Table des matières

Objet de ce document

Ce document est une déclaration de travail d'architecture pour le projet d'évolution et de migration de l'intégralité des systèmes d'informations numériques de l'entreprise Foosus.

Cette déclaration permet d'apporter un maximum de précision concernant le projet, nous y approfondissons le contexte général, nous le décrivons de manière assez précise, nous y apportons une vue d'ensemble et nous spécifions la stratégie qui nous a été préconisé par Foosus, le commanditaire.

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.

Nous verrons aussi le rôle précis de chacune des parties prenantes ainsi que leurs niveaux de responsabilité au sein du projet. Pour ce faire, nous choisirons de nous appuyer sur des techniques de modélisation, de manière à ce que ce soit le plus claire possible, tel qu'un arbre de gouvernance et une matrice RACI.

Nous approfondirons, de manière complète, le cadre d'architecture qui a été choisi, les différentes phases qu'il comporte, le contenu de chaque phase et les étapes préconisées de réalisation ainsi que les livrables devant être, impérativement, produit pour une assimilation et une compréhension parfaite du projet.

Par la suite, nous étudierons, de manière très détaillée, le plan d'implémentation que nous proposons, d'une part, en nous attardant sur l'architecture actuelle afin d'en saisir la réelle substance et, d'autre part, en développant, aussi précisément que possible, l'architecture ainsi que les différents composants que nous préconisons, et ceci en apportant un grand nombre de justifications de façon à ce que tout lecteur puisse émettre un avis objectif ainsi que constructif vis-à-vis d'une solution concrète et complète.

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 verrons également 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 des risques encourus dans la réalisation, les critères d'acceptation propre à ce projet ainsi que les procédures de validation nécessaire à toute mise en service.

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.

Déclaration de travail d’architecture

Requête du projet et contexte

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.

Description du projet et périmètre

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.

Vue d’ensemble

L’entreprise veut permettre à ses équipes techniques de donner le meilleur d'elles-mêmes en créant une nouvelle plateforme qui pourra faire franchir le prochain million d'utilisateurs inscrits à sa base de clientèle.

Elle veut impulser des campagnes de marketing dans plusieurs grandes villes en étant sûrs que l’intégralité de sa plateforme restera utilisable et réactive, tout en offrant une expérience utilisateur de premier plan.

Alignement stratégique

Foosus ne peut pas abandonner les outils actuels pendant qu'elle en élabore de nouveau car cela impliquerait la mise hors service de la plateforme existante. Pour pouvoir continuer à accepter de nouvelles adhésions de fournisseurs et de consommateurs, nous devons, en outre, dissocier les nouvelles livraisons de l'architecture et de l'infrastructure existantes afin de limiter les interruptions de service.

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.

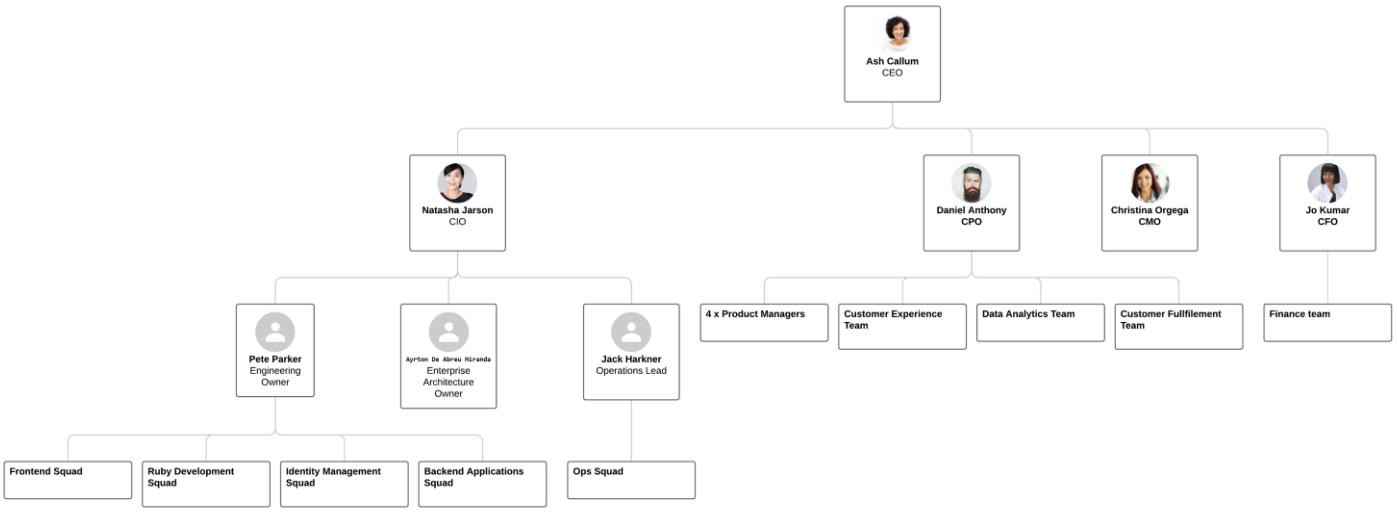
Les inscriptions constituent une métrique clé aux yeux de leurs investisseurs et ne peuvent être améliorées que par l'agilité nécessaire pour innover rapidement et ainsi avoir la capacité d'expérimenter avec des variantes d'offres produit existantes.

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.

Rôles et hiérarchisation

Structure de gouvernance

Le schéma, ci-dessous, vise à apporter des informations concernant la structure de gouvernance adopté au sein de l'entreprise. On va y retrouver les différentes parties prenantes accompagnées des équipes dont ils sont responsables. Il est à noter la mise en exergue de la hiérarchisation de l'ensemble du personnel.



Gouvernance actuelle de Foosus

On peut remarquer que le schéma, fourni par Foosus, n'est pas exhaustif, effectivement il manque la représentation de certaines équipes ainsi que de leur responsable et directeur. Dans certains cas, la représentation est bien présente mais de manière incomplète. On pense, par exemple, aux personnels responsables de la gestion des données, des ressources humaines ou encore du marketing.

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

Approche architecturale

Process d'architecture

La méthode de développement d'architecture TOGAF (ou ADM pour « Architecture Development Method ») décrit une méthodologie des meilleures pratiques pour le développement architectural. Néanmoins, toutes les phases ne sont pas également pertinentes pour chaque projet.

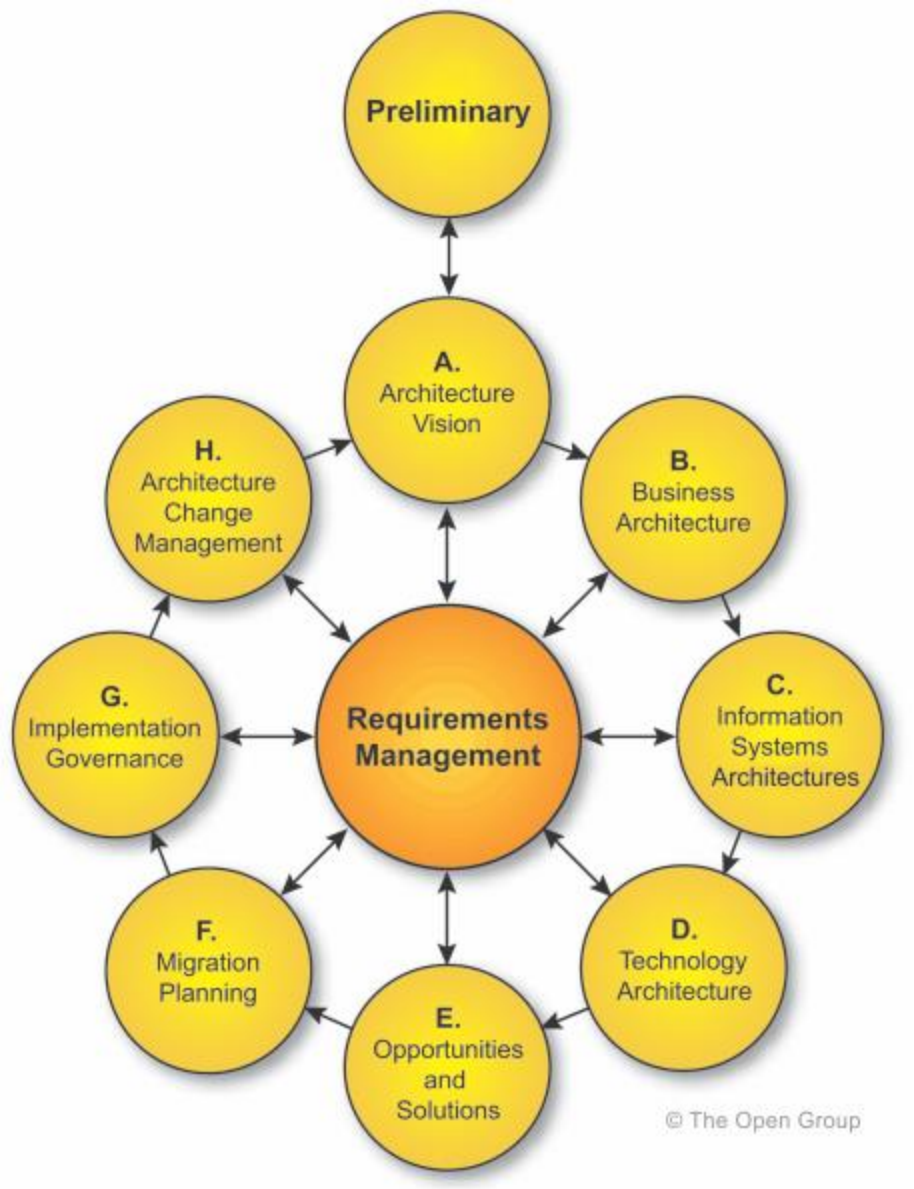


Illustration du processus TOGAF

Phase préliminaire

Cette phase décrit les activités de préparation et de lancement nécessaires pour répondre à la directive métier pour une nouvelle architecture d'entreprise, en incluant la définition d'un cadre d'architecture spécifique à l'organisation et la définition de principes.

