

Document de définition d'architecture

Projet de streaming vidéo interactif



Auteur(s) et contributeur(s)

Nom & Coordonnées	Qualité & Rôle	Société
Gérald ATTARD	Architecte logiciel	Gibberish

Historique des modifications et des révisions

N° version	Date	Description et circonstance de la modification	Auteur
0.01	08/12/20--	Document de vision préliminaire	
1.0	08/10/2022	Création du document	Gérald ATTARD

Validation

N° version	Nom & Qualité	Date & Signature	Commentaires & Réserves
1.0	Alex Z Directeur technique		

Tableau des abréviations

Abr.	Sémantique
COTS	Commercial Off-The-Shell (trad. <i>application commerciale sur étagère</i>)
ESB	Entreprise service bus (trad. <i>bus de service aux entreprises</i>)
GOTS	Government Off-The-Shell (trad. <i>application gouvernementale sur étagère</i>)
OSSI	Officier de Sécurité des Systèmes d'Information
RPC	Remote Procedure Call (trad. <i>appel de procédure à distance</i>)
SI	Système d'Informations
SOA	Service-Oriented Architecture (trad. <i>architecture orientée services</i>)
SOAP	<i>Simple Object Access Protocol</i> (trad. <i>protocole d'accès aux objets simple</i>)
UDDI	Universal Description Discovery and Integration (trad. <i>découverte et intégration de la description universelle</i>)
WDSL	Web Services Description Language (trad. <i>langage de description des services Web</i>)
XML	Extensible Markup Language (trad. <i>langage de balisage extensible</i>)

Table des matières

I. Contexte, objectifs et contraintes.....	6
II. Principes architecturaux.....	7
II.A. Principes de données.....	9
II.A.1. D1 : Les données en tant qu'actif.....	9
II.A.1.a. Déclaration.....	9
II.A.1.b. Raisonnement.....	9
II.A.1.c. Conséquences.....	9
II.A.2. D2 : Le partage des données.....	10
II.A.2.a. Déclaration.....	10
II.A.2.b. Raisonnement.....	10
II.A.2.c. Conséquences.....	11
II.A.3. D3 : L'accès aux données.....	12
II.A.3.a. Déclaration.....	12
II.A.3.b. Raisonnement.....	12
II.A.3.c. Conséquences.....	12
II.A.4. D4 : Le dépositaire des données.....	13
II.A.4.a. Déclaration.....	13
II.A.4.b. Raisonnement.....	13
II.A.4.c. Conséquences.....	13
II.A.5. D5 : le vocabulaire commun et les définitions des données.....	14
II.A.5.a. Déclaration.....	14
II.A.5.b. Raisonnement.....	14
II.A.5.c. Conséquences.....	14
II.A.6. D6 : la sécurité des données.....	15
II.A.6.a. Déclaration.....	15
II.A.6.b. Raisonnement.....	15
II.A.6.c. Conséquences.....	15
II.B. Principes d'application.....	16
II.B.1. A1 : l'indépendance technologique.....	16
II.B.1.a. Déclaration.....	16
II.B.1.b. Raisonnement.....	16
II.B.1.c. Conséquences.....	17
II.B.2. A2 : le facilité d'utilisation.....	17
II.B.2.a. Déclaration.....	17
II.B.2.b. Raisonnement.....	17
II.B.2.c. Conséquences.....	18
II.C. Principe technologique.....	18
II.C.1. T1 : le changement basé sur les exigences.....	18
II.C.1.a. Déclaration.....	18
II.C.1.b. Raisonnement.....	18
II.C.1.c. Conséquences.....	18
II.C.2. T2 : la gestion du changement.....	19
II.C.2.a. Déclaration.....	19
II.C.2.b. Raisonnement.....	19
II.C.2.c. Conséquences.....	19
II.C.3. T3 : la maîtrise de la diversité technique.....	20

II.C.3.a. Déclaration.....	20
II.C.3.b. Raisonnement.....	20
II.C.3.c. Conséquence.....	20
II.C.4. T4 : l'interopérabilité.....	21
II.C.4.a. Déclaration.....	21
II.C.4.b. Raisonnement.....	21
II.C.4.c. Conséquences.....	21
III. Architectures.....	22
III.A. Architecture de base.....	22
III.A.1. Définition.....	22
III.A.2. Représentation organisationnelle et fonctionnelle.....	22
III.A.3. Représentation fonctionnelle.....	23
III.B. Architecture cible proposée par <i>Gibberish</i>	24
III.B.1. Définition.....	24
III.B.2. Représentation organisationnelle et fonctionnellement.....	24
III.B.3. Représentation technique.....	24
III.C. Architecture cible proposée par l'architecte logiciel.....	24
III.C.1. Représentation organisationnelle et fonctionnelle.....	24
III.C.2. Représentation technique.....	28
IV. Logique et justification de l'approche architecturale.....	30
V. Mappage vers le référentiel d'architecture.....	33
V.A. Architecture Orientée Services.....	33
V.A.1. Avantages.....	34
V.A.2. Inconvénients.....	34
V.A.3. Utilisation.....	34
V.A.4. Principes.....	35
V.A.4.a. Application d'un contrat standard.....	35
V.A.4.b. Réalisation d'un couplage faible.....	35
V.A.4.c. Abstraction des services.....	35
V.A.4.d. Développement de service réutilisable.....	36
V.A.4.e. Conception de service autonome.....	36
V.A.4.f. Certification de services sans état (stateless).....	36
V.A.4.g. Assurance de la <i>découvrabilité</i>	37
V.A.4.h. Mise en œuvre de la composabilité.....	37
V.A.4.i. Notions théoriques.....	38
V.A.4.i.i. I pour <i>Interesting</i>	38
V.A.4.i.ii. R pour <i>Reusable</i>	38
V.A.4.i.iii. A pour <i>Atomic</i>	39
V.A.4.i.iv. ESB pour <i>Enterprise Service Bus</i>	39
V.B. La norme ACID.....	41
V.B.1. Atomicité.....	41
V.B.2. Cohérence.....	42
V.B.3. Isolation.....	42
V.B.4. Durabilité.....	42
V.C. Le Règlement Général pour la Protection des Données.....	43
V.D. Évaluation de la réutilisation.....	44
V.D.1. Définition de la réutilisation.....	44
V.D.2. Indicateurs de réutilisation.....	44

VI. Analyse des écarts.....46

VII. Évaluation de l’impact.....47

VIII. Architecture de transition.....48



I. Contexte, objectifs et contraintes

Créée en 2009, la société *Gibberish* est une société basant sa politique d'entreprise sur le maintien à jour de ses technologies pour produire, puis vendre, des vidéos toujours plus innovantes et attractives.

En appliquant cette politique à ce projet de streaming de vidéo interactive, les actionnaires souhaitent changer la façon avec laquelle le public appréhende le visionnage de vidéos et l'utilisation ludique de contenu interactif.

Aussi, avec l'appui de ces mêmes actionnaires, ce projet a pour objectif de mettre en œuvre l'organisation, l'infrastructure et les outils nécessaires à la réalisation de trois grands domaines :

- la production de séquences vidéo, au contenu interactif, offrant des fonctionnalités inédites de changement d'angle de visionnage, de navigation temporelle au sein de l'Histoire ou, encore, de téléchargement et de partage de contenu vidéo créé de façon personnalisée ;
- l'utilisation de toutes les possibilités offertes par le contenu interactif des vidéos produites, à partir duquel l'utilisateur pourra décider, lui-même, de la tournure de l'Histoire qu'il souhaite voir se dérouler ;
- la vente de vidéos source au contenu interactif aux différents clients (utilisateurs) de la société.

Pour réaliser les objectifs définis ci-dessus, le cahier des charges a d'ores et déjà défini les fonctionnalités à réaliser pour satisfaire les différentes parties prenantes décisionnaires.

Aussi, en partant de cette base définie par le cahier des charges, le présente document aura pour principal objectif de définir l'architecture à mettre en place pour répondre à ces attentes.

De par sa politique de modernisation continue, la société *Gibberish* a laissé tout le champs de recherche nécessaire à ce projet pour ne se fermer aucune option de recherche.

En ce sens, il sera tout à fait possible d'établir une architecture sur des COTS existants ou de définir des tâches fonctionnelles nécessitant un développement particulier, dévolu à ses propres équipes de développement internes.

Enfin, en plus de proposer une solution d'architecture fonctionnelle, ce document proposera aussi un modèle de conception, ainsi que les étapes à suivre pour concrétiser la solution conceptuelle proposée.



II. Principes architecturaux

Les principes d'architecture définissent les règles et directives générales sous-jacentes pour l'utilisation et le déploiement de toutes les ressources et actifs informatiques de *Gibberish*. Ils refléteront un niveau de consensus entre les différents éléments de cette entreprise et constitueront la base de la prise de décisions informatiques futures.

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

Les principes d'architecture sont utilisés pour capturer les vérités fondamentales sur la façon dont *Gibberish* utilisera et déploiera les ressources et les actifs informatiques.

Ainsi, ces principes architecturaux seront utilisés de différentes manières :

- fournir un cadre dans lequel *Gibberish* peut commencer à prendre des décisions conscientes concernant l'architecture d'entreprise et les projets qui implémentent l'architecture d'entreprise cible ;
- établir des critères d'évaluation pertinents, exerçant ainsi une forte influence sur la sélection de produits, de solutions ou d'architectures de solutions dans les étapes ultérieures de la gestion de la conformité à l'architecture d'entreprise ;
- définir les exigences fonctionnelles de l'architecture ;
- contribuer à l'évaluation des implémentations existantes et du portefeuille stratégique, pour la conformité avec les architectures définies. Ces évaluations fourniront des informations précieuses sur les activités de transition nécessaires à la mise en œuvre d'une architecture, à l'appui des objectifs et des priorités de l'entreprise ;
- mettre en évidence la valeur commerciale des implémentations conformes au principe et fournir des conseils pour les décisions difficiles avec des moteurs ou des objectifs contradictoires ;
- fournir un aperçu des tâches clés, des ressources et des coûts potentiels pour l'entreprise de suivre le principe. Ainsi, les déclarations d'implication dans un principe d'architecture devront répondre à des questions relatives aux contributions de futures initiatives de transition et autres activités de planification ;
- soutenir les activités de gouvernance de l'architecture permettant de :
 - fournir un "back-stop" pour les évaluations de conformité de l'architecture standard lorsqu'une certaine interprétation est autorisée ou requise ;
 - soutenir la décision d'initier une demande de dérogation lorsque les implications d'une modification particulière de l'architecture ne peuvent être résolues dans le cadre de la procédure d'exploitation locale.

De plus, les principes architecturaux peuvent être interdépendants et devront être appliqués comme un ensemble. En effet, il est probable que des principes rivaliseront parfois, par exemple, les principes d'« *accessibilité* » et de « *sécurité* » tendent vers des décisions contradictoires. En tout état de cause, chaque principe doit être considéré dans le contexte de "*toutes choses étant égales par ailleurs*".

Parfois, une décision sera nécessaire quant au principe qui prévaudra sur une question particulière. La justification de ces décisions se verra alors toujours documentée pour étayer le choix

Une réaction courante à la première lecture d'un principe est "*c'est évident et n'a pas besoin d'être documenté*". Le simple fait qu'un principe semble aller de soi ne signifie pas que les indications de ce principe soient suivies : avoir des principes qui semblent évidents aide à garantir que les décisions suivent réellement le résultat souhaité.

Bien que des sanctions spécifiques ne soient pas prescrites dans un énoncé de principes architecturaux, les violations de ces principes causent généralement des problèmes opérationnels et entravent la capacité de l'organisation à remplir sa mission.

Ainsi, au sein de cette étude, trois domaines de principes seront à prendre en compte :

- principes de données ;
- principes d'application ;
- principes technologiques.

Ces trois domaines de principes seront à appliquer au sein de toutes les entités métier au travers du SI associé.

II.A. Principes de données

II.A.1. D1 : Les données en tant qu'actif

II.A.1.a. Déclaration

Les données sont un actif propre à *Gibberish* et sont gérées en conséquence.

II.A.1.b. Raisonement

Les données sont une ressource d'entreprise précieuse qui ont une valeur réelle et mesurable.

La finalité des données est d'aider à la prise de décision.

Des données précises et opportunes sont essentielles pour prendre des décisions précises et opportunes.

Les données de *Gibberish* doivent être gérées avec soin, autant en terme de localisation qu'en terme d'intégrité.

II.A.1.c. Conséquences

Il faudra dans un premier temps fournir des efforts internes relatifs à l'organisation des données.

Pour apprendre aux collaborateurs de *Gibberish* à bien gérer ses propres données, des actions de formations seront organisées dès l'embauche des collaborateurs, et des séances de sensibilisation de leurs utilisations seront organisées périodiquement pour l'ensemble des collaborateurs.

En outre, il sera nécessaire à *Gibberish* d'identifier un OSSI qui sera garant de la gestion des données. Ce rôle devra s'assurer d'effectuer une transition culturelle d'une pensée « *propriété des données* » vers une pensée « *intendance des données* ».

Ce rôle sera essentiel pour l'intégrité des données car des données obsolètes, incorrectes ou incohérentes pourraient être transmises au personnel de *Gibberish* et affecter négativement les décisions.

Le *service Client* de *Gibberish* sera responsable de la Qualité des données. Ce service devra développer et utiliser des processus de garantie de saisie pour prévenir et corriger les erreurs dans les informations. En complément, ce service devra identifier et améliorer les processus qui produisent des informations erronées.

La qualité des données et des mesures prises devront être mesurées pour améliorer l'ensemble de la Qualité des données détenues par *Gibberish* - il est probable que des politiques et des procédures devront également être élaborées à cet effet.

Il est conseillé de mettre en place, au sein de *Gibberish*, un forum avec une représentation complète, à l'échelle de l'entreprise, afin de décider des changements de processus suggérés par le *service Client* et/ou l'OSSI.

En considérant que les données représentent un actif de valeur pour *Gibberish*, des rôles de « *gestionnaires de données* », responsables de la bonne gestion des données, devront être identifiés au niveau de l'entreprise.

II.A.2. D2 : Le partage des données

II.A.2.a. Déclaration

Les collaborateurs de *Gibberish* ont accès aux données nécessaires à l'exercice de leurs fonctions.

Par conséquent, les données sont partagées à toutes les fonctions et les organisations de l'entreprise, en tenant compte du critère « *Besoin d'en connaître* ».

II.A.2.b. Raisonnement

L'accès rapide à des données précises est essentiel pour améliorer la qualité et l'efficacité de la prise de décision de *Gibberish*.

Il est moins coûteux de conserver des données précises et actualisées, puis de les partager, que de conserver des données en double dans plusieurs emplacements.

