Spécification des Conditions requises pour l’Architecture

Projet : Approvisionnement alimentaire géographiquement responsable

Client : Foosus

Préparé par : Bastien GRISVARD

N° de Version du Document : 0.1

Titre : Spécification des Conditions requises pour l’Architecture

Date de Version du Document :

Revu par :

Date de Révision :

Table des Matières

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.

Mesures du succès

Métrique	Valeur cible	Technique de mesure	Justification
Nombre d'adhésions utilisateurs	+10% / mois	Suivi via analytics de la plateforme	Indicateur d'adoption par les utilisateurs finaux
Adhésion producteurs alimentaires	4 nouvelles adhésions/mois	back office métier	Indique la croissance de l'offre côté producteurs
Délai moyen de parution d'article	< 1 semaine	Temps entre rédaction et publication	Mesure l'efficacité du pipeline de publication
Taux d'incidents en production	< 1 incident/mois	Suivi via système de monitoring / logs	Indicateur de stabilité de l'architecture en prod

Conditions requises pour l'architecture

Besoins fonctionnels

- Nouvelles interfaces utilisateurs pour la recherche géolocalisée.
- Refonte des anciennes interfaces pour qu'elles soient mobile-first.
- Relevés de métriques pour mesurer le succès de la nouvelle architecture.

Besoins non-fonctionnels

- **Scalable**: Capable de supporter la croissance de l'entreprise, de +1M d'utilisateurs par exemple, sans interruption du service.
- **Performance** : Le service doit être accessible par des utilisateurs dans des zones géographiques spécifiques avec une bande passante limitée.
- **Sécurité** : Authentification des utilisateurs et chiffrement des données
- **Déploiement** : Faciliter le déploiement en production sans interruption.

Contrats de service business

Accords de niveau de service

- Offrir une **expérience utilisateur fluide et sans interruption** pour les producteurs comme pour les consommateurs.
- Garantir la **publication rapide des contenus** une fois x.
- Fournir un **support utilisateur réactif** et efficace.

Métrique	Technique de mesure	Cible	Justification
Taux de satisfaction utilisateur	Analyse des inscriptions via base de données / outil d'analytics	≥ 90% de retours positifs	Assure la qualité perçue par les utilisateurs finaux
Adhésion de producteurs alimentaires	Comptage des producteurs enregistrés par mois	4/mois	Mesure la traction côté offre
Délai moyen de parution d'article	Temps entre la demande de publication et la mise en ligne, via système de logs	< 1 semaine	Vise à améliorer l'efficacité du processus interne
Taux d'incidents de production	Nombre de tickets / alertes de production par mois (outil monitoring)	< 1/mois	Vise la stabilité
Taux de rétention mensuel	Comparaison mensuelle des utilisateurs actifs (via analytics ou backend)	≥ 70%	Indique la fidélisation

Contrats de service application

Objectifs de niveau de service

Assurer la disponibilité du service dans le monde entier 24h/24.

Indicateurs de niveau de service

Niveaux de service définis

- **Disponibilité**: 95%
- **Temps de réponse** :
    - Incident critique : 5min
    - Incident mineur : 1h
- **Performance** :
    - Temps de chargement maximum : 5s

Suivi et reporting :

Type de métrique	**Fréquence de la mesure**
Durée moyenne de résolution d'un incident	A chaque ticket
Temps moyen de réponse à un ticket	A chaque ticket
Temps d'activité du système	Toutes les minutes

Parties prenantes :

- Ash Callum (CEO), en cas d'incidents majeurs
- Natasha Jarson (CIO), en cas d'incidents
- Enterprise Architecture Owner et Jack Harkner (Ops Lead), seront en charge d'informer des pannes, d'ajuster les niveaux de services et de la mise en place des maintenances.

Lignes directrices pour l’implémentation

Ces lignes directrices visent à assurer une mise en œuvre cohérente et efficace de l'architecture, en alignement avec les objectifs stratégiques de l'entreprise.

