Dev, Archi, QA | Feedback sur qualité, documentation, décision, collaboration |

### Canaux et formats (Slack, Git, Notion, Miro, etc.)

| **Canal** | **Usage principal** |
| --- | --- |
| Slack / Discord | Discussion temps réel, alerte rapide, partage de lien (ADR, specs…) |
| GitHub / GitLab | Suivi des livrables techniques (PR, versions, lints, ADRs, OpenAPI) |
| Notion / Confluence | Documentation collaborative, modèles C4, backlog d’artefacts |
| Jira / Trello | Suivi opérationnel des tâches architecture, dépendances |
| Miro / FigJam | Schémas collaboratifs, design sprint, cadrage fonctionnel |
| Email (formel) | Validation finale, partages asynchrones avec sponsors externes |

Une base de connaissances centralisée est tenue à jour via Notion (ou Confluence), contenant les modèles, les décisions (ADRs), les versions de specs et les outils d’alignement UX–Dev–Ops.

### Fréquence et rythme (hebdo, sprint, jalons…)

**Formats & Fréquence :**

| **Format** | **Support** | **Fréquence** |
| --- | --- | --- |
| Revue hebdomadaire d’artefacts | Checklist partagée + revue de PRs | 1× / semaine |
| Présentation des décisions (ADRs) | Fiche synthétique + présentation orale | À chaque décision structurante |
| Tableaux de conformité | Fichier partagé / Notion / Excel | 1× / jalon (sprint ou mensuel) |
| Dashboards observabilité & performance | Grafana, Datadog, Kibana | Temps réel + revue mensuelle |

**Rythme de communication :**

* **Daily technique** : Intégration architecture dans les blocages et avancées
* **Sprint Planning** : Intégration explicite des livrables archi à planifier
* **Sprint Review** : Démonstration des artefacts mis à jour / livrés
* **Sprint Retro** : Feedback sur l’applicabilité des décisions d’architecture

è Ce plan de communication favorise la **co-responsabilité** des livrables et décisions. Il s’intègre directement dans le framework Agile/DevOps en place chez Foosus.

## Gouvernance, risques et arbitrage

Une gouvernance efficace est essentielle pour garantir que l’architecture cible est mise en œuvre de façon rigoureuse, cohérente et évolutive. Cette section décrit les **structures de décision**, les **mécanismes de contrôle**, et la **gestion active des risques** liés à l’implémentation technique.

### Structure de gouvernance (comités, règles, outils)

| **Instance** | **Responsabilités** | **Fréquence / Mode** | **Participants** |
| --- | --- | --- | --- |
| Comité de gouvernance architecture | Arbitrage, validation d’écarts, suivi conformité | Mensuel ou à la demande | Architecte, CTO, Tech Leads, PO |
| Revue d’architecture technique | Validation des livrables, alignement pratique ↔ théorie | 1× / sprint | Architecte, Dev Leads, QA, DevOps |
| Point escalade / arbitrage | Résolution de conflits bloquants / techniques | À la demande | Architecte + Direction / Sponsors |
| Groupe de validation des ADR | Approbation ou refus des décisions structurantes | Async + Revue PR | Architecte, Tech Leads, Devs référents |

Les décisions structurantes sont enregistrées dans un **registre d’ADRs versionné** (voir section 6), et **toute dérive ou dérogation est documentée, évaluée et justifiée**.

### Analyse des risques techniques & organisationnels

Voici les **principaux risques identifiés** concernant l’exécution de ce contrat d’architecture, ainsi que les **mesures de réduction associées** :

| **ID** | **Risque** | **Gravité** | **Probabilité** | **Facteur de réduction** | **Propriétaire** |
| --- | --- | --- | --- | --- | --- |
| R1 | Non-conformité des livrables aux décisions d’architecture | Élevée | Moyenne | Revue architecture systématique en sprint | Architecte |
| R2 | Manque de documentation ou d’artefacts techniques | Moyenne | Moyenne | Checklist de livrables, relecture PR obligatoire | Dev Lead |
| R3 | Refus implicite de certains principes par les équipes (shadow decision) | Moyenne | Élevée | Revue croisée ADR, pédagogie, présentation active | Tech Lead / Archi |
| R4 | Retard dans la production des specs contractuelles d’API | Élevée | Moyenne | CI avec fail si pas de spec, planification préalable | Dev Backend |
| R5 | Incompatibilité entre vision design et contraintes back-end | Moyenne | Moyenne | Points hebdo UX-Tech, prototypage rapide | UI Designer / Dev Front |

Les risques sont suivis dans un **registre partagé** (type Notion / Excel / Git), mis à jour à chaque rétrospective architecture.

