SHELTER

Blueprinting Report

MANIFESTO

Table of Contents

Introduction

Purpose of this Document

This document is a summary of the Blueprinting work undertaken by Manifesto for Shelter including an assessment of the current digital landscape and recommendations for future improvements. It summarises key findings both from the work undertaken by Manifesto and previously by Shelter and made available to us. Where appropriate, this document will outline the methodology used to come to a recommendation.

Document Scope

This document is intended to provide a clear summation of the findings and recommendations and act as a high-level blueprint for future change and the underlying structures and paradigms that should be put in place to achieve these. Where appropriate, specific technologies or providers will be recommended for implementation, although these should be treated as a recommendation based on the landscape at the time of the issue.

Related Documents

Name	Role	Version
User-Stories-Workshops-with-Capabilities-v1.0.xlsx	Shelter User Story document linked to capability model	1.0
shelter-cms-product-evaluation-scoresheets-v1.0.xlsx	CMS Product selection scoring	1.0
shelter-current-state-architecture-v.1.0.pdf	Current state architecture diagram	1.0
shelter-future-state-architecture-v.1.0.pdf	Future state architecture diagram	1.0
shelter-product-backlog-v2.0.xlsx	Initial product backlog for delivery	2.0
shelter-digital-engagement-capability-model-v2.0.xlsx	Digital Engagement Capability model tool	2.0

High-Level Solution Design

Project background

As part of Shelter's ongoing TIDE programme which aims to unify your digital estate, you asked us to make clear recommendations for the future of Shelter's digital publishing ecosystem – one that's flexible, meets your business needs to engage and support end users, and improves workflow and processes for content creation and governance.

Ultimately, this will enable Shelter to manage and strengthen its relationship with internal and external audiences – enabling deeper, more fulfilling engagement with Shelter that will benefit the organisation as a whole.

Our proposed solution is designed to future-proof your organisation and move away from your current fragmented, inflexible and developer-dependent architecture to one which allows you to scale up and devolve content creation, drive quality and efficiency, unify your users' interaction experiences – and implement data-driven feedback into your workflows.

Goals

As part of the production of this report, we spoke to a number of key stakeholders in your organisation, digested a range of internal documents and reviewed the preliminary work undertaken by Shelter in preparation for this project.

The following high-level needs were outlined to us:

- Allow for the devolution of digital publishing responsibilities across the business
- Allow for flexible content modelling
- Create transparent workflows that standardise the digital publishing process
- Integrate with a variety of tools and services
- Futureproof Shelter's Digital publishing capabilities

The scope of the work we have been tasked by Shelter to undertake can be summarised as:

"Recommend a best-in-class digital platform architecture that replaces and devolves your current content management solution, and empowers and future-proofs your organisation within a rapidly moving world."

In order to make clear recommendations to you about how best to meet these needs, we undertook the following activities:

- We described a current and future state architecture for Shelter's digital publishing platform
- We described a capability and maturity framework that provides the basis for mapping current state digital capabilities and identifying future ambition
- We performed a Web Content Management System Product Selection
- We detailed a series of epics that form a roadmap

Overview of Solution

In order to decide what architecture best answers the question about Shelter's future state, we believe there are three possible answers:

1. Rebuilding the current application in the same platform
2. Reimplementing the website using a different CMS platform
3. Implementing a Digital Engagement Platform by integrating best of breed products

We believe the answer is the third option – implementing a Digital Engagement Platform, which we describe as a suite of tools an organisation uses to engage with its users.

To understand why we think this is the answer, we understand that Shelter requires the following:

- You need a Digital Engagement Platform that you can use to build and deliver new products and services at pace
- You need a Digital Engagement Platform that helps to democratise tasks such as managing content changes into production
- You need a digital engagement platform that can evolve as you deliver new products for example a replacement for your CRM

The solution described in this document supports the following high-level goals:

- Be a flexible platform that supports regular, incremental change – in order, for example, to facilitate the introduction of new channels and products
- Minimise the requirement for Shelter DevOps team to manage servers – preferring Software as a Service (SaaS) products or products with lower operational overhead
- Support an operating model that describes Shelter managing subsequent development on the platform – preferring a model that utilises widely available skills and experience

We've described a modular platform that supports the integration of best of breed products and applications. Whilst this introduces some complexity in terms of the number of applications and the need to integrate them, it delivers an architecture that better supports evolution.

An evolutionary architecture is one that supports incremental, guided change as a first principle across multiple dimensions. In this case when we say dimensions we mean the lenses used to help understand a digital architecture such as technical, operational, data and security. Evolutionary architecture is primarily about optimising for evolvability across each of these dimensions.

The ability to evolve recognises the need for the architecture of your software to change over time. In order to deliver an underlying platform that provides stability over a longer period, we believe you should invest in describing an architecture that supports change as a first-order capability. What this means in practice is a platform that is capable of withstanding the change of the introduction of a new product or the withdrawal of a legacy one, without significant disruption.

A platform that provides stability over time whilst supporting the ability to evolve and change, will remain fit for purpose and relevant for longer.

Summary of Recommendations

The key components required to deliver an evolvable digital architecture for Shelter can be described as:

- Separating the concerns of content management, workflow and delivery
- Implementing a component-based architecture for front-end development
- Providing a unified view of content and data through the use of APIs

Separating the concerns of content management, workflow and delivery

By separating content management, workflow and delivery you enable greater flexibility in your architectural system. If offers you the ability to utilise specialist skills where they can add the most value and allows you the opportunity to evolve each of those products on their own roadmap. The skills that already exist within the Shelter digital team mean they are able to take ownership of each of the element as a separate concern.

Having looked at the elements of the architecture as discrete components, it is then important to evaluate how we would design Shelter's architecture to bring the capabilities those products and frameworks offer and tie them together to create a seamless system that utilises the power and strengths of those products as a holistic system.

