Acquia
EXPERIENCE DIGITAL FREEDOM

ハイブリッドヘッドレスCMS とデカップルアプリ利用ガイド

Webベースアプリケーションをサポートする 適切なアーキテクチャを選ぶ

目次

第1章

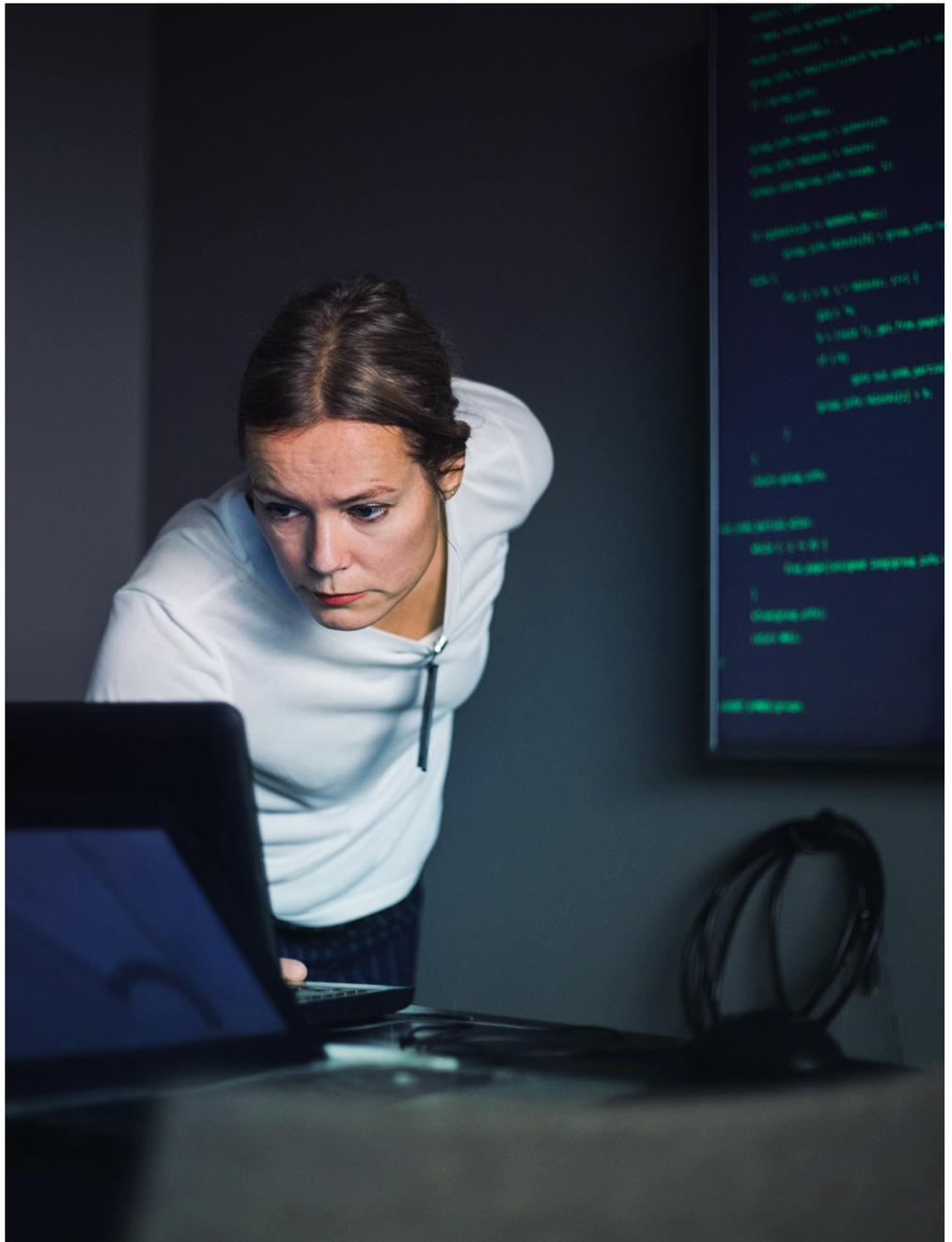
イントロ

21世紀に入ってからは、ほとんどのWebサイトが統一されたアーキテクチャで構築されています。このモデルでは、コンテンツ管理システム（CMS）がWebアプリケーションのフロントエンドとバックエンドの両方を管理します。これは、キャッシュ、セキュリティ、Webアプリケーションの進化など、多くの理由からレジリエンスが高いことが証明されています。

しかしその一番の理由は、開発者ではない人たちにローコードツールやWebベースのユーザーインターフェース（UI）を提供する必要があるからです。これらのツールによって、彼らが担当するエクスペリエンスを、彼ら自身がコントロールすることが可能になります。実際、Webエクスペリエンスの管理能力を必要とする人が増えているため、ローコードツールの台頭は続いています。同時に、新しいインターフェースやアプリケーションの増加により、サポートすべきチャネルが目まぐるしく変化。これらのデジタルチャネルはハイコードソリューションが必要なため、新しい課題となってしまっています。

ほとんどの場合、理想的なのは統一されたアーキテクチャを1つまたは複数の「デカップル」ピースに分割するソリューションです。これは通常、バックエンドのデータ管理をCMSに依存し、フロントエンドのエクスペリエンスを別のアプリケーションで管理できるようにするものです。言い換えれば、これはヘッドレスCMSのバックエンドとデカップリングされたフロントエンドの組み合わせです。

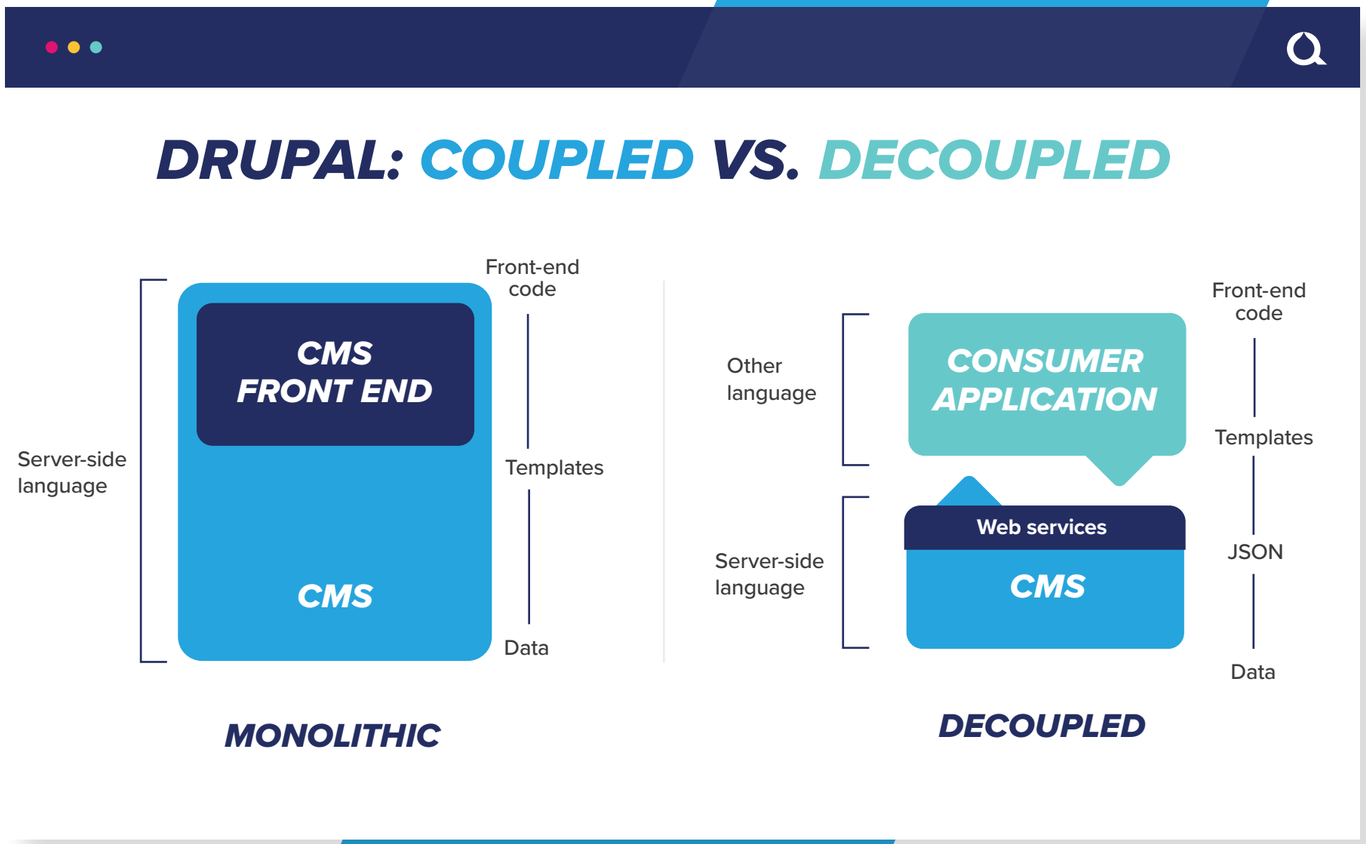
フォレスター社と**ガートナー社**は、2016年初頭にヘッドレスCMSとデカップル型アーキテクチャを重要な能力として取り上げました。最近、**ガートナー社**はアドバイスを修正し、APIオンリーではなく、APIファーストのハイブリッドCMSに焦点を変更しました。これは、21世紀へのさらなる移行に伴い、同じプラットフォームを共有するデジタルアプリケーション開発の重要性が増していることを示しています。



