Spécification des Conditions requises pour l'Architecture

Projet : Foosus Géoconscient

Client : Foosus

Préparé par : Loïc PIRIOU

N° de Version du Document : 0.1

Titre : Spécification des Conditions requises pour l'Architecture

Date de Version du Document : 14/03/2022

Revu par :

Date de Révision :

Table des Matières

Objet de ce document (Cahier des charges)

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

Métrique	Technique de mesure	Valeur cible	Justification
Nombre d'adhésions d'utilisateurs par jour	Requêtes en BDD sur le nombre d'adhésions par jour	Augmentation de 10 %	Cette métrique a chuté rapidement au cours des derniers mois et doit être améliorée en priorité.

Adhésion de producteurs alimentaires	Requêtes en BDD sur le nombre d'adhésions par mois	Passer de 1,4/mois à 4/mois	
Délai moyen de parution d'une évolution/modification		Réduit de 3,5 semaines à moins d'une semaine	Nous ne pouvons plus accepter de désactiver la plateforme à chaque installation d'une nouvelle version ou à chaque modification du schéma de la base de données.
Taux d'incidents de production P1		Pour commencer : réduit de >25/mois à moins de 1/mois.	L'an dernier, 12 de nos pannes ont été provoquées par la publication par une ou plusieurs équipes de modifications lourdes qui n'ont pas eu les résultats escomptés.

Conditions requises pour l'architecture

Les conditions requises pour l'architecture sont la mise en place d'une architecture microservices prenant en charge les solutions Web et mobiles, ayant une base de données standard.
L'architecture devra être évolutive pour que nous puissions déployer nos services sur diverses régions, dans des villes et des pays donnés.
L'architecture peut inclure de nouveaux composants personnalisés ou des composants du commerce pour favoriser la flexibilité, la stabilité et l'extensibilité.
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.

Contrats de service business

Accords de niveau de service

Les équipes de développement doivent itérer sur des périodes courtes afin de produire de nouvelles versions plus rapidement auprès des utilisateurs et éviter des temps longs de déploiement.
Les équipes de développement doivent mettre en place des tests de performances afin de s'assurer que l'application peut gérer un trafic intense, d'obtenir une qualité similaire sur n'importe quels supports.
La mesure effective du temps d'activité doit être de 99,97%.
Une réplication de la base de données doit être mises en place sur chaque action faite en base de données afin d'éviter toute perte de données.
Les services de support devront effectuer une réponse sous 24h pour tout incident grave (indisponibilité de l'application, perte de données, …).

Contrats de service application

Objectifs de niveau de service

Chaque nouvelle version doit être de taille réduite, présenter peu de risques et rester accessible en tout lieu et à tout moment.
L'application doit avoir des performances similaires quelques soit la zone géographique et le système utilisé.
L'application doit être disponible sur différents serveurs en parallèle afin de gérer la surcharge de connexion.
L'application doit être active 99,95% du temps.
L'application doit toujours permettre aux clients d'obtenir les informations associées à son compte.

Indicateurs de niveau de service

Le déploiement des nouvelles versions de l'application doit être invisible pour les clients. Nos utilisateurs doivent pouvoir accéder facilement à nos services.
Les utilisateurs situés dans des zones géographiques spécifiques, sur des connexions lentes (par exemple, avec des téléphones portables) aussi bien que sur des réseaux haut débit doivent pouvoir avoir les mêmes performances.
Si le système est surchargé, les utilisateurs connectés doivent pouvoir continuer à accéder à tous les services.

Les utilisateurs doivent pouvoir accès à l'application à n'importe quel moment sans interruption de service.
Une réplication de la base de données doit être effectuées pour éviter toutes pertes de données.

