Bloc de construction de la solution GED

Projet SCS Magazine GED

Auteur(s) et contributeur(s)

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

Historique des modifications et des révisions

N° version	Date	Description et circonstance de la modification	Auteur
1.0	15/11/2022	Création du document	Gérald ATTARD

Validation

N° version	Nom & Qualité	Date & Signature	Commentaires & Réserves
1.0	NICOLAS Software Architect Team Leader		

Tableau des abréviations

Abr.	Sémantique
ABB	Architecture Buiding Block (trad. *bloc de construction d'architecture*)
ASP	Active Server Page (trad. *page du serveur actif*) / Application Service Provider (trad. *fournisseur de service applicatif*)
BDD	Base De Données
BLOB	Binary Large Object (trad. *gros objet de données binaire*)
CEI	Commission électrotechnique internationale
CGI	Common Gateway Interface (trad. *interface de passerelle commune*)
CMS	Content Management System (trad. *système de gestion de contenu*)
COTS	Commercial Off The Shelf (trad. *produit commercial sur étagère*)
DAAS	Desktop As A Service (trad. *bureau en tant que service*)
DDA	Document de Définition d'Architecture
EJB	Enterprise JavaBeans
ERP	Entreprise Resource Planning (trad. *progiciels de gestion intégrés*)
GED	Gestion Electronique de Documents
GOTS	Gouvernemental Off The Shelf (trad. *produit gouvernemental sur étagère*)
HTML	HyperText Markup Language (trad. *langage signalétique hypertexte*)
HTTP	HyperText Transfer Protocol (trad. *protocole de transfert hypertexte*)
IaaS	Infrastructure As A Service (trad. *infrastructure en tant que service*)
ICR	Intelligent Character Recognition (trad. *reconnaissance intelligente de caractères*)
IoT	Internet Of Object (trad. *Internet des objets*)
ISO	International Standardization Organization (trad. *organisation internationale de normalisation*)
J2EE	Java 2 platform, Entreprise Edition (trad. *java version 2 plateforme, édition entreprise*)
JSP	Java Server Page (trad. *page du serveur*)
MVC	Modèle – Vue - Contrôleur
MVP	Modèle – Vue - Présentation
PaaS	Platform As A Service (trad. *plateforme en tant que service*)
OCR	Optical Character Recognition (trad. *reconnaissance optique de caractères*)
RAD	Reconnaissance Automatique de Documents

Abr.	Sémantique
SaaS	Software As A Service (trad. *logiciel en tant que service*)
SAE	Système d'Archivage Électronique
SBB	Solution building Block (trad. *bloc de construction de la solution*)
SGBDR	Système de Gestion de Base de Données Relationnnel
SQL	Structured Query Language (trad. *langage de requêtage structuré*)
TIC	Technologie de l'Information et de la Communication
WfMC	Workflow management coalition (trad. *coalition de gestion des flux de travail*)
XML	eXtensbile Markup Language (trad. *langage de balisage extensible*)
XPDL	XML Process Definition Language (trad. *langage de définition de processus XML*)

Table des matières

I. Fonctionnalités et attributs

Lecteur cible de ce paragraphe : toute partie prenante

Relativement au contexte de ce projet décrit dans le document de définition de l'architecture, la solution de GED collaboratives décrite ici, sera orientée pour être hébergée sur un cloud. Ce choix architectural se justifie afin de permettre un travail efficace entre les collaborateurs à distance, notamment via le système des *workflows*.

Bien loin de se cantonner uniquement à l'archivage et à la sécurisation des documents, la GED de ce projet constitue une alternative efficace pour partager les documents, les annoter, gérer leur *versionning* et les valider efficacement à distance en mode collaboratif.

En effet, bien que le contexte d'utilisation actuel se focalise sur la production éditoriale d'article documentaire, une GED collaborative doit permettre à tout un chacun de partager des documents de toute nature (factures, courriers, commandes, procédures, contrats...) et de toutes origines (email, papier, PDF, ERP...), sans effort et en toute sécurité, de les distribuer, les annoter, les versionner, et d'alerter sur leur disponibilité.

Aussi, ce document proposera ci-dessous les principales fonctionnalités à retrouver au sein de la future, ainsi que les attributs associées à celles-ci.

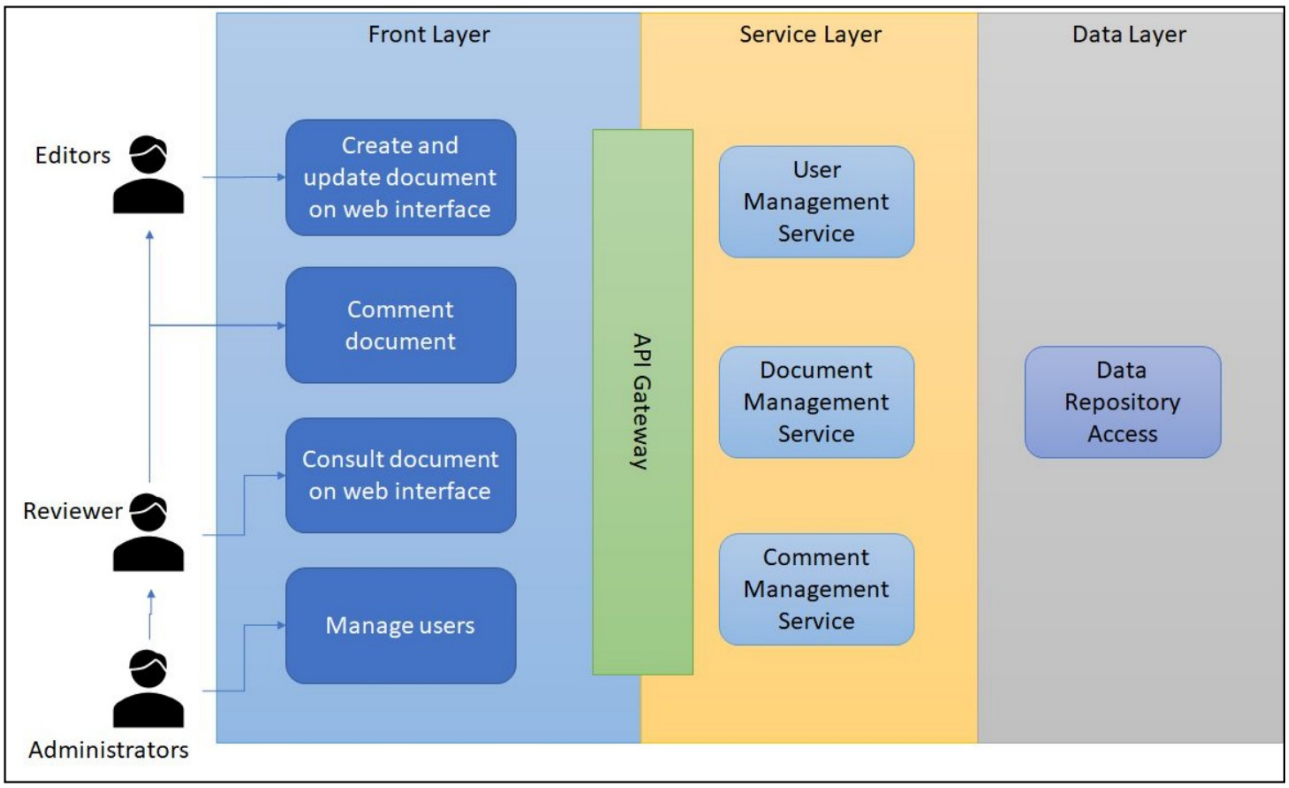
Fonctionnalité	Attribut	Description
Numériser ou Intégrer	**Type du document source**	Identifier la nature du document d'origine : numérique ou physique.
Indexer, classifier et organiser	**Identifiant unique**	Ranger le document de façon cohérente et pertinente.
Rechercher et trouver des documents	**Type de recherche**	Désigne la méthode de recherche d'un document : thématique, par mot clé...
Créer des documents	**Nom du document**	Nommer un document créé de façon pertinente.
Annoter des documents	**Annotations du document**	Conserver les annotations d'un document en tant que métadonnées.
Modifier des documents	*N/A*	Sauvegarder un document modifié.
Consulter des documents	*N/A*	Afficher un document.
Echanger des documents	**Droits du collaborateur** **Droits appliqués au document**	Partager des documents entre collaborateurs.
Gérer la sécurité	**Droits du collaborateur** **Droits appliqués au document**	Permettre de gérer la sécurité en paramètrant les différents droits d'accès.

Fonctionnalité	Attribut	Description
Archiver	**Nom de l'archive**	Conserver le document durant un délai prédéfini.
Automatisation des processus métier	**Workflow**	Identifier le nom du workflow au sein duquel le document est géré.
Accesibilité	**Emplacement**	Permettre l'accès à un document peu importe l'emplacement de l'utilisateur et le type de périphérique utilisé pour la consultation.
Intégration des autres outils du SI	*N/A*	Connecter la GED aux autres outils de la société : CRM, ERP, etc.

II. Relations entre ABB et SBB

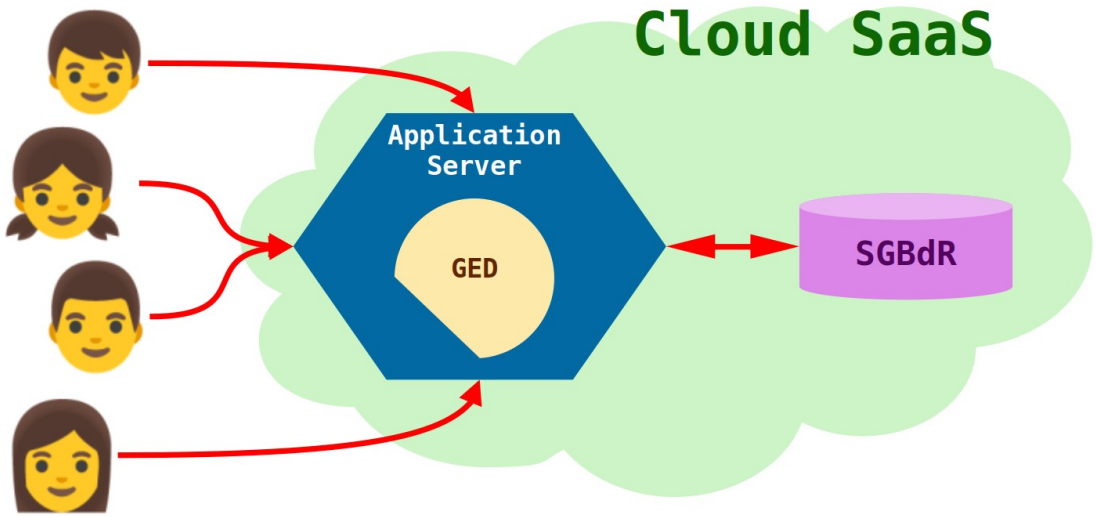
Lecteur cible de ce paragraphe : toute partie prenante

Tel qu'il a été décrit dans le DDA, l'ABB de ce projet est représenté par le diagramme d'architecture cible ci-dessous :



Il est alors possible d'y distinguer des acteurs (Editors, Reviewers et Administrators) et également trois couches successives permettant l'interfaçage entre le système et les acteurs (Front Layer), le traitement des données impliquées (Service Layer) et leur stockage (Data Layer).

