



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



Beebuzziness

Membres du Jury :

Gwenn Boussard

Michel Poitevin - Tuteur entreprise

Francis Brunet-Manquat - Tuteur universitaire

Auteur : Marie Kersalé

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 mis 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

I. Introduction.....	6
II. Beebuzziness, entreprise d'accueil.....	7
II.1 Présentation.....	7
II.2 Activité.....	7
Le Hub.....	7
Les statistiques.....	7
II.3 Organisation.....	7
Méthodes agiles, Scrum.....	7
Equipes de développement.....	8
III. Ma mission, Le Speed layer parcours détaillé.....	9
III.1 Contexte.....	9
III.2 L'architecture Lambda.....	9
Description générale.....	9
Les composant de l'Architecture lambda.....	10
Avantages.....	11
III.3 Les données.....	11
Données utilisateurs.....	11
Données.....	12
III.4 Cadre technique.....	12
III.5 Découpage prévisionnel.....	14
III.6 Bac à sables.....	15
III.7 Implémentation classes du SpeedLayer code.....	15
SpeedLayerManager.....	15
SpeedLayerAggregation.....	15
Double buffer .....	15
Buffer.....	16
III.8 Interfaces.....	16
Interface avec le BatchLayer.....	17
Interface avec le Querylayer.....	17
Interface avec d'Initialisation.....	17
III.9 Test development driven.....	17
Définition.....	17
Intérêts.....	17
Mise en situation dans le Speed Layer.....	17
III.10 Intégration.....	18
Configuration.....	20
Application.....	20
Consumer.....	20
Tests d'intégration et d'application.....	21
IV. Conclusion.....	24
V. Glossaire.....	25
VI. Bibliographie.....	28

## Table des figures

- Figure 1 : Site officiel Ustream. [p6]  
Figure 2 : Hub d'Auchan. [p7]  
Figure 3 : Lecteur proposé par Beebuzziness pour consulter des edocuments. [p7]  
Figure 4 : Le studio est un outil d'enrichissements de documents virtuels. [p7]  
Figure 5 : Catalogue Auchan en ligne. [p8]  
Figure 6 : Clients de Beebuzziness. [p8]  
Figure 7 : Représentation générale des équipes de développement. [p10]  
Figure 8 : Tableau des tâches à réaliser durant le sprint en cours. [p12]  
Figure 9 : Accès des statistiques via le Hub. [p12]  
Figure 10 : Analyse des consultations de médias dans le panneau de statistiques. [p13]  
Figure 11 : Ici on peut voir sur quel réseau le document a été partagé. [P13]  
Figure 12 : Ancien système de statistiques mis en place fin 2014. [p14]  
Figure 13 : Architecture mise en place cette année. [p14]  
Figure 14 : Architecture une fois ma mission achevée. [p17]

# I. 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.

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 du Digital Signage\* (service de diffusion interactif de médias\*), du Reporting et des statistiques.

Ma mission consiste à intervenir au niveau du système de statistiques récemment disponible en production puisque mon équipe est responsable de cette partie de la plateforme Ustream.

L'architecture Lambda\* a été choisie et mise en production cette année et j'ai pour mission d'intervenir sur le Speed Layer, pièce manquante aujourd'hui de ce système.

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 SpeedLayer et cette dernière, implique l'intégration du composant dans le système Lambda (détails chap. III).

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, informations évoquées lors de la soutenance de janvier dernier. Je détaillerai par la suite / ultérieurement ma mission, son contexte, son découpage ainsi que ces objectifs.

## II. Beebuzziness, entreprise d'accueil

---

### II.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\*. 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, utilisé par exemple dans les grandes entreprises pour la diffusion de rapport annuels ou plaquettes d'information interne ou de proposer des catalogue en ligne avec leur fiches produit pour des marques telle que Auchan. Dans un premier temps il était possible d'enrichir ces documents avec des ZZO\*, des liens internes, externes, qrcode et animations pour une consultation fluide et interactive.

En 2016, Beebuzziness déploie le Hubstream\* qui rassemble toutes les fonctions nécessaires à la gestion et à la diffusion de médias\* (détaillé ci-dessous).

### II.2 *Activité*

#### **Le Hub**

Le Hub\* (Figure 2) est un espace de stockage de documents virtuels autour duquel s'articule différentes fonctionnalités de consultation, de diffusion et d'analyse. On va s'intéresser aujourd'hui à la partie analyse avec des statistique proposées aux clients.

