Contrat d'architecture avec les fonctions de conception et de développement

Table des matières

Objet du document 3

Description du projet 4

Introduction 4

Contexte 4

Objectifs 5

Périmètre 6

Parties prenantes 6

Approche managériale 6

Approche technique 7

Présentation de l'architecture 8

Phase 1 : Mise en place d'une architecture microservices 8

Les outils indispensables 9

Frontend 9

Backend 10

Stockage des données 12

Réutilisation du système existant 12

Phase 2 : Migration complète du système 12

Frontend 13

Backend 13

Stockage des données 14

Principes stratégiques 16

Principes généraux 16

Principes business 16

Principes data 16

Principes d'application 16

Principes technologiques 17

Plan de travail 18

Détails d'implémentation de la première phase 18

Détails d'implémentation de la seconde phase 18

Plan de communication 19

Gouvernance et risques 20

Rôles et responsabilités 20

Analyse des risques 21

Hypothèses 22

Critères d'acceptation et procédures 23

Critères d'acceptation 23

Procédures d'acceptation 23

Vérification d'aptitude du bon fonctionnement 23

Vérification du service régulier 24

Procédures de changement de périmètre 24

Approbations 25

Objet de ce document

Ce document est un contrat avec les utilisateurs business pour le projet d'évolution et de migration de l'intégralité des systèmes d'informations numériques de l'entreprise Foosus.

Ce contrat est, simplement, un accord entre les parties prenantes, il nous permet également d'apporter un maximum de précision concernant le projet et son contexte.

Nous détaillons également les objectifs, quelle que soit leur nature, qui nous ont été transmis et, donc, que devra remplir le système à concevoir dans le cadre de ce projet.

Nous définirons, de manière aussi précise que possible, les parties prenantes gravitant autour de ce projet, les approches managériale et technique sur nous préconisons d'adopter afin d'optimiser la réalisation.

Par la suite, nous étudierons, de manière très détaillée, l'architecture que nous proposons ainsi que ses différents composants. Cette description de l'architecture et de ses composants s'attarde vivement sur les technologies et autres outils à mettre en œuvre pour rendre fonctionnel et optimiser au maximum le système à développer.

Nous abordons aussi les principes à prendre en compte dans le cadre de la réalisation de ce projet.

Ce document propose aussi le plan d'implémentation, complet et détaillé, de l'architecture que nous proposons ainsi que le plan de communication que nous conseillons d'adopter pour une interaction efficace entre les différentes parties prenantes au cours de ce projet.

Ce document comportera aussi une analyse de la gouvernance, au sens le plus général, des tâches nécessaires à l'accomplissement du projet. De plus, nous verrons les risques encourus liés au projet.

Dans ce document, nous verrons également les différentes hypothèses, quelle que soit leur nature, qui ont exprimées par Foosus.

Les critères d'acceptation du projet, les procédures de validation nécessaire à toute mise en service ainsi qu'un rappel de la marche à suivre lors du changement de périmètre seront également étudier.

Enfin, il ne restera plus qu'une dernière partie, très concise, elle concerne les signatures à apposer sur ce document de façon à signifier l'acceptation de ce qui y est spécifié.

Description du projet

Introduction

La plateforme actuelle de Foosus a atteint un point au-delà duquel elle ne peut plus soutenir les projets de croissance et d'expansion de l'entreprise, effectivement, après plusieurs années de développement, la solution technique complexe n'évolue plus au rythme de l'activité et risque d'entraver la croissance.

Les équipes de développement sont pleinement investies dans l'extinction d'incendies et dans le maintien en état de marche du système, ce qui a ralenti la capacité à livrer de nouvelles fonctionnalités et à rester compétitifs au sein d'un marché nouveau et imprévisible.

Les études de marché et les analyses commerciales montrent que leurs clients souhaitent acheter local et soutiennent les producteurs locaux, de plus, pour le moment, leurs concurrents n'ont pas ciblé cette niche.

Ils donc veulent s'appuyer sur les connaissances acquises pendant ces trois dernières années et créer une plateforme qui mettra en contact des consommateurs avec des producteurs et des artisans locaux dans toutes les catégories de besoins.

Contexte

L'objectif commercial 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.

Une nouvelle plateforme d'e-commerce est nécessaire afin d'améliorer sa compétitivité par rapport aux grandes entreprises d'e-commerce internationales.

Foosus a identifié plusieurs objectifs généraux qui doivent être satisfaits quelle que soit la nouvelle direction technique adoptée pour améliorer sa capacité opérationnelle.

La nouvelle plateforme devra également permettre à leurs 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.

Objectifs

Afin de rendre son application la plus attractive possible et ainsi gagner un grand nombre d'utilisateurs, de tous bords, au travers d'un système sécurisé et fiable, Foosus a défini plusieurs objectifs à atteindre.

En premier lieu, il souhaite tirer parti de la géolocalisation afin relier des fournisseurs et des consommateurs, ainsi ils proposeront 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.

Ils désirent également une architecture évolutive, scalable et résiliente, qui est, à la fois, capable de supporter un déploiement de leurs services sur diverses région, pays et villes et des pays donnés, mais également en mesure de s'adapter à un grand nombre d'utilisateurs simultanément sans causer de quelconque problème. L'architecture doit pouvoir s'équiper d'outils permettant une stratégie de scalabilité avancée ainsi qu'un déploiement efficace, ayant la capacité d'être distribué au besoin.

De cette diversité géographique, un nouvel aspect entre en jeu, les arrêts du système volontaires doivent être supprimer, effectivement, des décalages horaires résulte le fait que le système est amené à être utilisé tout au long des 24 heures journalière, c'est pourquoi, Foosus a bien spécifié que les améliorations et autres modifications apportées aux systèmes de production doivent limiter ou supprimer la nécessité d'interrompre le service pour procéder au déploiement.

Il est aussi à mettre en exergue le fait que leurs fournisseurs et leurs consommateurs doivent pouvoir accéder à leur solution où qu'ils se trouvent sans perte de performance. Cette solution doit donc être utilisable avec des appareils fixes comme mobiles et elle doit tenir compte des contraintes de bande passante pour les réseaux cellulaires et les connexions Internet haut débit.

