Contrat de Conception et de Développement de l'Architecture

Projet *:* Concevez une Nouvelle Architecture afin de Soutenir le Développement de votre Entreprise

Client *:* FOOSUS

Préparé par *: COSTA, Elias*

N° de Version du Document *:* 0.1

Titre *:* Contrat de Conception et de Développement de l'Architecture

Date de Version du Document *: 9/03/2021*

Revu par *: Jean Noel Gérard - Mentor*

Date de Révision *: 8/03/2021*

Table des matières

1. Introduction et Contexte

L'objectif d'une architecture d'entreprise telle que proposée par TOGAF est d'optimiser les processus existants, souvent fragmentés, pour construire un environnement intégré, capable de répondre aux changements et de soutenir la stratégie de l'entreprise.

TOGAF couvre quatre domaines d'architecture. L'architecture d'affaires définit la stratégie d'affaires et la gouvernance. L'architecture des données définit la structure et la gouvernance des données logiques et physiques de l'organisation. L'architecture applicative offre une vision des applications, leurs relations avec les processus d'affaires réalisés et leurs interrelations avec les autres applications. L'architecture technologique représente l'infrastructure sur laquelle reposent les applications d'affaires et les données.

2. La Nature de l'accord

L'essentiel du contenu des trois phases suivantes B (métier), C (système d'information) et D (technique) consiste à détailler l'architecture cible et initiale, à mesurer l'écart entre les deux, puis à évaluer les impacts des évolutions sur l'ensemble des facettes de l'entreprise. La combinaison de ces éléments permet d'établir un premier scénario de la feuille de route de transition.

3. Objectifs et périmètre

3.1. Objectifs business

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. Après plusieurs années de développement, notre solution technique complexe n'évolue plus au rythme de l'activité et risque d'entraver notre croissance. Les études de marché et les analyses commerciales montrent que nos clients souhaitent acheter local et soutiennent les producteurs locaux.

Nos concurrents n'ont pas ciblé cette niche. Nous voulons nous appuyer sur les

connaissances acquises 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.

Pour atteindre ses objectifs, Foosus propose de continuer à 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, grâce à l'élaboration d'une nouvelle plateforme d'e-commerce afin d'améliorer sa compétitivité par rapport aux grandes entreprises d'e-commerce internationales.

Notre 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.

3.2. Périmètre

1. Tirer parti de la géolocalisation pour relier des fournisseurs et des consommateurs et pour proposer des produits disponibles près des lieux de résidence de ces derniers. Un calculateur de distance devra être inclus pour permettre aux consommateurs de trouver les fournisseurs les plus proches d'eux.

2. Mettre en place une architecture évolutive pour que nous puissions déployer nos services sur diverses régions, dans des villes et des pays donnés.

3. Introduire des améliorations et des modifications aux systèmes de production afin de limiter ou supprimer la nécessité d'interrompre le service pour procéder au déploiement.

4. Faciliter l'accès des fournisseurs et des consommateurs à la plateforme où qu'ils se trouvent. Cette solution doit être utilisable avec des appareils mobiles et fixes. Elle doit tenir compte des contraintes de bande passante pour les réseaux cellulaires et les connexions Internet haut débit.

5. Concevoir une solution qui prend en charge divers types d'utilisateurs (par exemple, fournisseurs, back-office, consommateurs), avec des fonctionnalités et des services spécifiques pour ces catégories.

6. Fournir des livrables à intervalles réguliers pour que le nouveau système soit rapidement opérationnel et puisse être doté de nouvelles fonctionnalités au fil

du temps.

7. Permettre aux 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.

8. Faire évoluer la pile technologique naturellement au même rythme que sa base de clientèle.

9. Créer des infrastructures qui permettent à la plateforme d'absorber le trafic et d'être capable d'évoluer pour gérer les augmentations de charges.

10. Renforcer les mécanismes de sécurité afin d'éviter les risques pour l'image de marque de l'entreprise et assurer que la plateforme soit disponible 24h / 24 et 7j / 7.

11. Faire en sorte que chaque nouvelle version soit de taille réduite, présentant peu de risques, et qu'elle soit transparente pour les. C'est quand nos utilisateurs peuvent accéder facilement à nos services et apprécient notre produit que nous réussissons.

3.3. Parties prenantes, préoccupations et visions

Partie prenante	Préoccupation	Vision
Ash Callum, CEO	L'architecture métier	1. Soutenir l'alimentation locale et mettre les consommateurs en contact avec des producteurs et des artisans locaux. 2. Créer une plateforme de commerce électronique polyvalente pour faire passer l'entreprise à un niveau supérieur. 3. Innover pour soutenir la croissance de l'entreprise. 4. Pouvoir concurrencer les grandes entreprises mondiales de commerce électronique. 5. Maintenir un taux positif

		d'inscriptions de nouveaux utilisateurs 6. Améliorer la réputation de Foosus
Natasha Jarson, CIO	L'architecture métier, L'architecture des données L'architecture applicative L'architecture technologique	1. Soutenir l'innovation technique rapide et l'expérimentation 2. Construire une solution géociblée avec une nouvelle architecture. 3. Construire une solution résiliente, évolutive, performante, de haute disponibilité, facile à utiliser et sécurisée.
Daniel Anthony, CPO	L'architecture métier, L'architecture des données	1. Soutenir l'innovation technique rapide et l'expérimentation 2. Avoir un design d'architecture qui nous offre en temps réel des connaissances et une vision de la santé de la plateforme techniquement et d'un point de vue commercial. 3. Innover pour soutenir la croissance de l'entreprise. 4. Obtenir des informations précises sur les habitudes de consommation des clients
Christina Orgega CMO	L'architecture métier, L'architecture des données	1. Soutenir l'innovation technique rapide et l'expérimentation 2. Avoir un design d'architecture qui nous offre en temps réel des connaissances et une vision de la santé de la plateforme techniquement et d'un point de vue commercial. 3. Obtenir des informations précises sur les habitudes de consommation des clients

Jo Kumar CFO	L'architecture métier	1. Soutenir l'innovation technique rapide et l'expérimentation 2. Innover pour soutenir la croissance de l'entreprise. 3. Maintenir un taux positif d'inscriptions de nouveaux utilisateurs.
Elias Costa Promoteur de l'Architecture	L'architecture métier, L'architecture des données L'architecture applicative L'architecture technologique	1. Proposer une nouvelle architecture. 2. Éliminer la dette technique et le manque de cohérence qui impacte de manière significative le développement de fonctionnalités.
Pete Parker Responsable Ingénierie	L'architecture des données L'architecture applicative L'architecture technologique	1. Mettre en œuvre la nouvelle architecture. 2. Éliminer la dette technique et le manque de cohérence qui impacte de manière significative le développement de fonctionnalités.
Jack Harkner, Directeur des Opérations	L'architecture des données L'architecture applicative L'architecture technologique	1. Mettre en œuvre la nouvelle architecture. 2. Éliminer la dette technique et le manque de cohérence qui impacte de manière significative le développement de fonctionnalités. 3. Avoir un design d'architecture qui nous offre en temps réel des connaissances et une vision de la santé de la plateforme techniquement et d'un point de vue commercial.
Équipe de Développement	L'architecture des données L'architecture applicative L'architecture technologique	1. Mettre en œuvre la nouvelle architecture. 2. Construire une solution résiliente, évolutive, performante, de haute disponibilité, facile à utiliser et sécurisée.

		3. Proposer des idées d'innovation technique rapide et d'expérimentation
Équipe Commerciale	L'architecture métier L'architecture des données	1. Promouvoir les services Foosus 2. Avoir un design d'architecture qui nous offre en temps réel des connaissances et une vision de la santé de la plateforme techniquement et d'un point de vue commercial.
Producteurs	L'architecture métier L'architecture des données L'architecture applicative	1. Disposer de mécanismes de mise à jour des stocks. 2. Bénéficier des mécanismes de paiement en ligne. 3. Pouvoir bénéficier de services de transport de colis efficaces.
Clients	L'architecture métier L'architecture applicative	1. Avoir un mécanisme de recherche de produits basé sur la géolocalisation des producteurs. 2. Bénéficier des mécanismes de paiement en ligne. 3. Pouvoir bénéficier de services de transport de colis efficaces.

4. Description de l'architecture, principes stratégiques et conditions requises

4.1. Description

La plateforme historique de Foosus a naturellement évolué vers la complexité en raison du changement rapide et d'un manque de vision long terme. En pratique, les backends sont de grosses applications monolithiques qui effectuent plus que de simples passages de commandes. Les migrations architecturales et technologiques sont essentielles pour la croissance de Foosus.

Le marché actuel voit nos concurrents directs prendre rapidement l'avantage en pivotant en réponse à de nouvelles informations apprises. L'apprentissage doit être au cœur de notre état cible de l'architecture, étant donné que cela a été verrouillé par des solutions par le passé, d'une manière qui a généré davantage d'instabilité et de dette technique.

La plateforme doit être conçue en gardant à l'idée l'extensibilité et la personnalisation des fonctionnalités.

Le comportement technique de la plateforme, non plus que sa performance d'un point de vue du business, n'est pas clair. Toutes les connaissances acquises actuellement nécessitent des analyses de registres et de feuilles de calcul, avant de pouvoir rechercher l'intelligence business.

La marque Foosus doit être renforcée en réduisant les interruptions de service visibles par les utilisateurs. Cela implique :

- Des process pour réduire le risque de sortir des solutions qui échouent ou qui soient de mauvaise qualité
- La capacité de sortir de nouvelles versions de notre plateforme sans impacter l'utilisateur par des interruptions de service.

Les sorties à 3 h du matin semblaient fonctionner lorsque nos utilisateurs se trouvaient principalement dans la même zone géographique, mais nous ne devons plus dépendre de cela.

Nous voulons mener des campagnes de marketing Foosus dans de nombreuses

grandes villes, avec l'assurance que notre plateforme demeurera utilisable, réactive, et délivrera une expérience client de première classe. Pour parvenir à ce but, nous avons besoin de concevoir une solution d'architecture et un processus de gouvernance qui nous aident à atteindre l'ensemble des objectifs business actuels, ainsi que la vision globale.

4.2. Principes stratégiques

4.2.1. Principes généraux

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

4.2.2. Principes Business

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

4.2.3. Principes Data

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

4.2.4. Principes d'application

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

4.2.5. Principes technologiques

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

4.3. Référence aux Conditions requises pour l'architecture

1. Recherche dans l'interface client et commande de produits de consommation.
2. Les fournisseurs alimentaires soumettront à Foosus un inventaire des produits alimentaires disponibles. Afin de maintenir constamment les stocks de produits à jour, Foosus doit impliquer les producteurs dans le processus de mise à jour de leurs stocks, en facilitant les mécanismes de mise à jour automatisés ou manuels.
3. Les clients des produits de consommation trouveront et commanderont des

produits alimentaires.

4. Les fournisseurs alimentaires recevront des commandes.
5. L'équipe finance de Foosus recevra les paiements. Les systèmes de facturation doivent garantir que les fournisseurs alimentaires soient facturés pour une commission, comme tous les paiements soient effectués directement à la livraison.

5. Livrables architecturaux

Le schéma suivant montre le processus des producteurs dans le logiciel Foosus:

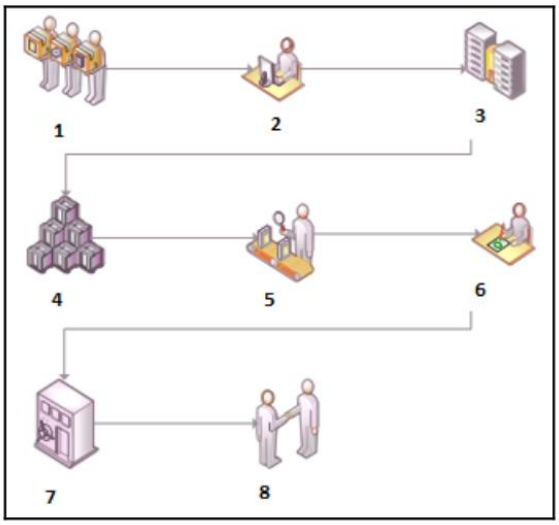


Figure 1 - Processus des producteurs dans le logiciel Foosus

Le diagramme précédent définit le flux de travail des producteurs qui utilisent Foosus, comme suit:

1. Le producteur visite l'application Foosus.
2. Le producteur s'inscrit au Foosus.
3. Ensuite, le producteur enregistre ses types de produits.
4. Le producteur crée l'inventaire.
5. L'inventaire est validé/approuvé par les administrateurs Foosus.

6. Foosus demande au producteur d'approuver l'inventaire.
7. Le producteur paie les frais d'inscription.
8. Un accord de partenariat Foosus-producteur est conclu.

