Document
de définition
d’architecture

Gestion et Maintenance
de sites Web

Auteur(s) et contributeur(s)

Nom & Coordonnées	Qualité & Rôle	Société
Gérald ATTARD	Architecte logicielle	Xxxxxxxxxx

Historique des modifications et des révisions

N° version	Date	Description et circonstance de la modification	Auteur
1.0	dd/mm/ yyyy	Création du document	Gérald ATTARD

Validation

N° version	Nom & Qualité	Date & Signature	Commentaires & Réserves
1.0	Nom Qualité		

Tableau des abréviations

Abr.	Sémantique
BDD	Base De Données
Design Pattern	Modèle de conception
KPI	Key Performance Indicator (trad. *indicateur de performance clé*)
MVC	Modèle-Vue-Contrôleur
MVP	Modèle-Vue-Présentation
ORB	Obkect Request Broker (trad. *courtier de requêtes objet*)
SGBDR	Système de Gestion de Base De Données Relationnel
UI	User Interface (trad. *interface utilisateur*)

Table des matières

I. Portée

WebStreet est une agence Web spécialisée dans la création rapide de sites Web pour les clients en utilisant des modèles catégorisés selon plusieurs secteurs d’activités, ayant chacun des styles différents.

Les équipes en développement ont été submergées par l'augmentation des ventes des services de WebStreet à des clients du monde entier.

Ainsi, après ce retour d’expérience et plusieurs années de développement, une nouvelle architecture de production a émergé, embarcant un environnement logiciel complexe ne s'adaptant clairement pas à l'entreprise, voire même, entravant sa croissance.

En parallèle, il a été conclu, suite au retour d’expérience, que les temps de développement doivent être réduits en mettant à profit de nouveaux outils et techniques. Ces enrichiseements devront permettre aux développeurs de livrer, plus rapidement et à moindre coût, en utilisant des blocs de construction et des modèles pour les sites Web ; ces composants logiciels étant appelés atomes de site.

Cet objectif ne pourra être atteint que grâce à une nouvelle architecture logicielle, évoluant dans un contexte agile et permettant une communication fluide par l’emploi d’un langage commun au sein de WebStreet.

II. Buts, objectifs et contraintes

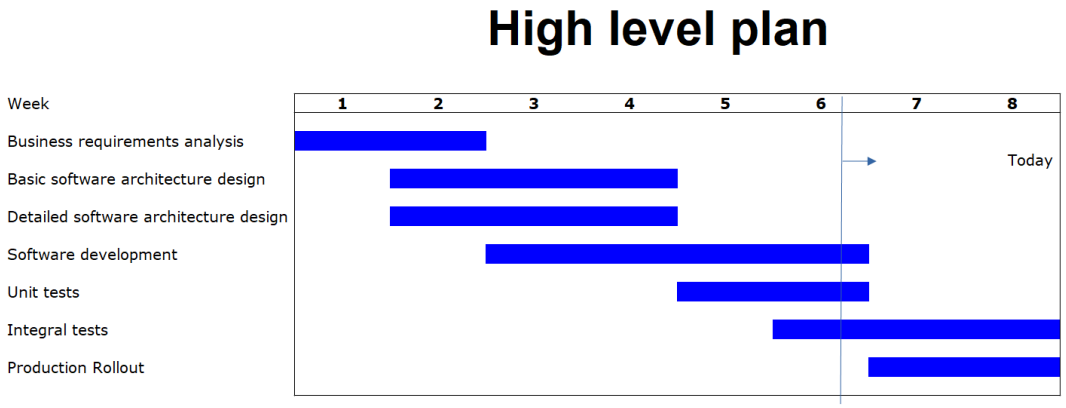
Le but à atteindre est de réduire de moitié les temps de cycle : par exemple, le délai actuel de livraison aux clients étant de 72 heures, celui-ci devra être raccourci à 36 heures. Ici, le processus de livraison est basé sur le principe "*Follow The Sun*", c’est à dire que dès la commande d’un un site Web par un client, l’équipe Principale commence à le développer, peu importe sa localisation :

- si cette dernière termine son travail à la fin de la première journée, elle transfère le livrable de développement à une autre équipe située à 8 à 12 heures de distance.
- s'ils ne terminent pas le premier jour, ils travaillent sur le paquet un jour de plus.

Cette pratique induit que le livrable puisse être transféré, plusieurs fois dans le cycle de développement, jusqu'à ce qu’il soit finalisé. L'équipe Principale approuve alors l'ensemble du cycle à la fin et en informe le client.

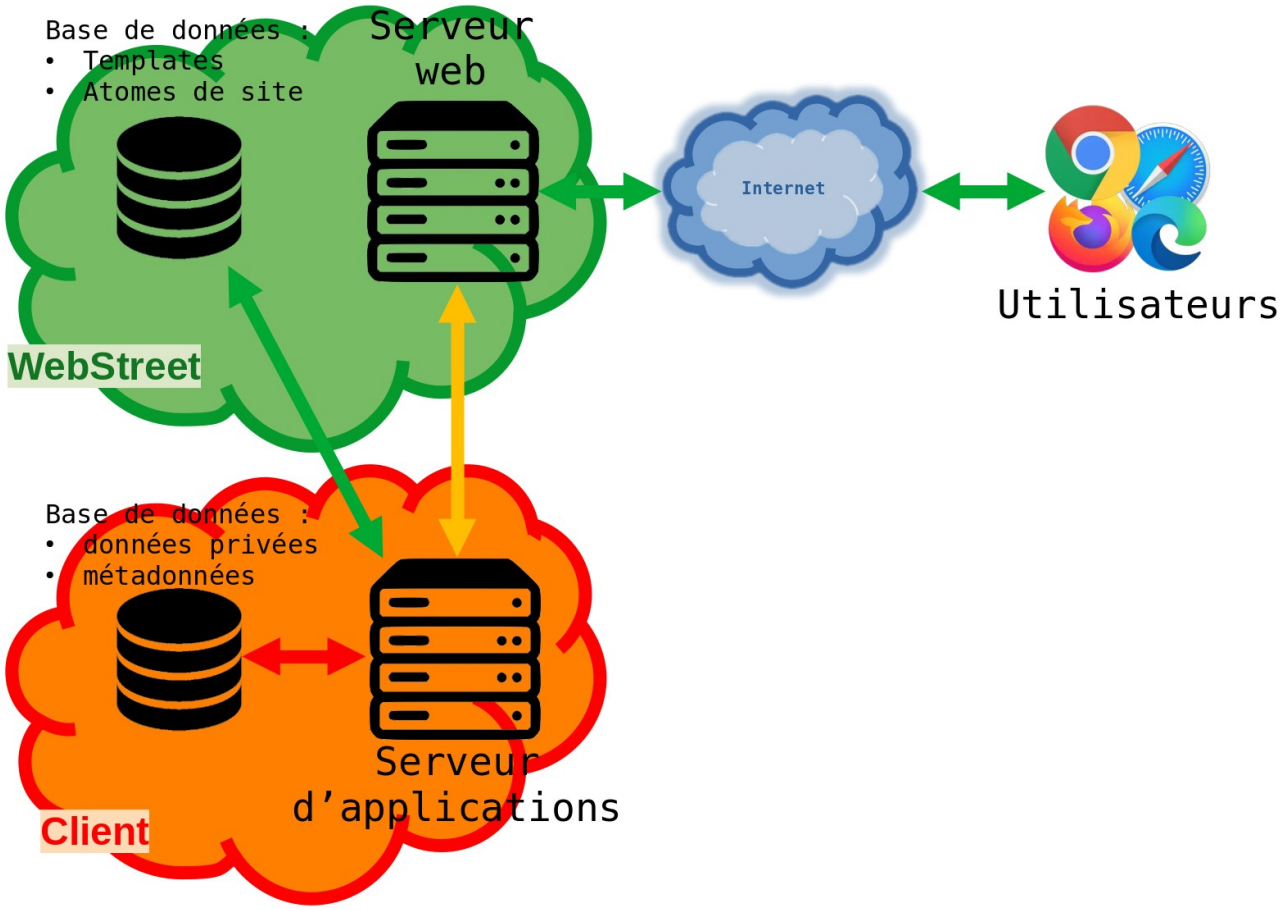
L’objectif final est de diminuer le nombre de transfert de livrables à 3 à 4 fois, tout au plus, au lieu des 6 à 7 fois actuellement.

Pour y parvenir une nouvelle architecture logicielle est nécessaire pour y parvenir.
Ce projet a pour contrainte temporelle 8 semaines de réalisation, selon le diagramme de Gantt ci-dessous :



III. Solution préconisée

La solution décrite ici reprend celle présentée dans le document du cadre d'architecture sur mesure dans le paragraphe §*Hébergement des données sur site Client*, reprise ci-dessous *:*



Cette solution est composée de :

- un **serveur d'applications** dont la fonction est d'agencer les modèles de site et leurs atomes afférents afin de constituer dynamiquement les pages des sites web répondant aux besoins fonctionnels du client.
- Un **serveur web** dont la fonction est de rendre accessible et d'afficher les pages web générées par le serveur d'application.
- Une **base de données hébergée au sein des infrastructures de WebStreet** contenant les données génériques relatives aux modèles de site et aux atomes de site.
- Une **base de données hébergée au sein des infrastructures du client** contenant les données privées/sensibles du client lui-même et leurs méta-données associées.

IV. Principes d’architecture

Les principes édictés dans ce paragraphe définissent les caractéristiques générales de la plateforme commune créée pour WebStreet.

Ces caractéristiques couvrent quatre domaines complémentaires et définissent les bases sur lesquelles la nouvelle solution applicative devra se fonder pour se construire de façon robuste et progressive, à savoir les domaines :

- métiers ;
- applicatifs ;
- données ;
- technologiques.

IV.A. Principes métiers

IV.A.1. Principe A1 : Primauté des principes

IV.A.1.a. Déclaration :

Les principes énoncés ici s'appliquent à tous les membres de WebStreet, que nous appellerons collectivement l'entreprise.

IV.A.1.b. Raisonnement :

La seule façon de fournir aux décideurs un niveau cohérent et mesurable d'informations de qualité est que l’ensemble des collaborateurs de WebStreet respecte ces principes.

IV.A.1.c. Implications :

Sans ce principe, des exclusions, du favoritisme et des incohérences mineraient rapidement la gestion et la pertinence des décisions concernant l’architecture.

