

# Création et intégration du Speed layer, sous-composant de l'architecture Lambda

Beebuzziness



Membres du Jury :  
Gwenn Boussard  
Michel Poitevin - Tuteur entreprise  
Francis Brunet-Manquat - Tuteur universitaire

Auteur : Marie Kersalé  
Licence professionnelle Métiers de l'informatique Application Web 2020



## **Déclaration de respect des droits d’auteurs**

Par la présente, je déclare être le seul auteur de ce rapport et assure qu’aucune autre ressource que celles indiquées n’ont été utilisées pour la réalisation de ce travail. Tout emprunt (citation ou référence) littéral ou non à des documents publiés ou inédits est référencé comme tel.

Je suis informé(e) qu’en cas de flagrant délit de fraude, les sanctions prévues dans le règlement des études en cas de fraude aux examens par application du décret 92-657 du 13 juillet 1992 peuvent s’appliquer. Elles seront décidées par la commission disciplinaire de l’UGA.

A Grenoble,

Le 9 janvier 2020,

Signature

KERSALE MARIE

## Remerciements

J'adresse tout d'abord mes remerciements à Pierre Nicodème-Taslé, directeur général et Bastien Guillard, directeur du développement, pour m'avoir accordée leur confiance durant cette année d'alternance et proposée un tuteur hors pair.

Je tiens donc à remercier considérablement mon tuteur, Michel Poitevin, développeur senior, pour avoir mis en place une mission sur mesure en s'adaptant au rythme de l'alternance une semaine de cours/une semaine d'entreprise, rythme qui n'était pas approprié au sprint de 3 semaines mises en place dans l'équipe. Il a su m'accompagner, m'aiguiller ainsi que valoriser mon travail lors de cette mission, un grand merci à lui.

Je remercie également tous les membres de mon équipe pour m'avoir soutenue et guidée tout au long de cette année scolaire grâce à leurs précieux conseils, leurs méthodologies et leur pédagogie: Antoine Begon, Julien Vienne, Clément Salin et Mokhtar Mial.

## Table des matières

Introduction.....	5
I. Beebuzziness, entreprise d'accueil.....	6
I.1 Présentation.....	6
I.2 Activité.....	7
I.3 Organisation.....	9
II. Ma mission.....	12
II.1 Contexte.....	12
II.2 L'architecture Lambda.....	12
II.3 Cadre technique.....	16
III. Réalisation.....	18
III.1 Implémentation.....	18
III.2 Intégration du Service Speed layer.....	24
III.3 Objectifs atteints, résultats obtenus.....	26
IV. Conclusion.....	27
V. Glossaire.....	28
VI. Sitographie.....	30
VII. Annexes.....	31

## Table des figures

Figure 1 : Player	[p7]
Figure 2 : Site officiel Ustream	[p8]
Figure 3 : Hub officiel Ustream	[p8]
Figure 4 : Hub description	[p8]
Figure 5 : Panneau de statistiques	[p9]
Figure 6 : Organigramme des services	[p10]
Figure 7 : Schéma répartitions équipes de développement	[p11]
Figure 8 : Schéma simplifié de l'architecture Lambda	[p14]
Figure 9 : Schéma détaillé de l'architecture Lambda.	[p15]
Figure 10 : Diagramme de séquence d'un évènement	[p25]
Figure 11 : Code implémenté avec le test unitaire	[31]
Figure 12 : Schéma d'architecture du Speed layer service	[p32]
Figure 13 : RabbitMQ	[p33]

# Introduction

---

Étudiante en licence professionnelle Métiers de l'Informatique Application web à l'IUT 2 à Grenoble, je suis en alternance au poste de développeuse web au sein de Beebuzziness, entreprise éditrice de logiciels, dans le service développement.

Le choix d'effectuer cette alternance a été motivé par le fait de pouvoir à la fois valider mes acquis et enrichir mes compétences en développement pour une meilleure adaptation lors de nouveaux projets.

L'entreprise Beebuzziness propose sa propre solution, Ubstream qui est une plateforme permettant d'aider les marques et les propriétaires de contenus digitaux à optimiser leur organisation et leur diffusion sur le web. Le module principal est appelé hub, espace de stockage de documents digitaux, autour duquel s'articulent différents modules de consultation, de partage et d'analyse de médias.

Dans ce rapport nous nous intéresserons à la fonctionnalité d'analyse d'audience permettant de suivre le trafic des médias d'un hub avec des statistiques telles que le temps moyen par page, le nombre de consultations, le partage des réseaux sociaux...

En effet à l'heure du Big Data des problématiques complexes liées à l'exploitation des données amènent les entreprises à choisir la technologie la plus adaptée pour le stockage, l'analyse et l'affichage de volumes importants de données. Beebuzziness a choisi l'architecture Lambda, l'une des trois plus célèbres architectures Big Data avec Kappa et Datalake.

Lambda est utilisé dans mon entreprise depuis un an mais reste incomplet et ma mission consiste à intégrer le dernier composant, le Speed layer, qui va permettre d'affiner les résultats des statistiques grâce au traitement de données temps réel.

Je vais vous présenter dans un premier temps l'entreprise dans laquelle se déroule cette année d'alternance. J'évoquerai ensuite ma mission de manière globale puis ses spécificités techniques, ses objectifs, son découpage ainsi que sa réalisation.

# I. Beebuzziness, entreprise d'accueil

## I.1 Présentation

Beebuzziness est une entreprise éditrice de logiciels, fondée en 2001 par Pierre-Nicodème Taslé. Les équipes de Beebuzziness développent Ustream\*, une plateforme de gestion de contenu en mode SaaS (Software as a Service) Logiciel en tant que Service en Français. C'est un modèle de distribution de logiciels au sein duquel les applications sont disponibles pour ses clients par l'intermédiaire d'internet. Le siège social et les activités commerciales sont implantés à Paris tandis que la production et R&D sont à Grenoble (51 employés).

La société Beebuzziness a été créée en 2001 dans le but de concevoir une technologie de dématérialisation et d'enrichissement de documents. La dématérialisation permet de convertir un document PDF en document virtuel, utilisée dans deux cas majeurs :

- Avec les grandes entreprises ou organisations pour la diffusion de leur communication corporate et aux actionnaires (rapport annuels, plaquettes d'information interne, lettre aux actionnaires...). Exemples de clients du dispositif : SPIE, BPI France, Fayat Énergie Services, CMA de l'Isère.
- Avec le secteur du retail dans lequel les consommateurs peuvent consulter les documents en ligne pour y trouver les promotions courantes sous forme de feuilletable avec des fiches produit et des informations détaillées sur les articles en vente dans les magasins.

La consultation de ces documents est facilitée par des enrichissement tels que des ZZO (zoom image), des liens internes, externes, QR code et animations et disponible dans un player pour une consultation fluide et interactive [Figure 1].



Figure 1: Player - consultation de documents en ligne

En 2016, Beebuzziness élargi ses services au-delà de la consultation et de l'enrichissement de documents digitaux et déploie la plateforme Ustream qui rassemble toutes les fonctions nécessaires à la gestion et à la diffusion de médias. Les médias représentent des documents, des photos, des vidéos et des liens. Je vous invite à aller sur le site officiel d'Ustream [Figure2].



Figure 2: Site officiel d'Ustream



Figure 3: Hub de Beebuzziness

## I.2 Activité

### Le hub

Le hub est un espace de stockage de documents virtuels et la pièce maîtresse d'Ustream autour duquel s'articulent différentes fonctionnalités de consultation, de diffusion et d'analyse d'activités autour des médias. Découvrez le hub de Beebuzziness [Figure 3] et l'image ci-dessous [Figure 4]. On va s'intéresser aujourd'hui à la partie analyse d'audience qui présente le trafic de hubs et de médias.



Figure 4: 1-Bannière / 2 -Menu du hub / 3-Barre d'outils / 4- Arborescence / 5-Espace médiast

## Les statistiques

Les statistiques sont disponibles sur un panneau accessible via le menu d'un hub et permettent d'avoir une vue sur l'activité de vos médias tels que le nombre de visites sur un document, les pages les plus consultées, le temps moyen passé sur une page, le nombre de partage sur les réseaux [Figure 5]. Vous retrouverez le panneau détaillé en [annexe A](#) p38.

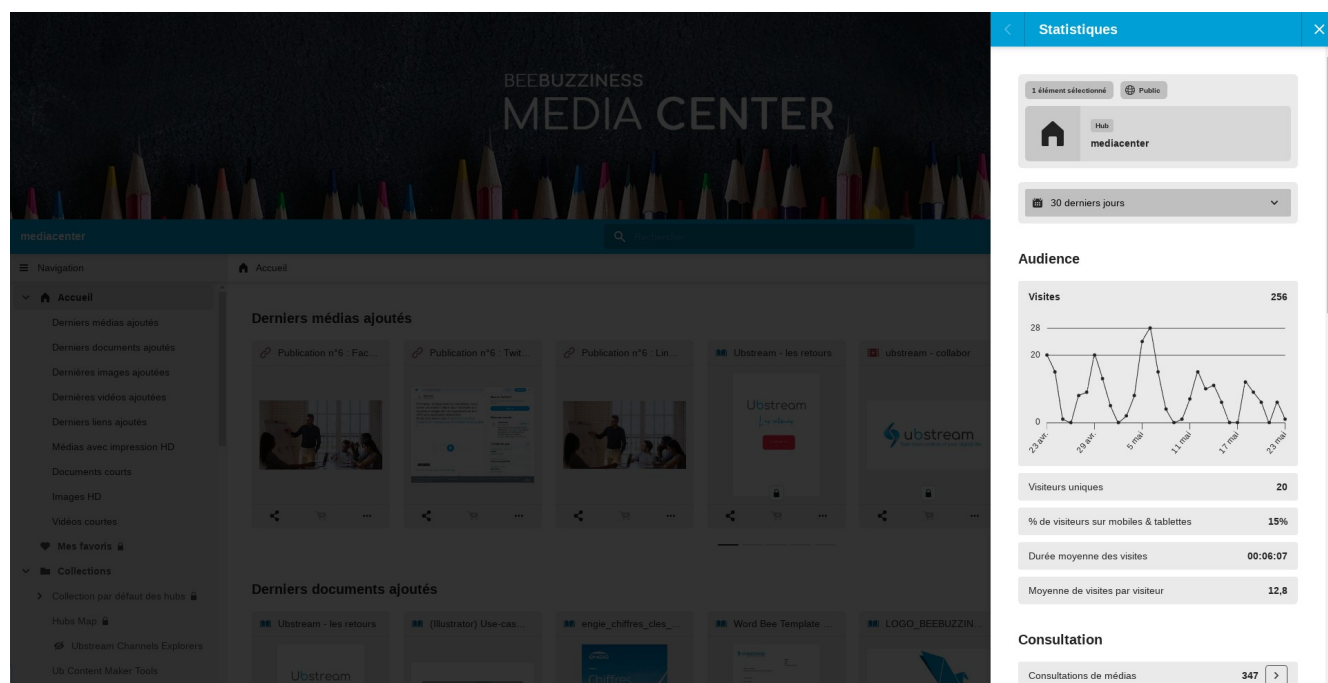


Figure 5: Panneau de statistiques d'un hub



## I.3 Organisation

### Services

Beebuzziness comprend 51 salariés repartis dans 5 services, la production, le marketing, la création, l'administration et le développement [Figure 6].