Dans le contexte de *Gibberish*, tous les services détiennent chacun une mine de données. Néanmoins, ces données sont stockées dans des différentes bases de données cloisonnées, et pas toujours compatibles.

La vitesse de collecte, de création, de transfert et d'assimilation des données dépend de la capacité de *Gibberish* à partager efficacement ces îlots de données dans l'ensemble de son organisation.

Les données partagées se traduiront par de meilleures décisions en diminuant drastiquement le nombre de sources de données. Ces sources seront alors gérées de façon plus précise et opportune pour l'ensemble des parties prenantes de décision.

Les données partagées électroniquement se traduiront par une efficacité accrue lorsque les entités de données existantes pourront être utilisées, sans ressaisie, pour créer de nouvelles entités.

II.A.2.c. Conséquences

Seul le *service Client* sera à même de partager des données en tenant compte du critère « *Besoin d'en connaître* ».

Les collaborateurs exerçant leurs fonctions au sein de ce service devront recevoir une formation particulière relative au « *Besoin d'en connaître* ».

L'action de mise à disposition d'une information au profit d'un autre collaborateur de *Gibberish* sera tracé au sein d'un journal d'accès.

En outre, pour permettre le partage des données, *Gibberish* devra développer et respecter un ensemble commun de politiques, de procédures et de normes régissant la gestion et l'accès aux données à court et à long terme.

À court terme, pour préserver son investissement dans les systèmes hérités, *Gibberish* devra investir dans des logiciels capables de migrer les données du système hérité vers un environnement de données partagé. Pour le développement et l'utilisation de modèles de données standard, la création d'éléments de données, telles que des métadonnées, est fortement conseillée. Ainsi, il sera alors possible de définir un environnement partagé et de développer un système de référentiel pour stocker les données et les métadonnées associées, afin de les rendre accessibles.

À long terme, à mesure que les anciens systèmes seront remplacés, *Gibberish* devra adopter et appliquer des politiques et des directives communes d'accès aux données. Ainsi, les collaborateurs, en tant qu'utilisateur de nouvelles applications, devront garantir que les données de ces nouvelles applications demeurent disponibles pour l'environnement partagé et que les données de cet environnement partagé puissent continuer à être utilisées par les nouvelles applications.

À court et à long terme, *Gibberish* adoptera des méthodes et des outils **communs** pour créer, maintenir et accéder à ses données partagées.

Le partage de données nécessitera un changement culturel important.

Ce principe de partage de données « *se heurtera* » probablement au principe de sécurité des données. Il sera alors nécessaire et indispensable de considérer qu'en aucun cas le principe de partage de données ne devra entraîner la compromission de données confidentielles.

Les données mises à disposition pour le partage devront être utilisées par tous les collaborateurs, en fonction du « *besoin d'en connaître* », pour exécuter leurs tâches respectives.

Cela garantira que seules les données les plus précises et les plus récentes seront utilisées pour la prise de décision.

Les données partagées deviendront la « *source unique virtuelle* » de toutes les données à l'échelle de l'entreprise.

II.A.3. D3 : L'accès aux données

II.A.3.a. Déclaration

Les données sont accessibles aux collaborateurs pour exercer leurs fonctions.

II.A.3.b. Raisonnement

Un large accès aux données conduit à l'efficacité et à l'efficacité de la prise de décision.

Cela permet également une réponse rapide aux demandes d'informations et à la prestation de services.

L'utilisation des informations doit être considérée du point de vue de *Gibberish* pour permettre l'accès à une grande variété de collaborateurs. Cet aspect sera à prendre en compte lors du processus de prospection des clients.

La recherche d'une données utilise du Temps et le Temps du personnel est économisé si la cohérence des données est améliorée.

II.A.3.c. Conséquences

L'accessibilité implique la facilité avec laquelle les collaborateurs obtiennent des informations.

Les méthodes d'accès et la manière dont les informations sont accessibles et affichées doivent être suffisamment adaptables pour répondre à un large éventail de collaborateurs.

L'accès aux données ne constitue pas une compréhension propre des données – chaque collaborateur doit veiller à ne pas mal interpréter les informations.

L'accès aux données n'accorde pas au collaborateur des droits d'accès pour modifier ou divulguer les données – la modification d'une donnée sera une action réservée au *service Client* de *Gibberish*.

Ce principe nécessitera un processus d'éducation et un changement dans la culture organisationnelle qui soutient actuellement une croyance en la « *propriété* » des données par les différents services de *Gibberish*.

II.A.4. D4 : Le dépositaire des données

II.A.4.a. Déclaration

Chaque élément de données a un administrateur responsable de la Qualité des données.

II.A.4.b. Raisonnement

Comme il est indiqué dans un autre principe architectural, l'un des avantages d'un environnement architectural est la possibilité de partager des données (du texte, de la vidéo, du son...) dans toute l'entreprise.

À mesure que le degré de partage des données augmente et que les différents services s'appuient sur des informations communes, il devient essentiel que seul l'administrateur des données prenne des décisions sur le contenu des données.

En effet, les données peuvent perdre leur intégrité lorsqu'elles sont saisies plusieurs fois, le dépositaire des données devra donc être le seul responsable de la saisie des données, ce qui élimine les efforts humains et les ressources de stockage de données redondants. Seuls les collaborateurs exerçant au sein du *service Client* seront habilités à saisir ou modifier une donnée.

Il sera alors nécessaire d'identifier deux rôles différents et complémentaires de gestion des données :

- **Responsable de l'intégrité** : ce rôle s'assurera de l'exactitude et de l'actualité des données ;
- **Responsable de la sécurité des données** : ce rôle a des implications plus larges et inclut des tâches de normalisation et de définition des données.

II.A.4.c. Conséquences

La notion de *tutelle réelle* résout les problèmes de "*propriété*" des données et permet à celles-ci d'être disponibles pour répondre aux besoins des collaborateurs de *Gibberish*.

Cette notion implique un changement culturel de la notion de « *propriété* » des données à la notion de « *tutelle* » des données.

Le *service Client* de *Gibberish* sera responsable du respect des exigences de qualité imposées sur les données.

Il est essentiel que le *service Client* ait la capacité de donner confiance aux collaborateurs dans les données sur la base critères concrets, tels que la source de données unique.

Cela ne signifie pas que les sources classifiées seront révélées ni que la source en sera le dépositaire. Les informations devront être saisies électroniquement une seule fois et immédiatement validées aussi près que possible de la source.

Des mesures de contrôle de la Qualité devront être mises en place pour assurer l'intégrité des données.

En raison du partage de données, le *service Client* sera à la fois *comptable* et *responsable* de l'exactitude et de l'actualité de ses éléments de données désignés et, par la suite, devra alors reconnaître l'importance de cette responsabilité de tutelle.

II.A.5. D5 : le vocabulaire commun et les définitions des données

II.A.5.a. Déclaration

Les données sont définies de manière cohérente au sein de *Gibberish*, et les définitions sont compréhensibles et disponibles pour tous les collaborateurs.

II.A.5.b. Raisonnement

Les données qui seront utilisées doivent avoir une définition commune à l'ensemble de l'entreprise pour permettre le partage des données.

Un vocabulaire commun facilitera également les communications et permettra un dialogue efficace.

De plus, il sera nécessaire d'interfacer les systèmes et d'échanger des données.

II.A.5.c. Conséquences

La clé du succès des efforts visant à améliorer l'environnement de l'information sera d'attribuer des fonctions dont l'intitulé retranscrit fidèlement les fonctions du poste.

Ceci est distinct mais lié à la question de la définition des éléments de données – il faudra définir à la fois un vocabulaire ET une définition.

Ainsi, *Gibberish* établira le vocabulaire commun initial pour chacun de ses métiers et la définition associée sera utilisée uniformément dans toute l'entreprise.

Chaque fois qu'une nouvelle définition de données est requise, l'effort de définition sera coordonné et réconcilié avec le "glossaire" de *Gibberish* des descriptions de données.

Le *service Client* assurera cette coordination. Les ambiguïtés résultant de multiples définitions paroissiales des données doivent céder la place à des définitions et à une compréhension acceptées à l'échelle de l'entreprise

Les initiatives de normalisation des données seront coordonnées par le *service Client* pour qui des responsabilités fonctionnelles en matière d'administration des données seront attribuées.

II.A.6. D6 : la sécurité des données

II.A.6.a. Déclaration

Les données sont protégées contre toute utilisation et divulgation non autorisées.

En plus des aspects traditionnels de la classification de sécurité nationale, cela inclut, mais sans s'y limiter, la protection des informations pré-décisionnelles et sensibles à la sélection de sources exclusives.

II.A.6.b. Raisonnement

Le partage ouvert d'informations et la diffusion d'informations via la législation pertinente devront être mis en balance avec la nécessité de restreindre la disponibilité d'informations classifiées, exclusives et sensibles.

Les lois et réglementations existantes exigent la sauvegarde de la sécurité nationale et de la confidentialité des données.

Les informations pré-décisionnelles (travail en cours, dont la diffusion n'est pas encore autorisée) devront être protégées pour éviter les spéculations injustifiées, les interprétations erronées et les utilisations inappropriées.

II.A.6.c. Conséquences

L'agrégation des données, classifiées ou non, créera une cible nécessitant des procédures d'examen et de déclassification pour maintenir un contrôle approprié.

En tant que propriétaire de données, *Gibberish* devra déterminer si leur agrégation entraînera une augmentation du niveau de classification. Cette décision sera orientée par les apports de l'OSSI.

Une politique et des procédures appropriées seront nécessaires pour gérer cet examen et cette déclassification.

L'accès à l'information fondé sur une politique du « *Besoin d'en connaître* » forcera des examens réguliers de l'ensemble de l'information par le *service Client*.

La pratique actuelle consistant à avoir des systèmes séparés pour contenir différentes classifications doit être repensée. Ainsi, il sera approprié de choisir un moyen pour séparer les données classifiées au travers d'une solution logicielle éprouvée et reconnue.

En vue de son évolution vers la production et la vente de vidéos au contenu interactif, la solution matérielle actuelle pourrait être considérée comme lourde, inefficace et coûteuse ; de plus, il est très coûteux de gérer des données non classifiées sur un système classifié.

Actuellement, la seule façon de combiner les deux est de placer les données non classifiées sur le système classifié, où elles doivent demeurer.

Afin de fournir un accès adéquat à des informations ouvertes tout en maintenant des informations sécurisées, les besoins de sécurité doivent être identifiés et développés au niveau des données, et non au niveau du SI.

Des mesures de sécurité des données devront être mises en place pour restreindre l'accès à « *afficher uniquement* » ou « *ne jamais voir* ». L'étiquetage de sensibilité pour l'accès aux informations pré-décisionnelles, décisionnelles, classifiées, sensibles ou exclusives doit être déterminé.

La Sécurité doit être intégrée aux éléments de données dès le départ.

Les systèmes, les données et les technologies devront être protégés contre l'accès et la manipulation non autorisés.

Les informations de *Gibberish* se doivent d'être protégées contre toute altération, sabotage, catastrophe ou divulgation accidentelle ou non autorisée.

L'OSSI fournira de nouvelles politiques sur la gestion de la durée de protection des informations pré-décisionnelles, en tenant compte de la fraîcheur du contenu.

II.B. Principes d'application

II.B.1. A1 : l'indépendance technologique

II.B.1.a. Déclaration

Les applications sont indépendantes des choix technologiques spécifiques et peuvent donc fonctionner sur une variété de plates-formes technologiques.

II.B.1.b. Raisonnement

L'indépendance des applications par rapport à la technologie sous-jacente permet aux applications d'être développées, mises à niveau et exploitées de la manière la plus rentable et la plus opportune.

De par sa politique d'amélioration technologique constante, *Gibberish* ne doit pas utiliser de vivier de technologies hétérogènes et disparates, pas toujours compatibles les unes avec les autres.

L'objectif sera donc de normaliser l'emploi de ces technologies sujettes à une obsolescence continue et à la dépendance vis-à-vis des fournisseurs, afin d'éviter que leur utilisation ne devienne le moteur en lieu et place des besoins des utilisateurs eux-mêmes.

Sachant que chaque décision prise en matière d'informatique rend dépendant de cette technologie, l'intention de ce principe est de garantir que le logiciel d'application ne dépend pas de matériel et de logiciels de systèmes d'exploitation spécifiques.

II.B.1.c. Conséquences

Ce principe nécessitera des normes qui prennent en charge la portabilité. A cela, il faudra également tenir des comptes du fait que les informations détenues par *Gibberish* doivent rester accessibles et affichables sans considération de technologie spécifique.

Pour les COTS et les GOTS, les choix actuels peuvent être limités, car bon nombre de ces applications dépendent de la technologie et de la plate-forme.

Des interfaces de sous-système devront être développées pour permettre aux applications héritées d'interagir avec les applications et les environnements d'exploitation développés dans le cadre de l'architecture d'entreprise.

Le middleware doit être utilisé pour découpler les applications de solutions logicielles spécifiques

À titre d'exemple spécifique au contexte technologique de *Gibberish*, ce principe pourrait conduire à la généralisation de l'utilisation de Java® en tant que langage *back-end*, afin d'accorder une grande priorité à l'indépendance de la plate-forme.

En ce qui concerne les notions de *front-end*, l'utilisation de *React.js*®, *Angular*® ou *Vue.js*® seront préconisés.

II.B.2. A2 : le facilité d'utilisation

II.B.2.a. Déclaration

Les applications de *Gibberish* doivent être faciles d'utilisation.

La technologie sous-jacente doit être transparente pour les collaborateurs, afin qu'ils puissent se concentrer sur leurs tâches à accomplir.

II.B.2.b. Raisonnement

Plus un collaborateur doit comprendre la technologie sous-jacente, moins il est productif.

La facilité d'utilisation est une incitation positive à l'utilisation des applications qui encourage les collaborateurs à travailler dans l'environnement d'information intégré, au lieu de développer des systèmes isolés pour accomplir la tâche en dehors du SI de l'entreprise.

L'utilisation d'une application doit être intuitive.

II.B.2.c. Conséquences

Les applications devront avoir une apparence et une convivialité communes et répondre à des exigences ergonomiques définies par *Gibberish*, au travers, par exemple, d'une charte graphique.

Par conséquent, la norme d'apparence commune doit être conçue et des critères de test d'utilisabilité doivent être développés.

Les lignes directrices pour les interfaces utilisateur ne doivent pas être limitées par des hypothèses étroites sur l'emplacement de l'utilisateur, la langue, la formation aux systèmes ou la capacité physique.

Des facteurs tels que la linguistique, les infirmités physiques du collaborateur (acuité visuelle, capacité à utiliser le clavier/la souris) et la maîtrise de l'utilisation de la technologie ont de vastes ramifications pour déterminer la facilité d'utilisation d'une application.

