Web y extranet Priorat

Rafael Soler Pérez

Resumen– El propósito principal de este proyecto es el de desarrollar un sitio web para los socios y entidades que trabajan en defensa del paisaje de la comarca del Priorat, así como la integración de una herramienta de trabajo colaborativo que suponga una mejora en la organización del día a día de los socios. Además, veremos el proceso de incorporación de un framework JavaScript moderno para ciertos componentes de un sitio web que ha sido construido con tecnologías web tradicionales como PHP.

Palabras clave– Desarrollo web, extranet, WordPress, framework, Vue.js, PHP, Bootstrap, LAMP

Abstract– The main purpose of this project is to develop a website for the partners and entities that work in defense of the landscape of the Priorat region, as well as the integration of a collaborative work tool that supposes an improvement in the organization of the partners day to day. Additionally, we will go through the process of incorporating a modern JavaScript framework for certain components of a website that has been built using traditional web technologies such as PHP.

Keywords– Web development, groupware, WordPress, framework, Vue.js, PHP, Bootstrap, LAMP

1 Introducción

Las páginas web se han convertido en un elemento esencial de las empresas y organizaciones de hoy en día, independientemente de su sector y del objetivo que tengan. Aún así, todavía hay empresas que no se dan cuenta o no son conscientes de la oportunidad que están perdiendo al no tener una página web o tenerla desactualizada. Ya no solo por el hecho de que es la máxima representación digital de una empresa en la red, sino que hay muchos motivos por los que cualquier empresa debería tener una página web, tal y como explica la revista Forbes [1]:

- Es una oportunidad de causar una buena primera impresión y de aumentar la credibilidad de la empresa, mostrando de una forma atractiva información relevante y de calidad a los usuarios interesados.
- Sirve para comunicar de una forma clara quién es, a quién representa y cuáles son sus objetivos.
- Una web optimizada para el *Search Engine Optimization* (SEO) aparecerá en los resultados de búsqueda a millones de clientes potenciales o de usuarios interesados en conocer más información sobre la empresa en todo el mundo.
- Permite mantener informado en todo momento sobre las últimas noticias o sobre las actividades que están llevando a cabo. Esto también ayuda a reducir de forma considerable el tiempo invertido en atender a los clientes, ya que toda la información importante estará disponible en la web.

Por este motivo, los socios y entidades que trabajan en defensa del patrimonio cultural y paisajístico del *Priorat-Montsant-Siurana*, han encargado la creación de un sitio web que haga visibles las características y valores principales de esta comarca –que es uno de los candidatos a patrimonio mundial de la UNESCO [2]–, así como de las entidades que trabajan en ella y de las actividades que están llevando a cabo en todo momento.

Por otro lado, hoy en día disponemos de una gran cantidad de herramientas de trabajo colaborativo de diferentes proveedores donde escoger, cada una con sus ventajas y desventajas, por lo que al final es probable que acabemos usando herramientas de diferentes proveedores al mismo tiempo. Por poner un ejemplo, una empresa podría estar usando Dropbox para almacenar archivos en un entorno compartido, Google Calendar para tener una agenda compartida, Zoom para las reuniones de equipo, WhatsApp para comunicarse y planificar las reuniones, etc.

• E-mail de contacte: rafasoler10@gmail.com
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Aitor Alsina Rodríguez (departament)
• Curs 2021/22

Este es el problema con el que se han encontrado los socios que trabajan en el Priorat al trabajar de forma conjunta, por lo que están buscando unificar en un único entorno los servicios y herramientas que utilizan en su día a día para organizarse y colaborar entre ellos. A lo largo del artículo, nos referiremos a este entorno colaborativo usando los términos *Extranet* o *Groupware*.

Es importante resaltar que, al tratarse de un proyecto que ha sido encargado por un cliente real, tendremos ciertas restricciones en cuanto a las funcionalidades implementadas y a las herramientas utilizadas, impuestas tanto por el cliente –la asociación Prioritat–, como por nuestra empresa –La Tempesta–.

A lo largo de este artículo detallaremos los objetivos de este trabajo y hasta donde queremos llegar, explicaremos la metodología que hemos seguido para cumplir con estos objetivos, mostraremos las diferentes fases del proyecto por las que hemos pasado, explicaremos el proceso de desarrollo de la aplicación y acabaremos mostrando los resultados obtenidos y unas conclusiones finales.

2 Objetivos

Puesto que este proyecto tiene dos componentes bien diferenciados, dividiremos este apartado en varios subapartados, uno en el que detallaremos los objetivos del sitio web, otro para los objetivos de la extranet y un tercer apartado con unos objetivos más generales.

2.1. Sitio web

Por un lado, tenemos el desarrollo de un sitio web, cuyo principal objetivo es exponer las características y los valores ambientales y patrimoniales del paisaje cultural del Priorat-Montsant-Siurana de una forma visualmente atractiva. Al mismo tiempo, el sitio web debe servir para representar a los socios y entidades colaboradoras del Priorat y proporcionar transparencia en las actividades y proyectos que están llevando a cabo en todo momento. Para cumplir con este objetivo, se tiene que poder gestionar de una forma sencilla e intuitiva –incluso para personas sin conocimientos técnicos– un blog donde publicar avisos o noticias, para tener la web siempre actualizada con la información más reciente.

2.2. Extranet

Por otro lado, tenemos el desarrollo y/o integración de una herramienta de trabajo colaborativo, con el objetivo de unificar los múltiples servicios y herramientas que utilizan los socios del Priorat en un único entorno, para así facilitar la gestión del trabajo y la comunicación entre ellos.

Esta herramienta debe permitir la autenticación de usuarios con diferentes permisos o perfiles personalizables para poder acceder únicamente a los grupos de trabajo que cada miembro tenga asignado. Respecto a las funcionalidades que debería incluir, encontramos un calendario o agenda compartida, un tablón de anuncios en el que se pueden publicar comentarios o respuestas, un espacio compartido donde almacenar documentos y una herramienta de planificación y seguimiento de proyectos estratégicos, con la posibilidad de añadir más funcionalidades en el futuro.

Siguiendo los requisitos del cliente, tendremos que utilizar una herramienta de código abierto existente en el mercado que ya incluya la mayoría de las funcionalidades requeridas y, partiendo de esta herramienta, realizar las adaptaciones que sean necesarias y la integración con el sitio web.

Puesto que el desarrollo de una herramienta de este tipo hubiera sido interesante a nivel académico, hemos decidido realizar por cuenta propia el desarrollo a medida de algunas de estas herramientas, como el tablón de anuncios o el calendario compartido. Con el objetivo de aportar modularidad, flexibilidad y facilidad de integración con el sitio web, desarrollaremos estas herramientas como *plugins* de WordPress, que podrán ser activados y desactivados de forma independiente.

2.3. Objetivos generales

También hemos identificado algunos objetivos más generales, que afectan tanto al sitio web como a la extranet:

- Todas las funcionalidades de la web y extranet han de ser compatibles con cualquier dispositivo, sistema operativo y navegador moderno (Chrome, Firefox, Edge, Safari, Opera...). La compatibilidad con Internet Explorer no la hemos considerado importante ahora que Microsoft ha anunciado que el fin de este navegador será en junio de 2022 [5].
- El diseño ha de ser *responsive*, es decir, se ha de adaptar perfectamente a cualquier tamaño de pantalla, ya que tanto la web como la extranet se accederán desde dispositivos muy diversos.
- En la medida de lo posible, intentaremos cumplir con el estándar de accesibilidad web WCAG 2.1.
- Tiene que haber algún tipo de interconexión entre el sitio web y la extranet, por ejemplo, para mostrar el calendario compartido públicamente en la web, ya sea con una *Application Programming Interface* (API) de tipo *Representational State Transfer* (REST), o con cualquier otro método que consideremos oportuno.