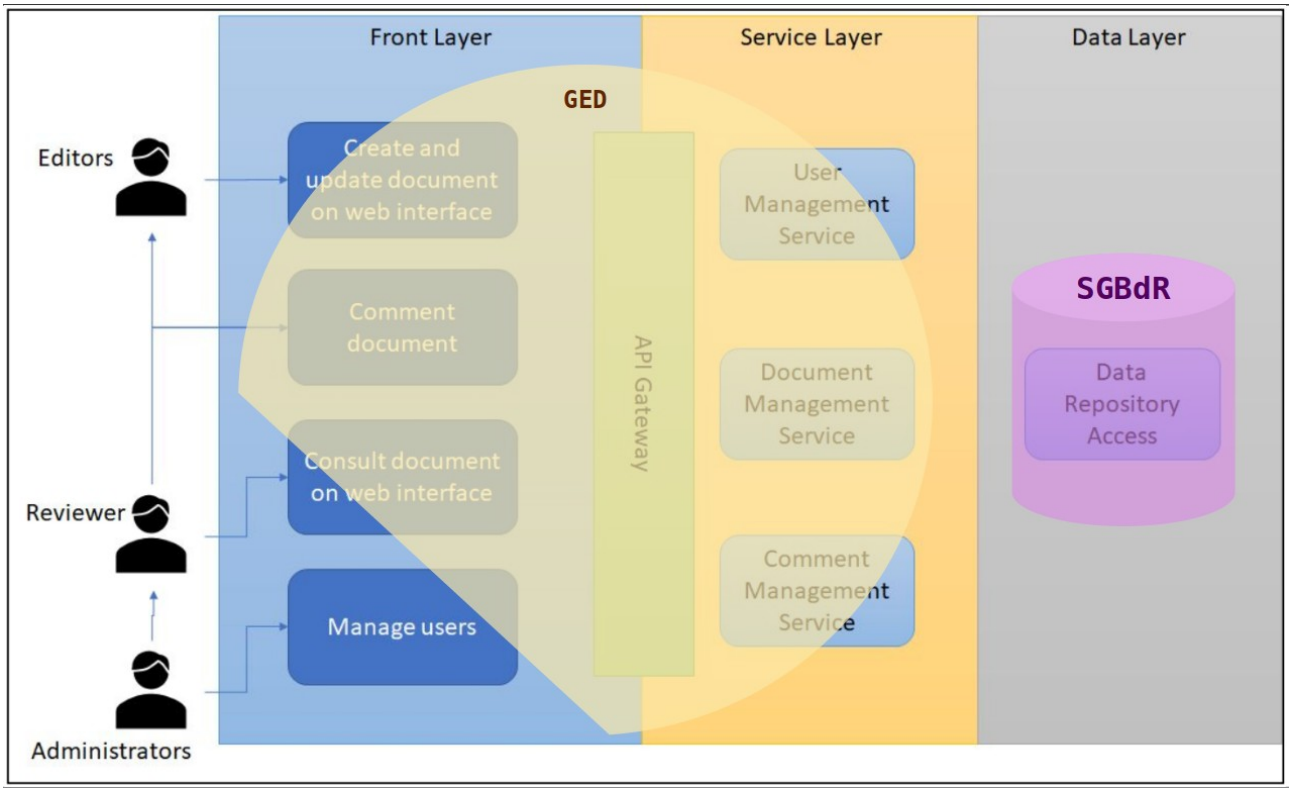
En outre, tel qu'il a été décrit dans le document d'évaluation de la conformité, les ressources nécessaires à l'implémentation d'une solution répondant au diagramme d'architecture cible précédent, sont décrites au sein du schéma suivant :



Il est alors possible de dénombrer deux ressources pricipales en tant que SBB :

- une **GED** ;
- un **module de stockage** (SGBDR) spécifiquement choisi pour gérer des entités de type BLOB.

Ainsi, il est possible de fusionner les diagrammes pour mettre en évidence les relations entre ABB et SBB présentés par la solution, selon le diagramme suivant :



Il est possible de constater dans cette vue que toutes les fonctionnalités assurées par les couches *Front Layer* et *Service Layer* sont prises en charge par le module de GED, et que la fonctionnalité de stockage des données de la couche *Data Layer* est assurée par le SGBdR.

III. Interfaces

Lecteur cible de ce paragraphe : toute partie prenante

Afin de répondre aux fonctionnalités détaillées dans le paragraphe précédent, il sera nécessaire d'utiliser quatre types d'interfaces répondant aux quatre notions présentées dans le document d'évaluation de la conformité, à savoir :

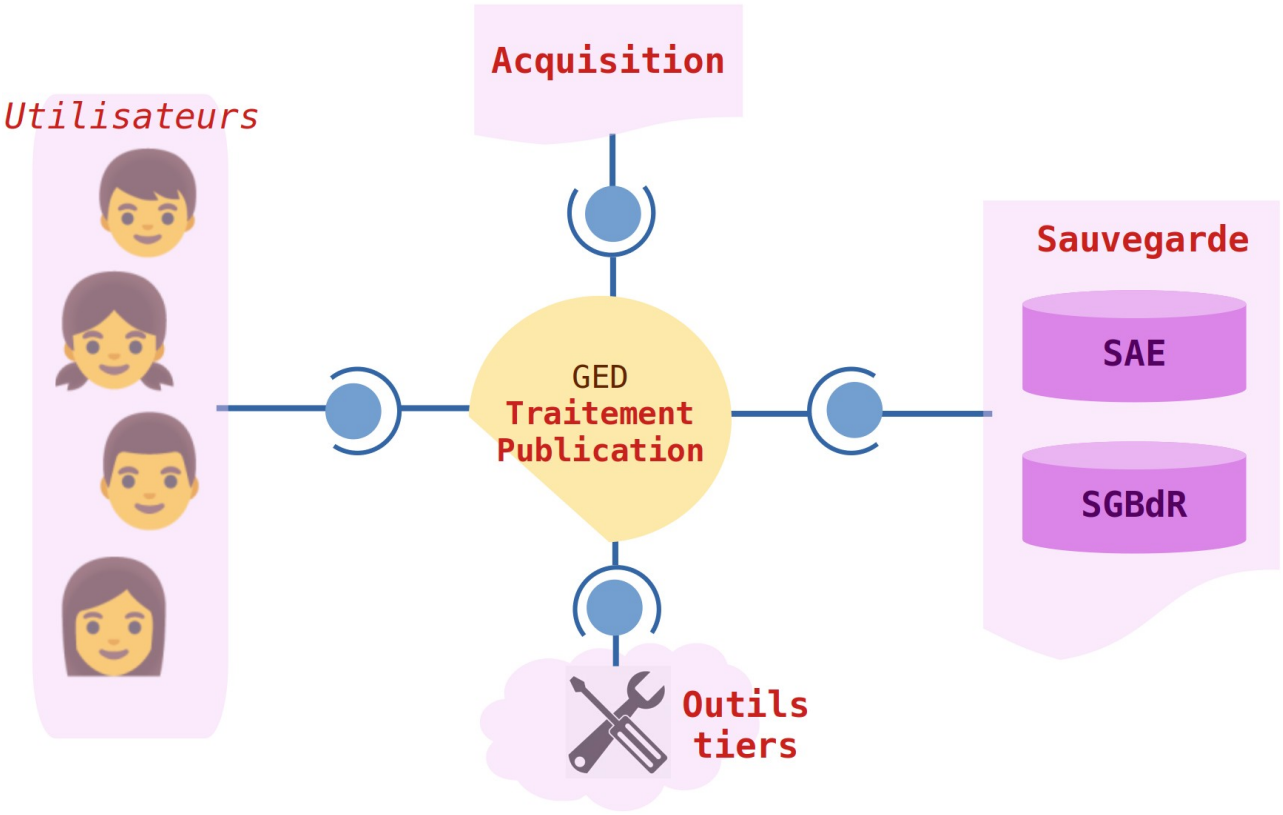
- L'**acquisition des documents** : dans un premier temps, les documents existants doivent être intégrés à la solution. Pour cela, il est possible de numériser les documents papier ou d'intégrer des documents électroniques déjà existants. Ces derniers peuvent également être produits à partir de la solution ou provenir d'autres sources, par exemple dans le cadre de l'échange des données informatiques (EDI).

- Le **traitement des documents** : une fois intégrés, les documents doivront être indexés. L'indexation documentaire permet de rationaliser l'organisation et de faciliter la recherche. Concrètement, cela consiste à décrire la forme et le contenu du document : type (PDF, mail, image, vidéo, photo…), auteur, date, sujet, source, mots clés, concept… Grâce aux technologies de reconnaissance des caractères (OCR), l'indexation peut être automatisée, tel qu'il est décrit dans le paragraphe *§II.E.Applications utilisatrices de la GED* du document d'évaluation de la conformité.

- Le **stockage des documents électroniques** : le support de stockage doit être dimensionné pour répondre aux exigences de volume documentaire de votre entreprise, mais aussi à la fréquence d'utilisation du système et au degré de confidentialité des données. Le stockage peut également prendre en compte des durées de conservation propres à chaque type de documents et des périodes d'épuration destinées à alléger la structure et faciliter son exploitation. L'archivage est généralement assumé par deux autres outils : la BDDs et le SAE.

- La **diffusion des documents** : les documents sont accessibles depuis une plateforme munie d'une interface ergonomique. Ils répondent à des règles et des droits d'accès : lecture seule, modification, en écriture…

Ainsi, il sera alors nécessaire de proposer des interfaces répondant aux quatre critères ci-dessus, sans omettre celles associées aux interactions avec les outils tiers, telles que décrit dans le diagramme ci-dessous :

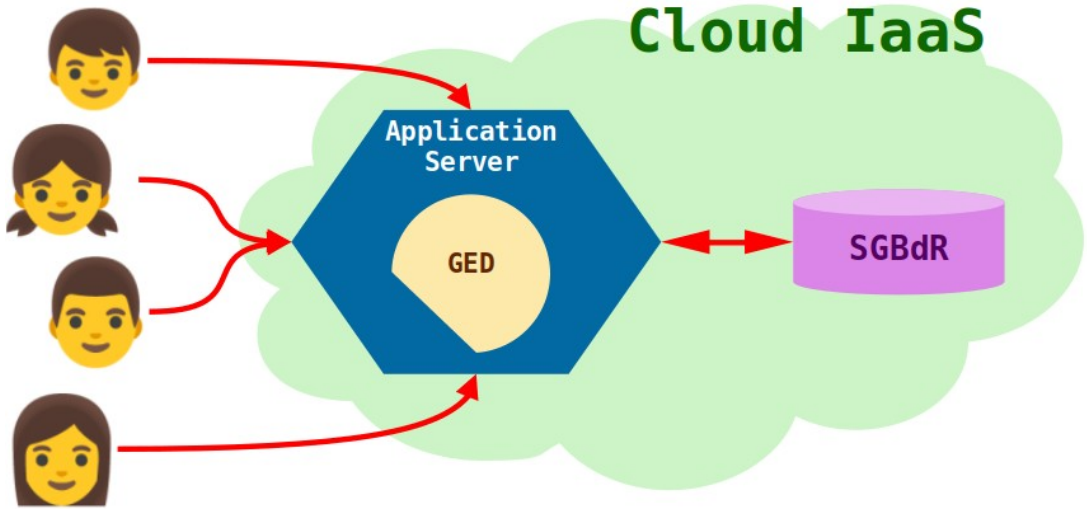


IV. SBB requis

Lecteur cible de ce paragraphe : partie prenante technique
Associé au SBB spécifique à une GED, cette dernière devra également collaborer avec deux solutions pour pouvoir être pleinement opérationnelle ; ces deux autres solutions sont :

- un serveur d'application
- une base de données prenant en compte la gestion des BLOB.

En effet, tel qu'il a été décrit le document d'évaluation de la conformité, la GED sera elle-même hébergée par un serveur d'applications et devra collaborer étroitement avec un SGBDR gérant les BLOB, comme l'expose dans le diagramme ci-dessous :



A noter, qu'en fonction du choix de la GED en tant que COTS, cette solution peut très bien incorporer nativement une BDD prenant en compte les BLOB.

Ainsi, dans un soucis d'exhaustivité, cette étude proposera la description d'un serveur d'applications et présentera dans ce document une BDD prenant en charge les BLOB.

IV.A. Serveur d'application

IV.A.1. Définition

Un serveur d'applications est une solution serveur placée sur un réseau distribué qui orchestre la logique métier d'une application. Le serveur d'application est fréquemment considéré comme une application trois-tiers, voire n-tiers, comportant :

- un serveur d'interface utilisateur graphique,
- un application serveur,
- une base de données,
- un serveur transactionnel.

D’une façon plus descriptive et sommaire, il peut être segmenté en trois parties :

1. un premier tiers, le **front-end**, qui est une interface utilisateur graphique s’exécutant dans un navigateur Web ;
2. une application, ou un groupe d’applications, dédié à la **logique métier** ;
3. un **back-end**, comme une base de données et un serveur transactionnel.

Les bases de données patrimoniales ainsi que les applications de gestion de transactions plus anciennes font partie du back-end. Le serveur d’applications se place au centre, entre les front-ends reposant sur le navigateur et les back-ends (bases de données et systèmes patrimoniaux).