II.C. Principe technologique

II.C.1. T1 : le changement basé sur les exigences

II.C.1.a. Déclaration

Ce n'est qu'en réponse aux besoins de *Gibberish* que des modifications devront être apportées aux applications et à la technologie.

II.C.1.b. Raisonnement

Ce principe favorisera une atmosphère où l'environnement de l'information change en réponse aux besoins de *Gibberish*, plutôt que de voir l'entreprise changer en réponse aux changements informatiques.

Il s'agit d'assurer l'objectif du support d'information - la transaction commerciale - est la base de tout changement proposé.

Les effets imprévus sur l'activité dus aux changements informatiques seront minimisés.

Un changement de technologie peut fournir une opportunité d'améliorer le processus métier et, par conséquent, de modifier les besoins de l'entreprise.

II.C.1.c. Conséquences

Les changements dans la mise en œuvre suivront un examen complet des changements proposés à l'aide de l'architecture d'entreprise.

Il ne devrait exister de financement pour une amélioration technique ou un développement de système que s'il n'existe un besoin commercial proprement argumenté et documenté.

Des processus de gestion du changement conformes à ce principe devront être élaborés et mis en œuvre.

Ce principe peut se heurter au principe du changement réactif, néanmoins *Gibberish* se doit de s'assurer que le processus de documentation des exigences n'entrave pas les changements réactifs pour répondre aux besoins commerciaux légitimes.

L'objectif de ce principe est de maintenir l'accent sur les besoins commerciaux, et non sur les besoins technologiques – en fonction du contexte, un changement réactif peut être considéré comme un besoin commercial.

II.C.2. T2 : la gestion du changement

II.C.2.a. Déclaration

Les modifications apportées à l'environnement d'information de *Gibberish* seront mises en œuvre en temps opportun.

II.C.2.b. Raisonnement

Le fait de demander à ses collaborateurs de travailler dans un environnement informatique défini oblige *Gibberish* à répondre à leurs besoins grâce à cet environnement informatique.

II.C.2.c. Conséquences

Les processus de gestion et de mise en œuvre du changement seront développés avec l'objectif de ne pas créer de retard.

Un collaborateur qui ressent un besoin de changement devra se connecter avec un "*expert métier*" pour faciliter l'explication et la mise en œuvre de ce besoin.

Si des modifications doivent être apportées, l'architecture doit être maintenue à jour.

L'adoption de ce principe pourrait nécessiter des ressources supplémentaires et cela pourrait même entrer en conflit avec d'autres principes (par exemple, avantage maximal à l'échelle de l'entreprise, applications à l'échelle de l'entreprise, etc.).

II.C.3. T3 : la maîtrise de la diversité technique

II.C.3.a. Déclaration

La diversité technologique est contrôlée pour minimiser le coût non négligeable du maintien de l'expertise et de la connectivité entre plusieurs environnements de traitement.

II.C.3.b. Raisonnement

Il existe un coût réel et non négligeable de l'infrastructure requise pour prendre en charge les technologies alternatives pour les environnements de traitement.

D'autres coûts d'infrastructure sont encourus pour maintenir l'interconnexion et la maintenance de plusieurs constructions de processeurs.

Limiter le nombre de composants pris en charge simplifiera la maintenabilité, réduira les coûts et surtout limitera les erreurs ou anomalies quant à l'utilisation de ces environnements.

Les avantages commerciaux d'une diversité technique minimale comprennent :

- un conditionnement standard des composants ;
- un impact prévisible de la mise en œuvre ;
- des évaluations et des rendements prévisibles ;
- des tests redéfinis ;
- un statut d'utilité ;
- une flexibilité accrue pour s'adapter aux progrès technologiques.

La technologie commune à l'ensemble de *Gibberish* apportera des avantages économiques à l'échelle de l'entreprise.

Les coûts d'administration technique et de support seront mieux maîtrisés avec des ressources limitées permettant au responsable informatique de se concentrer sur cet ensemble de technologies partagées.

II.C.3.c. Conséquence

Les politiques, les normes et les procédures qui régissent l'acquisition d'une technologie doivent être directement liées à ce principe.

Les choix technologiques doivent être limités par les choix disponibles dans le plan technologique.

Des procédures visant à augmenter l'ensemble de technologies acceptables, pour répondre aux exigences en constante évolution, devront être élaborées et mises en place.

La base technologique ne doit pas être figée et les avancées technologiques, considérés comme bienvenues, ne devront pas modifier en profondeur le modèle technologique de l'infrastructure actuelle. Cette avancée technologique ne devra en aucun cas freiner l'amélioration de l'efficacité opérationnelle ou nécessiter une capacité requise dispendieuse.

II.C.4. T4 : l'interopérabilité

II.C.4.a. Déclaration

Les logiciels et le matériel doivent être conformes aux normes définies qui favorisent l'interopérabilité des données, des applications et de la technologie.

II.C.4.b. Raisonnement

Les normes aident à assurer la cohérence, améliorant ainsi la capacité à :

- gérer les systèmes ;
- améliorer la satisfaction des clients ;
- protéger les investissements informatiques existants ;
- maximiser les retours sur investissement ;
- réduire les coûts.

Les normes d'interopérabilité aident, en outre, à assurer le support de plusieurs clients pour leurs produits et, ainsi, faciliter l'intégration de la chaîne d'approvisionnement de données pour *Gibberish*.

II.C.4.c. Conséquences

Les normes d'interopérabilité et les normes de l'industrie seront suivies, à moins qu'il n'y ait une raison commerciale impérieuse de mettre en œuvre une solution non standard.

Un processus pour établir des normes, les examiner, les réviser périodiquement et accorder des exceptions devra être établi.

Les plateformes informatiques existantes seront identifiées et documentées.



III. Architectures

III.A. Architecture de base

III.A.1. Définition



Pour compléter cette section, cette étude se basera sur la définition suivante :

« l'architecture de base est l'ensemble de produits qui décrivent l'entreprise existante, les pratiques commerciales actuelles et l'infrastructure technique. »

Aussi, la description de l'architecture de base sera constituée de différents éléments, autant structurels qu'organisationnels ou techniques, relevés AVANT la mise en place de la nouvelle architecture de migration.

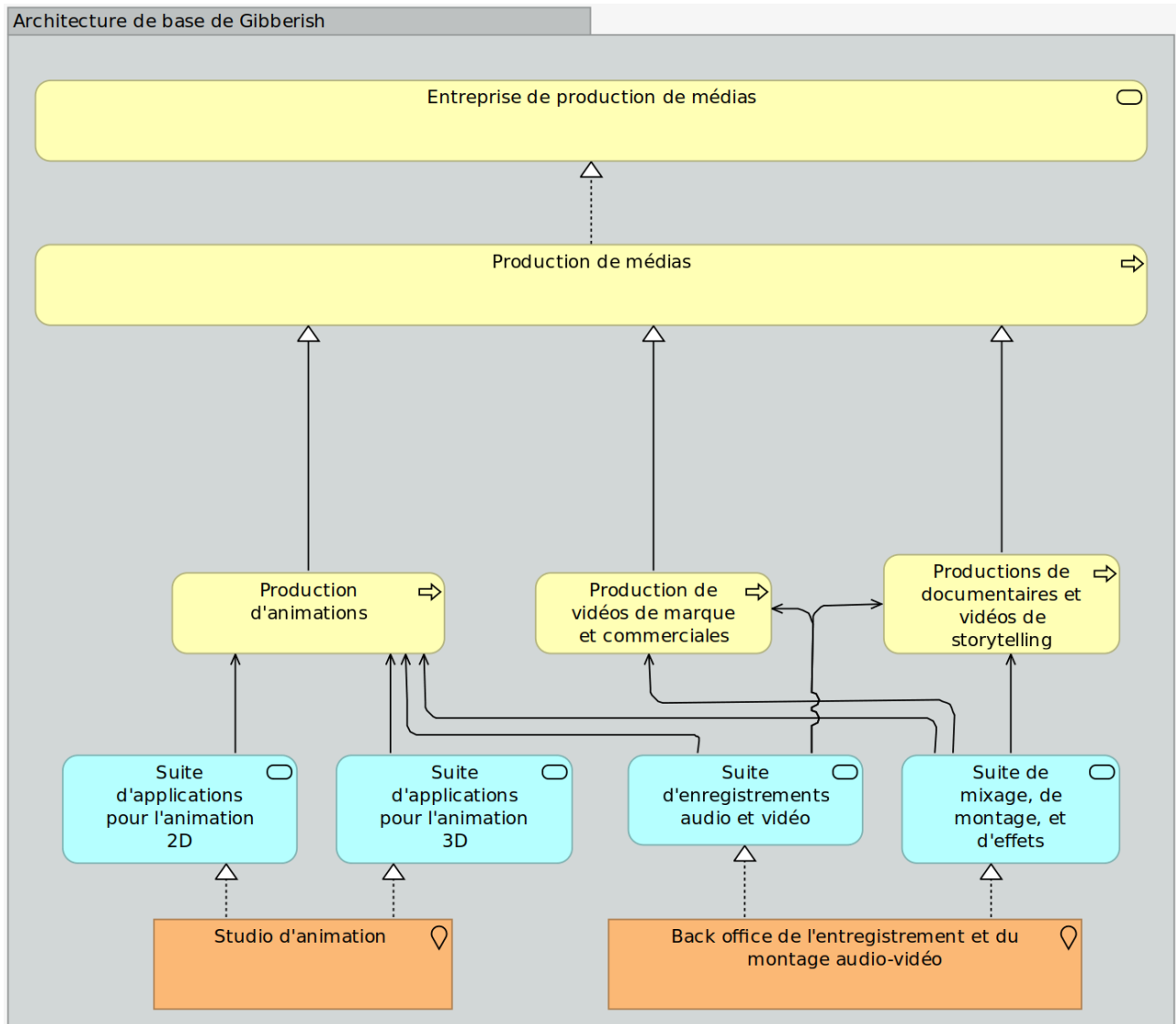
III.A.2. Représentation organisationnelle et fonctionnelle

La société *Gibberish* dispose actuellement de deux entités réalisant l'intégralité des vidéos qu'elle propose :

- un  **studio d'animation** :
 - cette entité est chargée de produire toutes les animations des différentes séquences vidéo ;
 - au moyen de suites applicatives logicielles spécialisées, le studio réalise les animations, aussi bien 2D que 3D, de toutes les séquences vidéos nécessitantes ;
- un  **département back office** en charge de :
 - produire les vidéos en mixant, puis en montant l'ensemble des séquences nécessitant des effets vidéos ;
 - enregistrer les séquences ainsi produites en synchronisant les bandes audio et vidéo de l'ensemble des séquences.

III.A.3. Représentation fonctionnelle

Tel qu'il a été décrit textuellement dans le paragraphe précédent, le diagramme suivant propose une représentation graphique de l'architecture actuelle de *Gibberish* :



III.B. Architecture cible proposée par Gibberish


III.B.1. Définition

Pour compléter cette section, cette étude se basera sur la définition suivante :

« *L'architecture cible du système est un modèle holistique des applications nécessaires pour satisfaire aux besoins d'entreprise et pour soutenir les processus cible.* »

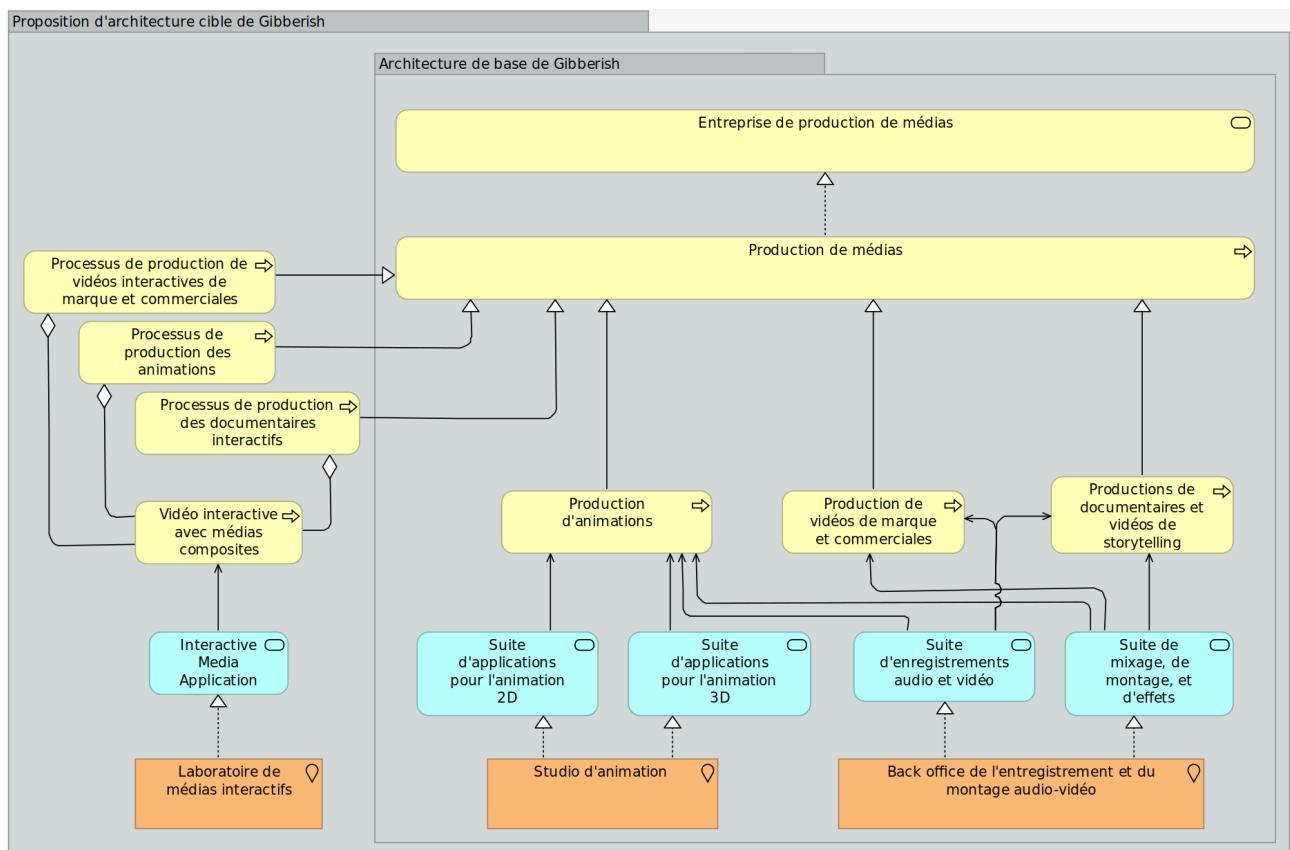
Tout comme l'architecture de base, la description de l'architecture cible sera constituée de différents éléments, autant structurels qu'organisationnels ou techniques, relevés APRÈS la mise en place de la nouvelle architecture de migration.