Éléments en entrées

Voici, ci-dessous, une description, aussi exhaustive que possible, des éléments devant être présents en entrée afin de veiller à la bonne réalisation de la phase préliminaire.

Documents de référence externes à l'entreprise

- TOGAF
- Autre cadre d'architecture (si nécessaire)

Entrées non architecturales

- Stratégies du conseil d'administration et plans d'affaires du conseil d'administration, stratégie commerciale, stratégie informatique, principes commerciaux, objectifs commerciaux et moteurs commerciaux, lorsqu'ils sont préexistants
- Principaux cadres opérant dans l'entreprise ; par exemple, gestion de portefeuille et de projet
- Gouvernance et cadres juridiques, y compris la stratégie de gouvernance de l'architecture, lorsqu'ils sont préexistants
- Capacité d'architecture
- Accords de partenariat et contractuels

Entrées architecturales

- Modèle organisationnel pour l'architecture d'entreprise incluant :
    - Portée des organisations touchées
    - Évaluation de la maturité, lacunes et approche de résolution
    - Rôles et responsabilités des équipes d'architectures
    - Exigences budgétaires
    - Gouvernance et stratégie d'accompagnement
- Cadre d'architecture existant (le cas échéant) incluant :
    - Méthode d'architecture
    - Contenu de l'architecture
    - Outils configurés et déployés
    - Principes d'architecture
    - Dépôt d'architecture

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la phase préliminaire comme accomplie :

- Modèle organisationnel pour l'architecture d'entreprise, incluant :
  - Portée des organisations touchées
  - Évaluation de la maturité, lacunes et approche de résolution
  - Rôles et responsabilités des équipes d'architecture
  - Contraintes sur le travail d'architecture
  - Exigences budgétaires
  - Gouvernance et stratégie d'accompagnement
- Cadre d'architecture sur mesure, incluant :
  - Méthode d'architecture sur mesure
  - Contenu d'architecture sur mesure (livrables et artefacts)
  - Principes d'architecture
  - Outils configurés et déployés
- Référentiel d'architecture initial, rempli avec le contenu du cadre
- Référence aux principes commerciaux, aux objectifs commerciaux et aux moteurs commerciaux
- Demande de travaux d'architecture (facultatif)
- Cadre de gouvernance de l'architecture
- Catalogue de principes

Phase A : Vision de l’architecture

Cette phase initiale de la méthode de développement d'architecture (ADM) présente les informations sur la définition de la portée, l'identification des parties prenantes, la création de la vision de l'architecture et l'obtention des approbations.

Éléments en entrées

Les éléments devant être présents en entée afin de veiller à la bonne réalisation de la phase A se rapportant à la vision de l’architecture correspondent aux éléments de sorties de la phase préliminaire.

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la phase A, se rapportant à la vision architecturale, comme accomplie :

- Énoncé des travaux d'architecture approuvé incluant :
  - Description et portée du projet d'architecture
  - Vue d'ensemble de la vision de l'architecture
  - Plan et calendrier du projet d'architecture
- Énoncés affinés des principes commerciaux, des objectifs commerciaux et des moteurs commerciaux

- Principes d'architecture
- Évaluation de la capacité
- Cadre d'architecture sur mesure incluant :
    - Méthode d'architecture sur mesure
    - Contenu d'architecture sur mesure (livrables et artefacts)
    - Outils configurés et déployés
- Vision de l'architecture incluant :
    - Description du problème
    - Objectif de l'énoncé des travaux d'architecture
    - Récapitulatif des visions
    - Scénario commercial (facultatif)
    - Exigences des parties prenantes clés de haut niveau affinées
- Projet de document de définition d'architecture, incluant (si possible) :
    - Architecture d'entreprise de base, version 0.1
    - Architecture technologique de base, version 0.1
    - Architecture de données de base, version 0.1
    - Architecture d'application de base, version 0.1
    - Architecture métier cible, version 0.1
    - Architecture de technologie cible, version 0.1
    - Architecture de données cible, version 0.1
    - Architecture d'application cible, version 0.1
- Plan de communication
- Référentiel d'architecture
- Matrices :
    - Matrice de la carte des parties prenantes
- Diagrammes :
    - Diagramme de la chaîne de valeur
    - Diagramme de concept de solution

Phase B : Architecture business

Cette seconde phase permet de décrire le développement d'une architecture d'entreprise afin de soutenir une vision d'architecture convenue.

Éléments en entrées

Les éléments devant être présents en entée afin de veiller à la bonne réalisation de la phase B se rapportant à l'architecture business correspondent aux éléments de sorties des phases antérieures, en particulier de la précédente, à savoir la phase A qui concerne la vision architecturale.

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la phase B, se rapportant à l'architecture business, comme accomplie :

- Mises à jour des livrables de la phase A concernant la vision architecturale, notamment :
  - Énoncé des travaux d'architecture
  - Principes commerciaux, objectifs commerciaux et moteurs commerciaux validés
  - Principes d'architecture
- Document de définition d'architecture incluant :
  - Architecture business de base, version 1.0 (détaillée)
  - Architecture business cible, version 1.0 (détaillée), comprenant :
    - Structure organisationnelle - identifier les sites commerciaux et les relier aux unités organisationnelles
    - Buts et objectifs commerciaux - pour l'entreprise et chaque unité organisationnelle
    - Fonctions commerciales - une étape détaillée et récursive impliquant la décomposition successive des principaux domaines fonctionnels en sous-fonctions
    - Services aux entreprises - les services que l'entreprise et chaque unité d'entreprise fournissent à ses clients, à la fois en interne et en externe
    - Processus commerciaux, y compris les mesures et les livrables
    - Rôles commerciaux, y compris le développement et la modification des exigences en matière de compétences
    - Modèle de données d'entreprise
    - Corrélation de l'organisation et des fonctions - reliez les fonctions commerciales aux unités organisationnelles sous la forme d'un rapport matriciel
  - Correspondance entre l'architecture business et la vision des principales préoccupations des parties prenantes
- Document des spécifications des exigences d'architecture, y compris les exigences d'architecture d'entreprise telles que :
  - Résultats de l'analyse des écarts
  - Exigences techniques - identification, catégorisation et hiérarchisation des implications pour le travail dans les domaines d'architecture restants ; par exemple, par une matrice de dépendance / priorité (par exemple, guider le compromis entre la vitesse de traitement des transactions et la sécurité) ; lister les modèles spécifiques qui devraient être produits (par exemple, exprimés sous forme de primitives du framework Zachman)
  - Exigences commerciales mises à jour
- Composantes de l'architecture d'entreprise d'une feuille de route de l'architecture
- Catalogues :
  - Catalogue des organisations / acteurs
  - Catalogue Pilotes / Buts / Objectifs
  - Catalogue de rôles
  - Catalogue des services / fonctions aux entreprises
  - Catalogue des emplacements
  - Catalogue des processus / événements / contrôles / produits
  - Catalogue de contrats / mesures

- Matrices :
    - Matrice d'interaction commerciale
    - Matrice acteur / rôle
- Diagrammes :
    - Diagramme d'empreinte d'entreprise
    - Diagramme de service / information d'entreprise
    - Diagramme de décomposition fonctionnelle
    - Diagramme du cycle de vie du produit
    - Diagramme des buts / objectifs / services
    - Diagramme de cas d'utilisation
    - Diagramme de décomposition de l'organisation
    - Diagramme de flux de processus
    - Diagramme d'événement

Phase C : Architecture des systèmes d'information

Cette phase permet de décrire les architectures de données et applicative dans un système d'information.

Éléments en entrées

Les éléments devant être présents en entée afin de veiller à la bonne réalisation de la phase C se rapportant à l'architecture des systèmes d'information correspondent aux éléments de sorties des phases antérieures.

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la phase C, se rapportant aux architectures des systèmes d'information, comme accomplie :

- Enrichissement et mise à jour des livrables de la phase A concernant la vision architecturale notamment :
    - L'énoncé des travaux d'architecture
- Enrichissement et mise à jour du document de définition d'architecture, incluant :
    - Architecture de données de base, version 1.0
    - Architecture de données cible, version 1.0
    - Architecture applicative de base, version 1.0
    - Architecture applicative cible, version 1.0
    - Correspondance entre l'architecture des données et la vision des principales préoccupations des parties prenantes
    - Correspondance entre l'architecture applicative et la vision des principales préoccupations des parties prenantes

- Document de spécification des exigences d'architecture, incluant les exigences d'architecture des systèmes d'information telles que :
    - Résultats de l'analyse des écarts
    - Exigences techniques pertinentes qui s'appliqueront à l'évolution du cycle de développement de l'architecture
    - Contraintes de l'architecture technologique sur le point d'être conçue
    - Exigences opérationnelles mises à jour
- Roadmap de l'architecture applicative avec les composants du système d'information

Phase D : Architecture technologique

Cette phase permet de décrire le développement de l'architecture technologique.