Además de los objetivos que en parte ha definido el cliente, el desarrollo de este proyecto aportará a nivel personal experiencia en la gestión de proyectos reales, no solo en la fase de desarrollo, también en la toma de requisitos haciendo reuniones con el cliente, en las fases de diseño y planificación de las tareas y en todo su ciclo de vida.

3 Estado del arte

En cuanto a la investigación del estado actual y la búsqueda de herramientas que puedan solucionar el problema planteado, nos hemos visto en cierto modo limitados, ya que el proyecto proviene de una empresa y de un cliente real, y al final lo que necesitan es una aplicación que proporcione fiabilidad y seguridad.

Siguiendo el flujo de trabajo de nuestra empresa, usaremos WordPress para la creación del sitio web, el *Content Management System* (CMS) más utilizado en la actualidad, usado en un 43,3 % de todas las webs existentes [3].

Además, hemos decidido utilizar *Understrap*, que es un *framework* de código abierto de creación de temas WordPress a medida [4] ya que esto nos dará mucha flexibiliad a la hora de construir el sitio web, tal y como explicaremos más adelante en el apartado 6.1. Esto nos permitirá profundizar en el desarrollo de WordPress, así como en conceptos avanzados de HTML5, SCSS, JavaScript, AJAX y PHP, entre otros.

Respecto a la extranet, el cliente solicitaba el uso de una herramienta de trabajo colaborativo de código abierto que tuviera la posibilidad de instalarse en sus propios servidores. Después de investigar herramientas de este tipo hemos llegado a la conclusión de que Nextcloud es la mejor opción pues es una de las pocas aplicaciones mínimamente serias que hemos encontrado que es open source y que cumple con prácticamente todos los requisitos del cliente. Además, puede ser ampliado mediante aplicaciones, de las cuales podemos encontrar cientos de ellas en el repositorio oficial de Nextcloud y, en caso de que fuera necesario, ofrecen documentación para desarrollar nuestras propias aplicaciones para extender Nextcloud.

4 METODOLOGÍA

En este apartado explicaremos las herramientas que hemos utilizado para llevar a cabo una correcta gestión y planificación del proyecto, y detallaremos cómo tenemos organizado nuestro entorno de desarrollo así como las tecnologías que están involucradas.

4.1. Gestión y planificación de las tareas

Para la gestión del proyecto usaremos la metodología Kanban, una metodología ágil que permite ver el estado del proyecto y el flujo de trabajo de una forma muy visual, en un tablero dividido en varias columnas, donde cada columna representa una etapa del desarrollo. Destaca por su flexibilidad para adaptarse a cualquier entorno de trabajo, por su capacidad de respuesta ante imprevistos y por su facilidad de uso.

Concretamente, seguiremos un flujo de trabajo habitual en el desarrollo web, con las siguientes columnas:

- *Backlog*: Lista con todas las tareas, requisitos del cliente o ideas que queremos o deberíamos cumplir, ordenadas según su prioridad.
- *To do*: Tareas que hemos decidido realizar en un período de tiempo determinado.
- *In Progress*: Contiene las tareas en las que actualmente se está trabajando. Para mejorar la productividad, limitaremos el número de tareas que puede haber al mismo tiempo en este estado a tres. De esta forma nos centraremos en completar las tareas que hemos empezado y evitaremos dispersarnos trabajando en muchas cosas al mismo tiempo.
- *Testing*: Las tareas que requieran una fase de pruebas pasarán por esta columna antes de ser desplegadas en el entorno de producción.
- *Done*: Una vez finalizada la tarea, pasará a esta columna, donde finalizará su recorrido.

4.2. Entorno de desarrollo

En un principio teníamos pensado usar Windows como servidor web usando la aplicación Local WP para gestionar el entorno local de desarrollo, pero hemos visto que Nextcloud no admite Windows como servidor, por lo que hemos decidido usar *Windows Subsystem for Linux* (WSL) con una distribución de Ubuntu 20.04.

WSL es una capa de compatibilidad que permite ejecutar un entorno GNU/Linux, incluida la mayoría de herramientas de línea de comandos, utilidades y aplicaciones, directamente en Windows, sin la sobrecarga de una máquina virtual o un arranque dual [10].

4.2.1. Arquitectura

Para montar el servidor seguiremos la arquitectura LAMP, compuesta por Linux como sistema operativo, Apache como servidor web, MariaDB como sistema de gestión de base de datos y PHP como lenguaje de scripting del lado del servidor.

Aprovecharemos el mismo servidor tanto para la web como para la extranet, pero cada herramienta tendrá una base de datos dedicada. Para conseguir esto, haremos uso de los hosts virtuales de Apache de forma que la web estará alojada en el dominio *prioritat.local* y la extranet en el subdominio *cloud.prioritat.local*. Este dominio únicamente estará presente en nuestro entorno de desarrollo y, para poder acceder a él, tendremos que modificar el archivo `c:\Windows\System32\Drivers\etc\hosts` para asignar este dominio a la dirección IP de nuestro servidor web linux.

En la figura 1 podemos ver una visión más general de la arquitectura de nuestra aplicación y de los componentes que están involucrados.

4.2.2. Creación de un certificado *Secure Sockets Layer* (SSL)

Más adelante, cuando tengamos un dominio público solicitaremos un certificado SSL a una Autoridad de Certificación (CA) de confianza, pero mientras estemos trabajando en un entorno de desarrollo local, para poder usar el protocolo HTTPS en la web y en la extranet necesitaremos un certificado autofirmado.

Usando la herramienta `openssl`, lo primero que haremos será generar la clave privada de la CA –`CA.key`– y el certificado raíz autofirmado –`CA.crt`–. También tendremos que generar la clave privada del servidor web local –`localhost.key`– y un *Certificate Signing Request* (CSR) –`localhost.csr`–.

Una vez completados estos pasos, solicitaremos a la CA que firme nuestro CSR, lo que generará el archivo `localhost.crt`, el certificado de nuestro servidor web local. Además de asignar el certificado a localhost, también añadiremos los dominios *prioritat.local* y *cloud.prioritat.local*, que serán los que usaremos en nuestro entorno de desarrollo para la web y la extranet respectivamente.

Ahora solo queda habilitar el módulo SSL de Apache con el comando `a2enmod ssl`, modificar los archivos de configuración de Apache para señalar la ubicación del certificado local y de la clave privada, y añadir la CA a nuestro

navegador o dispositivo. Sin este último paso, al acceder al servidor mediante HTTPS nos encontraríamos con un error indicando que la CA no es de confianza.

4.3. Tecnologías utilizadas

Dividiremos este apartado en web y extranet ya que usaremos tecnologías bastante distintas para cada una.

4.3.1. Web

Tal y como hemos mencionado anteriormente, para el sitio web usaremos WordPress junto con Understrap, que hará la función de un tema padre. Todo el desarrollo a medida lo realizaremos en un tema hijo que hereda las funcionalidades básicas del tema padre y al mismo tiempo nos permite realizar todas las modificaciones y ampliaciones que hagan falta.

Necesitaremos tener *Node.js* instalado en nuestro dispositivo, ya que Understrap usa herramientas como *Rollup.js* y *PostCSS* para automatizar ciertos procesos durante el desarrollo, como la compilación de archivos SCSS a CSS, la minificación de archivos JavaScript y la concatenación de diversos archivos [6]. También viene incluido *Bootstrap 5*, para agilizar el proceso de creación de un frontend responsive.

