Spécification des Conditions requises pour l'Architecture

Projet *:* Concevez une Nouvelle Architecture afin de Soutenir le Développement de votre Entreprise

Client *:* FOOSUS

Préparé par *: COSTA, Elias*

N° de Version du Document *:* 0.1

Titre *:* Spécification des Conditions requises pour l'Architecture

Date de Version du Document *: 9/03/2021*

Revu par *: Jean Noel Gérard - Mentor*

Date de Révision *: 8/03/2021*

Table des Matières

1. 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.

2. Mesures du succès

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. L'architecture devra être évolutive pour que nous puissions déployer nos services sur diverses régions, dans des villes et des pays donnés.
3. Les améliorations et autres modifications apportées aux systèmes de production devront limiter ou supprimer la nécessité d'interrompre le service pour procéder au déploiement.
4. 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. 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.

3. Conditions requises pour l'architecture

- 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.

4. Contrats de service business

Gardant à l'esprit que l'objectif principal de Foosus est de soutenir la consommation de produits alimentaires locaux et de mettre en contact les clients avec des producteurs et artisans locaux pour satisfaire tous leurs besoins, le contrat de services business comprend trois aspects essentiels :

- La relation entre les producteurs et Fosus
    - Les producteurs locaux ont le devoir de produire des produits de qualité à des prix attractifs et de tenir à jour leurs informations sur les stocks.
- La relation entre Fosus et les consommateurs
    - Fournir un moteur de recherche de produits basé sur la géolocalisation des producteurs ;
    - Assurer la disponibilité des produits ;
    - Fournir des services de vente en ligne et de livraison de colis.
- La relation entre Fosus et ses employées
    - Tous les utilisateurs de Business se voient attribuer un profil reflétant leur rôle professionnel, le département dans lequel ils travaillent ou une autre catégorisation. Les profils permettent aux administrateurs de défi-

nir et de gérer de manière centralisée ce que différents types d'utilisateurs peuvent voir et faire dans l'interface utilisateur afin de pouvoir effectuer leurs tâches de manière efficace.

- Modéliser, planifier, développer, implémenter, tester, déployer, modifier, corriger, mettre à jour, maintenir, gérer, ..., tous les aspects et phases de la plateforme Foosus.

4.1. Accords de niveau de service

Pour que la vente de produits en ligne soit viable, Foosus doit avoir des accords pour la fourniture de services de transport de colis avec d'autres entreprises. L'externalisation de ce service est essentielle, car ce n'est pas un domaine de vocation pour Foosus.

5. Contrats de service application

5.1. Objectifs de niveau de service

Notre but est de libérer la créativité et l'expérience de nos équipes techniques. Nous voulons 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 notre base de clientèle. Nous voulons impulser des campagnes de marketing Foosus dans plusieurs grandes villes en étant sûrs que notre plateforme restera utilisable et réactive, tout en offrant une expérience utilisateur de premier plan.

Nous ne pouvons pas abandonner les outils actuels pendant que nous en élaborons de nouveaux car cela impliquerait la mise hors service de la plateforme existante. 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.

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

5.2. Indicateurs de niveau de service

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.
Amélioration de la capacité de recherche	Premier trimestre

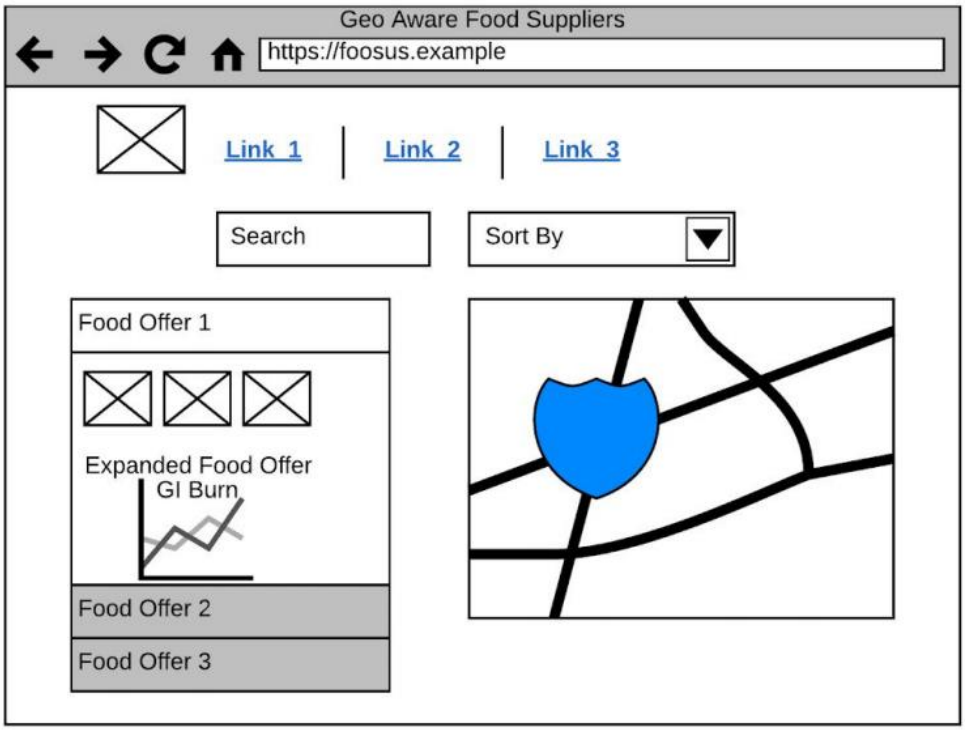
6. Lignes directrices pour l'implémentation