First let's look at the separation of content management and delivery. This is typically described as decoupled content management.

Decoupled content management

Decoupled content management describes a separation between the application used to manage content and the application used to deliver it.

One of the primary benefits for Shelter is that a decoupled content management system allows the front-end developers to work independently from the constraints of the content management system, enabling faster iteration and empowering the model of Product teams that Shelter is moving towards. Furthermore, it brings the following additional benefits:

- It allows your repository system and publishing system to be on different architectures
- It is easier to scale a decoupled delivery tier as there are typically fewer moving parts and those parts themselves are often stateless
- Different architectures provide the ability to build teams with different skill sets to work on each of the applications.

Although decoupled content management includes systems where the delivery application is also a dynamic application (i.e. one where the data and presentation is pulled together

when a user requests it) we're recommending a model where your primary web presence is delivered through the use of static HTML.

Utilising an automated, static publishing model

A static publishing model focuses on producing static HTML pages from a catalogue of flexible user interface components and relevant data sources, as part of a build process. The produced static HTML pages are then subsequently deployed to hosting such as a Content Delivery Network. In addition to the benefits of a decoupled model, the use of a static publishing model brings the following benefits:

- It provides the ability to make your delivery layer more secure. By using a static publishing model, the size and range of potential attack vectors is significantly reduced
- Higher reliability. No CMS in the delivery tier means fewer moving parts. Static HTML files usually don't throw exceptions.

An automated static publishing model extends this model linking it to a workflow tool so that deployments of updated web pages are triggered automatically in response to workflow transitions.

The following principles describe this model a bit more practically:

- Webpages are created as static HTML during the deployment of the site
- Any dynamic programming during the request/response cycle is handled by JavaScript, running entirely on the client
- All server-side processes or database actions are abstracted into reusable APIs, accessed over HTTP with JavaScript
- The process of generating HTML pages is triggered by workflow.

In conclusion such a model is in general, easier to secure, performs better, and is easier to release and rollback - creating static versions of the site, you are able to more easily manage versions, allowing you to be able to quickly generate new versions and deploy to a testing, staging or production server for review with limited dependencies, or generate versions from a given point in time or commit.

Implementing a visual workflow tool to manage content publishing

In order to successfully devolve the responsibility of publishing changes to the Shelter website our architecture describes a separate visual workflow tool. The key benefit of separating the workflow tool from the content management system is that it provides the ability to manage the full end to end process of content development and generation, even before the content is uploaded or created in the CMS

Facilitating integration using APIs and an event-based model

Successfully integrating a range of products to deliver a coherent digital publishing platform requires the ability for the various products to communicate changes in state to each other.

This capability is particularly important in our context for the workflow tool so that it can communicate a business decision such as an approval to the content management and delivery applications. All of the proposed products here utilise a communication style called WebHooks. WebHooks provide a light-weight way for applications to communicate about changes in the application by calling a described HTTP endpoint in response to an event. Our solution proposes using application WebHooks in conjunction with a cloud platform provided pub/sub provider such as Azure Event Grid.

Implementing a component-based architecture for front-end development

Component-based architecture

A component-based front-end architecture in this context means that the user interface of our digital products is built from composable, reusable units. This method of organising our user interface elements brings the following benefits:

- Allows different parts of the UI to more easily, grow and change at different paces
- Makes it easier to ensure UX consistency across a portfolio of products
- Optimises the requirements and design process for new products by providing a baseline to work from

One of the artefacts typically created as part of implementing a component based front-end architecture is a UI/UX pattern library. This library presents the full suite of components grouped together in a way that makes it easy for designers and developers to understand how to build new products. By adding additional context to the UI pattern library such as documentation detailing how to use the current components as well as how to create new ones we begin to describe a broader organisational resource that is often referred to as a design system.

Design Systems

A design system is a collection of reusable components, guided by clear standards, that can be assembled together to build any number of applications. A design system will typically consist of the following:

- Visual design language
- UI/UX pattern library
- Documentation detailing standards and the usage of components

In order to quickly iterate with confidence, design teams need access to a single source of truth that allows for a scalable UI language and streamlined UX guidelines. With brand touch-points reaching over multiple channels and platforms, consistent user experience can be assisted by leveraging a central design language.

Providing a unified view of content and data through the use of APIs

Introducing a centralised API management tier that provides a facade over the APIs provided by your existing products allows you to do a number of things.

