Acquia
EXPERIENCE DIGITAL FREEDOM

DRUPAL HEADLESS ET HYBRIDE : LES BASES

Comment déterminer la meilleure approche CMS pour votre entreprise ?

SOMMAIRE

03 INTRODUCTION

07 COMMENT FONCTIONNE UNE ARCHITECTURE UNIFIÉE ?

09 COMMENT FONCTIONNE UNE ARCHITECTURE DÉCOUPLÉE ?

11 DIFFÉRENCE ENTRE HEADLESS ET DÉCOUPLÉ

13 OPTIONS POUR LES SERVICES WEB EN BACK-END

16 OPTIONS POUR LES SDK EN FRONT-END

19 ALLER AU-DELÀ DE « L'API-ONLY »

22 LA QUESTION QUI RÉVÈLE LA BONNE APPROCHE

27 FLEXIBILITÉ D'UNE ARCHITECTURE HYBRIDE

30 AVANTAGES DU DÉCOUPLAGE PROGRESSIF

33 DRUPAL VOUS PERMET DE CHOISIR LA BONNE APPROCHE POUR CHAQUE PROJET

INTRODUCTION

Dans le monde du développement web, peu de tendances se sont répandues aussi rapidement que les systèmes de gestion de contenu (CMS) headless et les applications découplées.

Il est clair que cette évolution va bien au-delà de l'approche traditionnelle dans laquelle la couche de présentation est étroitement liée au CMS en back-end. Disposer d'un plus grand nombre d'options implique toutefois de mieux comprendre le fonctionnement de chacune de ces options afin de choisir l'approche la mieux adaptée à vos objectifs.

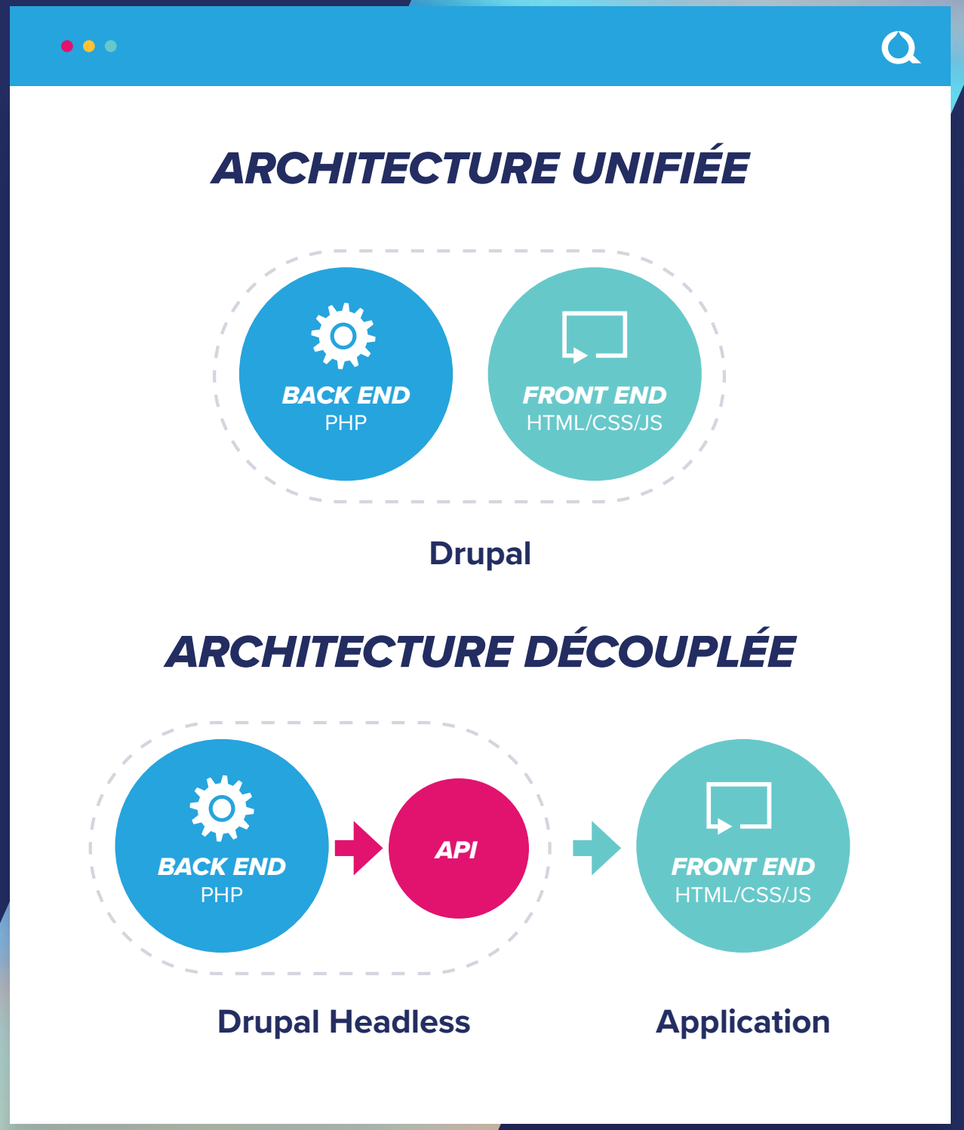
Dans cet e-book, nous expliquons comment fonctionne une architecture découplée, quand envisager le découplage et comment marketeurs et développeurs peuvent exploiter Drupal headless pour offrir de superbes expériences digitales. Nous verrons également comment Drupal peut offrir des fonctionnalités « headless hybrides » avancées, difficiles à trouver sur d'autres systèmes.

Commençons par différencier les architectures « unifiées » et « découplées ».

Dans une **architecture** traditionnelle ou **unifiée**, les responsabilités du front-end et du back-end sont contenues dans un seul système.

Le CMS peut gérer le contenu, mais aussi le balisage de l'expérience front-end (HTML) à l'aide de modèles ou d'outils low-code. Cet ensemble d'outils constitue un système unique et **unifié** qui convient à une grande variété de sites web et d'applications sans nécessiter de complexités supplémentaires.

Avec une **architecture découplée**, le back-end de Drupal fournit une couche de service d'API pour la présentation d'un contenu structuré. Ici, le développeur n'utilise pas le moteur de modélisation Twig fourni en standard, mais s'appuie sur une autre technologie pour le rendu de l'expérience front-end. Étant donné le nombre croissant d'appareils permettant de collecter des données et d'interagir avec le contenu, cette approche paraît vraiment logique.