Le schéma suivant montre le processus des acheteurs au sein du logiciel Foosus:

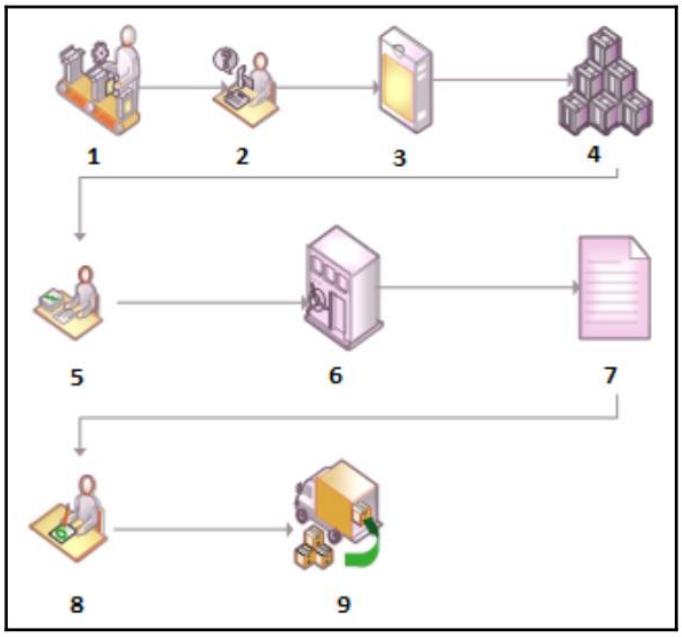


Figure 2 - Processus des acheteurs au sein du logiciel Foosus

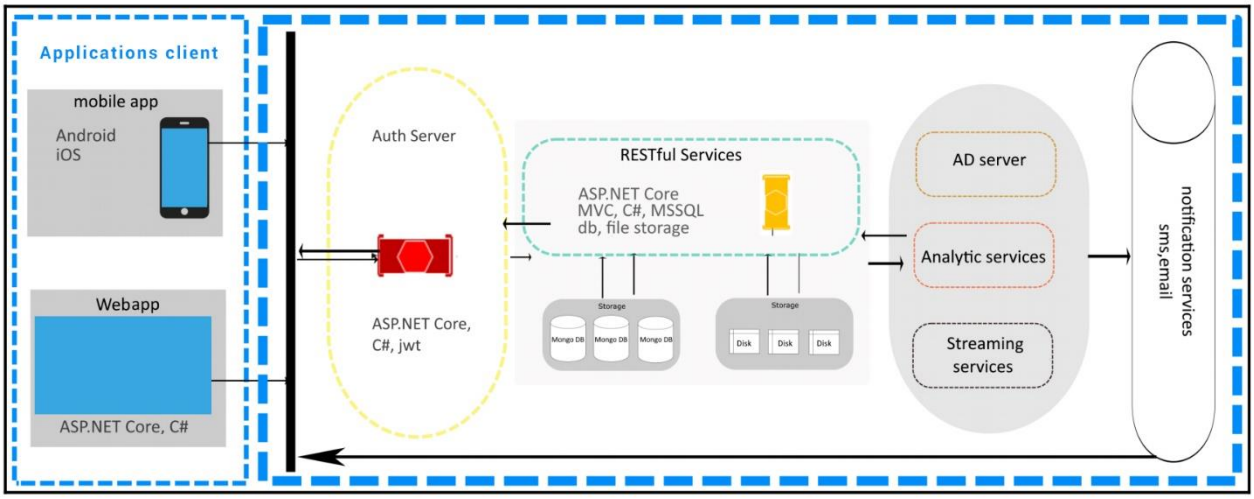
Le diagramme précédent définit le flux de travail des acheteurs qui utilisent Foosus, comme suit:

1. L'acheteur visite l'application Foosus.
2. L'acheteur s'inscrit auprès du Foosus.
3. Les produits et services disponibles sont proposés.
4. L'acheteur vérifie l'inventaire disponible pour le produit qu'il souhaite acheter.
5. L'acheteur passe une commande.
6. L'acheteur paie son article.
7. Un processus de contrat intelligent se produit.
8. La confirmation de la transaction récente est fournie à l'acheteur.
9. L'article est expédié à l'emplacement souhaité.

L'image complète du système Foosus contient les éléments suivants:

- **Panneau d'administration:** il s'agit du backend géré par l'équipe Foosus.
- **Panel des producteurs:** c'est l'interface pour tous les producteurs, afin qu'ils puissent gérer leurs produits et leurs stocks, y compris les ventes, etc.
- **Panneau utilisateur:** Cette interface est l'interface principale de l'utilisateur final qui va acheter les produits.

Le schéma suivant donne un aperçu de notre application:



Le diagramme précédent est une représentation picturale de la vue d'ensemble fonctionnelle de notre application, et il montre le flux qui contient les éléments suivants:

- **Applications clientes**: les applications mobiles et Web sont les applications clientes que l'utilisateur final va utiliser.
- **Serveur d'authentification**: il valide l'utilisateur et génère le jeton JWT, pour un traitement ultérieur.
- **Services RESTful**: ce sont les différents services qui vont aider notre application. Ces services ont leurs propres bases de données. Le stockage de fichiers serait un CDN ou un serveur séparé, qui serait utilisé pour stocker divers éléments de contenu, y compris des documents.
- **Services de notification:** il s'agit des services externes utilisés pour générer un mot de passe à usage unique (OTP), pour authentifier l'utilisateur et pour l'informer de la commande qu'il a générée, d'un produit qu'il a réservé, etc.

En dehors de ceux-ci, notre diagramme contient un serveur AD, des services d'analyse (Analytics) et des services de streaming, qui sont nécessaires si nous avons besoin de podcaster certaines de nos vidéos. Pour cela, nous aurions besoin de services de streaming.

6. Développement de l'architecture

Maintenant, nous allons définir le Produit Minimum Viable (Minimum Viable Product - MVP) pour mettre en valeur la force du logiciel. Ce logiciel peut être étendu à n'importe quel niveau.

En adoptant une approche MVP, la portée d'un travail est limitée au plus petit ensemble d'exigences, afin de produire un livrable fonctionnel. MVP est souvent combiné avec le développement de logiciels Agile en limitant les exigences à un montant gérable qui peut être conçu, développé, testé et livré.

Nous allons développer une application Foosus qui possède les fonctionnalités suivantes:

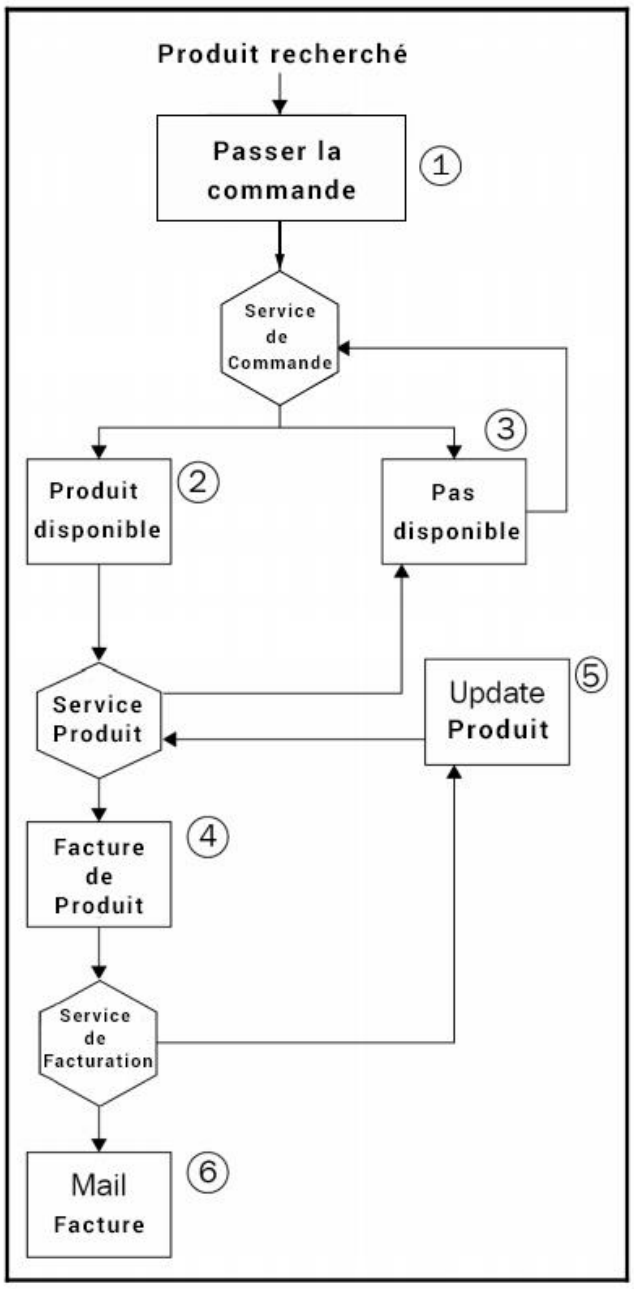
- Fonctionnalité de connexion (Login)
- Fonctionnalité d'inscription
- Fonction de recherche de produits
- Fonctionnalité de commande
- Fonctionnalité de notification

Jetons un coup d'œil au diagramme suivant, qui définit le flux de notre fonctionnalité de recherche. Dans ce diagramme, les services sont représentés par des hexagones, tandis que les événements sont représentés par des cases rectangulaires. Le flux décrit le scénario d'un client passant une commande, après avoir recherché les articles qu'il recherche.

Voici comment ça se passe :

1. L'événement Passer une commande est transmis au service Commande.

2. En réponse à cet événement, notre service analyse les arguments, tels que le produit et la quantité commandés, et déclenche l'événement Produit disponible au Service Produit.



.

3. À partir de là, deux résultats sont possibles: soit le produit demandé est disponible et à la quantité requise, soit il n'est pas disponible ou n'a pas la quantité requise.

4. Si les produits sont disponibles, le service Produit déclenche un événement appelé Générer une facture vers le service Facture. Étant donné que le fait de monter la facture signifie confirmer la commande, les produits sur la facture ne se-

raient plus en stock; nous devons nous en occuper et mettre à jour le stock en conséquence.

5. Pour gérer cela, notre service de facturation déclenche un événement appelé Mettre à jour la quantité de produit vers le service Produit et prend en charge cette exigence. Dans un souci de simplicité, je n'entrerai pas dans le détail sur l'événement de la facture Mail.

6. Il peut y avoir un scénario où un produit n'est pas disponible dans le magasin. Conformément à nos besoins commerciaux, nous devons marquer le produit comme Me notifier une fois disponible. Cela dépend uniquement de l'utilisateur final; c'est-à-dire que si l'utilisateur final a opté pour cette option, il recevra une notification lorsque le produit sera disponible. La même action sera déclenchée pour les produits de prévente, comme le signifie les produits de prévente de Foosus: un produit a été ajouté mais n'est pas encore disponible à l'achat.

7. Justification du choix technologique

7.1. Conditions préalables

Nous utiliserons les outils et technologies suivants lors de la transition de notre application monolithique vers une architecture de type microservice:

- Visual Studio 2019 ou version ultérieure
- C # 8.0 ou version ultérieure
- API ASP.NET Core MVC / Web
- Entity Framework Core
- SQL Server 2019 ou version ultérieure
- Microsof Azure
- Messagerie Asynchrone

7.2. Avantages des Microservices

- **Déploiements indépendants**. Vous pouvez mettre à jour un service sans redéployer toute l'application, et annuler ou restaurer par progression une mise à

jour en cas de problème. Les résolutions de bogues et les publications de fonctionnalités sont plus faciles à gérer et moins risquées.

- **Développement indépendant**. Une équipe de développement unique peut créer, tester et déployer un service. Cela se traduit par une innovation continue et un rythme de publication plus élevé.
- **Équipes restreintes et appliquées**. Les équipes peuvent se concentrer sur un seul service. Du fait de l'étendue réduite de chaque service, la base de code est plus simple à comprendre et les nouveaux membres d'équipe peuvent maîtriser plus facilement les choses.
- **Isolation des pannes**. Si un service connaît une défaillance, cela n'entraîne pas la mise hors ligne complète de l'application. Toutefois, vous ne bénéficiez pas pour autant automatiquement de la résilience de l'application. Vous devez toujours suivre les meilleures pratiques et les patrons de conception en matière de résilience. Pour plus d'informations, consultez Conception d'applications résilientes pour Azure. • Piles technologiques mixtes. Les équipes peuvent choisir la technologie la mieux adaptée à leur service.
- **Mise à l'échelle granulaire**. Les services peuvent être mis à l'échelle indépendamment. Par ailleurs, la plus forte densité de services par machine virtuelle signifie que les ressources de machines virtuelles sont pleinement utilisées. À l'aide de contraintes de positionnement, il est possible d'associer un service à un profil de machine virtuelle (processeur rapide, mémoire élevée, etc.).
- **Alignement avec les objectifs commerciaux**. Il est extrêmement important de comprendre que répondre rapidement aux besoins de l'entreprise et s'adapter aux tendances marketing ne sont pas des sous-produits des microservices, mais des objectifs. La capacité à atteindre ces objectifs avec des équipes plus petites ne fait que rendre les microservices plus adaptés aux propriétaires d'entreprise.
- **Avantages en termes de coûts.** Chaque microservice devient un investissement pour l'entreprise, car il peut facilement être consommé par d'autres microservices, sans avoir à refaire le même code encore et encore. Chaque fois qu'un microservice est réutilisé, du temps est gagné en évitant les tests et le déploiement de cette partie. L'expérience utilisateur est améliorée, car les temps d'arrêt sont soit éliminés, soit réduits au minimum.
- **Évolutivité(scalabilité) facile.** Avec l'isolation verticale en place et chaque microservice rendant un service spécifique à l'ensemble du système, il est facile

à mettre à l'échelle. Non seulement l'identification est plus facile pour les candidats à l'échelle, mais le coût est moindre. En effet, nous ne développons qu'une partie de l'ensemble de l'écosystème des microservices.