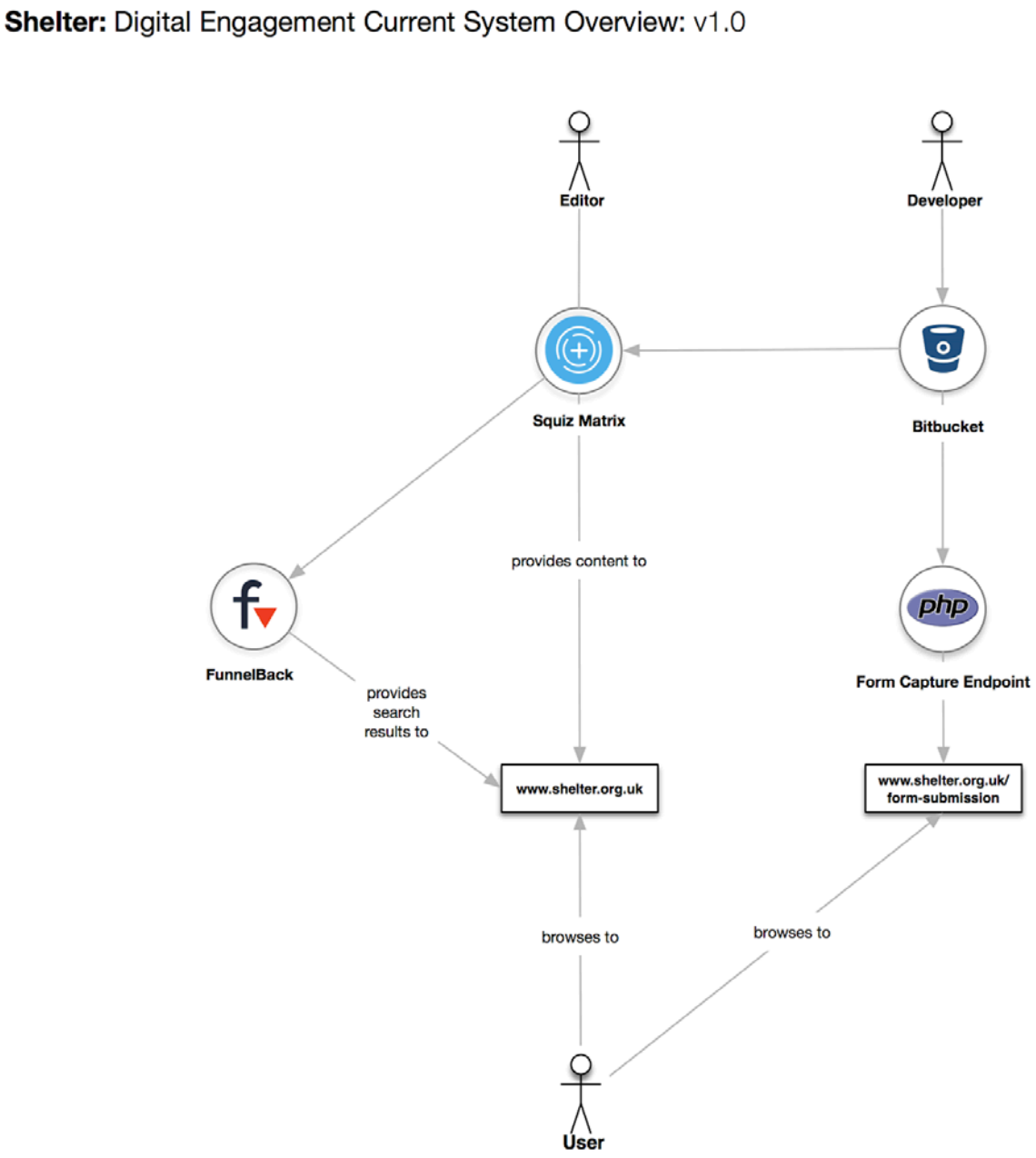
Firstly, it makes data available for delivery channels other than the Web - mobile is a good example here

Secondly it makes data from your systems available so that they can be integrated together, this provides opportunities for you to provide real value for your users

Thirdly by introducing a facade over existing APIs you provide a way to add and remove services from your platform without affecting the platforms underlying capability

Current State

In materials supplied to us and further outlined and expanded on during meetings with stakeholders, Shelter are constrained considerably by their current content management system, Squiz Matrix. The asset-based nature of the CMS means that there is a reliance on technical teams within Shelter to deliver content updates which causes considerable inefficiencies. As a result of this, there is a lack of a clear and transparent workflow that results in long lead times and an oblique understanding around the business of the state of a change.



Future State

The following diagram provides an overview of the proposed Digital Engagement System. It includes the following selected products:

- **Jira** - Jira provides Agile Project Management functionality including the ability to design workflows and track work visually using Scrum and KanBan-like boards
- **Contentful** - Contentful is a headless Content Management System
- **Gatsby** - Gatsby is a static site building tool that takes templated components and a datasource and returns static HTML pages
- **Netlify** - Netlify is a Content Delivery Network that hosts static HTML. In addition, it provides DNS integration that makes the provision of multiple environments simple and straightforward to implement
- **BitBucket** - BitBucket provides source code hosting and is required to manage the versioning of the static HTML delivered by Netlify
- **BitBucket Pipelines** - BitBucket pipelines provides a managed continuous delivery service that can be configured to trigger automated deployment and release processes

Further narrative describing the product choices is available later in this document. The following sections describe the functional role of each of the products within the solution architecture.

Jira manages editorial workflow

Jira is used to visualise the editorial workflow and is integrated with Contentful to provide an easy way for users to swap between the applications and to reflect changes that are made a content items state from one application to the other. For example, a content change could be approved by a user in Jira and that change would be reflected in Contentful. These integrations are implemented using WebHooks.

Contentful provides content management

Contentful is used to provide core content management functionality. This is distinct to Content Delivery in the sense that it doesn't provide any direct capability to serve, for example, web pages to users. Contentful provides the capability to model and manage content directly.

In our solution we've described how we take the content stored in Contentful and mix it together with templates stored in BitBucket to create static HTML. The application that performs this static HTML build process is Gatsby.

Gatsby provides static site building

Gatsby is a static site generator that uses React Components in combination with GraphQL data sources to produce static HTML pages. Gatsby is integrated with Contentful to provide the source of content for the website and with Netlify so that once content is approved and published through Jira/Contentful a static version of the site is created and pushed to Netlify

Netlify provides CDN hosting

Netlify is a Content Delivery Network that at its core provides static hosting for HTML. We're most interested in the additional value that the platform provides which include the following capabilities:

- Automated deployments - deployments to Netlify can be automated easily by linking them to branches in source control
- Automated management of https
- Integrated end user authentication - Netlify provides the functionality to integrate user management

BitBucket and Pipelines manage releases and deployments

BitBucket is a version control system that provides versioned storage for both Netlify and for engineers managing software development projects.

BitBucket Pipelines is an optional additional product, tightly integrated with BitBucket that provides a managed continuous delivery service. In practice and in this context, it is used to automate the deployment of applications and documentation supporting the main website. Deployment for the main website is handled by Netlify.

