Contrat de Conception et de Développement de l'Architecture

- *Projet :* Foosus Géoconscient
- *Client :* Foosus
- *Préparé par :* Ludovic SOUPLET

Table des matières

Objet de ce document

Les Contrats d'Architecture sont les accords communs entre les partenaires de développement et les sponsors sur les livrables, la qualité, et la correspondance à l'objectif d'une architecture. L'implémentation réussie de ces accords sera livrée grâce à une gouvernance de l'architecture efficace (voir TOGAF Partie VII, Gouvernance de l'architecture). En implémentant une approche dirigée du management de contrats, les éléments suivants seront garantis :

- *Un système de contrôle continu pour vérifier l'intégrité, les changements, les prises de décisions, et l'audit de toutes les activités relatives à l'architecture au sein de l'organisation.*
- *L'adhésion aux principes, standards et conditions requises des architectures existantes ou en développement*
- *L'identification des risques dans tous les aspects du développement et de l'implémentation des/de l'architecture(s), y compris le développement interne en fonction des standards acceptés, des politiques, des technologies et des produits, de même que les aspects opérationnels des architectures de façon à ce que l'organisation puisse poursuivre son business au sein d'un environnement résilient.*
    - *Un ensemble de processus et de pratiques qui garantissent la transparence, la responsabilité et la discipline au regard du développement et de l'utilisation de tous les artefacts architecturaux*
    - *Un accord formel sur l'organe de gouvernance responsable du contrat, son degré d'autorité, et le périmètre de l'architecture sous la gouvernance de cet organe*

Ceci est une déclaration d'intention signée sur la conception et le développement de l'architecture d'entreprise, ou de parties significatives de celles-ci, de la part d'organisations partenaires, y compris les intégrateurs système, fournisseurs d'applications, et fournisseurs de service.

De plus en plus, le développement d'un ou plusieurs domaine(s) d'architecture (business, données, application, technologie) peut être externalisé, avec la fonction d'architecture de l'entreprise fournissant une vue d'ensemble de l'architecture d'entreprise globale, ainsi que la coordination et le contrôle de l'effort total. Dans certains cas, même ce rôle de supervision peut être externalisé, bien que la plupart des entreprises préfèrent conserver cette responsabilité clé en interne.

Quelles que soient les spécificités des dispositions d'externalisation, les dispositions elles-mêmes seront normalement gouvernées par un Contrat d'Architecture qui définit les livrables, la qualité, et la correspondance à l'objectif de l'architecture développée, ainsi que les processus de collaboration pour les partenaires du développement de l'architecture.

Introduction et Contexte

La plateforme historique de Foosus a atteint un point critique où elle n'est plus adaptée à son objectif initial. Les équipes de développement sont actuellement entièrement concentrées sur la résolution des problèmes et le maintien en état de fonctionnement, ce qui a considérablement ralenti la capacité à introduire de nouvelles fonctionnalités et à rester compétitifs sur un marché en constante évolution et qui est imprévisible.

Les analyses de marché indiquent que la pertinence par rapport au marché a été éclipsée par l'instabilité de la plateforme et une réputation négative due à des interruptions de service visibles par le public.

En réponse à une forte baisse des inscriptions d'utilisateurs, l'objectif est de maintenir la plateforme actuelle en mode maintenance tout en restructurant les équipes afin de créer une nouvelle plateforme avec une architecture plus réfléchie. Cette nouvelle plateforme devrait permettre de croître en alignement avec la vision commerciale qui vise à soutenir les marchés locaux. Les inscriptions d'utilisateurs sont une mesure cruciale pour les investisseurs, et leurs améliorations ne peut être réalisées qu'en adoptant l'agilité nécessaire pour innover rapidement et expérimenter avec différentes offres de produits.

L'objectif est de développer un nouveau produit de manière rapide et itérative, qui pourra coexister initialement avec la plateforme actuelle avant de la remplacer complètement.

La Nature de l'accord

Les Contrats d'architecture représentent des ententes essentielles entre diverses parties impliquées dans le développement d'un projet, notamment les partenaires de développement, les commanditaires et l'équipe de projet. Ces accords portent sur plusieurs aspects cruciaux, tels que les livrables attendus, les normes de qualité à respecter et la conformité à l'objectif global de l'architecture de l'entreprise. En d'autres termes, ils constituent une déclaration d'intention commune, dans laquelle toutes les parties s'engagent à suivre les principes et les directives de l'architecture d'entreprise pour garantir la cohérence, la performance et la réussite du projet. Ces contrats servent de cadre pour orienter les décisions et les actions tout au long du processus de développement.

Objectifs et périmètre

Objectifs

Les objectifs business de ce Travail d'Architecture sont les suivants :

Objectif Business	Notes
Évoluer avec notre base clientèle	La pile technologique doit être conçue pour évoluer en parallèle avec la base de clientèle. Des pannes ont été observées en raison de l'incapacité du système logiciel à gérer les pics d'utilisation résultant des activités des clients et des programmes marketing. Les contraintes de performances du système actuel ne permettent pas de prendre en charge le niveau d'engagement et de croissance anticipé pour les futurs programmes marketing. Même en cas de surcharge du système, l'accès à tous les services doit rester disponible pour les utilisateurs connectés, ce qui n'est pas le cas actuellement.
Une plateforme sécurisée, utilisable et réactive	Il est envisagé de lancer des campagnes marketing Foosus dans diverses régions géographiques, et nous aspirons à ce que la plateforme puisse non seulement gérer le trafic existant, mais qu'elle soit également adaptable pour faire face à des augmentations de la charge. De plus, il est demandé qu'elle soit facilement personnalisable pour répondre aux particularités locales et aux exigences de nos clients. Dans le passé, il a été privilégié la convivialité au détriment de la sécurité, ce qui a parfois mis en péril la réputation de Foosus. Il est demandé d'éviter tout risque pour l'image de marque, et il est recherché une approche qui garantira la sécurité à chaque expansion de la plateforme.
Une technologie transparente	L'arrêt de la plateforme à chaque nouvelle version ou modification de la base de données n'est plus envisageable. Le marché cible englobe des villes à travers le monde, et l'époque où il était possible bénéficier de pauses au cœur de la nuit est révolue. Être opérationnel 24h/24 est la nouvelle norme !