En d'autres termes, Foosus ne recherche pas une solution désuète mais plutôt tendancielle et bénéficiant d'une optimisation au niveau réseau, ce qui pourrait se traduire par de multiples manières tel que l'utilisation accrue de la mise en cache, la limitation des données superflus lors des échanges, des images de qualité standard ou encore l'utilisation, pour la partie web, d'une « Single Page Application », une page se chargeant une seule et unique fois, de façon à éviter les rechargements inutiles aux changements de page.

En outre, le nouveau système doit réaliser, au moins, les mêmes fonctionnalités que l'actuel, comme par exemple, pouvoir prendre en charge divers types d'utilisateurs, tel que des fournisseurs, des consommateurs, des administrateurs, et ceci avec des fonctionnalités et des services spécifiques pour ces catégories.

Enfin, 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. Il faudrait donc, à minima, instaurer la livraison continue, si possible, le déploiement continu.

Périmètre

Parties prenantes

Le tableau, ci-dessous, vise à apporter un maximum de détails concernant les parties prenantes concernées par le projet. On va y retrouver leur nom, leur poste ainsi que leurs fonctions principales.

Nom	Poste	Rôle
Pete Parker	Responsable ingénieur	Coordonne les équipes techniques et veille au respect de la mise en œuvre technique, que ce soit aux niveaux des piles technologiques ou de la durée du développement.
Ayrton De Abreu Miranda	Architecte logiciel	Conçoit les architectures logicielles en fonction des besoins et contraintes exprimés et veille à son strict respect. En fonction des projets, d'autres tâches peuvent lui être confiées tel que la gestion de projet, les études, la recette, l'exploitation, les infrastructures, le support , la méthode et la qualité, la sécurité, le support et l'assistance aux utilisateurs .
Joe Harkner	Responsable infrastructure	Définit et met en œuvre la stratégie de production informatique. Il a pour mission de garantir la cohérence de l'infrastructure du système d'information et la qualité du service rendu aux utilisateurs dans un souci de productivité, maîtrise des coûts et respect des délais.
Natasha Jarson	Directeur informatique	Est en charge de l'implémentation informatique de la stratégie de l'entreprise dans laquelle il travaille. Il est responsable de la sécurité et de la fiabilité du système informatique ainsi que de son évolution au fil des changements techniques fréquents dans ce domaine.
Daniel Anthony	Directeur produit	Accompagne, structure et accélère la croissance économique de l'entreprise. Il est notamment en charge de la stratégie du produit et supervise tous les éléments de celui-ci, de sa conceptualisation à ses performances de lancement.
Christina Orgega	Directeur marketing	Dirige l'activité marketing globale d'une organisation. Il est également chargé de rediriger de l'étude de marché à la stratégie publicitaire en passant par l'orientation du développement des offres d'une entreprise, le ciblage de la clientèle et l'image de marque.
Jo Humar	Directeur financier	Détermine une stratégie et supervise la mise en œuvre des instruments requis : plans de financement, suivi de leur mise en œuvre, gestion de la trésorerie pour se procurer les fonds en veillant à ce que l'argent soit utilisé de la façon la plus performante.
Ash Callum	Directeur général	Établit les stratégies d'évolution et de développement d'une structure, tant au point de vue comptable, financier, managérial que technique. Il est en tête de l'ensemble des opérations et en donne les orientations à long et court termes.

Principales parties prenantes concernées par le projet

Approche managériale

Le précédent responsable de l'architecture nourrissait une culture où les équipes de développement étaient encouragées à expérimenter et essayer librement de nouvelles approches techniques.

Cette approche managériale a résulté dans la construction d'une équipe de 15 développeurs aimant travailler chez Foosus et étant vivement investis dans la satisfaction des clients et dans le succès du produit.

Bien que cette culture, en plus d'être plaisante pour les équipes techniques, possède de nombreux avantages, il est nécessaire de souligner qu'elle entraine aussi d'importants défauts tel que la diversification des modèles employés, augmentant le coût de maintenance et affaiblissant le potentiel de réutilisabilité.

Dans un contexte compétitif où la mise en place de nouvelles fonctionnalités ainsi que la résolution des bugs doivent sans délais, il n'est pas possible de continuer dans cette direction, effectivement, une certaine standardisation des méthodes doit être imposé à l'ensemble du personnel technique.

En d’autres termes, pour faire face à une concurrence féroce, nous pouvons dire qu’il faut davantage privilégier l’homogénéité technique d’un système que l’hétérogénéité, ce qui favorisera le potentiel de croissance de l’entreprise en rendant son système adaptable aux multiples cas d’utilisation.

Approche technique

Les analyses de marché indiquent que la correspondance de Foosus avec le marché a été éclipsée par l’instabilité de la plateforme et par une image de marque négative causée par des interruptions de service visibles par le public.

En réponse à un fort déclin des inscriptions utilisateurs, l'entreprise souhaite conserver la plateforme existante en mode maintenance et ainsi restructurer les équipes afin de livrer une plateforme à l’architecture davantage travaillée, qui lui permettra de grandir de manière alignée sur sa vision business de soutien aux marchés locaux et ceci en évitant les pannes quelconques du système.

Les inscriptions constituent une métrique clé aux yeux de leurs investisseurs et ne peuvent être améliorées que par l’agilité nécessaire pour innover rapidement et ainsi avoir la capacité d’expérimenter avec des variantes d’offres produit existantes.

En somme, l’objectif business est de sortir de manière rapide et itérative un nouveau produit qui pourra coexister dans un premier temps avec la plateforme existante, avant de la remplacer dans son intégralité.

Présentation de l'architecture

Afin de décrire, au mieux, l'architecture cible, il est important de commencer par dire que son accomplissement sera effectif à l'issue de deux phases distinctes.

Une première architecture que l'on peut qualifier de première phase, permettant de développer un système fonctionnel rapidement pour faire face à la concurrence, en priorisant l'utilisation d'outil open source. Ces deux motivations phares ont bien été exprimées par Foosus.

Une seconde architecture, de fait, la seconde phase, pensée pour s'intégrer parfaitement à la première, effectivement il s'agit d'une évolution de la phase 1. Cette seconde phase permet principalement d'unifier le style architectural et ainsi faire disparaitre l'ancien système hétéroclite.

Bien que cette section présente les choix architecturaux et technologiques devant être mis en œuvre dans le cadre de ce projet, elle n'a en aucune façon vocation à les justifier.