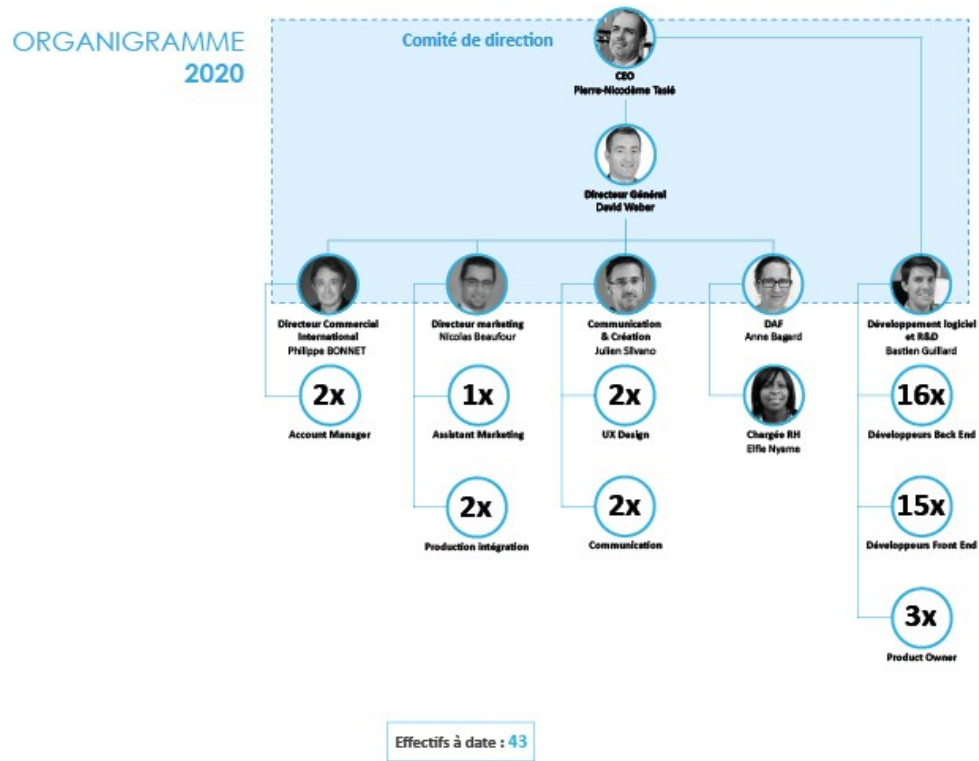


Figure 6: Organigramme des services de Beebuzziness avec la mise en valeur du comité directionnel.

## Equipes de développement

Le pôle de développement axé sur la R&D (Recherche et Développement), regroupe 30 employés, 26 CDI dont 4 externes et 4 alternants. Voici la structure de ces équipes dirigées par Bastien Guillard notre directeur du développement [Figure 7].

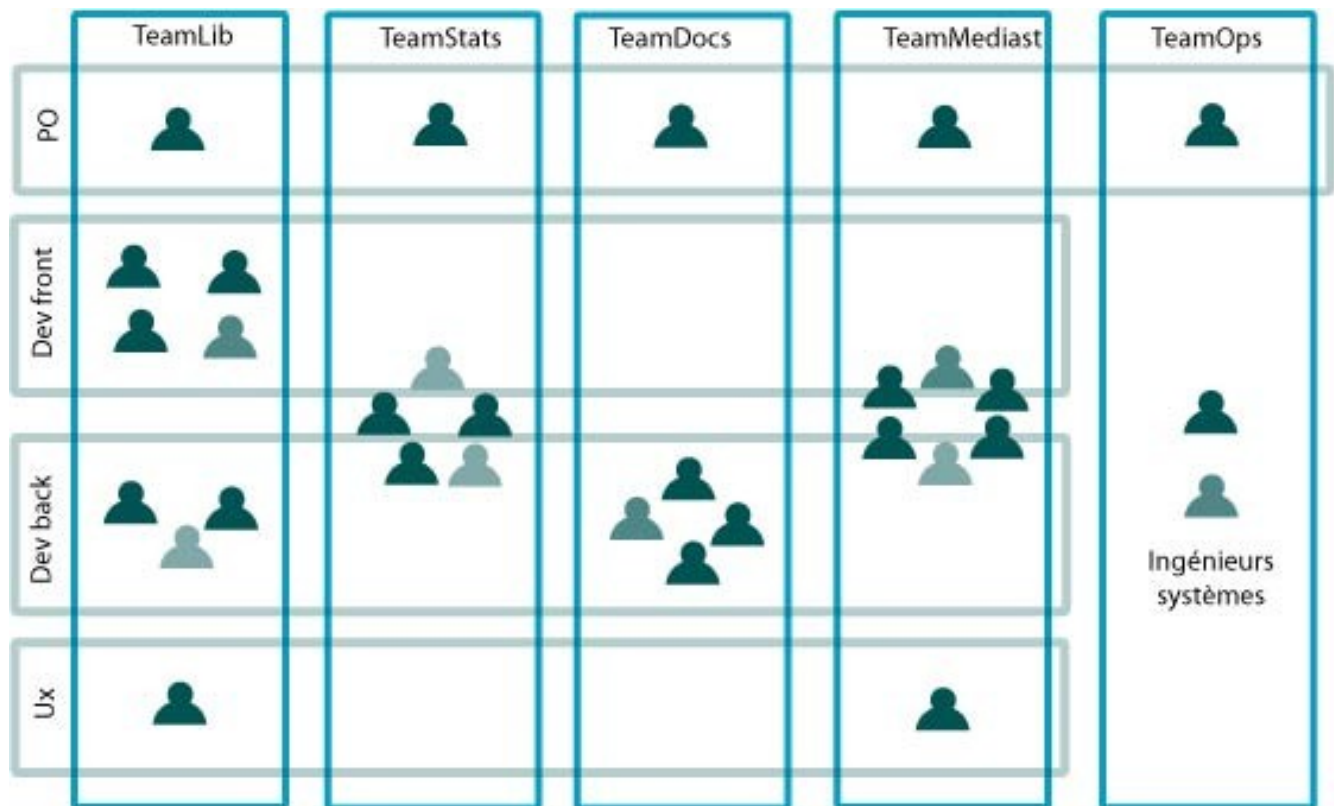


Figure 7: Composition des équipes de développement

La «TeamLib» s’occupe de tout ce qui attrait au hub ainsi qu’au player de documents virtualisés.

La «TeamStats» dans laquelle je me trouve, se charge des modules s’articulant autour d’un hub c’est à dire les statistiques, l’impression à la demande des documents d’un hub et l’Interactive Smart Signage\* permettant la diffusion de contenu interactifs à partir des médias stockés dans un hub.

La «TeamDocs» se charge de l’upload, la virtualisation et le stockage des différents types de documents numériques (PDF, Microsoft Office, Open Office) supportés et de manière plus globale de la scalabilité et de la robustesse de la plateforme Ustream.

La «TeamMedias» traite les sujets d’upload, de virtualisation, de stockage et de download de tous les types de médias hors documents gérés par le hub (images, vidéos, sons), ainsi que les sujets de permissions (accès, modification, suppression,... ) associés aux médias et aux hubs.

La «TeamOps» se charge d’automatiser et de programmer l’infrastructure.

## Méthodes agiles, Scrum

L'organisation de ce service de développement s'appuie sur les méthodes agiles depuis 2015.

Les **méthodes Agile** sont des pratiques de gestion de projet prônant une approche itérative et collaborative, capable de prendre en compte les besoins initiaux du client et ceux liés aux évolutions. Les méthodes Agile se basent sur un cycle de développement qui porte le client au centre. Le client est impliqué dans la réalisation du début à la fin du projet. Grâce à ces méthodes le demandeur obtient une meilleure visibilité de la gestion des travaux qu'avec une méthode classique. L'implication du client dans le processus permet à l'équipe d'obtenir un feedback régulier afin d'appliquer directement les changements nécessaires. Ces méthodes visent à accélérer le développement d'un logiciel. De plus, elle assure la réalisation d'un logiciel fonctionnel tout au long de la durée de sa création.

**Scrum** est l'une des méthodes agile, Ce terme signifie « mêlée » au rugby. La méthode scrum s'appuie sur des « sprints » qui sont des espaces temps assez courts, pouvant aller de quelques heures jusqu'à un mois. Généralement à Beebuzziness un sprint s'étend sur trois semaines. À la fin de chaque sprint, l'équipe présente ce qu'elle a ajouté au module dont elle est en charge devant l'ensemble de l'entreprise. L'équipe clôture le sprint par une rétrospective, bilan du déroulement de cette période qui permet de souligner les points positifs et définir de nouveaux axes d'amélioration.

## II. Ma mission

---

### II.1 Contexte

Ma mission consiste à intervenir sur l'outil d'analyse d'audience fourni aux utilisateurs via le panneau de statistiques d'un hub.

Début 2019, l'équipe a été contrainte de faire évoluer l'ancien système de statistiques et a misé sur le système Lambda. Une première version a été mise en ligne par la suite. Mais celle-ci ne comprenait pas la brique « Speed-layer » responsable du traitement des événements temps réel.

L'objectif de ma mission est l'implémentation de cette brique « speed-layer » dans notre produit et son intégration dans le système global. Elle sera déployée en phase de test à la fin de ma mission et disponible sur tous les hubs ultérieurement après une batterie de tests d'application et l'accord de notre directeur du développement.

Je travaille au sein de l'équipe nommée « TeamStats » qui comprend un product owner, Clément Salin et 5 développeurs fullstack\*, Michel Poitevin, Julien Vienne, Antoine Begon, Mokthar Mial et Rony Dormevil alternant. Un UI Designer\* intervient ponctuellement afin de nous délivrer des specs\*.

Je travaille avec mon tuteur entreprise, Michel Poitevin, sur cette mission depuis le début de mon alternance et nous avons remplis nos objectifs dans les 9 mois qu'a duré la formation.

### II.2 L'architecture Lambda

#### Description générale

Le but de l'architecture Lambda est de fournir des traitements en temps réel sur des volumes importants de données (Big Data). Ce modèle d'architecture propose de faire simultanément du **traitement de type batch** (traitement par lots de données) et du **traitement en temps réel** (de manière continue) permettant d'obtenir une vision complète des données. Le système lambda est composé de 3 couches. [Figure 8]

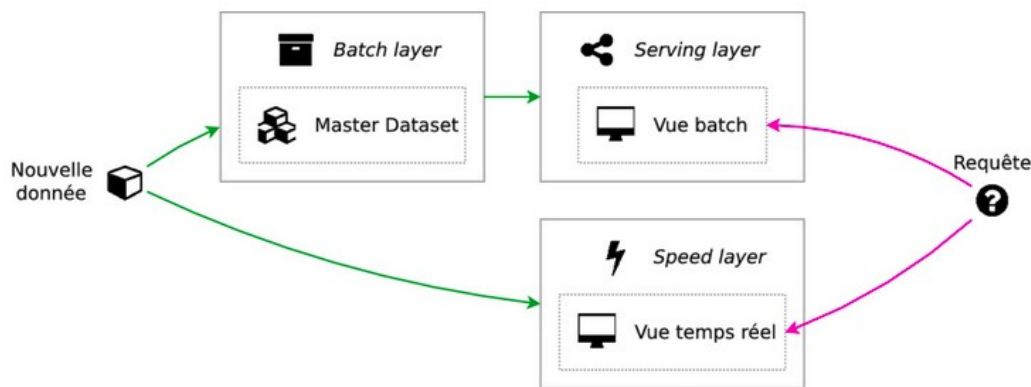


Figure 8: Schéma simplifié de l'architecture Lambda

Le **Batch layer**, qui se charge du traitement par lots, vise une précision parfaite en permettant de traiter toutes les données disponibles lors de la génération de vues. Un traitement par lot ( ou Batch) est lancé toutes les heures (en se basant sur les heures pleines) ou tous les jours (de minuit à minuit à l'heure GMT\*). Ces traitements parcourent le puits de données (ou Event Store), une base de données où nous enregistrons tous les événements générés par l'activité des utilisateurs de manière brute et immuable, pour produire des vues, des agrégations de l'activité des utilisateurs sur une période donnée. Ces vues précalculées sont enregistrées dans une autre base de données (partagée avec le View layer) où elles servent à répondre aux requêtes ad hoc ou comme données sources pour d'autres agrégations. Le Batch layer peut corriger les erreurs en recalculant l'ensemble des données, puis en mettant à jour les vues existantes. Le fait de garder les données dans un format brut permet aucune perte de données et cela assure une plus grande flexibilité. Des calculs peuvent être refaits sans toucher aux données brutes.

