Spécification des conditions requises pour l'architecture

Nom de l'entreprise : Foosus

Nom du projet : Refonte de la plateforme Foosus

Personne responsable du dossier : Emrys Callait

Table des matières

Version

Date	Version	Commentaires
10/07/2025 JJ/MM/AAAA	V.1.0	Document original Revue par :

Objectif du document

La spécification des conditions requises pour l'architecture fournit un ensemble de déclarations quantitatives qui dessinent ce que doit faire un projet d'implémentation afin d'être conforme à l'architecture.

Une spécification des conditions requises pour l'architecture constitue généralement un composant majeur du contrat d'implémentation, ou du contrat pour une définition de l'architecture plus détaillée.

Comme mentionné ci-dessus, la spécification des conditions requises pour l'architecture accompagne le document de définition de l'architecture, avec un objectif complémentaire : le document de définition de l'architecture fournit une vision qualitative de la solution et tâche de communiquer l'intention de l'architecte. La spécification des conditions requises pour l'architecture fournit une vision quantitative de la solution, énumérant des critères mesurables qui doivent être remplis durant l'implémentation de l'architecture.

Mesures du succès

Le projet sera considéré comme réussit si tu les indicateurs sont validés

Indicateur	Changement souhaité pour l'indicateur
Nombre d'adhésions d'utilisateurs par jour	Augmentation de 10 %
Adhésion de producteurs alimentaires	Passer de 1,4/mois à 4/mois
Délai moyen de parution	Réduit de 3,5 semaines à moins d'une semaine
Taux d'incidents de production P1	Pour commencer : réduit de >25/mois à moins de 1/mois.

Conditions requises pour l'implémentation

Cette section définit les principes, indicateurs et contraintes que l'implémentation doit respecter pour être considérée comme réussie et alignée avec les objectifs de l'entreprise.

Principes directeurs

- **Croissance et scalabilité :** L'architecture doit pouvoir soutenir une croissance jusqu'à un million d'utilisateurs et une expansion dans de nouvelles villes et de nouveaux pays. Elle doit pouvoir gérer les pics de charge sans dégrader les performances

- **Fiabilité et disponibilité :** La plateforme doit être active 24/7 et garantir une disponibilité constante. Les déploiements et les mises à jour ne doivent entraîner aucune interruption de service pour les utilisateurs

- **Réduction du « Time to Market » :** L'architecture doit permettre des cycles de développement courts et des déploiements fréquents et sécurisés pour lancer rapidement de nouvelles fonctionnalités

- **Sécurité dès la conception (security by design) :** La sécurité est un fondement non négociable et doit être intégrée à chaque service pour protéger les utilisateurs et la réputation de la marque, inversant la tendance passée qui privilégie la facilité d'utilisation

- **Agilité et innovation :** La structure doit permettre aux équipes d'innover rapidement, de tester des variantes et de réagir aux retours des clients sans être freinées par la complexité technique

Contrats de service business

Fonctionnalités Fondamentales

- **Gestion des utilisateurs :** La plateforme doit prendre en charge différents types d'utilisateurs (consommateurs, fournisseurs / producteurs, back-office) avec des fonctionnalités spécifiques à chaque rôle

- **Recherche et géolocalisation :** Cette fonctionnalité est prioritaire et doit permettre aux utilisateurs de trouver des fournisseurs par proximité, de calculer des distances et de prioriser les résultats locaux. Elle doit aussi afficher l'emplacement des offres

- **Informations produits :** Le système doit permettre d'afficher des informations secondaires sur les produits (indice glycémique, …) et de trier les offres selon divers critères, en se basant sur le processus actuel jugé positif

- **Commande :** Le processus de commande existant (ajout au panier, accord pour paiement à la livraison, envoi d'instructions par e-mail) doit être maintenu. L'architecture doit cependant prévoir l'intégration future de prestataires de paiement tiers et d'une interface de communication centralisée avec les fournisseurs

- **Expérience utilisateur :** L'interface doit être une progressive web app responsive, s'adaptant à tous les types de supports (mobiles et fixes),

elle doit aussi être performante même sur des réseaux à faible bande passante

- **Capacités d'expérimentation :** La plateforme doit fournir aux équipes produit la capacité de lancer des tests A/B et de collecter des données sur l'utilisation des fonctionnalités

