DECLARATION DE TRAVAIL D'ARCHITECTURE

Nom de l'entreprise : Foosus
Nom du projet : Développement d'entreprise
Adresse :
Email :
Tel :

Table des matières

1. OBJET DU DOCUMENT

Ce document est une déclaration de travail d'architecture pour le projet de développement de l'entreprise FOOSUS. Il spécifie une articulation et une direction d'architecture qui permet à l'entreprise de développer les capacités nécessaires pour réussir sur le marché ; un état cible d'architecture vers lequel l'entreprise doit itérer ; il décrit un processus et une approche d'architecture sur mesure qui conviennent aux structures des équipes de l'entreprise et à la topologie de son organisation.

2. DECLARATION DE TRAVAIL D'ARCHITECTURE

2.1. Requête du projet et contexte

Foosus est une start-up de 3 ans dans le secteur de l'alimentation durable qui souhaite construire une solution géociblée avec une nouvelle architecture. Avec un volume de dette technique très élevé et un manque de cohérence qui ont commencé récemment à impacter de manière significative le développement de fonctionnalités, l'entreprise a besoin de frontières claires pour pouvoir développer une plate-forme qui permette de l'innovation rapide et se mette à l'échelle du business.

La plate-forme actuelle de Foosus a atteint un point au-delà duquel elle ne peut plus soutenir les projets de croissance et d'expansion de l'entreprise. Après plusieurs années de développement, sa solution technique complexe n'évolue plus au rythme de l'activité et risque d'entraver sa croissance. Les études de marché et les analyses commerciales montrent que ses clients souhaitent acheter local et soutiennent les producteurs locaux. Elle veut s'appuyer sur les connaissances acquises ces trois dernières années et créer une plateforme qui mettra en contact des consommateurs avec des producteurs et des artisans locaux dans toutes les catégories de besoins.

2.2. Description du projet et périmètre

Dans le cadre du développement de l'entreprise FOOSUS, l'entreprise souhaite développer une plate-forme de commerce électronique sécurisé polyvalente pour faire passer l'entreprise à un niveau supérieur. Le projet consiste à développer une nouvelle architecture tenant compte des différents objectifs de l'entreprise dont les objectifs Business, les contraintes organisationnelles, fonctionnelles et techniques.

3. OJECTIFS ET PERIMETRE

3.1. Objectifs

Les principaux objectifs de l'entreprise sont les suivants.

1. Tirer parti de la géolocalisation pour relier des fournisseurs et des consommateurs et pour proposer des produits disponibles à proximité des lieux de résidence de ces derniers.
2. L'architecture devra être évolutive pour permettre à nos services de se déployer sur diverses régions à travers des villes et des pays donnés.
3. La solution doit être disponible pour nos fournisseurs et nos consommateurs, où qu'ils se trouvent. Cette solution doit être utilisable avec des appareils mobiles et fixes. Elle doit tenir compte des contraintes de bande passante pour les réseaux cellulaires et les connexions Internet haut débit.
4. La solution doit pouvoir prendre en charge différents types d'utilisateurs (par exemple, fournisseurs, back-office, consommateurs) avec des fonctionnalités et des services spécifiques pour ces catégories.
5. Assurer un service en continue sans interruption du système.

3.2. Périmètre

Le projet consiste à développer une nouvelle architecture pour développer une nouvelle plate-forme de commerce électronique sécurisée pour l'entreprise FOOSUS. Ayant déjà une plate-forme existante qui ne répond plus aux exigences de l'entreprise car peu évolutive, l'entreprise souhaite ne plus investir dans celle-ci mais la conserver en mode maintenance. Donc, aucune nouvelle fonctionnalité ne sera développée.
La nouvelle architecture sera construite avec en fonctions des technologies actuelles et avec la capacité de s'adapter à de nouvelles technologies.

3.3. Parties prenantes, préoccupations et vision

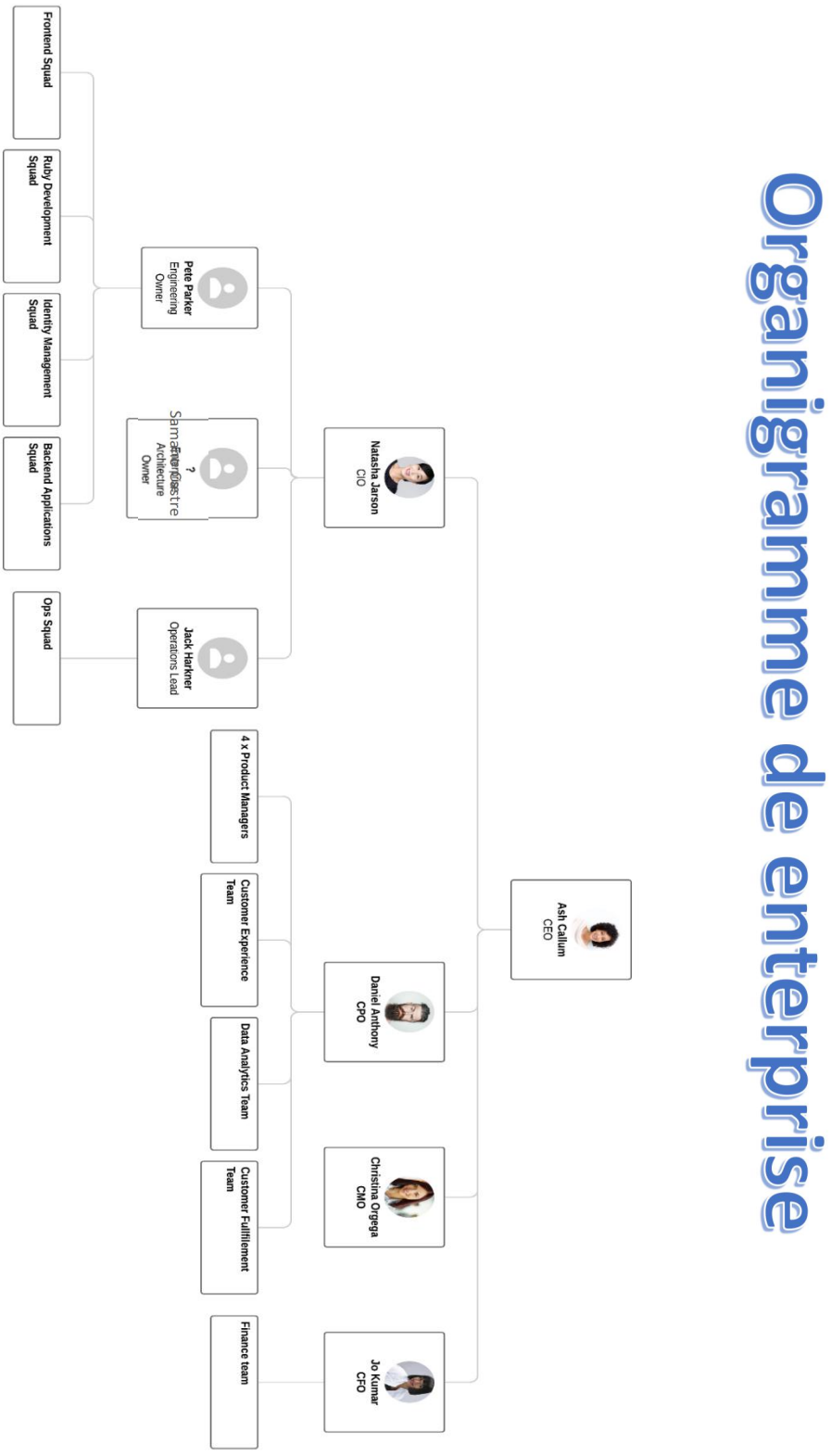
Le tableau suivant montre les parties prenantes du projet, leurs préoccupations, et la façon dont le travail d'architecture répondra à leurs préoccupations par l'expression de plusieurs visions.