Le **Speed layer** ne traite que les données récentes et vient compléter, réguler, les chiffres du Batch layer. Il reçoit les mêmes données que le Batch layer mais calcule au fil de l'eau des agrégations de l'activité de l'utilisateur pour une période récente et limitée. Les données du Speed layer garantissent le temps réel du système, elles sont par définition éphémères et seront oubliées lorsque l'agrégation du Batch layer sera à nouveau calculée. Ainsi, on peut se permettre des approximations ou des erreurs, qui seront temporaires et minimiser le temps de latence de mise à disposition de l'utilisateur final le résultat du calcul des agrégations car il est effectué de manière incrémentale à la réception de chaque événements.

Le **View layer** permet de stocker et d'exposer aux clients/utilisateurs les vues créées par la couche Batch layer dès qu'elles sont disponibles, c'est-à-dire dès que le calcul par la couche Batch layer est complété. C'est de la donnée grise, donnée construite qui peut être reconstruite au besoin. Cette couche renvoie les vues précalculées pour répondre aux requêtes et peut y ajouter les agrégations temporaires générées par le Speed layer pour avoir un résultat au plus proche du moment de la requête. C'est le View layer qui a l'intelligence pour associer les résultats des deux autres couches.

## Avantages

Ce système global va apporter précision et richesse à la donnée collectée et permettre d'améliorer les performances de l'outil de statistiques en séparant le stockage, la consommation et la complexité. Il permet de dépasser les limitations d'une base de données classique : compromis entre la disponibilité et la latence, la robustesse et le partitionnement des données.

Ce système est évolutif puisque chaque niveau est géré de manière indépendante et conçu pour gérer de grands volumes de données.

Il est robuste car il se base sur des clusters de base de données distribuée MongoDB et Elasticsearch qui améliorent notre tolérance aux pannes et notre scalabilité. En cas de panne d'un des serveurs MongoDB ou Elasticsearch, le cluster gère la redondance. Pour se protéger du risque de surcharge d'événements entrant, nous avons installé une file d'attente (queue) basée sur le serveur d'échange RabbitMQ. Cela permet d'encaisser les pics de charge soudain.

Le système est tolérant aux erreurs puisque l'insertion des événements est idempotente ce qui signifie qu'une opération a le même effet qu'on l'applique une ou plusieurs fois, ou encore qu'en la réappliquant on ne modifiera pas le résultat, dans notre cas cela signifie qu'un événement ne sera inséré qu'une seule fois dans le puits de données. C'est la donnée brute qui est insérée. Dans le batch layer, chaque agrégation est idempotente pour une plage horaire (ou bucket) donnée dont la fin est passée.

Ce système est rapide, avec une latence très faible pour l'écriture des événements, l'exécution des requêtes et l'affichage de données même complexes en temps réel grâce à ses différentes couches. Enfin la normalisation des événements permettant de répondre à des besoins futurs avec les données accumulées permet une grande évolutivité.

L'architecture Lambda est indépendante de la technologie car elle précalcule des résultats, les stocke en base, puis interroge cette base pour répondre aux requêtes du demandeur, comme chaque étape est faite par un module indépendant, chacun de ces modules peut être remplacé facilement.

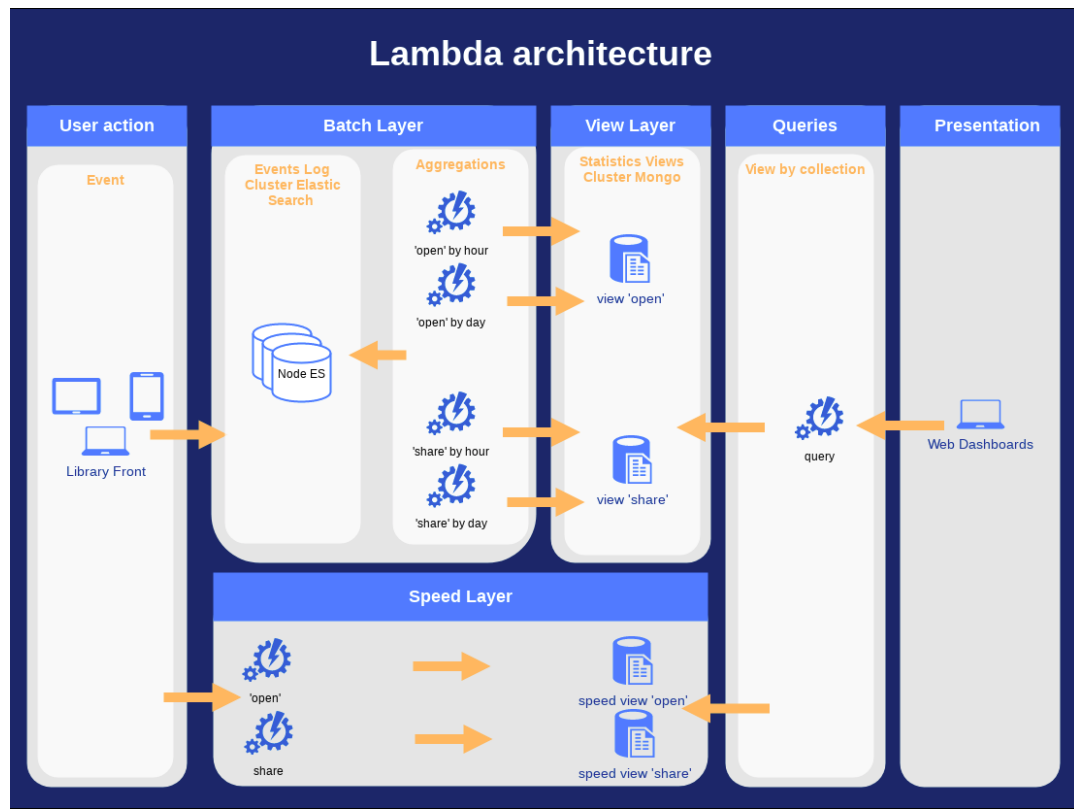


Figure 9: Schéma de l'architecture Lambda mis en place à terme en production

Vous trouverez en annexe B le schéma de l'ancienne organisation des anciennes statistiques et celui de l'architecture Lambda avant l'arrivée du Spee layer.

## II.3 Cadre technique

### Langages

Nodes.js est une plateforme de développement Javascript. C'est le langage Javascript avec des bibliothèques pour interagir avec un operating system. Node propose une solution rapide pour développer des applications back end et coder les deux parties d'une application web dans le même langage. C'est un gain de temps pour le développeur, une flexibilité pour les membres d'une équipe qui peuvent travailler sur le front end et le back end et donc une économie d'argent pour l'entreprise.

#### TypeScript

Langage de programmation libre et open source développé par Microsoft, il améliore et sécurise la production de code Javascript en typant les objets manipulés. Nous utilisons ce langage avec node.js.

Lodash est une bibliothèque JavaScript réunissant des outils bien pratiques pour manipuler des données, là où peuvent manquer des instructions natives :

- pour des tableaux, objets, chaînes de texte (notamment des itérations, du clonage)
- pour tester et manipuler des valeurs
- pour créer des fonctions composites

### Outils

RabbitMQ est un système de file d'attente dans lequel les événements sont stockés temporairement et permet de transporter et router les messages depuis les producteurs vers les consommateurs.

Elasticsearch est une base de données scalable horizontalement, un outil de recherche distribué en temps réel et un outil d'analyse. Une base de données scalable horizontalement permet de rajouter des espaces de stockage sans ré-indexer la donnée existante.

MongoDB est une base de données NoSQL orientée documents ce qui signifie qu'elle stocke les données au format de documents JSON.

Chai est une bibliothèque d'assertions BDD / TDD qui nous permet de vérifier et comparer la valeur de variables ou de propriétés avec la valeur attendue. Différentes possibilités d'écriture, le style assert ou le style BDD décliné en deux colorations: expect et should.

Sinon est la bibliothèque la plus utilisée dans le monde de JavaScript pour générer des espions\*, des bouchons et autres mocks (objets simulés). Sinon.js embarque même un ensemble d'assertions rendant son utilisation encore plus simple.

Mocha est un framework de test JavaScript riche en fonctionnalités exécuté sur node.js et le navigateur, ce qui rend les tests asynchrones simples. Ces tests s'exécutent en série, permettant des rapports flexibles et précis, tout en mappant les exceptions non capturées aux cas de test corrects.



## Découpage prévisionnel

<b>Octobre</b>	Découverte et compréhension de l'architecture Lambda.
<b>Nov-Déc</b>	Écriture des composants de base de ce sous système : aggregation, aggregationManager.
<b>Janvier-Fev</b>	Gestion d'un double buffering sur le Speed layer.
<b>Mars</b>	Création des composants de communication entre le Batch layer et le Speed layer.
<b>Avril</b>	Intégration du Speed layer avec le service de consommation des évènements.
<b>Mai</b>	Le Speed layer est synchronisé avec le Batch layer pour effectuer des calculs sur un double buffer. Les calculs ne sont pas exploités par l'IHM.
<b>Fin Mai</b>	Validation et déploiement en production.
<b>Bonus</b>	Création d'un service externe pour gérer le Speed layer.
<b>Fin mission</b>	L'IHM bénéficie des données calculées par le Speed layer.

---

## III. Réalisation

---

### III.1 Implémentation

Les données du Speed layer garantissent le temps réel du système et le calcul des agrégations se fait de manière incrémentale.

Afin de traiter la donnée en temps réel le Speed layer doit pouvoir agréger des actions utilisateur appelées des évènements qui généreront des statistiques servant à répondre aux requêtes utilisateurs.

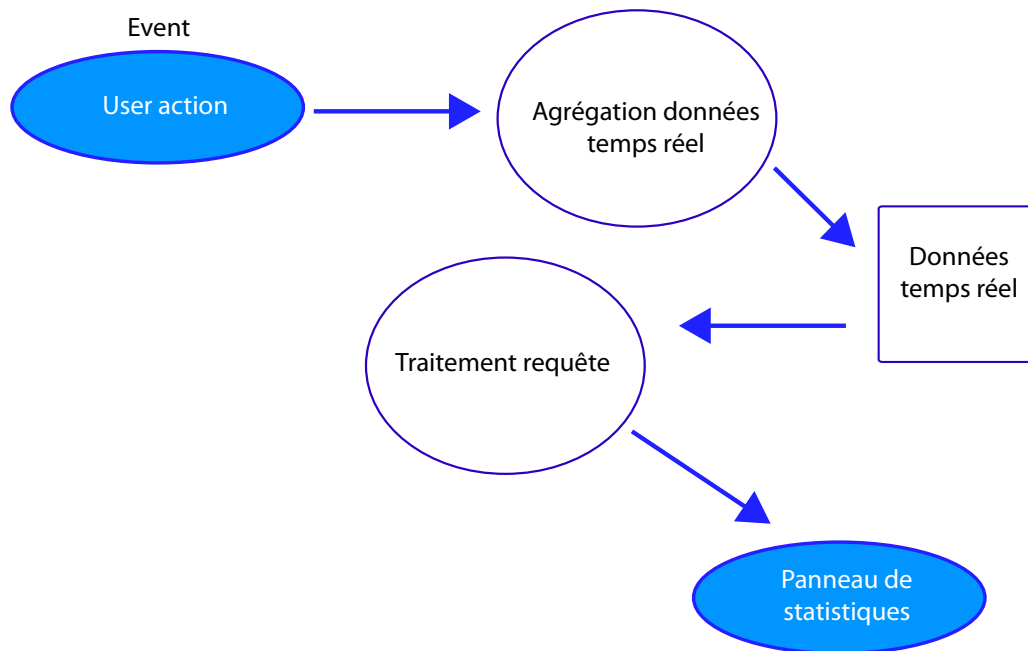


Figure 10: Schéma simplifié du Speed layer

Les responsabilités ont été séparées en différentes classes pour traiter les données du Speed layer.

