F00SUS

Déclaration de Travail d’Architecture

v 0.1

Ce document ainsi que toutes les informations qu'il contient sont des informations confidentielles appartenant à Foosus.

Table des matières

Information sur le document

Nom du projet	Conception d’une nouvelle architecture
Préparé par :	Clément Bastion
N° de version du document :	0.1
Titre :	Déclaration de travail d’architecture
Date de version du document :	12/07/2024
Revu par :	
Date de révision :	
Types d’action :	Approbation, Révision, Information, Classement, Action requise, Participation à une réunion, Autre (à spécifier)
Historique de versions du document	Voir git

Objet de ce document

Ce document est une **Déclaration de travail d'architecture** pour le projet de conception d'une nouvelle architecture.

Ce document spécifie :

- Une articulation claire d'une vision et d'une direction d'architecture qui permettent à Foosus de développer les capacités nécessaires pour réussir sur le marché.
- Un État Cible de l'Architecture vers lequel l'organisation doit itérer.
- Un process et une approche d'architecture sur mesure, mais flexibles, qui conviennent aux structures de nos équipes et à la topologie de notre organisation.

Déclaration de travail d’architecture

Requête du projet et contexte

Foosus est une start-up innovante et dynamique, spécialisée dans l’alimentation durable depuis trois ans. Leur mission est de promouvoir l’alimentation locale en connectant les consommateurs directement avec les producteurs et artisans locaux. Cependant, les choix technologiques historiques de Foosus ont conduit à une accumulation de dette technique et à un manque de cohérence, ce qui a récemment commencé à affecter significativement le développement de l'entreprise.

Avec le temps, le système d’information de Foosus est devenu trop complexe et n'évolue plus au rythme de l'activité, menaçant ainsi la croissance de l’entreprise. Pour remédier à cette situation, notre objectif commercial est de lancer rapidement et de manière itérative un nouveau produit qui pourra d'abord coexister avec la plateforme existante avant de la remplacer progressivement.

Le projet vise donc à établir les contraintes et les orientations architecturales nécessaires pour itérer rapidement vers nos objectifs commerciaux, assurant ainsi une transition efficace et durable vers une nouvelle architecture plus moderne et évolutive.

Description du projet et périmètre

L’objectif est de développer une plateforme d'e-commerce polyvalente et géo-ciblée pour propulser l'entreprise à un niveau supérieur. Pour rivaliser avec les grandes entreprises mondiales d'e-commerce qui dominent le marché de l'alimentation durable, il est indispensable d'optimiser l'efficacité, la flexibilité et de mettre en place des approches de prise de décision cohérentes.

Les principaux objectifs de l'entreprise sont les suivants :

1. **Utilisation de la géolocalisation :** La solution doit exploiter la géolocalisation pour connecter les fournisseurs et les consommateurs, en proposant des produits disponibles à proximité des lieux de résidence des utilisateurs.
2. **Architecture évolutive :** L'architecture doit être évolutive pour permettre le déploiement des services sur diverses régions, que ce soit dans différentes villes ou pays.

3. **Disponibilité multi-plateformes :** La solution doit être accessible pour les fournisseurs et les consommateurs, quel que soit leur emplacement. Elle doit être utilisable sur des appareils mobiles et fixes, tout en tenant compte des contraintes de bande passante pour les réseaux cellulaires et les connexions Internet haut débit.
4. **Support multi-utilisateurs :** La solution doit pouvoir gérer différents types d'utilisateurs, avec des fonctionnalités et des services spécifiques adaptés à chaque catégorie.

Vue d’ensemble

La géolocalisation constitue le cœur des nouvelles fonctionnalités de la plateforme. D’autres fonctionnalités, bien que non prioritaires, peuvent également être envisagées. Parmi celles-ci figurent le suivi des commandes avec un service client dédié pour les clients et un service de communication pour les fournisseurs, ainsi que l’approvisionnement en temps réel des offres par les fournisseurs.

Cette vue d’ensemble ne représente pas un état cible définitif, mais plutôt une vision évolutive. Elle a pour but de fournir une orientation stratégique pour Foosus, permettant à l'entreprise de s'adapter et de grandir en fonction des besoins et des opportunités du marché.

Alignement stratégique

Le projet Foosus de conception d’une nouvelle architecture est confronté à plusieurs contraintes, notamment en termes de budget et de délais. Il est donc pertinent de définir une vision d’une architecture évolutive, basée sur une mise à niveau progressive de l’architecture existante. Cette nouvelle architecture coexistera dans un premier temps avec la plateforme actuelle avant de la remplacer, une fois l’architecture cible de transition complétée, permettant ainsi une évolution continue.

Cette architecture ciblée de transition se présente comme une évolution de l’architecture historique, déployée dans le cloud, avec l’ajout de fonctionnalités répondant aux exigences de Foosus via des micro-services. La vision d’une architecture évolutive implique de futures évolutions continues, incluant l’extraction, la réécriture, l’amélioration et le redéploiement des différents composants de l’architecture historique. Ces évolutions seront réalisées via des technologies standardisées au sein de micro-services faciles à maintenir, tout en développant continuellement de nouvelles fonctionnalités.

Cela nécessite la mise en place de processus et d'une approche d'architecture sur mesure, mais flexible, adaptée aux structures des équipes et à la topologie de Foosus. Ainsi, ce projet vise à établir les contraintes et les orientations architecturales nécessaires pour itérer rapidement vers les objectifs commerciaux de Foosus, assurant ainsi une transition efficace et durable vers une nouvelle architecture plus moderne et évolutive.

Objectifs et périmètre

Objectifs

Objectif Business	Notes
Accélérer le Time-to-Market	Lancer rapidement de nouvelles fonctionnalités et produits pour répondre aux besoins évolutifs des consommateurs et du marché.
Améliorer l'efficacité opérationnelle	Optimiser les processus internes pour réduire les coûts et améliorer la productivité.
Soutenir la croissance de l’entreprise	Faciliter l'expansion dans de nouveaux marchés géographiques et l'augmentation de la base de clients.
Promouvoir l’innovation	Créer un environnement technologique flexible permettant une expérimentation rapide et sûre de nouvelles idées.
Renforcer l'expérience utilisateur	Offrir une performance élevée, une disponibilité continue et des fonctionnalités personnalisées pour les utilisateurs.
Faciliter l’intégration avec les partenaires externes	Assurer une intégration facile et rapide avec les systèmes des partenaires logistiques, des fournisseurs et des plateformes de paiement.
Assurer la conformité et la sécurité des données	Intégrer des mécanismes robustes de sécurité et de protection des données pour se conformer aux régulations en vigueur.

Périmètre

Les parties prenantes, leurs préoccupations et visions sont décrites dans le tableau suivant. Ce tableau montre les parties prenantes qui utilisent ce document, leurs préoccupations, et la façon dont le travail d'architecture répondra à ces préoccupations par l'expression de plusieurs visions.