Les initiatives ne débuteront pas tant que leur conformité aux principes n'aura pas été examinée. Un conflit avec un principe sera résolu en modifiant le cadre de l'initiative.

IV.A.2. Principe A2 : Conformité aux lois et aux règlements

IV.A.2.a. Déclaration :

Le système d’information, les processus métier et les livrables doivent être conformes à toutes les lois, politiques et réglementations pertinentes.

IV.A.2.b. Raisonnement :

La politique de WebStreet exige le respect des lois, politiques et réglementations. Cela n'exclut pas les améliorations des processus métier qui conduisent à des changements de politiques et de réglementations.

IV.A.2.c. Implications :

WebStreet doit se conformer aux lois, réglementations et politiques externes concernant la collecte, la conservation et la gestion des données, formation et accès aux réglementations. L'efficacité, le besoin et le bon sens ne sont pas les seuls moteurs. Les changements au niveau des lois et des réglementations peuvent entraîner des changements dans nos processus ou applications.

IV.A.3. Principe A3 : Gérer l'information est l'affaire de tous

IV.A.3.a. Déclaration :

Toutes les collaborateurs de WebStreet participent aux décisions de gestion de l'information nécessaires pour atteindre les objectifs d’amélioration des services aux clients.

IV.A.3.b. Raisonnement :

Les utilisateurs de l'information sont les principales parties prenantes dans l'application de la technologie pour répondre à un besoin client.

Afin de s'assurer que la gestion de l'information est alignée sur la politique de WebStreet, toutes les entité de l’entreprise seront impliquées dans tous les aspects de l'environnement de l'information.

Les parties prenantes et le personnel technique responsable du développement et de la maintenance de l'environnement informatique se réuniront en équipe pour définir conjointement les buts et les objectifs de chaque cas d’usage.

IV.A.3.c. Conséquences :

Pour fonctionner en équipe, chaque partie prenante, ou client, devra accepter la responsabilité du développement de l'environnement de l'information.

Un engagement de ressources sera nécessaire pour mettre en œuvre ce principe.

IV.A.4. Principe A4 : Application à usage commun

IV.A.4.a. Déclaration :

Le développement de la plateforme commune, utilisée et développée par et pour WebStreet, est préférable au développement d'applications similaires ou en double qui ne sont fournies qu'à une seule entité particulière de l'entreprise, exception faite en cas de décision spécifique de WebStreet pour répondre à un besoin avéré.

IV.A.4.b. Raisonnement :

La capacité de duplication est coûteuse et prolifère des données contradictoires.

IV.A.4.c. Conséquences :

Les entités consituant WebStreet, prises indépendamment, ne devront pas dépendre d'une capacité qui ne sert pas l'ensemble de l'entreprise. Elles devront donc se doter de la capacité de remplacer leur système prioritaire par un système à l'échelle de WebStreet ; cela nécessitera l'établissement et le respect de politiques exigeantes et strictes.

Indépendamment de l'entreprise, les entités ne seront pas autorisées à développer des capacités pour leur propre usage qui sont similaires et/ou dupliquées des capacités des outils à l'échelle de WebStreet. De cette manière, les dépenses de ressources rares pour développer essentiellement la même capacité de manière légèrement différente seront réduites et réparties dans les secteurs présentant à la fois de la valeur-ajoutée et un besoin avéré.

Les données et les informations utilisées pour soutenir la prise de décision d'une seule entité seront normalisées dans une bien plus grande mesure qu'auparavant, puisqu'à partir de l'application de ce principe, ces données et informations serviront la prise de décision à l'échelle de l'entreprise.

En effet, les capacités organisationnelles plus petites qui produisaient des données différentes, et qui n'étaient pas partagées entre d'autres entités, seront remplacées par des capacités à l'échelle de WebStreet. L'impulsion pour l'ajout à l'ensemble des capacités à l'échelle de l'entreprise peut bien provenir d'une entité qui aura su démontrer, de manière convaincante, la valeur des données et/ou des informations précédemment produites par sa capacité organisationnelle. Néanmoins, une fois que de telles données et/ou informations auront été qualifiée de Valeur, la capacité résultante fera partie du système à l'échelle de la société, et ces données et/ou informations produites seront partagées au sein de l'ensemble de WebStreet.

IV.A.5. Principe A5 : Responsabilité technologique

IV.A.5.a. Déclaration :

WebStreet est responsable de l'organisation technologique, de la possession et de la mise en œuvre des processus et de l'infrastructure web nécessaires, permettant aux sites web client de répondre aux exigences définies pour les clients eux-mêmes, en matière de fonctionnalité, de niveaux de service, de coût et de délai de mise en oeuvre.

IV.A.5.b. Raisonnement :

WebStreet s’impose d’aligner efficacement les attentes des patients avec les capacités et les coûts afin que les services prodigués soient les plus efficaces possibles : des solutions efficientes et efficaces ont des coûts raisonnables et des avantages évidents.

IV.A.5.c. Conséquences :

Un processus spécifique à chaque service prodigué doit être créé pour spécifier à la fois ses processus de réalisation et son contexte d’utilisation.

Ainsi, chaque fonction technologique nécessaire à un service doit définir des processus pour gérer les attentes du client.

En outre, des modèles de données, d'applications et de technologies doivent être créés pour permettre des solutions web intégrées de qualité, afin de maximiser les résultats web.

IV.A.6. Principe A6 : Protection de la propriété intellectuelle

IV.A.6.a. Déclaration :

La propriété intellectuelle (PI) de WebStreet doit être protégée. Cette protection doit se refléter dans l'architecture technologique, la mise en œuvre et les processus de gouvernance.

IV.A.6.b. Raisonnement :

Une grande partie de la propriété intellectuelle de WebStreet est hébergée dans son propre domaine technologique.

IV.A.6.c. Conséquences :

Bien que la protection des actifs de propriété intellectuelle soit l'affaire de tous, une grande partie de la protection réelle est mise en œuvre dans le domaine technologique - même la confiance dans les processus non technologiques peut être gérée par des processus technologiques (e-mail, notes de service, documentation opératoire, etc.)

Une politique de sécurité, régissant les acteurs humains et technologiques, sera nécessaire pour améliorer considérablement la protection de la propriété intellectuelle. Cette démarche se veut, à la fois, d’éviter les compromis et de réduire les responsabilités des collaborateurs médicaux.

IV.B. Principes applicatifs

IV.B.1. Principe B1 : Indépendance technologique

IV.B.1.a. Déclaration :

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

IV.B.1.b. Raisonnement :

L'indépendance des applications informatiques, par rapport à la technologie sous-jacente, permet à celles-ci d'être développées, mises à niveau et exploitées de la manière la plus rentable et la plus opportune. Dans le cas contraire, une technologie, sujette à une obsolescence continue et à sa dépendance vis-à-vis de fournisseurs extérieurs, deviendra un moteur d'investissement, en lieu et place des services prodigués à un client, tel qu'énoncé au sein du **principe A5 : Responsabilité technologique**.

WebStreet est conscient que chaque décision technique, prise en fonction d'un résultat fonctionnel, rend le client dépendant de cette technologie. Ainsi, l'intention de ce principe est de garantir que le logiciel d'application ne dépend aucunement du matériel et des logiciels de systèmes d'exploitation spécifiques.

IV.B.1.c. Conséquences :

Ce principe nécessitera des normes qui prennent en charge la portabilité.

L'utilisation de modèles de sites et d'atomes de sites devront donc être les plus génériques et portables possibles, tout en permettant une personnalisation maximale de leur contenant et contenu.

Des interfaces de sous-système devront être développées pour permettre aux modèles et atomes de site héritées d'interagir avec les applications et les environnements d'exploitation développés dans le cadre de plateforme du client.

Le middleware devra être adapté et utilisé pour découpler les modèles et atomes de site spécifiques.

IV.B.2. Principe B2 : Facilité d’utilisation

IV.B.2.a. Déclaration :

Les applications informatiques devont être faciles à utiliser.

La technologie sous-jacente est transparente pour les utilisateurs, afin qu'ils puissent se concentrer sur les tâches techniques à accomplir.

IV.B.2.b. Raisonnement :

Plus un personnel technique doit comprendre la technologie sous-jacente, moins il est productif ; la facilité d'utilisation est une incitation positive à l'utilisation d’applications informatiques. Il encourage ces mêmes personnels techniques à travailler dans un environnement d'information intégré au lieu de développer des systèmes isolés pour accomplir leur(s) tâche(s) technique(s) en dehors de l'environnement d'information intégré de WebStreet.

Ainsi, les connaissances informatiques requises pour faire fonctionner un système technique ne devront pas être un frein et/ou un obstacle direct ou indirect aux services prodigués à un client.

La formation informatique devra être réduite au minimum et le risque d'utiliser un système de manière inappropriée devra être faible. L'utilisation d'une application informatique doit être la plus intuitive possible pour n’importe quel personnle technique.

IV.B.2.c. Conséquences :

Les applications devront avoir une « *apparence et une convivialité* » communes et répondre à des exigences ergonomiques strictes, issues d’une charte graphique adaptée.

Par conséquent, la norme d'apparence commune doit être conçue et des critères de test d'utilisabilité doivent être développés en vue d’homogénéiser tous les services offerts par la plateforme de WebStreet.

Les lignes directrices pour les interfaces utilisateur ne doivent pas être limitées par des hypothèses étroites sur le niveau de formation du personnel technique, tel que la langue, la formation aux systèmes technologiques ou la capacité physique...

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

IV.C. Principes de données

IV.C.1. Principe C1 : Données en tant qu’atouts

IV.C.1.a. Déclaration :

Les données sont un actif qui a de la valeur pour l'entreprise et qui est gérée en conséquence.

IV.C.1.b. Raisonnement :

Les données sont une ressource de WebStreet précieuse; ayant une valeur réelle et mesurable. En termes simples, 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, peu importe le contexte d’application.
Les actifs de la société seront gérés avec soin et les données n’y feront pas exception.
Les données sont le fondement de la prise de décision, WebStreet gérera donc soigneusement ses données pour s’assurer de :

- leur positionnement,
- leur authenticité,
- leur exactitude,
- leur disponibilité.

IV.C.1.c. Conséquences :

Ce principe représente un pilier de ceux régissant les données :

- **les données sont un atout,**
- les données sont partagées,
- les données sont facilement accessibles.

En conséquence, WebStreet s’engage à fournir la formation indispensable, nécessaire et suffisante au personnel technique le constituant. Cet engagement de formation a pour objectif de s'assurer que tout le personnel technique de WebStreet comprenne la relation entre la valeur des données, leur partage et leur accessibilité.