Éléments en entrées

Les éléments devant être présents en entée afin de veiller à la bonne réalisation de la phase D se rapportant à l'architecture technologique correspondent aux éléments de sorties des phases antérieures.

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la phase D, se rapportant à l'architecture technologique, comme accomplie :

- Enrichissement et mise à jour des livrables de la phase A concernant la vision architecturale notamment :
    - Énoncé des travaux d'architecture
    - Principes technologiques validés ou nouveaux principes technologiques
- Enrichissement et mise à jour du document de définition d'architecture, incluant:
    - Architecture technologique cible, version 1.0 (détaillée), comprenant:
        - Composants technologiques et leurs relations avec les systèmes d'information
        - Plates-formes technologiques et leur décomposition, montrant les combinaisons de technologies nécessaires pour réaliser une «pile» technologique particulière
        - Environnements et emplacements - un regroupement de la technologie requise dans des environnements informatiques (par exemple, développement, production)
        - Charge de traitement attendue et répartition de la charge entre les composants technologiques
        - Communications physiques (réseau)
        - Spécifications matérielles et réseau
    - Architecture technologique de base, version 1.0 (détaillée)
    - Correspondance entre l'architecture technologique et la vision des principales préoccupations des parties prenantes
- Enrichissement et mise à jour du document de spécification des exigences d'architecture, incluant les exigences d'architecture technologique telles que:

    - Résultats de l'analyse des écarts
    - Exigences issues des phases B et C
    - Exigences technologiques mises à jour
- Roadmap de l'architecture applicative avec les composants du système d'information
- Catalogues :
    - Catalogue des normes technologiques
    - Catalogue du portefeuille technologique
- Matrices :
    - Matrice application / technologie
- Diagrammes :
    - Diagramme des environnements et des emplacements
    - Diagramme de décomposition de la plate-forme
    - Diagramme de traitement
    - Diagramme du matériels en réseau
    - Diagramme d'ingénierie des communications

Phase E : Opportunités et solutions

Cette phase permet de décrire les éléments de livraison (projets, programmes et portefeuilles) qui fournissent efficacement l'architecture cible identifiée dans les phases précédentes.

Éléments en entrées

Les éléments devant être présents en entée afin de veiller à la bonne réalisation de la phase E se rapportant aux opportunités et solutions correspondent aux éléments de sorties des phases antérieures.

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la phase E, se rapportant aux opportunités et solutions, comme accomplie :

- Enrichissement et mise à jour des livrables de la phase A concernant la vision architecturale notamment :
    - Énoncé des travaux d'architecture
    - Vision de l'architecture, comprenant la définition des types et des degrés d'interopérabilité
- Enrichissement et mise à jour du document de définition d'architecture, incluant:
    - Baseline Business Architecture, version 1.0 mise à jour si nécessaire
    - Target Business Architecture, version 1.0 mise à jour si nécessaire
    - Architecture des données de base, version 1.0 mise à jour si nécessaire
    - Target Data Architecture, version 1.0 mise à jour si nécessaire
    - Architecture d'application de base, version 1.0 mise à jour si nécessaire
    - Architecture d'application cible, version 1.0 mise à jour si nécessaire
    - Architecture technologique de base, version 1.0 mise à jour si nécessaire

    - Architecture de la technologie cible, version 1.0 mise à jour si nécessaire
    - Architecture de transition, nombre et portée si nécessaire
    - Correspondance entre les architectures et la vision des principales préoccupations des parties prenantes
- Document de spécification des exigences d'architecture, incluant :
    - Évaluation consolidée des lacunes, des solutions et des dépendances
- Évaluations des capacités, comprenant :
    - Évaluation des capacités commerciales
    - Évaluation des capacités informatiques
- Roadmap de l'architecture, comprenant :
    - Portefeuille de lots de travaux :
        - Description du lot de travaux (nom, description, objectifs)
        - Exigences fonctionnelles
        - Dépendances
        - Relation à l'opportunité
        - Relation avec le document de définition d'architecture et la spécification des exigences d'architecture
        - Relation avec les incréments de capacité
        - Valeur commerciale
        - Évaluation des facteurs de mise en œuvre et matrice de déduction
        - Impact
    - Identification des architectures de transition, le cas échéant, comprenant :
        - Relation avec le document de définition d'architecture
    - Recommandations de mise en œuvre :
    - Critères de mesure de l'efficacité
    - Risques et problèmes
    - Blocs de construction de solutions
- Plan de mise en œuvre et de migration, version 0.1, incluant :
    - Stratégie de mise en œuvre et de migration
- Diagrammes :
    - Diagramme de contexte de projet
    - Diagramme des avantages

Phase F : Planning de migration

Cette phase traite de la planification de la migration; c'est-à-dire comment passer des architectures de base aux architectures cibles en finalisant un plan détaillé de mise en œuvre et de migration.

Éléments en entrées

Les éléments devant être présents en entée afin de veiller à la bonne réalisation de la phase F se rapportant à la planification de la migration correspondent aux éléments de sorties des phases antérieures.

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la phase F, se rapportant à la planification de la migration, comme accomplie :

- Plan de mise en œuvre et de migration, version 1.0, incluant :
  - Stratégie de mise en œuvre et de migration
  - Répartition du projet et du portefeuille de la mise en œuvre :
    - Attribution des lots de travaux au projet et au portefeuille
    - Capacités fournies par les projets
    - Relation avec l'architecture cible et les architectures de transition
    - Jalons et calendrier
    - Structure de répartition du travail
  - Chartes de projet (facultatif) :
    - Dossiers de travail associés
    - Valeur commerciale
    - Risque, problèmes, hypothèses, dépendances
    - Besoins en ressources et coûts
    - Avantages de la migration
    - Estimation des coûts des options de migration
- Document de définition d'architecture finalisé, incluant :
  - Architectures de transition finalisées
- Spécification des exigences d'architecture finalisée
- Feuille de route de l'architecture finalisée
- Blocs de construction d'architecture réutilisables
- Demandes de travaux d'architecture pour une nouvelle itération du cycle ADM
- Modèle de gouvernance mis en œuvre
- Demandes de changement pour la capacité d'architecture découlant des leçons apprises

Phase G : Gouvernance de l'implémentation

Cette phase permet d'avoir une vision globale entre l'architecture et son implémentation.

Éléments en entrées

Les éléments devant être présents en entée afin de veiller à la bonne réalisation de la phase G se rapportant à la gouvernance de l'implémentation correspondent aux éléments de sorties des phases antérieures.

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la phase G, se rapportant à la gouvernance de l'implémentation, comme accomplie :

- Contrat d'architecture (signé)

- Évaluations de la conformité
- Demandes de changement
- Solutions conformes à l'architecture déployées, notamment :
    - Le système implémenté conforme à l'architecture
    - Référentiel d'architecture rempli
    - Recommandations et dispenses de conformité de l'architecture
    - Recommandations sur les exigences de prestation de services
    - Recommandations sur les mesures de performance
    - Accords de niveau de service
    - Vision de l'architecture, mise à jour après la mise en œuvre
    - Document de définition d'architecture, mis à jour après la mise en œuvre
    - Modèles d'exploitation commerciaux et informatiques pour la solution mise en œuvre

Phase H : Management du changement d'architecture

Cette phase permet la mise en place de procédures de gestion des modifications de la nouvelle architecture.

Éléments en entrées

Les éléments devant être présents en entée afin de veiller à la bonne réalisation de la phase H, se rapportant à la gestion du changement d'architecture, correspondent aux éléments de sorties des phases antérieures.

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la phase H, se rapportant au management du changement d'architecture, comme accomplie :

- Mises à jour de l'architecture (pour les modifications de maintenance)
- Modifications du cadre et des principes de l'architecture (pour les modifications de maintenance)
- Nouvelle demande de travaux d'architecture, pour passer à un autre cycle (pour les changements majeurs)
- Énoncé des travaux d'architecture, mis à jour si nécessaire
- Contrat d'architecture, mis à jour si nécessaire
- Évaluations de la conformité, mises à jour si nécessaire

Management des conditions requises

Cette section permet d'examiner le processus de gestion des exigences d'architecture dans l'ADM.

Éléments en entrées

Voici une description, aussi exhaustive que possible, des éléments devant être présents en entrée afin de veiller à la bonne réalisation de cette section se rapportant au management des conditions requises.

- Un référentiel d'architecture rempli
- Modèle organisationnel pour l'architecture d'entreprise comprenant :
    - Portée des organisations touchées
    - Évaluation de la maturité, lacunes et approche de résolution
    - Rôles et responsabilités des équipes d'architecture
    - Contraintes sur le travail d'architecture
    - Exigences budgétaires
    - Gouvernance et stratégie d'accompagnement
- Cadre d'architecture sur mesure, comprenant :
    - Méthode d'architecture sur mesure
    - Contenu d'architecture sur mesure (livrables et artefacts)
    - Outils configurés et déployés
- Énoncé des travaux d'architecture
- Vision de l'architecture
- Exigences d'architecture, remplissant une spécification d'exigences d'architecture
- Analyse d'impact des exigences

Etant donné le rôle de cette section, il est évident que son contenu est amené à évoluer ainsi qu'à s'enrichir en fonction de l'accomplissement des différentes phases.