Dans de nombreux cas d’usage, le serveur d’applications est associé ou fonctionne de pair avec un serveur Web (bassé sur de l’HTTP) et est appelé un serveur d’applications Web. Le navigateur Web supporte alors un front-end HTML, côté utilisateur. Le serveur Web fournit plusieurs moyens pour pousser une requête vers un serveur d’applications et pour remonter une nouvelle page, ou une modifiée, à l’utilisateur.

IV.A.2. Rôles

Utilisés initialement pour concevoir des pages web dynamiques, les serveurs d’application sont devenus les infrastructures de nouvelles applications bâties à l’aide de composants métier et intégrées aux systèmes d’information d’une entreprise.

Inadaptée pour répondre à ce nouveau besoin, l’infrastructure des sites statiques historiques a évolué afin, notamment, d’introduire des données enregistrées dans un SGBD au sein des pages statiques, et ainsi poser les prémices des pages web dynamiques.

IV.A.2.a. scripts CGI

Dans un premier temps, les éditeurs de serveurs web ont développé plusieurs types d’interfaces afin d’insérer des données dynamiques dans les pages et la CGI demeure l’une des plus populaires.

En effet, lors de la réception d’une requête HTTP, le serveur web lance l’exécution d’un programme accédant au SGBD et fournit, en retour, les pages contenant les données lues et prêtes pour l’affichage.

Néanmoins, malgré ses avantages, la CGI présente quelques défauts par de piètres performances et un fonctionnement assez complexe. Ainsi, à chaque requête HTTP, le serveur web lance un programme qui ouvre la base de données, lit les informations, puis ferme le SGBD ; les accès au SGBD étant très long, la CGI n’assure pas le maintien de l’état des sessions utilisateurs entre les requêtes HTTP.

IV.A.2.b. un web plus interactif

Pour pallier aux problèmes liés à l'inertie des pages web statiques, une nouvelle classe d'infrastructures logicielles, appelées serveurs d'application, a été élaborée.

Le principal rôle de cette première génération de plates-formes consistait à relier les serveurs web aux SGBD, sans ployer sous la montée en charge.

Les technologies employées sont les servlets et les JSP du monde J2EE, les ASP de Microsoft ou encore les scripts PHP du monde open source. Toutes ce stechnologies fournissent des moyens simples pour gérer des interactions dynamiques avec le web, au travers de sessions dont le contexte est maintenu entre les requêtes HTTP.

IV.A.2.c. Accroître les performances

Les serveurs d'application prennent également en charge le parallélisme du traitement des requêtes en provenance des internautes.

Ces serveurs facilitent la bonne tenue de l'infrastructure face à la montée en charge, en assurant un partage des ressources des serveurs entre les différents internautes. Ainsi, la majorité des technologies de serveurs d'application gèrent un pool de connexions au SGBD. Ces dernières sont affectées successivement aux différentes requêtes des internautes et évitent de rouvrir la base de données à chaque fois.

Les performances et l'aptitude à monter en charge sont en conséquence grandement améliorées vis-à-vis des anciens scripts CGI.

IV.A.2.d. Une programmation proche du client-serveur

Bien que les technologies d'affichage aient évolué, grâce au remplacement de l'interface locale Windows par une interface web standardisée et déportée sur les postes clients des internautes, le modèle de programmation des applications n'a que peu changé.

En effet, la logique métier et celle de présentation sont toujours mélangées dans le code source de la plupart des applications web, ce qui rend impossible toute réutilisation de code source.

Face à ce constat, les éditeurs de serveurs d'application proposent des modèles de composants métier (Com+ pour la plate-forme Microsoft, EJB pour la plate-forme J2EE) en complément des pages ASP/JSP, de façon à mieux structurer en couches la logique applicative.

Le résultat obtenu est alors une découpe en 4 couches :

- client web,
- logique de présentation web,
- logique métier
- accès aux données.

Cette segmentation facilite grandement la réutilisation de la logique métier. Toutefois, les composants métier restent encore peu utilisés, car ils impliquent de nouvelles méthodes de travail et une réflexion architecturale hors de portée sans un chagement drastique de la façon de penser les nouvelles architectures de systèmes d'informations de bon nombre d'entreprises.

IV.A.2.e. Une architecture de composants

Le couplage croissant des applications web novatrices avec les systèmes d'informations des entreprises implique des services *middlewares* aux complexités bien plus élevées.

Naturellement, la mise au point de ces services se révèle assez longue, étant donnée la réalisation de composants facilitant leur utilisation. Ainsi, à partir des version 2 ou 3, tant du côté Com+ que du côté EJB, les architectures de composants modulaires matures sont devenues un référence pour l'élaboration de nouvelles solutions modulaires et robustes aux pannes.

IV.A.2.f. Les services offerts aux développeurs

Les modèles de composants Com+/EJB fournis avec les serveurs d'application simplifient grandement l'usage des services *middlewares* de ces derniers, tels que la gestion des transactions et de la sécurité, l'activation et la désactivation des composants.

Ces invocations de services *middlewares* sont gérées par la structure d'accueil des composants, dénommée "*conteneur de composant*" intégré naitvement au serveur d'application.

Le comportement de chaque composant vis-à-vis de ces aspects est déclaré par un descripteur de déploiement, fixé le plus souvent lors de l'intégration de l'application.

Les développeurs n'ont donc plus à s'occuper de ces aspects, à la différence, par exemple, de CORBA.

IV.A.2.g. Une interopérabilité avec de nombreux systèmes

Les serveurs d'application proposent de nombreuses interfaces d'accès, chargées d'assurer l'interopérabilité des composants métier avec le monde extérieur :

- communication synchrone prise en charge par un courtier d'objet (DCom, Corba…),
- communication asynchrone via un *middleware* orienté message (MQ Series),
- accès aux bases de données et aux applications d'entreprise via des connecteurs adéquats,
- etc.

Cette interopérabilité place de fait les serveurs d'application au centre des nouvelles architectures informatiques.

IV.A.2.h. Des composants disponibles

Les plates-formes de serveur d'applications assurent également la disponibilité et la montée en charge des composants, au travers de services d'équilibrage de charge et de basculement sur panne à l'intérieur de serveurs multiprocesseurs ou de fermes de serveurs.

L'ensemble de ces services est géré grâce aux consoles d'administration, fournies nativement par la plupart des serveurs d'application.

IV.A.2.i. L’engouement pour les composants

Enfin atteinte, la maturité rend ces modèles de composants réellement exploitables et autorise une certaine réutilisabilité.

Ces composants sont d’ailleurs de plus en plus employés dans les applications stratégiques des entreprises, telles que les CMS ou les ERP.

IV.B. BDD gérant les BLOB

IV.B.1. Définition

Une base de données permet de stocker et de retrouver des données structurées, semi-structurées ou des données brutes ou de l'information, souvent en rapport avec un thème ou une activité ; celles-ci peuvent être de natures différentes et plus ou moins reliées entre elles.

Leurs données peuvent être stockées sous une forme très structurée (base de données relationnelles par exemple), ou bien sous la forme de données brutes peu structurées (avec les bases de données NoSQL par exemple). Une base de données peut être localisée dans un même lieu et sur un même support informatisé, ou répartie sur plusieurs machines à plusieurs endroits.

La base de données est au centre des dispositifs informatiques de collecte, mise en forme, stockage et utilisation d'informations. Le dispositif comporte un SGBD, un logiciel moteur qui manipule la base de données et dirige l'accès à son contenu. De tels dispositifs comportent également des logiciels applicatifs, et un ensemble de règles relatives à l'accès et l'utilisation des informations.

Toutes les BDD seront interrogeables à partir du langage SQL décrit dans ce document et devront être capable de contenir des données de type BLOB.

IV.B.2. Gestion

IV.B.2.a. Définition d’un BLOB

Un BLOB est un type intégré qui stocke un grand objet binaire en tant que valeur de colonne dans une ligne d'une table de base de données.

Par défaut, les pilotes de base de données implémentent les BLOB à l'aide d'un localisateur SQL , de type BLOB, ce qui signifie qu'un objet BLOB contient un pointeur logique vers les données SQL BLOB plutôt que les données elles-mêmes.

Un objet BLOB est valide pour la durée de la transaction dans laquelle il a été créé.

Les interface BLOB d’un pilote de base de données fournissent ainsi des méthodes pour obtenir la longueur d'une valeur SQL BLOB (Binary Large Object), pour matérialiser une valeur BLOB sur le client et pour déterminer la position d'un modèle d'octets dans une valeur BLOB.

De plus, cette interface dispose de méthodes pour mettre à jour une valeur BLOB. Toutes les méthodes de l'interface Blob doivent être entièrement implémentées si le pilote de la base de données prend en charge ce type de données.

IV.B.2.b. Options de stockage des objets BLOB

Flux de fichiers

Les flux de fichiers, ou *filestream*, permettent aux applications de stocker des données non structurées, telles que des documents et des images, sur le système de fichiers.

Les applications peuvent tirer parti des riches API de streaming et des performances du système de fichiers tout en maintenant la cohérence transactionnelle entre les données non structurées et les données structurées correspondantes.

Tables de fichiers

La fonctionnalité des tables de fichiers apporte la prise en charge de l'espace de noms de fichiers et la compatibilité avec les applications pour stoquer les données d’un ou plusieurs fichiers.

Les tables de fichiers permettent ainsi à une application d'intégrer ses composants de stockage et de gestion des données, et fournit des services SQL intégrés, y compris la recherche en texte intégral et la recherche sémantique, sur des données et des métadonnées non structurées.

En outre, il est possible de stocker des fichiers et des documents dans des tables spéciales de SQL appelées *Tables de fichiers* et permettre au système d’exploitation d’y accéder indirectement à partir d'applications installées. Ces dernières, avec qui l’OS communique, mettent alors les données recueillies à disposition de l’OS, comme si elles étaient stockés au sein de son propre système de fichiers et sans apporter de modifications aux applications clientes.

Magasin d'objets BLOB

Le magasin de BLOB permet aux administrateurs de bases de données de stocker des objets binaires volumineux (BLOB) dans des solutions de stockage standard plutôt que directement sur le serveur lui-même.

Cette fonctionnalité permet d'économiser une quantité importante d'espace et évite de gaspiller des ressources matérielles de serveur coûteuses.

En outre, le magasin de BLOB fournit un ensemble de bibliothèques d'API qui définissent un modèle standardisé pour que les applications accèdent aux données du BLOB.

En complément, ces API comprennent également des outils de maintenance, tels que la récupération de place, pour aider à gérer les données BLOB distantes, un peu comme le *garbage collector* en JAVA.

V. Mappage

V.A. Mappage aux normes

Lecteur cible de ce paragraphe : partie prenante technique

V.A.1. Standard d'architecture Client-Serveur

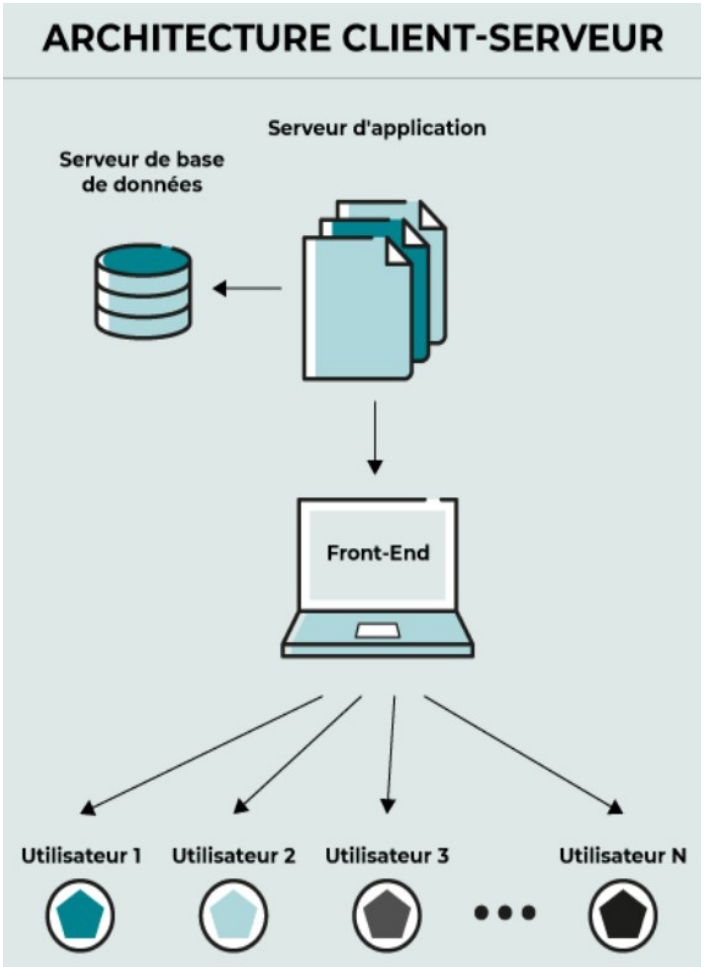
Notion à appliquer au niveau APPLICATIF

V.A.1.a. Description

L'architecture Client-Serveur fonctionne selon un principe de répartition des tâches entre les fournisseurs d'un service, appelés **Serveurs**, et les consommateurs du service, appelés **Clients**.