En la figura 2 podemos ver la estructura de directorios de nuestro tema `prioritat-LaTempesta`. Para empezar a trabajar en la aplicación lo primero que haremos será ubicarnos en la carpeta raíz del tema y ejecutar el comando `npm install`, lo que creará la carpeta `node_modules` con todas las dependencias que estén indicadas en el archivo `package.json`. Una vez hecho esto, tendremos acceso al comando `npm run watch-bs` que será el que usaremos a lo largo de todo el desarrollo de nuestra aplicación. Este comando tiene dos funciones principales:

- Por un lado, usando la herramienta `nodemon`, hace un seguimiento de los cambios en los archivos JS y SCSS que están ubicados en la carpeta `src/`, realiza unas tareas de compilación, minificación y concatenación –entre otras–, y genera los archivos finales en las carpetas `css/` y `js/`.
- Por otro lado, usando la herramienta `Browsersync`, cada vez que hay un cambio en el código fuente y finaliza el proceso de compilación, se sincroniza con una pestaña del navegador, recargando la página cuando esté lista para mostrar los últimos cambios. En el caso de que sean cambios en los estilos CSS, se mostrarán directamente en el navegador sin tener que recargar la página.

4.3.2. Extranet

Para la parte de la extranet, usaremos *Nextcloud* como herramienta de groupware, ya que incluye la mayoría de las funcionalidades requeridas por el cliente y, al ser de código abierto, puede ser alojada en nuestros propios servidores y adaptada según nuestras necesidades.

Asimismo, crearemos una versión más simplificada de estas herramientas de trabajo colaborativo desarrollando plugins para WordPress. Normalmente estos plugins se construyen siguiendo el manual de creación de plugins para desarrolladores [7], pero trataremos de integrar el *framework Vue.js*, al menos en el plugin del calendario compartido, para proporcionar una experiencia de usuario más satisfactoria, con una interfaz más ágil e interactiva. Esta práctica no está muy extendida, pero hay algunos plugins como *WP Rocket* –uno de los plugins de caché más populares de WordPress–, que usan *Vue.js* para tener un panel de administración más personalizado [8].

5 PLANIFICACIÓN

En este apartado identificaremos los pasos a seguir para desarrollar el proyecto de una forma existosa y estableceremos una planificación de las diferentes fases del proyecto. En el diagrama de Gantt de la figura 3 se puede ver una planificación más detallada de todas las tareas a realizar.

En una primera fase, nuestra empresa se centrará únicamente en realizar reuniones periódicas con los socios del Priorat para recopilar todos los requisitos, los objetivos y la finalidad del proyecto, tanto de la parte web como de la extranet. La asistencia a estas reuniones dependerá del objetivo que tengan. En nuestro caso, iremos a las reuniones que estén relacionadas directamente con el desarrollo de la aplicación.

Una vez tengamos esta parte bien definida la diseñadora web de nuestra empresa realizará unos *wireframes* para empezar a tener una idea general de cómo estará estructurada la web. Una vez validado con el cliente, pasará a diseñar unos *mockups* más detallados y que serán más parecidos al diseño final del producto, incluyendo colores, tipografía, imágenes, iconos, etc. Por último, usando el programa *Adobe XD*, creará un prototipo de alta fidelidad del diseño final de la web y de las principales interacciones que el usuario tendrá con el sistema.

Mientras la diseñadora va trabajando en los mockups, podemos ir realizando un estudio comparativo de los diferentes servicios y tecnologías que hay disponibles en el mercado, con tal de verificar que nuestra idea inicial de usar WordPress para la web y Nextcloud para la extranet es la solución óptima para este caso. Relacionado con lo anterior, también tendremos que analizar cual es el mejor servicio donde alojar la web y extranet, pues hay muchas combinaciones posibles: pueden estar ambos alojados en un *hosting* compartido o en un *Virtual Private Server* (VPS), puede estar la web en un servidor y la extranet en otro servidor independiente, etc.

La siguiente fase será íntegramente para el desarrollo de la aplicación. Para empezar, realizaremos algunas pruebas y prototipos para ver cómo se comportan las herramientas finalmente seleccionadas para desarrollar la web y la extranet. El sitio web lo iremos desarrollando a medida que vayamos recibiendo los prototipos de la diseñadora y, mientras tranto, podremos ir avanzando en otros aspectos de la extranet y de los plugins de WordPress, así como en la integración de la extranet con la web mediante una API REST.

Para acabar, tendremos una fase de despliegue, que ejecutaremos primero en un entorno de *staging* o preproducción lo más parecido posible al entorno final para verificar que todo funciona correctamente. Durante esta fase también estaremos aplicando todos aquellos cambios que nos solicite el cliente después de ver y probar la aplicación en un entorno prácticamente definitivo.

6 DESARROLLO DE LA APLICACIÓN

En este apartado explicaremos como ha sido el proceso de desarrollo de la aplicación, tanto de la parte web como de la extranet.

6.1. Web

Puesto que el diseño del frontend está hecho a medida por la diseñadora de nuestra empresa siguiendo las necesidades del cliente, sería muy complicado encontrar un tema de WordPress que cumpla con todos los requisitos, por eso decidimos usar el framework Understrap. Este tema prácticamente no incluye estilos ni tiene una estructura predeterminada para las páginas, sino que únicamente incluye las funcionalidades básicas de un tema WordPress, como la inclusión de meta etiquetas, hojas de estilo y scripts en el *head* de todas las páginas, la gestión de los menús de navegación y la paginación de los *posts* entre otros.

Para cada página desarrollada, crearemos una plantilla PHP en el directorio `page-templates/`, en la que tendremos que llamar a las funciones *get_header()* justo al inicio del archivo y *get_footer()* justo al final para construir la estructura básica de una página HTML, y todo lo que incluyamos entre estas dos funciones estará dentro del *body* de la página.

6.1.1. Bootstrap

Para construir un diseño completamente responsive de una forma más ágill, hemos usado Bootstrap. A continuación explicaremos algunas de las utilidades de Bootstrap que hemos encontrado más interesantes.

Grid: Hemos usado el sistema de cuadriculas de Bootstrap, que se basa en el método CSS *flexbox*, para organizar la disposición de elementos en la página. Este sistema está formado por contenedores, filas y columnas. Normalmente en cada página solo hay un contenedor que agrupa todo el contenido. Dentro puede haber cualquier número de filas que, a su vez, están divididas en doce columnas [11].

En el código de la figura 1 se puede ver un ejemplo del Grid de Bootstrap usado para crear la sección de agenda en la página de inicio. Las clases que nos interesan son `.row` –que define una fila–, y `.col-{breakpoint}-{nColumns}` –que define una columna–. En este caso usamos el *breakpoint* `md` para que las columnas solo se formen en pantallas superiores a 768 pixeles de anchura.

```
<div class="row g-0 align-items-center">
  <div class="col-md-5 p-3">
    <div class="calendar-monthly-view">...</div>
  </div>
  <div class="col-md-7 ps-0 ps-sm-4">
    <div class="calendar-list-view">...</div>
  </div>
</div>
```

Listing 1: Ejemplo del grid de Bootstrap

Espaciado: Siguiendo con el anterior código de la figura 1, otra de las utilidades que hemos usado mucho son las clases del estilo *.p-{n}*, *.m-{n}* y *.g-{n}*. Estas clases sirven, respectivamente, para definir el espaciado interior de los elementos –*padding*–, el margen exterior –*margin*– y el espaciado entre columnas –*gap*–.

