SPECIFICATION DES CONDITIONS REQUISES POUR L'ARCHITECTURE

Projet : Foosus géoconscient

Client : Foosus

Information sur le document

Nom du projet	Foosus géoconscient (titre provisoire)
Préparé par	Noémie BARRAL
N° de version	1.0
Titre	Spécification des Conditions requises pour l'Architecture
Date de version	27 / 10 / 2022
Revu par	N/A
Date de révision	N/A
Historique de version	v. 1.0 (27 / 10 / 2022)

Table des matières

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.

Mesures du succès

Les indicateurs de réussite ont été défini précédemment lors de la rédaction de la *Requête de travail*.

Indicateur	Changement attendu
Nombre d'adhésions d'utilisateurs par jour	Augmenter de 10 %
Adhésion de producteurs alimentaires	Passer de 1,4 / mois à 4 / mois
Délai moyen de parution	Passer de 3,5 semaines à moins d'une semaine
Taux d'incident de production P1	Passer de > 25 / mois à < 1 / mois

En plus de ces indicateurs, nous pourrons aussi ternir compte des indicateurs de niveau de service défini plus bas afin d'évaluer la réponse aux accords de niveau de service.

Conditions requises pour l’architecture

Plusieurs conditions ont été retenus comme bonnes pratiques afin d’avancer sur le travail d’architecture.

Choix technologiques

- 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.

Principe data

- Toujours modéliser comme si vous n’aviez pas encore la vision d’ensemble.
- Toujours protéger les données permettant l’identification personnelle.
- Concevoir pour l’accès aux données ou la mutabilité en fonction du problème.
- Appliquer la cohérence en fonction du scénario pour satisfaire au mieux le besoin business. (Ne partez pas du principe que toutes les données doivent être cohérentes immédiatement ou même à terme.)
- Refléter le modèle de domaine au sein d’un contexte délimité de façon appropriée.

Principe d’application

- Responsabilité unique et couplage faible des applications.
- Concevoir des interfaces ouvertes et extensibles en systèmes, sur lesquelles il est facile d’itérer.
- Appliquer une approche pilotée par le contrat client, où les interfaces entre les systèmes reflètent uniquement les données et opérations nécessaires à leur intégration.
- Éviter les dépendances cycliques entre les systèmes.

Contrats de service business

Accords de niveau de service

Les échanges avec les utilisateurs ont mis en avant un ensemble d'éléments indispensables que la nouvelle architecture doit permettre d'atteindre.

Il est d'abord attendu que l'application puisse soutenir la croissance de Foosus, notamment l'évolution de son nombre d'utilisateur.

A terme, il est attendu que le nombre d'utilisateurs global dépasse le million. L'application doit être en mesure de gérer la charge de données associée.

Par ailleurs, le nombre d'utilisateur simultané augmente logiquement, et l'application doit être en mesure de soutenir les pics utilisateurs à tout moment. Le service ne doit pas tomber en panne et créer une indisponibilité, quitte à proposer un service dégradé en cas de trop forte utilisation.

Ensuite, il est attendu que l'application puisse être disponible de manière quasiment permanente.

Cette disponibilité est attendue techniquement par la suppression du besoin d'interrompre le service en cas de déploiement et par la limitation, voire la suppression, des pannes dû à des déploiement à risque.

Afin de parvenir à ce dernier objectif, il a été demandé que le temps entre chaque version soit moins espacé afin de limiter leur impact et de pouvoir réagir plus facilement pour apporter correctif ou rollback.

Cette disponibilité est aussi attendue d'un point de vue géographique, l'application devant être disponible en tout lieu, et ce quelque soit la qualité du débit.

L'application devra être disponible depuis un appareil fixe ou mobile.

Enfin, l'application doit garantir la sécurité des données qu'elles conservent et permettre la mise en place à terme d'une solution de transaction bancaire par le biais d'un service tiers.