Ainsi, via un dispositif quelconque (ordinateur, tablette, smartphone), un **Client** peut demander un service à un **Serveur.** Le dispositif sert alors de médiateur entre le consommateur de service (le **Client**) et le fournisseur de service (le **Serveur**), et sait comment parler avec les deux.

Le diagramme ci-dessous décrit ce processus :



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

- Le **Front-End :** il s'agit de la partie du logiciel qui interagit avec les utilisateurs, même s'ils se trouvent sur des plateformes différentes avec des technologies différentes. Dans une architecture Client-Serveur, les modules *front-end* sont conçus pour interagir avec tous les appareils existant sur le marché. Cela comprend les écrans de connexion, les menus, les écrans de données et les rapports qui fournissent et reçoivent des informations des utilisateurs. Par exemple, la plupart des outils et frameworks de développement permettent de créer une version du programme qui fonctionne à la fois pour les PC, les tablettes et les téléphones.
- Le **serveur d'application (Application Server)** : il s'agit du Serveur où sont installés les modules logiciels de l'application. Il se connecte à la base de données et interagit avec les utilisateurs. Le serveur d'application est comme le cuisinier du restaurant de notre exemple.
- Le **serveur de base de données (Database Server)** : il contient les tables, les index et les données gérés par l'application. Les recherches et les opérations d'insertion/de suppression/de mise à jour sont exécutées ici.

V.A.1.b. Avantages

L'architecture Client-Serveur sépare le matériel, le logiciel et la fonctionnalité du système. Par exemple, si une adaptation du logiciel est nécessaire pour un pays spécifique, autrement dit si un changement de fonctionnalité est nécessaire, celui-ci peut être adapté dans le système sans avoir à développer une nouvelle version pour les téléphones, les tablettes ou les ordinateurs portables.

En outre, puisque cette architecture sépare le matériel, le logiciel et la fonctionnalité du système, seule la partie *front-end* doit être adaptée pour communiquer avec différents appareils.

V.A.1.c. Inconvénients

Dans le cas où énormément de Clients demanderaient simultanément des données au Serveur, celui-ci pourrait se voir surchargé, et donc cela pourrait provoquer des interruptions de services.

De plus, si ces interruptions de services sont provoquées par la panne du Serveur, alors aucun utilisateur ne pourrait plus utiliser le système.

V.A.1.d. Utilisation

Ce modèle d’architecture est particulièrement adapté pour le déploiement de solutions logicielles telles que des Progiciels de gestion intégré (ERP), des serveurs d’impression ou encore des serveurs de gestion électronique de documents.

V.A.2. La norme SQL

Notion à appliquer au niveau COMPOSANT

Après avoir créé la BDD au sein du SGBDR, les collaborateurs vont devoir interagir avec elle. Pour cela, ils devront utiliser un langage de communication normé, dénommé SQL. La première norme SQL est l'ISO/CEI 9075 adopté par l'ISO en 1987.

Ce langage de manipulation de données permettra aux collaborateurs de LAE d'accéder aux données de la base, de modifier leur contenu, d'insérer ou de supprimer des lignes (enregistrements).

Ce langage s'appuie sur quatre commandes de base qui sont *SELECT*, *INSERT*, *DELETE* et *UPDATE*. Néanmoins, le SGBDR veille au grain et ces quatre ordres ne seront pas toujours autorisés par l'administrateur de la BDD qui est le seul à pouvoir attribuer ou non les droits d'utilisation sur ces commandes.

Pour le collaborateur lambda, l'administrateur de la base pourra indiquer que seul l'ordre *SELECT* est utilisable. Les ordres de modification de la base ne devront alors être accessibles qu'aux collaborateurs ayant le rôle *Editor* pour des raisons de sécurité.

V.A.2.a. La sélection des données

La commande SELECT va permettre aux collaborateurs de réaliser des requêtes simples, rapidement, même sans connaissances approfondies en langage de programmation.
C'est l'ordre de base qui permet d'indiquer au serveur un souhait d'extraction des données.
C'est une commande très puissante, pour peu que toutes ses fonctions et possibilités soient connues par le collaborateur. Il est alors possible de réaliser des requêtes complexes, avec de nombreuses tables. Cependant, il sera nécessaire de faire attention aux performances qui peuvent se dégrader très rapidement sur une commande SQL mal construite ou n'utilisant pas les bons index de tables.
En outre, il est possible de compléter la commande SELECT avec d'autres expressions de sélection ; le tableau ci-dessous listant les plus utilisées :

Clause	Expression
SELECT	*Liste des colonnes et/ou des éléments d'extraction*
FROM	*Table(s) source(s)*
WHERE	*Condition(s) ou restriction(s), optionnelle*
GROUP BY	*Regroupement(s), optionnelle*
HAVING	*Condition(s) ou restriction(s) sur le(s) regroupement(s), optionnelle*
ORDER BY	*Tri(s)*

Nota : d'autres expressions existent, telles que DEFAULT ou ALL, néanmoins celles listées ci-dessus couvriront 90 % des requêtes…

La syntaxe de l'expression *SELECT* est :

```
SELECT table1_colonne1 ([, table1_colonne2, table2_colonne1…])
```

```
FROM table1 ([, table2, …])
```

V.A.2.b. La modification des données

l'insertion

L'opération d'insertion d'une donnée est réalisée par la commande SQL *INSERT*.

Cette commande va ajouter un ou plusieurs *tuples* dans une base de données relationnelles.

Un tuple est un type de données composé, servant à stocker une collection d'éléments. Les éléments d'un tuple sont des valeurs qui ne sont pas nécessairement du même type ; ils sont, en général, placés entre accolades et séparés par des virgules.

La syntaxe de la commande INSERT est la suivante :

```
INSERT INTO table (colonne1 [,colonne2, colonne3, ...])
VALUES (value1 [, valeur2, valeur3…)]
```

Le nombre de colonnes doit être identique au nombre de valeurs.

Si une colonne n'est pas spécifiée, sa valeur par défaut lui sera affectée.

Les valeurs insérées doivent respecter toutes les contraintes tel que les *clés étrangères, clés primaires,* et les colonnes **NOT NULL**.

Si la commande contient une erreur de syntaxe, ou si une contrainte n'est pas respectée, les valeurs ne sont pas insérées et une erreur est rapportée.

la suppression

La commande *DELETE* en SQL permet de supprimer des lignes dans une table.

En utilisant cette commande associé à *WHERE,* il est alors possible de sélectionner les lignes concernées qui seront supprimées.

AVERTISSEMENT *: avant d'essayer de supprimer des lignes, il est recommandé d'effectuer une* ***sauvegarde de la base de données****, ou tout du moins de la table concernée par la suppression. Ainsi, s'il y a une mauvaise manipulation, il est toujours possible de restaurer les données.*

La syntaxe pour supprimer des ligne est la suivante :

```
DELETE FROM 'table'
WHERE condition
```

AVERTISSEMENT : *si aucune condition* `WHERE` *n'est spécifiée, alors toutes les lignes de la table* `'table'` *seront supprimées et la table sera alors vide.*

la mise à jour

La commande *UPDATE* permet d'effectuer des modifications sur des lignes existantes.

Cette commande peut est utilisée avec *WHERE* pour spécifier sur quelles lignes doivent porter la ou les modifications.

La syntaxe d'une requête *UPDATE* est la suivante :

```
UPDATE table
SET nom_colonne1 = 'nouvelle_valeur1' ([, nom_colonne2 = nouvelle_valeur2, ...)]
WHERE condition
```

Cette syntaxe permet d'attribuer une nouvelle valeur à la colonne `nom_colonne1` pour les lignes qui respectent la condition stipulé avec *WHERE*.

Il est aussi possible d'attribuer la même valeur à la colonne `nom_colonne1` pour toutes les lignes d'une table si la condition *WHERE* n'était pas utilisée.

V.A.3. La norme ACID

Notion à appliquer au niveau COMPOSANT

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.A.3.a. 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.A.3.b. 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 été antérieur en appliquant le principe d'*Atomicité*.

V.A.3.c. 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.A.3.d. 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.

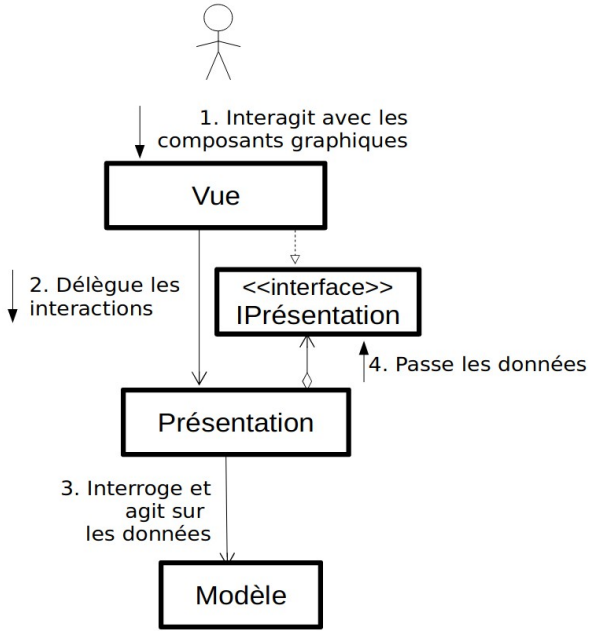
Pour permettre cette durabilité, des « **changelogs »** sont utilisés et pris pour références chaque fois que la BDD est redémarrée.

V.A.4. Théorie du MVP

Le Design Pattern MVP est un dérivé de son parent le patron de conception MVC. Ce patron définit trois types de rôles au sein d'une architecture logicielle :

- **Modèle** : ce composant représente les données manipulées à travers l'interface utilisateur. Ces dernières sont, en général, contenues au sein de base de données ou de fichier d'échange de données en XML ou en JSON, par exemple.
- **Vue** : ce composant affiche une représentation graphique des données (Vue) à l'utilisateur. Cette dernière n'est ni plus ni moins que l'UI d'une application.
- **Présentation** : ce composant est la partie communicant avec les deux précédentes. Dans un premier temps, ce composant traduit et transmet les commandes de l'utilisateur envoyées de la Vue vers le Modèle. Dans un deuxième temps, ce composant formate et affiche les données de réponse envoyées par le Modèle dans la Vue (lol ?).

Le diagramme fonctionnel ci-dessous représente le principe MVP :



En appliquant ce modèle, la logique métier pourra évoluer dynamiquement AVEC le composant Présentation.

De plus, en raison de l'interface utilisateur et des mécanismes d'accès aux données étroitement couplés, les couches de Présentation et de Vue ne relèveront pas de la même activité ou du même fragment, et pourront donc évoluer indépendamment.

En complément, la modularité et la testabilité de ce modèle lui confère un fort potentiel de maintenabilité.

Enfin, les points clés de l'architecture MVP sont :

- la communication entre le composant Vue et celui de Présentation se fait via une interface appelée tout simplement **Contrat** ;
- le composant Présentation ne gère qu'un seul composant à la fois, c'est à dire qu'il existe une relation *un à un* entre le composant Vue et celui de Présentation ;
- Les composants Modèle et Vue n'ont, chacun, aucune connaissance de l'existence de l'autre.

