



Ingeniería Web - Proyecto Web Colaborativo

Título: Book Store - Diseño e implementación de una librería online

Curso: 4º Grado en ADE + Ingeniería Informática (1º semestre)

Asignatura: Ingeniería Web

Estudiantes:

Lorea Intxausti Oregi
Naroa Jauregi Amiano
Ainhoa Jauregiberri Illarramendi
Nora Royo Iturrioz

Grupo: IW-01

Profesor: Beñat Galdós

Facultad de Deusto Business School

Donostia - San Sebastián, 13 de enero de 2022

Resumen

Durante este proyecto hemos recreado una página web donde se muestra información sobre diferentes libros, autores y editoriales mediante la librería online “Book Store”. Para ello, será necesaria la identificación del mismo iniciando sesión en la plataforma. Una vez dentro, los usuarios podrán ver los detalles de cada uno de los libros, de los autores y de las editoriales en función de su interés. Asimismo, podrán interactuar con la página si deciden valorar con una puntuación del 0-5 algún libro.

Entre las funcionalidades adicionales implementadas, nos ha parecido interesante emplear la interacción enriquecida con el cliente con JavaScript, el uso de vistas basadas en clases, el uso de formularios, personalizar la aplicación para diferentes usuarios y el multilingüismo para poder disponer de la información en más de un idioma.

Por otro lado, hemos desarrollado una agenda de VUE con el objetivo de obtener una agenda de contactos y su gestión. Además, hemos trabajado con VueFire los datos de la agenda en Firebase. Del mismo modo, hemos incluido capacidades sociales en Twitter y Facebook. Además, también hemos empleado las capacidades micro-semánticas de JSON.

Descriptores

- Django
- Editorial
- Libro
- Autor
- Vue

Índice

EXPLICACIÓN GENERAL DEL PROYECTO DJANGO	2
1.1. DESCRIPCIÓN DE LOS ESCENARIOS Y FUNCIONALIDADES BÁSICAS PLANTEADAS	3
1.2. DESCRIPCIÓN DEL MODELO DE DATOS DISEÑADO	8
1.3. DESCRIPCIÓN DE LAS PLANTILLAS UTILIZADAS PARA LAS VISTAS (DESCRIPCIÓN DE LA ESTRUCTURA HTML Y ESTILOS CSS)	9
1.4. DESCRIPCIÓN DE LA ESTRUCTURA DEL PROYECTO WEB (FICHEROS CORRESPONDIENTES A CADA COMPONENTE DEL PATRÓN MVC)	12
1.5. DESCRIPCIÓN DE LAS FUNCIONALIDADES AÑADIDAS IMPLEMENTADAS	13
1.5.1. Interacción enriquecida en el cliente con JavaScript	14
1.5.2. Uso de vistas basadas en clases	14
1.5.3. Uso de formularios en la aplicación pública	14
1.5.4. Personalizar la aplicación de administración	14
1.5.5. Multilingüismo - i18n	16
EXPLICACIÓN GENERAL DEL PROYECTO VUE	16
2.1. DESCRIPCIÓN DE LA ESTRUCTURA DEL PROYECTO	16
2.2. DESCRIPCIÓN DE LAS FUNCIONALIDADES AÑADIDAS IMPLEMENTADAS	19
2.2.1. Persistir con VueFire los datos de Agenda en Firebase	19
2.2.2. Incluir capacidades sociales (Twitter, Facebook)	25
2.2.3. Incluir capacidades micro-semánticas y de la web de datos	27
CONCLUSIONES	30
BIBLIOGRAFÍA	31

