Spécification des Conditions requises pour l'Architecture

19 juin 2021

Aftis Saadi

Table des matières

1 Informations sur le document

FIGURE 1 – Documents info

2 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.

3 Mesures du succès

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.

FIGURE 2 – Indicateurs de succès

4 Conditions requises pour l'architecture

— Tirer profit 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.

— L'architecture devra être évolutive pour que Foosus puisse déployer ses services sur diverses régions, dans des villes et des pays donnés.
— Les améliorations et autres modifications apportées aux systèmes de production devront limiter ou supprimer la nécessité d'interrompre le service pour procéder au déploiement.
— Les fournisseurs et les consommateurs de Foosus doivent pouvoir accéder à la 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.
— Le système 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.
— 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

— Assurance d'avoir un service qui fonctionne avec tous les types d'appareils.
— La solution doit être utilisable partout dans le monde.
— Le système sera fonctionnel a 99.5 % du temps (Hors problèmes externe : panne serveur cloud...)

6 Contrats de service application

6.1 Objectifs de niveau de service

— Un système qui n'a pas besoin d'être mis hors service pour l'ajout de mises à jour.
— Une application qui tourne 24/24 et 7/7.
— L'application doit fonctionner partout dans le monde.
— Le système doit fonctionner avec n'importe quel type d'appareil et s'adapter aux bandes passantes.
— Elle doit pouvoir prendre en charge divers types d'utilisateurs (par exemple, fournisseurs, back-office, consommateurs).

7 Lignes directrices pour l'implémentation

— 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

8 Spécifications pour l'implémentation

8.1 Recodage du front et du back

Toutes les fonctionnalités du back-end devront être recodés en Spring-Boot v-2.5.0, en utilisant Spring Data JPA comme ORM au lieu de Hibernate, concernant le Front on devra aussi tout recoder en Angular v- 11.0.

8.2 Géolocalisation

8.2.1 Avoir la localisation d'un utilisateur

Il y a 2 scénarios qui s'offrent à nous :
— L'utilisateur accepte d'être géolicaliser en acceptant la pop-up sur son navigateur.
— L'utilisateur préfère écrire son adresse manuellement (où choisi une autre adresse), dans ce cas on créera dans nos "fichier css" un champ de saisie pour l'utilisateur, puis on ajoutera la fonctionnalité "Auto-complete" qui restreindra les suggestions Autocomplete pour ne renvoyer que des adresses.

8.2.2 Suggestions de magasins proche

On commence par instancier un objet "DistanceMatrixService", ensuite on crée une méthode nommée "calculateDistances", cette dernière prend en entrée la destination de départ de l'utilisateur ainsi qu'un arrayList contenant la liste des magasins proches, la arrayList est constitué en envoyant une requête à notre base de données qui contient tous les magasins et en affinant la recherche avec le pays + code postal du client, ce qui nous donne en résultat une liste de magasins aux alentours du client. "calculateDistances" crée ensuite un tableau d'objets incluant l'ID du magasin, la distance exprimée sous la forme d'une chaîne lisible ainsi que la distance exprimée sous la forme d'une valeur numérique (en mètres), et elle organise ce tableau.

8.3 CDN

Un CDN (Content Delivery Network) est une plateforme de serveurs hautement distribuée qui permet de minimiser les délais de chargement du contenu des pages web en réduisant la distance physique entre le serveur et l'utilisateur. Les utilisateurs du monde entier peuvent ainsi visualiser le même contenu de haute qualité sans temps de chargement trop longs. Pour ce faire nous utiliserons le service aws "CloudFront".

8.4 Reverse Proxy

Un reverse proxy est un type de serveur proxy qui se trouve généralement derrière le pare-feu d'un réseau privé et qui dirige les demandes des clients vers le serveur Back-End approprié. Un reverse proxy fournit un niveau supplémentaire d'abstraction et de contrôle pour assurer la fluidité du trafic réseau entre les clients et les serveurs. Notre reverse Proxy sera compris dans l'offre de aws "CloudFront".

8.5 Déploiement

Foosus a explicitement demandé à ce qu'il n'y ai plus de "DownTime" lors du déploiement d'une nouvelle mise à jour, pour ce faire on va utiliser la stratégie de déploiement "Blue/Green" qui consiste à déployer la nouvelle version dans de nouveaux serveur et rediriger les utilisateurs vers ces derniers, si un dysfonctionnement arrive on repasse sur les anciens serveurs qui contiennent toujours l'ancienne version qui elle est stable, c'est donc une solution risquée mais, on limitera les risques au maximum avec des pratiques tels que "Environnement de pré-production", "Feature Toggeling", "tests d'intégration", "Code Coverage >70 %".

9 Standards pour l'implémentation

— 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.
— Les solutions open source sont préférables aux solutions payantes.

10 Conditions requises pour l'interopérabilité

— La base de données relationnel communiquera avec le Back-end grâce à Spring Data JPA qui est un ORM (Object Relational Mapping).
— Le transfère de données entre le Back-End et le Front-end se fait avec format JSON (Jackson permet de transformer des objets Java en Json et vice versa).

— L'api utilisé pour calculer les distances implémente l'utilisation de Spring-boot.

11 Conditions requises pour le management du service IT

L'équipe devra suivre le protocole suivant :
— On utilisera plus la méthode Kanban mais la méthode Agile Scrum.
— le lead-tech de chaque équipe sera en charge de l'attribution des tâches, ceci se fera via la plateforme Jira.
— Chaque employé travaillera sur sa branche Git et ne devra "push" que ce dont il est sûr que ça fonctionne (le code coverage lors des tests doit être supérieur a 70%).
— Chaque journée commencera par une réunion de 30 minutes "daily scrum" pour parler de ce qui a été fait la veille, les problèmes rencontrés et ce qu'on compte faire le jour même.
— Des "Sprint" seront faits toute les semaines pour la livraison du livrable.
— Des réunions mensuelles avec les hauts dirigeant de l'entreprise pour la présentation des livrables et de la prochaine étape à suivre seront au programme , ainsi que des réunions d'urgence en cas d'avancée/problème majeur.

12 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.

13 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.
— 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.