#### **Les statistiques**

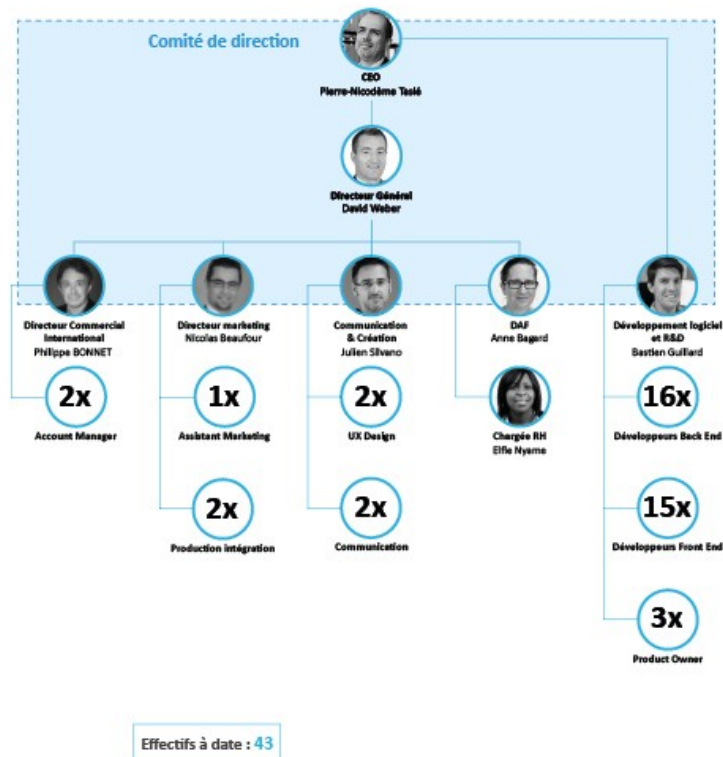
Un accès aux statistiques permet d'avoir une vue sur l'activité de vos médias telle que le nombre de visites sur un documents, quelles pages ont été consultées, combien de temps l'utilisateur est resté sur telle page, le nombre de partage sur les réseaux. Vous retrouverez une liste des statistiques disponibles, proposées aux clients en partie 2.a.

### II.3 *Organisation*

#### **Méthodes agiles, Scrum**

Beebuzziness comprend 48 salariés repartis dans 5 services, l'administration, la production, marketing, la création et le développement. L'organisation de ces services est basée sur les méthodes agiles depuis 2015.

## ORGANIGRAMME 2020



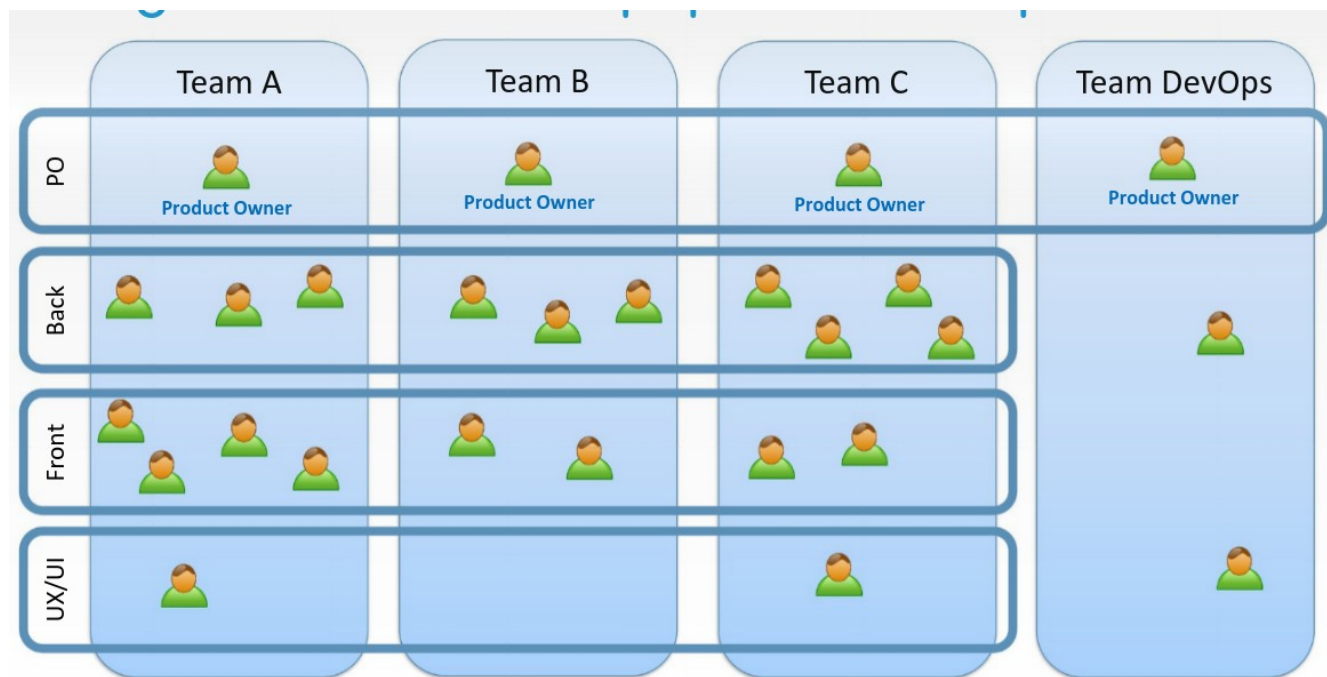
Une **méthode Agile** est une approche itérative et collaborative, capable de prendre en compte les besoins initiaux du client et ceux liés aux évolutions. La méthode Agile se base sur un cycle de développement qui porte le client au centre. Le client est impliqué dans la réalisation du début à la fin 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 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 produit.



## Equipes de développement

Le pôle de développement axé sur la R&D (Recherche et Développement), regroupe 30 employés, 5 contrats d'apprentissage, 25 CDI dont 10 externes. Voici la structure type de ces équipes dirigé par Bastien Guillard notre directeur de développement.



Je suis rattachée à l'équipe en charge des statistiques et du reporting depuis 2017. 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\*.

