CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE INDAIATUBA

DR. ARCHIMEDES LAMMOGLIA

CURSO DE TECNOLOGIA EM REDES DE COMPUTADORES

VINICIUS AUGUSTO DE SOUZA

Estudo do conceito da tecnologia *headless* em um Sistema de Gerenciamento de Conteúdo (CMS)

Indaiatuba
2024

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE INDAIATUBA

DR. ARCHIMEDES LAMMOGLIA

CURSO DE TECNOLOGIA EM REDES DE COMPUTADORES

VINICIUS AUGUSTO DE SOUZA

Estudo do conceito da tecnologia *headless* em um Sistema de Gerenciamento de Conteúdo (CMS)

Trabalho de Graduação apresentado por Vinicius Augusto de Souza como pré-requisito para a conclusão do Curso Superior de Tecnologia em Redes de Computadores, da Faculdade de Tecnologia de Indaiatuba, elaborado sob a orientação do Prof. Dr. Wellington Roque.

Indaiatuba
2024

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE INDAIATUBA

DR. ARCHIMEDES LAMMOGLIA

CURSO DE TECNOLOGIA EM REDES DE COMPUTADORES

VINICIUS AUGUSTO DE SOUZA

Banca Avaliadora:

Prof. Wellington Roque	Orientador
Prof.ª Waldinelly Costa	Especialista na Área
Profª Giovana	Convidado

Data da defesa: 21/06/2024

AGRADECIMENTOS

Em primeiro lugar, a minha família que me auxiliou no meu processo de formação e realização deste trabalho. Ao professor Wellington que tanto me auxiliou no desenvolvimento como meu professor orientador. A FATEC por disponibilizar a oportunidade de crescimento acadêmico, profissional e pessoal.

“Alguns homens vêem as coisas como são, e dizem ‘Por quê?’ Eu sonho com as coisas que nunca foram e digo ‘Por que não?’”

(Geroge Bernard Shaw)

RESUMO

Atualmente, grandes empresas que trabalham no ramo digital necessitam de tecnologias que entreguem conteúdos de forma facilitada para suas plataformas, mas são encontrados obstáculos de como encontrar um sistema efetivo de gerenciamento de conteúdo que seja além de simplificado algo completo para seu negócio. Partindo dessas informações, o trabalho teve como objetivo apresentar e comparar o conceito da tecnologia *headless* dentro de um Sistema de Gerenciamento de Conteúdo (CMS) e a implementação tradicional deste sistema, além de desenvolver uma API e sua integração com uma camada de *front-end* para análise de suas vantagens. Para isso foram utilizadas tecnologias que são frequentemente procuradas no mercado, como o Adobe Experience Management (AEM), um CMS capaz de integrar o conceito *headless* dinamicamente feito com o GraphiQL, que é uma tecnologia já implementada na ferramenta, isto que auxiliou no estudo para os resultados de desenvolvimento de uma página simples, pois trouxe evidências de um sistema completo que com sua arquitetura desacoplada atendeu a facilidade e trouxe o conceito de flexibilidade e inovação. Os resultados mostram que com o desenvolvimento e integração entre as camadas de *back-end* e *front-end* foi possível alcançar o conceito de um *headless* CMS, evidenciando sua capacidade de facilitar a entrega de conteúdo para cumprir o objetivo deste trabalho, conclui-se que foi possível por meio do CMS escolhido, criar e editar conteúdos que podem atender grandes empresas em sua gestão digital para melhorar a eficiência entre integrações das camadas de desenvolvimento.

Palavras-chave: CMS, Sistemas de Gerenciamento de Conteúdo, Desacoplamento, Arquitetura *Headless*.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

AEM	*Adobe Experience Management*
API	*Application Programming Interface*
CAGR	*Compound Annual Growth Rate*
CMS	*Content Management System*
CRM	*Customer Relationship Management*
CSS	*Cascading Style Sheets*
DAM	*Digital Asset Management*
HTML	*Hyper Text Markup Language*
IA	Inteligência artificial
IoT	*Internet of Things*
JCR	*Java Content Repository*
PHP	*Hypertext Preprocessor*
REST	*Representational State Transfer*
SEO	*Search Engine Optimization*
URL	*Uniform Resource Locator*

SUMÁRIO

INTRODUÇÃO

No mundo tecnológico a busca por facilidade, eficiência, flexibilidade e adaptabilidade são constantemente observados como prioridade para as pessoas, estes aspectos são usados com certa prioridade quando uma empresa busca marcar sua presença online, frequentemente procuram manter suas plataformas ágeis, relevantes e com o potencial de entrega de conteúdo para o usuário. Nesta Perspectiva o *Content Management System* (CMS) surge como alternativa primordial para a criação e manutenção de plataformas digitais.

Ultimamente há uma visão inovadora que vem se destacando entre os especialistas da área de tecnologia, os chamados *headless* CMS, de acordo com Niemeyer (2021), um *headless* CMS fornece uma maneira de gerenciar conteúdo sem estar vinculado a um *front-end* específico. Isso permite que os desenvolvedores usem qualquer pilha de tecnologia para apresentar conteúdo, proporcionando maior flexibilidade e desempenho.

Este trabalho propões um estudo sobre *headless* CMS contextualizando seu funcionamento, suas características, origens, benefícios e desafios. "Os *headless* CMS representam uma mudança fundamental na forma como as empresas gerenciam e entregam conteúdo digital" (Jones, 2019). Nos leva a conhecer como essa tecnologia está transformando o cenário digital, formando pessoas para atuar em empresas e oferecer experiências inovadoras em meio a um ambiente cada vez mais tecnológico.

No decorrer deste estudo, analisaremos a arquitetura *headless* no cenário atual e as perspectivas para o futuro, com o objetivo de fornecer uma visão abrangente das oportunidades e desafios pertinentes ao *headless* CMS, com isso este estudo visa colaborar para um entendimento mais amplo das últimas tendências em CMS, contribuindo com o aperfeiçoamento dos profissionais que atuam como desenvolvedores de softwares, promovendo táticas para melhorar o potencial das plataformas online.

Quanto à sua estrutura, o trabalho está organizado em dois capítulos. No capítulo I foram apresentados os conceitos do CMS, onde é mencionado toda a fundamentação teórica no que se refere às características, funcionamento e contexto histórico. No capítulo II apresentamos a parte prática contendo toda a metodologia do projeto. No capítulo III foi apresentado os resultados obtidos através da implementação de um *headless* CMS.

CAPÍTULO I

1. Fundamentação teórica

Neste capítulo são apresentados os componentes a serem utilizados no desenvolvimento deste trabalho, além da sua fundamentação teórica, a fim de auxiliar no seu entendimento, com base em artigos acadêmicos que possuem alguma correlação com este trabalho.

1.1 Sistema de Gerenciamento de Conteúdo (CMS)

Um *Content Management System* (CMS), é um software executado principalmente dentro do navegador e permite que você crie, gerencie e modifique um website e seu conteúdo sem a necessidade de conhecimento de programação. Assim sendo, ele possui uma interface gráfica para ajudar no gerenciamento de todos os aspectos do seu site. Você pode criar e editar *templates* de conteúdos, adicionar *assets* como imagens e vídeos, e montar o layout do site.

Segundo Salutes (2022) Saber o que é um CMS é essencial para desenvolvedores de portais na web. Isso porque esse sistema permite criar, publicar e gerenciar um site e seu conteúdo de forma simples e rápida.

1.2 Detalhando o funcionamento de um CMS

Sem um sistema CMS, é necessário o conhecimento em algumas linguagens de programação para criar um site, ou usar um criador de sites para isso. Além disso, seria necessário fazer o upload de todos os arquivos para o servidor manualmente, afinal, um site moderno consiste em duas partes principais: *front-end* e *back-end*.

O *front-end* é a parte do seu site que os visitantes conseguem visualizar, ou seja, posts do blog, imagens, vídeos, páginas de contato, formulário de inscrição etc. No entanto, a parte de texto é mostrada com a linguagem de marcação padrão HTML, enquanto o design é adicionado com CSS e Java Script.

Ao passo que o *back-end* consiste no banco de dados e nas funcionalidades do site. O conteúdo é armazenado no banco de dados e é disponibilizado no *front-end* sempre que o

usuário faz uma solicitação pela página. Portanto as funcionalidades do *back-end* podem ser escritas em diferentes linguagens de programação, como PHP, Python, Java script e outras.

Com a união entre as camadas e a integração de entrega de conteúdos e leitura de dados temos o CMS tradicional, onde é possível gerenciar tudo exclusivamente dentro do CMS, sendo necessário o uso de tecnologias fixas e com interações apenas dentro do sistema.

1.3 Definição e características principais de um headless CMS

Um CMS com arquitetura *headless* é um sistema que foca no *back-end*, ou seja, possibilitando que uma maior flexibilidade e customização tanto para os desenvolvedores quando para os gestores/criadores de conteúdo. Um CMS *headless* é um sistema que visa disponibilizar o conteúdo gerenciado a partir de uma Application Programming Interface (API), na maioria dos casos essas APIs seguem a abordagem Representional State Transfer (REST) ou GraphiQL, ambas utilizadas com frequência na construção de aplicações para distribuição de conteúdo, isso separa seus dados/conteúdos, que representariam o "corpo", de como eles são apresentados, que seria a "cabeça", e disso surge o termo *headless* (sem cabeça).

Graças a essa separação, se tem uma grande liberdade em como e onde utilizar os conteúdos cadastrados. Os desenvolvedores podem usar qualquer tecnologia desejada sem se preocupar com o impacto da integração com o *front-end*, e os gestores podem reutilizar seus conteúdos em qualquer número de dispositivos e plataformas diferentes, não estando limitados a um único tipo de aplicação.

1.4 O início da tecnologia *headless*

A tecnologia *headless* é frequentemente associada a duas áreas: comércio *headless* e gerenciamento de conteúdo *headless*. Para o comércio, a tecnologia *headless*, como muitos outros avanços, surgiu em resposta aos problemas enfrentados pela indústria do comércio eletrônico onde os monolíticos, que são uma única aplicação de software em camadas no qual a interface de usuário e código de acesso aos dados são combinados em um único programa a partir de uma única plataforma, prosperaram em uma época em que o conjunto de produtos não exigia atualizações frequentes e o varejo na Internet era simples. A era pós-desktop alterou a forma como os consumidores consomem conteúdo, os negócios online já não eram o único mercado e as lojas de comércio eletrônico tiveram de lidar com todos os desenvolvimentos

digitais que ocorreram num curto período. Para os comerciantes, tornou-se um fardo que precisava ser aliviado o mais rápido possível. Os monóliticos, por outro lado, tornaram-se um obstáculo ainda maior. A tecnologia *headless* surgiu na busca de uma solução iterativa que também garantisse alterações futuras e tornasse um sistema maleável, não tendo mais o conceito de um único software com várias camadas.

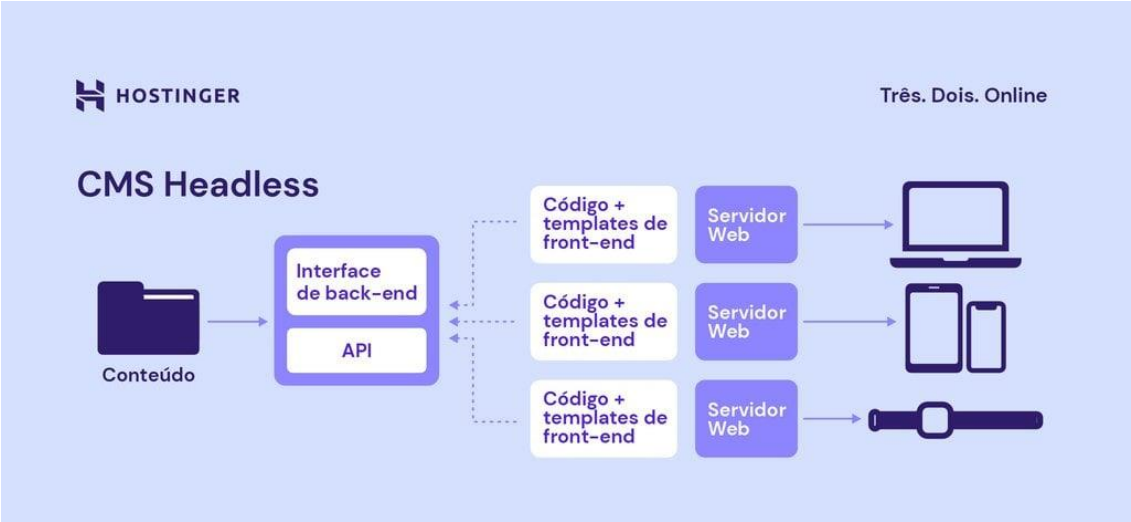
O termo *headless* "sem cabeça" é creditado a Dirk Hoerig em 2012. O conceito ganhou força à medida que as empresas de comércio eletrônico visavam oferecer experiências de compra dinâmicas em diferentes dispositivos. Empresas como a Nike começaram a explorar o comércio sem cabeça por volta de 2017, com o objetivo de aprimorar suas interfaces de compras on-line e fornecer uma experiência unificada na web, em dispositivos móveis e em outros pontos de contato. Desde o seu lançamento em 2018, a plataforma de comércio eletrônico direto ao cliente da Nike, alimentada por tecnologia sem cabeça, aumentou a receita em 6 mil milhões de libras.