Accords de niveau de service

Les objectifs de niveau de service identifiés pour la nouvelle architecture Foosus :

- **Disponibilité et fiabilité du service :** L'objectif principal est de fournir une plateforme fiable et constamment accessible

- **Disponibilité continue :** La plateforme doit être active et opérationnelle 24/7. Les déploiements de nouvelles versions ou les modifications de la base de données ne doivent plus nécessiter de désactiver la plateforme

- **Réduction des incidents critiques :** Le taux d'incidents de production de priorité 1 (P1) doit être radicalement réduit, passant de plus de 25 par mois à moins de 1 par mois

- **Gestion de la surcharge :** Le système doit pourvoir scaler horizontalement automatiquement

- **Adaptabilité au réseau :** La solution doit prendre en compte les contraintes de bande passante et être performante aussi bien sur des connexions lentes (mobiles) que sur des réseaux haut débit

Contrats de service application

Objectifs de niveau de service

Le service doit être disponible 99% du temps dans toutes les régions

Indicateurs de niveau de service

Criticité	Description	Temps de prise en charge	Temps de résolution
P1	Le site n'est plus accessible dans sa globalité	30 min	2 heures
P2	Un module important n'est disponible / une partie des utilisateurs ne peuvent pas se connecter	1 heure	4 heures
P3	Dysfonctionnement n'impactant pas les modules principaux (cosmétique / pas dans le flux de recherche, de commande, de paiement, d'inscription)	4 heures	48 heures

Lignes directrices pour l'implémentation

Les principes fondamentaux pour la conception de la nouvelle architecture sont :

- **La scalabilité** afin d'assurer un redimensionnement et une mise à l'échelle du système selon les besoins. L'architecture doit pouvoir absorber les pics d'utilisation sans pannes, évoluer avec la base de clientèle et permettre le déploiement sur de nouvelles régions géographiques

- **La continuité de service** afin de garantir une plateforme active 24/7 et d'éliminer les interruptions lors des déploiements. Même en cas de surcharge, le système doit rester accessible, quitte à fonctionner de manière dégradée, pour éviter toute panne complète

- **La sécurité** afin de garantir la protection des données et de la plateforme. Ce principe est une priorité pour éviter les risques pour l'image de marque, qui a été mise en péril par le passé lorsque la facilité d'utilisation a été préférée à la sécurité

- **L'évolutivité** afin de permettre à l'entreprise d'innover à nouveau rapidement et de manière responsable. L'architecture doit donner aux équipes produit la capacité d'expérimenter avec de nouvelles modifications, de réorienter des solutions existantes et d'exécuter des comparaisons de différentes solutions auprès des utilisateurs

Spécifications pour l’implémentation

Exigences applicatives et de conception

- Modularité des Services : L'architecture doit être décomposée en services indépendants (microservices), chacun responsable d'un domaine métier unique, afin de supporter le principe de réactivité au changement

- Communication par APIs : Toute communication entre les services doit se faire via des APIs standardisées et versionnées pour garantir l'interopérabilité et le découplage

- Déploiement indépendant : Chaque service doit pouvoir être déployé de manière autonome, sans interruption de service pour les autres composants (continuité de service)

Exigences non fonctionnelles

- Haute disponibilité : Le système doit garantir une disponibilité de service 24/7, avec pour objectif la réduction des incidents critiques (P1) à moins de 1 par mois

- Scalabilité : L'architecture doit être conçue pour gérer la croissance prévue (prochain million d'utilisateurs, pics de charge marketing) par une mise à l'échelle horizontale et automatisée des services

- Performance : La plateforme doit offrir un temps de réponse optimal, y compris sur des connexions à faible bande passante (mobile), pour garantir une expérience utilisateur fluide.

- Fiabilité : En cas de défaillance d'un service non critique, le système global doit rester opérationnel en mode dégradé plutôt que de subir une panne complète

Exigences technologiques

- Containerisation : Tous les services applicatifs devront être packagés dans des conteneurs (ex : Docker) pour assurer la portabilité et la standardisation des déploiements

- Orchestration : Un orchestrateur de conteneurs (ex : Kubernetes) doit être utilisé pour gérer automatiquement la mise à l'échelle, le déploiement et la résilience des services

Exigences de Sécurité

- **Sécurité dès la conception ("Security by Design") :** Les mécanismes de sécurité (authentification, autorisation, chiffrement des données) doivent être intégrés nativement dans l'architecture et non ajoutés après coup

- **Isolation des services :** La compromission d'un service ne doit pas permettre d'accéder aux autres services ou à leurs données.