## III. Ma mission, Le Speed layer parcours détaillé

---

### III.1 Contexte

Je réalise cette mission au sein de l'équipe ISS en charge des statistiques.

Depuis début 2019, l'équipe ISS est chargée de développer une nouvelle version de l'outil de statistiques délivré avec un Hub. J'interviens directement sur le panneau de statistiques dont l'accès se fait via un Hub\*.

Je dois intervenir sur l'architecture Lambda (Figure 5). Elle est composée de 3 niveaux, le Batch Layer (traitement de données par lots), le View Layer et le Speed Layer (traiter le flux des nouveaux événements en temps réel). Ma mission consiste à réaliser ce dernier niveau.

Je travaille sur cette mission avec mon tuteur entreprise qui a conçu un accompagnement sur mesure ainsi que celui des membres de l'équipe. Un développeur me rejoindra sur les phases d'intégration. La durée prévisionnelle de cette mission est de 9 mois soit le temps de l'alternance.

Je dois réaliser et intégrer un sous composant de l'architecture Lambda, le SpeedLayer.

### III.2 L'architecture Lambda

#### Description générale

Le but de l'architecture Lambda est de fournir un modèle de traitement presque 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 pannes, aux erreurs et la traçabilité de la correction des erreurs.
- la scalabilité en gérant de manière indépendante chaque niveau.
- la 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 pré-calculées dans des vues.

### Les composant de l'Architecture lambda

Le système lambda est composé de 3 couches :

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. On va stocker, dans le View Layer des événements dans un format brut toutes les heures en se basant sur les heures pleines. 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 fait en sorte qu'il n'y a 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. Cela permet de minimiser le temps de latence en fournissant des vues en temps réel des données les plus récentes pas encore disponible avec le Batch Layer en récupérant les événements de manière incrémentale.

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 (ou le recalcul) par la couche batch est complété. Elle renvoyant des vues précalculées ou en construisant des vues à partir des données traitées. C'est donc de la donnée grise, qui peut être reconstruite au besoin et qui correspond à une couche serveur pour répondre aux requêtes.

Ceux sont les requêtes qui interrogent les vues pré-calculées dans le View Layer et le Speed Layer pour répondre avec une latence la plus faible possible aux requêtes des applications clientes. La sortie est généralement stockée sur des bases de données NoSQL rapides.

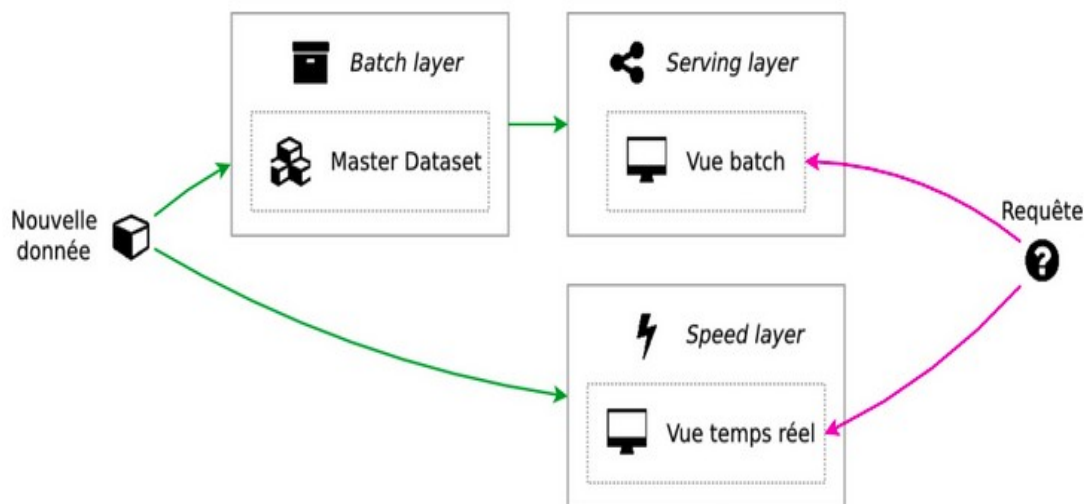


Figure 3: Architecture Lambda simplifiée

### 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 donnée classique : compromis entre la disponibilité et la latence, la robustesse et la cohérence des données ou 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 également robuste avec une tolérance aux pannes et aux erreurs et rapide, avec une latence très faible pour l'écriture des événements, l'exécution des requêtes et donc l'affichage de données même complexes en temps réel. 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 pré-calcule des résultats, les stocke en base, puis interroge cette base pour répondre aux requêtes du demandeur.

Vous trouverez en annexe [trois schémas](#) comprenant les statistiques legacy et l'architecture mise en place à la fin de la mission.

### III.3 Les données

#### Données utilisateurs

#### Données


##### Consultation

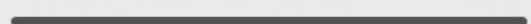
Consultations de médias 1 491 553


Médias consultés au moins une fois 75

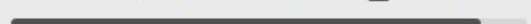
##### Médias les plus consultés




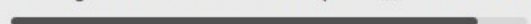
La course aux cadeaux est ouverte  158 407



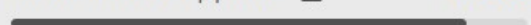
Trouvez toutes vos idées cadeaux  144 080




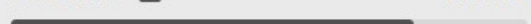
La magie d'être ensemble ! On p...  143 364




Galette des Rois Epiphanie  139 664



Les Soldes  123 633



Médias déposés 32 

[Vous trouverez plus de détails en annexes](#)

### *III.4 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 et 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.

