MÉMOIRE DE FIN DE FORMATION

Comment visualiser une communauté dans un contexte d'éco-fidélisation

Présenté par Kylian ROUVEURE

soutenu le 03/06/2024

letmotiv
Engage your community

Entreprise : letmotiv

Tuteur Entreprise : Steven TITREN, CTO letmotiv

David THOIRON, COO letmotiv

Tuteur IUT : Hélène CHANVILLARD, Enseignante

Candide : Antoine ROLLAND, Enseignant

BUT 3 - Science des Données

Année Universitaire 2023 – 2024

MÉMOIRE DE FIN DE FORMATION

Comment visualiser une communauté dans un contexte d'éco-fidélisation

Quelles innovations peuvent être mises en place pour industrialiser le développement des outils de pilotage pour l'éco-fidélisation ?

Présenté par Kylian ROUVEURE

AVERTISSEMENT

IUT Lumière Lyon 2, composante de l'Université Lumière Lyon 2, n'entend donner aucune approbation ni improbation aux opinions émises dans les mémoires des candidats aux DUT en alternance : ces opinions doivent être considérées comme propres à leur auteur.

Tenant compte de la confidentialité des informations ayant trait à telle ou telle entreprise, une éventuelle diffusion relève de la seule responsabilité de l'auteur et ne peut être faite sans son accord.

DÉCLARATION ANTI-PLAGIAT

Ce travail est le fruit d'un travail personnel et constitue un document original. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.

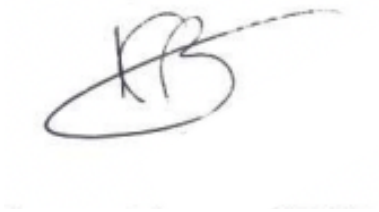
Je m'engage sur l'honneur à signaler, dans le présent mémoire, et selon les règles habituelles de citation des sources utilisées, les emprunts effectués à la littérature existante et à ne commettre ainsi aucun plagiat.

NOM : Rouveure

Prénom : Kylian

Date : 26/05/2024

Signature :



Remerciements

Je souhaite exprimer ma gratitude à toutes les personnes qui ont joué un rôle déterminant dans la réussite de mon alternance et dans la rédaction de ce mémoire.

Tout d'abord, je tiens à remercier mes tuteurs en entreprise, Steven Titren et David Thoiron, pour leurs enseignements, leur patience et leur suivi constant. Leur soutien m'a permis d'acquérir des compétences essentielles, tant sur le plan technique que managérial.

Je remercie également Madame Hélène Chanvillard pour son suivi rigoureux de mon alternance et pour ses conseils avisés, qui ont été importants dans l'accomplissement de ces deux ans.

Enfin, je suis reconnaissant envers toute l'équipe de Letmotiv pour leur accueil chaleureux, leurs conseils et la confiance qu'ils m'ont accordée en me permettant de travailler en autonomie sur beaucoup projets. Grâce à eux, j'ai pu me construire une expérience solide et développer des compétences de qualité, en apprenant de mes échecs comme de mes succès.

Sommaire

Glossaire

1. **Néo-fidélisation** • Concept de fidélisation qui s'appuie sur l'utilisation de technologies avancées et d'une approche multicanale pour interagir avec les clients, améliorant ainsi leur engagement et leur satisfaction.

2. **Éco-fidélisation** • Stratégie de fidélisation axée sur l'incitation à des comportements respectueux de l'environnement, où les clients sont récompensés pour leurs actions écoresponsables, contribuant ainsi à réduire leur empreinte environnementale tout en favorisant un engagement durable.

3. **Responsabilité Sociale des Entreprises (RSE)** • Démarche adoptée par les entreprises pour intégrer les préoccupations sociales, éthiques et écologiques dans leurs activités commerciales et leurs interactions avec les parties prenantes, visant à améliorer leur impact global.

4. **B2C (Business to Consumer)** • Modèle commercial direct où les entreprises vendent leurs produits ou services directement aux consommateurs, en utilisant divers canaux de distribution pour toucher un large public.

5. **B2B (Business to Business)** • Modèle commercial où les transactions de produits ou services se font entre entreprises.

6. **Data développeur** • Rôle spécialisé dans la création et la gestion de systèmes de données, incluant le développement de rapports et d'outils d'analyse, pour fournir des analyses stratégiques qui aident les entreprises à optimiser leurs décisions basées sur des données concrètes.

7. **Reporting** • Pratique consistant à élaborer des rapports détaillés qui synthétisent les données opérationnelles, financières ou de performance. Ces rapports sont essentiels pour évaluer l'efficacité des stratégies de fidélisation et pour ajuster les campagnes marketing en conséquence.

8. **Interactions clients** • Ensemble des échanges et communications entre une entreprise et ses clients, englobant tous les points de contact qui peuvent aller de la promotion au support client, en passant par les ventes et le service après-vente.

9. **Multicanal** • Approche qui consiste à interagir avec les clients à travers plusieurs canaux de communication et de vente, tels que le web, le mobile, le magasin physique, les réseaux sociaux, etc., offrant une expérience client cohérente et intégrée.

10. **Laravel PHP** • Framework de développement PHP open source, très apprécié pour sa capacité à faciliter des tâches complexes de programmation web grâce à une architecture bien structurée et un code propre, ce qui en fait un outil idéal pour des applications web robustes.

11. **Vue JS** • Framework JavaScript progressif utilisé pour construire des interfaces utilisateur. Il est conçu pour être incrémentalement adoptable, ce qui le rend particulièrement efficace pour le développement de single-page applications et des interfaces riches et interactives.

12. **Pinia JS** • Librairie de gestion de variables pour Vue JS, conçue pour être intuitive et flexible.

13. **Tailwind CSS** • Framework CSS qui permet de styler rapidement les applications web en utilisant des classes utilitaires directement dans le code HTML. Il est prisé pour sa flexibilité et sa capacité à accélérer considérablement le développement front-end en évitant de rédiger des CSS complexes.

14. **Dataviz (Visualisation de données)** • Pratique consistant à représenter des ensembles de données sous forme graphique pour en faciliter la compréhension et l'analyse.

15. **Widgets** • Composants d'interface utilisateur qui affichent des informations ou fournissent une fonctionnalité spécifique dans le contexte d'une application plus large.

16. **KPI (Indicateur clé de performance)** • Mesure évaluative utilisée pour juger de la performance d'une activité par rapport à des objectifs prédéfinis.

17. **Framework** • Ensemble cohérent de bibliothèques logicielles et de conventions destinées à aider le développement d'applications en fournissant une structure prête à l'emploi, afin de promouvoir des pratiques de codage uniformes et efficaces.

18. **Package** • Collection de modules ou de codes prêts à l'emploi qui peuvent être intégrés dans une application pour ajouter de nouvelles fonctionnalités sans nécessiter un développement à partir de zéro.

19. **NoSQL** • Type de base de données permettant de stocker et récupérer des données de manière flexible et scalable.

20. **CRUD (Create, Read, Update, Delete)** • Opérations de base permettant de manipuler les données dans une base.

21. **Endpoint** • URL spécifique permettant de récupérer ou envoyer des données vers un serveur à travers une API.

22. **Filtre** • Outil permettant de segmenter et sélectionner des données spécifiques pour une analyse plus fine.

23. **Séries temporelles** • Données chronologiques permettant d'analyser des tendances et des variations dans le temps.

24. **Industrialisation** • Transformation de processus manuels en processus automatisés pour améliorer l'efficacité, la qualité et la scalabilité des opérations.

25. **Rationalisation** • Processus visant à rendre plus efficace et économique la gestion des ressources, en simplifiant et optimisant les procédures et les structures. Diffère de l'industrialisation en se concentrant sur l'efficacité et la simplicité plutôt que sur l'automatisation.

26. **Segmentation** • Processus de division des données en sous-groupes pour en faciliter l'analyse et la compréhension.

27. **Scalabilité** • Capacité d'un système à gérer une augmentation du volume de travail ou de données sans compromettre ses performances.

28. **Paradigme** • Modèle ou cadre de référence qui influence la façon dont un problème est compris et abordé.

29. **Click-and-drop** • Méthode de création d'interfaces utilisateur en utilisant une méthode de glisser-déposer des éléments sans écrire de code.

30. **Outils SaaS** • Logiciels disponibles en tant que services sur le cloud, permettant aux utilisateurs d'accéder et de travailler sur des applications via Internet sans avoir à les installer ou les créer.

Introduction

Dans le contexte actuel marqué par une transformation digitale intense, chaque entreprise moderne se trouve confrontée à un défi majeur: comprendre et motiver sa communauté, que ce soit ses clients (B2C)[4], ses partenaires (B2B)[5] ou ses propres employés. Ce défi est au cœur de l'activité de letmotiv, une start-up lyonnaise spécialisée dans le webmarketing et fondée en 2017. Initialement concentrée sur la néo-fidélisation[1] multicanale[9], elle a pris un virage stratégique en 2023 vers l'éco-fidélisation[2], enrichissant ainsi son offre pour répondre aux nouvelles pratiques des consommateurs soucieux de leur impact environnemental.

Les systèmes de fidélisation développés par letmotiv visent à :

1. Maximiser les ventes,
2. Enrichir la connaissance client,
3. Améliorer l'image de marque,
4. Dynamiser les communautés.

L'adoption d'un système de fidélisation permet ainsi d'automatiser et de moderniser les interactions avec les clients ou collaborateurs grâce à des missions sur-mesure visant à atteindre un ou plusieurs de ces objectifs. L'éco-fidélité ajoute une dimension supplémentaire en récompensant les gestes écologiques de leurs utilisateurs, contribuant ainsi à l'amélioration de la Responsabilité Sociale des Entreprises (RSE)[3], un aspect de plus en plus valorisé par les consommateurs.

L'intégration de l'éco-fidélisation ne répond pas seulement à un besoin de durabilité, mais transforme aussi notre façon d'engager les clients d'une marque. En récompensant des actions écologiques, nous souhaitons renforcer les initiatives environnementales de nos utilisateurs, et ainsi renforcer l'engagement pour des habitudes durables.

En tant que data développeur[6], mon rôle consiste à concevoir et développer des reportings pour les différentes plateformes de fidélité que nous vendons à nos clients. Ces outils de pilotage sont essentiels, car ils permettent d'analyser l'efficacité des investissements, l'activité des plateformes et d'ajuster les campagnes pour optimiser les interactions clients[8]. Néanmoins, avec l'arrivée de l'éco-fidélisation, mon rôle prend un nouvel aspect avec la visualisation des impacts écologiques des missions. Cependant, développer nos reportings sur les technologies que nous utilisons chez letmotiv, telles que Laravel PHP[10] et Vue JS[11], complexifie la tâche. Coder les tableaux de bord directement nous permet de réaliser précisément ce que nous voulons en

termes de data visualisation et nous garantissant une restitution de qualité haut de gamme. Toutefois, cela nous confronte à des problèmes importants. Le codage de reporting sur-mesure est une tâche chronophage et complexe dans la programmation et la configuration, d'où des délais de développement allongés.

Face à cette complexité et à la maturité croissante des missions, il est devenu nécessaire d'optimiser et d'industrialiser ce type de développement. Ce mémoire abordera donc la problématique suivante :

Quelles innovations peuvent être mises en place pour industrialiser le développement des outils de pilotage pour l'éco-fidélisation ?

Le mémoire vise à expliciter les objectifs, l'impact et les méthodologies utilisés pour industrialiser les développements des outils de pilotage au sein de letmotiv. En soulignant, les problématiques que nous avons rencontrées lors de la création des reportings clients et en mettant en lumière les améliorations obtenues grâce à cette optimisation. Je vais aussi montrer comment ces développements ont renforcé nos relations avec les clients et ont fourni un avantage concurrentiel, ouvrant de nouvelles perspectives stratégiques pour l'entreprise.

1/ Les défis de production des reportings clients

Pour comprendre pleinement les défis auxquels letmotiv a dû faire face au cours des deux dernières années, il est primordial de se pencher sur la complexité croissante des reportings[7] clients qui nécessitent des ressources et des compétences spécifiques. En effet, la maturité de ces outils a permis de mettre en lumière des problématiques essentielles. Leur résolution offre des avancées considérables en termes de productivité et de compétitivité. Dans cette section, nous allons explorer les défis liés à la production des reportings sur les plateformes en ligne, en retraçant le parcours réalisé, en évaluant les méthodologies mises en place, leurs limites, et les évolutions incontournables à adopter.

1.1 - Métriques et outils précédemment utilisés

Depuis sa création et jusqu'à mon arrivée, letmotiv s'est appuyé sur des systèmes de reporting classiques. Les premiers tableaux de bord, conçus pour la simplicité, pouvaient contenir entre deux et quatre widgets[15], se concentrant sur des métriques élémentaires telles que le nombre, le nombre distinct, la somme et la moyenne. L'objectif principal était de fournir des informations directes et facilement compréhensibles, sans insister sur une analyse croisée des données pour une compréhension plus profonde.

Pour répondre à ces besoins initiaux, un package[18] spécifique avait été développé. Ce package s'est révélé très pratique au début, mais a montré ses limites avec le temps. L'un des principaux avantages des frameworks[17] comme Laravel PHP est leur capacité à faciliter la gestion et le développement de ces packages, permettant leur intégration rapide dans divers projets grâce à quelques commandes. Cette facilité d'intégration marque un premier pas vers l'industrialisation[24] de nos produits[(1)].

---

[1] Clouddevs, "Laravel Packages: Extending Functionality with Third-Party Libraries.", URL : https://clouddevs.com/laravel/packages/

Durant ma première année d'alternance, les tableaux de bord se sont graduellement complexifiés, évoluant vers des configurations plus élaborées qui permettent un meilleur pilotage grâce à des visualisations croisées et un filtrage efficace des données. Cette transformation est illustrée par les captures d'écran suivantes, montrant l'évolution des tableaux de bord, passant d'une configuration simple (*Figure 1 - Reporting simple*) à une autre plus complexe (*Figure 2 - Reporting complexe*).

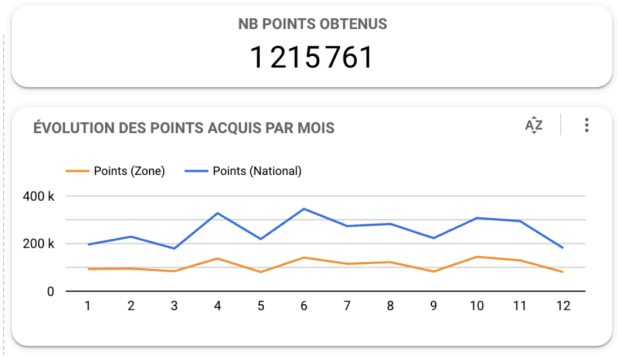


Figure 1 - Reporting simple

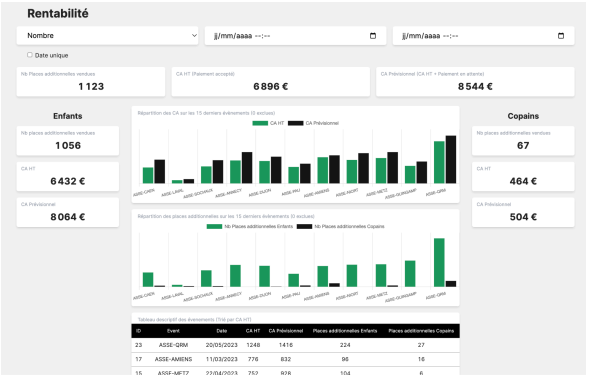


Figure 2 - Reporting complexe

Pour répondre aux exigences fonctionnelles croissantes et à la complexification des tâches, j'ai adopté une méthodologie structurée pour le développement des reportings. Ce processus débute par la création de maquettes préliminaires avec Google Data Studio permettant de visualiser de manière globale les informations requises par les clients et leur représentation graphique avant le développement. Après validation des maquettes par le client, nous procédons au développement des tableaux de bord sur leur plateforme de gestion. Ce plan préventif évite de débuter trop tôt la production et le risque de devoir tout recommencer si les résultats ne correspondent pas aux attentes du client.