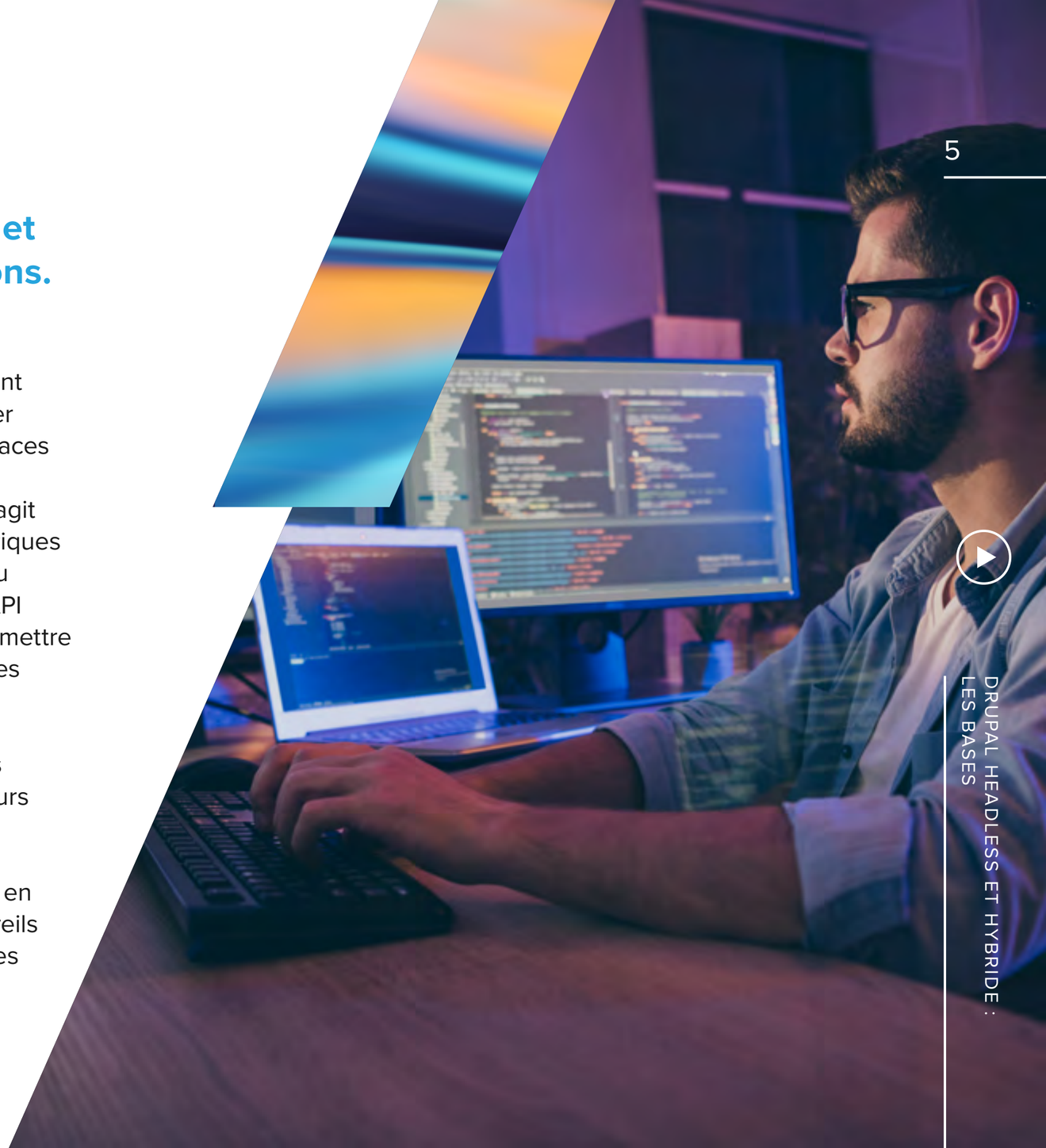
Dans ce modèle, Drupal est considéré comme un référentiel CMS headless, qui présente le contenu et les données pour utilisation par d'autres applications. Ces applications pourraient inclure :

- Les **applications natives**, développées pour un appareil ou une plateforme spécifique, par exemple un poste de travail ou un appareil mobile. Il s'agit par exemple des applications mobiles souvent développées pour des smartphones ou des montres connectées spécifiques.

- Les **applications JavaScript** qui actualisent les pages de manière dynamique via des requêtes asynchrones vers le serveur, ce qui évite d'actualiser toute une page. Les applications web peuvent donc faire appel au serveur sans que le navigateur recharge la page.

- Les **panneaux numériques** sont des moyens courants d'afficher des informations dans les espaces publics, les restaurants, les bureaux ou d'autres lieux. Il s'agit généralement d'écrans numériques capables d'afficher du contenu et des images à partir d'une API externe. Cette API permet de mettre à jour et de gérer facilement les panneaux à distance.

- Les **applications IoT**, avec des appareils tels que les téléviseurs intelligents, l'Echo d'Amazon, l'Apple Watch ou les capteurs d'activité. C'est un espace est en pleine expansion et ces appareils dépendent souvent de services externes pour le contenu et les données.



POURQUOI OPTER POUR UNE APPROCHE « HEADLESS » ?

Plusieurs raisons justifient l'adoption d'une approche Drupal headless. Certaines entreprises mettent en œuvre une stratégie « headless » pour exploiter Drupal en tant que référentiel de contenu. Elles peuvent ainsi diffuser du contenu sur tout appareil d'un écosystème digital complexe. D'autres préfèrent le découplage pour permettre aux équipes front-end d'exploiter les frameworks JavaScript (JS) les plus couramment utilisés, tout en conservant les capacités back-end de Drupal. De nombreuses organisations utilisent par exemple Drupal headless en combinaison avec des frameworks JavaScript, tels que React, Svelte, NextJS ou VueJS.

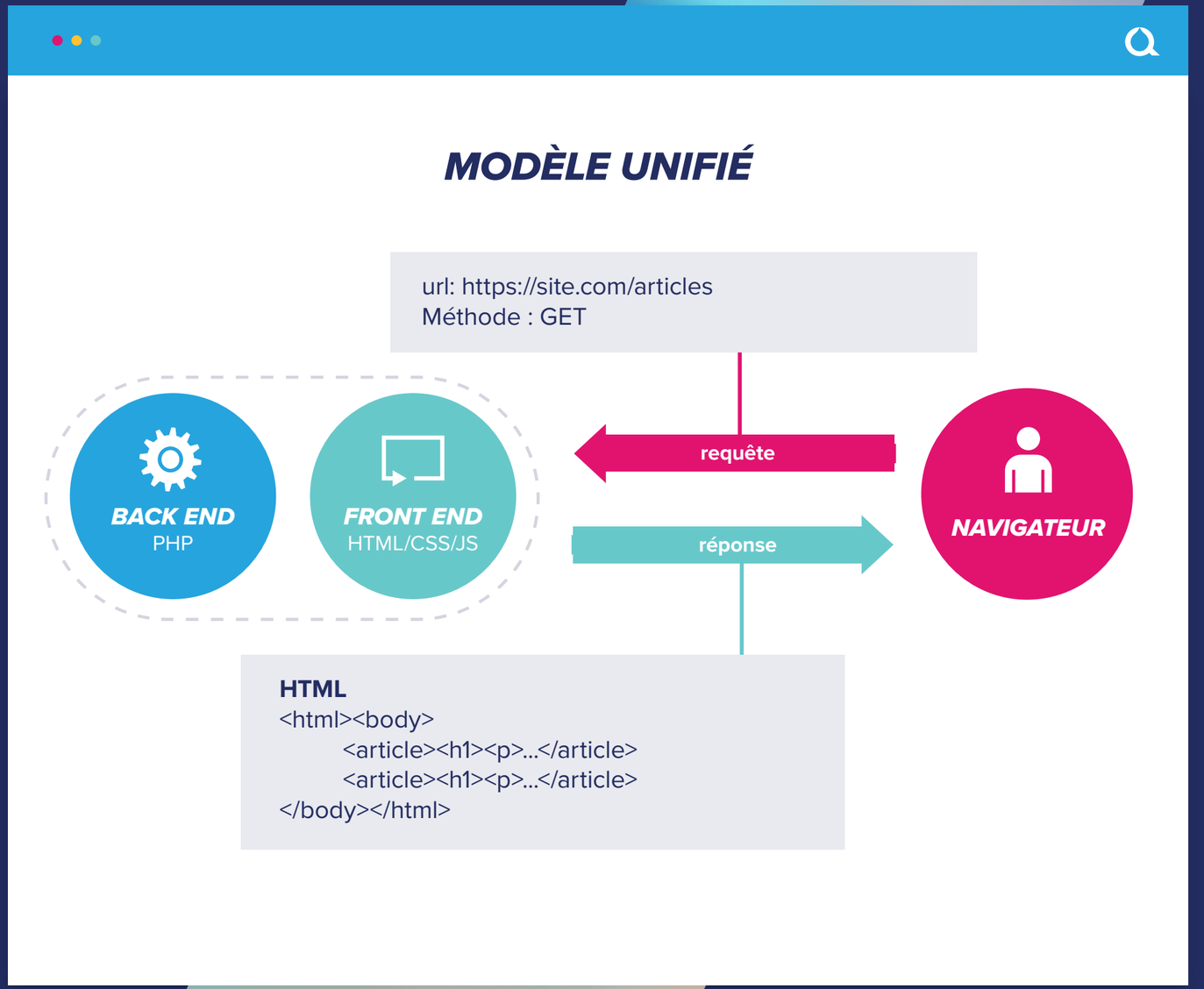
Point important : quel que soit le motif d'adoption d'un CMS headless, il convient de s'assurer que développeurs web et marketeurs obtiennent ce dont ils ont besoin.

COMMENT FONCTIONNE UNE ARCHITECTURE UNIFIÉE ?