Éléments en sorties

Voici une description, aussi exhaustive que possible, des éléments devant être présents en sortie afin de déclarer la section de management des conditions requises comme accomplie :

- Analyse d'impact des exigences
- Spécification des exigences d'architecture, si nécessaire

Le référentiel des exigences sera mis à jour dans le cadre de la phase de gestion des exigences et devrait contenir toutes les informations sur les exigences.

Lorsque de nouvelles exigences surviennent ou que des exigences existantes sont modifiées, une étude d'impact des exigences est produite, qui identifie les phases de l'ADM qui doivent être revues pour tenir compte des changements. La déclaration passe par diverses itérations jusqu'à la version finale, qui comprend toutes les implications des exigences (par exemple, les coûts, les délais et les mesures commerciales) sur le développement de l'architecture. Une fois que les exigences du cycle ADM actuel ont été finalisées, la spécification des exigences d'architecture doit être mise à jour.

Foosus

Contenu de l’architecture

Le cadre de contenu d’architecture TOGAF (ou ACF pour « Architecture Content Framework ») fournit une catégorisation des meilleures pratiques pour le contenu de l’architecture. Néanmoins, tous les éléments ne sont pas également pertinents pour chaque projet. Le tableau ci-dessous décrit les zones de contenu pertinentes pour ce projet spécifique.

Principes, vision et conditions requises

Principes

Les principes d'architecture définissent les règles et directives générales sous-jacentes pour l'utilisation et le déploiement de toutes les ressources et actifs informatiques dans l'entreprise. Ils reflètent un niveau de consensus entre les différents éléments de l'entreprise et constituent la base des futures décisions informatiques.

Chaque principe d'architecture doit être clairement lié aux objectifs commerciaux et aux principaux moteurs de l'architecture.

Les principes d'architecture sont généralement développés par les architectes de l'entreprise, en collaboration avec les principales parties prenantes, et sont approuvés par le Conseil d'architecture.

Les principes d'architecture seront éclairés par des principes au niveau de l'entreprise, s'ils existent.

Les principes d'architecture doivent être clairement traçables et clairement articulés pour guider la prise de décision. Ils sont choisis de manière à assurer l'alignement de l'architecture et de la mise en œuvre de l'architecture cible avec les stratégies et visions commerciales.

Plus précisément, le développement des principes d'architecture est généralement influencé par les éléments suivants:

- Mission et plans d'entreprise: la mission, les plans et l'infrastructure organisationnelle de l'entreprise.
- Initiatives stratégiques de l'entreprise: les caractéristiques de l'entreprise - ses forces, ses faiblesses, ses opportunités et ses menaces - et ses initiatives actuelles à l'échelle de l'entreprise (telles que l'amélioration des processus et la gestion de la qualité).
- Contraintes externes: facteurs de marché (impératifs des délais de mise sur le marché, attentes des clients, etc.); législation existante et potentielle.
- Systèmes et technologies actuels: l'ensemble des ressources d'information déployées au sein de l'entreprise, y compris la documentation des systèmes, les inventaires des équipements, les diagrammes de configuration du réseau, les politiques et les procédures.
- Tendances émergentes de l'industrie: prévisions sur les facteurs économiques, politiques, techniques et de marché qui influencent l'environnement de l'entreprise.

Vision

Les étapes permettant d'établir la vision de l'architecture sont les suivantes:

- Établir le projet d'architecture
- Identifier les parties prenantes, les préoccupations et les exigences opérationnelles
- Confirmer et élaborer les objectifs commerciaux, les moteurs commerciaux et les contraintes
- Évaluer les capacités de l'entreprise
- Évaluer la préparation à la transformation de l'entreprise
- Définir la portée
- Confirmer et élaborer les principes d'architecture, y compris les principes commerciaux
- Développer la vision de l'architecture
- Définir les propositions de valeur d'architecture cible et les KPI
- Identifier les risques liés à la transformation de l'entreprise et les activités d'atténuation
- Élaborer un énoncé des travaux d'architecture; Approbation sécurisée

Conditions requises

1. Phase de l'ADM :

Identifier / documenter les exigences - utiliser des scénarios commerciaux ou une technique analogue

2. Gestion des exigences

Exigences de base :

a. Déterminer les priorités découlant de la phase actuelle d'ADM
b. Confirmer l'adhésion des parties prenantes aux priorités résultantes
c. Enregistrer les priorités des exigences et les placer dans le référentiel des exigences

3. Gestion des exigences

Monitorer les exigences de base

4. Phase de l'ADM

Identifier les exigences modifiées :

a. Supprimer ou réévaluer les priorités
b. Ajouter des exigences et réévaluer les priorités
c. Modifier les exigences existantes

5. Gestion des exigences

Identifier les exigences modifiées et consigner les priorités :

a. Identifier les exigences modifiées et s'assurer que les exigences sont hiérarchisées par le ou les architectes responsables de la phase actuelle et par les parties prenantes concernées
b. Enregistrer de nouvelles priorités
c. S'assurer que tous les conflits sont identifiés et gérés tout au long des phases jusqu'à une conclusion et une hiérarchisation réussies
d. Générer une étude d'impact des exigences pour diriger l'équipe d'architecture

Remarques :
Pour s'assurer que les exigences sont correctement évaluées et classées par ordre de priorité, ce processus doit diriger les phases de l'ADM et enregistrer les décisions liées aux exigences.
La phase de gestion des exigences doit déterminer la satisfaction des parties prenantes à l'égard des décisions. En cas de mécontentement, la phase reste responsable pour garantir la résolution des problèmes et déterminer les prochaines étapes.

6. Phase de l'ADM

a. Évaluer l'impact des exigences modifiées sur la phase actuelle (active)
b. Évaluer l'impact des exigences modifiées sur les phases précédentes
c. Déterminer s'il faut mettre en œuvre le changement ou s'en remettre au cycle ADM ultérieur; si la décision est de mettre en œuvre, évaluer le calendrier de mise en œuvre de la gestion du changement
d. Étude d'impact sur les exigences du problème, version n + 1

7. Phase de l'ADM

Mettre en œuvre les exigences découlant de la phase H.

L'architecture peut être modifiée tout au long de son cycle de vie par la phase de gestion du changement d'architecture (phase H). Le processus de gestion des exigences garantit que les exigences nouvelles ou changeantes dérivées de la phase H sont gérées en conséquence.

8. Gestion des exigences

Mettre à jour le référentiel des exigences avec des informations relatives aux changements demandés, y compris les visions des parties prenantes concernées

9. Phase de l'ADM

Mettre en œuvre le changement dans la phase actuelle

10. Phase de l'ADM

Évaluer et réviser l'analyse des écarts pour les phases passées.

L'analyse des écarts dans les phases B à D d'ADM identifie les écarts entre les architectures de base et cible. Certains types d'écart peuvent donner lieu à des exigences d'écart.

L'ADM décrit deux types d'écart:

a. Quelque chose qui est présent dans la ligne de base, mais pas dans la cible (c'est-à-dire éliminé - par accident ou par conception)
b. Quelque chose qui n'est pas dans la ligne de base, mais présent dans la cible (c'est-à-dire nouveau)

Une «exigence d'écart» est tout ce qui a été éliminé par accident et nécessite donc une modification de l'architecture cible.

Si l'analyse des écarts génère des exigences relatives aux écarts, cette étape garantira qu'elles soient traitées, documentées et enregistrées dans le référentiel des exigences, et que l'architecture cible est révisée en conséquence.

Architecture Business

Les étapes de la phase se rapportant à l'architecture business sont les suivantes:

- Sélectionner des modèles de référence, des points de vue et des outils
- Élaborer une description de l'architecture business de base
- Élaborer une description de l'architecture business cible
- Effectuer une analyse des écarts
- Définir la roadmap des composants envisagés
- Résoudre les impacts en rapport avec le domaine architectural
- Effectuer un examen formel des parties prenantes
- Finaliser l'architecture business
- Documenter au travers d'un document de définition d'architecture

Architecture des systèmes d'information — Données

Les étapes de la phase se rapportant à l'architecture des données sont les suivantes:

- Sélectionner des modèles de référence, des points de vue et des outils
- Élaborer une description de l'architecture des données de base
- Élaborer une description de l'architecture des données cibles
- Effectuer une analyse des écarts
- Définir la roadmap des composants envisagés
- Résoudre les impacts en rapport avec le domaine architectural
- Effectuer un examen formel des parties prenantes

- Finaliser l'architecture des données
- Documenter au travers du document de définition d'architecture

Architecture des systèmes d'information — Applications

Les étapes de la phase se rapportant à l'architecture applicative sont les suivantes:

- Sélectionner des modèles de référence, des points de vue et des outils
- Élaborer une description de l'architecture applicative de base
- Élaborer une description de l'architecture applicative cible
- Effectuer une analyse des écarts
- Définir la roadmap des composants envisagés
- Résoudre les impacts en rapport avec le domaine architectural
- Effectuer un examen formel des parties prenantes
- Finaliser l'architecture applicative
- Documenter au travers du document de définition d'architecture

Architecture technologique

Les étapes de la phase se rapportant à l'architecture technologique sont les suivantes:

- Sélectionner des modèles de référence, des points de vue et des outils
- Élaborer une description de l'architecture technologique de base
- Élaborer une description de l'architecture technologique cible
- Effectuer une analyse des écarts
- Définir la roadmap des composants envisagés
- Résoudre les impacts en rapport avec le domaine architectural
- Effectuer un examen formel des parties prenantes
- Finaliser l'architecture technologique
- Documenter au travers du document de définition d'architecture

Mise en œuvre de l'architecture

Les étapes de la phase se rapportant à la mise en œuvre de l'architecture sont les suivantes:

- Confirmer la portée et les priorités du déploiement avec la gestion du développement
- Identifier les ressources et les compétences de déploiement
- Guide de développement du déploiement de solutions
- Effectuer des examens de conformité de l'architecture d'entreprise
- Mettre en œuvre les opérations business et informatiques
- Effectuer un examen post-implémentation et clôturer l'implémentation

Technologies et normes de l'industrie

Au cours de cette partie nous allons étudier l'état architectural de base ainsi que l'état cible, puis la place de l'itération dans l'ADM et, enfin, les bonnes pratiques de développement à adopter dans ce projet.