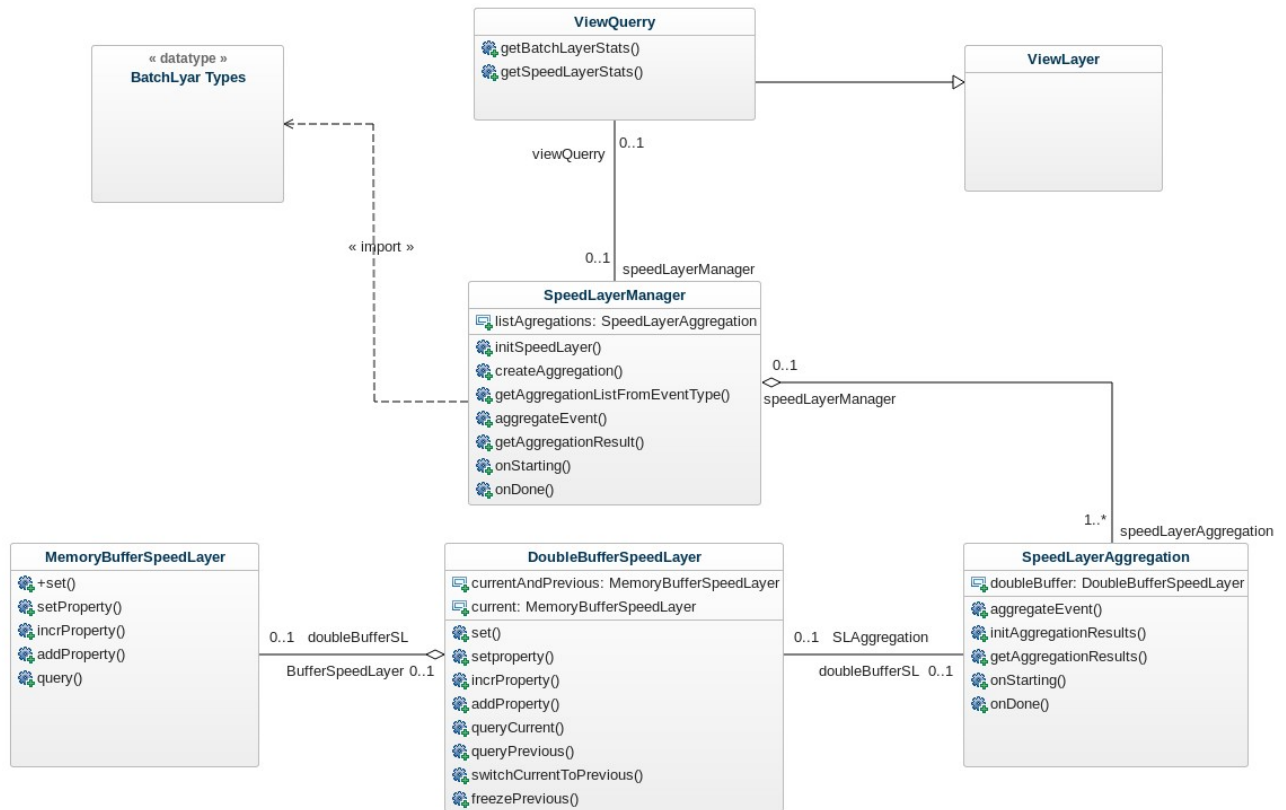


Figure 11: Diagramme de classes du Speed layer

## SpeedLayerManager

A l'arrivée d'un évènement en temps réel, l'objet de type SpeedLayerManager sélectionne les instances de SpeedLayerAggregation concernées par ce type d'évènement et transmet à chaque instance de la classe SpeedLayerAggregation l'évènement correspondant pour qu'il soit agrégé.

Afin de trouver les instances de SpeedLayerAggregation adéquates, on consulte la liste des configurations d'agréations disponibles du Batch layer type, il faut aussi que l'option Speed layer soit activée.

Le manager est également en charge du routage des évènements de synchronisation des calculs de chaque agrégation du Batch Layer vers chaque agrégation du Speed Layer.

Enfin, pour les requêtes le manager récupère le résultat de l'instance agrégée et la transmet au view Query qui renvoie le résultat vers le panneau de suivi des statistiques.

## SpeedLayerAggregation

Dans cette classe s'opère la partie agrégation.

Les évènements sont récupérés avec l'instance créée par le manager afin de les agréger. On agrège libraryDuration en appelant la méthode AggregateEvent() sur l'évènement concerné.

Cette classe gère la synchronisation avec le Batch layer qui signale le début et la fin de son travail avec les fonctions que celui-ci déclenche avec un appel distant (RPC\*) onStart et onDone et peut ainsi dispatcher la donnée agrégée dans le previous ou current buffer. Ces méthodes nous sont très utiles lorsque l'on arrive à la frontière entre la donnée agrégée et la donnée temps réel.

La synchronisation entre le calcul des agrégations dans le Speed layer et les instances du Batch associées s'opère à ce niveau-là.

Pour les requêtes, on récupère les informations stockées et c'est ici que l'on va agréger la donnée en fonction des opérateurs d'agréation. On va donc construire une liste de résultat de valeurs agrégées pour chaque propriété.

## Double buffer

Cette classe gère deux buffers de données, l'un représente l'heure courante et le second l'heure précédente. Elle ne connaît pas le code métier et les concepts d'évènements et d'agréations. C'est une zone d'aiguillage pour le stockage des données vers l'un ou l'autre des buffers. Elle sait router les ajouts de données vers l'un des deux buffers et interroger chaque buffer pour extraire de la donnée.

Pendant toute la durée d'une agrégation la donnée est stockée dans les deux buffers, les opérations se font en double et cela permet de ne pas avoir à merger ultérieurement et d'éviter des bogues lors de la fusion d'une propriété plus complexe.

## Bufferannexe F

Cette classe gère le stockage de la donnée en temps réel par type d'évènement lors de l'arrivée d'un évènement et l'extraction de cette donnée brute pour effectuer le calcul incrémentiel des agrégations

impliquées dans la réponse à une requête. Elle s'occupe donc de compacter de la donnée en réalisant des calculs simples afin d'optimiser l'espace mémoire.

**La notion de Buffer** correspond à la mémoire tampon, zone située dans un disque dur ou dans la mémoire vive. Pour ce projet, le travail s'effectue en mémoire vive. Si le trafic est amené à augmenter fortement, nous envisagerons d'utiliser Redis\* comme stockage afin de partager le buffer entre différentes instances du Speed layer.

Dans notre cas les buffers nous permettent de stocker nos résultats du Speed layer car on ne veut pas les conserver dans le temps, le travail se fait au travers de données éphémères. On travaille au maximum sur deux heures et plus précisément, annexe F sur les deux dernières heures, l'heure précédente et l'heure courante.

Lorsque le calcul commence sur la troisième heure, on supprime l'heure précédente, l'heure en cours devient l'heure précédente et l'heure en cours se réinitialise. Ce concept évite l'encombrement des données. Le fonctionnement du buffer permet de stocker des données sur une courte période et conserver les données avant leur utilisation.

## Les entrées du speed layer

Le Speed layer fonctionne avec 3 entrées appelées également des flots d'exécution :

- Agrégation d'un évènement [Figure 15]
- Synchronisation entre le Batch et le Speed layer. Schéma disponible en [annexe D](#)
- Réponse à une requête. Schéma disponible en [annexe E](#)

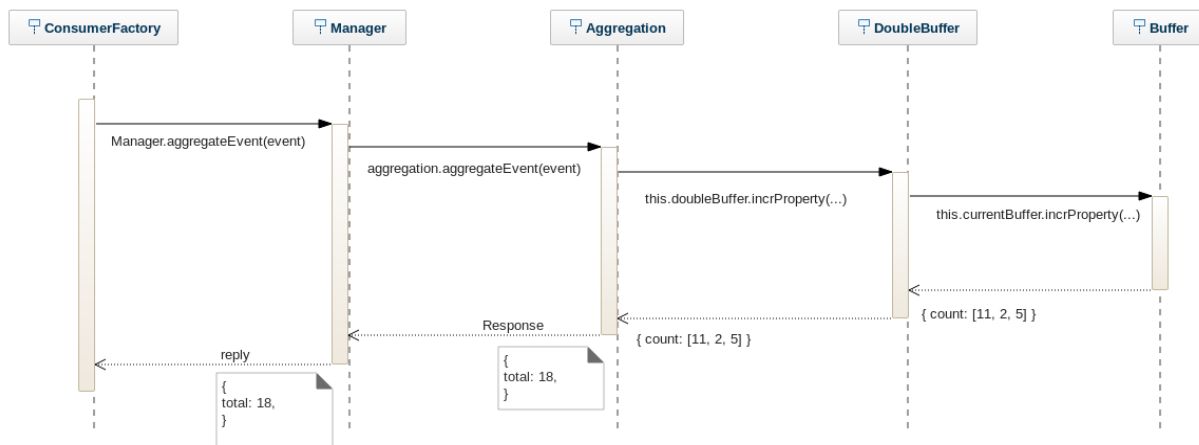


Figure 10: Agréger un évènement

## Mise en situation dans le Speed layer

J'ai implémenté les nouvelles méthodes des classes du Speed layer à partir des tests unitaires créés par mon tuteur. Notamment celle-ci, extraite de la classe `SpeedLayerAggregation`, qui gère l'agrégation d'un évènement.

```
80
81 public aggregateEvent(event: IEvent): IEventAggregation {
82
83     const resultOutput = this.settings.output;
84
85     const keyIncr = _.pick(event, this.settings.eventKey);
86
87     return _.mapValues(resultOutput, (callback: (settingsOneAggregation, key) => {
88         switch (settingsOneAggregation.aggregator) {
89             case Aggregator.count:
90                 const incrResult = this.doubleBuffer.incrProperty(keyIncr, key);
91                 return incrResult[key];
92             case Aggregator.add:
93                 const addResult = this.doubleBuffer.addProperty(keyIncr, key, event[settingsOneAggregation.inputPropertyName]);
94                 return addResult[key];
95             case Aggregator.exists:
96                 const setResult = this.doubleBuffer.setProperty(keyIncr, key, {value: 1});
97                 return setResult[key];
98             case Aggregator.buildProjectionAdd:
99                 const groupByPropertyName = settingsOneAggregation.groupBy;
100                 const groupByProperty = event[groupByPropertyName];
101                 const eventValue = event[settingsOneAggregation.inputPropertyName];
102                 const projectionAddResult = this.doubleBuffer.projectionAddProperty(keyIncr, key, groupByProperty, eventValue);
103                 return projectionAddResult[key];
104             default:
105                 throw new Error(`Invalid aggregator ${key}->${settingsOneAggregation} on event ${event}`);
106         }
107     })};
```

Figure 12: Méthode d'agrégation d'un évènement ( classe `SpeedLayerAggregation`)

Dans cette fonction on crée une variable de « result » à laquelle on affecte un objet vide. On parcourt les sorties des settings avec la méthode lodash « mapValues » qui va regrouper le résultat des clés identiques. On compare ensuite la clé de l'évènement avec l'opérateur d'agrégations pour appliquer la méthode adéquate construite dans le double buffer et retourner l'objet « result ».

