



# 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 de développement, pour m'avoir accordée leur confiance durant cette année d'alternance et proposé 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.....	6
I.3 Organisation.....	7
II. Ma mission, Le Speed layer parcours détaillé.....	10
II.1 Contexte.....	10
II.2 L'architecture Lambda.....	10
II.3 Les données.....	13
II.4 Cadre technique.....	14
III. Réalisations.....	15
III.1 Implémentation.....	15
III.2 Interfaces.....	18
III.3 Test development driven.....	19
III.4 Intégration.....	21
III.5 Objectifs atteints, résultats obtenus.....	21
IV. Conclusion.....	22
V. Glossaire.....	23
VI. Bibliographie.....	26
VII. Annexes.....	27

## Table des figures

Figure 1 : Player. [p7]
Figure 2 : Site officiel Ustream. [p6]
Figure 3 : Hub officiel Ustream. [p6]
Figure 4 : Hub description. [p7]
Figure 5 : Panneau de statistiques. [p7]
Figure 6 : Organigramme des services. [p7]
Figure 7 : Schéma répartitions équipes de développement. [p7]
Figure 8 : Schéma simplifié de l'architecture Lambda. [p7]
Figure 9 : Schéma détaillé de l'architecture Lambda. [p8]
Figure 10 : Liste events et event. [p8]
Figure 11 : Liste requêtes et requête. [p8]
Figure 12 : Aggrations. [p8]
Figure 13 : Diagramme de classes. [p10]
Figure 14 : Diagramme de séquence d'un évènement [p8]
Figure 15 : Diagramme de séquence du batch [p8]
Figure 16 : Diagramme de séquence d'une requête [p8]
Figure 17 : Interfaces [p8]

Figure 18 : Tdd test [p8]

Figure 19 : TDD code [p8]

Figure 20 : Schéma d'architecture du Speed layer service [p12]

Figure 21 : RabbitMQ [p12]

# 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.

Cette entreprise 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'articule différents modules de consultations, de partage et d'analyse de médias.

J'ai rejoint cette entreprise Grenobloise en tant qu'intégratrice il y a 4 ans. Depuis 2017 j'ai intégré une équipe de développeurs et commencé mes premiers pas dans le développement. J'ai pu ainsi participer à la conception de l'ISS (Interactive Smart Signage - service de diffusion interactif de médias\*) et des statistiques.

Ma mission est le système de statistiques disponibles dans les options d'un hub. L'architecture Lambda, système permettant de traiter des volumes importants de données, a été choisie et mise en production cette année. J'ai pour mission de réaliser et d'intégrer le Speed layer, pièce manquante aujourd'hui de ce système qui va permettre d'affiner les résultats de statistiques grâce au traitement de données temps réel.

Le déroulement de cette année d'alternance se découpe en 3 étapes, la première consiste à analyser et comprendre le système Lambda dans son ensemble, la seconde concerne l'implémentation du code du Speed layer et cette dernière implique l'intégration de ce sous-composant dans le système Lambda.

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.

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 ces spécificités techniques, son contexte, son découpage, sa réalisation ainsi que ces objectifs.

# 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 (45 employés).

La société Beebuzziness fût 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 via le Qrcode [Figure3]. On va s'intéresser aujourd'hui à la partie analyse avec des statistiques proposées aux clients retraçant le trafic de hubs et de médias.

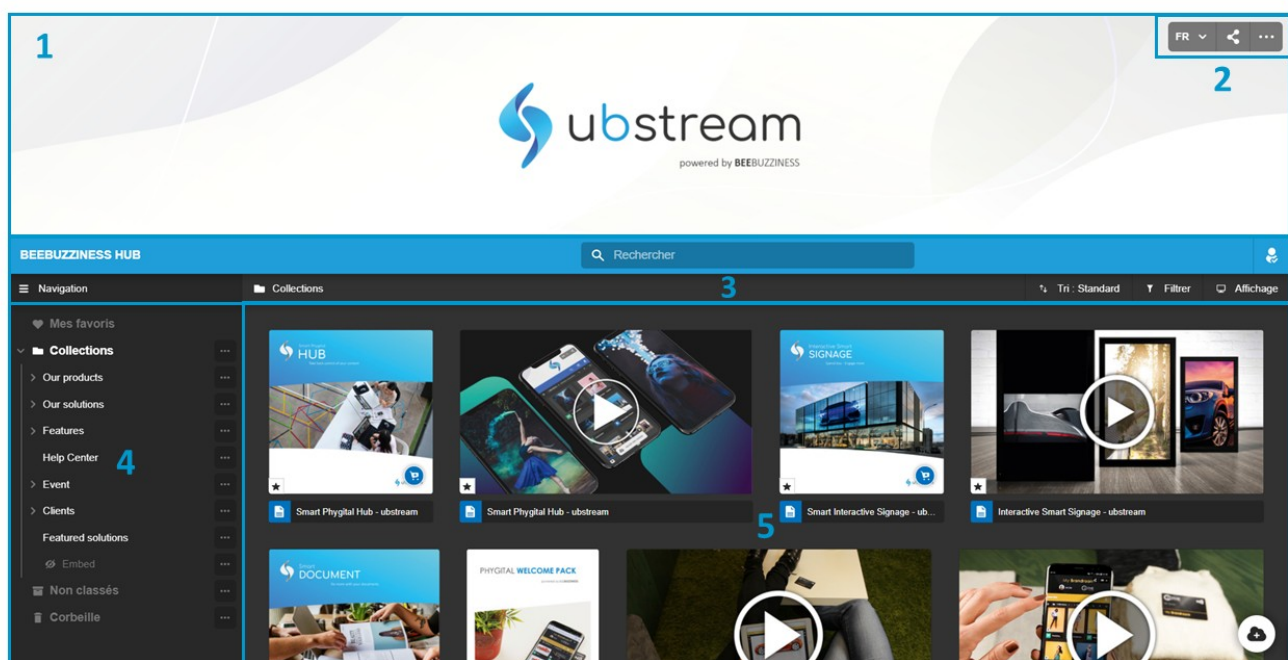


Figure 4: 1-Bannière / 2 -Menu du hub / 3-Barre d'outils / 4- Arborescence / 5-Espace médias



## 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 4]. Vous retrouverez le panneau détaillé en [annexe](#).

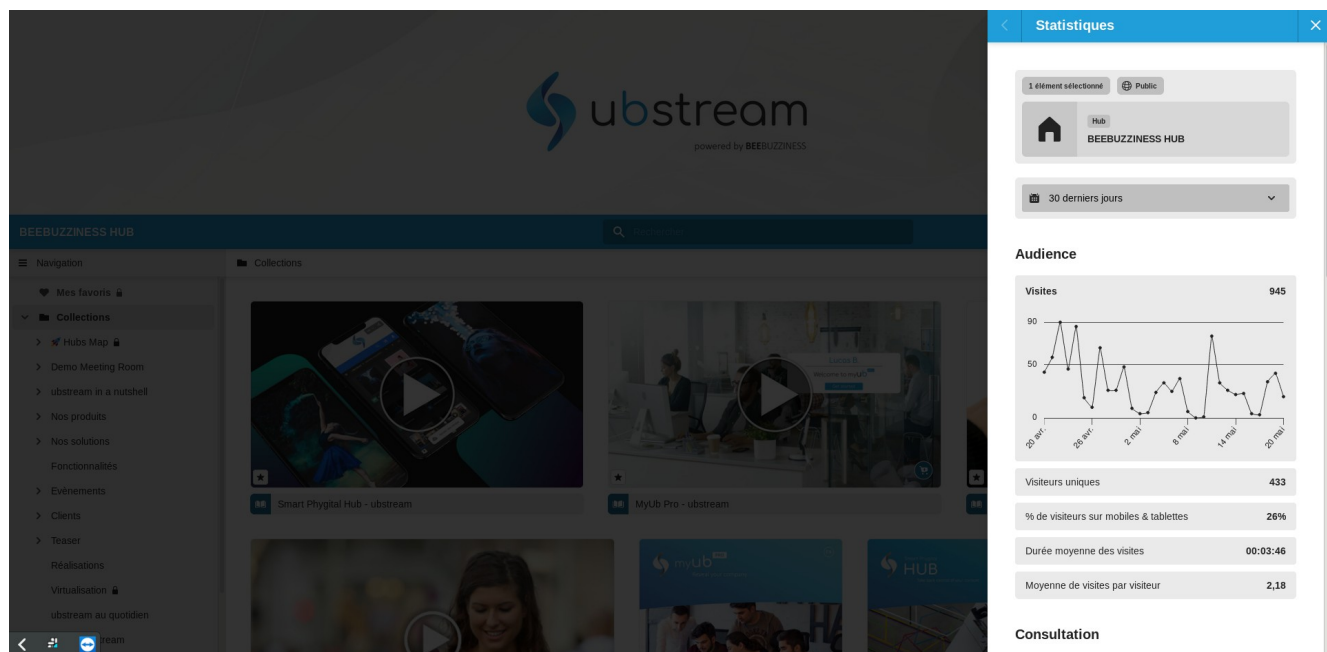


Figure 5: Panneau de statistiques d'un hub

## I.3 Organisation

### Services

Beebuzziness comprend 49 salariés repartis dans 5 services, la production, le marketing, la création, l'administration et le développement [Figure 6].