Sur le plan organisationnel, cette nouvelle méthodologie nous a permis d'établir une première normalisation dans la production des tableaux de bord sur mesure. Toutefois, les défis techniques et de production n'ont pas tardé à se manifester, compromettant la conservation de ces nouvelles compétences par letmotiv.

1.2 - Limitation de production

Le premier défi lié à la production des tableaux de bord concerne l'augmentation notable du temps de développement nécessaire. Initialement, il était possible de concevoir un reporting en une journée. Cependant, avec l'évolution des exigences et la complexité croissante des

tableaux, ce délai s'est étendu jusqu'à deux à trois semaines. Cette augmentation significative du temps de développement n'est pas seulement un problème de coût, mais aussi de réactivité. Face à des délais allongés, la capacité de l'entreprise à répondre rapidement aux besoins des clients s'en trouvent réduite, ce qui peut affecter la satisfaction client et la compétitivité sur le marché.

Le deuxième défi réside dans la complexité technique associée à la création de ces tableaux de bord complexes. Bien que le résultat final soit fonctionnel et visuellement agréable, le processus de développement est devenu si spécifique qu'il nécessite des compétences spécialisées pour concevoir et maintenir ces types de tableaux. Cette dépendance vis-à-vis d'une compétence spécifique crée un blocage, remettant en question la viabilité à long terme de ces outils au sein de l'entreprise.

Enfin, le troisième défi concerne le temps de réponse des tableaux. L'intégration de fonctionnalités avancées, telles que les widgets interactifs, les options de filtrage et les jointures de données, a conduit à un allongement notable des temps de réponse. Bien que les temps de chargement restent acceptables (entre 3 et 5 secondes) et que Laravel PHP gère efficacement ces données à travers ses modèles et ses systèmes de mise en cache, letmotiv est conscient qu'avec le développement l'éco-fidélité et à la croissance de l'entreprise impose une réflexion sur la sobriété informatique par l'optimisation de la gestion des requêtes pour le reporting. Ce constat nous oriente vers une réflexion sur l'adoption d'un paradigme No SQL pour améliorer la performance et la scalabilité[27] de nos données.

Ces enjeux ont été cruciaux pour la suite de mon alternance et pour letmotiv. Ils posent des questions stratégiques importantes sur la conservation de ces compétences en reporting, notamment en envisageant la possibilité de mon départ et cette interrogation : letmotiv a-t-elle intérêt à maintenir ces fonctionnalités, étant donné qu'elles pénalisent l'entreprise avec des délais de développement prolongés et une accessibilité utilisateur limitée, voire inexistante ?

1.3 - Les besoins de letmotiv et de ses clients

Avant de présenter les solutions mises en œuvre, il est essentiel de définir clairement les besoins de chaque partie, en alignant le design des tâches sur les ressources que letmotiv peut raisonnablement engager.

Deux axes principaux de rationalisation[25] se dégagent des problématiques précédemment évoquées. Le premier, et le plus crucial, est la création de tableaux de bord. Le second concerne la migration automatique des données. Il est crucial de souligner que, en tant que PME aux moyens limités, chaque ressource chez letmotiv est précieuse. Le reporting, bien qu'important, n'est pas une priorité immédiate, car il ne contribue pas directement à la construction du produit. Il est donc contre-intuitif de mobiliser une ressource pendant 2-3 semaines sur une même tâche, un luxe que seules les entreprises plus grandes peuvent se permettre.

Pour la création de reportings, letmotiv a besoin d'une solution pérenne, qui ne requiert pas de compétences spécialisées pour fonctionner, et qui soit facile à utiliser et à prendre en main. L'objectif est qu'un reporting ne prenne pas plus de quelques jours à être développé. Il serait également bénéfique de pouvoir concevoir les maquettes en définissant directement les structures, titres, widgets et types de graphiques des indicateurs. À court terme, l'idéal serait d'éliminer l'étape du Google Data Studio pour les petits projets, afin de gagner encore un peu plus de temps. En résumé, l'objectif n'est pas nécessairement d'adopter des solutions de type click-and-drop[29] ou des outils SaaS[30] sans contraintes de codage, mais plutôt de permettre à tout développeur de proposer des reportings, sans restreindre ces développements à un développeur spécialisé en data.

Pour concevoir correctement la solution, il est crucial de comprendre les besoins de nos clients afin de proposer une solution adaptée le plus rapidement possible. Du point de vue de nos clients, les analyses statistiques poussées ne sont pas cruciales et ne constituent pas un argument de vente déterminant pour notre solution. La solution doit donc privilégier la polyvalence des graphiques et la possibilité de les combiner. Nous nous adressons principalement à des administrateurs et non à des spécialistes de la data, il est donc impératif de proposer des widgets à la fois agréables et très compréhensibles. Une forte demande existe également pour les tableaux dynamiques offrant des fonctionnalités telles que le clic pour le tri de colonnes.

Les besoins des clients, comme nous l'avons vu, ne sont pas nécessairement complexes en termes de dataviz[14]. C'est aussi une raison pour laquelle le précédent package développé par letmotiv a bien fonctionné. Nos clients recherchent des informations faciles à comprendre et une accessibilité des tableaux d'analyses pour les examiner en interne. Cependant, leurs demandes ont évolué vers des interfaces plus avancées, mais toujours fondées sur les mêmes besoins de base, tels que la répartition des données, les calculs simples (chiffre d'affaires, nombre d'utilisateurs, somme des missions accomplies), des graphiques accessibles, des tableaux riches et des KPI[16] simples. Leur intérêt réside maintenant dans le croisement visuel des sources de données.

Enfin, pour la scalabilité des données, les clients ont besoin que la génération des analyses soit plus efficace. Pour letmotiv, cela implique une mise en place rapide avec des fonctions prêtes à être intégrées dans une routine quotidienne. L'objectif est de créer un automatisme dans le traitement des données par des routines, réduisant ainsi les ressources nécessaires pour les requêtes en effectuant les calculs en amont, ce qui permet de rendre les informations plus rapidement accessibles aux utilisateurs.

2/ Processus d'industrialisation et optimisation de la business intelligence

Après avoir identifié les problèmes et les besoins dans la production des reportings, nous abordons maintenant la phase technique visant à répondre efficacement à ces demandes. Cette section détaille le processus de planification, de décision et d'implémentation des solutions optimisées pour les outils de letmotiv.

2.1 - Cahier des charges techniques des solutions

Pour développer une version bêta de notre solution, il est crucial de définir un cahier des charges techniques précis. Ce document doit formaliser les attentes et les exigences techniques, et établir un cadre clair pour le projet. Voici les éléments principaux de notre cahier des charges :

Libertés accordées :

1. **Logiciel**, aucune restriction quant au choix du logiciel, à condition que la solution fonctionne efficacement et soit sécurisée.
2. **Possibilité d'externalisation**, à condition que les résultats puissent être intégrés via une API ou des iframes, facilitant ainsi l'interactivité avec nos systèmes existants.

Critères essentiels :

1. **Utilisation simple et viable**, elle doit être facile à utiliser pour tous les membres de l'équipe, indépendamment de leur niveau de compétence technique, afin de permettre une adoption rapide et large.
2. **Qualité du rendu**, il doit être aussi qualitatif que ce qui pourrait être développé en interne, garantissant ainsi la continuité et la qualité des processus décisionnels basés sur les données produites.
3. **Coût**, l'investissement doit être en adéquation avec les bénéfices potentiels. Il est essentiel que le coût total reste dans un budget raisonnable pour une PME, assurant un bon retour sur investissement.

En structurant notre cahier des charges avec ses contraintes, nous nous assurons de créer une solution qui non seulement répond efficacement à nos besoins, mais qui est également durable

et évolutive. Ce cadre permettra de guider le développement de la solution tout en restant flexible face aux besoins changeants de l'entreprise et ses clients dans le futur.

2.2 - Externalisation ou développement en interne ?

La question de choisir entre l'externalisation de la création de tableaux de bord ou de leur développement en interne est cruciale, surtout pour une startup comme la nôtre, où le temps et les ressources sont limités et précieux.

Voici les principaux avantages et inconvénients que nous avons pris en compte pour trancher cette question :

	Développement en interne	Externalisation
Avantages	• Contrôle complet sur la personnalisation et l'intégration avec les systèmes existants • Amélioration des compétences internes et de la capacité à réagir rapidement aux changements de besoins • Moyen terme, possibilité d'amortir complètement l'investissement	• Accès à une expertise spécialisée • Réduction des coûts à long terme liés à la maintenance et aux mises à jour • Délais de développement potentiellement plus courts
Inconvénients	• Coûts initiaux plus élevés pour le développement et la formation • Risque de rallonger les délais de développement si les ressources ne sont pas suffisamment qualifiées	• Baisse en qualité, car moins sur-mesure • Moins de contrôle sur les processus de développement • Dépendance vis-à-vis du fournisseur pour les mises à jour et les personnalisations

Tableau 1 - Comparaison des solutions offertes à letmotiv

Exploration des solutions externalisées :

Pour notre exploration, nous avons envisagé différentes solutions externalisées, notamment des logiciels SaaS, des packages de back-office et des packages de widgets.

Chaque option présente des avantages et des inconvénients qui influencent notre stratégie globale :

	Avantages	Inconvénients
Logiciels SaaS	• Récupération facile des données des API (Meta, Google Analytics, Twitter) • Traitements statistiques supplémentaires • Beaucoup de fonctionnalités liées aux données • Reporting en mode click-and-drop	• Intégration complexe des dashboards • Dashboard moins polyvalent et flexible • Compétences et développement pour pouvoir exploiter dans nos solutions ces traitements • Coûts très élevés • Complexités d'utilisation avec nos technologies
Packages sur-mesure ou homemade	• Amélioration des packages comme on le souhaite • Méthode CRUD[20] directement intégrée • Facilité de création de manager • Widgets prêts en main • Base Echarts, ChartsJS, HighCharts (Facile à en créer d'autres graphes)	• Codage requis pour les faire fonctionner • Grosse configuration requise • Connaissance du langage pour l'intégrer à des outils externes aux packages
Packages Widgets	• Package avec l'essentiel requis • Coûts bas	• Conçu pour créer des graphiques moins pour des tableaux de bord complexes • Créé par la communauté • Peu de fonctionnalités annexes

Tableau 2 - Comparaison des possibilités d'externalisations

Comme le montrent les différents tableaux comparatifs, chaque solution externe apporte ses propres avantages et défis, notamment en termes de complexité de gestion de la data. Opter pour ces solutions pourrait nous permettre de résoudre simultanément plusieurs de nos problèmes.

Avant de prendre une décision, examinons les solutions externes qui ont retenu notre attention :

	Licence / Prix	Utilisation avec Laravel	Offres
Power BI	Élevé	Package utilisable	• Tableaux de bord avancés, • Intégration de diverses sources de données
Tableau Software	Très élevé	Package utilisable	• Options de visualisation puissantes, • Bonne gestion des grands ensembles de données
Google Suite	Modéré	Package utilisable	• Intégration avec les services Google, • Facilité d'utilisation
Post Hog	Modéré	Inclus	• Analyse / Dashboard • Tracker Web • Back-Up • Test BDD • Data Warehouse • Pipeline / Profilage
Amplitude	Élevé	Package semi-complet	• Dashboard • Data Analyse User • Tracker Web • Back-Up • Test BDD • Data Warehouse • Pipeline / Profilage
MixPanel	Élevé	Package semi-complet	• Dashboard • Data Warehouse • Pipeline / Profilage • Analyse événementielle, • Tracking d'interactions
Databricks, Snowflack	Très élevé	Endpoint[21]	• Plateforme d'analyse basée sur Apache Spark, • Scalabilité

Tableau 3 - Comparaison de toutes les solutions SaaS[(6)]

[6] JEMS Group, "Our Data Science Offer.", URL : https://www.jems-group.com/en/our-offers/data-science/

	Licence / Prix	Offres
Laravel Nova	Privée • 99€ / projet	• Manager • Dashboards • Widgets • Filtres[22] • CRUD • Recherche
BackPack[5]	Privée • 69€ / projet	• Manager • Dashboards • Widgets • Filtres • CRUD • Plug-in inclus
Orchid	Open Source • Gratuit	• Manager • Dashboards • Widgets • CRUD
Filament[4]	Open Source • Gratuit	• Manager • CRUD • Dashboards • Widgets • Filtres • Recherche • Market place plug-in communautaire

Tableau 4 - Comparaison de tous les packages Back Offices[3-4-5]

	Licence / Prix	Offres
Laravel Charts	Open Source • Gratuit	• Widgets • Filtres
Lavacharts	Open Source • Gratuit	• Widgets • Filtres
Laravel ApexCharts	Privée	• Widgets

Tableau 5 - Comparaison de tous les packages de widgets

[3] Retool, "Best Laravel Admin Panels.", URL : https://retool.com/blog/best-laravel-admin-panels

[4] Filament, "Installation Back Office", URL : https://filamentphp.com/docs/3.x/panels/installation

[5] Backpack for Laravel, "Documentation.", URL : https://backpackforlaravel.com/docs

Maintenant que nous avons examiné en détail une partie des solutions externes disponibles, il est crucial de considérer sérieusement le développement de solutions internes. Si les options externes présentent des arguments convaincants, elles comportent également des limitations significatives qui pourraient ne pas répondre pleinement à nos besoins spécifiques et même nous imposer des contraintes à long terme.

En effet, l'utilisation de logiciels SaaS, bien qu'offrant de nombreuses fonctionnalités, s'accompagne de coûts élevés et présente des limites d'intégration avec Laravel, qui pourraient compromettre à la fois la santé financière de l'entreprise ainsi que la qualité des rapports produits. Opter pour de telles solutions, malgré leurs avantages, pourrait donc résoudre certains de nos problèmes tout en nous créant de nouveaux, un dilemme qui nous met face à un choix complexe.

D'un autre côté, choisir des packages qui s'intégreraient parfaitement à notre technologie pourrait sembler une solution idéale. Cependant, la complexité de leur configuration nécessiterait un investissement en temps et en ressources humaines non négligeable, ce qui pourrait retarder d'autres projets. De plus, leur évolution dépendant de nous, la gestion de leur adaptabilité pourrait s'avérer complexe, nous laissant potentiellement dépendants des fournisseurs de ces packages.

C'est pourquoi le développement interne émerge comme la voie la plus prometteuse. Bien que plus exigeant en termes de temps et d'effort initial, il nous offrirait une maîtrise totale sur la qualité du code et garantirait une indépendance sur cette technologie. Cette approche nous permettrait non seulement de personnaliser les solutions en fonction de nos besoins, mais aussi d'assimiler et d'adapter les meilleurs aspects du design des technologies existantes. En intégrant ces solutions à nos systèmes, nous maximiserions leur efficacité et leur pertinence, assurant ainsi une base solide pour le futur développement et l'innovation de letmotiv.