### III.2 Intégration du Service Speed layer

Une fois le traitement des évènements et des requêtes terminé avec les quatre classes vues précédemment il nous reste à créer le nouveau service et à le connecter à l'ensemble du système.

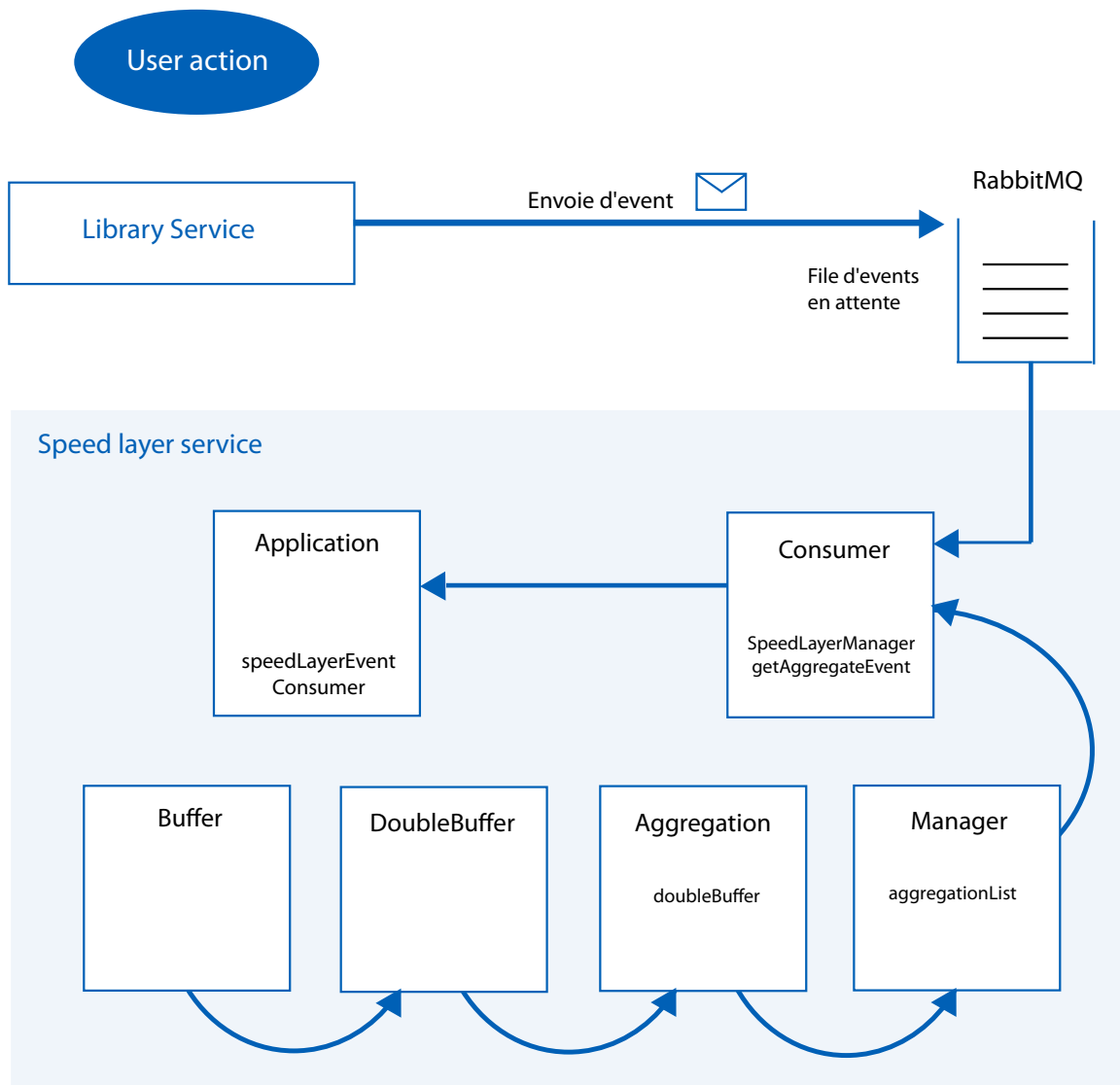


Figure 12: Schéma d'architecture du service Speed layer



En figure 12 une action utilisateur sera envoyée à RabbitMQ qui activera l'application du service Speed layer pour agréger cet évènement.

### Etapes 1

Nous avons créé le service Speed layer qui contient l'application, le consommateur d'évènements et ajouté les classes du Speed layer puis on a relié ce service aux statistiques globales de l'application et aux dockers.

### Etapes 2

Dans cette étape on vient ajouter les nouvelles routes et câbler le Speed layer à RabbitMq.

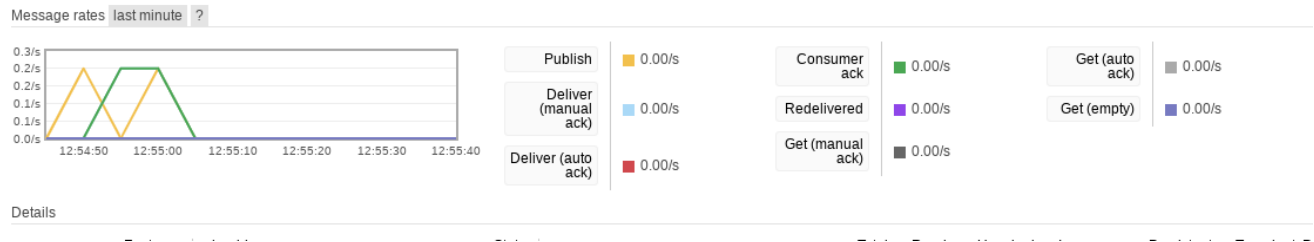


Figure 13: Extrait de RabbitMQ lors de l'ouverture d'un document sur un hub

La partie ascendante de la courbe nous montre la prise en charge de l'évènement par RabbitMQ qui le transmet au Consommateur et la partie descendant représente la prise en charge de l'évènement par le Speed layer.

### Etape 3

On a intégré les 3 entrées qui sont l'agrégation des évènements, la gestion des requêtes et la synchronisation des calculs Batch layer / Speed layer dans le service Speed layer. On a également ajouté l'environnement client/serveur.

### *III.3 Objectifs atteints, résultats obtenus*

#### **Objectifs réalisés**

Le Speed layer est désormais disponible sur nos docker locaux. Cela va nous permettre de réaliser une batterie de tests d'application avant le déploiement sur tous les hubs.

#### **Changements en cours de processus :**

Des changements ont eu lieu au cours de ce projet puisque nous avons travaillé de manière itérative.

Le premier changement est la mise en place des classes buffer et Double buffer avec l'arrivée d'un second bucket. Il était convenu de travailler seulement avec un bucket géré dans la classe Aggregation. Nous nous sommes rendu compte comme le temps d'une agrégation qui est flexible, nous avons un recouvrement à considérer lors de la suppression de données.

Le temps de recouvrement correspond au temps d'une agrégation. Le fait d'ajouter un autre bucket nous a permis de supprimer une partie de la donnée après s'être assuré que le Batch layer ait fini ses calculs. La création des deux nouvelles classes, le buffer et Double buffer permet d'alléger le code de la classe Aggregation pour plus de clarté dans le code et également dans les tests.

L'utilisation de Redis\* se fera plus tard sachant que la donnée est conditionnée au format clé valeur cela nous permettra une mise en place rapide. Pour l'instant la donnée du Speed layer est stockée en mémoire.

La première implémentation du code dans le ViewQuery consistait à récupérer en parallèle les données du Speed layer et du Batch layer puis de les merger. Maintenant on part des données du Speed layer et le Batch layer vient compléter les données déjà emmagasinées par le Speed layer. Ceci nous évite un merge qui pourrait mal se passer et aussi de réduire fortement le code.

## IV. Conclusion

---

Cette immersion dans l'architecture Lambda fût très enrichissante puisqu'elle reposait sur la création d'un nouveau service avec ses classes, son environnement client/serveur et ses interactions avec de nombreux services.

La réalisation et l'intégration du Speed layer, à la fois complexe et très intéressant ont été menées à bien, les objectifs d'intégration de ce sous composant, déployé sur les dockers locaux étant respectés.

Lors de cette mission j'ai bénéficié d'un encadrement technique et humain poussé, cela m'a permis de prendre confiance et d'appréhender cette mission au mieux.

Avec le confinement lié à la crise sanitaire l'activité de l'entreprise a été maintenue en télétravail. Cela a entraîné des difficultés supplémentaires ne me permettant pas d'interroger directement les membres de mon équipe. La disponibilité de mon tuteur entreprise a été bénéfique durant cette période.

Cette licence en alternance m'a permise de consolider une base technique, de la rigueur et une adaptation sur les différents langages. L'investissement de travail a été conséquent.

Les notions de développement apprises en cours et en entreprise ont été complémentaires, les langages utilisés étant différents. La réalisation de nouveaux projets en cours a été bénéfique afin de mieux appréhender l'orienté objet puisque le code en entreprise est dispatché dans de nombreux services.

## V. Glossaire

---

**Docker** : est une plateforme logicielle open source permettant de créer, de déployer et de gérer des containers d'applications virtualisées sur un système d'exploitation. Les services ou fonctions de d'application et ses différentes bibliothèques, fichiers de configuration, dépendances et autres composants sont regroupés au sein du container. Chaque container exécuté partage les services du système d'exploitation.

**Docker locaux** : est l'environnement de développement basé sur Docker créé par Beebuzziness. Il permet la distribution et l'utilisation de l'entièreté de l'application Ustream sur les PC des développeurs.

**Fullstack** : Un développeur intervenant sur le back-end et le front-end d'un site Web ou d'une application.

**GMT** : signifie "Greenwich Mean Time" (temps moyen de Greenwich). Il s'agit de l'heure locale calculée à l'observatoire astronomique de Greenwich, situé près de Londres en Angleterre. Cette heure sert de référence dans le monde entier (on dit par exemple "GMT+5h").

**Interactive Smart Signage** : Application de diffusion de contenus interactifs, accessible via un hub\* et faisant partie de l'écosystème Ustream\*. Elle est utilisée sur des écrans mis à disposition dans des points de vente et permet de créer du contenu personnalisé en fonction des stocks à écouler. Ces programmes/playlists publicitaires personnalisés sont diffusables à la demande sur tous types d'écrans, y compris sur les terminaux mobiles.

**Product Owner** : Le PO est le responsable de la définition et de la conception d'un produit. Il fait le lien entre la partie métier (bonne vision business) et la partie technique du projet. Il doit porter la vision du produit. Il est l'interface entre l'utilisateur, le Scrum Master et les équipes chargées du développement.

**Redis** est une base clé-valeur en mémoire, qui peut éventuellement persister sur disque les données. Redis est classé dans la catégorie des bases NoSQL, une solution open-source codée intégralement en C. Outil très rapide. Il est possible d'attribuer une durée de vie aux données insérées, ce qui permet de mettre en place un système de cache. Il est possible d'utiliser des streams pour travailler sur l'ensemble des données qui sont stockées. On peut le partitionner avec des préfixes pour regrouper/isoler de la donnée.

