

# Spécification des conditions requises pour l'architecture



Dylan J. Gerrits

## Tables des matières

1-	Information sur le document .....	4
2-	Objet de ce document .....	5
3-	Mesures du succès .....	6
	Indicateurs de réussite .....	6
	SLA, SLO et SLI .....	6
4-	Conditions requises pour l'architecture .....	6
5-	Contrats de service .....	7
A-	Accords de niveau de service (SLA) .....	7
	Lexique .....	7
	Disponibilité .....	8
	Gestion des problèmes .....	8
	Incidents   Temps de réponse et temps de résolution .....	10
B-	Objectifs de niveau de service (SLO) et indicateurs de niveau de service (SLI)	10
	Latence .....	11
	Disponibilité .....	11
6-	Lignes directrices pour l'implémentation .....	12
7-	Spécifications pour l'implémentation .....	14
A-	Cloud .....	14
B-	Géolocalisation .....	14
C-	Micro-services .....	15
	Extraction de fonctionnalités .....	15
	Ajout futur de fonctionnalités .....	16
8-	Standards pour l'implémentation .....	17
A-	Technologies et outils .....	17
	Frameworks .....	17
	Base de données .....	18
	Containerisation .....	<b>Error! Bookmark not defined.</b>
	Communication .....	19
	Versionning .....	19
B-	Clean code .....	20
9-	Conditions requises pour l'interopérabilité .....	21
	Interopérabilité technique et sémantique .....	21
	Interopérabilité organisationnelle .....	22
10-	Conditions requises pour le management du service IT .....	24

11-	Contraintes .....	25
12-	Hypothèses .....	26
13-	Approbations signées.....	27

# 1- Information sur le document

Nom du projet	Foosus - Conception d'une nouvelle architecture
Préparé par :	Dylan J. Gerrits
N° de version du document :	1.0
Titre :	Spécification des conditions requises pour l'architecture
Types d'action :	Approbation, Révision, Information, Classement, Action requise, Participation à une réunion

## 2- Objet de ce document

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

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

Comme mentionné ci-dessus, la spécification des conditions requises pour l'architecture accompagne le document de définition de l'architecture, avec un objectif complémentaire : le document de définition de l'architecture fournit une vision qualitative de la solution et tâche de communiquer l'intention de l'architecte.

La spécification des conditions requises pour l'architecture fournit une vision quantitative de la solution, énumérant des critères mesurables qui doivent être remplis durant l'implémentation de l'architecture.

## 3- Mesures du succès

### Indicateurs de réussite

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

### SLA, SLO et SLI

## 4- Conditions requises pour l'architecture

L'entreprise veut construire une solution géo-ciblée avec une nouvelle architecture.

L'entreprise a besoin de frontières claires pour pouvoir développer une plateforme qui permette de l'innovation rapide et de se mettre à l'échelle. Cela consiste à l'élaboration d'une architecture stratégique pour le nouveau projet et ceux qui suivront. Cela passe par la mise en place une certaine standardisation pour la maintenance des développements.

L'objectif est de créer une plateforme de e-commerce polyvalente pour faire passer l'entreprise à un niveau supérieur. L'efficacité, la flexibilité et des approches cohérentes dans la prise de décision sont nécessaires pour pouvoir concurrencer les grandes entreprises mondiales de e-commerce qui dominent le marché de l'alimentation durable. Les principaux objectifs de l'entreprise sont les suivants :

- La solution doit 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.
- L'architecture doit être évolutive pour permettre aux services de se déployer sur diverses régions à travers des villes et des pays donnés.
- La solution doit être disponible pour les fournisseurs et les 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.

- La solution doit pouvoir prendre en charge différents types d'utilisateurs avec des fonctionnalités et des services spécifiques pour ces catégories.

La nouvelle plateforme devra également permettre aux équipes d'innover rapidement en réorientant des solutions existantes, en expérimentant de nouvelles modifications et en facilitant l'intégration avec des partenaires internes et externes.

La plateforme doit facilement s'adapter aux particularités locales et répondre aux exigences d'utilisation de nos clients. Les utilisateurs situés dans différentes régions doivent pouvoir espérer des performances similaires. L'entreprise souhaite cibler les consommateurs dans des zones géographiques spécifiques, sur des connexions lentes aussi bien que sur des réseaux haut débit.

Il est nécessaire d'adopter une approche qui garantisse la sécurité à chaque évolution de la plateforme.

Même si le système est surchargé, les utilisateurs connectés doivent pouvoir continuer à accéder à tous les services de façon dégradée. Chaque nouvelle version doit être de taille réduite, présenter peu de risques, être transparente pour les utilisateurs et le système doit rester accessible en tout lieu et à tout moment.

## 5- Contrats de service

### A- Accords de niveau de service (SLA)

Les accords de niveau de service spécifient des engagements qui sont des niveaux de service définis entre le fournisseur de services et le client. Les engagements SLA peuvent être mesurés de manière qualitative ou quantitative. Ils peuvent être associés à une ou plusieurs escalades qui définissent les actions nécessaires lorsque l'engagement n'est pas respecté.

### Lexique

#### Dégradation des performances

Une qualité inférieure de service telle que décrite dans le présent SLA (comme une fonctionnalité temporairement en panne ou temporairement indisponible).

#### Indisponibilité

La période pendant laquelle le système est totalement indisponible, y compris la

maintenance, réalisée en dehors des heures de maintenance, notifiée moins de 24 heures au préalable. Cependant, l'indisponibilité ne comprend pas la maintenance programmée.

Il est à noter que des facteurs échappant au contrôle de Foosus, y compris les cas de force majeure, les pannes d'internet, ou encore l'application de règlements de l'État ou du gouvernement.

### **Temps de résolution**

Le temps qui s'écoule entre le temps de réponse et la résolution de l'alerte.

### **Temps de réponse**

Mesure le temps qui s'écoule entre la réception d'une alerte et l'heure de commencement des travaux pour résoudre le problème.

### **Entretien programmé | Maintenance programmée**

Coupures planifiées, suspendant le service en tout ou en partie, que Foosus s'efforcera d'annoncer au moins 5 jours à l'avance, et en tout cas pas moins de 24 heures à l'avance, qui ne dépassera pas une période raisonnable pour la maintenance requise.

