SPÉCIFICATION DES CONDITIONS REQUISES POUR L'ARCHITECTURE

Projet Foosus géoconscient

Version 1.0

Auteur : Hervé Prevost - Architecte Logiciel - Foosus

Date version : 19/09/2022

Version : 1.0

SOMMAIRE

OBJET DU 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.

Il accompagne le Document de Définition de l'Architecture en permettant d'en fournir une vision plus détaillée.

MESURES DU SUCCÈS

Voici un rappel du tableau des métriques fourni dans le document Déclaration de Travail d'Architecture :

Indicateur	**Changement souhaité**
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

EXIGENCES D'ARCHITECTURE

Voici listées par catégorie les besoins exprimés par les parties prenantes concernant l'architecture cible :

Catégorie	**Id**	**Exigence**
Evolutivité	AR1	L'architecture devra être évolutive pour permettre à nos services de se déployer sur diverses régions à travers des villes et des pays donnés.
	AR2	Les livrables doivent pouvoir être fournis à intervalles réguliers pour que le nouveau système soit rapidement opérationnel et puisse être doté de nouvelles fonctionnalités au fil du temps.
	AR3	Offrir la possibilité de réorienter des solutions existantes, en expérimentant de nouvelles modifications et en facilitant l'intégration avec des partenaires internes et externes.
	AR4	Réduire le temps entre le développement et sa validation afin notamment d'analyser les réactions de nos clients vis-à-vis de nouvelles fonctionnalités au fur et à mesure de leur développement.
	AR5	Possibilité d'exécuter diverses variantes ou réaliser des comparaisons de différentes solutions auprès de nos utilisateurs.
	AR6	Besoin de visibilité sur la façon dont nos logiciels sont utilisés.
	AR7	Nous devons pouvoir inverser des décisions d'architecture tant que cela reste peu onéreux.

Disponibilité	AR8	Notre solution doit être disponible pour nos fournisseurs et nos consommateurs, où qu'ils se trouvent.
	AR9	Cette solution doit être utilisable avec des appareils mobiles et fixes.
	AR10	La solution doit tenir compte des contraintes de bande passante pour les réseaux cellulaires et les connexions Internet haut débit.
	AR11	Les améliorations et autres modifications apportées aux systèmes de production devront limiter ou supprimer la nécessité d'interrompre le service pour procéder au déploiement.
	AR12	Choisir une architecture permettant de limiter les risques techniques.
	AR13	Même si le système est surchargé, les utilisateurs connectés doivent pouvoir continuer à accéder à tous les services de façon dégradée.
	AR14	Chaque nouvelle version doit être de taille réduite, présenter peu de risques, être transparente pour nos utilisateurs et rester accessible en tout lieu et à tout moment.
Spécificité	AR15	La solution doit pouvoir prendre en charge différents types d'utilisateurs (par exemple, fournisseurs, back-office, consommateurs) avec des fonctionnalités et des services spécifiques pour ces catégories.
Simplicité	AR16	La solution doit offrir une expérience utilisateur de premier plan (simplicité, UX Design).

ACCORDS DE NIVEAU DE SERVICE

Les accords de niveau de service, imposés par Foosus pour satisfaire aux demandes des investisseurs, déterminent les niveaux de service à atteindre sur le système de production.

Contrats de service business

Ils doivent répondre aux exigences exprimées par le demandeur : combien de temps peut-on s'arrêter en cas de sinistre mineur (l'arrêt d'une machine par exemple) ou de sinistre majeur (destruction du datacenter) ?

Ce double besoin est souvent exprimé sous forme de DIMA ou Durée d'Interruption Maximum Admissible (RTO en anglais) en heures ou encore sous la forme d'un objectif de service (SLO en anglais) en pourcentage du type 99,999% du temps soit 5,26 minutes d'interruption par an.

