F00SUS

Spécification des Conditions requises pour l’Architecture

v 0.1

Table des matières

Information sur le document....3
Objet de ce document....4
Mesures du succès....5
Indicateur technique....5
Indicateur business....5
Conditions requises pour l'architecture....5
Contrats de service business....7
Accords de niveau de service....7
Contrats de service application....8
Lignes directrices pour l'implémentation....9
Spécifications pour l'implémentation....10
Backend....10
Frontend....10
Bases de Données....12
Communication entre Services....12
Infrastructure et Déploiement....13
Standards pour l'implémentation....14
Sécurité....14
Conteneurisation et Orchestration....15
Monitoring et Logging....15
Pratiques de Développement....16
Conditions requises pour l'interopérabilité....18
Intégration des systèmes existants....18
Compatibilité avec des systèmes tiers....19
Interopérabilité des microservices....20
Conformité aux standards de l'industrie....21
Conditions requises pour le management du service IT....22
Monitoring et Supervision....22
Gestion des Incidents et des Problèmes....23
Sécurité et Conformité....23
Gestion des Configurations et des Changements....24
Support Utilisateur....25
Contraintes....27
Contrainte Budgétaire....27
Contrainte de Temps....28
Coexistence des Systèmes Actuels et Nouveaux....29
Contraintes Technologiques....30
Hypothèses....31
Checklists d'audit....32
Approbations signées....35

Information sur le document

Nom du projet	Conception d’une nouvelle architecture
Préparé par :	Clément Bastion
N° de version du document :	0.1
Titre :	Spécification des Conditions requises pour l’Architecture
Date de version du document :	12/07/2024
Revu par :	
Date de révision :	

Objet de ce document

La Spécification des Conditions requises pour l'Architecture fournit un ensemble de déclarations quantitatives qui dessinent ce que doit faire un projet d'implémentation afin d'être conforme à l'architecture.

Une Spécification des Conditions requises pour l'Architecture constitue généralement un composant majeur du contrat d'implémentation, ou du contrat pour une Définition de l'Architecture plus détaillée. Comme mentionné ci-dessus, la Spécification des Conditions requises pour l'Architecture accompagne le Document de Définition de l'Architecture, avec un objectif complémentaire : le Document de Définition de l'Architecture fournit une vision qualitative de la solution et tâche de communiquer l'intention de l'architecte.

La Spécification des Conditions requises pour l'Architecture fournit une vision quantitative de la solution, énumérant des critères mesurables qui doivent être remplis durant l'implémentation de l'architecture.

Ce document est une **Spécification des Conditions Requises pour l'Architecture** qui supprime toute ambiguïté et qui spécifie :

- Une description des conditions requises pour l'implémentation de l'architecture.
- Une description des conditions requises pour la conformité de l'implémentation.

Mesures du succès

Les mesures du succès permettent de suivre et d'évaluer la performance du projet par rapport aux objectifs stratégiques de l'entreprise. Ces mesures incluent des indicateurs clés de performance (KPIs) tels que l'augmentation des utilisateurs, l'expansion du réseau de producteurs, l'efficacité de la publication des contenus et la stabilité de la plateforme.

Indicateur technique

Indicateurs	**Valeur cible**	**Description**
Taux d'incidents de production sur 1 mois	Réduit de >25/mois à moins de 1/mois	Taux d'incidents de production sur 1 mois
Taux de disponibilité (SLA)	99,95 %	Taux de disponibilité (SLA) sur 1 mois

Indicateur business

Métrique	Technique de mesure	Valeur cible	Justification
Nombre d'adhésions d'utilisateurs par jour	Monitoring	+10%	Augmenter la base d'utilisateurs
Adhésion de producteurs alimentaires	Monitoring	4/mois	Augmenter le réseau de fournisseurs
Délai moyen de parution	Monitoring	Moins d'une semaine	Améliorer la réactivité

Conditions requises pour l'architecture

Ces conditions sont essentielles pour garantir que la nouvelle plateforme Foosus est capable de répondre aux attentes des utilisateurs, de soutenir la croissance de l'entreprise et de maintenir un avantage concurrentiel sur le marché.

L’architecture cible devra respecter les conditions suivantes :

- **Scalabilité et Robustesse :** L'architecture doit être capable de s'adapter aux variations de charge, supportant efficacement les pics d'affluence sans dégradation des performances. Cela implique l'utilisation de mécanismes **d'élasticité** pour ajuster dynamiquement les **ressources** en fonction de la demande.

- **Géolocalisation :** Intégration d’une solution de géolocalisation avancée permettant de proposer aux clients des produits disponibles à **proximité** de leur emplacement géographique. Cette fonctionnalité doit améliorer l'expérience utilisateur en offrant des options personnalisées et **localisées**.

- **Innovation et Flexibilité :** Favoriser l'innovation en adoptant une architecture maintenable et flexible. Il doit être simple et rapide d'itérer sur cette architecture pour ajouter de nouvelles fonctionnalités, améliorer les existantes, et répondre aux besoins changeants du marché. Cela inclut l'utilisation de **microservices** et de pratiques **DevOps** pour des cycles de développement et de déploiement continus.