デカップルされた進化

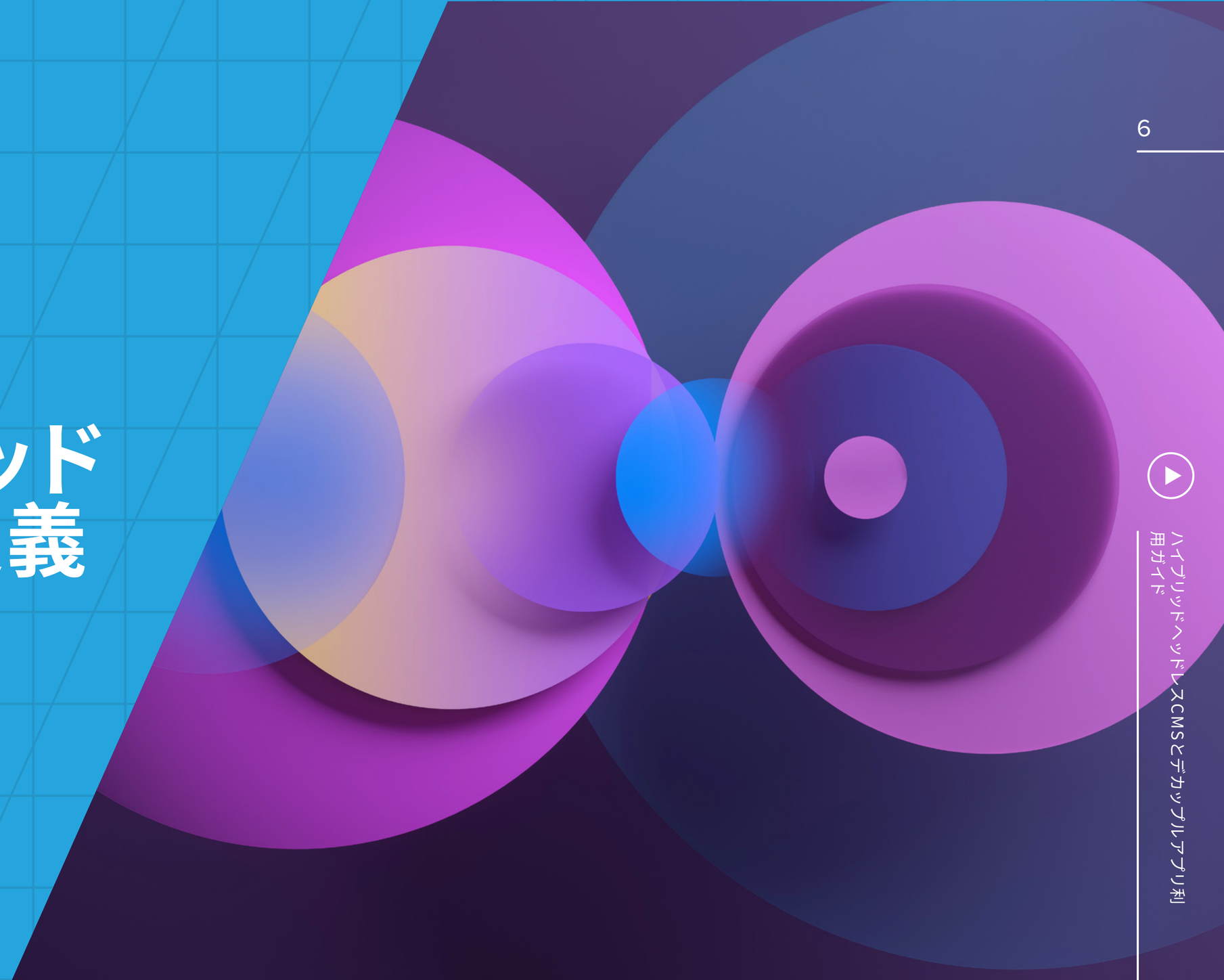
これまでのWebサイトは、バックエンドのCMSと密接したテンプレート型のソリューションでコンテンツを提供するような、単一構造のアーキテクチャで構築されていました。俊敏（アジャイル）な組織は柔軟性を求め、さまざまなプレゼンテーション層にわたって構造化されたコンテンツを管理することに努めています。これを実現するには、現代のデジタル環境を支えるフロントエンドフレームワークに柔軟性を持たせることが必要なのです。

だからこそ、デカップル型CMSやヘッドレスCMSが今注目を浴びており、あなたがこの資料を手にしているのです。しかしながら、あなたに必要なのはWebの次の段階、そしてその先をサポートするためのテクノロジーです。