Flexbox: En algunos casos es mejor utilizar las utilidades de *flexbox* en lugar del *grid* de Bootstrap, ya que nos permite crear un diseño aún más flexible. Por ejemplo, en el código de la figura 2 podemos ver como hemos definido la lista de los cuatro valores del Priorat para conseguir un diseño que se ajusta perfectamente a cualquier tamaño de pantalla, independientemente del número de elementos que haya. Únicamente hemos añadido tres atributos CSS adicionales (*flex*, *min-width* y *max-width*), sin utilizar en ningún momento *media queries* para hacer ajustes en tamaños de pantalla determinados.

```
<div class="values-list d-flex flex-wrap gap-3
    justify-content-center">
 <div class="value p-3" id="mermership">...</div>
 <div class="value p-3" id="harmony">...</div>
 <div class="value p-3" id="culture">...</div>
 <div class="value p-3" id="heritage">...</div>
</div>
```

Listing 2: Ejemplo de las utilidades flexbox

6.1.2. The Loop

Un aspecto esencial de WordPress que debemos mencionar es el concepto conocido como *The Loop*. *The Loop* es un código PHP que se usa para mostrar todo tipo de posts [12]. En Wordpress, un post no se refiere únicamente a una entrada del blog, sino que es una estructura que se usa como base para gestionar otros tipos de contenido, entre los que se incluyen las páginas, los archivos de la biblioteca de medios y los menus de navegación, entre otros. Esta estructura también puede ser usada como base para definir nuestras propias estructuras de datos, conocidas como *Custom Post Types* (CPT).

En el código de la figura 3 podemos ver un ejemplo del Loop de WordPress que hemos implementado en la página de inicio para mostrar una lista de las cuatro últimas entradas del blog, ordenadas de forma descendente usando la fecha de publicación como criterio.

```
<?php
$args = array(
  'post_type' => 'post',
  'posts_per_page' => 4,
  'orderby' => 'date',
  'order' => 'DESC',
);
$query = new WP_Query( $args );
if ( $query->have_posts() ) {
  while ( $query->have_posts() ) {
    $query->the_post();
    get_template_part(
        'loop-templates/content', get_post_type()
    );
  }
}
wp_reset_postdata();
?>
```

Listing 3: Ejemplo del Loop de WordPress

El funcionamiento es el siguiente:

- Hacemos una consulta a la base de datos para recoger los datos necesarios usando la clase *WP_Query*.
- Iniciamos un bucle si la consulta a generado resultados.

- Dentro del bucle, llamamos a la función *the_post()* que se encarga de iterar sobre los posts, usando un contador interno que se incrementa cada vez que es llamado.
- Importamos la plantilla específica que hemos desarrollado para mostrar los posts y, como estamos dentro del Loop, podremos llamar a funciones como *the_title*, *the_content*, etc. para obtener los datos del post actual.
- Para finalizar, llamamos a la función *wp_reset_postdata()* para restaurar el loop principal.

6.1.3. Contenido dinámico

Puesto que el cliente puede necesitar en algún momento hacer cambios a los textos de la web, tendremos que idear alguna forma de que pueda hacerlo sin tener que editar los archivos PHP. Para ello, usaremos los plugins *Custom Post Types UI* y *Advanced Custom Fields* del repositorio de WordPress para añadir algunos campos que el cliente pueda gestionar fácilmente desde la página de administración de WordPress.

Por ejemplo, la lista de valores que hay en la página de inicio, inicialmente tiene cuatro elementos previamente definidos, pero el cliente podria decidir en cualquier momento hacer ajustes en estos elementos o incluso añadir o borrar elementos. Para ello, en lugar de *hardcodear* estos elementos en la plantilla PHP, usaremos el Loop de WordPress y la función `get_field()` del plugin *Advanced Custom Fields* para obtener los datos que defina el cliente.

6.1.4. Multilingüe

Una funcionalidad importante que debe ofrecer nuestra aplicación es la posibilidad de mostrarse en diferentes idiomas según las preferencias del usuario. Esto es algo que deberíamos tener en cuenta desde un inicio, ya que todos los textos que hayamos puesto en las plantillas PHP deberán ser incluidos usando una función de WordPress encargada de mostrar el texto en el idioma adecuado.

Podemos ver un ejemplo de un texto preparado para una web multilingüe en el código de la figura 4. El primer parámetro de la función es el texto a traducir y el segundo parámetro es el grupo o dominio al que pertenece, en nuestro caso, hemos definido el grupo “prioritat” para todos los textos de nuestro tema WordPress.

```
<h2 class="bg-title bg-title-tertiary mb-4">
  <?= __( 'Ultimes noticies', 'prioritat' ) ?>
</h2>
```

Listing 4: Texto preparado para ser traducido

6.1.5. Accesibilidad

Desde el inicio del proyecto le hemos dado cierta importancia a la accesibilidad de nuestra aplicación, ya que es uno de los requisitos que ha marcado el cliente. Algunos de los aspectos que hemos tenido en cuenta para mejorar la accesibilidad son:

- Hemos comprobado que los colores del texto y de fondo tienen suficiente contraste.
- Siempre que sea posible, utilizamos etiquetas HTML5 más específicas en lugar de usar siempre la etiqueta general `<div>`, por ejemplo: `<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<footer>`, `<ul>`, `<li>`, `<time>`, etc.
- Se sigue un orden y una estructura semántica de cabezeras con sentido, es decir, se usa la etiqueta `<h1>` para el título de la página, `<h2>` para las principales secciones, `<h3>` para las subsecciones y así sucesivamente.
- Se puede navegar fácilmente por la web usando únicamente el teclado y el usuario sabe en todo momento que elemento tiene el *focus*.
- Se hace uso del atributo `alt` para proporcionar un texto alternativo a las imágenes que lo necesiten.
- Los elementos interactivos –como botones y enlaces– disponen de un texto con el que queda claro su propósito, incluso para usuarios que tienen que usar tecnologías de asistencia. En el código de la figura 5 se puede ver un ejemplo de como hemos hecho accesible los enlaces a las redes sociales. Lo que hacemos es ocultar el icono en formato SVG a los usuarios de *screen readers* –usando el atributo `aria-hidden="true"`– ya que no les aportaría nada útil y, en su lugar, ponemos un texto describiendo el propósito del enlace, usando la clase `.visually-hidden` para que este texto solo sea visible en el *screen reader*. Además, hemos añadido un aviso de que este enlace se abre en una nueva ventana, ya que esto podria desorientar a las personas que tienen discapacidades cognitivas o dificultades para percibir el contenido visual [13]

```
<a class="icon icon-twitter" href="https://
    twitter.com/prioritat" title="Twitter" target
    ="_blank" rel="noopener noreferrer">
  <span class="visually-hidden">
    <?= __( 'Twitter de Prioritat (Obre en una
    finestra nova)', 'prioritat' ) ?>
  </span>
  <svg aria-hidden="true" [...]>...</svg>
</a>
```

Listing 5: Enlace a las redes sociales accesible

6.2. Extranet: Nextcloud

Como hemos mencionado anteriormente, para la parte de la extranet usaremos Nextcloud, que es una aplicación de código abierto con herramientas de trabajo colaborativo, tales como un espacio para almacenar documentos, un calendario compartido y un tablón de anuncions, entre otras muchas funcionalidades que pueden ser añadidas desde el repositorio de aplicaciones de Nextcloud.