- **Disponibilité Globale :** Assurer une disponibilité continue de la plateforme, quel que soit l'endroit dans le monde où les utilisateurs se trouvent. L'architecture doit minimiser la perte de bande passante et optimiser les performances réseau, en s'appuyant sur des technologies de **CDN** (Content Delivery Network) et des infrastructures cloud globales pour réduire la latence et améliorer l'expérience utilisateur.
- **Disponibilité multi-plateformes :** La plateforme doit être accessible depuis divers appareils, y compris les mobiles et les ordinateurs, avec des optimisations adaptées aux différents niveaux de bande passante. Cela inclut la création de sites web responsive et d'applications natives ou hybrides, assurant une expérience utilisateur cohérente et performante, quel que soit le dispositif utilisé.

Contrats de service business

Accords de niveau de service

SLO (Service Level Objective)	SLI (Service Level Indicator)	SLA (Service Level Agreement)
Disponibilité de 99.9%	Taux de disponibilité : Pourcentage de temps pendant lequel le service est opérationnel	Le service doit être disponible 99.9% du temps chaque mois.
Résolution des incidents critiques en moins de 4 heures	Temps moyen de résolution (MTTR) : Moins de 30 minutes pour les incidents critiques	Les incidents critiques doivent être résolus en moins de quatre heures.
Taux d'erreur des transactions inférieur à 0.1%	Taux d'erreur : Pourcentage de requêtes échouées par rapport au nombre total de requêtes	Le taux d'erreur des transactions ne doit pas dépasser 0.1%.
Temps de chargement des pages de moins de 3 secondes	Latence : Temps moyen de chargement des pages	Les utilisateurs doivent pouvoir se connecter à la plateforme avec un accès spécifique correspondant à leur profil.
Détection des incidents en moins de 5 minutes	Temps de détection des incidents : Temps moyen de détection des incidents	La nouvelle architecture doit faciliter l'intégration de nouvelles solutions fournies par des partenaires ou développées en interne, et ces intégrations doivent être facilement réversibles.
Maintenance planifiée	Annonce 48h à l'avance	Maintenance avec durée maximale de 4 heures par mois.

Contrats de service application

Accords de niveau de service

SLO (Service Level Objective)	SLI (Service Level Indicator)	SLA (Service Level Agreement)
Disponibilité des applications de 99.5%	Taux de disponibilité des applications : Pourcentage de temps pendant lequel les applications sont opérationnelles	La plateforme doit être accessible 24/24h pour tous les fuseaux horaires où elle est disponible.
Détection et résolution des incidents critiques en moins de 4 heures	Temps de détection des incidents critiques : Temps moyen de détection des incidents	Les incidents critiques doivent être résolus en moins de 4 heures.
Performances de recherche et de géolocalisation optimisées	Temps moyen de réponse pour les requêtes de recherche et de géolocalisation : Moins de 200 ms	Les utilisateurs doivent pouvoir rechercher les producteurs proches d'eux en fonction de leur adresse ou de leur position géographique.

Lignes directrices pour l’implémentation

Les lignes directrices pour l’implémentation sont des recommandations visant à garantir la qualité et la maintenabilité du code, ainsi que l'efficacité du processus de développement. Elles incluent l'utilisation de normes de codage, de frameworks partagés, et d'outils de gestion de configuration.

Ligne directrice	Détail
Normes de codage	Adoption de normes de codage claires pour garantir la qualité et la cohérence du code.
Cadres et bibliothèques partagés	Utilisation de frameworks et bibliothèques standardisés pour éviter la duplication et promouvoir la réutilisabilité.
Documentation technique	Création et maintenance de documentations détaillées pour faciliter la prise en main par les nouveaux membres de l'équipe.
Outils de gestion de configuration	Mise en place d'outils pour suivre les modifications du code et gérer les versions.
Intégration continue et déploiement continu (CI/CD)	Utilisation de pipelines CI/CD pour automatiser les tests et les déploiements.
Utilisation de méthodologies Agile	Adoption de Lean Kanban pour gérer les cycles de développement.

Spécifications pour l’implémentation

Backend

Technologie : Spring Boot (Java)

- **Description :** Spring Boot est un framework open-source de Java qui facilite la création de microservices robustes et maintenables. Il offre une configuration simplifiée et des outils intégrés pour la gestion des dépendances, la sécurité, et les tests.
- **Justification :** La robustesse de Spring Boot, combinée à une communauté large et active, en fait un choix idéal pour développer des microservices. Il permet une gestion efficace des configurations et une intégration facile avec d'autres technologies Java. Spring Boot est conçu pour faciliter la création de microservices, avec des capacités natives pour la gestion des sessions, la sécurité, et la communication inter-services.

Tableau : Spécifications du Backend

Caractéristique	Détail	Avantage
Framework	Spring Boot	Robustesse, large communauté, facilité d'intégration
Langage	Java	Fiabilité, support étendu, performance
Architecture	Microservices	Scalabilité, modularité, maintenabilité

Frontend

Technologie : React.js

- **Description :** React.js est une bibliothèque JavaScript pour la construction d'interfaces utilisateur. Elle permet de créer des composants réutilisables et une gestion efficace de l'état de l'application, grâce à des outils comme Redux.
- **Justification :** React.js est choisi pour ses performances élevées et son écosystème riche. La bibliothèque utilise un DOM virtuel pour améliorer la performance des interfaces utilisateur et minimise les mises à jour inutiles de l'interface. De plus, la réutilisabilité des composants facilite la maintenance et l'évolution de l'application.