### **Ticket**

Une demande électronique envoyée à Foosus par un utilisateur (demandant une solution à un incident).

## **Disponibilité**

Foosus garantit une disponibilité internationale, avec un temps utilisable de 99% chaque mois, 24 heures sur 24, 7 jours sur 7.

*Le temps utilisable est mesuré en se basant sur la moyenne de disponibilité par mois, arrondie à la minute inférieure.*

Le déploiement de nouvelles versions de la plateforme devra avoir lieu, dans la mesure du possible, sans interruption de services.

Cela dit, si cela s'avère vraiment nécessaire, des coupures peuvent être planifiés, suspendant le service en tout ou en partie, que Foosus s'efforcera d'annoncer à l'avance, et qui ne dépassera pas une période raisonnable pour la maintenance requise. Ces coupures, si nécessaires, devront avoir lieu en considérant les heures d'affluence avec, notamment, des déploiements différés selon le fuseau horaire.

## **Gestion des problèmes**

Les équipes de Foosus s'engagent à analyser régulièrement tous les tickets des utilisateurs de façon à identifier les tendances et les goulets d'étranglement. Sur la base de ce



constat, une base de connaissances doit régulièrement être mise à jour avec des informations expliquant la solution aux erreurs connues.

Il est envisageable de partager publiquement tout ou en partie de la base de connaissances pour permettre aux utilisateurs d'avoir connaissance des éléments problématiques.

On distingue différents types de ticket :

- Les problèmes techniques sont liés à un bug particulier, des pannes de sécurité ou de sauvegarde, ou tout autre type de dysfonctionnement de la plateforme.
- Les questions des utilisateurs proviennent de cas où la plateforme n'est pas suffisamment intuitive.
- Les demandes concernent des changements au sein de la plateforme, notamment au niveau des fonctionnalités ou des réglages.
- Les questions de contenu ont trait aux contenus des données des utilisateurs. C'est Foosus qui est responsable de ses données, aussi bien en base de données qu'affichées publiquement sur la plateforme.

# Incidents | Temps de réponse et temps de résolution

Le temps de réponse et le temps de résolution dépendront de la priorité de ou des éléments affectés et de la gravité de l'alerte, comme indiqué dans le programme suivant :

Niveau de sévérité	Description	Temps de réponse	Temps de résolution
Extrême	Alerte extrêmement critique. La plateforme n'est pas disponible à l'emploi ou une part significative des fonctionnalités contractées n'est pas disponible.	Moins d'une heure	Dans les 4 heures
Critique	Alerte critique. Un ou plusieurs éléments de la plateforme ont cessé de répondre totalement ou répond de façon extrêmement lente.	Dans les 4 heures	Dans les 12 heures
Important	Alerte non-critique. Un ou plusieurs éléments de la plateforme ont cessé de répondre complètement ou répondent lentement et une solution est disponible.	Dans les 12 heures	Dans les 24 heures
Informatif	Notification de problème mineur qui n'empêche pas l'utilisation de la plateforme.	Dans les 24 heures	Dans les 48 heures

## B- Objectifs de niveau de service (SLO) et indicateurs de niveau de service (SLI)

Foosus, en tant qu'entité internationale doit intégrer la notion de fuseau horaire dans son approche. Pour cela le site ne doit pas souffrir de latence et donc avoir un tube d'accès conséquent pour répondre au pic de sollicitation et cela quel que soit le pays.

Foosus devra s'assurer de disposer de copie de son système d'information au sein de différents centres de données à travers le monde, avec une gestion automatisée concernant la mise à l'échelle horizontale pour une adaptation en fonction de l'affluence

et la charge sur chaque service.

Il est recommandé l'utilisation d'indicateurs variées pour mesurer le niveau de service. Il est possible d'obtenir des indicateurs concernant la disponibilité et la latence de la plateforme en examinant les requêtes envoyées vers le serveur et les réponses obtenues.

## Latence

La latence de réponse sera mesurée en effectuant la moyenne des délais de réponse des requêtes, avec un temps de réponse moyen du serveur qui devrait être en dessous de 250ms pour 90% des requêtes et en dessous de 750ms pour 95% des requêtes. 99% des requêtes devront avoir un temps de réponse moyen du serveur en dessous de 1250ms.

Le temps de chargement moyen des pages internet côté utilisateur devra aussi faire l'objet de mesures, afin de permettre une navigation fluide sur tout type d'appareil et tout type de connexion, avec un temps de chargement moyen des pages internet qui devrait être en dessous de 2500ms pour 90% des utilisateurs, et en dessous de 7500ms pour 95% des utilisateurs. 99% des utilisateurs devraient avoir un temps de chargement moyen des pages internet en dessous de 12500ms.

Les requêtes avec un temps de réponse trop élevés devront faire l'objet d'une attention particulière, avec l'ouverture d'un incident, afin d'optimiser ces délais. Il est à noter que les deux sources les plus communes sont l'absence d'un système de cache ou un manque de capacité du côté serveur (mémoire, puissance de calcul, bande passante).

## Disponibilité

Afin de disposer de données concrètes sur les temps d'indisponibilité ou de dégradation des performances, il est important de conserver un suivi des différents incidents, et les de documenter. Le nombre, la durée et la sévérité des interruptions permettra d'indiquer avec précision le niveau de service.

L'indisponibilité totale de la plateforme, par mois, ne devra pas excéder 7,31 heures (99%).

L'indisponibilité et la dégradation des performances, cumulée des services, par mois, ne devra pas excéder 36 heures (95%). Ainsi, que cela concerne une partie ou toutes les fonctionnalités, le décompte de ce temps sera effectué en effectuant la somme des durées des différentes interruptions.

La disponibilité sera, entre autres, mesurée en examinant la proportion de requêtes avec, en réponse, un statut autre que 500-599. Cette proportion devrait être d'au moins 99%.

Toute requête renvoyant un statut 500-599 provoquera l'ouverture d'un incident.