Id.	Objectif de niveau de service	Mesure
B-SLA1	Le taux de disponibilité de la plateforme doit être > 99,7%. (- de 2h d'indisponibilité / mois)	Analyse des incidents entraînant une perte de disponibilité de l'application.
B-SLA2	La solution doit être utilisable convenablement sur l'ensemble des connexions internet mobile ou filaire > 2,5Mb (3G+)	Analyse régulière des temps de réponses des API sur les versions mobiles.
B-SLA3	La localisation géographique de l'utilisateur ne doit pas impacter les performances. Toute les actions (chargement de page ou action utilisateur) ne devront nécessiter un temps de réponse > 500 ms.	Plan de test régulier et analyse des temps de réponses des API sur les versions mobiles.
B-SLA4	La sécurité de la solution doit être totale. Aucune fuite ne devra être constatée.	Analyse des logs système et API.
B-SLA5	Le RPO (Recovery Point Objective - Temps de récupération de données maximal) devra être < 10min au maximum.	Plan de test.
B-SLA6	Le RTO (Recovery Time Objective – Temps d'indisponibilité de la plateforme suite à un incident) devra être < 1h au maximum.	Plan de test.

Contrats de service application

Id.	Objectif de niveau de service	Mesure
A-SLA1	Le délai de déploiement d'une fonctionnalité achevée doit être >= 7 jours.	Délai moyen entre merge-request sur une branche « master » et déploiement de la branche.
A-SLA2	Le taux de couverture de code par des tests automatisés devra être >= 75%.	Rapport de couverture de code par les tests.
A-SLA3	Le nombre de vulnérabilités / mauvaises pratiques de sécurité détectées dans les applications devront être égal à 0.	Analyse statique et code-review.
A-SLA4	Le code des logiciels fournis devra être en conformité avec les chartes de développement établi et les mesures d'analyse automatisé de la « dette technique » devront être < 1h / dépôt de code source.	Analyse statique et code review.
A-SLA5	Les logiciels livrés devront être totalement conformes aux principes de conception définis par la gouvernance d'architecture.	Validation des logiciels par la gouvernance d'architecture.
A-SLA6	Les navigateurs web disposant d'une part de marché > 2% des utilisateurs (Mesures Foosus ou GlobalStats, la plus haute étant retenue) mobile ou desktop seront supportés.	Analyse statistiques + Plan de test.

Outils de test de code utilisés

Tests unitaires : Jest/MochaJs/Karma/Jasmine
Tests fonctionnels : Puppeteer, cypress
Tests de montée en charge : gremlins.js/Chaos Monkey
Supervision : Uptime Robot

LIGNES DIRECTRICES POUR L'IMPLÉMENTATION

Architecture générale

Les lignes directrices d'implémentation fournissent le cadre général des principes à mettre en œuvre pour la construction et le déploiement de la nouvelle solution.

Une évaluation permanente de la cohérence et de la conformité de l'architecture devra être mise en place.

La conception du système devra s'effectuer dans le respect de l'état de l'art du développement.

Les technologies employées devront être standardisées et open-source. La stack technologique devra rester cohérente, devra disposer d'un support large et d'un écosystème complet et devra avoir été éprouvée.

Enfin le système doit posséder les caractéristiques principales d'un bon système d'information :

Sécurité

Du point de vue de l'infrastructure, celle-ci sera assurée via la mise en place d'environnements **cloud AWS**.

La sécurité s'appuie sur quatre principes qui peuvent être résumés à l'aide de l'acronyme suivant **DICT** : **Disponibilité, Intégrité, Confidentialité, Traçabilité** en y ajoutant la **PDMA**, indiquant la Perte de Données Maximum Admissible (**RPO** en anglais).

La sécurité c'est donc ce qui permet d'être conforme à la DICT et à la PDMA demandées.

Disponibilité

C'est le reflet de la capacité d'une application à être résistante à tout événement pouvant générer un arrêt de service.