- **Sécurité.** La sécurité est similaire à ce qui est fourni par l'architecture en couches traditionnelle; les microservices peuvent être sécurisés aussi facilement. Différentes configurations peuvent être utilisées pour sécuriser différents microservices. Vous pouvez avoir une partie de l'écosystème de microservices derrière les pare-feu et une autre partie pour le chiffrement des utilisateurs. Les microservices Web peuvent être sécurisés différemment du reste des microservices. Vous pouvez répondre à vos besoins selon votre choix, votre technologie ou votre budget.
- **Gestion de données**. Étant donné que chaque microservice a sa propre base de données indépendante, la prise de décision liée aux modifications requises dans la base de données peut être facilement déléguée à l'équipe respective. Nous n'avons pas à nous soucier de l'impact sur le reste du système, car il n'y en aura pas. Au même temps, cette séparation de la base de données ouvre la possibilité à l'équipe de s'auto-organiser.

7.3. Problématiques

- **Complexité**. Une application basée sur des microservices présente plus d'éléments mobiles que l'application monolithique équivalente. Chaque service est plus simple, mais le système dans son ensemble est plus complexe.
- **Manque de gouvernance.** L'approche décentralisée associée à la création de microservices présente des avantages, mais elle peut aussi entraîner des problèmes. Vous pouvez vous retrouver avec un tel nombre de langues et d'infrastructures différentes que l'application devient difficile à mettre à jour. Il peut être utile de mettre en place des normes à l'échelle du projet, sans restreindre outre mesure la flexibilité des équipes. Cela s'applique tout particulièrement aux fonctionnalités transversales, telles que la journalisation.
- **Intégrité des données**. Chaque microservice est lui-même chargé de la persistance de ses données. Par conséquent, la cohérence des données peut s'avérer difficile. Quand cela est possible, adoptez un patron de cohérence éventuelle.
- **Surcharge et latence du réseau.** L'utilisation de nombreux services de petite taille et granulaires peut donner lieu à une communication interservices ac-

crue. En outre, si la chaîne de dépendances de services devient trop longue (le service A appelle le service B, qui appelle le service C, etc.), la latence supplémentaire peut devenir problématique. Vous devrez concevoir les API avec soin. Évitez les API trop « bavardes », prenez en considération les formats de sérialisation et cherchez des endroits où utiliser des patrons de communication asynchrone.

- **Gestion**. La réussite de la mise en œuvre de microservices passe par une culture DevOps mature. La journalisation corrélée pour l'ensemble des services peut être complexe. En général, la journalisation doit mettre en corrélation plusieurs appels de service pour une même opération utilisateur.
- **Gestion des versions.** Les mises à jour d'un service ne doivent pas interrompre les services qui en dépendent. Comme plusieurs services peuvent être mis à jour à tout moment, sans une conception soignée, vous pourriez avoir des problèmes de compatibilité descendante ou ascendante.
- **Développement et test.** Le développement en fonction des dépendances de services exige une approche différente. Les outils existants ne sont pas forcément conçus pour fonctionner avec les dépendances de services. La refactorisation au-delà des limites des services peut s'avérer complexe. Il est également difficile de tester les dépendances de services, surtout quand l'application évolue rapidement.
- **Compétences.** Les microservices sont des systèmes hautement distribués. Évaluez soigneusement si l'équipe possède les compétences et l'expérience requises pour réussir.

7.4. Meilleures pratiques

- Modélisez les services autour du domaine de l'entreprise.
- Décentralisez tout. Des équipes individuelles sont chargées de concevoir et de créer les services. Évitez le partage de code ou de schémas de données.
- Le stockage de données doit être accessible uniquement au service auquel les données appartiennent. Utilisez le meilleur stockage pour chaque type de service et de données.

- Les services communiquent via des API bien conçues. Évitez les fuites des détails de la mise en œuvre. Les API doivent modéliser le domaine et non la mise en œuvre interne du service.
- Évitez le couplage entre les services. Le couplage peut notamment être dû à des schémas de base de données partagés et à des protocoles de communication rigides.
- Confiez les responsabilités transversales, telles que l'authentification et la terminaison SSL, à la passerelle.
- Gardez les connaissances du domaine hors de la passerelle. La passerelle doit gérer et acheminer les requêtes de clients sans aucune connaissance des règles métier ou de la logique de domaine. Dans le cas contraire, la passerelle devient une dépendance et peut causer un couplage entre les services.
- Les services doivent présenter un couplage faible et une forte cohésion fonctionnelle. Les fonctions qui sont susceptibles de changer en même temps doivent être empaquetées et déployées ensemble. S'ils résident dans des services distincts, ces services se retrouvent fortement couplés, car un changement de l'un d'eux exigera la mise à jour de l'autre. Une communication trop importante entre deux services peut être un signe de couplage fort et de faible cohésion.
- Isolez les défaillances. Utilisez des stratégies de résilience afin d'empêcher que les défaillances au sein d'un service n'entraînent une réaction en chaîne. Pour plus d'informations, consultez Conception d'applications résilientes pour Azure.

7.5. La passerelle API

Notre application sera divisée en différents microservices tels que:

- Service produit
- Service de commande
- Service de facturation
- Service Clients

Dans l'interface utilisateur de notre application Foosus, nous devons montrer quelques détails:

- Titre du produit, nom de l'auteur, prix, remise, etc.

- Disponibilité
- Avis sur les produits
- Évaluations de produits
- Classement des producteurs et autres informations sur les producteurs

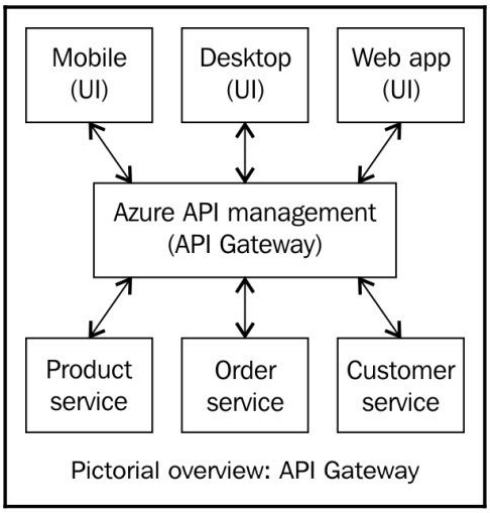
Avant de vérifier l'implémentation, parlons de la passerelle API.

La passerelle API est un point d'entrée unique pour tous les clients. Il agit comme un proxy entre les applications clientes et les services. Dans notre projet, nous utilisons Azure API Management (APIM) comme notre passerelle API.

La passerelle API est responsable des fonctionnalités suivantes:

- Accepter les appels d'API et les acheminer vers nos backends
- Vérification des clés API, des jetons JWT et des certificats
- Prise en charge de l'authentification via Azure AD et le jeton d'accès OAuth 2.0
- Application des quotas d'utilisation et des limites de taux
- Transformer notre API à la volée, sans modifications de code
- Mise en cache des réponses backend, où qu'elles soient configurées
- Journalisation des métadonnées des appels à des fins d'analyse

Le diagramme suivant montre la gestion des API Azure fonctionnant comme une passerelle d'API:



Pictorial overview: API Gateway

Dans l'organigramme précédent, nous avons différents clients, tels qu'une application mobile, une application de bureau et une application Web, qui utilisent des microservices.

Notre client ne sait pas sur quel serveur nos services peuvent être trouvés. La passerelle API fournit l'adresse de son propre serveur et authentifie en interne la demande du client, à l'aide d'une clé de souscription Ocp-Apim-Subscription-Key valide.

Étant donné que nous utilisons Azure API Management en tant que passerelle API, nous bénéficierons de certains avantages:

- Nous pouvons gérer nos différentes API à partir d'une seule plateforme; par exemple, ProductService, OrderService et d'autres services peuvent être facilement gérés et appelés par de nombreux clients.
- Parce que nous utilisons la gestion des API, cela ne nous fournit pas seulement un serveur proxy; cela nous permet également de créer et de maintenir la documentation de nos API.
- Il fournit une fonction intégrée, afin que nous puissions définir diverses politiques pour les quotas, les formats de sortie et les conversions de format, telles que XML en JSON ou vice versa.

7.6. Le service client

Le service client de notre application aura l'ensemble de fonctionnalités suivantes disponibles:

- Authentification d'utilisateur
- Recherche dans les produits disponibles
- Filtrer les produits sur la base des catégories
- Ajout de produits au panier
- Apporter des modifications au panier
- Passer une commande à partir du panier

7.6.1. Le microservice pour lister des produits

On va décomposer la première fonctionnalité de recherche dans les produits. Pour permettre à nos clients de parcourir la boutique à la recherche de produits, nous devons maintenir une liste de produits que nous proposons. Ici, nous avons notre premier candidat à être taillé comme un microservice. Le service de catalogue de produits serait chargé non seulement de rechercher dans les produits disponibles, mais également de maintenir le magasin de données qui contiendrait toutes les informations relatives aux produits. Le microservice doit être en mesure de gérer diverses mises à jour requises pour les produits disponibles dans le système. Nous l'appellerons le **microservice de catalogue de produits**, et il aura son propre magasin de données de produits.

7.6.2. Le microservice de recherche de produits

L'examen de la fonctionnalité suivante de filtrage des produits semble relever de la compétence du microservice de catalogue de produits lui-même. Cependant, cela dit, confirmons-le en remettant en question notre propre compréhension du domaine des affaires ici. La question qui me vient à l'esprit est liée à l'impact de toutes les recherches que nos utilisateurs effectueraient, ce qui ferait tomber le service. Alors, la fonctionnalité de recherche de produits devrait-elle être un service différent? Ici, la réponse réside dans le fait que le microservice doit avoir son propre magasin de données. En ayant le catalogue de produits et la fonction de recherche de catalogue de produits comme des services différents, cela nous obligerait à maintenir une liste de produits dans deux endroits différents avec des défis supplémentaires, comme devoir les synchroniser. La solution est simple: nous avons besoin d'un seul microservice, et si nécessaire, nous devons mettre à l'échelle et équilibrer la charge du microservice produits -catalogue.

7.6.3. Le microservice du panier

Cela devrait nous permettre d'ajouter ou de supprimer des produits de notre panier, avant que nous décidions enfin de les vérifier et de les payer. Il n'y a aucun doute quant à savoir s'il doit s'agir d'un microservice distinct ou non. Cependant, cela soulève une question intéressante de savoir s'il s'agit ou non du magasin de données du produit; il aurait besoin de le faire pour recevoir certains détails fondamentaux, tels que la disponibilité (ce qui est en stock). L'accès au magasin de données à travers le service est hors de question, car c'est l'une des conditions préalables les plus fondamentales pour les microservices. La réponse à notre question est la

communication interservices. Un microservice peut utiliser un service fourni par un autre microservice. Nous appellerons cela notre microservice de panier.

7.6.4. Le microservice de commande

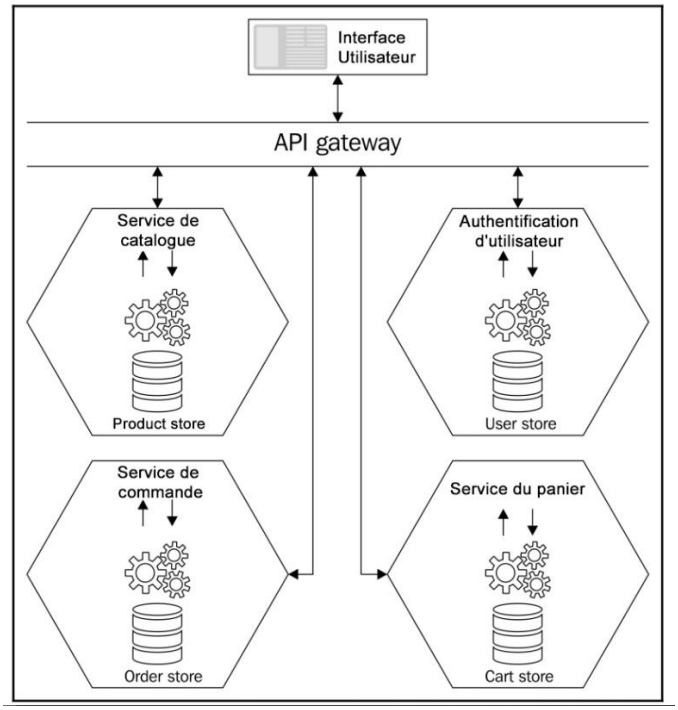
Vient ensuite la fonctionnalité commerciale de passer une commande. Lorsqu'un utilisateur décide que son panier contient les bons produits, il décide de passer une commande. À ce moment, certaines informations relatives à la commande doivent être confirmées/transmises à divers autres microservices. Par exemple, avant que la commande ne soit confirmée, nous devons confirmer à partir du catalogue de produits qu'il y a suffisamment de quantité disponible en stock pour exécuter la commande. Après cette confirmation, le bon nombre d'articles est censé être déduit du catalogue de produits. Le panier devra également être vidé après la confirmation réussie de la commande.