Objectif business	Accords de niveau de service
La solution doit pouvoir soutenir la croissance utilisateur de Foosus	• Gérer la charge des données associées • Rester accessible même en cas de pic d'utilisateurs (mode dégradé)
La solution doit être disponible à tout moment	• Limiter l'interruption de service en cas de déploiement • Limiter les pannes dues à des déploiements à risques • *Rester accessible même en cas de pic d'utilisateurs (mode dégradé)*
La solution doit être disponible en tout lieu	• Disponibilité géographique étendue • Disponibilité quelque soit la qualité du débit internet
La solution doit être accessible sur toutes les plateformes	• Interopérabilité mobile et fixe

La solution doit être fonctionnellement évolutive	• Mise en place de cycle court (AGILE)
La solution doit être sécurisée	• Garantir la sécurité des données conservées • Permettre la mise en place d'une solution de transaction bancaire

Contrats de service application

Objectifs de niveau de service

Les accords précédents nous permettent de définir des objectifs précis à atteindre pour chaque engagement que nous prenons.

Les incidences en production, qu'elles soient dû au nombre d'utilisateurs simultanés ou au déploiement d'une nouvelle version devront passer de plus de 25 par mois à moins de 1 par mois.

L'application devra être disponible en moyenne au moins 99% du temps. Cela implique que tout incident de production puisse rapidement être résolu, qu'une solution de rollback soit facilement disponible et que le déploiement d'une nouvelle version ne crée pas d'indisponibilité importante.

Le délai entre chaque nouvelle version doit être inférieur à une semaine.

L'application devant être disponible en tout lieu, nous avons pour objectif de couvrir au moins 90% des pays, certains pays exerçant des restrictions trop importantes pour que l'application puisse être rendu disponible. Cette disponibilité ne tient pas compte de l'existence d'infrastructure sur place permettant son accès, cela n'étant pas de notre ressort.

L'application devant être disponible quelque soit la qualité du débit à compter du moment où des infrastructures sont présentes, il est attendu un temps de réponse moyen acceptable selon la qualité du débit :

- Moins d'1 seconde pour un débit de bonne qualité
- 1 à 2 secondes pour un débit de qualité moyenne
- Moins de 10 secondes pour un débit de mauvaise qualité

Enfin, l'application devant être disponible sur des supports fixe et mobile, l'accès sera garanti sur Web, Android et IOS pour couvrir toutes les plateformes. Pour le Web, l'accès sera garanti sur les navigateurs suivant : Edge, Chrome, Firefox et Safari afin de couvrir un maximum d'utilisateurs. Ce choix pourra être revu s'il apparait qu'un nombre important d'utilisateurs fait usage d'un explorateur différent.

Accords de niveau de service	Objectifs de niveau de service
• Limiter l'interruption de service en cas de déploiement	Application disponible 99% du temps
• Limiter les pannes dues à des déploiements à risques • Rester accessible même en cas de pic d'utilisateurs (mode dégradé)	Incidence en production : de 25 / mois, passer à < 1 / mois
• Disponibilité géographique étendue	Disponibilité dans 90% des pays
• Disponibilité quelque soit la qualité du débit internet	Temps de réponse selon débit : • Bon débit : < 1 seconde • Débit moyen : 1 à 2 secondes • Mauvais débit : < 10 secondes
• Interopérabilité mobile et fixe	Support sur les plateformes IOS, Android et Web : Edge, Chrome, Firefox et Safari

• Mise en place de cycle court (AGILE)	Délai entre chaque version : < 1 semaine

Indicateurs de niveau de service

Indicateur	Mode de calcul	Cible	Réel
Taux d'incidence en production	Bug rapportés et interruptions involontaires	< 1 / mois	
Taux de disponibilité	% du temps moyen pendant lequel les serveurs sont disponibles	> 99%	
Délai entre chaque version	Temps depuis la version précédente	< 1 semaine	
Disponibilité géographique	% de pays dans lequel la plateforme est accessible	> 90%	

Interopérabilité	Plateformes supportées :
	- Android
	- IOS
	- Web :
	  o Edge
	  o Chrome
	  o Firefox
	  o Safari
		Support OK	

Lignes directrices pour l’implémentation

Les lignes directrices identifiées jusqu’à présent sont les suivantes :

- La solution doit permettre une standardisation des technologies au regard d'une réflexion stratégique.
- La solution doit être fonctionnellement évolutive en fonction du rythme de l'entreprise, en facilitant l'ajout ou la modification de services.
- La solution doit permettre la création et le déploiement de services pour des régions ou des utilisateurs spécifiques.
- La solution doit être sécurisée en tout lieu et toute circonstance.
- La solution doit pouvoir évoluer avec le nombre d'utilisateurs en absorbant les pics d'utilisateurs et permettant la croissance du nombre d'utilisateur grâce à un mécanisme de mise à l'échelle.
- La solution doit être disponible et performante quelque soit la zone géographique.
- La solution doit permettre de supprimer (ou du moins limiter) les interruptions de services lors des déploiements avec la possibilité de rollback facilement en cas de livraison défectueuse.
- La solution doit favoriser l'innovation.
- La solution doit permettre d'effectuer des livraisons régulières stables et fiables.