V.A.5. Définition du Modèle, de la Vue et de la Présentation

En appliquant le Design Pattern MVP, défini précédemment, à l'architecture cible fournie par le cabinet IT extérieur, il va alors être possible de factoriser plusieurs éléments éclatés et éparpillés au sein de ce schéma conceptuel.

Ainsi, pour la cohérence du document, cette étude va considérer séquentiellement chacun des trois composant du principe MVP :

- en ce qui concerne le **composant Modèle**, les bases de données prévues au sein de l'architecture cible ont été définies pour répondre spécifiquement à chaque besoin fonctionnel ; ces dernières s'avéreront donc indispensables, ne pourront être modifiées et devront être considérées comme **INVARIANTES** ;

- en ce qui concerne le **composant Vue**, les IHM définies dans l'architecture cible ont été prévues pour faire communiquer les acteurs en présence avec les différents systèmes/outils ; ainsi, l'existence même de ces Interfaces Homme-Machine ne pouvant être remise en cause, elles seront, elles aussi, considérées comme **INVARIANTES** ;

- en ce qui concerne le **composant Présentation**, celui **aura deux principales responsabilités** :
    - récupérer les données des différents modèles (bases de données),
    - prendre des mesures en fonction des notifications d'entrées des utilisateurs au travers du composant Vue.

En adossant ces responsabilités, le composant Présentation va permettre d'adapter dynamiquement les interfaces graphiques de rendu, en fonction du contexte rencontré ; ce dernier pouvant être composé par l'acteur (application web), le besoin fonctionnel, les périphériques impactés…C'est donc cet unique élément, le composant Présentation, qu'il faudra adapter de façon à ce qu'il réagisse dynamiquement à tous les contextes qui se présenteront.

En conclusion, il est maintenant nécessaire de définir précisément ce composant Présentation en annonçant l'**utilisation d'un Système GED** au sein du contexte de SCS Magazine.

V.B.Mappage à la topologie informatique

V.B.1. IaaS

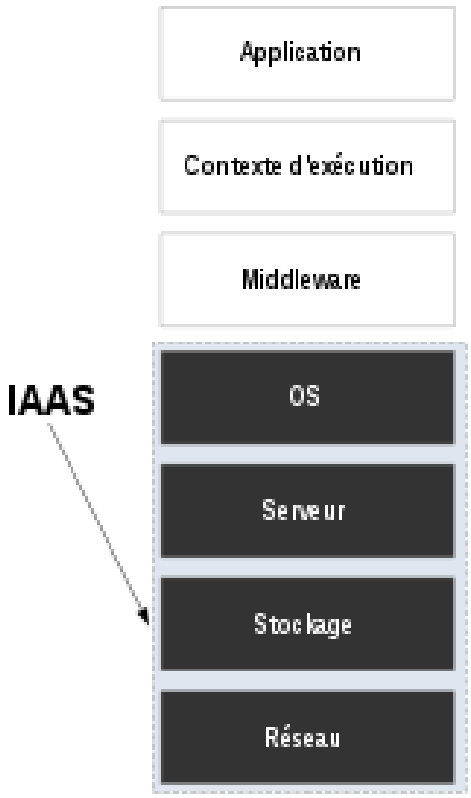
Tel qu'il a été développé dans le document d'évaluation de la conformité puis dans ce document, ce projet sera basé sur une infrastructure de type IaaS.

V.B.1.a. Définition

L'IaaS est un type de modèle de service de cloud computing dans lequel les ressources de calcul sont hébergées dans un cloud public, un cloud privé ou un hybrid cloud.

Le modèle IaaS permet aux entreprises de transférer tout ou partie de leur utilisation de l'infrastructure de datacenter sur site ou colocalisée vers le cloud, où elle est détenue et gérée par un fournisseur de cloud. Ces éléments d'infrastructure rentables peuvent inclure du matériel de calcul, de réseau et de stockage, ainsi que d'autres composants et logiciels.

Dans le modèle IaaS, le fournisseur de Cloud possède et exploite le matériel et les logiciels et possède ou loue également le datacenter. Lorsque qu'une solution IaaS est mise à disposition, celle-ci comprend alors la location des ressources de calcul ou de stockage, par exemple, leur mise en service au juste besoin et la facturation des ressources consommée. Pour certaines ressources, notamment de calcul, seules les ressources utilisées sont facturées, alors que pour d'autres, tel que le stockage, c'est la capacité utilisée qui est prise en compte.



V.B.1.b. Fonctionnement

Dans un modèle IaaS classique, une entreprise utilise des services comme le calcul, le stockage et les bases de données d'un fournisseur de Cloud. Ce dernier offre ces services en hébergeant du matériel et des logiciels dans le Cloud. L'entreprise n'a alors plus besoin d'acquérir et de gérer son propre équipement, ni d'espace pour héberger cet équipement, et le coût passe à un modèle de paiement à l'utilisation.

Lorsque l'entreprise a besoin de moins de ressources, elle paie moins. Et à mesure qu'elle se développe, elle peut mettre en service des ressources de calcul supplémentaires et d'autres technologies en quelques minutes.

Dans un scénario classique sur site, une entreprise gère et entretient son propre datacenter. Cette entreprise doit investir dans des serveurs, des capacités de stockage, des logiciels et d'autres technologies et embaucher du personnel ou des sous-traitants informatiques pour acheter, gérer et mettre à niveau tout l'équipement et les licences. Le datacenter doit être conçu pour répondre aux pics de demande, même si les charges de travail diminuent parfois et que ces ressources restent inactives. Inversement, si l'entreprise se développe rapidement, le service informatique pourrait avoir du mal à suivre le rythme.

Au minimum, l'infrastructure cloud comprend les services de base de calcul, de stockage et de réseau. En outre, elle y inclut désormais également des services de plus haut niveau (parfois appelés PaaS), tels que les bases de données relationnelles et NoSQL, le traitement des données en temps réel et par lots, les pipelines et services de développement, les conteneurs et les fonctions.

Contrairement au modèle SaaS, l'IaaS n'est pas destiné à l'utilisateur final type. L'IaaS est destiné aux applications et aux opérations informatiques, aux DevOps, aux administrateurs système et de base de données et aux développeurs qui travaillent sur l'intégralité du système.

V.B.1.c. Avantages

L’IaaS offre quatre avantages principaux qui permettent aux entreprises d'avancer plus rapidement et d'atteindre leurs objectifs de transformation numérique :

- L'infrastructure cloud réduit le temps et le coût de l'approvisionnement et de la mise à l'échelle des environnements pour les tests de développement et la production. Cela donne aux développeurs et aux équipes DevOps davantage de liberté pour expérimenter et innover.

- En rendant les services informatiques disponibles à la demande, les entreprises peuvent faire évoluer leur infrastructure vers le haut ou vers le bas selon leurs besoins, en ne payant que ce qu'elles utilisent sur une base horaire, quotidienne ou mensuelle, tout en gérant des pics d'activité plus importants que ce qui est possible dans la plupart des environnements sur site.

- L’IaaS peut donner aux entreprises l’accès à des équipements et services nouveaux et améliorés (tels que les processeurs, le stockage, le matériel de mise en réseau et l’orchestration de conteneurs les plus récents) que de nombreuses entreprises ne pourraient se permettre d’acquérir sur site ou auxquels elles ne pourraient pas accéder aussi rapidement.

- L’IaaS est disponible dans la plupart des zones géographiques, avec une présence régionale à proximité des grands centres de population, ce qui permet aux entreprises de développer leur empreinte en ligne plus rapidement.

Le passage à un modèle IaaS peut être transformationnel pour les entreprises, en particulier pour leurs services informatiques. Au lieu de consacrer une grande partie de leur temps à la gestion et à la prise en charge de l’infrastructure sur site, le personnel informatique peut consacrer davantage d’heures à des activités à forte valeur ajoutée qui rendent l’entreprise plus efficace et productive. Le modèle de paiement à l’utilisation permet également de réduire les erreurs de prévision et de s’assurer que les coûts sont bien en phase avec les besoins réels.

L’IaaS accroît la stabilité, la fiabilité et la capacité de soutien : les entreprises choisissent le modèle IaaS pour leurs charges de travail stratégiques en raison de sa stabilité, fiabilité et soutenabilité inégalées. Par rapport aux systèmes sur site, le modèle IaaS offre plus de temps de disponibilité, une redondance intégrée à chaque couche, de meilleures options de sécurité et de protection contre les catastrophes, et une évolutivité avec laquelle les environnements sur site ne peuvent pas rivaliser.

V.B.1.d. Inconvénients

Cependant, l’IaaS n’est pas exemplt d’inconvient et il est possible d’en dénombrer quatre principaux :

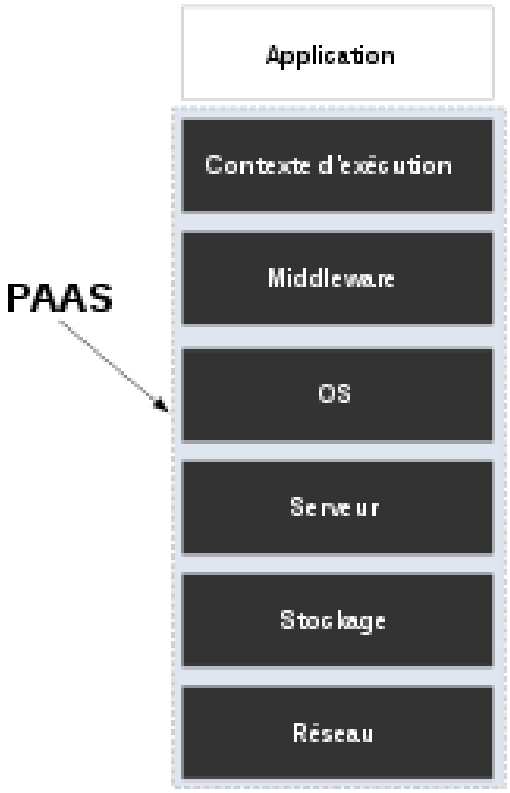
- La dépendance vis-à-vis du fournisseur, dont la seule responsabilité est la disponibilité et la sécurité du service.
- L'accès à Internet est indispensable (les problèmes de connexion à Internet causent aussi des problèmes avec l’environnement IaaS).
- Le changement de fournisseur peut s’avérer très compliqué.
- Les problèmes éventuels liés à la politique de confidentialité en raison de l’emplacement des serveurs du fournisseur.

V.B.2. PaaS

V.B.2.a. Définition

Dans le modèle PaaS, les développeurs louent essentiellement tout ce dont ils ont besoin pour construire une application, en s'appuyant sur un fournisseur de cloud pour les outils de développement, l'infrastructure et les systèmes d'exploitation. C'est l'un des trois modèles de service de l'informatique en cloud.

Le PaaS simplifie considérablement le développement d'applications web ; du point de vue du développeur, toute la gestion du backend se fait en arrière-plan. Bien que le PaaS présente certaines similitudes avec l'informatique sans serveur, il existe de nombreuses différences essentielles entre eux.



V.B.2.b. Fonctionnement

Le PaaS est accessible via n'importe quelle connexion Internet, ce qui permet de créer une application complète dans un navigateur web. L'environnement de développement n'étant pas hébergé localement, les développeurs peuvent travailler sur l'application depuis n'importe quel endroit du monde. Cela permet à des équipes réparties sur plusieurs sites géographiques de collaborer. Cela signifie également que les développeurs ont moins de contrôle sur l'environnement de développement, bien que cela se traduise par des frais généraux beaucoup moins importants.

V.B.2.c. Avantages

Les sociétés utilisant le PaaS, l’ont choisi pour :

- **mettre leur produit sur le marché rapidement** : PaaS est utilisé pour créer des applications plus rapidement que ce qui serait possible si les développeurs devaient se soucier de la création, de la configuration et de l'approvisionnement de leurs propres plates-formes et infrastructures backend. Avec le PaaS, il leur suffit d'écrire le code et de tester l'application, et le fournisseur se charge du reste.

- **utiliser un seul environnement du début à la fin** : PaaS permet aux développeurs de construire, tester, déboguer, déployer, héberger et mettre à jour leurs applications dans le même environnement. Cela permet aux développeurs de s'assurer qu'une application web fonctionnera correctement telle qu'elle est hébergée avant sa sortie, et cela simplifie le cycle de vie du développement des applications.