1. Approche modulaire et évolutive

- **Principe** : Adopter une architecture basée sur des microservices pour faciliter l'évolutivité et la maintenance.
- **Justification** : Permet de déployer indépendamment des composants, réduisant ainsi les risques lors des mises à jour.

2. Déploiement continu et automatisation

- **Principe** : Mettre en place des pipelines CI/CD pour automatiser les tests, les intégrations et les déploiements.
- **Justification** : Réduit les erreurs humaines et accélère le temps de mise sur le marché des nouvelles fonctionnalités.

3. Accessibilité et performance

- **Principe** : Optimiser la plateforme pour une utilisation fluide sur des connexions à faible bande passante et sur des appareils mobiles.
- **Justification** : Assure une expérience utilisateur satisfaisante, indépendamment des contraintes techniques des utilisateurs.

4. Sécurité dès la conception

- **Principe** : Intégrer des mécanismes de sécurité dès les premières phases de développement (approche "Security by Design").
- **Justification** : Protège les données sensibles et renforce la confiance des utilisateurs envers la plateforme.

5. Intégration de la géolocalisation

- **Principe** : Utiliser des services de géolocalisation pour connecter les consommateurs aux producteurs locaux.
- **Justification** : Répond à la demande croissante pour des produits locaux et soutient les économies régionales.

Spécifications pour l’implémentation

Ces spécifications détaillent les exigences techniques et fonctionnelles nécessaires pour aligner l'implémentation avec l'architecture définie.

1. Architecture technique

- **Langages** : Utilisation de langages adaptés au développement web et mobile (par exemple, Typescript, Java).
- **Frameworks** : Adoption de frameworks éprouvés pour le développement rapide et sécurisé (par exemple, React, Spring).
- **Base de données** : Choix d'une base de données relationnelle ou NoSQL en fonction des besoins spécifiques des modules.

2. Infrastructure

- **Cloud** : Déploiement sur une infrastructure cloud pour assurer la scalabilité et la résilience.
- **Conteneurisation** : Utilisation de conteneurs (par exemple, Docker) pour faciliter le déploiement et la gestion des services.
- **Orchestration** : Mise en place d'un système d'orchestration (par exemple, Kubernetes) pour gérer les conteneurs à grande échelle.

3. Sécurité

- **Authentification** : Implémentation d'un système d'authentification robuste (par exemple, OAuth 2.0).
- **Chiffrement** : Chiffrement des données sensibles en transit et au repos.
- **Surveillance** : Mise en place de mécanismes de détection et de réponse aux incidents de sécurité.

4. Performance et disponibilité

- **Temps de réponse** : Objectif de temps de réponse inférieur à 5 secondes pour 95% des requêtes.
- **Disponibilité** : Assurer une disponibilité du service de 95% sur une base mensuelle.
- **Monitoring** : Surveillance continue des performances et alertes en cas de dégradation.

5. Interopérabilité

- **API** : Développement d'API RESTful pour faciliter l'intégration avec des services tiers.
- **Standards** : Conformité aux standards ouverts pour assurer la compatibilité avec d'autres systèmes.

Standards pour l’implémentation

Ces standards garantissent la cohérence, la maintenabilité et la robustesse de la plateforme dans un contexte de croissance progressive et d’ouverture vers des partenaires publics/privés.

1. Standards de développement

- Langages typés (TypeScript, Kotlin, Java) pour éviter les erreurs à l'exécution.
- Convention de nommage et structuration uniforme des modules (inspirée Clean Architecture).

2. Standards API

- API RESTful respectant les normes OpenAPI v3 pour la documentation et la validation automatique.
- Respect du versionnage d’API (/v1/, /v2/, etc.) pour gérer les évolutions sans casser la rétrocompatibilité.