Partie prenante	Fonction	Préoccupation	Vision
Ash Callum	CEO : Chief Executive Officer En Français, PDG (Président-Directeur Général)	Taux d'inscriptions des utilisateurs	Rentabilité de la solution
Natacha Jarson	CIO : Chief Information Officer En Français, Directeur des systèmes d'information	Innovation technique dans le périmètre de l'architecture d'entreprise	- Stabilité - Disponibilité - sécurité - performance
Daniel Anthony	CPO : Chief Product Officer En Français, Directeur Produit	Innovation technique dans le périmètre de l'architecture d'entreprise	- Stabilité - Disponibilité - sécurité - performance - Évolutivité
Christina Orgega	CMO : Chief Marketing Officer En Français, Directeur Marketing	Visibilité de la plate-forme	- disponibilité - stabilité
Pete Parker	Resp. Ingénierie	- Processus techniques - Processus d'ingénierie - La veille technologique	- Stabilité - Disponibilité - sécurité - performance - Évolutivité - Qualité
Joe Kumar	CFO : Directeur financier	Coûts de la solution	- Rentabilité - Le coût
Samano CASTRE	Architecte logiciel	Viabilité du projet	Les principes architecturaux
Jack Harker	Operations Lead En Français, Responsable des opérations	Coût de la solution	- Rentabilité - Le coût

4. RÔLES ET RESPONSABILTES

4.1. Structure de gouvernance

Dans cette section nous allons présenter les rôles des responsables de l'entreprise Foosus.



Le tableau ci-dessous décrit l'organigramme de l'entreprise Foosus :

Employé	Fonction	Description
Ash Callum	CEO : Chief Executive Officer	En français PDG (Président-Directeur Général), est en charge à la fois de la vision stratégique et de son application opérationnelle.
Natacha Jarson	CIO : Chief Information Officer	En français DSI (Directeur des systèmes d'information), est responsable de la gestion, de la mise en œuvre et du bon fonctionnement des technologies de l'information et de l'informatique.
Daniel Anthony	CPO : Chief Product Officer	En français Directeur produit, est responsable du département produit. Concrètement, il se charge de la définition et de l'organisation de l'ensemble de la stratégie produit et orchestre les équipes qui y travaillent. Il doit s'assurer que le Produit réponde aux besoins des clients, tout en gardant en tête la stratégie business de l'entreprise.
Christina Orgega	CMO : Chief Marketing Officer	En français (Directeur marketing), Elle définit la stratégie marketing de l'entreprise et de coordonner toutes les équipes de la direction marketing.
Pete Parker	Resp. Ingénierie	En français, (Responsable Ingénierie), est le responsable qui élabore, pour le compte de son entreprise, des applications logicielles innovantes et efficaces, tout en tenant compte des contraintes temporelles, financières et sécuritaires imposées par ses responsables hiérarchiques et par le projet lui-même.
Joe Kumar	CFO : Directeur financier	En français (Directeur Financier), est responsable de la gestion de la trésorerie, de la dette et des analyses financières et fiscales de son entreprise. Il rend compte de ces chiffres aux dirigeants et propose des stratégies : plans de financement, investissements etc.
Samano CASTRE	Architecte logiciel	En français (Responsable Architecture Entreprise), il a pour rôle de conseiller son DSI et dirigeants sur la stratégie digitale de l'entreprise, mais aussi sur l'innovation et les transformations à mettre en œuvre.
Jack Harker	le Operations Lead	En français (Responsable des opérations) est responsable de la stratégie opérationnelle de l'entreprise au sein de laquelle il opère. Il est en charge d'optimiser les chiffres d'affaires et marges, mais également de participer au développement de l'activité en anticipant les évolutions du marché.

4.2. Responsabilités (RACI)

Un ensemble de responsabilités ont été définies. Et celles-ci seront confiées aux différents acteurs du projet. Pour la présentation des responsabilités/Rôle, on s'appuie sur la matrice RACI représentée par le tableau ci-dessous :

R.A.C.I. est un acronyme qui signifie :

1. R- Responsible : les personnes chargées de réaliser la tâche
2. A - Accountable : les autorités chargées de valider le travail
3. C - Consulted : les personnes à consulter
4. I - Informed : les personnes à informer

R Responsible A Accountable C Consulted I Informed	Directeur Général	Resp. Ingénierie	Architecte Logiciel	Directeur SI	Resp. Validation Qualification	Scrum Master	UX/UI Designer	Dév. Full-Stack	Dév. mobile	Chef de produit
Déclaration de travail d'architecture	A	C	R	I						
Spécification des conditions requises pour l'architecture	A	C	R	I						
Contrat d'architecture avec les utilisateurs business	A	C	R	I						
Contrat d'architecture avec les fonctions de développement et Design	A	C	R	I						
Cahiers des charges fonctionnel		A C	R							
Plan de gestion des parties prenantes		A	R							
Recueil des besoins client		R A	I							C
Réalisation des User stories		R A	C					C	I	
Charte graphique		A			I		R			
Cahiers des charges technique		R A			I			C	C	
Réunion du comité de pilotage		R A			I					
Réunion de lancement du projet		R A			I					
Réunions d'avancement du projet		R A								
Réunion de clôture du projet		R A			I					
Coordonner l'équipe de développement		R A	C							
Développement Back-End		A	I		I	I		R	I	
Développement Front-End(web)		A	I		I	I		R		

Développement Front-End(mobile)		A	I		I	I		C	R	
Organisation des tests		A			R			C	C	
Validation des tests		A			R			C	C	
Livraison version		R						C	C	
Correction des anomalies		A			I			R	R	
Livraison patch		A	C					R	R	

4.3. Process du projet

4.3.1. Réunions régulières

Dans le cadre de ce projet de développement d'une nouvelle plate-forme commerciale un ensemble de réunions professionnelles est nécessaires, telles que : La réunion du comité de pilotage, la réunion de lancement du projet, la revue de projet, point d'avancement, et des réunions de clôtures.

4.3.2. Comité de pilotage

Pour assurer le bon déroulement des opérations en fonctions des objectifs de ce projet, nous allons créer un comité de pilotage.
Ce comité se constituera de : du CIO (directeur principal de l'information), du CPO (Chef de produit) et du responsable ingénierie.

4.3.3. Intégration continue

Intégration continue ou (CI continious Integration) regroupe un ensemble de pratiques visant à accélérer le développement d'une application tout en garantissant la qualité du code. Ces consistent à tester de manière automatisée chaque révision de code avant de le déployer en production.
De nombreux outils permettent d'assurer l'intégration continue tels que : Jenkins, Travis CI, gitlab CI,Bamboo, TeamCity, etc…

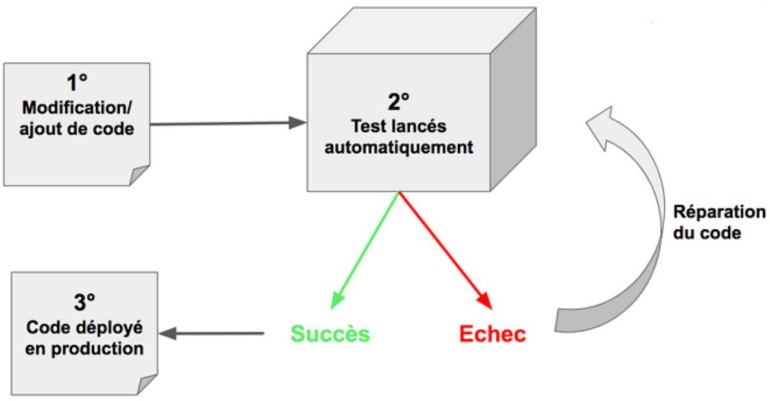


Figure 1 - illustration intégration continue

4.3.4. Procédure en cas de changement

Un processus de gestion du changement est une série d'étapes pour introduire et livrer une transformation**.**

Il existe alors 3 types de changements :

1. **Changement standard :** Changement qui représente peu de risques. Ce type de changement est pré-autorisé, voire automatisé.
2. **Changement urgent :** il s'agit ici de résoudre des erreurs, des incidents, mais aussi de s'adapter à des circonstances particulières. Il doit être mis en œuvre assez rapidement.
3. **Changement normal :** C'est le type de changement qui suit la procédure classique. Contrairement aux changements standards, il suit une chaîne d'approbation et de validation.