- **faire des économies financières** : dans de nombreux cas, le PaaS est plus économique que l'IaaS. Les coûts indirects sont réduits car les clients du PaaS n'ont pas besoin de gérer et de fournir des machines virtuelles. En outre, certains fournisseurs ont une structure de tarification par répartition, dans laquelle le vendeur ne facture que les ressources informatiques utilisées par l'application, ce qui permet généralement aux clients de réaliser des économies. Toutefois, chaque fournisseur a une structure tarifaire légèrement différente, et certains fournisseurs de plateformes facturent un montant fixe par mois.

- **faciliter l'octroi de licences** : les fournisseurs de PaaS s'occupent de toutes les licences pour les systèmes d'exploitation, les outils de développement et tout ce qui est inclus dans leur plate-forme.

V.B.2.d. Inconvénients

Néanmoins, le PaaS n’est pas exempt d’inconénients, et il est possible d’en énoncer trois principaux :

- Le **verrouillage propriétaire** : il peut devenir difficile de changer de fournisseur de PaaS, car l'application est construite au moyen des outils du fournisseur et spécifiquement pour sa plateforme. Chaque fournisseur peut avoir des exigences différentes en matière d'architecture. Les différents fournisseurs peuvent ne pas prendre en charge les mêmes langages, bibliothèques, API, architecture ou système d'exploitation utilisés pour construire et exécuter l'application. Pour changer de fournisseur, les développeurs peuvent avoir besoin de reconstruire ou de modifier sensiblement leur application.

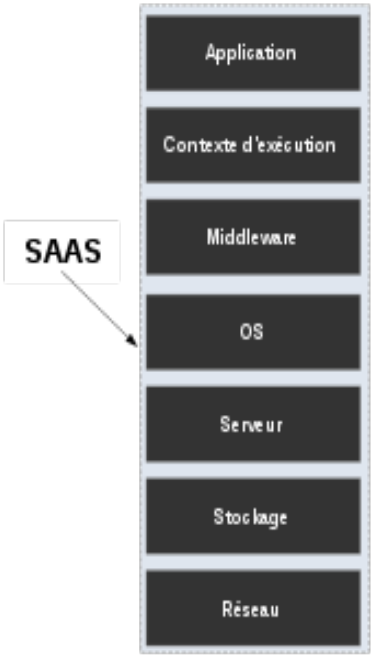
- La **dépendance vis-à-vis des fournisseurs** : les efforts et les ressources nécessaires pour changer de fournisseur de PaaS peuvent rendre les entreprises plus dépendantes de leur fournisseur actuel. Un petit changement dans les processus internes ou l'infrastructure du fournisseur pourrait avoir un impact énorme sur les performances d'une application conçue pour fonctionner efficacement sur l'ancienne configuration. En outre, si le fournisseur change son modèle de tarification, une application peut soudainement devenir plus coûteuse à exploiter.

- Les d**éfis en matière de sécurité et de conformité** : dans une architecture PaaS, le fournisseur externe stocke la plupart ou la totalité des données d'une application, tout en hébergeant son code. Dans certains cas, le fournisseur peut même stocker les bases de données par l'intermédiaire d'une autre tierce partie, un fournisseur IaaS. Bien que la plupart des fournisseurs de PaaS soient de grandes entreprises disposant d'une sécurité solide, il est difficile d'évaluer et de tester pleinement les mesures de sécurité protégeant l'application et ses données. En outre, pour les entreprises qui doivent se conformer à des réglementations strictes en matière de sécurité des données, la vérification de la conformité d'autres fournisseurs externes ajoutera encore plus d'obstacles à la mise sur le marché.

V.B.3. SaaS

V.B.3.a. Définition

Le SaaS est un modèle de distribution de logiciel au sein duquel un fournisseur tiers héberge les applications et les rend disponibles pour ses clients par l'intermédiaire d'internet. C'est l'une des quatre catégories principales de Cloud Computing, au même titre que l'IaaS, la PaaS ou encore le DaaS. Parmi les principaux fournisseurs d'un logiciel SaaS, on retrouve Salesforce, Oracle, IBM, Intuit ou encore Microsoft. La structure SaaS est représentée dans le diagramme ci-dessous :



Le modèle SaaS comprend une architecture multi-tenant qui permet à tous les utilisateurs et à toutes les applications de partager une infrastructure et une base de code uniques et communes, dont la maintenance est centralisée.

Grâce au modèle SaaS, les applications peuvent être rapidement personnalisées. De ce fait, les semaines ou les mois nécessaires à la mise à jour des logiciels d'entreprise traditionnels n'ont tout simplment plus lieu d'être.

L'architecture SaaS garantit l'unicité de ces personnalisations pour chaque entreprise ou utilisateur et les préserve toujours lors des mises à niveau. Cela signifie que les fournisseurs de SaaS peuvent effectuer des mises à niveau plus souvent, avec moins de risques pour les clients et un coût d'adoption beaucoup plus faible.

La solution SaaS offre un meilleur accès aux données depuis n'importe quel périphérique en réseau. Elle facilite la gestion des privilèges, le contrôle de l'utilisation des données et permet à chacun de consulter les mêmes informations au même moment.

V.B.3.b. Fonctionnement

Le SaaS est étroitement lié aux modèles de livraison de service de logiciel ASP et informatique à la demande. Le modèle de gestion d'application SaaS hébergé est similaire à l'ASP et dans les deux cas, le fournisseur héberge le logiciel du client et le délivre aux utilisateurs finaux via internet.

Avec le modèle de Logiciel à la Demande, le fournisseur offre aux clients un accès basé sur le réseau à une simple copie d'une application spécifiquement créée par le fournisseur pour la distribution Software as a Service. Le code source de l'application est le même pour tous les clients. Quand de nouvelles fonctionnalités sont déployées, tous les clients peuvent en profiter.

En fonction du niveau de service, les données du client peuvent être stockées localement, sur le Cloud, ou les deux à la fois. Les entreprises peuvent intégrer des applications à d'autres logiciels en utilisant des API. Par exemple, une entreprise peut développer ses propres outils logiciels et utiliser l'API du fournisseur de service pour intégrer ces outils à l'offre.

V.B.3.c. Avantages

Grâce à un logiciel SaaS, *SCS Magazine* n'aura plus besoin d'installer et de lancer des applications sur leurs propres ordinateurs ou sur leurs *Data Centers*. Le coût d'acquisition de matériel est ainsi éliminé, au même titre que les coûts d'approvisionnement et de maintenance, de licence de logiciel, d'installation et de support. On compte également plusieurs autres avantages.

Au lieu d'investir dans un logiciel à installer, et dans un équipement permettant de le prendre en charge, les utilisateurs souscrivent à une offre SaaS. En général, l'offre se présente sous la forme d'un abonnement mensuel dont le tarif est proportionnel à l'utilisation. Grâce à cette flexibilité, les entreprises peuvent organiser leur budget avec plus de précision et de facilité. De plus, il est possible de résilier l'abonnement à tout moment pour couper court aux dépenses.

Un autre avantage est la haute scalabilité. En fonction de ses besoins, l'utilisateur peut accéder à plus ou moins de services et à des fonctionnalités à la demande. Le Logiciel en tant que Service est donc adapté aux besoins propres à chaque business.

De même, plutôt que de devoir acheter régulièrement de nouveaux logiciels, les utilisateurs peuvent compter sur le fournisseur SaaS pour effectuer des mises à jour automatiquement et gérer l'ajout de patchs correctifs. L'entreprise a donc moins besoin d'une équipe d'informaticiens internes.

Enfin, étant donné que les applications SaaS sont délivrées via internet, les utilisateurs peuvent y accéder depuis n'importe quel appareil connecté et n'importe quelle position géographique. L'accessibilité est l'un des grands points forts de ce modèle.

Par ailleurs, une application SaaS peut être utilisée par des milliers, voire des millions d'utilisateurs finaux simultanément puisqu'elle est stockée sur le Cloud.

V.B.3.d. Inconvénients

Parmi les principaux inconvénients et faiblesses du SaaS, les risques liés à son utlisation sont nombreux :

- Les données sont confiées à un fournisseur : bien que le fournisseur de services s'engage contractuellement à préserver la sécurité et la confidentialité de vos données, vous confiez bel et bien les données de votre entreprise à un fournisseur tiers qui les enregistre chez lui. Les violations de données, les cyberattaques et autres incidents pouvant compromettre la confidentialité de vos données ne relèvent plus de votre contrôle. La sécurité des services en Cloud reste un thème particulièrement sensible. En Europe, le nouveau Règlement général sur la protection des données encadre relativement bien ces activités, à condition que le fournisseur du service soit bien soumis à ce règlement.
- Le risque d'interruption des services : si un fournisseur de SaaS venait à faire faillite, ou s'il est amené pour une raison quelconque à interrompre le service, vous ne pourrez plus en bénéficier. Il est même possible que vous puissiez perdre l'ensemble de vos données et de vos documents. Il est certes très rare qu'un SaaS soit interrompu. Généralement, en cas d'interruption des services, le fournisseur convient d'un délai avec ses utilisateurs pour leur permettre de rapatrier leurs données et leurs documents vers d'autres supports ou serveurs. Dans certains cas, un service vient alors en remplacer un autre, et les données doivent être souvent transférées.
- La nécessité d'une connexion Internet constante et rapide : le SaaS est un service en ligne qui nécessite immanquablement une bonne connexion internet. Plusieurs prestataires proposent un mode de fonctionnement hors ligne qui permet de travailler sans être connecté à Internet, et de synchroniser les données dès que la connexion est rétablie. Toutefois, pour pouvoir exploiter correctement un SaaS, vous avez besoin d'une connexion Internet constante. Les problèmes de réseau peuvent engendrer des délais de téléchargement très gênants avec, dans certains cas, des pertes d'exploitation.
- La non-disponibilité des logiciels en cas de panne du système : de la même manière, plusieurs SaaS ne peuvent pas être utilisés si un fournisseur doit suspendre temporairement ses services, notamment en cas de travaux d'entretien ou de panne de serveurs.
- La compatibilité nécessaire avec les systèmes d'exploitation et les navigateurs : les webtools ont un fonctionnement parfois différent en fonction du navigateur que vous utilisez. Vous pouvez rencontrer des problèmes de compatibilité liés aux systèmes d'exploitation, en particulier si vous êtes un adepte de macOS. La plupart des fournisseurs de SaaS optimisent cependant leurs utilitaires pour Windows. Quelques précisons cependant : les problèmes liés au choix du navigateur internet sont assez rares.
- L'utilisation de logiciels non terminés : comme les programmes SaaS ne sont ni développés, ni commercialisés sur un modèle traditionnel, il arrive parfois que les prestataires de SaaS mettent en ligne des applications encore en cours de développement. Ils le font parfois avant de lancer les béta-tests de grande envergure, ou avant d'implémenter de nouvelles fonctionnalités importantes. De manière générale, les produits SaaS sont soumis à des contrôles qualité moins rigoureux.

V.C. Mappage aux politiques opérationnelles

L'IaaS devient de plus en plus populaire dans tous les secteurs d'activité et ses applications s'élargissent. La principale base d'utilisateurs IaaS comprend les opérateurs informatiques, les développeurs d'applications, les équipes DevOps, les administrateurs système et de bases de données et les développeurs qui travaillent sur l'intégralité du système dans les entreprises qui créent et exécutent des applications. Il est également utilisé par les entreprises qui souhaitent disposer d'une infrastructure Cloud flexible pour prendre en charge leur ERP, leurs services financiers, leur Supply Chain et d'autres applications internes.

Initialement, l'IaaS était principalement utilisé par les organisations natives du Cloud pour des charges de travail temporaires, expérimentales ou susceptibles de changer de façon inattendue.

Aujourd'hui, de nombreuses grandes entreprises, attirées par les avantages de l'IaaS, se tournent de plus en plus vers ce modèle pour prendre en charge leurs back-office, systèmes d'enregistrement et d'autres charges de travail stratégiques.