Notre microservice de commande semble plus omniprésent, et il semble être en contradiction avec les règles de ne pas partager de données entre les microservices, mais ce n'est pas le cas, comme nous le verrons bientôt. Toutes les opérations seront terminées, tout en maintenant des limites claires, chaque microservice gérant son propre magasin de données.

7.6.5. Authentification d'utilisateur

Notre dernier candidat pour le service produit est le microservice d'authentification des utilisateurs, qui validerait les informations d'identification des clients qui se connectent à notre application. Le seul but de ce microservice est de confirmer si les informations d'identification fournies sont correctes ou non, de restreindre l'accès non autorisé. Cela semble assez simple pour un microservice; Cependant, nous devons nous rappeler qu'en intégrant cette fonctionnalité à tout autre microservice, cela aurait un impact sur plus d'une fonctionnalité métier si nous décidions de modifier votre mécanisme d'authentification. Le changement peut prendre la forme de l'utilisation de JWT généré et validé, sur la base du framework d'autorisation OAuth 2.0 et de l'authentification OpenID Connect 1.0.

Dans le diagramme suivant, nous on peut visualiser quatre microservices du service client:



7.6.6. Microservice d'enregistrement des clients

Pour que les clients puissent acheter des produits, ils doivent s'inscrire auprès de l'application Foosus. Des données telles que nom, prénom, adresse, e-mail, téléphone portable, adresse de livraison du produit seront nécessaires pour valider la création du compte client. Chaque client sera invité à valider son email et son numéro de téléphone portable.

7.6.7. Microservice d'enregistrement des producteurs

Les producteurs doivent s'inscrire sur la plateforme Fossus afin de pouvoir constituer des stocks de leurs produits et devenir partenaires dans la commercialisation des produits alimentaires. Lors de l'inscription, ils doivent fournir des données sur l'entreprise, y compris des données sur le compte bancaire à utiliser aux fins du transfert de l'argent. Il leur sera également demandé de fournir des données sur le(s) responsable(s) de l'entreprise, sur d'autres personnes de contact et sur les services de livraison des commandes qu'ils peuvent avoir. Chaque producteur sera invité à valider son email et son numéro de téléphone portable.

7.6.8. Microservice de mailing

Ce microservice sera utilisé pour les notifications des clients et des producteurs. Le microservice marketing utilisera également ce microservice pour promouvoir des campagnes marketing.

7.6.9. Microservice de SMS

Le microservice SMS sera principalement utilisé pour confirmer l'authenticité de l'utilisateur. Cependant, il peut également être utilisé pour les notifications et les campagnes marketing.

7.6.10. Microservice du type de produit

Il s'agit d'un petit microservice qui vise à créer une liste de produits autorisés à la vente sur la plateforme Foosus. Cette liste est gérée par les services commerciaux de Foosus.

7.6.11. Microservice d'enregistrement de produit

Avec ce microservice, le producteur peut créer la fiche produit, en introduisant des données qui caractérisent le produit d'une manière spécifique et détaillé (nom, photos, vidéos, taille, couleur, prix, caractéristiques, composition, date de fabrication, période de consommation, informations nutritionnelles, etc.). Ce microservice utilise les données du microservice de type de produit.

7.6.12. Microservice de gestion des stocks

Ce service permet de créer, valider et mettre à jour le stock. A cet effet, il communique avec le service d'enregistrement des produits qui dispose d'informations sur les produits existants.

7.6.13. Microservice de paiements et de remboursements

Grâce à ce service, les producteurs pourront payer les frais d'inscription sur la plateforme Foosus, les clients pourront payer leurs commandes et se faire rembourser. Foosus l'utilise également pour effectuer des paiements aux producteurs et aux transporteurs. Les systèmes de paiement habituels via cartes de débit et de crédit et virement bancaire seront disponibles.

7.6.14. Microservice accord de partenariat

Ce service a pour but de vérifier l'accomplissement de toutes les formalités par le producteur et de générer automatiquement la convention de partenariat à signer avec Foosus.

7.6.15. Microservice de tarification des produits

Ce microservice calcule le prix final des produits, y compris le montant des taxes/impôts et en ajoutant la marge de bénéfice de Foosus.

7.6.16. Microservice de facturation

Le microservice de facturation sert les clients et les producteurs. Pour chaque paiement effectué sur la plateforme Foosus, une facture/un reçu est émis et envoyé à l'email du client/producteur.

7.6.17. Microservice de gestion des commandes

Ce microservice enregistre tout le cycle de vie d'une commande jusqu'à sa livraison. Il est d'une grande utilité pour le service commercial de Foosus, pour ses clients et producteurs. Ce service nous permet d'évaluer les performances à chaque étape du cycle de commande.

7.6.18. Microservice de transport

La gestion du transport se fait avec ce microservice. Lorsque le producteur ne dispose pas d'un service de livraison de produits, avec ce microservice, Foosus contracte les services d'un transporteur, préalablement enregistré dans sa base de données.

7.6.19. Microservice de marketing

Le marketing est essentiel pour que Foosus atteigne ses objectifs. Grâce à ce microservice, Fossus pourra organiser des campagnes marketing automatisées visant les produits promotionnels, les lancements de produits, la fidélisation de la clientèle, etc.

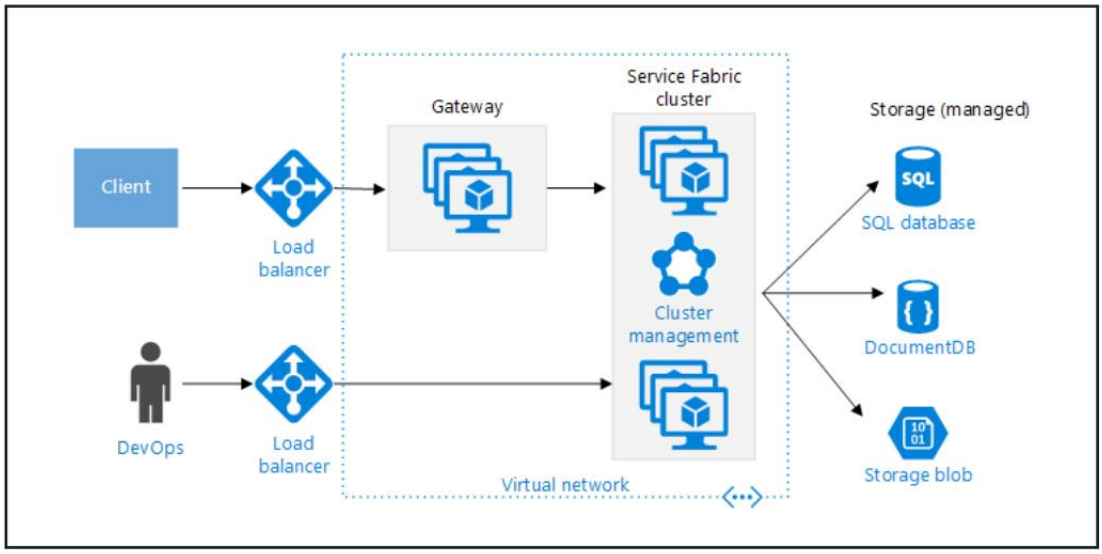
7.6.20. Microservice de finances

Avec ce microservice, Foosus aura une idée claire de sa rentabilité et de la rentabilité de ses producteurs. Il sera également possible de réparer les plus gros consommateurs ou clients de la plateforme.

7.7. Azure Service Fabric

Il s'agit d'une plate-forme qui nous aide à empaqueter, déployer et gérer facilement des microservices évolutifs et fiables (le conteneur est également comme Docker). Parfois, il est difficile de se concentrer sur votre responsabilité principale en tant que développeur, en raison de problèmes d'infrastructure complexes. Avec l'aide d'Azure Service Fabric, les développeurs n'ont plus à se soucier des problèmes d'infrastructure. Azure Service Fabric fournit diverses technologies et se présente sous la forme d'un bundle qui a la puissance d'Azure SQL Database, Cosmos DB, Microsoft Power BI, Azure Event Hubs, Azure IoT Hub et de nombreux autres services de base.

Le diagramme suivant illustre une architecture de microservices avec Azure Service Fabric.



Le cluster Service Fabric est déployé dans un ou plusieurs groupes de machines virtuelles identiques. Vous pouvez inclure plus d'un groupe de machines virtuelles identiques dans le cluster afin de disposer de différents types de machines virtuelles. Une passerelle API est placée devant le cluster Service Fabric, avec un équilibreur de charge externe pour recevoir les requêtes de clients.

Le runtime Service Fabric accomplit les tâches de gestion du cluster, y compris le positionnement des services, le basculement entre les nœuds et la vérification du fonctionnement. Le runtime est déployé sur les nœuds du cluster eux-mêmes. Il n'y a pas d'ensemble distinct de machines virtuelles pour la gestion du cluster.

Les services communiquent entre eux à l'aide du proxy inverse intégré dans Service Fabric. Service Fabric offre un service de découverte capable de résoudre le point de terminaison pour un service nommé.

7.8. Sécurité

L'approche traditionnelle, qui consiste à avoir un seul point d'authentification et d'autorisation, a bien fonctionné dans l'architecture monolithique. Cependant, dans le cas des microservices, vous devez le faire pour chaque service. Cela poserait un défi non seulement pour la mise en œuvre d'un service, mais aussi pour le maintenir synchronisé.

Le cadre d'autorisation OAuth 2.0 et les spécifications OpenID Connect 1.0 combinées peuvent résoudre le problème pour nous. OAuth 2.0 décrit assez bien tous les rôles impliqués dans le processus d'autorisation qui répondent assez bien à nos besoins. Nous devons simplement nous assurer que le bon type de subvention est choisi; sinon, la sécurité sera compromise. L'authentification OpenID Connect repose sur le protocole OAuth 2.0.

Azure Active Directory (Azure AD) est l'un des fournisseurs de spécifications OAuth 2.0 et OpenID Connect. Il est entendu ici qu'Azure AD évolue très bien avec les applications et s'intègre bien à tout Windows Server Active Directory organisationnel.

Il est important et intéressant de comprendre que les conteneurs sont très proches du noyau du système d'exploitation hôte. Les sécuriser est un autre aspect qui ne peut être surestimé. Docker est l'outil que j'ai envisagé et il fournit la sécurité nécessaire en utilisant le principe du moindre privilège.

Azure Active Directory (AAD) est une bonne adaptation pour valider les utilisateurs et authentifier les applications, afin que nous ne nous inquiétions pas de l'authentification des demandes. Mais pour terminer la validation, nous devons prendre quelques étapes supplémentaires lors de l'utilisation d'AAD. Outre la sécurité, la surveillance et la journalisation sont également des aspects importants d'une application.

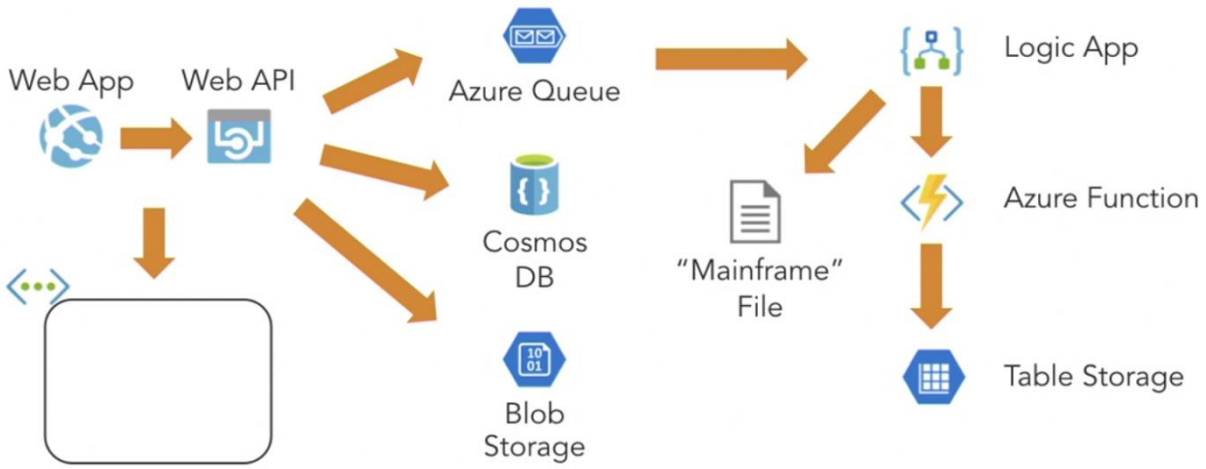
7.9. Surveillance de l'application

Contrairement à une architecture monolithique, la surveillance est très nécessaire, dès le début, dans une architecture basée sur des microservices. Il existe des outils disponibles, tels que AppDynamics et New Relic, qui nous permettront de visualiser les données pour peut-être jusqu'à 100 microservices. Cependant, dans les applications du monde réel, ce n'est qu'une fraction du nombre.