Elle doit répondre aux exigences exprimées par le demandeur : combien de temps peut-on s'arrêter en cas de sinistre mineur (l'arrêt d'une machine par exemple) ou de sinistre majeur (destruction du datacenter) ?

Ce double besoin est souvent exprimé sous forme de **DIMA** ou Durée d'Interruption Maximum Admissible (**RTO** en anglais) en heures ou encore sous la forme d'un objectif de service (**SLO** en anglais) en pourcentage du type 99,999% du temps soit 5,26 minutes d'interruption par an.

On parle ici de mécanismes tels que le **clustering, la haute disponibilité, la redondance**.

La haute disponibilité sera atteinte grâce à la mise en place des architectures suivantes :

- Le **Load Balancing** (ou répartition de charge) permet de répartir les flux entrants sur plusieurs équipements et permet de limiter une surcharge ou un dysfonctionnement lors d'un pic de trafic ou de connexions simultanées d'utilisateurs. Les requêtes sont ainsi redistribuées de façon intelligente vers les équipements (serveurs) les moins chargés.
- Le **FailOver** (tolérance aux pannes) consiste à rediriger un utilisateur vers un autre serveur lorsque le serveur principal tombe en panne laissant le temps aux administrateurs système de régler le problème. L'intérêt de ce type d'architecture est de pouvoir délivrer un service en continu en s'appuyant sur la redondance des matériels mis en place (duplication d'un composant ou matériel par des éléments identiques).

- **Clustering** : alors que le Load balancing distribue la charge entre plusieurs serveurs pour améliorer les performances, le clustering combine plusieurs serveurs pour travailler comme s'il s'agissait d'un seul.

Intégrité

Comme son nom l'indique, il s'agit de s'assurer qu'une donnée n'est pas modifiée indûment,le plus souvent dans un cadre sensible ou légal.

Ici, on utilisera des mécanismes de hash et des dispositifs à écriture unique type **WORM (Write-once, read-many)**.

Confidentialité

On effectue une classification de la sensibilité des données, en fonction de l'impact de leur publication. On peut distinguer deux types de données confidentielles : celles très sensibles qui touchent au métier de l'entreprise et dégraderaient sa capacité à continuer son activité en cas de fuite ou de corruption et celles, réglementaires, qui légalement doivent être protégées, gardées secrètes ou supprimées (on pense à la **RGPD**, notamment).

Afin de s'assurer de la confidentialité, on mettra en place du chiffrement, que ce soit au niveau des flux (**https**) ou des données stockées (**7-zip**).

Traçabilité

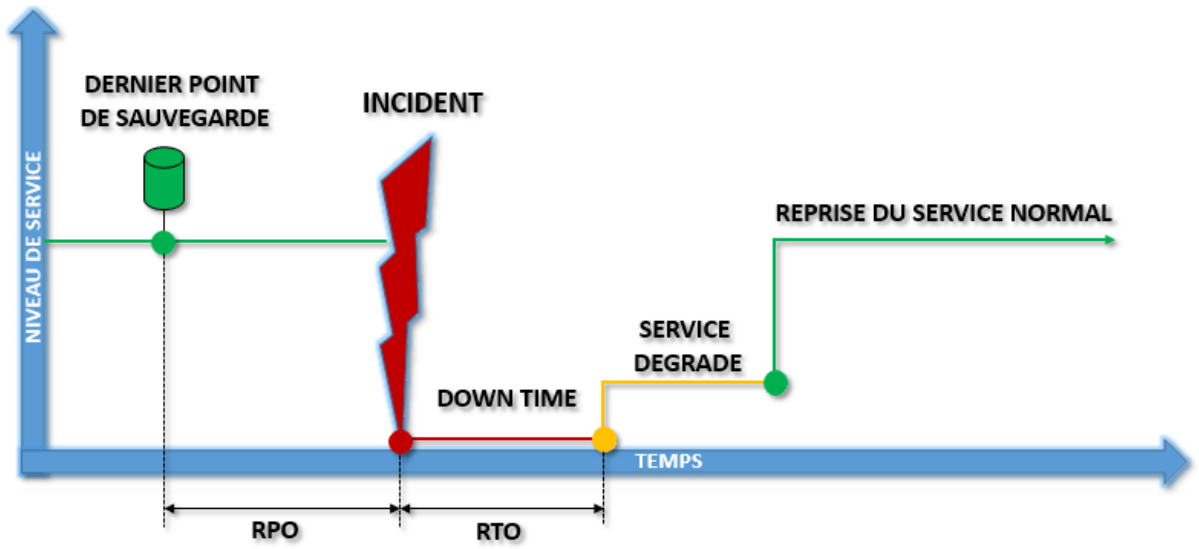
C'est la capacité à garder une preuve de l'accès ou de la modification d'une donnée, dans le temps. Cette notion est particulièrement pertinente pour les données réglementées ou celles, souvent confidentielles, qui ne doivent être consultées que par peu de personnes.