Puesto que Nextcloud ya es una aplicación lista para ser utilizada, por nuestra parte lo único que tendremos que hacer será crear la base de datos en el servidor, configurar Apache para hacer uso de un subdominio, lanzar el instalador de Nextcloud, establecer la configuración inicial y realizar algunos retoques para dejarlo a gusto del cliente.

Más adelante, en el apartado 6.3.1.5, explicaremos como hemos integrado el tablón de anuncios de Nextcloud en uno de los plugins WordPress que hemos construido, usando una API REST.

6.3. Extranet: Plugins WordPress

Por otro lado, también hemos desarrollado unos plugins para WordPress que nos permitirán tener directamente en nuestra web algunas funcionalidades similares a las de Nextcloud. De esta forma podremos hacer una comparación entre instalar en el servidor una herramienta open source lista para producción y tratar de integrarla con la web usando una API REST, o bien desarrollar desde cero unas herramientas especialmente pensadas para WordPress.

Para crear la estructura de directorios básica de un plugin WordPress estaremos usando el *WordPress Plugin Boilerplate Generator* [14].

6.3.1. Tablón de anuncios

El primer plugin que hemos desarrollado es el tablón de anuncios. Este plugin permite a los usuarios crear anuncios fácilmente desde la página de administración de WordPress y mostrarlos públicamente en un apartado de la web. Los anuncios estarán formados por un título, una descripción corta y un enlace –que puede ser a una página interna, para dar más detalles sobre este anuncio, o un enlace externo–.

6.3.1.1 Creación del Custom Post Type

Para empezar tendremos que crear un *Custom Post Type* (CPT), que nos permitirá tener una estructura de datos personalizada para almacenar los anuncios en la base de datos. En el código de la figura 6 podemos ver como hemos registrado un CPT al que hemos llamado `nb_announcements`. Algunos campos que vale la pena mencionar son el de `supports`, que define los campos que tendrá este CPT, el de `has_archive` que hemos definido como `false` ya que hemos construido nuestras propias plantillas usando el Loop de WordPress, y el de `show_in_rest` que nos permitirá usar el editor *Gutenberg* integrado en WordPress para el contenido del anuncio.

```
$args = array(
  'labels'        => $labels,
  'public'        => true,
  'show_in_menu'  => $this->plugin_name,
  'supports'      => array( 'title', 'editor', '
    custom_fields'),
  'has_archive'   => false
  'menu_icon'     => 'dashicons-megaphone'
  'rewrite'       => array( 'slug' =>
    _x( 'anuncis', 'slug', 'noticeboard' ),
  'show_in_rest'  => true
);
register_post_type( 'nb_announcements', $args );
```

Listing 6: Registro de un CPT

6.3.1.2 Agregar los campos personalizados

Siguiendo con el anterior código de la figura 6, podemos ver que nuestro CPT tiene un campo para el título, otro para el editor de texto y uno especial llamado `custom_fields`. Esto nos permitirá añadir nuestros propios campos completamente personalizados usando la función `add_meta_box()` de WordPress.

En el código de la figura 7 podemos ver un ejemplo del campo que hemos añadido para permitir un resumen del anuncio. Dentro de este campo llamamos a la función `get_post_meta()` que se encarga de recoger de la base de datos el contenido que haya introducido el usuario anteriormente. Cuando el usuario guarde los cambios, usaremos la función de WordPress `update_post_meta()` con el nuevo valor para que se actualice la base de datos.

```
<label for="announcement_summary">
  <?= __( 'Resum', 'noticeboard' ) ?>
</label>
<textarea name="announcement_summary" id="
    announcement_summary">
  <?= get_post_meta( $post->ID, '
    announcement_summary', true ); ?>
</textarea>
```

Listing 7: Campo de resumen de un anuncio

6.3.1.3 Añadir pestaña al menu

Para que los usuarios puedan gestionar los anuncios, deberemos añadir una nueva pestaña al menú de administración de WordPress desde la que podamos añadir, editar y borrar los anuncios. En el anterior código de la figura 6 ya hemos indicado con el parámetro `show_in_menu` que esta página debe aparecer en la pestaña dedicada a nuestro plugin, así que solo nos falta registrar esta pertaña usando la función `add_menu_page()` de WordPress.

6.3.1.4 Creación de un shortcode

Una vez tenemos toda la estructura lista, tenemos que decidir como se van a mostrar los anuncios en el frontend. En este caso, vamos a usar la API de *Shortcode* de WordPress para generar un código sencillo y fácil de recordar que podremos usar en cualquier parte, por ejemplo en una plantilla PHP, en el editor de WordPress o en un *widget* de nuestro tema.

Usaremos la función `add_shortcode()` para registrar nuestro shortcode y permitiremos dos parámetros: `limit` para establecer el número máximo de anuncios a mostrar, y `source` para establecer si la fuente de datos debería ser WordPress o Nextcloud –que implementaremos en el siguiente apartado 6.3.1.5–.

En el código de la figura 8 podemos ver como usamos este shortcode para generar una lista de los últimos 4 anuncios en la página de inicio.

```
<div class="noticeboard col-lg-4 d-flex flex-
    column ps-3">
  <h3>
    <?= __('Taulell d\'anuncis', 'prioritat') ?>
  </h3>
  <?= do_shortcode('[latest_announcements limit
    =4]') ?>
</div>
```

Listing 8: Uso del shortcode que hemos creado

6.3.1.5 Integración de la API REST de Nextcloud

Una funcionalidad adicional que le hemos añadido a nuestro plugin es la posibilidad de conectarse con una instancia de Nextcloud mediante una API REST para recoger los datos del tablón de anuncios que también incluye esta plataforma y así poder mostrarlos en el frontend de la web.

Nextcloud dispone de una API REST que sigue el estándar de *Open Collaboration Services* (OCS) para definir los *endpoints*, los métodos de autenticación y la estructura de las respuesta del servidor.

En el código de la figura 9 podemos ver un ejemplo de llamada a la API para obtener los datos de la aplicación *Announcement Center* de Nextcloud. Usaremos la función `get_option()` de WordPress para obtener algunos parametros –como la URL de la API y el usuario y contraseña de Nextcloud–, que habrán sido definidos previamente por el usuario desde la página de configuración de nuestro plugin. Veremos en más detalle como hemos creado esta página de configuración en el siguiente apartado 6.3.1.6. Para hacer la consulta a la API usando el método HTTP GET utilizaremos la función `wp_remote_get()`, pasándole como parámetros la URL del endpoint y algunas cabeceras adicionales.

```
$base_url = get_option( '
    noticeboard_nextcloud_url_setting' );
$user = get_option( '
    noticeboard_nextcloud_user_setting' );
$password = get_option( '
    noticeboard_nextcloud_password_setting' );
$url = trailingslashit( $base_url ) . 'ocs/v2.php
    /apps/announcementcenter/api/v1/announcements
    ';
$args = array(
  'headers' => array(
    'Authorization' => 'Basic ' . base64_encode(
    $user . ':' . $password ),
    'OCS-APIRequest' => true,
   ),
 );
$response = wp_remote_get( $url, $args );
```

Listing 9: Llamada a la API de Nextcloud para obtener los anuncios

Una vez obtenida la respuesta, deberemos parsear el XML para coger los datos necesarios y formatearlos para que puedan ser mostrados en el frontend. Un aspecto a tener en cuenta es que los anuncios de Nextcloud admiten texto en formato *Markdown*, así que usaremos la librería *Slimdown* de PHP para procesar el resultado devuelto por la API y transformar las etiquetas Markdown en etiquetas HTML [15].