Os primeiros usuários de CMS *headless* eram muitas vezes empresas de mídia e editores, reconhecendo a necessidade de distribuir conteúdo em vários canais, mantendo uma qualidade consistente. O New York Times, por exemplo, adotou uma abordagem de CMS *headless* em 2015 para agilizar a entrega de conteúdo em suas diversas plataformas. Provedores de CMS como o Drupal, que começou a negociar em 2001, começaram a falar sobre dissociar o front-end. Em 2016, Dries Buytaert do Drupal escreveu um blog falando pela primeira vez sobre o assunto. Contentful, foi um dos CMS *headless* mais populares que entrou no mercado em 2018. Embora essas tecnologias pareçam existir desde sempre, elas ainda são muito novas. Mas empresas como a Nike e o New York Times, como pioneiros, abriram caminho para uma adoção mais ampla de CMS e sistemas de comércio sem cabeça em vários setores.

1.5 Arquitetura *headless* e suas vantagens

Foi apresentado anteriormente que a arquitetura *headless* é um conceito de desenvolvimento de software que separa o *front-end* do *back-end*. Na Figura 1, é possível confirmar as características, onde se tem o conteúdo que se passa por uma API, a partir dela o *front-end* faz a leitura e exibe para quaisquer dispositivos.

Figura 1: Arquitetura de entrega de conteúdos via CMS *Headless*



Fonte: Hostinger, 2024.

É possível elencar quais são os mais importantes benefícios em utilizar esse tipo de arquitetura e quais os motivos que impulsionam sua adoção:

• Camada de apresentação separada: Uma abordagem *headless* te dá flexibilidade para escolher a melhor forma de construir o seu front-end. Graças a isso você não se encontra mais preso a uma tecnologia fixa ou ultrapassada.

• Para vários canais: permite que tenhamos possibilidades ilimitadas para que os gestores alterem diversos portais e canais sem a presença de um time de desenvolvimento.

• Experiência do cliente: O *front-end* é onde toda a jornada do consumidor ocorre, a habilidade de controlar essa camada do sistema separadamente ajuda a cliente a melhorar e personalizar a experiência do usuário, resultando em um número maior de conversões e faturamento.

• Vantagens competitivas: Outro benefício dessa arquitetura é a agilidade e flexibilidade para enfrentar um mercado altamente competitivo. Isso permite a utilização das melhores tecnologias disponíveis no momento e faz com que seu negócio se torne altamente escalável.

Essas são algumas das principais vantagens de optar pelo uso dessa solução.

1.6 *Headless* CMS x CMS tradicional

A principal diferença entre um CMS do tipo *headless* e um CMS tradicional é que o primeiro oferece apenas a funcionalidade de *back-end* e sua camada de apresentação fica por responsabilidade de outras aplicações, segundo Bruno S. (2024), é incluído também como diferenças para uma tecnologia com implementação *headless*, a flexibilidade, escalabilidade, velocidade de desenvolvimento, personalização e integração. Em contrapartida, a arquitetura do CMS tradicional oferece uma solução unificada para gerenciar o conteúdo e a camada de apresentação, onde se limita a utilizar apenas ferramentas disponibilizadas por aquele sistema.

Um CMS tradicional, também conhecido como sistema monolítico de gerenciamento de conteúdo, só permite que o conteúdo seja renderizado em um único front-end: uma página da Web. Assim, os profissionais de marketing precisam redirecionar suas peças de conteúdo se quiserem publicá-las em outras plataformas.

Normalmente, um CMS monolítico consiste em:

- Um banco de dados para armazenar, ler e gravar o conteúdo;
- Uma página de administrador para tudo que se relaciona à criação do conteúdo;
- Uma camada de apresentação para apresentar o conteúdo.

Por outro lado, a arquitetura *headless* utiliza a framework de modelo de conteúdo. Ela divide o conteúdo em partes separadas de acordo com sua finalidade, resultando em um conteúdo mais estruturado. Por exemplo, a modelagem de conteúdo permite que um CMS *headless* armazene o título, o corpo, os elementos visuais e as *tags* de um determinado conteúdo digital separadamente. Isso contribui para que os profissionais de marketing personalizem e reutilizem o mesmo conteúdo em diferentes plataformas de forma prática.

1.7 O que é arquitetura *headless*

Basicamente, uma arquitetura *headless* refere-se a uma abordagem de design de software em que o *front-end* e o *back-end* de um aplicativo que são dissociados, permitindo que funcionem de forma independente. A arquitetura *headless* dissocia esses componentes, diferentemente das configurações monolíticas tradicionais, onde o *front-end* e o *back-end* estão totalmente integrados.

Nesse paradigma, o *front-end* passa a ser uma interface que consome dados e serviços por meio de APIs fornecidas pelo *back-end*. Essa dissociação permite que os desenvolvedores

utilizem diversos front-end – abrangendo web, dispositivos móveis, dispositivos IoT etc., todos exibindo a interface com o mesmo *back-end*, promovendo agilidade, escalabilidade e iteração rápida.

Configurações monolíticas, caracterizadas por componentes de front-end e back-end totalmente integrados, podem prejudicar a escalabilidade e a flexibilidade à medida que os aplicativos aumentam em complexidade. O modelo monolítico muitas vezes exige que todo o aplicativo seja reimplantado, mesmo quando são feitas pequenas alterações, levando a ineficiências. As arquiteturas *headless* abordam esses problemas dissociando o *front-end* e o *back-end*, permitindo o desenvolvimento e a implantação independentes, superando assim as limitações dos sistemas monolíticos.

1.7.1 Casos de uso da arquitetura *headless*

Os sistemas *headless* podem ser aplicados em vários campos. Podemos nos deparar com CMS *headless* em sistemas de pagamento *headless* e mecanismos de pesquisa. Mas ainda assim, o caso de uso de *front-end headless* mais visto aparece no comércio eletrônico, onde essa arquitetura é uma solução muito popular.

Ao testar a versão *headless*, é possível analisar que as opções de design são ilimitadas e você terá controle total sobre o código em execução nos dispositivos de seus usuários, permitindo rastrear e agilizar praticamente todas as interações.

Quando o *front-end* é pesado, com um design fraco e interface de usuário muito complicada geralmente é uma descrição rápida de sua solução de comércio eletrônico mudar para a arquitetura *headless*, isto que simplificará os processos e implementará novas mudanças com mais rapidez.

Mais notavelmente, o *headless* oferece a flexibilidade necessária para atender a requisitos desafiadores. Pode ser difícil atingir seus objetivos se você precisar modificar fortemente um produto multifuncional, combinar uma ferramenta *headless* com um *front-end* menor pode ser a maneira mais fácil de entregar os designs e fluxos de usuário desejados.

Caso o projeto atenda perfeitamente aos requisitos de um projeto com uma ferramenta multifuncional, então as opções *headless* provavelmente serão as ideais. Da mesma forma, se no desenvolvimento o time estiver perfeitamente satisfeito e bem familiarizado com sua solução multifuncional atual, realmente não há necessidade de se preocupar em dividir as ferramentas de *front-end* e *back-end*. No entanto, se o time de desenvolvimento estiver enfrentando as

limitações de suas ferramentas, o *headless* permitirá que você resolva seus pontos problemáticos diretamente.

Este modelo resolve um dos maiores desafios dos sites tradicionais baseados em CMS, o conteúdo dentro desses sistemas só pode ser renderizado usando os temas do lado do servidor disponíveis para a plataforma. Construir um novo aplicativo Android ou iOS requer um mecanismo de ponte para expor o conteúdo do CMS de uma forma que seu código nativo possa acessar, a abordagem centrada em API *headless* evita esse problema, em vez de criar temas para o seu CMS, você carrega conteúdo dele em qualquer situação que desejar.

Ao observar atentamente o cenário *headless*, descobriremos que as ferramentas *headless* limitam intencionalmente seu escopo funcional e fornecem maneiras de integração em sistemas maiores. A separação de recursos específicos é benéfica quando os sistemas se tornam mais complexos, é mais fácil fazer escolhas específicas que limitem o custo, a segurança, a manutenção e os requisitos de hospedagem de espaços de código maiores quando você trabalha com ferramentas menores e focadas.

É importante notar que as opções *headless* geralmente exigem que você mesmo escreva algum código. No entanto, como os *front-end* são cada vez mais um conjunto de componentes anteriormente construídos e muitas vezes um design pronto para uso preenchido com seus próprios dados, não deveria ser muito presunçoso esperar mais maneiras de misturar e combinar ferramentas especializadas e integrar perfeitamente opções *headless* sem escrever código.

O *back-end* perfeito para um projeto pode ser apenas uma assinatura SAAS ou a instalação de um projeto de código aberto, isso que pode ser integrado sem código a um *front-end* pronto para uso que atenda a todas as suas necessidades.

Podemos configurar um novo site e então escolher entre um conjunto de opções de CMS *headless* de diferentes fornecedores para gerenciar os dados completos do site. As opções são atraentes, quer você esteja interessado nelas pela flexibilidade da arquitetura do aplicativo, pelo controle da experiência do usuário ou por pensar cuidadosamente sobre a longevidade do seu serviço.

1.8 O futuro da indústria de CMS *headless*

O mercado de CMS *headless* está crescendo rapidamente, com um CAGR projetado de 22,4% de 2021 a 2028. Este crescimento está sendo impulsionado pela crescente demanda por flexibilidade e escalabilidade no gerenciamento de conteúdo, bem como pela necessidade de

maior segurança e desempenho. Além do crescimento do mercado de CMS *headless*, há uma série de tendências e desafios emergentes que estão moldando o setor.

Esses incluem:

• A ascensão das arquiteturas combináveis: As arquiteturas combináveis são uma nova abordagem para a construção de software que permite às empresas misturarem e combinar diferentes componentes para criar uma solução personalizada. Essa abordagem está se tornando cada vez mais popular entre os CMSs *headless*, pois permite que as empresas criem um sistema de gerenciamento de conteúdo adaptado às suas necessidades específicas.

• O uso crescente de inteligência artificial (IA): a IA é usada de várias maneiras para melhorar a experiência de CMS sem cabeça, como automatizar a criação e entrega de conteúdo, personalizar experiências de conteúdo e melhorar a segurança.

• O crescimento da Internet das Coisas (IoT): A IoT está criando uma onda de dispositivos que precisam ser gerenciados e atualizados com conteúdo. CMSs *headless* são adequados para gerenciar esse tipo de conteúdo, pois podem entregar conteúdo facilmente para qualquer dispositivo, independentemente de seu formato ou sistema operacional.

Os CMS *headless* estão se tornando cada vez mais populares à medida que as empresas percebem os benefícios de separar o gerenciamento de conteúdo da apresentação. É provável que esta tendência continue à medida que mais e mais empresas adotam CMS *headless* para melhorar o gerenciamento e a entrega de conteúdo.

À medida que o mercado de CMS *headless* continua a crescer, esperamos ver ainda mais inovação e adoção desta tecnologia. Isso trará ainda mais benefícios para as empresas, como maior segurança, flexibilidade e escalabilidade. Quando a pessoa estiver escolhendo um novo CMS, definitivamente vale a pena considerar um CMS *headless*, pode ser a melhor opção para o seu negócio, especialmente se ela tiver requisitos de conteúdo complexos ou precisar entregar conteúdo para vários canais.

1.9 Como escolher um CMS

Antes de se preocupar em escolher o CMS certo para a necessidade do projeto, é essencial saber se realmente é preciso um, eles são projetados para agilizar o processo de criação, edição e gerenciamento de conteúdo digital, principalmente para sites.

Deve-se considerar alguns pontos para aquisição de um CMS:

• O site exige atualizações frequentes de conteúdo, como artigos ou listas de produtos.

•. Existem vários autores de conteúdo colaborando no conteúdo do site.

• O site exige estruturação e organização flexível de conteúdo para alterações futuras.

Se estes itens forem considerados, há uma boa chance de que o projeto se beneficie dos recursos oferecidos por um bom CMS.