En matière d'innovation, l'IaaS est également en passe de devenir une solution privilégiée. Les entreprises qui conservent encore leurs datacenters sur site trouvent qu'il est très difficile et coûteux de faire plus que d'assurer le simple fonctionnement. Pour innover et rester compétitives sur le marché, les entreprises tournées vers l'avenir transfèrent leurs datacenters vers le Cloud. En s'appuyant sur l'IaaS, le PaaS ou le SaaS, elles sont en mesure de libérer leurs talents et leurs ressources pour réaliser les innovations qu'elles imaginent et développer leur activité.

V.C.1. IaaS

V.C.1.a. Fonctionnalités et avantages

L'IaaS offre de nombreux avantages par rapport aux datacenters classiques sur site. Avec l'IaaS, SCS Magazine sera en mesure de :

Fonctionnalité	Avantage
Réduire leurs dépenses	Les entreprises qui sont passées à l'IaaS n'ont pas à acheter, gérer et entretenir leur infrastructure, et elles paient uniquement pour ce qu'elles utilisent, même sur des périodes d'amortissement de cinq ans ou plus.
Améliorer la continuité des activités	L'infrastructure Cloud fournit généralement un degré de disponibilité plus élevé et davantage d'options de récupération après sinistre que les déploiements sur site, car elle dispose d'une redondance intégrée à chaque couche, offre plusieurs domaines de défaillance et des emplacements répartis géographiquement, et est exécutée à grande échelle par des experts des opérations.
Accélérer l'innovation	L'IaaS permet de tester de nouveaux produits et concepts rapidement, facilement et à moindre coût. Au lieu d'avoir à développer des prévisions détaillées et à investir dans de nouvelles infrastructures, les entreprises peuvent augmenter leur infrastructure Cloud en quelques minutes, puis la faire évoluer à la hausse ou à la baisse selon les besoin.

Fonctionnallité	Avantage
Tirer parti des dernières technologies.	De nombreux fournisseurs de Cloud mettent en package et déploient de nouveaux matériels et logiciels, notamment des structures d'intelligence artificielle et de machine learning, et ce bien avant que les entreprises puissent les mettre en œuvre sur site.
Accélérer la mise en service	Même les infrastructures sur site virtualisées souffrent de longs délais de mise en service de plusieurs semaines, voire plusieurs mois. Avec l'IaaS, des environnements d'application entiers peuvent être mis en service en quelques minutes.
Se concentrer sur leur cœur de métier	L'IaaS libère les services informatiques et leur évite de consacrer jusqu'à la moitié de leurs ressources à la gestion et à la maintenance du matériel et des logiciels sur site. Avec l'IaaS, les entreprises peuvent également permettre aux équipes DevOps et à d'autres équipes d'accéder à l'infrastructure elles-mêmes, de sorte qu'elles puissent procéder à l'exécution et aux tests sans délai.
Évoluer plus rapidement	Les entreprises ont besoin de davantage de ressources lors des pics de charges de travail, par exemple pendant les périodes de reporting mensuelles. Avec l'IaaS, l'infrastructure peut évoluer en quelques minutes, de sorte que les rapports peuvent être exécutés rapidement et que le personnel peut se concentrer sur des activités plus stratégiques pour l'entreprise.

Le fournisseur de cloud œuvre à s'assurer que l'environnement IaaS soit aussi efficace que possible. Il dispose souvent de matériel de pointepour lequel SCS Magazine n'aura pas besoin de le rechercher et de l'acheter elle-même.

En outre, aucune formation spécialisée ni de longs cycles de mise en service pour mettre à niveau votre infrastructure ne sera nécessaire. Au lieu de cela, SCS Magazine disposera du temps et des ressources nécessaires pour se concentrer sur son activité métier.

V.C.1.b. Exemple d'utilisation

Ainsi, SCS Magazine pourra utiliser l'IaaS dans les situations suivantes :

- le Test et le développement : les équipes DevOps peuvent configurer et démanteler des environnements de test et de développement rapidement et à faible coût, ce qui leur permet de commercialiser plus rapidement de nouvelles applications.
- L'utilisation d'applications traditionnelles : l'IaaS prend en charge à la fois les applications natives du Cloud et les applications d'entreprise traditionnelles, y compris les systèmes ERP et les applications d'analytiques métiers.
- L'hébergement de sites Web et applications : de nombreuses entreprises gèrent leurs sites Web sur un modèle IaaS afin d'optimiser les coûts. L'IaaS prend également en charge les applications Web et mobiles, qui peuvent être rapidement déployées et évolutives.

- Le stockage, la sauvegarde et la récupération de données : le stockage et la sauvegarde des données sur site, ainsi que la planification de la prévention des sinistres et la récupération après sinistre, nécessitent énormément de temps et expertise. Le transfert de leur infrastructure vers le Cloud aide les entreprises à réduire les coûts et leur permet de se concentrer sur d'autres tâches.
- Le calcul informatique haute performance : avec son modèle de paiement à l'utilisation, l'IaaS rend plus abordables le calcul informatique haute performance (HPC) et d'autres tâches orientées projet, qui traitent d'énormes volumes de données.

V.C.2. PaaS

V.C.2.a. Fonctionnalités et avantages

Les principales offres des fournisseurs de PaaS sont les suivantes :

- Outils de développement
- Intergiciels
- Systèmes d'exploitation
- Gestion des bases de données
- Infrastructure

Les différents fournisseurs peuvent également proposer d'autres services, et ceux-ci sont les principaux services du PaaS, à savoir :

- Les outils de développement : les fournisseurs de PaaS proposent une variété d'outils nécessaires au développement de logiciels, notamment un éditeur de code source, un débogueur, un compilateur et d'autres outils essentiels. Ces outils peuvent être proposés ensemble comme un cadre. Les outils spécifiques proposés dépendront du fournisseur, mais les offres PaaS doivent inclure tout ce dont un développeur a besoin pour créer son application.
- Les middlewares : lLes plates-formes offertes en tant que service comprennent généralement des intergiciels, de sorte que les développeurs n'ont pas à les créer eux-mêmes. Un intergiciel est un logiciel qui se situe entre les applications destinées à l'utilisateur et le système d'exploitation de la machine ; par exemple, un intergiciel est ce qui permet à un logiciel d'accéder aux données saisies au clavier et à la souris. Un intergiciel est nécessaire pour exécuter une application, mais les utilisateurs finaux n'interagissent pas avec lui.
- Les systèmes d'exploitation : un fournisseur de PaaS fournira et maintiendra le système d'exploitation sur lequel les développeurs travaillent et sur lequel l'application fonctionne.
- Les bases de données : les fournisseurs de PaaS administrent et maintiennent des bases de données. Ils fournissent généralement aussi aux développeurs un système de gestion de base de données.
- L'infrastructure : PaaS est la couche qui précède IaaS dans le modèle de service de cloud computing, et tout ce qui est inclus dans IaaS est également inclus dans PaaS. Un fournisseur de PaaS gère les serveurs, le stockage et les datacenters physiques, ou les achète à un fournisseur de IaaS.

V.C.2.b. Exemple d’utilisation

En fournissant une plateforme intégrée et prête à l'emploi, et en permettant aux entreprises de se décharger de la gestion de l'infrastructure sur le fournisseur de cloud pour se concentrer sur la création, le déploiement et la gestion des applications, les solutions PaaS facilitent ou font progresser un certain nombre d'initiatives IT, notamment :

- Le développement et la gestion d'API : Grâce à ses structures intégrées, une solution PaaS simplifie considérablement le développement, l'exécution, la gestion et la sécurisation des API (interfaces de programmation d'application). Les équipes peuvent ainsi partager des données et des fonctionnalités entre les applications.
- l’IoT : Une solution PaaS peut prendre immédiatement en charge toute une gamme de langages de programmation (Java, Python, Swift, etc.), ainsi que des outils et des environnements d'application utilisés pour le développement d'applications IoT et le traitement en temps réel des données générées par les appareils IoT.
- Le développement Agile et DevOps : Une solution PaaS peut fournir des environnements entièrement configurés qui permettent d'automatiser le cycle de vie de l'application logicielle, notamment l'intégration, la distribution, la sécurité, les tests et le déploiement.
- La migration cloud et développement cloud natif : Grâce à ses outils prêts à l'emploi et à ses fonctions d'intégration, une solution PaaS peut simplifier la migration des applications existantes vers le cloud. Elle permet en particulier le changement de plateforme (déplacement d'une application vers le cloud, avec des modifications exploitant l'évolutivité du cloud, l'équilibrage de charge et d'autres fonctions), ou la restructuration (création d'une nouvelle architecture de tout ou partie d'une application à l'aide de microservices, de conteneurs et autres technologies natives cloud).
- La stratégie de cloud hybride : Le cloud hybride intègre des services de cloud public, des services de cloud privé et une infrastructure sur site. Il assure l'orchestration, la gestion et la portabilité des applications sur les trois. Il en résulte un environnement informatique réparti, unifié et flexible, dans lequel une entreprise peut exécuter et mettre à l'échelle ses charges de travail traditionnelles déjà existantes ou natives cloud en choisissant le modèle de traitement le mieux adapté. Une solution PaaS judicieusement choisie permet aux développeurs de créer une seule fois, puis de déployer et de gérer partout dans un environnement de cloud hybride.

V.C.3. SaaS

V.C.3.a. Fonctionnalités et avantages

Ces dernières années, le marché du SaaS a beaucoup évolué. Au commencement, personne n'était vraiment certain de la pertinence de ce modèle. Nul n'était sûr que les entreprises accepteraient de payer un abonnement pour accéder à un logiciel, et les banques avaient peur des risques.

Désormais, ce manque de certitudes a totalement disparu. De même, les prix ont baissé et la mise en place est beaucoup plus facile. Cependant, il est toujours nécessaire de proposer des logiciels de qualité à un prix raisonnable pour se démarquer sur le marché des SaaS.

De nos jours, l'intelligence artificielle (IA) est profondément ancrée dans la société. L'IA a le potentiel de perturber le paysage SaaS de diverses manières, en améliorant les caractéristiques clés du modèle SaaS à tous les niveaux. Lorsque le SaaS s'associe aux capacités de l'IA, il permet aux entreprises de tirer davantage de valeur de leurs données. Il peut également automatiser et personnaliser les services, améliorer la sécurité et compléter les capacités humaines.

Le Machine Learning (ML) est l'un des segments du logiciel qui connaît une forte croissance. Le ML est utilisé en SaaS pour automatiser la réactivité dans les rapports et les applications du service client, comme les opérations de chat alimentées par l'IA avec des chatbots en direct. Il permet également d'automatiser le processus d'intégration SaaS. Une nouvelle injection d'innovation ML offre aux services SaaS de s'auto-améliorer, offrant un niveau d'intelligence et d'efficacité opérationnelle.

La troisième des tendances clés du secteur SaaS est axée sur les données. Alors que la transformation numérique s'accélère dans toutes les industries, les entreprises de tous les secteurs se tournent vers les données. Ces dernières leur permettent de rationaliser leurs organisations tout en acquérant une meilleure compréhension de leurs clients ou utilisateurs. Les investissements dans les innovations logicielles en tant que service axés sur l'analyse devraient monter en flèche.

La quatrième tendance SaaS est le SaaS vertical. Alors que le SaaS horizontal se concentre sur les clients de toutes les industries et de tous les secteurs, le SaaS vertical est entièrement personnalisable. Il cible les clients d'industries et de chaînes d'approvisionnement spécifiques. Les logiciels d'analyse des soins de santé, de la vente au détail ou de la logistique moderne en sont des exemples.

À mesure que le secteur du logiciel-service évolue et que l'innovation augmente, de nombreux développeurs ou fournisseurs se concentreront sur la fidélisation des clients en plus de leur acquisition. Cela dit, le SaaS devrait maintenant migrer davantage vers le domaine du PaaS. Il s'agit de développements qui permettent aux entreprises de créer des applications personnalisées pour compléter leurs services originaux. L'un des principaux objectifs de l'évolution du PaaS sera d'aider les startups et les entreprises relativement récentes à se développer rapidement et avec succès.

Ainsi, passer au SaaS est un investissement très rentable pour les entreprises pour de nombreuses raisons.

Tout d'abord, les logiciels en tant que service permettent de réduire les coûts. Ils sont généralement moins chers que les systèmes on-premise, puisque l'entreprise n'a besoin d'implémenter que les logiciels nécessaires et peut ensuite y accéder par le biais d'une simple connexion internet.