6.3.1.6 Creación de una página de ajustes

Como hemos mencionado anteriormente, necesitaremos una página de ajustes en nuestro plugin para que el usuario pueda definir la URL base donde estará ubicado Nextcloud, así como el usuario y contraseña de esta plataforma, ya que todas las llamadas a la API deben estar correctamente autenticadas.

Para conseguir esto, deberemos llamar a la función `add_submenu_page()` que añadirá una nueva página al menú de administración de nuestro plugin, y crearemos una plantilla en PHP que contendrá un formulario con todos los campos necesarios. También tendremos que usar la API de *Settings* de WordPress para registrar en nuestro plugin los nuevos campos que hemos definido, usando las funciones `register_setting()` y `add_settings_field()`.

6.3.1.7 Aspectos de seguridad

Algunas de las medidas de seguridad que hemos implementado en el plugin son el uso de las funciones `sanitize_*()` para filtrar el contenido que introduce el usuario, el uso de `esc_*()` para filtrar el contenido mostrado al usuario –así evitamos ataques *Cross-site scripting* (XSS)–, la verificación del rol de usuario usando la función `current_user_can()` antes de ejecutar cualquier operación sensible, y el uso de *nonces* para proteger los formularios de usos no deseados y así evitar ataques de *Cross-site request forgery* (CSRF).

6.3.2. Calendario de eventos

El siguiente plugin que hemos decidido desarrollar es el calendario de eventos. Este plugin permite que aquellos usuarios que tengan suficientes permisos puedan gestionar un calendario compartido desde el panel de administración de WordPress, que podrá ser mostrado públicamente en la web. No repetiremos los pasos para configurar un plugin ni las formas de definir nuestra propia estructura de datos ya que esto lo hemos cubierto en la anterior sección 6.3.1 donde hemos explicado la creación del plugin del tablón de anuncios. En este caso nos centraremos en la parte más interesante e innovadora que es la integración del framework *Vue.js* para mostrar el calendario de eventos.

Antes de entrar en detalles, explicaremos a grandes rasgos qué es y que ventajas nos aporta este framework.

Vue.js es un framework de JavaScript que proporciona un modelo de programación declarativo y basado en componentes, que nos ayuda a desarrollar interfaces de usuario de cualquier complejidad de una manera eficiente [16]. La principal funcionalidad de este tipo de frameworks es que son reactivos, es decir, hacen un seguimiento de los cambios de estado de JavaScript y actualizan el *Document Object Model* (DOM) en consecuencia a estos cambios. Esto es algo que encaja perfectamente con el tipo de aplicación que queremos desarrollar, y es por este motivo que hemos decidido usar Vue.js para el calendario.

Lo primero que deberemos hacer será añadir a la API REST de WordPress nuestros propios endpoints usando las funciones `register_rest_route()` i `register_rest_field()`, para poder gestionar el calendario desde Vue. Concretamente, definiremos un endpoint para obtener los eventos en un rango de fechas determinado, y otro endpoint para actualizar los detalles del evento.

A continuación explicaremos algunos de los componentes más destacables que hemos desarrollado.

6.3.2.1 Componente raíz

Empezaremos con el componente raíz que es el encargado de agrupar el resto de componentes. Aquí guardaremos el estado del mes y año actual, los eventos obtenidos de la API y una variable que servirá para controlar cuando empieza y cuando acaba de cargarse los datos asíncronos, para

así poner algún tipo de *loader*. En el código de la figura 10 podemos ver una parte de la plantilla de este componente.

```
<month-indicator @change-month="updateDate" :
    month="currentMonth" :year="currentYear" />
<days-of-week />
<b-skeleton-wrapper :loading="isLoading">
  <template #loading>
    <date-grid-skeleton :month="currentMonth" :
    year="currentYear" />
  </template>
  <date-grid
    :month="currentMonth" :year="currentYear"
    :cachedEvents="cachedEvents"
    :isUpdating="isUpdating"
    @update:events="updateEvents"
    @create:event="createEvent" />
</b-skeleton-wrapper>
```

Listing 10: Plantilla del componente raíz de nuestra aplicación Vue

6.3.2.2 Componente indicador del mes

El componente `<month-indicator />` muestra el mes y el año seleccionados actualmente, junto con dos botones para avanzar o retroceder en el tiempo. Para establecer la comunicación con el componente padre y con el resto de componentes, recibe dos propiedades reactivas: el mes y el año. Además, este componente emite un evento personalizado que hemos llamado *change-month* y que servirá para actualizar la lista de eventos cuando el usuario cambie de mes.

6.3.2.3 Componente días de la semana

El componente `<days-of-week />` es uno muy básicoo que no tiene ningún tipo de interacción con el resto, únicamente muestra los días de la semana haciendo uso de la directiva v-for de Vue para actualizar el DOM a partir del contenido de un array.

6.3.2.4 Componente cuadrícula de días

El componente `<date-grid />` agrupa todos los días del mes actual mediante la propiedad grid de CSS. Como podemos ver en el código de la figura 11, este componente recibe un evento para actualizar y otro para crear eventos y, al mismo tiempo, estos eventos son transmitidos al padre –usando la función `$emit` de Vue– que será el encargado de actualizar la lista.

```
<div class="date-grid" :style="...">
 <single-date
   v-for="day in daysInMonth" :key="day"
   :day="day" :month="month" :year="year"
   :events="getSingleDateEvents(day)"
   :isUpdating="isUpdating"
   @update:events="$emit('update:events',$event)"
   @create:event="$emit('create:event',$event)">
 </single-date>
</div>
```

Listing 11: Plantilla del componente date-grid

6.3.2.5 Componente fecha única

Por último, tenemos el componente `<single-date />` que es el encargado de mostrar los detalles de los eventos disponibles para un día en concreto, así como de permitir la creación de nuevos eventos o modificar los existentes. Para visualizar toda esta información, usaremos el componente `b-popover` de la librería *BootstrapVue* que nos permite crear una ventana flotante que solo se muestra al seleccionar un día en concreto. En la figura 11 podemos ver el resultado de este formulario para editar los detalles del evento.

7 Resultados

En este apartado expondremos los resultados obtenidos a lo largo del desarrollo del proyecto.

7.1. Web

Para empezar tenemos el sitio web, del qual hemos podido desarrollar una parte importante hasta ahora.

En la figura 4 podemos ver la portada de nuestra aplicación web, esto es lo primero que ven los usuarios al entrar al sitio. Importante mencionar el botón de arriba a la derecha "*Àrea soci*", que nos lleva a un subdominio en el que está instalado Nextcloud.

Seguidamente tenemos un apartado en el que se detacan los valores más significativos del priorat. Como se puede ver en las figuras 5, 6, 7 y 8, estos valores se ajustan perfectamente a cualquier tamaño de pantalla, sin necesidad de usar CSS *media queries* en ningún momento.

Otro apartado importante es el de últimas notícias, como podemos ver en la figura 9. Aquí tenemos por un lado los cuatro últimos posts publicados en el blog, con un enlace a una página donde se puede leer el artículo completo, y por otro lado los últimos anuncios, que pueden provenir de WordPress o bien de Nextcloud.

No obstante, todavía queda trabajo por realizar en la web que se ha visto retrasado principalmente por el hecho de que la comunicación con el cliente, así como el diseño y validación de los prototipos ha tomado mucho más tiempo de lo previsto inicialmente. Por ese motivo, nos hemos centrado en tener el poco contenido que hay disponible actualmente lo mejor posible y hemos avanzado la fase de pruebas responsive. También hemos podido avanzar en aspectos relacionados con la accesibilidad web y la compatibilidad entre diferentes navegadores o dispositivos.