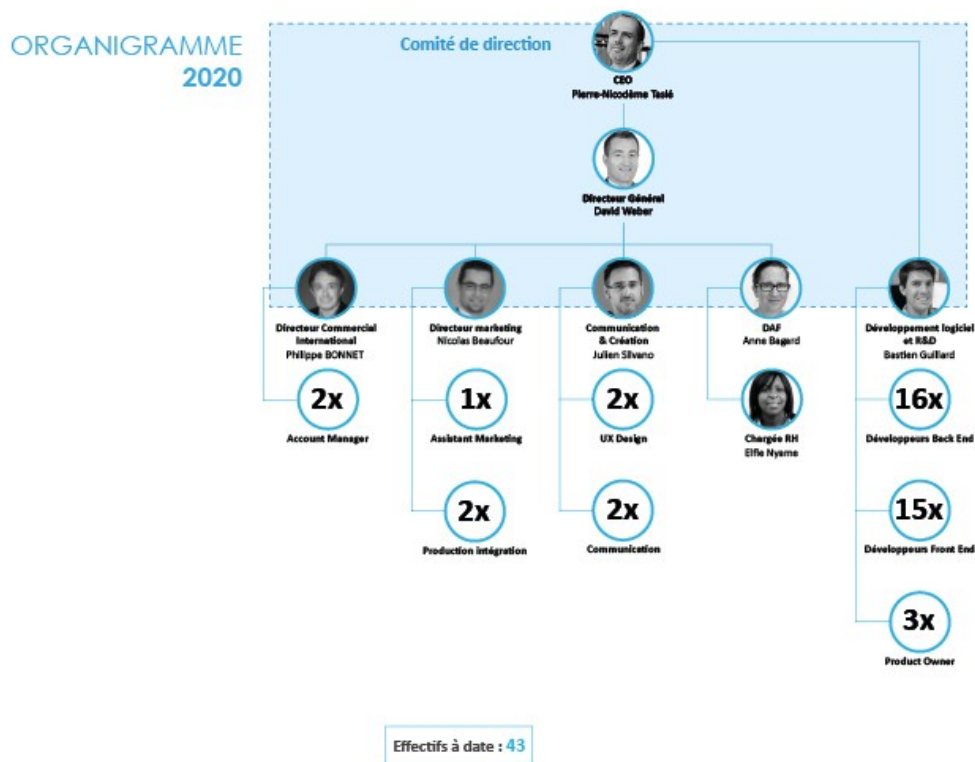


Figure 6: Figure 2: Organigramme des services de Beebuzziness ainsi que la mise en valeur du comité directionnel

### 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 type de ces équipes dirigées par Bastien Guillard notre directeur de développement [Figure 7].

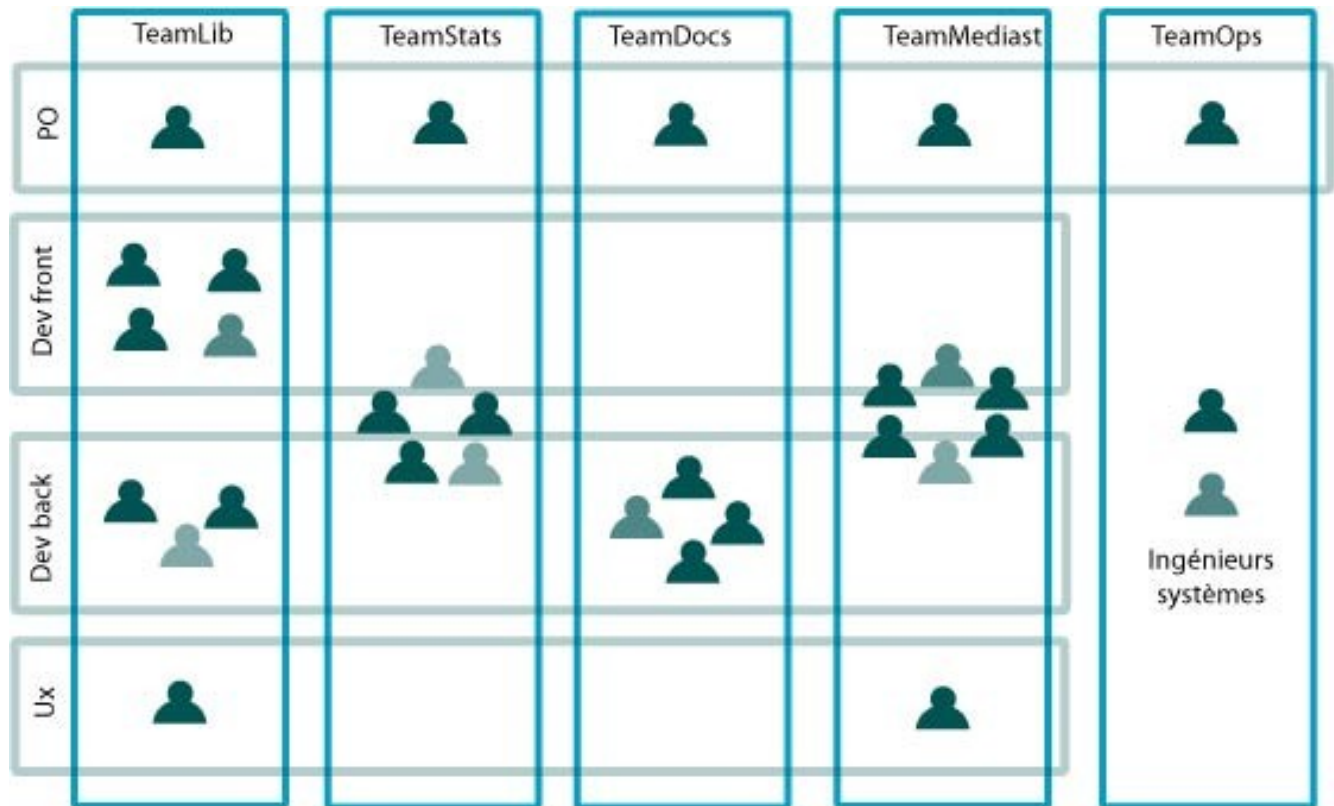


Figure 7: Composition des équipes de développement

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

La «TeamStats» 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'ISS, 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 transformation et le stockage de tous les types de documents gérés par le hub et de manière plus globale le 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 gérés par le hub (hors document), ainsi que les sujets de permissions (accès, modification, suppression,... ) associées 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.

Une **méthode Agile** est une approche itérative et collaborative, capable de prendre en compte les besoins initiaux du client et ceux liés aux évolutions. La méthode Agile se base 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 la méthode agile 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. Cette méthode vise 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 méthode 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, Le Speed layer

---

### Contexte

Ma mission consiste à implémenter le Speed layer, sous-composant de l'architecture Lambda. Je réalise cela au sein de la TeamStats en charge aujourd'hui de l'outil d'analyse. Depuis début 2019, cette équipe est en charge du développement de la nouvelle version de l'outil de statistiques délivrée avec un hub.

Cette équipe 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 sur cette mission et la durée prévisionnelle est de 9 mois soit le temps de l'alternance. J'interviens directement sur le panneau de statistiques dont l'accès se fait via un hub.

L'objectif principal est le déploiement du Speed layer sur nos dockers locaux et sur mon hub afin de montrer les différences entre les statistiques avec et sans le speed layer. Son intégration en production se fera ultérieurement après des ajustements en fonction d'une batterie de tests applicatifs.

### II.1 L'architecture Lambda

#### Description générale

Le but de l'architecture Lambda est de fournir un modèle de traitement temps réel sur des volumes importants de données (Big Data), en proposant un nouveau modèle de calcul.

La conception d'une architecture lambda est guidée par les contraintes suivantes :

- la robustesse avec la tolérance aux erreurs est garantie et la traçabilité de la correction des erreurs.
- la scalabilité en gérant de manière indépendante chaque niveau.
- réduire latence avec l'écriture des événements et l'exécution des requêtes dans les tableaux de bord statistiques.
- la normalisation des événements pour pouvoir répondre à de futurs besoins métiers avec les données accumulées.

- la facilité de maintenance puisque les événements sont simples et les agrégations répondent à un besoin précis.
- les requêtes à la demande sont guidées par le besoin utilisateur et construites sur les agrégations précalculées dans des vues.