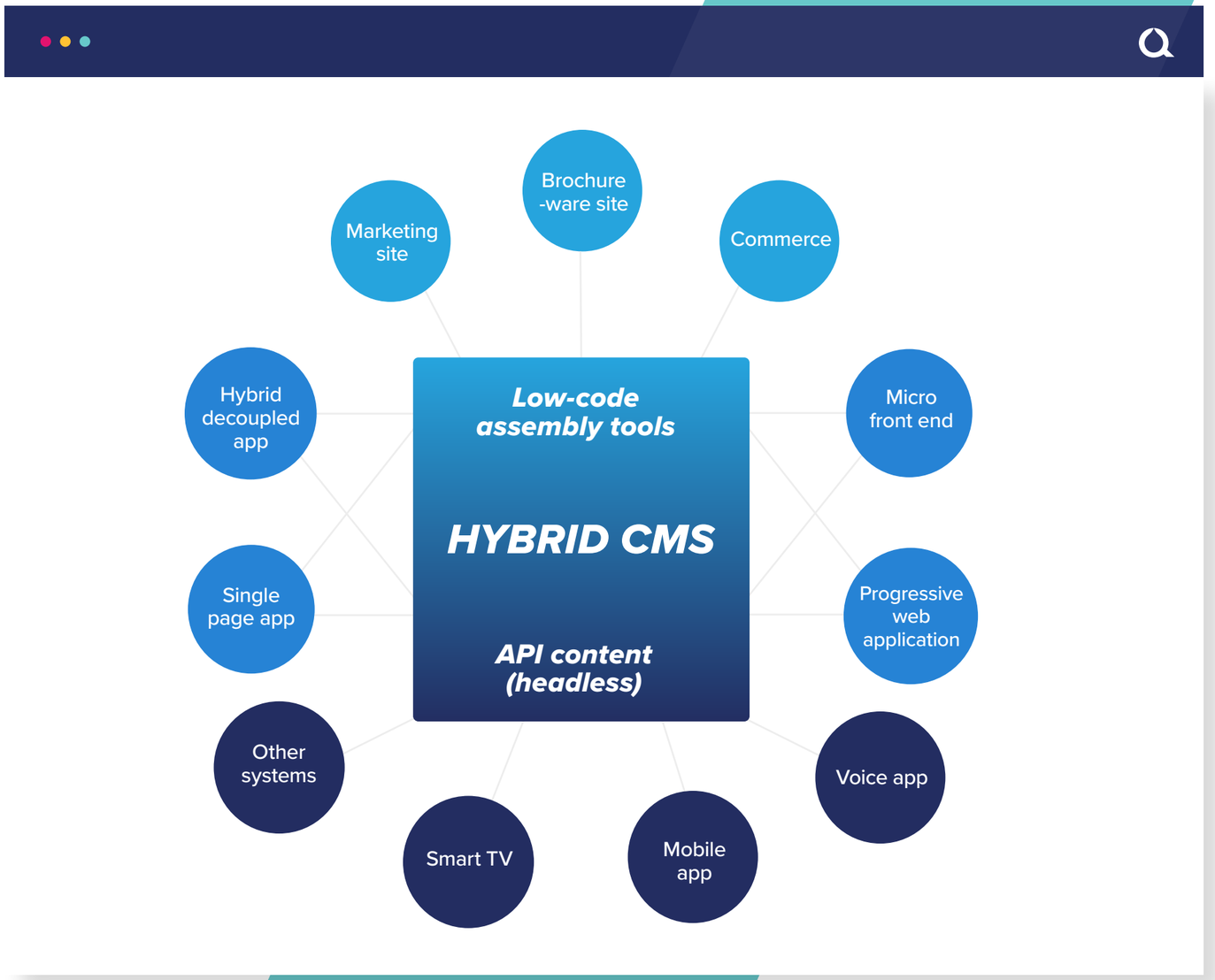
第2章

ハイブリッドCMSの定義



まず最初に、従来のCMSとヘッドレスCMS、ハイブリッドCMSの違いをご存知でしょうか？

- **従来のCMS:** ユーザーはエディターでコンテンツを作成し、それをデータベース（バックエンド）に保存。そのコンテンツは、バックエンドと緊密に結合したフロントエンドのレンダリングレイヤーに提供される。
- **ヘッドレスCMS:** ユーザーはエディターでコンテンツを作成し、APIでフロントされた独立したデータベースに保存。コンテンツは、これらのAPIを介して別のフロントエンドのレンダリングレイヤーによって取得される。
- **ハイブリッドCMS:** 従来のCMSとヘッドレスCMSを融合したもの。ユーザーはエディターでコンテンツを作成し、データベースに保存。コンテンツは、既存のフロントエンド・レンダリング層から提供することも、APIを介して全く別のフロントエンド・レンダリング層から取得することも可能で、柔軟に対応できる。



どの場合でも、フロントエンドのテンプレートは、デジタルアプリケーションにおけるコンテンツの表示方法を決定します。しかし、このタスクで使用できる言語とフレームワークの状況は変化します。チャネル間で一貫した体験を求めるビジネス（および消費者）の要求に応えるため、チームは複数のフロントエンド・フレームワークの専門知識を持つ必要があり、ここ数年でそのリストは変化しています。

従来のWebサイト開発で人気のある言語はPHP、.NET、Javaなどですが、アプリケーションサーバーとしての人気の高まりにより、Node.jsが爆発的な盛り上がりを見せています。2020年の**スタックオーバーフロー社の調査**によると、フロントエンドのWeb開発におけるプログラミング言語は、今もJavaScriptが主流であり、何年も前から変わっていません。

従来のCMSアーキテクチャと、ヘッドレスやデカップリング型アーキテクチャを比較検討する場合、そのアーキテクチャに最適な使い方を理解することが重要です。例えば、Node.jsのWebサーバ上でJavaScriptのフロントエンドフレームワークを使用するプロジェクトの大半は、Node.js固有の強みであるリアルタイム、または非同期の機能性に依存するのが一般的です。通常の使用例は、ノンブロッキングでシングルスレッドを維持する能力が中心となります。一般的な例としてはReactアプリが挙げられるでしょう。ページを更新することなく、複数のビューまたは APIエンドポイントを組み合わせて使用します。

この機能の他の一般的な例としては、以下のようなものがあります。

- **ダイナミックなAPIファクトリー**
- **リアルタイムまたはデータストリーミング**
- **チャットボット/チャットクライアント**
- **WebSocketによるメッセージ**

デカップル型アーキテクチャを検討している組織で、以下の要件に該当する場合は、あまり適していない可能性があります。

- **コンテンツチームのための本格的なエディトリアル体験**
- **表示やレイアウトを頻繁に操作する必要がある**
- **アプリケーション発売前のエンドツーエンドプレビュー**
- **デジタルアプリケーションのインフラを維持するために、開発者のリソースを最小限に抑えたい**