Partie prenante	Préoccupation	Vision
Ash Callum, CEO	Maintenir un taux positif d'inscriptions de nouveaux utilisateurs, expansion dans les marchés locaux	Une architecture scalable qui soutient la croissance de la base d'utilisateurs en fournissant du géociblage et de la flexibilité pour toucher une gamme plus large d'utilisateurs.
Jo Kumar, CFO	Maximiser les investissements en IT tout en assurant un retour sur investissement rapide	Une plateforme évolutive qui permet de nouvelles fonctionnalités sans augmentation significative des coûts opérationnels.
Natasha Jarson, CIO	Garantir la stabilité et la sécurité du système tout en facilitant l'innovation technique	Une architecture robuste avec des mécanismes de sécurité intégrés, supportant l'innovation et les expérimentations rapides.
Christina Orgega, CMO	Améliorer la réputation de Foosus sur le marché et garantir la satisfaction des clients	Une plateforme fiable et stable, minimisant les interruptions de service et offrant une expérience utilisateur exceptionnelle.
Daniel Anthony, CPO	Accélérer le développement de nouvelles fonctionnalités et améliorer l'expérience produit	Un environnement de développement flexible et rapide, avec des micro-services modulaires permettant des itérations rapides.
Pete Parker, Ingénieur	Assurer la maintenabilité et la cohérence du code, tout en réduisant la dette technique	Une architecture modulaire et standardisée facilitant la maintenance et l'évolution continue du système sans complexité accrue.

Approche managériale

Foosus a longtemps cultivé une culture où les développeurs pouvaient librement expérimenter de nouvelles approches techniques. Bien que cela ait permis une grande créativité et une innovation rapide, cela a également conduit à une diversité de modèles et d'idées souvent non réutilisables. Cette diversité a engendré des défis en matière de maintenabilité et de standardisation, compliquant les efforts de collaboration entre les équipes et augmentant la complexité technique du système global.

Standardisation et Maintenabilité

Aujourd'hui, notre objectif est d'instaurer une standardisation accrue pour faciliter la maintenance des développements futurs tout en préservant la capacité d'innovation. La standardisation impliquera l'adoption de pratiques et de cadres communs pour le développement, l'intégration et le déploiement de nouvelles fonctionnalités. Cela inclut :

1. **Normes de Codage :**
    - Adoption de normes de codage claires et cohérentes pour garantir que tous les développeurs suivent les mêmes directives, réduisant ainsi les erreurs et facilitant la relecture de code.
2. **Cadres et Bibliothèques Partagés :**
    - Utilisation de cadres et de bibliothèques partagés pour éviter la duplication d'efforts et promouvoir la réutilisabilité des composants.
3. **Documentation Complète :**
    - Création et maintien de documents techniques détaillés pour tous les aspects du système, permettant aux nouveaux membres de l'équipe de se familiariser rapidement avec l'architecture existante.
4. **Outils de Gestion de Configuration :**
    - Mise en place d'outils de gestion de configuration pour suivre les modifications du code et assurer une gestion efficace des versions.

Communication et Apprentissage Continu

Nous mettons l'accent sur la communication ouverte et l'apprentissage continu pour optimiser cette architecture stratégique. Une communication efficace est essentielle pour garantir que toutes les équipes sont alignées sur les objectifs du projet et peuvent collaborer efficacement. Cela sera facilité par :

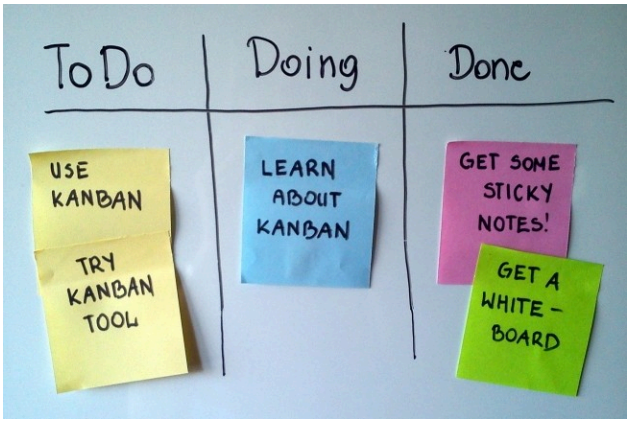
1. **Réunions de Synchronisation :**
    - Réunions quotidiennes (stand-ups) pour évaluer les progrès, identifier les obstacles et coordonner les efforts des différentes équipes.

    - Réunions hebdomadaires de revue des sprints pour discuter des réalisations, des défis rencontrés et des plans pour les itérations suivantes.
2. **Partage des Connaissances :**
    - Organisation de sessions de partage des connaissances où les développeurs peuvent présenter leurs travaux, partager leurs découvertes et apprendre des expériences des autres.
3. **Formation et Développement :**
    - Investissement dans la formation continue des équipes pour les tenir informées des dernières technologies, outils et pratiques du secteur.

Méthodologie Agile Kanban

Nous continuerons à adopter une méthodologie Agile Kanban, qui offre la flexibilité nécessaire pour s'adapter aux changements et aux besoins évolutifs de Foosus. En tant que praticiens Agiles du Lean, nous privilégions les personnes et les interactions plus que les processus et les outils. Cette approche permet de :

1. **Priorisation et Flexibilité :**
    - Utiliser des tableaux Kanban pour visualiser le flux de travail, prioriser les tâches et ajuster les priorités en fonction des besoins immédiats du projet.
2. **Livraisons Continues :**
    - Favoriser des cycles de développement courts et itératifs, permettant des livraisons continues de petites incrémentations de valeur, ce qui facilite les tests et l'intégration continue.
3. **Amélioration Continue :**
    - Encourager une culture d'amélioration continue en intégrant régulièrement des rétrospectives pour analyser ce qui fonctionne bien et identifier les domaines d'amélioration.



Environnement Collaboratif et Adaptatif

Cette approche favorise un environnement collaboratif et adaptatif, essentiel pour répondre aux besoins évolutifs de Foosus. Les équipes sont encouragées à travailler ensemble de manière transparente, à partager les responsabilités et à s'adapter rapidement aux nouvelles informations et aux changements de contexte. Les principaux éléments de cette collaboration incluent :

1. **Transparence :**
    - Maintenir une transparence totale sur les objectifs, les progrès et les défis du projet pour assurer que toutes les parties prenantes sont bien informées et engagées.
2. **Responsabilisation :**
    - Encourager les équipes à prendre des responsabilités et à s'approprier leurs tâches, renforçant ainsi l'engagement et la motivation.
3. **Réactivité aux Changements :**
    - Rester réactif aux retours des utilisateurs et aux changements du marché, permettant une adaptation rapide des priorités et des fonctionnalités développées.

En intégrant ces éléments dans notre approche managériale, Foosus sera mieux préparé à gérer les défis technologiques et organisationnels, tout en cultivant un environnement propice à l'innovation et à la croissance durable.