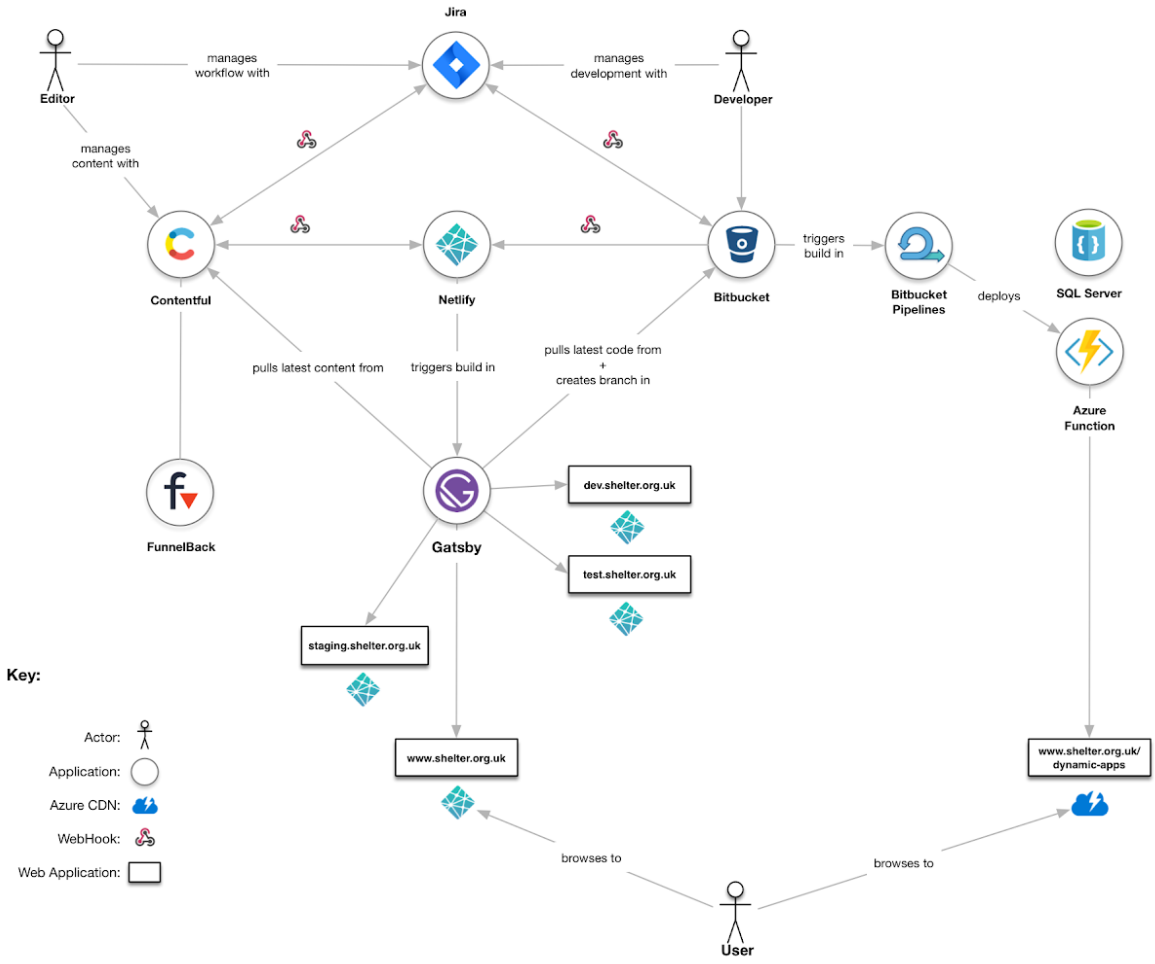
Jira manages development workflow

We've also included in the overview diagram some reference to the engineers involved in improving and shaping the platform. In this context Jira is used as the tool to manage development workflow.

Azure Functions provides Serverless Compute

Azure Functions is one of the serverless products offered as part of Microsoft's Azure Cloud platform. In our context we are using it to deliver the integration points between the main website and more dynamic functionality such as form capture endpoints.

Shelter: Digital Engagement Future System Overview: v1.0



Evaluation Methodology

Shelter provided a number of User Stories which clearly described organisational needs for the chosen CMS approach. We used these User Stories as the basis of our assessment by taking the following steps:

1. Analysing the set of Shelter user stories and validating any gaps with business teams by running a series of stakeholder workshops
2. Clustering the User Stories into Epics
3. Mapping Users Stories/ Epics to the core capability areas within the assessment framework
4. Reviewing, assessing and scoring CMS, content delivery and workflow applications

The 20 capability areas which form the framework for assessment are:

- Content delivery
- Preview
- Web Content Management
- Digital Assessment (DAM)/Enterprise Digital Assessment Management
- Social
- Personalisation
- Promotion Management
- Targeting
- Campaign Reporting
- Analytics (campaign)
- Workflow Engine
- Profile Management
- Ecommerce Services
- Multi Site Management
- Analytics
- Custom Business Analytics
- Directory Services

- Customer Data
- Web Services
- Content Management

Digital Engagement Capability and Maturity Model

This model includes six overarching categories (numbered 1 to 6) of core capabilities that are required for effective digital marketing solutions. It maps current capabilities, to define future state needs, and identifies the gaps that require solutions. This model describes foundational elements needed to support a comprehensive, extensible Digital Engagement Platform.

Categories

Content

The content category describes the creation and maintenance of content for optimum re-use across multiple channels.

Application

The application category describes services required for creating and presenting content to consumers.

Data

The data category describes the data layer constituting data access methods and platform repositories as well as corporate databases.

Integration

The integration category describes the integration layer responsible for interfacing with legacy and current systems.

Monitoring

The monitoring category describes the collection, analysis, and reporting on platform and site performance.

Governance and Operations

The governance and operations category describe governance and operational concerns.