1.10 Integração e compatibilidade do CMS

É importante considerar a facilidade de integração com o sistema ou estrutura escolhida. Você pode estar pensando em adicionar algumas funcionalidades ao seu site, como um sistema de comércio eletrônico ou software de gerenciamento de relacionamento com o cliente (CRM), ou você pode estar apenas começando um novo projeto do zero.

De qualquer forma, é crucial garantir que o CMS selecionado possa se integrar perfeitamente a esses sistemas.

A maioria dos frameworks, como Next.js, Gatsby ou Astro, oferecerá uma lista de plugins CMS oficiais que agilizam o processo de integração. Um CMS pode ser adicionado a qualquer projeto sem um plugin oficial, utilizando a API do CMS de sua escolha e codificando tudo manualmente.

Mas geralmente é aconselhável procurar um CMS que seja apoiado pela sua estrutura, pois ele elimina grande parte do trabalho pesado da equação e permite que você se concentre em outras preocupações urgentes. Por exemplo, meu blog tem uma lista dos melhores CMS para Next.js para você escolher. A maioria das estruturas possui vários gerenciadores de conteúdos compatíveis e criados por desenvolvedores terceirizados.

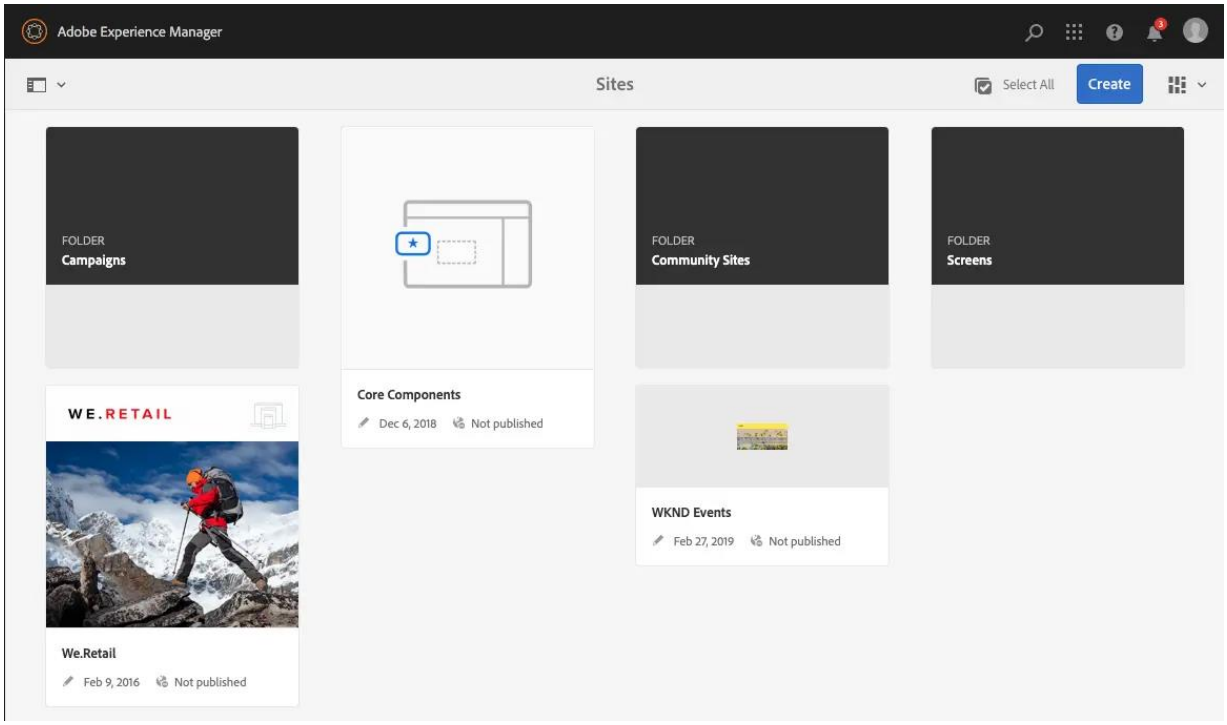
Ao procurar um CMS com integração e compatibilidade perfeitas, consulte a documentação oficial da sua estrutura para obter orientação, que deve informar quais bibliotecas públicas estão disponíveis.

1.11 Tecnologias utilizadas

A ferramenta utilizada para atender a necessidade de um *headless* CMS no projeto foi da Adobe, chamado de AEM (Adobe Experience Management) em sua versão de

desenvolvimento local, distribuído no próprio site para download, sua solução é para integrações a serviços tanto tradicionais como a optada de estudo no projeto que é o *headless*. Na Figura 2 é possível visualizar sua interface para desenvolvimento de templates.

Figura 2: Exemplo de página da interface do AEM



Fonte: Adobe *experience league*, 2024

Para a geração de APIs foi utilizado o GraphiQL, ele foi lançado publicamente em 2015 pelo Facebook e foi uma linguagem de consulta e um *runtime* para execução de queries em APIs. Os clientes podem solicitar exatamente os dados que precisam com GraphiQL, ao contrário das APIs REST convencionais. Isso aumenta a eficiência, a flexibilidade e a capacidade de interagir melhor com APIs complexas.

A fim de testar a obtenção da resposta dos dados enviados para uma suposta camada de front-end foi utilizada a ferramenta Postman que é uma plataforma popular para desenvolvimento, teste e documentação de APIs, inicialmente foi lançado como uma extensão do Google Chrome, agora é uma aplicação completa que funciona em várias plataformas, como Windows, macOS e Linux. Desenvolvedores e equipes de QA costumam usá-lo para interagir com APIs, o que torna os processos de desenvolvimento e teste mais produtivos.

Para atender as necessidades de testar os resultados do projeto *headless* CMS, foi realizada através de HTML, CSS e JavaScript uma página simples consumindo através da

biblioteca do jQuery onde facilita essa interação através de seus métodos AJAX, permitindo que desenvolvedores façam requisições a APIs de maneira simples e eficaz.

1.12 Trabalhos relacionados

Nesta seção são apresentados estudos e pesquisas relacionados ao tema deste trabalho. O levantamento realizado foi orientado pela busca de pesquisas científicas e/ou tecnológicas que têm em seus objetivos a análise de ferramentas, tecnologias para o *headless* CMS.

A ferramenta que serviu de referência para isso foi o Google Acadêmico por meio do qual se buscou mapear as pesquisas dessa natureza circunscritas nos últimos anos.

Edvaldo Donizeti Demarchi, Gabriel José de Deus Bonfim e Luciana Jose Garcia Ribeiro (2021), desenvolveram um blog com o objetivo de apresentar um projeto utilizando uma ferramenta CMS (Sistema de Gerenciamento de Conteúdo) para promover a comunicação e acompanhar o público-alvo com engajamentos com as redes sociais e canais de mídias expansivos. Para o desenvolvimento deste artigo, métodos e ferramentas foram utilizados para a criação de um passo a passo de como desenvolver um blog em formato CMS. A plataforma com layout e botões pré-definidas foi utilizada para a criação, facilitando o desenvolvimento do projeto e melhorar a interação com o desenvolvedor. A plataforma utilizada foi WordPress, um sistema Gerenciador de Conteúdo (CMS), como um dos mais utilizados da atualidade, e o objetivo era ajudar o usuário na criação de conteúdo sem iniciar uma linguagem de programação.

Daniela Alexandra Coelho Silva (2022), seu estudo sobre migração de dados em uma empresa multinacional foca no tema software de gerenciamento de dados e plataformas de comércio eletrônico. O estudo explica como é realizada a migração de dados e os diferentes métodos utilizados nos processos de migração. As empresas que adquirem seus CMS e sistemas de e-commerce há anos percebem que as tecnologias que utilizam não correspondem ao que os usuários procuram por diversos motivos. Muitos destes sistemas estão chegando ao fim do seu ciclo de vida e muitas vezes precisam de atualizações para acompanhar a evolução tecnológica. Muitas empresas começam a implementar a migração para os sistemas mais recentes, o que pode ser uma limitação, pois a maioria das empresas carece de conhecimento técnico sobre plataformas e projetos de migração. O estudo utilizou um estudo de caso de uma única organização, que apresentou suas origens, nova aplicação e a técnica utilizada para migração. A organização adquiriu uma empresa dinamarquesa de grande porte em 2019 e, desde então,

desenvolve um projeto de atualização de suas tecnologias. A organização reconhece a necessidade de alinhar a sua imagem e atividade digital através de uma ferramenta mais desenvolvida, com a intenção de expandir a plataforma online para promover a sua identidade, valores e essência.

Lennon Vinicius Moreno Del Antonio, Nidia Mara Melchiades Castelli (2023). O estudo demonstra que um Sistema de Gerenciamento de Conteúdo (CMS) pode ajudar uma microempresa a iniciar operações no mundo digital sem a necessidade de uma empresa especializada em criação de sites. O WordPress, um CMS popular e intuitivo, foi escolhido para este estudo, pois permite ao microempreendedor construir seu site sem amplo conhecimento de programação web. Outras ferramentas CMS, como Weebly e Shopify, podem ajudar as microempresas a entrarem na era digital com seus próprios sites, utilizando recursos financeiros mínimos. A ferramenta CMS gratuita, que permite a qualquer pessoa gerir e publicar conteúdos, incluindo e-commerce, foi apresentada a uma empresa de vestuário. Os proprietários da empresa ficaram satisfeitos com o design, layout, imagens, cores e facilidade de uso do site. Eles confirmaram que o carrinho de compras funciona corretamente e pode adicionar, remover ou modificar seus produtos de forma rápida e simples usando o WordPress. O estudo destaca o potencial das ferramentas CMS para permitir que as microempresas entrem na era digital.

Após o estudo desses trabalhos concluímos que o CMS tem muito a contribuir com diferentes empresas, de pequeno a grande porte e até mesmo para criação de conteúdos em redes sociais. Observamos que grandes empresas buscam por atualizações e até mesmo por novos sistemas com o intuito de atingir suas expectativas, pois os avanços tecnológicos são visíveis e cada vez mais se busca ferramentas mais desenvolvidas.

CAPÍTULO II

2. Procedimentos Metodológico

Neste tópico foi apresentado todo o procedimento metodológico para a realização deste trabalho, considerando ferramentas utilizadas, análises de produto e objetivos de um CMS *headless* descrevendo todos os passos que serão realizados para a execução do projeto.

2.1 Caracterização de Pesquisa

Este estudo metodológico visou analisar como a implementação de um sistema de gerenciamento de conteúdo *headless*, também conhecido como CMS *headless*, afeta o processo de desenvolvimento de websites. A pesquisa examinou os efeitos práticos do uso de um CMS *headless*.

A implementação do *headless* CMS permitiu a gestão dinâmica do conteúdo do estudo, o que facilitará a análise contínua e a reflexão sobre as práticas de desenvolvimento. Finalmente, pretendemos fornecer informações úteis sobre os aspectos úteis da adoção de *headless* CMS no contexto do desenvolvimento online moderno.

2.1.1 Quanto aos objetivos

Este trabalho possui como objetivo trazer os motivos para utilização de um *headless* CMS através de uma pesquisa exploratória, sendo seu propósito enriquecer e aprimorar ideias, Tais métodos são empregados para proporcionar uma compreensão mais profunda e contextualizada do tema em estudo, permitindo aos pesquisadores explorarem diferentes perspectivas e hipóteses emergentes.

Para seguir ao destino do projeto é importante desenvolver uma solução capaz de deixar evidente que a utilização de um sistema desacoplado traz muitas vantagens a toda a arquitetura de um site. Serão considerados alguns pontos para a análise do objeto de estudo, como:

1. Examinar as vantagens e desvantagens percebidas da implementação de um sistema de gerenciamento de conteúdo sendo ele *headless* em comparação com métodos convencionais de CMS.
2. Investigar a eficácia do processo de desenvolvimento de websites ao usar um *headless* CMS, levando em consideração fatores como facilidade de implementação, manutenção simples e flexibilidade de design.
3. Avaliar a capacidade de adaptação e escalabilidade de websites desenvolvidos com *headless* CMS em comparação com abordagens de CMS tradicionais.

2.1.2 Quanto ao delineamento

O delineamento de um projeto de pesquisa refere-se ao planejamento e estruturação detalhados dos passos que serão seguidos durante a realização do estudo.