Ainsi, en prenant en compte le fait que les données sont un actif de valeur pour l'ensemble de la société, des responsables de la bonne gestion de ces données seront affectés à des postes pertinents pour l’amélioration des services prodigués à un client par le personnel technique de WebStreet. Ces intendants, ou responsable de données, auront l'autorité et les moyens de gérer les données dont ils sont responsables.

En délégant cette responsabilité, WebStreet s’assurera de mettre en place une politique de transition culturelle pour passer d’un mode de pensée de *« propriété des données »* à celle d’une *« intendance des données »*.

Ce rôle de responsable de données est essentiel ; des données obsolètes, incorrectes ou incohérentes pourraient être transmises au personnel technique et affecter négativement les décisions prises de la société. Cela pourrait même avoir des répercutions sur la prise en charge d’un client.

Une partie du rôle du responsable de données est de s'assurer la qualité de celles-ci.

Des procédures seront développées et utilisées pour :

- prévenir et corriger les erreurs dans les informations ;
- améliorer les processus qui produisent des informations erronées.

En outre, la qualité des données devra être mesurée : ces métriques serviront alors à mettre en place des mesures d'amélioration de la qualité de ces données ; il est d'ailleurs probable que des politiques et des procédures seront également élaborées à cet effet.

Enfin, un forum avec une représentation complète, à l'échelle de WebStreet, devra décider des changements de processus suggérés par les délégués syndicaux représentant le personnel technique.

IV.C.2. Principe C2 : Partage des données

IV.C.2.a. Déclaration :

Le personnel technique a accès aux données nécessaires à l'exercice de ses fonctions.
Par conséquent, les données sont partagées entre les fonctions et les équipes de WebStreet, après avoir pris en compte le besoin d'en connaitre.

IV.C.2.b. Raisonnement :

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

Axiome : il est moins coûteux de conserver des données précises et actualisées dans une seule application, puis de les partager, que de conserver des données en double dans plusieurs applications.

Au vu de la quantité de données disponibles au sein de WebStreet, celles-ci seront stockées au sein d'un système de gestion de base unique.

La vitesse de collecte, de création, de transfert et d'assimilation des données sera proportionnelle à la capacité de la société à partager efficacement l'ensemble de ces mêmes données. Celles-ci devront donc être partagées pour traduire la volonté de prendre les meilleures décisions possibles en s'appuyant sur le moins de sources possibles. En effet, la gestion centralisées des données les rendra plus précises et opportunes pour l'ensemble du personnel médical en charge de cette prise de décision.

En outre, 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.

IV.C.2.c. Conséquences :

Ce principe représente un pilier de ceux régissant les données :

- les données sont un atout,
- **les données sont partagées,**
- les données sont facilement accessibles.

En conséquence, WebStreet s’engage à fournir la formation indispensable, nécessaire et suffisante au personnel technique le constituant. Cet engagement de formation a pour objectif de s'assurer que tout le personnel technique de WebStreet comprenne la relation entre la valeur des données, leur partage et leur accessibilité.

Pour permettre le partage des données, WebStreet devra développer et respecter un ensemble commun de politiques, de procédures et de normes régissant la gestion et l'accès aux données techniques à court et à long terme.

À court terme, en prenant en compte l’investissement important des systèmes hérités par chacune des entités et équipes, WebStreet devra investir dans des logiciels capables de migrer ces données du système hérité vers un environnement de données partagé.

WebStreet développera également des modèles de données standard, des atomes de site et d'autres métadonnées définissant l’environnement partagé. De plus, la société réalisera un système de référentiel unique pour stocker ces métadonnées afin de les rendre accessibles.

À long terme, à mesure que les anciens systèmes seront remplacés, WebStreet adoptera et appliquera des politiques et des directives communes d'accès aux données. Ainsi, les développeurs de nouveaux sites web pourront garantir que les données de ces nouvelles applications :

- restent disponibles pour l'environnement partagé,
- continuent à être utilisées par ces nouvelles applications.

À court et à long terme, WebStreet 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 continuellement au principe de sécurité des données, pour lequel, en aucun cas, le principe de partage de données ne devra compromettre des données confidentielles.

Les données mises à disposition pour le partage pourront être utilisées par le personnel technique pour préparer et/ou exécuter les services les plus pertinents et adaptés pour le client.

L’application de ce principe 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 données à l'échelle de l’entreprise.

IV.C.3. Principe C3 : Accessibilité des données

IV.C.3.a. Déclaration :

Les données sont accessibles au personnel technique pour exécuter leurs fonctions.

IV.C.3.b. Raisonnement :

Un large accès aux données conduit à l'efficience et à l'efficacité de la prise de décision technique et permet 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 WebStreet, pour permettre l'accès à une grande variété de spécialités techniques ; le temps du personnel technique est alors économisé et la cohérence des données est améliorée.

IV.C.3.c. Conséquences :

Ce principe représente un pilier de ceux régissant les données :

- les données sont un atout,
- les données sont partagées,
- **les données sont facilement accessibles.**

En conséquence, WebStreet s’engage à fournir la formation indispensable, nécessaire et suffisante au personnel technique le constituant. Cet engagement de formation a pour objectif de s'assurer que tout le personnel technique de l’entreprise comprenne la relation entre la valeur des données, leur partage et leur accessibilité.

Ainsi, l'accessibilité implique la facilité avec laquelle le personnel technique obtient des informations.

La manière dont ces informations sont accessibles et affichées doit être suffisamment adaptable pour répondre à un large éventail de spécialités techniques et leurs méthodes d'accès correspondantes.

L'accès aux données ne constitue pas une compréhension des données : le personnel doit veiller à la bonne interprétation de l’information.

L'accès aux données n'accorde pas nécessairement au personnel technique des droits d'accès pour modifier ou divulguer les données.

Ces concepts de Sécurité nécessiteront un processus de formation et un changement dans la culture organisationnelle. Ainsi, WebStreet définira et mettra en œuvre une politique de transition culturelle pour passer d’un mode de pensée de *« propriété des données »* à celle d’une *« intendance des données »*.

IV.C.4. Principe C4 : Intégrité des données

IV.C.4.a. Déclaration :

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

IV.C.4.b. Raisonnement :

L'un des avantages d'un environnement architecturé est la possibilité de partager des données (par exemple, du texte, de la vidéo, du son, etc.) au sein de l'entrprise.

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

En prenant en compte la perte d'intégrité des données saisies plusieurs fois, le dépositaire de ces données sera seul responsable de la saisie des données, ce qui élimine les efforts humains et les ressources de stockage de données redondants.

Nota: un administrateur de données est différent d'un responsable de données. Un administrateur de données est responsable de l'exactitude et de l'actualité des données, tandis que le responsabilité de données a à sa charge des tâches de normalisation et de définition des données.

IV.C.5. Conséquences :

La tutelle réelle résout les problèmes de "*propriété*" des données et permet aux données d'être disponibles pour répondre aux besoins de toule personnel médical.

Cet axiome implique qu'un changement culturel de la « *propriété* » des données à la « *tutelle* » des données est nécessaire.

Ainsi, le dépositaire des données sera responsable du respect des exigences de qualité imposées sur les données dont il est lui-même est responsable.

De plus, l'administrateur de données doit avoir la capacité de donner confiance au personnel technique dans les données, en affichant, par exemple, l'attribut relatif à la "*source de données*" dont elles sont issues.

WebStreet devra identifier une véritable source des données unique afin que l'autorité responsable des données puisse se voir attribuer cette responsabilité de dépositaire. Cela ne signifie pas que les sources classifiées seront révélées ni que la source en sera forcément le dépositaire.

Les informations doivent être saisies électroniquement une seule fois et immédiatement validées aussi près que possible de la source.

Des mesures de contrôle de la qualité doivent être mises en place pour assurer l'intégrité des données. En raison du partage de données au sein de l'entreprise, l'administrateur de données est comptable et responsable de l'exactitude et de l'actualité de ses éléments de données désignés. Il est de la responsabilité du consortium de reconnaître l'importance de cette responsabilité de tutelle.

IV.C.6. Principe C5 : Vocabulaire commun et définitions de données

IV.C.6.a. Déclaration :

Les données sont définies de manière cohérente au sein de WebStreet ; les définitions sont compréhensibles et disponibles pour tout le personnel technique.

IV.C.6.b. Raisonnement :

Les données qui seront utilisées dans le développement de sites doivent avoir une définition commune à l'ensemble de la société pour permettre le partage des données.

Un vocabulaire commun facilitera les communications et permettra au dialogue d'être efficace.

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

IV.C.6.c. Conséquences :

L’existence de poste intitulé "*administrateur de données*" et de forums avec des chartes impliquant une telle responsabilité ne doit pas être dénué de sémantique.

Un apport d’énergie et de ressources supplémentaires importantes doivent être consacrées à cette seule tâche ; ces efforts visant à améliorer l'environnement de l'information.

Ce principe est, à la fois, distinct et lié à la question de définition des éléments de données, telles que la constitution d’un vocabulaire et de définitions communs.

WebStreet établira le vocabulaire commun initial pour le métier en prenant en compte les spécificités médicales représentées.

En ce qui concerne les définitions, elles devront être utilisées uniformément au sein de l’entreprise.

Chaque fois qu'une nouvelle définition de données est requise, l'effort de définition sera coordonné et réconcilié avec un "*glossaire*", ou dictionnaire, comprenant les descriptions de données impliquées.

Les administrateurs de données de WebStreet assureront cette coordination.

Les ambiguïtés résultant de multiples définitions paroissiales des données céderont la place à des définitions et à une compréhension acceptées à l'échelle de la société.

Plusieurs initiatives de normalisation des données doivent être coordonnées.

Des responsabilités fonctionnelles en matière d'administration des données seront attribuées.

IV.C.7. Principe C6 : Sécurité des données

IV.C.7.a. Déclaration :

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

Outre les aspects traditionnels de la classification de sécurité nationale, cela inclut, sans s'y limiter, la protection des informations :

- pré-décisionnelles,
- sensibles,
- sensibles à la sélection des sources,
- exclusives.

IV.C.7.b. Raisonnement :

Le partage ouvert d'informations et la diffusion d'informations techniques, via la législation pertinente, doivent 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, tout en permettant un accès libre et ouvert : le client a accès aux informations qui le concerne. Les informations pré-décisionnelles (travail en cours, dont la diffusion n'est pas encore autorisée) doivent être protégées pour éviter les spéculations injustifiées, les interprétations erronées, les utilisations inappropriées ou toute autre sorte de tracas pour le client.