Technologie : Flutter

- **Description :** Flutter est un framework open-source de Google qui permet de développer des applications mobiles pour iOS et Android avec une base de code unique. Il compile en code natif, offrant des performances proches des applications natives.
- **Justification :** Flutter est choisi pour sa capacité à créer des interfaces utilisateur riches et cohérentes sur les deux plateformes mobiles (iOS et Android), ainsi que pour ses performances élevées. Le framework permet également une productivité accrue grâce au rechargement à chaud (hot reload) et à une large collection de widgets personnalisables.

Tableau : Spécifications du Frontend

Caractéristique	Détail	Avantage
Bibliothèque	React.js	Performance, composant réutilisable, écosystème riche
Gestion de l'état	Redux	Gestion efficace et centralisée de l'état
Compatibilité	Web, Mobile	Responsive design, adaptabilité multi-plateformes
Framework	Flutter	UI riche, performances natives, rechargement à chaud
Langage	Dart	Simplicité, support multiplateforme

Bases de Données

Technologies : MongoDB et PostgreSQL

- **MongoDB :**
  - **Description :** MongoDB est une base de données NoSQL orientée document, adaptée pour le stockage de données semi-structurées et non structurées.
  - **Justification :** Idéale pour les applications nécessitant une flexibilité dans la gestion des données, MongoDB supporte une scalabilité horizontale et une manipulation rapide des données.
- **PostgreSQL :**
  - **Description :** PostgreSQL est une base de données relationnelle open-source reconnue pour sa robustesse, sa conformité ACID (Atomicité, Cohérence, Isolation, Durabilité), et ses capacités d'extension.
  - **Justification :** PostgreSQL est choisi pour les transactions complexes nécessitant une intégrité des données stricte. Elle offre des performances élevées pour les opérations transactionnelles et une grande fiabilité.

Tableau : Spécifications des Bases de Données

Type de base de données	Technologie	Avantage
NoSQL	MongoDB	Flexibilité, scalabilité horizontale
Relationnelle	PostgreSQL	Conformité ACID, robustesse, extensibilité

Communication entre Services

Technologies : Apache Kafka et RabbitMQ

- **Apache Kafka :**
  - **Description :** Kafka est une plateforme de streaming distribuée qui permet de gérer des flux de données en temps réel. Il est utilisé pour la collecte, le traitement et le stockage de grands volumes de données.
  - **Justification :** Kafka est sélectionné pour sa capacité à gérer de gros volumes de données avec une latence faible, offrant une haute disponibilité et une tolérance aux pannes.
- **RabbitMQ :**
  - **Description :** RabbitMQ est un message broker qui facilite la communication asynchrone entre applications. Il permet de distribuer des messages de manière fiable entre les différentes parties d'un système.

- **Justification :** RabbitMQ est utilisé pour la simplicité de sa configuration et sa capacité à gérer des messages complexes avec des fonctionnalités de routage avancées.

Tableau : Spécifications de la Communication entre Services

Technologie	Description	Avantage
Kafka	Plateforme de streaming de données	Gestion des flux de données en temps réel, haute scalabilité
RabbitMQ	Message broker	Communication asynchrone, routage avancé

Infrastructure et Déploiement

Technologies : Docker et Kubernetes

- **Docker :**
  - **Description :** Docker est une plateforme qui permet de conteneuriser des applications, assurant leur portabilité et une gestion simplifiée des environnements.
  - **Justification :** Docker est essentiel pour garantir que les applications fonctionnent de manière cohérente sur différents environnements de développement, de test et de production.
- **Kubernetes :**
  - **Description :** Kubernetes est un orchestrateur de conteneurs qui automatise le déploiement, la gestion et la mise à l'échelle des applications conteneurisées.
  - **Justification :** Kubernetes facilite la gestion des clusters de conteneurs, assurant une haute disponibilité, une mise à l'échelle automatique et une tolérance aux pannes.

Tableau : Spécifications de l'Infrastructure et Déploiement

Technologie	Description	Avantage
Docker	Conteneurisation d'applications	Portabilité, isolation, cohérence
Kubernetes	Orchestration de conteneurs	Automatisation des déploiements, scalabilité, haute disponibilité

Standards pour l’implémentation

Les standards pour l’implémentation définissent les règles, les protocoles et les meilleures pratiques à suivre pour le développement, le déploiement et la gestion de la nouvelle architecture de la plateforme Foosus. Ces standards garantissent une cohérence dans le processus de développement, améliorent la qualité du produit final et assurent la sécurité et la fiabilité des systèmes.

Sécurité

Standards : OAuth 2.0 et JWT

- **OAuth 2.0 :**
    - **Description :** OAuth 2.0 est un protocole standard d'autorisation qui permet aux applications d'accéder aux ressources d'un utilisateur sans révéler ses identifiants. Il est largement utilisé pour sécuriser les API et permet une gestion fine des permissions.
    - **Utilisation :** OAuth 2.0 sera utilisé pour sécuriser l'accès aux services et aux ressources, notamment les API publiques et privées de la plateforme. Les jetons d'accès générés par OAuth 2.0 garantiront que seules les entités autorisées puissent accéder aux données sensibles.
- **JWT (JSON Web Tokens) :**
    - **Description :** JWT est un standard ouvert qui définit un moyen compact et sécurisé de transmettre des informations entre les parties sous forme d'objets JSON. Il est souvent utilisé pour l'authentification et l'autorisation.
    - **Utilisation :** Les JWT seront utilisés pour l'authentification des utilisateurs et l'autorisation des accès. Leur nature stateless permet une vérification rapide et sécurisée des utilisateurs, facilitant la gestion des sessions dans les environnements distribués.