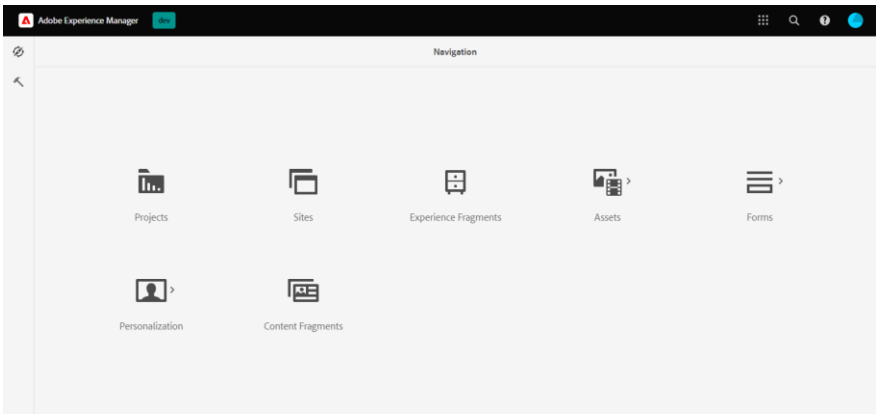
Dessa forma, todo o estudo para a validação do projeto é baseado nos tópicos feitos no capítulo, com isso o delineamento deste projeto visa investigar como facilitar a execução de projetos com o CMS sem esforço, sugerindo métodos e recursos que podem aumentar a eficácia e a eficiência do processo.

2.2 Configurando a Ferramenta de CMS

O AEM, é a plataforma de gerenciamento de conteúdo de negócios criada pelo Adobe Systems com ela foi possível criar, administrar e distribuir experiências digitais personalizadas em uma variedade de canais, como sites, aplicativos móveis e dispositivos Internet das Coisas (IoT).

A automação de marketing, o gerenciamento de ativos digitais, a personalização, a análise e o gerenciamento de conteúdo são alguns dos recursos do AEM que podem serem desenvolvidos no projeto. Empresas que buscam melhorar a experiência do cliente e otimizar os esforços de marketing digital costumam usá-lo. Na Figura 3 é mostrado sua interface inicial, onde se começam os desenvolvimentos.

Figura 3: Tela Inicial do AEM apresentados seus principais módulos



Fonte: Autoria própria

2.3 Estruturação de dados

O termo “modelo” (*model*) no *Adobe Experience Manager* (AEM) é basicamente um formulário utilizado para a organização e estrutura do conteúdo dentro de um sistema de gerenciamento de conteúdo (CMS). Uma representação ou estrutura de dados que descreve como o conteúdo é organizado, armazenado e mostrado no AEM é chamada de modelo.

Ao criar um modelo no AEM, foi possível definir quais componentes de conteúdo foram usados em uma página ou aplicativo. Ao garantir que o conteúdo seja consistente e padronizado, isso facilita a criação e o gerenciamento de conteúdo em larga escala.

Além disso, os modelos AEM podem ser usados para alimentar uma variedade de canais de saída, incluindo sites, aplicativos móveis e dispositivos IoT, garantindo uma experiência de usuário consistente em todos os pontos de contato. Em resumo, os modelos são essenciais para organizar e estruturar o conteúdo no Adobe Experience Manager.

2.3.1 Criando uma Model

Com o início de um Projeto em um *headless* CMS foi necessário estudar a estrutura de um site ou projeto para que o fosse possível separar cada parte do site em componentes definindo se eles devem ou não utilizar um modelo definido de conteúdo, este modelo geralmente é escolhido a componentes que se repetem várias vezes dentro de um site, como por

exemplo, um carrossel de banners onde geralmente dentro de um site ao navegar página a página é encontrado no topo imagens explicando o objetivo da página ou campanhas.

Para montar um componente deste foi analisado a Figura 4, o componente é mostrado pela área vermelha onde possuem alguns atributos dentro dele, como o título, subtítulo, texto do botão, link do botão e imagem do banner, então é essencial que seu modelo de conteúdo tenha estes mesmos atributos listados.

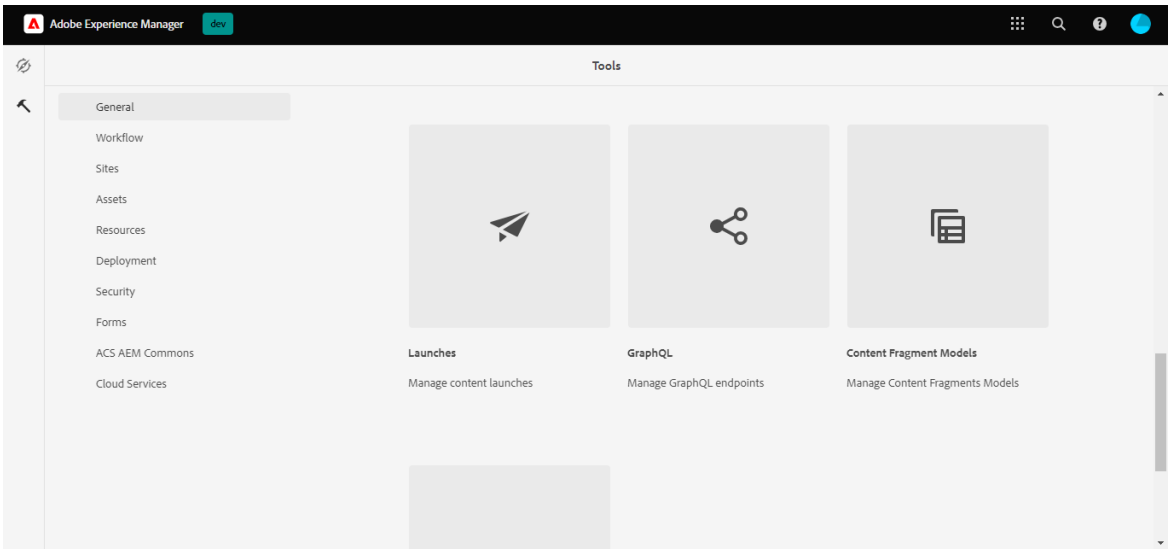
Figura 4: *front-end* para implementação da API do *headless* CMS

Fonte: Autoria própria

A Model também poderá servir de base para a construção de conteúdos informacionais a serem mandados para outras ferramentas, usando no contexto de IoT, por exemplo, para os painéis luminosos que é encontrado em ruas urbanas informando o estado da rua adiante, neste caso é possível usar os conteúdos dessa *model* para preencher um formulário que posteriormente dentro do contexto do AEM será chamado de um fragmento de conteúdo, este por fim será entregue via API criada por meio de uma *query* dentro do GraphiQL, uma solução de entrega de APIs adotada pela Adobe para o AEM.

As interfaces para criação e desenvolvimento ficam dentro de *tools*, demostrado na Figura 5, uma interface própria do console AEM para manipular *templates*, componentes, *tags*, *endpoints* através do GraphiQL, Modelos de Conteúdos (models) e acessar configurações da plataforma.

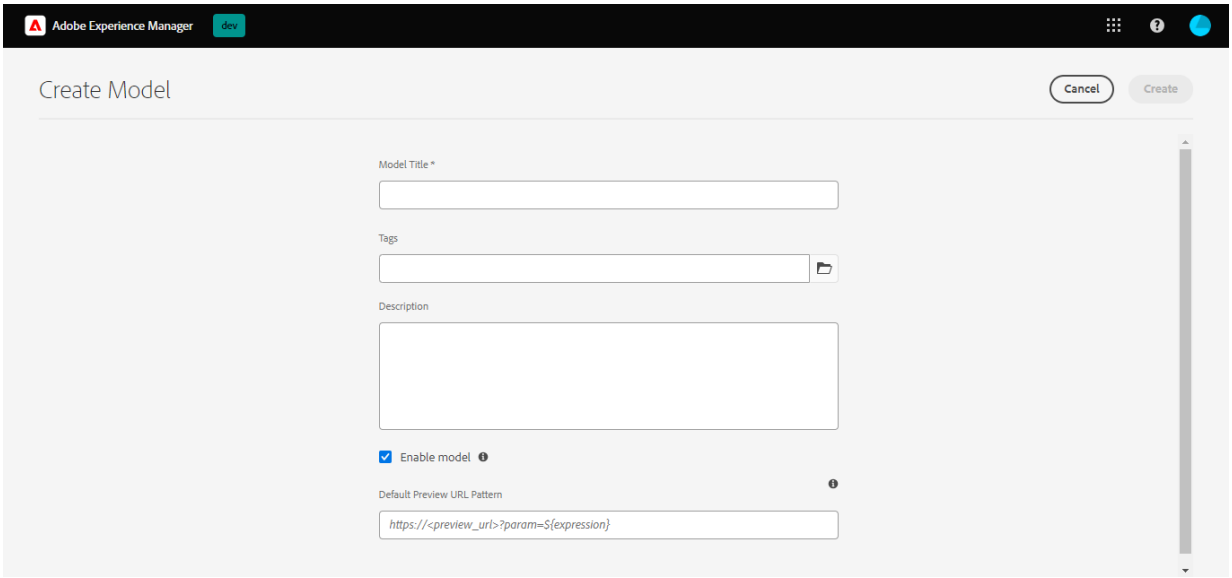
Figura 5: Interface de configurações do AEM



Fonte: Autoria própria

Ao selecionar a criação de uma nova *model*, foi necessário definirmos alguns atributos para a identificação, conforme é mostrada na Figura 6, o atributo obrigatório é O *Model Title*, nele atribuímos o nome para o model, os demais atributos *Tags* e *Description* são opcionais, mas são importantes para identificação deles dentro do projeto, como padrão deixamos a opção *Enable Model* ativa para que seja possível utilizar este modelo.

Figura 6: Tela de criação de um modelo de fragmento do conteúdo

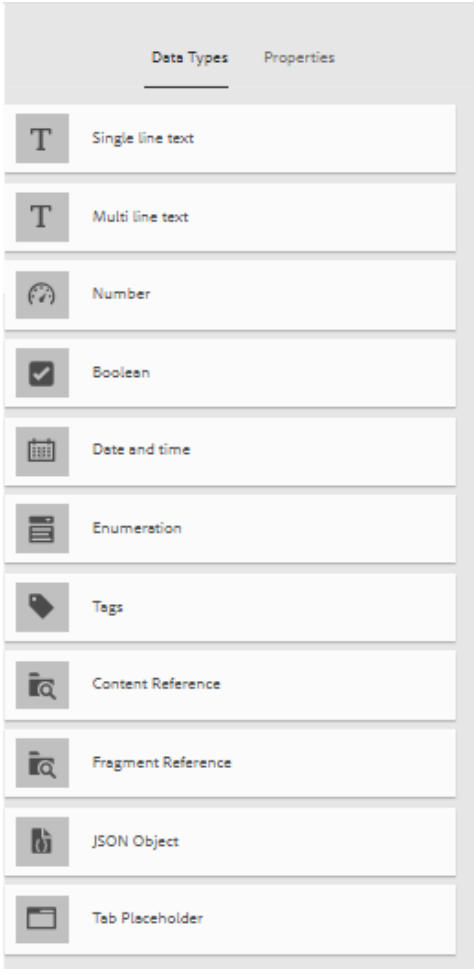


Fonte: Autoria própria

2.3.2 Escolhendo os *Data Types*

Os *Data Types* são basicamente tipos de dados que podem ser usados dentro de uma model para definir certas propriedades e configurações de atributos dos componentes, como demonstrado na Figura 7 tem alguns já definidos pela ferramenta, o AEM usa uma estrutura flexível que é o JCR (Java Content Repository), este é o responsável por armazenar e gerenciar os diferentes tipos de dados, também é possível criar outros tipos de dados usando o *nodetype definitions* na interface de desenvolvimento do AEM chamada CRXDE.

Figura 7: Lista de *Data Types* do AEM



Fonte: Autoria própria

A definição correta do *data type* no AEM foi fundamental para garantir que todos os dados sejam armazenados e manipulados corretamente e com isso ter uma gestão dos conteúdos eficiente, com isso é importante relacionar no Quadro 1 os *data types* identificados na Figura 7.