Pour tout changement dans le cadre de ce projet, nous allons procéder ainsi :

1. **Objectifs, Périmètre et Risques :** La règle des 7R nous permettra de définir les objectifs du changement et établir son périmètre. Cette règle consiste à se poser les 7 questions suivantes :
   a) **Requis** : qui a requis ce changement ?
   b) **Raison** : quelle est la raison de ce changement ?
   c) **Retour** : quel est le retour attendu de ce changement ? Quels sont les bénéfices attendus ?
   d) **Risque** : quels sont les risques encourus à la réalisation ou la non-réalisation de ce changement ?
   e) **Ressources** : quelles sont les ressources nécessaires pour réaliser ce changement ? De quoi aurons-nous besoin ?
   f) **Responsable** : qui est responsable de la réalisation de ce changement ?
   g) **Relation** : quelle relation avec d'autres changements ? Existe-t-il des dépendances ?

2. **Catégorisation :** Catégoriser un changement consiste à lui affecter une priorité. La méthode consiste à déterminer la priorité d'un changement grâce à une matrice prenant en compte l'impact et l'urgence de ce changement. Voir figure 2 ci-dessous.

Figure 2 - Matrice Impact/Urgence

3. **Planification :** Il s'agit d'établir un calendrier de changements autorisés. Ce plan de changement doit être en fonction des besoins du business en évitant que les changements s'interfacent avec d'autres processus de changement.

4. **Plan de remédiation :** On parle aussi de Backout Plan, Il s'agit de prévoir un plan de retour dans le cas où le déploiement se passe mal. Dans le cadre de ce projet, nous allons toujours réaliser un sauvegarde du système avant d'y apporter tout changement, afin de pouvoir revenir à l'état initial.

5. APPROCHE ARCHITECTURALE

5.1. Process d'architecture

Dans le cadre de ce projet, nous allons utiliser TOGAF. C'est un recueil de bonnes pratiques qui apporte un cadre complet pour l'architecture d'entreprise.

TOGAF est un cadre d'architecture d'entreprise qui aide à définir les objectifs commerciaux et à les aligner sur les objectifs d'architecture autour du développement de logiciels d'entreprise. Dans le cadre de ce projet, nous allons utiliser la méthode ADM de TOGAF.

L'ADM (Architecture Development Method) spécifie le cycle des étapes ou phases de la méthode et leurs transitions. Le plancher de la durée d'un cycle ADM peut être de 6 mois et le plafond de 2 ans.

Comme toutes méthodes, pour contribuer à atteindre les objectifs fixés depuis la vision jusqu'à la maintenance de l'architecture déployée, les étapes requièrent des artefacts en entrées et fournissent des produits en sortie.

5.1.1. Les phases de ADM

Cette méthode de développement architectural est une approche détaillée, divisée en phases, qui explique comment gérer l'ensemble du cycle de vie d'une architecture.

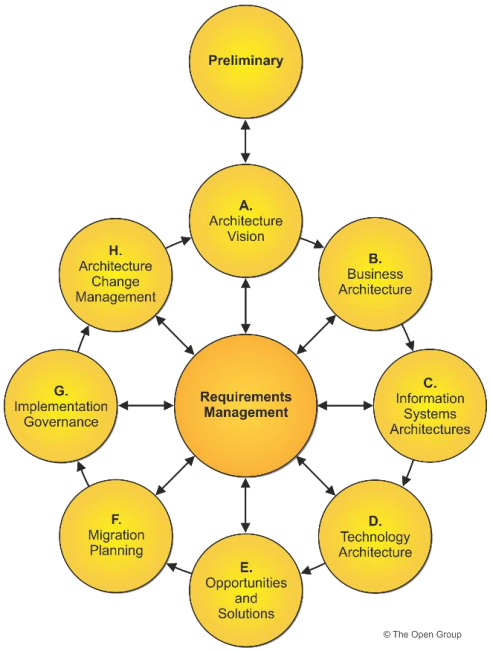


Figure 3 - illustration des phase ADM

Dans le cadre de ce projet, nous allons utiliser les phases suivantes :

Phase	Entrée/Sortie	Notes
Préliminaire	**Entrées :** Stratégies, objectifs, périmètre, budget **Sorties :** -Demande de mise en chantier d'architecture : Synthèse stratégie métier, but, objectifs, moteurs changement et principes à appliquer. - Organisation de l'équipe d'architecture d'entreprise	Identifier les besoins pour réaliser l'architecture
A – Vision de l'architecture	**Entrées :** - Principes et objectifs métiers - Cadre d'archi, adapté - Enterprise continuum **Sorties :** - Vision de l'architecture pour le cycle ADM en cours - Carte de acteurs concernés - Plan de communication	Développer une vision de haut niveau
B – Architecture Business	**Entrées :** - Les technologies existantes - Les exigences fonctionnelles et techniques - Les contraintes **Sorties :** - Architecture cible	- Initialisation du document de l'architecture. - Développer l'architecture cible. - Identifier les écart entre les exigences existantes et la cible
C – Architecture des systèmes d'information	**Entrées :** - Principes et objectifs métiers - Les exigences fonctionnelles **Sorties :** - Les fonctionnalités	Définir les services et interrelations et Définir la structure logique des données
D – Architecture technologique	**Entrées :** - Les fonctionnalités - Les exigences techniques - Les options d'architecture **Sorties :** - Le stack technologique	- Description des exigences techniques. - Description de l'infrastructure
E – Opportunités et solutions	**Entrées :** - Architecture cible **Sorties :** - Mise à jour des livrables des étapes précédentes - Architectures de transition - Plan de mise en œuvre et migration	Approche pour la mise en œuvre
F – Planning de migration	**Entrées :** - Architecture cible - Ressources	- Donner une valeur métier au projet

	Sorties : - Tous les livrables sont livrés en version finalisée - Produit les drafts de contrat d'architecture	- Estimer le besoin en ressources, planification et l'organisation des livrables
G – Gouvernance de l'implémentation	**Entrées :** - Equipes d'architecture - Equipe de mise en œuvre - Contrats d'architecture **Sorties :** - Contrats d'architecture signés	- Formulation des recommandations - Assurance de la conformité du projet avec l'architecture cible
H – Management du changement	**Entrées :** - Demande de changement - Capacité des équipes - Les exigences fonctionnelles et techniques **Sorties :** - Update référentiel d'archi - Demandes de changements avec décisions	Intégration des modifications dans le projet en cours.

6. CONTENU DE L'ARCHITECTURE

6.1. Phase A : Vision de l'architecture

6.1.1. Problèmes du système existant

Les problèmes soulevés dans le système actuel sont les suivants :

1. La plate-forme actuelle ne peut plus soutenir les projets de croissance et d'expansion de l'entreprise.
2. Un volume important de dette technique et un manque de cohérence commence à impacter le développement de nouvelles fonctionnalités.
3. Cette solution technique complexe n'évolue plus au rythme de l'activité.
4. Le système actuel ne permet pas la géolocalisation.
5. Lenteurs constatées dans l'utilisation de la fonctionnalité de recherche.

6.1.2. Contraintes de l'architecture cible

Les contraintes de l'architecture cible est d'ordre budgétaire, temporel et techniques :

1. Le projet initial est approuvé pour un coût de 50 000 USD (45 190 €) et une période de 6 mois est prévue pour définir l'architecture et préparer un projet de-suivi afin de développer un prototype.
2. L'architecture doit permettre d'obtenir le meilleur rapport qualité-coût.
3. L'architecture peut inclure de nouveaux composants personnalisés ou des composants du commerce pour favoriser la flexibilité, la stabilité et l'extensibilité.

4. L'architecture doit être évolutive de manière à accompagner la croissance de l'entreprise.

6.1.3. Objectifs et exigences

1. Tirer parti de la géolocalisation
2. L'architecture doit-être évolutive
3. La solution doit être fiable, stable et toujours disponible
4. Elle doit tenir compte des contraintes de bande passante
5. Prendre en charge différents types d'utilisateurs (fournisseurs, back-office, utilisateurs.
6. Les solutions open-source sont préférables aux solutions payantes
7. Support continu des composants doit être pris en compte lors de leur sélection ou lors de la prise de décision de création d'achat.
8. Les livrables doivent pouvoir être fournis à intervalles réguliers

6.2. La Phase B : Architecture métier

Phase B – Architecture Métier :
Le métier avec ses exigences, ses processus et ses entités, gouverne l'architecture d'entreprise. Elle permet de fixer l'architecture cible et de mesurer les impacts.