Exigences de Conformité

- **Conformité RGPD :** La gestion des données personnelles des utilisateurs et des producteurs doit être en stricte conformité avec le Règlement Général sur la Protection des Données (RGPD)

Standard de l'implémentation

Modèle architectural

- **Micro services :** L'application doit être décomposée en un ensemble de services indépendants, chacun responsable d'un domaine métier spécifique. Cette approche vise à augmenter l'agilité, à faciliter le déploiement autonome des équipes et à améliorer la fiabilité globale

- **API Gateway :** Un point d'entrée unique (API Gateway) doit être mis en place pour gérer, sécuriser et rediriger toutes les requêtes des clients vers les micros services appropriés du backend

- **Hébergement cloud (CaaS) :** La plateforme doit être hébergée sur une infrastructure cloud de type CaaS (Container as a Service), comme AWS, Google Cloud ou Azure, pour assurer la flexibilité, la scalabilité et la portée mondiale

- **Conteneurisation :** Les micro services doivent être conteneurisés et orchestrés pour permettre un dimensionnement indépendant et améliorer la sécurité

Gestion

- **Bases de données indépendantes :** Chaque micro service doit posséder sa propre base de données pour réduire l'interdépendance

- **Technologie de base de données adaptées :** Chaque service doit utiliser la technologie de base de données la plus appropriée à ses besoins spécifiques

- **Base de données spatiales :** Une base de données spatiale est requise pour stocker, interroger et manipuler des données géographiques, ce qui est essentiel pour la fonctionnalité de recherche géolocalisée

- **Base de données multilingues :** Une base de données capable de gérer et de restituer des données en plusieurs langues doit être intégrée pour adapter le contenu à la langue de l'utilisateur

Conditions requises pour l’interopérabilité

La nouvelle architecture doit être conçue pour faciliter l'intégration avec des partenaires internes et externes. L'un des objectifs à long terme les plus importants est de pouvoir s'intégrer à des prestataires de paiement tiers. Même si cette fonctionnalité n'est pas prioritaire pour la première phase du projet, la plateforme doit être conçue dès le départ pour pouvoir prendre en charge cette intégration future.

Architecture interne et normes

L'interopérabilité interne est une pierre angulaire de la nouvelle conception technique pour assurer la flexibilité et la réutilisation stratégique des composants.

- **Microservices :** Une approche d'architecture basée sur des microservices est envisagée. Cette approche favorise l'interopérabilité en s'appuyant sur des services indépendants qui communiquent via des APIs bien définies.

- **Normes** : L'architecture doit s'appuyer sur des normes qui prennent en charge des solutions web et mobiles afin d'assurer une communication cohérente entre les différentes parties du système.

Conditions requises pour le management du service IT

La gestion du service IT pour la nouvelle plateforme Foosus doit s'articuler autour de processus clairs visant l'agilité, la fiabilité et une gouvernance renforcée.

Gestion des déploiements et des changements

Gouvernance et responsabilité

La nouvelle approche de management doit établir une gouvernance plus explicite, tout en préservant l'autonomie et la responsabilisation des équipes. L'architecte travaillera en étroite collaboration avec le directeur produit (CPO) et le responsable ingénierie pour créer un plan menant à l'état cible de l'architecture. Pour assurer la transparence et la collaboration, tous les artefacts architecturaux seront centralisés et développés dans un dépôt GitHub.

Gestion des déploiements et des changements

Le processus de déploiement doit être radicalement amélioré pour supporter une innovation rapide. Cela implique d'assurer une disponibilité continue de la plateforme, qui doit être active 24/7 sans interruption de service pour les déploiements. Le délai moyen de parution (du code à la production) doit être réduit, en publiant des nouvelles versions de taille réduite et à faible risque qui sont transparentes pour les utilisateurs. Cette approche s'inscrit dans la culture de l'organisation qui pratique déjà le Kanban et l'amélioration continue.