En cas de désalignement grave ou conflit non résolu, le comité de gouvernance architecture détient **l’autorité de dernier recours**, avec possibilité d'escalade auprès du CTO ou de la direction générale si nécessaire.

### Procédures d’escalade

En cas de désaccord, de dérive significative, ou de non-respect avéré du présent contrat d’architecture, une procédure d’escalade structurée est activée pour :

* garantir une **résolution rapide et encadrée** des incidents d’implémentation,
* maintenir la **cohérence de l’architecture** face à des pressions locales ou des arbitrages tactiques,
* préserver la **gouvernance de la trajectoire cible**, en évitant la dette ou l’érosion des principes d’architecture.

**Escalade standard (technique ou méthodologique) :**

| **Niveau** | **Délai d’attente max.** | **Responsables** | **Action attendue** |
| --- | --- | --- | --- |
| **Niveau 1 — Équipe** | **24–48h** | **Dev / Design / QA / DevOps** | **Tentative de résolution en autonomie + documentation** |
| **Niveau 2 — Référent technique** | **48h** | **Lead Dev ou Tech Lead + Architecte** | **Analyse + proposition d’arbitrage conforme aux principes** |
| **Niveau 3 — Comité d’architecture** | **Hebdo ou ad hoc** | **Architecte, CTO, PO, QA, etc.** | **Décision formelle + documentation dans le registre** |
| **Niveau 4 — Direction / Sponsors** | **Exceptionnel** | **CTO, CEO, CPO** | **Arbitrage stratégique en cas de conflit bloquant ou de dérive majeure** |

**Principes d’escalade TOGAF appliqués**

* **Documentation obligatoire à chaque niveau (ticket, PR, Notion, commentaire d’ADR, etc.).**
* **Décisions formelles enregistrées dans le journal des arbitrages (ou ADRs versionnés).**
* **Responsabilité collective de porter les conflits de façon transparente et constructive.**

**è En cas d’escalade au niveau 3 ou 4, un compte rendu formel est produit et, si besoin, un avenant au contrat est proposé (voir section 12).**

## Hypothèses

Ce contrat d’architecture repose sur un ensemble d’**hypothèses de contexte et de disponibilité**, ainsi que sur **des dépendances techniques et organisationnelles** externes à son périmètre d’exécution direct.

Ces éléments sont documentés pour :

* Clarifier les **conditions de validité** du présent contrat,
* Identifier les **risques en cas de non-respect**,
* Servir de base aux éventuels **avenants ou ajustements** (voir Section 12).

**Hypothèses de départ :**

| **ID** | **Hypothèse** | **Impact si fausse** | **Propriétaire** |
| --- | --- | --- | --- |
| H1 | La plateforme cible s’appuie sur une infrastructure cloud managée (PaaS) | Retards, surcoût, rework architecture | DevOps |
| H2 | Les équipes de dev ont été formées aux modèles C4, ADR et OpenAPI | Livrables non conformes ou incohérents | Architecte / Lead Dev |
| H3 | Un outil de versioning (Git) centralisé est utilisé pour tous les artefacts d’architecture | Risque de perte d’information, audit impossible | Architecte |
| H4 | Le découpage en microservices a été validé et n’évoluera pas sur la période cible | Révision des interfaces, rework specs | CTO |
| H5 | Le backlog produit est priorisé et maintenu à jour côté PO | Décalage vision produit ↔ architecture | Product Owner |

**Dépendances externes :**

| **ID** | **Dépendance** | **Nature** | **Responsable externe** |
| --- | --- | --- | --- |
| D1 | Livraison du connecteur SAP / CRM | Interconnexion applicative | Intégrateur partenaire |
| D2 | Règlement RGPD / directives CNIL | Conformité sécurité / données | DPO |
| D3 | Design System validé par UI | Alignement front ↔ maquettes | Lead Design |
| D4 | Planning marketing des MVP | Roadmap alignée avec les sprints | CMO / CPO |

è Ces hypothèses sont **revues à chaque jalon ou comité de gouvernance architecture**. Tout changement significatif peut conduire à un **réexamen du contrat ou à un avenant** (voir section 12).

## Critères d’acceptation et validation

Les critères suivants définissent les conditions formelles de validation des livrables techniques, dans le cadre de ce contrat d’architecture. Ils visent à garantir une **conformité technique, une valeur métier observable et une traçabilité complète**.