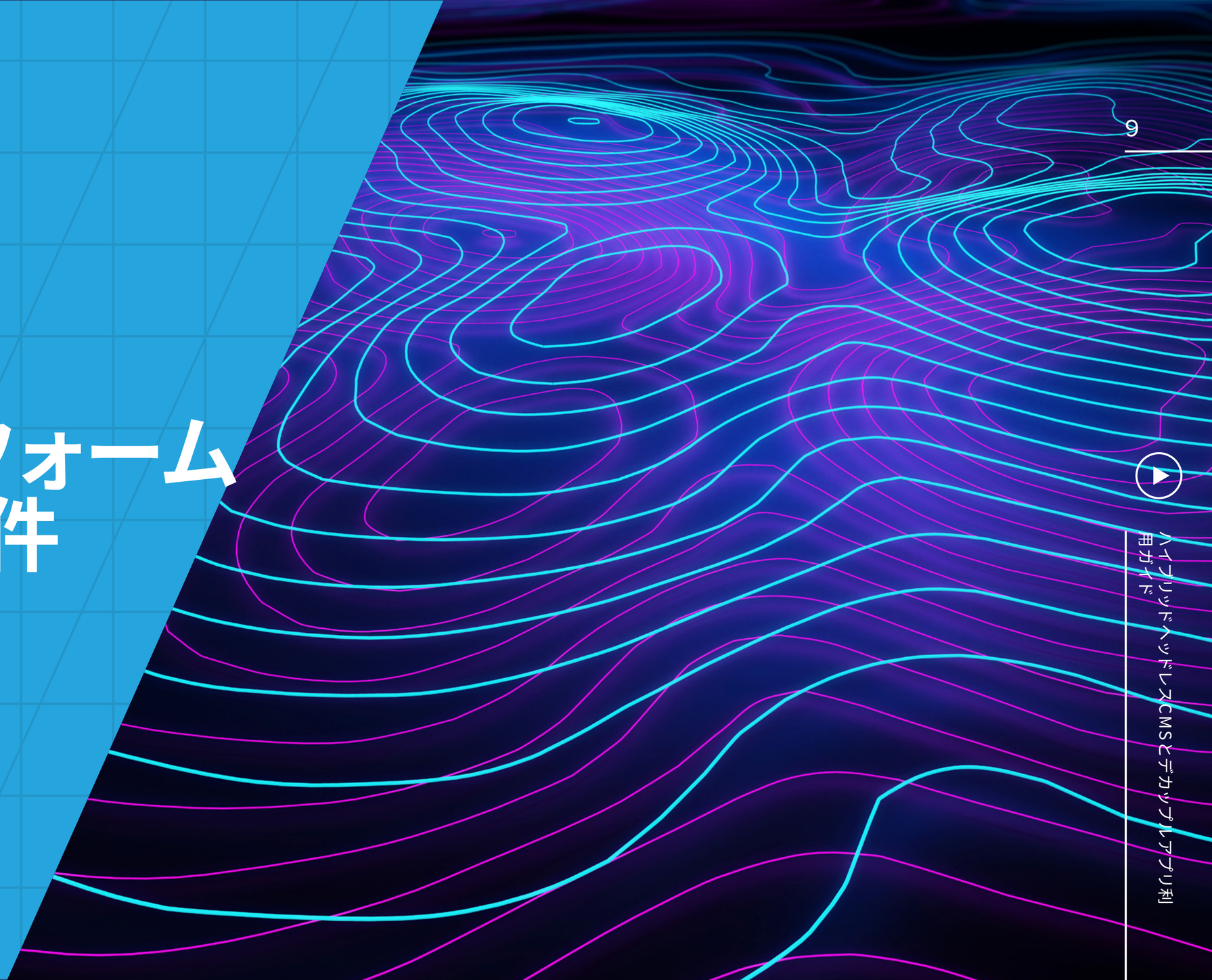
このため、デフォルトではハイブリッドCMSを使用することが推奨されています。ハイブリッドCMSはAPIファーストで、ヘッドレスと従来の実装の両方に使用することができます。これは、異なるソリューションを評価、購入、再トレーニングする必要なく、単一のツールセットを社内の異なるプロジェクトに使用、再利用できることを意味します。

DrupalはオープンソースかつAPIファーストのCMSで、最も強力なハイブリッドCMSソリューションの一つです。APIファーストとは、アプリケーションがコンテンツを取得するために、CMSに「電話をかける」ことでデカップリングを実現するものです。コンテンツは一カ所に集められ、外部に配信されます。Webサイトをサイロ化するのではなく、ハブ&スポーク方式で構築するのです。ハイブリッドCMSがハブで、スポークはシングルページのアプリやスマートテレビ、さらには他のアプリのバックエンドとなります。

これは、Drupalが従来のヘッドレス機能に「アドオン」することも、その逆も可能であることも意味しています。この柔軟性により、セキュリティ、パフォーマンス、開発スピードを犠牲にすることなく、チームに選択肢を与えることができます。

第3章

プラットフォーム選びの要件



デジタルプラットフォームのチームは、Web、モバイルWeb、モバイルアプリ、チャットボット、音声起動アプリケーションなどをサポートするためのシステムを必要としています。これらのチャネルをサポートするための選択肢としては、上記のアプローチによって異なります。Adobe、Sitecore、Episerver、Joomlaなど従来のCMSは、モノリシック・アーキテクチャに最も適しており、Built.io、Contentful、Prismic.ioのような、APIのみ（ヘッドレス）の新しいCMSシステムも人気を集めています。そして最後に、DrupalのようなAPIファーストの選択肢があり、デカップル型アーキテクチャを実現することができます。課題は、現在の事例だけでなく、組織の戦略ロードマップに適した選択肢を決定することです。

その決断は、予想される今後のデジタル環境によって決まるでしょう。チャットボット、音声対応アプリ、拡張現実は、これからさらに普及することが予想されます。問題は、デジタル環境に対して組織が対応する方法に、この変化がどう影響するかということです。

すべてをゼロから構築し、メンテナンスの負担だけでなく、膨大な時間がかかることを選択するのでしょうか？それとも、既製品のポイントソリューションを選択するのでしょうか？理想的なプラットフォームとは、再利用可能なコンポーネントやサービスのライブラリから特定のソリューションを構成することができ、なおかつ必要に応じて調整・カスタマイズできるものです。このスピード、ガバナンス、柔軟性のバランスが、現代のデジタル空間における成功の鍵と言えるでしょう。

信頼性の高いハイブリッドCMSを選択することで、単一のツールセットで複数の利用をサポートできるだけでなく、チームがお互いに邪魔になることなく、関連プロジェクトに取り組めるようになります。デジタルマーケティングのチームは、ローコードツールを使ってメインのWebサイトを構築・修正。モバイルアプリケーションのチームは、APIを介して同じコンテンツを利用することができます。　ハイブリッドにより、異なる方法で構築できるため、Webとモバイルのサイロがなくなり、デジタルアプリケーションは一つのチームになります。

これができれば、コンテンツの構成と見せ方を軸にチームを編成することができます。よって、フロントエンドの開発者をバックエンドの障害から解放することができるのです。

共通のツールセットを導入し、ボトルネックを解決することで、チームは継続的な開発手法を採用できるようになります。継続的開発では、アプリケーション開発のビルド、統合、テスト、デプロイの各ステージを自動化します。これにより、開発チームは新しいアプリケーションや機能をより早く、より効率的にユーザーに提供することができるようになります。ハイブリッド型のWebサイトやアプリケーションの開発では、このアプローチでフロントエンドとバックエンドの両チームが独立し、プロジェクトの目的に最適な構造化コンテンツモデルや、最新のプレゼンテーションを開発することができるようになります。

このアーキテクチャの利点を享受するためには、これらの利用をサポートするように設計されたプラットフォームが必要です。継続的な開発では、チームはコミュニケーションを円滑にする共通の開発ツールセットの恩恵を受けることができます。一つのプラットフォームによるフロントとバックエンドのアプリケーションホスティングのサポートは、インフラサプライヤーの統合、単一のサポート体制と品質保証など多くの利点があります。