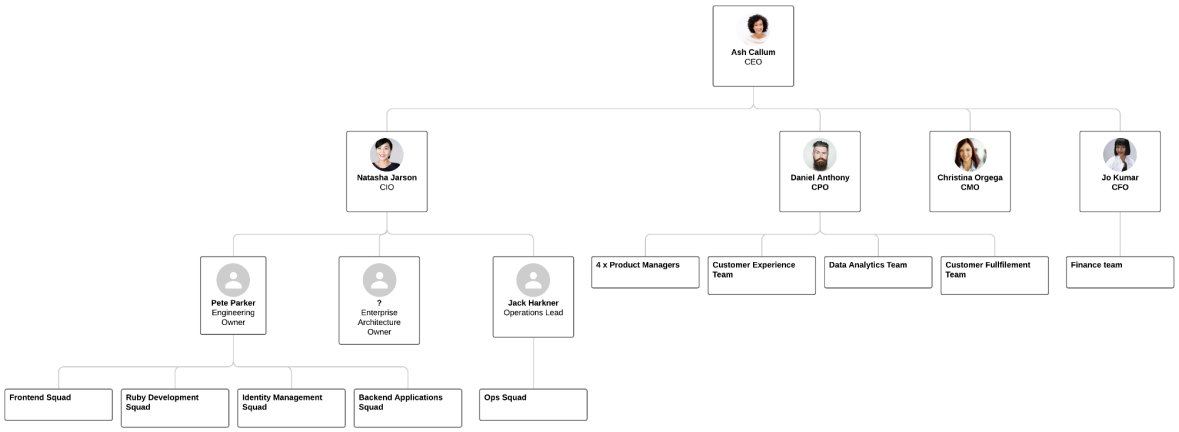
Procédures de changement de périmètre

Les changements de périmètre peuvent avoir un impact significatif sur les décisions et la direction du projet. Tout changement proposé doit être discuté et validé par Natasha Jarson et Ash Callum lors d’un comité de pilotage. Il est crucial de prendre toutes les mesures nécessaires pour assurer la cohérence et la continuité du projet.

Une fois les changements approuvés, les parties prenantes seront informées des décisions prises. Les modifications seront ensuite intégrées dans une nouvelle version du projet, avec une documentation mise à jour pour refléter ces ajustements. Cela garantit que toutes les parties restent alignées sur les objectifs et les directives du projet.

Rôles et responsabilités

Structure de Gouvernance



Rôle	Description	Responsable Hiérarchique
Chef de Projet	Responsable de la coordination globale du projet, de la planification et de la gestion des ressources.	Ash Callum, CEO
Architecte en Chef	Dirige la conception de l'architecture, assure la cohérence avec les objectifs stratégiques.	Natasha Jarson, CIO
Responsable DevOps	Gère les déploiements, l'infrastructure, et les opérations continues.	Natasha Jarson, CIO
Expert en Sécurité	Assure la mise en œuvre et la gestion des mesures de sécurité.	Natasha Jarson, CIO
Responsable Produit	Définit la vision produit, les fonctionnalités et les priorités de la roadmap.	Daniel Anthony, CPO

Analyste de Données	Analyse les données, fournit des insights et optimise les performances.	Daniel Anthony, CPO
Responsable Support Client	Fournit une assistance aux utilisateurs finaux, recueille et analyse les retours d'expérience.	Christina Orgega, CMO

Process du Projet

1. **Réunions régulières :**
    - Réunions quotidiennes (stand-up meetings) pour évaluer les progrès et résoudre les obstacles.
    - Réunions hebdomadaires pour la revue des sprints et la planification.
    - Comités de pilotage mensuels pour examiner l'avancement global et valider les décisions stratégiques.
2. **Comités de pilotage :**
    - Dirigés par Natasha Jarson et Ash Callum, ces comités valident les changements de périmètre et les décisions majeures.
3. **Répertoire de documents :**
    - Utilisation d'outils de gestion documentaire comme Confluence et Google Drive pour stocker et partager les documents de projet.
4. **Management de la configuration :**
    - Utilisation de Git pour la gestion du code source et la configuration des environnements de développement.
5. **Assurance qualité :**
    - Intégration continue et déploiement continu (CI/CD) avec des tests automatisés pour garantir la qualité du code.
6. **Procédure en cas d’escalade :**
    - Mise en place d'une procédure d'escalade pour traiter les problèmes critiques rapidement, en impliquant les niveaux hiérarchiques appropriés.
7. **Procédure en cas de changement :**
    - Tout changement doit être documenté et approuvé lors des comités de pilotage avant d'être intégré dans le projet.

Rôles et Responsabilités (RACI)

Le tableau suivant montre les rôles et responsabilités des parties prenantes clés en utilisant le modèle RACI (Responsible, Accountable, Consulted, Informed).

Activité/Processus	Responsable (R)	Accountable (A)	Consulté (C)	Informé (I)
Planification du Projet	Chef de Projet	Ash Callum	Architecte en Chef, Responsable Produit	Équipe de Développement, Équipe DevOps
Conception de l'Architecture	Architecte en Chef	Natasha Jarson	Chef de Projet, Responsable Produit	Équipe de Développement, Équipe Sécurité
Développement des Microservices	Équipe de Développement	Responsable Produit	Chef de Projet, Architecte en Chef	Équipe DevOps, Analyste de Données
Gestion de l'Infrastructure Cloud	Équipe DevOps	Responsable DevOps	Chef de Projet, Architecte en Chef	Équipe de Développement, Expert en Sécurité
Implémentation des Mesures de Sécurité	Expert en Sécurité	Natasha Jarson	Chef de Projet, Architecte en Chef	Équipe DevOps, Responsable Produit

Analyse des Données et Optimisation	Analyste de Données	Daniel Anthony	Responsable Produit, Chef de Projet	Équipe de Développement, Équipe DevOps
Support aux Utilisateurs Finaux	Responsable Support Client	Christina Orgega	Chef de Projet, Responsable Produit	Équipe DevOps, Analyste de Données

Approche architecturale

Process d'architecture

La méthode de développement d'architecture TOGAF (ou ADM pour «Architecture Development Method») décrit une méthodologie des meilleures pratiques pour le développement architectural. Néanmoins, toutes les phases ne sont pas également pertinentes pour chaque projet. Le tableau ci-dessous décrit l'utilisation de l'ADM pour ce projet spécifique.

Phase	Entrée/Sortie	Notes
Préliminaire	- Objectifs stratégiques - Cadre budgétaire - Plan stratégique	Établir les bases du projet, définir les parties prenantes et les contraintes initiales.

A — Vision de l'architecture	- Vision de l'architecture - Représentation générale des architectures initiales et cibles	Créer une vision globale et partagée de l'architecture cible.
B — Architecture business	- Architecture métier - Formalisation des exigences métier - Processus métier	Définir les processus métier et les exigences pour aligner l'architecture sur les objectifs métier.
C — Architecture des systèmes d'information	- Architecture du système - Constituants logiciels (applicatifs et données)	Spécifier les composants applicatifs et les données nécessaires à l'automatisation des processus métier.
D — Architecture technologique	- Architecture technique - Plateformes et environnements d'exécution	Décrire les infrastructures techniques supportant les applications et les données.
E — Opportunités et solutions	- Consolidation des phases B, C et D - Exigences et écarts	Identifier les opportunités d'amélioration et les solutions potentielles.
F — Planning de migration	- Planning de migration - Détails des projets de mise en œuvre	Planifier la transition vers l'architecture cible avec des projets spécifiques.
G — Gouvernance de l'implémentation	- Contrats de projets d'implémentation - Recommandations architecturales	Mettre en place des mécanismes de gouvernance pour superviser la mise en œuvre.
H — Management du changement d'architecture	- Gestion des modifications - Évaluation des demandes de changement	Gérer les évolutions et modifications de l'architecture tout au long de sa mise en œuvre.

Management des conditions requises	- Suivi des exigences - Ajustements en fonction des retours	Assurer que toutes les exigences sont bien prises en compte et ajustées si nécessaire.

Notes supplémentaires sur les phases, les itérations, etc.

