LIVRE BLANC

LES CMS À L'ÈRE DES MICROSERVICES

Impacts et usages

KALIOP
INTERACTIVE MEDIA

I - Introduction

Dans un monde numérique en évolution permanente, le développement des applications multimédia et des sites Internet n'échappe pas à la règle. Aujourd'hui, une toute nouvelle approche émerge dans la communauté des développeurs pour les systèmes de gestion de contenu. D'un CMS (*content management system*) monolithique traditionnel, on tend de plus en plus vers une **architecture microservices**.

Cette nouvelle conception offre une couverture fonctionnelle et applicative plus riche : **l'architecture Serverless** ou sans serveur permet ainsi de descendre au niveau le plus fin de vos besoins.

Mais les microservices matérialisent-il pour autant la fin des solutions CMS « classiques », comme Drupal, eZ Platform ou WordPress au profit des CMS « *As A Service* » comme Contentful ou Prismic ?

Examiner cette question implique d'abord de **découvrir et comprendre le fonctionnement de l'approche microservices** ou Serverless, ainsi que la couverture technique et fonctionnelle de ses différentes solutions. Nous comparerons également ces dernières avec les outils traditionnels comme Drupal, eZ Platform et Wordpress, en s'interrogeant sur leur valeur ajoutée, mais aussi leurs limites. Sans oublier que les CMS classiques n'ont pas dit leur dernier mot et apportent dorénavant des **solutions en mode dit *« Headless »***, qui peuvent s'avérer très intéressantes et pertinentes.

II - Du CMS monolithique aux microservices : changement de paradigme

1. Le confort des CMS traditionnels

Aujourd'hui, les CMS traditionnels présents sur le marché (Drupal, WordPress ou eZ Platform) présentent un panel de **fonctionnalités assez standard**, attendues tant par les développeurs que les utilisateurs :

- la gestion des informations : gestion éditoriale, des médias et des utilisateurs, le tout accompagné par des fonctionnalités de recherche avancées,
- des modèles (templates) et fonctionnalités facilement personnalisables,
- la possibilité de créer facilement des plateformes e-commerce, d'utiliser des APIs et d'étendre les fonctionnalités initiales du site ou de l'application.

De plus, pour refondre ou créer un site Web ou une application, vous attendez généralement d'un **CMS classique qu'il vous offre une solution tout-en-un**, claire et simple à mettre en place. Ainsi, un CMS doit posséder plusieurs atouts :

- rapidité de mise en œuvre,
- facilité de déploiement : le travail de développement est limité au profit du paramétrage et de la personnalisation de la solution,
- large couverture fonctionnelle : le CMS doit permettre de gérer l'ensemble de vos besoins et informations.

LES FACTEURS DE CHOIX D'UN CMS

Vous privilégiez une solution en fonction de :

- vos applications et sites existants,
- la couverture la plus complète possible de vos besoins,
- la dimension technique : langages et frameworks maîtrisés par vos équipes, compatibilité avec les technologies utilisées en interne, etc.

Pour couvrir le plus largement possible vos besoins, les CMS classiques tentent d'être performants dans tous les domaines. En réalité, ils brillent sur quelques fonctionnalités (généralement, leur cœur de métier historique), mais présentent souvent des limites pour toutes les autres.

Afin de combler ces lacunes, **des outils tiers ont émergé**, inventant parfois de nouvelles fonctionnalités. Ils constituent initialement des **briques qui viennent compléter les CMS traditionnels**, de manière de plus en plus complexe, ouvrant l'ère des microservices.

2. Le renouveau de l'approche microservices

Dans le cadre d'un projet de site Internet ou d'application, une stratégie microservices permet d'élaborer une solution qui répond parfaitement à vos enjeux et besoins. Une démarche impliquant une **nouvelle approche** et la déconstruction préalable de toute réflexion basée sur un CMS classique.

Ainsi, on ne recherche plus un CMS unique et exhaustif, mais **une ou plusieurs briques répondant à chaque besoin** ou fonctionnalité attendue. Il s'agit donc d'une **stratégie *API first*** (*Application Programming Interface* ou interface de programmation applicative).

LA STRATÉGIE API FIRST : LES ATOUTS

▸ Construire une solution globale en **rassemblant différents outils spécifiques via leurs APIs pour répondre à chaque besoin** de manière optimale.