Phase 1 : Mise en place d'une architecture microservices

Comme on va le voir sur le diagramme de composant ci-dessous, il a été choisi de mettre en place une architecture microservices.

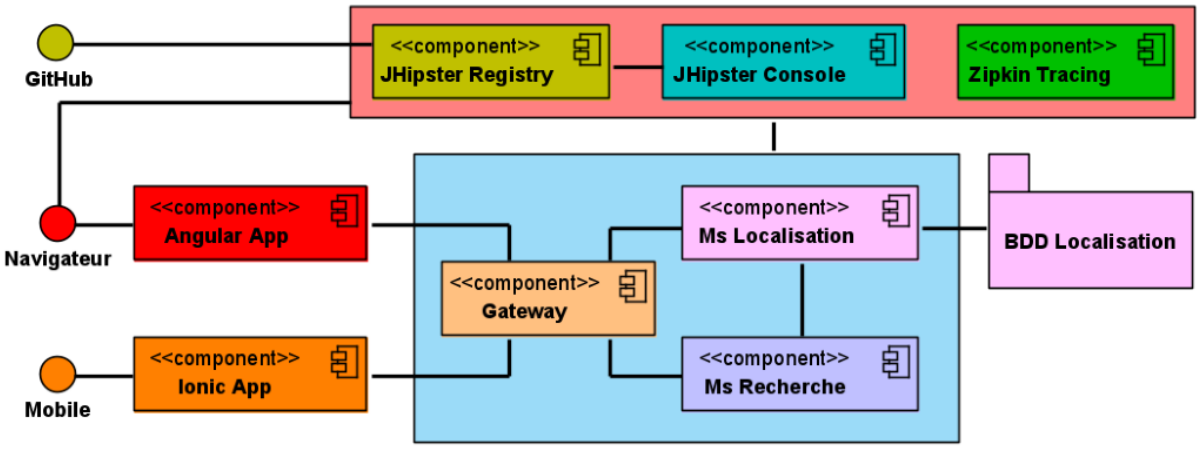

Diagramme de composant – Phase 1

Avant de nous attarder sur le diagramme de composant, nous allons faire une légère parenthèse et présenter les outils indispensables qui n'y sont pas représentés mais qui sont à ajouter au système.

Par la suite, afin d'apporter un maximum de clarté, nous proposons une présentation des composants de ce diagramme en 3 parties : le frontend, le backend et le stockage des données.

Enfin nous terminerons sur une dernière partie, elle non plus n'est pas représentée sur le diagramme et décrit les parties du système existant qui devront être réutilisées.

Les outils indispensables

Afin d'avoir le système le plus performant possible et de respecter au mieux les exigences du projet, certains outils vont être utilisés pour compléter le système proposé.

Docker sera utilisé comme outil de conteneurisation des composants, Kubernetes comme outil d'orchestration des conteneurs et Jenkins comme serveur d'automatisation des tests.

Ce triumvirat est mis en place pour, en autres, respecter le principe de déploiement continu.

Frontend

La première chose à prendre en compte en ce qui concerne la partie frontend est que nous ne ferons pas les évoluer les multiples clients web actuels ainsi que le client mobile existant. Nous prenons le parti de développer de nouveaux clients, web comme mobile.

Il est à noter que la partie frontend ne prendra en charge que le contexte backend prévu dans cette première phase. Les fonctionnalités déjà présentes dans le système seront, bien évidemment, intégrées dans les nouvelles applications lors de la seconde phase.

Application web – Angular App

Nous proposons une application web implémentée avec le langage Typescript et, plus précisément, avec le framework Angular. Une partie du code sourde sera généré avec l'outil JHipster, en excluant la partie backend lors de la création du composant afin de marquer la séparation entre ces parties.

En ce qui concerne le souhait de géolocalisation exprimé, nous proposons d'utiliser la librairie Javascript Geolocation qui a l'avantage de correspondre à l'esprit open source également émis par Foosus. Pour ce qui est de l'affichage des résultats de recherche sur une carte de manière à les rendre le plus clair possible, nous proposons d'utiliser la libraire Javascript de Google : Google Maps.

Application mobile – Ionic App

Nous proposons une application mobile cross-platform implémentée avec le langage Typescript et, plus précisément, avec le framework Ionic. Une partie du code sourde sera, d'une part, généré avec l'outil JHipster, en excluant la partie backend lors de la création du composant afin de marquer la séparation entre ces deux parties et, d'autre part, basé sur la réutilisation de code Angular, c'est à dire de l'application web, raccourcissant encore davantage le temps alloué au développement.

En ce qui concerne le souhait de géolocalisation exprimé, nous proposons d'utiliser la fonction native des mobiles qui a l'avantage, en quelque sorte, de correspondre à l'esprit open source émis par Foosus. Pour ce qui est de l'affichage des résultats de recherche sur une carte de manière à les rendre le plus clair possible, nous proposons d'utiliser la libraire Javascript de Google : Google Maps.

Backend

Pour la partie backend, le choix des technologies s'est également reposé sur l'utilisation de JHipster, cette fois ci en excluant les parties frontend des projets et en ne gardant donc que les sections implémentées Java et Spring.

Par exception, comme on va le voir, il est prévu un outil de traçage des requêtes qui n'est pas pris en charge par JHipster, ce qui semble évident étant donné qu'il ne nécessite aucun développement, il est effectivement disponible en téléchargement sous la forme d'un Jar ou directement sous la forme d'une image Docker.

Serveur passerelle – Gateway

La pile prête à l'emploi fournie par JHipster se compose principalement de 3 outils.

Zuul pour accomplir les fonctions de gateway, c'est à dire être le point d'entrée unique du système. Il a donc la responsabilité de contrôler la requête reçue, de la valider ou non et en fonction du résultat de sa validation, de la transmettre au service concerné.

Ribbon pour répartir la charge côté client, c'est à dire entre les différentes instances des microservices.

Spring Security permettant de sécuriser le composant et ceci de plusieurs manières possibles comme, par exemple, la simple identification par identifiant et mot de passe ou bien l'utilisation du standard JWT.

Microservice de gestion des localisations – Ms Localisation

Un microservice de localisation est nécessaire afin de répondre aux exigences imposées dans le cadre de ce projet. Pour rappel, ces exigences concernaient la mise en place de la géolocalisation ainsi que la prise en charge de multiples régions (pays, ville, etc.).