Tableau : Standards de Sécurité

Standard	Description	Utilisation	Avantage
OAuth 2.0	Protocole d'autorisation	Sécurisation des API et gestion des permissions	Contrôle d'accès granulaire
JWT	Tokens pour l'authentification	Authentification des utilisateurs	Vérification rapide, stateless

Conteneurisation et Orchestration

Standards : Docker et Kubernetes

- **Docker :**
  - **Description :** Docker est une technologie de conteneurisation qui encapsule une application et ses dépendances dans un conteneur, garantissant que l'application fonctionne de manière cohérente dans différents environnements.
  - **Utilisation :** Docker sera utilisé pour emballer les microservices et leurs dépendances, facilitant ainsi le déploiement et la gestion des applications à travers les environnements de développement, de test et de production.
- **Kubernetes :**
  - **Description :** Kubernetes est un système d'orchestration de conteneurs qui gère le déploiement, l'auto-scaling et la maintenance des applications conteneurisées.
  - **Utilisation :** Kubernetes sera employé pour orchestrer les conteneurs Docker, automatisant la gestion de l'infrastructure, y compris la mise à l'échelle des services, l'équilibrage de charge, et la gestion des défaillances.

Tableau : Standards de Conteneurisation et Orchestration

Standard	Description	Utilisation	Avantage
Docker	Conteneurisation des applications	Packaging des microservices	Portabilité, isolation des environnements
Kubernetes	Orchestration de conteneurs	Gestion automatisée des conteneurs	Scalabilité automatique, haute disponibilité

Monitoring et Logging

Standards : Prometheus et ELK Stack

- **Prometheus :**
  - **Description :** Prometheus est un système de surveillance open-source qui collecte et stocke des métriques de temps réel dans une base de données de séries temporelles. Il permet de surveiller les performances des applications et de générer des alertes.
  - **Utilisation :** Prometheus sera utilisé pour la surveillance des microservices et des infrastructures, offrant une visibilité sur les performances et la santé des

systèmes. Il aidera à détecter et à résoudre les problèmes avant qu'ils n'affectent les utilisateurs.

- **ELK Stack (Elasticsearch, Logstash, Kibana) :**
  - **Description :** La suite ELK est un ensemble d'outils open-source pour la recherche, l'analyse et la visualisation des logs générés par les applications. Elasticsearch est utilisé pour l'indexation et la recherche, Logstash pour la collecte et la transformation des données, et Kibana pour la visualisation.
  - **Utilisation :** L'ELK Stack sera employé pour la centralisation des logs, permettant une analyse approfondie des incidents et des performances. Elle fournira une interface conviviale pour visualiser les données de logs et détecter les anomalies.

Tableau : Standards de Monitoring et Logging

Standard	Description	Utilisation	Avantage
Prometheus	Système de monitoring et d'alerte	Surveillance des performances et de la santé des systèmes	Alertes en temps réel, visualisation des métriques
ELK Stack	Outils de gestion des logs	Centralisation et analyse des logs	Analyse approfondie, visualisation des données

Pratiques de Développement

Standards : DevOps, CI/CD

- **DevOps :**
  - **Description :** DevOps est une pratique qui combine le développement logiciel (Dev) et les opérations informatiques (Ops) pour améliorer la collaboration et la productivité en automatisant les processus d'infrastructure et en surveillant les performances des applications.
  - **Utilisation :** La culture DevOps sera adoptée pour promouvoir la collaboration entre les équipes de développement et d'opérations. Cela inclut l'automatisation des processus de déploiement, de tests, et de monitoring pour garantir des livraisons rapides et de haute qualité.

- **CI/CD (Intégration Continue et Déploiement Continu) :**

- **Description :** CI/CD est une méthodologie qui permet l'intégration fréquente des modifications de code, suivie d'un processus de déploiement automatisé. CI assure que chaque modification est validée par un processus automatisé de construction et de tests, tandis que CD permet le déploiement continu des modifications en production.
- **Utilisation :** Les pipelines CI/CD seront mis en place pour automatiser l'intégration et le déploiement du code. Cela inclut des tests automatisés pour détecter les erreurs rapidement, ainsi qu'un déploiement automatisé pour réduire le temps de mise sur le marché et minimiser les erreurs humaines.

Tableau : Standards de Pratiques de Développement

Standard	Description	Utilisation	Avantage
DevOps	Collaboration et automatisation	Automatisation des processus de déploiement et de monitoring	Livraison rapide, amélioration continue
CI/CD	Intégration et déploiement continus	Automatisation des tests et des déploiements	Réduction des erreurs, accélération du cycle de développement

Conditions requises pour l’interopérabilité

L'interopérabilité est une exigence cruciale pour la nouvelle architecture de la plateforme Foosus, assurant que divers systèmes, applications et services puissent fonctionner ensemble de manière fluide et efficace. Cette capacité permet de faciliter les échanges de données, de garantir la continuité des services, et d'optimiser les processus opérationnels à travers une variété d'environnements technologiques.

Intégration des systèmes existants

Description : La nouvelle architecture doit être capable de s'intégrer de manière transparente avec les systèmes existants de Foosus. Cela inclut les bases de données actuelles, les applications de gestion des clients (CRM), les systèmes de gestion des commandes (OMS), et les infrastructures de gestion des stocks.