III.B.2. Représentation organisationnelle et fonctionnellement

L'architecture cible proposée par *Gibberish* prend en compte l'architecture de base, décrite précédemment, et la complète à l'aide d'une entité supplémentaire, nommée  **Laboratoire de médias interactifs**. Cette dernière aura alors la charge de produire du contenu vidéo interactif pour toutes les vidéos produites par *Gibberish*, et nécessitant l'inclusion d'éléments interactifs au sein de celles-ci.

III.B.3. Représentation technique

Le diagramme ci-dessous reprend la représentation graphique de l'architecture de base et la complète avec celle de l'architecture cible proposée par *Gibbersih* :

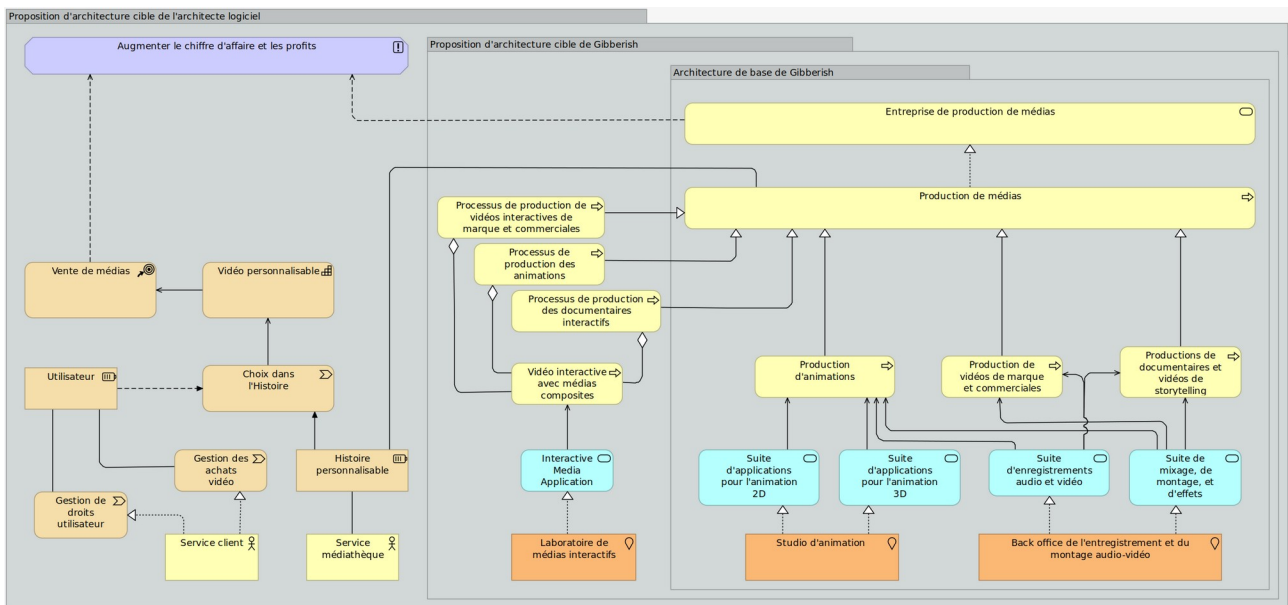


III.C. Architecture cible proposée par l'architecte logiciel

III.C.1. Représentation organisationnelle et fonctionnelle

L'architecture cible proposée par *Gibberish* ne cible que la production de vidéo au contenu interactif. Néanmoins, cette architecture ne tient compte ni des processus commerciaux ni des processus d'utilisation des contenus interactifs.

Aussi, cette étude proposera, en premier lieu, d'ajouter deux entités chargées de la mise en œuvre des processus d'utilisation des vidéos interactives. Les caractéristiques fonctionnelles de ces deux entités, nommées *Service médiathèque* et *Service client*, sont représentées dans le diagramme suivant :



Le premier élément à prendre en considération dans le diagramme ci-dessus est l'**objectif** issu du diagramme d'objectifs présenté dans le *cahier des charges*, à savoir **Augmenter le chiffre d'affaire et les profits**, à partir d'une ligne de conduite **Vente de médias** et du processus métier **Entreprise de production de médias**.





Associé à l'objectif précédent, deux autres éléments viennent compléter le diagramme ci-dessus :

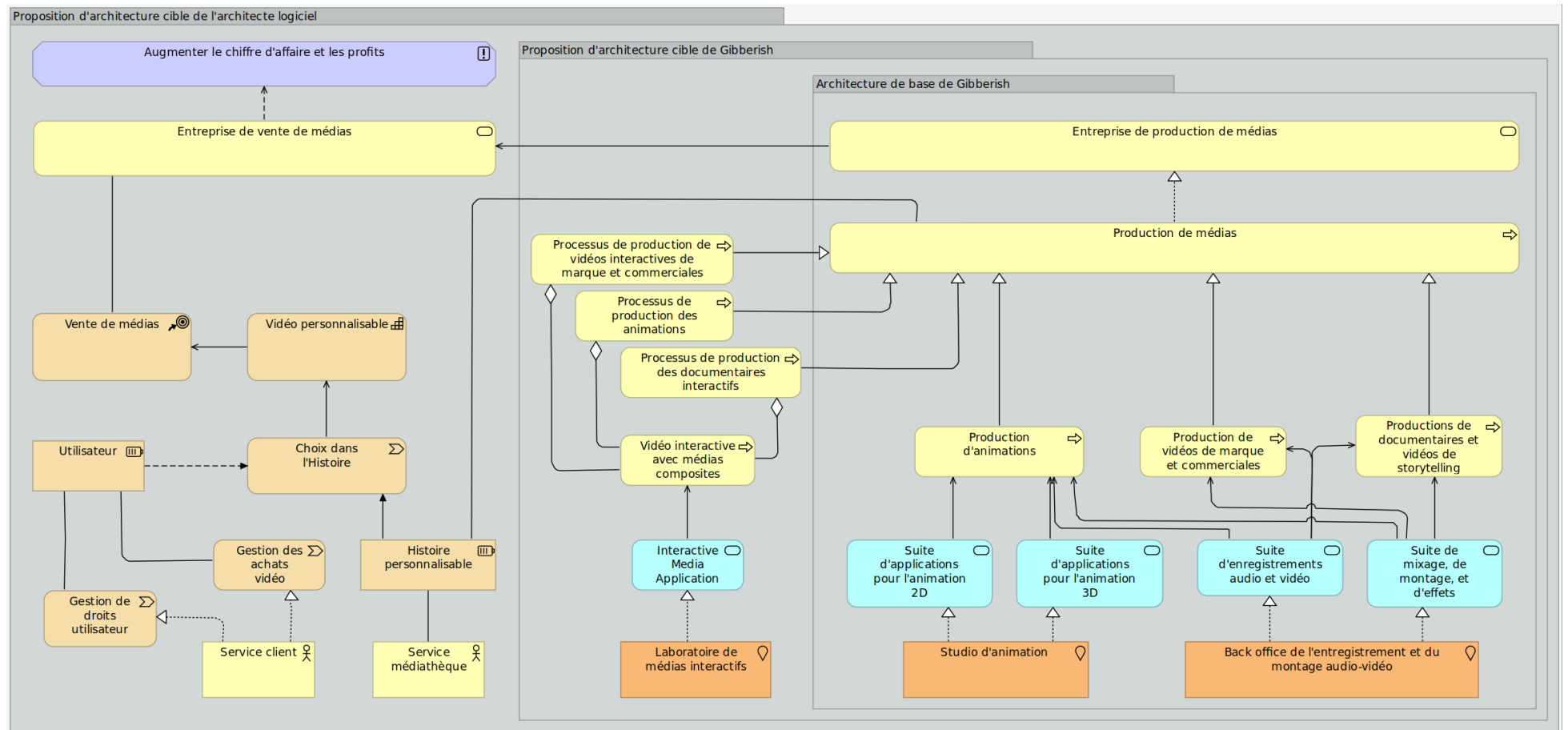
- le **service médiathèque** qui a la charge de gérer l'intégralité des vidéos produites l'entité *Production de médias* ;
- le **service client** qui a pour responsabilité de gérer les droits des utilisateurs et aussi les vidéos achetées par chaque utilisateur.

Il est à noter qu'un utilisateur sera considéré, dans la représentation ci-dessus, comme une **ressource** qui aura la possibilité de faire des choix, au sein de chaque vidéo, pour créer une Histoire qui lui est propre, de par les choix opérés.





Néanmoins, tout comme il est représentée au sein de l'architecture de base par l'entité *Entreprise de production de médias*, cette proposition d'architecture cible souffre de l'absence d'une entité globale gérant les entités *Service client* et *Service médiathèque*.

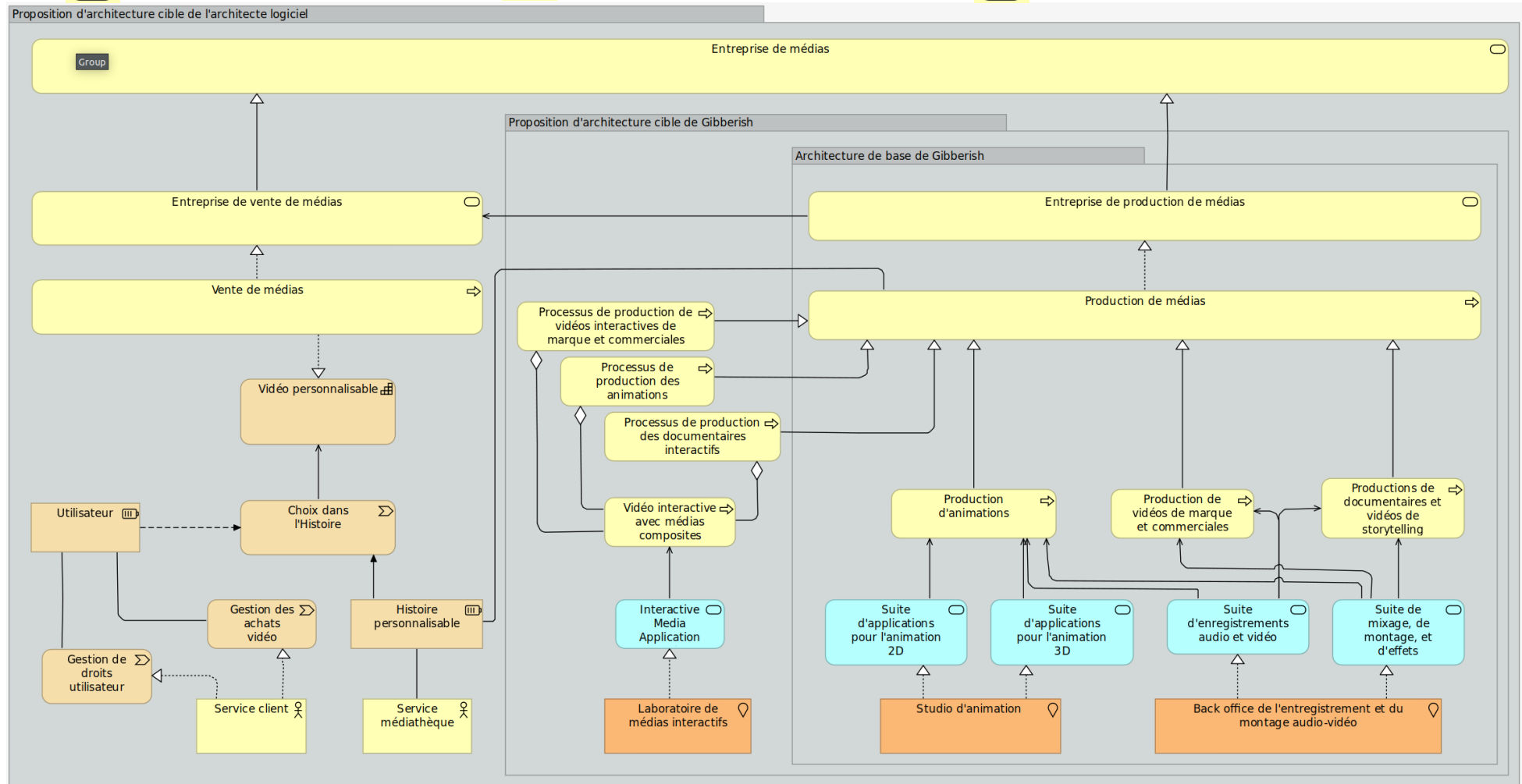
DIFFUSION RESTREINTE

A l'instar du processus métier  *Entreprise de production de médias*, le processus métier de gestion des acteurs métiers,  *Service médiathèque* et  *Service client*, sera nommée  *Entreprise de vente de médias* et pourra être représenté comme suit :