Lodash est une bibliothèque JavaScript réunissant des fonctions 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 permet de transporter et router les messages depuis les producteurs vers les consommateurs en utilisant les échanges et **bindings** pour savoir si il doit délivrer, ou non. Système de file d'attente dans lequel les événements sont stockés temporairement.

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.

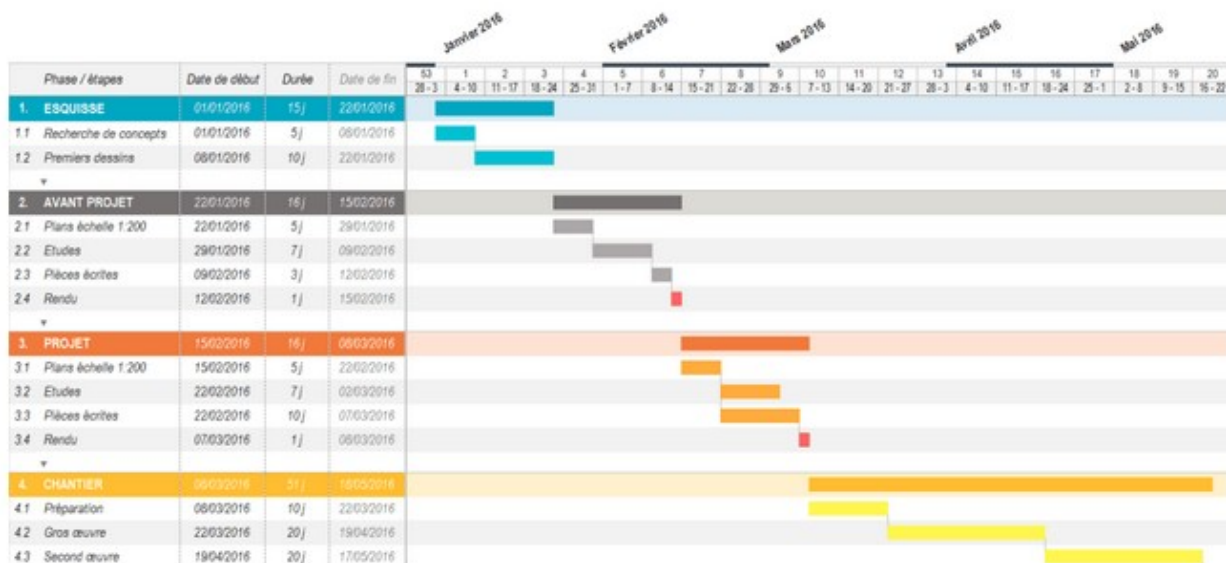
Chai est une bibliothèque d'assertions BDD / TDD qui nous permet de vérifier et comparer la valeur de variables ou de propriétés avec la valeur attendue. Différentes possibilités d'écriture, le style assert ou le style BDD décliné en deux colorations: expect et should.

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.

### III.5 Découpage prévisionnel



© MacroGantt / Softonic

Octobre	Découverte et compréhension de l'architecture Lambda.
Nov-Déc	Écriture des composants de base de ce sous système : <b>aggregation, aggregationManager.</b>
Janvier	Intégration du Speed Layer avec le service de consommation des évènements.
Fin Janv	Intégration du Speed Layer avec le service de consommation des évènements.
Février	Gestion d'un double <b>buffering</b> sur le Speed Layer.

Mars	Création des composants de communication entre le Batch Layer et le Speed Layer.
Avril-Mai	Intégration avec le composant Batch Layer, validation et déploiement en production.
Mi Avril	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.
Bonus	Création d'une tâche Kubernetes pour gérer le Speed Layer.
Fin mission	L'IHM bénéficie des données calculées par le Speed Layer.

### III.6 Bac à sables

### III.7 Implémentation classes du SpeedLayer code

Les responsabilités ont été séparées en différentes classes concernant la gestion des données du SpeedLayer au niveau de l'architecture.

#### SpeedLayerManager

A l'arrivée d'un évènement en temps réel, l'objet de type SpeedLayerManager sélectionne les instances de SpeedLayerAggreion concernées par ce type d'évènement et transmet à chaque instance de la classe SpeedLayerAggreion l'évènement correspondant pour qu'il soit agrégé. Afin de trouver les instances de SpeedLayerAggreion concernées/adéquates on consulte la liste des configuration d'agrégations disponibles du Batch Layer type, il faut aussi que l'option Speed Layer soit activée. Les configuration du Batch Layer type non activées telles que .... nous permettent ... puisqu'elles sont n'ont de sens seulement dans le cadre du Batch Layer.

A partir du moment où ces critères sont respectées on crée une instance de SpeedLayerAggreion et on appelle une nouvelle factory.

#### SpeedLayerAggregation

A l'arrivée d'une instance et de son event dans le SpeedLayerAggreion, elle est transmise au Double Buffer avec ces éléments pour qu'ils soient agrégés. En retour 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 SpeedLayer et les instances du SpeedLayerAggregation associées s'opère à ce niveau là.