En conclusion, bien que les arguments des solutions externes soient indéniables, les avantages à long terme du développement interne, notamment en termes de contrôle et d'adaptabilité, orientent clairement notre choix vers cette option. Cette décision stratégique favorisera une croissance durable et une évolution technologique avec les objectifs de cette entreprise.

2.3 - Décisions et résultats des solutions adoptées

Cette section est divisée en deux parties principales, la première concernant le développement interne du package de création de tableaux de bord, et la seconde se focalisant sur la migration des données vers NoSQL[19].

Développement interne des tableaux de bord

Suite à notre veille technologique approfondie sur les solutions de tableaux de bord, nous avons opté pour un développement interne pour plusieurs raisons essentielles :

1. **Adéquation avec nos besoins spécifiques**, Un développement interne nous offre la flexibilité nécessaire pour adapter la solution précisément à nos besoins.
2. **Rapidité de déploiement**, l'utilisation d'une solution externe aurait requis une mise à niveau importante de nos systèmes et une formation de ces outils, ce qui aurait pu ralentir le processus de déploiement.
3. **Intégration et flexibilité**, Laravel facilite le développement de packages personnalisés, nous permettant ainsi de créer des modules polyvalents qui s'intègrent parfaitement à notre écosystème sans complications majeures.

La première version du package intègre de nombreuses fonctionnalités similaires à celles proposées par les solutions SaaS, tout en étant parfaitement harmonisées avec notre écosystème afin de garantir une intégrité et une adaptabilité maximales.

Le package comprend trois composants clés :

1. la conception de la structure des tableaux,
2. la création de widgets,
3. l'intégration de filtres dynamiques pour une interaction utilisateur optimale.

Les structures

La structure est essentielle, car un tableau de bord bien conçu ne se contente pas de surveiller les performances, il motive et engage les utilisateurs à interagir avec les données. Notre objectif est de rendre les structures accessibles et de simplifier le codage pour éviter les complexités inutiles.

“Un tableau de bord bien conçu permet non seulement de surveiller la performance de votre entreprise, mais aussi de motiver votre équipe et de la mobiliser.” (2)

Anish Ambujakshan

Conseiller d’affaires principal, BDC Services-conseils

Pour simplifier l'utilisation des structures et éviter la complexité excessive des classes et des raisonnements, nous avons adopté un système basé sur des lignes et des colonnes. Ce système permet au développeur de créer une ligne, puis de définir les proportions et les subdivisions en colonnes au sein de cette ligne. Ainsi, nous avons conçu des classes faciles à comprendre et un paradigme[28] de codage intuitif pour faciliter l'acquisition et l'utilisation.

Exemples, de création de lignes afin de comprendre comment le package fonctionne

```
<div class="dashboard" >
    <div class="dashboard-row" >
      // exemple : Widget KPI
    </div>
</div>
```

Figure 3 - Création d’une ligne

```
<div class="dashboard" >
    <div class="dashboard-row" >
      // exemple : Widget KPI 2
    </div>
    <div class="dashboard-row" >
      // exemple : Widget KPI 2
    </div>
</div>
```

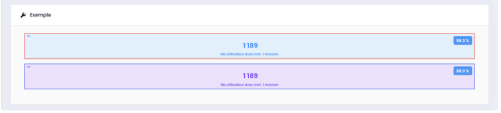


Figure 4 - Création de 2 lignes

Ensuite, voici comment on ajoute des colonnes :

```
<div class="dashboard" >
  <div class="dashboard-row" >
    <div class="three-column" >
      // exemple : 3 * Widget KPI
    </div>
  </div>
</div>
```

Figure 5 - Création de 3 colonnes

---

[2] Quels sont les avantages des tableaux de bord ?, URL : https://www.bdc.ca/fr/articles-outils/operations/efficacite-operationnelle/conseils-utilisation-avantageuse-tableaux-bord#:~:text=Un%20tableau%20de%20bord%20bien,%C3%A9quipe%20et%20de%20la%20mobiliser.

Ces exemples montrent comment, en seulement quelques lignes de code, nous pouvons commencer à construire un tableau de bord. Bien que les exemples utilisent des indicateurs de performance clés (KPI) simples, notre système est conçu pour être flexible et adaptable à divers types de data visualisations.

Nous avons défini des classes génériques pour les colonnes afin de standardiser et faciliter le développement :

1. Two-column : 2 colonnes
2. Three-column : 3 colonnes (Présent dans la *figure 5*)
3. Four-column : 4 colonnes

Nous proposons également différents templates de lignes pour varier les dispositions :

1. Middle-template, *figure 6*
2. Left-template, *figure 7*
3. Right-template, *figure 8*

Colonne centrale : MIDDLE-TEMPLATE

```
<div class="dashboard" >
  <div class="dashboard-row middle-template" >

    <div class="first-column one-line" >
      <widget-kpi
          :properties="{
                  'title': 'Label',
                  'theme': 'primary-soft',
              }"
      ></widget-kpi>
    </div>
    <div class="second-column two-line" >
      <widget-line
          :properties="{
                  'title': 'Title',
              }"
      ></widget-line>
    </div>
    <div class="third-column one-line" >
      <widget-kpi
          :properties="{
                  'title': 'Label',
                  'theme': 'primary-soft',
              }"
      ></widget-kpi>
    </div>

  </div>
</div>
```

Figure 6 - Mise en page d'un template avec 3 colonnes dont 1 colonne forte au milieu

Colonne gauche : LEFT-TEMPLATE

```
<div class="dashboard" >
  <div class="dashboard-row left-template { null || size-1 || siz

    <div class="first-column one-line" >
      <widget-kpi
          :properties="{
                  'title': 'Label',
                  'theme': 'primary-soft',
              }"
      ></widget-kpi>
    </div>
    <div class="second-column two-line" >
      <widget-line
          :properties="{
                  'title': 'Title',
              }"
      ></widget-line>
    </div>

  </div>
</div>
```

Figure 7 - Mise en page d'un template avec 2 colonnes dont 1 colonne faible à gauche

Colonne droite : RIGHT-TEMPLATE

```
<div class="dashboard" >
  <div class="dashboard-row right-template { null || size-1 || si
    <div class="first-column one-line" >
      <widget-kpi
          :properties="{
                  'title': 'Label',
                  'theme': 'primary-soft',
              }"
      ></widget-kpi>
    </div>
    <div class="second-column two-line" >
      <widget-line
          :properties="{
                  'title': 'Title',
              }"
      ></widget-line>
    </div>

  </div>
</div>
```

Figure 8 - Mise en page d'un template avec 2 colonnes dont 1 colonne faible à droite

Ces templates permettent de définir encore plus rapidement la structure d'une ligne d'un tableau de bord et pour encore plus de configuration, il est possible sur left-template et right-template de personnaliser les proportions. Par défaut, nous sommes à 70% pour la colonne forte et 30% pour la colonne faible, la classe size-1 permet de mettre une proportion de 90/10 et size-2 80/20.

L'utilisation de classes prédéfinies, basées sur notre expérience, facilite grandement la réflexion sur la disposition sans limiter excessivement les options. Cela permet de gagner du temps précieux lors de la planification des dispositions. Bien entendu, étant donné que nous travaillons dans un environnement de programmation, il est toujours possible d'ajouter des styles personnalisés à la demande du client pour répondre à des besoins spécifiques.

Pour illustrer la capacité du package à gérer des projets plus complexes, voici un exemple d'une disposition élaborée qui reste réalisable par tous, grâce à la simplicité de notre système :

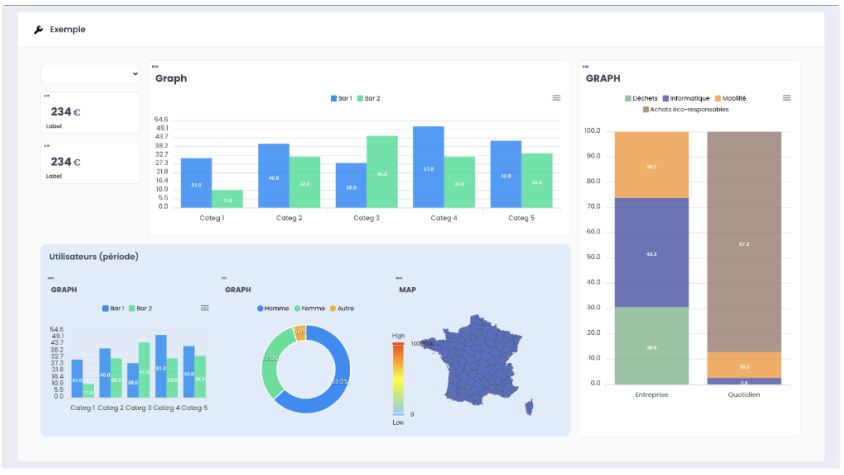


Figure 9 - Exemple de reporting complexe à développer

```
<div class="dashboard" >
    <div class="dashboard-row right-template" >
        <div class="first-column five-line" >
            <div class="dashboard-row left-template size-2" >
                <!-- Filter + KPI -->
                <div class="first-column four-line" >
                    <filter-select></filter-select>
                    <widget-kpi></widget-kpi>
                    <widget-kpi></widget-kpi>
                </div>

                <!-- Graph -->
                <div class="second-column five-line" >
                    <widget-bar
                        :properties="{
                                'title': 'Graph',
                            }"
                        :chart="{
                                'height':270,
                            }"
                    ></widget-bar>
                </div>
            </div>

            <!-- Users -->
            <div class="dashboard-row users" >
                <div class="title-section xl" >Utilisateurs (période)</div>
                <div class="three-column" style="gap:0px;" >
                    <widget-bar
                        :properties="{
                                'title': 'GRAPH',
                                'titleSize': 'lg',
                                'theme': 'nothing',
                                'evo_label': 'AVG',
                                'evo_theme': 'primary'
                            }"
                        :chart="{
                                'height':230,
                            }"
                    ></widget-bar>

                    <widget-pie
                        :properties="{
                                'title': 'GRAPH',
                                'titleSize': 'lg',
                                'theme': 'nothing'
                            }"
                        :chart="{
                                'type': 'donut',
                                'height':250,
                                'textColor': 'black',
                            }"
                    ></widget-pie>

                    <widget-map
                        :properties="{
                                'title': 'MAP',
                                'titleSize': 'lg',
                                'theme': 'nothing'
                            }"
                        :chart="{
                                'height':250,
                            }"
                    ></widget-map>
                </div>
            </div>
        </div>

        <!-- Graph -->
        <div class="second-column five-line" >
            <widget-bar
                :properties="{
                        'title': 'GRAPH',
                    }"
                :chart="{
                        'height':620,
                        'stacked': true,
                        'maxY': 100,
                    }"
            ></widget-bar>
        </div>
    </div>
</div>
```

Figure 10 - Code pour développer la Figure 7

Cet exemple démontre que même pour des configurations avancées, le code reste simple et accessible, permettant ainsi de répondre à un de nos objectifs principaux : rendre le développement de tableaux de bord viable sans se limiter à un spécialiste des données. De plus, pour les petits projets, nous pouvons omettre l'étape du maquettage et passer directement à la production, ce qui économise au moins une journée de travail par rapport à l'utilisation de Google Data Studio.

Ensuite, nous allons commencer la deuxième partie, celle des widgets. Étant l'une des parties les plus importantes et les plus longues à développer, letmotiv a décidé de prioriser le développement des graphiques qui constituent les dashboards d'éco-fidélité. Ainsi avec la création d'autres reportings, nous ajouterons les nouveaux graphiques à ce moment-là. Cette stratégie permet d'alléger le développement et d'éviter de développer des widgets inutiles à moyen terme.

Tout d'abord, qu'allons-nous visualiser pour l'éco-fidélité ?

Étant donné que ce projet est de grande envergure et que les maquettes ont été réalisées avant la rationalisation, les maquettes RSE ont été faites avec Google Data Studio. Voici les trois tableaux (*figures 9 à 11*) :

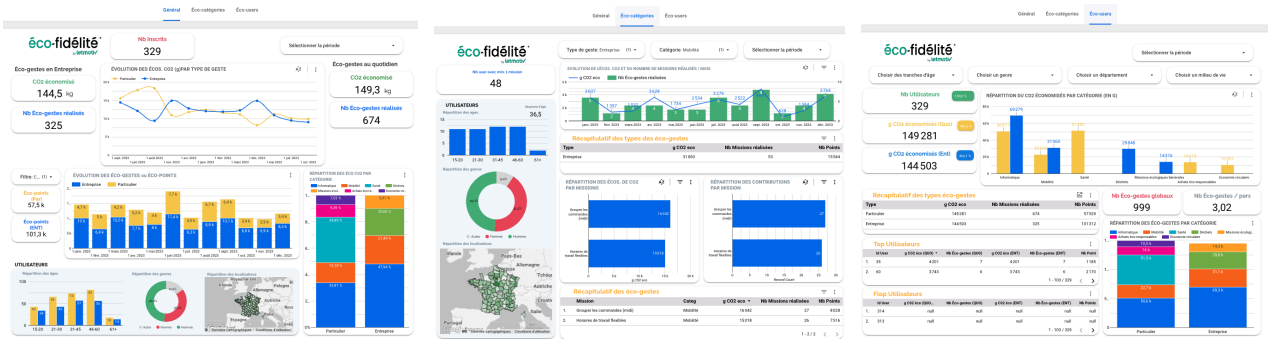


Figure 11 à 13 - Maquettes des dashboards RSE

Nous avons donc huit widgets à créer en priorité. Les widgets sont constitués de deux composants principaux : l'en-tête (*figure 15*) et le visuel (l'indicateur) (*figure 16*).

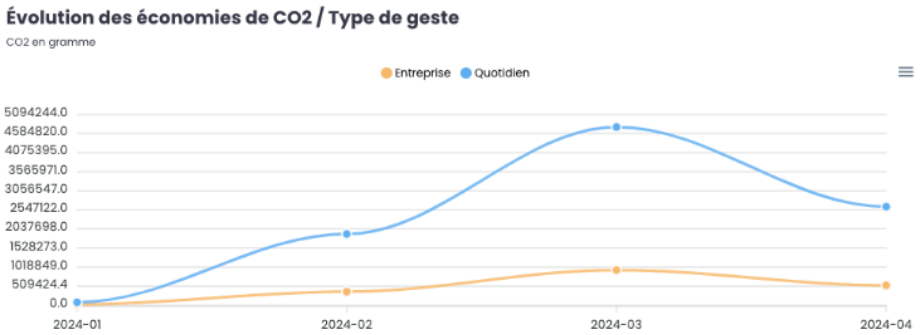


Figure 14 - Un widget d'un graphique en courbes

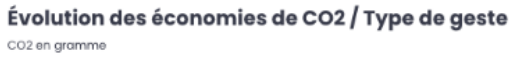


Figure 15 - L'entête du widget graphique en courbes

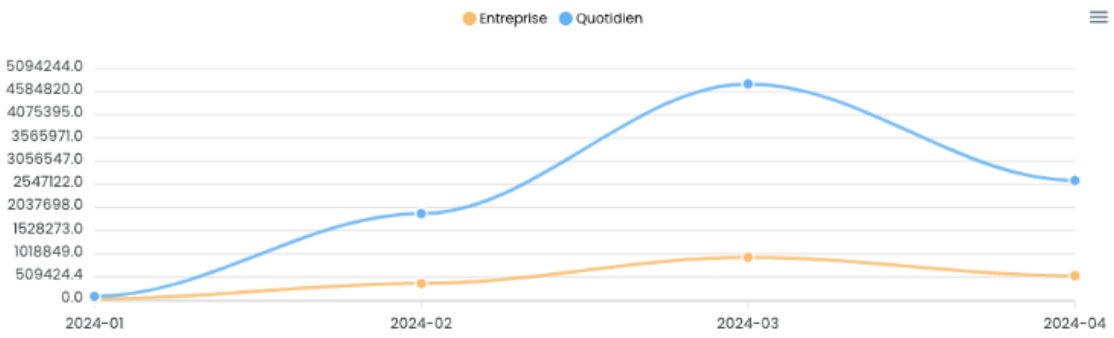


Figure 16 - Le contenu du widget graphique en courbes

L'en-tête est identique pour chaque widget, mais elle doit pouvoir être modifiable. C'est pourquoi, pour chaque widget, nous avons des attributs sous forme de JSON afin de la modifier.