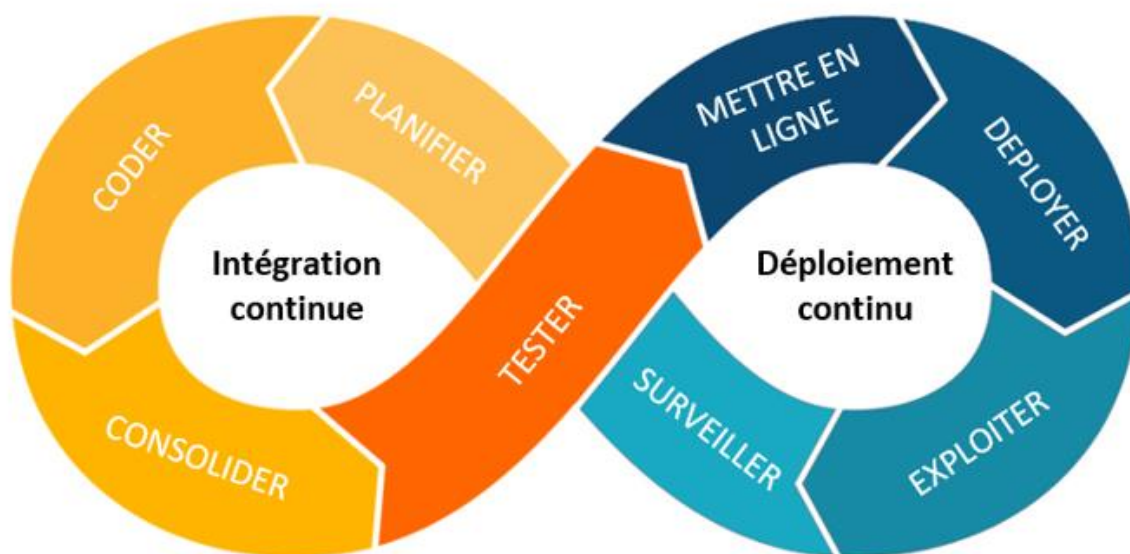
## 6- Lignes directrices pour l'implémentation

Une liste des directives relatives au projet a été fournie par Foosus :

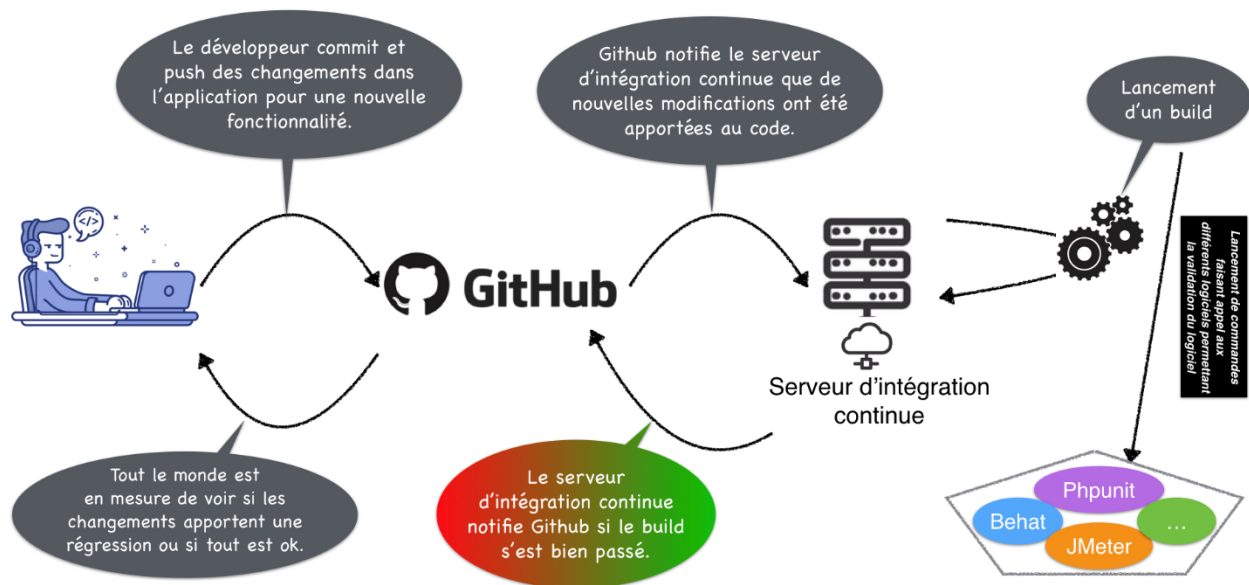
- Les solutions open-source sont préférables aux solutions payantes.
- Le support continu des composants doit être pris en compte lors de leur sélection ou lors des prises de décision de création ou d'achat.
- Toutes les solutions du commerce ou open source doivent, dans la mesure du possible, faire partie d'une même pile technologique afin de réduire les coûts de maintenance et de support continus.

Le déploiement de nouvelles versions de la plateforme devra avoir lieu, dans la mesure du possible, sans interruption de services.

L'implémentation s'inscrit dans une démarche d'intégration et de déploiement continu.

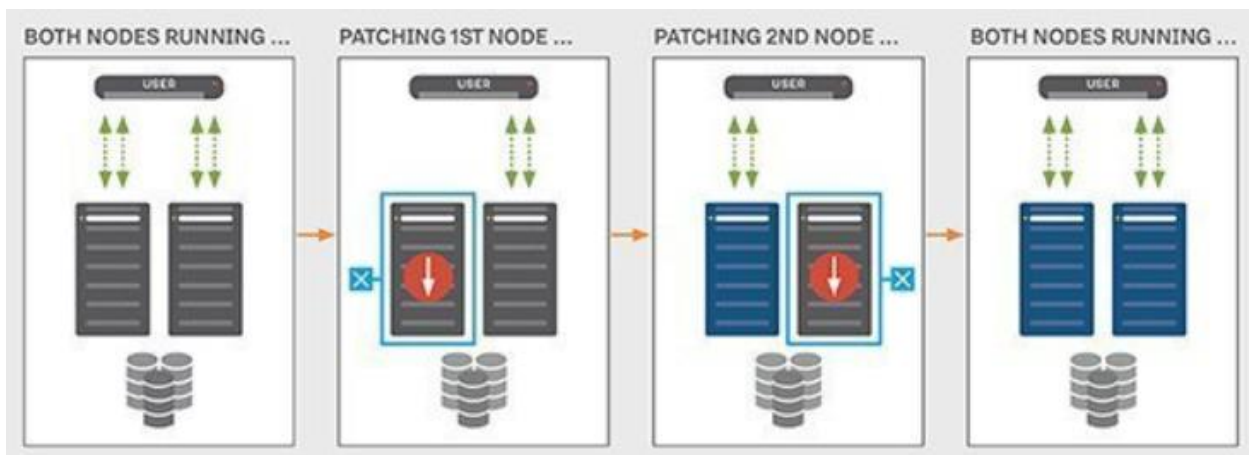


Ainsi, il convient de tester chacune des nouvelles fonctionnalités, avant de la mettre en ligne et la déployer. Pour cela, il est pertinent de mettre en place des tests automatisés pour vérifier et valider chacune des modifications. Le code sera systématiquement validé sur plusieurs environnements différents notamment sur un environnement de *build*.