Comme le reste du système backend, il sera sous Java avec Spring, cependant pour participer à l'accomplissement de la tâche de géolocalisation il faudra l'utilisation d'un système externe de conversion d'adresses en coordonnées géographiques (et inversement).

Bien qu'elle soit payante, l'API Geocoding de Google est préconisée, malheureusement aucun outil open source fiable n'a été trouvée pour concurrencer l'outil proposé par Google.

Microservice de gestion des recherches – Ms Recherche

Un microservice de recherche est également nécessaire afin de répondre aux exigences imposées dans le cadre de ce projet. En effet, bien qu'un système de recherche soit présent dans l'application existante, il ne prend pas en charge les fonctions de localisation qui vont être implémentées.

La pile technologique présente dans ce microservice est, encore une fois, le langage Java, accompagné du framework Spring ainsi que l'utilisation de l'outil JHipster.

Ce composant remplace donc « Foosus Search System » présent dans la solution initiale.

Serveur d'enregistrement – JHipster Registry

JHipster Registry est un composant quasiment clé-en-main fourni par JHipster. En plus de mettre à disposition plusieurs tableaux de bord de contrôle et de management, il est équipé de deux outils supplémentaires : Eureka Discovery Server et Spring Cloud Config Server.

Ce composant est donc implémenté avec Spring, framework s'appuyant sur le langage Java.

Eureka Discovery Server permet de mettre en place le serveur de découverte des instances et Spring Cloud Config Server sert à mettre en place le serveur de configuration externalisée fonctionnant, dans notre cas, avec GitHub.

Il faut souligner que les tableaux de bord de contrôle et de management fournis dans le composant sont accessibles à travers une interface web indépendante.

Serveur de gestion des logs – JHispter Console

JHipster Console fait également parti des composants fournis par JHipster et est aussi pratiquement clé-en-main. Il repose donc sur le langage Java et regroupe la pile ELK c'est à dire Elasticsearch, Logstash et Kibana, qui ensemble forme un triumvirat redoutable pour tout ce qui attrait à la gestion des logs.

Pour résumer simplement le fonctionnement de cette pile, on pourrait dire que Logstash fournit un flux d'entrée à ElasticSearch pour le stockage et la recherche, et Kibana accède aux données pour la visualisation, par exemple pour des tableaux de bord.

Tout comme le composant JHipster Registry, JHipster Console fourni une interface web indépendante permettant de manager ainsi que de visualiser les données extraites par le composant.

Serveur de traçage – Zipkin Tracing

Le composant que nous appelons Zipkin Tracing est un composant clé-en-main, s'appuyant intégralement sur l'outil Open Zipkin, directement téléchargeable sous la forme d'un Jar ou d'une image Docker, par conséquent, comme il a été spécifié lors de l'introduction de la section dédiée à la partie backend du système, il ne nécessite aucun développement.

Cet outil, en plus de tracer les requêtes, met à disposition une interface web indépendante, permettant une recherche approfondie des requêtes circulant dans le système ainsi qu'une visualisation détaillée de leur trajet et de leurs informations (en-têtes, temps de parcours, etc.).

Stockage des données

Comme on l'a vu sur le diagramme de composant ci-dessus, une seule base de données est requise pour cette première phase.

Base de données des localisations – BDD Localisation

La base de données dédié au microservice de localisation sera relationnelle et fonctionnera avec l'outil open source MySQL, même s'il est important de souligner, qu'au besoin, une offre professionnelle payante est disponible et qu'une migration vers cette version se fait naturellement.

Réutilisation du système existant

Toutes les systèmes frontend de la solution actuelle vont être réutiliser, y compris les middlewares composant l'architecture backend for frontend. Par exception, toutes les parties consacrées au système de recherche seront remplacées par les nouvelles applications web et mobile.

En ce qui concerne la partie backend de la solution actuelle qui est composée de 4 systèmes primaires, un seul ne sera pas réutilisé car il a été remplacé : le système de recherche.

Nous réutiliserons le système de gestion des commandes pour le management du panier et le processus de paiement ; le système de gestion des utilisateurs pour gérer les clients et les fournisseurs ; le système de gestion de l'inventaire qui ne sera donc plus connecter au client web de recherche actuel mais aux nouvelles applications web et mobile.

Toutes les communications inter-service, c'est à dire entre les microservices, comme extra-service, c'est à dire avec le système actuel, se feront avec la librairie open source Open Feign. Cet outil permet de simplifier et de minimiser l'écriture de code se rapportant au requêtage HTTP, tout en s'inscrivant dans la lignée des « Convention Over Configuration ».

Phase 2 : Migration complète du système

Comme son nom l'indique, cette seconde phase consiste à migrer l'intégralité du système vers un architecture microservices afin d'avoir, enfin, une solution unique et standardisée.

A l'issu de cette seconde phase, le projet qui nous a été transmis sera terminé, Foosus bénéficiera alors d'une solution professionnelle efficace permettant de concurrencer ses opposants dans la vente de proximité et pourra donc dédier son entière concentration à l'ajout de nouvelles fonctions de manière à ce que sa solution ne soit plus un frein mais un moteur dans son développement.

Comme on va le voir grâce au diagramme ci-dessous, seuls les composants du système actuel, qui n'ont pas été inclut lors de la première phase du projet, reste à intégrer au nouveau système.