État architectural de base

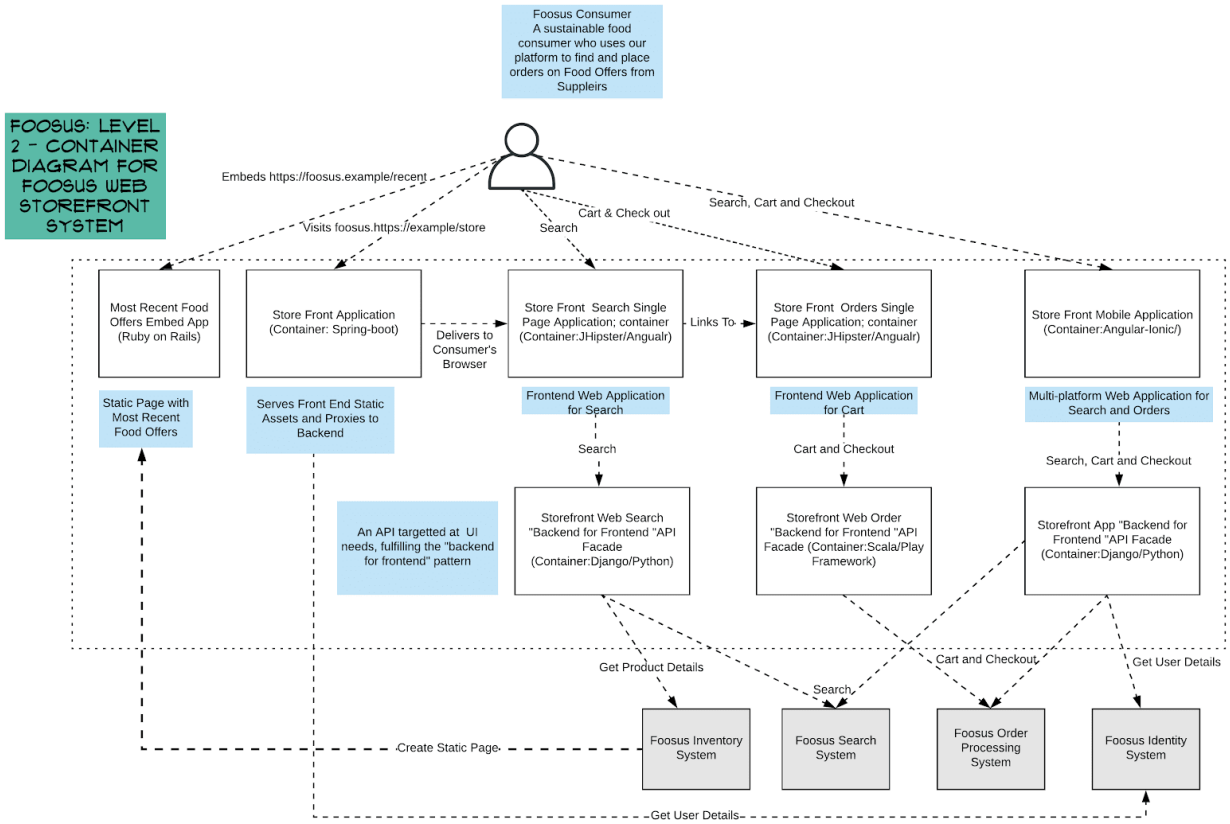


Diagramme de niveau conteneur

Le diagramme de niveau conteneur ci-dessus nous permet d'avoir une description de haut niveau du système informatique dont est actuellement équipé Foosus. Il est à noter qu'il n'est malheureusement pas exhaustif.

La première chose que l'on peut remarquer est la pluralité des technologies présentes au sein de ce système. Cette approche hétéroclite dans le choix des technologies rend un système difficilement compréhensible en plus de complexifier vivement toute stratégie de maintenabilité ainsi que d'augmenter les coûts.

Dans un second temps, nous pouvons nous attarder sur le pattern architectural mise en place, il s'agit du pattern « Backend For Frontend » (BFF). Pour faire simple, ce pattern consiste à inclure un intermédiaire entre chaque client frontend et la partie backend, et ceci pour plusieurs raisons, notamment :

- La diminution de la logique métier coté frontend et parfois backend.
- L'optimisation des requêtes en fonction des besoins du frontend.
- L'indépendance du backend vis-à-vis des interfaces.

Ces avantages sont toutefois lourds de conséquence, effectivement c'est un pattern qui a parait très bien sur le papier mais qui présente néanmoins des inconvénients significatifs, tels que :

- Des services additionnels à déployer et à maintenir entrainant une augmentation des coûts.
- Des services additionnels qui sont, bien souvent, développés et maintenus par les équipes frontend, qui deviennent, de fait, fullstack alors qu'elles ne sont, théoriquement, pas légitimes pour travailler dans un environnement backend qui inclue des problématiques bien différentes.
- L'ajout d'une couche d'abstraction supplémentaire qui complexifie davantage le système.

Foosus nous a spécifié que leurs équipes techniques passaient la majeure partie de leur temps à éteindre des incendies tant le système est désuet et mal optimisé, c'est pourquoi, il ne souhaite pas qu'on le fasse évoluer, il doit être, impérativement, passé en mode maintenance, jusqu'à son remplacement total.

Pour approfondir davantage la section dédiée aux technologies, dans le système nous allons retrouver :

- Une application embarquée, implémentée sous Ruby on Rails, donnant un accès direct aux offres alimentaires les plus récentes grâce à son interaction avec le système backend de gestion de l'inventaire.
- Une « passerelle » frontend, implémenté en Java avec l'utilisation du framework Spring, ayant trois fonctions principales :
    - L'interaction directe avec le système backend dédié à la gestion des utilisateurs.
    - L'interaction avec le client SPA dédié à la fonction de recherche, généré à l'aide de JHiptser, qui permet la création d'application implémenté avec le framework Angular pour la partie frontend et le framework Spring pour la partie backend. Ce client communique avec :
        - Un intermédiaire servant de façade à la partie backend, comme expliqué lors de la partie relatant le pattern BFF, développé en Python avec le framework Django. Cet intermédiaire communique donc avec :
            - Le système backend de gestion de l'inventaire.
            - Le système backend de recherche.
    - L'interaction avec le client SPA dédié aux fonctions de panier et de paiement, aussi généré à l'aide de JHiptser et communiquant avec :
        - Un intermédiaire, servant de façade à la partie backend, développé en Scala avec Play Framework. Cet intermédiaire communique donc avec :
            - Le système backend de gestion des commandes.
- Une application mobile cross-platform proposant les principales fonctions du système : recherche, panier ainsi que paiement. Cette application est implémentée à l'aide de Ionic, framework qui se base sur l'utilisation de code Angular, dans notre cas probablement celui des clients évoqués ci-dessus, et communiquant avec :

    - Un intermédiaire servant de façade à la partie backend, développé en Python avec le framework Django. Cet intermédiaire communique donc avec :
        - Le système backend de recherche.
        - Le système backend de gestion des commandes.
        - Le système backend de gestion des utilisateurs.
- L'ensemble des composants du système backend ont été évoqués au cours de cet approfondissement dédié aux technologies, aucun langage de programmation n'est spécifié concernant cette section, cependant nous sommes libres de penser que la partie backend a été implémentée intégralement à l'aide du langage Java et de son framework Spring, effectivement cette combinaison est la plus utilisée dans le système, en plus d'être la spécialité des développeurs.

État architectural cible

Le choix des technologies qui vont être proposées dans le cadre de la réalisation de ce projet a été influencé par le souhait de standardisation exprimé par Foosus. Effectivement, les langages et autres frameworks choisis sont déjà présents dans le système actuel garantissant que les équipes techniques ont les compétences pour les utiliser, cependant, comme on va le voir, le style architectural a évolué.

Nous allons rappeler, de manière simplifiée, les exigences de Foosus pour ce projet :

- Géolocalisation des offres alimentaires
- Prise en charge des régions (pays, ville, etc.)
- Amplification de la sécurité des systèmes
- Limitation des interruptions de services
- Prise en charge des appareils fixes et mobiles
- Prise en charge de plusieurs types d'utilisateurs
- Livraison à intervalle régulier
- Développement d'un système évolutif

Deux architectures cibles vont être proposées :

- Une première architecture que l'on peut qualifier de phase 1, 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 phase 2, 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.