Quadro 1: Quadro de comparação entre os tipos de dados

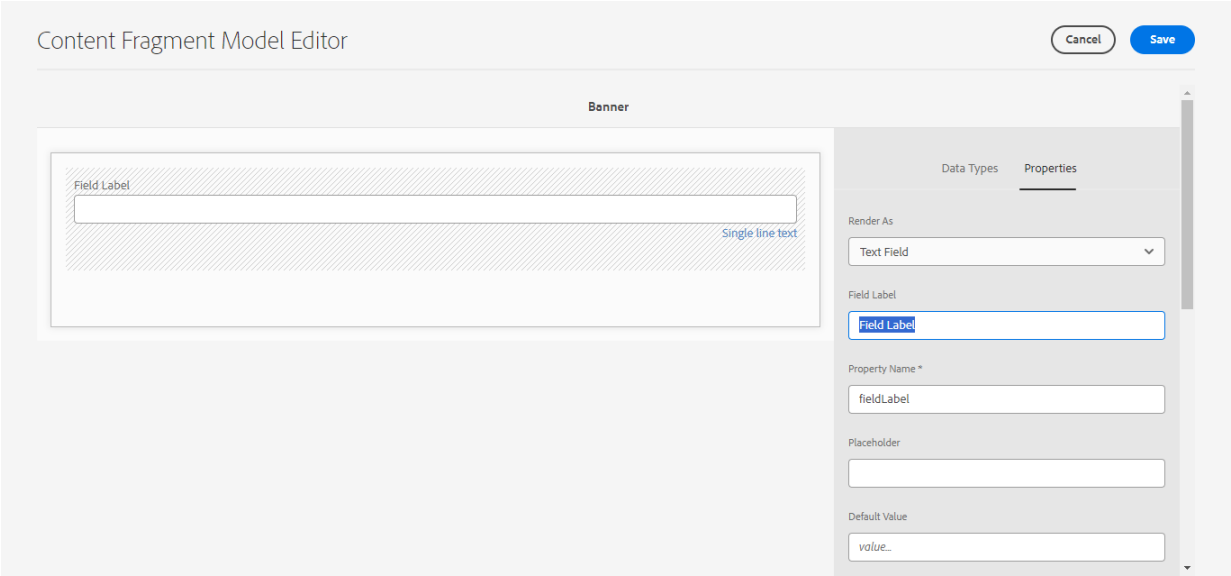
Dado do componente	Data Type	Função
Título	Single Line Text	*String* curta
Subtítulo	Multi Line Text	*String* grande
Texto do Botão	Single Line Text	*String* curta
Link do Botão	Single Line Text	*String* curta
Imagem do Banner	Content Reference	*Relator* para busca de conteúdos no DAM

Fonte: Autoria própria

2.3.2.1 Formulando um *Data Type*

Para usar um *Data Type* foi preciso definir suas propriedades conforme a Figura 8, o essencial foi escolher como foi a exibição de seu nome, este é o indicador que foi usado em consideração para cada atributo na hora do preenchimento, deveria ser um nome fácil de identificar e que condiz com o campo que foi apresentado na tela, a partir dele é preenchido o *Property Name*, este por padrão não pode conter espaço e nem caracteres especiais, dado que ele será o nome do atributo que foi chamado na lógica de programação das APIs para a entrega dos conteúdos no formato *Headless* API.

Figura 8: Criando um *data type*



Fonte: Autoria própria

Os demais campos para preenchimento de um *data type* são as possíveis personalizações para a exibição do campo na *model* ou padrões a serem preenchidos do campo na hora da exibição.

Assim como a estruturação de uma *model*, as propriedades de cada atributo também foram importantes de definirmos corretamente para a melhor eficiência, são com base nelas que posteriormente iremos chamar seus atributos para montar a API e depois consequentemente chamar o atributo numa camada de *front-end*, é possível verificar a relação das propriedades do *data type* com o dado do componente no Quadro 2.

Quadro 2: Quadro de comparação dos nomes das propriedades

Dado do componente	Find Label	Property Name
Título	Título	*titulo*
Subtítulo	Subtítulo	*subtitulo*
Texto do Botão	Texto do Botão	*textoDoBotao*
Link do Botão	Link do Botão	linkDoBotao
Imagem do Banner	Imagem do Banner	imagemDoBanner

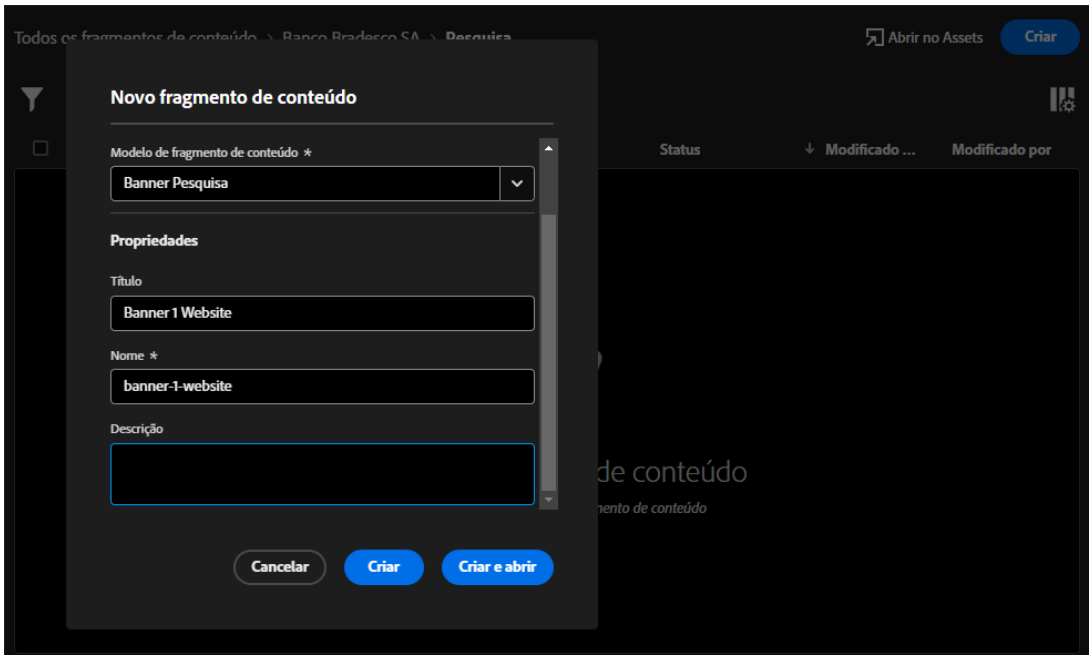
Fonte: Autoria própria

2.3.3 Criando um conteúdo através de uma model

Após estruturar a *model* e seus *data types* chegou a hora de finalmente criar um fragmento de conteúdo, ao iniciar o cadastro deste conteúdo é mostrada uma modal que exibe algumas informações a serem preenchidas, conforme é exibida na Figura 9, primeiramente precisamos definir o modelo de fragmento de conteúdo que irá ser usado, no caso será o intitulado Banner Pesquisa, este modelo está configurado com todos os *data types* necessários que suprem a necessidade do banner da página.

Com a definição do modelo a ser usado precisamos definir as propriedades como título, nome e descrição, estes atributos não necessariamente são para utilizar no componente, mas sim para identificação dentro da plataforma do AEM.

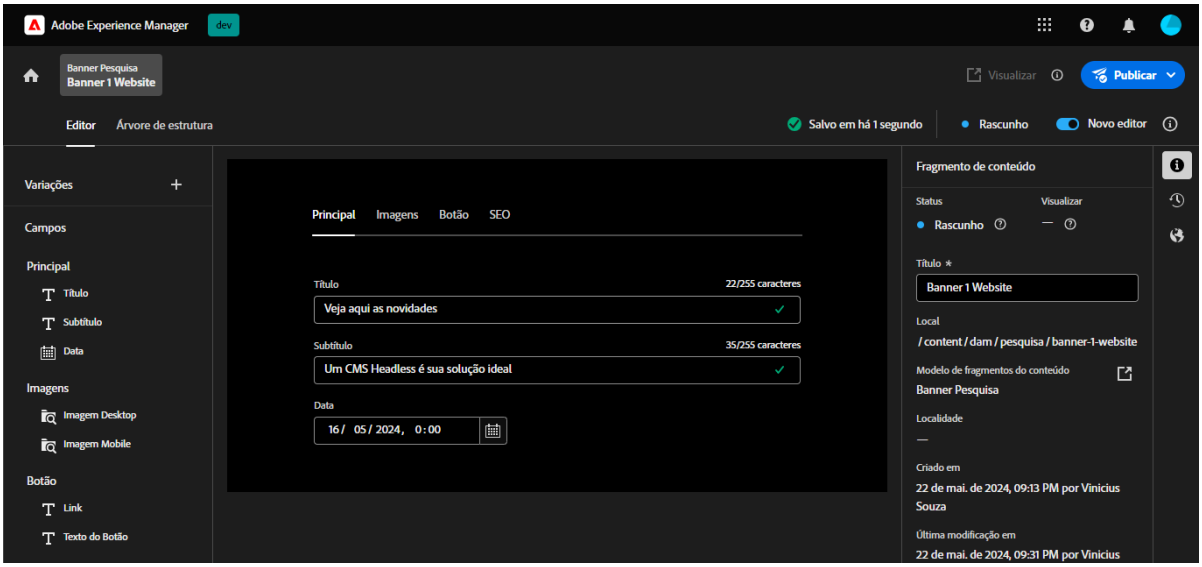
Figura 9: Exemplo de criação de um fragmento de conteúdo



Fonte: Autoria própria

O preenchimento do fragmento de conteúdo já com o modelo selecionado foi a última etapa da estruturação do componente, nele cadastramos o conteúdo a ser apresentado em uma estrutura parecida com um formulário, conforme mostrado na Figura 10, nesta tela é possível visualizar toda a estrutura que foi criada anteriormente na model e seus atributos, toda criação e alteração ficam registradas e são salvas, possibilitando usarmos em quaisquer lugares.

Figura 10: Exemplo de preenchimento de dados



Fonte: Autoria própria

Todos os conteúdos neste caso da plataforma AEM foram salvos em um JCR, que é uma especificação para o armazenamento, acesso e gerenciamento de conteúdo na aplicação Java, podemos visualizar suas alocações através do DAM já configurado nativamente na ferramenta, nele decidimos onde será salvo como numa estrutura de diretórios simples e acessá-los posteriormente.

2.4 Integração com o GraphiQL

Para o projeto foi utilizado o GraphiQL, pois essa é uma tecnologia já integrada com o AEM, Ele oferece uma maneira mais eficaz e adaptável de interagir com APIs e serve como uma alternativa às APIs REST convencionais.

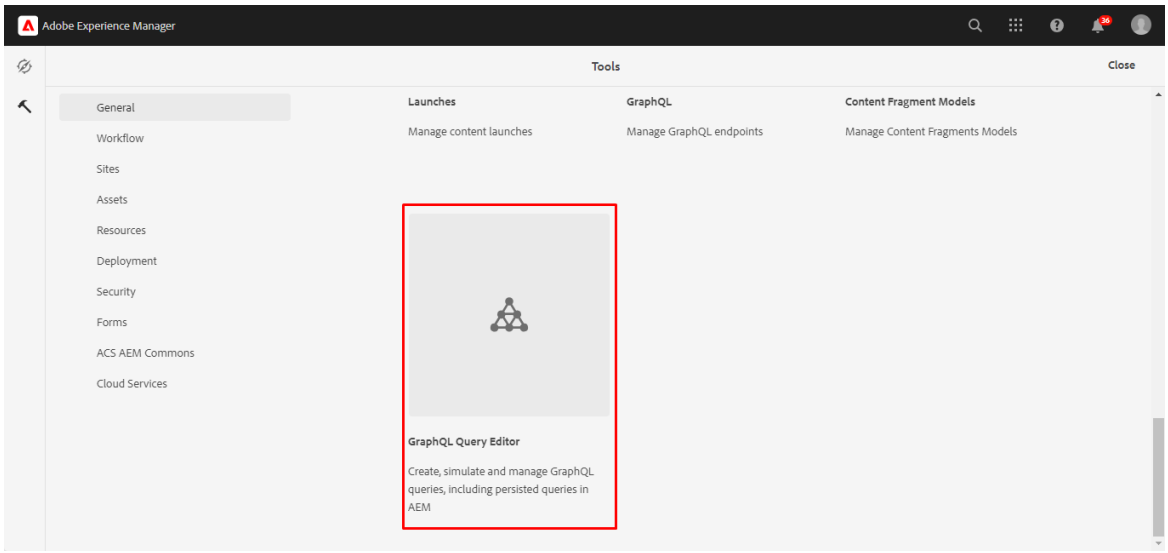
Como sendo uma tecnologia já integrada existe uma certa personalização com o intuito de facilitar a criação de queries, isto que ajuda muito o desenvolvedor a criação da API.

2.4.1 Criando uma Query no GraphiQL

Dentro de um conceito de *headless* CMS o objetivo é ter a separação entre as camadas de *back-end* e *front-end*, ou seja, a separação da camada de entrega dos dados com a camada de apresentação dos dados, com isso a parte essencial do projeto foi idealizar uma forma de extrair os conteúdos do CMS escolhido, isto foi possível através da ferramenta GraphiQL Query Editor do AEM.

A interface demonstrada na Figura 11, refere-se ao GraphiQL, esta é uma versão já integrada na ferramenta com personalização para a chamada de dados dos fragmentos de conteúdos gerados, essa integração facilitou o desenvolvimento a diante.