Lignes directrices pour l'implémentation

Une architecture de microservices se différencie d'une approche monolithique classique par le fait qu'elle décompose une application pour en isoler les fonctions clés. Chacune de ces fonctions est appelée « service » et ces services peuvent être développés et déployés indépendamment les uns des autres. Ainsi, chacun peut fonctionner sans affecter les autres.
Un microservice est donc une fonction essentielle d'une application, qui est exécutée de manière totalement indépendante des autres services. Cependant, une architecture de microservices ne se résume pas à la relation entre les fonctions fondamentales d'une application : elle permet aussi de restructurer les équipes de développement et la communication entre les services pour mieux se préparer aux inévitables pannes, mais aussi aux évolutions futures et à l'intégration de nouvelles fonctions.
Cela vous aide à adopter la dimension technologique du DevOps et à simplifier l'itération et la livraison continues (CI/CD) tout en les rendant plus accessibles.
Dans le but d'automatiser et de monitorer l'ensemble du cycle de vie des applications, des phases de test jusqu'à la mise en production, je préconise de mettre en place la pratique DevOps qui permet de prendre en compte les contraintes de déploiement dès la phase de programmation. En prenant en compte cela, les logiciels sont de facto plus performants. Ils engendrent ensuite moins de bugs et de failles de sécurité. Des problèmes qui peuvent provenir de l'application en tant que telle ou de ses dépendances avec d'autres couches du système d'information (serveur d'applications, serveur physique, clusters...). Les incidents sont solutionnés plus rapidement. L'enjeu du DevOps est ainsi d'améliorer la satisfaction des utilisateurs. Avec pour objectif d'appliquer la logique des méthodes agiles à l'ensemble de l'activité informatique, le DevOps se concrétise par la mise en place de pipelines d'intégration et de livraison continues (CI/CD) courts.

Spécifications pour l'implémentation

Une architecture de microservices permet d'agir sur plusieurs points :

- Scalabilité, le couplage faible permet de déployer une deuxième instance d'un module en limitant les effets de bords,
- Organisation, chaque équipe travaille sur son module sans impacter ou être

impacté par les modules gravitant autours ; sur la base de contrats de service et de jeux de tests/bouchonnages préalablement définis,
- Evolutivité, chaque module subit un cycle de gouvernance indépendant, qui permet de faire évoluer les modules indépendamment (versionning des services).

Pour permettre de renforcer l'architecture microservice, il faut passer par une rigueur inculquée aux équipes de développement. Ce qui permettra, dans le cas où les concepts liés aux microservices sont bien appliqués, d'avoir un code modulable, scalable et plus facilement évolutif/maintenable.
La livraison continue (CD) repose sur la livraison de nouvelles versions de code aussi rapidement que possible aux clients. Les tests automatisés sont essentiels pour atteindre cet objectif.
La CD fait partie d'un pipeline de déploiement plus large. Elle succède à l'intégration continue (CI), dont elle dépend également. La CI est entièrement responsable d'exécuter les tests automatisés pour tout nouveau changement de code et de garantir que ces changements ne font pas regresser les fonctionnalités établies ou n'introduisent pas de nouveaux bugs.
La CD est déclenchée lorsque le plan de tests automatisés valide l'étape de CI.

Standards pour l'implémentation

Comment industrialiser des microservices ? Des outils permettent de gérer le déploiement et l'orchestration des packages, tels que Docker et Kubernetes ce qui permet de déployer à la volée une instance d'un microservice sans se soucier de la couche OS et matériel ; tout en bénéficiant de l'isolation de Docker.
La problématique repose sur le fait de faire connaitre le nouveau package déployé aux autres packages déjà déployés, il faut pour cela un système de répartition de charge flexible.
La mise en place d'une API Gateway, passerelle entre les apis exposées par les microservices et l'application front consommant ces API, permettrait :

- Agréger ou désagréger différentes API
- Réaliser la transformation de protocoles ou de données, l'utilisation de protocoles non compatibles internet, la distribution de données sur différentes API mais aussi la mise en cache et l'orchestration des différentes API
- Centraliser l'implémentation de la sécurité évitant ainsi de le faire au niveau des API du serveur applicatif
- Participer à la disponibilité et l'évolutivité de l'application
- Contribuer à la simplicité de la consommation des API par les consommateurs

