Specification
Des
Conditions Requises
Pour l'Architecture

Architecte Logiciel
Wiam el yadri

TABLE DES MATIERES

1 INFORMATION SUR LE DOCUMENT

Nom du Projet	Foosus- Conception d'une nouvelle architecture
Préparé par	El yadri Wiam
N° de version du document	1.0
Titre	Déclaration de travail d'architecture
Type d'action	Approbation, Révision, Information, Classement, Action requise, Participation à une réunion, Autre (à spécifier)

2 OBJECTIFS 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 SUCCES

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.

4 CONDITIONS REQUIESES POUR L'ARCHITECTURE

Tirer parti de la géolocalisation pour relier des fournisseurs et des consommateurs et pour proposer des produits disponibles près des lieux de résidence de ces derniers. Un calculateur de distance devra être inclus pour permettre aux consommateurs de trouver les fournisseurs les plus proches d'eux.

1. L'architecture devra être évolutive pour que nous puissions déployer nos services sur diverses régions, dans des villes et des pays donnés.
2. Les améliorations et autres modifications apportées aux systèmes de production devront limiter ou supprimer la nécessité d'interrompre le service pour procéder au déploiement.
3. Nos fournisseurs et nos consommateurs doivent pouvoir accéder à notre solution où qu'ils se trouvent. Cette solution doit être utilisable avec des appareils mobiles et fixes.
4. Elle doit tenir compte des contraintes de bande passante pour les réseaux cellulaires et les connexions Internet haut débit.
5. Elle doit pouvoir prendre en charge divers types d'utilisateurs (par exemple, fournisseurs, back-office, consommateurs), avec des fonctionnalités et des services spécifiques pour ces catégories.
6. Les livrables doivent pouvoir être fournis à intervalles réguliers pour que le nouveau système soit rapidement opérationnel et puisse être doté de nouvelles fonctionnalités au fil du temps.

5 CONTRATS DE SERVICE BUSINESS

5-1 ACCORDS DE NIVEAU DE SERVICE

Le service-level agreement (SLA) ou « entente de niveau de service » est un document qui définit la qualité de service, prestation prescrite entre un fournisseur de service et un client. Autrement dit, il s'agit de clauses basées sur un contrat définissant les objectifs précis attendus et le niveau de service que souhaite obtenir un client de la part du prestataire et fixe les responsabilités.

1. Tirer parti de la géolocalisation pour relier des fournisseurs et des consommateurs et pour proposer des produits disponibles près des lieux de résidence de ces derniers. Un calculateur de distance devra être inclus pour permettre aux consommateurs de trouver les fournisseurs les plus proches d'eux.
2. Les utilisateurs situés dans différentes régions doivent pouvoir espérer des performances similaires. Nous voulons cibler les consommateurs dans des zones géographiques spécifiques, sur des connexions lentes (par exemple, avec des téléphones portables) aussi bien que sur des réseaux haut débit. Toutes les solutions doivent pouvoir répondre à cette exigence.

6 CONTRATS DE SERVICE APPLICATION

6-1 OBJECTIFS DE NIVEAU DE SERVICE

Le but est de libérer la créativité et l'expérience de l'équipe technique.Il faut leur permettre de donner le meilleur d'elles-mêmes en créant une nouvelle plateforme qui pourra faire franchir le prochain million d'utilisateurs inscrits à la base de clientèle. On va impulser des campagnes de marketing Foosus dans plusieurs grandes villes en étant sûrs que la plateforme restera utilisable et réactive, tout en offrant une expérience utilisateur de premier plan.

On ne peut pas abandonner les outils actuels pendant qu'on en élabore de nouveaux car celaimpliqueraitlamisehorsservicedelaplateformeexistante. Pourpouvoircontinuerà accepter de nouvelles adhésions de fournisseurs et de consommateurs, il faut dissocier les nouvelles livraisons de l'architecture et de l'infrastructure existantes afin de limiter les interruptions de service.