IV.C.7.c. Conséquences :

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

Le client et/ou le personnel technique doivent déterminer si l'agrégation entraîne une augmentation du niveau de classification. 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.

A l'échelle de WebStreet, il ne conviendra pas de pratiquer le sillotage de données au sein de systèmes séparés contenant différents niveaux de classifications...

En ce qui concerne la séparation des informations dites « *sensibles* »: afin de fournir un accès adéquat à des informations, tout en maintenant un certain niveau de sécurité au sein du système d'information, **ces besoins de sécurité seront identifiés et développés au niveau des données**, et non au niveau de l'application. Des mesures de sécurité des données seront mises en place pour restreindre l'accès au moyen de différents libellés d'affichage, tels que « *afficher uniquement* » ou « *information sensible* ». L'étiquetage de sensibilité pour l'accès aux informations pré-décisionnelles, décisionnelles, classifiées, sensibles ou exclusives doit être déterminé en amont de création de la données. Ainsi, la sécurité sera intégrée aux éléments de données dès le départ et ne saurait être ajouté plus tard.

Les systèmes, les données et les technologies doivent être protégés contre l'accès et la manipulation non autorisés. Les informations de WebStreet doivent être protégées contre toute altération, sabotage, catastrophe ou divulgation accidentelle ou non autorisée.

De nouvelles politiques nécessaires seront instaurées pour gérer la durée de protection des informations pré-décisionnelles et d'autres travaux en cours, en tenant compte de la fraîcheur du contenu et des décisions du patient.

IV.D. Principes technologiques

IV.D.1. Principe D1 : Changement basé sur les exigences

IV.D.1.a. Déclaration :

Ce n'est qu'en réponse aux besoins de WebStreet que des modifications seront apportées aux applications et à la technologie technique associée.

IV.D.1.b. Raisonnement :

Ce principe favorisera une atmosphère où l'environnement de l'information change en réponse aux besoins de l’entrprise, plutôt que de voir celle-ci changer en réponse aux transistions informatiques effectués. Il s'agit donc de s'assurer que l'objectif du support d'information, embarqué au sein du processus technique, est bien la base de tout changement proposé.

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

Un changement de technologie peut fournir une opportunité d'améliorer le processus technique et, par conséquent, de modifier les besoins de WebStreet. Néanmoins, ce principe n’est en rien prioritaire en comparaison du **Principe A5 : Responsabilité technologique.**

IV.D.1.c. Conséquences :

Les changements relatifs à la mise en œuvre des technologies informatiques devront suivre un examen complet de la part de l’ensemble des équipes techniques de WebStreet.

Il n'y aura pas de financement d’amélioration technique ou de développement de système technologique tiers sans un besoin technique avéré et documenté.

Des processus de gestion du changement conformes à ce principe seront élaborés et mis en œuvre à chaque transition ou migration informatique préconisée.

Ce principe peut se heurter au principe du changement réactif. Néanmoins, celui-ci devra être un critère décisionnel pour atteindre la plus grande valeur-ajoutée possible.

WebStreet devra s’assurer que le processus de documentation des exigences n'entrave pas les changements réactifs pour répondre aux besoins médicaux légitimes.

L'objectif de ce principe est de maintenir l'accent sur les besoins technico-fonctionnel, et non sur les besoins technologiques - un changement réactif est également un besoin technique.

IV.D.2. Principe D2 : Gestion réactive du changement

IV.D.2.a. Déclaration :

Les modifications apportées à l'environnement d'information de WebStreet sont mises en œuvre en temps opportun.

IV.D.2.b. Raisonnement :

WebStreet met en œuvre des systèmes d’information adaptés aux différents services prodigués à un client et imposera que ces systèmes d’information, utilisés quotidiennement par des personnels techniques, répondent à leurs besoins.

IV.D.2.c. Conséquences :

Des processus de gestion et de mise en œuvre du changement doivent être développés, afin que l’adaptation à ce changement ne créent pas de manquement vis à vis des services prodigués à un client.

Un personnel technique ressentant un besoin de changement devra communiquer avec un "spécialiste technique" pour faciliter l'explication et la mise en œuvre de ce besoin.

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

L'adoption de ce principe pourrait nécessiter des ressources supplémentaires et pourrait entrer en conflit avec d'autres principes. Il conviendra donc au personnel technique d’étayer ce besoin et à WebStreet de décider quant à sa prise en compte.

IV.D.3. Principe D3 : Maîtrise de la diversité technique

IV.D.3.a. Déclaration :

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

IV.D.3.b. Raisonnement :

Il existe un coût réel, 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.

A cet état de fait, WebStreet s’impose de limiter le nombre de composants pris en charge pour simplifier leur maintenabilité et en ainsi en réduir les coûts d’exploitation.

Les avantages technique d'une diversité technologique minimale comprennent :

- un conditionnement standard des composants ;
- un impact prévisible de la mise en œuvre ;
- des évaluations et des rendements prévisibles ;
- des tests redéfinis et maintenus à jour afin qu’ils soient toujours les plus pertinents possibles ;
- un statut d'utilité ;
- une flexibilité accrue pour s'adapter aux progrès technologiques.

Une technologie commune à l'ensemble de WebStreet apportera les avantages des économies d'échelle à celui-ci. Les coûts d'administration technique et de support en seront mieux maîtrisés lorsque des ressources limitées seront concentrées sur l’ensemble des technologies communes et partagées.

IV.D.3.c. Conséquences :

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

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

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

La base technologique n'est jamais gelée.

Les avancées technologiques sont toujours les bienvenues et modifieront le modèle technologique lorsque la compatibilité avec l'infrastructure actuelle, l'amélioration de l'efficacité opérationnelle ou une capacité technique requise aura été démontrée.

IV.D.4. Principe D4 : Interopérabilité

IV.D.4.a. Déclaration :

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

IV.D.4.b. Raisonnement :

Les normes aident à assurer la cohérence, améliorant ainsi la capacité à gérer les systèmes et à améliorer la satisfaction des clients, et à protéger les investissements technologiques existants, maximisant ainsi le retour sur investissement et réduisant les coûts. Les normes d'interopérabilité aident, en outre, à assurer le support de plusieurs fournisseurs pour leurs produits et facilitent l'intégration de la chaîne d'approvisionnement.

IV.D.4.c. Conséquences :

Les normes d'interopérabilité des systèmes, dont notamment les normes techniques, seront suivies à moins qu'il n'y ait une raison technique impérieuse de la mise en œuvre d’une solution non standard ; par exemple, ceci pourrait s’avérer dans un contexte au sein duquel un autre principe énoncé dans ce document viendrait imposer des processus non standards impérieux vis à vis des services technique d’un client.

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

Les systèmes technologiques existants doivent être identifiées et documentées.

V. Modèle d’architecture

V.A. Architecture d’entreprise

Pour la définition du modèle d’architecture de WebStreet, cette étude va appliquer les techniques et principes architecturaux de conception et de planification stratégiques dans un contexte d'entreprise/commercial.

Il s'agit d'une pratique stratégiquement définie pour effectuer l'analyse, la conception, la planification et la mise en œuvre de l'entreprise elle-même.

L'architecture d'entreprise de WebStreet visera à :

- fournir ses processus de fonctionnement ;
- réduire au strict minimum les éléments redondants ;
- répondre aux éventuels conflits avec des arguments factuels et objectifs ;
- diminuer la complexité aussi bien fonctionnelle que technique ;
- identifier et qualifier les risques commerciaux.

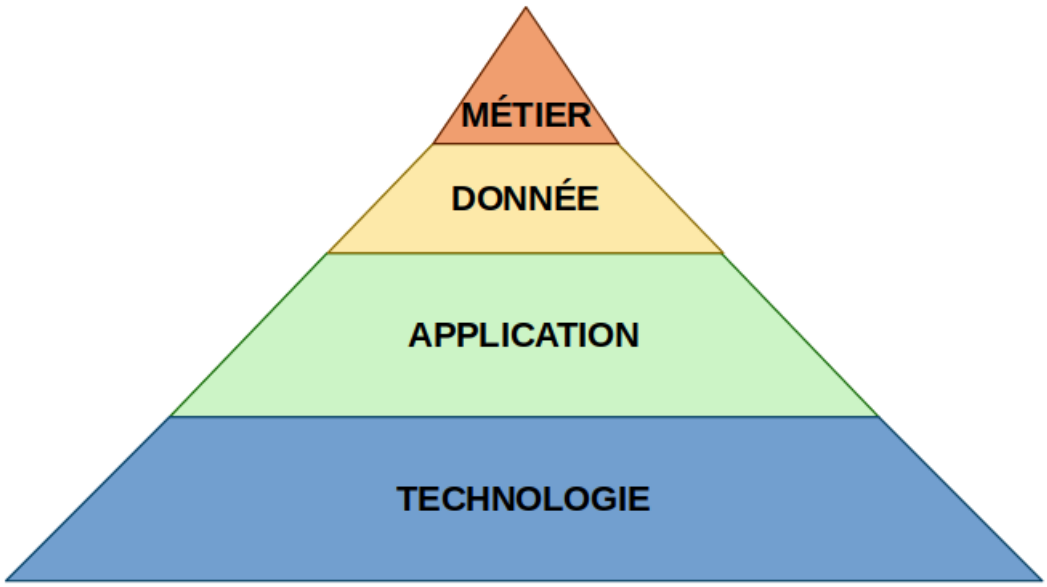
Il s’agira donc de définir des normes entre les services et les équipes pour unifier les efforts et les orienter vers des objectifs communs.

Au travers de l’organisation de WebStreet, celle-ci se verra alors capable de confirmer la stabilité de sa structure et la mise en place de ses processus. L’objectif de telles démarches sera d’aider à tirer des informations et des conclusions à partir des données collectées via l'architecture d'entreprise.

Ce partage des ressources se traduira alors ensuite par une efficacité, une agilité et la diminution du risque de conflit entre les parties prenantes.

La finalité sera de terminer un processus stratégique global de WebStreet pour promouvoir l'efficacité, le partage des ressources sans conflit et la réduction des risques.

Bien que WebStreet présente une structure grandement personnalisée, elle peut être décrite à l'aide quatre couches architecturales principales d'entreprise, tel que schématisées ci-dessous :



V.A.1. Follow the sun

En gestion de projet, en approche Agile ou en modèle d'organisation du digital, le modèle *follow the sun* est une méthode de service et d'assistance conçue pour répondre aux besoins des clients sans tenir compte de l'emplacement géographique ou de l'heure.