La stabilité du système sera, entre autres, garantie par la duplication des instances d'un micro-service et la substitution d'une instance par une autre en cas de panne. Cette technique sera également utilisée pour garantir le fonctionnement du service lors des mises à jour système.

Pour mettre en ligne et déployer une nouvelle version d'un micro-service, les équipes échelonnent les modifications de sorte que la mise à jour s'active sur certains serveurs ou instances avant d'autres, avant d'y acheminer une partie du trafic. Au fur et à mesure du déploiement, le trafic est redirigé vers les serveurs ou instances à jour.



Le système doit permettre un déploiement rapide et économiquement viable dans de nouvelles zones géographiques.

# 7- Spécifications l'implémentation

pour

## A- Cloud

Il semble pertinent de déployer le système d'information sur le *cloud*. Cela offre de nombreux avantages en termes d'agilité, d'accessibilité, de rapidité et de simplicité.

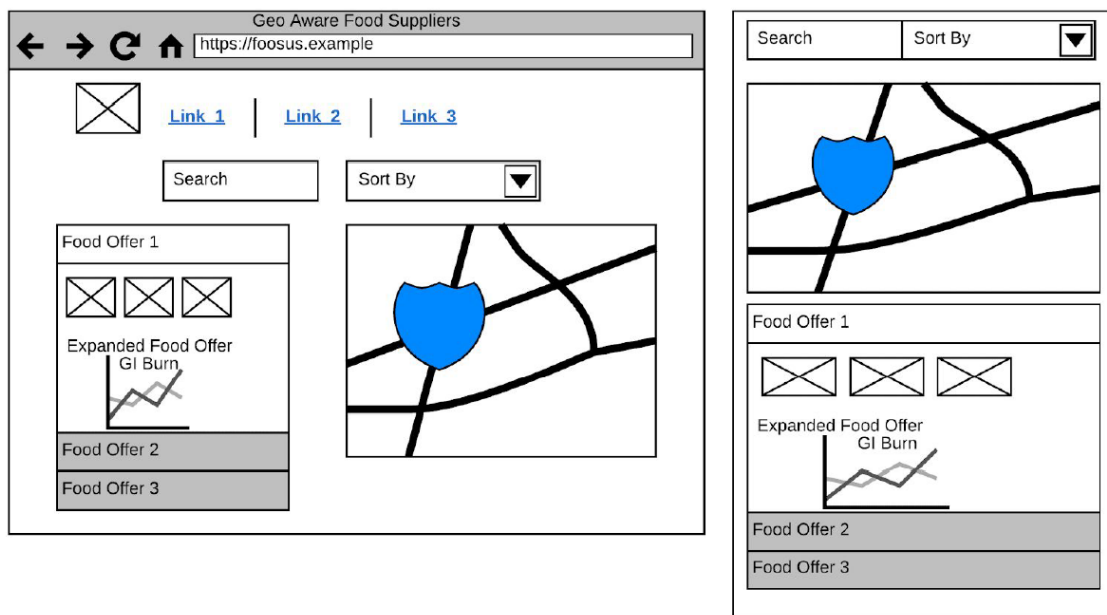
Dans un premier temps, il suffira de déployer l'architecture historique sur le *cloud* et de la dupliquer. L'une des architectures ne sera pas modifiée, et restera en mode maintenance, tandis que les évolutions seront apportées à une autre à travers différentes itérations jusqu'à complétion de l'architecture cible de transition.

La décision concernant le type de *cloud* et le fournisseur de service de *cloud* doit faire l'objet d'une analyse plus approfondie et de discussions entre les différentes parties prenantes.

## B- Géolocalisation

Afin d'intégrer la géolocalisation, il existe deux scénarios possibles :

- L'utilisateur accepte d'être géolocaliser en acceptant le pop-up sur son navigateur.
- L'utilisateur préfère écrire son adresse manuellement (où choisi une autre adresse).



Toute conception doit cependant tenir compte des éléments suivants :

- 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.

La géolocalisation permettra d'introduire d'autres innovations en fonction de l'emplacement de l'utilisateur ou du fournisseur alimentaire.

Il est considéré d'implémenter le système de recherche comme une API rattachée au système d'inventaire, utilisant les services externes de Google Maps Platform pour la géolocalisation.



Google Maps Platform inclut de nombreux outils de visualisation afin d'afficher des cartes au sein d'un navigateur internet ou d'une application mobile. Il est possible de faire communiquer le service « Foosus Search System » avec les API à disposition, notamment Distance Matrix, afin de filtrer les résultats et afficher les offres à proximité de l'utilisateur, ou de l'adresse renseignée manuellement.

## C- Micro-services

Il convient de disposer d'une architecture qui permette l'ajout de nouveaux composants sans impacter ceux déjà existant. Cela suppose donc un couplage faible et une forte cohésion.

### Extraction de fonctionnalités

L'ensemble des composants de l'architecture historique sont à mettre à niveau afin d'en extraire des micro-services distincts avec une pile technologique commune. Ceux-ci doivent respecter, au mieux, le principe de responsabilité unique.

La mise à niveau des composants s'inscrit dans la vision d'une architecture évolutive, et doit s'effectuer tout au long des itérations, notamment en fonction des besoins pour l'implémentation de nouvelles fonctionnalités.



Cela passe par l'extraction d'une ou plusieurs fonctionnalités au sein d'un composant existant, et des données relatives à ce composant, pour développer un micro-service unique prenant en charge cette fonctionnalité. Le composant historique pourra alors être désactiver,

## Ajout futur de fonctionnalités

Des fonctionnalités, qui ne sont pas prioritaires pour le moment, sont envisagés concernant notamment l'intégration de système de paiements via des prestataires tiers mais aussi une gestion de toutes les communications avec les fournisseurs alimentaires au sein d'une interface utilisateur personnalisée.