Objectifs :

- Maintenir la continuité des services tout en introduisant la nouvelle architecture.
- Assurer une transition en douceur avec une perturbation minimale pour les opérations existantes.
- Faciliter l'accès aux données historiques et leur migration vers les nouveaux systèmes lorsque nécessaire.

Approches recommandées :

- **Utilisation de passerelles d'API** pour faciliter la communication entre les anciens et les nouveaux systèmes.
- **Migration progressive des données** pour transférer les informations des systèmes hérités vers de nouvelles bases de données sans interruption de service.

Tableau : Intégration des systèmes existants

Composant	Intégration requise	Méthode recommandée
Bases de données existantes	Accès continu aux données	Passerelles d'API, ETL (Extract, Transform, Load)
CRM et OMS	Communication bidirectionnelle	Services web, middleware d'intégration
Systèmes de gestion des stocks	Synchronisation des données	Interfaces de programmation (API)

Compatibilité avec des systèmes tiers

Description : L'architecture doit permettre l'intégration facile avec des systèmes tiers, tels que les services de paiement en ligne, les plateformes de logistique, et les outils de marketing digital. Cette compatibilité est essentielle pour offrir une gamme complète de services aux clients et aux partenaires commerciaux.

Objectifs :

- Permettre l'intégration avec une variété de fournisseurs de services externes.
- Assurer la sécurité et la confidentialité des données lors des échanges avec des systèmes tiers.
- Supporter les standards de l'industrie pour les échanges de données et les transactions.

Approches recommandées :

- **Adoption de standards de communication universels** comme RESTful APIs, SOAP, ou GraphQL pour assurer une compatibilité large.
- **Utilisation de protocoles sécurisés** comme HTTPS et OAuth pour protéger les échanges de données.

Tableau : Compatibilité avec des systèmes tiers

Service tiers	Standard de communication	Protocole de sécurité
Services de paiement	RESTful API, SOAP	HTTPS, OAuth
Plateformes de logistique	EDI (Electronic Data Interchange), API	TLS (Transport Layer Security)
Outils de marketing digital	Webhooks, API	JWT, SSL (Secure Sockets Layer)

Interopérabilité des microservices

Description : Les microservices de la nouvelle architecture doivent être conçus pour communiquer entre eux de manière fluide et efficace, que ce soit à l'intérieur de l'infrastructure de Foosus ou à travers différents environnements de déploiement, comme les clouds publics ou privés.

Objectifs :

- Assurer une communication fiable et à faible latence entre les microservices.
- Permettre la scalabilité horizontale des services tout en maintenant la cohérence des données.
- Gérer les défaillances partielles sans interruption des services globaux.

Approches recommandées :

- **Utilisation de systèmes de message et de queues** comme RabbitMQ ou Kafka pour la communication asynchrone.
- **Implémentation de schémas de sérialisation** tels que JSON pour des échanges de données efficaces.

Tableau : Interopérabilité des microservices

Protocole de communication	Description	Avantages
RESTful API	API pour la communication inter-service	Simplicité, compatibilité universelle
gRPC	Framework de communication RPC	Performance, prise en charge du streaming
Message brokers (Kafka, RabbitMQ)	Systèmes de message pour la communication asynchrone	Résilience, tolérance aux pannes

Conformité aux standards de l'industrie

Description : L'architecture doit être conforme aux standards de l'industrie, garantissant que les systèmes puissent interagir sans friction avec des technologies et des plateformes tierces, tout en assurant la sécurité et la protection des données.

Objectifs :

- Adhérer aux normes de sécurité et de confidentialité des données, comme GRPD.
- Utiliser des standards de communication et de données largement acceptés pour garantir l'interopérabilité.

Approches recommandées :

- **Suivi des bonnes pratiques et des standards** établis par des organismes comme l'ISO, l'IETF, et l'IEEE.
- **Audit régulier des systèmes** pour vérifier la conformité continue aux normes applicables.

Tableau : Conformité aux standards de l'industrie

Standard	Domaine	Importance
GRPD	Protection des données personnelles	Conformité légale, protection de la vie privée
ISO/IEC 27001	Sécurité de l'information	Gestion des risques, protection des données
PCI-DSS	Sécurité des données de paiement	Sécurisation des transactions financières

Conditions requises pour le management du service IT

Le management du service IT est un aspect crucial de la gestion de la nouvelle architecture de la plateforme Foosus. Il englobe la supervision, le contrôle et la maintenance des services informatiques pour garantir leur disponibilité, performance, sécurité, et alignement avec les objectifs stratégiques de l'entreprise. Les conditions requises pour le management du service IT visent à établir des pratiques et des outils robustes pour soutenir une gestion efficace et proactive des ressources IT.

Monitoring et Supervision

Description : La mise en place d'un système de monitoring est essentielle pour surveiller en temps réel les performances des applications, des infrastructures, et des réseaux. Le monitoring permet de détecter rapidement les anomalies, de diagnostiquer les problèmes, et de mettre en place des mesures correctives avant qu'ils n'affectent les utilisateurs finaux.

Objectifs :

- Assurer une surveillance continue des systèmes pour garantir leur disponibilité et leur performance.
- Identifier et résoudre les incidents de manière proactive pour minimiser les interruptions de service.
- Fournir des données et des rapports pour l'analyse des tendances et l'optimisation des performances.