- **Phases itératives :** Certaines phases, comme l'architecture des systèmes d'information et l'architecture technologique, peuvent nécessiter des itérations pour affiner les détails en fonction des retours et des tests.
- **Revue continue :** Des revues régulières doivent être effectuées après chaque phase majeure pour garantir que l'architecture reste alignée avec les objectifs stratégiques et les besoins métier.
- **Flexibilité :** L'approche doit rester flexible pour s'adapter aux changements de périmètre ou aux nouvelles opportunités identifiées en cours de projet.
- **Documentation :** Une documentation complète et à jour doit être maintenue tout au long du projet pour assurer la traçabilité et faciliter les décisions futures.

Cette approche architecturale, basée sur TOGAF ADM, assure une structure méthodique et adaptable pour le développement de l'architecture, en garantissant que toutes les étapes clés sont couvertes et que les besoins de Foosus sont pleinement intégrés dans la solution finale.

Phases de l'ADM adaptées

La méthode de développement d'architecture TOGAF (ADM) a été adaptée pour répondre aux besoins spécifiques de Foosus, une petite entreprise. Voici comment les phases ont été ajustées :

- **Phase A — Vision de l'architecture :** La vision inclut des éléments spécifiques comme la géolocalisation et les micro-services modulaires, répondant directement aux objectifs stratégiques de l'entreprise.
- **Phase B — Architecture business :** Adaptée pour formaliser les exigences métier en cycles itératifs courts, permettant une adaptation rapide aux changements de besoins.
- **Phase C — Architecture des systèmes d'information :** Focus sur la modularité et l'intégration continue pour assurer une flexibilité maximale.
- **Phase D — Architecture technologique :** Utilisation de technologies adaptées aux petites entreprises avec une forte capacité de scalabilité.

Contenu de l’architecture

Le cadre de contenu d’architecture TOGAF (ou ACF pour «Architecture Content Framework») fournit une catégorisation des meilleures pratiques pour le contenu de l’architecture. Néanmoins, tous les éléments ne sont pas également pertinents pour chaque projet. Le tableau ci-dessous décrit les zones de contenu pertinentes pour ce projet spécifique.

Zone de contenu	Entrée/Sortie	Notes
Principes, Vision et Conditions requises de l’Architecture	- Principes d'architecture - Vision architecturale - Conditions requises	Inclut les principes directeurs, la vision à long terme et les exigences de haut niveau.
Architecture Business	- Processus métier - Exigences métier	Décrit les processus et les exigences métier, ainsi que leur alignement stratégique.
Architecture des systèmes d’information — Données	- Modèle de données - Entrepôts de données	Définit la structure des données, les flux de données et les entrepôts de données nécessaires.
Architecture des systèmes d’information — Applications	- Diagrammes d'applications - Interfaces et intégrations	Détaille les applications nécessaires, leurs interactions et leurs intégrations.
Architecture technologique	- Infrastructure technologique - Plateformes d'exécution	Spécifie les composants technologiques, les plateformes et l'infrastructure nécessaires pour supporter les applications et les données.
Réalisation de l’architecture	- Plan de mise en œuvre - Gouvernance et suivi	Inclut les plans de mise en œuvre, les mécanismes de gouvernance et les mesures de suivi pour assurer la réalisation conforme de l'architecture.

Méthodologies Pertinentes et Normes de l’Industrie

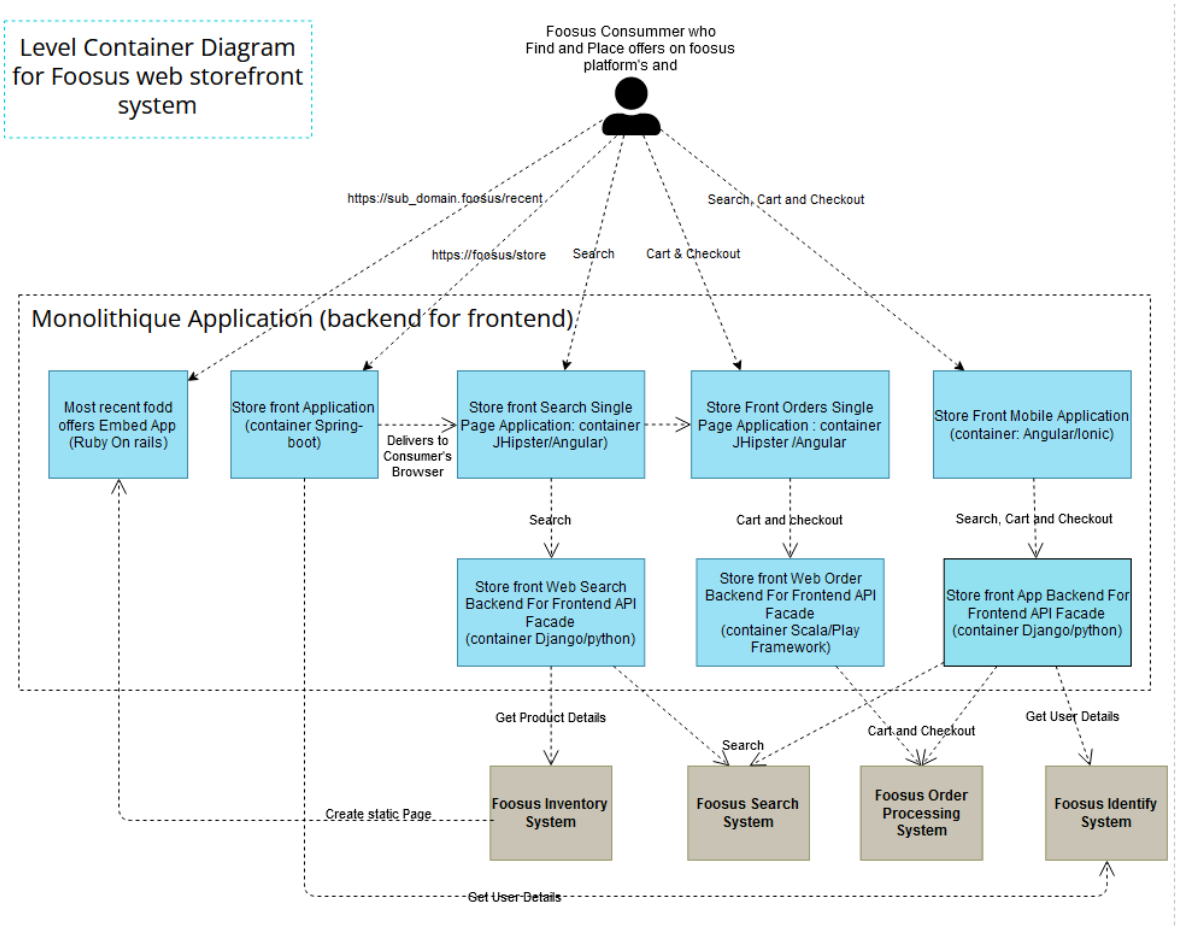
Les autres points notables relatifs à l’approche architecturale incluent :

- **Le niveau de détail :** Le travail d'architecture sera mené à un niveau stratégique, avec des segments spécifiques détaillés en fonction des besoins de Foosus.
- **La période de temps :** L'architecture couvrira une période de 3 à 5 ans, permettant des évolutions en fonction des changements du marché et des technologies.
- **Le sujet :** Les domaines couverts incluront l'infrastructure technologique, les applications métier, et les modèles de données.
- **Le niveau d’abstraction :** Une combinaison de représentations concrètes de solutions et d'architectures de référence abstraites sera utilisée.
- **La ligne de base vs la cible :** L'accent sera mis à la fois sur la documentation de l'architecture actuelle (ligne de base) et sur la proposition d'une architecture future cible. Les activités seront abordées séquentiellement, en commençant par l'évaluation de la ligne de base actuelle.
- **L’itération :** L'itération sera utilisée dans l’ADM pour permettre des ajustements continus et des améliorations basées sur les retours d'expérience.
- **Le partitionnement :** Le travail d'architecture tiendra compte des relations avec d'autres travaux d'architecture dans un environnement partitionné, assurant une intégration harmonieuse et une cohérence globale.