▸ Ouvrir de nouveaux horizons en créant des solutions sur-mesure : les possibilités quasi-infinies permettent d'aller plus loin qu'avec un CMS classique.

Gestion de médias (images, vidéos...)	Cloudinary
Gestion d'utilisateurs (données, authentification...)	WSO2
E-commerce (gestion des produits, commandes...)	shopify, CRYSTALLIZE
Moteur de recherche efficace	algolia, Elastic Site Search
Accélération des performances (génération de sites statiques à partir de contenus dynamiques...)	netlify
Fonctionnalités sur-mesure (scripts, formulaires...)	Google Cloud Functions, AWS Lambda

Changer d'approche en passant d'un CMS classique aux microservices implique de **concevoir différemment sa stratégie de création et de gestion** de site ou d'application.
Cette démarche vous permet de vous concentrer sur vos besoins sans être limité par une solution monolithique, car **il existe autant de services et APIs que de fonctionnalités possibles**.

III - Les CMS Serverless et leur architecture

1. Microservices et Serverless

On associe souvent la **notion de microservices à celle de *Serverless***, même si elles ne se recouvrent pas complètement. De fait, si un CMS classique et toutes ses fonctionnalités (base de données, moteur de recherche, langages...) peuvent être installés sur quelques serveurs, chacun des microservices ayant ses besoins propres, il serait contre productif de tous les installer et les gérer soi-même. On recourt donc à des **outils prêts à l'emploi, à choisir et configurer directement en ligne**. L'installation, le déploiement et la vie de chaque microservice sont donc délégués à une solution tierce, dite Serverless.

LES AVANTAGES DU SERVERLESS

- Pas de gestion de l'installation des services.
- Pas de gestion du déploiement des services.
- Outils prêts à l'emploi à connecter via des APIs.

2. Le CMS à l'ère des microservices : les exemples de Prismic et Contentful

Dans le cadre des microservices, la **gestion de contenu via un CMS (la « brique » principale de gestion du système) peut aussi être considérée comme un service**. Il s'agit là de l'approche *Content as a Service* (CaaS), représentée notamment par Prismic et Contentful.

Ces deux acteurs majeurs proposent des services proches, avec tout ce que l'on peut attendre d'un CMS moderne :

- un modèle de contenu et une structure de l'information totalement personnalisables,
- la gestion des médias avec un CDN intégré pour les images,
- un service full API,
- l'intégration du langage GraphQL pour gérer les contenus,
- de nombreux SDK,
- l'édition personnalisable, permettant d'étendre les backoffices.

Comme beaucoup de microservices, Prismic et Contentful couvrent de très nombreux SDK ou kits de développement logiciel, afin d'accélérer le développement de vos applications, quel que soit l'environnement technologique. Un atout évitant de se soucier de la technologie des solutions utilisées : les microservices sont choisis pour leur excellence dans un domaine, puis connectés via une API et des SDK incluant de nombreux langages (PHP, Ruby, JavaScript...).