### Les composant de l'Architecture lambda

Le système lambda est composé de 3 couches [Figure 8] :

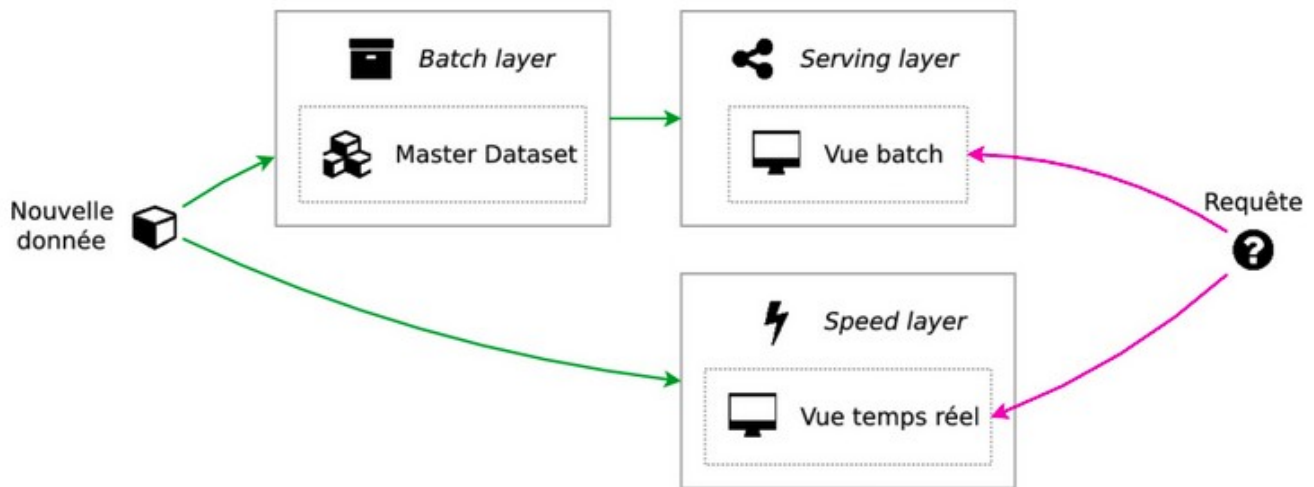


Figure 8: Schéma simplifié de l'architecture Lambda

Le **Batch layer** vise une précision parfaite en permettant de traiter toutes les données disponibles lors de la génération de vues, toutes les heures en se basant sur les heures pleines ou tous les jours en se basant sur l'heure GMT\*. Ce composant peut corriger les erreurs en recalculant l'ensemble des données, puis en mettant à jour les vues existantes. **La sortie des couches Batch layer répond aux requêtes ad hoc en renvoyant des vues précalculées ou en construisant des vues à partir des données traitées.** 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. 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 ainsi 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 les couches Batch layer dès qu'elles sont disponibles, c'est-à-dire dès que le calcul par la couche Batch layer est complété. Cette couche renvoie des vues précalculées ou construit des vues à partir des données traitées. C'est donc de la donnée grise (donnée construite, donnée intermédiaire qui peut être reconstruite au besoin).

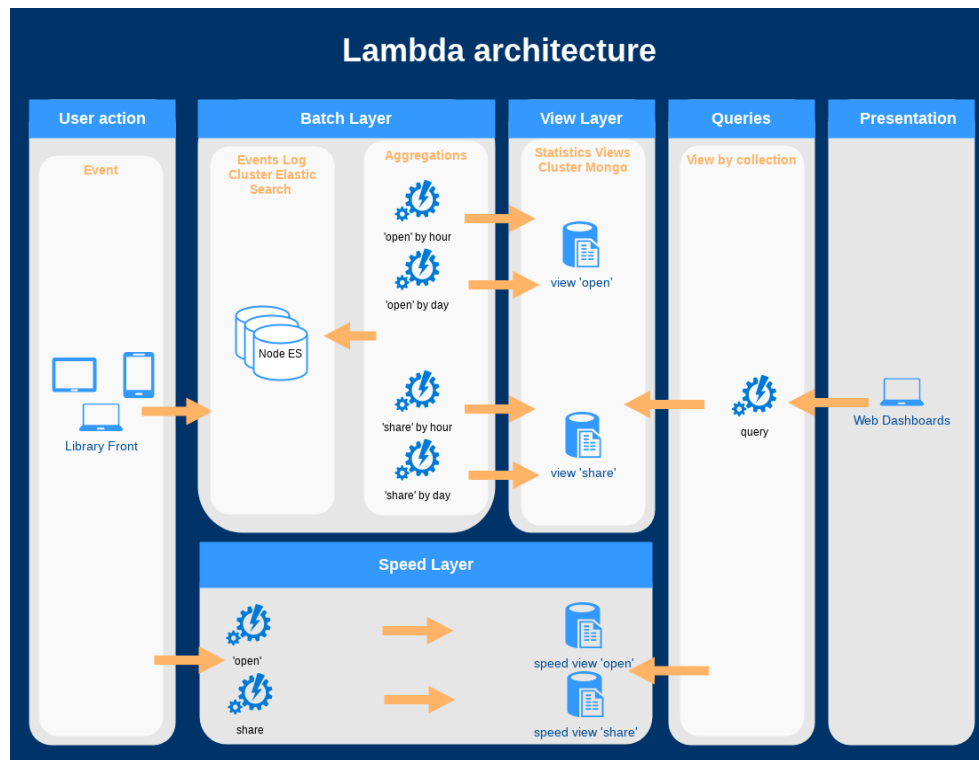


Figure 9: Schéma détaillé de l'architecture Lambda mis en place à terme en production et aujourd'hui disponible sur des dockers locaux

Vous trouverez en [annexe](#) le schéma de l'ancienne organisation des anciennes statistiques.

### 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 permet l'utilisation de cluster sur Mongo et Elastic search qui améliorent notre tolérance aux pannes. En cas de panne d'un des serveurs Mongo ou Elastic search, le cluster gère la redondance. En cas de surcharge sur les événements, c'est le système de messagerie RabbitMq qui nous permet d'absorber les pics de charge.

Les tolérances aux erreurs sont évitées 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 un bucket donné.

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.



## II.2 Cadre technique

### Langages

Nodes.js est une plateforme de développement Javascript. C'est le langage Javascript avec des bibliothèques. 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.

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.

## Objectifs, 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>	Intégration avec le composant Batch layer, validation et déploiement en production.
<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>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. Réalisations

### III.1 Les données

On a besoin de différents types de données afin de gérer les données temps réel, comme les données de configuration des événements et requêtes puis des données d'agrégations et de factory afin d'obtenir le format escompter. Dans un premier temps on va formater la donnée d'un événement puis celle d'une requête.

#### Liste de tous les events disponible

```
2  export enum LambdaEventType {
3      openMedia = 'openMedia',
4      openLib = 'openLib',
5      openMediaReferrer = 'openMediaReferrer',
6      session = 'session',
7      libraryDuration = 'libraryDuration',
8      mediaDuration = 'mediaDuration',
9      shareMedia = 'shareMedia',
10     shareLib = 'shareLib',
11     pageDuration = 'pageDuration',
12     openPage = 'openPage',
13     buyOnline = 'buyOnline',
14     objectStorageSize = 'objectStorageSize',
15 }
16
```

Figure 10: Liste des évènements disponibles de l'architecture lambda

```
107
108 export const openMediaCount: IAggregationOnEventSettings = {
109     buckets: [BucketGranularity.day, BucketGranularity.hour],
110     eventType: EventType.openMedia,
111     outputViewName: 'openMedia',
112     eventKey: ['libId', 'mediaId'],
113     output: {
114         count: {
115             aggregator: Aggregator.count,
116         },
117     },
118     enableSpeedLayer: true,
119 };

```

Figure 11: Définition de la configuration de l'agrégation 'OpenMedia'

## Queries

```
184 export const viewQueriesSettings: IViewQueriesSettings = {
185   [QueryType[QueryType.openMedia]]: openMediaQuery,
186   [QueryType[QueryType.libraryDuration]]: libraryDurationQuery,
187   [QueryType[QueryType.mediaDuration]]: mediaDurationQuery,
188   [QueryType[QueryType.openPage]]: openPageQuery,
189   [QueryType[QueryType.shareMedia]]: shareMediaQuery,
190   [QueryType[QueryType.session]]: sessionQuery,
191   [QueryType[QueryType.visitor]]: visitorQuery,
192   [QueryType[QueryType.mediaSizeInByte]]: mediaSizeInByteQuery,
193   [QueryType[QueryType.mediaDurationByOpenMediaId]]: mediaDurationByOpenMediaIdQuery,
194   [QueryType[QueryType.pagesDurationByOpenMediaId]]: pagesDurationByOpenMediaIdQuery,
195   [QueryType[QueryType.openMediaByMediaId]]: openMediaByMediaIdQuery,
196 };
197
```

Figure 12: Liste des requêtes disponibles

```
54
55 const openMediaQuery: IViewQuerySettings = {
56   inputViewName: 'openMedia', // nom collection dans mongo
57   output: {
58     openMediaTotal: {
59       aggregator: QueryAggregator.add,
60       inputPropertyName: 'count',
61     },
62     openMediaHistogram: {
63       aggregator: QueryAggregator.buildHistogram,
64       inputPropertyName: 'count',
65     },
66     mediaIds: {
67       aggregator: QueryAggregator.buildProjectionAdd,
68       groupId: 'mediaId',
69       inputPropertyName: 'count',
70     },
71   },
72 };
73
```

Figure 13: Définition de la configuration de la requête 'OpenMedia'

## Agrégations

```
5 export enum AggregationType {
6     shareMedia = 'shareMedia',
7     openMediaRefererCount = 'openMediaRefererCount',
8     openMediaCount = 'openMediaCount',
9     libraryDuration = 'libraryDuration',
10    mediaDuration = 'mediaDuration',
11    openPageCount = 'openPageCount',
12    session = 'session',
13    visitor = 'visitor',
14    visitorAggregated = 'visitorAggregated',
15    objectStorageSize = 'objectStorageSize',
16    logicalDatastoreSize = 'logicalDatastoreSize',
17    mediaDurationByOpenMediaId = 'mediaDurationByOpenMediaId',
18    pageDurationByOpenMediaId = 'pageDurationByOpenMediaId',
19    openMediaCountByMediaId = 'openMediaCountByMediaId',
20
21 }
```