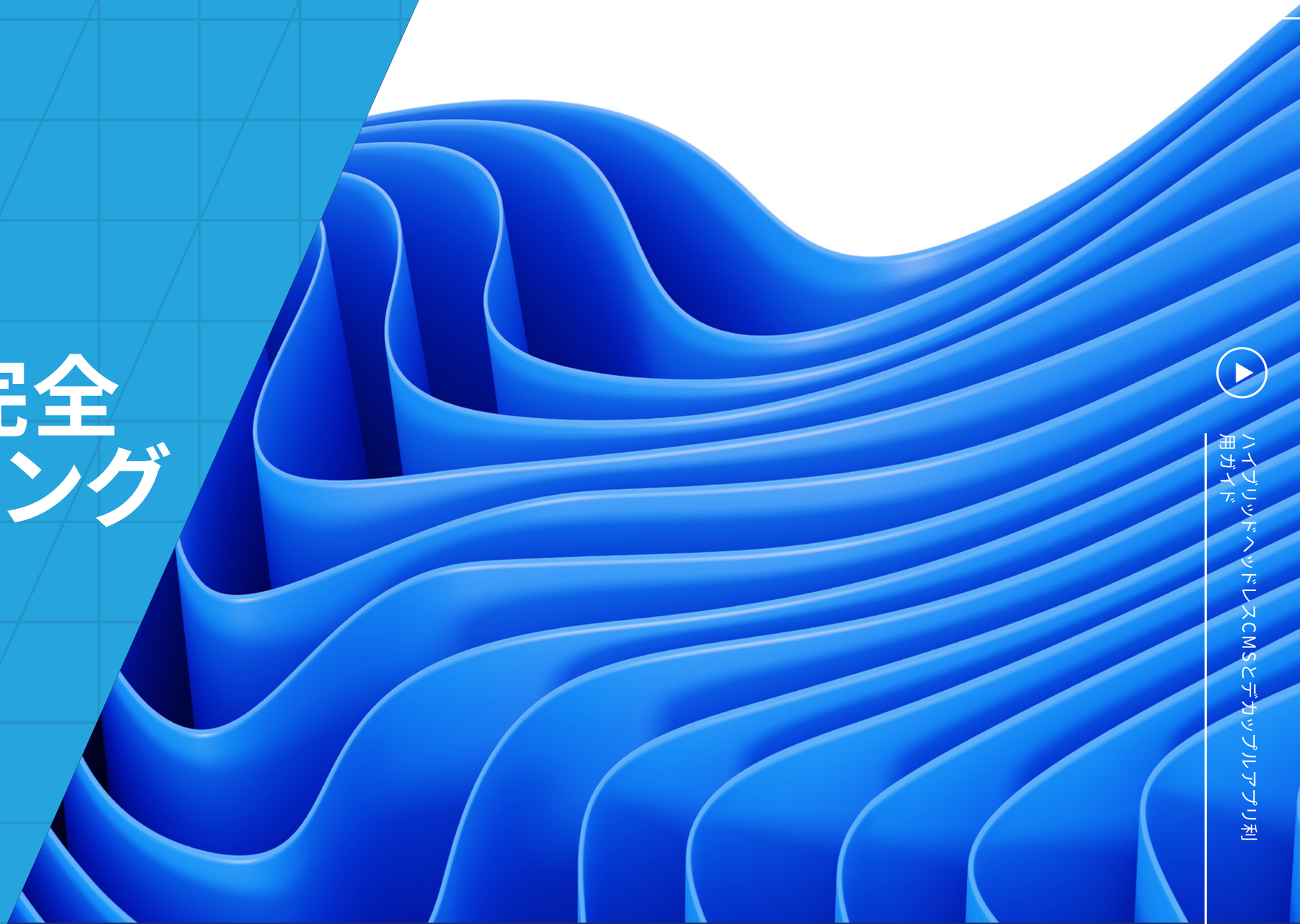
完全なオムニチャネルのデジタルエクスペリエンスを提供するには、プラットフォームを以下と簡単に統合できるか、これらを含める必要があります。

- **フロントエンドフレームワーク（例：Node.js）**
- **CMS （例：Drupal）**
- **クラウド型開発ツール**
- **パーソナライズドコンテンツの配信**
- **カスタマージャーニーの編成**
- **システムインテグレーション（Eコマース、マーケティングオートメーション）**
- **強固なAPIファーストのアプローチ**
- **世界クラスのチームによるアプリケーションとインフラのサポート**

シングルプラットフォームでJavaScriptによるデカップリングアプリケーションを作成する利点は、共通の開発ツールと、チームが使用する共通のUIです。何らかのビルドとテストの自動化を使用している場合、両者は同じツールを使用することができます。チームは、統一された管理コンソールと開発ツールを共有することで、より効率的に作業することができます。

第4章

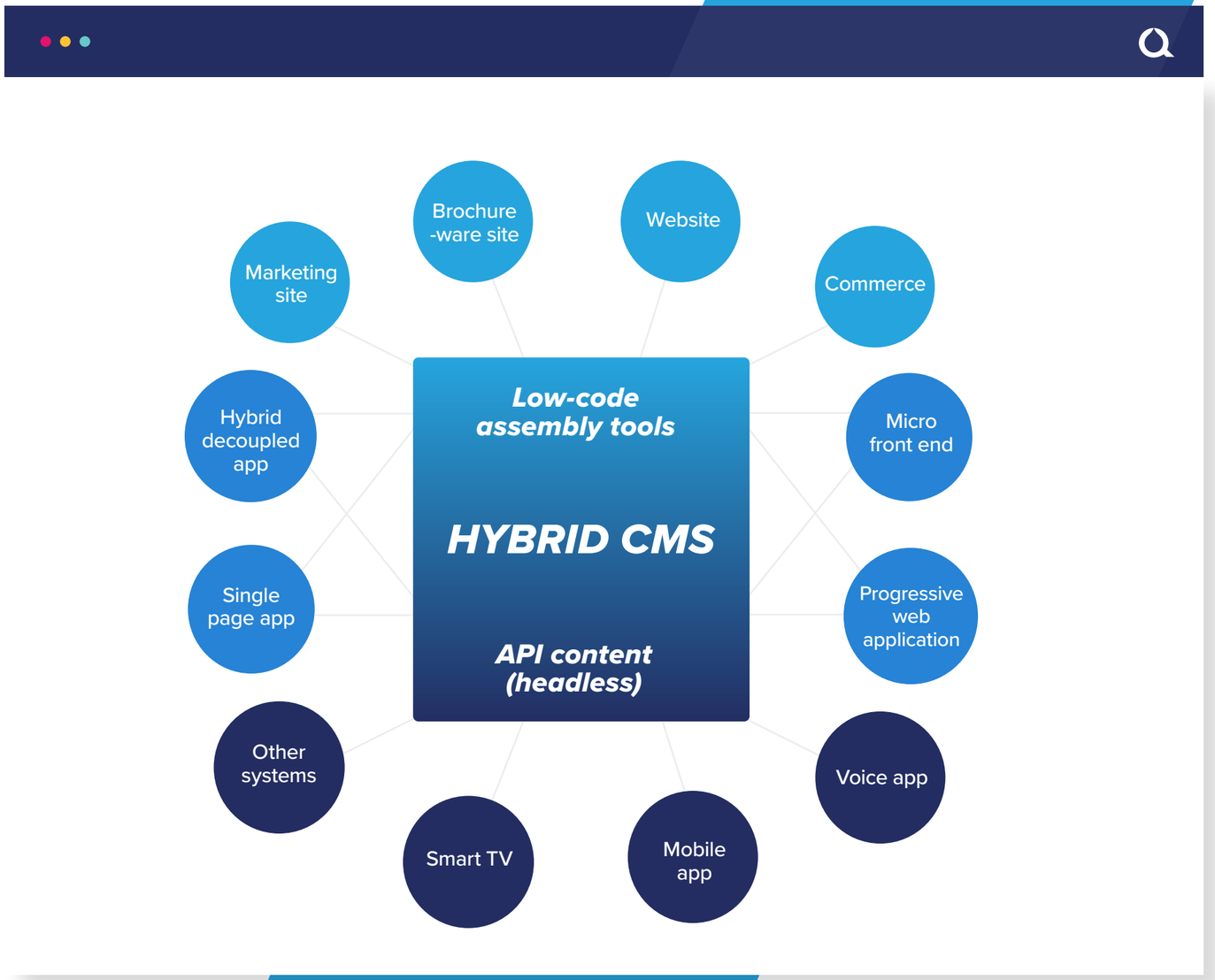
柔軟性：部分的と完全デカップリング



プロジェクトによっては、フルスタック、つまり統一されたDrupalアプリケーションを主要なソリューションとして選択する場合もあります。従来のモデルでは、フロントエンド全体がバックエンドサーバーによって管理されていました。しかし、Webサイトや追加のデジタルチャネルに対して、よりインタラクティブな技術への関心が高まっています。部分的なデカップリングアプローチは、ブラウザで特定のインタラクティブコンポーネントを表示するためにJavaScriptコンポーネントを使用できますが、Drupalのバックエンド内でキャッシュ可能なコンテンツと追加のワークフローも保ちます。