Voici une capture d'un code typique pour modifier un entête (*figure 17*). Centraliser l'entête permet d'avoir une structure commune pour les modifications d'un widget.

```
<widget-[type]
    :properties="{
            'type': '...', // Info du type de widget (defaut: type du widget)
            'title': 'Text', // Titre du widget (defaut: null)
            'titleSize': '', // Taille du titre
            'titleFilter': 'varaible_select', // Titre dynamique à partir
                                              d'un filtre CONCAT(title;' ';valeur filtre[titleFilter])
            'description': '', // Description du widget
            'icon': '', // une icone sur le header du widget (fontawesome pour l'instant)
            'evoLabel': '', // label de la card évolution
        }"
    :nostore="[variable_select']" // desactivé la recharge des données si une variable change
    url_data="/manager/..." // liens des requetes
></widget-[type]>
```

Figure 17 - Modification d'un entête

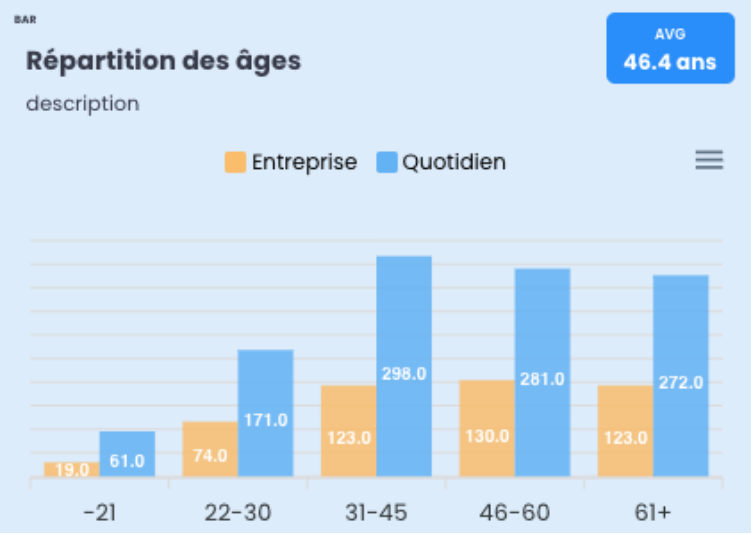


Figure 18 - Widget calibré

Nous avons donc sept variables disponibles pour chaque widget. Pour illustrer les modifications possibles, je vais faire le lien avec la *figure 18* qui est le rendu final. "Type" est une option pour renommer le 'BAR' en haut à gauche. Par défaut, il affiche le type du widget. "Title" permet de modifier 'Répartition des âges'. "Description" est utilisé pour ajouter une description, par défaut, il est nul donc il n'y a pas de ligne. "TitleSize" est une classe de customisation pour changer la taille du titre, par défaut, c'est 24 pixels. "TitleFilter" permet d'avoir un titre dynamique en ajoutant à côté de title la valeur d'une variable de filtre. "Icon" permet d'ajouter une icône. "EvoLabel" permet de donner une valeur à la place de 'AVG' dans le cadran bleu foncé et "class_custom" permet d'ajouter des modifications manuellement sur le widget.

Nous avons aussi des variables de style, qui permettent d'ajuster l'apparence visuelle du widget

de manière plus fine. Voici une capture des variables de style (*figure 19*) et des tableaux de correspondance des styles (*tableaux 6 à 10*).

```
<widget-kpi
  :properties="{
          'titleSize': '', // theme du tritre
          'titleTheme': '', // theme du tritre
          'theme': '', // theme de la card
          'style': '', // (default) weak or strong
          'evoTheme': '', // theme de la card évolution
      }"
  :class_content="" // pour ajouter des class Tailwind CSS dans le widget
></widget-kpi>
```

Figure 19 - Code des autres variables de style d'un widget

`titleSize` :

Valeurs	Content class
DEFAULT	`text-2xl`
sm	`text-sm`
lg	`text-lg`
xl	`text-xl`

Tableau 6 - Classe prise en compte pour la variable titleSize par le package (classes Tailwind CSS[13])

`titleTheme` :

Valeurs	Content class
DEFAULT	`text-black`
custom	color: white;
PRIMARY	color: #3699FF;
SUCCESS	color: #1BC5BD;
INFO	color: #8950FC;
WARNING	color: #FFA800;
DANGER	color: #F64E60;

Tableau 7 - Classe prise en compte pour la variable titleTheme par le package (classes Tailwind CSS)

`theme` :

Valeurs	Content class
DEFAULT light	`text-black bg-white`
dark	color: white;background-color: #1a202c;
nothing	`border-none bg-transparent`
custom	color: white;background-color: #283747;
PRIMARY	color: white; background-color: #3699FF;
PRIMARY-SOFT	color: #3699FF; background-color: #E1F0FF;
SUCCESS	color: white; background-color: #1BC5BD;
SUCCESS-SOFT	color: #1BC5BD; background-color: #C9F7F5;
INFO	color: white; background-color: #8950FC;
INFO-SOFT	color: #8950FC; background-color: #EEE5FF;
WARNING	color: white; background-color: #FFA800;
WARNING-SOFT	color: #FFA800; background-color: #FFF4DE;
DANGER	color: white; background-color: #F64E60;
DANGER-SOFT	color: #F64E60; background-color: #FFE2E5;

Tableau 8 - Classe prise en compte pour la variable theme par le package (classes Tailwind CSS)

`evoTheme` :

Valeurs	Content class
DEFAULT	`bg-light-gray text-black`
nothing	`bg-transparent`
PRIMARY	color: white; background-color: #3699FF;
PRIMARY-SOFT	color: #3699FF; background-color: #E1F0FF;
PRIMARY-TEXT	color: #3699FF; `bg-transparent`
SUCCESS	color: white; background-color: #1BC5BD;
SUCCESS-SOFT	color: #1BC5BD; background-color: #C9F7F5;
SUCCESS-TEXT	color: #1BC5BD; `bg-transparent`
INFO	color: white; background-color: #8950FC;
INFO-SOFT	color: #8950FC; background-color: #EEE5FF;
INFO-TEXT	color: #8950FC; `bg-transparent`
WARNING	color: white; background-color: #FFA800;
WARNING-SOFT	color: #FFA800; background-color: #FFF4DE;
WARNING-TEXT	color: #FFA800; `bg-transparent`
DANGER	color: white; background-color: #F64E60;
DANGER-SOFT	color: #F64E60; background-color: #FFE2E5;
DANGER-TEXT	color: #F64E60; `bg-transparent`

Tableau 9 - Classe prise en compte pour la variable evoTheme par le package (classes Tailwind CSS)

`style` :

Valeurs	Content class
DEFAULT weak	`rounded-md`
strong	`rounded-xl shadow-2xl`

Tableau 10 - Classe prise en compte pour la variable style par le package (classes Tailwind CSS)

Comment fonctionnent-ils ? Comment les câbler avec des vraies données ?

Pour qu'un widget affiche des données, il faut définir un endpoint (*figure 17 et 18*) et ajouter l'URL de l'endpoint dans l'attribut *'url_data'* du widget (*figure 19*).

Création d'un Endpoint

Un endpoint est une URL spécifique qui permet de récupérer les données nécessaires pour alimenter un widget. Dans notre cas, l'endpoint est implémenté avec Laravel. Voici un exemple de code pour un endpoint qui calcule le nombre d'utilisateurs ayant réalisé au moins une mission éco (*figure 20*).

```php
public function distinct_users(Request $request, UserRepository $userRepository): array
{
    $params = $this->init($request);

    $users_all = $userRepository->users_all($params['date_end']);

    $value = DB::table('contributions')->join('users as u', 'u.id', 'contributions.id_user')
        ->where('u.role', 'Participant')
        ->join('eco_missions as m', 'm.id', '=', 'contributions.id_mission')
        ->whereIn('m.destination', ['Tous', 'RSE'])
        ->whereIn('m.type', $params['gestes'])
        ->whereBetween('contributions.created_at', [$params['date_start'], $params['date_end']])
        ->distinct('contributions.id_user')->count();

    return [
        'value'=>$this->functionRepository->formatNumber($value, 0),
        'evo_data' => ($value / ($users_all > 0 ? $users_all : 1) * 100),
        'evo_suffix' => '%',
    ];
}
```

Figure 20 - Code d'un endpoint

```
Codiumate: Options | Test this function
Route::prefix('dashboard')->group(function () {
  Route::prefix('categ_request')->group(function () {
    Route::get('/distinct_users', [CategoryRepository::class, 'distinct_users'])->name('dashboard.categ.distinct_users');
  });
});
```

Figure 21 - Attribution de la fonction sur une route

Attribuer l'Endpoint au Widget

Une fois l'endpoint défini, nous devons l'attribuer au widget en utilisant l'attribut url_data. Voici un exemple de code pour attribuer un endpoint à un widget KPI (*figure 22*).

```
<widget-kpi
    :properties="{
            'type':' ',
            'title': 'Nb utilisateur avec min. 1 mission',
            'theme': 'primary-soft',
            'evoTheme': 'primary'
        }"
    class_content="text-center"
    url_data="/manager/dashboard/categ_request/distinct_users"
></widget-kpi>
```

Figure 22 - Code pour attribuer un endpoint

Rendu final

Une fois le widget configuré avec l'URL de l'endpoint, il peut récupérer et afficher les données. Voici un exemple de résultat affiché par le widget (figure 23).

Figure 23 - Résultat du endpoint

Pour clore cette partie sur les widgets, voici tous les widgets disponibles actuellement :

- KPI : widget simple avec une valeur mise en avant
- Graphique : nous avons des graphiques en barres, en courbes, en nuages de points, en radar
- Diagramme circulaire
- Tableau simple
- Carte : des départements français, heatmap et treemap

Intégration des Filtres

Pour rendre les dashboards plus interactifs et permettre une analyse plus fine, nous intégrons des filtres. Les filtres permettent aux utilisateurs de segmenter les données et de pouvoir focus leurs analyses. Voici quelques exemples de filtres que nous avons intégrés.

Filtre de Période

Le filtre de période permet aux utilisateurs de sélectionner une date de début et une date de fin pour afficher les données correspondantes. Voici un exemple de code pour intégrer des filtres de période (*figure 24*).

```
<filter-date
  size="sm"
  filter="date_start"
  label="Date de début"
></filter-date>
<filter-date
  size="sm"
  filter="date_end"
  label="Date de fin"
></filter-date>
```

Figure 24 - Intégration de filtres pour une période

Et voici le résultat de l'intégration de ces filtres (*figure 25*).

Figure 25 - Résultat de l'intégration

Filtre Select

Le filtre select permet aux utilisateurs de choisir parmi différentes options. Il peut être configuré pour permettre une sélection unique ou multiple. Voici un exemple de code pour un filtre select (*figure 26*).

```
<filter-select
    filter_select="type_gestes_select"
    store_data="type_gestes_list"
    label="Type de geste"
></filter-select>
```

Figure 26 - Intégration d'un filtre select

Voici le schéma d'initialisation d'un filtre select (*figure 27*).

```
<filter-select
    filter_select="variable_select"
    store_data="variable_list" ou url_data = '/filters/...' // url_data prioritaire
    :multiple="true" // plusieurs possibilités
></filter-select>
```

Figure 27 - Schéma d'initialisation d'un filtre select

Utilisation des Filtres

Pour utiliser les filtres, nous utilisons le store Pinia[12] de Vue.js. C'est une librairie qui permet de stocker des valeurs (*figure 28*) et de les utiliser ensuite. L'objectif est de centraliser nos

variables, et si la valeur contenue dans une variable change, les widgets envoient une requête sur les endpoints attribués en amont avec les nouvelles valeurs des filtres.

```
export const useFilterStore = defineStore('filters', {
    state: () => {
        const currentDate = new Date();
        const dateStart = new Date(currentDate.setDate(currentDate.getDate() - 90)).toISOString().split('T')[0];

        return {
            date_start: dateStart,
            date_end: new Date().toISOString().split('T')[0],

            // General
            point_geste_select: 'Éco-gestes',
            point_geste_list: ['Éco-gestes', 'Éco-points'],
        };
    },
    actions: {
        setData(name, value) {
            this[name] = value;
        },
    },
})
```

Figure 28 - Exemple de code pour le stockage des variables

Pour chaque filtre, on retrouve les attributs ***label*** pour ajouter un label au-dessus de l'input et ***size*** pour définir la taille de l'input. Enfin, pour avoir des données, les filtres de date utilisent l'attribut ***filter*** pour donner le nom de la variable Pinia qui va stocker ce filtre. Étant donné que les dates sont uniques, un seul champ est nécessaire. Cependant, pour les selects, on a un attribut vrai ou faux pour savoir si on veut une multiple sélection (***multiple***). Enfin, pour les données, on a l'attribut ***filter_select*** qui a la même fonction que ***filter*** pour les dates, à savoir récupérer la valeur de filtrage.

Voici le résultat du développement de la visualisation de l'éco-fidélité (étant câblées sur une démo, les données ont été générées aléatoirement) :

Figure 29 - Tableau de bord Général finalisé pour la démo éco-fidélité RSE

Figure 30 - Tableau de bord Catégorie finalisé pour la démo éco-fidélité RSE

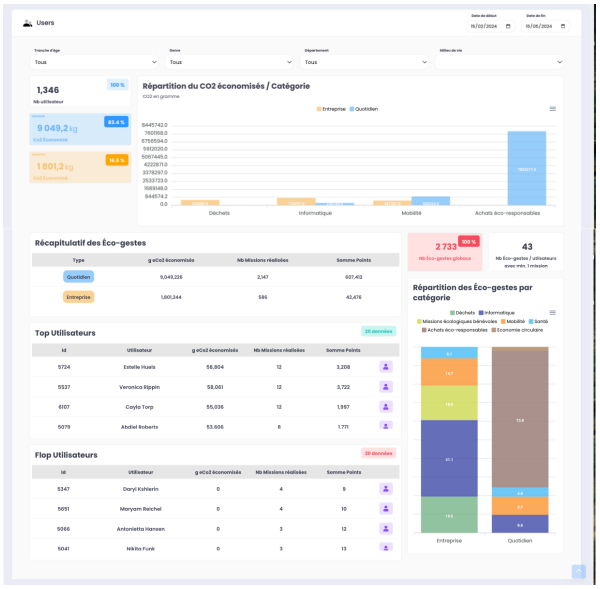


Figure 31 - Tableau de bord Utilisateurs finalisé démo éco-fidélité RSE

Quelles sont les évolutions pour ce package ?