6.2.1. Diagramme de contexte (architecture existante)

Modèle C4 détaillé de l'architecture historique de Foosus.

Le diagramme de composant ci-dessous montre les composants clés du système actuel. En rapport avec la loi de Conway, cette vision représente également la structure d'équipe d'origine de Foosus.

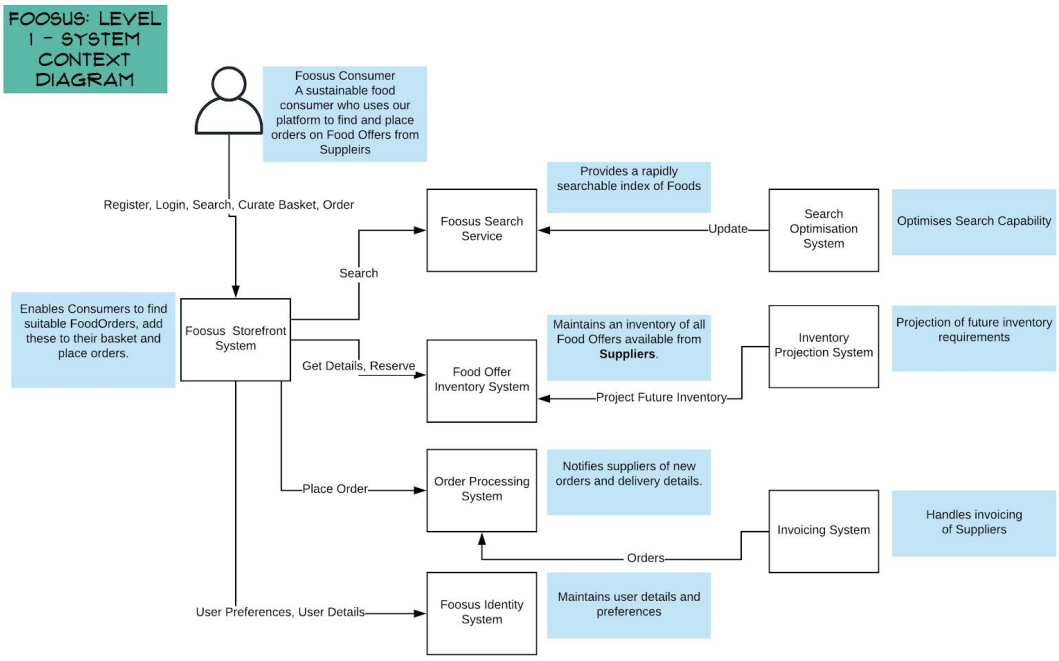


Figure 4 - Diagramme contexte

6.2.2. Description des composants

1. **Foosus Storefront System (En français, Système Storefront de Foosus) :** Ce système permet aux consommateurs de trouver des commandes d'aliments durables, de les ajouter à leur panier et de passer des commandes.
2. **Foosus Search Service (En français, Service de recherche de Foosus) :** Ce service fournit un index des aliments rapidement consultable.
3. **Search Optimisation System (En français, Système d'optimisation de recherches) :** Ce système permet d'optimiser la capacité de recherche.
4. **Food Offer Inventory System (En français, Système Stock offres alimentaires) :** Ce système permet de maintenir un inventaire de toutes les offres alimentaires disponibles auprès des fournisseurs.
5. **Inventory Projection System (En français, Système de projection d'inventaire) :** Ce système permet la projection des besoins futurs en inventaire.
6. **Order Processing System (En français, Système de traitement des commandes :** Ce système permet d'informer les fournisseurs des nouvelles commandes et des détails de livraison.
7. **Invoicing System (En français, Système de facturation) :** Ce système permet de gérer la facturation des fournisseurs.
8. **Foosus Identity System (En français, Système d'identité de Foosus) :** Ce système permet de maintenir les détails et les préférences des utilisateurs.

6.2.3. Diagramme de niveau conteneur

Ce diagramme représente les dépendances des conteneurs C4 (ou des applications, dans ce contexte) et les relations distribuées impliquées dans la satisfaction au système Storefront de Foosus.

Le Storefront utilise le modèle de design de backend pour le frontend et la propagation du comportement. En pratiques, les backends de commandes Storefront sont de grosses applications monolithiques qui effectuent plus que de simples passages de commandes.

Bien que Java soit la compétence clé au sein des équipes plate-forme, sélectionnées pour cela au moment du recrutement, la plate-forme en elle-même inclut une vaste gamme de choix techniques. Ceux-ci ont été mis en place de manière organique, avec peu de réflexion stratégique. Une impulsion vers la standardisation à l'avenir serait dans l'intérêt du business dans son ensemble.
Notez l'existence de plusieurs applications, services, et travaux prévus non inventoriés qui soutiennent supposément ces fonctions clés.

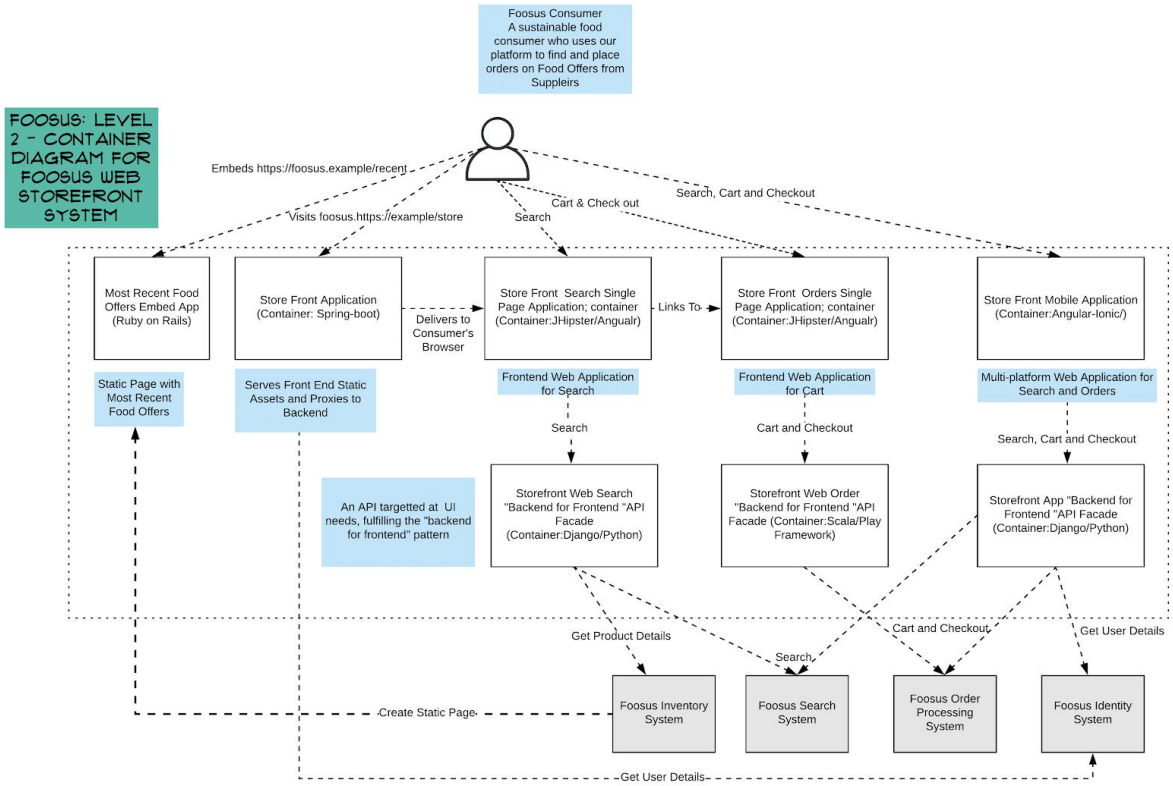


Figure 5 - Diagramme de niveau conteneur

6.2.4. Analyse des exigences

Dans cette section nous décrivons les différentes exigences ou conditions du projet. Elles sont regroupées en 2 groupes distincts : Exigences techniques et exigences opérationnelles. Le tableau ci-dessous nous présente les exigences et les solutions.