## Double buffer

Il permet une gestion de 2 buffers\* de données : current buffer et previous buffer qui gère la synchronisation entre les calculs d'agrégation du BatchLayer et les données stockées par chaque buffer (current et previous). Cette classe mémorise également les dates de début et de fin d'agrégation correspondante dans le batch layer et transmet les événements qui arrivent en temps réel vers le current buffer. Pour chaque requête, elle construit la réponse à cette requête sous la forme de liste de valeurs, en composant les données qui proviennent d'une requête sur le previous buffer et/ou sur le current buffer.

Les buffers contiennent du code générique (pas de code métier) et permettent seulement de stocker des informations, on a pas besoin à ce niveau là d'avoir des notions de types d'agrégations. Pour optimiser l'espace mémoire on compacte la donnée. Le DoubleBuffer a la notion de previous, current et date, cela permet de dispatcher les informations regroupées dans le Buffer entre le previous et le buffer en fonction de l'évolution du Batch Layer grâce aux dates.

## Buffer

La classe agrège des données pour un bucket (une période) et un type d'évènement donné en temps réel. Pour chaque requête sur le buffer (current ou previous), on construit une liste de résultat de valeurs agrégées pour chaque propriété. La synchronisation entre le calcul des agrégations dans le SpeedLayer et les instances du SpeedLayerAggregation associées s'opère à ce niveau là.

## Schéma de classes

Une fois récupérée l'instance avec l'évènement qui a été créé dans le M et transmet au SA puis DB( qui lui transmet en plus des autres classes une notion de temps), cette classe agrège des données pour un bucket (une période) et un type d'évènement donné en temps réel.

**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 SpeedLayer car on ne veut pas les conserver dans le temps, le travail se fait au travers de données éphémères. On travaille sur maximum 2 heures et plus précisément, sur les deux dernières heures, l'heure précédente et l'heure courante.

Lorsque que 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

évoque 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).

### *III.8 Interfaces*

Une interface définit un ensemble de méthodes sans leurs implémentations et sera tenue d'utiliser les méthodes de celle-ci. Permet de définir des contrats.

**Interface avec le BatchLayer**

**Interface avec le Querylayer**

**Interface avec d'Initialisation**

### *III.9 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.

La démarche à suivre pour mettre en place cette méthode est décomposée en trois phases appelé RGR (aussi appelé la Mantra). Les deux premières phases sont nommées d'après la couleur de la barre de progression dans les outils de test unitaires comme Chai (Red pour échec et Green pour réussite) :

- R (Red): écrire un code de test et les faire échouer,
- G (Green) : écrire le code métier qui valide le test,
- R (Refactor) : remaniement du code afin d'en améliorer la qualité.

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

```
describe('aggregateEvent', () => {
  const countOpenMediaAggregationSetting = aggregationSettingsWithOneResult;
  it('aggregate one event ie(cad) count the events "openMedia"', () => {
    const speedLayer = aggregationFactory(countOpenMediaAggregationSetting);
    const { doubleBufferForTest } = createStubDoubleBufferFactory(speedLayer);
    const spyDoubleBufferIncrProperty = sinon.spy(doubleBufferForTest,
'incrProperty');
    const eventOpenMedia: IOpenMediaEvent & IEvent = {
      timeStamp: 1,
      sessionId: 'session1',
      eventType: EventType.openMedia,
      libId: 'libId1',
      mediaId: 'mediaId1',
      documentId: 'documentId1',
      mediaType: MediaType.beevirtua,
      context: StatisticsContext.player,
    };
    const aggregationsStep1: IEventAggregation =
speedLayer.aggregateEvent(eventOpenMedia);
    sinon.assert.calledOnce(spyDoubleBufferIncrProperty.withArgs({ libId:
'libId1', mediaId: 'mediaId1' }, 'count'));
    expect(aggregationsStep1).toEqual({
      count: 1,
    });
  });
});
```

```
public aggregateEvent(event: IEvent): IEventAggregation {
  const { eventKey, output } = this.settings;
  const keysAsStringPrefix = _.join(eventKey.map(key => _.get(event, key)),
'-');
  const outputKeys = Object.keys(output);
  // outputKeys.forEach((outputKey) => {
  //   const keysAsString = keysAsStringPrefix.concat('-', outputKey);
  //   this.currentBucketResult[keysAsString] =
  //
  this.computeAggregationValue(this.currentBucketResult[keysAsString],
output[outputKey].aggregator, outputKey, event);
  // });
  // return this.currentBucketResult;
  return {};
}
```

explication code

### *III.10      Intégration*

Point d'entrée ViewQuery qui va aller chercher les infos dans le service client nommé bee-stat-speedlayer.

Le point d'entrée du front côté application se fait au niveau du panneau de stat et coté code se passe au niveau du fichier ViewQuery.ts avec ces params de query, de filters et de dates elle va appeler soit le B. soit le S ou les 2 dans ce cas le merge des 2 parties de se fera ici. Ex : si la requête concerne des events passés on a besoin seulement du B. dans le cas contraire on aura besoin de consulter les calculs à temps réel.

Création d'un nouveau service pour l'intégration des 3 interfaces, l'agrégation des événements, la gestion des requêtes et la synchronisation des calculs BatchLayer / SpeedLayer.