Architecture Source

L'architecture actuelle repose sur un modèle de conception backend pour le frontend, ce qui signifie que le backend est conçu spécifiquement pour répondre aux besoins du frontend. Cela inclut également la propagation du comportement, où les actions et les événements se propagent à travers les différentes couches de l'architecture.

Dans cette configuration, chaque conteneur représente un backend dédié à la gestion des commandes StoreFront. En pratique, ces backends sont des applications monolithiques, ce qui signifie qu'elles sont développées en tant qu'une seule et même application plutôt qu'en services indépendants. Avec le temps, ces monolithes sont devenus de plus en plus complexes et difficiles à gérer. Cette complexité accrue pose des défis importants aux équipes de développement de Foosus, rendant la maintenance, l'évolution et le débogage des applications plus laborieux et coûteux.



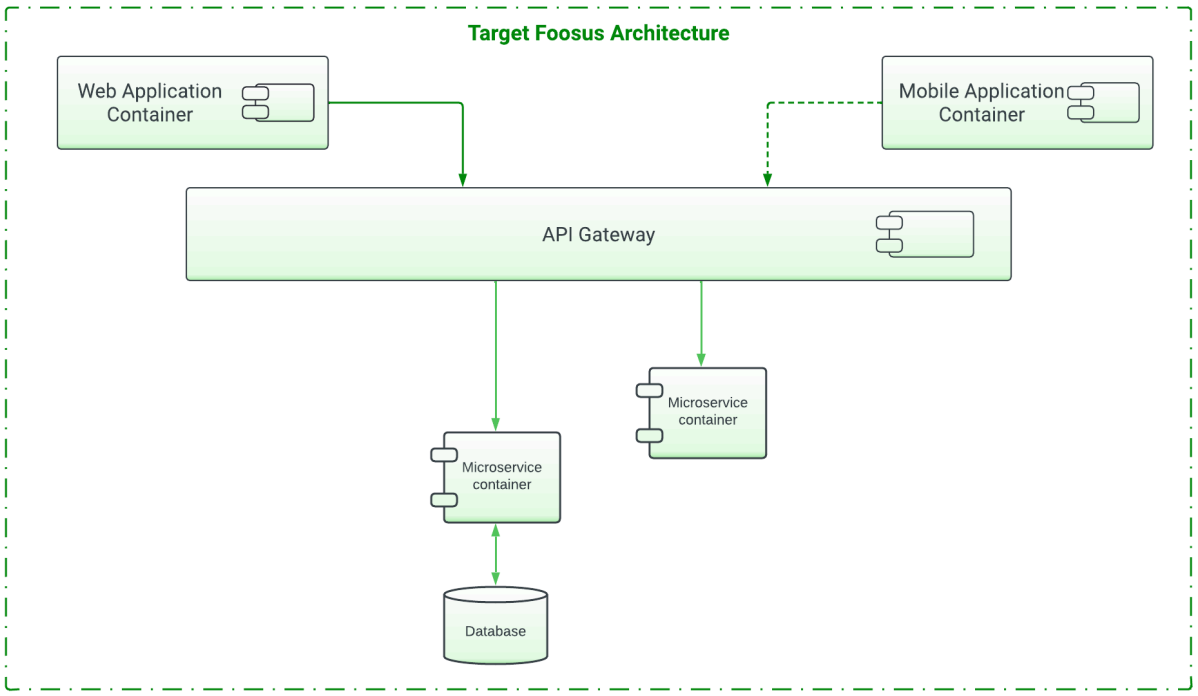
La nature monolithique de ces applications signifie que toute modification ou mise à jour nécessite une compréhension approfondie de l'ensemble du système, augmentant ainsi le risque d'introduire des erreurs. En outre, l'évolutivité est limitée, car il est difficile de déployer des parties spécifiques de l'application sans affecter l'ensemble du système.

Pour résumer, l'architecture actuelle, bien que basée sur un modèle de design sophistiqué, souffre de la lourdeur et de la complexité des applications monolithiques, rendant la tâche des équipes de Foosus plus ardue.

Base d’architecture

L'objectif ici est d'adopter une architecture modulaire en créant des microservices qui décomposeront les besoins métier en plusieurs applications, tout en respectant les principes SOLID.

Target Architecture Base



Il est important de comprendre les avantages que présentent les micro-services. Voici les avantages :

- Les services peuvent évoluer indépendamment selon les besoins des utilisateurs.
- Chaque service peut être mis à l'échelle individuellement pour répondre à la demande spécifique des utilisateurs.
- Au fil du temps, les cycles de développement deviennent plus rapides, car les nouvelles fonctionnalités peuvent être mises en production plus rapidement.
- Les services sont isolés, ce qui les rend plus résilients aux défaillances.
- Une défaillance d'un service unique n'entraîne pas la défaillance de l'ensemble de l'application.
- Les tests deviennent plus cohérents grâce au développement piloté par le comportement (BDD).

En adoptant cette approche, nous visons à améliorer la maintenabilité, la scalabilité et la résilience de nos systèmes.

Architecture Cible

Nous avons adapté notre architecture source à notre nouvelle base d'architecture en respectant le principe de micro-services. Cette transition permettra de décomposer les applications monolithiques en composants plus petits, indépendants et plus faciles à gérer, alignant ainsi notre infrastructure avec les besoins actuels et futurs de l'entreprise.