Spécifications pour l’implémentation

Standardisation technologique

Un choix de technologie uniforme doit être fait, aussi bien pour le développement des différents services de la solution que pour son infrastructure.

Mise à l’échelle

Un mécanisme de mise à l'échelle ne peut être atteint qu'avec une solution d'orchestration de containeur.

Disponibilité géographique

La mise en place de différents serveurs proxy stratégiques et d'une solution d'autoscaling selon la zone géographique de l'utilisateur permet de réduire le délai de réponse

Interruptions de service

L’autoscaling des différents containeurs permettra de conserver une ancienne version de l’application pendant le déploiement de la nouvelle version, évitant ainsi de générer une interruption du service pendant le déploiement.

Rollback

Le rollback peut être faciliter par une gestion du workflow git permettant un historique lisible avec une identification précise des versions. Idéalement, les éléments de configuration devront être contenus dans l’historique git pour faire partie du versionning. Ce workflow doit s’accompagner d’un process automatisé de CI/CD pour pouvoir redéployer une ancienne version rapidement.

Environnements d’expérimentation

La mise en place des environnements d'expérimentation et de tests séparés de la production et isolés des autres expérimentations en cours permettra de favoriser l’innovation en laissant les développeurs libres de leur développement.

Livraisons rapides, stables et fiables

La solution doit permettre d'effectuer des livraisons régulières pour soutenir le process AGILE initiée avec la possibilité de rollback facilement.

La solution doit offrir la possibilité d'introduire des éléments de vérification (ex : tests à passer, couverture de code minimum…) pour empêcher une mise en production instable.

Évolutivité

La mise en place d’une architecture microservice permet l'indépendance des services (donc l'ajout de nouveau service par création ou modification d'un service sans impacter les autres).

En parallèle, l'intégration d'un outil de gestion de build permet de faciliter la gestion des dépendances et donc l'ajout de services tiers.

Gestion des droits

La mise en place d’une solution de gestion des droits permettra la mise en place de services pour certains utilisateurs spécifiques.

En parallèle, la configuration de déploiement doit pouvoir se faire par zone géographique pour permettre une personnalisation selon la localisation, en se basant notamment sur les différents serveurs proxy mentionnés plus haut.

Sécurité

La sécurisation de l'application va passer par plusieurs éléments. D'une part, les communications web doivent être sécurisées par le bais d'un protocole de communication. D'autre part, la sécurité passe aussi par la sécurisation des transactions : soit une transaction s'effectue en entier, soit elle doit être totalement annulée (rollback).

Standards pour l'implémentation

Standardisation technologique

Java étant la compétence clé des équipes et pouvant parfaitement répondre aux exigences de l'application, il sera privilégié sur la partie back-end de notre solution.

Les autres choix technologiques seront à définir à l'aide de la définition d'architecture mais devront autant que possible faire partie d'une même pile technologique, autant pour les technologies de développement que pour celles d'infrastructure.

Mise à l'échelle

Kubernetes est une solution de gestion de containeur qui nous permet de mettre en place un scaling automatique de nos containeurs, les multipliant lorsque nécessaire et les détruisant lorsqu'ils ne sont plus utilisés, permettant ainsi d'absorber nos pics utilisateurs.

Disponibilité géographique

Kubernetes en plus de répliquer les containeurs pour mieux gérer la charge, peut répartir la charge à travers différents serveurs situés à travers le monde.

Interruptions de service

L'autoscaling apporté par Kubernetes peut aussi se faire de manière à supporter temporairement les deux versions en cours de l'application lors d'un déploiement, évitant ainsi toute interruption. Les nouvelles demandes seront progressivement envoyé vers des containeurs de la nouvelle version tandis que les containeurs de l'ancienne version seront fermés progressivement lorsque les utilisateurs termineront leur en-cours

Rollback

Dans le but d'assurer un historique lisible, il est conseillé de fonctionner avec une branche principale de référence (one trunk base) sur laquelle on viendra merge en fast forward sans commit de merge (ou rebase) afin d'éviter les commits superflus, les historiques illisibles et la duplication de commit. Cette méthode permettra aussi de limiter la gestion des conflits lors des étapes de merge. Dans le but d'assurer des merges request lisibles et une séparation de concern cohérente, il conviendra de bien effectuer une merge request par feature ou bugfix. Les versions seront identifiées par le biais d'un tag permettant de les retrouver facilement.

