SPÉCIFICATION DES CONDITIONS REQUISES POUR L'ARCHITECTURE

PROJET DE CONSTRUCTION D'UNE PLATEFORME D'E-COMMERCE GÉO-CONSCIENTE, RÉSILIENTE ET BASÉE SUR UNE ARCHITECTURE RESPONSABLE

TABLE DES MATIÈRES

OBJET DE CE DOCUMENT **3**

CONDITIONS REQUISES POUR L'ARCHITECTURE **4**

Conditions requises pour l'architecture 4
Conditions requises pour l'interopérabilité 4
Contraintes 5
Hypothèses 5
Mesures du succès de l'architecture 6

CONDITIONS REQUISES POUR LE MANAGEMENT DU SERVICE IT **8**

CONTRATS DE SERVICE **9**

Contrats de service business 9
Contrats de service application 10

IMPLÉMENTATION **11**

Lignes directrices pour l'implémentation 11
Spécifications pour l'implémentation 11
Architecture microservices 11
Conteneurisation 11
Passerelle d'API 12
Bases de données 12
Standards pour l'implémentation 13
Architecture REST 13
Formats d'échanges de données 13
ACID 14
SQL 14

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.

CONDITIONS REQUISES POUR L'ARCHITECTURE

Conditions requises pour l'architecture

L'approche architecturale doit privilégier les bonnes pratiques en s'organisant autour de microservices.

La nouvelle architecture peut faire appel à des solutions open source.

L'architecture doit être évolutive et permettre l'ajout, la modification ou la suppression d'un service sans impacts sur la disponibilité de la plateforme. Elle doit rester disponible 24/24h et ne doit pas nécessiter d'interruption de service.

La nouvelle architecture doit pouvoir être scalable et absorber les montées de charge.

La nouvelle architecture doit pouvoir coexister avec l'ancienne pendant la période de son déploiement progressif.

La nouvelle architecture doit permettre d'expérimenter auprès de nouvelles solutions sans impacter la plateforme de production.

La nouvelle architecture doit faciliter l'interopérabilité de ses composants.

L'architecture doit assurer la pérennité des données en intégrant un outil de sauvegarde parmi ses composants.

La nouvelle architecture et ses composants doivent être adaptés à l'internationalisation (i18n) et la régionalisation.

La nouvelle architecture doit être adaptée à tous types de terminaux et de connexions.

Conditions requises pour l'interopérabilité

Les solutions développées ou sélectionnées devront exposer des API REST.

Les solutions développées ou sélectionnées devront faire partie d'une pile technologique préconisée en début de projet.

Les solutions développées ou sélectionnées ne devront pas exploiter des langages, normes ou concepts propriétaires.

Les solutions développées ou sélectionnées devront respecter les standards de l'industrie.

Contraintes

Le budget alloué pour concevoir la nouvelle architecture, préparer un projet de suivi et développer un prototype de la nouvelle plateforme est de 50000 $. La durée allouée est de 6 mois.

Une phase d'expérimentation de 3 mois est prévue.

Les solutions open source sont préconisées.

Le support continu des composants sélectionnés doit être pris en charge.

Toutes les solutions sélectionnées doivent, dans la mesure du possible, faire partie d'une même pile technologique.

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.

Mesures du succès de l'architecture

La mesure du succès de l'architecture se fera selon les critères suivants (ces mesures incluent les métriques définies dans la Déclaration de travail d'architecture) :

Métrique	Technique de mesure	Valeur cible	Justification	Notes supplémentaires
Implémentation de l'industrialisation	Idem au KPI 'Délai moyen de parution"	Idem au KPI "Délai moyen de parution"		
Implémentation de l'environnement cloud	Vérification des conditions requises	Diagramme de base de l'architecture		
Développement des nouveaux fronts et implémentation du microservice d'identification	Vérification des conditions requises	Diagramme Architecture cible - Étape 1		
Implémentation des microservices de géolocalisation, de recherche et d'inventaire	Vérification des conditions requises	Diagramme Architecture cible - Étape 2		
Implémentation des microservices de commande et de finance	Vérification des conditions requises	Diagramme Architecture cible - Étape 3		
Tests de la nouvelle plateforme en interne	Vérification des conditions requises	Diagramme Architecture cible - Étape 4		
Expérimentation et désactivation de l'ancienne plateforme	Vérification des conditions requises	Diagramme Architecture cible		
Nombre d'adhésions d'utilisateurs par	Requêtes sur la BDD de l'application	Augmentation de 10 %	Nous n'attirons plus de nouveaux	La nouvelle architecture devrait permettre d'atteindre