Les entreprises, qu'elles soient grandes et petites, sont en mesure de fournir un service 24 heures sur 24 sans faire peser la charge sur un seul site et sans obliger les employés à travailler tard le soir.

L'adoption d'une méthodologie de support *follow the sun* aide à fournir un service et un support de haute qualité et dans les délais lorsque des clients internes et externes en ont besoin.

Cette méthode est également utilisé par les équipes de développement de logiciels qui sont réparties sur des sites internationaux.

Comme il fait toujours jour quelque part dans le monde, *Follow the sun* demeure un moyen d'assister les clients du monde entier, quel que soit le fuseau horaire où ils se trouvent. La méthode suit littéralement le soleil, de sorte que le travail est effectué pendant les heures normales de travail aux quatre coins du monde.

Si les journées de travail des bureaux satellites se chevauchent, le travail peut être transféré d'un bureau qui ferme pour la journée à un bureau qui est encore ouvert ou qui commence tout juste sa journée.

En suivant le soleil, il est alors possible de travailler sur un site web jusqu'à ce qu'il soit terminé. Les délais de réponse sont plus courts, les retards dans la résolution des problèmes sont réduits et les clients sont satisfaits.

V.A.1.a. 4 principes

Follow the sun répond alors à quatre principes complémentaires :

- réduire le temps de réponse pour résoudre plus rapidement les problèmes, et réduire les délais de développement et de production pour mettre plus rapidement les produits et services sur le marché.
- Les sites de service et de développement sont répartis sur de nombreux fuseaux horaires, ce qui permet de transmettre le travail d'un endroit à l'autre.
- À la fin d'une journée de travail de chaque équipe, les employés transfèrent le travail au site suivant qui commence une nouvelle journée sur un autre fuseaux horaires plus à l'ouest (car nous suivons le soleil d'Est en Ouest).
- Grâce à ces transferts, le travail appartient à un seul site à la fois, qui y travaille.

V.A.1.b. Bénéfices et avantages

Cette méthode présente les avantages suivants :

une résolution plus rapide des problèmes ;

des délais de mise sur le marché plus rapides ;

des niveaux de service cohérents : ce modèle garantit qu'aucun des sites de service n'est négligé ou ne bénéficie d'un traitement préférentiel.

Des horaires de travail raisonnables : la plupart des employés ne veulent pas être au travail 12 ou 16 heures par jour. Le modèle *follow the sun* permet aux employés de retrouver leur vie et de consacrer plus de temps à leur famille et à leurs loisirs. En travaillant aux heures normales de la journée, les employés restent heureux, en bonne santé et motivés pour travailler. WebStreet reste alors en mesure d'offrir un service 24 heures sur 24.

V.B. Modèles d’architecture de données

L’architecture de données dans son **état FINAL** sera prise en charge par un **Système de Gestion de Base de Données Relationnelle** (SGBDR). Selon ce modèle, les données sont placées dans des tables, avec lignes et colonnes, et n'importe quelle donnée contenue dans la **base de données** peut être retrouvée à l'aide du **nom de la table**, du **nom de la colonne** et de la **clé primaire**.

Le modèle relationnel est destiné à assurer l'indépendance des données et à offrir les moyens de contrôler la cohérence et d'éviter la redondance.

Il permet de manipuler les données comme des ensembles. Les règles de cohérence qui s'appliquent alors aux bases de données relationnelles sont l'absence de redondance ou de *nul* des *clés primaires*, et l'intégrité référentielle.

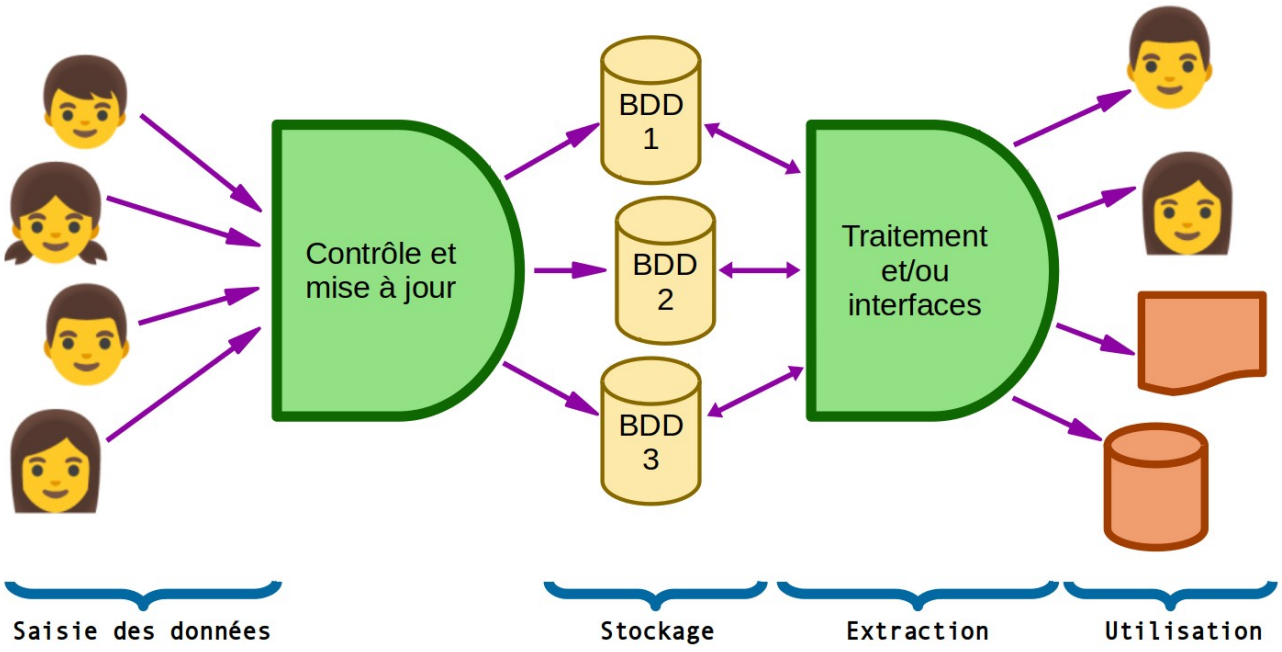
Les avantages d'un système de gestion de base de données relationnelle offrent une vue systématique des données, ce qui aidera WebStreet à faire évoluer ses processus décisionnels actuels, en améliorant différents domaines.

Ainsi, l’utilisation d’un SGBDR présente plusieurs avantages :

- **amélioration de la sécurité des données :** les fonctions d'autorisation et de contrôle d'accès du système de gestion de base de données relationnelle prennent en charge le cryptage et le décryptage avancés, permettant aux administrateurs de base de données de gérer l'accès aux données stockées. Cela offre des avantages importants en termes de sécurité. De plus, les développeurs peuvent modifier l'accès aux tables de la base de données et même limiter les données disponibles à d'autres. Cela fait des SGBDR une solution de stockage de données idéale pour les structures où une autorité décisionnelle doit contrôler l'accès aux données à la fois pour les collaborateurs et/ou pour les clients ;
- **conservation de la cohérence des données :** il est aisé d'ajouter de nouvelles données ou de modifier des tables existantes dans un SGBDR tout en maintenant la cohérence des données avec le format existant. Ceci est principalement dû au fait qu'un SGBDR est conforme aux normes ACID, décrites plus loin au sein de cette étude ;
- **flexibilité et évolutivité :** un SGBDR offre de grandes options en termes de flexibilité lors de la mise à jour des données car les modifications ne doivent être effectuées qu'une seule fois. Par exemple, la mise à jour des détails dans le tableau principal mettra automatiquement à jour les fichiers pertinents et évitera d'avoir à changer plusieurs fichiers un par un. De plus, chaque table peut être modifiée indépendamment sans déranger les autres. Cela rend les bases de données relationnelles évolutives pour des volumes de données croissants ;
- **maintenance facile** : l’utilisation d’une base de données relationnelle permettra aux collaborateurs de rapidement tester, réguler, corriger et sauvegarder les données car l'outil d'automatisation du SGBDR aide à systématiser ces tâches. L’ensemble de ces tâches étant automatisés, la maintenance en sera grandement facilitée ;

- **taux de risque d'erreur réduit :** au sein de WebStreet, le responsable informatique utilisant un logiciel de base de données relationnelle pourra facilement vérifier les erreurs par rapport aux données de différents enregistrements. De plus, comme chaque élément de données est stocké à un seul et unique emplacement, il n'y a aucune possibilité que les anciennes versions brouillent l'image ;
- **traçabilité** : le lignage des données (*data lineage* en anglais) est un processus permettant de créer une sorte de cartographie pour connaître l'origine et les étapes suivies par une donnée, ainsi que la manière et la raison de l'évolution de cette dernière au fil du temps. Cette traçabilité est documentée en répertoriant la source et la destination finale d'une information ainsi que toutes les transformations qu'elle a subi à chaque étape de son parcours dans l'entreprise. Ce processus simplifie le suivi opérationnel de la gestion des données quotidienne et facilite la résolution des erreurs liées aux datas.

Ainsi, au sein de WebStreet, l'architecture de données devra au final répondre au schéma de fonctionnement ci-dessous :



V.C. Modèle d’architecture applicatif

Le modèle d’architecture utilisé sera constituée de trois niveaux, et ainsi appelée architecture trois tiers. Elle est la résultante de l'application du modèle plus général qu'est le multi-tiers. L'architecture logique du système est ainsi divisée en trois niveaux ou couches :

- couche de présentation ;
- couche de traitement ;
- couche d'accès aux données.

C'est une architecture basée sur un environnement de type client–serveur qui est conçue sur un modèle logique d'architecture applicative, visant à modéliser une application comme un empilement de trois couches logicielles (ou niveaux, étages, tiers) dont le rôle est clairement défini :

- la **présentation des données**, correspondant à l'affichage, la restitution sur le poste de travail, le dialogue avec l'utilisateur ;
- le **traitement métier des données**, correspondant à la mise en œuvre de l'ensemble des règles de gestion et de la logique applicative ;
- l'**accès aux données persistantes,** correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive.