Gestion des incidents et de la performance

L'objectif est d'améliorer drastiquement la stabilité et la fiabilité de la plateforme. Un objectif de niveau de service clair a été fixé pour réduire le taux d'incidents de production de priorité 1 (P1) de plus de 25 par mois à moins de 1 par mois. De plus, en cas de forte charge, le système doit pouvoir rester accessible de façon dégradée plutôt que de subir une panne complète. Pendant cette transition, la plateforme existante sera conservée en mode de maintenance, sans aucun développement de nouvelles fonctionnalités.

Gestion de la configuration

Pour maîtriser les coûts et la complexité, la gestion de la configuration technique est une priorité. Les directives du projet imposent que toutes les solutions, qu'elles soient open source ou commerciales, fassent si possible partie d'une même pile technologique afin de réduire les coûts de maintenance et de support. En outre, les solutions open source sont à préférer aux solutions payantes.

Contraintes

- La phase initiale de définition de l'architecture est approuvée pour un coût de 50 000 USD sur une période de 6 mois

- L'architecture doit viser le meilleur rapport qualité-coût

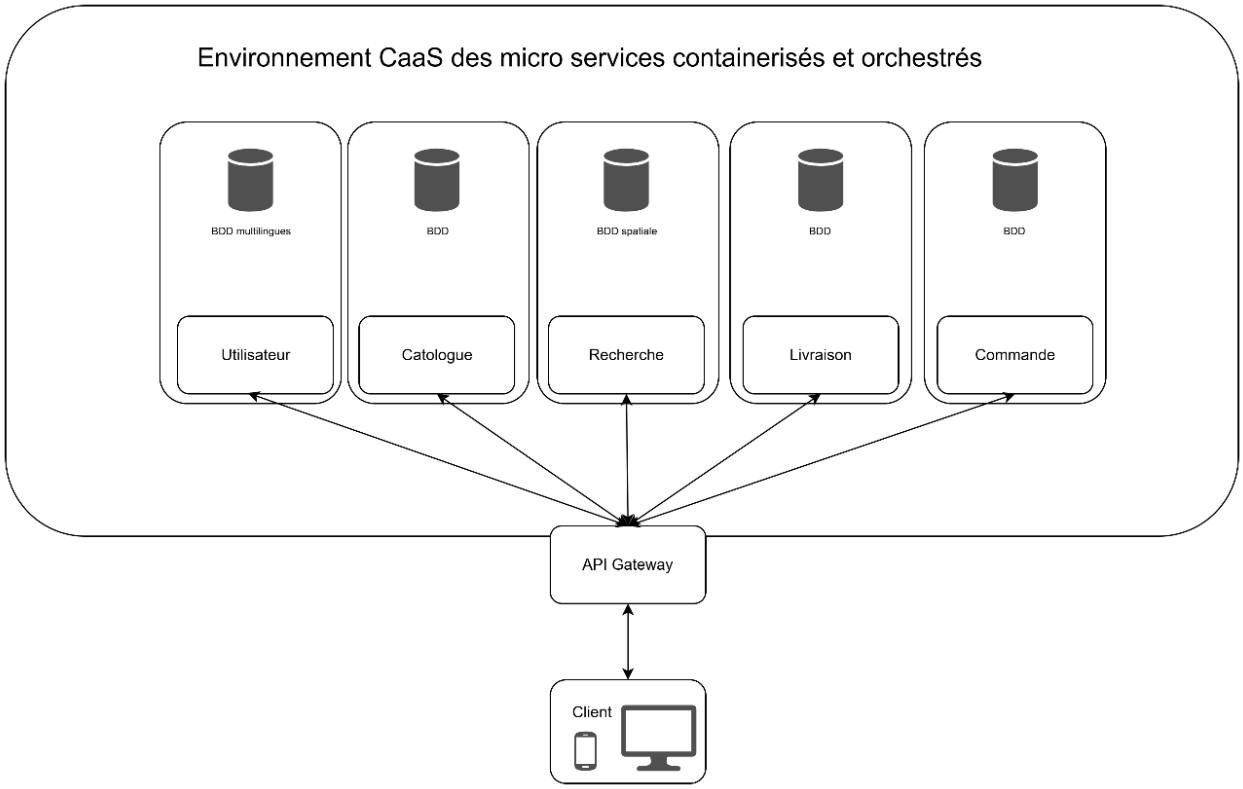
- Les solutions open source sont préférables aux solutions payantes, en tenant compte des coûts de support continus