Le package est voué à intégrer de nouveaux widgets comme des tableaux croisés ou encore des indicateurs d'objectifs. Il peut également inclure l'intégration d'analyses statistiques comme la courbe des tendances, des prévisions dynamiques et des tableaux d'analyses de séries temporelles[23]. De nouvelles classes de styles prédéfinis et des templates de disposition pourront aussi être ajoutés.

Pour conclure sur ce package, les impacts de ce package ont été immédiats. Par exemple, un tableau comme celui illustré ci-dessus aurait pris de deux à trois semaines à développer sans le package. Grâce à ce package, nous avons pu développer deux tableaux de bord en une semaine, réduisant ainsi le temps de production d'un reporting par cinq. Ce n'est qu'une première version et il est voué à être enrichi à l'avenir.

Migration de données vers NoSQL

Initialement, le package devait être une solution complète NoSQL comprenant les migrations et des fonctions CRUD. Cette solution visait à accélérer les temps de chargement des analyses de données, un aspect crucial pour la visualisation statistique.

Afin de répondre à cette problématique, nous avons exploré différentes solutions NoSQL comme MongoDB, Redis, Cassandra, et d'autres. Étant donné le nombre croissant d'utilisateurs et l'amélioration continue des systèmes, nous avons décidé de tester MongoDB. Ce choix repose sur plusieurs avantages :

1. **Diversité des modèles**, MongoDB implémente de nombreux modèles NoSQL (documents, clé-valeur, séries temporelles, etc.) ou est en train de les développer.
2. **Support technique**, MongoDB offre un package de requêtes simplifiées pour l'interaction avec Laravel.
3. **Modernité et fonctionnalité**, MongoDB continue d'ajouter de nouvelles fonctionnalités, ce qui correspond à notre besoin d'adopter des systèmes en évolution constante.

Pour valider ce choix, nous avons réalisé deux tests techniques avec Laravel :

1 - Test de requêtes simples :

L'objectif de ce test était de définir les gains pour la méthode CRUD de ce changement de systèmes avec des requêtes très simples, afin d'avoir un aperçu des bénéfices pour les mêmes finalités de requêtes sur une base de 30 000 lignes de données.

Voici les résultats (*tableau 11*) :

Table 30k lignes	Sequel Pro (SQL Classic)	MongoDB
Poids fichier	5,5 MB	0,5 MB MEILLEUR
Create	`User::create([ 'name' => 'Admin', 'email' => 'admin@letmotiv.io', 'password' => '...', ]);` Temps : 86 MS	`UserMongoDB::create([ 'name' => 'Admin', 'email' => 'admin@letmotiv.io', 'password' => '...', ]);` Temps : 55 MS
Read	`User::where('email', 'kylian@letmotiv.io')->get();` Temps : 27 MS (début 153 ms)	`UserMongoDB::where('email', 'kylian@letmotiv.io')->get();` Temps : 19,4 MS (début 114 ms)
Update	`User::where('email', 'kylian@letmotiv.io')->update(['name' => 'Admin']);` Temps : 47 MS	`UserMongoDB::where('email', 'kylian@letmotiv.io')->update(['name' => 'Admin']);` Temps : 55 MS
Delete	`User::where('email', 'admin@letmotiv.io')->delete();` Temps : 48 MS	`UserMongoDB::where('email', 'admin@letmotiv.io')->delete();` Temps : 55 MS

Tableau 11 - Comparatif des différents logiciels pour la méthode CRUD

Les résultats ont montré que MongoDB offrait des améliorations significatives en termes de vitesse de lecture et de stockage. Les gains sur les opérations de lecture étaient particulièrement notables avec une différence d'environ 7,6 ms. Le stockage était également

beaucoup plus efficace, avec une réduction de la taille par un facteur de 11. Cependant, les performances étaient inférieures pour les opérations de mise à jour et de suppression, dues à l'optimisation intrinsèque de Laravel pour les bases SQL. Bien que l'interaction directe avec les données n'était pas l'objectif principal de ce test, il a confirmé que MongoDB n'est pas optimal pour toutes les opérations, surtout celles nécessitant des interactions fréquentes avec des données existantes.

2 - Test de requêtes complexes pour le pilotage en condition réelle :

Ce test visait à évaluer si MongoDB pouvait gérer efficacement les requêtes complexes utilisées pour le reporting. Nous avons repris les requêtes utilisées sur l'éco-fidélité et les avons traduites en requêtes MongoDB.

Le défi était de taille et a révélé plusieurs limitations. La complexité des requêtes nécessaires pour le reporting a montré que le package MongoDB pour Laravel manquait de maturité.

Pour des requêtes très complexes, les requêtes MongoDB nécessitent une formation spécifique et un temps de développement important, car elles ne sont pas toujours capables de réaliser les mêmes opérations que les requêtes Éloquent. Cette limitation a confirmé que, malgré ses avantages pour certaines opérations, MongoDB ne pouvait pas encore remplacer entièrement nos bases de données SQL pour les besoins de reporting complexes.

Grâce à ces tests, nous avons pris la décision d'adopter une approche hybride. Pour toutes ces raisons, nous avons décidé de continuer à utiliser nos SGBD déjà en place pour les requêtes complexes, mais d'intégrer des fonctions de migration de données vers des paradigmes NoSQL pour optimiser la taille et le temps d'exécution des requêtes, permettant ainsi des analyses et des développements plus rapides.

Nous restons conscients que MongoDB sera nécessaire dans un avenir proche. Par conséquent, nous avons également décidé d'inclure MongoDB dans notre architecture, laissant ainsi le choix aux développeurs d'orienter ces migrations vers nos systèmes de base SQL ou vers MongoDB.

Pour l'instant, deux fonctions de migration existent :

Fusion de tables avec agrégats (*figure 32*) :

```php
public function migrate(CollectionController $collection) {
  $table = 'users'; // table à migrer
  $table_fields = ['id', 'role', 'lastname', 'firstname']; /* champs à garder pour la table à migrer,
                                                              si [], tout est pris */

  $type = 'nosql'; // possibilité de choisir la destination nosql => mongodb sinon SGBD classic

  $schema = [
    [ 'type' => 'all', 'table' => 'contributions', 'alias' => 'c', 'tableKey' => 'id_user',
      'originKey' => 'id', 'count' => true ],
    [ 'type' => 'join', 'table' => 'user_data_values', alias' => 'birthday', 'originKey' => 'id',
      'tableKey' => 'user_id', 'tableFields' => ['value'], 'conditions' => "birthday.key = 'birthday'" ],
    [ 'type' => 'join', 'table' => 'user_data_values', 'alias' => 'gender', 'originKey' => 'id',
      'tableKey' => 'user_id', 'tableFields' => ['value'], 'conditions' => "gender.key = 'gender'" ],
  ];

  $aggregate = [
    [ 'table' => 'contributions', 'alias' => 'c',
      'originKey' => 'id', 'tableKey' => 'id_user',
      'calculs' => [
        'id' => ['count'],
        'co2_economise' => ['sum', 'avg'],
        'points' => ['sum', 'avg'],
        'id_mission' => ['count_distinct'],
        'id_categorie' => ['count_distinct']
      ] ],
    [ 'table' => 'achats', 'alias' => 'a',
      'originKey' => 'id', 'tableKey' => 'id_user',
      'calculs' => [
        'id' => ['count'],
        'qte' => ['sum', 'avg'],
        'prix_final' => ['sum', 'avg'],
        'id_produit' => ['count_distinct'],
        'id_mission' => ['count_distinct']
      ] ],
  ];

  return $collection->migrate_table($table, $table_fields, $schema, $aggregate,
    $drop (default false), $rebuild (default false) );
  /* $drop = on supprime toutes les données avant de remplire à nouveau la base
     | $rebuild = on supprime la base avant de la refaire */
}
```

Figure 32 - Code commenté pour réussir une migration

Cette fonction permet de donner un schéma des tables et des agrégats souhaités. Elle fusionne les données de plusieurs tables en un document unique par individu, incluant des agrégats spécifiques. Par exemple, nous pouvons fusionner les tables "Users", "Contributions" et "User_data_values" pour créer des documents utilisateurs enrichis avec des métriques précalculées.

Migration en séries temporelles (*figure 33*) :

```php
public function timeseries(CollectionController $collection) {
    $table = 'users';
    $type = 'nosql'; // possibilité de choisir la destination nosql => mongodb sinon SGBD classic

    $time = [
      'day', // Type de periode : day  | month | year
      '2023-04-16', // Start période (default = null) si null il prend 2023-01-01
      '2024-04-18' // End période (default = null) si null il prend la Carbon::now()
    ];

    $schema = [
      [
        'users', 'u',
        [
          'id' => ['count'],
          'id_departement' => ['count_distinct'],
          'densite' => ['avg'],
        ]
      ],
      [
        'contributions', 'c',
        [
          'id' => ['count'],
          'co2_economise' => ['sum', 'avg'],
          'points' => ['sum', 'avg'],
          'id_mission' => ['count_distinct'],
        ]
      ],
    ];

    return $collection->migrate_timeseries($type, $table, $schema, $time,
        $drop (default false), $rebuild (default false) );
    /* $drop = on supprime toutes les données avant de remplire à nouveau la base
      | $rebuild = on supprime la base avant de la refaire */
  }
```

Figure 33 - Code commenté pour réussir une migration

Cette fonction permet de créer des séries temporelles avec les agrégats souhaités. Elle migre les données avec un identifiant de date, facilitant ainsi les requêtes sur des périodes spécifiques. Cela permet de réaliser des analyses temporelles sans les complexités des jointures et des calculs en temps réel, améliorant ainsi la performance des requêtes.

Pour finir sur ce package, bien que MongoDB ne soit pas encore entièrement adapté pour remplacer toutes les fonctionnalités de nos bases SQL, son utilisation stratégique pour certaines opérations d'analyse nous permet de tirer parti de ses nombreux avantages. Nous continuerons à surveiller les évolutions des outils NoSQL et à adapter notre stratégie en fonction des progrès technologiques, assurant ainsi une performance optimale et une évolutivité continue pour letmotiv.

En conclusion, cette deuxième partie a montré comment letmotiv a abordé les défis de l'industrialisation des processus de reporting en adoptant une approche hybride. Nous avons d'abord détaillé le développement interne des tableaux de bord, mettant en avant la création de structures flexibles et de widgets interactifs qui ont permis de réduire significativement les temps de développement. Ensuite, nous avons abordé la migration des données vers des solutions NoSQL, notamment MongoDB, pour optimiser les temps de réponse et améliorer les performances des analyses. Cette stratégie combinée nous permet de tirer parti des avantages des bases de données SQL et NoSQL, posant ainsi les bases pour une innovation continue et un positionnement renforcé de letmotiv sur le marché.

3/ Impacts commerciaux de l'industrialisation

La rationalisation des processus de reporting chez letmotiv a déjà eu un impact significatif sur l'entreprise. En optimisant nos outils et en intégrant des technologies avancées, nous avons non seulement amélioré notre compétitivité, mais aussi ouvert de nouvelles perspectives stratégiques.

3.1 - Avantages concurrentiels induits

L'industrialisation des reportings a permis à letmotiv de transformer et optimiser ses méthodes de travail. Cette transformation se traduit par une réduction notable des délais de développement, une amélioration de la qualité des produits livrés et une optimisation des ressources. En conséquence, letmotiv a pu offrir à ses clients des solutions plus robustes, plus rapides et plus fiables, tout en se distinguant de la concurrence par ses innovations techniques et son engagement en faveur de la durabilité.

L'industrialisation a permis à letmotiv de gagner en efficacité opérationnelle, réduisant les délais de développement et de livraison des reportings. La qualité et la précision des tableaux de bord ont renforcé la confiance de nos clients dans cet exercice. Ce package nous a permis d'avoir des améliorations sur plusieurs plans :

1. **Production**, nous avons réduit les temps et la complexité de développement, ce qui nous permet à terme d'avoir plus de ressources pour d'autres projets et de pérenniser ces compétences au sein de Letmotiv.
2. **Économique**, corrélée à la production, la diminution des temps et des compétences nécessaires à ces développements a un impact économique à moyen terme, car nous aurons une meilleure capacité de production.
3. **Commercial**, nous pourrons répondre plus rapidement aux demandes des clients en publiant des reportings dans des délais plus courts.

De plus, l'intégration de solutions NoSQL est très prometteuse pour améliorer la performance des analyses, offrant une expérience utilisateur optimisée. Bien plus qu'une simple optimisation des temps de chargement, ce type de paradigme va nous permettre de l'ajouter en argument positif pour l'éco-fidélité et pour letmotiv en général puisque, comme l'illustre le tableau 12, nous allons pouvoir économiser sur plusieurs plans [17] :

---

[17] LinkedIn, "How do you monitor and troubleshoot NoSQL performance issues ?", URL : https://www.linkedin.com/advice/0/how-do-you-monitor-troubleshoot-nosql-performance-issues

1. **Économique**, Bien que minime sur un an pour une entreprise, ce montant n'est pas négligeable, car il pourra être redistribué dans la R&D ou en avantages pour les employés.
2. **Technique**, En allégeant les ressources de nos serveurs et ainsi réduire leur utilisation par utilisateur, nous serons prêts à gérer une augmentation du trafic sans ajouter de nouveaux serveurs.
3. **Écologique**, Cet argument "green IT"peut être très pertinent pour l'éco-fidélité en montrant que nos serveurs consomment un minimum et que nos ressources sont gérées correctement.

Voici dans la *figure 34* les conditions des calculs.