DIFFUSION RESTREINTE

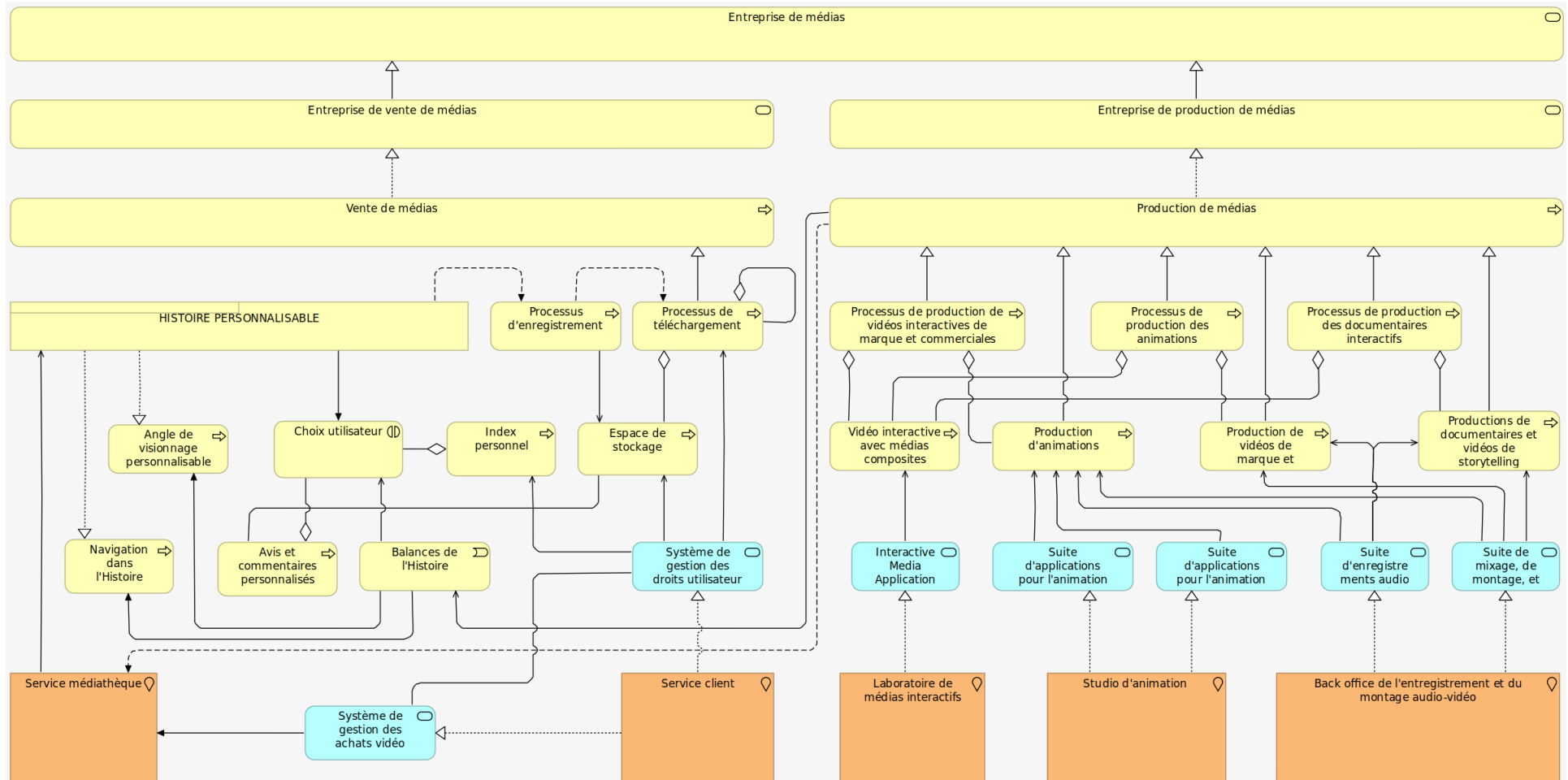
Enfin, pour que la solution proposée par cette étude s'aligne avec l'organisation de *Gibberish*, cette étude proposera de transformer la ligne de conduite  **Vente de médias** en processus métier du même nom. De plus, la solution proposée ajoutera un service métier, nommé  **Entreprise de médias**, pour gérer  l'**Entreprise de vente de médias** et  l'**Entreprise de production de médias** :



DIFFUSION RESTREINTE

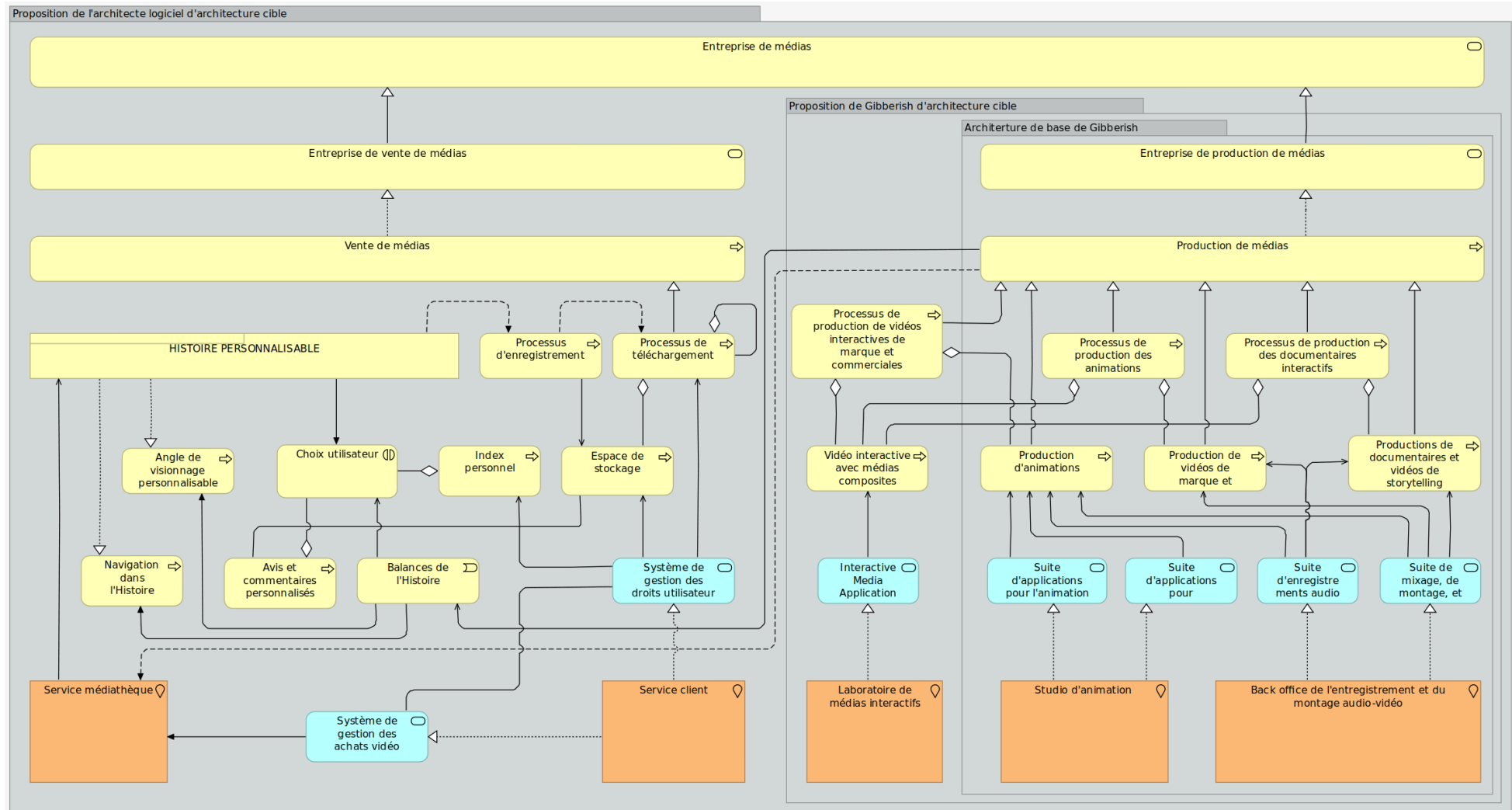
III.C.2. Représentation technique

La représentation technique ci-dessous expose tous les  processus, les  interactions, les  événements métiers et leurs différentes relations issus des entités purement fonctionnelles présentées dans le paragraphe précédent :



DIFFUSION RESTREINTE





Enfin, pour faire une transition cohérente avec la représentation organisationnelle et fonctionnelle présentée dans le paragraphe précédent, le diagramme suivant découpe les fonctionnalités techniques en trois domaines relatifs au proposition des différentes parties prenantes :




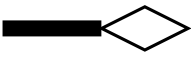





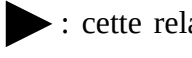


IV. Logique et justification de l'approche architecturale

Suite aux paragraphes précédents, cette étude a mis en évidence une importante utilisation de services indépendants, présentés sous forme de processus métier au sein de la représentation technique du paragraphe §Représentation technique de la solution préconisée.

Ainsi, dans son approche conceptuelle, cette étude a décomposé chaque  **flux de valeur** fonctionnel en  **processus métier** spécialisés, reliés par des  **événements métier** découlant d'  **interaction utilisateur**.

En outre, les différents liens de relation entre chacune des entités sus-mentionnées sont de plusieurs types :

- l'**association**  : cette relation permet de mettre deux entités quelconques en relation, indiquant ainsi un lien entre elles sans en préciser la nature ;
- l'**agrégation**  : cette relation implique que l'entité enfant peut exister indépendamment de l'entité parent ;
- la **généralisation**  : cette relation est un mécanisme combinant les types d'objet de même nature en un seul ;
- la **spécialisation**  : cette relation est le mécanisme inverse de la *généralisation*, c'est à dire qu'il est possible de créer une sous-entité à partir d'une entité *mère* existante, de même nature ;
- le **déclenchement**  : cette relation indique qu'une entité va en déclencher une autre, de façon événementielle ;
- le **service**  : cette relation indique qu'une entité va être au service d'une autre, en termes de fournisseur de ressources ;
- la **réalisation**  : cette relation indique qu'une entité est la source qui va engendrer/réaliser une autre ;
- le **flux**  : cette relation indique qu'une entité a un rapport de flux avec une autre, en termes temporels, c'est à dire d'antériorité.

A partir des éléments présentées ci-dessus et des principes architecturaux mentionnés précédemment, *Gibberish* se devra d'appliquer différentes bonnes pratiques afin de faire face à :

- l'explosion du volume de données collectées, produites et stockées ;
- la persistance des données relative au développement de la logique de silos, due en partie à l'essor de technologies innovantes ;
- l'augmentation de la part des données dites non-structurées ;
- le « *durcissement* » de la réglementation sur la protection des données personnelles (RGPD) qui impose une bonne gestion des données clients que l'entreprise traite (au sens large).

Ainsi, dans le contexte de *Gibbersish*, bien gérer les données clients sera un des principaux axes stratégiques pour cette entreprise.

Une stratégie solide de gestion des données client fera la différence en permettant de :

- mieux connaître le comportement et les préférences de ses clients,
- devenir plus pertinente dans les produits proposés, dans la manière de les promouvoir, dans sa relation clients au sens large et ses prises de décisions stratégiques.

La gestion des données client désignera ainsi la stratégie et l'ensemble des procédures mises en œuvre pour gérer et analyser les données clients collectées, produites, stockées et utilisées par *Gibberish*.

L'objectif sera de mieux comprendre les besoins, attentes, préférences et comportements des clients en vue d'améliorer leur rétention et leur satisfaction.

Cet objectif transformera le gisement des données clients en intelligence et en connaissance client, et sera géré par un ensemble de services inter-opérant les uns avec les autres.

Aussi, au sein de chacun de ces services, il sera nécessaire de mettre en place des processus spécifiques afin de gérer toutes les données à disposition de ceux-ci, selon les critères suivants:

- **catégorisation** : les données devront être classées en catégories et sous-catégories afin de les organiser et les structurer pour éviter toute erreur d'interprétation ;
- **correction** : chaque service sera responsable de vérifier la précision et la cohérence des données collectées et de les mettre à jour, quand c'est nécessaire, aidant ainsi à l'identification des données dupliquées ou obsolètes ;
- **enrichissement** : chaque service sera responsable d'enrichir les données incomplètes ;
- **collecte** : chaque service organisera la collecte des données dont il a besoin à partir de différents systèmes et différentes sources.
- **organisation** : les données seront organisées et partagées par tous les services, ainsi l'un des grands enjeux derrière cette gestion de données sera l'**unification des données** et leur **centralisation**.

Comme cela a été relaté précédemment, les volumes de données collectés et ou produits seront appelés à exploser, puisque plus de clients est synonyme de plus de données et aussi de plus d'outils/services proposés : l'utilisation de services spécialisés sera alors nécessaire pour décomplexifier cette masse d'informations associée à cette explosion de la quantité de données.

L'ensemble de ces service s'appuiera sur une architecture de type SOA pour :

- gérer les données clients en les rendant fiables et utiles ;
- rendre l'intégralité des services indépendants les uns des autres ;
- permettre à tous les services définis de répondre aux questions suivantes :
 - comment indiquer à un service où trouver les fonctionnalités dont il a besoin et comment y faire appel ?
 - Réponse : il faut établir un contrat (format d'un document XML appelé WSDL) entre les service qui explique clairement comment fonctionne chaque service.
 - comment l'ensemble des services peuvent trouver l'ensemble des fonctionnalités à disposition ?
 - Réponse : il suffit de centraliser tous les documents WSDL dans un serveur qui tient un annuaire de ceux-ci, appelé UDDI.
 - comment faire communiquer des services aux langages hétérogènes et aux technologies variées ?
 - Réponse : il suffit d'utiliser des technologies XML couplé à un protocole SOAP pour fixer les règles et les bonnes pratiques d'échange de messages entre services.
 - comment faire pour créer des messages conformes pour faire communiquer tous les services entre eux ?
 - Réponse : tous les langages de nouvelle génération comprennent leurs propres structures de gestion de fichier XML et de norme SOAP ; ainsi, dans les faits, il n'y aura jamais lieu d'en développer de spécifique.



V. Mappage vers le référentiel d'architecture

V.A. Architecture Orientée Services

Tel qu'il a été annoncé dans le paragraphe précédent, le mappage vers le référentiel d'architecture choisi sera effectuée à l'aide d'une architecture orientée services (SOA).

La SOA est un modèle de conception largement utilisé, regroupant des services modulaires qui « *dialoguent* » entre eux pour prendre en charge les applications et leur déploiement par l'intermédiaire d'un service Bus.

Les files d'attente de messages assurent donc la coordination entre ces applications distribuées. Elles peuvent considérablement simplifier le codage des applications découplées, fluidifier les pics de charge, tout en améliorant les performances, la fiabilité et l'évolutivité.

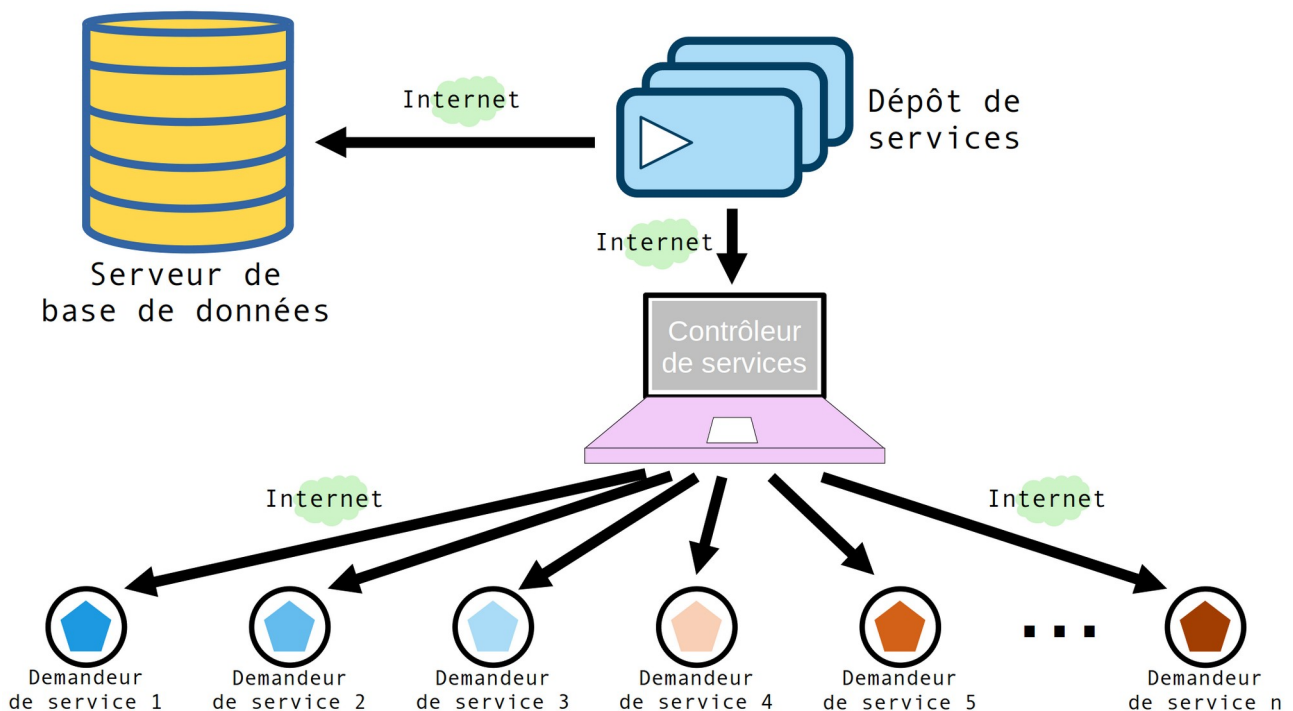
Cette architecture est une forme d'architecture de médiation qui est un modèle d'interaction applicative mettant en œuvre des **services** (composants logiciels) :

- avec une forte cohérence interne (par l'utilisation d'un format d'échange pivot, le plus souvent XML ou JSON) ;
- des couplages externes « lâches » (par l'utilisation d'une couche d'interface interoperable, le plus souvent un service web).