**Refactorisation** : Permet d'assurer un suivi de l'existant, de faire un ménage qui en facilitera la maintenance. La refactorisation se traduit par éliminer du code mort, documenter, renommer et optimiser du code.

**RPC** : (*remote procedure call*) est un protocole réseaux permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'applications. Ce protocole est utilisé dans le modèle client-serveur pour assurer la communication entre le client, le serveur et d'éventuels intermédiaires.

**Spec :** Les spécifications décrivent le fonctionnement du dispositif digital. L'UX Designer\* et l'UI Designer\* spécifie le comportement de chaque écran de l'interface utilisateur. On clarifie la réponse que l'interface doit apporter lorsque l'utilisateur réalise une action ou active une fonction. Il faut détailler tous les interactions à l'écran, c'est à dire tout ce qui peut se passer sur le plan interactionnel dans la manipulation de l'interface utilisateur. Elles peuvent aussi bien s'appuyer sur des maquettes fonctionnelles (wireframe\*) que sur des maquettes graphiques. Le périmètre couvert par la spécification fonctionnelle se rapporte au fonctionnement de l'interface côté utilisateur. Le périmètre couvert par la spécification technique concerne le fonctionnement technique de l'interface côté administrateur, *en back-office*

**UI Designer :** S'occupe du traitement graphique du wireframe\* et applique une charte. Il intervient quand la maquette est réalisée avec les recommandations de l'Expérience Utilisateur placées par l'UX designer dans le maquettage. Il se charge ainsi de réaliser une interface agréable et utile pour les utilisateurs. Plus axé sur le graphisme et la création, il conçoit et positionne les éléments graphiques et contenu texte d'une interface web par exemple.

**Ux Designer :** Améliore et optimiser l'Expérience Utilisateur quelque soit leur support (smartphone, tablette, grands écrans...). Il peut être amené à faire des interviews pour comprendre les besoins et les habitudes des utilisateur par des questionnaires.

**Wireframe :** Maquette fonctionnelle est un schéma utilisé lors de la conception d'une interface utilisateur pour définir les zones et composants qu'elle doit contenir. À partir d'un wireframe peut être réalisée l'interface proprement dite par un graphiste. La démarche de recourir à des wireframes s'inscrit dans une recherche d'ergonomie. Elle est surtout utilisée dans le cadre du développement et des sites et applications Web. Le wireframe consiste concrètement en un croquis, un collage papier ou un schéma numérique.

## VI. Sitographie

---

Site de Beebuzziness :

<https://ubstream.com/>

Lexique Agile Scrum

<https://agiliste.fr/lexique-agile-scrum/>

Méthode agile

<https://toucantoco.com/blog/methodes-agiles-kanban-revolutionner-management/>

<https://www.ideematic.com/actualites/2015/01/methodes-agiles-definition/>

BDM :

<https://www.blogdumoderateur.com/definition-developpement-web/ss>

rabbitMQ

<https://www.rabbitmq.com/>

TDD :

<https://putaindecode.io/articles/se-lancer-dans-le-tdd/>

UX Designer / UI Designer :

<https://www.journalducmm.com/metier-ux-designer/>

Sinon.js

<https://www.editions-eni.fr/open/mediabook.aspx?idR=39ae4beb4e4edad9da04dfda4034eb36>

# VII. Annexes

## A. Panneau de statistiques détaillé:

Les informations sont classées en trois catégories : Audience, consultation, engagement.

### Consultation



### Consultations

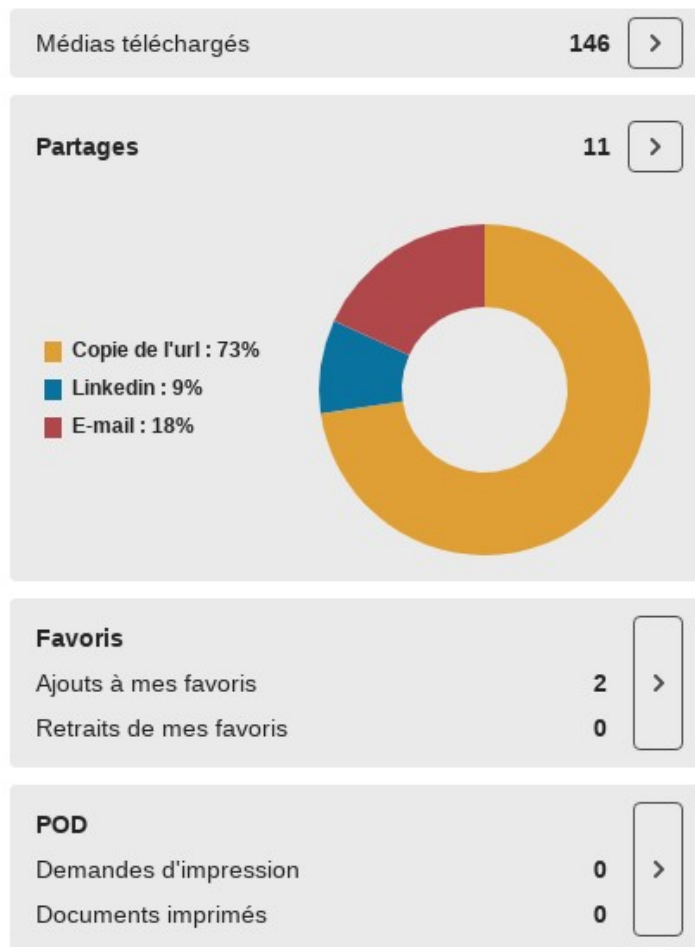
- \* Consultations des médias
- \* Médias consultés au moins une fois
- \* Médias les plus consultés / pages les plus consultées ( stat d'un média de type document)
- \* Confirmation de lecture
- \* Médias déposés

Figure 13: Statistiques disponibles dans la catégorie Consultation

## Engagement

- \* Médias téléchargés
- \* Partages
- \* POD
- \* Prise de contact
  - Demande de rappel
  - Appel audio
  - Appel vidéo

## Engagement



## Abonnements

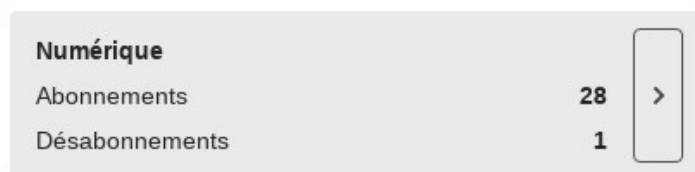


Figure 14: Statistiques disponibles dans la catégorie Engagement et Abonnements

[Retour partie statistiques](#)



## B. Statistiques

### Anciennes organisations

L'ancien système de statistiques datant de 2014 arrivait à ses limites car l'écriture des données et le calcul à la demande des agrégations sur une même base était très coûteuse en puissance machine et le format de données qui avait été créé était peu évolutif.

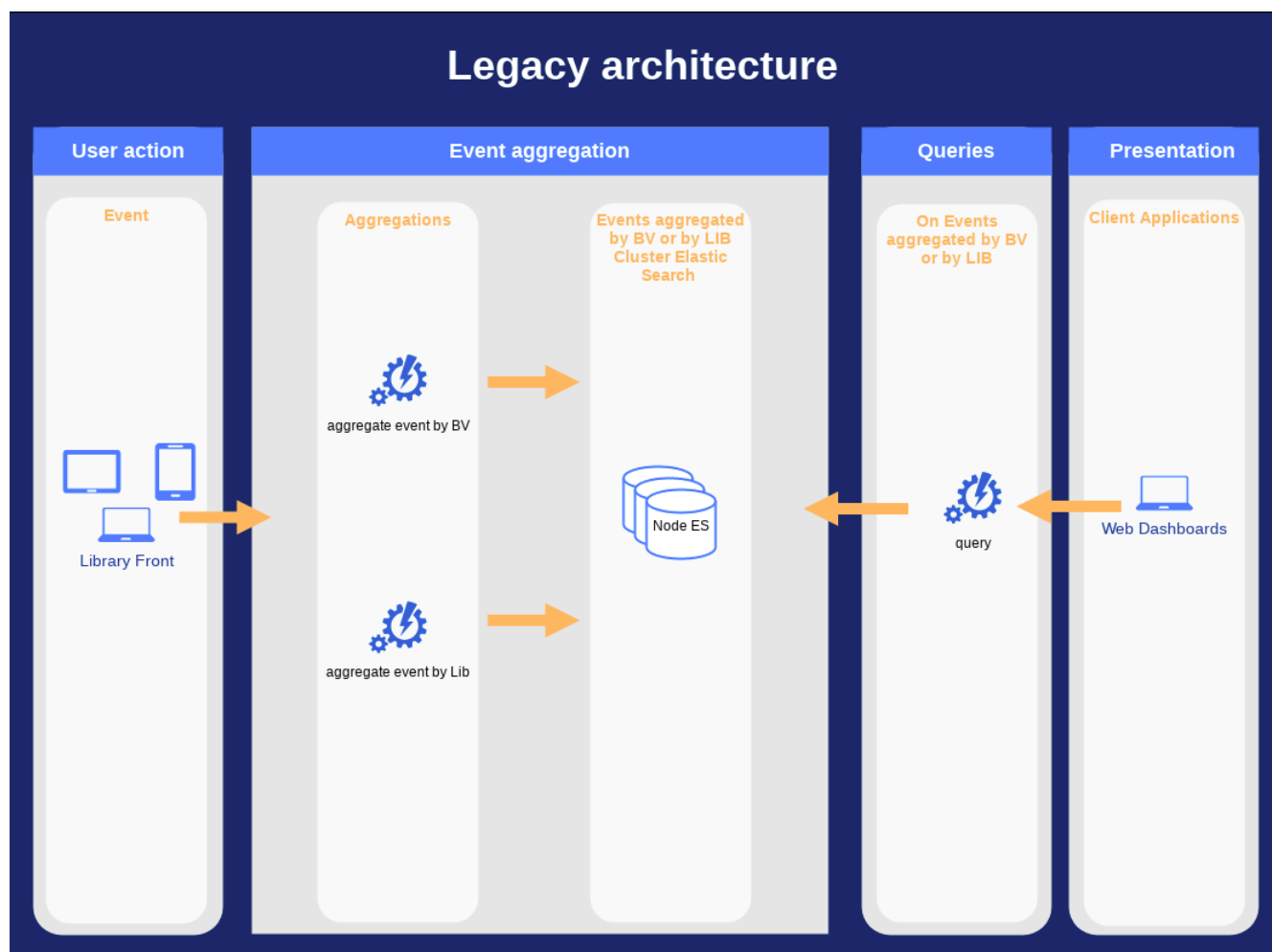


Figure 15: Modèle utilisé avant la mise en place de l'architecture Lambda

Ce schéma représente la nouvelle organisation du système de statistiques sans le sous-composant Speed layer. Les événements côté Aggregation layer et les requêtes côté View layer sont traités séparément avec l'intégration de RabbitMQ pour réguler les événements entrants avec un mode de file d'attente.