La traçabilité passe la plupart du temps par la mise en place de journaux (**logs**) répertoriant qui a accédé à quoi et ce qui a été fait.

PDMA – La Perte de Données Maximale Admissible

La combinaison des techniques liées à la redondance des systèmes permettra de garantir la haute disponibilité de l'infrastructure informatique et le fonctionnement en continu des services mais il y a également des risques de perte de données en cas de catastrophe naturelle, d' incendie ou autres entraînant une perte d'équipements. La redondance seule ne suffit donc pas. C'est pourquoi il faut mettre en place un système de sauvegarde régulier pour pallier ce risque. Les cas de corruption et surtout de suppression accidentelle sont assez courants et nécessitent d'être envisagés en amont lors de la conception d'un système.

Afin de mitiger ces risques, on s'appuiera sur des mécanismes de **réplication**, de **sauvegarde** et d'**externalisation** des données. A noter, seule la politique de sauvegarde permet d'assurer réellement une PDMA.



Evolutivité

Flexibilité

La flexibilité concerne la partie logiciel, elle doit être assurée d'un point de vue fonctionnel et technique.

L'évolutivité fonctionnelle est permise par la simplification du noyau en couches du back office. Tout fonctionnement non standard sortant du cadre du noyau sera développé via un micro-service. Cette approche permettra de consacrer les ressources aux développements à haute valeur ajoutée.

L'évolutivité technique sera assurée par la pérennité et l'indépendance de ses composantes :

- Utilisation systématique des briques les plus reconnues et disposant de la communauté la plus active dans chaque domaine.
- Découpage du code en composants autonomes, écrits dans un langage unique et de taille la plus réduite possible.

Elasticité

Composante majeure des architectures modernes, l'élasticité (scalabilité) est la capacité d'un système à répondre à des contraintes changeantes. On distingue en général deux types d'élasticité : une « verticale » dans laquelle on ajoute des ressources supplémentaires aux éléments existants (plus de processeurs, plus de mémoire, plus de disques…), l'autre « horizontale » où l'on ajoute de nouveaux éléments (un serveur web supplémentaire, un switch additionnel). Ces aspects sont gérés via la plateforme AWS.

Adaptabilité

Elle est la capacité d'un système à s'adapter dynamiquement au dispositif de consultation (tablette, smartphone…), au débit de la communication, aux personnes atteintes de troubles visuels, à la langue de l'utilisateur…. Ces aspects sont gérés en appuyant les développements sur des bibliothèques ou framework front tels que Bootstrap, JQuery …

Couplage lâche

Le couplage est la nature de l'adhérence qu'ont deux systèmes entre eux. Une architecture bien pensée doit permettre une adhérence faible avec ses interfaces externes (les autres applications utilisées) et internes (ses propres services, par exemple sa base de données).

Simplicité

Un système simple sera par essence compréhensible donc opérable, extensible et stable dans la durée.