	Chaque nouvelle version doit être légère, minimiser les risques, rester transparente pour nos utilisateurs, et demeurer accessible partout et en permanence. Le succès dépend de la facilité d'accès aux services par les utilisateurs et de leur satisfaction envers le produit. Garantir des performances similaires pour les utilisateurs dans différentes régions, qu'ils se trouvent dans des zones géographiques spécifiques avec des connexions lentes (comme sur des téléphones portables) ou sur des réseaux haut débit est ce qui doit primer. Toutes les solutions doivent être en mesure de répondre à cette exigence.
Une évolutivité capable d'accompagner la croissance	L'année dernière, 12 des incidents rencontrés par la plateforme ont été déclenchés par la mise en œuvre de modifications importantes par une ou plusieurs équipes, sans obtenir les résultats escomptés. Il a également été rencontré des difficultés à harmoniser les travaux réalisés par différentes équipes sur des modifications de la plateforme qui n'étaient pas liées entre elles. En tant que petite entreprise, être confrontés à ce genre de problèmes n'est absolument pas normal. La principale difficulté réside dans le laps de temps nécessaire pour que chaque nouvelle version logicielle soit vue par les autres équipes ou testée dans nos environnements de production. Réduire l'écart entre le moment où une ligne de code est écrite et le moment où elle est validée dans un environnement intégré est nécessaire. Cela permettra également d'évaluer les réactions des clients par rapport aux nouvelles fonctionnalités au fur et à mesure de leurs développements.
Expérimentation	Les équipes produits aimeraient pouvoir exécuter diverses variantes ou réaliser des comparaisons de différentes solutions auprès de nos utilisateurs. Pour y parvenir, les équipes ont besoin de visibilités sur la façon dont les logiciels sont utilisés et doivent pouvoir inverser des décisions d'architecture tant que cela reste peu onéreux. Ou alors répliquer sur une plateforme qui permette d'essayer de nouveaux produits d'une façon compatible avec les objectifs commerciaux fondamentaux.

Parties prenantes, préoccupations et visions

Partie prenante	Fonction	Préoccupation	Vision
Ash CALLUM	CEO	Création d'une plateforme polyvalente, fiable et économique pour soutenir l'entreprise dans sa croissance et concurrencer les entreprises du même domaine	Soutenir les producteurs locaux et les mettre en relation avec leurs clients afin d'avoir une consommation responsable
Natasha JARSON	CIO	Avoir une architecture responsable et des équipes de développement impliqués dans la culture lean Servir les clients	Avoir une plateforme performante pour soutenir la vision de l'entreprise
Daniel ANTHONY	CPO	Avoir une architecture responsable et des équipes de développement impliqués dans la culture lean Servir les clients Accompagner la croissance	Avoir une plateforme performante pour soutenir la vision de l'entreprise. Créer une expérience utilisateur exceptionnelle qui révolutionne la manière dont nos clients interagissent avec nos produits
Pete Parker	Engineering Owner	Avoir des équipes disponibles pour le développement de fonctionnalités plutôt que de faire de la réparation d'incidents Avoir une voie unique pour le développement	Avoir une vision stratégique afin de construire une plateforme résiliente et offrir aux ingénieurs un ensemble cohérent et motivant
Jack HARKNESS	Operations Lead	Assurer des déploiements rapides et sans interruptions Éviter les incidents en phase d'exploitation	Avoir une plateforme résiliente, performante et sécurisée
Ludovic SOUPLET	EAO	Proposer une architecture résiliente et évolutive Prendre en compte les besoins de chacun en trouvant les meilleurs compromis	Définir et maintenir une architecture cohérente et stratégique pour soutenir les objectifs de l'entreprise, favorisant l'efficacité

			opérationnelle et l'innovation tout en garantissant la sécurité.
Product Managers	PM	La planification, le suivi, le contrôle et la gestion des ressources pour assurer la réussite d'un projet Respect des délais, du budget et les attentes des parties prenantes. Gérer les risques, la qualité, les communications et les relations avec l'équipe et les clients	
Products Owner	PO	Respect des délais Respect des besoins métiers	Avoir une vision du produit clair afin de pouvoir apporter la bonne valeur produit aux clients
Development Teams	DT	Avoir une vision claire sur le produit pour avancer dans la même direction et respecter les bonnes pratiques Pouvoir innover malgré tout	
Équipe commerciale		Avoir une plateforme opérationnelle et de nouvelles fonctionnalités régulièrement	
Clients		Avoir accès à la plateforme en permanence Facilité d'utilisation	

Description de l’architecture, principes stratégiques et conditions requises

Description

La nouvelle plateforme devra répondre à des exigences de scalabilité, performances et de livraisons rapides. De plus, dans un premier temps elle devra fonctionner en parallèle de l’ancienne plateforme avant son décommissionnement.
Afin de répondre à ces exigences, une solution Micro-Service avec hébergement dans le cloud est proposé. Les détails de cette solution se trouve dans la section suivante.

Principes stratégiques

Nous ne pouvons pas abandonner les outils actuels pendant que le développement de nouveaux, car cela entraînerait l'indisponibilité de la plateforme existante. Afin de continuer à accepter de nouveaux fournisseurs et consommateurs, il doit être séparer les nouveaux développements de l'architecture et de l'infrastructure actuelles pour minimiser les interruptions de service.

L’objectif est de libérer la créativité et le savoir-faire des équipes techniques. Ils doivent avoir la possibilité de s'exprimer pleinement en créant une nouvelle plateforme capable d'attirer le prochain million d'utilisateurs dans la base client.
Il sera lancé des campagnes marketing Foosus dans plusieurs grandes villes tout en ayant la certitude que la plateforme restera fonctionnelle, réactive, et qu'elle offrira une expérience utilisateur exceptionnelle.

Référence aux Conditions requises pour l’architecture

L'architecture **micro services** est un style de conception logicielle où une application est décomposée en petits services autonomes qui travaillent ensemble pour fournir une fonctionnalité complète. Chaque service est indépendant, communique via des interfaces, peut être développé et déployé séparément, et offre une meilleure évolutivité et résilience. Cette approche est particulièrement adaptée aux applications complexes et évolutives.