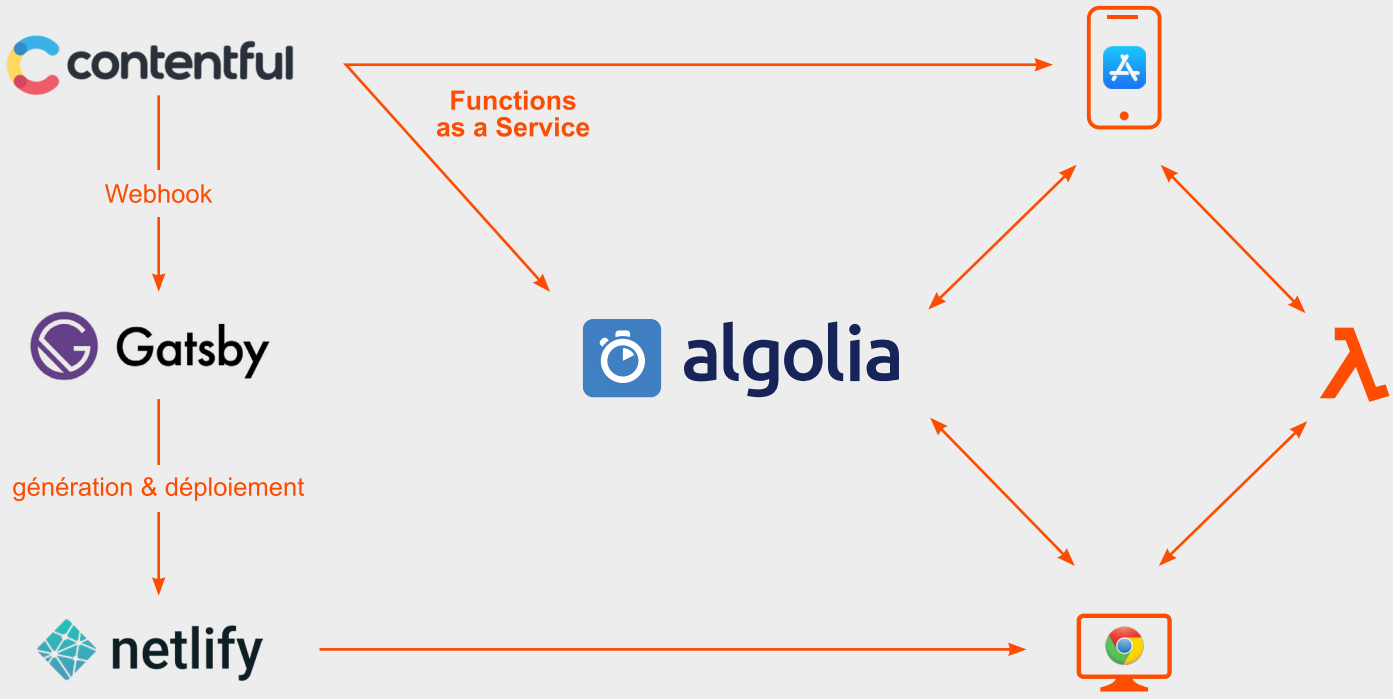
Les fonctionnalités coup de cœur de nos experts

prismic.io	contentful
Publications groupées : gestion des workspaces et des releases en une seule fois, programmables à une date particulière.	**Technique :** gestion du cycle de vie et de déploiement efficace et fluide, limitant les interactions dans l'interface au profit du code.
Slices : groupes d'attributs réutilisables pour des pages dynamiques.	**API d'édition :** possibilité d'éditer directement depuis vos outils.

3. Quelle architecture pour organiser ses services ?

Une fois les différents microservices identifiés et paramétrés, se pose la question d'utiliser et de faire fonctionner ensemble toutes ces briques.