Figure 14: Liste des types d'agrégations disponibles

## III.2 Implémentation

### Les classes

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

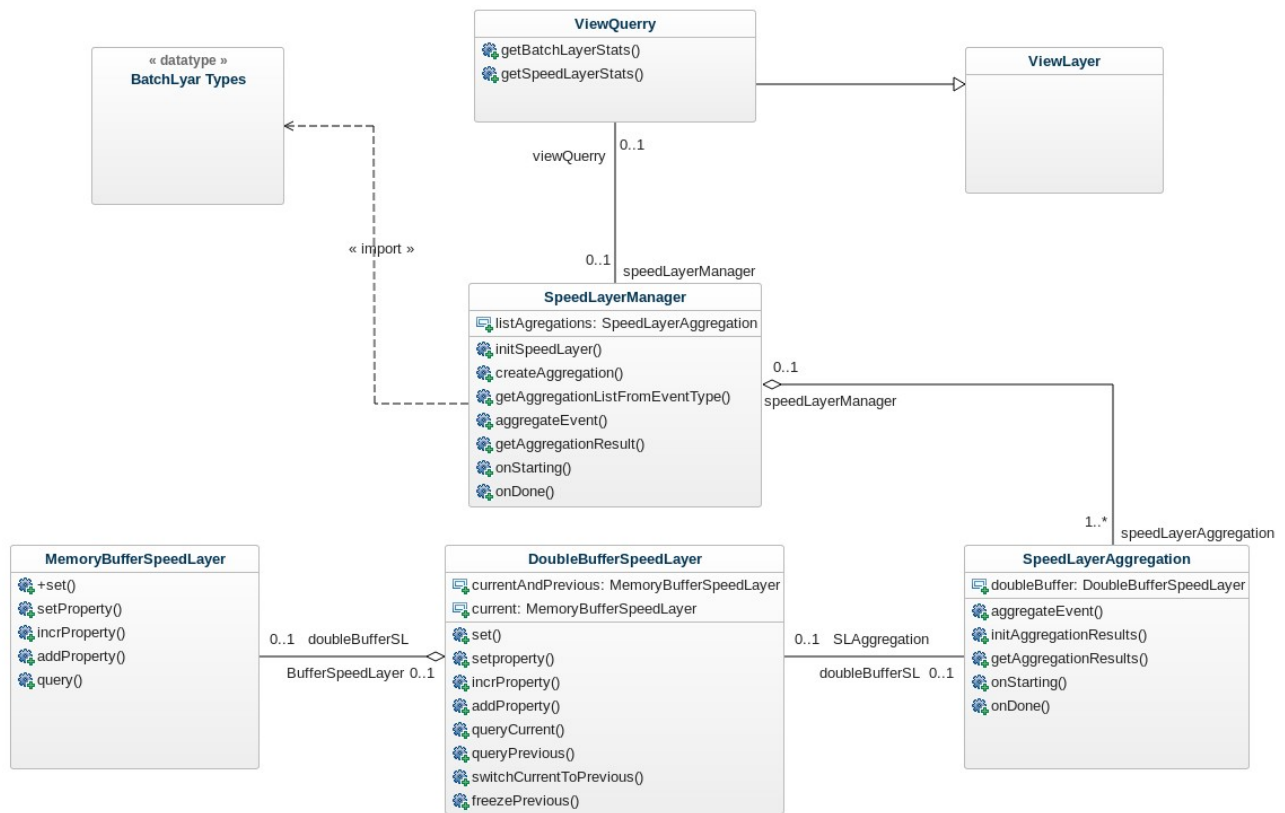


Figure 15: Diagramme de classes du Speed layer

## 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 concernées/adéquates on consulte la liste des configurations d'agrégaions disponibles du Batch layer type, il faut aussi que l'option Speed layer soit activée. Les configurations du Batch layer type non activées tel que *visitorAggregated* ou encore *objectStorageSize* nous permettent de garder dans un même fichier l'ensemble des configurations de l'architecture Lambda. Elles n'ont pas d'utilité dans le cadre du Speed layer puisque aujourd'hui, car nous n'avons pas encore eu la réflexion pour calculer les extrapolations de ces valeurs dans le speed layer, car un calcul incrémental sur l'intervalle de temps connu du speed layer n'est pas suffisant pour construire un résultat. Une autre raison de ne pas activer le speed layer est liée au ratio coût/valeur ajoutée. Si le calcul temps réel coûte très cher, il faut évaluer la valeur ajoutée de ce calcul dans l'application. De même, pour certaines valeurs, nous n'avons pas la nécessité de calculer des valeurs en temps réel, par exemple pour le calcul de la performance des influenceurs....

Le manager en retour récupère l'instance agrégée et la transmet au view Query.

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, le manager transmet les requêtes à l'agrégation du Speed Layer concernée et renvoie le résultat vers le panneau de suivi des statistiques.

## SpeedLayerAggregation

Dans cette classe s'opère la partie agrégation. Cette classe gère la synchronisation avec le Batch layer afin de savoir où il en est grâce aux fonctions *onStarting* et *onDone* afin de pouvoir 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.

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

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



## Double buffer

Le double buffer gère 2 buffers de données. Il ne connaît pas le code métier et les concepts d'événements et d'agrégations. C'est une zone d'aiguillage pour le stockage des données vers l'un ou l'autre des buffers. Il sait router les ajouts de données vers un des 2 buffers, et interroger chaque buffer pour extraire de la donnée.

A ce niveau la donnée est stockée dans les 2 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. Temps de recouvrement d'une agrégation

## Buffer

La classe **s'occupe de compacter de la donnée en** réalisant des calculs simples afin d'optimiser l'espace mémoire.

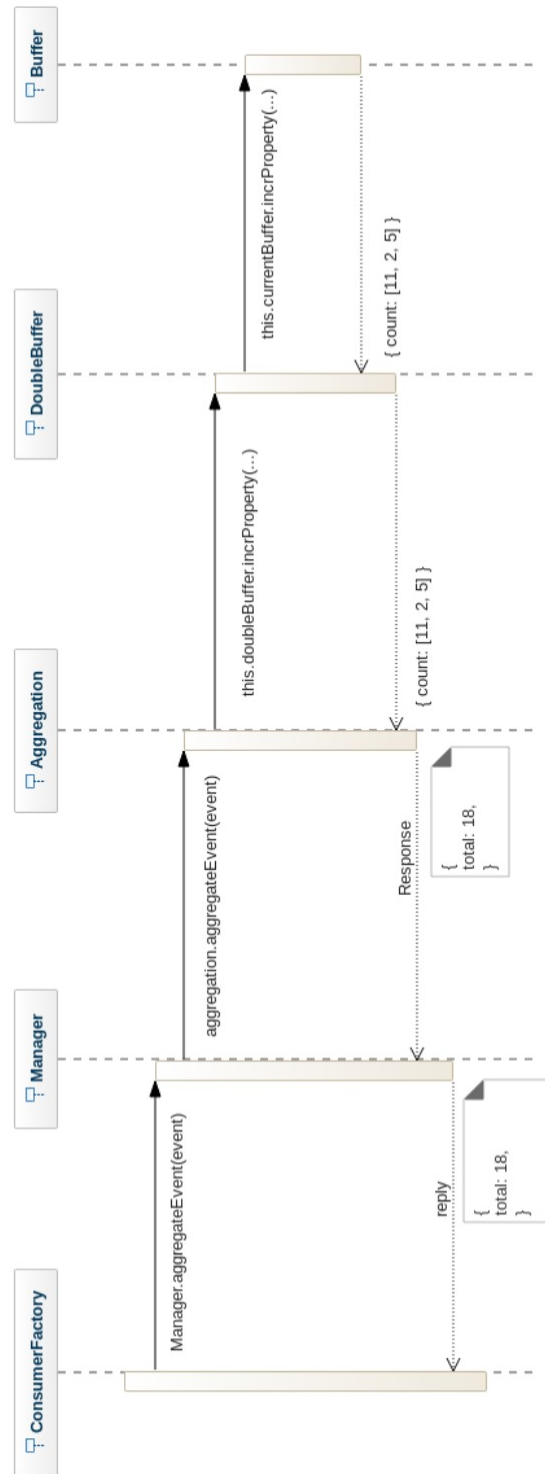
**La notion de Buffer** correspond à la mémoire tampon, zone située dans un disque dur ou dans la mémoire vive d'un ordinateur (pour ce projet le travail s'effectue sur Redis ou sur la mémoire vive) afin de conserver des données pendant un moment en vue de leur utilisation. La mémoire tampon sert à stocker temporairement des données, une sorte de salle d'attente des données qui transitent de manière éphémère. Sans la mémoire tampon et les tampons l'ordinateur fonctionnera moins efficacement et le temps d'attente sera trop long.

Dans notre cas cela permet 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 2 heures et plus précisément, 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 d'un port entrant à un port sortant. Le fonctionnement du buffer permet de stocker des données sur une courte période et conserve les données avant leur utilisation. Les articles traitant du buffer évoquent un inconvénient concernant le dépassement de tampon, qui est évité au sein de ce projet avec l'utilisation de Redis (données stockées en base de données).