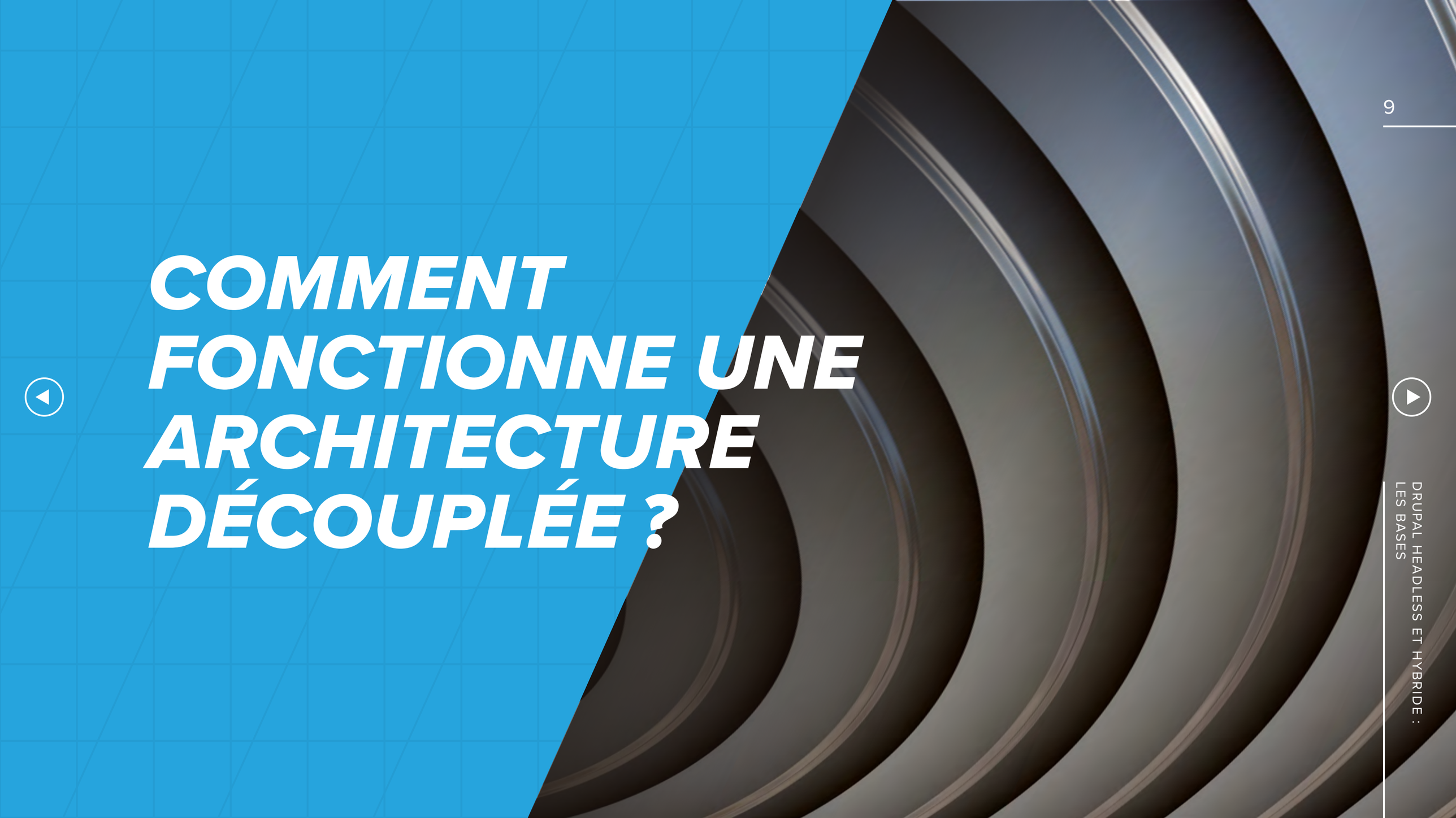
La plupart des interactions qui ont lieu sur le web reposent sur un concept de type **requête/réponse.** Un utilisateur demande des données et un système répond en rassemblant, formatant et présentant le contenu approprié selon un modèle HTML prédéfini.

Il s'agit du mode de fonctionnement traditionnel d'Internet : le navigateur envoie une requête HTML, le serveur retourne les données, puis le navigateur présente la page à l'utilisateur. Très simple, très évolutif, et vraiment fondamental pour le web.

Dans un modèle unifié, Drupal fournit le système de gestion de contenu en back-end et un moteur de rendu HTML (Twig) dans un framework composable. Certains CMS traditionnels fournissent des résultats similaires. Toutefois, Drupal est différent, dans la mesure où il est **API-first** et basé sur des systèmes discrets et composables. Selon les besoins, il peut être utilisé dans une architecture unifiée ou découplée.



COMMENT FONCTIONNE UNE ARCHITECTURE DÉCOUPLÉE ?

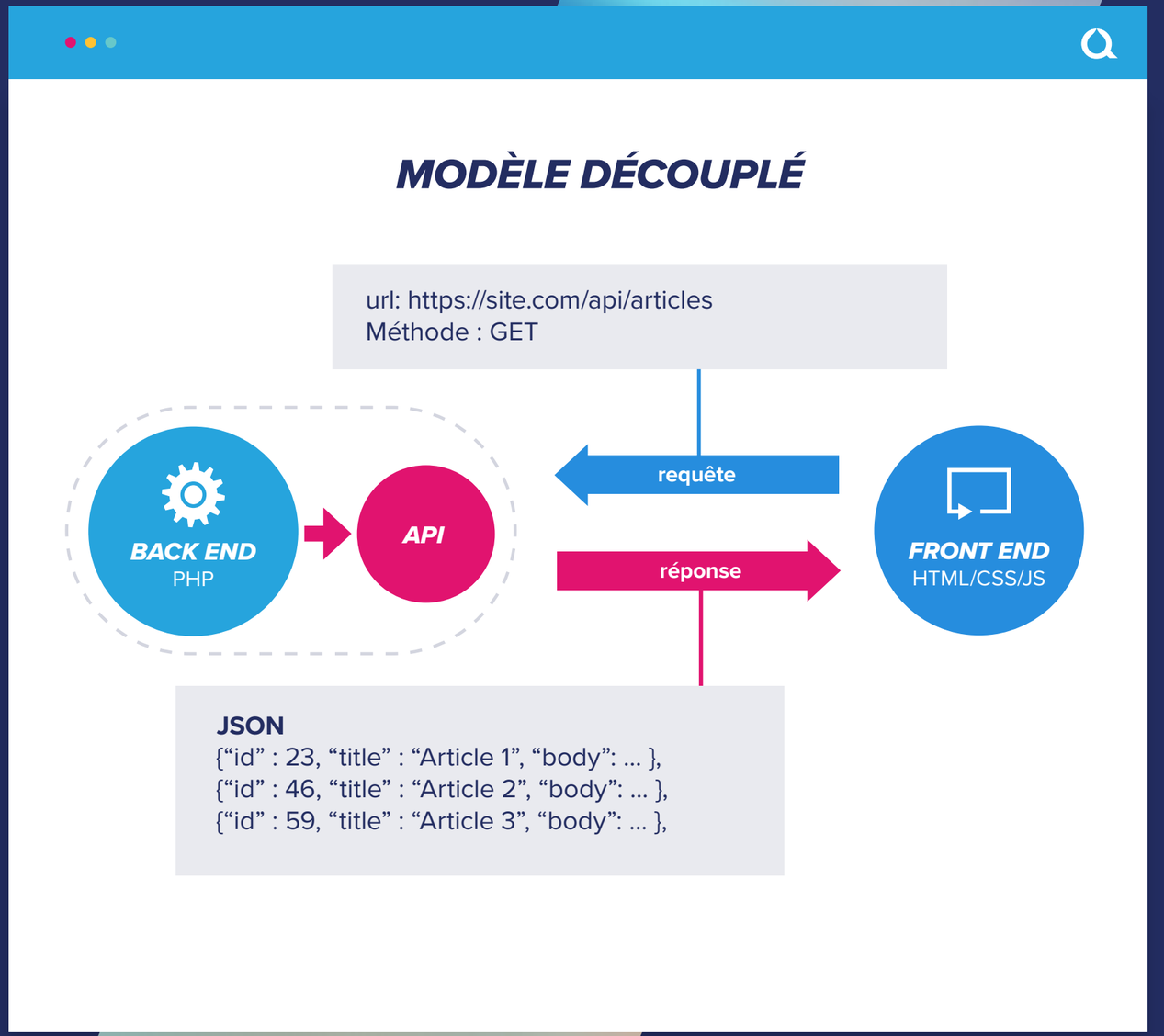


Une architecture découplée adopte la même approche de base, mais avec une légère différence. Au lieu d'un seul système et unifié, elle répartit les différentes responsabilités entre plusieurs systèmes.