Para acabar con este apartado, destacar que ya hemos conseguido migrar el sitio web desde nuestro entorno local a un entorno de pruebas desde el cual el cliente podrá ir viendo el progreso y hacer comentarios.

7.2. Extranet: Nextcloud

En el caso de Nextcloud, una vez instalado y ejecutado la configuración inicial, ya teníamos una extranet lista para ser utilizada en un entorno de producción. Quedaremos a la espera por si el cliente nos pide hacer algunos ajustes en esta herramienta para más adelante.

7.3. Extranet: Plugins WordPress

Como hemos podido ver en la figura 9, ya tenemos un plugin de tablón de anuncios completamente funcional. Este plugin permite configurar fácilmente algunos parámetros como el límite de notícias a mostrar, o si queremos obtener los anuncios de la base de datos de WordPress o de la API de Nextcloud.

En el caso del calendario de eventos, hemos conseguido integrar Vue en un plugin que permite añadir, modificar y eliminar eventos desde el panel de administración de WordPress, para luego mostrarlos en el frontend. Podemos ver la interfaz del calendario en la figura 10

8 Conclusiones

Como hemos podido ver a lo largo de este artículo, el desarrollo de una aplicación web con WordPress puede llegar a ser muy interesante. No se trata únicamente de instalar un tema y ajustar el contenido como puedan pensar algunas personas, sino que hemos usado la API de WordPress para desarrolladores para crear una aplicación totalmente a medida y con funcionalidades que un tema predefinido no nos podría ofrecer.

Por otro lado, hemos conseguido integrar existosamente una herramienta de trabajo colaborativo –Nextcloud– con el sitio web mediante una API REST, tal y como nos propusimos al inicio del proyecto.

Como posibles extensiones futuras de este proyecto, creemos que sería posible conseguir una mayor integración entre la web y la extranet, por ejemplo, que los usuarios de una plataforma puedan entrar en la otra con el mismo usuario y contraseña. También deberíamos acabar de desarrollar los apartados que faltan de la web en cuanto recibamos los diseños y contenidos definitivos por parte del cliente y, una vez todo listo, hacer la migración a un entorno de producción.

Para acabar, consideramos que el desarrollo del proyecto ha sido satisfactorio, ya que en general hemos cumplido con los objetivos marcados –aunque hemos necesitado hacer algunos ajustes en la planificación– y las opiniones del cliente han sido muy satisfactorias hasta el momento.

Agradecimientos

Me gustaría agradecer a mi familia por el apoyo que me han dado estos años, a mi tutor Aitor Alsina por los consejos aportados y a Marc Hernández –director de La Tempesta– por darme la oportunidad de usar un proyecto real como trabajo de fin de grado.

Referencias

[1] K. Kaplan, “Why Every Business Needs A Website”, Forbes, https://www.forbes.com/sites/theyec/2020/02/03/why-every-business-needs-a-website/. 2020.

[2] Agencias Priorat, “El Priorat quiere volver a presentar su candidatura a Patrimonio Mundial de la Unesco”, La Vanguardia, https://www.lavanguardia.com/local/tarragona/20210427/7359228/priorat-quiere-volver-presentar-candidatura-patrimonio-mundial-unesco-2022.html. 2021.

[3] “Usage statistics and market share of WordPress”, W3Techs, https://w3techs.com/technologies/details/cm-wordpress. 2022.

[4] “The Free WordPress Theme Framework For Handcrafted Sites”, Understrap, https://understrap.com/. 2022.

[5] S. Lyndersay, “The future of Internet Explorer on Windows 10 is in Microsoft Edge”, Windows blog, https://blogs.windows.com/windowsexperience/2021/05/19/the-future-of-internet-explorer-on-windows-10-is-in-microsoft-edge/. 2021.

[6] “NPM and the Build Process”, Understrap, https://docs.understrap.com/#/understrap-child/npm. 2022.

[7] “Plugin Handbook”, WordPress, https://developer.wordpress.org/plugins/. 2022.

[8] M. Futari, “Yes, this is how to use Vue JS with WordPress in 3 unique ways”, ITNEXT, https://itnext.io/yes-this-is-how-to-use-vue-js-with-wordpress-in-3-unique-ways-7a676c085bfe. 2021.

[9] “Local WordPress development made simple”, Local, https://localwp.com/. 2022.

[10] “¿Qué es el Subsistema de Windows para Linux?”, Microsoft, https://docs.microsoft.com/es-es/windows/wsl/about. 2022.

[11] “Grid system”, Bootstrap, https://getbootstrap.com/docs/5.2/layout/grid/. 2022.

[12] “The Loop”, WordPress Codex, https://codex.wordpress.org/The_Loop.

[13] “G201: Giving users advanced warning when opening a new window”, M. Cooper (W3C), A. Kirkpatrick (Adobe Systems Inc.), J. O Connor (InterAccess) https://www.w3.org/TR/WCAG20-TECHS/G201.html. 2016.

[14] “WordPress Plugin Boilerplate Generator”, E. Chavez, https://wppb.me/. 2014.

[15] “A simple regex-based Markdown parser in PHP”, J. Broadway, https://github.com/jbroadway/slimdown. 2013.

[16] “Introduction to Vue.js”, Vue.js, https://vuejs.org/guide/introduction.html.

Apéndice

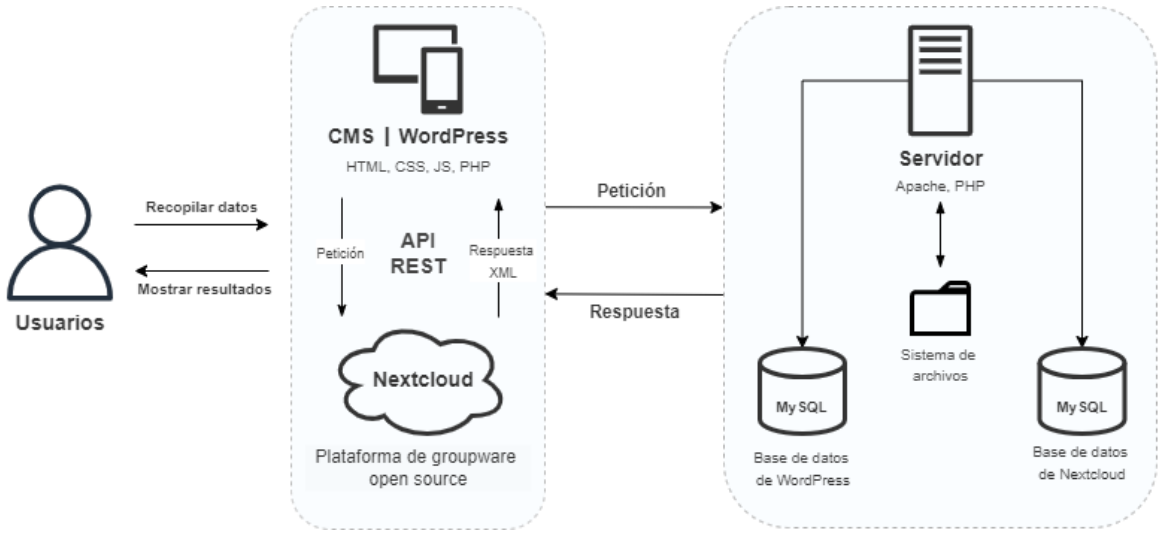

Fig. 1: Arquitectura de nuestra aplicación web

```
prioritat-LaTempesta/
├── css/
├── fonts/
├── global-templates/
├── js/
├── node_modules/
├── loop-templates/
├── page-templates/
│   ├── homepage-template.php
│   └── ...
├── src/
│   ├── build/
│   ├── js/
│   └── sass/
│       ├── assets/
│       ├── theme/
│       │   ├── _child_theme.scss
│       │   └── _child_theme_variables.scss
│       │
│       └── child-theme.scss
├── .gitignore
├── functions.php
├── header.php
├── package.json
├── screenshot.png
└── style.css
understrap/
```