Exigences	Solutions
Exigences techniques	
Une nouvelle plate-forme qui pourra faire franchir le prochain million d'utilisateurs inscrits à notre base de clientèle.	- Gestion des charges avec l'utilisation de load balancer.
La pile technologique doit être conçue de façon à évoluer naturellement au même rythme que la base de clientèle de l'entreprise.	- Utilisation d'une architecture avec un couplage faible - L'indépendance des modules - Maintenance facile - La réutilisation des composants développés
La plate-forme doit absorber le trafic, mais être également capable d'évoluer pour gérer les augmentations de charges.	- Gestion des charges avec l'utilisation de load balancer.
La plate-forme ne doit pas être désactivée à chaque installation d'une nouvelle version ou à chaque modification du schéma de la base de données.	- Choix d'une architecture permettant l'évolutivité du système, donc de la maintenance facile.
Doit permettre l'intégration des travaux des différentes équipes de la plate-forme sans créer de pannes du système	- Pratique de l'intégration continue.
Les solutions opensources sont préférables aux solutions payantes	- Choix des technologies opensources en accord avec l'architecture.
Exigences opérationnelles	
Gestion de la sécurité des données	- Sécurité par HTTPS : - Sécurité par TLS : - Mise en place d'un firewall - Mise en place d'un système d'authentification sécurisé par login/mot de passe.
Gestion d'une version web et mobile de la plate-forme	- Utiliser une architecture qui offre hétérogénéité des plate-formes.
Optimisation de la fonctionnalité «recherche fournisseurs alimentaires»	- Utilisation d'une base de données NoSQL
Trier parti de la géolocalisation	- Gestion de la position géographique de l'utilisateur
Exigences opérationnelles futures	
Gestion d'intégrer à des prestataires tiers de paiements	- Choix d'une architecture permettant l'évolutivité du système.
Gestion de toutes les communications avec les fournisseurs alimentaires au sein d'une interface utilisateur personnalisée.	- Choix d'une architecture permettant l'évolutivité du système. - Choix des outils de communication adaptée

Au vu des objectifs et des différentes contraintes techniques et fonctionnelles du projet, telles que :

1. Gestion des charges
2. Sécurité des données
3. Gestion d'un grand nombre d'utilisateurs simultanés.
4. Amélioration et modification future du système (évolutivité).
5. Forte croissance de l'entreprise (scalabilité).
6. Stabilité et fiabilité de la plate-forme.

L'architecture recommandée est une architecture en micro-services. En effet, l'architecture Microservices propose une solution en principe simple : découper une application en petits services, appelés Microservices, parfaitement autonomes qui exposent une API *REST* que les autres Microservices pourront consommer.

6.2.5. Architecture cible (Microservices)

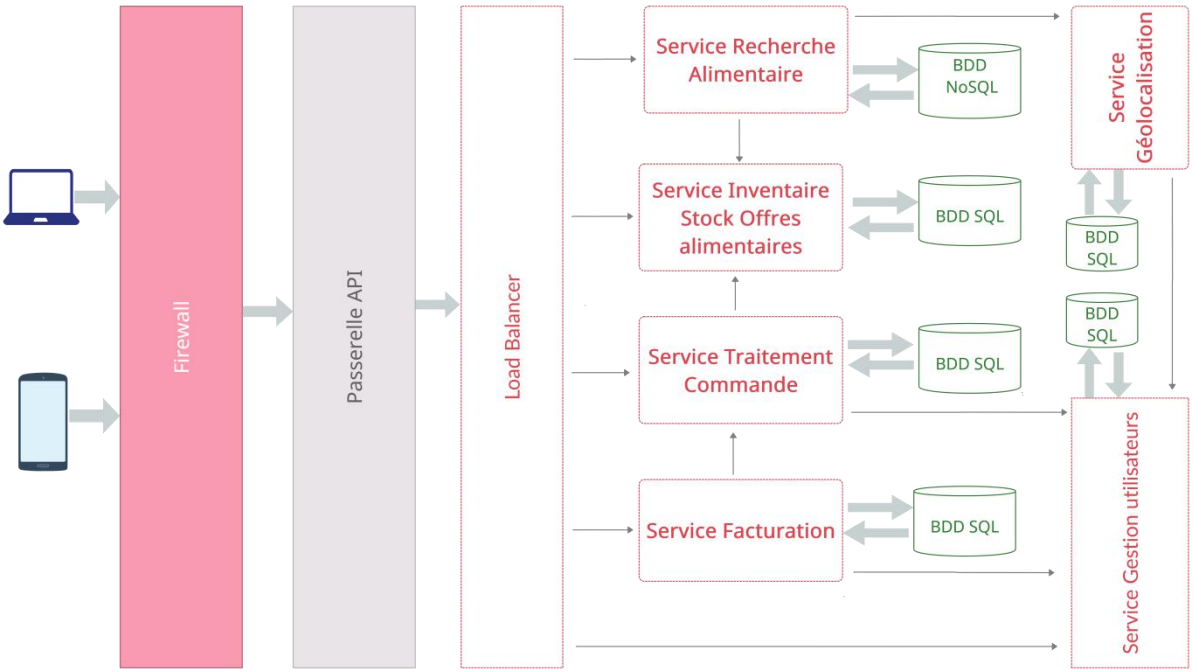


Figure 6 - Architecture cible

6.2.6. Description des composants

Le tableau ci-dessous nous présente la liste des composants à ajouter pour optimiser l'architecture actuelle de l'entreprise Foosus

Composants	Description
Firewall	Il permet d'assurer la sécurité des informations d'un réseau en filtrant les entrées et en contrôlant les sorties selon des règles définies par son administrateur.
Une API Gateway	(Également appelée passerelle API) est le point d'entrée unique pour les API et microservices backend définis (qui peuvent être à la fois internes et externes). Assise devant les API, l'API Gateway agit en tant que protecteur, renforçant la sécurité et assurant l'évolutivité et la haute disponibilité.
Load balancing	Répartition de charge, est une technologie conçue pour distribuer la charge de travail entre différents serveurs ou applications. Le but : optimiser la performance globale de l'infrastructure, son rendement et sa capacité.
Service Recherche Alimentaire	Ce composant assure la recherche des aliments, fournissant ainsi un index des aliments rapidement consultable.
Service Stock Offres Alimentaires	Ce composant permet de maintenir un inventaire de toutes les offres alimentaires disponibles auprès des fournisseurs.
Service Traitement Commandes	Ce composant permet aux consommateurs de passer commande, informant ainsi les fournisseurs des nouvelles commandes et des détails de livraison.
Service facturation	Ce composant permet de gérer la facturation des fournisseurs.
Service Gestion des utilisateurs	Ce composant permet de gérer les utilisateurs : fournisseurs et consommateurs, permettant ainsi de les identifier.
Service géolocalisation	Ce composant permet de géolocaliser les utilisateurs (connaître leur position exacte)
API REST	Une API REST (également appelée API RESTful) est une interface de programmation d'application (API ou API web) qui respecte les contraintes du style d'architecture REST et permet d'interagir avec les services web RESTful.