Dans une architecture découplée, le back-end de Drupal fait office de référentiel de contenu. Le référentiel Drupal headless et l'application découplée échangent des données par le biais de méthodes HTTP standard. L'application envoie une requête et transmet les paramètres à l'API. Ensuite, le CMS Drupal headless retourne la réponse, qui est généralement au format JSON.

Drupal est API-first et le module RESTful Web Services est inclus dans le noyau de Drupal (le package standard de base qui définit la dernière version de Drupal). Ce module fournit une API de données (API RESTful personnalisable et extensible) gérées par Drupal. La réponse HTTP peut être au format JSON, XML ou utiliser d'autres représentations.

Sur la figure ci-dessus, l'API REST et le client HTTP font office de médiateurs dans l'architecture découplée. Ils permettent aux développeurs back-end et front-end de travailler avec les frameworks de leur choix. Drupal fournit des API REST de base et un format JSON:API entièrement fonctionnel dans le noyau, avec GraphQL et d'autres formats disponibles dans l'écosystème.



DIFFÉRENCE ENTRE HEADLESS ET DÉCOUPLÉ

Dans ce guide, nous avons choisi d'utiliser les termes headless et découplé de manière très spécifique. Les distinctions entre « headless » et « découplé » peuvent être assez nuancées et difficiles à définir.

Voici une bonne règle de base : Le terme « headless » fait référence au CMS ou au service fournissant des données via l'API et le terme « découplé » fait référence à l'architecture ou à l'application en front-end.

En termes simples, un **CMS headless** ne fournit aucun affichage à l'utilisateur final. Il fournit uniquement l'API permettant la mise à disposition du contenu et des données. Grâce à cette approche, toute expérience front-end, quelle qu'elle soit, peut utiliser le contenu. En tant que CMS headless, Drupal est un choix très intéressant et largement utilisé, car il est « API-first » (et non « API-only »). Il a été spécifiquement conçu avec une couche de services d'API résidant dans son noyau.

Une **architecture découplée** fait référence à un type d'application qui fournit l'expérience front-end, avec un ou plusieurs services d'API pour la fourniture du contenu et des données. Nous incluons généralement l'interface utilisateur dans l'architecture découplée, qu'il s'agisse d'une application JS, d'une application mobile, d'un téléviseur intelligent, d'un affichage dynamique ou d'autre chose.

AVEC UNE ARCHITECTURE DRUPAL HEADLESS, LES ÉQUIPES DE DÉVELOPPEMENT WEB PEUVENT :

- **Alimenter en données une multitude d'appareils :** Grâce à ses API et ses services web flexibles, Drupal peut être le cerveau de tous vos systèmes pour diffuser du contenu partout.
- **Exploiter d'autres technologies front-end :** Drupal peut fonctionner comme une couche de services pour permettre au contenu créé dans le CMS Drupal d'être présenté via un framework JavaScript, tel que React, VueJS et Angular.
- **Contrôler tous les aspects des actifs :** Drupal Headless peut servir de référentiel central pour envoyer des vidéos et des données aux nombreux points de diffusion disponibles sur la marketplace actuelle des médias.
- **Intégrer plusieurs systèmes :** Drupal peut être intégré en back-end pour prendre en charge les systèmes techniques existants.

OPTIONS POUR LES SERVICES WEB EN BACK-END

Les différents modules de services web Drupal permettent aujourd'hui d'utiliser facilement des données à partir de Drupal. Ces modules incluent des modules communautaires en complément aux modules déjà disponibles dans le noyau.

- **Services web RESTful de base :** Drupal fournit en standard une API REST. Cette API inclut les opérations create, read, update et delete (CRUD) à appliquer aux entités de contenu. L'API REST permet également la lecture des entités de configuration. Quatre modules REST principaux résident également dans le noyau : **Serialization**, **RESTful Web Services, HAL,** and **Basic Auth.** Ces modules (Core REST) nécessitent peu de configuration tout en offrant de nombreuses fonctionnalités.

- **JSON:API : JSON:API** est de plus en plus utilisée en raison de son adoption par les développeurs comme format commun et du fait de sa capacité à prendre en charge les données complexes. Spécification destinée aux API REST, JSON:API utilise le format JSON et offre des fonctionnalités allant au-delà de la couche des services de base.

- **GraphQL :** Développé à l'origine par Facebook, GraphQL est un langage de requête développé pour des requêtes personnalisées. Il permet aux clients de Drupal headless d'extraire facilement des données du back-end. Ils peuvent récupérer des ensembles de données personnalisés via une seule requête. Drupal prend en charge le module **GraphQL.**

- **Générateur de requêtes low-code :** Drupal fournit un module de base appelé **Views.** Son interface permet de créer des composants qui extraient du contenu de la base de données d'un site web et le présentent à l'utilisateur. Ce générateur de requêtes dynamique peut fournir des points de terminaison REST entièrement gérés dans l'interface utilisateur. Vous pouvez donc créer et gérer des points de terminaison personnalisés sans écrire de code.

- **Code personnalisé :** La couche de services de Drupal est conçue pour être « pluggable » et composable dès le départ. Même si vous avez des intégrations de services très spécifiques, voire propriétaires, pour la fourniture de contenu, vous pouvez construire ce dont vous avez besoin au-dessus de Drupal.

Cette personnalisation sera beaucoup plus rapide, plus facile et plus stable que toute autre approche sur mesure, car il vous suffit d'étendre le CMS pour obtenir ce dont vous avez besoin, au lieu de partir de zéro.

OPTIONS POUR LES SDK FRONT-END

En plus d'une robuste collection de services web, Drupal offre maintenant un écosystème de kits de développement logiciel (SDK) qui accélèrent le développement d'applications avec d'autres technologies. Les SDK peuvent étendre la portée de Drupal au-delà de PHP.

Auparavant, la consommation de contenu Drupal nécessitait une certaine compréhension des détails de mise en œuvre de l'API REST et du développement d'applications personnalisées. Drupal étant un CMS véritablement composable, la communauté Drupal a continué à fournir un support pour les frameworks et les plateformes de développement d'applications les plus utilisés.

Vous pouvez disposer d'une application fonctionnelle dès le premier jour avec une multitude d'options :

- **Drupal State** offre un ensemble d'utilitaires qui permettent aux développeurs JavaScript ayant une connaissance limitée de Drupal ou de la spécification JSON:API de tirer parti des meilleures fonctionnalités de l'API Drupal.

  Étant indépendant des frameworks, Drupal State peut être utilisé avec le framework JavaScript classique ou tout autre framework JavaScript courant. Drupal State se veut modulaire, extensible et modifiable, afin d'aider à la prise en charge de l'écosystème Drupal JavaScript.

- **Next.js pour Drupal** ou Next-Drupal, permet de remplacer votre front-end Drupal par Next.js tout en conservant les principales fonctions d'édition de Drupal, soit le meilleur des deux mondes. Next-Drupal permet de prévisualiser instantanément le site Next.js dans le workflow éditorial de Drupal. Vous pouvez également créer une architecture de contenu personnalisée sans vous embarrasser de modules complexes et de modèles Twig encombrants. Enfin, Next-Drupal prend en charge les vues multiples et multisites.

- **Gatsby,** générateur de sites statiques, conçu avec React et GraphQL, atténue les problèmes d'évolutivité et de performances liés à la création d'applications React. En back-end, vous pouvez exploiter les outils de modélisation, de création et d'édition de contenu de Drupal avec le module JSON:API pour fournir du contenu au front-end Gatsby. Vous obtenez une alternative puissante, complète, open source et gratuite aux coûteux CMS d'entreprise.

- **Druxt.js** est une passerelle open source entre deux frameworks, NuxtJS et Drupal, avec possibilité de tirer parti du système d'affichage entités/champs, des régions Block, des vues, etc. de Drupal. DruxtJS prend en charge le client JSON:API de Drupal avec la mise en cache Vuex et agit comme une couche thématique Vue.js pour Drupal. Les composants Druxt peuvent être thématisés à l'aide de composants Wrapper aux côtés de slots Vue.js, de $attrs et de props.

ALLER AU-DELÀ DE « L'API-ONLY »

Aujourd'hui, les plateformes CMS headless courantes adoptent une approche **API-only.** En utilisant l'architecture découplée, les organisations disposent simplement d'une API et d'un back-end pour entrer les données dans le CMS. L'expérience front-end requise fait défaut.