Pour ce faire, chaque micro service doit être conteneurisé. La **conteneurisation** est une méthode qui permet d'encapsuler des applications et leurs dépendances dans des environnements isolés et portables, appelés conteneurs. Cela facilite le déploiement, la gestion

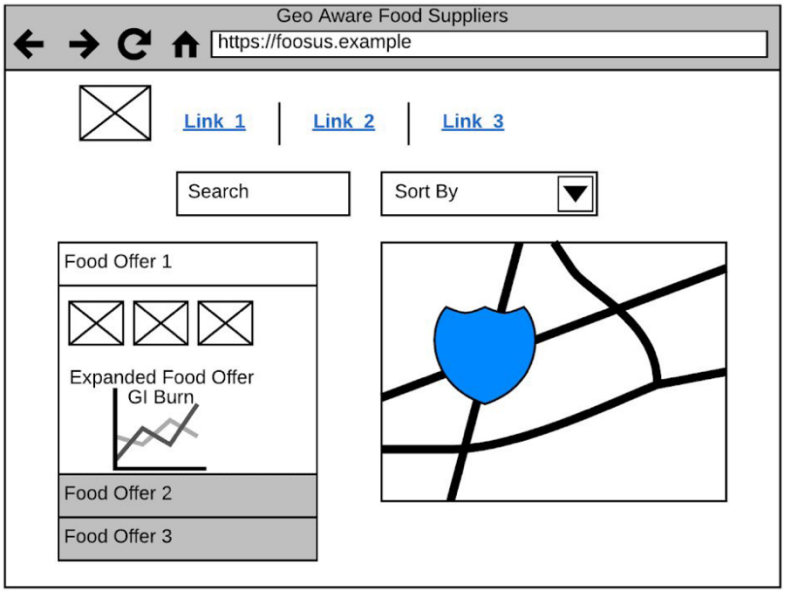
et la portabilité des applications tout en offrant une isolation et une flexibilité accrues. Elle est essentielle pour le développement et le déploiement d'applications modernes.

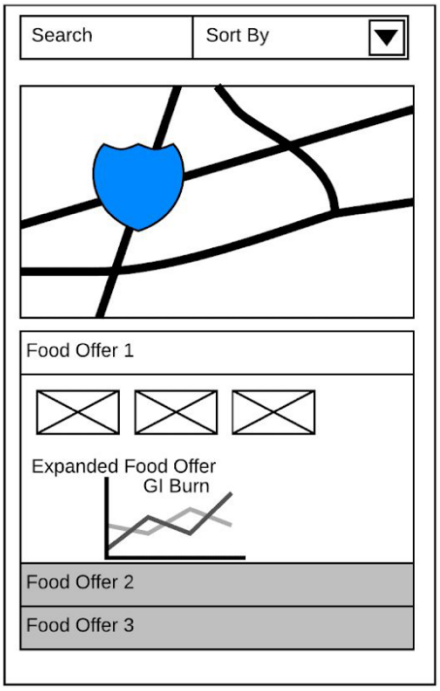
Afin de gérer l'ensemble des conteneurs (un conteneur = un micro service) qui constitue notre application, nous utiliserons l'orchestration. **L'orchestration** pour une application micro services consiste à coordonner et à gérer les différents services micro services qui composent l'application, en s'assurant qu'ils fonctionnent harmonieusement ensemble. Elle inclut la gestion du déploiement, de la mise à l'échelle, de la supervision et de la coordination des services pour garantir la performance et la fiabilité de l'ensemble de l'application.

Afin de répondre au besoin de performance tout autour du monde, nous utiliserons le principe du cloud régionalisé. Le **cloud régionalisé** pour une application micro services signifie que l'application est déployée sur des centres de données dans différentes régions géographiques du cloud. Cela permet d'améliorer la disponibilité, la résilience et les performances de l'application en fonction de la localisation des utilisateurs, tout en garantissant que les services micro services sont répartis de manière stratégique pour servir efficacement différentes régions.

Afin d'automatiser le déploiement des nouvelles itérations nous utiliserons une **chaîne CI/CD**. Une CI/CD (Intégration Continue/Livraison Continue) pour une application micro services est un processus automatisé qui permet de développer, tester et déployer rapidement et de manière cohérente chaque micro service, garantissant ainsi une mise à jour continue et efficace de l'application. Cela favorise l'agilité, la qualité et la stabilité de l'ensemble de l'application.

Une spécification **d'UI** pour l'implémentation de la nouvelle fonctionnalité est également fournie. Voir les maquettes ci-dessous :





- Emplacement des offres alimentaires proposées par les fournisseurs

- Proximité de l'utilisateur effectuant la recherche en cours
- Visualisation des informations statistiques secondaires et sectorielles relatives au produit alimentaire concerné. Par exemple, détails sur son indice glycémique

Livrables architecturaux

Ces éléments sont présents dans la Déclaration de travail d’architecture que vous pouvez trouver dans le repository d’artefacts architecturaux.

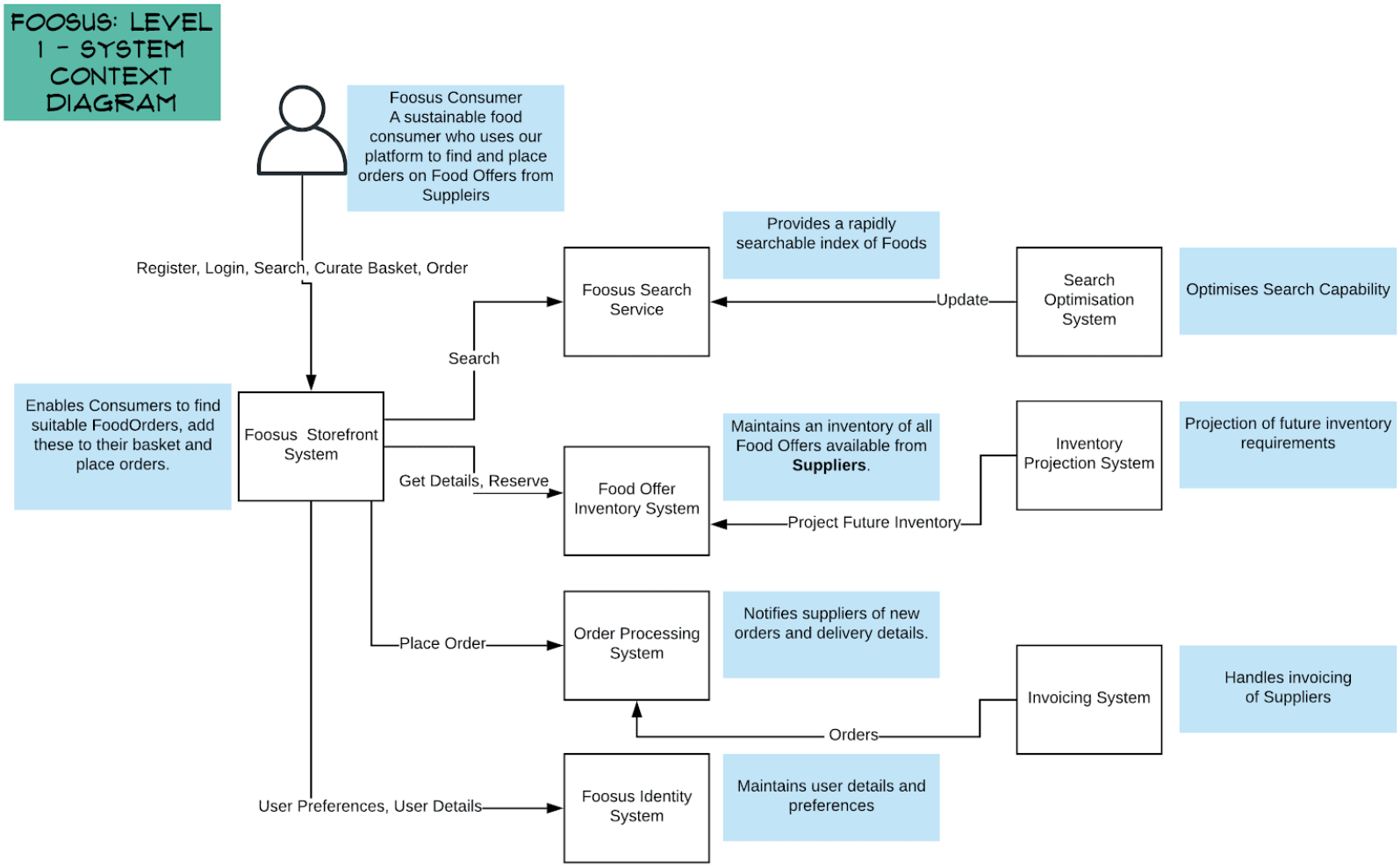
Chapitre :