Ce terme, apparu dans les années 2000 concernait à l'origine essentiellement les problématiques d'interopérabilité syntaxique en relation avec les technologies d'informatique utilisées en entreprise.

Cette conception a évolué pour désigner maintenant le sous-ensemble particulier d'architecture de médiation en fonction de la technologie disponible.

Le schéma ci-dessous représente une architecture orientée services :



Dans le schéma précédent, il est possible de dénombrer quatre parties principales :

- Le **dépôt de services (*service repository*)** : il s'agit d'une bibliothèque de services conçue pour répondre à des demandes d'informations externes. L'information fournie est généralement un petit élément, comme un numéro, un mot, quelques variables, etc. Par exemple, un numéro de vol, un numéro de suivi de colis, le statut d'une commande (une lettre), etc. Cette bibliothèque est généralement documentée de manière très détaillée, car des applications externes font appel aux fonctions qu'elle contient ;
- Le **contrôleur de services (*service controller*)** : ce module communique les informations contenues dans le dépôt de services web aux demandeurs de services. Lorsqu'un demandeur de service externe appelle une certaine fonction du dépôt de services web, le contrôleur de services interprète la demande et recherche la fonction dans le dépôt de services. Il exécute ensuite cette fonction et renvoie une valeur au demandeur ;
- Le **serveur de base de données (*Database Server*)** : ce serveur contient les tables, les index et les données gérés par l'application. Les recherches et les opérations d'insertion, de suppression et de mise à jour sont exécutées ici ;
- Les **demandeurs de services (*Service Requester*)** : il s'agit d'applications externes qui demandent des services au dépôt de services par l'intermédiaire d'Internet, comme une organisation demandant des informations sur les vols à une compagnie aérienne, ou une autre entreprise demandant à un transporteur la localisation d'un colis à un moment donné.

V.A.1. Avantages

Ce modèle permet de collaborer avec des acteurs externes sans les laisser accéder au système interne. Ainsi, les fournisseurs de services permettent aux clients d'obtenir les informations nécessaires aux services qu'ils proposent, tout en se réservant l'accès à son système interne.

Ainsi, les entreprises peuvent partager avec le monde entier, de manière ordonnée et contrôlée, la sélection des fonctions qu'elles choisissent de laisser à disposition de leurs clients.

V.A.2. Inconvénients

Les services peuvent représenter une faiblesse au niveau du système d'informations pour les pirates qui veulent engorger ce système. Certaines formes d'attaques sont des « *dénis de service* ». Elles consistent à demander le même service des millions de fois par seconde, jusqu'à ce que le serveur tombe en panne. Il existe aujourd'hui une technologie permettant de résoudre ce problème, mais c'est toujours un problème à prendre en compte dans les architectures orientées services.

V.A.3. Utilisation

Ce modèle d'architecture pourra se rencontrer, par exemple, au sein de service de mise à disposition de vidéos à télécharger, d'informations sur les vols ou encore au sein des convertisseurs de devises.

Ainsi, le mappage vers le référentiel d'architecture présenté ci-dessus devra suivre les principes et notions décrits dans les paragraphes suivants.

V.A.4. Principes

V.A.4.a. Application d'un contrat standard

il sera primordial que les documents WSDL soient standardisés au maximum afin de faciliter leur réutilisabilité vis à vis services. Cette standardisation pourra impliquer, par exemple, des conventions de nommage standardisées, des types de données standards et toutes autres bonnes pratiques visant la standardisation de l'utilisation de ressources.

V.A.4.b. Réalisation d'un couplage faible

Tous les services devront être indépendants et faiblement couplés aux autres composants du SI. Ceci se concrétisera en veillant à ce que tous les échanges de service avec le monde extérieur se fassent uniquement via SOAP. Le fait d'aller interroger directement un service pourra, par exemple, créer une dépendance entre ces deux services. Ainsi si demain le service appelé directement change d'adresse ou est supprimé, le service appelant ne pourra plus être utilisable non plus : c'est pour cela que le couplage doit être le plus minimaliste possible. Le seul couplage toléré dans une SOA est celui avec un UDDI contenant vos contrats publiés.

V.A.4.c. Abstraction des services

Les contrats publiés devront TOUJOURS indiquer le minimum vital nécessaire à l'invocation du service ciblé ; il sera donc nécessaire de ne JAMAIS publier d'information relative au fonctionnement interne dudit service.

Toute information non utile directement à l'invocation du service pourrait être utilisée par d'autres services et générer des réponses inattendues et inappropriées.

Cet état de fait crée ici un couplage fort car les autres services du SI deviendrait alors dépendants de l'implémentation réalisée : **SITUATION A ÉVITER ABSOLUMENT.**

Imaginons, par exemple, qu'une fonction permettant d'avoir la liste complète des clients soit exposée. Pour obtenir cette liste, une méthode interne, prenant un paramètre "*limit*" fixé à -1, retournerait la liste complète de tous les clients. Le fait de publier cette fonction interne dans le WSDL pourrait « tenter » d'autres services qui veulent avoir juste les 10 premiers clients à appeler votre fonction interne : le résultat serait donc inapproprié et dangereux quant à son utilisation potentielle.

En conclusion, il sera TOUJOURS nécessaire et suffisant de publier le strict minimum l'invocation d'un service ; techniquement et par ordre de priorité, le *scope* d'une fonction sera préférentiellement *private* ou *protected*.

V.A.4.d. Développement de service réutilisable

La réutilisabilité d'un service se mesure à son indépendance vis à vis de la logique de son processus métier.

En d'autres termes, le service devra retourner les réponses les plus neutres possibles afin que celles-ci soient réutilisables dans différents autres composants du SI.

Par exemple, si un service retourne une liste de produits avec leurs détails (nom, image, description, prix, etc.), celui-ci pourra être appelé par une application web affichant des promotions pendant la période de Noël, ou par un tout autre service retournant simplement tous les produits en boutique...

En conclusion, il sera toujours nécessaire et suffisant de ne retourner, autant que faire se peut, que des données neutres. En d'autres termes, c'est au service appelant d'appliquer sa propre logique métier et de manipuler les données retournées pour les adapter à son cas d'utilisation, ce n'est pas au service fournisseur de ressources d'appliquer une quelconque logique métier.

V.A.4.e. Conception de service autonome

Lors du développement d'un service, il faudra toujours faire en sorte qu'il soit le plus autonome possible, c'est-à-dire qu'il dispose de ses propres ressources, librairies, conteneurs, etc.

En termes de SOA, un bon service est un service pouvant être, par exemple, embarqué dans une archive *war* ou *jar*, et fonctionner dans n'importe quelle type de machine, de façon indépendante.

V.A.4.f. Certification de services sans état (stateless)

Lors du développement d'un service, il sera nécessaire de veiller qu'aucune requête ne dépende de la réponse d'une autre.

Si, par exemple, un service gère le panier d'un e-commerce, il serait envisageable d'implémenter le fonctionnement suivant :

- Un service externe appelle le service de panier en ajoutant un produit A pour le client Robert, puis fait un autre appel pour ajouter un produit B pour le même client.
- Une session gérant le panier de cet utilisateur est alors gardée en mémoire de façon à ce que quand le client souhaite passer la commande, le service externe passe la simple requête "commander" et comme la session possède en mémoire tout ce qu'il faut pour traiter le nouveau panier, il suffit d'en enregistrer la commande.

Ce service a donc maintenu une session en **état** pendant une période assez longue, et le service externe comptait sur cet état précis pour passer la commande.

Un des inconvénients de cette approche est que s'il existe 2000 clients sur la boutique, la mémoire vive sera saturée avec 2000 sessions actives. De plus, ce service devra rester actif pour maintenir les sessions ce qui réduit la **scalabilité** du système : en d'autres termes il serait souhaitable de pouvoir ajouter des instances à ce service et d'en supprimer en fonction du trafic. Sauf que comme ce service a un état, il est difficile de supprimer une de ces instances sans perdre des paniers.

La solution à ce problème est représentée par un processus plus léger : quand un service externe demande à ajouter un produit A au client Robert, il suffira alors d'enregistrer cette donnée dans le système de gestion des données (base de données par exemple) et de retourner l'identifiant du panier. Ainsi, quand le service externe souhaitera ajouter un autre produit ou passer la commande, il sera alors de sa responsabilité de passer l'identifiant du panier sur lequel il faut faire l'opération. ; n'importe quelle instance de service pourra alors répondre à une telle demande.

Ce type de service est appelé **stateless** ou **sans état** in french ;)

V.A.4.g. Assurance de la découvrabilité

Tel qu'énoncé précédemment, un service devra toujours pouvoir être trouvé facilement par les autres services grâce à la publication de son contrat (WSDL) dans un annuaire de type UDDI.

Le WSDL servira à décrire :

- le protocole de communication (*SOAP RPC* ou *SOAP orienté message*)
- le format de messages requis pour communiquer entre tous les services ;
- les méthodes publiques à disposition que le client pourra invoquer ;
- la localisation du service.

En outre, une description WSDL sera représentée au sein d'un document XML commençant par la balise <definitions> et contenant les balises suivantes :

- <binding> : définit le protocole à utiliser pour invoquer le service ;
- <port> : spécifie l'emplacement effectif du service ;
- <service> : décrit un ensemble de points finaux du réseau.

V.A.4.h. Mise en œuvre de la composabilité

Ce principe est l'aboutissement des sept principes précédents. Maintenant qu'un service à couplage faible, abstrait, réutilisable, autonome, stateless et découvrable est créé, il va falloir profiter de toutes ces avantages au sein du SI.

Ainsi, ce service participera maintenant à la composition d'une application ou d'un SI plus large en proposant ses fonctions à différents composants.

La réutilisabilité du service devra alors être exploitée au maximum afin d'optimiser le système final.

V.A.4.i. Notions théoriques

Les principes énoncés précédemment décrivent la façon de se représenter un service au sein d'une architecture SOA. Néanmoins, il est nécessaire maintenant d'appréhender cette architecture de façon globale afin de justifier la conception de la solution préconisée.

L'architecture ainsi proposée prend en compte quatre grands concepts :

- I pour *Interesting*;
- R pour *Reusable*;
- A pour *Atomic*;
- ESB pour Entreprise Service Bus.

V.A.4.i.i. I pour *Interesting*

Les fonctionnalités d'un service devront être intéressantes pour servir la finalité du composant à développer et, également, pour l'ensemble de tous les services du système.

Par exemple, un service qui met à jour un champ dans une table de base de données dès qu'une commande est ajoutée, est totalement inintéressant ! Dans cette situation, il s'agira d'un service qui traite du fonctionnement interne du processus de commande (par exemple, dans la boutique en ligne). En effet, aucun autre service ne sera intéressé par la mise à jour d'un champ obscur dans une table aux tréfonds de la base de données d'un outil quelconque...

Par contre, si un service propose des fonctionnalités de téléchargement de fichier et d'enregistrement de celui-ci dans un emplacement spécifique, alors n'importe quel autre service pourrait être intéressé par de telles fonctionnalités proposées.

V.A.4.i.ii. R pour *Reusable*

Bien que cette fonction ait déjà été abordée au sein des principes énoncés précédemment, ce principe est à considérer de façon concrète si un service doit être réutilisable dans d'autres contextes et scénarios. Ainsi, comme décrit précédemment, le service en question ne doit pas être lié à la logique du processus dans lequel il intervient.

Par exemple, un service renvoyant les 10 derniers produits ajoutés au stock n'est intéressant que dans un cadre très précis, par exemple, pour afficher les nouveautés dans une page de la boutique.

Alors qu'un service qui renvoie n'importe quel nombre de produits demandé est potentiellement intéressant pour beaucoup d'autres services, peu importe le contexte et les situations rencontrées.

V.A.4.i.iii. A pour Atomic

Le mot d'ordre pour cette notion pourrait être : *faites une seule chose et faites-la bien!*

'Atomique' veut dire indivisible et élémentaire : il sera alors nécessaire d'éviter la création de service aux multiples fonctions, tels que le passage des commandes, la gestion des retours, la suppression des produits et le service du café.

Néanmoins, il faudra également ne pas entrer dans les extrêmes et éviter aussi une trop grande atomicité : il ne faudra pas créer un service récupérant le nom d'un produit, un autre sa description, un troisième le nombre de grain de café nécessaire...

L'important dans cette notion est de trouver un juste milieu, équilibré pour que le service développé soit un bloc naturel et élémentaire.

Par exemple, un service ayant une fonction qui renvoie toutes les informations sur tous les grossistes et leurs commandes depuis la création de leurs comptes, n'est clairement pas un service atomique. Ce genre de service sera composé d'une dizaine de composants allant chercher les informations des grossistes, leurs commandes, leurs prix, les informations des produits de chaque commandes, etc.

Un service atomique serait plutôt un service aux fonctions séparées : une pour renvoyer la liste des grossistes, une autre pour la liste des commandes, etc.

La logique étant que si toutes ces informations sont nécessaires, alors il suffira d'appeler chacune de ces fonctions indépendamment, puis d'agréger finalement l'ensemble des résultats renvoyés par chacune d'entre elles.