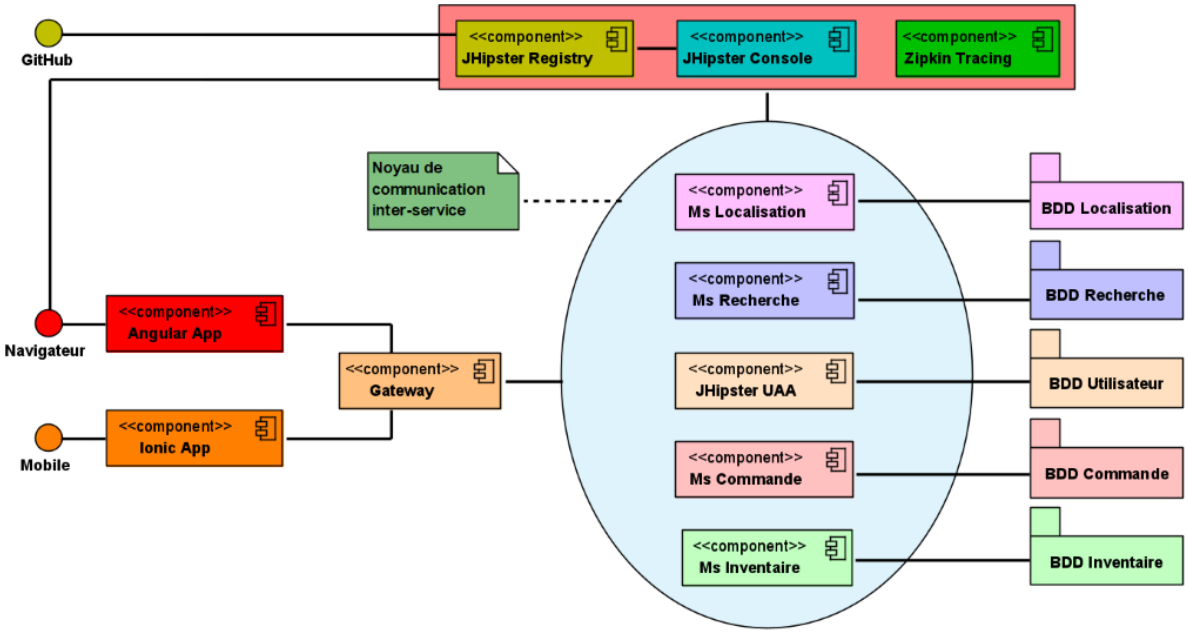

Diagramme de composant – Phase 2

Tout comme pour la première phase, nous proposons une présentation des composants de ce diagramme et ceci en 3 parties : le frontend, le backend et le stockage des données.

Frontend

Les composants frontend présents sur le diagramme de cette seconde phase sont les mêmes que ceux de la première. C'est tout à fait normal, il faudra simplement les faire évoluer de manière à prendre en compte l'amplification du contexte backend et ceci au fur et à mesure de l'implémentation des composants.

La partie frontend reposant sur une évolution progressive de ses composants, les applications web et mobile, nous n'allons pas nous y attarder davantage.

Backend

Comme on peut le remarquer sur le diagramme de composant présent ci-dessus, de nouveaux éléments ont été ajoutés à la partie backend, cependant il ne faut pas négliger le fait que des éléments présents depuis la première phase doivent être modifiés afin de prendre en compte le nouveau contexte backend, à savoir les composants Gateway et JHipster Console.

De plus, en fonction de la manière dont l'implémentation sera réalisée, il est également possible qu'une modification des deux microservices présents en phase 1, recherche et localisation, soit nécessaire pour mettre en place les fonctions de communication inter-service.

Enfin, il est important de préciser que ces communication inter-service se feront, comme pour la première phase, à l'aide de la librairie open source Open Feign de manière à minimiser ainsi qu'à simplifier considérablement le code qui en résulterait.

Microservice de gestion des utilisateurs et des autorisations – JHipster UAA

JHipster UAA (UAA signifiant « User Accounting and Authorizing ») fait également parti des composants fournis par JHipster et est aussi pratiquement clé-en-main. Il repose donc sur le langage Java ainsi que le framework Spring et a deux fonctions principales : la gestion des autorisations et la gestion des utilisateurs.

Microservice de gestion des commandes – Ms Commande

Un système de gestion des commandes est fondamental afin que la solution puisse fonctionner correctement. Un composant similaire est présent dans la solution initiale mais n'est, hélas, pas compatible avec l'architecture ainsi que la philosophie des microservices.

La pile technologique présente dans ce microservice est le langage Java, accompagné du framework Spring, ainsi que l'utilisation de l'outil JHipster.

Ce composant remplace donc « Foosus Order and Processing System » de la solution initiale.

Microservice de gestion des inventaires – Ms Inventaire

Un système de gestion des inventaires est également fondamental afin que la solution puisse fonctionner correctement. Un composant similaire est aussi présent dans la solution initiale mais n'est, hélas, pas compatible avec l'architecture ainsi que la philosophie des microservices.

La pile technologique présente dans ce microservice est le langage Java, accompagné du framework Spring, ainsi que l'utilisation de l'outil JHipster.

Ce composant remplace donc « Foosus Inventory System » présent dans la solution initiale.

Stockage des données

Pour cette seconde phase, nous avons requis la mise en place de 4 bases de données de plus.

Base de données des recherches – BDD Recherche

La base de données dédié au microservice de recherche sera relationnelle et fonctionnera avec l'outil open source MySQL, même si, encore une fois, il est important de souligner, qu'au besoin, une offre professionnelle payante est disponible et qu'une migration vers cette version se fait naturellement.

Base de données des utilisateurs – BDD Utilisateur

La base de données dédié au microservice utilisateur sera relationnelle et fonctionnera avec l'outil open source MySQL, même si, encore une fois, il est important de souligner, qu'au besoin, une offre professionnelle payante est disponible et qu'une migration vers cette version se fait naturellement.

Base de données des commandes – BDD Commande

La base de données dédié au microservice commande sera relationnelle et fonctionnera avec l'outil open source MySQL, même si, encore une fois, il est important de souligner, qu'au besoin, une offre professionnelle payante est disponible et qu'une migration vers cette version se fait naturellement.

Base de données des inventaires – BDD Inventaire

La base de données dédié au microservice inventaire sera relationnelle et fonctionnera avec l'outil open source MySQL, même si, encore une fois, il est important de souligner, qu'au besoin, une offre professionnelle payante est disponible et qu'une migration vers cette version se fait naturellement.

Principes stratégiques

Au cours de ce chapitre, nous allons énumérer les listes des différents principes stratégiques dont nous devons tenir compte et ceci de la manière la plus générale possible, afin de couvrir le maximum de facettes qui y sont liées.

Principes généraux

Il y a 4 principes généraux à prendre en compte :

- Décisions pilotées par le feed-back et l'apprentissage.
- Faire des choix qui soutiennent les objectifs long terme.
- Accepter le fait que les erreurs se produisent.
- Nous assurer que nous concevons l'architecture pour échouer vite et nous améliorer.

Principes business

Il y a 2 principes business à prendre en compte :

- Soutenir l'innovation et l'agilité du business grâce à l'extensibilité
- Soutenir la réputation de la marque grâce à la stabilité

Principes data

Il y a 5 principes data à prendre en compte :

- 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 partons 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.