Microsoft vous donne une image complète des performances de notre application Web en fournissant une analyse des journaux, une surveillance des applications et des alertes de sécurité. La meilleure partie est que ces outils sont intégrés à Azure, nous n'avons donc pas à installer de nouveau logiciel. Microsoft Cloud Monitoring est parfait pour les entreprises qui recherchent une solution simple pour surveiller leur pile Microsoft.

SolarWinds dispose d'un outil appelé Virtualization Manager qui fait des recommandations prédictives pour améliorer les performances de votre environnement virtuel. Si vous gérez une infrastructure virtuelle, vous souhaitez vérifier cela. Virtualization Manager facilite également la planification de la capacité et vous alerte en cas de problème.

7.10. Ressources Azure utilisées dans cette application



Le terme *calcul* fait référence au modèle d'hébergement des ressources de calcul utilisées par une application. L'App Service est un service géré pour l'hébergement d'applications web, de serveurs principaux d'applications mobiles, d'API RESTful ou de processus métier automatisés. Azure Functions est une solution serverless qui nous

permet d'écrire moins de code, de maintenir une infrastructure plus légère et de réduire les coûts. Au lieu de nous préoccuper du déploiement et de la maintenance des serveurs, l'infrastructure cloud met à notre disposition tous les serveurs à jour nécessaires pour assurer l'exécution de nos applications.

Du point de vue du calcul, notre application se compose principalement d'une application Web et d'une API Web qui vont être mises à l'échelle indépendamment. De plus, je souhaite que le processus de commande soit disponible non seulement dans l'application, mais également à partir d'autres applications de l'organisation. L'application Web et l'API Web seront hébergées dans Azure App Services et l'application Web pourra communiquer avec l'API Web afin de fonctionner avec les produits et les commandes. Cela me permettra de profiter de la publication simple à partir de Visual Studio ou de passer à un modèle de déploiement continu dans lequel je peux mettre à l'échelle chacun de ces composants.

Le processus de commande principal sera dans Azure Function. Ainsi, il peut être appelé depuis cette application mais aussi depuis d'autres applications de mon organisation selon les besoins. Cela signifie également que la fonction de traitement des commandes peut évoluer indépendamment. Et comme il s'agit d'une fonction, elle peut évoluer selon les besoins en fonction de la demande. L'utilisation d'App Services pour la partie calcul signifie que je dispose d'une évolutivité facile et d'une faible gestion de tous ces composants de mon application qui seront facilement déployés dans le modèle App Service.

Au lieu d'un seul service de stockage de données, j'en utiliserai plusieurs dans cette application car nous avons différents types de données et différentes exigences pour notre stockage. Premièrement, nos données produit sont de nature flexible car nous avons certains produits qui nécessitent une taille et d'autres qui n'en ont pas. Les bases de données de documents comme CosmoDB fournissent une bonne solution pour stocker ce type de contenu. Pour les images de produits, je vais utiliser Azure Blob Storage, car il convient parfaitement au type de données.

Je vais également activer la mise en cache des métadonnées sur les images lorsque je les ai enregistrées afin qu'elles puissent être mises en cache par des intermédiaires ou je peux ajouter la prise en charge CDN plus tard. Un autre avantage de l'utilisation d'Azure Blob est que ces images peuvent être associées aux enregistrements CosmoDB à l'aide de la fonctionnalité d'attachement. Notre historique des commandes sera stocké dans le stockage Azure Table car il fournit l'échelle dont j'ai besoin à faible coût et les données que je stocke sont principalement des données d'écriture qui seront rarement lues. Ces différents modèles de stockage de données peuvent être utili-

sés dans la même application, car ils répondent à des besoins différents dans mes données d'application. Toutes mes données ne doivent pas nécessairement être rassemblées au même endroit.

Notre application va utiliser Logic App pour coordonner le processus de commande. Comme il va nécessiter plusieurs points d'intégration dans les connexions aux systèmes hérités, Logic App sera chargé d'appeler la Fonction Azure pour stocker les informations de commande. Étant donné que les commandes peuvent être traitées de manière asynchrone, l'API Web passe en fait des commandes sur une file d'attente Azure (Azure Queue) qui sera le déclencheur pour lancer le processus de Logic App.

La solution ne nécessite vraiment qu'un réseau de base. Cependant, je vais configurer un réseau virtuel associé à mon App Service. Le fait d'avoir ce réseau en place me permet d'adapter l'application Web à ce réseau. Ensuite, je peux activer les points de terminaison de service pour mon stockage CosmoDB et Azure Blob, me permettant de verrouiller l'accès à ceux-ci uniquement à partir du réseau virtuel.

8. Architecture événementielle (Event-Driven)

Foosus a déjà adopté la notion de gestion et de traitement des événements dans ses limites.

Ce modèle architectural est appliqué par la conception et la mise en œuvre d'applications et de systèmes qui transmettent des événements entre des composants logiciels et des services faiblement couplés. Un système événementiel se compose généralement d'émetteurs d'événements (ou d'agents), de consommateurs d'événements (ou récepteurs) et de canaux d'événements. Les émetteurs ont la responsabilité de détecter, collecter et transférer les événements. Un émetteur d'événement ne connaît pas les consommateurs de l'événement, il ne sait même pas si un consommateur existe, et s'il existe, il ne sait pas comment l'événement est utilisé ou traité ultérieurement.

Un modèle courant qui peut être observé dans de nombreuses implémentations de systèmes événementiels est celui de la duplication des données. Dans un système avec des domaines indépendants, chaque domaine stockera des informations pertinentes pour son fonctionnement. Ces informations peuvent être stockées dans un autre domaine pour d'autres opérations, etc. Cela peut entraîner une augmentation des besoins de stockage en fonction de la quantité de duplication réellement effectuée.

9. Livraison de l'architecture et métriques business

Nous pouvons utiliser un microservice à l'échelle de l'entreprise pour être géré avec la livraison continue (CD) et l'intégration continue (CI). Dès les premières étapes, l'exigence de CI et de CD est si forte que sans eux, l'étape de production ne verra peut-être jamais le jour.

Les conteneurs peuvent répondre aux exigences d'isolation beaucoup plus efficacement que leurs concurrentes les plus proches, les machines virtuelles. Docker et Kubernetes sont les candidats les plus populaires pour la conteneurisation.

Azure DevOps Services, en tant que fournisseur de services logiciels en ligne, offre une cadence plus rapide de livraison de fonctionnalités et permet aux équipes d'éviter de gérer et de s'occuper des serveurs locaux.

Après avoir complété le code, l'application doit être testée dans l'environnement approprié, et cela est possible avec le déploiement; CI / CD fonctionne ici.

Nous utiliserons l'approche mock-and-stub pour rendre les tests indépendants des autres microservices. Cela facilite également les tests avec les bases de données, car nous pouvons également simuler les interactions de bases de données.

La métrique principal repose sur la capacité de Foosus à maintenir un taux positif d'inscriptions de nouveaux utilisateurs.

9.1. Indicateurs de réussite

Indicateur	Changement souhaité pour l'indicateur
Nombre d'adhésions d'utilisateurs par jour	Augmentation de 10 %
Adhésion de producteurs alimentaires	Passer de 1,4/mois à 4/mois
Délai moyen de parution	Réduit de 3,5 semaines à moins d'une semaine
Taux d'incidents de production P1	Pour commencer : réduit de >25/mois à moins de 1/mois.

Amélioration de la capacité de re-cherche	Premier trimestre

L'analyse joue également un rôle important en aidant à déterminer la ou les mesures à prendre, des aspects simples tels que les rapports d'inventaire aux domaines plus complexes tels que la demande de stock et l'analyse de la rotation.

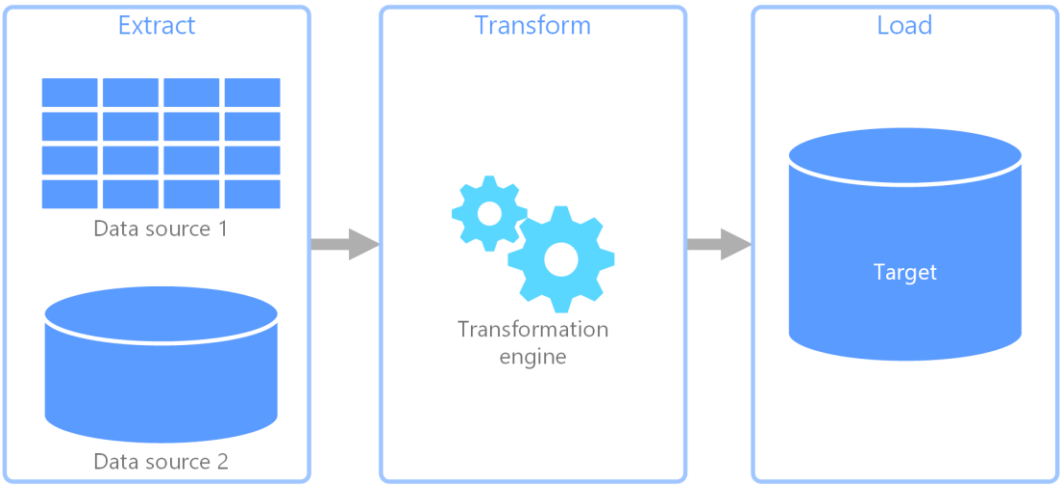
10. Architecture des données

Le cloud change la façon dont les applications sont conçues, y compris la façon dont les données sont traitées et stockées. Au lieu d'une seule base de données à usage général qui gère toutes les données d'une solution, les solutions de persistance polyglotte utilisent plusieurs banques de données spécialisées, chacune optimisée pour fournir des fonctions spécifiques. La perspective sur les données de la solution est par conséquent modifiée. Il ne s'agit plus de plusieurs couches de logique métier qui lisent et écrivent sur une couche de données unique. Les solutions sont plutôt conçues autour d'un pipeline de données qui décrit le flux des données via une solution, l'endroit où elles sont traitées, stockées et comment elles sont consommées par le composant suivant dans le pipeline.

10.1. Processus ETL (extraction, transformation et chargement)

ETL est un pipeline de données utilisé pour collecter des données provenant de différentes sources, transformer les données en fonction des règles métier et charger les données dans un magasin de données de destination. Le travail de transformation dans ETL a lieu dans un moteur spécialisé et implique souvent l'utilisation de tables intermédiaires pour conserver temporairement les données lors de leur transformation et leur chargement final vers leur destination.

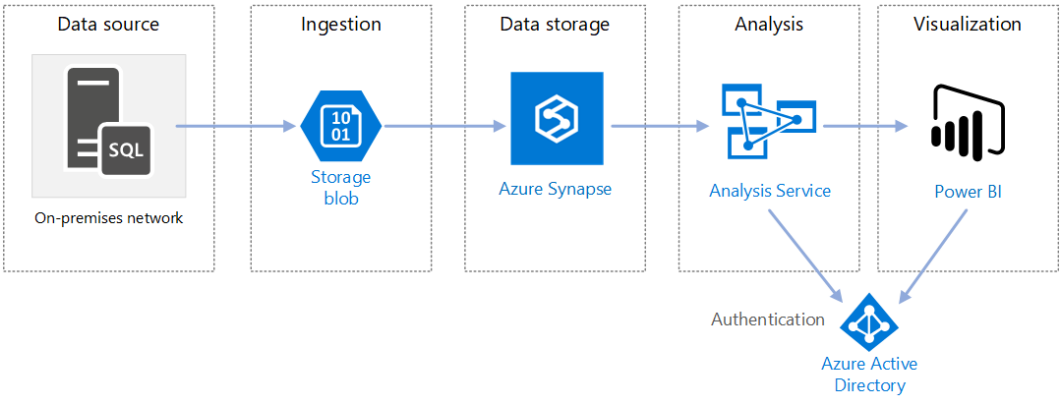
La transformation des données qui a lieu implique généralement plusieurs opérations, comme le filtrage, le tri, l'agrégation, la jointure des données, le nettoyage des données, la déduplication et la validation des données.



Souvent, les trois phases ETL sont exécutées en parallèle pour gagner du temps. Par exemple, tandis que les données sont extraites, un processus de transformation peut travailler sur les données déjà reçues et les préparer pour le chargement, et un processus de chargement peut commencer à travailler sur les données préparées, au lieu d'attendre la fin du processus d'extraction complet.