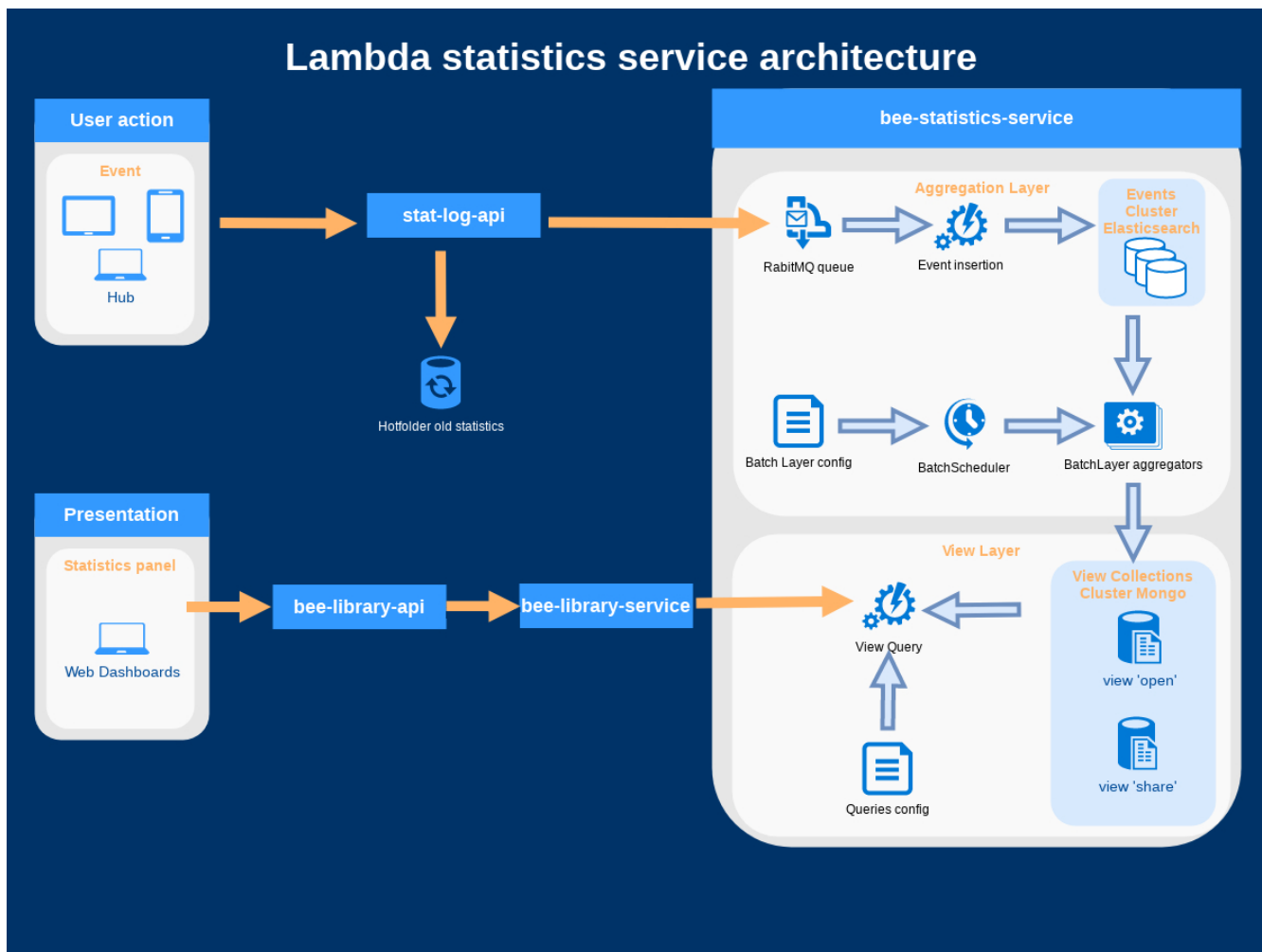


Figure 16: Architecture Lambda sans le traitement de données en temps réel

Le Batch layer est sollicité plus régulièrement sans le Speed layer, environ toutes les minutes alors qu'avec l'intégration de ce dernier il sera sollicité toutes les heures.

**Retour chapitre architecture Lambda**




## C Cheminement des évènements à un instant T

Ce fichier excel permet de visualiser les différents cas possible lorsqu'un évènement est déclenché en détaillant les différents statuts du Batch layer et du Speed layer à un moment donné.

Cheminement de l'évènement « OpenMedia » :

Spécification cas pratiques liées à un évènement : OpenMedia					
Jour J	Time	Batch Layer = BL	Event from BL	Speed Layer Current = SPC	Speed Layer Previous = SPP
10-01-2020	9h59	[ origine BL .. 9h [		[ 9h .. Time [	[ 8h .. 9h [
10-01-2020	10h00	[ origine BL .. 9h [	Starting Computing BL	[ 10h .. Time [	[ 9h .. 10h [
10-01-2020	10h01	[ origine BL .. 9h [		[ 10h .. Time [	[ 9h .. 10h [
10-01-2020	10h02	[ origine BL .. 10h [	BL Compured	[ 10h .. Time [	[ 9h .. 10h [
10-01-2020	10h15	[ origine BL .. 10h [		[ 10h .. Time [	[ 9h .. 10h [
10-01-2020	10h59	[ origine BL .. 10h [		[ 10h .. Time [	[ 9h .. 10h [

On peut regrouper les colonnes en différentes catégories :

-  Temps qui passe
-  État du Batch layer et du Speed layer à l'instant T
-  Évènement du Batch layer envoyé au Speed layer

**Cas 1** => Evènement pris en charge par le current Speed layer

Colonne 1-2 : Lorsque l'évènement est déclenché le 10 janv à 9h59

Colonne 3 : Le Batch layer travaille sur la plage horaire de l'origine jusqu'à 9h

Colonne 4 : Le Batch layer est en cours de traitement sur cette heure (9h à 10h02) donc n'envoie pas l'évènement au Speed layer.

Colonne 5 : C'est le current buffer qui va traiter cet évènement car il travaille de 9h à 10h02

Colonne 6 : Le previous buffer a travaillé de 8h à 9h

**Cas 2** => Evènement pris en charge par le current Speed layer

Colonne 1-2 : Lorsque l'évènement est déclenché le 10 janv à 10h00

Colonne 3 : Le Batch layer travaille sur la plage horaire de l'origine jusqu'à 9h

Colonne 4 : Le Batch layer démarre une nouvelle plage horaire et envoie l'évènement onStarting au Batch layer au Speed layer ce qui va autoriser le switch des données du current au previous et vider le current – On est autorisé à changer de buffer.

Colonne 5 : Le current travaille désormais de 10h à maintenant.

Colonne 6 : Le previous a travaillé de 9h à 10h.

**Cas 3** => Event pris en charge

Colonne 1-2 : Lorsque l'évènement est déclenché le 10 janv à 10h01.

Colonne 3 : Le B. travaille sur la plage horaire de l'origine jusqu'à 9h.

Colonne 4 : Le B. est en cours de traitement sur cette heure (9h à 10h02) donc n'envoie pas d'évènement au Speed layer.

Colonne 5 : C'est le Current qui va traiter cet event car il travaille de 10h à maintenant.

Colonne 6 : Le previous a travaillé de 9h à 10h.

**Cas 4** => Event pris en charge

Colonne 1-2 : Evènement est déclenché le 10 janv à 10h02.

Colonne 3 : le Batch layer travaille sur la plage horaire de l'origine jusqu'à 10h

Colonne 4 : Le Batch layer a terminé le traitement jusqu'à 10h incluse et envoie l'évènement onDone au Speed layer . A ce moment Le contenu du current est copié dans le previous.

Colonne 5 : C'est le Current qui va traiter cet évènement car il travaille de 10h à maintenant.

Colonne 6 : Le previous a travaillé de 9h à 10h et garde son contenu jusqu'au prochain évènement onStarting.

**CAS 5** => Evènement pris en charge

Colonne 1-2 : L'évènement est déclenché le 10 janv à 10h15.

Colonne 3 : Le Batch layer travaille sur la plage horaire de l'origine jusqu'à 10h.

Colonne 4 : Le Batch layer est en cours de traitement sur cette heure (9h à 10h02) donc n'envoie pas d'évènement au Speed layer.

Colonne 5 : C'est le Current qui va traiter cet évènement car il travaille de 10h à maintenant.

Colonne 6 : Le previous a travaillé de 9h à 10h.

**Cas 6** => Evènement pris en charge

Colonne 1-2 : L'évènement est déclenché le 10 janv à 10h59.

Colonne 3 : Le Batch layer travaille sur la plage horaire de l'origine jusqu'à 10h.

Colonne 4 : Le Batch layer est en cours de traitement sur cette heure (9h à 10h02) donc n'envoie pas d'évènement au Speed layer.

Colonne 5 : C'est le current qui va traiter cet évènement car il travaille de 10h à maintenant.

Colonne 6 : Le previous a travaillé de 9h à 10h.

Cheminement de la requête «Combien de médias ont été consultés sur les 30 derniers jours» :

Spécification cas pratiques liées à une requête utilisateur : Combien de médias ont été consultés les 30 derniers jours									
Jour J	Time	Batch Layer = BL	Event from BL	Speed Layer Current = SPC	Speed Layer Previous = SPP	Query period	query period	query result	
10-01-2020	9h59	[ origine BL .. 9h [	Starting Computing BL BL Computed	[ 9h .. Time [	[ 9h .. 9h [	les 30 dernier jours	[ J-29 0h00 ..aujourd'hui 9h59 [	BL[J-29 0h00..J 9h00[ + SPC[9h..Time[	
10-01-2020	10h00	[ origine BL .. 9h [		[ 10h .. Time [	[ 9h .. 10h [	les 30 dernier jours	[ J-29 0h00 ..aujourd'hui 10h00 [	BL[J-29 0h00..J 9h00[ + SPP[9h..10h[ + SPC[10h..Time[	
10-01-2020	10h01	[ origine BL .. 9h [		[ 10h .. Time [	[ 9h .. 10h [	les 30 dernier jours	[ J-29 0h00 ..aujourd'hui 10h01 [	BL[J-29 0h00..J 9h00[ + SPP[9h..10h[ + SPC[10h..Time[	
10-01-2020	10h02	[ origine BL .. 10h [		[ 10h .. Time [	[ 9h .. 10h [	les 30 dernier jours	[ J-29 0h00 ..aujourd'hui 10h02 [	BL[J-29 0h00..J 10h00[ + SPC[10h..Time[	
10-01-2020	10h15	[ origine BL .. 10h [		[ 10h .. Time [	[ 9h .. 10h [	les 30 dernier jours	[ J-29 0h00 ..aujourd'hui 10h02 [	BL[J-29 0h00..J 10h00[ + SPC[10h..Time[	
10-01-2020	10h59	[ origine BL .. 10h [		[ 10h .. Time [	[ 9h .. 10h [	les 30 dernier jours	[ J-29 0h00 ..aujourd'hui 10h02 [	BL[J-29 0h00..J 10h00[ + SPC[10h..Time[	

On peut catégoriser les colonnes :

- Temps qui passe
- État du Batch layer et du Speed layer à l'instant T
- Évènement du Batch layer envoyé au Speed layer
- Requête actions

**CAS 1** => Cheminement requête à 9h59

Colonne 1 : Une requête est lancée sur les 30 derniers jours (l'utilisateur d'un hub ouvre le panneau de statistiques).

Colonne 2 : La période concernée est de J-29 0h00 à aujourd'hui 9h59.

Colonne 3 : On récupère les infos du Batch layer de j-29 0h00 à 9h00 et les informations récoltées du current de 9h à maintenant.

**CAS 2** => Cheminement requête à 10h00

Colonne 1 : Une requête est lancée sur les 30 derniers jours.

Colonne 2 : La période concernée est de J-29 0h00 à aujourd'hui 10h00.

Colonne 3 : On récupère les infos du B. de j-29 0h00 à 9h00, les infos récoltées du previous de 9h à 10h et les infos du current de 10h à maintenant.

sachant qu'à ce moment là le B. envoie l'évènement onStarting au Speed layer ce qui permet de spécifier à celui-ci que la période de 9h à 10h n'est pas prise en charge par le Batch layer.