6.2.7. Interactions entre les micro-services

Le diagramme ci-dessous présente les liens de communication existant entre les différents micro-services de l'architecture :

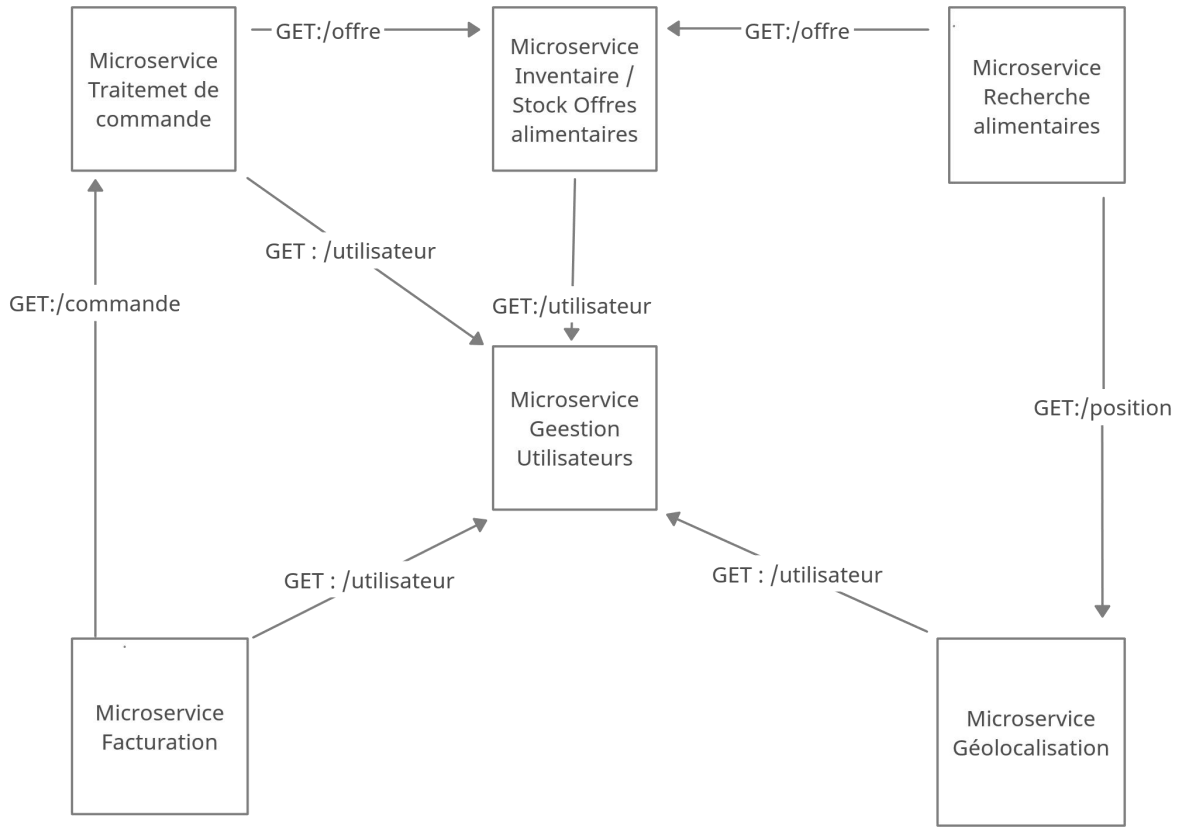


Figure 7 - Diagramme interactions entre les microservices

6.2.8. Description des dépendances entre les micro-services

1. **Le micro-service « Recherche alimentaire » :** a besoin des offres alimentaires provenant du micro-service Inventaire/Stock Offre Alimentaire pour pouvoir lancer la recherche.
2. **Le micro-service « Stock Offre Alimentaire» :** a besoin de la ressource utilisateur du micro-service « Gestion utilisateurs » pour pouvoir identifier l'utilisateur.
3. **Le micro-service « Traitement de commande» :** a besoin de la ressource utilisateur/consommateur du micro-service « Gestion Utilisateurs » pour pour l'identification, mais aussi la ressource «Offre» du micro-service « Stock Offre alimentaires» pour pouvoir traiter la commande.
4. **Le micro-service «Facturation» :** a besoin de la ressource commande du microservice «Traitement de commandes» et de la ressource utilisateur/fournisseur pour pouvoir générer a facture pour le fournisseur.
5. **Le micro-service Géolocalisation :** a besoin de la ressource utilisateurs pour pouvoir fournir s position géographique

6.2.9. Les endpoints des micro-services

Le tableau ci-dessous présente quelques endpoints des micro-services du projet :

Micro-services	Endpoints	Description action
Recherche fournisseurs alimentaires	GET:/fournisseur	Permet de rechercher un fournisseur alimentaire
Inventaire/Sotck Offres Alimentaires	GET:/offre	Permet de consulter une offre alimentaire
Traitement commandes	GET:/commande	Permet de consulter la commande
	POST:/commande	Permet de valider la commande
	PUT:/commande	Permet de annuler la commande
	GET:/commande/informations	Permet d'informer le fournisseur du passage de commande et des détails de livraison
Facturation	POST:/facture	Permet de générer la facture
Gestion utilisateurs	POST:/utilisateur	Permet de créer un utilisateur (fournisseur ou consommateur)
	PUT:/utilisateur	Permet de mettre à jour un utilisateur
	GET:/utilisateur	Permet d'afficher un utilisateur
	DELETE:/utilisateur	Permet de supprimer un utilisateur
Géolocalisation	GET:/position	Permet de fournir la position géographique d'un utilisateur

6.2.10. Les fonctionnalités

Le tableau ci-dessous nous présente les différentes fonctionnalités du projet avec les micro-services responsables de ces fonctionnalités ainsi que les ressources exposées et nécessaires.

Micro-services	Fonctionnalités	Ressources exposées	Ressources nécessaires
Recherche fournisseurs alimentaires	- Rechercher des fournisseurs alimentaires - Tri des offres alimentaires	- Aliment	- Offre - fournisseur
Inventaire/Stock Offres alimentaires	- Consulter une offre alimentaire	- Offre	
Traitement commande	- Ajouter commande - Consulter commande - Supprimer panier - Valider commande - Annuler commande - Informer fournisseur	- Commande	- Consommateur - Fournisseur - Offre
Facturation	- Éditer facture	- Facture	- Utilisateur - Commande
Gestion des utilisateurs	- Gérer consommateur - Gérer fournisseurs	- Consommateurs - Fournisseurs	-
Géolocalisation	- Géolocaliser utilisateur	- Position géographique	- utilisateurs

6.2.11. Enjeux de sécurité : protection des données

La sécurité des micro-services consiste à protéger l'intégrité des API :

1. **API GATEWAY :** On fait une authentification centralisée grâce à la passerelle d'API pour limiter l'accès aux ressources et aux API comme le montre la figure représentant l'architecture modifiée.

2. **CHIFFREMENT DES DONNES QUI TRANSITENT**

   Pour un accès plus sécurisé, on va utiliser un certificat SSL (Secure Socket Layer) ou plutôt TLS (Transport Layer Security): il s'agit d'une technologie standard destinée à la sécurité de la connexion Internet et à la protection des données sensibles qui sont transmises entre deux systèmes, empêchant les criminels de lire et de modifier les informations transférées, y compris d'éventuelles informations personnelles.

   TLS est avant tout un protocole de cryptage des données circulant sur le web. Il s'agit d'un système de sécurisation qui agit entre les serveurs et le réseau. TLS est en quelque sorte la version améliorée de SSL. C'est un protocole en deux couches, qui reprend les caractéristiques principales du SSL tout en améliorant certaines fonctions, de manière à mieux sécuriser les échanges de données. Les deux couches désignent deux clés qui agissent l'une en parallèle de l'autre lorsqu'un utilisateur souhaite par exemple rejoindre une navigation sécurisée.

   Lorsqu'un visiteur s'authentifie, il fait appel à ce certificat SSL mis en place par le

serveur. Ce certificat est en quelque sorte la carte d'identité électronique du site sécurisé. Il a également pour rôle de chiffrer les données entrées par l'utilisateur afin de leur garantir un premier niveau de protection.

Ce certificat contient deux clés distinctes : d'un côté, une clé publique qui valide le certificat racine et de l'autre, une clé privée, plus adaptée au chiffrement des échanges de données afin de leur assurer un niveau de protection maximal.

3. **Système de gestion des utilisateurs :** Système pour gérer les utilisateurs de la plate-forme y compris les consommateurs et les fournisseurs. C'est ce système qui gère identification sur la plate-forme.

4. **Firewall application web :** Il permet de filtrer les requêtes entrant sur un serveur HTTP Apache. Il se présente sous la forme d'un module apache, qui analyse les requêtes reçues grâce à l'emploi d'une base des règles de requêtes considérées comme non souhaitées.

6.2.12. Technologies / Outils

Le tableau ci-dessous présente les outils retenus pour chaque microservice, compte tenu des avantages, inconvénients et prix.