Priority

The model serves as an aid by capturing the priority of each capability to help systematically determine in what order capabilities should be built and launched. The model allows for setting the priority for each capability within the platform using the following key.

Rating	Description
3	Must have capability
2	Should have capability
1	Nice to have capability
0	Out of scope

Assessment

In order to frame the range of capabilities we require to deliver the described solution, we've utilised a digital engagement capability model. It's worth outlining how we've used the model for the purposes of product selection and more specifically the parts of the model that we haven't used.

The capability model aims to provide a collection of foundational elements needed to support a comprehensive, extensible digital marketing platform. In our scenario we're using it as a way of describing simply the capabilities that we are most interested in for the purposes of selecting tools and products for the solution. Put simply we've used the tool as a way of identifying specific capabilities that form part of a broader ecosystem of digital engagement capabilities as way of providing greater context.

The model can also be used as a broader organisational tool to assess current levels of capability and future aspiration. For our purposes here however, this use of the tool is out of scope for the blueprinting project.

Product Selection - Web Content Management System

Goal - to provide a framework for choosing a Web Content Management System that best supports the needs of Shelter.

The content management system is the main system that administrative users use to create, manage and publish content across a range of delivery channels. The CMS will typically consist of two main components:

- A content management application (CMA) is the front-end user interface that allows a user to add, modify, and remove content from a website without the intervention of a developer.
- A content delivery application (CDA) the delivers the information to end users. Content Delivery as a product is discussed separately in this document.

Overview and recommendations

Whilst we describe the needs of the organisation fairly granularly by mapping requirements generated from user stories to capabilities described in the digital engagement capability model, to focus exclusively at this level of detail risks ignoring at least one key principle used to help us make both architectural and product decisions.

This principle is determined by identifying and selecting a future state operating model. Or to put it another way, once we determine who should manage the platform and who should be responsible for future development of the platform, we can use this decision to guide the choice of products that best meet our use cases.

There are three main operating models to choose from. These options can be characterised as:

1. Responsibility for ongoing development of the platform is managed by an internal team at Shelter using the knowledge and experience of the current team
2. Responsibility for ongoing development of the platform is managed by an internal team at Shelter created by recruiting people with the required specialist skills
3. Responsibility for ongoing development of the platform is managed by a development partner who provides a team with the required specialist skills

All three of these options are valid models, however Shelter have said to us that you want to ensure that you have an internal capability that can manage future development of the digital publishing platform and that preference whilst only explicitly removing one option, does impact the choice between options 1 and 2.

Our experience of working with organisations similar to Shelter has lead us to believe that building a capable and stable internal team of engineers with specialist product knowledge is becoming increasingly difficult to do successfully.

Competition for engineers with deep knowledge of platforms such as Drupal & BloomReach continues to intensify and a significant proportion of Manifesto's customers are organisations who have devolved responsibility for their content management platforms to an agency.

With this in mind our recommendation is for Shelter to invest in a platform that can be enhanced and developed by an internal team using currently available skills. This is described as option 1.

Selection recommendation - Contentful

In our opinion, **Contentful** provides the best fit for;

- the planned operating model at Shelter
- the new architecture that is being proposed (in the former sections of this document)
- and functionality such as the lightweight dynamic image resizing and tools for managing the import and export of content give Contentful the edge in comparison.

Of the five products we looked at as part of the detailed analysis and comparison, we felt there were three in particular that stood out as potential candidates, two that are more specifically API focussed content management systems that would fit in excellently, and a more classic CMS that also has the capability to fit within the proposed architecture.

The ones that stand out are Contentful and Prismic as API first options, and WordPress as a more 'classic' CMS option. The API first options provide the best fit in terms of restructuring how content is produced and managed with regards to a decoupled content delivery tier, allowing for a lot of efficient reuse of content across multiple channels. Whilst Prismic's concept of 'Slices' is an interesting approach to grouping and reusing common modules, its lack of a formal write API potentially limits the future proof nature of the selection.

A full breakdown of scoring can be found in the document; *shelter-cms-product-evaluation-scoresheets-v1.0.xlsx*

Comparison Highlights

Contentful

+ Excellent API features

+ Good support for content model migrations

+ Good open source tools for managing content import / export

+ Good lightweight dynamic image resizing / cropping functionality

- Lack of reporting functionality (would need to build)

Prismic

+ Slices functionality provides a good way of grouping together common module infrastructure

+ Releases concept provides good control over point in time publishing for more dynamic sites

- Lack of validation for content entry fields

- Undercooked import / export interface

- Would probably require additional DAM or software to manage renditions

- Lacks a formal write API which limits its use from other applications

WordPress

+ Easy to use, well understood editorial interface

+ Good content API

+ Provides broader options than simply API (can render and deliver full sites)

+ Easily extendable

- Relies heavily on plugins such as ACF to make it viable as a Content Management Platform

Methodology

- Determine a long list of options
- Determine a shortlist of five prospective products
- Identify based on user needs a range of grouped assessment criteria
- Apply a weighting to describe our priorities
- Score each product against the criteria described

Market research

In order to determine a list of prospective Content Management Systems to pick from, we performed some initial market research.

Hosting, support and maintenance costs

Both Contentful and Prismic are SaaS platforms, so hosting is tied in alongside licence costs. These products both offer various tiers of service, with options around size of instance, numbers of users permitted and tiers of service and support.

All figures are based on available information from the platform providers, either via publicly available figures or from direct conversations with the vendors and should be treated as indicative pending further negotiations with the vendors directly

Contentful Platform Costs

Contentful separate costs for service and costs for the environments (referred to as Spaces in Contentful parlance). The service tiers come with additional features (such as SSO, dedicated infrastructure) as well as being bundled with service features such as response SLA's and access to account management and solution architects.