Par ailleurs, les SaaS suppriment les besoins en maintenance et en réparation. Les services cloud offrent des options de backups, et les éventuelles pannes de service ont un impact minime car les fournisseurs de services ont davantage de personnel que la plupart des entreprises, et peuvent donc remédier rapidement aux problèmes. La restauration et les sauvegardes de données sont prises en charge au sein de data centers.

Ils permettent également de gagner de l'espace. Les solutions on-premise nécessitent de la place pour les serveurs, le matériel informatique et le personnel, sans parler des câbles et de la ventilation. Au contraire, ils ne nécessitent aucun espace physique supplémentaire. La sécurité des serveurs quant à elle est laissée entre les mains des vendeurs.

En outre, les mises à jour sont gérées par les fournisseurs de service, ce qui permet de s'assurer que les logiciels sont toujours à jour. Certains logiciels on-site peuvent être personnalisés, mais ces personnalisations sont généralement liées à la version du logiciel actuellement déployée.

Enfin, la plateforme Saas permet de réduire considérablement les temps de déploiement par rapport aux systèmes on-premise. Un système cloud peut être déployé dans plusieurs régions, au sein de diverses divisions de l'entreprise, et d'éviter les coûts liés à ces déploiements. Aucun matériel additionnel n'est nécessaire. Ainsi, les entreprises n'ont pas besoin de perdre de temps se procurer une infrastructure informatique et un accès VPN sur différents sites.

V.C.3.b. Exemple d'utilisation

Il est possible de dénombrer des applications SaaS pour les technologies fondamentales des entreprises, comme les emails, la gestion de ventes, la gestion de relations client (CRM), la gestion électronique de documents, la gestion de finances, la gestion des ressources humaines, la facturation et la collaboration.

Néanmoins, il faut garder à l'esprit que l'apparition de ce modèle d' application SaaS dans le jeu vidéo (Gaming as a service) et surtout dans l'Internet des Objets qui se base sur cette infrastructure logicielle afin d'organiser les données récoltées depuis des milliers de capteurs.

VI. Spécifications des attributs partagés

Lecteur cible de ce paragraphe : toute partie prenante

Les attributs spécifiés dans ce paragraphe sont des attributs de type APPLICATIF ; ils ne sont pas associés à des composants spécifiques mais plutôt aux fonctionnalités générales de la solution.

Fonction	Attribut	Objectif	Description
Moteur de recherche	**#i_mr.v**	Trouver facilement et rapidement la bonne version d'un document	• « **mr** » identifie le moteur de recherche utilisé. • « **v** » identifie la version du moteur de recherche utilisé. Application permettant à un utilisateur d'effectuer une recherche locale ou en ligne, c'est-à-dire de trouver des ressources à partir d'une requête composée de termes.
Indexation full texte	**#i_ISO13249_index**	Avoir une meilleure gouvernance documentaire au sein de l'entreprise	• «**index**» représente l'année relative au cadre générale de la norme ISO/IEC 13249 utilisée. La recherche plain texte permet de retrouver des données contenant certains mots, expressions ou formes fléchies de mots, synonymes, etc. dans les lignes des tables, y compris pour des colonnes de type LOB pouvant contenir de grand textes (CLOB, NCLOB) voire des fichiers électroniques binarisés (BLOB).
Génération de PDF	**#b_PDF**	Réduire les coûts de traitement	• « **b_PDF** » binaire prenant deux valeurs TRUE ou FALSE pour indiquer si l'exportation vers un format PDF est disponible dans le contexte du document considéré. La méthode recommandée pour générer des fichiers PDF est à la fois fonction des logiciels à disposition et du type de flux de production auquel la ressource documentaire est destinée.

Fonction	Attribut	Objectif	Description
Gestion automatique de documents	**#s_mgmt_ method**	Gérer plus efficacement des documents au quotidien	• « **mgmt_method** » identifie la méthode de gestion documentaire utilisée pour la ressource documentaire, par exemple, hiérarchique, alphabétique, numéraire… Dans un bon système d'organisation, les reccources le constituant doivent être faciles à ranger, faciles à trouver et réutilisables.
Mode brouillon	**#b_draft**	Gérer les documents de leur création à leur archivage	• « **b_draft** » binaire pouvant prendre deux valeurs TRUE ou FALSE pour indiquer que le dmode brouillon est activé ou non. Le mode brouillon permet de passer en revue les erreurs relatives à un type de document. Ainsi, ce mode permet d'exécuter rapidement des vérifications sur le type de document même avec des configurations incomplètes.
Alerte	**#i_alert**	Avoir une meilleure gouvernance documentaire au sein de l'entreprise	• « **i_alert** » indique qu'une alerte est positionnée sur la ressource en en spécifiant le type. L'alerte identifiée est relative au tableau d'alertes (à produire) et indique s'il s'agit d'un document à modifier, à reprendre, à publier…
Gestion des utilisateurs	**#s_user**	Donner la possibilité à plusieurs collaborateurs de mettre à jour un document gérer plus facilement des documents partagés entre collaborateurs	• « **s_user** » identifie un utilisateur de façon unique
Gestion des droits des documents	**#s_ doc-rights**	Uniformiser les pratiques documentaires	• « s**_doc-rights** » indique le type de droit d'un document selon le modèle UNIX RWXRWXRWX
Circuit de validation	**#s_flow**	Avoir une meilleure gouvernance documentaire au sein de l'entreprise	• « **s_flow** » cette chaîne de caractères identifiera de manière formelle et univoque le flux de travail utilisé

Fonction	Attribut	Objectif	Description
Partage	**N/A**	Partager des données avec un certain nombre de personnes autorisées	Bien que cette notion puisse être considéré comme un attribut, ce ne sera en réalité qu'une conséquence issue de l'utilisation de l'attribut « **s_doc-rigths** »
Gestion des métadonnées	**#hash_ metadata**	Avoir une meilleure gouvernance documentaire au sein de l'entreprise	• « **hash_metadata** » résultat d'une fonction de hachage identifiant de manière formelle et univoque les métadonnées associées à une ressrouce documentaire, telles que l'Editor, les droits du document, le date de création… Dans le hachage, il existe une fonction de hachage qui mappe les clés sur certaines valeurs. Cependant, ces fonctions de hachage peuvent entraîner une collision, c'est-à-dire que deux clés ou plus sont mappées à la même valeur. Le hachage de chaîne évite les collisions. L'idée est de faire en sorte que chaque cellule de la table de hachage pointe vers une liste liée d'enregistrements ayant la même valeur de fonction de hachage. Ici, une fonction de hachage sera crée de sorte que la table de hachage ait un nombre "n" de compartiments. Pour insérer un nœud dans la table de hachage, nous devons trouver l'index de hachage pour la clé donnée. Et il sera calculé en utilisant la fonction de hachage.
Sauvegarde des modifications de métadonnées	**#hash_ hash_ metadata**	Faire coexister plusieurs versions d'un même document	• « **hash_hash_metadata** » résultat d'une fonction de hachage permettant de retrouver les métadonnées d'une ressource documentaire, si celles-ci sont corrompues. En outre, ce hash pourra également servir de vérification pour s'assurer que la ressource documentaire soit pérenne et intègrer.

VII. Performance et configurabilité

Lecteur cible de ce paragraphe : partie prenante technique

Les critères de performance et de configurabilité énoncés ci-dessous seront basés sur des critères minimum pour assurer le bon fonctionnement l'implémentation de la solution préconisée. Ces critères seront à considérer comme des prérequis et reprendront quatre domaines relatifs aux éléments décrits plus avant dans ce document, à savoir :

- L'infrastructure,
- La base de données,
- L'observabilité et la gestion,
- Les services supplémentaires (services non-vitaux pour l'exécution du système à prendre en compte pour le confort d'utilisation).

Infrastructure
• 2 machines virtuelles de calcul avec 1/8 OCPU et 1Go de mémoire chacune ; • Des cœurs Ampere A1 basés sur Arm et 24 Go de mémoire utilisables comme 1 VM ou jusqu'à 4 VM avec 3000heures d'OCPU et 18000Go-heures par mois ; • 2volumes de blocs de stockage, 200Go au total ; • 10GB de stockage d'objets normal ; • 10GB de stockage d'objets avec un accès peu fréquent ; • 10Go de stockage de niveau archive ; • Resource Manager: Terraform géré ; • 5bastions OCI.

• Base de données
• Oracle Autonomous Transaction Processing, Autonomous Data Warehouse, la base de données JSON autonome ou APEX Application Development, vous avez le choix. Deux bases de données au total, chacune avec 1 OCPU et 20Go de stockage ; • Base de données SQL avec 133millions de lectures par mois, 133millions d'écritures par mois, 25Go de stockage par table, jusqu'à 3 tables.

Observabilité et Gestion
• Surveillance: 500millions de points de données d'ingestion, 1milliard de points de données de récupération ; • Application Performance Monitoring : 1000événements de suivi par heure ; • Logging: 10 Go par mois ; • Notifications: 1million d'envois via HTTPS par mois, 1000envois par e-mail par mois ; • Service Connector Hub: 2connecteurs de service.

Services supplémentaires

- Équilibreur de charge flexible: 1 instance, 10Mbps ;
- Flexible Network Load Balancer ;
- Transfert de données sortant: 10To par mois ;
- Réseaux cloud virtuels (VCN): 2réseaux cloud virtuels au maximum, prenant en charge l'IPv4 et l'Ipv6 ;
- Journaux de flux VCN: jusqu'à 10Go par mois partagés entre les services de OCI Logging
- VPN de site à site : 50 connexions IPSec ;
- Content Management Starter Edition : 5000ressources par mois ;
- Certificates : 5certificats CA privés et 150certificats TLS privés ;
- Livraison d'e-mails: 100envois par jour.

VIII. Contraintes

Lecteur cible de ce paragraphe : toute partie prenante

VIII.A. Contraintes de conception

Les contraintes liées à la conception de cette solution sont principalement :

- Les **contraintes de délais** : les délais relatifs aux corrections de chaque collaborateurs doivent être drastiquement diminués, par rapport au système initial, afin de réduire au maximum les temps d'attente, en asynchrone, de chaque production collaborative.

- Les **commentaires des collaborateurs** : chaque collaborateur apportant des annotations relatives à une production devra pouvoir identifer son auteur, la date de rédaction, le contexte rédactionnel et être conservée.

- Les **caractéristique des périphériques** : de par l'hétérogénéité des périphériqes utilisés par les collaborateurs pour rédiger chaque production documentaire, leur type ou marque ne devra en aucun être un frein au travail collaboratif général.

- Le **versionning de document** : cette contrainte englobe celle relative aux commentaires des collaborateurs. En effet, le versionning de document va devoir s'appliquer à tout élément rédactionnel, peu importe sa granularité, du livre, en passant pas l'article, et en descendant jusqu'au commentaires spécifiques à une certaine partie.

- le **suivi des commentaires** : cette contrainte est également liée à celle relative aux commmentaires des collaborateurs, en y spécifiant, à l'aide de fil de discussion par exemple, certains éléments de métadonnées associés audit commentaire.

- les **flux de travail** : l'organisation des productions documentaires ainsi que la structuration de celles-ci sera un élément déterminant dans la conception de la solution finale.

- Les **pensées finales** : malgré toutes les contraintes énumérées précédemment, les collaborateurs devront s'élever au-dessus d'elles pour créer leur meilleur travail. Ainsi, les contraintes de conception ne devront en aucun cas les encadrer et devront permettre d'en faire plus en moins de temps car ils peuvent aider les collaborateurs à se concentrer et à accomplir plus, en affinant la vision du projet rédactionnel.

VIII.B. Contraintes d’architecture

Ces contraintes d’architecture sont intrinsèquement liées aux principes architecturaux énoncés au sein du document d’évaluation de la conformité et reprendront ainsi :

- La qualité, la nature et le type des ressources documentaires utilisées ;
- le partage de ces données ;
- l’accès à ces données ;
- La qualification du dépositaire de ces données ;
- le vocabulaire et les définitions des termes utilisés pour ces données ;
- la sécurité mise en place pour assurer ces données (protection physique, logique, structure…) ;
- l’indépendance technologique des outils utilisés pour traiter les données ;
- la facilité d’utilisation et la convivialité d’accès aux outils et aux données impliquées.


SuperTechSoft