Dans cette approche, les couches communiquent entre elles au travers d'un « *modèle d'échange* », et chacune d'entre elles propose un ensemble de services rendus. Les services d'une couche sont mis à disposition de la couche supérieure. On s'interdit par conséquent qu'une couche invoque les services d'une couche plus basse que la couche immédiatement inférieure ou plus haute que la couche immédiatement supérieure (chaque couche ne communique qu'avec ses voisins immédiats).

Le rôle de chacune des couches et leur interface de communication étant bien définis, les fonctionnalités de chacune d'entre elles peuvent évoluer sans induire de changement dans les autres couches. Cependant, une nouvelle fonctionnalité de l'application peut avoir des répercussions dans plusieurs d'entre elles. Il est donc essentiel de définir un modèle d'échange assez souple, pour permettre une maintenance aisée de l'application.

L'architecture trois tiers a pour objectif de répondre aux préoccupations suivantes :

- l'allègement du poste de travail client, notamment vis-à-vis des architectures classiques client-serveur de données ;
- la prise en compte de l'hétérogénéité des plates-formes (serveurs, clients, langages, etc.) ;
- l'introduction de clients dits « *légers* » (plus liée aux technologies Intranet/HTML qu'à l'architecture trois tiers proprement dite) ;
- l'amélioration de la sécurité des données, en supprimant le lien entre le client et les données. Le serveur a pour tâche, en plus des traitements purement métiers, de vérifier l'intégrité et la validité des données avant de les envoyer dans la couche d'accès aux données ;
- la rupture du lien de propriété exclusive entre application et données. Dans ce modèle, la base de données peut être plus facilement normalisée et intégrée à un entrepôt de données ;
- une meilleure répartition de la charge entre différents serveurs d'applications.

Précédemment, dans les architectures client-serveur classiques, les couches de présentation et de traitement étaient imbriquées, ce qui posait des problèmes à chaque fois que l'on voulait modifier l'interface homme-machine du système.

L'activation à distance (entre la station et le serveur d'applications) des objets et de leurs méthodes (cas d'invocation) peut se faire au travers d'un ORB. Cette architecture ouverte permet également de répartir les objets sur différents serveurs d'applications (soit pour prendre en compte un existant hétérogène, soit pour optimiser la charge).

Il s'agit d'une architecture logique qui se répartit ensuite selon une architecture technique sur différentes machines physiques, bien souvent au nombre de trois, quatre ou plus.

V.D. Modèles d’architecture technologique

En ce qui concerne la réalisation de site devant répondre aux besoins du client, cette étude propose l’utilisation de la technologie ***low-code.*** En effet, cette dernière reprend les processus habituels de développement, débogage et autre revue de code, tout en réduisant le temps et le besoin d’effort humain.

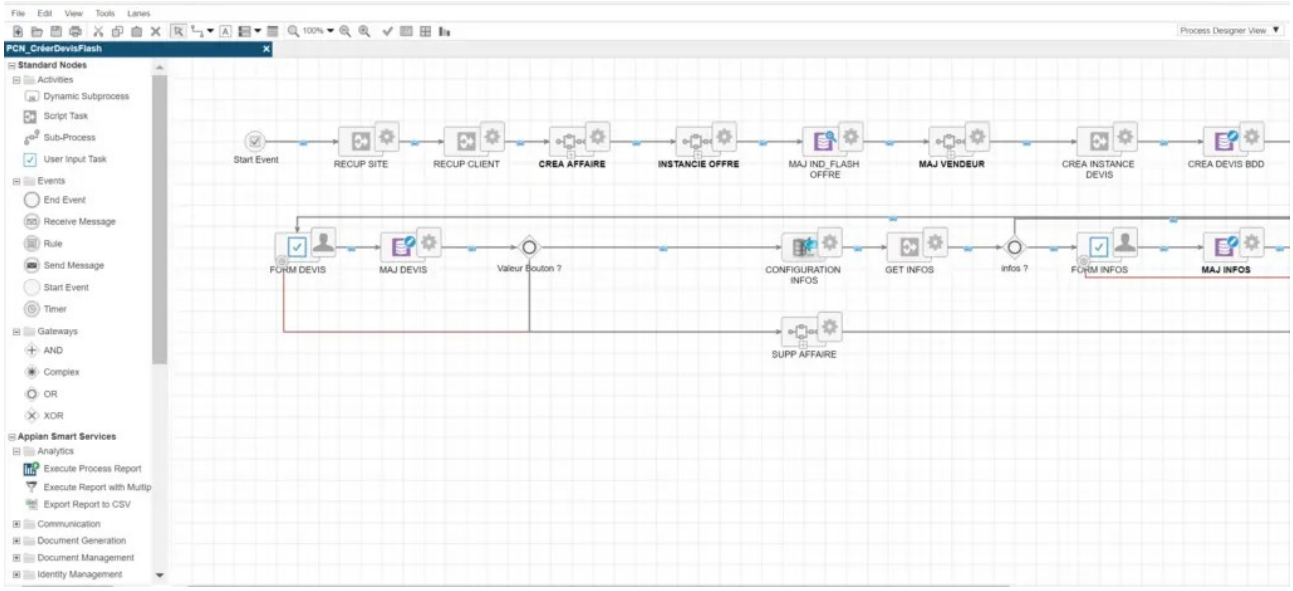
Ce genre de technologie utilise une IHM de réalisation de type WYSIWYG permettant de construire et relier entre eux des blocs d’action, et disposant d’outils de gestion des variables, conditions de déclenchement, etc. Leur objectif est ainsi de réaliser une application en limitant fortement l’utilisation de code. Ce genre de plateformes présentent de nombreuses ruptures et innovations par rapport au schéma de développement classique.

Ce genre de technologie est adapté à la politique d’entreprise de Web Street et présente plusieurs avantages :

- une amélioration de la fiabilité et des performances des outils, qui sont désormais exempts d’erreurs de compilation, de lenteurs ou d’outils de débogage perfectibles.
- La prise en charge des normes modernes est à l’ordre du jour, comme par exemple BPM.
- Enfin, le cloud computing qui tend à s’installer dans une majorité d’entreprises permet aux employés de tester et développer des applications en SaaS et non plus sur leur seule machine, avec des conséquences positives en terme de simplicité et de rapidité.

L’utilisation de cette technologie est synonyme d’un paradygme de développement spécifique. En effet, il n’est ici pas question de pas penser le cycle de vie d’un logiciel de sa conception à sa première version : le suivi, les mises à jour et l’amélioration des applications métiers sont également des points sur lesquels les outils de low-coding se démarquent, la difficulté à la transmission étant considérablement réduite, et l’ajout de nouvelles fonctionnalités ayant un risque bien moindre d’avoir des effets de bord dans le reste du code.

L’exemple d’atelier logiciel ci-dessous donne un exemple de l’utilisation de cette technologie :



VI. Logique et justification de l’approche architecturale

Le fait d’utiliser une architecture constituée de plusieurs couches se justifie à l’aide des exigences rencontrées dans ce projet.

En effet, le fait de découpler les bases de données relève d’une exigence du client lui-même indiquant le souhait de disposer des données privées et confidentielles sur ses propres infrastructures, alors que l’administration du site demeurera au sein des infrzastructures de WebStreet et sera dévolue à ses équipes techniques.

En outre, l’utilisation de technologie low-code est justifiée par les exigences et la politique d’entreprise de WebStreet souhaitant la réalisation, la livraison et la mise en production de site les plus simples et conviviales possibles, à la fois pour les concepteurs que pour le client.

VII. Mappage

VII.A. Mappage aux normes

Lecteur cible de ce paragraphe : partie prenante technique

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

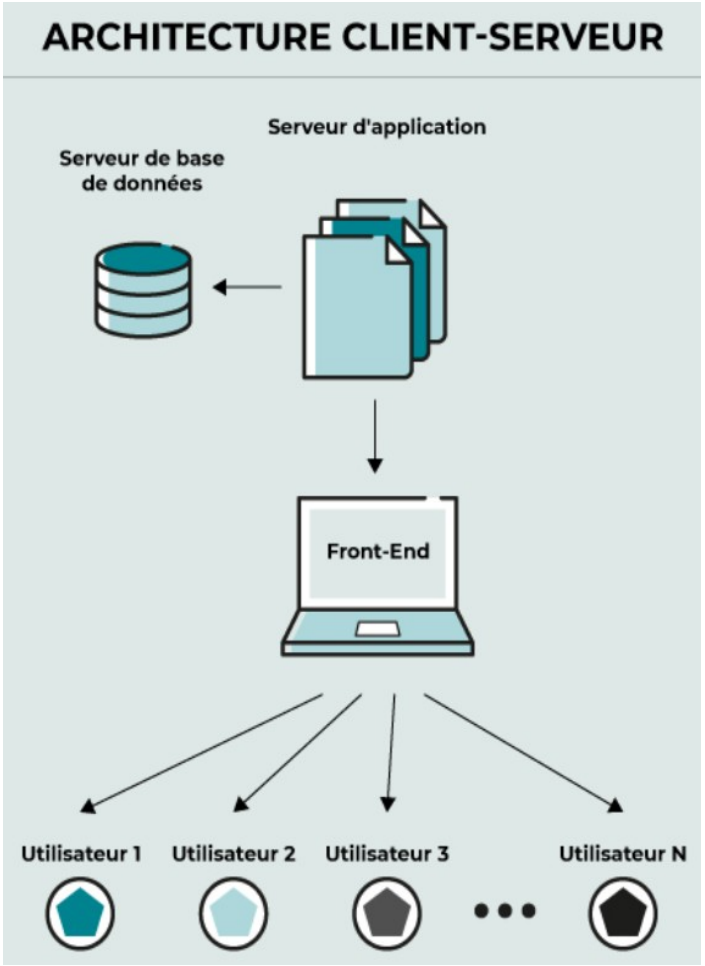
Notion à appliquer au niveau APPLICATIF

VII.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.

VII.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.

VII.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.

VII.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.

VII.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.