Tier	**Indicative Costs**	**Notes**
Enterprise Grade Space, Silver tier platform features	£35,000 - £45,000/ year*	Includes SLA's, access to account management and solution architects
Large Space, Basic tier platform features	£8049/ year**	• 4 roles • 48 Content Types • 50,000 records • 10 free users

* Price converted from Euros

** Price converted from Dollar's

** Note additional users are $15/ user/ month.

Contentful offer a pro-bono account for not for profit organisations, equivalent to a large space if Shelter are prepared to participate in co-marketing.

Prismic Platform Costs

Prismic have a variety of options for tiers of service, with numbers of users, support functionality and SLA responsibility.

Tier	Indicative Costs	Notes
Professional Tier, Platinum	£4,500/ year*	• 99.9% Uptime SLA • Priority Support • Unlimited users
Professional Tier, Unlimited	£2,750/ year*	• 99.9% Uptime SLA • Basic Support • Unlimited users

* Price converted from Dollar's. Includes 20% annual payment discount

WordPress Platform Costs

WordPress is free to license as it is open source. There are numerous options for hosting a WordPress site. For ongoing hosting, we recommend WP Engine who are a WordPress specialist managed hosting provider. We've been working together for a number of years and are one of their preferred partners, with a number of sites maintained with them including the UNICEF UK multisite.

Tier	Indicative Costs	Notes
Managed Hosting on WPEngine	~£500 - £1000/ month	• Cost is dependent on support/ SLA requirements

Longlist

The following list comes from this report - https://www.realstorygroup.com/Reports/CMS/ and details 34 separate Web CMS products and platforms.

- Acquia - Drupal
- Adobe - AEM Sites
- Atex - Polopoly
- BloomReach - Digital Experience Platform
- CCI Europe - Escenic
- Contentful - Contentful
- CoreMedia - Digital Experience Platform
- Crafter Software - Crafter CMS
- CrownPeak Technology - CrownPeak CMS
- DotNetNuke - DotNetNuke
- e-Spirit - FirstSpirit
- Episerver - Episerver
- eZ Systems - eZ Enterprise
- GX Software - XperienCentral
- IBM - Web Content Manager
- Ingeniux - Ingeniux Content Management System
- Joomla Project - Joomla!
- Kentico - Kentico CMS
- Magnolia - Magnolia CMS
- Microsoft - SharePoint 2016 WCM
- MODX - Revolution
- OmniUpdate - OU Campus
- OpenText - TeamSite
- OpenText - OpenText Web Experience Management
- Oracle - WebCenter Sites

- Perfect Sense Digital - Brightspot
- Plone - Plone CMS
- Progress Software - Sitefinity CMS
- SDL - SDL Web
- Sitecore - Experience Platform
- TERMINALFOUR - Site Manager
- TYPO3 - TYPO3
- Umbraco - Umbraco CMS
- WordPress - WordPress

In addition, we've added the following API first CMS solutions to the list

- Prismic
- IBM Watson Content Hub
- Netlify CMS
- Directus
- ContentStack

From this long list we filtered down to a list of five options based on the following criteria:

- Excluding legacy platforms
- Costs for implementation and licensing are within the range
- Excluding products that were a poor fit for our scenario, this included products that:
    - Focused on specific use cases such as multilingual
    - Provided too little pertinent functionality
- Prefer API first CMS solutions

Exclusions

Legacy

- IBM: Web Content Manager
- Microsoft: SharePoint Server 2016
- OpenText: TeamSite
- OpenText: Web Experience Management

- Oracle: WebCenter Sites

Poor fit

- **Atex - Polopoly** (caters primarily for media and newspaper market)
- **CCI Europe** - **Escenic** (caters primarily for large scale media organisations - more a publishing tools than an engagement platform)
- **CoreMedia - Digital Experience Platform** (caters primarily for large scale organisations with complex media focussed needs, pricing starts at $28,000 per core with an additional per-user cost)
- **Crafter Software - Crafter CMS** (based on Alfresco would suit organisation with a pre-existing commitment to that technology)
- **CrownPeak Technology - CrownPeak CMS** (fully hosted CMS platform, with licensing costs starting at $72k per year. Hosted exclusively on AWS)
- **DotNetNuke - DotNetNuke** (Open source .NET product, poor content production functionality including metadata support and internal search)
- **e-Spirit - FirstSpirit** (focussed on integration with enterprise portals such as SAP Portal)
- **eZ Systems - eZ Enterprise** (PHP based, recently re-written, licensing costs range from $36,500 - $66,000 per year with cloud costs additional)
- **GX Software - XperienCentral** (Java based platform with costs in the range of $60k - $150k per year)
- **Ingeniux - Ingeniux Content Management System** (.net based)
- **Joomla Project - Joomla!** (Open source PHP product, lack of support for multisite scenarios)
- **Magnolia - Magnolia CMS** (Focused on integrations)
- **MODX - Revolution** (Simpler open source PHP product, focusses on small webshops, interactive agencies and nonprofits with internal dev resource.)
- **OmniUpdate - OU Campus** (caters primarily to higher education institutions)
- **Perfect Sense Digital - Brightspot** (caters to larger media organisations, implementation costs ranging from $500K - $1.5M)
- **Plone - Plone CMS** (Python based open source platform, requires development to deliver anything beyond simple informational sites)
- **SDL - SDL Web** (caters primarily for large multinational enterprise customers)
- **TERMINALFOUR - Site Manager** (focussed on higher education customers, licensing model charges per content item.)