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

6-1 INDICATEUR DE NIVEAU DE SERVICE

Le niveau de service requis, ou en anglais service-level requirement (SLR), est une définition, en fonction d'indicateurs prédéfinis, du niveau de service convenu entre un fournisseur de service et son client.

Lors de la création de back logs, les équipes produites peuvent développer à partir de comportements spécifiques, mais toute nouvelle conception doit cependant tenir compte des éléments suivants :

1- Emplacement des offres alimentaires proposées par les fournisseurs.
2- Proximité de l'utilisateur effectuant la recherche en cours.
3- Visualisation des informations statistiques secondaires et sectorielles relatives au produit

Alimentaire concerné. Par exemple, détails sur son indice glycémique.

7 LIGNES DIRECTRICES POUR L'IMPLEMENTATION

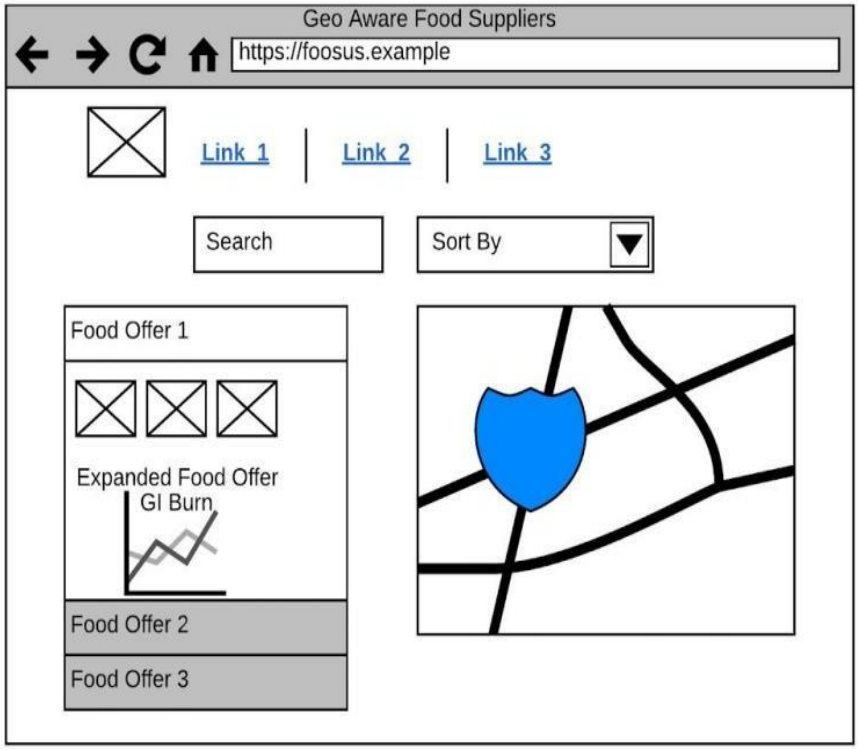
- Pour pouvoir continuer à accepter de nouvelles adhésions de fournisseurs et de consommateurs, nous devons en outre dissocier les nouvelles livraisons de l'architecture et de l'infrastructure existantes afin de limiter les interruptions de service.
- L'an dernier, 12 des pannes ont été provoquées par la publication par une ou plusieurs équipes de modifications lourdes qui n'ont pas eu les résultats escomptés. L'entreprise a également eu des difficultés à intégrer les travaux réalisés par différentes équipes sur des modifications de la plateforme qui n'avaient pas de lien entre elles. Il ne faut pas avoir ce genre de problème en tant que petite entreprise. La difficulté vient du temps nécessaire pour que chaque nouvelle version logicielle soit vue par les autres équipes ou testée dans nos environnements de production. Il faut combler le fossé entre le moment où une ligne de code est écrite et celui où elle est validée dans un environnement intégré. Cela permet également d'aider l'équipe à déterminer les réactions des clients vis-à-vis de nouvelles fonctionnalités au fur et à mesure qu'elle développe ces dernières.