Phase 1 : Mise en place d'une architecture microservices ainsi que des composants fondamentaux

Comme on va le voir sur le diagramme de composant ci-dessous, il a été choisi de mettre en place une architecture microservices, qui, en plus de correspondre en tous points aux exigences, a été conseillé par Foosus. Ce type architectural est parmi les plus utiliser dans le monde professionnel et permet :

- Une amplification de la sécurité, comme il a été spécifié dans les exigences.
- Une limitation des interruptions de services, comme il a été spécifié dans les exigences.
- La prise en charge des appareils fixes et mobiles, comme il a été spécifié dans les exigences.
- Livraison à intervalle régulier, comme il a été spécifié dans les exigences.
- Développement d'un système évolutif, comme il a été spécifié dans les exigences.

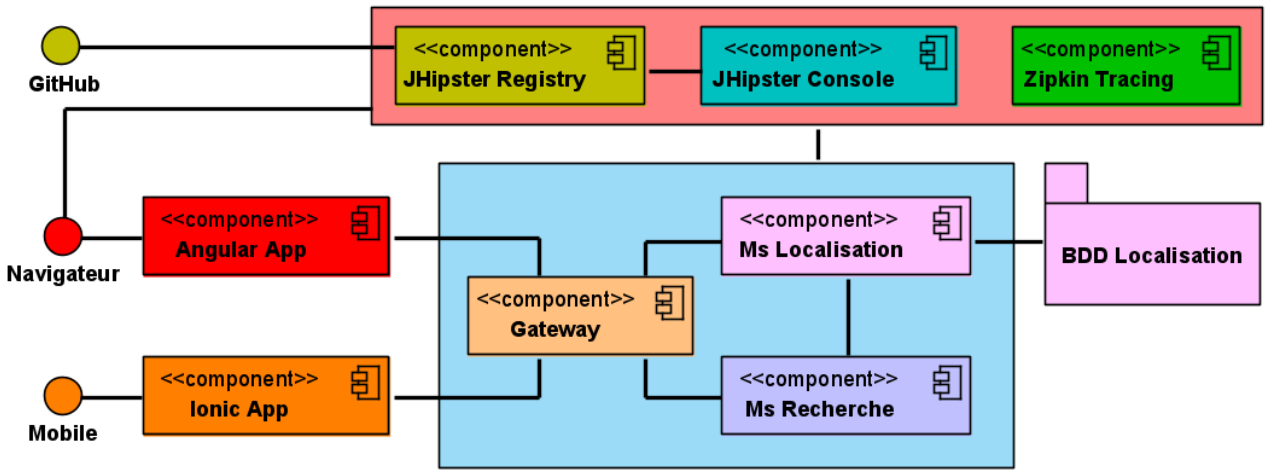


Diagramme de composant – Phase 1

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

Par la suite, afin d'apporter un maximum de clarté, nous proposons une description complète et détaillée des composants de ce diagramme et ceci 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, à savoir : Docker, Kubernetes et Jenkins.

Docker — Outil de conteneurisation

L'utilisation d'un outil de conteneurisation tel que Docker permet une amélioration des performances en permettant une économie considérable des ressources systèmes. Effectivement, Construit sur des capacités du noyau Linux, un conteneur Docker, à l'opposé de machines virtuelles traditionnelles, ne requiert aucun système d'exploitation séparé et n'en fournit aucun.

Ajouter à cela, une facilité accrue à la gestion des dépendances, un conteneur permet d'isoler chaque service : le serveur web, la base de données, des applications pouvant être exécutées de façon indépendante dans leur conteneur dédié, contenant uniquement les dépendances nécessaires.

De plus, la gestion des déploiements est également allégée, utiliser Docker pour créer et gérer des conteneurs simplifie la mise en œuvre de systèmes distribués en permettant à de multiples applications, tâches de fond et autres processus de s'exécuter de façon autonome sur une seule machine physique ou à travers un éventail de machines isolées.

Enfin, même si cela semble évident, il est tout de même de bon ton de rappeler qu'utiliser un outil de conteneurisation permet également d'utiliser un orchestrateur de conteneurs et de bénéficier des nombreux avantages qu'il apporte à tous systèmes.

Kubernetes — Orchestrateur de conteneurs

L'utilisation d'un orchestrateur de conteneurs tel que Kubernetes apporte beaucoup à un quelconque système, effectivement, on pourrait résumer son utiliser à cette phrase assez éloquente : « Kubernetes fourni une plate-forme permettant d'automatiser le déploiement, la montée en charge et la mise en œuvre de conteneurs d'application sur des clusters de serveurs ».

Outre la facilité dans le déploiement distribué ainsi que l'automatisation de la montée en charge offrant à un système un potentiel de scalabilité hors du commun, se traduisant par une résilience exceptionnelle, comme évoqué dans le paragraphe ci-dessus, Kubernetes permet de limiter voire même d'empêcher les interruptions de service, comme désiré par Foosus, en offrant la possibilité de manager les composants sous de multiples versions au travers d'une stratégie efficace de déploiement continu.

Jenkins — Serveur d'automatisation

Jenkins est un serveur d'automatisation permettant un développement basé sur le principe de l'intégration continue, qui consiste à tester le système au fur et à mesure de son avancement (de façon unitaire comme intégrée) et ainsi éviter les bugs conséquents de fin de projet, bien souvent, difficile à résoudre.

De plus, il est à noter qu'en présence d'un outil de conteneurisation associé à un orchestrateur de conteneur, nous pouvons passer d'un développement prônant l'intégration continu à un développement sur le principe du déploiement continu de la solution.

Frontend

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

La première est, comme spécifié par Foosus dans ses recommandations concernant le projet, que ce serait un faux raccourci, effectivement Foosus a beaucoup de problèmes divers dans son système et une évolution pourrait révéler de mauvaises surprises et alourdir considérablement la tâche des équipes techniques pour des résultats qui n'en vaudront probablement pas la peine.

Secondement, pour rejoindre le premièrement, les outils qui ont été choisi pour développer les frontend ne sont pas anodins. Nous retrouvons Angular, un framework de programmation par composant, qui demande une grande réflexion architecturale dans la mise en place des composants afin d'éviter de se retrouver avec une application surchargée, avec un faible potentiel de réutilisabilité, et non optimisée, qui engendrerait, sans aucun doute, des problèmes de performances au fil de l'évolution de l'application. En ce qui concerne le client mobile Ionic, se basant sur du code Angular, les mêmes soucis sont donc à prévoir.

Troisièmement, le système possède beaucoup trop d'application web différentes à notre goût, nous souhaitons une unification de ces systèmes en un seul et unique, optimisé et pensé pour cette fonction.

Enfin, 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.

<u>Application web – Angular App</u>

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 deux parties.

Notre choix s'est porté sur cette pile technologique pour plusieurs raisons, dont le fait que cette technologie est déjà présente dans le système actuel, ce qui signifie qu'elle est connue des équipes techniques ce qui facilitera considérablement sa mise en place.

Il faut également souligner que Foosus a fait un très bon choix technologique en choisissant cette pile, elle est dans les standards actuels et permet, comme on va le voir avec la partie mobile, de gagner un temps considérable dans le développement d'une application mobile cross-platform.

De plus, il est à noter que c'est une solution rapide à mettre en œuvre, effectivement, nous nous reposons sur l'outil JHipster qui a été développé dans un souci d'augmentation de la rapidité dans le développement d'application professionnelle.

Ces principales raisons rendent le choix de cette pile efficace et cohérente que ce soit au niveau des souhaits de Foosus, qui pour mémoire désire une mise en œuvre rapide pour supplanter la concurrence ou que ce soit au niveau de l'intégration avec le reste du système actuel qui est déjà managé par JHipster.

Enfin, 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.

Notre choix s'est porté sur cette pile technologique pour les mêmes raisons que pour l'application web, à savoir que cette technologie est déjà présente dans le système actuel, qu'elle se base aussi sur JHipster, ainsi qu'elle est à la fois efficace grâce à la présence d'une application web Angular dans le système et à la fois cohérente par rapport aux souhaits exprimés par Foosus.

Enfin, 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, de nouveau, la libraire Javascript de Google : Google Maps.

Backend

Le choix des technologies backend a reposé, tout comme le frontend, sur un désire de rapidité dans la mise en place du nouveau système ainsi que de la maitrise des outils par les équipes techniques.

Il parait donc évident que l'intégralité des composants se sont reposés sur JHipster, cette fois ci en excluant les parties frontend des projets et en ne gardant donc que les sections sous 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.

Le principal inconvénient de JHipster, coté backend, est qu'il n'est pas compatible avec tous les frameworks disponible sur le marché, il faut donc attendre l'intégration de ces derniers dans la pile par les équipes techniques de cette compagnie.

Néanmoins il est tout de même important de souligner que la pile est déjà bien fournie, elle offre beaucoup de possibilités, les intégrations se font au fur et à mesure, lorsque les outils font leurs preuves dans le monde professionnel. On peut par exemple penser à Spring Cloud Gateway ou bien à Spring Cloud LoadBalancer qui sont de nouveaux outils, trop récents pour faire partie de la pile, dont on attend les preuves d'efficacité.

Serveur passerelle – Gateway

Une passerelle dans une architecture microservices est un standard permettant d'avoir un seul point d'entrée dans le système backend et par conséquent d'augmenter considérablement la sécurité de la solution.

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.