部分的なデカップリングは、プレゼンテーション層を管理するためのローコードツールを活用し、必要な場所にデカップリングコンポーネントを挿入することで、完全デカップリングアーキテクチャの多くの制限を緩和します。これにより、チームは再利用可能なコンポーネントのライブラリからエクスペリエンスを構成し、コードを展開する必要なく成果をもたらすことができます。ローコードのアセンブリによって、バックエンドのセルフサービスツールを使い、開発者でなくても安全かつ確実にエクスペリエンスを管理できるようになります。

完全なデカップリングは、開発者の手に大きな力を与え、より広くチームに展開されれば、効果的かつ効率的な開発ワークフローにつながるでしょう。CMSは完全にヘッドレスモードになり、デカップリングされたフロントエンドアプリケーションが消費するコンテンツとデータの作成と管理のみに使用されます。



部分的デカップリング

部分的なデカップリングは、従来のDrupalアプリケーションの利点と、標準のCMSコンテンツと組み合わせてDrupalで組み立てられる別のJavaScriptコンポーネントを使用するような、フロントエンドフレームワークの利点を組み合わせたものです。コンポーネントモジュールを使えば、JavaScriptの開発者はPHPやDrupalの専門知識がなくても、簡単にコンポーネントを追加することができます。彼らはコンポーネントデータを使用してフォーマットされたYAMLファイルを作成するだけで、Drupalがすべての依存関係とともにコンポーネントを自動検出してロードします。

CMSがそれを発見すると、コンテンツ制作者は他のコンテンツと同じように簡単にデカップリングコンポーネントを使用でき、ビジュアルページの作成者は、簡単にドラッグアンドドロップできるインターフェースでコンテンツを構成、組み立てができます。これは、すべてのエクスペリエンスではなく一部を切り離すのに最適な方法と言えます。

部分的なデカップリングにより可能になることは以下です。

- **スケルトンボイラープレートと、ページのダイナミックセクションのレンダリングと配信をDrupalに依存**
- **ダイナミックまたはキャッシュされない必要があるページのコンポーネントを切り離す**
- **DrupalのBigPipeモジュールを活用し、非同期で結合されたダイナミックコンポーネントの投入を行いながら、ページロードを高速化**
- **フロントエンドのレンダリングを一部オフロードすることで、サーバーリソースの使用量を削減し、ページの読み込みを高速化**
- **APIリクエストを最適化し、リクエストをより小さく、軽量にする**
- **JavaScriptコンポーネントの共有ライブラリの使用と貢献**

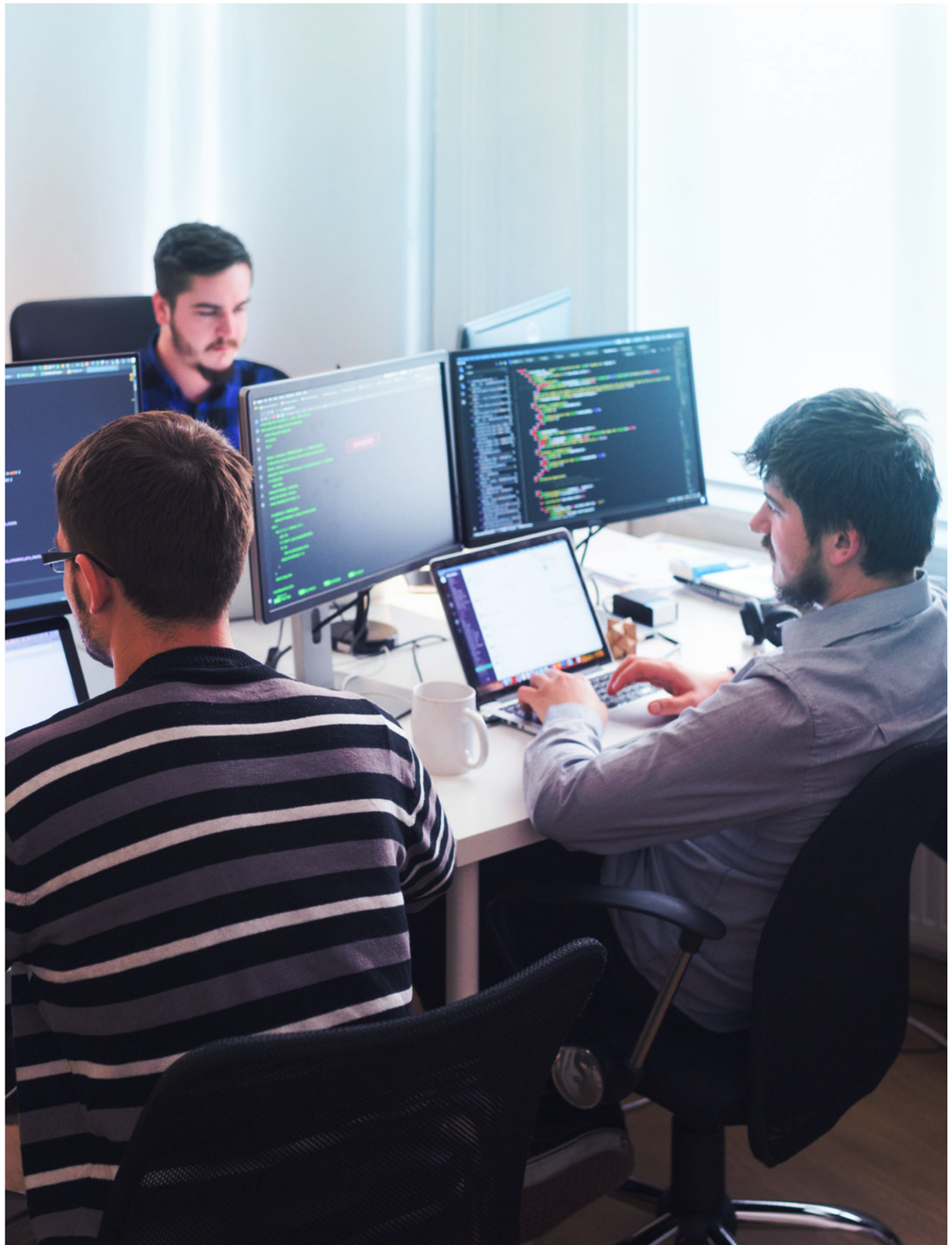
完全デカップリング

ReactなどのJavaScriptアプリケーションをイメージするときに思い浮かべるのが、フルデカップリングです。これは、エクスペリエンス管理の責任の多くを開発者チームに移行させる方法です。これにより、マルチチャネルのコミュニケーション管理が容易になる場合もあれば、コンテンツ（スマートデバイスなど）を表示する唯一の方法となる場合もあります。