jour	(avant et après la MEP de l'architecture)		utilisateurs	cet objectif grâce à la géolocalisation
Adhésion de producteurs alimentaires	Requêtes sur la BDD de l'application (avant et après la MEP de l'architecture)	4 / mois	Nous n'attirons plus de nouveaux producteurs (1,4 / mois)	La nouvelle architecture devrait permettre d'atteindre cet objectif grâce à la géolocalisation
Délai moyen de parution	Requête sur notre outil de gestion de projet pour contrôler le délai de traitement des US	Réduction à moins d'une semaine	Délai de parution trop long (3,5 semaines) pour être compétitif face aux grands acteurs de notre marché	La réduction du temps de sprint à une semaine devrait permettre d'atteindre cet objectif (en plus d'une hausse du niveau de qualité)
Taux d'incidents de production P1	Requête sur notre outil de gestion de projet pour contrôler le nombre de tickets créés sur 1 mois	Réduction à moins de 1 / mois	Trop d'incidents rencontrés en production (> 25 / mois)	Les pratiques agiles et lean devraient permettre d'atteindre cet objectif

CONDITIONS REQUISES POUR LE MANAGEMENT DU SERVICE IT

Les collaborateurs du service IT doivent adopter la culture Lean de l'entreprise en orientant les résultats de leur travail vers la satisfaction des besoins de nos utilisateurs.

Les pratiques agiles doivent être mises en place afin d'atteindre les critères de mesure de succès concernant la rapidité de MEP et la qualité des développements livrés.

Les équipes doivent donc se structurer selon les pratiques agiles en nommant un scrum master et un product owner dans chaque équipe et en organisant les cérémonies agiles permettant ainsi d'améliorer la communication au sein des équipes.

L'approche Scrum doit être privilégiée à Kanban (malgré ses avantages en termes de rapidité de MEP) car cela permet aux équipes de s'engager à réaliser un incrément de travail, qui est potentiellement livrable, selon des intervalles définis (sprints). L'objectif de cette approche est de créer des boucles d'apprentissage afin de rassembler et d'intégrer rapidement le feedback des clients. La durée de sprint préconisée est d'une semaine.

Afin d'augmenter la qualité des développements, les phases de code review devront être systématisées avant toute MEP.

L'approche CI/CD doit être implémentés pour accélérer au maximum le délai entre la prise en charge d'une demande d'amélioration/modification/correction et sa MEP.

Des outils de gestion de projet et collaboratifs devront être mis en place au sein des équipes de développement afin de faciliter la communication et le partage d'informations.

Il est fortement recommandé de mettre en place des environnements de pré-production ainsi que des tests automatisés.

Les équipes de développements pourront continuer à laisser libre court à leur créativité afin d'être force de proposition en livrant des nouvelles fonctionnalités (ou de nouvelles versions de fonctionnalités existantes) s'appuyant sur des technologies innovantes.