## Les entrées du speed layer

### 1. Agréger un event



**2. Synchro batch vers speed pour informer le speed qu'une agrégation est en cours ou se termine sur le batch. => event sous forme d'appel RPC**

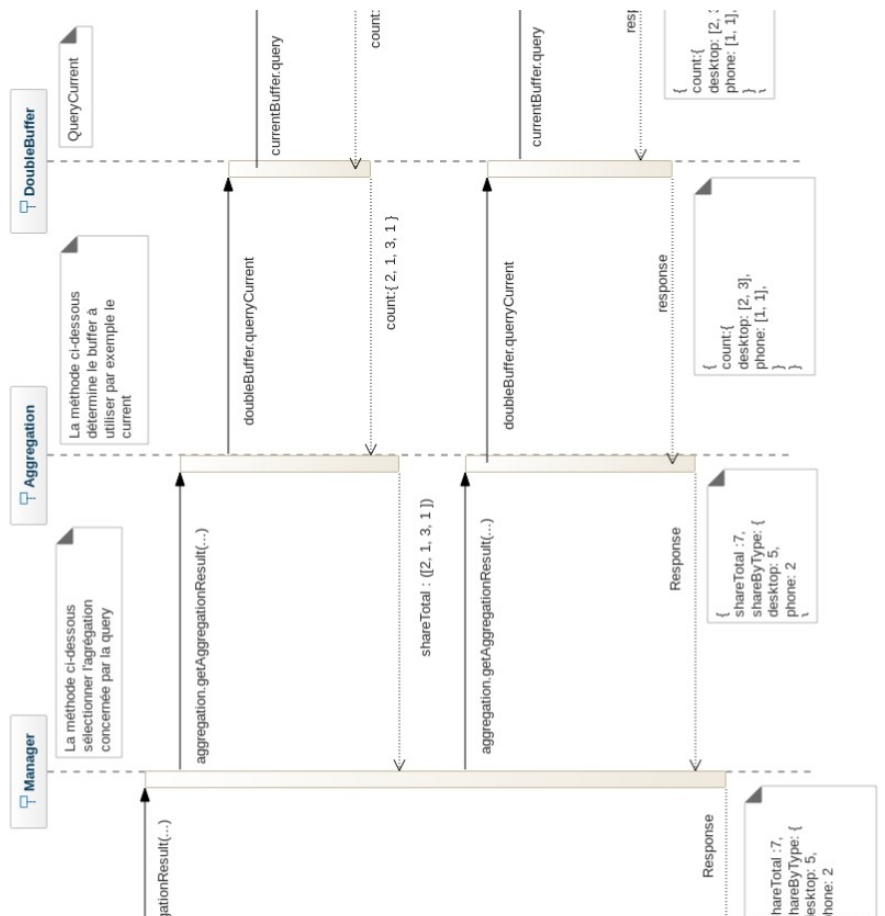
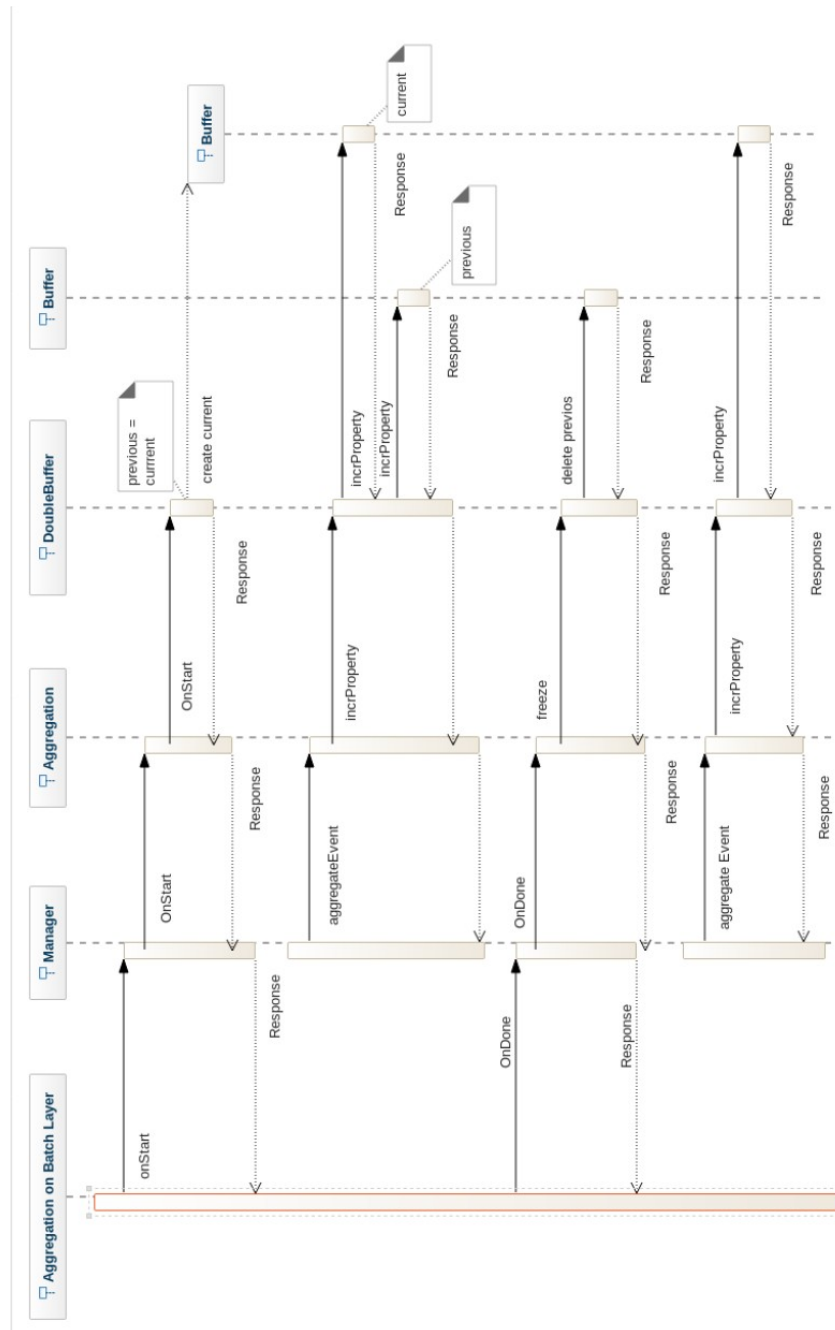


Figure 16: test

en entrée queryId, filtre, date debut, date de fin



### III.3 Interfaces

Une interface définit un ensemble de méthodes sans leurs implémentations et les classes associées seront tenues d'utiliser les méthodes de l'interface mise en place. Cela permet de définir des contrats.

#### Interface d'Initialisation

#### Interface pour le BatchLayer

#### Interface pour le Querylayer

```
25 export interface IInitSpeedLayer {
26   initSpeedLayer(aggregationsSettings?: ISpeedLayerAggregationSettings): void;
27
28   createOneAggregation(aggregationSettings: IAggregationSettings): SpeedLayerAggregation;
29 }
30
31 export interface ISpeedLayerAggregation {
32   aggregateEvent(event: IEvent): IMapAggregation;
33 }
34
35 export interface IQuerySpeedLayer {
36
37   getAggregationResult(query: IViewQuerySettings, params: IFilterKeys, startTimestamp: number, endTimestamp: number): IViewSummary;
38 }
39
40 export interface ISyncSpeedLayerFromBatchLayer {
41
42   onStartCurrentBucketAggregationOnBatchLayer(aggregationType: AggregationType, timestampBeginningBatchLayerAggregation: number): void;
43
44   onCurrentBucketAggregationDoneOnBatchLayer(aggregationType: AggregationType, timestampEndingBatchLayerAggregation: number): void;
45 }
46
47 export interface ISpeedLayerAggregationManager extends IInitSpeedLayer, ISpeedLayerAggregation, IQuerySpeedLayer, ISyncSpeedLayerFromBatchLayer {}
```

1

## III.4 Test development driven

### Définition

Le TDD (Test Driven Development) est une technique de développement mêlant l'écriture des tests unitaires, la programmation et l'amélioration continue du code (encore appelée refactorisation). La méthode traditionnelle de la rédaction des tests unitaires consiste à rédiger les tests d'une portion d'un programme (appelé unité) afin de vérifier la validité de l'unité implémentée. On rédige donc les tests unitaires avant de procéder à la phase de codage.

Le cycle de développement préconisé par TDD comporte cinq étapes comprenant l'écriture d'un test, exécuter ce test et vérifier qu'il échoue (car le code qu'il teste n'a pas encore été implémenté), puis écrire l'implémentation pour faire passer le test, pour ensuite exécuter les tests afin de contrôler que les tests passent et enfin remanier (Refactorer) du code afin d'en améliorer la qualité tout en conservant les mêmes fonctionnalités.

### Intérêts

Commencer par les tests permet de s'assurer que leur écriture ne sera remise à plus tard voir pas du tout. Cela nous pousse à penser aux détails de la méthode dont on a besoin pour écrire la spécification. On va également s'interroger sur le nom de la méthode, sa valeur de retour, ses paramètres, son comportement. Cela permet de clarifier la conception et d'écrire seulement du code utile. Le fait de disposer d'un grand nombre de test permet de s'assurer de la solidité et la garantie du code. Lorsqu'on modifie une méthode existante, on peut relancer les tests unitaires afin de s'assurer que sa modification n'a pas impacté l'existant, cela offre donc un feedback immédiat.