### KPIs de l’architecture cible

Ces indicateurs mesurent l’**adéquation des livrables techniques à l’architecture cible** :

| **Métrique** | **Objectif** | **Méthode de mesure** | **Justification** |
| --- | --- | --- | --- |
| Conformité des APIs aux specs OpenAPI | 100 % | Analyse via CI + PR review | Garantit l’interopérabilité, la lisibilité et la documentation |
| Respect des modèles C4 (niveau 2/3) | 100 % composants critiques | Revue architecture + template imposé | Assure la traçabilité et la vision système |
| Taux de couverture ADRs / décisions majeures | 100 % | Check ADR Git + validation comité | Trace toutes les décisions structurantes |
| Intégration sécurité (OWASP, RGPD) | 100 % des composants exposés | Scan code + validation DPO | Conformité réglementaire et sécurisation early-stage |
| Taux de test automatisé | ≥ 80 % (unitaires + intégration) | Outils CI / Coverage report | Garantit la stabilité et la maintenabilité |

### KPIs de livraison / valeur métier

| **Indicateur métier** | **Cible** | **Responsable** | **Observé via** |
| --- | --- | --- | --- |
| Time-to-market (MVP) | ≤ 3 sprints | PO / Dev Lead | Suivi Jira |
| Nombre de fonctionnalités isolables | ≥ 70 % | Tech Lead | Architecture + toggles |
| Erreurs sur endpoints critiques en production | < 1 % | DevOps | Observabilité (Grafana / logs) |
| Disponibilité des services clés | ≥ 99.9 % | DevOps / SRE | SLO / SLA Dashboard |
| Retours correctifs post-release | ≤ 10 % | QA / PO | Taux de bug dans backlog |

### Processus de validation mutuelle

1. **Pré-validation technique** par Dev Lead (CI/CD, lint, tests, specs) ;
2. **Revue croisée architecture** (PRs + artefacts associés : C4, ADR, API) ;
3. **Validation métier** si les KPIs de valeur sont mesurables dès la release ;
4. **Checklists de conformité** signées ou validées par les parties (via Notion, GitHub, ou autre) ;
5. **Archivage de la décision** (release note, issue, merge, ou commentaire Git).

è Aucun livrable n’est considéré comme finalisé sans validation croisée architecture & PO. En cas de désalignement, la **procédure d’escalade** décrite en section 9.c s’applique.

## Phasage, jalons et calendrier

Ce contrat d’architecture s’applique sur une **trajectoire multi-phasée**, alignée avec la feuille de route produit, les cycles de développement agiles, et les livraisons incrémentales de la plateforme Foosus.

### Conditions requises pour la conformité

Afin qu’un jalon soit considéré comme **validé d’un point de vue architectural**, les **conditions suivantes doivent être réunies** :

| **Type de livrable** | **Critères de conformité** | **Vérification** |
| --- | --- | --- |
| API / Microservice | Spéc OpenAPI validée, code review passée, CI/CD verte, tests ≥ 80% | PR + badge Git |
| Modèle C4 / ADR | Modèle mis à jour, revue architecture, lien vers ADR dans PR | Lien vers doc + validation croisée |
| Sécurité | Analyse OWASP passée, données conformes RGPD, logs activés | Scan + validation DPO |
| Documentation | Présente, versionnée, accessible à l'équipe | PR + validation archi |
| Observabilité | Logs, métriques, alertes intégrés à la stack cible | Dashboard + test de validation |

### Développement et propriété de l’architecture

| **Élément** | **Responsable du développement** | **Responsable de la validation finale** |
| --- | --- | --- |
| ADRs | Architecte / Lead Dev | Architecte |
| Diagrammes C4 | Dev Lead / Architecte | Comité archi |
| Spécs d’API | Dev Back / PO | Architecte + PO |
| Dashboards SLO | DevOps | DevOps + Archi |
| Documentation technique | Devs / QA | Lead technique |

è La **propriété collective** de l’architecture est encouragée. Toutefois, les décisions stratégiques restent pilotées par l’équipe d’architecture, en concertation avec les fonctions Design/Dev/QA.