Approches recommandées :

- **Utilisation de Prometheus** pour la collecte de métriques de performance et la génération d'alertes en temps réel.
- **Intégration de Grafana** pour la visualisation des données de monitoring, permettant de suivre les KPI (Key Performance Indicators) et les SLA (Service Level Agreements).

Tableau : Outils de Monitoring et Supervision

Outil	Fonctionnalité	Avantage
Prometheus	Collecte de métriques et alertes	Surveillance en temps réel, personnalisation des alertes
Grafana	Visualisation des données de monitoring	Interface intuitive, tableaux de bord personnalisés

Gestion des Incidents et des Problèmes

Description : La gestion des incidents et des problèmes vise à réagir rapidement aux interruptions de service, à en minimiser l'impact et à prévenir leur récurrence. Une approche structurée de gestion des incidents permet d'assurer une reprise rapide des services, tandis que la gestion des problèmes se concentre sur l'identification et la résolution des causes profondes des incidents.

Objectifs :

- Minimiser le temps de résolution des incidents pour réduire les impacts négatifs sur les utilisateurs.
- Documenter et analyser les incidents pour éviter les récurrences.
- Améliorer les processus de gestion des incidents et des problèmes en continu.

Approches recommandées :

- **Mise en place d'un système de tickets** pour la gestion des incidents, avec des priorités définies en fonction de la criticité des problèmes.
- **Utilisation de la méthode ITIL** (Information Technology Infrastructure Library) pour les processus de gestion des incidents et des problèmes.

Tableau : Processus de Gestion des Incidents et Problèmes

Processus	Description	Outil recommandé
Gestion des incidents	Réponse rapide aux interruptions de service	ServiceNow, Jira Service Desk
Gestion des problèmes	Identification et résolution des causes racines	ITIL, Root Cause Analysis (RCA)

Sécurité et Conformité

Description : La sécurité et la conformité sont des aspects critiques du management du service IT, visant à protéger les données sensibles et à se conformer aux réglementations en vigueur. Cela inclut la mise en place de politiques de sécurité, de contrôles d'accès, et de procédures de réponse aux incidents de sécurité.

Objectifs :

- Assurer la confidentialité, l'intégrité, et la disponibilité des données.

- Se conformer aux régulations telles que le RGPD (Règlement Général sur la Protection des Données) pour protéger les informations personnelles.
- Mettre en place des audits de sécurité réguliers pour vérifier l'efficacité des mesures de sécurité.

Approches recommandées :

- **Déploiement de solutions de sécurité** comme les pare-feu, les systèmes de détection d'intrusion (IDS), et les systèmes de prévention des intrusions (IPS).
- **Utilisation de protocoles de sécurité** tels que SSL/TLS pour sécuriser les communications et le chiffrement des données sensibles.

Tableau : Outils et Protocoles de Sécurité

Outil/Protocole	Fonctionnalité	Avantage
SSL/TLS	Sécurisation des communications	Chiffrement des données en transit
IDS/IPS	Détection et prévention des intrusions	Surveillance proactive des menaces
Audit de sécurité	Évaluation des pratiques de sécurité	Identification des vulnérabilités, conformité

Gestion des Configurations et des Changements

Description : La gestion des configurations et des changements est essentielle pour contrôler et documenter les modifications apportées à l'infrastructure IT. Cela inclut la gestion des configurations matérielles et logicielles, ainsi que la supervision des processus de changement pour minimiser les risques d'erreurs et d'interruptions.

Objectifs :

- Assurer une gestion cohérente et transparente des configurations IT.
- Évaluer l'impact des changements avant leur mise en œuvre pour éviter les interruptions imprévues.
- Maintenir une documentation à jour des configurations et des changements.

Approches recommandées :

- **Utilisation d'outils de gestion de configuration** comme Ansible ou Puppet pour automatiser la gestion des configurations.
- **Mise en œuvre d'un processus de gestion des changements** basé sur les meilleures pratiques ITIL pour gérer et approuver les changements.

Tableau : Gestion des Configurations et des Changements

Outil/Processus	Description	Avantage
Ansible/Puppet	Gestion des configurations	Automatisation, cohérence des configurations
Gestion des changements ITIL	Processus de gestion des modifications	Réduction des risques, documentation des changements

Support Utilisateur

Description : Le support utilisateur est un élément clé du service IT, offrant assistance et résolution de problèmes aux utilisateurs finaux. Ce service comprend des mécanismes pour recevoir, suivre, et résoudre les demandes d'assistance, ainsi qu'un système pour collecter et analyser les feedbacks utilisateurs.

Objectifs :

- Fournir une assistance rapide et efficace pour résoudre les problèmes rencontrés par les utilisateurs finaux.
- Maintenir une communication claire et ouverte avec les utilisateurs pour les informer sur l'état de leurs requêtes et les solutions apportées.
- Utiliser les retours des utilisateurs pour identifier les points à améliorer dans les services et les systèmes.

Approches recommandées :

- **Mise en place d'un centre d'assistance (Help Desk)** avec des canaux multiples (téléphone, email, chat en ligne) pour permettre aux utilisateurs de soumettre leurs demandes.
- **Utilisation d'un système de gestion des tickets** pour suivre les demandes des utilisateurs depuis leur soumission jusqu'à leur résolution, avec des priorités définies en fonction de l'urgence et de l'impact.
- **Élaboration de bases de connaissances et de FAQ** pour fournir des solutions rapides aux problèmes courants, réduisant ainsi le volume de requêtes nécessitant une intervention directe.