完全にカスタム化されたアプリケーション間で無限のレベルの変動のバランスをとるためには、すべての分離されたアプリケーションの構築と管理に使用されるツールセットの概要を説明することが重要です。この場合、Webアプリケーションとともに、1つ以上のネイティブアプリケーションのサポートが必要になる可能性を考慮する必要があります。

ほとんどのメジャーなJavaScriptフレームワークは通常問題なく機能しますが、開発者が何を使いこなし、多様なツールがどの進化・成長しているのかも把握しておく必要があります。理想は、多目的ツールと標準的なアーキテクチャを選択し、できるだけシンプルにすることです。できる限り複雑さを排除し、アプリケーション開発に使用する方法は標準的なものにすることがベターでしょう。

デカップル型アプリケーションは、古いWeb（スマートフォン以前）の長所と、現在のWebが現在そして将来提供するものの長所を融合させたものです。しかし、これらの新しいものはすべて、すべてがつながるように開発される必要があります。デカップル型アプリケーションの構築方法を見ていると、1つのことが明らかになります。それは、JavaScriptが必要だということです。JavaScriptとAngular、React、Vueなどのフレームワークは、スタックオーバーフロー社が5万人以上の開発者を対象に行った調査で、2013年から最も人気のあるスクリプト言語としてランクインしています。



第5章

適切なJAVASCRIPTフレームワークを選ぶ

JavaScriptのフレームワークはたくさんあります。Drupalコアに追加された、最も人気のあるフレームワークの1つであるReactや、Express.js、Vue、Svelte、Angularなどが代表的です。開発者の立場から見たJavaScriptの課題は西部劇のようなもので、フレームワークの選び方は使い方と開発者の好みにかかっています。

しかし、Drupalを使用するのであれば、サポートされている統合パターンと強力なサンプルコードを持つ既存のフレームワークに注目することが自然でしょう。例えば、Contenta ディストリビューションには、いくつかサンプルのデカップル型アプリケーションが用意されていますし、Next.js インテグレーションでは、デカップル型アプリケーションのリアルタイムプレビューを提供する、最適化されたインテグレーションが提供されています。

検討すべきリソースをいくつか紹介します。

- **Acquia CMS**
- **Drupal State JavaScript SDK**
- **React用Drupal Next.js**
- **Drupal Tome - 静的サイトジェネレータ**
- **Drupal Gatsby**
- **Componentモジュール**
- **Decoupled Pagesモジュール**
- **Decoupled Kitモジュール**
- **すべてのデカップリング関連モジュール**

特定のソリューションでデカップル型アーキテクチャを選択したチームにとって、典型的な疑問は、デカップル型アプリケーションをどうするかということです。フロントエンドをまったく別のスタック、この場合はランタイムにNode.jsを含むJavaScriptスタックで実行する場合、CMSをサポートするスタックをセットアップしなければなりません。そして、Google CloudやAWSなど別の場所でフロントエンドのランタイムをセットアップし、そこで実行する必要があります。

アクイアが2017年9月にNode.jsのサポートを開始したとき、フロントエンドとバックエンドの両方のスタックを1つのUIで実行できるようになったため、お客様にとって複雑さが大幅に軽減されました。Node.jsのサポートは、デカップル型アプリケーションを開発するための、最適なスタック構成を提供することを中心に設計されています。Acquia Cloud Platform 上では、完全な Node.js スタックと LAMP スタックを構築するのではなく、完全な LAMP スタックと Node.js ランタイムのすべてを一つのプラットフォームでサポートすることで、デカップリングアプリケーションを実現します。

このように、真のハイブリッドCMSのサポートとデカップルアプリケーションサポートの組み合わせにより、アクイアのプラットフォームは従来のアプリケーション、ヘッドレス、デカップル、静的、ローコードアプリケーションのすべてを同じプラットフォームで作成できる強力なツールとなっています。

第6章

デカップル型API を使った ベストプラクティス

2017年10月、弊社がAcquia CloudでのNode.jsのサポートをリリースした矢先、私たちは自社という新しい顧客を獲得しました。

アクイアは、我々の多くのお客様が日常的に直面しているのと同じ状況、つまり、イベントと結びついたデジタル体験の壁に直面していました。弊社の場合、当社のグローバル年次イベント「Acquia Engage」のためのデカップル型アプリケーションを構築している時でした。

デカップリングプロジェクトを成功させるための最初のステップは、要件に関するアラインメントです。Acquia Engage の場合、それらは次のようなものでした。

- **プレゼンテーションや登壇者の情報をリアルタイムで提供**
- **エンゲージアワードのファイナリスト情報を部門別に紹介**
- **その他、カンファレンスに関連したコンテンツの紹介**
- **JSONなどの標準的なAPIからアプリケーションへコンテンツを統合**
- **複数のデバイスや画面サイズに対応したレスポンシブな動作**

要件がすべて決まると、次はデカップルされたDrupalのワークフローを理解することが必要でした。これを効果的に行うために行ったことは以下です。

- **タスクに応じた労力レベルの想定を対比させるために開発を活用**
- **同等に動作する2つのアプリケーションを構築する際の、デプロイメント要件に沿ったワークフローの標準化**
- **潜在的な問題点に関するコミュニケーションを合理化し、対象者の技術レベルに応じたビルドのトラブルシューティング方法を理解**

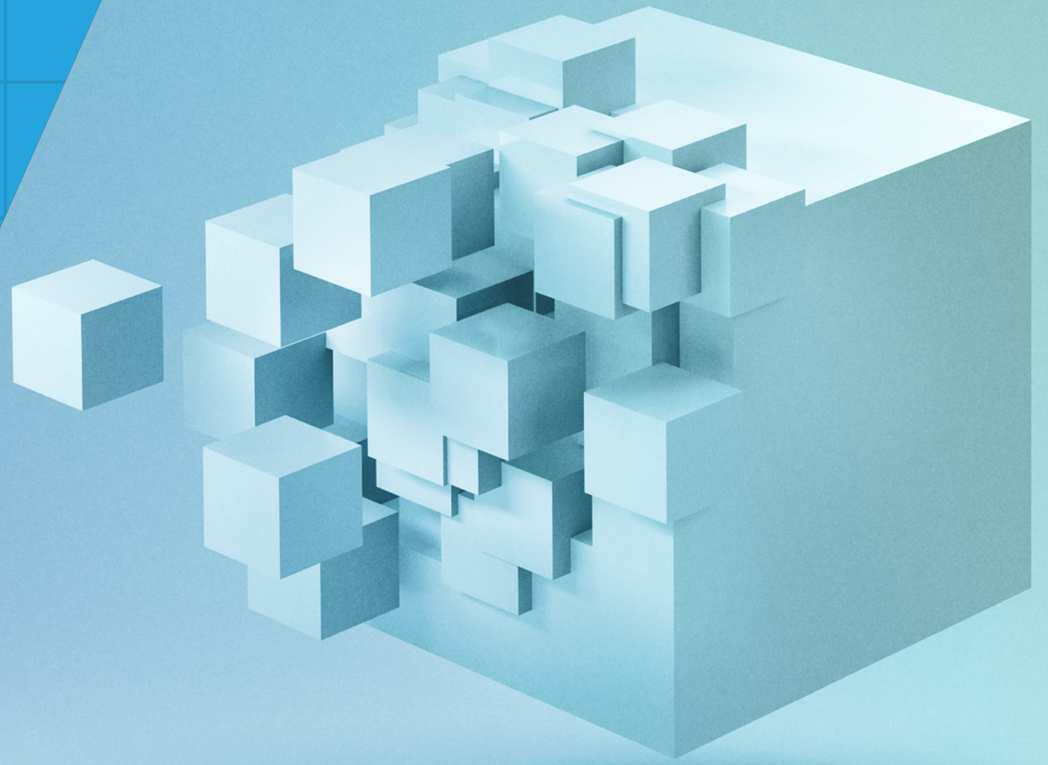
Acquia Engageのアプリは、Acquia CloudのNode.js環境上で実行されるようにJavaScriptで構築されています。開始する前に、Acquia Cloud Node.js ホスティングの制限や技術的な意見をチェックすることが重要でした。