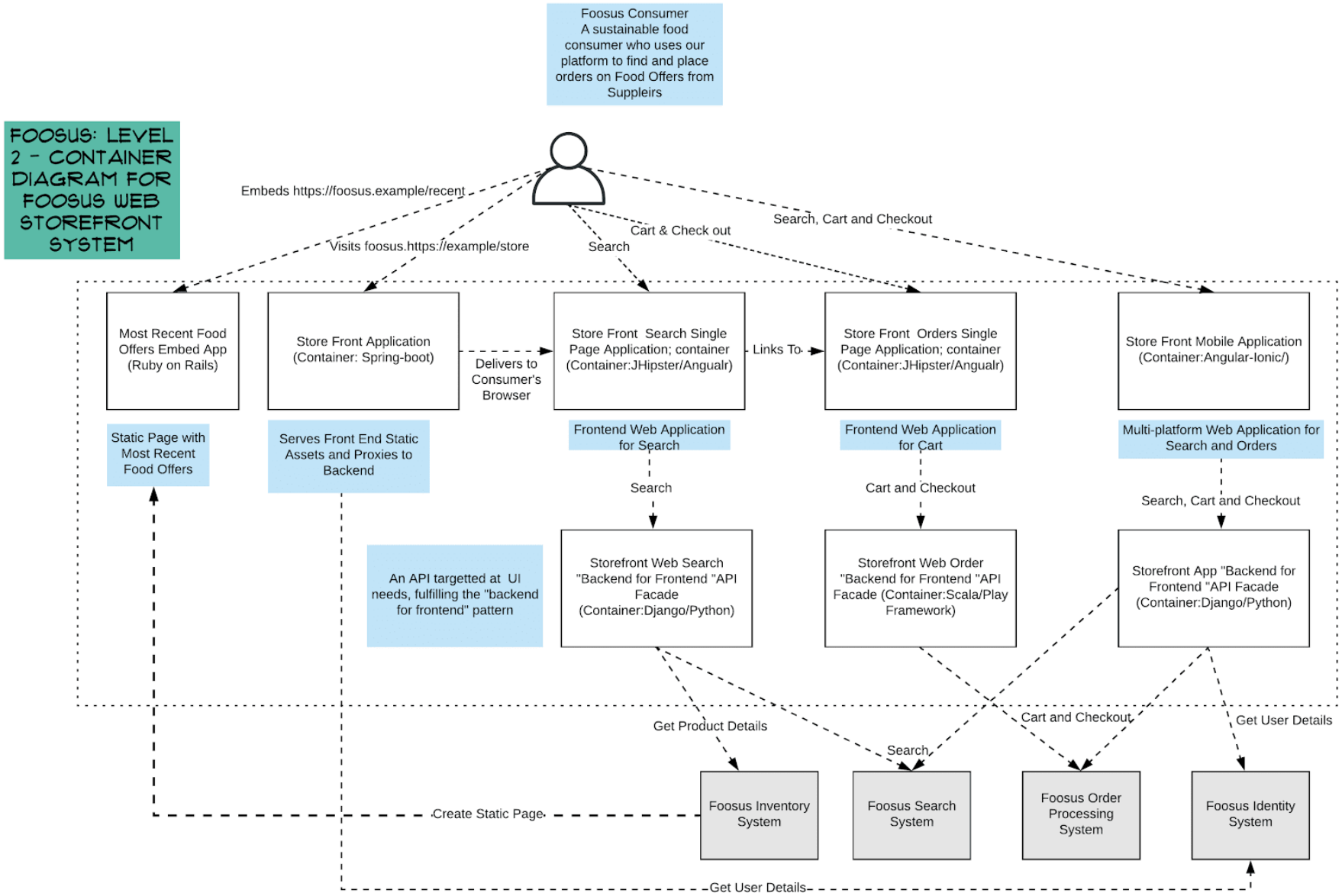
- Process d’architecture

Développement de l’architecture

Analyse de l'état actuel :

On trouvera ci-dessous les diagramme C4 de niveau 1 et 2 de la plateforme actuelle.




FOOSUS: LEVEL 2 - CONTAINER DIAGRAM FOR FOOSUS WEB STOREFRONT SYSTEM
Foosus Consumer
A sustainable food consumer who uses our platform to find and place orders on Food Offers from Suppleirs
Embeds https://foosus.example/recent
Visits foosus.https://example/store
Search
Cart & Check out
Search, Cart and Checkout
Most Recent Food Offers Embed App (Ruby on Rails)
Store Front Application (Container: Spring-boot)
Delivers to Consumer's Browser
Store Front Search Single Page Application; container (Container:JHipster/Angualr)
Links To
Store Front Orders Single Page Application; container (Container:JHipster/Angualr)
Store Front Mobile Application (Container:Angular-Ionic/)
Static Page with Most Recent Food Offers
Serves Front End Static Assets and Proxies to Backend
Frontend Web Application for Search
Frontend Web Application for Cart
Multi-platform Web Application for Search and Orders
Search
Cart and Checkout
Search, Cart and Checkout
An API targetted at UI needs, fulfilling the "backend for frontend" pattern
Storefront Web Search "Backend for Frontend "API Facade (Container:Django/Python)
Storefront Web Order "Backend for Frontend "API Facade (Container:Scala/Play Framework)
Storefront App "Backend for Frontend "API Facade (Container:Django/Python)
Get Product Details
Search
Cart and Checkout
Get User Details
Create Static Page
Foosus Inventory System
Foosus Search System
Foosus Order Processing System
Foosus Identity System
Get User Details

Technologie actuelle

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.