### Jalonisation prévisionnelle

| **Jalon** | **Objectif** | **Date cible** |
| --- | --- | --- |
| J1 | Finalisation architecture cible (ADRs + C4 v1) | Semaine 34 |
| J2 | Spécification des APIs Produits / Panier | Semaine 36 |
| J3 | Livraison MVP 1 avec observabilité activée | Semaine 40 |
| J4 | Revue architecture intermédiaire (v1.1) | Semaine 42 |
| J5 | Livraison finale phase 1 + validation contrat | Semaine 46 |

## Signataires et approbation

Ce contrat d’architecture entre en vigueur à compter de sa validation par les parties listées ci-dessous. Chaque signataire atteste avoir :

* Pris connaissance de la vision, des objectifs, des responsabilités et des contraintes décrites,
* Accepté les modalités d’exécution, de gouvernance et de validation,
* Compris son rôle dans l’exécution et la réussite de cette architecture.

**Tableau des signatures :**

| **Nom** | **Fonction** | **Équipe / Org.** | **Signature** | **Date** |
| --- | --- | --- | --- | --- |
| Hedi Dhib | Architecte Logiciel | Foosus |  | 20/08/2025 |
| Sarah Benali | Lead Developer | Foosus |  |  |
| Thomas Giraud | UX Designer Senior | Foosus |  |  |
| Léa Dubois | QA / Test Lead | Foosus |  |  |
| Julien Lefèvre | DevOps / SRE | Foosus |  |  |
| Claire Morel | CPO / Représentante Métier | Foosus |  |  |
| Marc Girard | CTO | Foosus |  |  |

**Annexe 1 – Références et standards d’architecture :**

| **Standard** | **Lien / Référence** | **Utilisation dans ce projet** |
| --- | --- | --- |
| OWASP Top 10 | <https://owasp.org/Top10> | Conformité sécurité applicative |
| OpenAPI 3.x | <https://swagger.io/specification> | Contrat API et documentation |
| C4 Model | <https://c4model.com> | Diagrammes d’architecture |
| Git Conventional Commits | <https://www.conventionalcommits.org> | Historique clair dans Git |
| Clean Architecture | Robert C. Martin | Structuration code + responsabilités |
| RGPD | <https://cnil.fr> | Conformité traitement des données |
| CI/CD (GitHub Actions) | <https://docs.github.com/actions> | Déploiement automatisé |

**Annexe 2 – Glossaire des acronymes :**

| **Acronyme** | **Signification** |
| --- | --- |
| ADR | Architecture Decision Record |
| API | Application Programming Interface |
| CI/CD | Continuous Integration / Continuous Delivery |
| C4 | Context, Container, Component, Code (modèle d’architecture) |
| DPO | Data Protection Officer |
| EDA | Event-Driven Architecture |
| KPI | Key Performance Indicator |
| RGPD | Règlement Général sur la Protection des Données |
| SLO | Service Level Objective |
| TDD | Test-Driven Development |
| UX/UI | User Experience / User Interface |

**Annexe 3 – Cartographie de l’architecture cible (niveau C4) :**

Ce diagramme représente une **vue d’ensemble de la solution Foosus** dans son environnement technique et fonctionnel. Il suit le **modèle C4 de Simon Brown**, niveau **Contexte** (macro) ou **Container** (composants logiques déployés).

Il permet :

* une **vision partagée entre métiers, dev, architecture et partenaires externes**,
* un support d’**onboarding rapide** des nouveaux arrivants,
* une **traduction graphique du périmètre d’application** du présent contrat d’architecture.

Structure du diagramme C4 (niveau Container) :

| **Élément** | **Description** |
| --- | --- |
| Front Web Client | Interface React utilisée par les utilisateurs finaux |
| Mobile App | Application Flutter/React Native pour iOS/Android |
| API Gateway | Point d’entrée unique, gestion des routes et sécurité |
| Service Produits | Microservice métier pour la gestion des produits locaux |
| Service Commande | Gestion du panier, paiement, tracking |
| Service Utilisateur | Authentification, SSO, gestion des profils |
| Base de données | PostgreSQL ou MongoDB selon contexte |
| Système de Logs / Monitoring | Stack ELK ou Grafana + Prometheus |
| Système d’événements | RabbitMQ ou Kafka pour communication asynchrone |
| Externe : CRM / SAP | Connecteurs vers systèmes externes |
| Externe : Stripe / Email | Intégrations services tiers |

**diagramme C4 (niveau Container)**

Une image contenant texte, capture d’écran, diagramme, Plan

Le contenu généré par l’IA peut être incorrect.

[Lien vers Dépot Github](https://github.com/OpenClassrooms-Student-Center/Concevez-une-nouvelle-architecture-afin-de-soutenir-le-d-veloppement-de-votre-entreprise)