### Mise en situation dans le Speed layer

```
316
317
318   it('expectation: Should invoke getAggregationResult only on the instance of SpeedLayerAggregation for libraryDuration', () => {
319       const fakeAggregationsSettings = {
320           libraryDuration,
321           mediaDuration,
322       };
323       const speedLayerManager =
324           new SpeedLayerAggregationManager(createStubSpeedLayerAggregationFactory, doubleBufferFactory, memoryBufferSpeedLayerFactory);
325       speedLayerManager.initSpeedLayer(fakeAggregationsSettings);
326       const spyGetAggregationResultOnMediaDurationAggregation = sinon.spy(stubSpeedLayerAggregation[mediaDuration.eventType], 'getAggregationResult');
327       const spyGetAggregationResultOnLibraryDurationAggregation = sinon.spy(stubSpeedLayerAggregation[libraryDuration.eventType], 'getAggregationResult');
328       const startTimestamp = new Date('10-02-2020').getTime();
329       const endTimestamp = new Date('20-02-2020').getTime();
330       const eventTimestamp = new Date('15-02-2020').getTime();
331       const filter: IFilterKeys = { libId: [], mediaId: [] };
332       const libraryDurationEvent: ILibraryDurationEvent & IEvent = {
333           eventType: EventType.libraryDuration,
334           timeStamp: eventTimestamp,
335           libId: '1',
336           sessionId: 'session1',
337           duration: 300,
338       };
339       speedLayerManager.aggregateEvent(libraryDurationEvent);
340
341       const libraryDurationQuery = viewQueriesSettings[QueryType.libraryDuration];
342       speedLayerManager.getAggregationResult(libraryDurationQuery, filter, startTimestamp, endTimestamp);
343
344       sinon.assert.notCalled(spyGetAggregationResultOnMediaDurationAggregation);
345       sinon.assert.calledOnce(spyGetAggregationResultOnLibraryDurationAggregation);
346   });
```

### explication code

A partir de la liste d'agrégations existantes on crée une instance de Speed layer, on l'initialise et on définit les paramètres utiles pour la fonction `getAggregateResult()`.

On agrège `libraryDuration` en appelant la méthode `AggregateEvent()` sur l'évènement concerné.

On vérifie que la méthode est appelée sur l'agrégation `LibraryDuration` et non sur `mediaDuration` grâce aux méthodes prédéfinies de Chai `assert called once` / `assert notCalled`.

Dans ce test on a pas besoin de vérifier le résultat de retour mais l'appelle de la méthode `getAggregateResult()`.

```
86 | public getAggregationResult(query: IViewQuerySettings, params: IFilterKeys, startTimestamp: number, endTimestamp: number): IViewSummary {
87 |
88 |     let result = {};
89 |     _forEach(this.aggregationList, (aggregation, outputViewName) => {
90 |         const inputViewName = query.inputViewName;
91 |
92 |         if (aggregation?.isUsingViewName(inputViewName)) {
93 |             result = aggregation.getAggregationResult(query.output, params, startTimestamp, endTimestamp);
94 |         }
95 |     });
96 |
97 |     return result;
98 | }
99 |
```

Ici on parcourt la liste des agrégations disponibles dans l'architecture lambda et si l'évènement se trouve dans cette liste alors on appelle la méthode sur l'agrégation avec en paramètre la requête, mes filtres, la date de début et de fin de l'évènement.

### III.5 Intégration

Le Speed layer est accessible sur 3 points d'entrée, celui des évènements, des requêtes et du Batch layer.

Schéma architecture speed service



Le point d'entrée du front se fait au niveau du service Library au niveau du fichier ViewQuery, avec ces paramètres de requête, de filtres et de dates, elle va appeler soit le View layer soit le Speed layer ou les deux. Dans ce dernier cas le merge des 2 parties se fera dans le ViewQuery, par exemple si la requête concerne des évènements passés on aura besoin seulement du Batch layer dans le cas contraire on aura besoin de compléter avec les calculs à temps réel.

#### **Etapas 1**

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

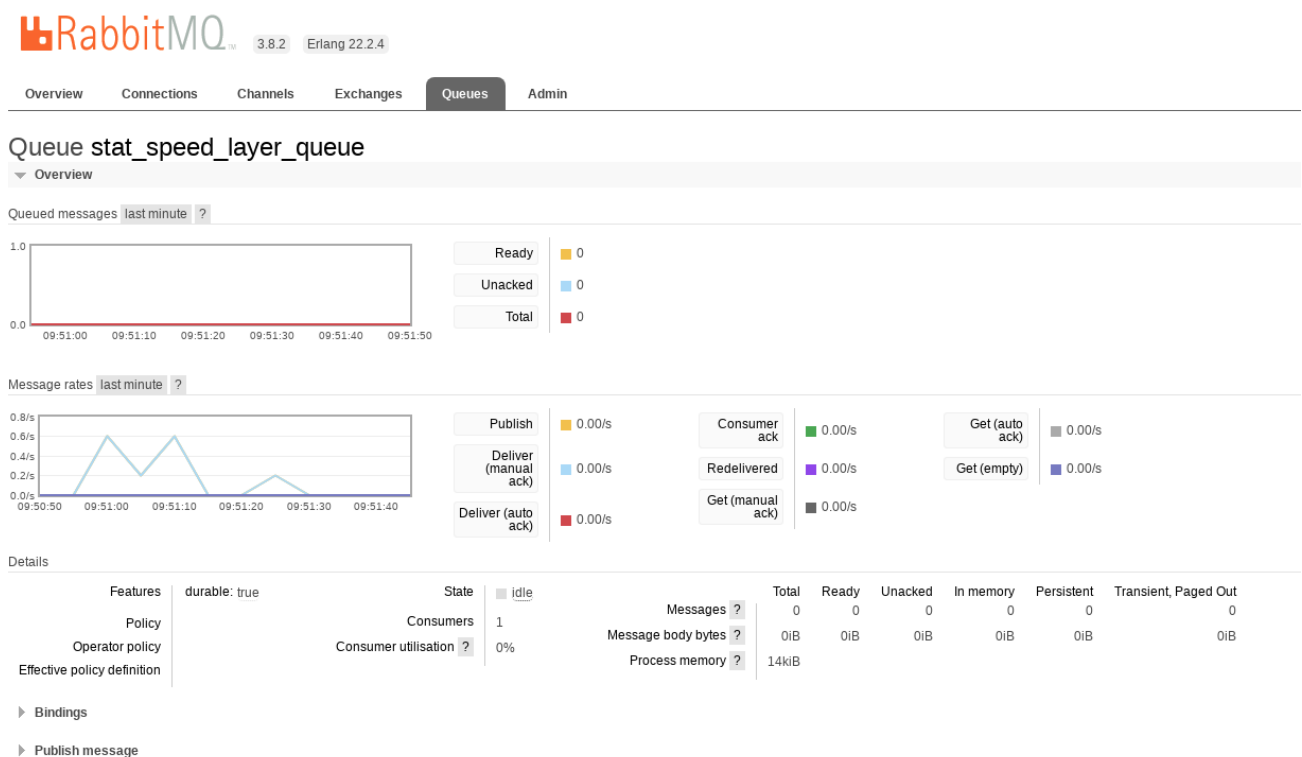
On a intégré les 3 interfaces, l'agrégation des évènements, la gestion des requêtes et la synchronisation des calculs Batch layer / Speed layer.



## Etapes 2

- Câblage du Speed layer avec RabbitMQ avec la création de nouvelles routes.
- Les évènements de statistiques reçus par le Speed layer ne doivent pas être persistant dans rabbitMQ :

La donnée temps réel ne doit pas être gardée sur du long terme, contrairement aux données du Batch layer, et sera effacée une fois la donnée accessible dans le Batch layer afin d'éviter toutes confusions avec une donnée ancienne qui serait gardée par défaut. Pour se faire on a besoin de spécifier à RabbitMQ qu'une persistance des données n'est pas utile puisque cela se fait par défaut.



## Etape 4

- Créer la partie serveur et le service client du Speed layer.

## Etapes 5

- Gestion des requêtes.

### **Etapes restantes gérées par mon tuteur**

- Ajouter feature flag afin de pouvoir activer/désactiver le Speed layer. Ceci permet d'isoler nos développements en encapsulant justement tous ces développements dans ce qu'on appelle des features et ainsi de pouvoir les activer ou pas à la demande de l'utilisateur. Très utile pour tester l'application en interne.
- Un feature flag sera créé pour activer / désactiver dans bee-cluster
- Création d'un `helmChart` Speed layer pour le déploiement.

### *III.6 Objectifs atteints, résultats obtenus*

Le Speed layer est désormais disponible sur nos docker locaux et sur nos hubs locaux.

Les objectifs ont donc été atteints puisqu'il est désormais possible de voir la différence entre les statistiques avec et sans Speed layer.

Des changements ont eu lieu au cours de ce projet puisque nous travaillons de manière itérative :

Le premier changement est la mise en place du buffer et du Double buffer avec l'arrivée du previous. Au début il était prévu de travailler seulement avec un bucket current géré dans le SpeedLayerAggregation et on s'est rendu compte qu'avec le temps d'une agrégation on avait un recouvrement à prendre en compte et que nous ne pouvions supprimer la donnée du Speed layer sans avoir la certitude de perdre de données. Le temps de recouvrement correspond au temps d'une agrégation. Ces deux buffers nous permettent dans un premier temps de supprimer le previous puis celle du current en temps voulu.

Ce qui nous a amené à créer 2 nouvelles classes, le buffer et Double buffer afin d'alléger le code du SpeedLayer 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, lorsque nous aurons un plus grand panel de clients il sera temps de le mettre en place. Cette mise en place a été anticipée et peut se faire rapidement en temps voulu. Pour l'instant la donnée du Speed est stockée en mémoire.

Le choix d'implémentation : le formatage de la donnée du buffer et du double buffer, plusieurs choses ont été testées jusqu'à trouver la forme la plus adaptée à nos besoins. Le premier essai s'est fait avec des maps mais ....

## IV. Conclusion

---

En conclusion, la mission suit son cours, le sujet traité est complexe mais très intéressant avec un encadrement technique et humain poussé.