Les technologies utilisées dans la stack technique sont les suivantes :

- Ruby on rails
- Springboot
- Java
- JHipster
- Angular
- Angular-ionic
- Django
- Python
- Scala
- Play Framework

Bien que Java soit la compétence clé au sein des équipes plateforme, sélectionnées pour cela au moment du recrutement, la plateforme 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.

Définition de la Vision d'Architecture :

Vision

A court terme l'architecture restera dans son état actuel, hormis des travaux de correction pour maintenir la plateforme fonctionnelle, aucun nouveau développement ne sera entrepris lors de la phase de développement du prototype et du MVP.

A moyen terme, lorsque le MVP sera déployé, un double run aura lieu afin de maintenir les anciennes fonctionnalités non basculées et afin de permettre la migration de données et d'utilisateurs vers la nouvelle plateforme ainsi que la reprise des fonctionnalités toujours présente sur l'ancienne plateforme sur la nouvelle.

A long terme, décommissionnement de la plateforme historique et run unique sur la nouvelle plateforme.

Principes d'architecture définissant les directives générales.

La nouvelle plateforme devra répondre à des exigences de scalabilité, performances et de livraisons rapides. De plus, dans un premier temps elle devra fonctionner en parallèle de l'ancienne plateforme avant son décommissionnement.
Afin de répondre à ces exigences, une solution Micro-Service avec hébergement dans le cloud est proposé. Les détails de cette solution se trouve dans la section suivante.

Architecture Cible :

Description de l'architecture souhaitée à terme

L'architecture **micro services** est un style de conception logicielle où une application est décomposée en petits services autonomes qui travaillent ensemble pour fournir une fonctionnalité complète. Chaque service est indépendant, communique via des interfaces, peut être développé et déployé séparément, et offre une meilleure évolutivité et résilience. Cette approche est particulièrement adaptée aux applications complexes et évolutives.

Pour ce faire, chaque micro service doit être conteneurisé. La **conteneurisation** est une méthode qui permet d'encapsuler des applications et leurs dépendances dans des environnements isolés et portables, appelés conteneurs. Cela facilite le déploiement, la gestion et la portabilité des applications tout en offrant une isolation et une flexibilité accrues. Elle est essentielle pour le développement et le déploiement d'applications modernes.

Afin de gérer l'ensemble des conteneurs (un conteneur = un micro service) qui constitue notre application, nous utiliserons l'orchestration. **L'orchestration** pour une application micro services consiste à coordonner et à gérer les différents services micro services qui composent l'application, en s'assurant qu'ils fonctionnent harmonieusement ensemble. Elle inclut la gestion du déploiement, de la mise à l'échelle, de la supervision et de la coordination des services pour garantir la performance et la fiabilité de l'ensemble de l'application.

Afin de répondre au besoin de performance tout autour du monde, nous utiliserons le principe du cloud régionalisé. Le **cloud régionalisé** pour une application micro services signifie que l'application est déployée sur des centres de données dans différentes régions géographiques du cloud. Cela permet d'améliorer la disponibilité, la résilience et les performances de l'application en fonction de la localisation des utilisateurs, tout en garantissant que les services micro services sont répartis de manière stratégique pour servir efficacement différentes régions.

Afin d'automatiser le déploiement des nouvelles itérations nous utiliserons une **chaîne CI/CD**.

Une CI/CD (Intégration Continue/Livraison Continue) pour une application micro services est un processus automatisé qui permet de développer, tester et déployer rapidement et de manière cohérente chaque micro service, garantissant ainsi une mise à jour continue et efficace de l'application. Cela favorise l'agilité, la qualité et la stabilité de l'ensemble de l'application.

Le front-end sera développé en single page application et s'appuiera sur le principe du micro front-end.

Architecture Technique:

Spécification des standards technologiques

Les composants devront être développés selon les standards suivant :

Java 8 avec le framework Quarkus :

Quarkus est un framework Java natif pour Kubernetes complet, conçu pour les machines virtuelles Java (JVM) et la compilation native, qui permet d'optimiser Java spécifiquement pour les conteneurs afin d'en faire une plateforme efficace pour les environnements serverless, cloud et Kubernetes. Ce qui sera notre cas car nous avons fait le choix d'un environnements cloud.

Quarkus est conçu pour fonctionner avec les normes, frameworks et bibliothèques Java les plus utilisés, tels que Spring et Eclipse MicroProfile, ainsi qu'Apache Kafka, RESTEasy (JAX-RS), Hibernate ORM (JPA), Infinispan, Camel et beaucoup d'autres.

La solution d'injection de dépendances de Quarkus utilise la spécification CDI (Contexts and Dependency Injection). Quarkus inclut aussi un framework d'extension qui permet de l'enrichir et de configurer, de démarrer et d'intégrer un framework dans votre application. Il est aussi simple d'ajouter une extension qu'une dépendance. Vous pouvez sinon utiliser les outils de Quarkus.

Ce framework permet aussi la compilation native de vos applications avec GraalVM (une machine virtuelle universelle qui exécute des applications rédigées dans de nombreux langages, y compris Java et JavaScript).

React :

Pour le front-end, nous avons choisi le SPA React. La plateforme historique s'appuyait sur Angular, mais à la vue des dernières State of JS, sa perte de popularité et la baisse de son utilisation par le reste de l'industrie nous amène à choisir stratégiquement React qui bénéficie d'une popularité grandissante.

PostgreSqQL :

PostgreSQL est un système de gestion de base de données relationnelle et objet. C'est un outil libre disponible selon les termes d'une licence de type BSD. Ce système est comparable à d'autres systèmes de gestion de base de données, qu'ils soient libres, ou propriétaires.