Principes d'application

Il y a 4 principes d'application à prendre en compte :

- 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.

Principes technologiques

Il y a 6 principes technologiques à prendre en compte :

- Faire des choix ouverts et aisés à modifier.
- Les choix de construction vs achat doivent être raisonnés et toujours pris en compte.
- Les choix technologiques doivent s'aligner sur la capacité et la correspondance avec le business.
- Soutenir les sorties logiciel dès que possible.
- S'assurer que tous les composants de l'architecture sont conçus pour être faciles, à cataloguer et à ne pas perdre de vue.
- Privilégier la prévisibilité et la répétabilité plutôt que le non-déterminisme.

Plan de travail

Détails d'implémentation de la première phase

Le tableau, ci-dessous, vise à apporter un maximum de détails concernant l'implémentation de la première phase. On va y retrouver l'ordre chronologique des tâches à réaliser, les équipes devant les accomplir, les livrables qui y sont attendus ainsi que la durée de travail estimée.

Num	Tâche	Équipe	Livraison	Durée (jours)
1	Mise en place du composant JHipster Registry	Backend	Code source, image Docker et documentations.	4
2	Mise en place du composant Gateway	Backend	Code source, image Docker, cahier de recette et documentations.	10
3	Mise en place du composant JHipster Console	Backend	Code source, image Docker et documentations.	4
4	Mise en place du composant Zipkin Tracing	Backend	Image Docker et documentations.	1
5	Mise en place du composant Ms Localisation	Backend	Code source, image Docker, cahier de recette et documentations.	10
6	Mise en place de la base de données BDD Localisation	Data	Scripts SQL d'initialisation, image Docker, dataset et documentations	6
7	Mise en place du composant Ms Recherche	Backend	Code source, image Docker, cahier de recette et documentations.	10
8	Mise en place de l'application Angular App	Frontend	Code source, image Docker, cahier de recette et documentations.	20
9	Mise en place de l'application Ionic App	Frontend	Code source, image Docker, cahier de recette et documentations.	15
			Durée totale de la première phase :	80

Plan d'implémentation – Phase 1

Détails d'implémentation de la seconde phase

Le tableau, ci-dessous, vise à apporter un maximum de détails concernant l'implémentation de la seconde phase. On va y retrouver l'ordre chronologique des tâches à réaliser, les équipes devant les accomplir, les livrables qui y sont attendus ainsi que la durée de travail estimée.

Num	Tâche	Équipe	Livraison	Durée (jours)
1	Migration de la gestion des utilisateurs et des autorisations			35
1.1	Mise en place du composant JHipster UAA	Backend	Code source, image Docker, cahier de recette et documentations.	10
1.2	Mise en place de la base de données BDD Utilisateur	Data	Scripts SQL d'initialisation ainsi que de migration, image Docker, dataset et documentations.	8
1.3	Prise en compte dans les composants (gateway, etc.)	Backend	Code source, image Docker, cahier de recette et documentations.	5
1.4	Prise en compte dans l'application Angular App	Frontend	Code source, image Docker, cahier de recette et documentations.	6
1.5	Prise en compte dans l'application Ionic App	Frontend	Code source, image Docker, cahier de recette et documentations.	6
2	Migration de la gestion des inventaires			35
2.1	Mise en place du composant Ms Inventaire	Backend	Code source, image Docker, cahier de recette et documentations.	10
2.2	Mise en place de la base de données BDD Inventaire	Data	Scripts SQL d'initialisation ainsi que de migration, image Docker, dataset et documentations.	8
2.3	Prise en compte dans les composants (gateway, etc.)	Backend	Code source, image Docker, cahier de recette et documentations.	5
2.4	Prise en compte dans l'application Angular App	Frontend	Code source, image Docker, cahier de recette et documentations.	6
2.5	Prise en compte dans l'application Ionic App	Frontend	Code source, image Docker, cahier de recette et documentations.	6

3	**Migration de la gestion des commandes et du paiement**			**35**
3.1	Mise en place du composant Ms Commande	Backend	Code source, image Docker, cahier de recette et documentations.	10
3.2	Mise en place de la base de données BDD Commande	Data	Scripts SQL d'initialisation ainsi que de migration, image Docker, dataset et documentations.	8
3.3	Prise en compte dans les composants (gateway, etc.)	Backend	Code source, image Docker, cahier de recette et documentations.	5
3.4	Prise en compte dans l'application Angular App	Frontend	Code source, image Docker, cahier de recette et documentations.	6
3.5	Prise en compte dans l'application Ionic App	Frontend	Code source, image Docker, cahier de recette et documentations.	6
4	**Amélioration du système de recherche**			**25**
4.1	Mise en place de la base de données BDD Recherche	Data	Scripts SQL d'initialisation, image Docker, dataset et documentations.	6
4.2	Ajout des nouvelles fonctions dans Ms Recherche (ex: historique, etc.)	Backend	Code source, image Docker, cahier de recette et documentations.	7
4.3	Prise en compte dans l'application Angular App	Frontend	Code source, image Docker, cahier de recette et documentations.	6
4.4	Prise en compte dans l'application Ionic App	Frontend	Code source, image Docker, cahier de recette et documentations.	6
			Durée totale de la seconde phase :	**130**

Plan d'implémentation – Phase 2

Plan de communication

Le tableau, ci-dessous, vise à apporter un maximum de détails concernant le plan de communication adopté dans le cadre de ce projet. On va y retrouver les différentes réunions, leurs objets, le support de réalisation, la fréquence ainsi que le responsable organisation et managérial.

Ce tableau n'est, bien évidemment, pas exhaustif, d'autres réunions devront avoir lieu au cours de ce projet, il permet, de fait, de réunir celles qui sont primordiales afin d'assurer une réalisation sereine et pérenne.