Les étapes à venir concernent l'intégration du Speed layer avec le service de consommation des événements, gestion d'un double « buffering » sur le Speed layer et la création des composants de communication entre le Batch layer et le Speed layer.

Le rythme soutenu des cours ainsi que les flots d'informations à emmagasiner rapidement demande de la rigueur et beaucoup d'investissement avec lesquels je compte acquérir des automatismes et une rapidité d'exécution.

Je suis satisfaite de ...

Suite au confinement, la charge de travail est restée la même mais l'éloignement physique de mes collègues a ajouté des difficultés. Pas de possibilité de poser mes questions en direct. Michel est très présent et beaucoup d'échanges avec lui ont comblé le manque d'échanges avec le reste de l'équipe.

Le rapprochement que j'ai pu faire entre les cours et le travail en entreprise se fait plutôt dans sa globalité et non de manière spécifique puisque les langages utilisés sont différents. Les cours et le fait de partir sur des projets nouveaux m'ont permis de mieux appréhender l'orienté objet puisqu'en entreprise le code est complexe et dispatché dans de nombreux services.

La difficulté a été de trouver un juste milieu pour poser des questions entre chercher suffisamment et ne pas rester enlisée. Puisque M. est le seul à connaître ce sujet par coeur alors qu'avec les autres de l'équipe il y avait un coup d'entrée le temps qu'ils prennent connaissance du sujet et puisse répondre à mes questions. Faire attention à ne pas faire perdre trop de temps à l'équipe qui a rencontré des périodes de rush lors de périodes de livrables. Fev/ mars et avril.

## V. Glossaire

---

[6] **Saas** : La gestion en mode Saas (Software as a Service) ou logiciel en tant que service est un concept reposant sur le cloud. Il s'agit de s'abonner à des logiciels sous forme de services délivrés via internet plutôt que de les installer sur les serveurs de l'entreprise.

[7] **Ustream** : Solution complète de gestion de contenu en mode Saas, qui permet aux organisations de toutes les tailles d'organiser, de valoriser et de diffuser efficacement leurs médias, grâce à des modules simples et rapides à prendre en main.

[0] **Digital 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.

[0] **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").

[0] **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.

[0] **Previous buffer**

[0] **Current buffer**

[0] **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*

[0] **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.

[0] **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 maquetage. 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.

[0] **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.

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

[0] **Impression à la demande:**

[0] **Espions** : Les espions sont des fonctions qui enregistrent un certain nombre de paramètres, comme le nombre de fois qu'elles ont été appelées, avec quels paramètres...  
Ils sont utiles pour tester le comportement de composants qui manipulent des fonctions, comme un émetteur d'événements.

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>

# VI. Annexes

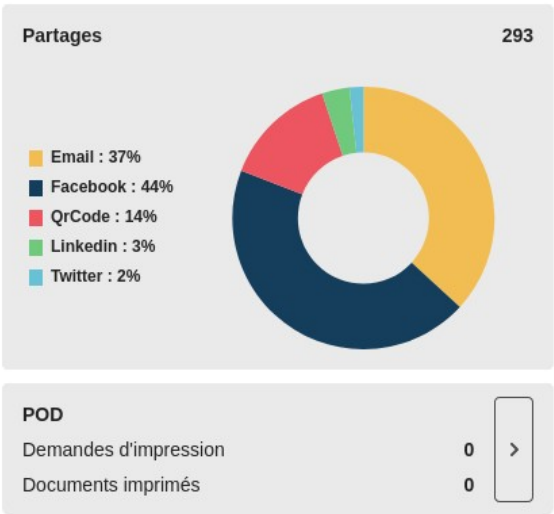
**Panneau de statiistiques détaillé:**

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

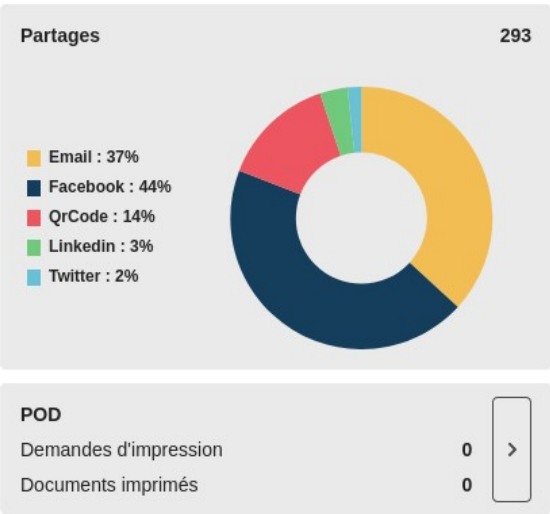
**Audience**

- \* Visites
- \* % de visiteurs sur mobiles et tablettes
- \* Durée moyenne des visites
- \* Moyenne de visites par visiteur

**Engagement**



**Engagement**



**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



## Engagement

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


## Consultation

Consultations de médias **1 491 553**


Médias consultés au moins une fois **75**

### Médias les plus consultés




La course aux cadeaux est ouverte  **158 407**



Trouvez toutes vos idées cadeaux  **144 080**





La magie d'être ensemble ! On p...  **143 364**



Galette des Rois Epiphanie  **139 664**



Les Soldes  **123 633**

Médias déposés **32** 

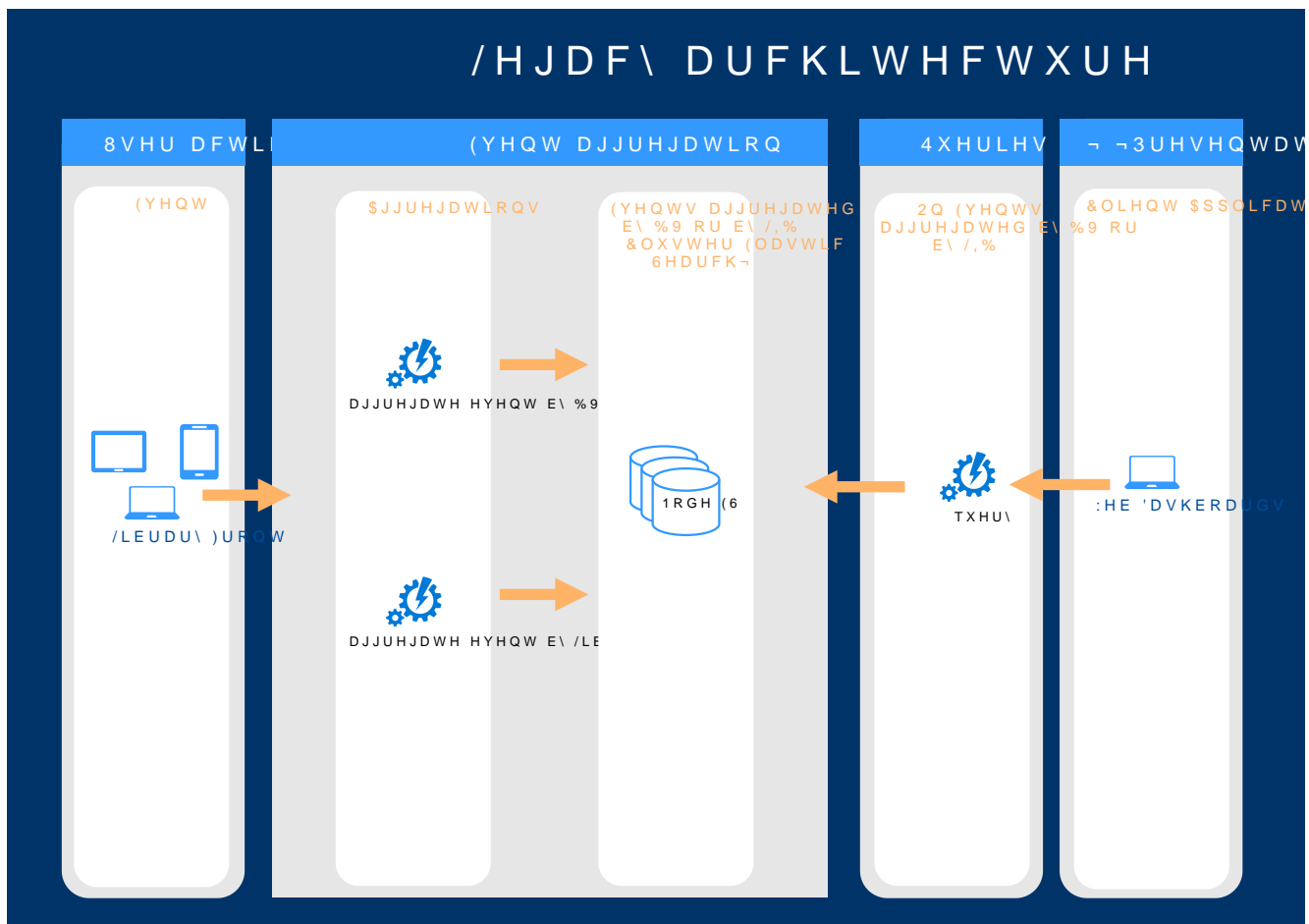
### Statistiques futures :

- Trier les médias d'un hub par nombre de consultations ou de partage
- Consulter les statistiques de médias détaillées par variante (langue) d'un même média
- Consulter les statistiques d'une sélection manuelle de médias
- Consulter les statistiques globales de Ustream
- Consulter le parcours d'un utilisateur et son historique
  - o Nombre de sessions
  - o Géolocalisation
  - o Devices utilisés
  - o Durée des sessions
  - o Contenu des sessions
    - Consultations de médias
    - Durée de consultation
    - Partages / Ajout aux favoris
    - Changement de hub
    - Changement de collection
- Suivre le parcours de dissémination (Lib-it/Ajout aux favoris) d'un média entre différents hubs et pouvoir calculer des éléments de rétribution (Rétribution d'un influenceur générant beaucoup de consultations/partages/... de médias appartenant à des marques depuis son hub)