Ces micro-services sont considérés dans la vision d'une architecture évolutive, et donc les choix subséquents pour l'implémentation de celle-ci, mais n'entre pas dans le périmètre du projet actuel.



## 8- Standards l'implémentation

**pour**

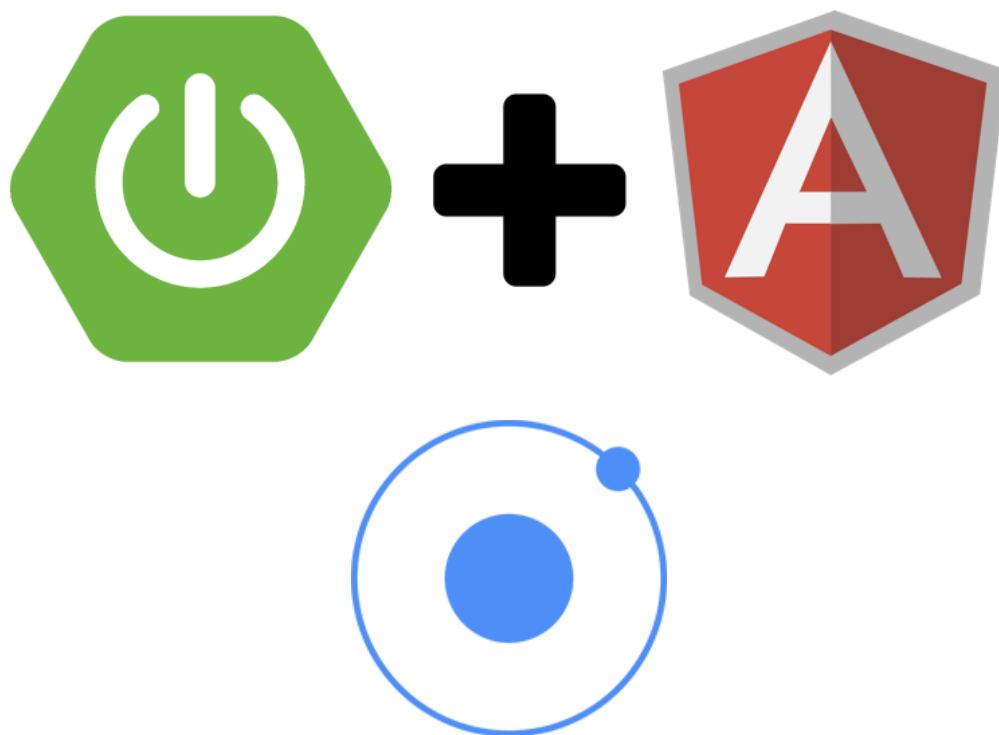
Différents principes ont déjà été mis en avant par Foosus. Ceux-ci sont énumérés dans le document de [déclaration de travail d'architecture](#) (7-B- Méthodologies pertinentes et normes de l'industrie).

Il semble important, à la suite des problèmes reportés par Foosus, d'apporter un cadre standardisé concernant l'implémentation des nouvelles fonctionnalités et évolutions futures.

### A- Technologies et outils

Toutes les solutions du commerce ou open source doivent, dans la mesure du possible, faire partie d'une même pile technologique afin de réduire les coûts de maintenance et de support continus.

#### Frameworks



Malgré le fait qu'une architecture de micro-services offre une flexibilité concernant le choix de technologie, il apparaît pertinent d'offrir plus de flexibilité au sein de l'organisation, via une standardisation des technologies utilisées. En effet, l'utilisation d'une base commune permettrait aux différents développeurs de facilement de collaborer

Concevez une nouvelle architecture afin de soutenir le développement de votre entreprise  
[Foosus] Spécification des conditions requises pour l'architecture / 17

et s'adapter à différents projets.

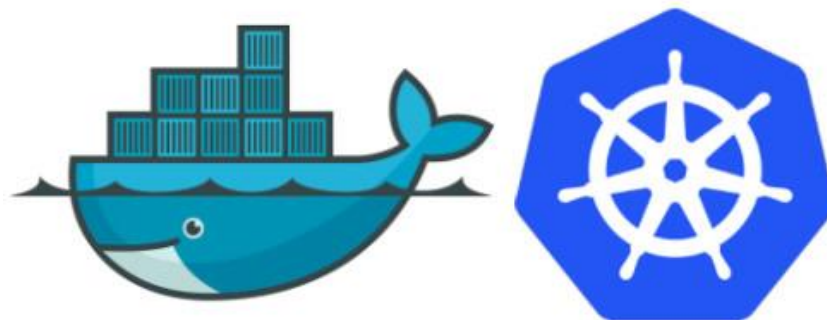
Foosus est déjà familier avec les technologies Java, notamment Spring Boot, ainsi qu'Angular et Ionic. Il semble donc pertinent de standardiser l'écriture des différents composants en utilisant ces frameworks.

## Base de données



Une architecture de micro-service suppose l'existence de micro-services autonomes avec leurs propres bases de données. Il semble pertinent, en considérant l'absence de relations directes entre les bases de données, d'utiliser des bases de données NoSQL, qui peuvent être mises en place et mises à l'échelle rapidement. Il est proposé d'utiliser MongoDB.

## Conteneurisation



Alors que Docker permet de créer des conteneurs, Kubernetes permet l'orchestration et la gestion de ces conteneurs. Il est possible d'utiliser Docker afin de conteneuriser chaque composant, notamment chaque micro-service, du système d'information, et Kubernetes pour le déploiement et la mise à l'échelle de ces composants.

Si Foosus dispose de peu de conteneurs, il est possible de les gérer sans Kubernetes, mais cela devient beaucoup plus difficile lorsque le nombre de conteneurs augmente. Afin de considérer les futures évolutions, il semble pertinent d'utiliser les Docker et Kubernetes conjointement.

Ensemble, ces deux outils constituent une part essentielle de l'architecture cloud moderne et de la transformation numérique. Ils sont désormais couramment utilisés conjointement pour accélérer le déploiement et la relance d'applications.