Exemple 1 : architecture d'un site de contenu et de son application mobile

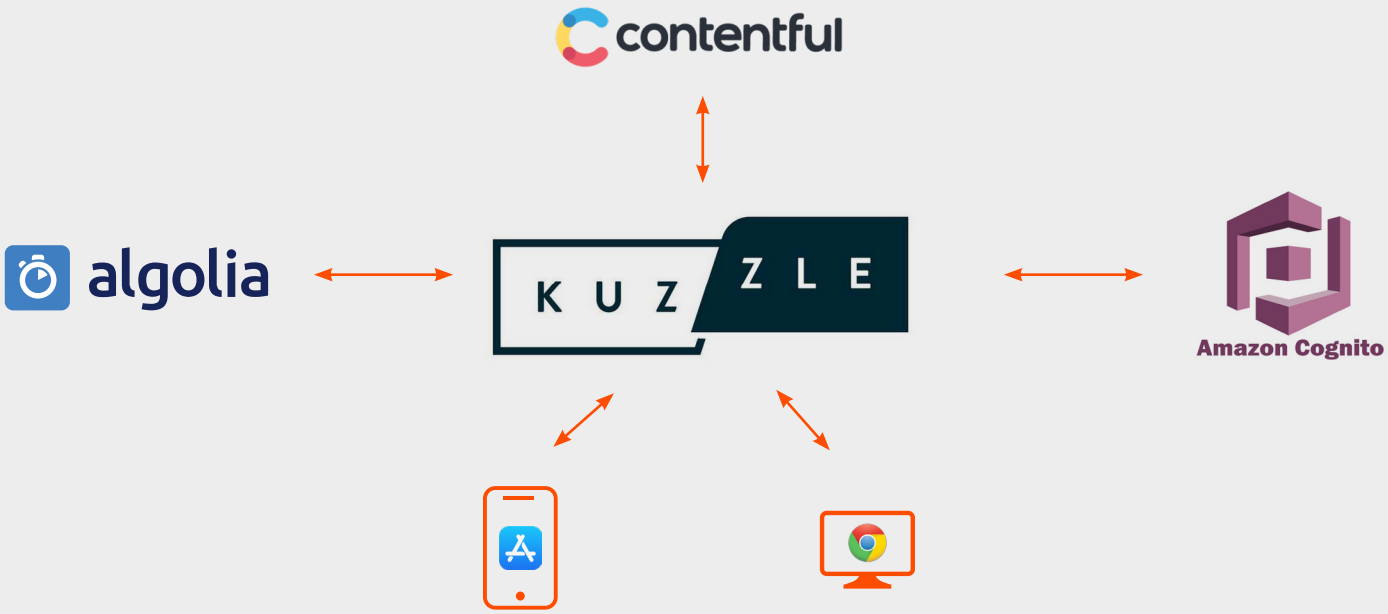


Pour fournir du **contenu à l'application mobile, Contentful peut être directement interrogé**, sans étape intermédiaire. Le contenu géré par Contentful est aussi envoyé pour indexation sur Algolia en utilisant une fonction Lambda entre les deux.

Pour créer un site Internet avec des mises à jour régulières, on utilise ensuite un **framework comme Gatsby**, permettant de générer et de déployer le site au travers d'une solution comme Netlify. Ainsi, à chaque fois qu'un nouveau contenu est publié dans Contentful, **un message est envoyé via un webhook à Netlify**, qui génère le site et le met à disposition en statique.

On constate un retour des sites statiques lorsque le contenu est peu souvent publié ou modifié, facilitant l'hébergement et l'utilisation d'outils simples à mettre en œuvre comme Netlify et répondant aux exigences du SEO.

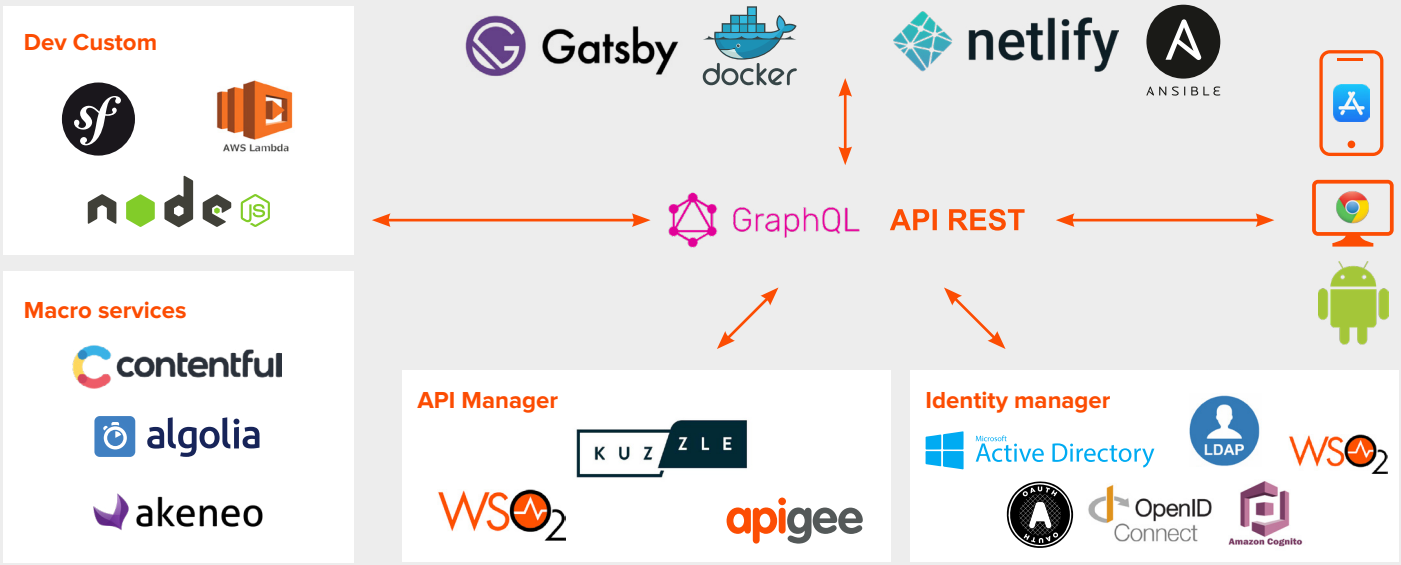
Exemple 2 : architecture d'un site de contenu avec des fonctionnalités métier



Pour élaborer un site de contenu public et privé doté de fonctionnalités métier, on introduit un **middleware** (Serverless ou sous forme de développement classique), permettant une **connexion avec des outils tiers** (dont ceux de l'entreprise) et une **coordination de leurs différentes fonctionnalités**. Ici, le middleware Kuzzle centralise les fonctionnalités d'Algolia, Amazon Cognito et Contentful avant de générer le site et l'application mobile.