10.2. Entrepôt de données

Un entrepôt de données est un référentiel central de données intégrées provenant d'une ou plusieurs sources hétérogènes. Les entrepôts de données stockent des données historiques et actuelles, et permettent de créer des rapports et d'analyser des données.



Les données à déplacer dans un entrepôt de données sont régulièrement extraites de différentes sources qui contiennent d'importantes informations d'entreprise. Durant leur déplacement, les données peuvent être mises en forme, nettoyées, validées, synthétisées et réorganisées. Elles peuvent également être stockées au plus bas niveau de détail, avec des vues agrégées fournies dans l'entrepôt pour la création de rapports. Dans les deux cas, l'entrepôt de données devient un magasin de données

permanent pour la création de rapports, l'analyse et l'analyse décisionnelle (BI).

10.3. Architectures d'entrepôt de données

Les architectures de référence suivantes illustrent des architectures de l'entrepôt de données de bout en bout sur Azure :

- ***Décisionnel d'entreprise dans Azure avec Azure Synapse Analytics***. Cette architecture de référence implémente un pipeline ELT (extract/extraire), load/charger, transform/transformer) qui déplace des données d'une base de données SQL Server locale vers Azure Synapse.

- ***Décisionnel d'entreprise automatisé avec Azure Synapse et Azure Data Factory***. Cette architecture de référence montre un pipeline ELT avec un chargement incrémentiel, automatisé à l'aide d'Azure Data Factory.

10.4. Pourquoi utiliser cette solution ?

Foosus doit adopter un entrepôt de données parce qu'elle a besoin de convertir un grand nombre de données dans un format facile à comprendre.

Elle a aussi besoin de conserver les données d'historique dans un emplacement autre que les systèmes transactionnels sources pour des raisons de performances. Les entrepôts de données facilitent l'accès aux données d'historique à partir de plusieurs emplacements, en fournissant un emplacement centralisé utilisant des formats, des clés et des modèles de données courants.

Les entrepôts de données ne doivent pas forcément suivre la même structure de données sobre que celle que vous pouvez utiliser dans vos bases de données OLTP. Vous pouvez utiliser des noms de colonne pertinents pour les utilisateurs professionnels et les analystes, restructurer le schéma pour simplifier les relations entre les données, et consolider plusieurs tables en une seule. Ces étapes guident les utilisateurs qui ont besoin de créer des rapports et d'analyser les données dans des systèmes de BI, sans l'aide d'un administrateur de base de données (DBA) ou d'un développeur de données.

Étant donné que les entrepôts de données sont optimisés pour l'accès en lecture, la génération de rapports est plus rapide qu'avec le système de transactions source.

Voici d'autres avantages :

- Un entrepôt de données peut consolider des données provenant de différents logiciels.
- Les entrepôts de données facilitent la mise en place d'un accès sécurisé aux utilisateurs autorisés, tout en limitant l'accès aux autres utilisateurs. Les utilisateurs professionnels n'ayant pas besoin d'accéder aux données sources, cela supprimer un vecteur d'attaque potentiel.
- Les entrepôts de données facilitent la création de solutions d'aide à la décision, telles que des cubes OLAP.

10.5. Entreposage de données dans Azure

Vous pouvez avoir une ou plusieurs sources de données, qu'il s'agisse de transactions client ou d'applications professionnelles. Ces données sont généralement stockées dans une ou plusieurs bases de données OLTP. Les données peuvent être persistantes dans d'autres supports de stockage tels que des partages réseau, des objets blob de stockage Azure ou un Data Lake. Elles peuvent également être stockées dans l'entrepôt de données lui-même ou dans une base de données relationnelle comme Azure SQL Database. L'objectif de la couche du magasin de données analytique est de satisfaire les requêtes émises par les outils d'analyse et de création de rapports au niveau de l'entrepôt de données. Dans Azure, cette fonctionnalité de magasin analytique est disponible avec Azure Synapse, ou avec HDInsight Azure en utilisant une requête Hive ou interactive. Vous avez également besoin d'un certain niveau d'orchestration pour déplacer ou copier des données du stockage de données vers l'entrepôt de données, ce qui peut être effectué à l'aide d'Azure Data Factory ou d'Oozie sur Azure HDInsight.

11. Phases de livraison définies

Livrables architecturaux qui satisfont aux conditions requises pour le business.

11.1. Principales étapes du processus métier

1. **Processus d'approvisionnement**: acquisition d'équipements et de services, préparation des prestataires de services et coordination des expéditions.
2. **Tarification:** identification de nouveaux producteurs locaux potentiels, renouvèlement des contrats existants et signature de nouveaux contrats avec

des producteurs. Dans cette étape les spécialistes parcourront diverses sources pour déterminer le meilleur prix du marché pour chaque article.

3. **Inventaire** : Un nouvel ajout d'inventaire est créé pour chaque nouveau produit, ce qui obligera l'équipe chargée de l'inventaire à télécharger l'image sur le système, à saisir les détails pertinents du pack d'actifs et à marquer l'article comme prêt pour le stock mais pas généralement disponible à l'achat.

4. **Stockage**: Chaque producteur met à jour les données sur les quantités de produits qu'il a en stock disponibles à la vente.

5. **Service Clients**: Au fur et à mesure que les clients recherchent des produits en ligne, les commandes sont rassemblées et les vendeurs répondent aux besoins des clients.

6. **Livraison des commandes**: Les services de transport des producteurs ou d'un prestataire livrent les commandes.

11.2. Plan de travail commun priorisé

Cette section décrit toutes les activités et tous les livrables pour le travail d'architecture.

Fournir un plan pour le travail d'architecture.

Item de travail n°1 :		
Nom du Produit	**Activités**	**Livrables**
Abonnement Microsoft Azure	1. Abonnement au service cloud hybride Microsoft Azure. 2. Étendre le réseau local à l'aide d'un VPN. 3. Configuration du réseau virtuel Azure. 4. Configuration des services Appliance VPN et Passerelle VPN Azure. 5. Configuration de la sécurité du réseaux virtuel (Firewall, HTTPS).	Abonnement au service Microsoft Azure IaaS hybride.

Item de travail n°2 :		
Nom du Produit	**Activités**	**Livrables**
Entrepôt de données Foosus	1. Création de l'entrepôt de données Foosus sur Microsoft Azure. 2. Élaboration des vues agrégées pour la création de rapports, l'analyse et l'analyse décisionnelle (BI). 3. Activation des services de sauvegardes et restaurations de base de données. 4. Activation du service de cryptage des données.	Création de l'entrepôt de données Foosus
Item de travail n°3 :		
Nom du Produit	**Activités**	**Livrables**
Services de surveillance de l'application	1. Activation des services de surveillance de l'application	Services de surveillance de l'application.
Item de travail n°4 :		
Nom du Produit	**Activités**	**Livrables**
Services de Azure Fabric et DevOps.	1. Activation et configuration du service Azure Fabric. 2. Activation et configuration du service Azure DevOps.	Services de Azure Fabric et DevOps.
Item de travail n°5 :		
Nom du Produit	**Activités**	**Livrables**
Services de sécurité de l'application.	1. Activation et configuration d'Azure Active Directory. 2. Configuration du service de contrôle d'accès basé sur les rôles (RBAC). 3. Activation et configuration du Suite de Management d'Operations et du Centre de	Services de sécurité de l'application.

	Sécurité d'Azure.	
Item de travail n°6 :		
Nom du Produit	**Activités**	**Livrables**
Microservice d'enregistrement des producteurs	1. Création du microservice d'enregistrement des producteurs	Microservice d'enregistrement des producteurs
Item de travail n°7 :		
Nom du Produit	**Activités**	**Livrables**
Microservice du type de produit	1. Création du microservice du type de produit.	Microservice du type de produit
Item de travail n°8 :		
Nom du Produit	**Activités**	**Livrables**
Microservice d'enregistrement de produit.	1. Création du microservice d'enregistrement de produit.	Microservice d'enregistrement de produit.
Item de travail n°9 :		
Nom du Produit	**Activités**	**Livrables**
Microservice tarification des produits.	1. Création du microservice tarification des produits.	Microservice tarification des produits.
Item de travail n°10 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de mailing.	1. Création du microservice de mailing.	Microservice de mailing.
Item de travail n°11 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de SMS.	1. Création du microservice de SMS.	Microservice de SMS.
Item de travail n°12 :		

Nom du Produit	Activités	Livrables
Microservice accord de partenariat.	1. Création du microservice accord de partenariat.	Microservice accord de partenariat.
Item de travail n°13 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de paiements et de remboursements.	1. Création du microservice de paiements et de remboursements.	Microservice de paiements et de remboursements.
Item de travail n°14:		
Nom du Produit	**Activités**	**Livrables**
Microservice d'enregistrement des clients.	1. Création du microservice d'enregistrement des clients.	Microservice d'enregistrement des clients.
Item de travail n°15 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de gestion des stocks.	1. Création du microservice de gestion des stocks.	Microservice de gestion des stocks.
Item de travail n°16 :		
Nom du Produit	**Activités**	**Livrables**
Microservice pour lister des produits.	1. Création du microservice pour lister des produits.	Microservice pour lister des produits.
Item de travail n°17 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de recherche de produits.	1. Création du microservice de recherche de produits.	Microservice de recherche de produits.
Item de travail n°18 :		
Nom du Produit	**Activités**	**Livrables**

Microservice authentification d'utilisateur.	1. Création du microservice authentification d'utilisateur.	Microservice authentification d'utilisateur.
Item de travail n°19 :		
Nom du Produit	**Activités**	**Livrables**
Microservice du panier.	1. Création du microservice du panier.	Microservice du panier.
Item de travail n°20 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de commande.	1. Création du microservice de commande.	Microservice de commande.
Item de travail n°21 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de facturation.	1. Création du microservice de facturation.	Microservice de facturation.
Item de travail n°22 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de gestion des commandes.	1. Création du microservice de gestion des commandes.	Microservice de gestion des commandes.
Item de travail n°23 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de transport.	1. Création du microservice de transport.	Microservice de transport.
Item de travail n°24 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de finances.	1. Création du microservice de finances.	Microservice de finances.

Item de travail n°25 :		
Nom du Produit	**Activités**	**Livrables**
Microservice de marketing.	1. Création du microservice de marketing.	Microservice de marketing.

Observation *: Au moment de la création de chaque microservice, sa base de données et l'API de communication avec d'autres microservices sont également créés. Une mise à l'échelle de l'infrastructure est également réalisée afin d'optimiser les performances de l'ensemble de l'application.*

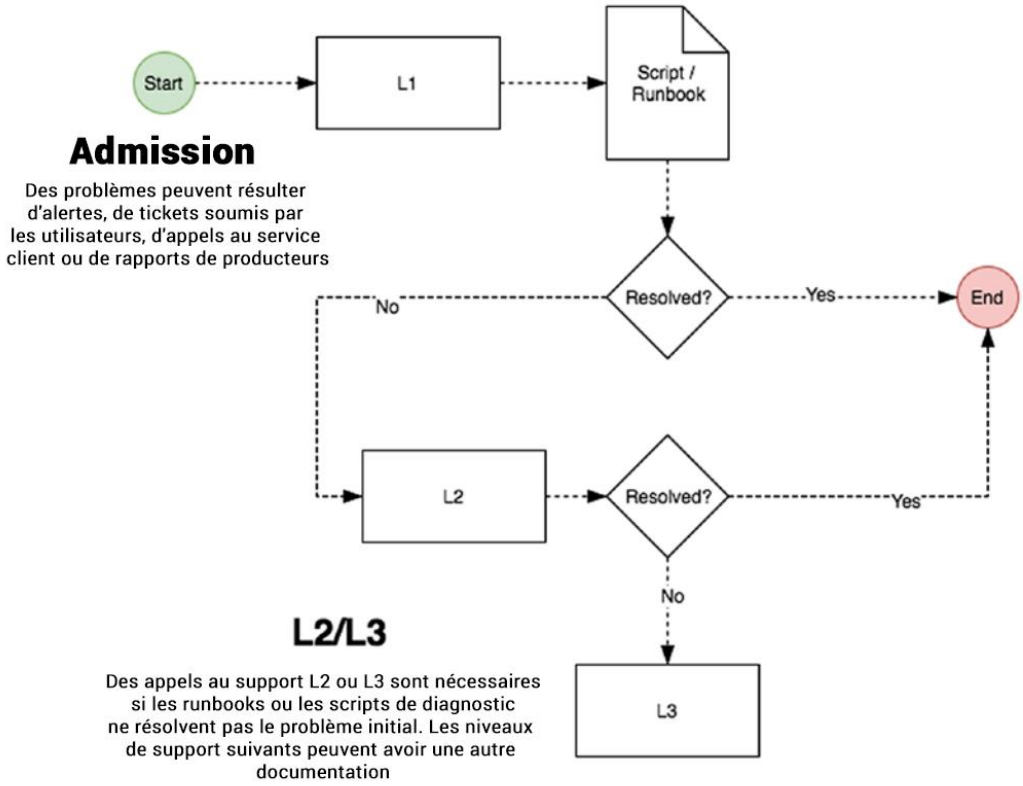
12. Plan de communication

12.1. Évènements

12.1.1. Réponse standardisée

Disposer d'une méthode standard pour la réception, le triage et la résolution des problèmes peut considérablement rationaliser tout effort d'assistance. Tout aussi importante est la notion selon laquelle une structure logique et bien définie d'escalade devrait être en place pour faciliter la résolution des problèmes.