V.A.4.i.iv. ESB pour Entreprise Service Bus

Un ESB est un composant central qui se positionnera comme un interlocuteur unique pour l'ensemble de tous les composants et services d'un SI.

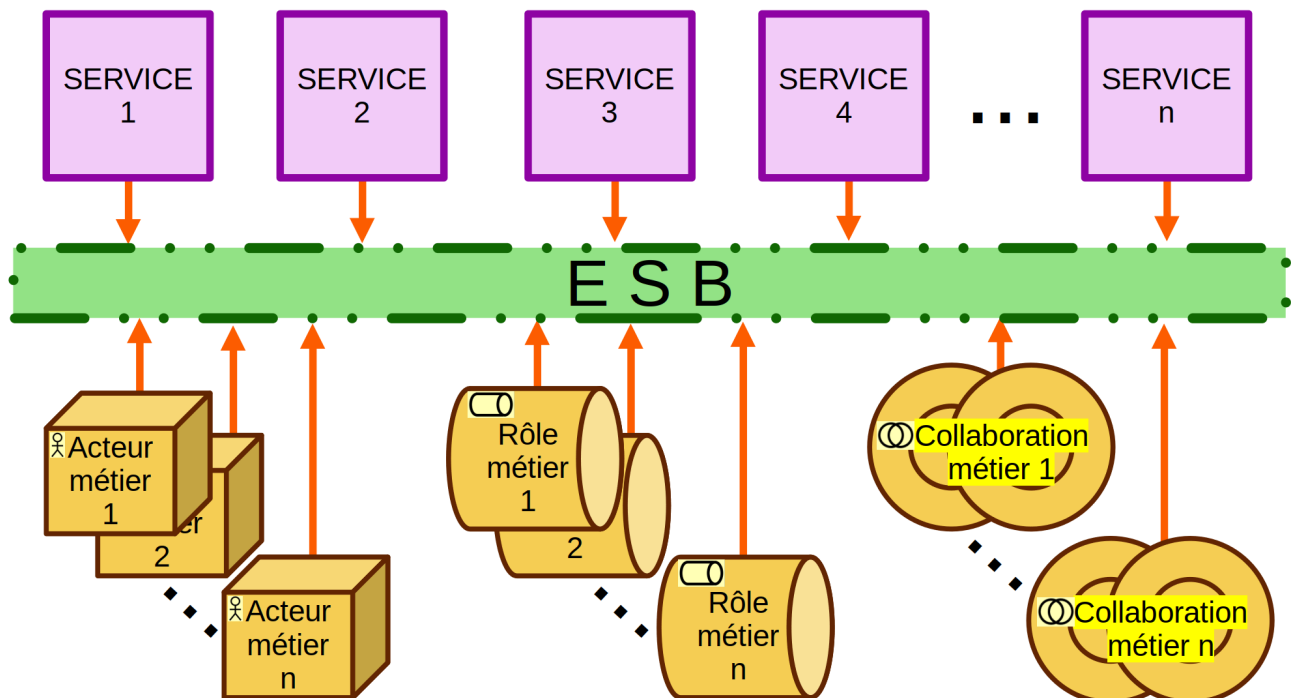
Ainsi, l'appel d'un composant X ne sera plus JAMAIS direct ; il ne sera même pas nécessaire de connaître quel protocole il accepte, ni quelle est son URL.

Il suffira d'envoyer une requête via le protocole préféré (par exemple, SOAP) à l'ESB, et celui-ci s'occupera ensuite de faire le nécessaire pour transformer le message et l'adapter avant de le transférer au composant demandé. Il fera ensuite la même chose pour la réponse avant de la servir, sur un plateau d'argent, au service demandeur ; il ne lui restera plus qu'à déguster...

Pour réaliser toutes ces étapes, il sera nécessaire d'écrire des adaptateurs à fournir à l'ESB afin que celui-ci sache comment traduire les messages en question. Néanmoins, l'énorme avantage ici est la représentation de l'architecture considérée, qui ne sera plus un plat de spaghetti de complexité. En effet, les adaptateurs développés seront tous centralisés au même endroit, et il ne sera donc nécessaire de les dupliquer entre les différents composants.

Par exemple, si dix services ont besoin aussi de communiquer avec un SI, il suffira de fournir à chacun d'entre eux l'adaptateur développé : un pour les diriger tous !

Un ESB pourrait alors être représenté selon le diagramme ci-dessous :



Ainsi, lors du développement d'un adaptateur, il n'en existera qu'une seule copie placée dans l'ESB et il sera alors possible de le modifier et l'améliorer à volonté en toute transparence.

De plus, l'utilisation d'une telle topologie offre plusieurs avantages :

- **Couplage faible** : les composants du SI n'ont plus à se connaître pour communiquer. Chaque service ne communique plus qu'avec l'ESB et il est donc possible de remplacer un service par un autre et gagner en liberté dans le choix des langages et des environnements.
- **Standardisation** : lors de la création d'un nouveau service, son intégration sera très simple et se fait uniquement au sein de l'ESB. Ceci va aider également *Gibberish* à établir des standards dans les développements de services, sachant qu'il n'y a plus de contrainte d'intégration. A terme, tous les services se ressembleront dans leurs structures et leurs modes de fonctionnement, ce qui simplifiera grandement la maintenance, l'interchangeabilité des technologies et, bien sûr, diminuera proportionnellement le coût global.
- **Scalabilité** : avec l'utilisation d'un ESB, il sera beaucoup plus facile d'avoir plusieurs instances de service pour répondre à un trop grand nombre de requêtes. Ainsi, si un service est en surcharge et ne répond plus, l'ESB pourra rediriger les requêtes vers une autre instance de celui-ci permettant ainsi au système de continuer à fonctionner sans ralentissement ou interruption.
- **Routing** : l'ESB permettra de rediriger les requêtes très finement en fonction des paramètres d'entrée, par exemple, lors de la création d'une version améliorée d'un service impliquant des changements dans la façon de l'appeler. Ainsi, grâce à l'ESB, les requêtes entrantes seront redirigées vers la bonne version du service de destination, en fonction de leur compatibilité avec la V1 ou la V2.

V.B.La norme ACID

L'approche ACID (pour **Atomicité**, **Cohérence**, **Isolation** et **Durabilité**) permet d'assurer l'intégrité des données au sein d'une base de données traitant d'important **volumes de données**.

Comme cela a traité précédemment dans cette étude, une BDD est un ensemble de données, structuré ou non, stockées sur un ou des ordinateurs.

Ces architectures sont complexes, et il est nécessaire d'utiliser des SGBDR pour les manipuler. Ces systèmes permettent la gestion du stockage, et la récupération de données au sein des BDD.

Se pose alors la question relative à la méthode utilisée pour gérer les BDD et leur contenu d'une manière optimale. L'une des méthodes répondant à cette question est l'approche ACID.

Les quatre principes énoncés plus haut, **Atomicité**, **Cohérence**, **Isolation** et **Durabilité**, permettent d'assurer que les transactions de bases de données soient traitées de façon fiable.

Pour rappel, le terme de transaction désigne n'importe quelle opération effectuée au sein d'une BDD. Il peut s'agir, par exemple, de la création d'un nouvel enregistrement ou d'une mise à jour des données.

Or, le moindre changement apporté à une BDD doit être effectué avec une extrême rigueur. Dans le cas contraire, les données risquent d'être corrompues.

En appliquant les propriétés ACID, à chaque modification effectuée dans une BDD, il est plus facile de maintenir son exactitude et sa fiabilité.

À présent, cette étude va détailler les quatre composants de cette approche ACID.

V.B.1. Atomicité

L'atomicité d'une transaction de BDD signifie que **tout changement effectué doit être accompli jusqu'au bout**.

En cas d'interruption, par exemple, une perte de connexion au beau milieu de l'opération, le changement doit être annulé et la base de données doit revenir automatiquement à son état antérieur au début de la transaction.

Ce principe permet d'**éviter qu'une transaction soit partiellement terminée**, à cause d'une panne ou d'un plantage. Dans le cas contraire, il est impossible de savoir à quel niveau d'avancée le processus a été interrompu ; d'importantes complications peuvent s'en suivre.

V.B.2. Cohérence

La cohérence, ou *consistency* en anglais, est un principe permettant de garantir qu'une transaction n'enfreigne **les contraintes d'intégrité des données** fixées pour une BDD.

Ainsi, **si la BDD entre dans un état « illégal » en enfreignant ces règles, le processus de transaction sera automatiquement abandonné**. La base de données retournera alors automatiquement à son état antérieur en appliquant le principe d'*Atomicité*.

V.B.3. Isolation

Une transaction isolée est considérée comme « sérialisable ». Cela signifie que les transactions surviennent dans un ordre successif, plutôt que d'être effectuées en une fois.

Toute écriture ou lecture effectuée dans la BDD n'impacte pas l'écriture ou la lecture d'autres transactions survenant sur cette même BDD. Un ordre global est créé, et chaque transaction s'ajoute à une file d'attente. Ce n'est que lorsqu'une transaction est totalement complète que les autres débutent.

Cela ne veut pas dire que deux opérations ne peuvent survenir simultanément. Plusieurs transactions peuvent être effectuées en même temps, à condition qu'elles ne puissent pas s'impacter entre elles.

Bien évidemment, cette méthode peut avoir des conséquences sur la vitesse des transactions. De nombreuses opérations devront attendre avant de pouvoir commencer.

Toutefois, il s'agit d'un sacrifice permettant d'assurer la sécurité et l'intégrité des données.

Pour réaliser cette isolation, il est possible d'opter pour un **schéma de transaction « optimiste »** ou **« pessimiste »**.

Dans le cas d'un **schéma de transaction optimiste**, les autres transactions seront effectuées sans lire ou écrire au même emplacement. Si deux transactions se confrontent, elles seront automatiquement abandonnées et réessayées.

Un **schéma de transaction pessimiste** laisse moins de liberté à la base de données. Les ressources seront limitées, partant du principe que les transactions s'impacteront entre elles. Ceci réduit le nombre d'abandons et d'essais, mais force plus souvent les transactions à patienter dans la file d'attente.

V.B.4. Durabilité

La durabilité est le quatrième principe de l'approche ACID. Il permet d'assurer que **tout changement apporté à la base de données soit permanent**, même en cas de panne du système.

Ceci permet d'éviter que les données soient corrompues ou altérées par une panne de service, un crash ou tout autre problème technique.

Pour permettre cette durabilité, des « **changelogs** » sont utilisés et pris pour références chaque fois que la BDD est redémarrée.

V.C. Le Règlement Général pour la Protection des Données

Comme il a été précisé dans le contexte général relatif à *Gibberish*, cette entreprise capitalise sur l'ensemble des données de ses clients. C'est grâce à ses données que des **HISTOIRE PERSONNALISABLE** spécifiques à chaque client sont réalisées selon les souhaits spécifiques des utilisateurs.

Cependant, il faut considérer que la possession de ces données n'est pas sans conséquence ; c'est à ce titre qu'a été rédigé le RGPD, ou Règlement Général pour la Protection des Données.

Depuis le 25 mai 2018, toutes les entreprises gérant et collectant des données sur les personnes se doivent de respecter chacune des obligations du règlement européen pour la protection des données. Toutes les entreprises sont donc concernées par ce règlement quelque soit leurs modes opératoires.

En ce sens, le RGPD vise à protéger toutes les données à caractère personnel des individus au sein de l'Union Européenne à travers **trois objectifs** précis :

- l'uniformisation européenne de la réglementation sur la protection des données ;
- la responsabilisation des entreprises ;
- le renforcement du droit des personnes.

Le règlement n'a besoin d'aucune transposition légale en fonction du pays de l'entreprise : son application concerne directement tous les pays européens, à partir d'un même texte.

L'intérêt majeur de cette uniformisation à échelle européenne est la simplification des mesures, centralisées vers un interlocuteur unique. Les entreprises doivent s'adresser directement à l'autorité de protection des données pour l'Etat membre dans lequel se situe l'établissement principal.

La simplification des formalités pour les entreprises doit aussi permettre de les responsabiliser dans le traitement et la gestion des données. Les rôles, les responsabilités, les fonctions sont réparties et précisées avec un ensemble de points à suivre : chaque entreprise doit mettre en place une politique de protection des données personnelles, et s'assurer qu'à chaque étape de la gestion des données, le RGPD est respecté.

En outre, de nouveaux droits pour les personnes ont été introduits pour assurer l'accès à leurs informations :

- le droit à la portabilité qui permet aux personnes de récupérer les données fournies, pour un contrôle total de ses propres données ;
- le droit à réparation des dommages matériels et moraux ;
- des droits spécifiques pour les enfants et le traitement de leurs données ;
- des droits aux recours collectifs...

Le RGPD est applicable sur toutes les données à caractère personnel de chacun des citoyens et résidents européens, soit « *toute information se rapportant à une personne physique identifiée ou identifiable* » selon la définition du RGPD. Il permet même de faire valoir ses droits face à une entreprise non européenne. Dès lors qu'une entreprise européenne traite des données personnelles – noms, e-mails, numéro de téléphone... elle est concernée pour la collecte, l'enregistrement, la conservation, le classement, l'utilisation, la diffusion de ces données.

V.D. Évaluation de la réutilisation

Avant de pouvoir évaluer le taux de réutilisation des ressources disponibles au sein de *Gibberish*, il est nécessaire de définir en quoi consiste ce procédé.

V.D.1. Définition de la réutilisation

La réutilisation est une opération qui s’amorce lorsque le détenteur d’une ressource s’en défait sans l’effacer directement. Il peut alors mettre cette ressource à disposition dans une structure spécifique au sein de laquelle la ressource attendra qu’un nouveau besoin soit décelé pour son réemploi.

Cette structure peut être un dépôt, ou conteneur virtuel, au sein duquel la ressource se verra attribuée un nouveau statut : *DISPONIBLE*.

Lorsqu’un nouveau besoin survient et que la ressource a un statut *DISPONIBLE* au sein du dépôt d’attente, elle se voit alors attribuer à ce nouveau besoin ; c’est une sorte de seconde vie...

À noter que le dépôt désigné pour héberger les ressources en attente d’utilisation doit répondre à des règles strictes afin de ne véritablement contenir que des ressources inutilisées. Il s’avérerait dommageable qu’une ressource encore en cours d’utilisation y soit placée en tant que ressource *DISPONIBLE* alors qu’elle est déjà utilisée par un autre processus. Des problématiques liées à l’exclusivité pourraient alors apparaître...

V.D.2. Indicateurs de réutilisation

Avant de pouvoir réutiliser des ressources, il faut les quantifier et la quantification de ces ressources et la définition de leur statut vont permettre d’évaluer leur réutilisabilité.