Livraison de l’architecture et métriques business

Métrique	Technique de mesure	Valeur cible	Justification
Nombre d’adhésions d’utilisateurs par jour	Requêtage sur la plateforme et affichage sur un dashboard de suivi	Augmentation de 10%	Recul des inscriptions sur les derniers mois. C’est l’indicateur clé de la réussite du projet
Adhésion de producteurs alimentaires	Requêtage sur la plateforme et affichage sur un dashboard de suivi	Passer de 1,4/mois à 4/mois	Plus de producteurs c’est une attractivité supplémentaire pour attirer de nouveaux clients
Délai moyen de mise en production	Modification automatique de la date de « dernière MEP » sur le dashboard de suivi. Alerte en cas de dépassement de la valeur cible et valeur cible +2j	Réduit de 3,5 semaines à moins d'une semaine	Réduire la taille des MEP afin de réduire les impacts sur la plateforme Retour utilisateur plus aisé sur les fonctionnalités car moins « perdues » dans un lot de nouvelle fonctionnalités
Taux d'incidents de production P1	Suivi de l’uptime via les outils du clouder choisi (cloudwatch par exemple pour AWS)	Réduit de >25/mois à moins de 1/mois	Augmentation de la satisfaction clientèle et de la réputation de l’application
Délai de réponse des requêtes	Suivi du délai de réponses via les outils du clouder choisi (cloudwatch par exemple pour AWS)	Moins de 1 secondes en moyennes	Satisfaction utilisateurs Compatibilité maximum avec les connexions lentes et performances similaires entre les utilisateurs
Taux de conversion après une recherche	Requêtage sur la plateforme et affichage sur un dashboard de suivi	75%	Actuellement, le taux de conversion après une recherche est de 52%, c’est trop bas

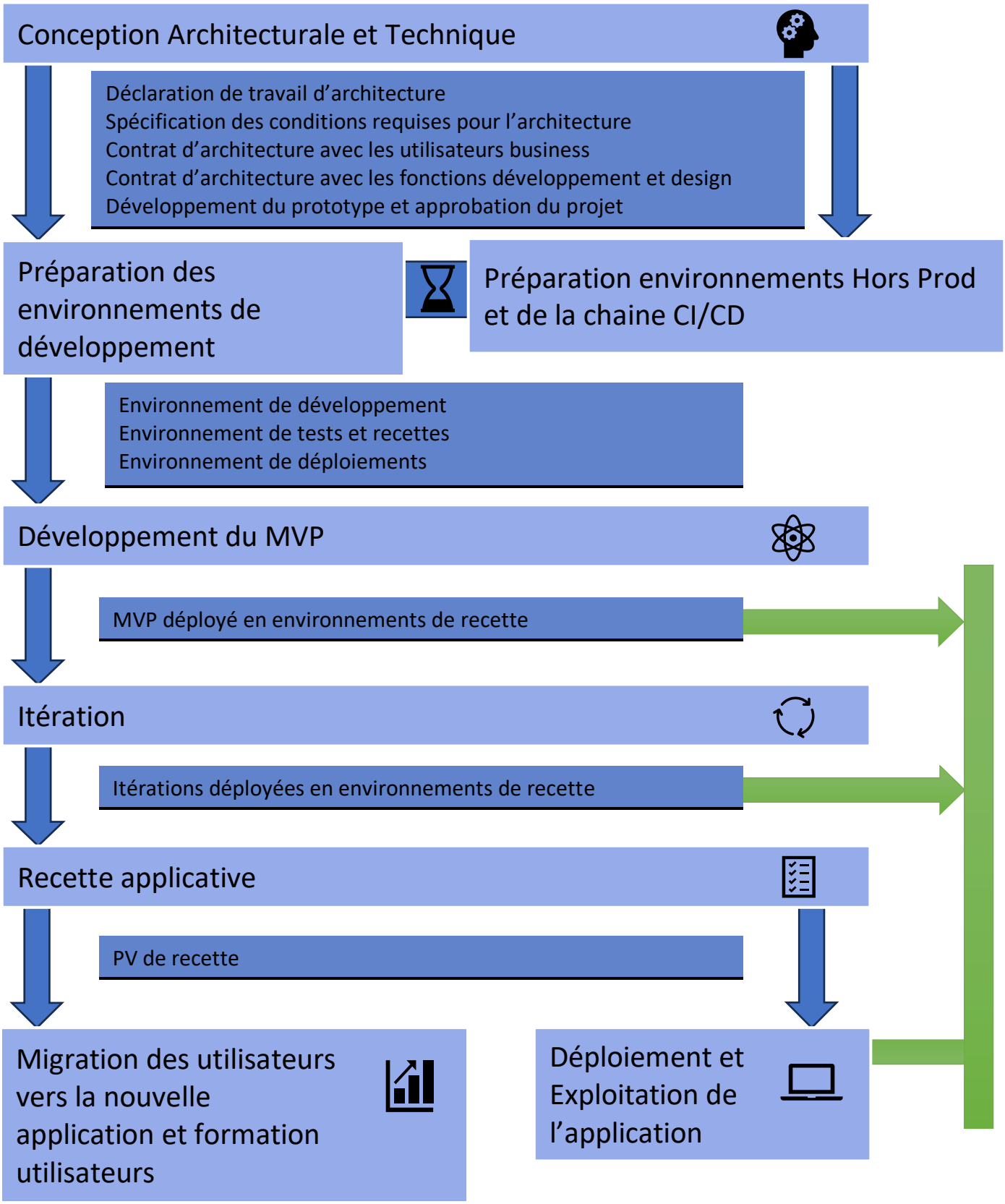
Phases de livraison définies

Ces éléments sont présents dans la Déclaration de travail d'architecture que vous pouvez trouver dans le repository d'artefacts architecturaux.

Chapitre :