Composants	Technologies/outils	Avantages
Load balancer	✓ Nginx	✓ Architecture de Nginx qui gère le forte charge ✓ Excellentes performances ✓ Très réactif sur forte charge
API Gateway	✓ Kong Gateway	✓ Sécurité des API ✓ Evolutivité et élasticité des API ✓ Répartition des charges ✓ Haute disponibilité
Firewall application web	✓ ModSecurity	✓ Accessibilité 24/7 ✓ Prise en charge des environnements physiques et virtuels ✓ Visibilité et contrôle des applications ✓ Configuration facile
Base de données relationnelles	✓ Mysql	✓ Excellente performance ✓ Open-source ✓ Léger, portable et gratuit ✓ Rapide sur les requêtes simples
Base de données non relationnelles	✓ NoSQL	✓ Capable de gérer un volume important de données structurées, semi-structurées et non structurées. ✓ Offre une performance supérieure par rapport à MySQL ✓ Gestion des données volumineuses qui gèrent la vitesse, la variété, le volume et la complexité des données.
Géolocalisa-tion	✓ OpenStreetMap	✓ Open-source et gratuit ✓ Rapidité du chargement de la carte ✓ Les cartes sont très détaillées et toujours à jours

Intégration continue (CI)	✓ Jenkins	✓ Open-source et gratuit ✓ Facile d'installation ✓ Une très grande communauté d'utilisateurs ✓ Plus de 1000 plugins à notre disposition ✓ C'est écrit en Java, donc peut fonctionner dans la plus part des plate-formes
Back-end	✓ Spring boot	✓ Configuration facile ✓ Absolument pas de génération de code et pas de configuration XML requise ✓ Gestion des dépendances simplifiée grâce à platform-bom
Frontend	✓ Angular	✓ Open-source ✓ Framework à execution rapide ✓ Fournit des composants réutilisables ✓ Création des PWA robustes, fiables et réactifs. PWA (Progressive Web Apps). ✓ Utilisation des fonctionnalités de la plate-forme web ✓ Les applications sont adaptées aux plates-formes iOS et Android.
Dévelop-pement mobile (PWA: Progressive Web App)	✓ Angular	
Cloud	✓ Hébergement Amazon Web Cloud (hosting)	✓ coût faible ✓ niveaux de fiabilité élevés ✓ aucune administration de serveur et évolutivité permettant de gérer le trafic au niveau de l'entreprise ✓ Serveurs web ultra-rapides

7. PLAN DE TRAVAIL

7.1. Initialisation du projet

L'initialisation de projet est la première phase du cycle de vie d'un projet.
Lors de cette phase il est essentiel de :

1. Préciser l'objectif du projet ;
2. Identifier les acteurs et les parties prenantes ;
3. Identifier les risques ;
4. Définir les délais, les ressources, la démarche, les outils de pilotage et le plan de communication ;
5. Définir les ressources humaines et financière ;
6. Définir la démarche, les outils de pilotage et le plan de communication.

Les livrables :

1. **Déclaration de travail d'architecture sous format PDF** : c'est un document qui présente la vision, la direction d'architecture, un état cible, un process et une approche d'architecture sur mesure.
2. **Spécifications des conditions requises pour l'architecture** : C'est un document sous format PDF qui décrit les conditions pour l'implémentation de l'architecture et la conformité de l'implémentation.
3. **Contrat d'architecture avec les utilisateurs business** : accords communs entre les

partenaires de développement et les sponsors sur les livrables, la qualité, et la correspondance à l'objectif d'une architecture

4. **Contrat d'architecture avec les fonctions de développement et de design :** accords communs entre les partenaires de développement et les sponsors sur les livrables, la qualité, et la correspondance à l'objectif d'une architecture

7.2. Les différentes phases

Compte tenu de possibles évolutions du projet, nous avons prévu d'avoir une approche agile avec la méthode Scrum et d'utiliser la technique d'intégration continue (CI) pour éviter des régressions lors des mises à jour.

Puis, réaliser **chacune des fonctionnalités** de chaque micro-service en suivant le principe du Schema ci-dessous :

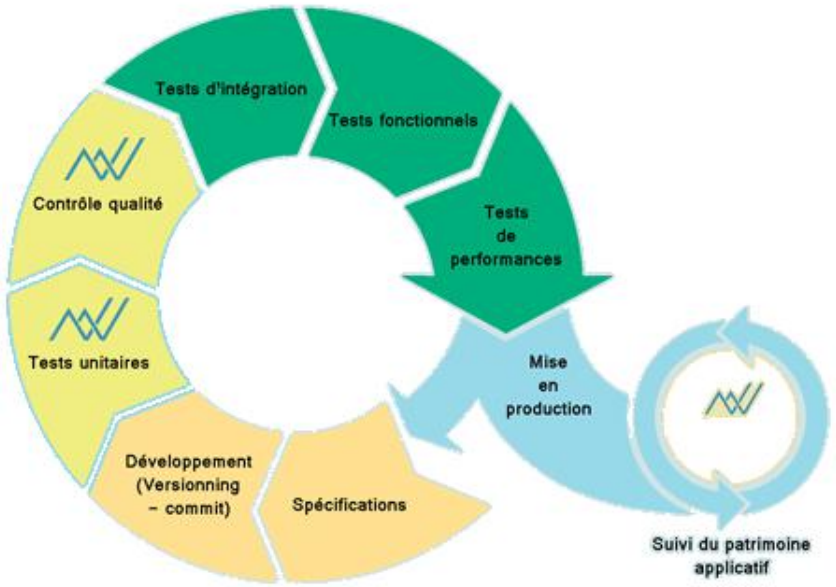


Table 1- intégration continue

Ainsi chacune des fonctionnalités de chacun de nos microservices auront à faire ce parcours :

7.2.1. Analyse

Cette phase consiste à définir l'objectif de cette fonctionnalité, la faisabilité et le degré de souplesse qui pourra être accordée.

7.2.2. Conception

Cette phase consiste à établir une liste de tâches associées à la fonctionnalité, à les ordonner tenant compte et à et à indiquer leur enchaînement logique en tenant compte des

ressources disponibles et de leur charge de travail maximale

7.2.3. Spécifications

Cette phase consiste à définir l'ensemble d'exigences à satisfaire par la fonctionnalité.

7.2.4. Développement (code)

C'est la phase où les développeurs codent la fonctionnalité (Front-End et Back-End) et connectent les interfaces pour atteindre les objectifs définis.

7.2.5. Tests Unitaires

C'est la phase qui consiste à s'assurer du fonctionnement correct d'une partie précise de la fonctionnalité. Elle a pour objectif d'isoler le comportement de la partie de code à tester de tout facteur extérieur et de vérifier qu'il est conforme à ce qui est attendu.

7.2.6. Contrôle de qualité

C'est la phase où la conformité de la fonctionnalité développée est vérifiée. Le résultat peut être conforme, non conforme mais avec possibilité de retouches ou non conforme et tout recommencer.

7.2.7. Test d'intégration

C'est la phase où on s'assure que toutes les parties développées indécemment fonctionnent bien ensemble.

7.2.8. Test fonctionnel

C'est la phase où on on va tester la fonctionnalité développée via des parcours en simulant les actions de l'utilisateur (clics, saisies claviers, mouvement de souris, …).

7.2.9. Test de performances (Benchmark)

C'est la phase où on mesure la performance de la fonctionnalité développé. Le test de performances a pour objectif de mesurer le temps de réponse.

7.2.10. Mise en production

Cette phase consiste à déployer la fonctionnalité entièrement testée et totalement fonctionnelle.

7.3. Livrables

Un ensemble de livrables seront livré au client lors du déploiement de l'application :

1. **Documentation technique** : Document qui décrit l'utilisation, la fonctionnalité ou l'architecture d'un produit, d'un système ou d'un service.
2. **Cahier de test /Recette :** Document qui regroupe de nombreux points permettant de mener à bien votre mise en production.
3. **Guide utilisateur** : Documents PDF qui sert de support aux utilisateurs expliquant comment fonctionne les différentes fonctionnalités de l'application.
4. **Document de déploiement :** Document qui réunit toutes les bonnes pratiques pour la mise en service de l'application.
5. **Les bases de données** : Les différentes bases qui servent à stocker les données utilisateurs.
6. **Les différentes applications :** Les applications sous formats archives, APK et IPA.