Le code nécessaire pour créer une expérience complète, qu'il s'agisse d'un site web, d'une application ou d'autres expériences omnicanales, doit être créé. C'est pourquoi l'architecture découplée est souvent qualifiée de « high code ». Elle s'appuie sur les développeurs pour écrire et maintenir le code de l'expérience front-end. Une architecture unifiée est souvent qualifiée de « low code ». Elle fournit des outils d'interface permettant aux non-développeurs d'assembler l'expérience.

Pour certaines applications, l'approche « high-code » est acceptable et même souhaitée. Dans certains cas, l'expérience et le code doivent être liés, par exemple lorsque vous créez une application mobile, web ou IoT. Si le développeur est responsable de l'expérience, c'est souvent pour lui le moyen le plus efficace de créer et de maintenir le front-end.

Il s'agit toutefois d'une limitation importante si vous avez des besoins différents, par exemple des outils low-code pour autonomiser les utilisateurs métier. Dans ce modèle, la mise en œuvre de tout type de présentation personnalisée nécessite l'intervention d'un développeur, ce qui ralentit généralement la mise sur le marché. Pour aggraver les choses, les plateformes CMS headless API-only ont tendance à pousser les organisations vers des niveaux plus coûteux en imposant des limites à l'architecture et à la gouvernance du site.

Vous pouvez par exemple être limité quant au nombre de types de contenu, de champs ou d'utilisateurs. Vous pourriez également disposer de possibilités limitées en termes de rôles et d'autorisations personnalisés. Au fur et à mesure de votre utilisation de ces plateformes, leur coût peut augmenter.

Drupal au contraire utilise une approche **API-first.** Une API est toujours disponible, mais n'est pas *obligatoire*.

Il n’y a aucune limite quant au nombre de types de contenu, à la complexité des données, au nombre d’intégrations, au nombre d’utilisateurs, ni même aux options avancées pour les rôles et les utilisateurs.

Drupal offre toutes les fonctionnalités d’une plateforme CMS headless, sans les limitations. Les organisations ont en outre la possibilité de choisir l’architecture qui convient à chaque projet.

Lorsque nécessaire, l’architecture unifiée va plus loin en apportant aux organisations les outils, les modèles et les options de rendu nécessaires pour construire un site, sans aucune limitation. C’est un avantage considérable. **En optant pour Drupal, vous disposez du meilleur de tous les mondes.** Vous avez la possibilité d’utiliser l’architecture unifiée ou découplée sur la même plateforme selon vos besoins, et parfois en même temps. C’est l’approche CMS hybride, qui offre le plus grand degré de liberté et de flexibilité.

LA QUESTION QUI RÉVÈLE LA BONNE APPROCHE

Les architectures découplées font l'objet d'un grand battage médiatique. Avant de se lancer dans un projet, il est donc important de procéder à une analyse impartiale. Si vous choisissez d'utiliser Drupal comme référentiel de contenu pour servir plusieurs applications, une architecture découplée pourrait vous convenir. Vous pouvez le déterminer en posant une seule question simple : ***Qui a la responsabilité de l'assemblage de l'expérience ?***

RESPONSABILITÉ DÉTENUE PAR LES DÉVELOPPEURS = ASSEMBLAGE HIGH-CODE

Si l'expérience est axée sur le code et détenue par l'équipe informatique (les développeurs), une approche high-code utilisant l'architecture découplée pourrait être la plus appropriée. L'utilisateur métier dispose ici d'un ensemble plus limité d'outils pour optimiser la création de contenu, tandis que le développeur a toute latitude pour créer l'expérience, les modèles et le code permettant d'afficher le contenu. L'expérience et le code sont gérés et déployés ensemble.

UNE ARCHITECTURE ENTIÈREMENT DÉCOUPLÉE OFFRE PLUSIEURS AVANTAGES :

- **Séparation des tâches**
  L'utilisation de Drupal uniquement en tant que référentiel de contenu peut renforcer la séparation des tâches. Avec une architecture entièrement découplée, la manipulation du contenu est réservée au back-end. Cette tâche est séparée du front-end, lequel ne concerne que la présentation et la fourniture du contenu à l'utilisateur.

- **Développement en pipeline**
  Cette séparation s'étend également aux équipes back-end et front-end et permet aux développeurs de travailler à leur propre rythme. Les développeurs front-end sont libres de contrôler le balisage et le rendu, tandis que les développeurs back-end peuvent concentrer leurs efforts sur le développement d'une API robuste.

- **Gestion de contenu rationalisée**
  Transférer la responsabilité de la gestion de l'expérience à l'équipe de développement, libère les utilisateurs métier qui peuvent se concentrer sur le seul contenu structuré. Les responsabilités liées à la création et à la gestion du contenu étant moins nombreuses, le CMS peut être rationalisé pour offrir un workflow plus simple, et parfois plus rapide, pour la création de contenu. Lorsque le contenu est parfaitement standardisé, cette efficacité peut représenter un avantage considérable.

Cependant, lorsque l'ensemble du front-end est contrôlé par une application découplée, les équipes techniques ne peuvent pas tirer parti des capacités de Drupal auxquelles tiennent de nombreux utilisateurs.

Cette stratégie de découplage total annule les capacités de Drupal telles que l'édition « in-place » et la gestion de l'affichage. Elle introduit également des points de défaillance supplémentaires et le risque d'une dette technique accrue.

Elle génère en outre d'autres risques :

- **Aucune protection contre le cross-site scripting, ni d'assainissement des entrées (surtout si vous n'utilisez pas de framework)**
- **Aucune gestion de la présentation et de l'affichage pour les utilisateurs métier**
- **Pas de workflow de contenu prévisualisable**
- **Pas de modules prêts à l'emploi ni d'intégrations pour le front-end**
- **Pas de notification ou de message du système**
- **Pas de chargement progressif BigPipe ou de stratégies de mise en cache avancées**
- **Pas de balisage accessible en standard ni d'avantages pour l'expérience utilisateur**
- **Pas de gestion multilingue en standard ni de possibilité de changement de langue**

RESPONSABILITÉ DÉTENUE PAR LES MARKETEURS = ASSEMBLAGE LOW-CODE

En revanche, si l'expérience est axée sur le contenu et détenue par les marketeurs (non-développeurs), une approche low-code utilisant l'architecture unifiée est plus efficace. Cette approche fournit des outils en libre-service qui permettent à l'utilisateur métier de contrôler la présentation, et à l'équipe des développeurs de construire les fonctionnalités et les composants découplés. Les deux équipes peuvent travailler en parallèle. L'expérience et le code sont gérés séparément.

*Les **avantages** liés à une architecture unifiée avec Drupal sont nombreux :*

- **Outils low-code**
  L'un des avantages les plus importants et les plus perceptibles est la possibilité pour les non-développeurs de contrôler l'assemblage et la création du contenu à l'aide d'outils low-code. Ils peuvent utiliser l'interface par glisser-déposer pour « construire » du contenu et l'adapter à leurs besoins spécifiques. Au lieu de faire appel à un développeur pour créer des modèles, l'utilisateur métier peut créer directement l'expérience dont il a besoin, beaucoup plus vite.

- **Prévisualisation de l'expérience et workflow**
  Avec des outils low-code, le créateur de contenu peut construire visuellement les pages et les expériences dont il a besoin. Ce feedback visuel immédiat augmente la rapidité d'exécution et permet de préparer le travail de publication en quelques heures et non en quelques jours. Avec des outils de workflow, les processus d'approbation de contenu peuvent imposer la gouvernance, mais aussi faire bénéficier les approbateurs des options de visualisation.

- **De superbes expériences digitales**
  Outre l'assemblage de beaux contenus statiques, les outils low-code permettent d'assembler des composants dynamiques avancés. Ces composants, fournis par les développeurs, sont souvent les mêmes que ceux d'une application entièrement découplée. Ils sont décomposés en éléments réutilisables, de taille réduite. Les non-développeurs peuvent créer des expériences innovantes et attrayantes, fonctionnellement similaires à celles d'une application entièrement découplée.

Ces avantages expliquent pourquoi autant d'équipes marketing apprécient Drupal. Toutefois, avec le pouvoir vient la responsabilité. Une approche « low-code » exige de consacrer davantage de réflexion et d'efforts à la gouvernance et à la gestion du processus de création de contenu. La facilité avec laquelle le système peut être augmenté et modifié peut le rendre trop facile à étendre.

Il peut devenir difficile de gérer les options de configuration les plus complexes et le CMS risque de devenir un système tentaculaire et inefficace. Trouver le juste équilibre entre la croissance et une standardisation plus limitative exige souvent de la discipline et un solide encadrement technique.