- Approche Architecturale

Plan de travail commun priorisé

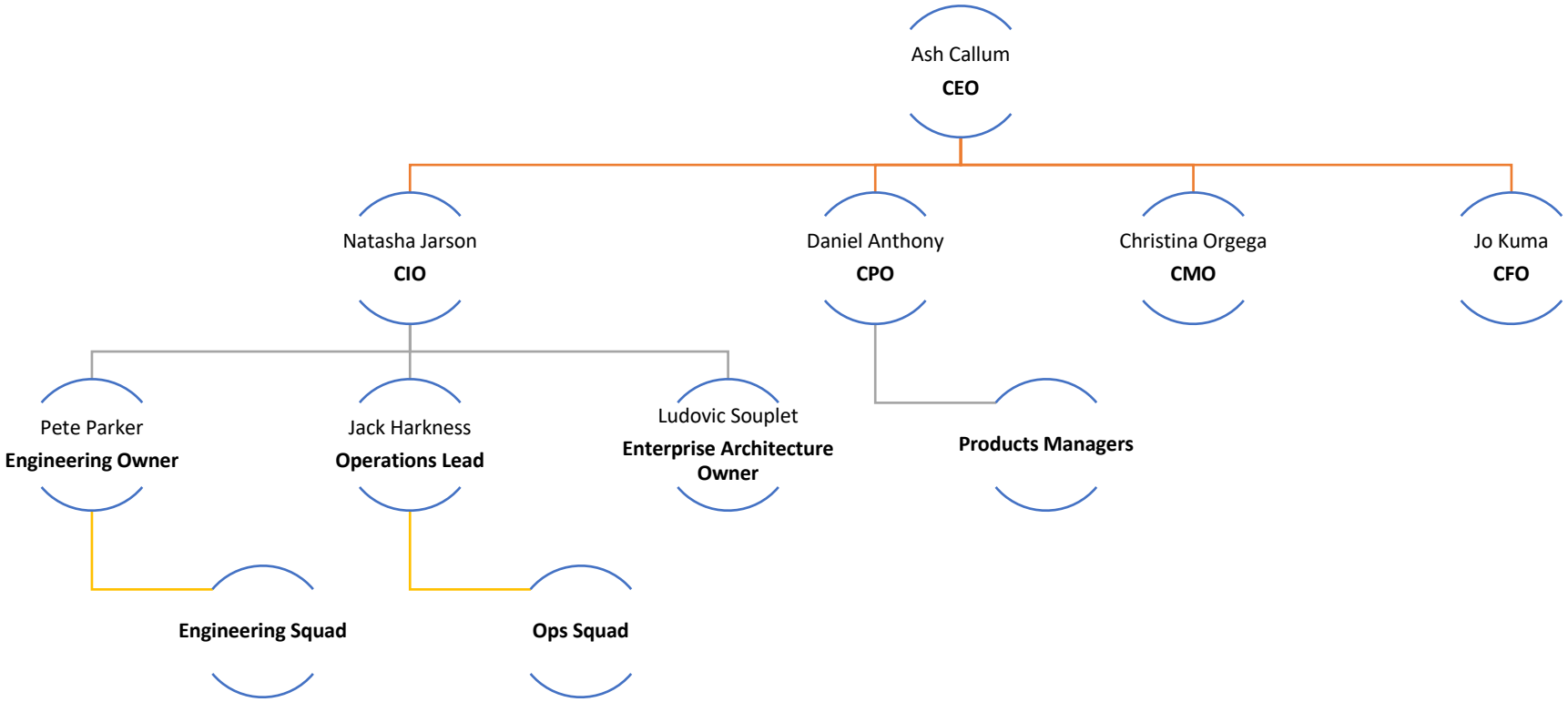


Plan de communication

Évènement	Destinataire	Canal	Format	Contenu	Fréquence
Communication sur le lancement des développements	Tous les salariés	Mail	Texte	Présentation de la stratégie macro et des fonctionnalités.	-
Communication sur le lancement des développements	Salariés impliqués dans le développement et l'exploitation	Physique	Meeting	Kick-off de lancement du projet. Présentation de la stratégie micro, de la stack technique, des fonctionnalités et de l'architecture cible	-
Communication sur l'état d'avancement du projet	Tous les salariées	Mail	Texte	Détails sur l'avancement du projet	Bimensuel
Fin du projet	Tous les salariés	Physique et virtuel	Meeting	Présentation de la nouvelle version	-
Fin du projet	Utilisateurs de l'application	Mail et Notifications Push	Texte	Information sur la nouvelle version avec date de disponibilité pour l'utilisateur Invitation à l'utiliser pour les utilisateurs ayant une certaine ancienneté	-
Communication interne à l'équipe	Équipes de développement	Jira et Confluence	Texte et tâches	Tâches de développements, documentation interne à l'équipe	-

Risques et facteurs de réduction

Structure de gouvernance



Analyse des risques

ID	Risque	Gravité	Probabilité	Action Préventive	Propriétaire
1.	Estimation erronée des travaux	4	2	Mener des discussions préliminaires avec les équipes de développement pour aligner la stratégie de développement	EAO
2.	Manque de collaboration entre les équipes	2	2	S'assurer, par le biais du Contrat de Conception et de Développement de l'Architecture, de l'engagement de toutes les équipes de développement à collaborer efficacement. Organiser des réunions régulières entre les équipes pour favoriser la communication	EAO EO
3.	Dépassement de budget	4	2	Mettre en place un suivi rigoureux du budget	EAO CFO
4.	Expertise des développeurs sur la nouvelle solution	2	3	Affecter des experts en technologie aux équipes de développement pour renforcer les compétences des développeurs moins familiers avec la technologie. Prévoir des sessions de formation, du pair programming et des sessions de revue de code	EO
5.	Non-respect des contrats établis dans les documents de Conception et de Développement de l'Architecture	4	2	S'assurer que tous les intervenants adhèrent aux documents lors de leur rédaction. Prévoir un plan de conduite du changement.	EAO EO

Hypothèses

Le tableau suivant résume les hypothèses pour cette Déclaration de travail d'architecture.

ID	Hypothèse	Impact
1.	Maintien de la plateforme actuel	Plus de développement sur la plateforme historique
2.	Intégrer les technologies historiques de la plateforme et permettre l'utilisation de nouvelles technologies	Concevoir une plateforme évolutive
3.	Accès des utilisateurs aux nouvelles fonctionnalités de manière progressive	Prévoir un double run (ancienne et nouvelle application) Prévoir les ressources nécessaires pour la montée en charge
4.	Élaboration sur mesure d'une approche architecturale type « lean »	Cela contribue à la bonne exécution de la feuille de route et cela évitera de priver les équipes de leur autonomie et de compromettre la rapidité des cycles

Critères d’acceptation et procédures

Métriques et KPIs de l’État Cible de l’Architecture

Métrique	Technique de mesure	Valeur cible	Justification
Nombre d’adhésions d’utilisateurs par jour	Requêtage sur la plateforme et affichage sur un dashboard de suivi	Augmentation de 10%	Recul des inscriptions sur les derniers mois. C’est l’indicateur clé de la réussite du projet
Adhésion de producteurs alimentaires	Requêtage sur la plateforme et affichage sur un dashboard de suivi	Passer de 1,4/mois à 4/mois	Plus de producteurs c’est une attractivité supplémentaire pour attirer de nouveaux clients
Délai moyen de mise en production	Modification automatique de la date de « dernière MEP » sur le dashboard de suivi. Alerte en cas de dépassement de la valeur cible et valeur cible +2j	Réduit de 3,5 semaines à moins d'une semaine	Réduire la taille des MEP afin de réduire les impacts sur la plateforme Retour utilisateur plus aisé sur les fonctionnalités car moins « perdues » dans un lot de nouvelle fonctionnalités
Taux d'incidents de production P1	Suivi de l’uptime via les outils du clouder choisi (cloudwatch par exemple pour AWS)	Réduit de >25/mois à moins de 1/mois	Augmentation de la satisfaction clientèle et de la réputation de l’application
Délai de réponse des requêtes	Suivi du délai de réponses via les outils du clouder choisi (cloudwatch par exemple pour AWS)	Moins de 1 secondes en moyennes	Satisfaction utilisateurs Compatibilité maximum avec les connexions lentes et performances similaires entre les utilisateurs
Taux de conversion après une recherche	Requêtage sur la plateforme et affichage sur un dashboard de suivi	75%	Actuellement, le taux de conversion après une recherche est de 52%, c’est trop bas