Cette lisibilité, associée à une démarche d'infra-as-code, permettant de versionner la configuration de l'infrastructure, et à une CI/CD automatisé permettra d'atteindre l'objectif de pouvoir rollback facilement en cas de livraison défectueuse.

Environnements d'expérimentation

Environnement de développement

L'environnement de développement va permettre de déployer les modifications en cours avec l'ensemble des composants de l'application afin de tester les modifications de manière indépendante les unes des autres. Chaque merge request dispose donc de son propre environnement de développement.

Environnement d'intégration

L'environnement d'intégration va permettre de déployer la branche principale afin de tester l'intégration des features et des bugfix ensembles.

Environnement de QA

L'environnement de QA permet le déploiement des releases candidates. Ces dernières seront identifiées sur la branche principale par le biais de tag avec une nomenclature spécifique

propre aux release candidate. Cet environnement permet d'effectuer la phase de recette et de soumettre les services au tests d'acceptance avant le passage en production.

Environnement de production

Une fois la release candidate validée, un tag de release est ajouté sur la branche principale afin de l'identifier, afin de pouvoir rollback vers cette version ou les précédentes en cas de dysfonctionnement. Cet environnement est destiné à l'utilisateur final.

Livraisons rapides, stables et fiables

Afin de garantir la fiabilité des livraisons, une CI/CD avec d'indicateurs spécifiques à respecter pour valider son passage doit être instaurée, tel que par exemple le taux de couverture du code.

En parallèle, chaque version doit faire l'objet de tests d'acceptance dans l'environnement de QA afin de s'assurer que les modifications apportées fonctionnent et correspondent aux attentes métier.

Les livraisons rapides et régulières garantissent par ailleurs d'obtenir des retours utilisateurs efficaces afin de pouvoir réagir rapidement en cas de dysfonctionnement ou de mauvaise définition du besoin.

Évolutivité

La mise en place d'une architecture microservice va pouvoir se faire autour du Gateway Pattern qui va nous permettre de développer nos services de manière indépendante puis de gérer les appels à ces services par le biais d'une Gateway centralisant l'ensemble de ses appels et les redirigeant vers les bons services.

Gestion des droits

Tenant compte de l'utilisation majoritaire de Java pour le développement de back-end, une solution telle que Spring Security permettant la gestion de rôle et de droits avec un certain niveau de granularité permettrait de répondre au besoin de rendre disponible des services selon le type d'utilisateur.

En parallèle, Kubernetes peut être configurer pour déployer certains services selon certaines zones géographiques spécifiques par le biais de la configuration des Pods.

Sécurité

De manière générale, mais plus particulièrement pour les éléments ayant trait avec la sécurité, il conviendra de toujours s'appuyer sur les dernières normes et version en cours pour garantir

Actuellement, le protocole de communication sécurisée à respecter dans le web est le protocole HTTPS.

Conditions requises pour l'interopérabilité

La nouvelle architecture doit permettre la disponibilité de la solution sur une majorité de plateforme afin de rendre l'application accessible depuis un appareil fixe ou mobile.

Au regard du marché actuel, le choix a été fait de rendre l'application sur appareil mobile pour les plateformes IOS et Android et sur appareil fixe depuis un navigateur internet. Les navigateurs sur lesquels nous garantiront un support sont ceux recouvrant la majeure partie des utilisateurs, à savoir Chrome, Safari, Edge (remplaçant définitivement Internet Explorer qui n'est plus supporté depuis le 15 juin 2022) et Firefox. Ce choix pourra être revu s'il apparait qu'un nombre important d'utilisateurs fait usage d'un navigateur différent