Par exemple, en prenant en considération la ressource *Utilisateur*, celle-ci est spécifique autant en termes de définition que d’utilisation. Néanmoins, si un utilisateur décide de se désabonner des services proposés par *Gibberish*, cette ressource n’est plus d’actualité et ne doit donc plus avoir accès à ses services.

Cependant, ce n’est pas parce qu’elle n’est plus comptabilisée aujourd’hui, qu’elle ne sera pas utile demain ou l’année prochaine. Il est donc primordiale d’en garder une trace et de la stocker dans l’attente qu’elle soit à nouveau disponible.

Ainsi, au sein de l’exemple précédent, plusieurs éléments peuvent être relevés :

- la ressource,
- son statut (*ACTIVE* ou *DISPONIBLE*),
- sa trace,
- son stockage.

En ce qui concerne les ressources de *Gibberish*, celles-ci seront créées par le processus métier

⇒ **Production de médias.**

En ce qui concerne, leur statut, les différents responsables de processus utilisant ces ressources, devront, pour tous les tuples de chaque ressource, définir si celles-ci sont encore utilisées ou méritent d'être archivées.

En ce qui concerne leur trace d'utilisation, il va falloir mettre en place un système d'historique aidant à assurer le suivi des ressources. Cet historique devra lui-même être caractérisé par un historique d'activités décrivant sommairement les expériences personnalisées et donnant ainsi des suggestions pertinentes. Un des critères pouvant être utilisés pour ce système est la chronologie événementielle.

Ce procédé se base principalement sur des événements survenus à l'entreprise, classés chronologiquement, pour garder des informations relatives à ces événements, tels que les problèmes rencontrés, leur(s) solution(s), les décisions prises... Il définit d'ailleurs parfaitement le processus métier d'⇒ **Index personnel** d'un utilisateur.

Il serait aussi pertinent d'y inclure une gestion de versions qui consiste à gérer l'ensemble des versions d'une ou plusieurs ressources. C'est une activité fastidieuse et complexe qui nécessite l'utilisation d'un logiciel spécialisé. C'est une sorte de journal au sein duquel sont répertoriés tous les événements, tels que ceux stockés au sein de l'évènement métier D **Balances de l'Histoire**, et ayant nécessité des modifications entre deux versions de ressources.

Les ressources ainsi versionnées sont mises à disposition au sein d'un dépôt, c'est à dire d'un ⇒ **espace de stockage** géré par le logiciel de version.

Lorsqu'une modification de ressource est nécessaire, il est alors possible de créer un branche pour constituer un déroulé historique, parallèle à la réalité ; cela évite les divergences et donc les conflits. Chaque branche peut alors être étiquetée, ou marquée, afin de disposer d'un repère pour pouvoir y accéder ultérieurement, grâce, par exemple à la ⇒ **Navigation dans l'Histoire.**

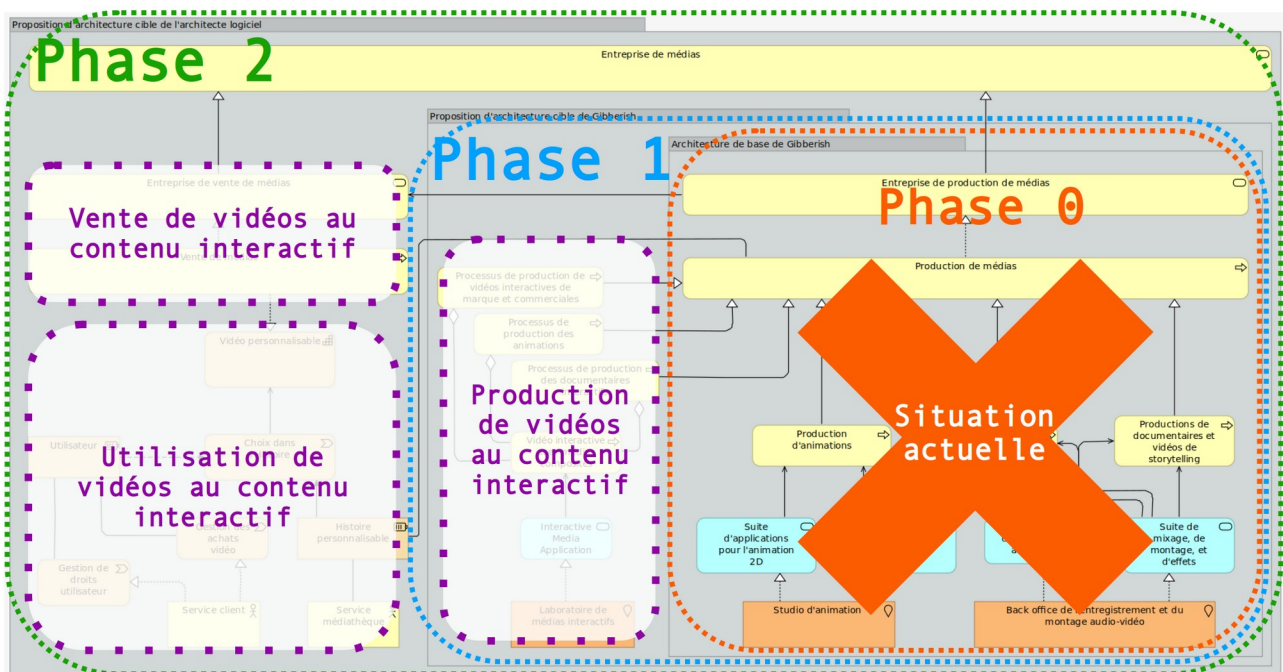


VI. Analyse des écarts

L'analyse des écarts présentée dans ce document se basera sur trois phases successives à atteindre séquentiellement :

- **Phase 0** : cette phase correspond à la situation organisationnelle et structurelle de *Gibberish*. En tant qu'architecture de base, cette phase sera le point de départ et la référence de base de l'évolution vers l'architecture cible.
- **Phase 1** : cette phase correspond à l'architecture proposée par *Gibberish* et qui sera désignée comme *Intermédiaire* au sein de cette étude.
- **Phase 2** : cette phase sera considérée comme l'architecture cible et, donc, comme phase finale à atteindre pour effectuer l'évolution du SI de *Gibberish*.

L'écart estimé ici, sera donc un différentiel entre la phase 2 représentée par l'architecture cible et la phase 0 correspondant à l'architecture de base, tel que représenté ci-dessous :



Ainsi, les fonctionnalités résultantes de cette opération différentielle se résume à trois domaines principaux :



- le domaine de production de vidéo interactive ;
- le domaine d'utilisation des possibilités offertes par les vidéos interactives à l'utilisateur pour créer sa propre Histoire ;
- le domaine de vente de vidéos source au contenu interactif aux différentes clients de *Gibberish*.



Ces trois domaines rejoignant bien les objectifs décrits dans le paragraphe §Contexte, objectifs et contraintes, il est désormais possible d'assumer le fait que cette étude ait bien répondu aux attentes initiales.



















VII. Évaluation de l'impact

En considérant que la **Phase 0** est l'état actuel de la société *Gibberish*, l'évaluation de l'impact considérera les **Phase 1** et **Phase 2** comme objectifs à atteindre.

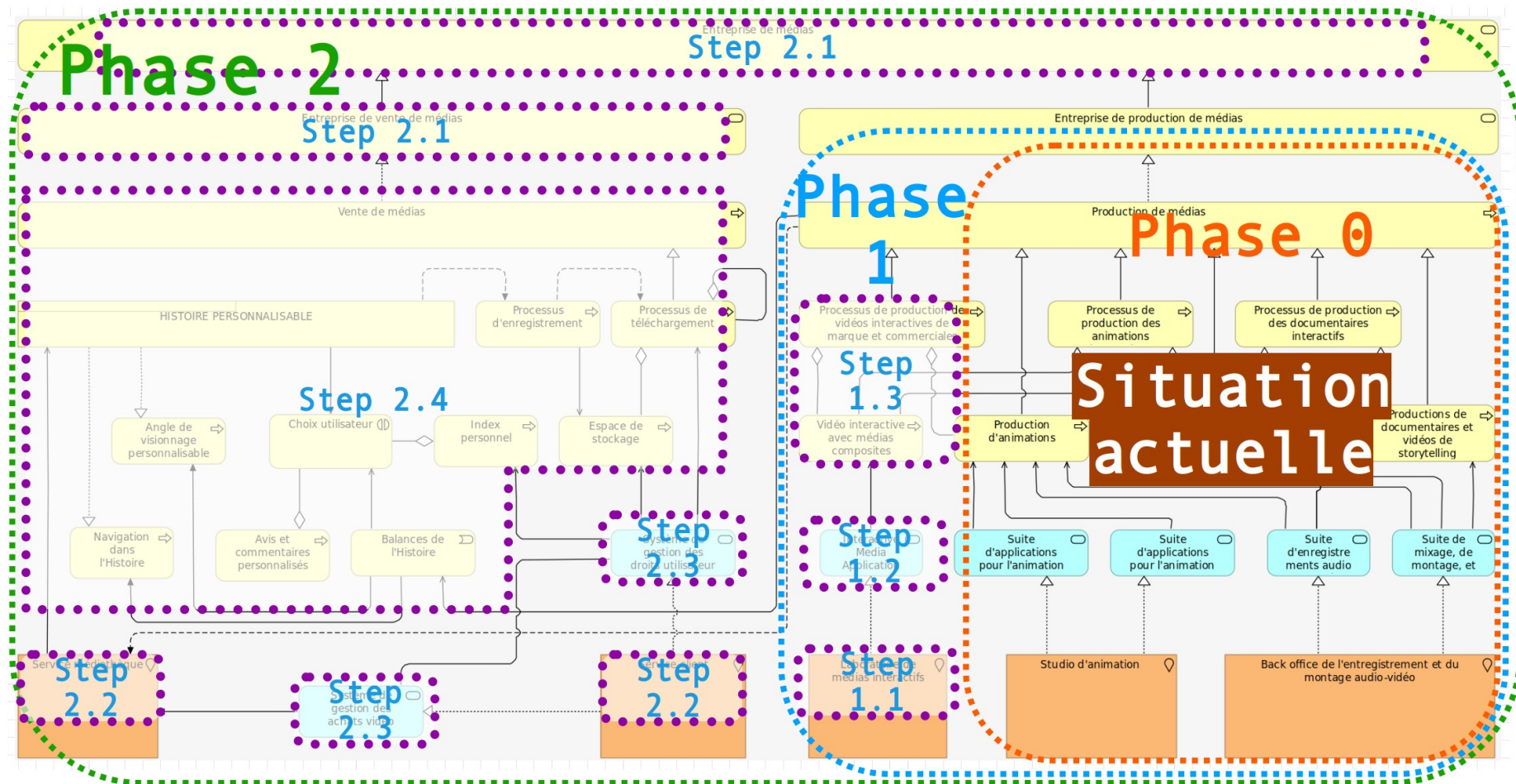
En ce qui concerne spécifiquement la **Phase 1**, cette architecture ayant été proposée par *Gibberish* même, il sera nécessaire de vérifier si l'acquisition du  *Laboratoire de médias interactifs* et de l' *application de Media interactif* déjà était prévue par les parties prenantes ayant conçues cette architecture.

En ce qui concerne spécifiquement la **Phase 2**, il sera nécessaire de considérer deux  *localisations* et deux  *services métier* supplémentaires :

















-  **Service client** : ce service aura la charge de deux services applicatifs :
 - le  **Système de gestion des droits utilisateurs** associés aux différents clients de *Gibberish* aura la charge de sécuriser les processus métier d' **index personnel**, d' **Espace de stockage** et de  **téléchargement** des vidéos ;
 - le  **Système de gestion des achats de vidéo** sera un service applicatif qui fournira les accès au  **Service médiathèque** pour distribuer les vidéos aux clients.
-  **Service médiathèque** : ce service aura la responsabilité de centraliser toutes les vidéos interactives (**HISTOIRE PERSONNALISABLE**) produites par *Gibberish* et de les mettre à disposition des utilisateurs en fonction de leurs droits gérés par le  **Système de gestion des droits utilisateur**.
-  **Entreprise de vente de médias** : à l'instar du service métier  **Entreprise de production de médias** prenant à charge la  **Production de médias**, ce nouveau service métier prendra à sa charge la  **vente de médias** à contenu interactif produit par *Gibberish*.
-  **Entreprise de médias** : ce service métier aura la responsabilité de gérer les services métier  **Entreprise de vente de médias** et  **Entreprise de production de médias** en ce qui concerne les domaines commercial, financier, juridique, socio-environnemental...

VIII. Architecture de transition

Comme il a présenté précédemment dans ce document, la transition architecturale, d'une architecture de base à une architecture cible, se déroulera en trois phases consécutives, telle représentées dans le diagramme suivant :



Ainsi, les trois phases en question vont de se dérouler selon la décomposition ci-dessous :

- **Phase 0** : cette phase est la situation organisationnelle et structurelle actuelle de *Gibberish* (*état initial*).
- **Phase 1** : cette phase ayant été conçue par *Gibberish* même (*état intermédiaire*), il sera nécessaire de vérifier, avec les parties prenantes l'ayant définie, que :
 - **Step 1.1** : la localisation du  **Laboratoire de médias interactifs** a bien été prévue ;
 - **Step 1.2** : la suite logicielle  **Interactive Media Application** a bien été commandée et financée ;
 - **Step 1.3** : les processus métiers associés à la production de médias interactif aient été conceptualisés et concrétisés.
- **Phase 2** : cette phase sera considérée comme l'*état final* à atteindre suite à cette migration et sera composée de quatre différentes étapes :
 - **Step 2.1** : dans un premier temps, il sera nécessaire de s'occuper de la création de deux entreprises :
 -  **Entreprise de médias** : responsable des  **Entreprise de vente de médias** et  **Entreprise de production de médias**.
 -  **Entreprise de vente de médias** : chargée de tous les processus et entités responsable de vendre et, ainsi, mettre à disposition des clients tous les produits et services liés au streaming de vidéos au contenu interactif.
 - **Step 2.2** : il sera nécessaire ici d'acquérir des locaux propices aux  **Service médiathèque** et  **Service client**.
 - **Step 2.3** : cette étape sera celle de définition des  **Système de gestion des droits utilisateur** et  **Système de gestion vidéos** au contenu interactif de tous les produits de *Gibberish*.
 - **Step 2.4** : enfin, cette étape définira :
 - la  **Navigation dans l'Histoire** interactive ;
 - la modification d'  **Angle de visionnage personnalisable** ;
 - les  **Balances de l'Histoire** possibles et les  **Choix utilisateur** associés ;
 - les  **Processus d'enregistrement** et  **Processus de téléchargement** nécessaires à la mise à disposition des différentes vidéos au contenu interactif.

GIBBERISH