**Configuration**

**Application**

**Consumer**

**Tests d'intégration et d'application**

On va créer un nouveau service, bee-stat-speedlayer qui contiendra la partie consumer, et bdd.

## IV. Conclusion

---

### Objectifs atteints / à venir

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 concerne 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 le 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 ...

## V. Glossaire

---

[1] Un **Hub** : Espace de stockage, consultation et diffusion de médias en ligne.

[3] **Médias** : Un type de documents, images, vidéos ou liens.

[4] **Méthode agile** : Processus itératif et adaptatif où les équipes travaillent dans une série de cycles courts, incorporant un retour d'information rapide, pour livrer des solutions adaptées aux besoins des clients.

[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] **Ubstream** : 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] **ISS** : Nom de l'équipe dans laquelle je me trouve. ISS signifie Interactive Smart Signage et correspond au premier projet réalisé par cette équipe qui consistait à réaliser une application de diffusion de contenus interactifs à partir des médias stockés dans un Hub. Le nom de l'équipe n'a pas changé même si elle est désormais en charge des statistiques.

[0] **Digital signage** : Application de diffusion de contenus interactifs, accessible via un Hub\* et faisant partie de l'écosystème Ubstream\*. 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] **TDD** : Test Driven Development (*Développement Dirigé par les Tests*), est une technique de développement qui impose l'écriture de tests avant même l'écriture de la première ligne de code.

[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] **ZZO** :

[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] **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.



## VI. Bibliographie

---

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 :

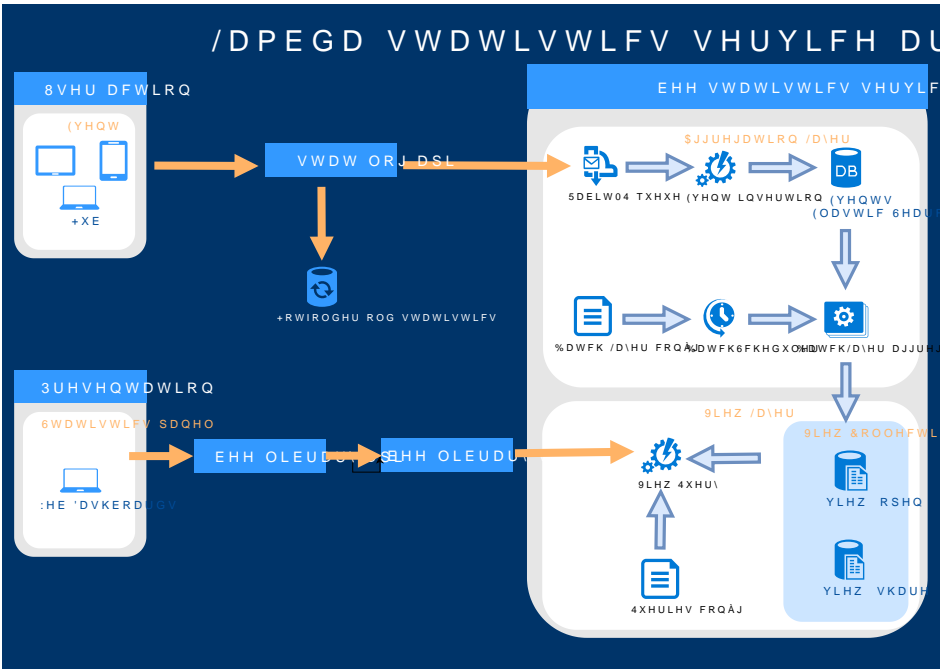
<https://www.journalducsm.com/metier-ux-designer/>