Exemples de requêtes

SQL	NoSQL
CREATE TABLE timeseries_invoices AS SELECT DATE_FORMAT(i.created_at, '%Y-%m-%d') AS jour, DATE_FORMAT(i.created_at, '%M') AS mois, COUNT(DISTINCT i.user_id) AS nb_user, SUM(CASE WHEN i.status = 3 THEN d.qte ELSE 0 END) AS sum_qte, SUM(CASE WHEN i.status = 3 THEN d.amount ELSE 0 END) AS sum_ca, SUM(CASE WHEN i.status = 3 THEN 1 ELSE 0 END) AS nb_invoices_valid, SUM(CASE WHEN i.status = 1 THEN 1 ELSE 0 END) AS nb_invoices_refuse, SUM(CASE WHEN i.status = 0 THEN 1 ELSE 0 END) AS nb_invoices_new, SUM(CASE WHEN i.status = 2 THEN 1 ELSE 0 END) AS nb_invoices_error, SUM(1) AS nb_invoices FROM table_invoices i INNER JOIN table_invoice_details d ON d.invoice_id = i.id GROUP BY jour	SELECT * FROM timeseries_invoices

Gains	SQL	NoSQL
Lignes	993 043	365
Temps	13,2s	0,009s

Tableau des gains

Formules

Jour SQL	N JOUR * X USER REQUÊTES * SQL TEMPS REQUÊTES
Jour NoSQL	(N JOUR * X USER REQUÊTES * NOSQL TEMPS REQUÊTES) + (N JOUR * SQL TEMPS REQUÊTES)

Figure 34 - Contexte des calcules

Jours	H de server (SQL)	H de server (NoSQL)	H de server économisées (SQL - NoSQL)	kWh économisés	Kg Co2e économisés	€ Electricités économisés
30	1 100	0,977	1 099	401,14	127,16	93,67
60	2 200	1,953	2 198	802,29	254,32	187,33
90	3 300	2,930	3 297,1	1 203,43	381,49	281
120	4 400	3,907	4 396,1	1 604,57	508,65	374,67
150	5 500	4,883	5 495,1	2 005,72	635,81	468,34
180	6 600	5,860	6 594,1	2 406,86	762,97	562
210	7 700	6,837	7 693,2	2 808,01	890,14	655,67
240	8 800	7,813	8 792,2	3 209,15	1 017,30	749,34
270	9 900	8,790	9 891,2	3 610,29	1 144,46	843
300	11 000	9,767	10 990,2	4 011,44	1 271,62	936,67
330	12 100	10,743	12 089,3	4 412,58	1 398,79	1 030,34
360	13 200	11,72	13 188,3	4 813,72	1 525,95	1 124

Tableau 12 - Comparatif des économies potentielles en passant sur du NoSQL [18-19-20]

[18] EasyVirt, "Consommation d'énergie des centres de données.", URL : https://www.easyvirt.com/consommation-energie-centres-de-donnees/#:~:text=La%20puissance%20consomm%C3%A9e%20est%20fonction,les%20serveurs%20%C3%A0%20deux%20prises

[19] Ministère de la Transition Écologique, "Émissions de CO2 de l'industrie.", URL : https://www.statistiques.developpement-durable.gouv.fr/edition-numerique/chiffres-cles-du-climat/10-emissions-de-ges-de-lindustrie#:~:text=%C3%89missions%20de%20CO2%20pour,d%27%C3%A9lectricit%C3%A9%20dans%20l%27UE&text=Depuis%201990%2C%20les%20%C3%A9missions%20de,CO2%2FkWh%20en%202018

[20] Ekwateur, "Prix de l'électricité en Europe.", URL : https://ekwateur.fr/blog/marche-de-l-energie/prix-electricite-europe-pays/

Enfin, toutes ces avancées ont permis de poser des bases solides dans la manipulation des données. Bien plus que cela, elles vont permettre d'optimiser mon poste sur de nombreux aspects. De plus, il va y avoir de conséquences positives pour l'entreprise. Évidemment, le reporting ne fait pas le produit, mais c'est un outil essentiel pour une plateforme et les bénéfices vont se répercuter sur de nombreux aspects.

3.2 - Stratégies futures

Avec les gains réalisés grâce à l'industrialisation des processus de reporting, letmotiv est bien positionnée pour envisager des stratégies futures ambitieuses et innovantes. Ces stratégies s'appuieront sur les avancées technologiques et organisationnelles récentes pour renforcer notre position sur le marché et explorer de nouvelles opportunités.

L'un des axes majeurs de notre stratégie future sera l'expansion de nos capacités analytiques. En intégrant davantage de solutions NoSQL et en développant des outils d'analyse avancée, nous pourrons offrir à nos clients des insights encore plus précis et pertinents. Cela inclura :

1. **Analyse prédictive**, utiliser des algorithmes pour prévoir les tendances futures basées sur les données historiques, comme des prédictions sur séries temporelles, des Analyses de Composantes Principales pour vectoriser les utilisateurs pour du profilage, etc.
2. **Segmentation**[26] **avancée**, Développer des outils pour segmenter les utilisateurs avancée comme des outils de profilage, permettant des campagnes marketing hyper-ciblées ou optimiser l'impact de nos plateformes.
3. **Tableaux de bord interactifs**, améliorer l'interactivité des tableaux de bord avec de nouveaux widgets et filtres, l'objectif est d'enrichir le package par des options de personnalisations des données à représenter et de segmentations.

Néanmoins, cette industrialisation nécessitera une maintenance continue pour optimiser encore plus les performances et pour rester compétitif sur ce type d'outils.

Pour conclure, les stratégies futures de letmotiv viseront à consolider nos acquis tout en explorant de nouvelles voies de croissance et d'innovation. Grâce à l'industrialisation de nos processus de reporting et à notre engagement envers la durabilité, nous sommes prêts à relever les défis de demain et à saisir les opportunités qui se présenteront.

Conclusion

Le processus d'industrialisation des outils de reporting chez letmotiv a permis d'atteindre plusieurs objectifs clés :

- Premièrement, la décision de développer en interne des tableaux de bord personnalisés a offert une flexibilité et une adaptation précises aux besoins spécifiques de nos clients et de l'entreprise. Cette approche a non seulement réduit les délais de développement, mais a également amélioré la qualité des produits livrés, renforçant ainsi la confiance de nos clients et améliorant notre compétitivité sur le marché.

- Deuxièmement, l'intégration des paradigmes NoSQL est vouée à optimiser les performances des analyses de données en réduisant significativement les temps d'exécution des requêtes, améliorant l'expérience utilisateur et ainsi proposer une solution éco-friendly pour l'éco-fidélité. Cette transition vers une architecture hybride, combinant les avantages des paradigmes SQL et NoSQL, pose les bases pour une innovation continue et des réflexions constructives. En intégrant progressivement des fonctions de migration de données vers des paradigmes NoSQL, nous préparons aussi letmotiv à gérer efficacement une augmentation future du volume de données, tout en maintenant une utilisation optimale des ressources.

En outre, l'industrialisation de nos outils de reporting a permis d'optimiser les ressources et de renforcer nos capacités internes. Cette démarche a ouvert de nouvelles perspectives stratégiques et d'innovations pour letmotiv. En améliorant notre capacité à analyser et visualiser des données de manière plus efficace, nous sommes mieux préparés à développer des solutions innovantes répondant aux attentes croissantes de nos clients en matière de techniques, de durabilité et de performance. Ces avancées pourront nous positionner comme un acteur clé capable de relever les défis technologiques de demain grâce à l'éco-fidélité tout en contribuant à une gestion responsable et durable des ressources.

Bibliographies

1. Clouddevs, "Laravel Packages: Extending Functionality with Third-Party Libraries.", URL : https://clouddevs.com/laravel/packages/

2. BDC, "Quels sont les avantages des tableaux de bord ?", URL : https://www.bdc.ca/fr/articles-outils/operations/efficacite-operationnelle/conseils-utilisation-avantageuse-tableaux-bord

3. Retool, "Best Laravel Admin Panels.", URL : https://retool.com/blog/best-laravel-admin-panels

4. Filament, "Installation Back Office", URL : https://filamentphp.com/docs/3.x/panels/installation

5. Backpack for Laravel, "Documentation.", URL : https://backpackforlaravel.com/docs

6. JEMS Group, "Our Data Science Offer.", URL : https://www.jems-group.com/en/our-offers/data-science/

7. Oracle, "Base de données relationnelle vs. Non relationnelle.", URL : https://www.oracle.com/fr/database/base-donnees-relationnelle-difference-non-relationnelle/

8. AWS, "AWS Free Database.", URL : https://aws.amazon.com/fr/free/database/?gclid=CjwKCAiAzc2tBhA6EiwArv-i6UE1P9sUYaVDt8BzwvPepE9vF_gCheQFJxP1qvyFGCRx9-ytyaTULhoCBAcQAvD_BwE&trk=d5944ab8-a2e2-4f6e-98b6-cafb66144b54&sc_channel=ps&ef_id=CjwKCAiAzc2tBhA6EiwArv-i6UE1P9sUYaVDt8BzwvPepE9vF_gCheQFJxP1qvyFGCRx9-ytyaTULhoCBAcQAvD_BwE:G:s&s_kwcid=AL!4422!3!548727697912!e!!g!!aws%20sql%20database!12260822385!119890183120

9. DataStax, "What is NoSQL?", URL : https://www.datastax.com/fr/what-is/nosql

10. Microsoft Azure, "Non-relational data.", URL : https://learn.microsoft.com/fr-fr/azure/architecture/data-guide/big-data/non-relational-data

11. DataScientest, "SQL vs NoSQL: Differences, Utilisations, Avantages et Inconvénients.", URL : https://www.google.com/url?q=https://datascientest.com/sql-vs-nosql-differences-utilisations-avantages-et-inconvenients&sa=D&source=editors&ust=1709743361873266&usg=AOvVaw1HJeGMRqcfUHW5yr41CmaH

12. LeHibou, "Qui utilise le Big Data ?", URL : https://www.lehibou.com/communaute/qui-utilise-big-data

13. Les Echos, "Quels sont les impacts du Big Data sur l'entreprise ?", URL : https://www.lesechos.fr/idees-debats/cercle/quels-sont-les-impacts-du-big-data-sur-lentreprise-131763

14. LeHibou, "Fidélité Client et Big Data." URL : https://www.lehibou.com/communaute/fidelite-client-big-data

15. ODBMS.org, "ODBMS Industry Watch: Interview avec Jutta Bremm et Peter Palm.", URL : https://www.odbms.org/blog/2014/09/odbms-industry-watch-interview-jutta-bremm-peter-palm/

16. OpenClassrooms, "Choisissez votre famille NoSQL.", URL : https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462433-choisissez-votre-famille-nosql

17. LinkedIn, "How do you monitor and troubleshoot NoSQL performance issues ?", URL : https://www.linkedin.com/advice/0/how-do-you-monitor-troubleshoot-nosql-performance-issues

18. EasyVirt, "Consommation d'énergie des centres de données.", URL : https://www.easyvirt.com/consommation-energie-centres-de-donnees/#:~:text=La%20puissance%20consomm%C3%A9e%20est%20fonction,les%20serveurs%20%C3%A0%20deux%20prises

19. Ministère de la Transition Écologique, "Émissions de CO2 de l'industrie.", URL : https://www.statistiques.developpement-durable.gouv.fr/edition-numerique/chiffres-cles-du-climat/10-emissions-de-ges-de-lindustrie#:~:text=%C3%89missions%20de%20CO2%20pour,d%27%C3%A9lectricit%C3%A9%20dans%20l%27UE&text=Depuis%201990%2C%20les%20%C3%A9missions%20de,CO2%2FkWh%20en%202018

20. Ekwateur, "Prix de l'électricité en Europe.", URL : https://ekwateur.fr/blog/marche-de-l-energie/prix-electricite-europe-pays/

Liste des illustrations

Toutes les Figures

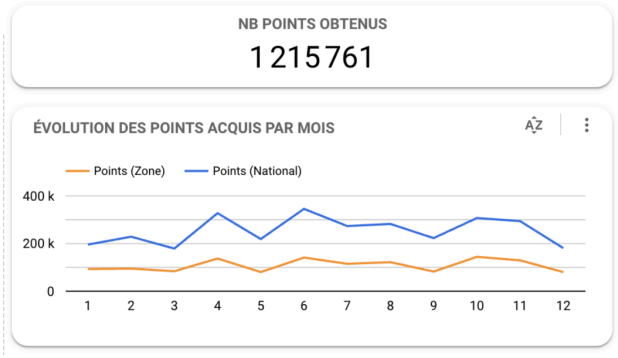


Figure 1 - Reporting simple

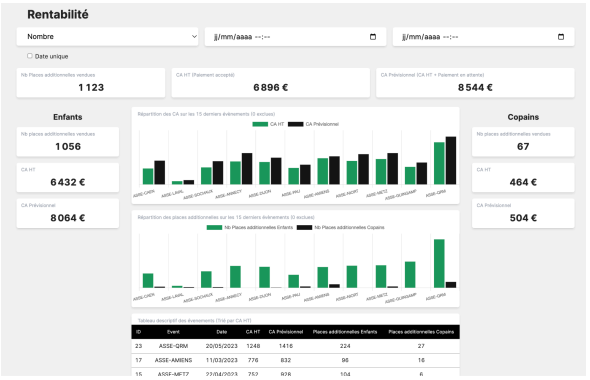


Figure 2 - Reporting complexe

```
<div class="dashboard" >
    <div class="dashboard-row" >
      // exemple : Widget KPI
    </div>
</div>
```

Figure 3 - Création d'une ligne

```
<div class="dashboard" >
    <div class="dashboard-row" >
      // exemple : Widget KPI 2
    </div>
    <div class="dashboard-row" >
      // exemple : Widget KPI 2
    </div>
</div>
```

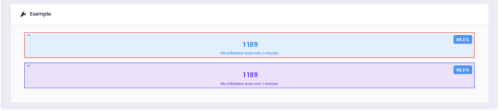


Figure 4 - Création de 2 lignes

```
<div class="dashboard" >
  <div class="dashboard-row" >
    <div class="three-column" >
      // exemple : 3 * Widget KPI
    </div>
  </div>
</div>
```

Figure 5 - Création de 3 colonnes

Colonne centrale : .MIDDLE-TEMPLATE

```
<div class="dashboard" >
  <div class="dashboard-row middle-template" >

    <div class="first-column one-line" >
      <widget-kpi
          :properties="{
                  'title': 'Label',
                  'theme': 'primary-soft',
              }"
      ></widget-kpi>
    </div>
    <div class="second-column two-line" >
      <widget-line
          :properties="{
                  'title': 'Title',
              }"
      ></widget-line>
    </div>
    <div class="third-column one-line" >
      <widget-kpi
          :properties="{
                  'title': 'Label',
                  'theme': 'primary-soft',
              }"
      ></widget-kpi>
    </div>

  </div>
</div>
```

Figure 6 - Mise en page d'un template avec 3 colonnes dont 1 colonne forte au milieu

Colonne gauche : .LEFT-TEMPLATE

```
<div class="dashboard" >
  <div class="dashboard-row left-template { null || size-1 || siz

    <div class="first-column one-line" >
      <widget-kpi
          :properties="{
                  'title': 'Label',
                  'theme': 'primary-soft',
              }"
      ></widget-kpi>
    </div>
    <div class="second-column two-line" >
      <widget-line
          :properties="{
                  'title': 'Title',
              }"
      ></widget-line>
    </div>

  </div>
</div>
```

Figure 7 - Mise en page d'un template avec 2 colonnes dont 1 colonne faible à gauche

Colonne droite : .RIGHT-TEMPLATE

```
<div class="dashboard" >
  <div class="dashboard-row right-template { null || size-1 || si

    <div class="first-column one-line" >
      <widget-kpi
          :properties="{
                  'title': 'Label',
                  'theme': 'primary-soft',
              }"
      ></widget-kpi>
    </div>
    <div class="second-column two-line" >
      <widget-line
          :properties="{
                  'title': 'Title',
              }"
      ></widget-line>
    </div>

  </div>
</div>
```

Figure 8 - Mise en page d'un template avec 2 colonnes dont 1 colonne faible à droite