CONTRATS DE SERVICE

Contrats de service business

L'accord de niveau de service (ou Service Level Agreement - SLA - en anglais) du contrat de service business stipule les éléments suivants :

- L'ancienne plateforme doit pouvoir continuer à être exploitée (nouvelles livraisons sur l'ancienne architecture, nouvelles adhésions d'utilisateurs, ...) pendant la période d'implémentation de la nouvelle architecture sans interruption de service.
- La nouvelle architecture doit être évolutive et permettre l'ajout, la modification ou la suppression d'un service sans impacts sur la disponibilité de la plateforme. Le déploiement d'une nouvelle version d'un service ne pas avoir d'impact visible et/ou de disponibilité de service pour les utilisateurs.
- La nouvelle architecture doit pouvoir être déployable dans plusieurs nouvelles régions, en restant disponible 24/24h quel que soit le fuseau horaire sur lequel la plateforme est implantée. Elle doit pouvoir s'adapter facilement aux particularités locales.
- La nouvelle architecture doit tenir compte des contraintes de connexion de nos utilisateurs en s'adaptant à leur bande passante.
- La nouvelle architecture doit être scalable. et suffisamment performante afin de supporter le niveau d'engagement et de croissance attendus de nos futurs programmes marketing. En cas de surcharge, les services doivent continuer à être accessibles, à minima de façon dégradée.
- La nouvelle architecture doit être sécurisée et être capable de maintenir un niveau de sécurité élevé lors des phases d'élargissement de la plateforme.
- La nouvelle architecture doit pouvoir faciliter l'intégration de nouvelles solutions fournies par des partenaires ou développées en interne. Celles-ci doivent pouvoir être facilement réversibles.

La mesure du succès du contrat de service business se fera selon les critères de mesure du succès de l'architecture définis dans les conditions requises pour l'architecture.

Contrats de service application

L'accord de niveau de service (ou Service Level Agreement - SLA - en anglais) du contrat de service application stipule les éléments suivants :

- La plateforme doit être accessible 24/24h pour tous les fuseaux horaires où elle est et sera disponible
- Les utilisateurs de type consommateurs, producteurs ou encore back-office doivent pouvoir se connecter à la plateforme avec un accès spécifique correspondant à leur profil.
- Les utilisateurs de type consommateurs doivent pouvoir rechercher les producteurs proches d'eux (en fonction de leur adresse ou de leur position géographique au moment de la recherche).
- Les utilisateurs doivent pouvoir se connecter à la plateforme avec des appareils mobiles et fixes et ce quelle que soit la qualité de la connexion qu'ils utilisent.

La mesure du succès du contrat de service application se fera selon les critères de mesure du succès de l'architecture définis dans les conditions requises pour l'architecture.

IMPLÉMENTATION

Lignes directrices pour l'implémentation

L'implémentation doit se faire par phases successives lors desquelles chaque ancien service monolithique ("Foosus … System') sera remplacé par un ou plusieurs nouveaux microservices fournissant une couverture fonctionnelle identique.

Les 2 architectures doivent pouvoir cohabiter et les services de l'une ou de l'autre doivent pouvoir être exploités par tous les utilisateurs.

Chaque étape de l'implémentation doit être préalablement validée sur un environnement de pré-production puis être soumise à l'approbation par un panel d'utilisateurs avant sa mise en production.

Spécifications pour l'implémentation

Architecture microservices

Le but d'une architecture microservices est de concevoir une application en une suite de petits services, appelés microservices, découplés le plus possible les uns des autres,

Chaque microservice est aussi simple que possible, autonome et répond à une fonctionnalité métier unique, Il possède sa propre base de données. Il peut être développé de manière indépendante des autres microservices en utilisant son propre langage de programmation et est déployable indépendamment des autres, de façon continue et automatisée. Il s'exécute dans son propre processus, communique avec un protocole léger et est automatiquement scalable.

Conteneurisation

Le principe de conteneurisation est préconisé pour l'hébergement des microservices car il permet de développer des applications de façon plus efficiente, en utilisant moins de ressources, et de déployer ces applications plus rapidement.

Un conteneur est une enveloppe virtuelle qui permet de distribuer une application avec tous les éléments dont elle a besoin pour fonctionner : fichiers source, environnement d'exécution, librairies, outils et fichiers.

Ils sont assemblés en un ensemble cohérent et prêt à être déployé sur un serveur et son système d'exploitation (OS). Contrairement à la virtualisation de serveurs et à une machine virtuelle, le conteneur n'intègre pas de noyau, il s'appuie directement sur le noyau de l'ordinateur sur lequel il est déployé.