LES DEUX APPROCHES SONT VALIDES

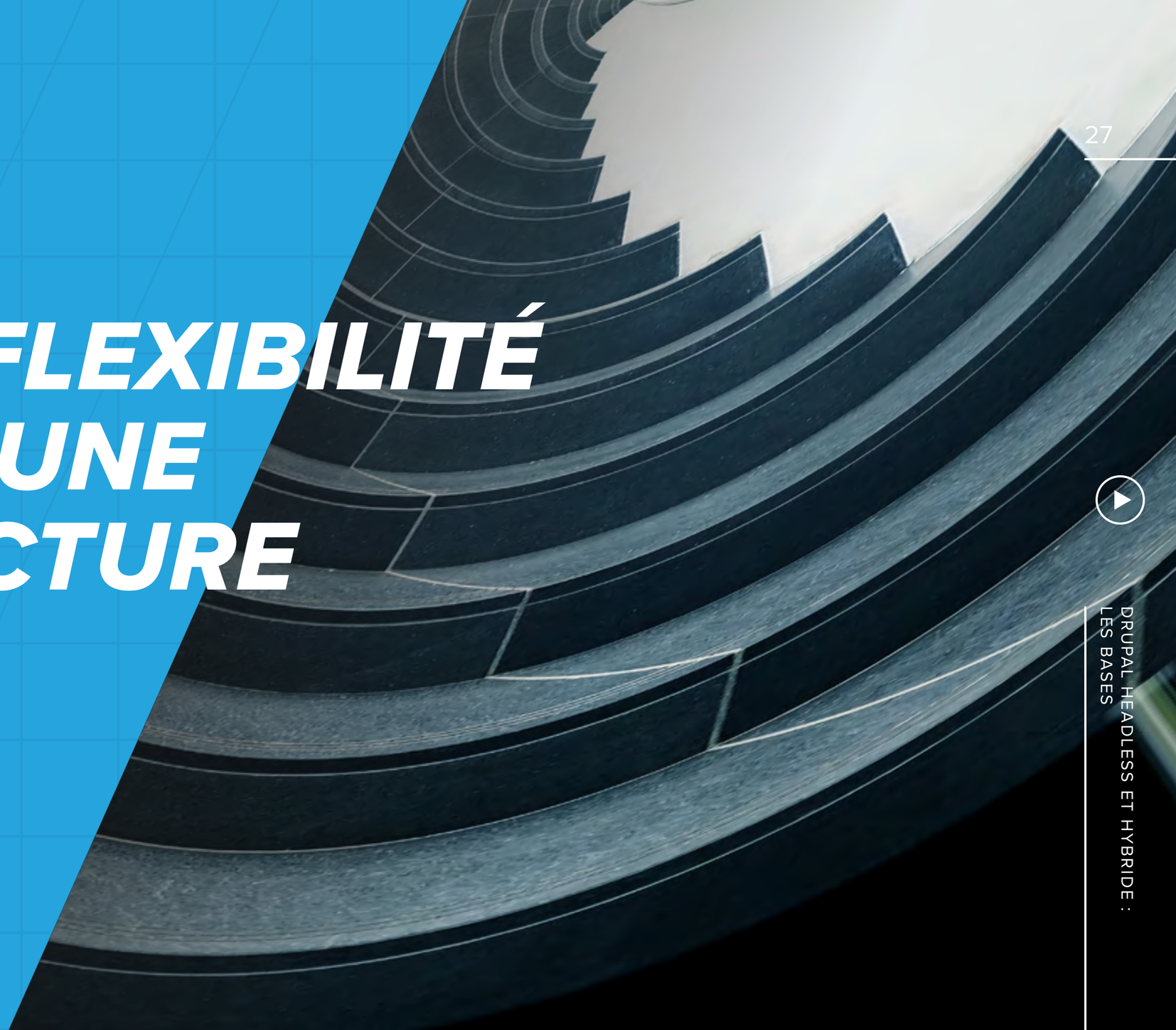
Les deux approches présentent des avantages et des inconvénients. Il n'existe pas une « bonne » façon de procéder. C'est pourquoi il est important de déterminer différents éléments. Qui sera responsable de l'assemblage de l'expérience ? Comment les changements seront-ils mis en œuvre ? L'expérience doit-elle être séparée des déploiements de code ? Quels sont les besoins du projet concerné ?

Quelle que soit l'option adoptée, Drupal constitue le meilleur choix de CMS, car il fonctionne avec toutes les architectures, contrairement aux outils API-only. À partir d'une seule base de code, vous pouvez servir de multiples applications ayant des besoins différents. Drupal est en effet une plateforme composable et prend en charge la gestion multisite de manière native.

À partir de la base de code commune, chaque application peut simplement activer les modules et les fonctionnalités spécifiques dont elle a besoin.

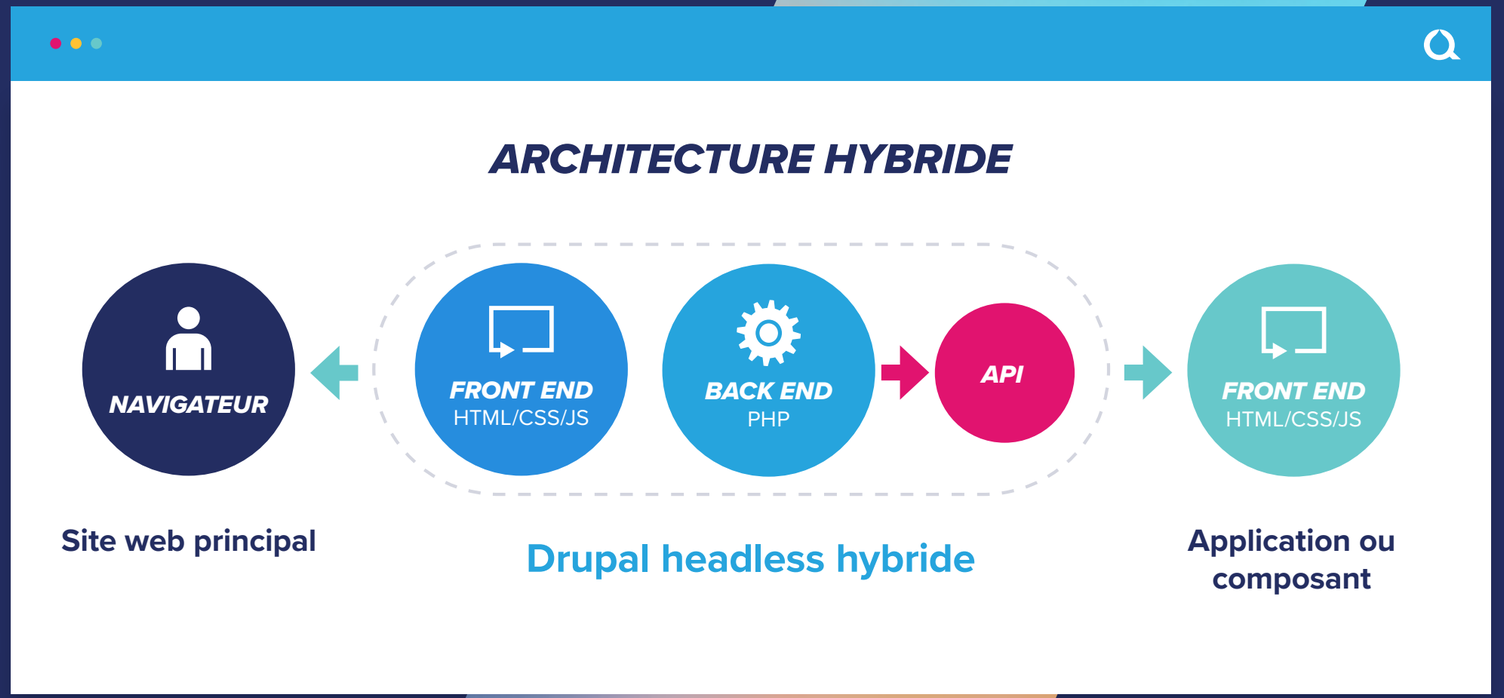
Les organisations disposent d'un outil standard qui peut être reconfiguré de multiples façons. Vous n'êtes jamais « bloqué » par votre architecture. Vous pouvez réutiliser les mêmes outils de différentes manières pour résoudre les problèmes métier qui se présentent. C'est pourquoi Drupal est souvent dit « future-ready ». Il est conçu pour évoluer au gré des changements du marché.