3. Format des données

- JSON standardisé pour tous les échanges.
- Structure des entités partagée via des schémas communs (ex. producteurs, produits, commandes).

4. Sécurité

- Utilisation systématique de HTTPS (TLS 1.3).
- Authentification OAuth 2.0 avec scopes limités par rôle (ex : consommateur, producteur).

5. Conformité réglementaire

- Conformité RGPD : suppression, anonymisation, et export des données personnelles prévues dès la conception.

Conditions requises pour l’interopérabilité

L’interopérabilité est cruciale pour connecter la plateforme à d’autres SI (collectivités, bases open data, services de livraison).

1. Normes d’échange

- Adoption de standards ouverts (JSON, GeoJSON, CSV, OpenAPI).
- Acceptation de connecteurs externes via des webhooks ou API REST sécurisées.

2. Intégration de données tiers

- Prise en charge des formats Open Data (ex: référentiels géographiques, INSEE).
- Mécanismes de synchronisation régulière avec sources externes (via pull API par exemple).

3. Interopérabilité sémantique

- Vocabulaire métier unifié, exposé via un dictionnaire de données partagé.
- Mapping prévu pour les métadonnées de produits, certifications, unités de mesure.

4. Compatibilité mobile et web

- Interfaces accessibles via navigateur et PWA, support mobile natif si nécessaire.
- Standardisation des endpoints pour permettre un SDK ou une appli mobile futur

Conditions requises pour le management du service IT

Pour garantir un service fiable, sécurisé et scalable dans la durée, voici les conditions essentielles de gestion IT.

1. Supervision et monitoring

- Mise en place de dashboards de monitoring (Prometheus, Grafana) avec alertes. Logs centralisés (ELK ou équivalent) pour analyse post-incidents.

2. Support et maintenance

- Politique claire de gestion des incidents (SLA internes).
- Rétention des logs (30 jours minimum), backups réguliers (journaliers).

3. Gouvernance des accès

- Gestion des comptes et rôles via IAM (Identity & Access Management).
- Historique des actions utilisateurs critiques.

4. Documentation et transfert de connaissance

- Documentation technique versionnée (Github).

Contraintes

Ces contraintes influencent directement les choix techniques, fonctionnels et organisationnels.

Budget limité au lancement

- ➔ Privilégier les services open source ou gratuits (PostgreSQL, Keycloak, MinIO…).
- ➔ Ne pas surdimensionner l'infra initiale (scaling horizontal prévu à moyen terme).

Hébergement souverain souhaité

- ➔ Pas d'hébergement sur AWS, GCP ou Azure si souveraineté imposée.

- ➔ Compatibilité assurée avec des hébergeurs comme Clever Cloud, OVH.

Diversité des utilisateurs finaux

- ➔ Interface pensée pour des utilisateurs non techniques (producteurs locaux, collectivités).
- ➔ Accessibilité (RGAA), UX claire, responsive dès la V1.

Connexion réseau instable sur certains territoires

- ➔ Optimisation des temps de chargement, compression d'images, caching.

Hypothèses

Ces hypothèses orientent la conception mais devront être validées ou ajustées lors des itérations projet.

- **Adoption progressive par les parties prenantes**
  On suppose que les utilisateurs seront onboardés progressivement via des campagnes locales. La plateforme doit donc rester stable et performante à faible trafic comme à trafic croissant.

- **Disponibilité de jeux de données fiables**
  On part du principe que les bases de données publiques (géolocalisation, villes, etc…) sont disponibles, maintenues et de qualité suffisante.

- **Capacité des producteurs à s'auto-déclarer**
  Hypothèse : les producteurs peuvent renseigner eux-mêmes leurs infos produit via une interface dédiée (plutôt que tout passer par les collectivités).

- **Ouverture à des extensions tierces**
  On anticipe le besoin futur d'extensions partenaires (livraison, paiement, CRM). La plateforme est donc pensée pour exposer une couche API propre dès la V1.