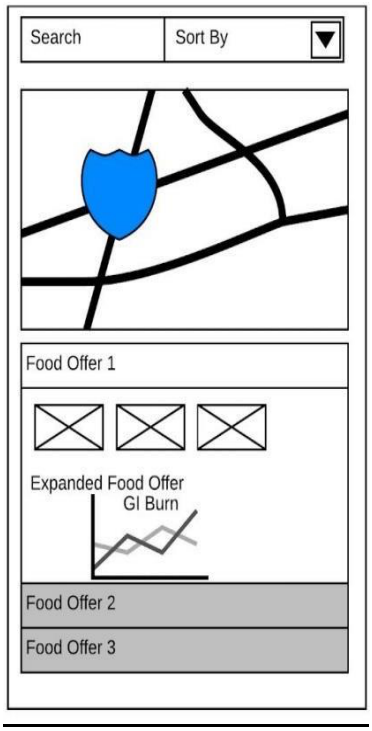
8 SPECIFICATIONS POUR L'IMPLEMENTATION

1. Déploiements indépendants. Il est tout à fait possible de mettre à jour un service sans redéployer toute l'application et annuler ou restaurer par progression une mise à jour en cas de problème. Les résolutions de bogues et les publications de fonctionnalités doivent être plus faciles à gérer et moins risquées.
2. Développement indépendant. Il est nécessaire de créer des équipes de développement uniquespourchaqueserviceafindefavoriserl'innovation continueetunrythmede publication plus élevé.

3. Résilience. Le système doit résister aux pannes de n'importe quel composant, telles que les pannes de serveur, les pannes de disque dur, les partitions réseau, etc. Tirer parti de la réplication empêche la perte de données et permet à un service de continuer à utiliser les instances restantes. Tirer parti de l'isolation empêche les échecs en cascade.
4. Élasticité. Vous pouvez vous attendre à ce que la charge varie considérable- ment au cours de la durée de vie du service. Il est essentiel de mettre en œuvre une scalabilité dynamique et automatique, à la fois vers le haut et vers le bas,en fonction de la charge.
5. Utilisation simple. La simplicité de l'architecture améliore son efficacité, sa gestion et sa maintenance.
6. Sécurité. Une architecture logicielle doit reposer sur un système qui dispose de contrôles de sécurité intégrés aux composants matériels et logiciels, ainsi que de protections supplémentaires contre les menaces telles que les attaques DDoS (distributed denial-of-service attack).
7. Le coût est l'un des aspects pertinents à prendre en compte lors du choix d'une architecture. Nous devons choisir une architecture dont le système dans lequel elle est intégrée nous offre des avantages pour une utilisation efficace et à moindre coût des ressources.

Les premiers tests de wireframes se sont traduits par les deux structures suivantes de point de rupture pour les appareils fixes et mobiles, tous deux privilégiant la proximité entre un fournisseur et l'utilisateur.





Lors de la création de backlogs, les équipes produites peuvent développer à partir de comportements spécifiques, mais toute nouvelle 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.

Nous prévoyons de faire reposer les nouvelles conceptions sur le processus tri des offres alimentaires existant, qui comporte les étapes suivantes :

- Recherche et identification des produits alimentaires requis.
- Ajout des offres alimentaires au panier.
- Recherche d'un accord pour payer à la livraison.
- Instructions de livraison et facture de la commission par e-mail au fournisseur alimentaire. La vision du produit à long terme consiste à modifier ces deux dernières étapes afinque nous puissions :

- L'intégrer à des prestataires de paiement tiers ;
- Gérer toutes les communications avec les fournisseurs alimentaires au sein d'une interface utilisateur personnalisée.

Les premières études sur les meilleures pratiques en matière d'architecture en font apparaître plusieurs qui présentent des risques techniques réduits. Il s'agit notamment de micro-services potentiels, de normes prenant en charge des solutions Web et mobiles, de bases de données standard et d'autres approches similaires.