PLUS DE FLEXIBILITÉ GRÂCE À UNE ARCHITECTURE HYBRIDE



Jusqu'à présent, nous nous sommes principalement concentrés sur les architectures unifiées comparées aux architectures découplées. Conserver une architecture CMS unifiée est une option parfaitement valide pour de nombreux sites et applications. Toutefois, si vos besoins vont au-delà des offres typiques de Drupal, vous pouvez opter pour une stratégie **entièrement headless** ou **hybride.** Il ne s'agit pas d'un choix binaire : unifié ou non. Une troisième voie connaît un succès croissant auprès des utilisateurs de Drupal. Il s'agit des architectures hybrides.

Avec une application entièrement découplée, vous risquez d'avoir à recréer et à gérer de nombreuses capacités et fonctionnalités déjà fournies par Drupal dans un modèle unifié. Conséquence : des coûts plus élevés, des cycles de développement plus longs et une assistance réduite pour les utilisateurs non développeurs. C'est pourquoi de nombreuses organisations cherchent aujourd'hui à utiliser Drupal en tant que **CMS hybride**.



En utilisant Drupal dans une architecture hybride, vous pouvez combiner les avantages d'un système unifié et d'un CMS headless. Ceci parce que Drupal est API-first. En d'autres termes, la couche des services API est disponible dans le noyau de Drupal, avec tous les outils de présentation. Cette approche hybride est souvent l'option idéale pour les entreprises qui doivent trouver le juste milieu entre les besoins de l'équipe marketing et ceux des développeurs. Le marketing doit en effet prendre la responsabilité de l'expérience et les développeurs doivent mettre en œuvre des fonctionnalités avancées.

Avec les architectures hybrides, les choses peuvent se compliquer, car il ne s'agit plus d'un choix binaire, mais de toute une variété d'options et d'opportunités. Cette flexibilité est l'un des aspects les plus appréciés par les développeurs qui utilisent Drupal.

IL EXISTE DEUX APPROCHES PRINCIPALES POUR LA MISE EN ŒUVRE D'UNE ARCHITECTURE HYBRIDE :

1. Composants découplés intégrés
En utilisant Drupal comme outil d'assemblage low-code, les non-développeurs peuvent créer du contenu et assembler différents composants pour construire l'expérience. Cette expérience est ensuite présentée telle quelle par le navigateur. Drupal permet également d'ajouter une bibliothèque de composants découplés à assembler avec les autres actifs CMS.

Il peut s'agir de composants web natifs ou de tout autre composant pris en charge par le framework JS. Ces composants peuvent fournir des options d'interaction avancées ou être eux-mêmes des mini-applications interagissant avec tout service web comme une application JS complète. Ces composants peuvent également utiliser la couche d'API de Drupal pour interagir avec le système en tant que CMS headless. Ils peuvent aussi se connecter à des systèmes externes, par exemple un système d'encaissement pour les ventes en ligne.

2. Applications satellites découplées
L'autre approche majeure consiste à utiliser Drupal dans une architecture unifiée pour le site principal, tout en offrant du contenu via une API pour utilisation par des applications « satellites ». Il peut s'agir de n'importe laquelle des applications découplées dont nous avons parlé plus haut : Frameworks JS, applications mobiles, téléviseurs intelligents et autres applications IoT.

Cette flexibilité explique pourquoi les développeurs, mais aussi les **analystes,** vantent les mérites de l'architecture hybride. Cette architecture offre en effet les capacités d'adaptation promises par les services headless, avec les outils et le support d'une application unifiée.

LES AVANTAGES DU DÉCOUPLAGE PROGRESSIF

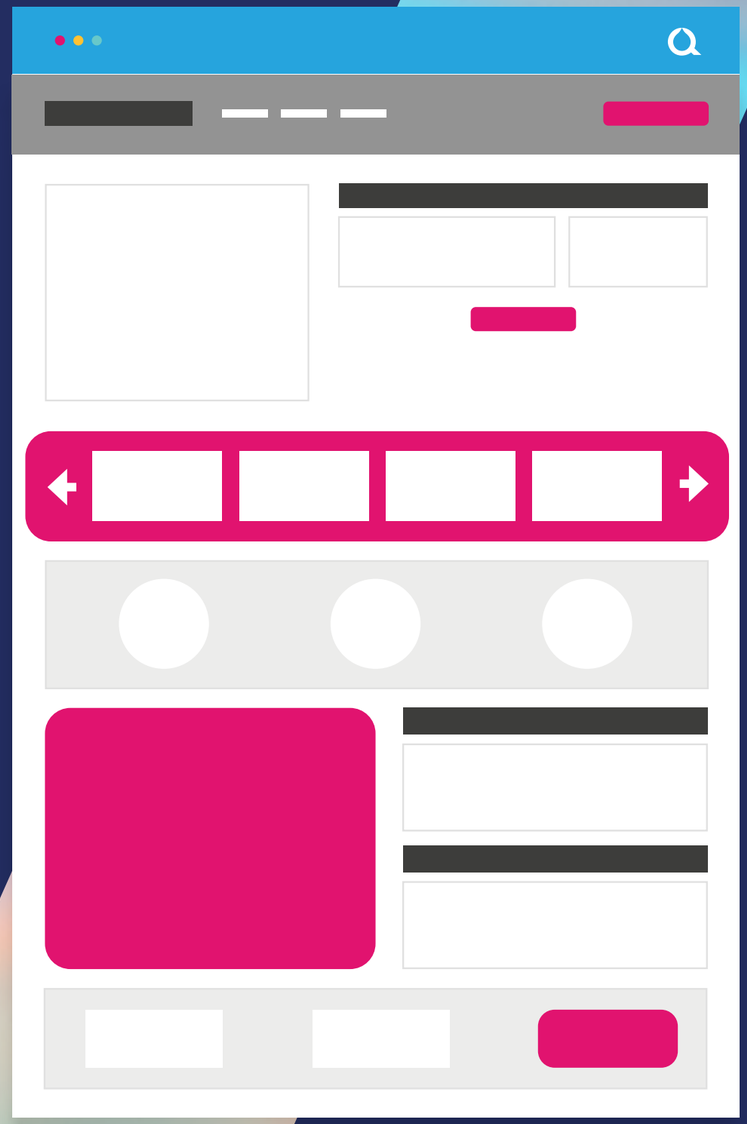
La **stratégie de découplage progressif** est une solution de plus en plus répandue pour atténuer les risques d'un découplage total de Drupal. Contrairement à une architecture entièrement découplée, les implémentations hybrides insèrent un framework JavaScript dans le front-end d'un site Drupal sous forme de composants découplés. Les frameworks JavaScript continuent à utiliser l'API REST. Toutefois, certaines zones de contenu peuvent être contrôlées et rendues par Drupal, tandis que d'autres continuent à exploiter les fonctions de rendu côté client.

Cette approche hybride permet de bénéficier de l'expérience front-end de JavaScript, tandis que les équipes éditoriales et techniques continuent à exploiter les précieuses fonctionnalités de Drupal. Ces composants découplés peuvent être créés et mis en œuvre selon les besoins, de manière progressive. Certaines parties du site peuvent être découplées sans qu'il soit nécessaire de s'engager dans une architecture unifiée ou totalement découplée.

Les utilisateurs métier peuvent utiliser des outils low-code pour assembler et réorganiser les composants, sans faire appel aux développeurs.

UNE STRATÉGIE DE DÉCOUPLAGE PROGRESSIF PERMET DE GÉRER DES ÉLÉMENTS DE PAGE SPÉCIFIQUES DE DIFFÉRENTES MANIÈRES.

Ici, les composants découplés en rose peuvent être gérés par page selon les besoins. En fonction des besoins de l'expérience client, l'application peut choisir de manière sélective quelle quantité d'une page particulière découpler.



ÉTUDE DE CAS

GEORGIA TECHNOLOGY AUTHORITY

La Georgia Technology Authority est l'agence de services numériques qui dessert l'État de Géorgie, aux États-Unis. Avec l'aide d'Acquia, l'équipe de GTA a conçu une stratégie pour migrer, reconstruire et remodeler le CMS de l'État de Géorgie et le transformer en un seul système cohérent.