Le processus d'admission est déclenché lorsqu'un événement se produit, que ce soit via une alerte, un ticket soumis par l'utilisateur, un appel d'un client ou un fournisseur signalant un problème. Les alertes sont déclenchées en fonction de plusieurs points, notamment les erreurs liées aux applications, les mesures de performances, la surveillance standard de montée/descente des serveurs et les limites d'espace disque atteintes. La figure suivante illustre les points d'admission, de triage et d'escalade pour tout problème entrant.



La suite de produits de Management des Operations proposée par Microsoft permet la surveillance traditionnelle des ressources, ainsi que la gestion des stocks, la gestion des mises à jour (correctifs) et l'analyse des journaux par le biais de l'agrégation de journaux à partir de plusieurs systèmes d'exploitation et de plusieurs sources, y compris le journal personnalisé appelée ingestion. Azure Monitor ajoute également à ces fonctionnalités en permettant la capture des alertes de manière centralisée et le déploiement de résolutions automatisées en fonction de ces alertes, à l'aide de Logic Apps, de runbooks Azure Automation, etc.

12.2. Canaux

Un aspect qui n'apparaît pas immédiatement est celui de la communication. À chaque étape du processus, une communication est envoyée à toutes les parties impliquées pour les informer de l'état d'avancement du processus. Alors que la plupart des communications sont gérées directement par le système de billetterie, d'autres points de communication sont réalisés par contact direct avec l'utilisateur ou le producteur concerné. Les contacts avec les membres de l'équipe de support sont gérés par l'utilisation de PagerDuty, une application qui regroupe des informations et envoie des appels téléphoniques, des e-mails, des notifications push et/ou des messages SMS aux membres d'une équipe préétablie.

De plus, PagerDuty s'intègre très bien avec les produits Atlassian, notamment avec Jira Software qui sera utilisé par toute l'équipe Foosus, et avec Microsoft Azure.

En ce qui concerne l'intégration des logs, la plateforme Azure offre de bonnes options, notamment des alertes et surveillance.

12.3. Rythme de communication

Sachant que Foosus est une organisation Agile, le rythme de communication doit être quotidien et permanent.

13. Risques et facteurs de réduction

13.1. Structure de gouvernance

13.1.1. Domaine Business

Responsabilités	Rôles clés	Fonctions clés
Gestion des risques et remédiation: l'identification et la correction de tout risque pour l'entreprise, que ce soit pour le secteur d'activité ou pour l'entreprise	1. Natasha Jarson, CIO. 2. Daniel Anthony, CPO. 3. Pete Parker, Responsable Ingénierie. 4. Jack Harkner, Responsable des opérations. 5. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Administrateurs de base de données).	• Construction de solutions en utilisant les meilleures pratiques et les normes de conformité. • Suivi de toutes les opérations au sein de la plateforme. • S'assurer que les utilisateurs internes adhèrent aux normes d'intégrité et de conformité les plus élevées.
Achats: les transactions financières associées à l'achat de logiciels, de technologies ou de services de plate-forme auprès d'un fournisseur de cloud ou de technologie préféré.	1. Jo Kumar, CFO. 2. Équipe des finances. 3. Natasha Jarson, CIO.	• Exécuter des contrats ou des bons de commande pour obtenir des tarifs préférentiels pour les technologies et les services. • Assurer l'utilisation de

		fournisseurs approuvés.
Licences et gouvernance de la plate-forme: l'établissement d'accords commerciaux avec les fournisseurs de technologie et la gouvernance de ces processus.	1. Natasha Jarson, CIO. 2. Daniel Anthony, CPO.	• Négocier des rabais et des tarifs de service préférentiels. • Assurer la conformité des licences avec toutes les technologies, logiciels et services.
Gestion de la relation entre Foosus et Microsoft Azure et avec autres types de fournisseurs.	1. Natasha Jarson, CIO. 2. Pete Parker, Responsable Ingénierie. 3. Jack Harkner, Responsable des opérations.	• Établir des relations avec les fournisseurs de services préférés. • Maintenir les contacts et les négociations pour assurer une utilisation optimale. • Veiller à ce que les offres des fournisseurs continuent à apporter des avantages significatifs à l'entreprise.

13.1.2. Domaine d'application

Responsabilités	Rôles clés	Fonctions clés
Gestion du cycle de vie: gestion du cycle de vie de l'application, de la conception à la livraison	1. Natasha Jarson, CIO. 2. Pete Parker, Responsable Ingénierie. 3. Jack Harkner, Responsable des opérations. 4. Architectes logiciels 5. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Ingénieurs qualité).	• Définition des étapes du cycle de vie - Mappage des rôles et des responsabilités à chaque étape. • Suivi des éléments de travail / planification du travail. • Établissement de métriques, définition de prêt, définition de terminé. • Portes de qualité. • Test (unité, intégration, fonctionnel, fumée, charge, performance). • Processus de création et de publication de composants d'application.

		• Surveillance / télémétrie des applications. • Gestion des incidents (accueil, triage, résolution).
Conception d'application: acte collaboratif de conception de comportements techniques et fonctionnels dans une application, aboutissant à une ou plusieurs fonctionnalités du produit,	1. Natasha Jarson, CIO. 2. Pete Parker, Responsable Ingénierie. 3. Jack Harkner, Responsable des opérations. 4. Architectes logiciels 5. Équipe de Développement (Ingénieurs logiciels, Web-designers). 6. Daniel Anthony, CPO.	• Analyse des exigences et / ou documentation des spécifications (fonctionnelles et techniques). • Intégration des composants applicatifs • Accords de niveau de service pour le traitement des composants et l'achèvement des transactions. • Créer un aperçu de la conception technique qui permettra à la fonctionnalité demandée de livrer avec succès de la valeur.
Gestion de la plate-forme d'application: gestion d'une application, d'une suite d'applications ou d'une suite de composants comprenant une plate-forme fonctionnelle plus étendue.	Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs logiciels).	• Capture de télémétrie par composant. • Capture des performances par composant. • Capture de télémétrie par transaction de valeur (mission critique).
Gestion de la capacité du produit: gestion de la capacité d'une application logicielle (produit) à gérer le trafic utilisateur et la charge de calcul.	1. Architectes logiciels 2. Équipe de Développement (Ingénieurs logiciels, Ingénieurs en fiabilité). 3. Pete Parker, Responsable Ingénierie. 4. Jack Harkner, Responsable des opérations.	• Analyse du recensement des utilisateurs clients. • Analyse des demandes de fonctionnalités. • Analyser et atténuer les goulots d'étranglement des performances.
Planification de la capacité: la planification proactive et	1. Architectes logiciels 2. Équipe de Développement	• Prévision de la capacité en fonction de la tendance

l'anticipation des changements dans l'utilisation des applications.	(Ingénieurs logiciels, Ingénieurs en fiabilité, Ingénieurs DevOps). 3. Pete Parker, Responsable Ingénierie. 4. Jack Harkner, Responsable des opérations.	d'utilisation et des données de surveillance des performances. • Analyse de la clientèle nouvelle et existante pour les changements dans le recensement des utilisateurs.

13.1.3. Domaine d'infrastructure

Responsabilités	Rôles clés	Fonctions clés
Conception de l'infrastructure d'application: la conception de tous les fondements d'application qui permettent la fonctionnalité, y compris les composants PaaS et/ou IaaS.	1. Architectes logiciels/systèmes. 2. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Architectes Cloud).	• Sélection des composants en fonction des exigences de l'application. • Respect des principaux facteurs d'utilisation des composants (temps, coût, effort, fiabilité, maintenabilité, opérabilité).
Gestion de l'infrastructure d'application: la surveillance, l'approvisionnement, la maintenance et la mise à jour de toute infrastructure liée aux applications.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps).	• Correctif des composants (IaaS, si vous utilisez OMS). • Surveillance des composants. • Création de workflows ou de politiques pour le provisionnement des ressources.
Patch au niveau de la plate-forme: correctif des composants de base de la plate-forme, tels que les systèmes d'exploitation. Applicable aux composants IaaS.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps).	• Planification de l'acquisition et du déploiement des correctifs. • Application (ou restauration) de correctifs pendant les fenêtres de maintenance appropriées. • Sélection du fournisseur de correctifs (Corporate IT ou

		OMS).
Gestion et optimisation de la capacité de la plate-forme: la gestion, la surveillance et la planification associées à la garantie que le parc cloud de l'entreprise est en mesure de répondre aux besoins de l'entreprise.	1. Architectes logiciels/systèmes. 2. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Architectes Cloud).	• Établir les besoins d'abonnement et de ressources pour chaque LOB. • Planifier et prévoir la croissance du patrimoine par secteur d'activité et par entreprise. • Le cas échéant, analyser et consolider les pools de ressources. • Assurez-vous que les blocs CIDR applicables sont disponibles pour l'appairage. • Demander de nouveaux blocs CIDR si nécessaire.

13.1.4. Domaine technologique

Responsabilités	Rôles clés	Fonctions clés
Outils de plate-forme: la conservation ou l'approvisionnement de tout produit technique qui offre des ensembles d'outils à des collègues dans le but d'interagir avec une plate-forme cloud.	1. Architectes Logiciels/Systèmes/Cloud. 2. Équipe de Développement (Opérations cloud).	• Évangélisation de plate-forme. • Création de produits personnalisés ou évaluation de produits tiers destinés à rationaliser les opérations cloud.
Allocation d'abonnements: action de répondre à une demande de création d'un ou plusieurs abonnements Azure sous l'inscription d'entreprise.	1. Architectes Logiciels/Systèmes/Cloud. 2. Équipe de Développement (Opérations cloud). 3. Pete Parker, Responsable Ingénierie. 4. Jack Harkner, Responsable des opérations.	• Choix de l'offre de plateforme qui correspond le mieux aux besoins de l'entreprise. • Évangélisation de la plateforme préférée aux parties prenantes. • Mise à disposition de nouveaux abonnements dans le cadre de l'inscription

		d'entreprise. • Fourniture des exigences de base (politique, réseau, etc.).
Support de plate-forme de première ligne: action de fournir une assistance aux consommateurs de la plate-forme cloud en cas d'obstacles techniques, d'erreurs de plate-forme ou de défaillances de la plate-forme.	1. Équipe de Développement (Opérations cloud, Équipe de support).	• Dépannage des fonctionnalités réseau. • Dépannage d'ExpressRoute. • Intégrations avec les produits du fournisseur de choix (par exemple, Azure et Azure DevOps Services). • Dysfonctionnements des composants de la plate-forme.
Réponse à un incident majeur: toute réponse à un événement qui compromet les intérêts des parties prenantes ou des actionnaires et implique des conseils réglementaires ou juridiques.	1. Daniel Anthony, CPO. 2. Département légal.	• Exposition et atténuation des risques. • Notification des autorités compétentes.
Conception de la plate-forme: tout travail de conception lié à l'exploitation ou à la maintenance de la plate-forme, dont d'autres seront les consommateurs.	1. Équipe de Développement (Opérations cloud). 2. Architectes Logi-ciels/Systèmes/Cloud.	• Cas d'utilisation des composants de plate-forme. • Cas d'utilisation IaaS. • Conseils et normes de mise en réseau.
Planification et exécution de la reprise après sinistre: action de planifier et de tester avec succès le plan de reprise après sinistre d'une application.	1. Équipe de Développement (Ingénieurs DevOps).	• Planifier la construction pour DR (runbooks, procédures). • Mise à disposition des ressources cibles. • Simulation du basculement des ressources existantes vers les ressources de sauvegarde.
Mises à niveau et correctifs: l'acquisition et l'application de	1. Pete Parker, Responsable	• Organisation d'équipe.

mises à niveau de composants ou de correctifs de sécurité pertinents pour l'application LOB	Ingénierie.	• Gestion des demandes.
Support de plate-forme de deuxième / troisième ligne: action de fournir un soutien aux opérations commerciales, aux parties prenantes ou aux utilisateurs finaux en cas d'erreur liée au code de l'application, à une mauvaise configuration ou à une mauvaise utilisation.	1. Architectes logiciels/systèmes. 2. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs logiciel).	• Dépannage d'un dysfonctionnement du Web, du service, du middleware ou de la base de données. • Dépannage d'une configuration incorrecte au niveau des composants de l'application. • Dépannage des erreurs liées au code. • Prise en charge, triage et résolution des incidents.
Approvisionnement du système d'exploitation: la création de systèmes d'exploitation de machines virtuelles conformément aux normes établies par Corporate IT.	1. Architectes logiciels/systèmes. 2. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs logiciel, Ingénieurs DevOps).	• Provisionnement du système d'exploitation versionné. • Application des correctifs, paramètres, politiques et logiciels requis. • Placement de l'objet ordinateur dans l'unité organisationnelle appropriée (si vous utilisez Active Directory). • Établissement de la responsabilité des correctifs du système d'exploitation.
Prise en charge de l'environnement: la maintenance des environnements de déploiement logiques, en fonction des exigences de l'application et du secteur d'activité.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs logiciel, Ingénieurs DevOps, Ingénieurs qualité).	• Mise à disposition de nouveaux environnements • Dépannage des erreurs de construction ou de publication • Dépannage des problèmes de performances de l'environnement • Gestion de l'utilisation (cycles de mise sous / hors tension)