VII.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, …])
```

VII.A.2.b. La modification des données

VII.A.2.b.i. 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.

VII.A.2.b.ii. 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.*

VII.A.2.b.iii. 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.

VII.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.

VII.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.

VII.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*.

VII.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.

VII.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.

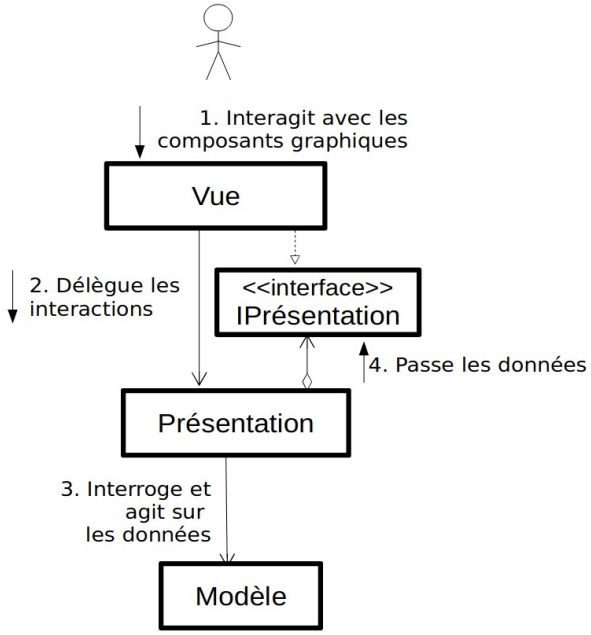
Pour permettre cette durabilité, des « **changelogs »** sont utilisés et pris pour références chaque fois que la BDD est redémarrée.

VII.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.

VII.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.

VII.B. Mappage à la topologie informatique

Le besoin premier de ce projet étant avant tout de réaliser des sites web à l'aide de plusieurs équipes techniques réparties sur des sites géographiques aux fuseaux horaires différents, il va être nécessaire d'appréhender des outils de travail permettant de distribuer et partager cette charge de travail.

Aussi, l'utilisation de cloud n'est alors pas à exclure selon les typologie IaaS, SaaS et PaaS.

VII.B.1. IaaS

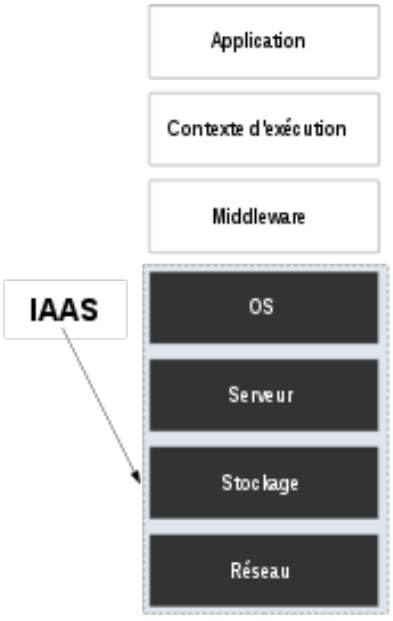
Tel qu'il a été développé dans le document d'évaluation de la conformité puis dans ce document, ce projet sera basé sur une infrastructure de type IaaS.

VII.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.



VII.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.

VII.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.

VII.B.1.d. Inconvénients

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

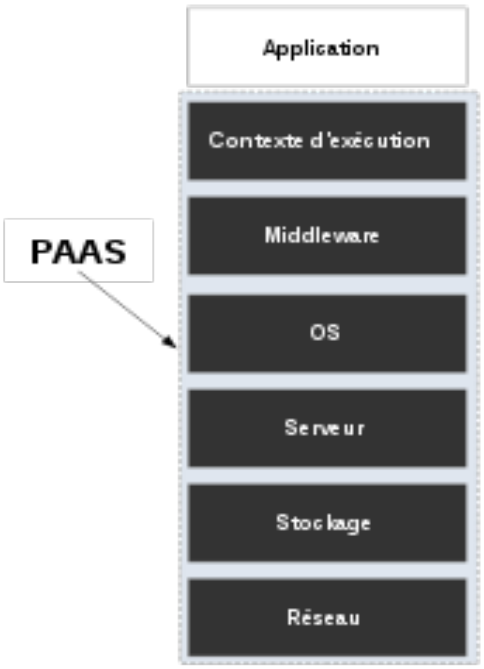
- La dépendance vis-à-vis du fournisseur, dont la seule responsabilité est la disponibilité et la sécurité du service.
- L'accès à Internet est indispensable (les problèmes de connexion à Internet causent aussi des problèmes avec l’environnement IaaS).
- Le changement de fournisseur peut s’avérer très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

VII.B.2. PaaS

VII.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.



VII.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.

VII.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.

VII.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.

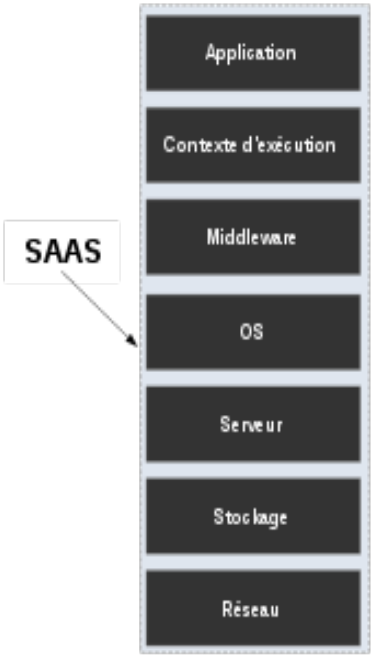
- La **dépendance vis-à-vis des fournisseurs** : les efforts et les ressources nécessaires pour changer de fournisseur de PaaS peuvent rendre les entreprises plus dépendantes de leur fournisseur actuel. Un petit changement dans les processus internes ou l'infrastructure du fournisseur pourrait avoir 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.

VII.B.3. SaaS

VII.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.

VII.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.

VII.B.3.c. Avantages

Grâce à un logiciel SaaS, WebStreet 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.

VII.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.

VII.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.

VII.C.1. IaaS

VII.C.1.a. Fonctionnalités et avantages

L'IaaS offre de nombreux avantages par rapport aux datacenters classiques sur site. Avec l'IaaS, WebStreet 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 WebStreet 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, WebStreet disposera du temps et des ressources nécessaires pour se concentrer sur son activité métier.

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

Ainsi, WebStreet 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.

VII.C.2. PaaS

VII.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.

VII.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.

VII.C.3. SaaS

VII.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.

VII.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.

En outre, il faut garder à l'esprit 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.

VII.D. Mappage à la technologie

VII.D.1. Utilisation du low-code

VII.D.1.a. Définition

Le terme *low-code* signifie en français « Peu de code » ou « Peu de programmation ». Avec une programmation de type *low-code*, le développeur a très peu recours à une programmation classique et manuelle.

Le travail se fait plutôt à partir d'une interface graphique sur laquelle le développeur va utiliser des blocs visuels qui ont été préprogrammés. Le développement d'un logiciel est, de ce fait, grandement facilité.

Le gain de temps est alors conséquent puisqu'il n'est pas nécessaire de programmer chaque élément séparément. Le *low-code* est donc une forme simplifiée du développement logiciel, demandant un minimum de connaissances en programmation.

De plus, comme son nom l'indique, les plateformes *low-code* ne dispensent pas totalement de la programmation manuelle. En effet, environ 80% du temps de codage ne nécessite pas de code; les plateformes ne nécessitant aucun code sont appelées des plateformes *no-code*. Sur de telles plateformes, le temps économisé est encore plus conséquent. Nénamoins, ces plateformes sont moins flexibles que les plateformes low-code, qui elles, permettent des ajustements personnalisés.

VII.D.1.b. Singularité

VII.D.1.b.i. Des méthodes de modélisation graphique

Avec leur interface-utilisateur graphique, les plateformes *low-code* favorisent un travail intuitif, basé sur le principe des blocs. L'utilisateur a accès à des modèles visuels qu'il peut sélectionner, généralement par un *Drag-and-Drop*, pour les insérer où bon lui semble dans son projet personnel. Cette modélisation visuelle permet le développement rapide d'applications Web et mobiles, et entraîne un gain de productivité énorme.

VII.D.1.b.ii. Le caractère réutilisable

Le gain de temps est un élément essentiel dans le développement *low-code*. Ce n'est pas seulement lié à la modélisation visuelle qui remplace la programmation manuelle. Les utilisateurs améliorent leur productivité grâce à l'usage de modèles, de *plugins* et de *widgets*. Ces outils sont très souvent exploités même pendant la phase de production.

A noter que certaines entreprises mettent même à disposition de tous les développeurs leurs propres composants, au moyen d'un magasin en ligne privé ou publiques, véritable mine de ressources.

VII.D.1.b.iii. L'accès par un Cloud

La plupart des plateformes de *low-code* permettent à leurs utilisateurs de gérer et d'administrer leur application dans un Cloud. En cas de changement de Cloud, ou lors de la mise en place d'une nouvelle base de données, il n'y aura alors rien à (re)programmer.