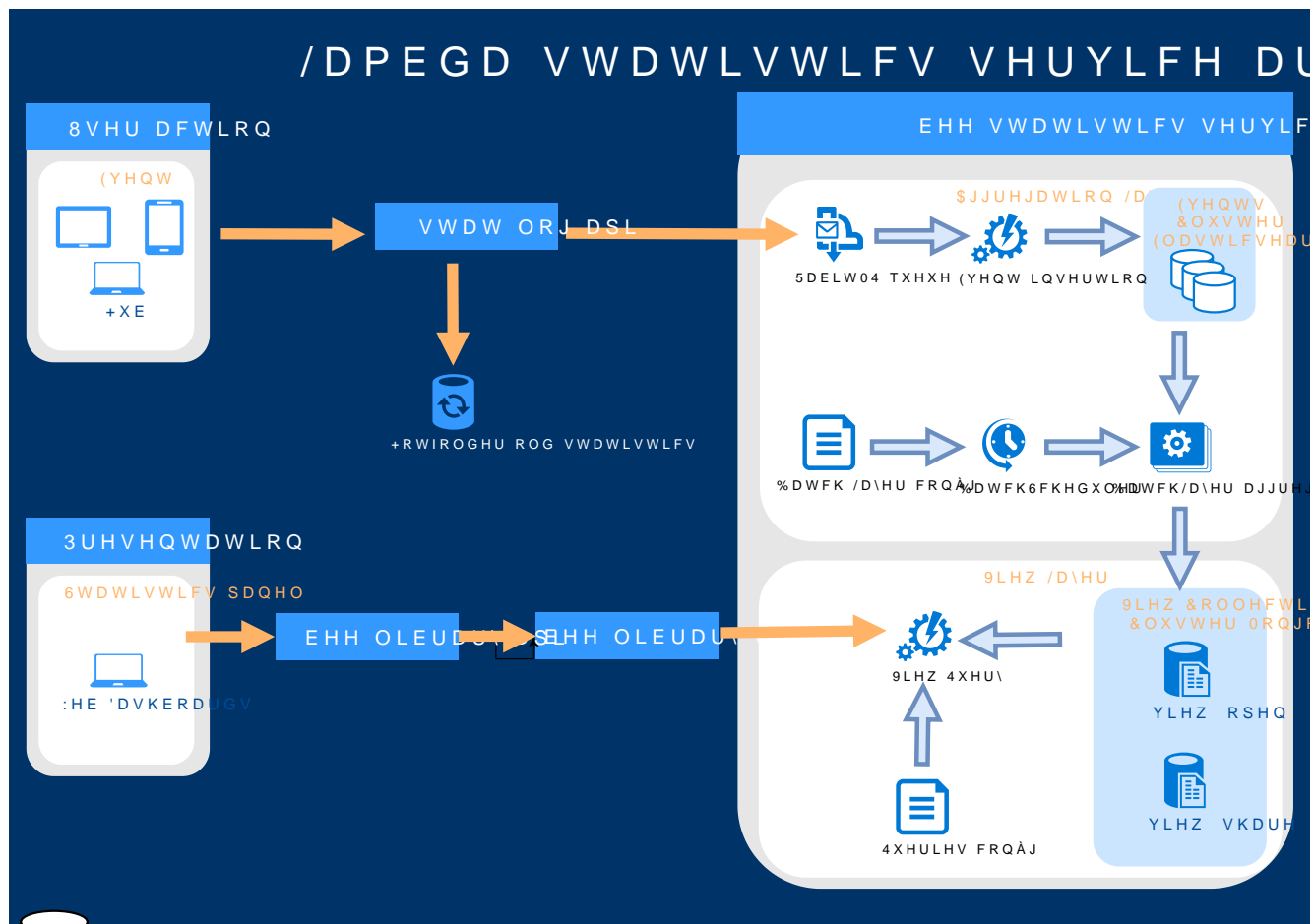
## Statistiques

### Anciennes organisations

L'ancien système de statistiques datant de 2014 arrivait à ses limites car celui-ci ne permettait pas d'ajouter de nouvelles agrégations aisément, engageant des coûts importants et ne supportait pas la montée de charge puisque les événements et les requêtes reposaient essentiellement sur Elastic Search. Cela demandait une maintenance journalière qui monopolisait un développeur à temps plein pour vérifier le bon fonctionnement et récupérer la donnée non intégrée en cas de surcharge (lors de la mise en ligne des catalogues Auchan par exemple).



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



[Retour chapitre archi Lambda](#)

## 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 status 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 Computed	[ 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
- Évènements du Batch layer envoyés au Speed layer

**Cas 1** => Event 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 qui va traiter cet évènement car il travaille de 9h à 10h02

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

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

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

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

Colonne 4 : Le B. démarre une nouvelle plage horaire et envoie l'événement Starting Computing au BL au S. ce qui va autoriser le switch des données du current au previous et vider le current // Autoriser le changement de buffer

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

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

**Cas 3** => Event pris en charge .... le current S. tjrs

Colonne 1-2 : Lorsque l'événement 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'événement au S.

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

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

**Cas 4** => Event pris en charge .... le current S. tjrs

Colonne 1-2 : Lorsque l'événement est déclenché le 10 janv à 10h02

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

Colonne 4 : Le B. est terminé le traitement jusqu'à 10h inclus et envoie l'événement BL computed au S. ce qui va "faire sauter" le previous et travaille uniquement le current

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

Colonne 6 : Le previous a travaillé de 9h à 10h et garde son contenu jusqu'au prochain événement Starting Computing au BL

**CAS 5** => Event pris en charge .... le current S. tjrs

Colonne 1-2 : Lorsque l'événement est déclenché le 10 janv à 10h15

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

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

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

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

**Cas 6** => Event pris en charge .... le current S. tjrs

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

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

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

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

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	[ 9h ... Time [	[ 8h ... 9h [	les 30 dernier jours	[ J-29 0h00..aujourd'hui 9h59 [	BL[J-29 0h00..J 9h00[ + SPP[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[ + SPP[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[ + SPP[10h..Time[	
10-01-2020	10h59	[ origine BL ... 10h [	BL Computed	[ 10h ... Time [	[ 9h ... 10h [	les 30 dernier jours	[ J-29 0h00..aujourd'hui 10h02 [	BL[J-29 0h00..J 10h00[ + SPP[10h..Time[	

On peut catégoriser les colonnes :

	Temps qui passe
	État du Batch layer et du Speed layer à l'instant T
	event du B. envoyé au speed
	Requête

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

Colonne 1 : Une requête est lancé sur les 30 derniers jours (l'utilisateur d'un hub ouvre le panneau de stat

Colonne 2 : La période concernée est de J-29 0h00 à auj 9h59

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

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

Colonne 1 : Une requête est lancé sur les 30 derniers jours

Colonne 2 : La période concernée est de J-29 0h00 à auj 10h00

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

sachant qu'à ce moment là le B. envoie l'event Starting Computing BL au S. ce qui permet de spécifier au S. que la période de 9h à 10h n'est pas prise en charge par le B.

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

Colonne 1 : Une requête est lancé sur les 30 derniers jours

Colonne 2 : La période concernée est de J-29 0h00 à auj 10h01

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

sachant qu'à ce moment là le B. envoie l'événement BL computed au S. ce qui permet de ne plus se baser sur le previous puisque le B. 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 infos du B. de j-29 0h00 à 10h00 et les infos du S 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 infos du B. de j-29 0h00 à 10h00 et les infos du S current de 10h à maintenant



## **Résumé : Immersion dans 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.

Ma mission est liée à la fonctionnalité d'analyse proposée aux clients via un hub. Cette fonctionnalités permet 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. Je dois intégrer la pièce manquante de cet outil reposant sur l'architecture Lambda, le Speedlayer qui va amener plus de précisions sur les statistiques en prenant en compte les événements en temps réel.

La mission consiste à concevoir l'écriture du code du Speed layer et intégrer ce sous-composant dans l'architecture Lambda. L'implémentation du code c'est fait en TDD.

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

## **Abstract : Immersion in lambda architecture**

Marie kersalé

Beebuzziness, located in grenoble, offers a global solution for digital communication who combines the advantages of digital and rematerialization on demand.

Ustream, platform brings together all the functionalities necessary for the management and distribution of content.

My mission is linked to statistics since my team is in charge of creating a new version of the statistics tool since the beginning of 2019. The new system is based on the Lambda architecture and I am involved in this project. My mission is to realize the Speed layer, the last missing piece of the Lambda architecture.

The mission is in 3 steps:

- analysis and understanding of the Lambda system as a whole,
- design and writing of the Speed layer code, seen as a system autonomous,
- integration of the component into the system, including communication and synchronization of the component with the other two components of the architecture Lambda.

The new version of the statistics allows to easily add new analysis information according to customer requests without any additional cost for the company.

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

Les classes représentent un concept, un type de boîte (un tiroir).

Les instances sont des objets, où l'on trouve des valeurs (on remplit le tiroir)

**Les spécifications décrivent le fonctionnement du dispositif digital.** On 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 toutes 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.