Performance

La rapidité de fonctionnement d'un système est un élément clef de la satisfaction de l'utilisateur. Celle-ci sera assurée par la qualité des développements, l'utilisation de CDN pour les bibliothèques tierces, et enfin par la redondance géographique des systèmes à travers les régions AWS et la mise à disposition d'infrastructures à haute disponibilité.

Transparence

La possibilité de construire des rapports personnalisés à tout moment sur la globalité des données du système est un élément clef pour permettre une orientation stratégique efficace.

SPÉCIFICATIONS POUR L'IMPLÉMENTATION

Les spécifications d'implémentation de la solution Foosus ont pour objectif de définir les méthodes et outils mis en œuvre pour répondre aux lignes directrices définies dans la section précédente.

Catégorie	Id	Type	Spécification
Architecture	IS1	Gouvernance	Les missions principales de la gouvernance porteront sur la conception et le contrôle de conformité de l'architecture.
	IS2	Modèle	L'architecture en couche / orientée données est préconisée pour la partie BO, les fonctionnalités spécifiques seront établies selon une architecture en Microservices.
	IS3	Stack Technique	Front : Javascript/ Bootstrap / JQuery / REACT (JS et native)/ NextJS Back : Node / Express / PostGreSQL
Confidentialité	IS4	Contrôle d'accès	L'authentification à 2 facteurs (2FA) est préconisée.
	IS5	Transfert	Les données en transit seront cryptées selon TLS (https, sFTP, SSH ...)
	IS6	Chiffrement	Les données sensibles sont stockées de manière non lisible (7-zip).
	IS7	WAF	L'utilisation d'un pare-feu logiciel est préconisée.
Disponibilité	IS8	Plateforme CLoud	Utilisation d'une plateforme PAAS. L'écosystème AWS répond aux besoins définis. (Passerelle, Load Balancer, montée en charge dynamique)

	IS9	Redondance	Sur AWS, l'infrastructure est redondée en standard à raison de 2 zones de disponibilité par région.
	IS10	RTO / RPO	Le RPO après incident doit être < 10min. Le RTO après incident doit être < 1h.
	IS11	Upgrade	Les mises à jour seront déployables à chaud sans arrêt des services par bascule sur le serveur secondaire le temps de la mise à jour.
Evolutivité	IS12	Intégration continue	La solution doit disposer de processus d'intégration et de livraison continu, avec versionning Git et archivage sur GitLab.
	IS13	Tests	Les tests automatisés seront réalisés à l'aide des éléments suivants : Tests unitaires : Jest/MochaJs/Karma/Jasmine Tests fonctionnels : Pupeeteer Tests de montée en charge : gremlins.js/Chaos Monkey/Artillery Supervision : Uptime Robot
Performances	IS14	CDN	Appel individuel des plugins standard front sur des CDN répandus à accès rapide quelque soit la région (cloudflare...)
	IS15	Cache Applicatif	Utilisation de Memcached pour réduire les temps d'accès aux données et paramètres de l'application.
	IS16	Conception	Conception et développements selon les principes SOLID

	IS17	Traitements Asynchrones	Utilisation des possibilités asynchrones de NodeJs et Express pour l'architecture en couche..
	IS18	Responsive	Développement responsive de l'application pour une adaptation dynamique au dispositif de consultation.
	IS19	Réseaux lents	Transfert de données limité à leur plus simple expression, les pages sont construites dynamiquement sur le client à partir des paramètres compressés des pages.

STANDARDS POUR L'IMPLÉMENTATION

Les standards d'implémentation pour l'architecture décrivent un premier niveau de standards pour l'architecture qui seront appliqués tout au long du cycle de construction de la solution. Ces standards seront complétés par l'instance de gouvernance d'architecture.

Id.	**Nom**	**Description**
S1	Accès centralisé	Tout accès à un service disposant d'une interface API Web doit être effectué par l'intermédiaire de l'API Gateway.
S2	Authentification des accès et Cryptage des données.	Toutes les interfaces de service (API) doivent être construites selon les standards de l'industrie pour l'authentification utilisateur et les meilleures pratiques de sécurité.