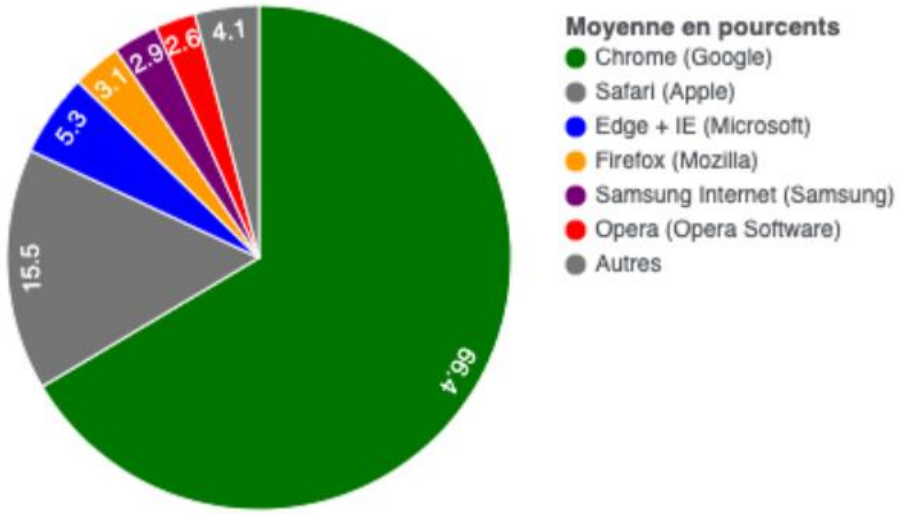

Figure 1 : Part de marché des navigateurs Web, Wikipédia (mars 2022)

Le support mobile implique de devoir choisir entre le développement d'une application mobile et un site web adapté pour mobile.

Ce choix est à confirmer avec les équipes marketing, mais il apparait que l'acquisition de nouveaux clients pour un site de e-commerce passe principalement par une recherche de produit via un moteur de recherche[1], rendant indispensable la présence d'un site web adapté pour mobile.

Cependant, il est soulevé que l'application mobile permet de fidéliser les utilisateurs en leur permettant une expérience mobile optimale, notamment pour des services tels que la géolocalisation ou des services de notification (dans notre cas, nous pourrions avoir des notifications sur les informations de livraison des produits).

Aussi, ces deux solutions semblent indissociables pour atteindre l'objectif de croissance de Foosus, bien que la priorité évidente soit la mise à disposition d'un site web accessible pour mobile et s'adaptant à tout type de plateforme et de taille d'écran.

Un site web pour mobile peut être atteint soit par le développement d'un site spécifiquement adapté pour mobile en plus d'une version desktop, impliquant notamment le développement d'un front spécifique, soit par le développement d'un site web responsive qui s'adapte à chaque situation selon la taille d'écran disponible.

[1] https://aventique.paris/application-mobile-ou-site-web

Cette deuxième solution comporte l'avantage de rester sur une même base de code, avec des pages HTML partagées, ce qui permet de faciliter la maintenance et l'évolution de la solution. Aussi, nous nous tournerons vers ce choix.

Le responsive design est rendu possible grâce à la sortie de CSS3, mais tous les technologies de développement de front faisant appel à CSS, cet élément ne sera pas déterminant dans notre choix technologique.

Une approche « Mobile First » en termes de UX design nous permettra de garantir une déclinaison plus facile de la version mobile vers la version desktop afin de bénéficier des mêmes fonctionnalités quelque soit le support.

Afin de garantir une interopérabilité entre les différents services de notre plateforme et la possibilité de faire coexister différents clients sans recoder l'ensemble de l'application, nous aurons une approche REST tout au long de la conception de notre architecture.

Enfin, une stratégie de containerisation comme mentionnée au préalable nous permet de garantir un déploiement interopérable sur de nombreux serveurs.

Conditions requises pour le management de service IT

Télémétrie et Logging

La nouvelle architecture doit permettre de remonter en temps réel une vision de la santé de la plateforme, que ce soit d'un point de vue technique afin de corriger d'éventuel bugs et d'un point de vue commercial afin de pouvoir prendre des décisions stratégiques. Ainsi la nouvelle architecture doit intégrer dès sa conception une solution de télémétrie et de logging permettant de faire remonter les informations nécessaires.

Les informations commerciales à remonter seront à définir avec les équipes marketing notamment afin d'être le plus complètes possibles.

Les informations techniques devront permettre de contrôler notamment la qualité de l'application par le biais d'indicateurs permettant de détecter d'éventuelles régressions. Les indicateurs définitifs devront être confirmés avec les équipes IT, mais on peut déjà penser au taux de couverture des tests, à la détection des vulnérabilités nouvelles ou encore à la qualité du code tel que la détection de code dupliqué.