Procédure d’acceptation

L'approbation du projet est conditionnée par les deux éléments suivants :

La confirmation de la conformité des livrables par les parties prenantes à la fin de chaque étape du cycle ADM
La présentation d’un prototype afin de vérifier la viabilité de l'architecture proposée

Procédures de changement de périmètre

L’organisation en Agile permet les changements de périmètre, ce qui implique la mise à jour de tous les artefacts concernés. Voici la procédure à suivre :

1. **Identification des changements potentiels de périmètre :**
   Commencez par établir une liste des éléments qui pourraient être sujets à des changements de périmètre. Cela peut inclure des besoins du client qui évoluent, des priorités qui changent, des découvertes lors du développement, etc.

2. **Évaluation de l'impact :**
   Pour chaque changement potentiel de périmètre, évaluez son impact sur les artefacts existants. Cela pourrait inclure des documents d'architecture, des diagrammes, des spécifications techniques, des plans de développement, etc. Identifiez quels artefacts sont susceptibles d'être affectés.

3. **Création d'une demande de changement :**
   Lorsqu'un changement de périmètre est identifié et que son impact est compris, créez une demande de changement formelle. Cette demande devrait inclure une description du changement, les raisons de ce changement, l'impact attendu et les besoins du client.

4. **Évaluation et approbation :**
   La demande de changement doit être évaluée par les parties prenantes concernées, y compris l'équipe d'architecture, les développeurs, le client, etc. Une décision doit être prise quant à l'approbation ou au rejet du changement. Cela peut nécessiter des discussions et des négociations pour parvenir à un consensus.

5. **Mise à jour des artefacts :**
   Si le changement de périmètre est approuvé, identifiez les artefacts spécifiques qui doivent être mis à jour en conséquence. Cela peut inclure des révisions de documents, des ajustements de plans, des modifications de diagrammes, etc.

6. **Documentation du changement :**
   Toutes les modifications apportées aux artefacts doivent être documentées de manière claire et précise. Cela garantit que toutes les parties prenantes sont informées des changements et de leur justification.

7. **Communication :**
   Communiquez les modifications aux membres de l'équipe, au client et à toute autre partie prenante concernée. Assurez-vous que tout le monde est au courant des changements apportés au périmètre et à leurs implications.

8. **Suivi et vérification :**
   Assurez-vous que les modifications sont mises en œuvre conformément à la demande de changement. Effectuez des vérifications pour garantir que les artefacts ont été mis à jour correctement et que le périmètre est aligné sur les nouvelles exigences.

9. **Gestion des versions :**
   Si nécessaire, gérez les versions des artefacts pour suivre les changements apportés et disposer d'un historique des versions précédentes.

10. **Réévaluation continue :**
    Tout au long du projet, continuez d'évaluer les besoins et les changements potentiels de périmètre. Adaptez votre procédure de changement de périmètre en conséquence pour une gestion agile et réactive des modifications.

En mettant en place cette procédure de changement de périmètre, nous serons en mesure de gérer efficacement les modifications tout en maintenant l'intégrité des artefacts d'architecture dans notre contexte Agile.

Conditions requises pour la conformité

La conformité sera validée lorsque les éléments compris dans l’artefact « Statement of architecture work » compris dans notre repository seront validés.

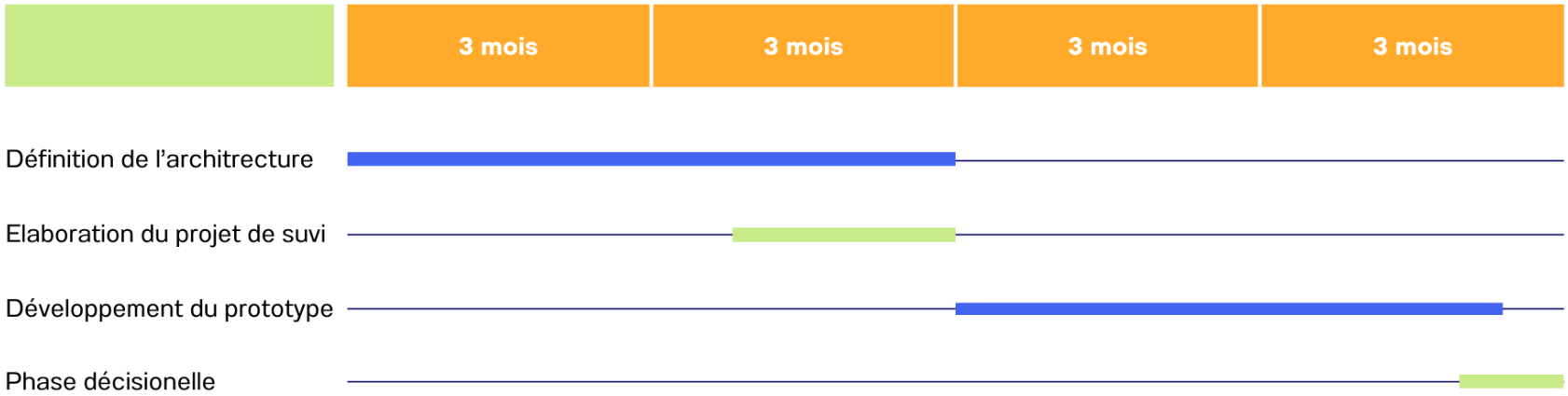
Développement et propriété de l'architecture

	CEO	CIO	CPO	EO	EAO	OL	DT
Conception architecture nouvelle application	A	A	A	C	R	C	I
Développement de la nouvelle application	I	A	A	C	C	C	R
Déploiement de la nouvelle application	I	A	A	C	C	R	C
Exploitation de l'application	I	I	I	C	I	R	C

(R)esponsable
(A)pprobateur
(C)onsulté
(I)nformé

EO : Engineering Owner
EAO : Enterprise Architecture Owner
OL : Operations Lead

Calendrier et Phases de livrables définies



Personnes approuvant ce plan

Ash Callum CEO	**Natasha Jarson** CIO	**Daniel Anthony** CPO	**Jo Kumar** CFO	**Christina Orgega** CMO
Date et signature :	Date et signature :	Date et signature :	Date et signature :	Date et signature :