Sinon.js

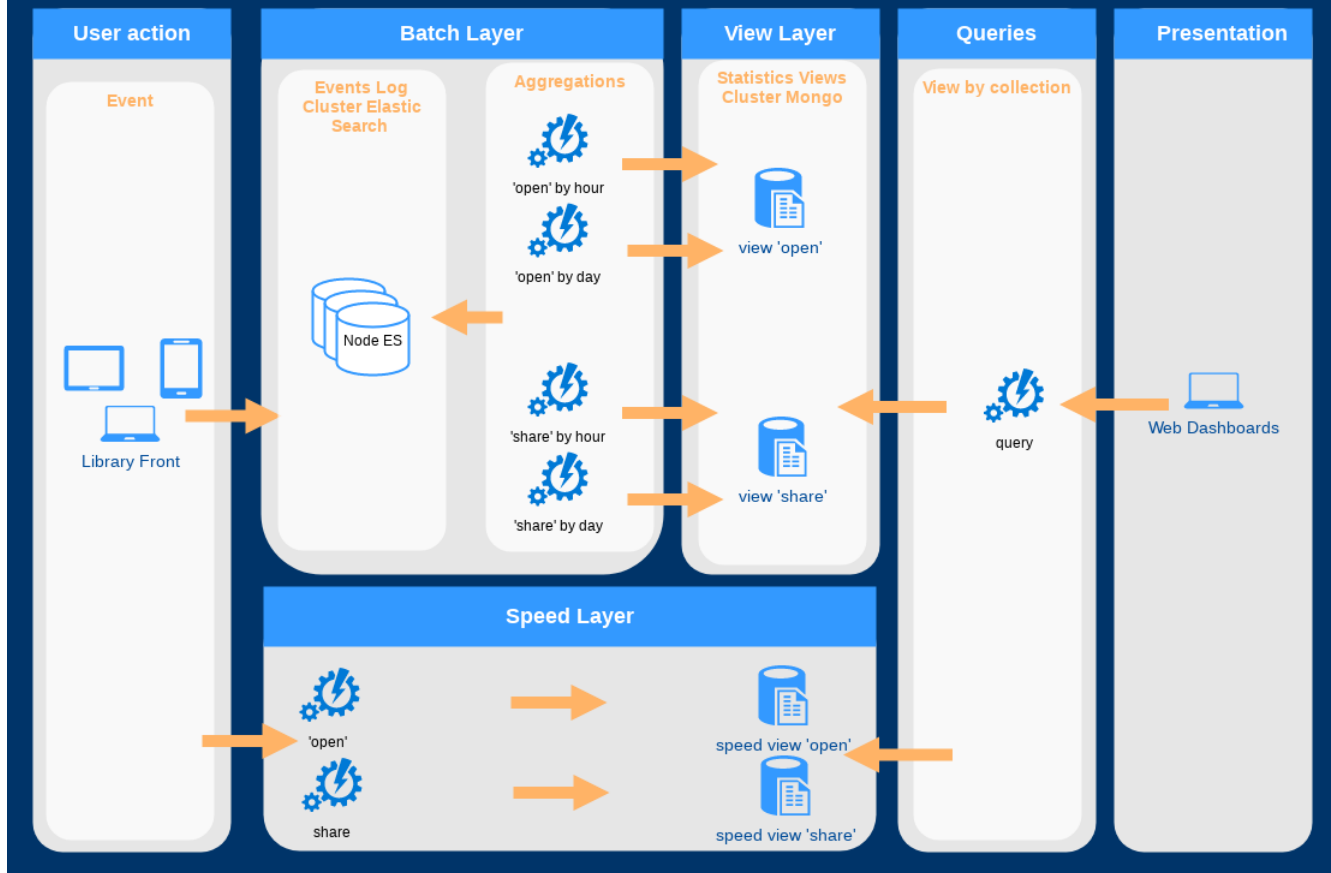
<https://www.editions-eni.fr/open/mediabook.aspx?idR=39ae4beb4e4edad9da04dfda4034eb36>

# VII. Annexes

## Architecture L amba évolutions

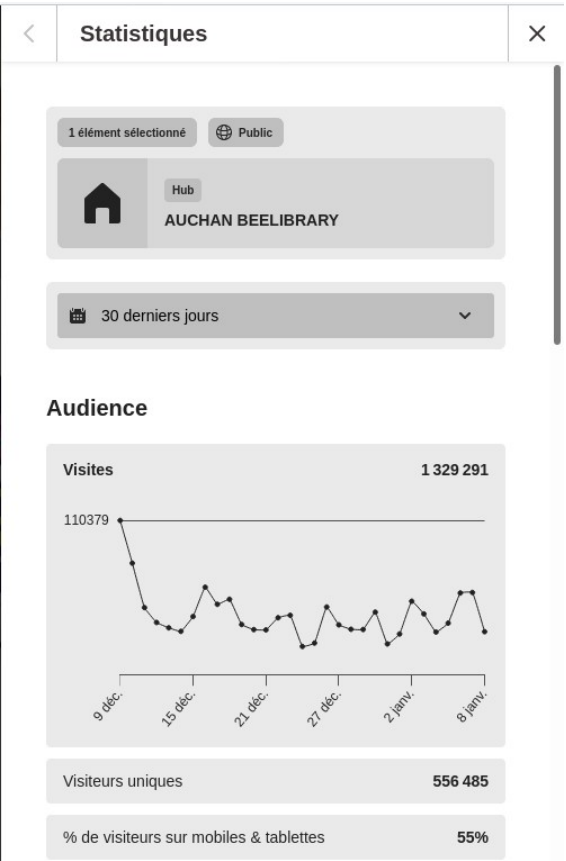
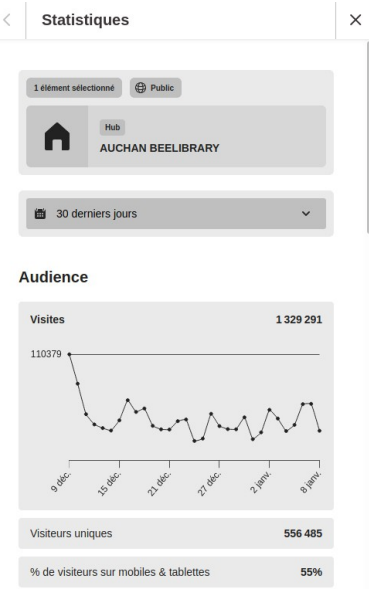
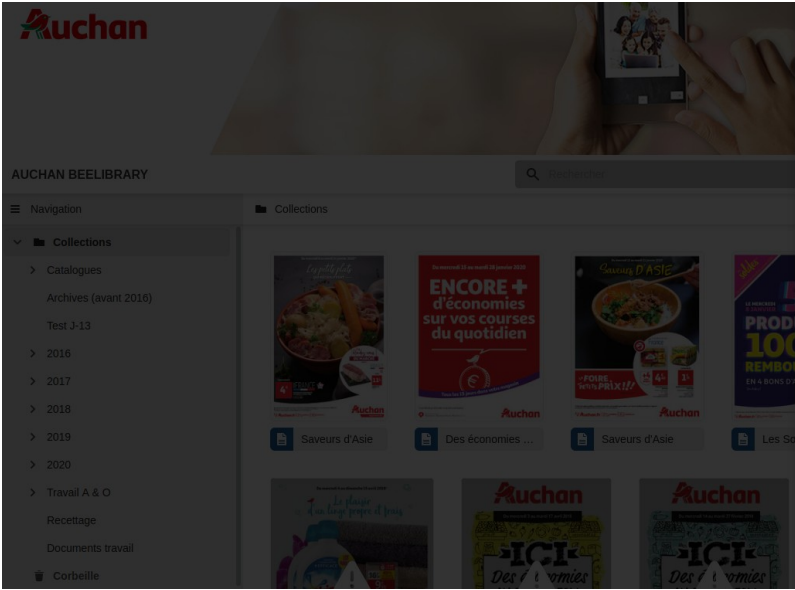