La mise en place d'une stratégie de log permettra de garantir une traçabilité en cas de bug en production et facilitera la maintenance de l'application. Les logs seront catégorisés selon une échelle de valeur permettant de facilement identifier les informations nécessaires selon les situations :

Niveau	Usage
TRACE	Utilisé pour le débogage fin en mode verbeux de l'application
DEBUG	Utilisé pour le débogage courant de l'application. Les informations loguées avec ce niveau intéressent plus particulièrement le développeur.
INFO	Traces fournissant des informations contextualisées sur le fonctionnement général de l'application et son utilisation. Ces traces à destination du métier. Les logs d'audit sont également logués avec le niveau INFO.
WARN	Trace d'anomalies ne déstabilisant pas l'application et ne demandant pas une intervention immédiate (erreur d'ordre fonctionnel par exemple). Problème non bloquant ne faisant pas échouer la transaction métier. Permet de ne pas spammer les logs avec des logs de niveau ERROR.
ERROR	Trace d'anomalie signalant une erreur importante mais ne remettant pas en cause le fonctionnement général de l'application
FATAL	Trace d'erreurs critiques empêchant tout fonctionnement ultérieur de l'application

Les logs seront ensuite centralisés et journalisés de manière à être accessible via une IHM ou une API REST permettant de les retrouver facilement en cas de besoin, puis supprimer

automatiquement au-delà de 6 mois si aucune action n'est faite afin de ne pas encombrer les espaces de stockage inutilement.

Cycle de vie du développement

Le développement de la nouvelle plateforme devra s'effectuer en mode agile en privilégiant des cycles courts entre deux mises en production et en favorisant une collaboration constante entre les différentes parties prenantes du projet. Des mises à jour régulière permettront d'obtenir un retour rapide des utilisateurs sur les nouvelles fonctionnalités et de pouvoir apporter d'éventuelles modifications de manière réactive.

Afin d'assurer un historique de développement lisible, nous privilégierons un workflow git sur la base du One Trunk Based Development, c'est à dire avec une branche principale de référence sur laquelle on viendra merge en fast forward sans commit de merge (ou rebase) afin d'éviter des commits superflus, les historiques illisibles et la duplication de commit. Cette méthode permettra aussi de limiter la gestion des conflits lors des étapes de merge. Chaque version de logiciel sera identifiée par le biais d'un tag. Ainsi, en cas de version défaillante, il sera facile d'identifier la version précédente pour effectuer un rollback en urgence.

Chaque nouvelle version devra faire l'objet de vérification avant son déploiement que l'on pourra automatiser par le biais d'une pipeline de CI/CD. Ainsi, si l'ensemble des vérifications n'a pas été validé, le déploiement pourra être automatiquement interrompu afin d'éviter la mise en production d'une version deffectueuse.

Contraintes

International

De manière générale, Foosus doit s'attacher à respecter les lois et normes en vigueur dans les territoires sur lesquels nous dirigeons une activité.

Qualité

L'ensemble de la conception et de la mise en œuvre de la nouvelle architecture doit respecter la norme ISO 9001 afin d'assurer un niveau de qualité élevé dans nos processus.

Dans le même objectif, tout code doit respecter les normes qualité propre à son langage. Par exemple, tout code HTML doit respecter la norme W3C.

Protection des données

En Europe, la collecte et le traitement de données personnelles (nom, prénom, adresse, numéro de sécurité sociale, etc.) par les entreprises sont soumis à des obligations destinées à protéger la vie privée et les libertés individuelles des personnes dont les données sont collectées.

Le règlement de protection des données (RGPD) s'applique à tous les organismes qui procèdent au traitement de données à caractère personnel dès lors qu'un résidant européen est directement visé par une collecte ou un traitement de données.

L'ensemble des données de tout utilisateur (consommateurs ou fournisseurs, voire même salariés) de la plateforme résidant sur le territoire européen est ainsi concernés par cette contrainte.

Paiement bancaires

Si un système de transaction bancaire est mis en place sur la plateforme, y compris par un tiers, il conviendra de respecter les normes les plus récentes en matière de paiement en ligne afin de sécuriser les transactions, notamment la norme PCI DSS portant sur les standards de sécurité des données pour l'industrie des cartes de paiement.

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.