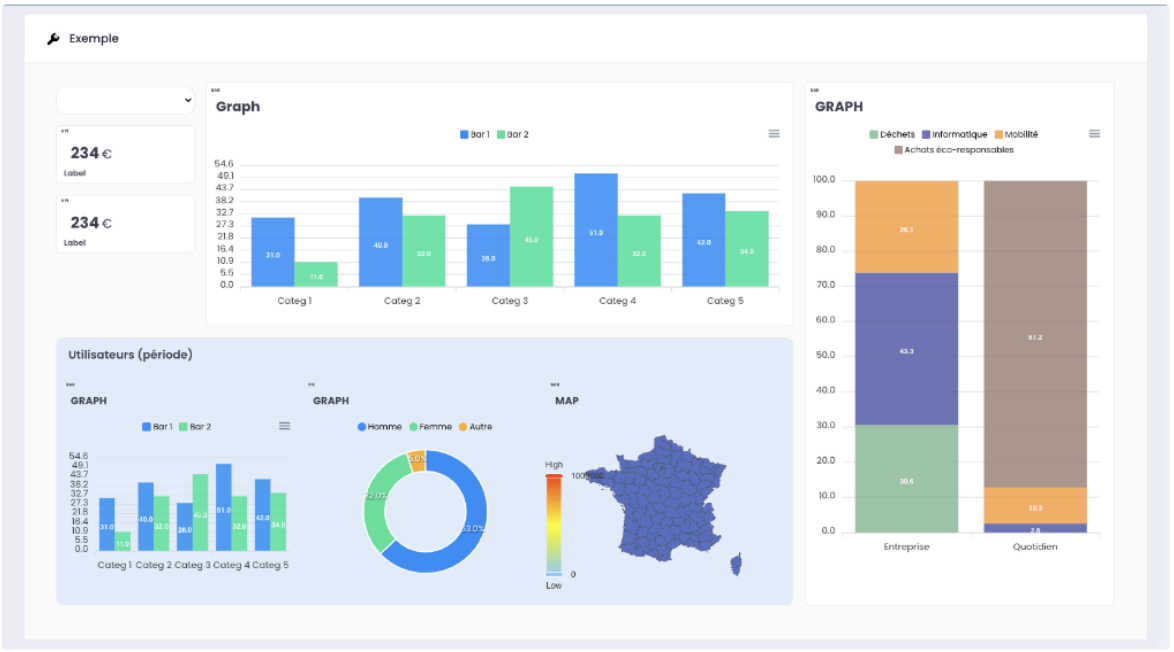


Figure 9 - Exemple de reporting complexe à développer

```
<div class="dashboard" >
    <div class="dashboard-row right-template" >
        <div class="first-column five-line" >
            <div class="dashboard-row left-template size-2" >
                <!-- Filter + KPI -->
                <div class="first-column four-line" >
                    <filter-select></filter-select>
                    <widget-kpi></widget-kpi>
                    <widget-kpi></widget-kpi>
                </div>

                <!-- Graph -->
                <div class="second-column five-line" >
                    <widget-bar
                        :properties="{
                                'title': 'Graph',
                            }"
                        :chart="{
                                'height':270,
                            }"
                    ></widget-bar>
                </div>
            </div>

            <!-- Users -->
            <div class="dashboard-row users" >
                <div class="title-section xl" >Utilisateurs (période)</div>
                <div class="three-column" style="gap:0px;" >
                    <widget-bar
                        :properties="{
                                'title': 'GRAPH',
                                'titleSize': 'lg',
                                'theme': 'nothing',
                                'evo_label': 'AVG',
                                'evo_theme': 'primary'
                            }"
                        :chart="{
                                'height':230,
                            }"
                    ></widget-bar>

                    <widget-pie
                        :properties="{
                                'title': 'GRAPH',
                                'titleSize': 'lg',
                                'theme': 'nothing'
                            }"
                        :chart="{
                                'type': 'donut',
                                'height':250,
                                'textColor': 'black',
                            }"
                    ></widget-pie>

                    <widget-map
                        :properties="{
                                'title': 'MAP',
                                'titleSize': 'lg',
                                'theme': 'nothing'
                            }"
                        :chart="{
                                'height':250,
                            }"
                    ></widget-map>
                </div>
            </div>
        </div>

        <!-- Graph -->
        <div class="second-column five-line" >
            <widget-bar
                :properties="{
                        'title': 'GRAPH',
                    }"
                :chart="{
                        'height':620,
                        'stacked': true,
                        'maxY': 100,
                    }"
            ></widget-bar>
        </div>
    </div>
</div>
```

Figure 10 - Code pour développer la Figure 7

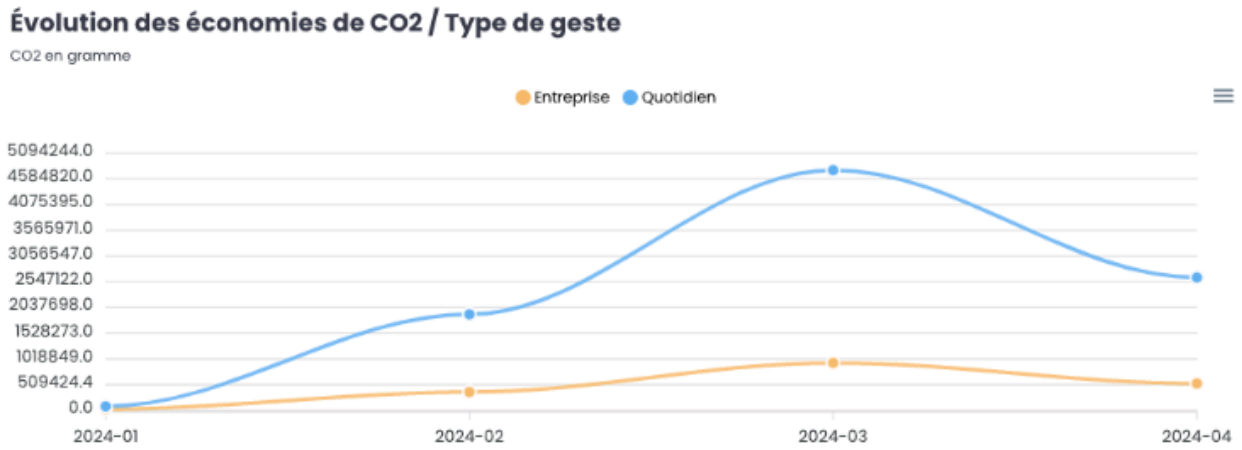


Figure 14 - Un widget d'un graphique en courbes

Figure 15 - L'entête du widget graphique en courbes

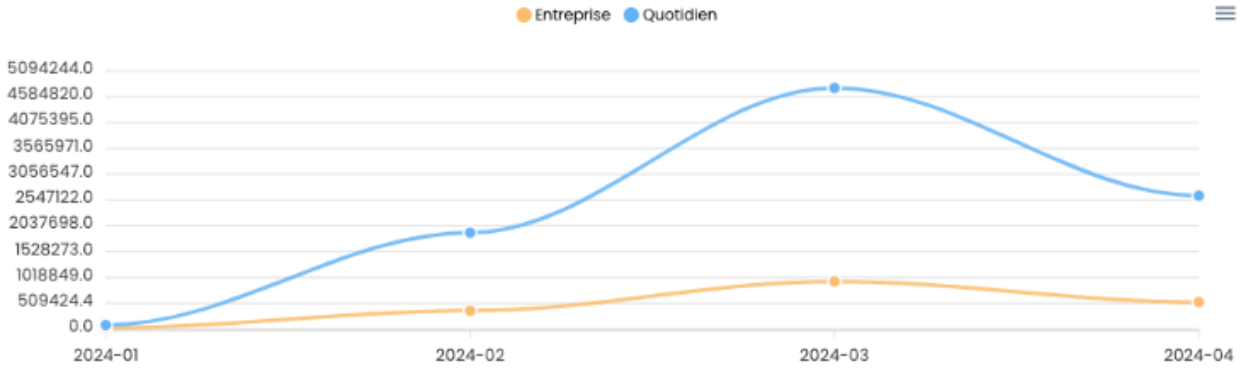


Figure 16 - Le contenu du widget graphique en courbes

```
<widget-[type]
    :properties="{
            'type': '...', // Info du type de widget (defaut: type du widget)
            'title': 'Text', // Titre du widget (defaut: null)
            'titleSize': '', // Taille du titre
            'titleFilter': 'varaible_select', // Titre dynamique à partir
                                              d'un filtre CONCAT(title;' ';valeur filtre[titleFilter])
            'description': '', // Description du widget
            'icon': '', // une icone sur le header du widget (fontawesome pour l'instant)
            'evoLabel': '', // label de la card évolution
        }"
    :nostore="[variable_select']" // desactivé la recharge des données si une variable change
    url_data="/manager/..." // liens des requetes
></widget-[type]>
```

Figure 17 - Modification d'un entête

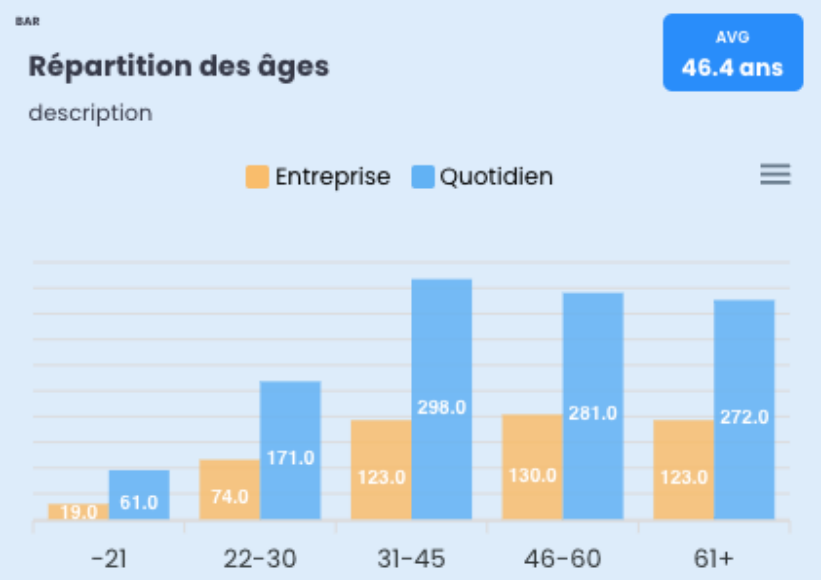


Figure 18 - Widget calibré

```
<widget-kpi
  :properties="{
          'titleSize': '', // theme du tritre
          'titleTheme': '', // theme du tritre
          'theme': '', // theme de la card
          'style': '', // (default) weak or strong
          'evoTheme': '', // theme de la card évolution
      }"
  :class_content="" // pour ajouter des class Tailwind CSS dans le widget
></widget-kpi>
```

Figure 19 - Code des autres variables de style d'un widget

```
public function distinct_users(Request $request, UserRepository $userRepository): array
{
    $params = $this->init($request);

    $users_all = $userRepository->users_all($params['date_end']);

    $value = DB::table('contributions')->join('users as u', 'u.id', 'contributions.id_user')
        ->where('u.role', 'Participant')
        ->join('eco_missions as m', 'm.id', '=', 'contributions.id_mission')
        ->whereIn('m.destination', ['Tous', 'RSE'])
        ->whereIn('m.type', $params['gestes'])
        ->whereBetween('contributions.created_at', [$params['date_start'], $params['date_end']])
        ->distinct('contributions.id_user')->count();

    return [
        'value'=>$this->functionRepository->formatNumber($value, 0),
        'evo_data' => ($value / ($users_all > 0 ? $users_all : 1) * 100),
        'evo_suffix' => '%',
    ];
}
```

Figure 20 - Code d'un endpoint

```php
Codiumate: Options | Test this function
Route::prefix('dashboard')->group(function () {
    Route::prefix('categ_request')->group(function () {
        Route::get('/distinct_users', [CategoryRepository::class, 'distinct_users'])->name('dashboard.categ.distinct_users');
    });
});
```

Figure 21 - Attribution de la fonction sur une route

```
<widget-kpi
    :properties="{
            'type':' ',
            'title': 'Nb utilisateur avec min. 1 mission',
            'theme': 'primary-soft',
            'evoTheme': 'primary'
        }"
    class_content="text-center"
    url_data="/manager/dashboard/categ_request/distinct_users"
></widget-kpi>
```

Figure 22 - Code pour attribuer un endpoint

Figure 23 - Résultat du endpoint

```
<filter-date
    size="sm"
    filter="date_start"
    label="Date de début"
></filter-date>
<filter-date
    size="sm"
    filter="date_end"
    label="Date de fin"
></filter-date>
```

Figure 24 - Intégration de filtres pour une période

Figure 25 - Résultat de l'intégration

```
<filter-select
    filter_select="type_gestes_select"
    store_data="type_gestes_list"
    label="Type de geste"
></filter-select>
```

Figure 26 - Intégration d'un filtre select

```
<filter-select
    filter_select="variable_select"
    store_data="variable_list" ou url_data = '/filters/...' // url_data prioritaire
    :multiple="true" // plusieurs possibilités
></filter-select>
```

Figure 27 - Schéma d'initialisation d'un filtre select

```
export const useFilterStore = defineStore('filters', {
    state: () => {
        const currentDate = new Date();
        const dateStart = new Date(currentDate.setDate(currentDate.getDate() - 90)).toISOString().split('T')[0];

        return {
            date_start: dateStart,
            date_end: new Date().toISOString().split('T')[0],

            // General
            point_geste_select: 'Éco-gestes',
            point_geste_list: ['Éco-gestes', 'Éco-points'],
        };
    },
    actions: {
        setData(name, value) {
            this[name] = value;
        },
    },
})
```