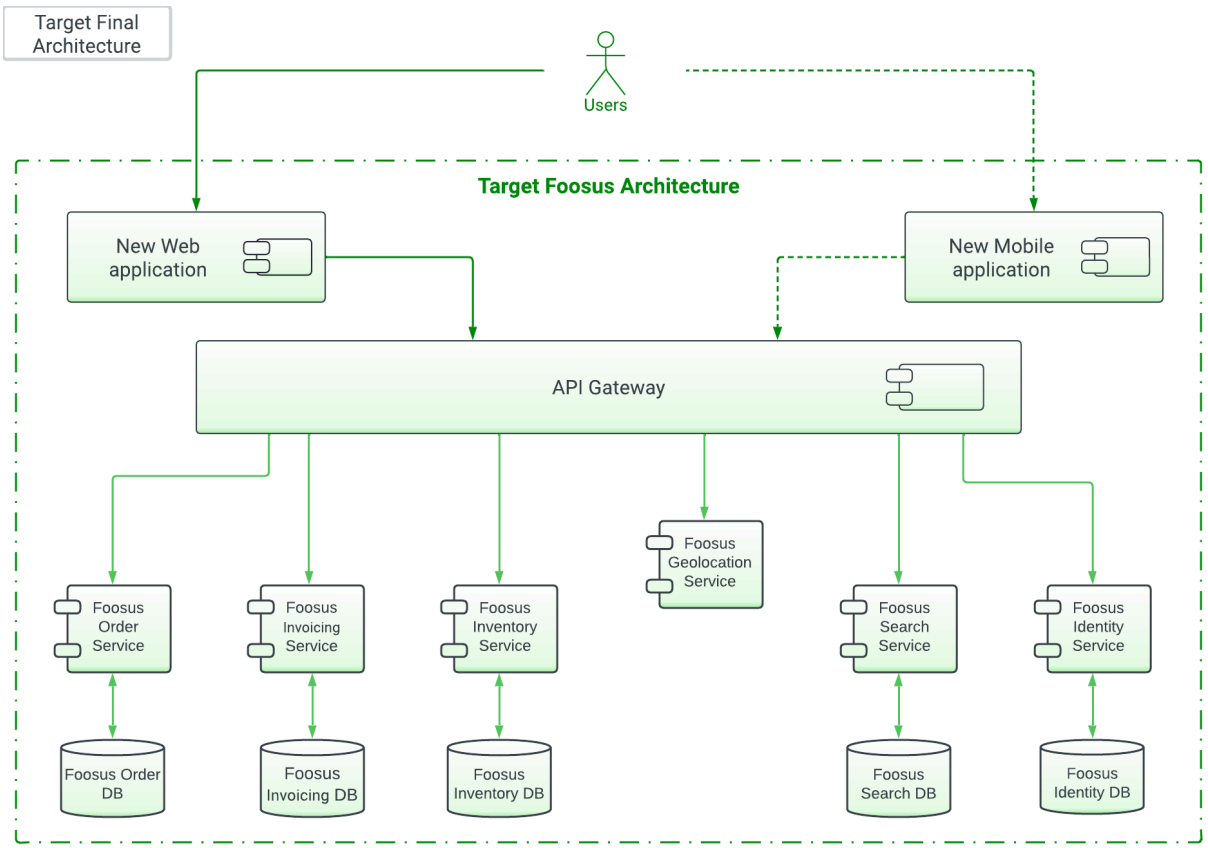
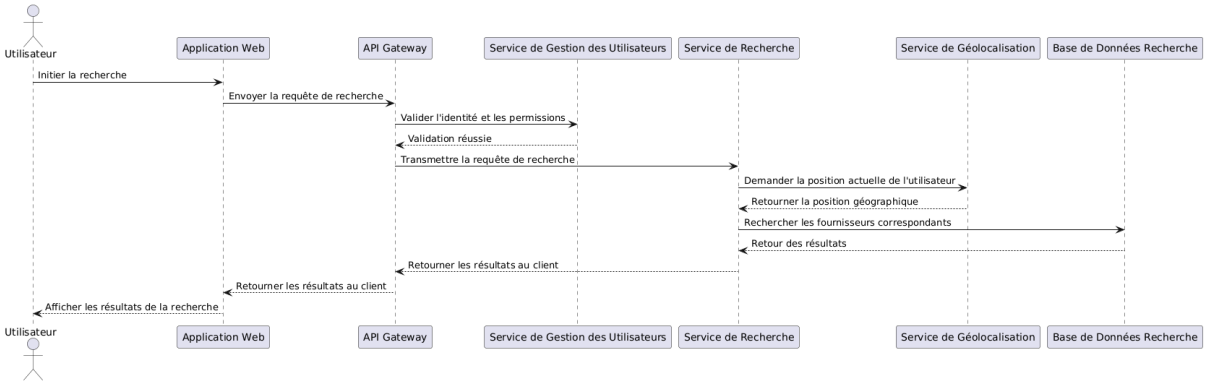


Diagramme de séquence



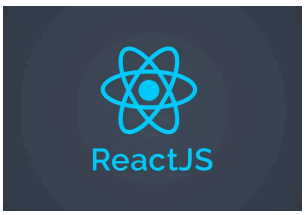
Technologies Utilisées

L'adoption de technologies appropriées est cruciale pour atteindre les objectifs de l'architecture modulaire basée sur les microservices. Voici un aperçu des technologies clés qui seront utilisées :

Backend

- **Spring Boot (Java) :**
  - **Robustesse :** Spring Boot est un framework mature et largement adopté, reconnu pour sa robustesse et sa stabilité.
  - **Micro-services :** Conçu pour faciliter la création de micro-services, avec des fonctionnalités intégrées pour la gestion des configurations et des dépendances.
  - **Communauté :** Une large communauté et une excellente documentation, facilitant le support et la résolution de problèmes.
  - **Sécurité :** Intégration facile avec des standards de sécurité et des mécanismes d'authentification.

Frontend



- **React.js :** Bibliothèque JavaScript pour construire des interfaces utilisateur interactives.
    - **Performances :** Virtual DOM et rendu efficace, améliorant les performances des applications interactives.
    - **Écosystème :** Un écosystème riche avec des outils comme Redux pour la gestion de l'état et une forte adoption dans l'industrie.
    - **Réutilisabilité :** Composants réutilisables qui facilitent la maintenance et la scalabilité du code.

      Support : Soutenu par Facebook et une grande communauté, garantissant une évolution continue et un bon support.

- **Flutter (Dart)** :

    - **Cross-Platform** : Permet le développement d'applications natives pour iOS et Android avec une base de code unique.
    - **Performances :** Compilé en code natif, offrant des performances proches des applications natives.
    - **UI Riches :** Prend en charge le développement d'interfaces utilisateur modernes et animées, cohérentes entre les plateformes.
    - **Communauté et Plugins :** Large collection de plugins et une communauté en croissance rapide, offrant un support solide pour de nombreuses fonctionnalités natives.

Bases de Données

- **MongoDB :**
  - MongoDB est une base de données NoSQL qui permet de stocker des données non structurées et semi-structurées de manière flexible. Elle est adaptée aux applications nécessitant une scalabilité horizontale et une gestion de données évolutive.

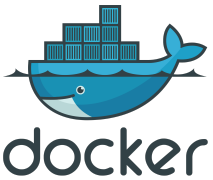
- **PostgreSQL :**
  - PostgreSQL est une base de données relationnelle open source connue pour sa robustesse, sa conformité ACID, et ses capacités d'extension. Elle est idéale pour des applications nécessitant des transactions complexes et une intégrité des données.

Communication entre Services

- **Kafka :**
  - Apache Kafka est une plateforme de streaming distribuée qui permet de gérer le flux de données en temps réel. Il est adapté aux systèmes nécessitant une grande scalabilité et une tolérance aux pannes, en particulier pour les applications de traitement de données en temps réel.

- **RabbitMQ :**
  - RabbitMQ est un message broker qui facilite la communication asynchrone entre les micro-services. Il est simple à configurer et offre des fonctionnalités avancées de routage des messages, ce qui est essentiel pour des systèmes distribués.

Infrastructure et Déploiement



- **Docker :**
    - Docker permet de conteneuriser les applications, assurant ainsi leur portabilité et une scalabilité simplifiée. Les conteneurs garantissent que les applications fonctionnent de manière cohérente dans différents environnements.

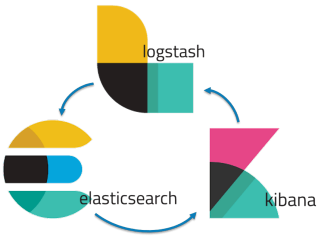
- **Kubernetes :**
    - Kubernetes est un orchestrateur de conteneurs qui automatise le déploiement, la gestion et la mise à l'échelle des applications conteneurisées. Il est indispensable pour gérer des clusters de conteneurs et assurer une haute disponibilité et une tolérance aux pannes.
- **AWS/GCP/Azure :**
    - Les services cloud comme AWS, GCP, et Azure offrent une infrastructure scalable et fiable, avec une multitude de services gérés pour le déploiement, la gestion et la sécurisation des applications. Le choix du fournisseur dépendra des besoins spécifiques et des accords commerciaux existants.