## Communication



Les processus de communication et d'échange des données de l'architecture sont clairement établies et univoques. L'implémentation d'une architecture de micro-services communiquant à travers des API REST en JSON est la norme définie pour la conception de l'architecture.

## Versionning



Il est recommandé d'utiliser Git pour le contrôle des versions, avec GitHub.

Il semble pertinent de disposer d'un dépôt principal, pour le cœur du système, autour duquel gravitent d'autres dépôts dédiés aux différents micro-services. Chaque dépôt pourra déposer de différentes branches correspondant aux différents environnements, permettant, entre autres, de tester les nouvelles fonctionnalités avant déploiement en

production.

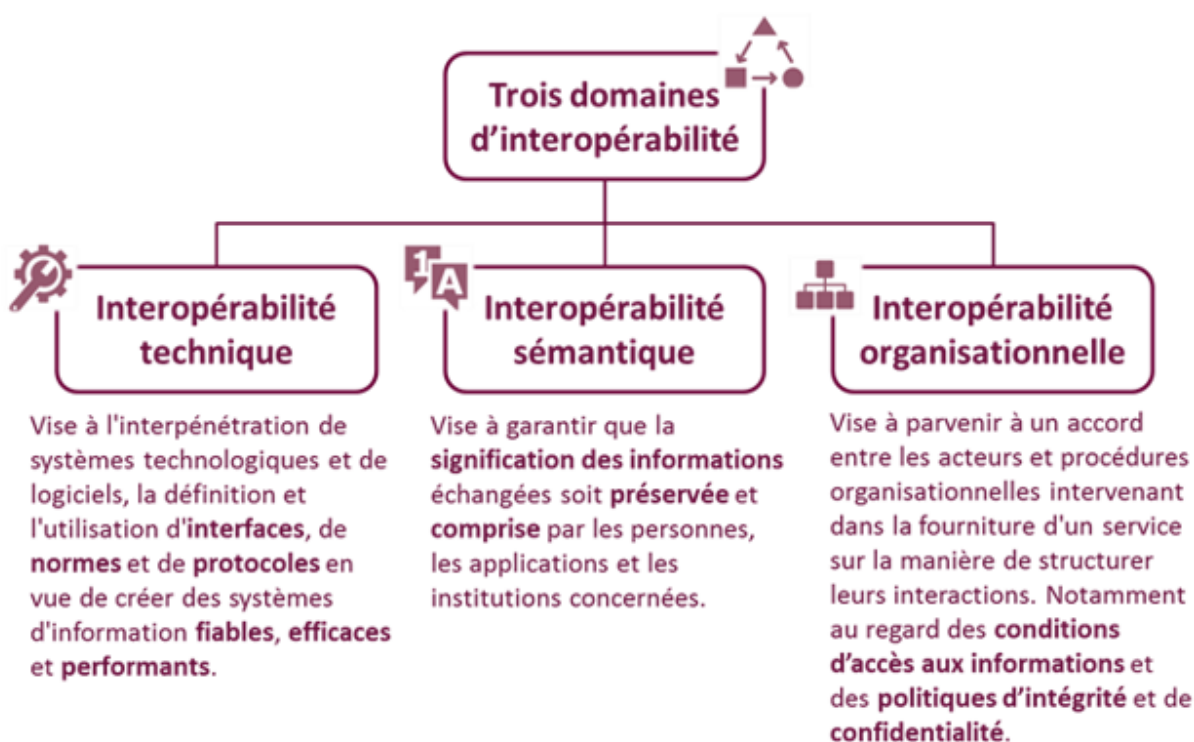
## B- Clean code

Une standardisation des pratiques liées à l'écriture du code apporterait une valeur ajoutée à la start-up. Le code doit être élégant, efficace, lisible, simple, sans duplications et bien écrit afin d'offrir qualité et compréhension. Il est nécessaire que le code de chacun soit propre et lisible pour que tout le monde puisse le comprendre facilement et éviter de faire perdre du temps aux autres. Cela passe notamment par la mise en place de conventions de nommages pour l'ensemble des éléments.

## 9- Conditions requises pour l'interopérabilité

L'interopérabilité est ainsi un terme informatique désignant des systèmes capables de s'adapter et de collaborer avec d'autres systèmes indépendants déjà existants ou encore à créer.

Foosus dispose de différents principes, visant à soutenir l'innovation et l'agilité grâce à l'extensibilité et soutenir sa réputation grâce à la stabilité. Les décisions sont pilotées par le feed-back et l'apprentissage, avec des choix qui soutiennent les objectifs long terme. La start-up est consciente que des erreurs peuvent être amenées à se produire et cherche s'assurer de concevoir une architecture qui permet de commettre ces erreurs rapidement, pour en tirer des enseignements.



### Interopérabilité technique et sémantique

Les lignes directrices, ainsi que les standards évoqués pour l'implémentation, que ce soit au niveau des technologies et outils ou des pratiques de « clean code » sont amenés à favoriser l'interopérabilité technique et sémantique, mais aussi opérationnelle.

Foosus souhaite disposer d'une architecture tenant compte du principe de responsabilité unique et couplage faible des composants.

Il convient de concevoir des interfaces ouvertes et extensibles en systèmes, sur

lesquelles il est facile d'itérer, et d'appliquer une approche pilotée par le contrat client, où les interfaces entre les composants, micro-services, reflètent uniquement les données et opérations nécessaires à leur intégration. Il faut éviter les dépendances cycliques entre les composants.

Il semble important de documenter chaque micro-service en listant notamment l'ensemble des points d'accès aux APIs, la structure des requêtes et des réponses. Cela peut être fait en utilisant, par exemple, la norme [OpenAPI](#).

Il faut s'assurer que tous les composants de l'architecture sont conçus pour être faciles à cataloguer et à ne pas perdre de vue. Il convient de privilégier la prévisibilité et la répétabilité plutôt que le non-déterminisme.

Les choix technologiques, bien que standardisés, doivent être ouverts et aisés à modifier. Il est important d'être raisonné et de toujours considéré le choix entre développer un ou plusieurs composants de l'architecture, utiliser ou acheter un service déjà existant (tel que les services Google Maps par exemple).

Les choix technologiques doivent s'aligner sur la capacité et la correspondance avec le métier.