- Pour minimiser les coûts de maintenance, les composants sélectionnés doivent, dans la mesure du possible, faire partie d'une pile technologique cohérente

- La nouvelle architecture coexistera initialement avec la plateforme existante, qui sera maintenue sans développement de nouvelles fonctionnalités

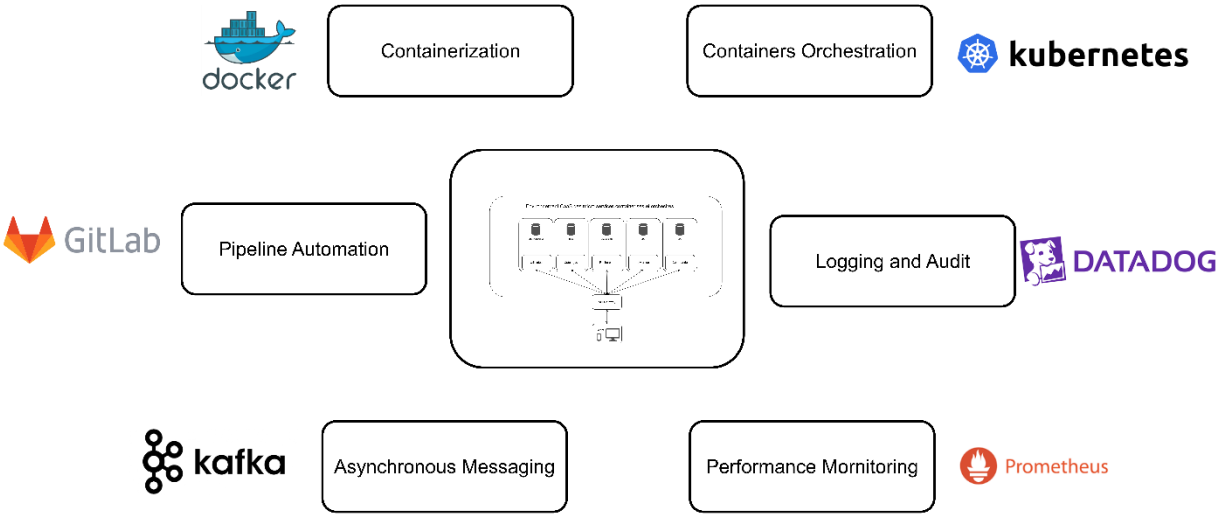
Hypothèses

Microservice



L'hypothèse d'adopter une architecture en microservices repose sur la nécessité de surmonter les limitations de la plateforme actuelle, qui freine l'innovation et ne peut plus évoluer au rythme de l'entreprise. L'architecture existante a accumulé une dette technique importante, rendant les déploiements risqués, nécessitant des interruptions de service et compliquant l'intégration des travaux de différentes équipes. Le passage à des microservices est envisagé comme une solution stratégique pour décomposer l'application en services plus petits, indépendants et à responsabilité unique. Cette approche permettrait aux équipes de développer, tester et déployer leurs services de manière autonome, réduisant ainsi le délai de mise sur le marché de nouvelles fonctionnalités. De plus, elle offrirait la scalabilité nécessaire pour supporter la croissance du nombre d'utilisateurs et l'expansion géographique, tout en améliorant la résilience globale du système, ce qui est un objectif clé pour l'entreprise.

Ecosystème



Pour déployer et gérer efficacement une architecture microservices, un écosystème d'outils spécifiques est indispensable. Bien que la sélection finale puisse varier, des technologies comme Docker sont nécessaires pour conteneuriser les applications, garantissant leur portabilité. L'orchestration de ces conteneurs, incluant la montée en charge et la résilience, est typiquement gérée par Kubernetes. L'automatisation des déploiements (CI/CD) requiert une plateforme telle que GitLab, tandis que la communication asynchrone et découplée entre services est souvent assurée par Kafka. Enfin, l'observabilité et la supervision de la santé des services sont cruciales, nécessitant des outils de monitoring comme Prometheus ou Datadog.