Fig. 2: Estructura de nuestro tema WordPress

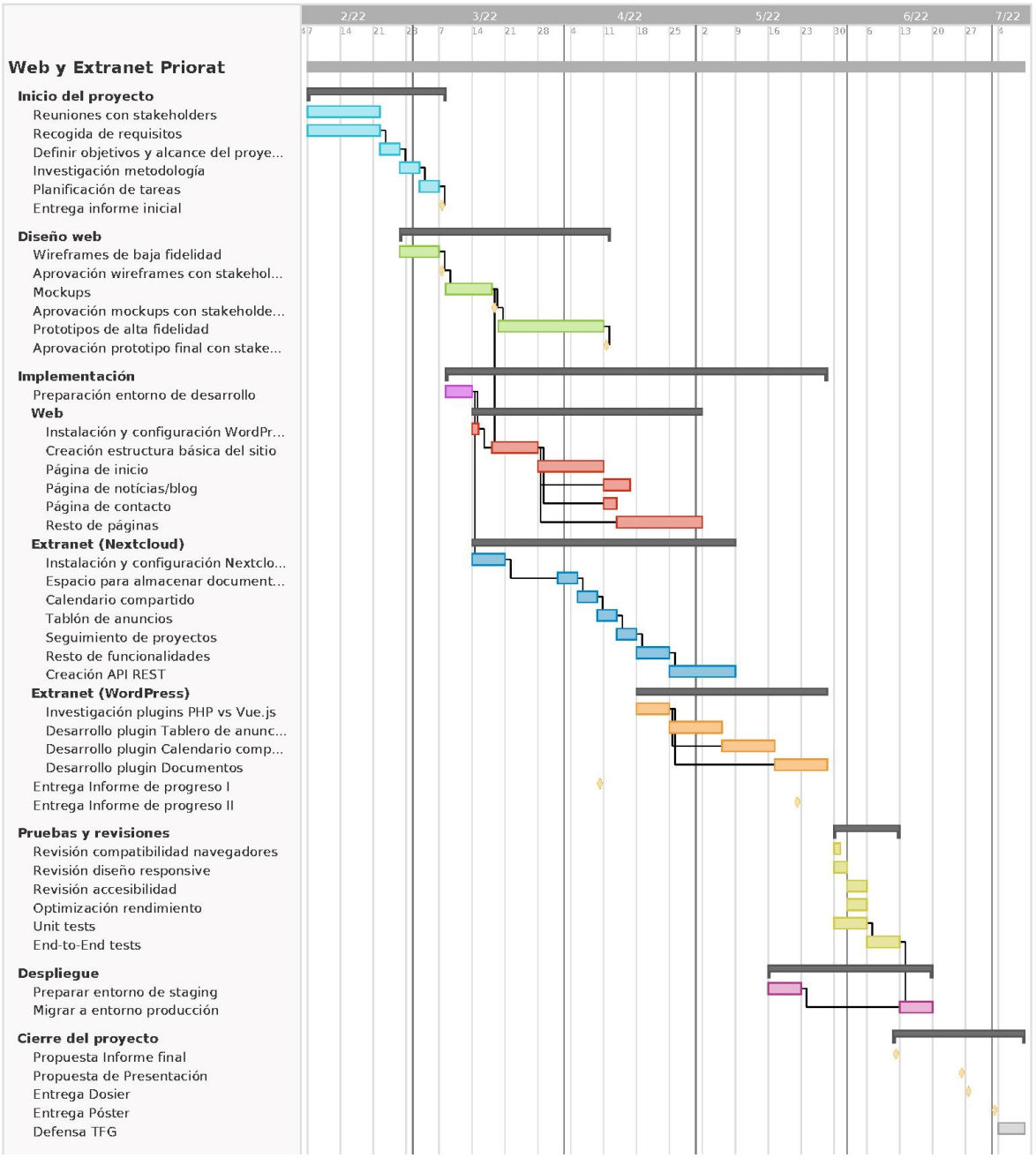

Fig. 3: Planificación inicial del proyecto

Fig. 4: Portada del sitio web

Fig. 5: Sección de valores en un ordenador de 1280px de ancho

Fig. 6: Sección de valores en una tablet de 768px de ancho

Fig. 7: Sección de valores en un dispositivo de 576px de ancho

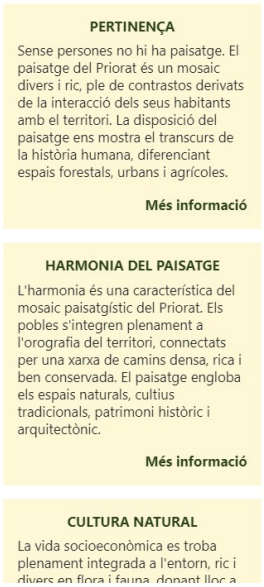

Fig. 8: Sección de valores en un móvil de 320px de ancho

Fig. 9: Sección de últimas noticias y anuncios

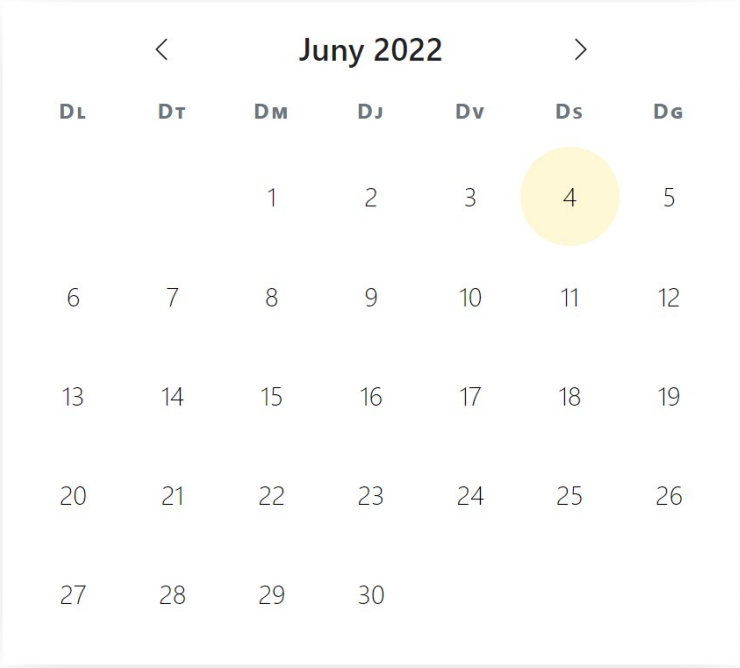

Fig. 10: Interfaz del calendario

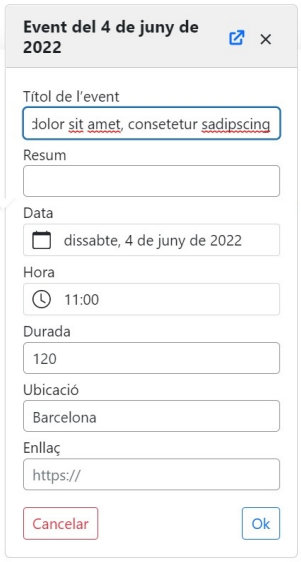

Fig. 11: Ventana flotante que permite modificar los detalles de un evento