Sécurité

- **OAuth 2.0 :**
    - OAuth 2.0 est un standard d'autorisation sécurisé largement adopté pour permettre aux applications d'accéder aux ressources sans partager les informations de connexion des utilisateurs. Il est essentiel pour protéger les API et les données utilisateur.

- **JWT (JSON Web Tokens) :**
    - Les JWT sont utilisés pour créer des tokens d'accès sécurisés et stateless, facilitant l'authentification et l'autorisation dans des environnements distribués. Ils permettent une vérification rapide et sécurisée des utilisateurs.

- **Prometheus :**
    - Prometheus est un système de monitoring et d'alerte open source conçu pour la scalabilité et la fiabilité. Il est capable de surveiller des milliers de services et de serveurs, ce qui est crucial pour des applications en production.



- **ELK Stack (Elasticsearch, Logstash, Kibana) :**
    - La suite ELK offre une solution complète pour la gestion et l'analyse des logs. Elasticsearch permet une recherche rapide, Logstash collecte et transforme les données, et Kibana offre des visualisations puissantes. Cette combinaison est idéale pour le suivi et le diagnostic des performances des applications.

Justification des Choix Technologiques

Les technologies listées ci-dessus ont été choisies pour leur capacité à :

- **Scalabilité :** Supporter une augmentation du nombre d'utilisateurs et de transactions.
- **Performance :** Fournir des temps de réponse rapides et une haute disponibilité.
- **Sécurité :** Assurer la protection des données et des communications.
- **Flexibilité :** Permettre des itérations rapides et des déploiements fréquents.
- **Compatibilité :** Intégration facile avec les systèmes existants et futurs.

Recommandations CI/CD (pipeline)

Pour assurer une intégration et un déploiement continus efficaces, les outils et processus recommandés sont :

- Outils CI/CD : Jenkins, GitLab CI, CircleCI.
- Étapes du pipeline :
    1. Build : Compilation du code et génération d'artefacts.
    2. Test : Exécution de tests unitaires et d'intégration.
    3. Deploy : Déploiement automatisé dans des environnements de staging.
    4. Monitor : Surveillance des performances et des logs post-déploiement.

Plan de transition sans interruption

Introduction

Le plan de transition vise à garantir que la migration vers la nouvelle architecture se déroule sans interruption de service pour les utilisateurs finaux. Cela implique la coexistence des deux systèmes pendant une période de transition, l'utilisation de techniques de déploiement avancées, et des stratégies de gestion des risques pour minimiser les impacts potentiels.

Techniques de Déploiement

Déploiement Canari

Le déploiement canari consiste à déployer progressivement la nouvelle version du service à un sous-ensemble d'utilisateurs avant de l'étendre à l'ensemble de la base d'utilisateurs. Cela permet de détecter et de résoudre les problèmes potentiels avec un impact limité.

- Étape 1 : **Sélection d'un sous-ensemble d'utilisateurs :** Identifier un groupe représentatif d'utilisateurs pour tester la nouvelle architecture.

- Étape 2 : Déploiement initial : **Déployer la nouvelle version à ce sous-ensemble** d'utilisateurs tout en maintenant l'ancienne version pour le reste des utilisateurs.
- Étape 3 : **Surveillance et ajustements :** Surveiller les performances et les retours des utilisateurs, effectuer les ajustements nécessaires.
- Étape 4 : Extension progressive : **Étendre graduellement** le déploiement à l'ensemble des utilisateurs après validation des résultats.

Blue-Green Deployment

Le Blue-Green Deployment maintient deux environnements de production identiques (Blue et Green). Une nouvelle version est déployée dans l'environnement inactif, puis la bascule vers le nouvel environnement se fait instantanément.

- Étape 1 : Préparation des environnements : Maintenir deux environnements de production identiques, Blue et Green.
- Étape 2 : Déploiement sur l'environnement inactif : Déployer la nouvelle version sur l'environnement inactif (Green) tandis que l'environnement actif (Blue) continue de servir les utilisateurs.
- Étape 3 : Tests et validation : Effectuer des tests approfondis sur l'environnement Green pour valider les fonctionnalités et la stabilité.
- Étape 4 : Bascule vers l'environnement Green : Une fois validé, basculer instantanément le trafic utilisateur de l'environnement Blue vers l'environnement Green.
- Étape 5 : Surveillance post-bascule : Surveiller les performances et résoudre rapidement les problèmes éventuels.

Stratégies de Gestion des Risques

Plan de Repli

Un plan de repli (rollback) doit être en place pour revenir à l'ancienne version en cas de défaillance majeure.

- Détection de problème : Utiliser des outils de monitoring pour détecter rapidement les anomalies post-déploiement.
- Exécution du rollback : Avoir des scripts prêts pour un retour rapide à l'ancienne version si nécessaire.

Plan de travail

Cette section décrit toutes les activités et tous les livrables du travail d'architecture.

Élément de travail 1 : Analyse Initiale

Activités	Livrables
Collecte des besoins auprès des parties prenantes.	Déclaration de travail d'architecture : Définit la portée et l'approche utilisées pour terminer un cycle de développement d'architecture. Sert de base pour le contrôle du succès du projet d'architecture et peut-être considéré comme un accord contractuel entre les différentes parties.
Analyse des systèmes existants et identification des points faibles.	Rapport d'analyse initiale : Document détaillant les besoins, les points faibles des systèmes actuels, et les recommandations pour l'architecture cible.
Évaluation des technologies actuelles et émergentes pertinentes pour le projet.	

Élément de travail 2 : Conception de l'Architecture

Activités	Livrables
Développement de la vision architecturale.	Document de définition d'architecture : Fournit une vue qualitative de la solution et communique l'intention des architectes. Contient les artefacts architecturaux de base et les informations connexes importantes, couvrant tous les domaines de l'architecture (entreprise, données, application et technologie) et tous les états pertinents de l'architecture (base, transition et cible).
Conception des modèles d'architecture cible.	Diagrammes d'architecture : Représentations visuelles des composants et de leurs interactions.

Définition des interfaces et des interactions entre les composants.	

Élément de travail 3 : Spécification des Conditions Requises pour l'Architecture

Activités	Livrables
Définition des exigences quantitatives pour la mise en œuvre de l'architecture.	Spécification des conditions requises pour l'architecture : Ensemble d'énoncés quantitatifs décrivant ce qu'un projet de mise en œuvre doit faire pour se conformer à l'architecture. Complément au document de définition d'architecture.
Validation des spécifications avec les parties prenantes.	

Élément de travail 4 : Contrats d'Architecture

Activités	Livrables
Élaboration des contrats d'architecture avec les partenaires de conception et développement.	Contrat d'architecture avec la conception et le développement : Déclaration d'intention sur la conception et le développement de l'architecture d'entreprise par des organisations partenaires internes ou externes.
Négociation des accords de niveau de service (SLA).	Contrat d'architecture avec les utilisateurs professionnels : Déclaration d'intention de se conformer à l'architecture d'entreprise, délivrée par les utilisateurs métiers.