Plutôt que de modifier le système de recherche actuel pour y inclure les nouvelles fonctionnalités, nous préférons nous concentrer sur la création d'un nouveau service remplissant les nouveaux standards architecturaux et ainsi éviter les mauvaises surprises qui pourraient se déclarer à la suite de l'évolution du système de recherche existant qui n'a, par ailleurs, pas été conçu dans les règles de l'art, c'est à dire sous la supervision d'un professionnel de l'architecture ainsi que dans un but évolutif.

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 un serveur de découverte des instances qui, dans une architecture microservices, est un standard permettant d'enregistrer les multiples instances des différents microservices déployées.

Pour faire simple, on peut considérer le serveur de découverte des instances comme l'annuaire dont le gateway se servira pour contacter les microservices, ce qui, en autres, permet éviter l'utilisation d'URL absolue qui peut s'avérer problématique lorsque plusieurs instances d'un microservice sont déployées.

L'utilisation d'un serveur de découverte ne se résume pas seulement à ça, son utilisation apporte de nombreux avantages, cette simplification permet à tous lecteurs de comprendre son utilité primaire au sein d'un système, à savoir : servir d'annuaire.

Spring Cloud Config Server sert à mettre en place un serveur de configuration externalisée. Les configurations ne sont donc plus compilées dans les livrables en même temps que le code source mais sont présentes dans un dépôt en ligne, dans notre cas GitHub, ce qui permet d'effectuer des modifications en temps réelles sans devoir recompiler puis redéployer les composants.

Il est à noter que ce type d'outil a également un rôle dans la limitation des interruptions de services et la réduction de la charge de travail inutile.

Enfin, 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.

Elasticsearch permet l'indexation ainsi que la recherche de données. Pour ce faire il fournit un moteur de recherche distribué et multi-entité à travers une interface REST.

Logstash est un outil informatique de collecte, analyse et stockage de logs qui peut être considéré comme un ETL (Extract-transform-load). Il permet de centraliser les différentes traces et d'en faire une analyse efficace. Il est capable de gérer pratiquement tous les types de logs : journaux du système, journaux du serveur Web, journaux d'erreurs et journaux des applications. Il se positionne côté serveur et permet de filtrer les messages, d'extraire des informations utiles et de les stocker pour les indexer.

Kibana est un greffon de visualisation de données pour Elasticsearch, il fournit des fonctions de visualisation sur du contenu indexé dans une grappe Elasticsearch. Les utilisateurs peuvent créer des diagrammes en barre, en ligne, des nuages de points, des camemberts et des cartes de grands volumes de données.

Pour résumer simplement ce qui a été dit, 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 (date, en-têtes, temps de parcours, etc.).

Avec la multiplication des architectures orientée services et microservices, Open Zipkin est aujourd'hui un incontournable dans le monde professionnel, il permet, bien souvent, de localiser précisément le composant souffrant d'un bug ou autres problèmes, ce qui facilite vivement le débogage.

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. On aurait pu penser qu'une base de données par microservice aurait été mis en place mais il ne nous est pas apparu nécessaire d'avoir une base de données pour le microservice de recherche.

Effectivement, ce microservice de recherche n'a pas besoin d'enregistrer de données primordiales pour fonctionner contrairement au microservice de localisation qui devra au moins enregistrer les adresses où les produits locaux devront être récupéré sous un format adapter à la recherche par rayon, tel que les coordonnées géographiques.

Base de données des localisations – BDD Localisation

Comme il été vu ci-dessus, une base de données dédiée au microservice de localisation est obligatoire pour que le système ait le fonctionnement attendu. On aurait pu penser s'en passer et se reposer sur l'adresse fourni par les utilisateurs dans leurs profils, en interrogeant le système dédié dans la solution existante, mais le format d'enregistrement textuel pour les adresses est inadapté pour la recherche et aurait impliqué des temps de réponses records au mieux et un crash du système au pire, c'est à dire si les requêtes s'enchainent.

Nous proposons une base de données relationnelle MySQL qui est une technologie open source ayant déjà fait ses preuves dans le milieu professionnel. De plus, elle possède de nombreux outils permettant son contrôle, sa surveillance, ainsi que le choix entre un déploiement cloud ou physique.

Enfin, il faut 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 présents dans 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, sur le diagramme décrivant la solution actuelle étudiée au chapitre précédent (architecture de base), on peut remarquer qu'elle est composée de 4 systèmes primaires. Parmi ces systèmes, 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, ce qui est possible sans modification préalable car il a bien été spécifié que cette distinction est déjà présente actuellement ; 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 vers une architecture microservices

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 fonctionnalités 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 à notre nouveau style architectural.

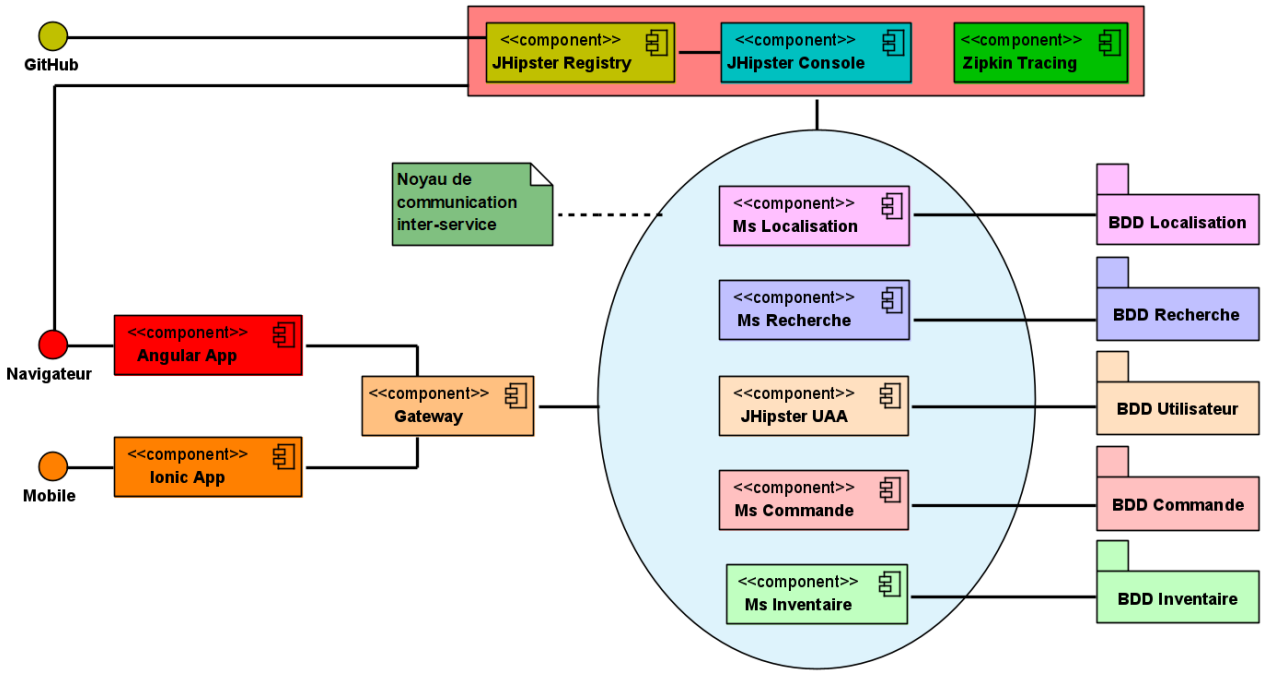


Diagramme de composant – Phase 2

Tout comme pour la première phase, nous proposons une description complète et détaillée 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.

Son premier rôle de serveur de ressources est tout à fait lambda et consiste simplement en une gestion des utilisateurs et de leurs informations. Néanmoins, il est tout de même à noter que cette première fonction est indispensable au système afin de pour pouvoir réaliser la deuxième.

Sa seconde fonction de serveur d'autorisation joue un rôle essentiel dans le système et consiste en une communication quasi-permanente avec la passerelle dans le but de valider chaque requête devant traverser le système à l'aide de l'outil OAuth 2.0 qui est, de manière simplifiée, un protocole de délégation d'autorisation fonctionnant avec un système de jeton et permettant d'autoriser l'accès à une ressource en protégeant les informations de l'utilisateur.

Ce composant remplace donc « Foosus Identity System » présent dans la solution initiale, et ceci en offrant une sécurité bien supérieure à ce qui était implémenté.

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.

Le microservice de gestion des commandes remplira donc les mêmes fonctionnalités que le composant initial mais aura l'avantage d'être adapté pour fonctionner avec le nouveau style architectural.

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 » présent dans 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.

Le microservice de gestion des inventaires remplira donc les mêmes fonctionnalités que le composant initial mais aura l'avantage d'être adapté pour fonctionner avec le nouveau style architectural.

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

Bien que trois microservices ont été ajoutés dans cette seconde phase, nous avons requis la mise en place de 4 bases de données supplémentaires. Effectivement, à la suite de la première phase, le système a le fonctionnement attendu, par conséquent, notre approche lors de cette seconde phase ne se focalise plus uniquement sur la rapidité de livraison.

Nous pensons donc que le moment est bien choisi pour prendre en compte les besoins secondaires que pourrait avoir le microservice de gestion des recherches avec la mise en place de sa base de données.

Néanmoins, ce n'est pas le plus important dans cette seconde phase, alors il faudra veiller à ce que le déploiement de son stockage de données dédié soit bien orchestrer par rapport aux autres tâches.