Ingénierie et livraison de plate-forme: le fait de travailler avec la plate-forme cloud pour créer et fournir une solution en utilisant les composants disponibles.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs logiciel, Ingénieurs DevOps). 2. Architectes logiciels/systèmes.	• Évaluer les composants pour déterminer la meilleure option fonctionnelle. • Orchestration de la construction et du déploiement de logiciels sur des composants sélectionnés.
Gestion du réseau et du DNS: l'approvisionnement, l'utilisation et la maintenance des composants réseau requis au sein de la plate-forme.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Opérations Cloud).	• Mise à disposition de réseaux virtuels. • Mise à disposition de groupes de sécurité réseau. • Mise à disposition d'enregistrements DNS pour des objets internes ou externes. • Approvisionnement des circuits ExpressRoute. • Établissement de l'homologation de réseaux virtuels privés et publics.
Gestion de la configuration: gestion des paramètres de configuration des applications, des logiciels ou de la plate-forme cloud.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Opérations Cloud, Ingénieurs logiciel).	• Ajout de composants aux réseaux virtuels approuvés. • Appairage de plages d'adresses IP privées à des circuits ExpressRoute. • Établissement de chaînes de connexion à la base de données. • Installation des fonctionnalités au niveau du système d'exploitation nécessaires pour la fonctionnalité de l'application (par exemple, Internet Information Services). • Utilisation de la configuration comme code (par exemple, configuration

		de l'état souhaité).
Assistance à la production de première ligne: responsabilité partagée de la gestion des problèmes de production pouvant être liés à des défaillances de la plate-forme cloud ou de la plate-forme d'application.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Opérations IT, Opérations Cloud, Ingénieurs logiciel, Administrateurs de base de données).	• Prise et triage des pannes ou rapport d'erreur. • Acheminement des incidents vers l'équipe de première ligne appropriée. • Communication avec les parties prenantes, le cas échéant. • Résolution d'incident ou escalade.

13.1.5. Domaine de données

Responsabilités	Rôles clés	Fonctions clés
Sauvegardes de base de données: action de lancer une sauvegarde d'une base de données d'application ou d'exploiter un composant de plate-forme qui effectue la même opération.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Administrateurs de base de données).	• Déclencher une opération de sauvegarde au niveau du serveur de base de données. • Stockage des actifs de sauvegarde. • Définition et respect des politiques de rétention.
Restaurations de base de données: restauration d'une base de données d'application à un moment précis dans le temps.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Administrateurs de base de données).	• Restauration d'une sauvegarde de base de données en production. • Restauration d'une sauvegarde de base de données sur un serveur hors production.
Approvisionnement de la base de données d'application: création de nouvelles instances de base de données d'application.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Administrateurs de base de données).	• Création de base de données. • Dénomination de la base de données. • Allocation des ressources

		(cœurs, CPU, DTU).
Constructions / modifications / versions de la base de données d'application: gestion des modifications apportées aux données ou au schéma d'une application et à l'avancement de ces modifications du développement à la production.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Administrateurs de base de données).	• Création, maintenance et publication de scripts de migration. • La gestion du changement. • Test et validation des modifications.
Conception de base de données d'application: action de concevoir un schéma et un objet de données en corrélation avec les fonctionnalités de l'application.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs logiciel, Administrateurs de base de données).	• Conception d'objets ou d'entités de données. • Désignation des relations entre entités. • Authentification, autorisation et portée. • Conception de schéma.
Maintenance et réglage de la base de données: exécution de travaux de maintenance, manuels ou automatiques, qui maintiennent les opérations de la base de données en ligne avec les mesures de performance attendues.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs Logiciel, Administrateurs de base de données). 2. Architectes logiciels.	• Indexation de table. • Réglage de la procédure stockée. • Afficher le réglage. • Revue et optimisation du plan d'exécution. • Réduction du fichier de données.
Surveillance de la base de données d'application: observation des performances et de l'utilisation de la base de données par des moyens manuels ou automatiques.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs logiciel, Administrateurs de base de données).	• Définition d'alertes sur les valeurs de seuil de performance (CPU élevé, E / S élevées). • Collecte de la télémétrie sur les temps d'exécution des requêtes, des procédures stockées, des fonctions. • Assurer la disponibilité grâce à l'utilisation de rapports d'agent.

13.1.6. Domaine de sécurité

Responsabilités	Rôles clés	Fonctions clés
Mise en œuvre et validation du contrôle de sécurité: Compréhension et mise en œuvre des contrôles de sécurité prescrits par le plus grand organe directeur de l'entreprise, dans le but de sécuriser les données de l'utilisateur final et d'atténuer les risques pour l'entreprise.	1. Équipe de Développement (Ingénieurs en fiabilité, Ingénieurs DevOps, Ingénieurs logiciels). 2. Pete Parker, Responsable Ingénierie.	• Utilisation du cryptage au repos, en transit. • Fournir une autorisation de niveau application étendue aux utilisateurs administratifs. • Demander et mettre en œuvre des comptes de service. • Demander et mettre en place des comptes administratifs. • Adhésion à l'architecture de référence de sécurité.
Opérations de sécurité: examen et jugement de toute violation des politiques ou contrôles de sécurité au niveau de l'entreprise.	1. Équipe de Sécurité (Audit et conformité, Analystes Info-Sec, Managers InfoSec). 2. Architecte Système.	• Déterminer les arbres de décision et les politiques d'escalade en cas de violation. • Examiner régulièrement les exigences de sécurité de l'entreprise en conjonction avec toutes les obligations juridiques nationales et internationales.
Administration et surveillance des politiques: application et audit des politiques et normes de la plate-forme cloud d'entreprise.	1. Architecte Cloud 2. Cloud Operations 3. Analystes InfoSec.	
Politique et normes de la plate-forme: L'établissement de politiques et de normes qui régissent l'utilisation et la configuration des composants de la plate-forme cloud au sein	1. Architecte Cloud. 2. Architecte Système. 3. Audit et conformité.	• Rédiger, collaborer et publier des politiques de plate-forme cloud. • S'assurer que les politiques respectent toutes les exigences légales et de conformité au nom du

du parc cloud de l'entreprise.		cabinet. • S'assurer que les normes sont conformes aux meilleures pratiques déterminées par l'industrie ainsi que par le fournisseur de la plate-forme.

14. Analyse des risques

ID	Risque	Gravité	Probabilité	Facteur de réduction	Propriétaire
1	Confidentialité	4	1	1. Registre des risques. 2. Utilisation du Suite de Management d'Operations et du Centre de Sécurité d'Azure. 3. Microsoft Azure. 4. Audits.	Équipe de développement.
2	Intégrité	4	1	1. Registre des risques. 2. Microsoft Azure. 3. Audits.	Équipe de développement.
3	Disponibilité	4	1	1. Registre des risques. 2. Utilisation du Suite de Management d'Operations et du Centre de Sécurité d'Azure. 3. Microsoft Azure. 4. Audits.	Équipe de développement.

4	Vol d'identité	3	1	1. Azure AD. 2. Contrôle d'accès basé sur les rôles (RBAC). 3. Migration vers Azure Active Directory Business to Consumer. 4. Utilisation du Suite de Management d'Operations et du Centre de Sécurité d'Azure. 5. Registre des risques. 6. Audits.	Équipe de développement.
	Canaux de transport et de livraison	4	1	1. Cryptographie ou cryptage (HTTPS). 2. Verrouillages sur un certain nombre de tentatives de connexion infructueuses consécutives. 3. Registre des risques. 4. Infrastructure réseau Azure. 5. Audits.	Équipe de développement.
5	Sécurité du réseaux	4	1	1. Firewall. 2. Cryptographie ou cryptage (HTTPS). 3. Équipements réseau tels que routeurs physiques, commutateurs, périphériques pare-feu et points d'accès sans fil. 4. Registre des risques. 5. Infrastructure réseau Azure	Équipe de développement.

				6. Audits.	
6	Sécurité du déploiement	4	1	1. Clés SSH 2. Azure AD. 3. Contrôle d'accès basé sur les rôles (RBAC). 4. Registre des risques. 5. Utilisation du Suite de Management d'Operations et du Centre de Sécurité d'Azure. 6. Microsoft Azure. 7. Audits.	Équipe de développement.
7	Sécurité des applications	4	1	1. Des outils tels que NDepend peuvent aider à éliminer les dépendances dans une base de code et à déterminer plus facilement où des conflits potentiels pourraient survenir à partir du graphe de dépendances. 2. Registre des risques. 3. Utilisation du Suite de Management d'Operations et du Centre de Sécurité d'Azure. 4. Microsoft Azure. 5. Audits.	Équipe de développement.

Note : *La gravité et la probabilité de chaque risque sont généralement évalués sur une échelle d'un à quatre*

15. Hypothèses

Le tableau suivant résume les hypothèses pour cette Déclaration de travail d'architecture.

ID	Hypothèse	Impact	Propriétaire
1	Plutôt que d'investir davantage dans la plateforme existante, nous la conserverons en mode de maintenance. Aucune nouvelle fonctionnalité ne sera développée.	L'ensemble de l'équipe technique sera disponible et focalisée sur les opérations de migration de la plateforme vers le Cloud, sur la base d'une architecture de microservices.	Équipe Technique (+CIO, + Responsable Ingénierie, + Responsable des opérations)
2	La nouvelle architecture sera construite en fonction des technologies actuelles et avec la capacité de s'adapter à de nouvelles technologies lorsque celles-ci seront disponibles.	1. Début d'un nouveau processus d'innovation technologique continue de la plateforme Foosus. 2. Amélioration des performances de l'ensemble de l'application. 3. Augmentation de la capacité de compétitivité de Foosus.	Équipe Technique (+CIO, + Responsable Ingénierie, + Responsable des opérations)
3	Les équipes étant attachées à la plateforme existante, les dirigeants devront éviter de prendre de faux raccourcis en intégrant un nouveau comportement dans le système existant.	L'ensemble de l'équipe technique sera disponible et focalisée sur les opérations de migration de la plateforme vers le Cloud, sur la base d'une architecture de microservices.	Équipe Technique (+CIO, + Responsable Ingénierie, + Responsable des opérations)
4	La géolocalisation, si elle est modélisée suffisamment tôt dans la nouvelle plateforme, permettra d'introduire	1. Booster la consommation de produits locaux. 2. Augmenter les	Équipe Technique (+CIO, + Responsable Ingénierie, + Responsable des opérations)

	d'autres innovations en fonction de l'emplacement de l'utilisateur ou du fournisseur alimentaire.	abonnements.	
5	L'élaboration sur mesure d'une approche architecturale de type « lean » pourra contribuer à la réalisation de cette feuille de route, ce qui évitera de priver les équipes de leur autonomie et de compromettre la rapidité des cycles de versions.	1. Faciliter et accélérer le processus de migration.	Équipe Technique (+CIO, + Responsable Ingénierie, + Responsable des opérations)

16. Procédures de changement de périmètre

Sachant que le processus de migration implique le transfert par étapes de l'application monolithique hébergée on-premise vers le cloud afin d'éviter toute interruption de service, je propose l'utilisation d'une approche hybride. Cela signifie que certains composants résideront dans le cloud et certains continueront à résider dans leur domicile existant on-premise.

Cela réduit vraisemblablement le risque de déplacer tous les aspects d'une plate-forme à la fois, mais peut poser des défis en fonction des décisions prises et des composants migrés. Certains domaines à garder à l'esprit comprennent:

- Routes de trafic réseau acceptables
    - VPN sur Internet public
    - Internet public direct
    - Intranet privé
- Les actifs du réseau, tels que
    - Blocs d'adresses CIDR internes
    - Tables de routage
    - Pare-feu

- IPs virtuelles
- Zone DNS, A, enregistrements CNAME

- Latence et cohérence attendues ou réelles

17. Calendrier

Afin de simplifier la présentation du calendrier, prenant en considération que chaque livrable aura une durée maximale de 3 semaines et sachant que 25 livrables clés ont été définis, il nous faudra 75 semaines, soit 525 jours ouvrables, pour mener à bien le processus de migration.

Cependant, en fonction du nombre d'équipes technique que Foosus sera capable de créer, ce temps peut être considérablement réduit.

18. Personnes approuvant ce plan

Validateurs	Domaine de responsabilité	Date
La Direction de l'Entreprise		10/03/2021
Natasha Jarson	CIO	09/03/2021
Le comité d'architecture		08/03/2021