Tableau : Support Utilisateur

Service	Description	Avantage
Centre d'assistance	Point de contact pour les demandes des utilisateurs	Accessibilité, centralisation des demandes
Système de gestion des tickets	Suivi des requêtes et gestion des priorités	Transparence, efficacité dans la résolution des problèmes
Bases de connaissances/FAQ	Ressources pour l'auto-assistance des utilisateurs	Réduction du temps de résolution, accès rapide à l'information

Contraintes

Les contraintes identifiées pour la mise en œuvre de la nouvelle architecture de la plateforme Foosus sont des limitations ou des restrictions qui peuvent influencer la planification, le développement, et le déploiement du projet. Comprendre et gérer ces contraintes est essentiel pour assurer le succès du projet tout en respectant les attentes des parties prenantes.

Contrainte Budgétaire

Description : Les ressources financières disponibles pour le projet sont limitées. Cette contrainte budgétaire impose des limites sur les dépenses pour le développement, les infrastructures, les outils et les ressources humaines.

Objectifs :

- Optimiser l'utilisation des ressources financières disponibles pour maximiser la valeur ajoutée du projet.
- Prioriser les fonctionnalités essentielles et les investissements nécessaires pour atteindre les objectifs stratégiques.
- Éviter les dépassements de budget en surveillant attentivement les coûts et en ajustant les priorités en fonction des disponibilités financières.

Approches recommandées :

- **Budgétisation et suivi rigoureux des dépenses** pour s'assurer que le projet reste dans les limites budgétaires allouées.
- **Priorisation des fonctionnalités** selon leur valeur ajoutée et leur coût de mise en œuvre.
- **Recherche de solutions open-source** ou de logiciels avec un bon rapport qualité-prix pour réduire les coûts sans sacrifier la qualité.

Tableau : Contrainte Budgétaire

Élément	Description	Stratégie
Budget limité	Fonds restreints pour le projet	Budgétisation stricte, priorisation des dépenses
Allocation des ressources	Distribution efficace des ressources financières	Analyse coût-bénéfice, sélection de solutions rentables

Contrainte de Temps

Description : Le projet doit être livré dans des délais stricts, souvent imposés par des engagements commerciaux, des attentes des utilisateurs, ou des compétitions sur le marché.

Objectifs :

- Respecter les échéances pour le lancement de la nouvelle architecture afin de minimiser l'impact sur les opérations existantes.
- Assurer la coordination efficace des équipes et des activités pour maintenir le calendrier projet.
- Adapter rapidement les priorités et les plans en cas de retards imprévus ou de changements de contexte.

Approches recommandées :

- **Méthodologies Agile et Scrum** pour une gestion flexible du projet, permettant des ajustements rapides et une livraison itérative.
- **Définition de jalons clairs** et surveillance constante de l'avancement pour identifier les écarts par rapport au planning.
- **Gestion proactive des risques** pour anticiper et atténuer les causes possibles de retard.

Tableau : Contrainte de Temps

Élément	Description	Stratégie
Délais stricts	Période limitée pour le développement et le déploiement	Planification Agile, gestion des priorités
Gestion des imprévus	Capacité à réagir aux retards et aux changements	Gestion des risques, flexibilité des équipes

Coexistence des Systèmes Actuels et Nouveaux

Description : Pendant la transition vers la nouvelle architecture, il est crucial de maintenir les systèmes existants en fonctionnement pour éviter les interruptions de service.

Objectifs :

- Assurer la continuité des services pour les utilisateurs finaux tout au long de la transition.
- Gérer les interfaces entre les systèmes actuels et la nouvelle architecture pour une intégration fluide.
- Préparer et planifier la migration progressive des données et des fonctionnalités sans perturber les opérations courantes.

Approches recommandées :

- **Stratégie de migration progressive** pour transférer les fonctions et les données par étapes, minimisant ainsi les risques de perturbation.
- **Tests approfondis** dans des environnements de pré-production pour s'assurer de la compatibilité et de la stabilité avant la mise en production.
- **Planification de la coexistence** avec des plans de repli (rollback) en cas de problème majeur lors de la transition.

Tableau : Coexistence des Systèmes

Élément	Description	Stratégie
Maintien des systèmes existants	Continuité des services durant la transition	Migration progressive, plan de repli
Intégration et interopérabilité	Fonctionnement conjoint des anciennes et nouvelles architectures	Tests de compatibilité, gestion des interfaces

Contraintes Technologiques

Description : Les choix technologiques sont parfois limités par des facteurs tels que l'infrastructure existante, les compétences des équipes, ou les réglementations en vigueur.

Objectifs :

- Utiliser des technologies compatibles avec l'infrastructure existante pour éviter des investissements supplémentaires.
- S'assurer que les technologies choisies soient maîtrisées par les équipes de développement et d'exploitation.
- Respecter les régulations locales et internationales concernant la technologie et la protection des données.

Approches recommandées :

- **Évaluation des technologies existantes** et des compétences internes avant de faire des choix technologiques.
- **Formation et développement des compétences** pour les équipes sur les nouvelles technologies.
- **Conformité réglementaire** en intégrant les exigences légales dès le début de la conception.