Il est important de souligner, comme on peut le constater sur le diagramme, que chaque microservice à sa propre base de données qui lui est totalement dédiée. Cette configuration permet de respecter au mieux l'esprit véhiculé par l'architecture microservices, à savoir, une indépendance forte des composants induisant donc un couplage faible entre eux. Cette configuration prend toute son importance dans plusieurs types d'opération tel que la mise à l'échelle des composants. À contrario, dans une architecture orientée services une seule base commune aurait été préconisée.

Base de données des recherches – BDD Recherche

Comme il été vu ci-dessus, une base de données dédiée au microservice de gestion des recherches est nécessaire pour que le système ait le fonctionnement attendu.

Nous proposons une base de données relationnelle MySQL qui est une technologie open source ayant déjà fait ses preuves dans le milieu professionnel. De plus, elle possède de nombreux outils permettant son contrôle, sa surveillance, ainsi que le choix entre un déploiement cloud ou physique.

Enfin, il faut 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

Comme il été vu, une base de données dédiée au microservice de gestion des utilisateurs est nécessaire pour que le système ait le fonctionnement attendu.

Nous proposons, encore une fois, une base de données relationnelle MySQL qui est une technologie open source ayant déjà fait ses preuves dans le milieu professionnel. De plus, elle possède de nombreux outils permettant son contrôle, sa surveillance, ainsi que le choix entre un déploiement cloud ou physique.

Enfin, il faut, encore une fois, 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

Comme il été vu, une base de données dédiée au microservice de gestion des commandes est nécessaire pour que le système ait le fonctionnement attendu.

Nous proposons, encore une fois, une base de données relationnelle MySQL qui est une technologie open source ayant déjà fait ses preuves dans le milieu professionnel. De plus, elle possède de nombreux outils permettant son contrôle, sa surveillance, ainsi que le choix entre un déploiement cloud ou physique.

Enfin, il faut, encore une fois, 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

Comme il été vu, une base de données dédiée au microservice de gestion des inventaires est nécessaire pour que le système ait le fonctionnement attendu.

Nous proposons, encore une fois, une base de données relationnelle MySQL qui est une technologie open source ayant déjà fait ses preuves dans le milieu professionnel. De plus, elle possède de nombreux outils permettant son contrôle, sa surveillance, ainsi que le choix entre un déploiement cloud ou physique.

Enfin, il faut, encore une fois, souligner, qu'au besoin, une offre professionnelle payante est disponible et qu'une migration vers cette version se fait naturellement.

Place de l'itération dans l'ADM

Le processus itératif est le meilleur moyen pour parvenir à ses objectifs c'est-à-dire la transformation de l'architecture d'entreprise pour avoir une vision globale des aspects stratégiques, métiers, organisationnels et pour maîtriser l'alignement entre le métier et la technique, clé d'un Système d'Information agile.

Comme toutes bonnes méthodes, TOGAF est un ensemble de bonnes pratiques qu'il faut adapter au contexte et qu'on est libre d'implémenter ou non.

TOGAF conseille d'avoir 4 cycles de plusieurs itérations où chaque itération regroupe les phases suivantes :

- Itération de contexte, composé des phases « préliminaire » et « A. Vision de l'architecture ».
- Itération de définition de l'architecture, composé des phases « B. Architecture business », « C. Architecture des systèmes d'Information » et « D. Architecture technologique ».
- Itération de transition et de planning, composé des phases « E. Opportunités et Solutions » et « F. Planning de migration ».
- Itération de gouvernance, composé des phases « G. Gouvernance de l'implémentation », « H. Management du changement d'architecture ».

Par exemple, afin d'aboutir par incrément à l'architecture métier, système et technique avant de passer aux itérations de transition et de planning (phases E et F), on peut planifier plusieurs itérations de définition (phases B, C et D) :

- Itération 1 : « B. Architecture business 1 », « C. Architecture des systèmes d'Information 1 », « D. Architecture technologique 1 ».
- Itération 2 : « B. Architecture business 2 », « C. Architecture des systèmes d'Information 2 », « D. Architecture technologique 2 ».
- Itération 3 : « B. Architecture business 3 », « C. Architecture des systèmes d'Information 3 », « D. Architecture technologique 3 ».

Dans un contexte de refonte du Système d'Information, où on doit concevoir des solutions dans un laps de temps réduit, il est conseillé que la première itération de définition (« B. Architecture business 1 », « C. Architecture des systèmes d'Information 1 », « D. Architecture technologique 1 ») se concentre sur l'architecture cible et ensuite dans la seconde itération insister sur l'architecture existante.

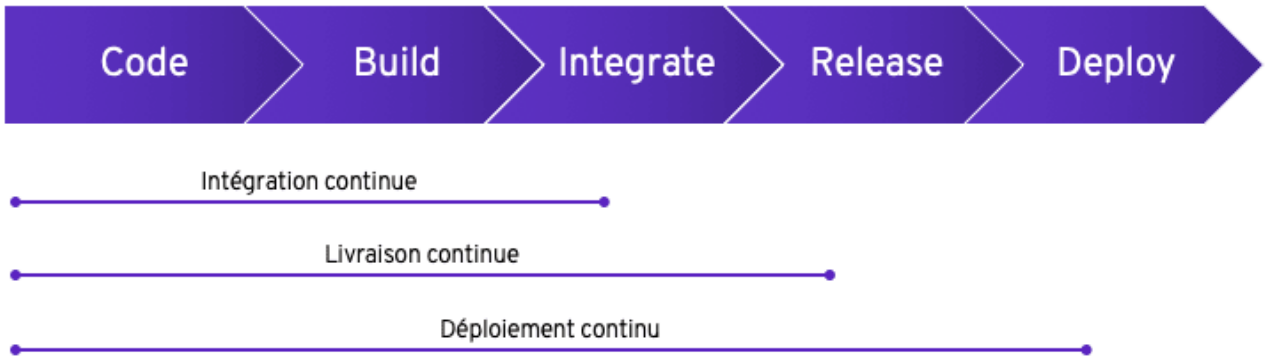
Il faut simplement garder à l'esprit qu'il faut s'adapter au contexte.

En fonction de la situation, on peut aussi prendre parti pour une planification opposée de la précédente, c'est-à-dire la première itération de définition (« B. Architecture business 1 », « C. Architecture des systèmes d'Information 1 », « D. Architecture technologique 1 ») se concentre sur l'architecture existante et ensuite dans la seconde itération insister sur l'architecture cible.

Bonne pratique de développement

Cette partie sera consacrée aux bonnes pratiques de développement non techniques. Il sera donc laissé de côté tout ce qui attrait directement au code source (style d'écriture, qualité, métriques, couplage des classes, instructions à éviter, etc.), à sa documentation ainsi qu'aux règles de conception.

Nous nous concentrerons donc sur des points, comme on va le voir, qui ont déjà été évoqués dans la partie, ci-dessus, traitant de l'architecture cible.



Intégration continue

Cette méthode repose souvent sur la mise en place d'une brique logicielle permettant l'automatisation de tâches : compilation, tests unitaires et fonctionnels, validation produit, tests de performances… À chaque changement du code, cette brique logicielle va exécuter un ensemble de tâches et produire un ensemble de résultats, que le développeur peut par la suite consulter. Cette intégration permet ainsi de ne pas oublier d'éléments lors de la mise en production et donc ainsi améliorer la qualité du produit final.

Pour appliquer cette technique, il faut d'abord que :

- Le code source soit partagé (en utilisant des logiciels de gestion de versions tels que CVS, Subversion, git, Mercurial, etc.) ;
- Les développeurs intègrent (commit) quotidiennement (au moins) leurs modifications ;
- Des tests d'intégration soient développés pour valider l'application ;
- Un outil d'intégration continue est ensuite nécessaire, tel que CruiseControl ou Jenkins. D'autres outils, comme SonarQube ou Jacoco, peuvent être mis en place afin de superviser la qualité du code.

Les principaux avantages d'une telle technique de développement sont :

- Le test immédiat des modifications ;
- La notification rapide en cas de code incompatible ou manquant ;
- Les problèmes d'intégration sont détectés et réparés de façon continue, évitant les problèmes de dernière minute ;
- Une version est toujours disponible pour un test, une démonstration ou une distribution.

Livraison continue

La livraison continue est une approche d'ingénierie logicielle dans laquelle les équipes produisent des logiciels dans des cycles courts, ce qui permet de le mettre à disposition à n'importe quel moment. Le but est de construire, tester et diffuser un logiciel plus rapidement.

L'approche aide à réduire le temps d'évaluation des risques, et les risques associés à la livraison de changement en adoptant une approche plus incrémentielle des modifications en production. A terme, l'objectif est une réduction du coût. Un processus simple et répétable de déploiement est un élément clé

Déploiement continu

Le déploiement continu, est une approche d'ingénierie logicielle dans laquelle les fonctionnalités logicielles sont livrées fréquemment par le biais de déploiements automatisés. Le déploiement continu diffère de la livraison continue, une approche similaire, détaillée ci-dessus, dans laquelle des fonctionnalités logicielles sont également livrées fréquemment et considérées comme pouvant potentiellement être déployées, mais qui ne le sont pas pour autant, effectivement le déploiement restant dans ce cas un processus manuel.

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

Foosus

Risques et acceptation

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

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 apportées 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.

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