Contrairement au test manuel, le test automatisé s'exécute sans l'intervention d'un humain. Cette méthode nécessite l'utilisation de solutions informatiques afin d'exécuter

les actions prédéterminées dans un script et analyser le produit sur des parcours bien précis.
Le test automatisé a pour objectif de simplifier autant que possible les efforts de test grâce aux scripts. Le test est alors exécuté selon celui-ci, les résultats sont signalés et comparés aux résultats des essais antérieurs.
En ce qui concerne le temps, le caractère répétitif du test automatisé permet de tester les applications en continu mais aussi de tester plus et mieux. C'est notamment le cas pour les tests de non-régression. Cela favorise une mise en production plus rapide ainsi qu'une réduction des délais de livraison. Aussi, le test automatisé permet une flexibilité au niveau du temps: les tests peuvent être exécutés en dehors des horaires de travail.
Les campagnes de test peuvent être tracées grâce à l'automatisation des tests car les automates préservent l'historique de l'exécution des tests. Les chefs d'équipe peuvent ainsi assurer un suivi fiable de la qualité d'exécution des tests.

Conditions requises pour l'interopérabilité

L'interopérabilité se base sur des protocoles, normes ou règles de communication. C'est la capacité à communiquer de deux logiciels différents, issus d'équipes de développement différentes.
Les API (Application Programming Interface) désignent les définitions et les protocoles qui facilitent la création et l'intégration des logiciels d'applications. Elles indiquent à un système, comme un médiateur ou un interprète, ce que l'utilisateur attend de lui. Le standard REST (Representational State Transfer) est aujourd'hui un type d'architecture référence pour les API. Les échanges de données via les API s'effectueront dans le format JSON (JavaScript Object Notation).
Le JSON est un format standard utilisé pour représenter et transmettre des données structurées sur des sites web.
Les normes ISO (Organisation Internationale de Normalisation) : Une norme est un document de référence élaboré par un organe de normalisation reconnu comme l'ISO et l'AFNOR en France Une norme définit les règles, caractéristiques, bonnes pratiques et recommandations applicables à des produits, services, méthodes, processus ou organisations.
La norme ISO 9001 définit des exigences pour la mise en place d'un système de management de la qualité, cela se concrétiserait par une amélioration en permanence de la satisfaction des clients et de leur fournir des produits et services conformes.
Quant à la norme ISO 27001, c'est une norme de sécurité des systèmes d'informations. Elle définit les exigences pour la mise en place d'un système de management de la sécurité de l'information (SMSI). Le SMSI recense les mesures de sécurité, dans un périmètre défini, afin de garantir la protection des actifs de l'organisme. L'objectif est de protéger les fonctions et informations de toute perte, vol ou altération, et les systèmes

informatiques de toute intrusion et sinistre informatique. Cela apportera la confiance des parties prenantes.
Les RGPD (Règlement Général sur la Protection des Données) encadre le traitement des données personnelles sur le territoire de l'Union européenne. Ainsi, le traitement ou la collecte des données amène des obligations spécifiques pour garantir la protection des données qui nous sont confiées.

Conditions requises pour le management du service IT

La mise en place du projet s'effectuera en respectant la culture Lean de Foosus.
Le Lean consiste à la pérennité et la rentabilité de l'entreprise et des emplois en passant par la satisfaction des clients et de leur envie de revenir. Les moyens d'y parvenir sont tout aussi importants : en respectant la société et l'environnement, les employés et les fournisseurs partenaires.
En dynamisant toute l'entreprise. La dynamique vient à la fois de l'engagement de tous et de la fluidité.
L'engagement des clients et leur envie de promouvoir les produits ou services qu'ils ont acheté naît de l'engagement des salariés de l'entreprise. L'engagement des salariés part d'un postulat simple : chaque employé est riche de savoir-faire, de choses observées et d'idées. Les outils Lean sont des outils d'apprentissage, qui permettent l'expérimentation, au sein d'équipes qui s'approprient leurs résultats, collaborent avec d'autres équipes, dans une optique de développement continu. Le manager n'est plus celui qui fait faire mais celui qui fait apprendre. Le manager réapprend à être sur le terrain. Il atteint ses objectifs en développant ses collaborateurs.

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.

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.