9 STANDARDS POUR L'IMPLEMENTATION

Les premières études sur les meilleures pratiques en matière d'architecture en font apparaître plusieurs qui présentent des risques techniques réduits. Il s'agit notamment de micro-services potentiels, de normes prenant en charge des solutions Web et mobiles, de bases de données standard et d'autres approches similaires

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.

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.

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.

Frameworks

Malgré le fait qu'une architecture de micro-services offre une flexibilité concernant le choix de technologie, il apparait 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 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 relaxe 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 gravite 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.

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.

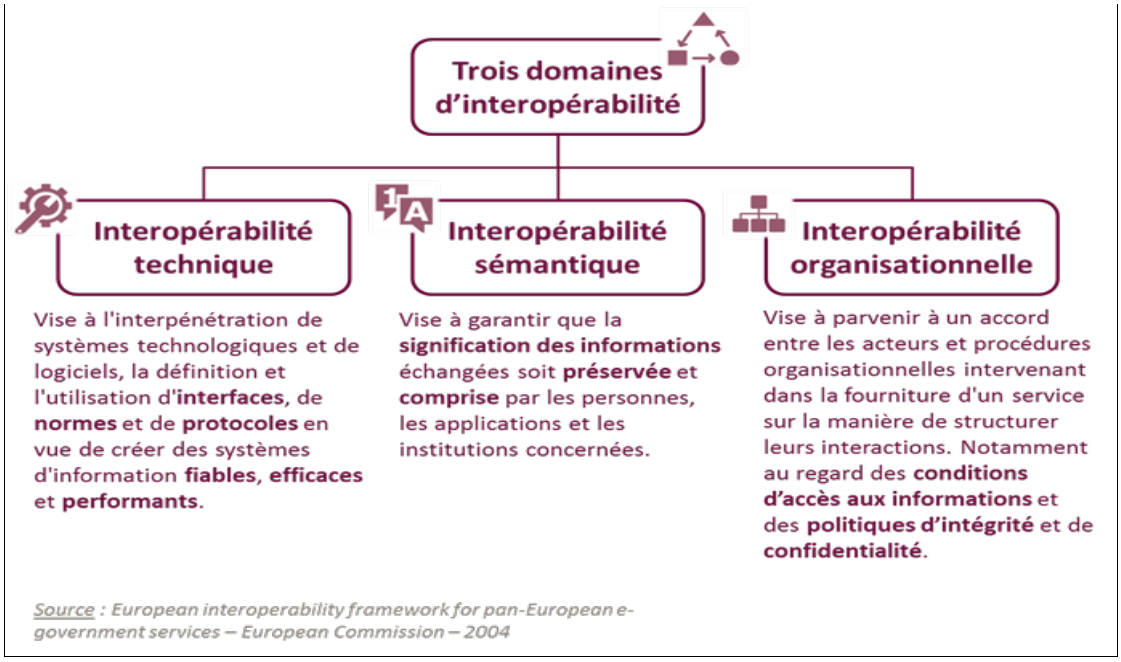
10 CONDITIONS REQUISES POUR L'INTEROPERABILITE

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.

L'interopérabilité sera garantie avec l'adoption de l'architecture des micro-services.

Avec cette architecture, la migration de l'application actuelle vers l'application cible peut être réalisée par étapes puisque chaque micro-service peut utiliser différentes technologies, langages de programmation et bases de données.

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 .



11 CONDITIONS REQUIES 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.

12 CONTRAINTES

Ci-après figure une liste des contraintes relatives au projet approuvé.

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.

L'objectif de cette phase du projet étant la définition de l'architecture, des projets de suivi seront créés pour compléter les détails avec les équipes internes.

13 HYPOTHESES

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.

14 ANNEXES

https://en.wikipedia.org/wiki/Denial-of-service_attack

https://en.wikipedia.org/wiki/Service-level_agreement