Tableau : Contraintes Technologiques

Élément	Description	Stratégie
Compatibilité	Technologies compatibles avec l'infrastructure existante	Évaluation préalable, compatibilité descendante
Compétences des équipes	Expertise requise pour les nouvelles technologies	Formation, recrutement ciblé
Régulations	Conformité aux normes et lois	Intégration des exigences réglementaires, audits réguliers

Hypothèses

Ces hypothèses doivent être vérifiées régulièrement tout au long du projet. Si l'une d'elles se révèle incorrecte, cela peut entraîner des ajustements dans la planification et la gestion du projet pour garantir le succès de l'initiative.

Hypothèse	**Impact**	**Propriétaire**
Les ressources nécessaires (humaines et financières) seront disponibles tout au long du projet	Une disponibilité continue des ressources garantit l'achèvement du projet dans les délais prévus et prévient les retards et les dépassements de budget.	Chef de Projet
Les parties prenantes resteront engagées et fourniront des retours rapides et constructifs	Un engagement continu des parties prenantes assure la pertinence et l'acceptation des livrables, facilitant une prise de décision rapide et efficace.	Responsable Produit
Les technologies sélectionnées seront compatibles avec les systèmes existants	La compatibilité technologique réduit les risques de réintégration, minimise les coûts supplémentaires et assure une transition en douceur.	Architecte en Chef
Les besoins des utilisateurs finaux ne changeront pas significativement au cours du projet	Une stabilité des besoins utilisateurs facilite la gestion du périmètre et aide à respecter les délais et les budgets.	Responsable Produit

Checklists d’audit

Cette checklist peut être utilisée pour faciliter l'audit et le suivi des conditions de conformité dans l’architecture.

1. Sécurité

- **Contrôles d'accès**
    - Les contrôles d'accès sont définis et appliqués pour chaque niveau du système (utilisateur, administrateur, etc.).
    - Les mécanismes d'authentification sont en place (par exemple, OAuth 2.0, SAML, JWT).
    - Les permissions sont correctement configurées pour éviter les accès non autorisés.
    - Les sessions utilisateur sont sécurisées (HTTPS, expiration de session, etc.).
- **Chiffrement**
    - Les données sensibles sont chiffrées au repos et en transit (SSL/TLS, AES).
    - Les clés de chiffrement sont stockées et gérées de manière sécurisée.
- **Gestion des incidents**
    - Un plan de réponse aux incidents de sécurité est en place.
    - Des mécanismes de détection d'intrusion (IDS/IPS) sont activés.
    - Les journaux de sécurité sont centralisés et analysés régulièrement.
- **Tests de sécurité**
    - Des tests de pénétration sont effectués régulièrement.
    - Des scans de vulnérabilité sont réalisés pour toutes les applications et infrastructures critiques.

2. Confidentialité

- **Protection des données personnelles**
    - Les données personnelles sont traitées conformément aux réglementations en vigueur (ex. RGPD).
    - Les utilisateurs sont informés et ont consenti au traitement de leurs données.
    - Un mécanisme est en place pour que les utilisateurs puissent exercer leurs droits (accès, rectification, suppression).
- **Anonymisation et pseudonymisation**
    - Les données personnelles sont anonymisées ou pseudonymisées là où cela est possible.
- **Audit et traçabilité**
    - Les accès et modifications des données personnelles sont journalisés.
    - Un plan de conservation des données est défini et respecté.

- **Partage de données avec des tiers**
    - Les contrats avec des tiers incluent des clauses de protection des données.
    - Les transferts de données hors de l'UE respectent les cadres légaux (par exemple, clauses contractuelles types, Privacy Shield).

3. Qualité

- **Conformité aux spécifications**
    - Les fonctionnalités développées respectent les spécifications fonctionnelles et non fonctionnelles.
    - Les exigences de performance sont définies et mesurées (temps de réponse, taux de disponibilité).
- **Tests de qualité**
    - Des tests unitaires, d'intégration et de bout en bout sont automatisés et exécutés régulièrement.
    - Les critères d'acceptation sont définis pour chaque fonctionnalité.
- **Gestion des versions**
    - Un processus de gestion des versions est en place, incluant la gestion des branches et les règles de fusion.
    - Les versions sont documentées avec des notes de version claires.
- **Revue et validation**
    - Des revues de code régulières sont organisées pour assurer la qualité du code.
    - Les processus d'intégration continue (CI) et de déploiement continu (CD) incluent des contrôles de qualité.

4. Audit et Conformité Générale

- **Documentation**
    - Tous les processus sont documentés et accessibles aux parties prenantes pertinentes.
    - La documentation est à jour et reflète les pratiques actuelles.
- **Suivi de la conformité**
    - Un calendrier d'audit interne est établi pour vérifier la conformité continue.
    - Les écarts identifiés lors des audits sont corrigés dans des délais raisonnables.
- **Formation et sensibilisation**
    - Le personnel est formé aux politiques de sécurité, de confidentialité et de qualité.
    - Des sessions de sensibilisation régulières sont organisées pour maintenir un haut niveau de conformité.

5. Gestion des Risques

- **Identification des risques**
    - Une analyse des risques est réalisée pour chaque projet majeur.
    - Les risques identifiés sont classés par ordre de priorité (probabilité d'occurrence et impact).
- **Plan de mitigation**
    - Un plan de mitigation est en place pour chaque risque identifié.
    - Les risques sont revus régulièrement et mis à jour en fonction des changements de contexte.

Approbations signées

Nom	Date	Signature
Ash Callum, CEO		
Daniel Anthony, CPO		
Natasha Jarson, CIO		