- **TYPO3 - TYPO3** (PHP based platform focussed on providing support for community features such as polls, forums and blogs. Complex user interface)
- **Adobe - AEM Sites** (provides an excellent digital marketing platform / ecosystem, is expensive to purchase, implement and run, would require internal dev resource to be cost effective to build on, external resource is expensive, implementations are large scale development projects)
- **Sitecore - Experience Platform** (similar to AEM, is an excellent platform, but is expensive to purchase and implement, because the development model doesn't always follow microsoft standards, you would need sitecore expertise - typical deal size is $350,000+)
- **Progress Software - Sitefinity CMS** (.NET standards based, widget-based page building provides flexible editorial interface)
- **Kentico - Kentico CMS** (.NET standards based, provides a good range of functionality, pricing is per domain / per server - $999 per month SAAS hosted)
- **Episerver - Episerver** (.NET standards based, provides good digital marketing tools, cloud based hosting is costed by pages views)
- **Umbraco - Umbraco CMS** (doesn't provide a native, granular content modelling paradigm)

Selections

Candidates

After excluding legacy and poor fit selections, we were left with the following candidates for further evaluation:

- **WPEngine – WordPress**
- **Acquia - Drupal**
- **BloomReach - Digital Experience Platform**
- **Contentful – Contentful**
- **Directus**
- **ContentStack**
- **IBM Watson Content Hub**
- **Prismic**
- **Netlify CMS**

After additional consideration we further discounted the following:

- **Acquia - Drupal**
    - Drupal is an excellent fit for community and digital marketing scenarios, with a large community supporting the creation of connectors, plugins and integrations with a variety of sources

- Whilst Drupal does have good API support, it does not fit the recommended architectural approach – it is a complete CMS and content delivery product in one, which whilst offering advantages in some aspects, such as potentially a simpler user experience, it prevents as clear a vision of the future of the platform and the ability to evolve and adapt over time.
- Drupal also requires specialist development knowledge of the CMS, which doesn't make it a good choice for the operating model recommendation as described above.
- Implementation and hosting costs for a Drupal build are also higher than some of the other items on the candidate list.

- **BloomReach - Digital Experience Platform**
  - BloomReach is best suited to scenarios that require a fine-grained content model and good integration capabilities. It is reasonably straightforward for content editors, but its UI can be limited, so it would require more significant customisation to support power users
  - BloomReach is able to function as a headless CMS, which would fit the architectural recommendation, however it's approach is still to tie the CMS to the delivery tier in a closer way, which would not suit every use case for Shelter, particularly in regard to the digital roadmap.
  - BloomReach also requires specialist development knowledge of the CMS, which doesn't make it a good choice for the operating model recommendation as described above.

- **Directus**
  - Directus is an open source Headless CMS with an excellent user interface and top-quality documentation. However, it lacks exposing certain features through the UI, which would require development intervention – the very circumstance Shelter is looking to move away from which is why it was ultimately discounted from consideration.

- **ContentStack**
  - ContentStack is also a Headless CMS, offering the same benefits as outlined above. It is easy to use and understand for content editors and offers a variety of options for developers, although it can lack significant customisation options out of the box.
  - The price of a ContentStack hosting and license is higher than the others in consideration
  - Overall, the out of the box experience for ContentStack is quite limited to some of the others in consideration, which is why it was discounted from the shortlist.

Shortlist

We are then left with the following 5 items on the shortlist for detailed evaluation and scoring.

- **Contentful**
- **Prismic**
- **IBM Watson Content Hub**
- **WPEngine - WordPress**
- **Netlify CMS**

Scoring

Scoring scale

The following describes the scale used to score the capabilities described for each product

Ratings	Description
4	Available out of the box, but some configuration required.
3	Available out of the box, but extensive configuration required.
2	Minimal custom development required to support the capability.
1	Extensive custom development required to support the capability.
0	No capability - completely custom solution.

Web Content Management Selection Scores

	Weighting	Contentful	Prismic	IBM Watson	Netlify CMS	WordPress
Content Authoring	20%	2.46	2.46	2.31	2.23	2.54
Content Delivery	20%	2.80	2.80	2.60	2.40	2.20
Integration APIs and Extension	20%	3.33	2.67	2.67	1.67	2.67
Digital Asset Management	10%	3.00	2.00	2.00	1.50	2.50
Cost of Ownership	10%	2.20	2.40	2.00	2.80	3.20
Platform Management Complexity	5%	3.00	3.00	3.00	2.00	3.00
Security	15%	2.50	2.50	2.50	2.50	2.00
Totals		**2.764**	**2.551**	**2.440**	**2.164**	**2.501**

Recommendation - Content Delivery Stack

Component Based Framework

Framework Evaluation

We evaluated three frameworks for the Front-End delivery - React.js, Vue.js and Angular.

Name	Notes
React.js	+ Strong component-based framework - State management is complex
Vue.js	+ Strong component-based framework + Use of HTML more formally as templating - Community is smaller
Angular	- Batteries included approach requires a full commitment to the framework

Recommendation - React.js

React and Vue are really quite similar from a functional perspective. Following conversations with the development team at Shelter we recommend React given the broader range of experience across the teams at Manifesto and Shelter.

Costs

As open source frameworks, there are no costs associated with these selections.

Content Delivery

Stacks Evaluated

Some parts of this product are typically supported by a content management system and a content delivery application is referenced in the Content Management product above. However in our context there are other technologies involved in the delivery of content to end users including caching frameworks such as Content Delivery Networks (CDNs).

Name	Notes
Netlify CDN	• Static Site Generation using Gatsby.js, React and Contentful - https://www.gatsbyjs.org/blog/2017-12-06-gatsby-plus-contentful-plus-netlify/ • Need to discuss Netlify functions (by default AWS) • Provides a good branch-based preview model
Azure (CDN, Static Hosting, WebApp Hosting)	• Good fit with Orgs strategic choice of Azure (more building than just Netlify) • Doesn't need to be just static site, could include dynamic node / React / Vue
AWS (CDN, Static Hosting, Web App Hosting)	• Less good fit with Org, better fit with Netlify • Doesn't need to be just static site, could include dynamic node / React / Vue

Recommendation - Netlify

Netlify is one of a series of modern, more managed CDNs in that as well as providing static hosting it includes additional functionality to support use cases such as authentication and integration with AWS functions as a service

Costs

Name	Indicative Costs	Notes
Netlify	£330/ month*	Includes Professional Tier of • Functions

		• Identity • Forms • Teams
Azure	~£100/ month	Azure hosting is considerably more self service, consisting of a range of components, and costs are based on usage/ number of users. This indicative price would need further refinement if this hosting option was selected.
AWS	~£100/ month	AWS hosting is considerably more self service, consisting of a range of components, and costs are based on usage/ number of users. This indicative price would need further refinement if this hosting option was selected.