これには、デカップルのDrupal用に構築されたNodeプラットフォームが、単独のNodeホスティングサービスとどう異なるかについての記述も含まれています。また、アクイアパイプラインに基づくデプロイサイクルの自動化という要件に沿って、アプリケーションを調整する必要がありました。

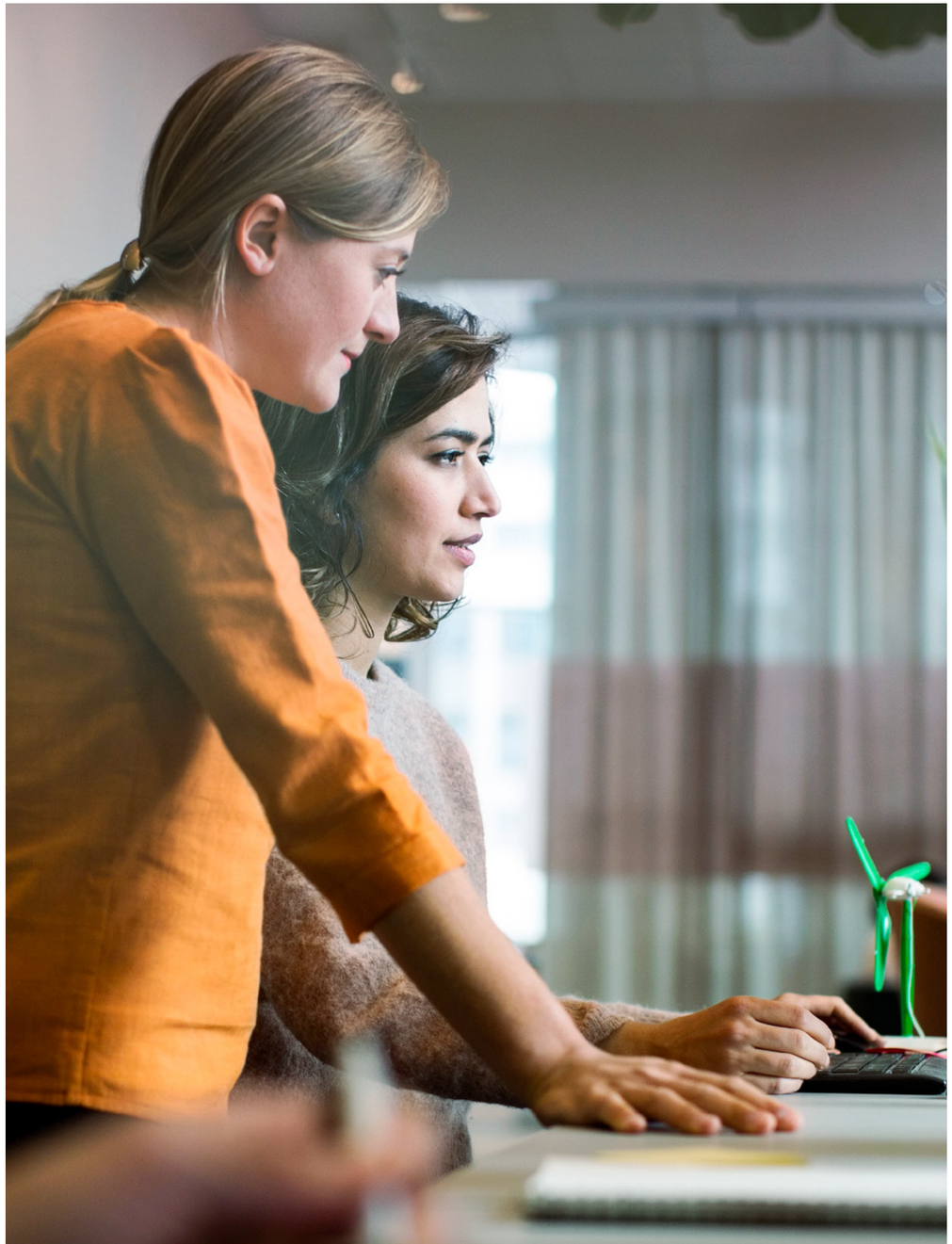
制限と技術的なオプションが確立されると、Acquia Engageのアプリ内でコンテンツを操作・更新する、開発者ではない作成者のためのコンテンツワークフローが有効になります。効率を最大化するために、マーケティングチームが必要としていたのは、標準的なCMSで管理されたワークフローでコンテンツを管理し、コードを展開することなくリアルタイムの更新を提供する機能でした。さらに、コンテンツの種類に応じてメディアアセットをアップロードする権限を必要としていました。これらの要件を満たすために、Drupalと統合されたカスタマイズをJavaScriptアプリケーションに引き継き可能にする必要がありました。

結果：

- **会議期間中、1日あたり約250人のユーザーをサポートすることに成功**
- **開発時間を30%短縮**
- **Node.jsの処理により、Engageアプリを通じてAcquia Engageの参加者にリアルタイムのデータを提示**
- **カンファレンス更新時のエディターワークフローが35%増加**



第7章
まとめ



デカップル型アーキテクチャは、21世紀に向けてアプリケーションを発展させるための基礎となるものです。実際、**フォレスター社**は次のように述べています。「マイクロサービスアプローチは、デジタルエクスペリエンスアーキテクチャの未来である。」にもかかわらず、このパターンに移行するには、よく考えて決断する必要があります。

使用例やデジタルアプリケーションを理解し、ニーズを満たすために最適なアーキテクチャに合わせるようにしましょう。APIを使ったコンテンツ配信、リアルタイムデータ、オムニチャネル対応などを優先するのであれば、デカップル型のDrupalを真剣に検討すべきです。Webやその他のデジタルアプリケーションのサクサク動くエクスペリエンスを目的とする場合、DrupalをNode.jsランタイムのサービスとして活用することで、素晴らしいフロントエンドエクスペリエンスを簡単に提供することができます。

一見すると無限の利点があるように見えますが、デカップル型のスタックを管理することの妥協点も考慮しましょう。アプリケーションを最適に動作させるために必要なサポートを、パートナーや社内チームは提供できますか？また、これまでとは異なる方法でアプリケーションを構築する、チームの準備はできていますか？デカップル型は、新しい開発ツール、言語、アプローチを導入する可能性を含みます。来るべき変化に対応できるように準備しましょう。最後に、フロントエンドとバックエンドの両チームのサポートに基づいて、プラットフォームのアプローチを評価することを確認しましょう。

次のステップ

アプリケーションに適したアーキテクチャの選択方法を学んだところで、Acquia CMSのアプローチ方法をチェックしてみましょう。

詳しくはこちら ▸

ACQUIA

ACQUIA.COM

アクイアについて

アクイアはオープンなデジタル体験プラットフォームを提供し、企業がウェブサイトやデジタルアプリケーションを通じて、大規模な規模で顧客とのコミュニケーションを構築、ホスト、分析、コミュニケーションを行うことを可能にします。信頼されるオープンソースのリーダーとして、柔軟なインテリジェンスを使用して、CXリーダーのためのより良いビジネス成果を生み出しています。