7.4. Plan de communication

Un large processus de consultation nécessite l'utilisation et la combinaison de différentes méthodes en tenant compte des caractéristiques du public cible. La stratégie de consultation est présentée dans le tableau ci-dessous pour ce Projet.

Étape projet	Type de message	Support	Fréquence	Cibles	Responsables
-Préparation et définition du projet	- Demande de compléments d'informations. - Risques Éventuels du projet	- Réunions - Entrevues - Courriels -Téléphones	- A chaque mise à jour du projet	- Pete Parker - Daniel Anthony	- Pete Parker - Samano CASTRE
- Définition du plan d'exécution du projet	- Réunion du comité de pilotage,	- Réunions	Au début et une fois par mois selon les besoins	MOA : Pete Parker, Daniel Anthony et Comité de direction.	-Pete Parker, -Samano CASTRE
- Exécution et le pilotae du projet	- Compte rendu réunion - Justification de l'avancement - Utilisation des ressources - Résultats intermédiaires - Demandes d'informations d'ordre techniques	- Courriels	Toutes les deux semaines	- MOA: Daniel Anthony, Comité de direction	Pete Parker
	- Point d'avancement - Réunion de travail	- Réunions, - Courriels,	Quotidien (Au fur et à	Équipe technique	MOE : -Pete Parker

	- Revu du projet - Tâches à réaliser - Résultats obtenus - Réunion d'équipe		mesure des mises à jour)		- Samano CASTRE
	- Rapport de validation de toute ou partie de fonctionnalités,	- Courriels	Récurrente	Pete Parker	- Pete Parker - Testeurs
	- Nouvelles demandes - Demandes d'évolution	- Réunions, - Courriels,	Récurrente	Pete Parker, Samano CASTRE	- Pete Parker - comité de direction
- Mise en exploitation et la clôture du projet	- Clôture COPIL - Clôture projet	- Réunions	1 Fois en fin de projet	- MOA : Pete Parker, Comité de Direction - MOE: Daniel Anthony	- MOE: Pete Parker

7.5. Analyse des risques

L'objectif d'une analyse de risques consiste à éliminer ou réduire le niveau de risques en mettant en place des mesures de prévention adéquates. Elle permet d'assurer un lieu de travail sain et sécuritaire pour tous. Dans le cadre de ce projet, nous avons identifié certains risques qui méritent d'être pris en compte.

On distingue différents types de risques :

1. **Financiers** : coût supérieur à l'estimation, manque de budget, etc.
2. **Humains** : manque de compétences, absentéisme, démission au cours du projet, conflits au sein de l'équipe, etc.
3. **Temporels** : retards ou mauvaise estimation des délais, etc.
4. **Techniques** : logiciel inadapté, pannes, matériel obsolète, etc.

Pour évaluer et gérer les risques nous avons élaboré la matrice de criticité suivante :

Probabilité							
	Certains	**5**	5	10	**15**	**20**	**25**
	Probable	**4**	**4**	8	12	**16**	**20**
	Peu probable	**3**	**3**	6	9	12	**15**
	Rare	**2**	**2**	**4**	6	8	10
	Extrêmement Rare	**1**	**1**	**2**	**3**	**4**	5
			1	**2**	**3**	**4**	**5**
			Mineur	**Significatif**	**Sévère**	**Critique**	**Catastrophique**
			Gravité (G)				

	Criticité (C Min)	**Criticité (C Max)**	**Description**
	13	25	Risque inacceptable, mesures indispensables de réduction du risque
	5	12	Risque à surveiller, mesures adaptées de réduction du risque
	1	4	Risque acceptable

Le tableau ci-dessous nous présente la gestion des risques du projet

Risque identifié	Causes	Conséquences /Impact	Risque Potentiel			Solution/Plan d'action (Barrières)	Risque Résiduel		
			P	G	C		P	G	I
Dépassement de budget	Variation du périmètre : nouvelles demandes. Sous-évaluation du budget	Retard / Blocage du projet	4	4	16	Regroupement de plusieurs estimations détaillées des charges, coût et planning. Remise en cause des demandes et éventuellement révision du budget en cas de demandes importantes.	3	4	12
Non-respect des délais	Tâches sous-évaluées	Retard sur les livrables	4	4	16	Analyse et estimation des tâches avec soin. Suivi de l'avancement du projet	3	3	9
Risque de sécurité	Données en transit mal protégées	Problèmes de confidentialités	4	4	16	Gestion des enjeu de sécurité par un Firewall, utilisation d'un certificat SSL	2	2	4
Fortes demandes	Changement	Non-respect de délai, augmentation du budget	5	2	10	Approche agile pour la réalisation du projet Une bonne gestion du changement	5	1	5
Lenteur de la plate-forme	Fortes charges	Plate-forme délaissée : diminution du nombre d'utilisateurs	4	4	16	Mise en place d'un API gateway pour la répartition des charges et d'un load balancer pour la gestion des fortes charges	2	2	4
Ralentissement de la production en phase de finalisation	Équipe démotivé	Retard sur les livrable	2	3	6	Réunion hebdomadaire «vie de groupe»	2	2	4
Réalisation d'un produit incohérent avec les besoins	Mauvaise interprétation du besoin	Produit inutilisable	3	5	15	Réalisation des cahiers de charges fonctionnel et technique et suivie du projet	1	1	1
Incompréhension sur des spécifications techniques	Manque de précisions	Développement non utilisable + retard sur le projet	3	5	15	Écriture claire et précises des spécifications	2	1	2
Indisponibilité ou absence d'un des acteurs du projet	Contraintes personnelles	Planification complexe des réunion de travail	4	3	12	Anticiper les absences quand c'est possible, réaliser des comptes rendus de réunions à partager sur un réseau de partage	4	1	4
Mise à l'écart d'un des membres	Manque d'investissement	Charge de travail supplémentaire pour les autres membres	2	3	6	Réunion hebdomadaire « Vie de groupe »	1	3	3
Problème de communication	Problématique linguistique	Incompréhension	2	2	4	Reformulations, échanges constructifs	2	1	2
Planification non exhaustive	Manque d'expériences	Retard dans le projet	2	2	4	Révision du planning à chaque réunion	1	1	1
Non-respect des tâches individuelles	Manque de temps, tâches inadaptées	Retard dans le projet	4	3	12	- Approche agile avec la méthode Scrum - Suivi de l'avancement des tâches	3	1	3
Climat conflictuel dans l'équipe	Opinions divergentes	Désintégration de l'équipe	3	5	15	Établissement des règles de fonctionnement, Réunions « vie de groupe »	2	3	6

8. CRITERES D'ACCEPTATION ET PROCEDURES

8.1. Métriques et KPIs

Le KPI c'est le suivi d'indicateurs de type ratios comparant le "prévisionnel" et le "réalisé" en termes de temps, de consommation de budget et de ressources.

Les métriques suivantes seront utilisées pour déterminer le succès de ce travail d'architecture :

Métrique	Technique de mesure	Valeur cible
Nombre d'adhésion d'utilisateur par jour	Comparaison avec la plate-forme actuelle	Augmentation de 10%
Adhésion de producteurs alimentaires	Comparaison avec la plate-forme actuelle	Passer de 1.4 par mois à 4 par mois
Délai moyen de parution	Temps	Réduit de 3.5 semaines à moins d'une semaine
Taux d'incidence de production	Incidents	Passer de plus de 25 par mois à moins de 1 par mois
Latence pour la recherche	Effectuer une recherche	Entre 1 à 5 secondes

9. APPROBATIONS SIGNEES

Partie prenante	Fonction	Signature
Ash Callum	PDG (Président-directeur Général)	

10. CONCLUSION

Nous avons établi cette déclaration de travail d'architecture conformément à la méthode ADM de TOGAF pour répondre aux besoins de l'entreprise Foosus en matière d'architecture.
Elle a pour but de spécifier :

1. 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.
2. Un État Cible de l'Architecture vers lequel l'organisation doit itérer.
3. Un process et une approche d'architecture sur mesure, mais flexibles, qui conviennent aux structures de ses équipes et à la topologie de son organisation.