Même les outils visuels utilisés sont basés sur des Clouds, ce qui favorise une mise en œuvre immédiate et une excellente disponibilité de l'application.

VII.D.1.b.iv. Maintenance assurée au-delà de la phase de développement

Le recours à une plateforme *low-code* pour développer une application, bénéficie d'un support qui va au-delà de la phase de développement.

Cela veut donc dire que la mise à disposition et la maintenance de l'application sont comprises dans le prix même de la licence de base.

Il est par ailleurs possible de mettre en pause les projets, de les reporter à plus tard, et de les reprendre en main ultérieurement.

VII.D.1.c. Avantage

De nombreux éléments plaident en faveur du *low-code* dont notamment :

- la **rapidité** : comme la programmation manuelle disparaît quasiment dans le développement *low-code*, les prototypes, voire les applications complètes, sont développés dans un délai réduit. L'efficacité des développeurs professionnels est ainsi améliorée. Ils peuvent ainsi consacrer plus de temps aux fonctions métiers essentielles de l'application au lieu de passer du temps à dénicher des erreurs de code.
- La **simplicité** : la simplicité d'utilisation facilite l'apprentissage avec peu de pré-acquis. Le code-source, généralement rédigé de façon manuelle par les développeurs, est généré de manière automatique en disposant intuitivement des éléments les uns après les autres. Il reste cependant possible de faire des ajustements après coup, de façon manuelle.
- La **réduction des coûts** : la réduction des coûts est étroitement liée au gain de temps. Les blocs visuels disponibles sur les plateformes de *low-code* sont réutilisables, et n'ont pas besoin d'être réinventés à chaque fois. Grâce à la simplicité d'utilisation, l'économie en formations techniques des employés , souvent onéreuses, est conséquente.
- La **flexibilité** : des outils de mise à disposition de l'application facilitent la publication dans un environnement privilégié. La simplicité d'utilisation des plateformes *low-code* favorisent par ailleurs une extrême adaptabilité. Les développeurs peuvent ainsi réagir très rapidement aux éventuelles fluctuations des exigences du marché.
- Une **meilleure qualité** : comme le développement du *low-code* est aussi conçu pour les non-programmeurs, la possibilité d'y inclure des compétences transversales est alors à prendre en compte. Des experts issus de secteurs divers peuvent intervenir en tant qu'intervenant dans le développement, et contribueront à trouver des solutions basées sur une approche créative. Le cloisonnement des compétences est alors grandement diminué et la qualité du produit final amélioré.

VII.D.1.d. Utilisation

Dans le cadre spécifique de WebStreet relatif au développement d'un site Internet, le recours à des plateformes *low-code v*a simplifier le travail du développement. Vu les nombreux atouts de cette méthode de développement, elle est déjà utilisée dans un grand nombre d'industries.

Le *low coding* s'avère particulièrement judicieux dans des processus récurrents. Les applications *low-code* favorisent par exemple une optimisation des processus internes qui ont tendance à engendrer beaucoup de *paperasse*. De telles solutions permettent d'économiser du temps et de l'argent, en termes de gestion du personnel.

Le *low-code* est utilisé non seulement à des fins internes, mais aussi à des fins externes. Dans la fonction publique, dans le eCommerce ou dans l'industrie, le développement d'applications logicielles, orientées clients, pouvant être utilisées très rapidement, est très apprécié. Les programmeurs non-initiés développent ainsi des applications destinées à l'efficacité opérationnelle, tandis que les développeurs professionnels utilisent les plateformes de *low-code* pour créer des prototypes d'applications innovantes. Il est également possible d'améliorer des systèmes hérités ou des anciens systèmes. Il est ainsi possible de modifier une application existante au moyen du *low-code* et de l'ajuster aux nouvelles exigences du marché ou des clients, et la rendre plus facile à utiliser.

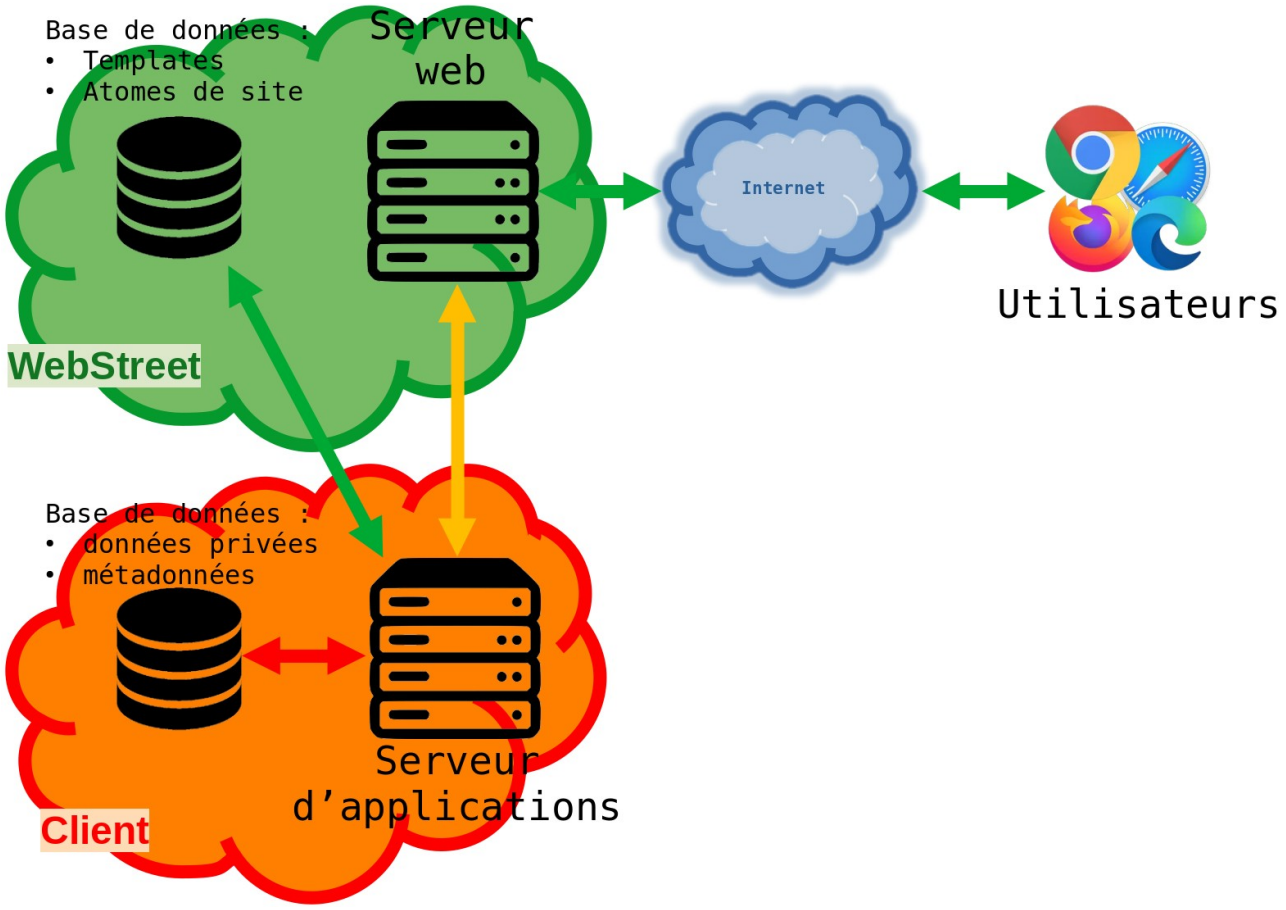
Son utilisation s'est également démocratisée au sein de population defreelances et de blogueurs qui se servent des plateformes de *low-code* pour développer leur propre site Internet. Le plus bel exemple de ce type de projet est certainement le fameux *CMS WordPress*. Pour commencer, les utilisateurs sélectionnent une mise en page prédéfinie pour leur site Internet. Ils ont ensuite l'occasion d'insérer du texte et divers médias. Comme le fournisseur a une approche *OpenSource*, il permet aux utilisateurs plus avancés en programmation d'accéder au code source et de l'adapter en fonction de leurs besoins. *WordPress* est donc une plateforme de *low-code* classique, conçue à la fois pour les débutants, sans compétence en programmation et pour les vrais développeurs.

Le *low-code* facilite donc la programmation de nouveaux logiciels et donne des résultats professionnels sans nécessiter un long processus d'apprentissage. Ce sont là les principaux atouts qui ont permis au développement *low-code* de devenir une méthode de développement intéressante, à la fois en termes de temps et de coûts.

VIII.Evaluation de l’impact

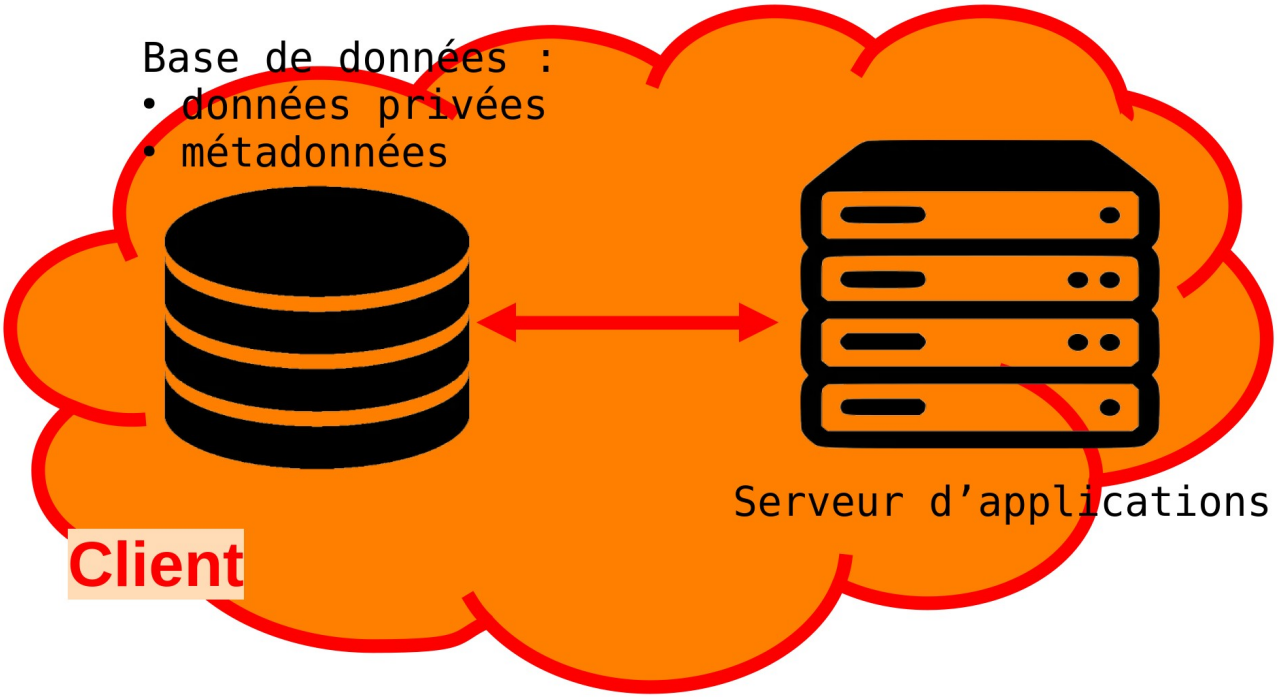
La solution d’architecture présentée dans cette étude tient compte des exigences énoncées, à savoir :

- un composant installé sur le serveur du client lui permettant de récupérer son site web ;
- toutes les données structurelles du site client sont hébergées par les infrastructures de WebStreet ;
- les validations transactionnelles sont envoyées et réalisées par le serveur d’application du client ;
- les mises à jour volumieuses sont contenues au sein de la base de données du client ;
- les mises à jour générales sont centralisées au sein de l’infrastructure de WebStreet via sa base de données associée ;
- toutes les mises à jours, effectuées sur les structures générales de sites, sont envoyées en même temps à tous les clients.



IX. Architecture de transition

L'architecture de transition présentée au sein de cette étude se focalisera sur l'infrastructure du client, issues des dernières exigences énoncées.



Aussi, la transition vers l'architecture finale passera par deux étapes principales :

- **Phase 1** : remplissage de la base de données contenant les données privées du client.
- **Phase 2** : intégration des éléments de personnalisation au sein du serveur d'applications hébergé par le client. Ces éléments de personnalisation modifieront les éléments d'architecture générale (modèles de site et atomes de site), en fonction des besoins du client.


WEBSTREET