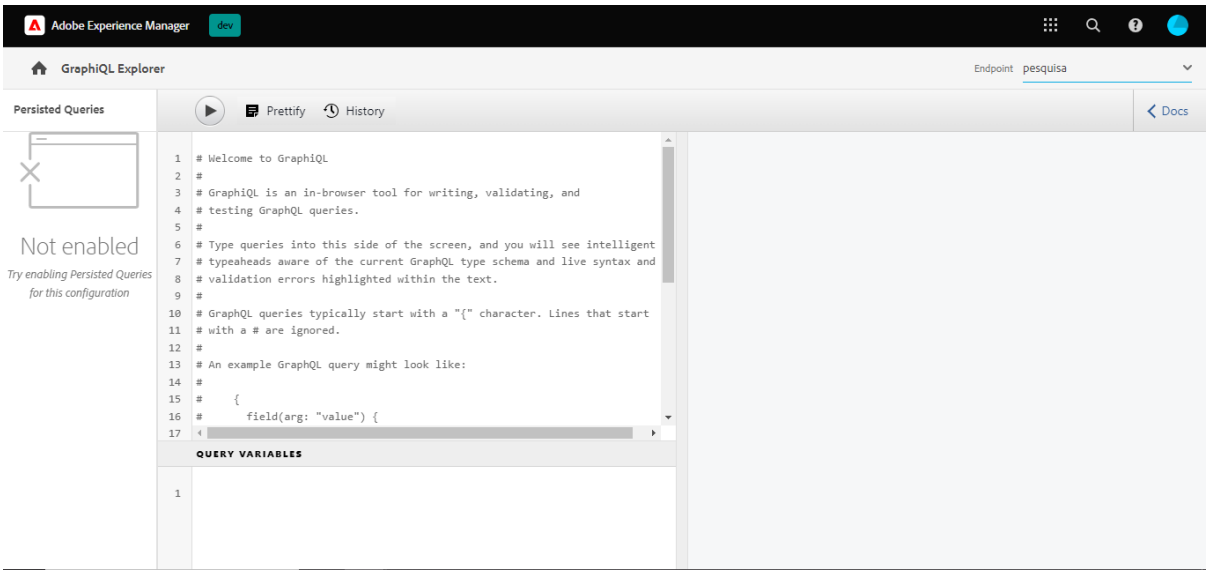
Figura 11: Tela de acesso as configurações do GraphiQL



Fonte: Autoria própria

Na interface GraphiQL *Explorer*, conforme demonstrado na Figura 12 no canto superior direito da tela foi necessário selecionar um *endpoint*, no caso do projeto foi criado um chamado pesquisa, essa escolha do *endpoint* é essencial pois faz parte da URL de requisição da API juntamente com o nome definido da *query*, após a escolha foi feita a lógica de programação no campo dedicado, está query foi basicamente a estrutura de dados a serem utilizadas para o projeto com todos os dados definidos e preenchidos através do fragmento de conteúdo.

Figura 12: Interface do GraphiQL no AEM

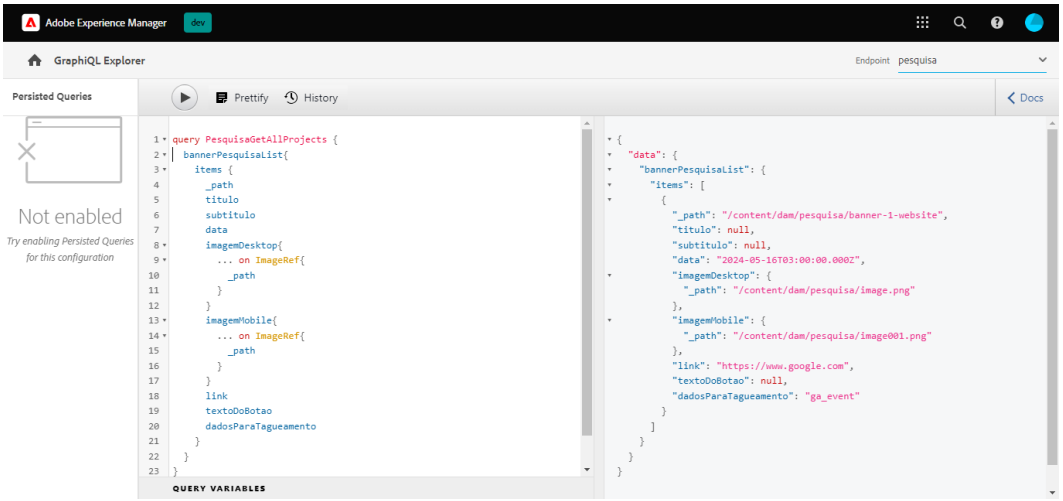


Fonte: Autoria própria

2.4.2 Entrega de dados de um Fragmento de Conteúdo via GraphiQL

A entrega dos dados foi feita pela query executada a partir de uma lista de conteúdos cadastrados do tipo Pesquisa Banner, conforme é possível visualizar na Figura 13, juntamente com os campos que foram cadastrados no modelo usado, é possível analisar que foram inseridos todos os campos possíveis para o cadastro de um conteúdo de banner, como o título, subtítulo, data, imagem para desktop e mobile, link de um botão, texto do botão e dados para *tags* de SEO caso existir.

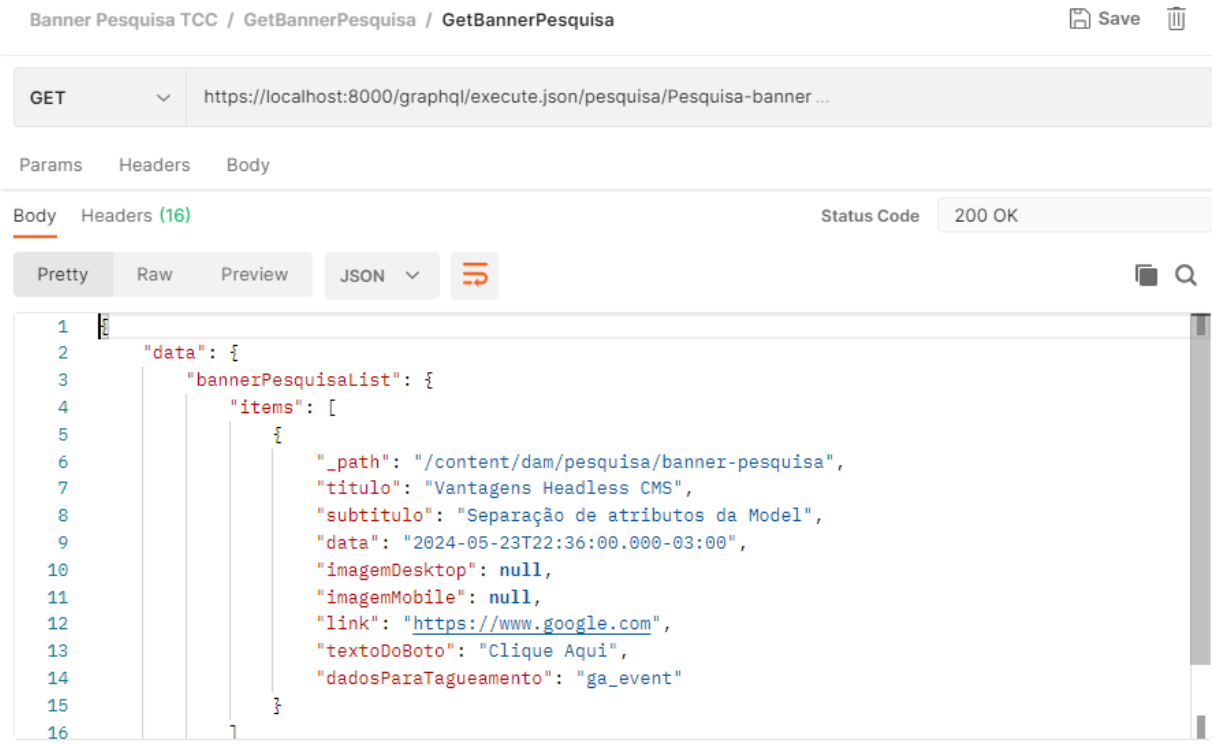
Figura 13: Criação da *query* para o endpoint



Fonte: Autoria própria

A consulta final da API é feita através de URLs, conforme demonstrado na Figura 14, com um *request* GET, é necessário passar algum tipo de autenticação que está configurada na ferramenta, com isso foi obtido o resultado dos dados entregues que podem serem usados a partir de quaisquer outros meios, sendo eles em um site na WEB ou um dispositivo de IOT, concluindo o objetivo de separar as camadas e entregando todos os conteúdos do CMS através de uma integração de API.

Figura 14: Consulta da API criada com os conteúdos



Fonte: Autoria própria

CAPÍTULO III

3. Apresentação e Análises de Dados

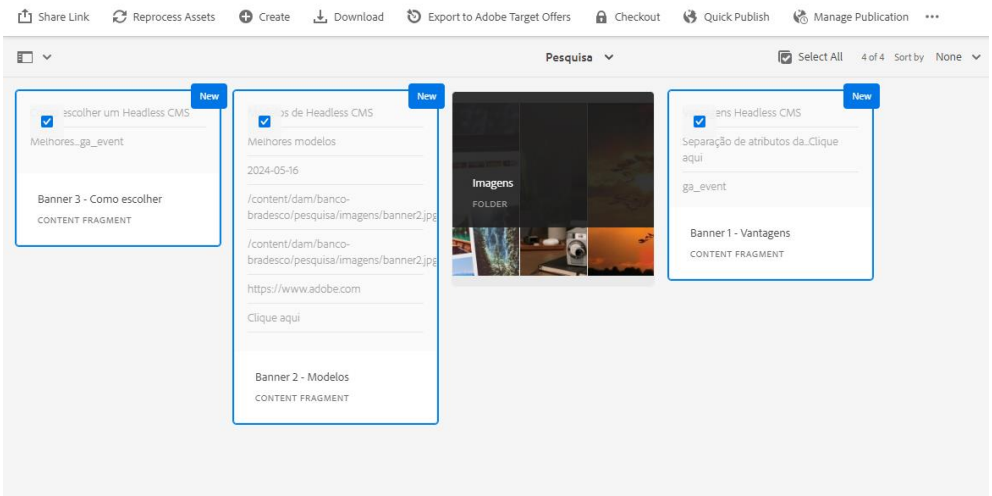
Neste tópico será apresentado os resultados e testes obtidos durante a produção deste trabalho.

3.1 Consulta de Dados

Como resultado de uma arquitetura *headless* foi preciso obter de alguma forma um desenvolvimento independente onde a camada de conteúdo é separada da camada de apresentação, promovendo agilidade no desenvolvimento.

Dado isso a resposta obtida ao desenvolvimento apresentado na metodologia, foi uma API apresentando dados para o banner de 3 itens de um site, conforme apresentado na Figura 15, foi possível observar o cadastro de três conteúdos na ferramenta AEM.