		Toutes les données doivent être encryptées lors de leur transfert et leur stockage.
S3	Développement en intégration continue et méthodologies Agile	Tous les services et applicatifs doivent s'intégrer dans le workflow CI/CD.
S4	Environnements	Tous les services et applicatifs doivent être disponibles sur au moins 3 environnements : Développement / Qualification / Production.
S5	Log / Traçabilité	L'ensemble des accès au système et aux ressources doit générer des logs conservés durant à minima 30 jours.
S6	Design UX	L'utilisateur doit être au centre des développements. Un comité s'assure de la simplicité d'utilisation de l'application, son adaptabilité au device de consultation, à la langue ainsi qu'aux capacités réseau liées à la situation géographique.

EXIGENCES D'INTEROPÉRABILITÉ

Id.	**Nom**	**Description**
IR1	Format des échanges	Les interfaces API Web doivent être construites selon le principe de conception REST niveau 3.

IR2	Format des données en transit	Le format des données en transit doit être basé sur le langage JSON.
IR3	Encodage	L'encodage de tous les documents et chaînes de caractères doit être UTF-8.
IR4	Compatibilité	La compatibilité avec l'ancien système (format des échanges, nomenclature) doit être maintenue autant que possible pour faciliter la migration.

CADRE DE MANAGEMENT DU SERVICE IT

Le cadre de management du service IT vise à présenter les grandes lignes directrices de la collaboration entre les pôles de compétence du service.

Product management

La gestion produit doit être placée au cœur de l'activité du service. Chaque produit devra être affecté à un gestionnaire (Product Owner) assisté d'éventuels collaborateurs.

Le Product Owner est le garant des fonctionnalités et de la traduction des besoins. Il organise et priorise les lots de travail fournis aux équipes de développeurs.

Gouvernance d'architecture

La gouvernance d'architecture est assurée par l'architecte logiciel qui la partage avec les référents techniques de chaque équipe. La gouvernance d'architecture aura pour rôle de s'assurer de :

- De la définition de l'architecture (sur l'ensemble des aspects),
- De la définition des standards d'implémentation,
- De la conception à haut niveau des solutions,

- Du contrôle de la conformité,
- Du maintien du référentiel d'architecture.

Organisation des développements

Les équipes de développements s'organisent autour d'équipes de développement respectant les principes LEAN et Scrum. Chaque équipe de développement est spécialisé dans une pile technologique précise (généralement sur un domaine applicatif restreint).
Le travail quotidien s'effectue selon la méthodologie « SCRUM » visant à travailler par cycle itératif court.

Infrastructure et exploitation

L'exploitation de la plateforme est assurée par une équipe à forte compétence DevOps. L'équipe est chargée de mettre en place la plateforme, de définir les règles d'automatisation, de contrôler son bon fonctionnement et gère le contrôle d'accès aux ressources internes.

CONTRAINTES

Le catalogue ci-après liste les contraintes transmises par Fossus pour le chantier d'architecture.

Id.	**Contraintes**
C1	Le projet initial est approuvé pour un coût de 50 000 USD (45 190 €) et une période de 6 mois est prévue pour définir l'architecture et préparer un projet de suivi afin de développer un prototype.
C2	L'architecture doit permettre d'obtenir le meilleur rapport qualité-coût.

C3	L'architecture peut inclure de nouveaux composants personnalisés ou des composants du commerce pour favoriser la flexibilité, la stabilité et l'extensibilité.
C4	Les solutions open source sont préférables aux solutions payantes.
C5	Le support continu des composants doit être pris en compte lors de leur sélection ou lors des prises de décision de création ou d'achat.
C6	Toutes les solutions du commerce ou open source doivent, dans la mesure du possible, faire partie d'une même pile technologique afin de réduire les coûts de maintenance et de support continus.