## Interopérabilité organisationnelle

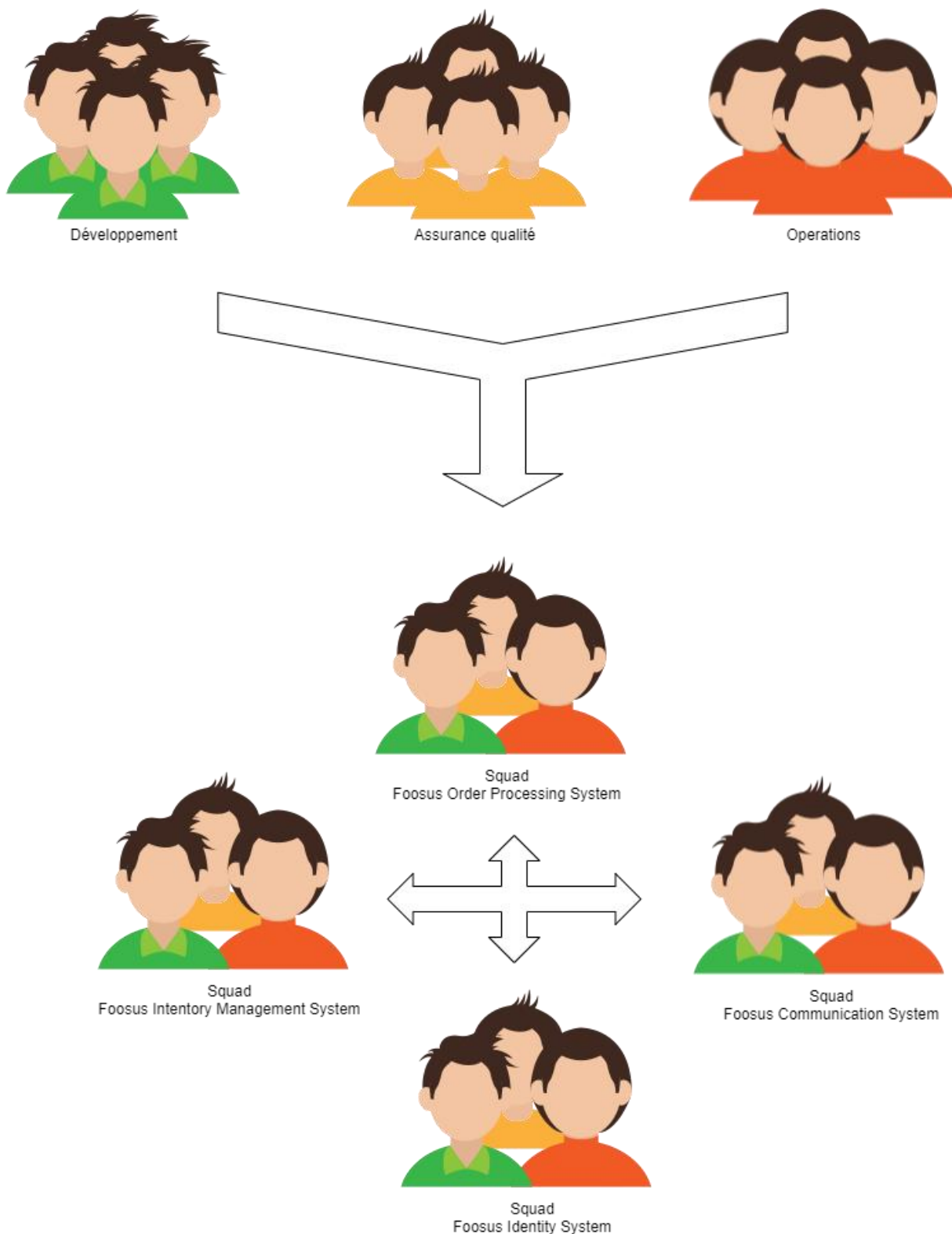
Apprendre, produire et mesurer sont au cœur de la méthodologie agile avec Lean. La communication, ainsi que la réalisation des activités s'articulent autour de ces trois principes fondamentaux. Kablan est utilisé afin de d'organiser et répartir les différentes tâches à effectuer.

Il apparaît pertinent de suivre la loi de Conway : « Les organisations qui conçoivent des systèmes [...] tendent inévitablement à produire des designs qui sont des copies de la structure de communication de leur organisation. »

Ainsi, l'équipe peut être divisé en différentes « squads » travaillant sur une fonctionnalité (correspondant à un micro-service), ou un ensemble de fonctionnalités liées (correspondant à un ensemble de micro-services).

Les différentes « squads » peuvent fonctionner de manière autonome et indépendante, tout en communiquant avec les autres équipes. Tout comme un micro-service peut nécessiter les données d'un autre micro-service en entrée, pour fournir une réponse adaptée en sortie, la communication entre les différentes « squads » est importante.

Il est recommandé l'utilisation de canaux propres à chaque « squad », travaillant sur un ou plusieurs micro-services, ainsi qu'au moins un canal commun à l'ensemble de l'équipe de développement. L'utilisation de Slack ou Discord est recommandé.



Cet exemple de division par « squads », ici incomplet, correspond à la vision de l'architecture évolutive.

De plus, des réunions régulières devront être mises en place. Celles-ci doivent être encadrées, afin de ne pas aller à l'encontre des démarches de standardisation des



processus de développement et de maintenance des différents éléments du système d'information voulues par Foosus.

Concernant les données, Foosus met en avant des principes de sécurité forts, considérant qu'il faut toujours protéger les données permettant l'identification personnelle. Cela dit, la start-up estime que les besoins métiers sont plus important qu'avoir une total cohérence au sein des données.

## 10- Conditions requises pour le management du service IT

Il convient de combler le fossé entre le moment où une ligne de code est écrite et celui où elle est validée dans un environnement intégré. Cela peut également aider à déterminer les réactions des clients vis-à-vis de nouvelles fonctionnalités à mesure que celles-ci sont développés.

Chaque nouvelle version doit être de taille réduite, présenter peu de risques, être transparente pour les utilisateurs et rester accessible en tout lieu et à tout moment.