Cible	Objet	Support	Fréquence	Responsable
Tout le personnel	Définition des objectifs journalier	Présentiel + récapitulatif sur tableau	Quotidien	Chef d'équipe concerné
Équipe(s) de developpement backend	Mise au point sur l'avancement	Présentiel / visioconférence + CR par courriel	Hebdomadaire	Responsable ingénieur
Équipe(s) de developpement frontend	Mise au point sur l'avancement	Présentiel / visioconférence + CR par courriel	Hebdomadaire	Responsable ingénieur
Équipe(s) de gestion des identités et des accès	Mise au point sur l'avancement	Présentiel / visioconférence + CR par courriel	Hebdomadaire	Responsable ingénieur
Équipe(s) de gestion des infrastructures	Mise au point sur l'avancement	Présentiel / visioconférence + CR par courriel	Hebdomadaire	Responsable infrastructure
Équipe(s) de gestion des données	Mise au point sur l'avancement	Présentiel / visioconférence + CR par courriel	Hebdomadaire	Responsable ingénieur
Équipe(s) produit	Mise au point sur l'avancement	Présentiel / visioconférence + CR par courriel	Hebdomadaire	Responsable produit
Équipe(s) expérience client	Rapport de l'expérience client et adaptation de la stratégie	Présentiel / visioconférence + CR par courriel	Bi-mensuel	Directeur produit
Équipe(s) d'analyse de données	Rapport sur l'analyse des données et adaptation de la stratégie	Présentiel / visioconférence + CR par courriel	Bi-mensuel	Directeur produit
Équipe(s) de satisfaction client	Rapport de satisfaction client et adaptation de la stratégie	Présentiel / visioconférence + CR par courriel	Bi-mensuel	Directeur produit
Équipe(s) de gestion financière	Contrôle apprfondie des coûts	Présentiel / visioconférence + CR par courriel	Bi-mensuel	Directeur financier
Responsables produits	Suivi des objectifs et de la ligne directrice	Présentiel / visioconférence + CR par courriel	Bi-mensuel	Directeur produit
Responsables techniques	Suivi des objectifs et de la ligne directrice	Présentiel / visioconférence + CR par courriel	Bi-mensuel	Directeur informatique
Directeurs techniques et non techniques	Suivi des objectifs et de la ligne directrice	Présentiel / visioconférence + CR par courriel	Mensuel	Directeur général

Plan de communication

Gouvernance et risques

Rôles et responsabilités

Le tableau, ci-dessous, vise à apporter un maximum d'informations sur la répartition des tâches dans le cadre de ce projet. Sur la gauche on va retrouver tous les champs d'expertise nécessaire à la bonne réalisation de ce projet, et en haut, l'intégralité des membres internes qui vont être concernés par le projet.

Il décrit, avec un code, pour chaque des tâches, les rôles et les responsabilités de chacun des acteurs

- En gris, n'est pas concerné par la tâche.
- En vert, doit être Informé des résultats de sortie.
- En jaune, doit Collaborer apporte les éléments, en entrée, nécessaires à la réalisation.
- En bleu, est un Acteur, doit réaliser la tâche.
- En rouge, est Responsable de la réalisation de la tâche.

	Équipes techniques	Équipe expérience client	Équipe satisfaction client	Équipe analyses des données	Équipe financière	Responsable ingénieur	Architecte logiciel	Responsable infrastructure	Responsable marketing	Responsable RH	Responsable Produit	Directeur marketing	Directeur RH	Directeur produit	Directeur financier	Directeur informatique	Directeur général
Définition des objectifs globaux							I				I			I		A	R
Définition précise du projet						I	A	I			A			C		R	I
Allocation des ressources humaines					I	C	I	C		A			R		C	I	I
Allocation des ressources matérielles					I	C	C	A							C	R	I
Allocation des ressources financières					A		I								R	I	I
Définition du budget					A		I								R	I	I
Veille au respect du budget					A		I								R	I	I
Estimation du temps de travail total						C	A	C							I	R	I
Veille au respect des délais						C	A	C								R	I
Conception de la solution	I					C	A	C								R	I
Définition des technologies de developpement	I					C	A	I								R	I
Définition des infrastructures de dev et de prod	I					I	A	C								R	I
Définition de la stratégie de déploiement et de scalabilité	I					I	C	A								R	I
Définition de la stratégie de sécurité informatique	I					A	C	A								R	I
Mise en oeuvre de la solution	A				I	R	I	I			I	I		I	I	I	I
Validation de la solution	A	C	C			A	A	I			A			I		R	I
Relation utilisateur		A	A	C							I	I		R			I
Veille aux performances du produit							I				A			R		I	I
Réalisation de l'étude de marché									A			R		I		I	I
Définition de la stratégie de communication		C	I	C					I			A		I	C	I	R
Met en oeuvre la stratégie de communication		I	I						A			R		I		I	I
Recherche de nouvelles fonctionnalités		C	C	C			I				A			R		I	I
Analyse des données				A										R			I

Matrice RACI du projet

Foosus

Analyse des risques

Le tableau, ci-dessous, vise à apporter un maximum de détails concernant les risques encourus dans le cadre de ce projet. On va y retrouver leur gravité, leur type, mettant en exergue, par ailleurs, la responsabilité mit en cause, leur point de départ ainsi que leurs conséquences directes ou indirectes et, enfin, les actions préventives permettant au mieux d'éviter le risque et au pire de le diminuer.

Gravité	Type	Cause	Conséquence	Préventif
1 - Mineure	Divers	Système non satisfaisant	Réajustements demandés Délais et coût allongés	Produire et valider un cahier des charges clair et précis
2 - Significative	Technique	Panne d'une ressource matériel	Délais et coût allongés	Prévoir une solution de remplacement pour les resources importantes
2 - Significative	Technique	Technologie mal/non maitrisée	Délais et coût allongés	Concerter les équipes techniques
3 - Grave	Humain	Départ d'un membre important	Délais et coût allongés	Prévoir une personne de remplacement pour les postes clés
3 - Grave	Technique	Perte de données	Délais et coût allongés	Faire des sauvegardes
3 - Grave	Divers	Locaux de travail non accessible	Délais et coût allongés Organisation perturbée	Prévoir un plan télétravail
4 - Critique	Technique	Besoin mal défini	Système non conforme	Produire et valider des documents définissant le besoin
4 - Critique	Technique	Périmètre flou autour du projet	Système non conforme	Produire et valider des documents définissant le périmètre
4 - Critique	Humain	Pandémie mondiale	Délais et coût allongés Probabilité de réaction en chaine	Prévoir un plan télétravail
4 - Critique	Gestion	Mauvaise organisation temporel	Délais et coût allongés	Faire valider le travail par un pair
4 - Critique	Gestion	Mauvaise estimation financière	Budget insuffisant	Faire valider le travail par un pair
5 - Catastrophique	Sécurité	Système final mal sécurisé	Fuite de données	Faire valider le travail par un pair et mener des tests spécifiques
5 - Catastrophique	Sécurité	Attaque informatique durant la réalisation	Fuite de données Paralysie des systèmes Délais et coût allongés	Faire intervenir l'équipe sécurité pour définir l'environnement de travail
5 - Catastrophique	Technique	Erreur de conception	Système non conforme	Faire valider le travail par un pair
5 - Catastrophique	Économique	Crise économique	Insolvabilité	Établir une stratégie avec un expert juridique
5 - Catastrophique	Juridique	Faillite d'une des parties prenantes	Insolvabilité	Établir une stratégie avec un expert juridique

Risques liés au projet

Hypothèses

De manière à ce que la prise en main du projet soit optimale et que nous saisissions tous les tenants et aboutissants, Foosus nous a transmis la liste des hypothèses qu'il préconise dans le cadre de la réalisation de ce projet. Libre à nous d'en tenir, strictement, compte.