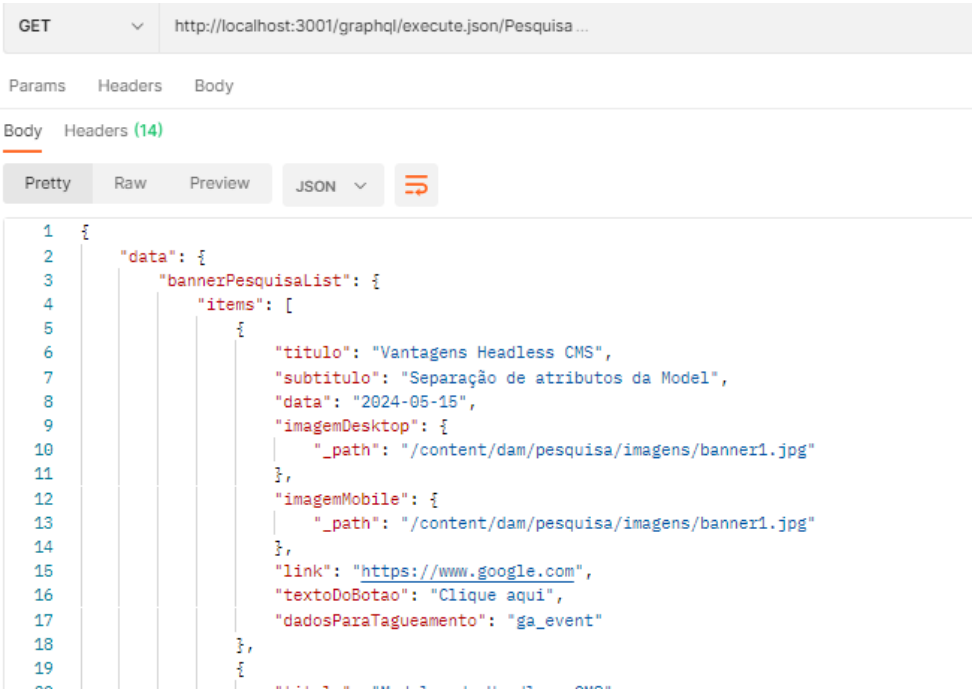
Figura 15: Fragmentos de Conteúdos no AEM



Fonte: Autoria própria

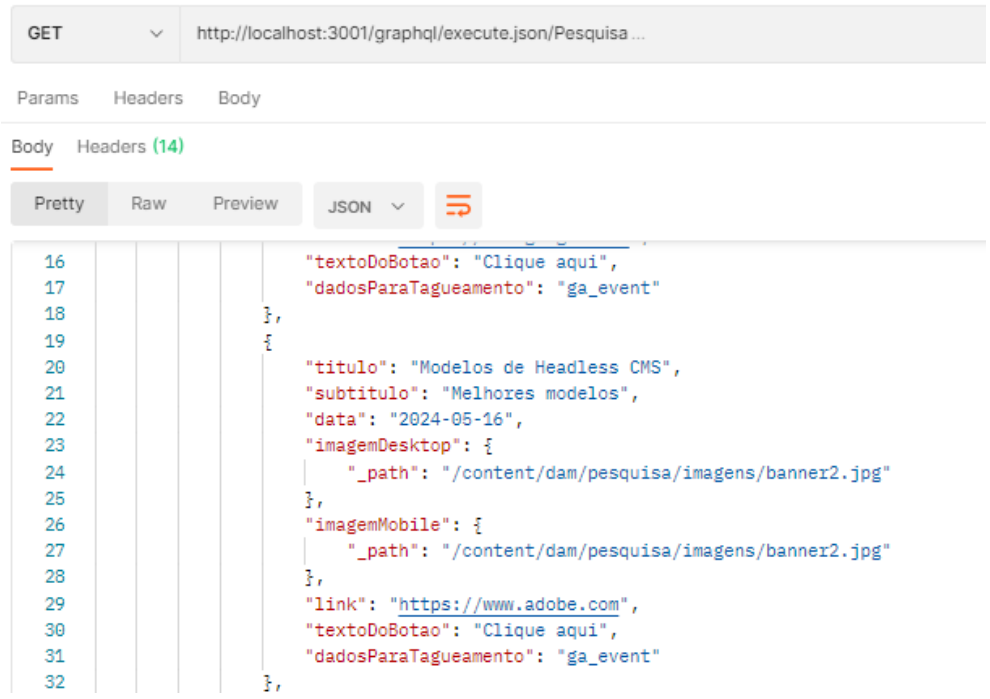
Através deste cadastro apresentado na Figura 15 foi consultado os mesmos através da query montada no GraphiQL, é fornecido um *endpoint* da API com 3 itens resultantes dos conteúdos criados a partir da *model* Banner Pesquisa, conforme apresentado na Figura 16 para o item 1, Figura 17 para o item 2 e a Figura 18 para o item 3.

Figura 16: Resultado do *request* para o item 1.



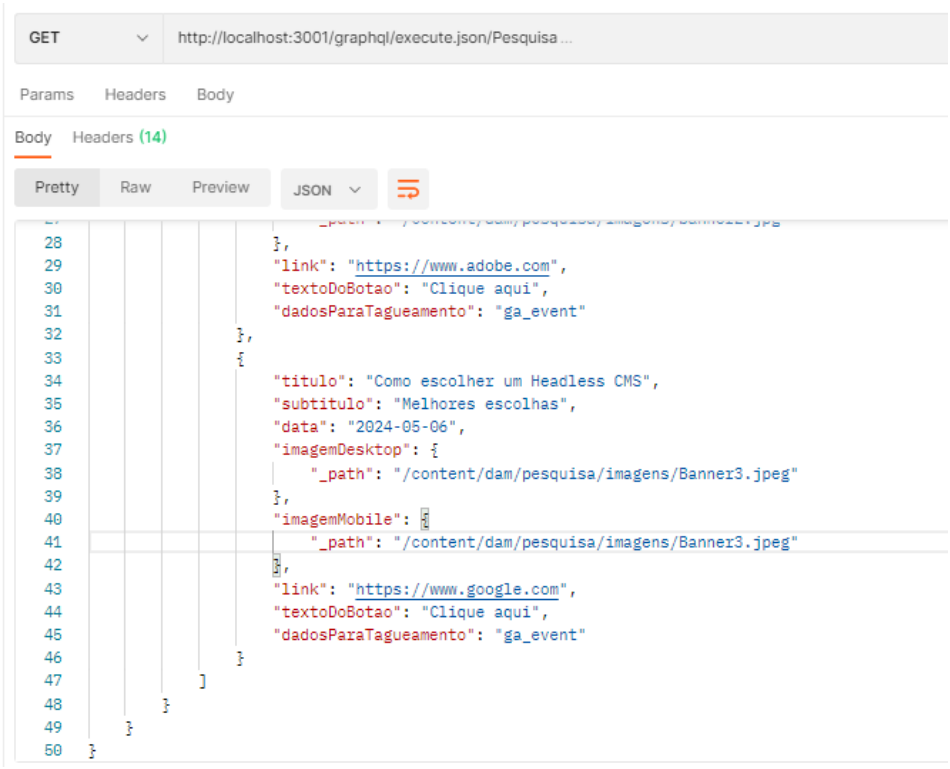
Fonte: Autoria própria

Figura 17: Resultado do *request* para o item 2.



Fonte: Autoria própria

Figura 18: Resultado do *request* para o item 3.

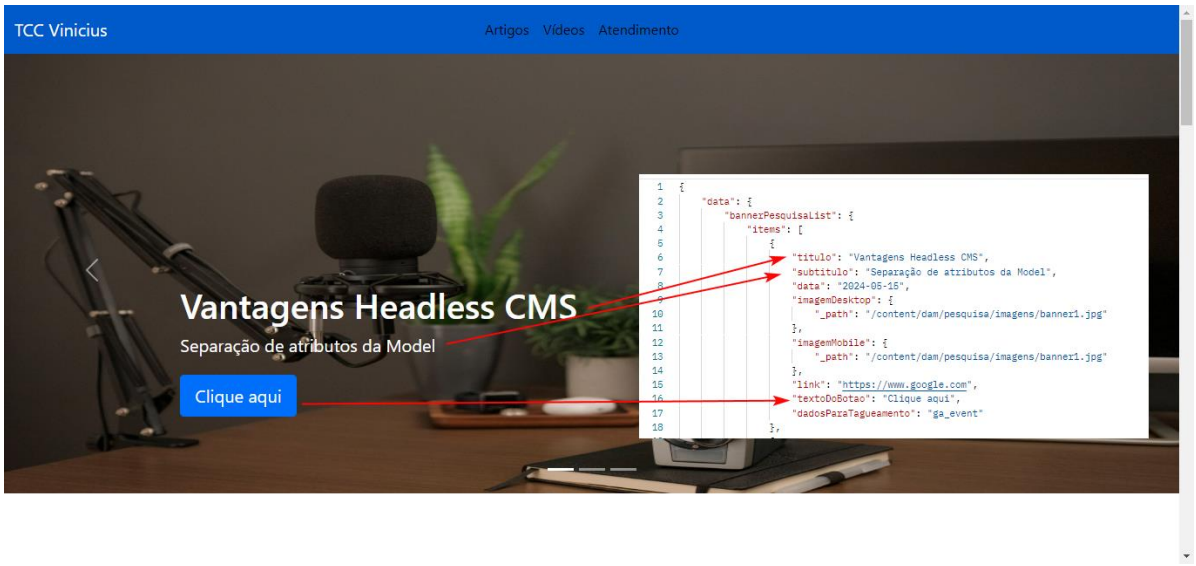


Fonte: Autoria própria

Com esses dados em mãos foi possível analisar que atendeu todas as necessidades para o projeto *headless*, podendo usar essa API para desenvolvimento em diversas plataformas, como em websites para computadores, dispositivos mobile, *smart* TVs ou dispositivos de IoT e totalmente desacopladas as camadas, além disto garantir uma experiência consistente e adaptada ao contexto do usuário, cumprindo o objetivo de um *headless* CMS, onde recebemos através de uma gestão centralizada no CMS os conteúdos, mas no final sendo distribuída de uma forma descentralizada, caso for a necessidade até mesmo em múltiplos canais.

No caso da pesquisa foram utilizados os *endpoints* acima identificados em um site desenvolvido exclusivamente para este trabalho, conforme apresentado na Figura 19, para que seja possível visualizar o consumo dos dados em um cenário real.

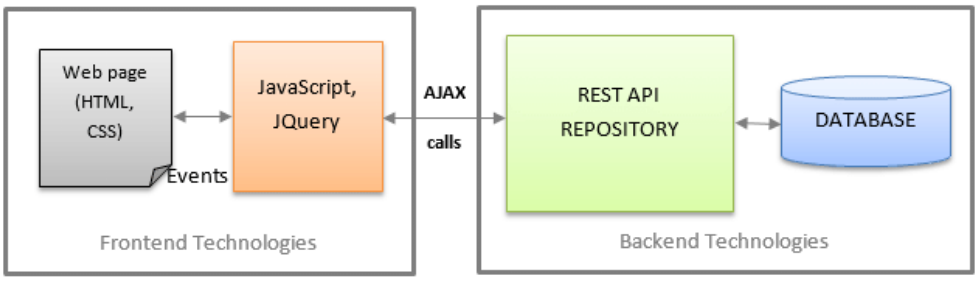
Figura 19: Relação entre API e página de exemplo.



Fonte: Autoria própria

A leitura de dados foi realizada através de um Javascript que faz a interação com a API via AJAX, conforme identificado na Figura 20, a lógica é que neste caso o JavaScript injeta os conteúdos recebidos através da API na *Web Page*, conseguindo assim neste cenário exibir a camada de apresentação.

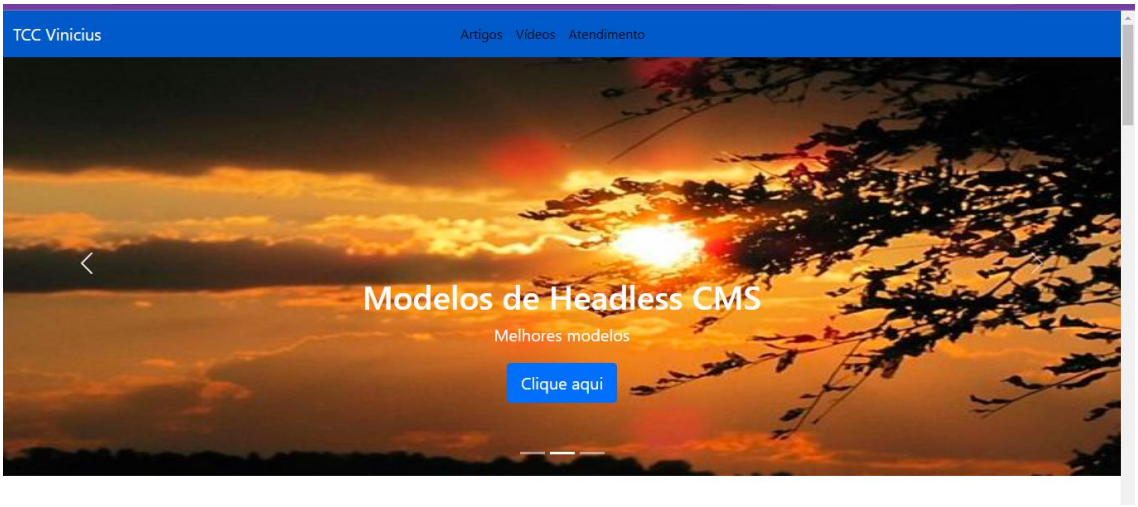
Figura 20: Diagrama de chamadas entre camadas



Fonte: ResearchGate, 2017.