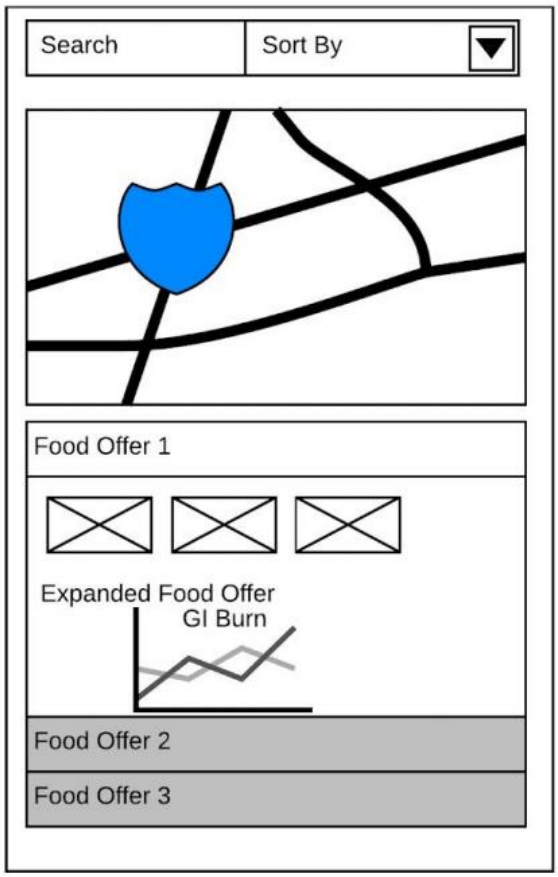
1. La pile technologique doit être conçue de façon à évoluer naturellement au même rythme que notre base de clientèle.
2. La plateforme doit être capable d'évoluer pour gérer les augmentations de charges.
3. Nous souhaitons en outre qu'elle soit facile à adapter aux particularités locales et qu'elle réponde aux exigences d'utilisation de nos clients.
4. La plateforme doit être disponible 24 h/24.
5. Chaque nouvelle version doit être de taille réduite, présenter peu de risques, être transparente pour nos utilisateurs et rester accessible en tout lieu et à tout moment.
6. Les utilisateurs situés dans différentes régions doivent pouvoir espérer des performances similaires.
7. Les différentes équipes doivent être capables de surmonter leurs difficultés d'intégration des travaux de modifications de la plateforme.
8. Mettre en place un mécanisme isolé pour la réalisation des tests des différentes solutions réalisées par l'équipe de développement.

7. Spécifications pour l’implémentation

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 uniques pour chaque service afin de favoriser l'innovation continue et un rythme de 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 considérablement 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.
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 produits 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 afin que 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.

8. Standards pour l'implémentation

1. La plateforme Foosus doit être hébergée dans le cloud. Le fournisseur de cloud que je propose d'utiliser est Microsoft Azure.
2. L'architecture des microservices est celle qui répond le mieux aux exigences de Foosus.
3. Nous utiliserons la plateforme Azure Service Fabric car elle nous aidera à empaqueter, déployer et gérer facilement des microservices évolutifs et fiables (le conteneur est également comme Docker). Avec l'aide d'Azure Service Fabric, les développeurs n'ont plus à se soucier des problèmes d'infrastructure. Azure Service Fabric fournit diverses technologies et se présente sous la forme d'un bundle qui a la puissance d'Azure SQL Database, Cosmos DB, Microsoft Power BI, Azure Event Hubs, Azure IoT Hub et de nombreux autres services de base.

4. Les méthodes de livraison continue (CD) et d'intégration continue (CI) seront largement utilisées dans la mise en œuvre de la nouvelle architecture. Les conteneurs peuvent répondre aux exigences d'isolation beaucoup plus efficacement que leurs concurrentes les plus proches, les machines virtuelles.