# Lambda architecture

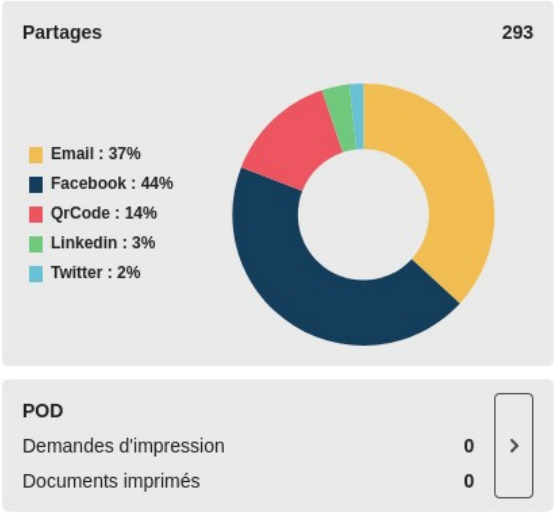


[Retour chapitre archi Lambda](#)

Accès aux statistiques dans un Hub :



Engagement



[Retour chapitre données](#)

## Cheminement des évènements à un instant T

Ce fichier excel permet de visualiser les différents cas possible lorsqu'un event est déclenché / pris en charge en détaillant les différents "status/comportement" de B et du S. à un moment donné.

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 [

////////// Cheminement event

On va s'intéresser à l'évènement OpenMedia

On peut regrouper les colonnes en différentes catégories :

1-2 = Temps qui passe

3-5-6 = Etat du B. et du S. à l'instant T

4 = events du B. envoyés au speed

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

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'event au S.

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

Colonne 6 : Puisque le previous a travaillé de 8h à 9h

=> Event pris en charge par le current SpeedLayer

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

=> 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

=> 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. a terminé le traitement jusqu'à 10h inclus et envoie l'événement BL computed au S. ce qui va "faire sauter" le previous et travailler 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

=> Event pris en charge .... le current S. tjrs

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 [	[ 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 [	BL Computed	[ 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[	

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

=> Event pris en charge .... le current S. tjrs

Colonne 1-2 : Lorsque l'évènement 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 événement car il travaille de 10h à maintenant  
Colonne 6 : Le previous a travaillé de 9h à 10h

=> Événement pris en charge .... le current S. tjrs

La query est combien de médias ont été consultés sur les 30 derniers jours :

On peut catégoriser les colonnes :

A/B = temps qui passe

C/E/F = B. et S.

D = événement du B. envoyé au speed

G/H/I = query

////////// Cheminement requête à 9h59

Colonne 1 : Une requête est lancée 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 aujourd'hui 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

////////// 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 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'événement 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.

////////// 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 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

////////// Cheminement requête à 10h02

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

Colonne 2 : La période concernée est de J-29 0h00 à auj 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. evoie l'event BL computed au S. ce qui permet de ne plus se baser sur le previous puisque le B. a pris le relais

////////// Cheminement requête à 10h15

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

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

////////// Cheminement requête à 10h59

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

Colonne 2 : La période concernée est de J-29 0h00 à auj 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, situé à Grenoble, propose une solution globale de communication numérique qui combine les avantages du numérique et de la rematérialisation à la demande en proposant Ustream, une plateforme regroupant toutes les fonctionnalités nécessaires à la gestion et à la diffusion des contenus virtuels.

Ma mission est liée aux statistiques puisque mon équipe est en charge, depuis début 2019, de réaliser une nouvelle version de l'outil de statistiques. Le nouveau système est basé sur l'architecture Lambda et j'interviens sur ce projet. Ma mission consiste à réaliser le Speed Layer, la dernière pièce manquante de l'architecture Lambda.

La mission se déroule en 3 phases :

- \* analyse et compréhension du système Lambda dans son ensemble,
- \* conception et écriture du code du Speed Layer, vu comme un système autonome,
- \* l'intégration du composant dans le système avec les deux autres composants de l'architecture Lambda.

La nouvelle version des statistiques permet d'ajouter facilement de nouvelles informations d'analyse en fonction des demandes clients sans coût supplémentaire pour l'entreprise.

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.

Ubstream, 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 : Ubstream, 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 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.