* Price converted from Dollar's

Design System

Describing a framework for how your users interact with all of your services, not just the pages that the content managed website provides a point of separation between the user experience that you provide to your users across your digital engagement platform and it's various implementations. In practice this means the creative design, user experience and front end build (HTML, CSS and Javascript) takes place as a separated project or phase from the main CMS build. By designing and building this part of the work separately we recognise the ability to progress and change our user experience and front end build with impacting the CMS implementation to the same extent. In summary a separate UX pattern library:

- Supports the ability to keep design and user experience consistent across multiple products
- Provides a style guide to provide clear guidance to collaborators and delivers practical example driven guidance for implementation
- Supports the ability to change and update the frontend user experience at a difference pace to the backend CMS implementation

Recommendation - Atomic Design

Atomic design - http://bradfrost.com/blog/post/atomic-web-design/ is a methodology for creating design systems. There are five distinct levels in atomic design:

- Atoms
- Molecules

- Organisms
- Templates
- Pages

Atoms

Atoms are the basic building blocks of matter. Applied to web interfaces, atoms are our HTML tags, such as a form label, an input or a button.

Molecules

Molecules are groups of atoms bonded together and are the smallest fundamental units of a compound. These molecules take on their own properties and serve as the backbone of our design systems. An example of a molecule might be a site search control including a label, input and search button.

Organisms

Molecules give us some building blocks to work with, and we can now combine them together to form organisms. Organisms are groups of molecules joined together to form a relatively complex, distinct section of an interface. An example of an organism might be the Header bar of a website, including things such as site search molecule, a navigation bar and a site logo.

Templates

At the template stage, we break our chemistry analogy to get into language that makes more sense to our clients and our final output. Templates consist mostly of groups of organisms stitched together to form a generalised view of a page type. An example here might be a landing page template.

Pages

Pages are specific instances of templates. Here, placeholder content is replaced with real representative content to give an accurate depiction of what a user will ultimately see.

Recommendation - Workflow and Delivery

Products Reviewed

Product	Notes

Jira	+ Good WebHook support + Visual workflow builder - Standard UI is maybe complicated for casual users?
Trello	+ Good WebHook support + Extensible through the creation of PowerUps - More limited functionality
LeanKit	+ Currently in use at Shelter - No WebHooks - would need to understand better how to integrate with the rest of the stack

Recommendation - Jira

Jira's excellent WebHook support makes it the best fit for integrating the different products that form the proposed architecture. Its visual representation of workflow also makes it an excellent fit for use as the source of truth for content moderation. Jira is an incredibly powerful and flexible tool - if not configured the options can be overwhelming for a casual user - so it's important that the interface is tuned, but its in depth permissions system mean it is suitable for a variety of users.

Costs

Name	Indicative Costs	Notes
Jira	£267/ month*	• Assumes 50 users - $7/ user/ month (prices reduce at 101 users plus)
Trello	£280/ month*	• Assumes 50 users - $9.99/ user/ month (billed annually)

* Price converted from Dollar's

Appendix A – Testing Approach

During the initial phase of the project, Manifesto will produce a Test Strategy document – outlining any our approach to testing on the project, establishing boundaries for Manifesto testing, and identifying any areas of high risk for consideration during testing. Attached below is a sample of the table of contents for a document for a project of a similar size.

Contents

Appendix B - Capabilities to Recommendations Index

Content Management

Capabilities

- 1.9.1 Intuitive content editing paradigm
- 1.9.2 WYSIWYG authoring
- 1.9.3 Modular authoring paradigm
- 1.9.5 Metadata management
- 1.9.11 Web forms authoring
- 1.9.13 Versioning
- 1.9.15 Publishing
- 1.9.20 Roles and permissions (author)
- 1.9.21 Visual/Creative design
- 1.9.24 Asset search and filter
- 1.9.25 Data entry validation
- 1.1.1 Scalability
- 1.1.2 Caching
- 1.1.6 Content syndication
- 1.1.8 Social media
- 1.1.12 Delivery failover
- 1.7.1 Content review / Live preview
- 1.10.1 Rich media asset management
- 1.10.8 Rendition creation and management
- 2.3.3 Multi-tenant architecture
- 3.1.1 Content import and export
- 4.3.2 Integration support

- 4.3.3 Content repository APIs

Workflow Engine

Capabilities

- 2.1.1 Publishing workflow

Component Based Framework

Capabilities

- 1.9.3 Modular authoring paradigm

Content Delivery

Capabilities

- 1.1.1 Scalability
- 1.1.2 Caching
- 1.1.12 Delivery failover
- 1.1.6 Content syndication

Integration

Capabilities

- 4.3.2 Integration support

Analytics

Capabilities

- 1.20.1 Campaign performance
- 1.20.2 3rd party referral tracking
- 2.6.1 Content analytics
- 2.6.2 Visitor analytics
- 2.6.3 Clickstream analysis

Ecommerce

Capabilities

- 2.13.1 Payment gateway

Email Distribution

Capabilities

- 1.1.9 Email distribution

Personalisation

Capabilities

- 1.16.1 Personalization
- 1.16.2 Personalised content
- 1.17.1 Segmentation
- 1.17.2 Campaign management
- 1.18.1 Multivariate testing
- 1.18.4 Content targeting
- 1.20.6 A/B and multivariate testing