4. Une brique centralisée pour gérer efficacement son site

Comment tout ça s'orchestre ?



Le fait d'utiliser plusieurs briques et donc différentes technologies, serveurs... peut poser des **problèmes d'orchestration de l'ensemble**. Au contraire d'une solution monolithique comme Drupal, le monitoring, le débogage, la gestion des droits et le déploiement des services risquent d'être démultipliés par le nombre de fonctionnalités à gérer.

C'est pourquoi l'exemple précédent montre bien l'intérêt de mettre en place une **brique centralisée pour la gestion et l'orchestration de tous les microservices**. De même, des middlewares peuvent être utilisés pour gérer les droits des utilisateurs (*Identity Management*) et les APIs (*API Management*).

QU'EST-CE QUE L'API MANAGEMENT ?

Permettant de faire communiquer correctement plusieurs APIs ensemble, l'API Management est une brique ou un middleware de votre système. La gestion des APIs de vos différents outils est ainsi regroupée et reliée directement à la brique centralisée.

La flexibilité et la gestion très fine des besoins sont les principaux atouts des microservices. Si le système mis en place peut sembler complexe, il offre la possibilité de faire appel uniquement aux microservices et aux briques qui vous sont réellement utiles. De même, il est tout à fait envisageable de débuter avec seulement quelques fonctionnalités puis de faire évoluer votre système progressivement en fonction de vos besoins.

III - Les CMS traditionnels se déclinent en mode “headless”

Face au développement des microservices, les systèmes de gestion de contenu traditionnels, majoritairement utilisés sur le marché, ne sont pas en reste.

1. Des APIs et connecteurs avancés

Les CMS comme eZ Platform, Drupal ou encore Wordpress **développent des APIs assez performantes, afin d’interagir avec le contenu**. Il est alors possible d’utiliser des connecteurs de plus en plus avancés qui permettent de changer de moteur de recherche, de recourir à une gestion de médias externe, etc.

> Par exemple, une gestion d’identité séparée du CMS est utile lorsque ces identités sont utilisées par différents services au sein d’une organisation.

eZ Platform, Drupal ou WordPress se positionnent aussi de plus en plus sur des **solutions headless**. Elles offrent l’avantage de pouvoir afficher les contenus non seulement sur le CMS, mais aussi au sein de moteurs de templates tiers.

> Par exemple, il est possible de développer une application React Native pour faire appel à plusieurs sources de contenu, et de l’utiliser avec l’API WordPress en front.

2. Des briques externalisées

En pratique, les CMS développent leur **compatibilité avec des briques et microservices présents sur le marché**. Vous pouvez ainsi conserver votre CMS, ainsi que vos bases de données, vos solutions de gestion d’utilisateurs ou de médias, etc., en les interconnectant.

> Par exemple, si vos besoins en médias sont très forts, il sera judicieux d’utiliser un système de médiathèque externalisé, connecté ensuite à l’ensemble de vos outils de publication de contenu, besoins métier, applications mobiles, etc.

Ensuite, la mise en œuvre du CMS implique d’**installer et de customiser le thème et les modules choisis**, avant d’effectuer les premiers tests fonctionnels. Après les développements spécifiques qui en découlent, vous pourrez alors procéder à une « recette » pour vérifier que le projet correspond à vos exigences initiales.

3. Qu’en est-il de l’Open Source ?

Face au développement des briques et microservices propriétaires, l’esprit et les solutions Open Source n’ont pas pour autant disparu. D’ailleurs, même au sein des outils propriétaires ou SaaS, il est possible de bénéficier de modes de fonctionnement assez souples et de recourir à des **systèmes de portabilité pour externaliser les données**, les importer et les exporter assez facilement. Ainsi, vous conservez une certaine liberté dans la gestion de vos données.