**CAS 3** => Cheminement requête à 10h01

Colonne 1 : Une requête est lancée sur les 30 derniers jours.

Colonne 2 : La période concernée est de J-29 0h00 à aujourd'hui 10h01.

Colonne 3 : On récupère les informations du Batch layer de j-29 0h00 à 9h00, les informations récoltées du previous de 9h à 10h et les informations du current de 10h à maintenant.

**CAS 4** => Cheminement requête à 10h02

Colonne 1 : Une requête est lancée sur les 30 derniers jours.

Colonne 2 : La période concernée est de J-29 0h00 à aujourd'hui 10h02.

Colonne 3 : On récupère les infos du Batch layer de j-29 0h00 à 10h00 et les informations du current de 10h à maintenant.annexe F

A ce moment là le Batch layer envoie l'aujourd'hui Batch layer onDone au Speed layer ce qui permet de ne plus se baser sur le previous puisque le Batch layer a pris le relais.

**CAS 5** => Cheminement requête à 10h15

Colonne 1 : Une requête est lancée sur les 30 derniers jours.

Colonne 2 : La période concernée est de J-29 0h00 à aujourd'hui 10h15.

Colonne 3 : On récupère les informations du Batch layer de j-29 0h00 à 10h00 et les informations du current de 10h à maintenant.

**CAS 6** => Cheminement requête à 10h59

Colonne 1 : Une requête est lancée sur les 30 derniers jours.

Colonne 2 : La période concernée est de J-29 0h00 à aujourd'hui 10h59.

Colonne 3 : On récupère les informations du Batch layer de j-29 0h00 à 10h00 et les informations du current de 10h à maintenant.

prendre en compte synonymes

[Retour aux diagrammes de séquences](#)

D. Réponse à une requête avec en entrée id, filtre, date début, date de fin [Figure 16].

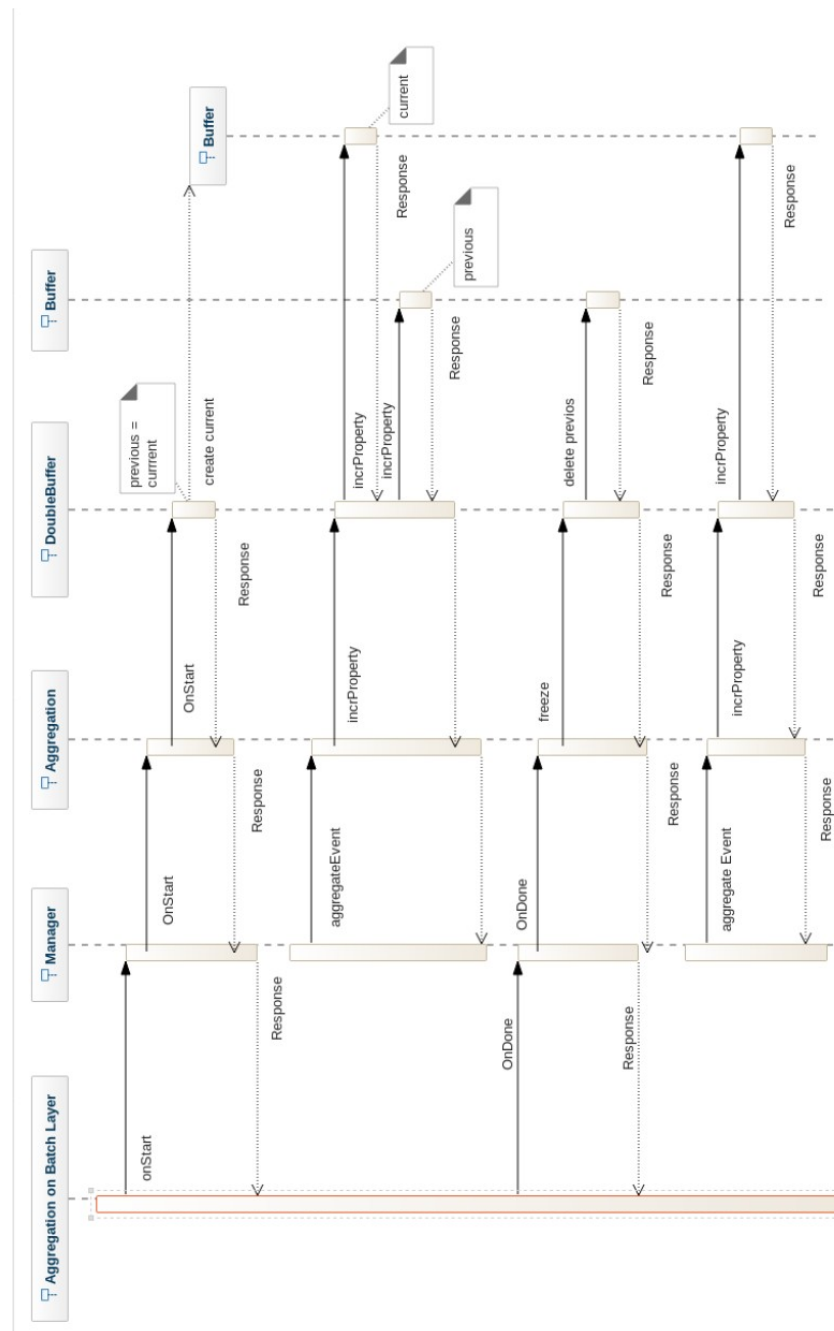
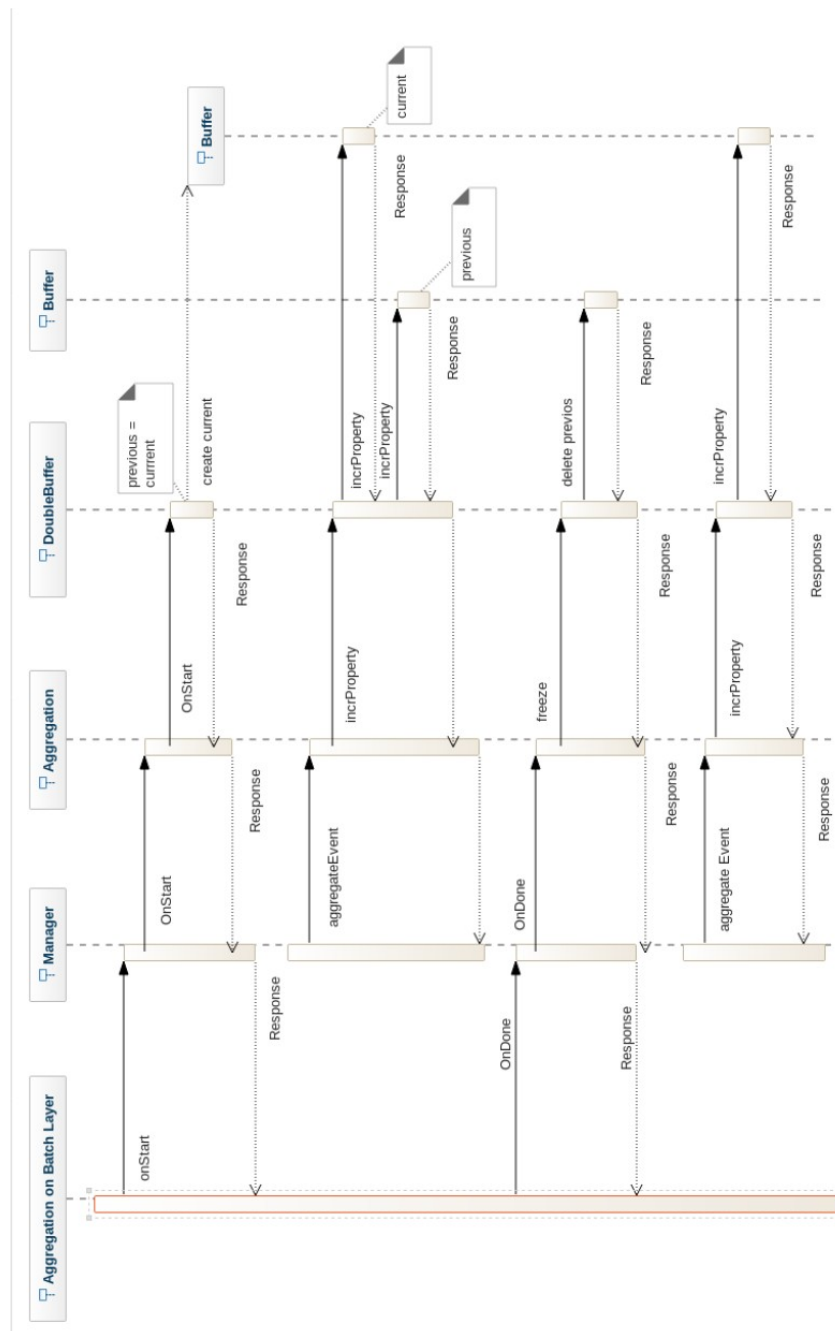


Figure 16 - Parcours d'une requête

[Retour à la partie Entrées du Speed layer](#)

E. Synchronisation du Batch layer vers Speed layer pour informer ce dernier qu'une agrégation est en cours ou se termine



[Retour à la partie Entrées du Speed layer](#)



## **Résumé : création et intégration du Speed layer, sous-composant de l'architecture Lambda.**

Marie kersalé

Beebuzziness, entreprise éditrice de logiciels, situé à Grenoble, propose sa propre solution, Ustream, une plateforme en ligne composée de différents modules permettant d'aider les marques et les propriétaires de contenus digitaux à optimiser leur organisation et la diffusion sur le web.

Un hub est un module central d'Ustream, espace de stockage de documents virtuels autour duquel s'articule des options de consultation, de partage et d'analyse.

Ma mission était basée sur la fonctionnalité d'analyse statistique de l'audience proposée aux clients via un hub. Ces statistiques permettent aux clients d'avoir une vue globale sur le trafic de leur médias au sein d'un hub tel que les médias les plus consultés, comment sont-ils partagés sur les réseaux sociaux.

Au cours de cette mission j'ai intégré la pièce manquante de cet outil reposant sur l'architecture Lambda, le Speed layer amène plus de précisions sur les statistiques en prenant en compte les événements en temps réel. La mission consistait donc à concevoir l'écriture du code du Speed layer et intégrer ce sous-composant dans l'architecture Lambda.

Les objectifs de la mission ont été atteints, le Speed layer est maintenant disponible sur nos dockers locaux et sur mon hub en production et prochainement sur tous les hubs après une série de tests applicatifs et l'accord du directeur du développement.

Mots clés : Ustream, hub, architecture Lambda, Speed layer

**Abstract :** Creation and integration of the Speed layer, subcomponent of the Lambda architecture.

Marie kersalé

Beebuzziness, a software publisher located in Grenoble, offers its own solution, Ustream, an online platform made up of different modules that help brands and owners of digital content to optimize their organization and distribution on the web.

A hub is a central module of abstream, a space for storing virtual documents around options for consultation, sharing and analysis.

My mission was based on the analysis functionality offered to customers via a hub. These statistics allow customers to have a global view on the traffic of their media within a hub such as the most consulted media, how are they shared on social networks.

During this mission I integrated the missing part of this tool based on the Lambda, the Speed layer brings more details on the statistics by taking into account the events in real time. The mission therefore consisted of designing the writing of the Speed layer code and integrating this subcomponent into the Lambda architecture.

The objectives of the mission have been reached, the Speed layer is now available on our local dockers and on my hub in production and soon on all the hubs after a series of application tests and the agreement of the development director.

Key words : Ustream, hub, Lambda architecture, Speed layer