Pour garantir un suivi des modifications de l'architecture, il est primordial qu'un suivi régulier soit effectué sur l'ensemble des activités. Pour une activité de développement, il est nécessaire de définir et documenter :

- Le champ d'application et la charge de travail associés à l'activité.
- Le découpage de l'activité en différents jalons et, si nécessaire, en différents micro-services.
- Les diagrammes d'architecture associés.
- Les logs d'activités et le suivi de la progression.
- Les résultats des tests.
- La conformité avec le cahier des charges.
- Les éventuels écarts avec les standards d'implémentations et les procédures standardisés mises en place par Foosus.
- Les informations relatives à l'activité telles que les dates clés et les ressources impliquées.



# 11- Contraintes

---

Ci-après figure une liste des contraintes relatives au projet, telles que présenté au sein du document d'autorisation du projet.

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

L'ensemble des spécifications et conditions requises, venant compléter cette liste de contraintes, ont été présentés au sein des sections précédentes, et se retrouvent aussi à travers les documents accompagnant l'activité d'analyse et conception d'une nouvelle architecture pour ce projet.

## 12- Hypothèses

---

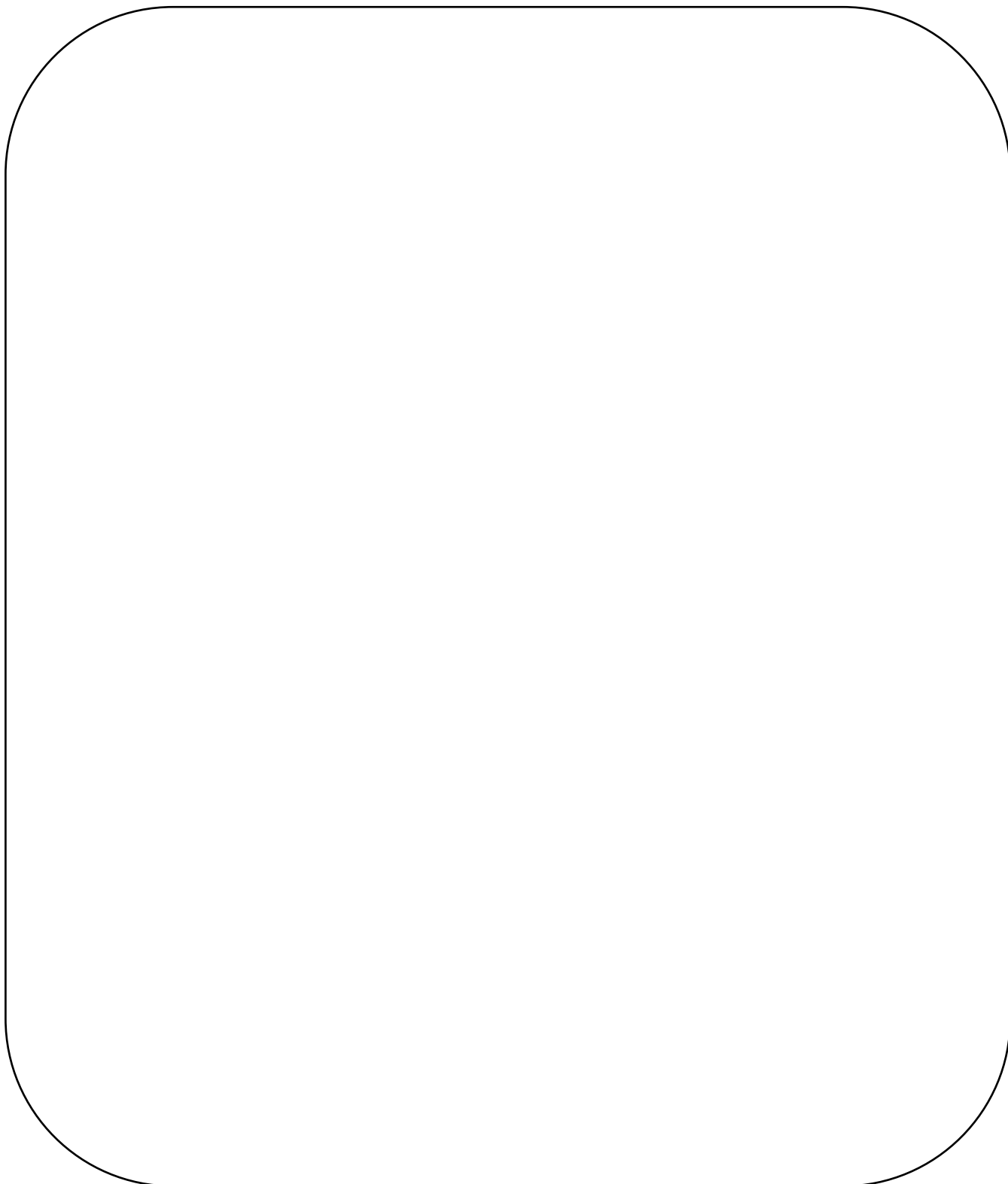
Ci-après figure une liste d'hypothèse présenté par Foosus :

- Plutôt que d'investir davantage dans la plateforme existante, celle-ci sera conservée en mode de maintenance. Aucune nouvelle fonctionnalité ne sera développée.
- La nouvelle architecture sera construite en fonction des technologies actuelles et avec la capacité de s'adapter à de nouvelles technologies lorsque celles-ci seront disponibles.
- Les équipes étant attachées à la plateforme existante, les dirigeants devront éviter de prendre de faux raccourcis en intégrant un nouveau comportement dans le système existant.
- L'offre initiale impliquera la coexistence de deux plateformes et la montée en puissance empirique du volume d'utilisateurs qui migreront vers la nouvelle plateforme à mesure que le produit évoluera. Cette augmentation sera proportionnelle à l'évolution des fonctionnalités.
- La géolocalisation, si elle est modélisée suffisamment tôt dans la nouvelle plateforme, permettra d'introduire d'autres innovations en fonction de l'emplacement de l'utilisateur ou du fournisseur alimentaire.
- L'élaboration sur mesure d'une approche architecturale de type Lean pourra contribuer à la réalisation de cette feuille de route, ce qui évitera de priver les équipes de leur autonomie et de compromettre la rapidité des cycles de versions.

L'ensemble de ces hypothèses a été pris en compte lors de la conception de la conception des documents accompagnant l'activité d'analyse et conception d'une nouvelle architecture pour ce projet.

## 13- Approbations signées

Commentaires, date de signatures :

A large, empty rounded rectangular box with a thin black border, intended for signatures and comments.