Sur Acquia Cloud Platform, ils ont construit une **architecture multisite** qui favorise la cohérence et la flexibilité pour tous les sites de l'agence. Grâce à cette stratégie, ils ont pu migrer le contenu de manière efficace, tout en créant une application de recherche personnalisée, réactive et « mobile-first ».

En l'espace de 12 mois, Georgia.gov a lancé 55 sites sur Acquia Platform. Les responsables estiment que le passage à Drupal et à Acquia Platform va leur permettre de réaliser des économies à hauteur de 4,7 millions de dollars sur cinq ans. Ce changement a permis à Georgia.gov de se libérer de la gestion d'au moins 20 serveurs et a fourni une approche standardisée pour gérer plus efficacement toutes ses propriétés web.

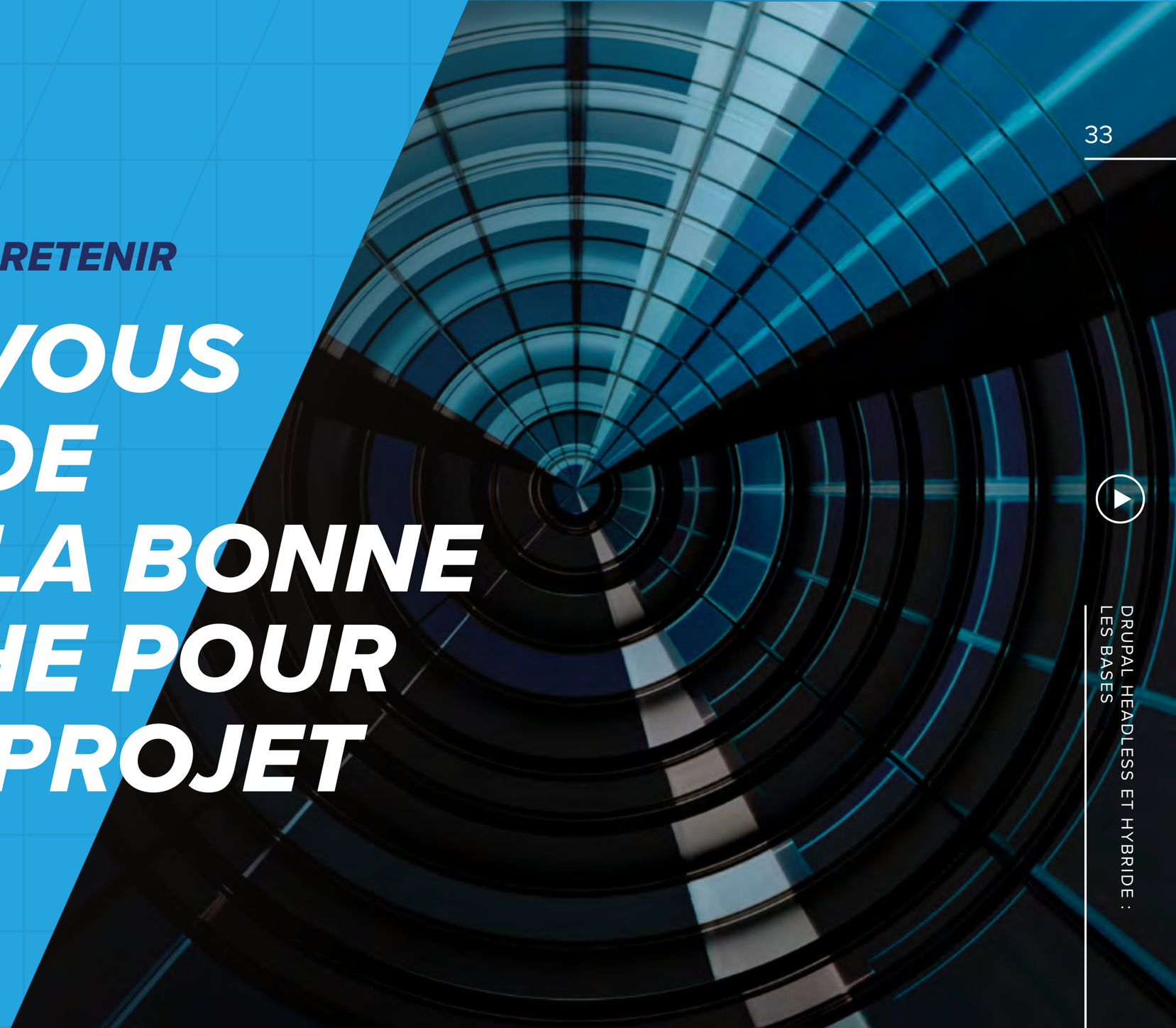
Depuis le lancement du site, Georgia.gov a poursuivi son partenariat avec Acquia afin de faire évoluer les agences « au-delà du navigateur » en tirant parti de l'approche CMS hybride.

Au cours d'un projet de trois mois, **Acquia a construit une compétence Alexa** que toute personne possédant un appareil Amazon Echo peut utiliser. Il est par exemple possible de demander des informations sur les bons alimentaires de Géorgie. Les visiteurs du site peuvent effectuer des demandes simples comme la validation d'un permis de conduire obtenu hors de l'État. Ils peuvent demander un bulletin pour un vote anticipé ou l'enregistrement d'un permis de pêche.

Le contenu, pour répondre à toutes ces demandes, est fourni par les sites web de Géorgie via l'API. La mise à jour de ces informations se fait en un seul endroit et toutes les instances sont alors mises à jour instantanément.

RÉSUMÉ ET POINTS À RETENIR

DRUPAL VOUS PERMET DE CHOISIR LA BONNE APPROCHE POUR CHAQUE PROJET



Les marques doivent offrir à leurs publics une expérience digitale rationalisée, optimisée et valorisante, quel que soit l'appareil utilisé pour interagir avec le contenu. Dans ce contexte, il est important de tirer parti d'une approche CMS optimale, tant pour les marketeurs que pour les développeurs web.

Avec une architecture Drupal headless, le back-end est découplé de la couche de présentation. Les équipes de développement web disposent du contrôle et la flexibilité nécessaires pour fournir des solutions créatives sur de nombreux canaux et appareils.

Toutefois, si cette approche augmente le nombre de problèmes techniques ou empêche les marketeurs de mettre en œuvre le contenu et les campagnes comme ils le souhaitent, un CMS headless n'est peut-être pas la solution idéale.

Il existe une autre approche, de plus en plus appréciée par les marketeurs et les développeurs.

Drupal headless hybride combine les avantages d'un système unifié et d'un CMS headless. En équilibrant ces deux approches, il est plus facile de créer, gérer et optimiser le contenu et les données pour offrir des expériences digitales exceptionnelles sur toute une variété d'appareils. Même si vous ne parvenez pas à trouver l'équilibre adéquat au début, un CMS hybride vous permet d'ajuster progressivement votre approche jusqu'à un état optimal.

Avec un système Drupal hybride headless, les équipes de développement web peuvent :

- **Travailler de manière indépendante, mais collaborative**
- **Rationaliser la gestion de contenu**
- **Accélérer les délais de production**
- **Fournir des informations essentielles sur un nombre croissant d'interfaces**
- **S'adapter rapidement aux technologies futures**
- **Optimiser les expériences digitales**

Pour les équipes de développement qui cherchent à déployer Drupal en tant que CMS headless, Acquia fournit une plateforme complète d'outils et de capacités pour la prise en charge de chaque étape du workflow entre développeurs et marketeurs. Créée par le fondateur de Drupal, pionnier de l'open source, Acquia est une entreprise spécialisée dans les solutions d'expérience digitale ouvertes. Elle permet aux marques d'adopter l'innovation et de créer des expériences client qui comptent.

VOUS SOUHAITEZ EN SAVOIR PLUS SUR LA FAÇON DONT UN CMS DRUPAL HEADLESS OU HYBRIDE PEUT APPORTER DAVANTAGE DE FLEXIBILITÉ À VOS ÉQUIPES ?

DEMANDEZ UNE DÉMONSTRATION

Acquia

ACQUIA.COM

À PROPOS D'ACQUIA

Acquia permet aux marques parmi les plus ambitieuses au monde de créer des expériences client digitales qui comptent. La Digital Experience Platform (DXP) d'Acquia est bâtie sur le CMS open source de Drupal. Elle permet aux marketeurs, aux développeurs et aux équipes IT, dans des milliers d'entreprises mondiales, de créer et de déployer rapidement des produits et des services digitaux attrayants. Ces services améliorent les conversions et favorisent la différenciation.