5. Nous utiliserons l'approche mock-and-stub pour rendre les tests indépendants des autres microservices. Cela facilite également les tests avec les bases de données, car nous pouvons également simuler les interactions de bases de données.

6. Quant aux aspects de sécurité, nous utiliserons Azure Active Directory (Azure AD) pour l'authentification des utilisateurs et des applications. Azure AD est l'un des fournisseurs de spécifications OAuth 2.0 et OpenID Connect. Il est entendu ici qu'Azure AD évolue très bien avec les applications et s'intègre bien à tout Windows Server Active Directory organisationnel.

7. En ce qui concerne la surveillance et la journalisation, Microsoft Azure nous donne une image complète des performances de notre application Web en fournissant une analyse des journaux, une surveillance des applications et des alertes de sécurité. Nous n'avons donc pas à installer de nouveau logiciel. Microsoft Cloud Monitoring est parfait pour les besoins de la plateforme Foosus.

9. Conditions requises pour l'interopérabilité

L'interopérabilité sera garantie avec l'adoption de l'architecture des microservices. Avec cette architecture, la migration de l'application actuelle vers l'application cible peut être réalisée par étapes puisque chaque microservice peut utiliser différentes technologies, langages de programmation et bases de données.

Il est très important d'examiner attentivement le choix du mécanisme de messagerie lorsqu'il s'agit d'architecture de microservice. Si cet aspect est ignoré, cela peut compromettre l'ensemble de l'objectif de la conception d'une architecture de microservice.

Pour implémenter ce projet, mon choix se porte sur la Messagerie Asynchrone.

Si le couplage faible est important, en particulier dans un système qui nécessite une résilience élevée et dont l'échelle est imprévisible, la meilleure option est la messagerie asynchrone.

La messagerie asynchrone est une approche fondamentale pour intégrer des systèmes indépendants ou pour construire un ensemble de systèmes faiblement couplés

qui peuvent fonctionner, évoluer et évoluer de manière indépendante et flexible. La messagerie asynchrone et la communication pilotée par les événements sont des fonctionnalités essentielles lors de la propagation de changements sur plusieurs microservices et leurs modèles de domaine connexes.

10. Recommandations pour le management du service IT

- Modélisez les services autour du domaine de l'entreprise.
- Décentralisez tout. Des équipes individuelles sont chargées de concevoir et de créer les services. Évitez le partage de code ou de schémas de données.
- Le stockage de données doit être accessible uniquement au service auquel les données appartiennent. Utilisez le meilleur stockage pour chaque type de service et de données.
- Gestion de données. Étant donné que chaque microservice a sa propre base de données indépendante, la prise de décision liée aux modifications requises dans la base de données peut être facilement déléguée à l'équipe respective. Nous n'avons pas à nous soucier de l'impact sur le reste du système, car il n'y en aura pas. Au même temps, cette séparation de la base de données ouvre la possibilité à l'équipe de s'auto-organiser.
- Les services communiquent via des API bien conçues. Évitez les fuites des détails de la mise en œuvre. Les API doivent modéliser le domaine et non la mise en œuvre interne du service.
- Évitez le couplage entre les services. Le couplage peut notamment être dû à des schémas de base de données partagés et à des protocoles de communication rigides.
- Confiez les responsabilités transversales, telles que l'authentification et la terminaison SSL, à la passerelle.
- Gardez les connaissances du domaine hors de la passerelle. La passerelle doit gérer et acheminer les requêtes de clients sans aucune connaissance des règles métier ou de la logique de domaine. Dans le cas contraire, la passerelle devient une dépendance et peut causer un couplage entre les services.
- Les services doivent présenter un couplage faible et une forte cohésion fonctionnelle. Les fonctions qui sont susceptibles de changer en même temps doi-

vent être empaquetées et déployées ensemble. S'ils résident dans des services distincts, ces services se retrouvent fortement couplés, car un changement de l'un d'eux exigera la mise à jour de l'autre. Une communication trop importante entre deux services peut être un signe de couplage fort et de faible cohésion.

- Isolez les défaillances. Utilisez des stratégies de résilience afin d'empêcher que les défaillances au sein d'un service n'entraînent une réaction en chaîne. Pour plus d'informations, consultez Conception d'applications résilientes pour Azure.

11. Contraintes

- 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.
- 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'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.

12. Hypothèses

- Plutôt que d'investir davantage dans la plateforme existante, nous la conserverons 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.
- Par exemple, les utilisateurs précoces pourront choisir d'utiliser les nouvelles fonctionnalités de recherche intégrées au processus de paiement existant.
- 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.