Élément de travail 5 : Développement et Implémentation

Activités	Livrables
Développement des microservices conformément aux spécifications.	Code source des microservices : Référentiel de code conforme aux standards de développement définis.
Intégration continue et déploiement continu (CI/CD) des composants.	Scripts de déploiement : Scripts automatisés pour le déploiement des microservices sur l'infrastructure cible.
Mise en place de l'infrastructure de monitoring et de logging.	Infrastructure de monitoring et de logging : Outils et configurations pour la surveillance continue et la gestion des logs.

Élément de travail 6 : Tests et Validation

Activités	Livrables
Conduite de tests unitaires, d'intégration, de performance et de sécurité.	Rapports de tests : Documentation des résultats de tests unitaires, d'intégration, de performance et de sécurité.
Validation des fonctionnalités avec les parties prenantes.	Plan de validation : Documentation des critères de validation et des résultats obtenus lors des tests de charge et en environnement de production simulé.
Exécution des tests de charge et des tests en environnement de production simulé.	

Élément de travail 7 : Formation et Transition

Activités	Livrables
Formation des équipes internes sur la nouvelle architecture et les technologies utilisées.	Documentation de formation : Manuels et guides pour les équipes de développement et d'exploitation.

Documentation des processus opérationnels et des guides utilisateurs.	Plans de transition : Stratégie détaillée pour la migration des systèmes existants vers la nouvelle architecture.
Transition progressive des systèmes existants vers la nouvelle architecture.	

Élément de travail 8 : Mise en Production et Support Post-Lancement

Activités	Livrables
Mise en production de la nouvelle architecture.	Documentation de mise en production : Check-lists et procédures pour la mise en production.
Surveillance post-lancement pour identifier et résoudre les problèmes éventuels.	Rapports de surveillance post-lancement : Suivi des performances et des incidents après le lancement.
Réception des feedbacks et ajustements continus.	Plan de support post-lancement : Stratégie de support et de maintenance continue.

Plan de Communication

Évènements	Canaux	Formats	Contenu
Réunions de lancement	En personne/visioconférence	Présentations PowerPoint	Objectifs et avancement du projet
Réunions hebdomadaires de suivi	En personne/visioconférence	Présentations PowerPoint	Décisions prises et actions à suivre
Comités de pilotage mensuels	En personne/visioconférence	Présentations PowerPoint	Changements de périmètre et ajustements nécessaires
Séances de validation avec les parties prenantes	En personne/visioconférence	Présentations PowerPoint	Suivi des livrables et des jalons
Emails	Email	Emails récapitulatifs	Résumé des réunions et décisions importantes
Intranet d'entreprise	Intranet	Tableaux de bord en ligne	Suivi des progrès et documentation à jour
Outils de gestion de projet (e.g., JIRA, Confluence)	Outils de gestion de projet	Documents Word/PDF, Tableaux de bord en ligne	Planification des tâches et suivi des actions

Durée et Effort

Phase	Durée Estimée	Effort Estimé
Analyse Initiale	1 mois	4 personnes x 1 mois = 4 PM (Person-Months)
Conception de l'Architecture	2 mois	4 personnes x 2 mois = 8 PM
Spécification des Conditions Requises	1 mois	3 personnes x 1 mois = 3 PM
Élaboration des Contrats	1 mois	2 personnes x 1 mois = 2 PM
Développement et Implémentation	4 mois	6 personnes x 4 mois = 24 PM
Tests et Validation	2 mois	4 personnes x 2 mois = 8 PM
Formation et Transition	1 mois	3 personnes x 1 mois = 3 PM
Mise en Production et Support Post-Lancement	2 mois	3 personnes x 2 mois = 6 PM
Total	**14 mois**	**58 PM**

Collaboration

Équipe de Développement	Participation active dans la conception et le développement.
Équipe DevOps	Gestion de l'infrastructure et des déploiements.
Experts Sécurité	Implémentation et gestion des mesures de sécurité.
Product Managers	Définition des fonctionnalités et priorités.
Analystes de Données	Analyse des données et optimisation des performances.
Support Client	Assistance aux utilisateurs finaux et recueil des feedbacks.
Partenaires Externes	Intégration avec les services logistiques, les plateformes de paiement, etc.

Risques et facteurs de réduction

Analyse des risques

Identification des Risques

ID	Risque	Conséquences
1	Dépassement du budget	- Projets retardés ou annulés - Réduction de la portée du projet - Perte de crédibilité
2	Exigences imprécises ou en constante évolution	- Délais prolongés - Coûts supplémentaires - Insatisfaction des parties prenantes
3	Vulnérabilité des données	- Violations de données - Perte de confiance des clients - Amendes réglementaires
4	Complexité accrue de l'architecture	- Maintenance difficile - Risque accru de dysfonctionnements - Délais de développement plus longs
5	Retards dans les délais de livraison	- Dépassement du budget - Insatisfaction des clients - Perte de parts de marché

Stratégies de réduction

ID	Probabilité (P)	Gravité (G)	Criticité (C) = P x G	Responsable	Facteur de réduction
1	Moyenne	Haute	15	Chef de Projet	Bonne planification budgétaire et suivi régulier des dépenses
2	Moyenne	Haute	15	Chef de Projet	Validation continue des exigences avec les parties prenantes et gestion des changements
3	Moyenne	Très haute	20	Expert en Sécurité	Mise en place de mesures de sécurité robustes, chiffrement des données et audits réguliers
4	Moyenne	Moyenne	9	Architecte en Chef	Validation de l'architecture par des experts externes, adoption de standards de codage
5	Moyenne	Haute	15	Chef de Projet	Suivi rigoureux du planning, gestion proactive des risques et des dépendances

Hypothèses

Ces hypothèses doivent être vérifiées régulièrement tout au long du projet. Si l'une d'elles se révèle incorrecte, cela peut entraîner des ajustements dans la planification et la gestion du projet pour garantir le succès de l'initiative.

ID	Hypothèse	Impact	Propriétaire
1	Les ressources nécessaires (humaines et financières) seront disponibles tout au long du projet	Une disponibilité continue des ressources garantit l'achèvement du projet dans les délais prévus et prévient les retards et les dépassements de budget.	Chef de Projet
2	Les parties prenantes resteront engagées et fourniront des retours rapides et constructifs	Un engagement continu des parties prenantes assure la pertinence et l'acceptation des livrables, facilitant une prise de décision rapide et efficace.	Responsable Produit
3	Les technologies sélectionnées seront compatibles avec les systèmes existants	La compatibilité technologique réduit les risques de réintégration, minimise les coûts supplémentaires et assure une transition en douceur.	Architecte en Chef
4	Les besoins des utilisateurs finaux ne changeront pas significativement au cours du projet	Une stabilité des besoins utilisateurs facilite la gestion du périmètre et aide à respecter les délais et les budgets.	Responsable Produit

Critères d’acceptation et procédures

Métriques et KPIs

Métrique	Technique de mesure	Valeur cible	Justification
Nombre d'adhésions d'utilisateurs par jour	Monitoring	+10%	Augmenter la base d'utilisateurs
Adhésion de producteurs alimentaires	Monitoring	4/mois	Augmenter le réseau de fournisseurs
Délai moyen de parution	Monitoring	Moins d’une semaine	Améliorer la réactivité
Taux d'incidents de production P1	Monitoring	Moins de 1/mois	Réduire les interruptions de service

Procédure d’acceptation

Le document devra être approuvé en comité de pilotage puis déposé signé sur le répertoire **GIT** dédié.

Approbations signées

Nom	Date	Signature
Ash Callum, CEO		
Daniel Anthony, CPO		
Natasha Jarson, CIO		