V - Conclusion

Les microservices permettent aujourd'hui de **répondre de manière très fine à chacun des besoins que vous avez identifiés**, en sélectionnant les meilleurs outils pour y parvenir. Interconnectés, connectés à des solutions existantes et/ou centralisés, ils offrent une alternative fonctionnelle et évolutive aux CMS monolithiques.

De fait, ces outils vont assez loin dans les fonctionnalités proposées. Couplés avec un mode de fonctionnement Serverless ou SaaS, ils impliquent donc **un certain coût de licence**. Il importe néanmoins de le mettre en regard des coûts de développement, de maintenance et d'hébergement qui auraient été occasionnés par une solution développée sur-mesure.

Ainsi, s'il est judicieux de se tourner vers les **microservices pour des besoins identifiés et existants sur le marché**, une solution centralisée sur-mesure reste préférable au développement d'une multitude de briques pour répondre à des spécifications et besoins ultra-personnalisés qui n'existent nulle part ailleurs.

Beaucoup d'entreprises envisagent de migrer vers AWS pour bénéficier des nombreux avantages offerts par cette solution. Cependant, les frameworks & CMS PHP populaires, comme Drupal, eZ ou Magento, n'ont pas été conçus à l'époque du Cloud. Leur migration vers AWS nécessite souvent des adaptations et de nouveaux apprentissages

Découvrez dans un webinar notre retour d'expérience sur **les migrations des CMS classiques vers AWS**.

Visionner le webinar

VI - Lexique

API
Application Programming Interface ou interface de programmation d'application. Il s'agit d'une interface regroupant un certain nombre de fonctionnalités (langages de programmation, bibliothèques, etc.), permettant à des applications de communiquer entre elles, par exemple pour des échanges de données.

« As a Service »
Cette expression signifiant littéralement « en tant que service ». Ici, les CMS, le contenu (Content as a Service ou CaaS) ou encore les logiciels (Software as a Service ou SaaS) sont hébergés sur des serveurs distants et non plus sur ceux de l'utilisateur. Gratuits ou payants, ils nécessitent généralement une inscription et/ou un abonnement.

CDN
Content Delivery Network ou réseau de diffusion de contenu. Généralement constitué d'un réseau de serveurs, un CDN permet la diffusion rapide et optimisée de contenus (médias, streaming...) au plus près de l'internaute.

CMS
Content Management System ou système de gestion de contenu. Les CMS sont des solutions permettant de concevoir et gérer des contenus par exemple pour des sites Internet et applications mobiles.

Framework
Aussi appelé infrastructure logicielle ou cadre d'applications, le framework propose l'architecture et les fonctionnalités de base nécessaires à la construction d'un logiciel.

Middleware
Il s'agit d'un logiciel qui assure la liaison entre plusieurs autres applications. Il est souvent utilisé pour centraliser les informations et fonctionnalités de différentes applications, ou comme brique de transmission entre le noyau d'une infrastructure et les applications utilisateurs.

Open Source
Les logiciels possédant un « code source ouvert » peuvent être librement distribués et faire l'objet de travaux dérivés suivant leur licence.

React Native
C'est un framework permettant de développer des applications mobiles, initialement développé par Facebook.

Release
Sortie de la nouvelle version d'un logiciel ou d'une application.

SDK
Software Development Kit ou devkit. Destiné aux développeurs, le SDK réunit plusieurs outils (bibliothèques, langages de programmation, etc.) qui permettent de développer rapidement une application.

Template
Modèle permettant la création d'une partie d'une application (comme un site Internet), le template offre un cadre cohérent pour son développement et sa mise en forme (on parle alors de thème).

Webhook
Il s'agit d'une fonction automatique, permettant de récupérer et stocker des événements, généralement externes à une application. Un webhook permet par exemple de recevoir des notifications ou d'exécuter une action après la publication d'un contenu.

À PROPOS DE KALIOP

Le groupe Kaliop est un spécialiste de la transformation numérique et de l'innovation. L'excellence technique est à la source de notre ADN, l'agilité est au cœur de notre méthodologie. L'open-source est une valeur centrale de nos choix technologiques. Grâce à ces engagements, Kaliop a mené à bien depuis plus de 15 ans des centaines de projets digitaux stratégiques, en associant l'expertise technique à une vision fonctionnelle centrée sur la création de valeur pour l'utilisateur final.

tech.kaliop.com - 01 80 49 30 00

KALIOP
INTERACTIVE MEDIA