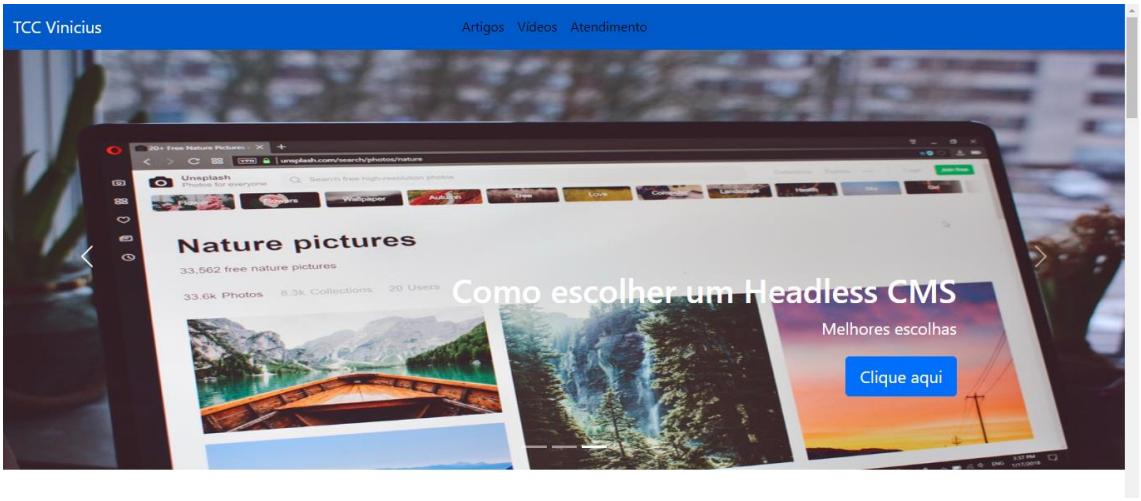
Seguindo com o objetivo do preenchimento dos dados foi possível trazer para a página web desenvolvida todos os conteúdos fornecidos pelo *endpoint* para completar o carrossel, conforme evidenciado nas Figuras 21 e 22.

Figura 21: Resultado da leitura do banner 2 da API



Fonte: Autoria própria

Figura 22: Resultado da leitura do banner 3 da API



Fonte: Autoria própria

Tanto a criação como manipulação de dados se mostraram muito simples utilizando a tecnologia tratada no trabalho, evidenciando que um projeto que utiliza este conceito de *headless* CMS é descomplicado e cumpre com sua proposta de flexibilidade, agilidade, melhor desempenho, facilidade de integração, produtividade e redução de custos, dado que sua manutenção também é simplificada ao serem separadas as camadas de dados e apresentação e possui uma escalabilidade no sentido que pode prever futuras alterações de aumento da demanda sem a necessidade de reestruturar a arquitetura do CMS.

3.2 Resultados da ferramenta AEM (Adobe Experience Management)

A ferramenta AEM foi capaz de atender as necessidades de estudo do projeto, onde nela foi possível criar o conteúdo a partir de *templates* pré-definidos e programados, que na ferramenta são chamados de *models*, a partir deles foram criados os fragmentos de conteúdos que serviram como base para os envios dos dados via API gerado pelo GraphiQl que é uma tecnologia já integrada na ferramenta.

Seu uso mostrou a facilidade e agilidade para a criação do projeto, mostrando a eficiência no caso de um *headless* CMS para o desenvolvimento de aplicações, comprovando no estudo do trabalho que esta tecnologia é uma boa alternativa para a escolha na hora das definições de quaisquer projetos.

3.3 Resultados do GraphiQL

A linguagem de consulta utilizada para execução de queries em APIs atendeu as necessidades da pesquisa, dado que se mostrou capaz de ser eficiente em dados, possui um desenvolvimento ágil e facilidades para o desenvolvedor.

É apenas uma possível tecnologia, pois dentro do contexto de um headless CMS podem ser usados outros métodos para a entrega de dados, mas cumpriu seu papel ainda mais estando integrado a um CMS, neste caso a facilidade de desenvolvimento se torna ainda maior, pois dado sua integração existem facilitadores dentro da ferramenta do AEM que ajudam na criação e execução de queries.

3.4 Resultados para a tecnologia *headless*

Com todos esses resultados, não é de admirar que o CMS *headless* esteja ganhando força. *Headless* é muito mais flexível, mas como muitos fatores a serem analisados, com flexibilidade e liberdade vêm responsabilidades maiores.

Headless não é para todos. Não existe uma “solução única”, pois as organizações têm necessidades diferentes. Antes de tomar uma decisão, primeiro é importante considerar como deseja que o conteúdo seja usado ou como ele é usado atualmente, analisar se é preciso que ele esteja disponível em diferentes formatos e para diferentes ocasiões. Para isso são necessários

recursos internos ou um bom provedor de serviços que garanta o gerenciamento adequado do modelo de conteúdo.

O desenvolvimento *front-end* deve ser considerado nesta solução headless. Como não existe uma opção pronta para uso, algumas empresas subestimam os recursos de desenvolvimento no front-end. O investimento em termos de tempo e dinheiro é uma consideração significativa. Colocar o sistema em funcionamento leva mais tempo. Desenvolver tudo do zero requer muitos recursos e alocação adequada de orçamento e tempo.

Considere a frequência com que o conteúdo muda, o que é necessário para o SEO e tome decisões sobre as necessidades do seu site. Como as visualizações de conteúdo nem sempre estão disponíveis, medidas também devem ser levadas em consideração sobre o resultado do front-end.

Normalmente, as organizações em crescimento têm como objetivo otimizar o tráfego orgânico por meio de SEO. Para isso, eles precisariam adaptar o conteúdo especificamente para o público-alvo, bem como qual canal esse público usará para consumir o conteúdo. Agora, isso significa que eles precisam trabalhar com design, *front-end* e modelo de implantação e garantir que cumpram a estratégia de SEO.

Para otimizar o site para SEO, é necessário haver flexibilidade nas diferentes configurações ou design de conteúdo para permitir testes rápidos no público-alvo. Isso significa que será necessário o envolvimento do marketing e dos desenvolvedores. Quanto mais fácil gerar conteúdo e testá-lo rapidamente, melhor e mais ágil a pessoa poderá personalizar suas experiências de usuário.

Com todas as tendências tecnológicas, deveria haver uma reflexão mais profunda sobre as funções e processos nas organizações. Um CMS *headless* muda as funções dos desenvolvedores, criadores de conteúdo e gerentes de marketing digital/comércio eletrônico. Os desenvolvedores precisam gerenciar o modelo de conteúdo e a arquitetura da camada de apresentação. Os criadores de conteúdo precisam mudar seu pensamento e forma de trabalhar, de centrado na página para centrado no conteúdo e multicanal. Quando o gerente de marketing digital ou comércio eletrônico está tomando decisões sobre segmentação comportamental e otimização contínua, ele precisa renunciar aos recursos prontos para uso do CMS anterior e escolher novas tecnologias para a otimização contínua da experiência.

Existem algumas armadilhas potenciais que se deve considerar, porque a utilização de um CMS nem sempre corresponde às expectativas. Aqui estão alguns desafios comuns:

• Complexidade avassaladora: isso pode parecer um paradoxo, considerando a afirmação anterior de que os CMSs deveriam simplificar as coisas. Mas cada caso é diferente,

um CMS pode não agradar qualquer outra pessoa. É uma boa ideia é buscar conferir, talvez até criar algum conteúdo, só para ter uma ideia se o CMS é adequado para a empresa.

• Custos ocultos: muitos CMSs afirmam que são gratuitos. Mas quase todos eles começam a cobrar quando se atinge um determinado limite de tráfego ou capacidade de armazenamento. É importante que o usuário faça sua pesquisa e certifique-se de considerar o projeto se aventurará no domínio do “serviço pago” mais cedo do que o esperado.

• Suporte e documentação limitados: Ao encontrar problemas ou precisar de assistência, suporte confiável e documentação abrangente são fundamentais. É essencial certificar de que o CMS escolhido tenha uma comunidade ativa, canais de suporte oficiais e extensa documentação ou guias do usuário. Essa rede de suporte pode ser inestimável na solução de problemas e no aprendizado de como aproveitar ao máximo os recursos do CMS.

CONCLUSÃO

Como vimos ao longo desse trabalho os sistemas de gerenciamento de conteúdo são parte essencial do ecossistema de programas que auxiliam as empresas e os desenvolvedores independentes manterem seus sites, portais e aplicativos em funcionamento e devidamente atualizados e preenchidos.

Porém também existem limitações e dificuldades conhecidas associadas ao uso dos CMSs mais tradicionais, para isso surgiu na indústria um novo modelo de gestão de conteúdos que vem sendo amplamente adotado por empresas que precisam da flexibilidade e rapidez no seu fluxo para disponibilização de conteúdo.

Esse novo modelo era a adoção da arquitetura *headless* dentro do contexto de gerenciamento de conteúdo, que ao separar as partes e suas responsabilidades trouxe uma inovação que foi muito bem recebida. Em um mundo onde o desacoplamento das aplicações está em alta, essa simples mudança trouxe um grande impacto em como vemos a gestão de conteúdo hoje.

Com isso temos algumas considerações a fazer, a arquitetura *headless* é de fato uma solução que vai se manter viável e suprir os requisitos cada vez maiores que temos no nesse mercado, ou será apenas uma tendência passageira dentre as várias que temos no campo da tecnologia e do desenvolvimento de software.

Isso só será respondido com o tempo e vasta utilização e popularização dessa tecnologia. O que podemos afirmar hoje é que ela vem desempenhando um papel importantíssimo em vários casos em que o dinamismo e a adaptabilidade são fatores determinantes para o sucesso.

REFERÊNCIAS

ADOBE**. Conceitos principais de AEM**. Publicado em 16 de abr. 2024. Disponível em: https://experienceleague.adobe.com/pt-br/docs/experience-manager-65/content/implementing/developing/introduction/the-basics Acesso em: 18 de mai. 2024.

ADOBE. **Documentação do Adobe Experience Manager**. Disponível em: https://business.adobe.com/br/products/experience-manager/sites/headless-cms.html Acesso em: 23 de fev. 2024.

ADOBE. **RECURSOS DO ADOBE EXPERIENCE MANAGER SITES: CMS headless**. Publicado em 2024. Disponível em: https://business.adobe.com/br/products/experience-manager/sites/headless-cms.html Acesso em: 20 de mai. de 2024

ANTONIO, Lennon Vinicius Moreno Del; CASTELLI, Nidia Mara Melchiades. **CMS PARA O MICROEMPREENDEDOR: DESENVOLVIMENTO DE UM E-COMERCE PARA MICROEMPRESAS UTILIZANDO FERRAMENTAS CMS GRATUITOS**. 2023. Artigo (Tecnólogo em Tecnologia em Gestão da Tecnologia da Informação) - Faculdade de Tecnologia de Mococa, Mococa, 2023.

BUTTI, Roberto**. Headless CMS Explained: What is it? Why does it matter?**. Storyblok. Disponível em: https://www.storyblok.com/tp/headless-cms-explained. Acesso em: 14 de nov. 2023.

CMS Headless: como funciona e quais as diferenças?. Sydle, 2023. Disponível em: https://www.sydle.com/br/blog/cms-headless-63bef5fdde93807eddd426ea/. Acesso em: 30 de nov. 2023.

DEMARCHI, Edvaldo Donizeti; BONFIM, Gabriel José de Deus; RIBEIRO, Luciana Jose Garcia. **Projeto de criação de um blog utilizando uma ferramenta CMS**. 2021. Artigo (Tecnólogo em Análises e Desenvolvimento de Sistemas) - Faculdade de Tecnologia de Mococa, Mococa, 2021.

FURQUIM, thiago. **O que é um CMS? | Content Management System**. canaltech. Agosto de 2022. Disponível em: https://canaltech.com.br/software/o-que-e-cms-content-management-system/. Acesso em: 18 de jan. 2024

GONÇALVES, Emanuel. **Headless CMS, o que são? onde vivem? o que comem?**. dev.to. 04 de jun. 2020. Disponível em: https://dev.to/codecasts/headless-cms-o-que-sao-onde-vivem-o-que-comem-3nm. Acesso em: 01 de jun. 2024

HANS, Aaron. **Going Headless: Use Cases And What It's Good For. Smashing Magazine**. Disponível em: https://www.smashingmagazine.com/2021/03/going-headless-use-cases/. Acesso em: 27 de nov. 2023.

Headless CMS na gestão da experiência dos seus clientes. Blog lumis. Disponível em: https://www.lumis.com.br/a-lumis/blog/cms-gestao-da-experiencia.htm. Acesso em: 24 de mar. 2024

ORACLE. **O que é um sistema de gerenciamento de conteúdo (CMS)?.** Oracle, 2023. Disponível em: https://www.oracle.com/br/content-management/what-is-cms/. Acesso em: 18 de nov. 2023.

SANTANA, Bruno. **O Que é Headless CMS e Qual a Diferença? Saiba Quando Usá-lo no Seu Site**. hostinger. Bahia, 21 de mai. 2024. Disponível em: https://www.hostinger.com.br/tutoriais/headless-cms. Acesso em: 07 de jun. 2024.

SILVA, Daniela Alexandra Coelho. **Estudo de Caso - Migração de Sistemas de Gestão de Conteúdo numa empresa Multinacional**. 2022. Monografia (Mestrado em Negócio Eletrônico) - Instituto Superior de Contabilidade e Administração do Porto Politécnico do Porto. 2022.