Figure 28 - Exemple de code pour le stockage des variables

Figure 29 - Tableau de bord Général finalisé pour la démo éco-fidélité RSE

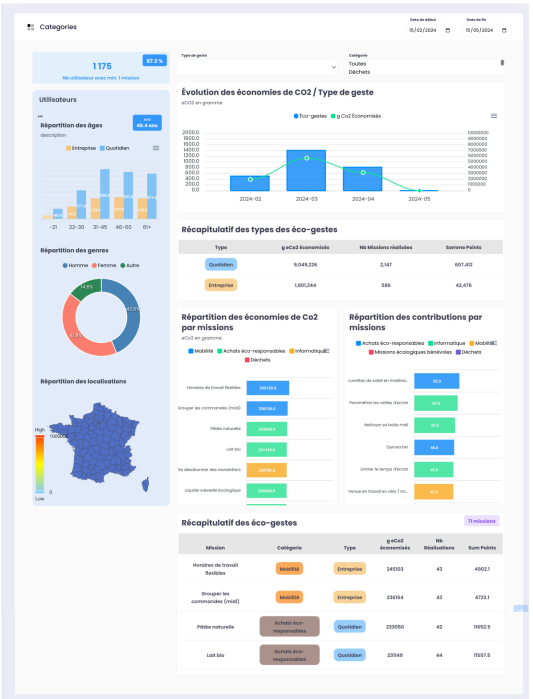


Figure 30 - Tableau de bord Catégorie finalisé pour la démo éco-fidélité RSE

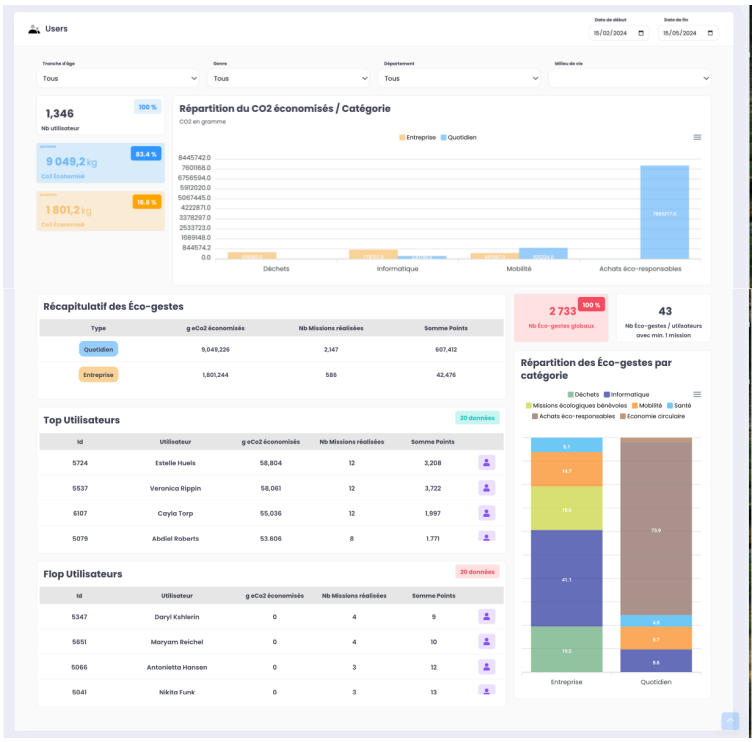


Figure 31 - Tableau de bord Utilisateurs finalisé démo éco-fidélité RSE

```php
public function migrate(CollectionController $collection) {
  $table = 'users'; // table à migrer
  $table_fields = ['id', 'role', 'lastname', 'firstname']; /* champs à garder pour la table à migrer,
                                                              si [], tout est pris */

  $type = 'nosql'; // possibilité de choisir la destination nosql => mongodb sinon SGBD classic

  $schema = [
    [ 'type' => 'all', 'table' => 'contributions', 'alias' => 'c', 'tableKey' => 'id_user',
      'originKey' => 'id', 'count' => true ],
    [ 'type' => 'join', 'table' => 'user_data_values', alias' => 'birthday', 'originKey' => 'id',
      'tableKey' => 'user_id', 'tableFields' => ['value'], 'conditions' => "birthday.key = 'birthday'" ],
    [ 'type' => 'join', 'table' => 'user_data_values', 'alias' => 'gender', 'originKey' => 'id',
      'tableKey' => 'user_id', 'tableFields' => ['value'], 'conditions' => "gender.key = 'gender'" ],
  ];

  $aggregate = [
    [ 'table' => 'contributions', 'alias' => 'c',
      'originKey' => 'id', 'tableKey' => 'id_user',
      'calculs' => [
        'id' => ['count'],
        'co2_economise' => ['sum', 'avg'],
        'points' => ['sum', 'avg'],
        'id_mission' => ['count_distinct'],
        'id_categorie' => ['count_distinct']
      ] ],
    [ 'table' => 'achats', 'alias' => 'a',
      'originKey' => 'id', 'tableKey' => 'id_user',
      'calculs' => [
        'id' => ['count'],
        'qte' => ['sum', 'avg'],
        'prix_final' => ['sum', 'avg'],
        'id_produit' => ['count_distinct'],
        'id_mission' => ['count_distinct']
      ] ],
  ];

  return $collection->migrate_table($table, $table_fields, $schema, $aggregate,
    $drop (default false), $rebuild (default false) );
  /* $drop = on supprime toutes les données avant de remplire à nouveau la base
     | $rebuild = on supprime la base avant de la refaire */
}
```

Figure 32 - Code commenté pour réussir une migration

```php
public function timeseries(CollectionController $collection) {
    $table = 'users';
    $type = 'nosql'; // possibilité de choisir la destination nosql => mongodb sinon SGBD classic

    $time = [
      'day', // Type de periode : day  | month | year
      '2023-04-16', // Start période (default = null) si null il prend 2023-01-01
      '2024-04-18' // End période (default = null) si null il prend la Carbon::now()
    ];

    $schema = [
      [
        'users', 'u',
        [
          'id' => ['count'],
          'id_departement' => ['count_distinct'],
          'densite' => ['avg'],
        ]
      ],
      [
        'contributions', 'c',
        [
          'id' => ['count'],
          'co2_economise' => ['sum', 'avg'],
          'points' => ['sum', 'avg'],
          'id_mission' => ['count_distinct'],
        ]
      ],
    ];

    return $collection->migrate_timeseries($type, $table, $schema, $time,
        $drop (default false), $rebuild (default false) );
    /* $drop = on supprime toutes les données avant de remplire à nouveau la base
      | $rebuild = on supprime la base avant de la refaire */
  }
```

Figure 33 - Code commenté pour réussir une migration

Exemples de requêtes

SQL	NoSQL
CREATE TABLE timeseries_invoices AS SELECT DATE_FORMAT(i.created_at, '%Y-%m-%d') AS jour, DATE_FORMAT(i.created_at, '%M') AS mois, COUNT(DISTINCT i.user_id) AS nb_user, SUM(CASE WHEN i.status = 3 THEN d.qte ELSE 0 END) AS sum_qte, SUM(CASE WHEN i.status = 3 THEN d.amount ELSE 0 END) AS sum_ca, SUM(CASE WHEN i.status = 3 THEN 1 ELSE 0 END) AS nb_invoices_valid, SUM(CASE WHEN i.status = 1 THEN 1 ELSE 0 END) AS nb_invoices_refuse, SUM(CASE WHEN i.status = 0 THEN 1 ELSE 0 END) AS nb_invoices_new, SUM(CASE WHEN i.status = 2 THEN 1 ELSE 0 END) AS nb_invoices_error, SUM(1) AS nb_invoices FROM table_invoices i INNER JOIN table_invoice_details d ON d.invoice_id = i.id GROUP BY jour	SELECT * FROM timeseries_invoices

Gains	SQL	NoSQL
Lignes	993 043	365
Temps	13,2s	0,009s

Tableau des gains

Formules

Jour SQL	N JOUR * X USER REQUÊTES * SQL TEMPS REQUÊTES
Jour NoSQL	(N JOUR * X USER REQUÊTES * NOSQL TEMPS REQUÊTES) + (N JOUR * SQL TEMPS REQUÊTES)

Figure 34 - Contexte des calcules

Tous les tableaux

	Développer en interne	Externalisation
Avantages	• Contrôle complet sur la personnalisation et l'intégration avec les systèmes existants • Amélioration des compétences internes et de la capacité à réagir rapidement aux changements de besoins • Moyen terme, possibilité d'amortir complètement l'investissement	• Accès à une expertise spécialisée • Réduction des coûts à long terme liés à la maintenance et aux mises à jour • Délais de développement potentiellement plus courts
Inconvénients	• Coûts initiaux plus élevés pour le développement et la formation • Risque de rallonger les délais de développement si les ressources ne sont pas suffisamment qualifiées	• Baisse en qualité, car moins sur-mesures • Moins de contrôle sur les processus de développement • Dépendance vis-à-vis du fournisseur pour les mises à jour et les personnalisations

Tableau 1 - Comparaison des solutions offertes à letmotiv

	Avantages	Inconvénients
Logiciels SaaS	• Récupération facile des données des API (Meta, Google Analytics, Twitter) • Traitements statistiques supplémentaires • Beaucoup de fonctionnalité liée aux données • Reporting en mode click-and-drop	• Intégration complexe des dashboards • Dashboard moins polyvalent et flexible • Compétences et dev pour pouvoir exploiter dans nos solutions ces traitements • Coûts très élevés • Complexités d'utilisation avec nos technologies
Packages Back Office	• Amélioration des packages comme on le souhaite • Méthode CRUD directement intégrée • Faciliter de création de manager • Widgets prêts en main • Base Echarts, ChartsJS, HighCharts (Facile à en créer d'autres graphes)	• Codage requis pour les faire fonctionner • Grosse configuration requise • Connaissance du langage pour l'intégrer à des outils externes aux packages
Packages Widgets	• Package avec l'essentiel requis • Coûts bas	• Conçu pour créer des graphiques moins pour des dashboards complexes • Créé par la communauté • Peu de fonctionnalités annexes

Tableau 2 - Comparaison des possibilités d'externalisations

	Licence / Prix	Utilisation avec Laravel	Offres
Power BI	Élevé	Package utilisable	• Tableaux de bord avancés, • Intégration de diverses sources de données
Tableau Software	Très élevé	Package utilisable	• Options de visualisation puissantes, • Bonne gestion des grands ensembles de données
Google Suite	Modéré	Package utilisable	• Intégration avec les services Google, • Facilité d'utilisation
Post Hog	Modéré	Inclus	• Analyse / Dashboard • Tracker Web • Back-Up • Test BDD • Data Warehouse • Pipeline / Profilage
Amplitude	Élevé	Package semi-complet	• Dashboard • Data Analyse User • Tracker Web • Back-Up • Test BDD • Data Warehouse • Pipeline / Profilage
MixPanel	Élevé	Package semi-complet	• Dashboard • Data Warehouse • Pipeline / Profilage • Analyse événementielle, • Tracking d'interactions
Databricks, Snowflack	Très élevé	Endpoint	• Plateforme d'analyse basée sur Apache Spark, • Scalabilité

Tableau 3 - Comparaison de toutes les solutions SaaS

	Licence / Prix	Offres
Laravel Nova	Privée • 99€ / projet	• Manager • Dashboards • Widgets • Filtres • CRUD • Recherche
BackPack	Privée • 69€ / projet	• Manager • Dashboards • Widgets • Filtres • CRUD • Plug-in inclus
Orchid	Open Source • Gratuit	• Manager • Dashboards • Widgets • CRUD
Filament	Open Source • Gratuit	• Manager • CRUD • Dashboards • Widgets • Filtres • Recherche • Market place plug-in communautaire

Tableau 4 - Comparaison de tous les packages Back Offices

	Licence / Prix	Offres
Laravel Charts	Open Source • Gratuit	• Widgets • Filtres
Lavacharts	Open Source • Gratuit	• Widgets • Filtres
Laravel ApexCharts	Privée	• Widgets

Tableau 5 - Comparaison de tous les packages de widgets

`titleSize` :

Valeurs	Content class
DEFAULT	`text-2xl`
sm	`text-sm`
lg	`text-lg`
xl	`text-xl`

Tableau 6 - Classe prise en compte pour la variable titleSize par le package (classes Tailwind CSS)

`titleTheme` :

Valeurs	Content class
DEFAULT	`text-black`
custom	color: `white`;
PRIMARY	color: `#3699FF`;
SUCCESS	color: `#1BC5BD`;
INFO	color: `#8950FC`;
WARNING	color: `#FFA800`;
DANGER	color: `#F64E60`;

Tableau 7 - Classe prise en compte pour la variable titleTheme par le package (classes Tailwind CSS)

`theme` :

Valeurs	Content class
`DEFAULT` light	`text-black bg-white`
dark	color: `white`;background-color: `#1a202c`;
`nothing`	`border-none bg-transparent`
custom	color: `white`;background-color: `#283747`;
`PRIMARY`	color: `white`; background-color: `#3699FF`;
`PRIMARY-SOFT`	color: `#3699FF`; background-color: `#E1F0FF`;
`SUCCESS`	color: `white`; background-color: `#1BC5BD`;
`SUCCESS-SOFT`	color: `#1BC5BD`; background-color: `#C9F7F5`;
`INFO`	color: `white`; background-color: `#8950FC`;
`INFO-SOFT`	color: `#8950FC`; background-color: `#EEE5FF`;
`WARNING`	color: `white`; background-color: `#FFA800`;
`WARNING-SOFT`	color: `#FFA800`; background-color: `#FFF4DE`;
`DANGER`	color: `white`; background-color: `#F64E60`;
`DANGER-SOFT`	color: `#F64E60`; background-color: `#FFE2E5`;

Tableau 8 - Classe prise en compte pour la variable theme par le package (classes Tailwind CSS)

`evoTheme` :

Valeurs	Content class
`DEFAULT`	`bg-light-gray text-black`
`nothing`	`bg-transparent`
`PRIMARY`	color: `white`; background-color: `#3699FF`;
`PRIMARY-SOFT`	color: `#3699FF`; background-color: `#E1F0FF`;
`PRIMARY-TEXT`	color: `#3699FF`; `bg-transparent`
`SUCCESS`	color: `white`; background-color: `#1BC5BD`;
`SUCCESS-SOFT`	color: `#1BC5BD`; background-color: `#C9F7F5`;
`SUCCESS-TEXT`	color: `#1BC5BD`; `bg-transparent`
`INFO`	color: `white`; background-color: `#8950FC`;
`INFO-SOFT`	color: `#8950FC`; background-color: `#EEE5FF`;
`INFO-TEXT`	color: `#8950FC`; `bg-transparent`
`WARNING`	color: `white`; background-color: `#FFA800`;
`WARNING-SOFT`	color: `#FFA800`; background-color: `#FFF4DE`;
`WARNING-TEXT`	color: `#FFA800`; `bg-transparent`
`DANGER`	color: `white`; background-color: `#F64E60`;
`DANGER-SOFT`	color: `#F64E60`; background-color: `#FFE2E5`;
`DANGER-TEXT`	color: `#F64E60`; `bg-transparent`

Tableau 9 - Classe prise en compte pour la variable evoTheme par le package (classes Tailwind CSS)

`style` :

Valeurs	Content class
`DEFAULT` weak	`rounded-md`
strong	`rounded-xl shadow-2xl`

Tableau 10 - Classe prise en compte pour la variable style par le package (classes Tailwind CSS)

Table 30k lignes	Sequel Pro (SQL Classic)	MongoDB
Poids fichier	5,5 MB	0,5 MB MEILLEUR
Create	`User::create([ 'name' => 'Admin', 'email' => 'admin@letmotiv.io', 'password' => '...', ]);` Temps : 86 MS	`UserMongoDB::create([ 'name' => 'Admin', 'email' => 'admin@letmotiv.io', 'password' => '...', ]);` Temps : 55 MS
Read	`User::where('email', 'kylian@letmotiv.io')->get();` Temps : 27 MS (début 153 ms)	`UserMongoDB::where('email', 'kylian@letmotiv.io')->get();` Temps : 19,4 MS (début 114 ms)
Update	`User::where('email', 'kylian@letmotiv.io')->update(['name' => 'Admin']);` Temps : 47 MS	`UserMongoDB::where('email', 'kylian@letmotiv.io')->update(['name' => 'Admin']);` Temps : 55 MS
Delete	`User::where('email', 'admin@letmotiv.io')->delete();` Temps : 48 MS	`UserMongoDB::where('email', 'admin@letmotiv.io')->delete();` Temps : 55 MS

Tableau 11 - Comparatif des différents logiciels pour la méthode CRUD

Jours	H de server (SQL)	H de server (NoSQL)	H de server économisées (SQL - NoSQL)	kWh économisés	Kg Co2e économisés	€ Electricités économisés
30	1 100	0,977	1 099	401,14	127,16	93,67
60	2 200	1,953	2 198	802,29	254,32	187,33
90	3 300	2,930	3 297,1	1 203,43	381,49	281
120	4 400	3,907	4 396,1	1 604,57	508,65	374,67
150	5 500	4,883	5 495,1	2 005,72	635,81	468,34
180	6 600	5,860	6 594,1	2 406,86	762,97	562
210	7 700	6,837	7 693,2	2 808,01	890,14	655,67
240	8 800	7,813	8 792,2	3 209,15	1 017,30	749,34
270	9 900	8,790	9 891,2	3 610,29	1 144,46	843
300	11 000	9,767	10 990,2	4 011,44	1 271,62	936,67
330	12 100	10,743	12 089,3	4 412,58	1 398,79	1 030,34
360	13 200	11,72	13 188,3	4 813,72	1 525,95	1 124

Tableau 12 - Comparatif des économies potentielles en passant sur du NoSQL