Passerelle d’API

Une passerelle d'API (ou API Gateway) est un outil de gestion des interfaces de programmation d'application (API) qui se positionne entre un client et une collection de services back-end. Elle agit comme un proxy inversé qui accepte tous les appels des API, rassemble les différents services requis pour y répondre et renvoie le résultat souhaité.

L'objectif principal d'une passerelle API est de simplifier et de stabiliser les interfaces exposées aux clients (mobiles, navigateurs...). De plus, en raison de la position unique d'une passerelle API dans l'architecture, divers avantages complémentaires sont activés, tels que la surveillance, la journalisation, la sécurité, l'équilibrage de charge et la manipulation du trafic.

Bases de données

Au vu des objectifs de déploiement international de Foosus, nous préconisons l’utilisation d’une base de données hébergée en cloud auprès d’une prestataire qui aura la capacité à distribuer rapidement la donnée là où elle est nécessaire.

Il est également conseillé d’utiliser une base de données relationnelle respectant les propriétés ACID garantissant ainsi que les transactions seront correctement exécutées.

De plus l’hébergeur devra proposer un outil de réplication afin d’améliorer la fiabilité, la tolérance aux pannes, ou la disponibilité des données.

Standards pour l'implémentation

Architecture REST

L'architecture REST (Representational State Transfer ou transfert d'état de représentation, en français) constitue un ensemble de normes, ou de lignes directrices architecturales qui structurent la façon de communiquer les données entre une application et le reste du monde, ou entre différents composants d'une application. Les API REST (ou RESTful) se basent sur le protocole HTTP pour transférer les informations.

Les avantages clés des API REST sont :

- la séparation du client et du serveur, qui aide à scaler plus facilement les applications ;
- le fait d'être stateless, ce qui rend les requêtes API très spécifiques et orientées vers le détail ;
- la possibilité de mise en cache, qui permet aux clients de sauvegarder les données, et donc de ne pas devoir constamment faire des requêtes aux serveurs.

Formats d'échanges de données

JSON est préconisé comme format d'échanges de données. JavaScript Object Notation (JSON) est un format de données textuelles dérivé de la notation des objets du langage JavaScript.

JSON a pour avantage d'être un format léger grâce à sa structure en arborescence et sa syntaxe simple et est donc compréhensible par tous (humain et machine).

Il ne dépend d'aucun langage (format d'échange de données ouvert). Comme ce format est très ouvert, il est pris en charge par de nombreux langages : JavaScript, PHP, Perl, Python, Ruby, Java,...

Il permet également de stocker des données de différents types : chaînes de caractères (y compris des images en base64), nombres, tableaux (array), objets, booléens (true, false), la valeur null.

Son principal inconvénient est qu'il nécessite, comme tout format d'échange de données, de sécuriser ces échanges en implémentant un algorithme de cryptage de bout en bout de la chaîne d'échange.

ACID

En informatique, les propriétés ACID (atomicité, cohérence, isolation et durabilité) sont un ensemble de propriétés qui garantissent qu'une transaction informatique est exécutée de façon fiable.

Les 4 propriétés ACID se définissent ainsi :

- Atomicité : tout ou rien, une modification des données doit être réalisée dans son intégralité ou pas du tout.
- Cohérence : les données doivent toujours être cohérentes entre elles, même en cas d'erreur. Dans ce cas là, on effectuera un RollBack.
- Isolation : Pas d'interférences entre les transactions. Utilisation des verrous et des points de synchronisation.
- Durabilité: Lorsqu'une transaction s'est achevée, avec succès (Commit) ou en erreur (Rollback), les données doivent être dans un état stable et cohérent.

SQL

SQL (pour Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles.

C'est un langage déclaratif c'est-à-dire qu'il permet de décrire le résultat escompté, sans décrire la manière de l'obtenir à l'aide d'instructions. Les instructions SQL s'écrivent d'une manière qui ressemble à celle de phrases ordinaires en anglais. Cette ressemblance voulue vise à faciliter l'apprentissage et la lecture.

SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.