Néanmoins, il n'en reste pas moins intéressant de connaitre leurs positions sur le sujet avant toute proposition de manière à avoir un second avis et ainsi pouvoir comparer nos réflexions.

Voici les 5 hypothèses de Foosus :

- 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 sentimentalement à la plateforme existante, les dirigeants devront éviter de prendre de faux raccourcis en intégrant un nouveau comportement dans le système déjà 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 fonctions.
    - 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

Critères d'acceptation et procédures

Critères d'acceptation

Le tableau, ci-dessous, vise à apporter un maximum de détails concernant les critères d'acceptation qui vont déterminer le succès du travail d'architecture dans le cadre de ce projet. On va y retrouver les métriques, leur méthode de mesure, leur valeur initiale et cible ainsi qu'une éventuelle remarque.

Métrique	Mesure	Valeur initiale	Valeur cible	Remarque
Adhésion journalière (utilisateur)	Requête SQL	NC	(+) 10%	La scalabilité du système utilisateur ne sera efficace qu'à partir de la phase 2.
Adhésion journalière (producteur)	Requête SQL	1,4/mois	4/mois	La scalabilité du système utilisateur ne sera efficace qu'à partir de la phase 2.
Délai de déploiement	Empirique	3,5 semaines	> 1 semaine	Métrique ne concernant que le nouveau système et ceci à partir de la phase 1.
Incident de production	Empirique	> 25/mois	> 1/mois	Métrique ne concernant que le nouveau système et ceci à partir de la phase 1.

Critères d'acceptation du projet

Procédures d'acceptation

Deux procédures majeures vont être instaurées dans le cadre de ce projet permettant d'approuver définitivement le travail effectué.

Vérification d'aptitude du bon fonctionnement

La VABF doit être bien préparée et planifiée par le chef de projet, qui doit piloter le client pour optimiser les résultats et les délais. Cette étape de validation précède la production qu'un quelconque composant.

Avant toute chose, il faut initialiser la recette client seulement une fois les préparatifs terminés. Cette étape préliminaire consiste à effectuer une recette interne, afin de garantir que le livrable est conforme à la demande et ainsi assurer que la recette peut s'opérer dans de bonne condition.

Le plus important est de vérifier la disponibilité des intervenants et leur capacité à accomplir cette tâche.

Il est nécessaire de découper la recette utilisateur en 3 temps :

- Le client fait son recueil d'anomalies et le communique
- Les corrections sont faites en d'itération et validées par le client (en continuant le recueil)
- Le client ne remonte plus de nouvelles anomalies, on termine les corrections à faire valider

Nous jugerons la durée de chacune des étapes afin de clore la recette à la date fixée avec le client. Il est important de communiquer ce planning de recette au client, le plus en amont possible afin que chaque partie puisse s'organiser.

Il sera également transmis un cahier de recette listant l'ensemble des tests à traiter. Pour cadrer la recette, un protocole de recette sera établi. Ce dernier doit expliciter le processus de remontée d'anomalies, il est impératif d'utiliser un gestionnaire de tickets pour garantir un suivi.

Il ne faut jamais perdre de vue le côté contractuel de cette phase. Et surtout la notion de pénalité qui peut jouer son rôle d'agitateur, la recette fonctionnelle est contractualisée dans un procès-verbal de recette.

La recette utilisateur, est comme son nom l'indique, avant tout une histoire humaine, de fait, à ce titre, le chef de projet doit intensifier sa communication auprès du client ainsi que de ses équipes et l'accompagner.

Dans un projet en interne, tel que celui-ci, la particularité étant que le client ainsi que le prestataire sont une seule entité, bien que pour certaines phases, des rôles puissent être attribués entre les collaborateurs.

Vérification du service régulier

Si la VABF se déroule correctement et est validée, on procède alors à la mise en service opérationnelle.

Une période de vérification de service régulier (VSR) commence donc par un déploiement, cette mise en production permet de valider le produit en conditions réelles.

À la différence des étapes précédentes, celle-ci se déroule en production avec des données réelles.

Procédures de changement de périmètre

Comme dans tout projet, il est probable qu'un changement de périmètre se produise au cours de la réalisation de celui-ci, ce qui reste, par ailleurs, de l'ordre de la normalité.

La responsabilité de ce changement de périmètre incombe à la partie prenante responsable de l'aspect concerné par ce changement. Effectivement, ce changement peut aussi bien résider dans la stratégie de mise en production initialement prévue, que dans l'ordre, initialement proposé, d'implémentation des composants, ou encore dans l'architecture en elle-même.

L'important réside dans la communication ainsi que l'accord des parties prenantes accompagnés du respect des processus spécifiés dans le framework d'architecture présent dans l'énoncé des travaux.

Bien évidemment, il faut veiller au bon respect des spécifications, critères de succès et de l'ensemble des procédures de conformités, comme ça a été le cas avec le système initialement proposé.

Approbations

Le présent document a été fait en 6 exemplaires, le …….. à ……..

Signature des parties prenantes précédée de la mention « Lu et approuvé ».

Monsieur Ayrton DE ABREU MIRANDA — Architecte logiciel

Signature

Monsieur Pete PARKER — Responsable ingénieur

Signature

Monsieur Jack HARKNER — Responsable infrastructure

Signature

Madame Natasha JARSON — Directeur informatique

Signature

Monsieur Daniel ANTHONY — Directeur produit

Signature

Madame Ash CALLUM — Directeur général

Signature