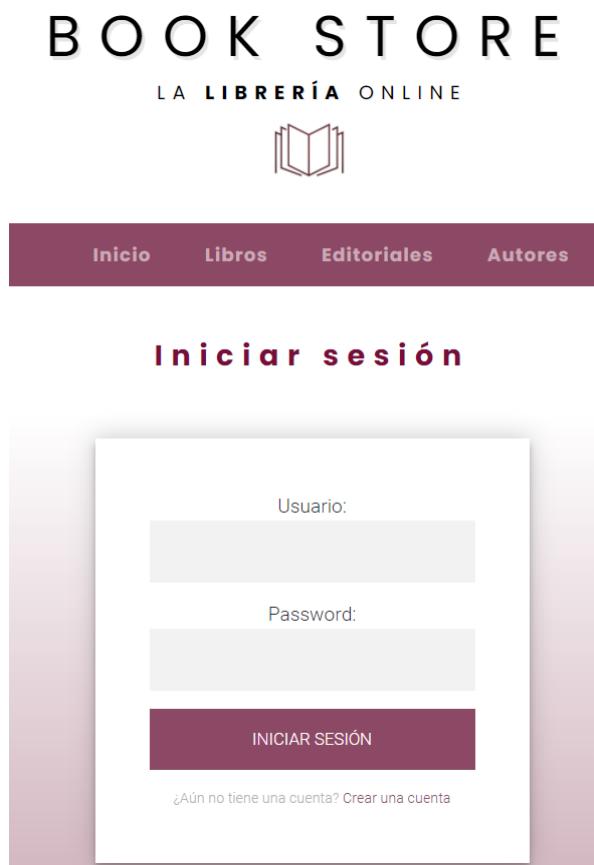
1. EXPLICACIÓN GENERAL DEL PROYECTO

DJANGO

1.1. DESCRIPCIÓN DE LOS ESCENARIOS Y FUNCIONALIDADES BÁSICAS PLANTEADAS

Primero de todo quisiéramos presentar nuestro proyecto: La librería “Book Store”. Este consiste en ilustrar una página web de una librería online donde el usuario podrá visualizar las últimas novedades de los editoriales, la lista de libros y los detalles de cada uno de ellos, la lista de autores con los detalles respectivos y las dos editoriales disponibles.

En lo que respecta al inicio de la web, lo primero que hay que hacer es identificarse mediante el nombre de usuario y contraseña.

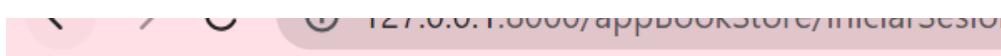


El usuario introducirá su nombre de usuario y contraseña en esta página. Si el usuario es incorrecto, saltará un aviso de este tipo:



El usuario es incorrecto. Inténtelo de nuevo.

Por el contrario, si la contraseña introducida es incorrecta el error será el siguiente:



La contraseña es incorrecta. Inténtelo de nuevo.

R e g i s t r a r s e

Nombre:

Usuario:

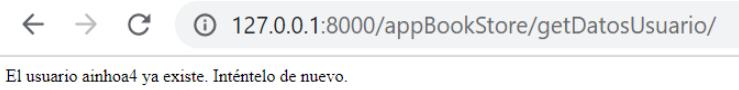
Password:

CREAR

[¿Ya tienes una cuenta? Iniciar sesión](#)

Por el contrario, si el usuario no dispone de una cuenta, tendrá la opción de poder crear una haciendo clic en “crear una cuenta”. Para ello, deberá introducir su nombre, el nombre de usuario y la contraseña. Si el nombre de usuario es incorrecto porque ya existe uno con ese nombre, saltará un aviso.

A continuación, podremos observar un ejemplo de error que se nos mostrará en la pantalla en el caso de que intentemos registrarnos con un nombre de usuario ya existente.



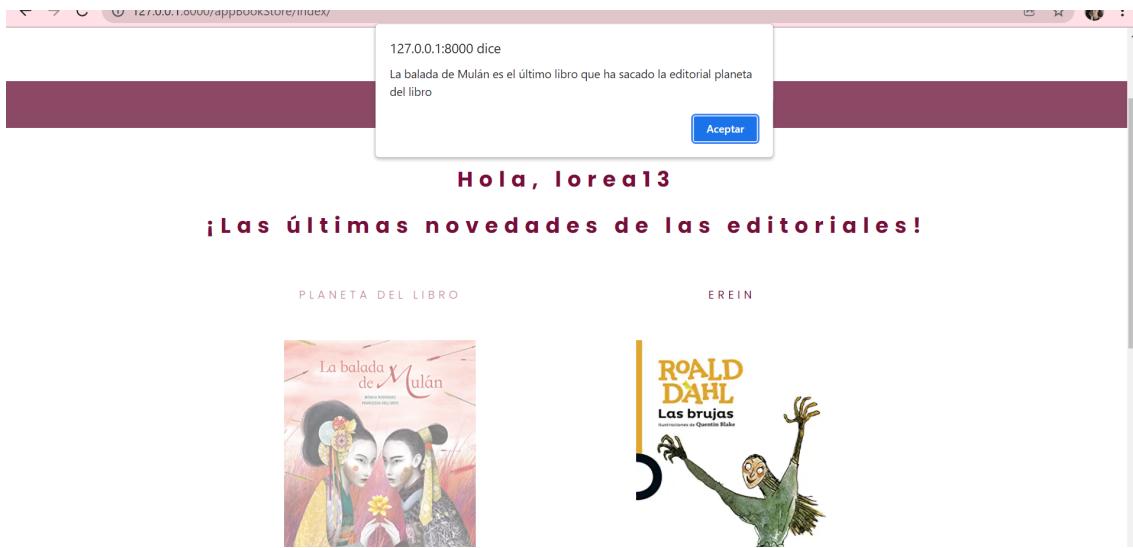
Una vez registrado el nuevo usuario, volverá a la página de inicio donde tendrá que iniciar sesión.

Si navegamos por la barra de navegación, esta incluye 4 apartados: inicio, libros, editoriales y autores. La página principal, la de inicio, mostrará el último libro publicado de cada editorial. Es decir, ordenamos los libros por fecha de publicación y mostramos solo el más reciente.

En la siguiente captura adjunta podemos observar que el último libro publicado por la editorial “Planeta del Libro” es “La balada de Mulán” y por la editorial “Erein” es “Las Brujas”.

The screenshot displays the homepage of the Book Store website. The header features the text "BOOK STORE" and "LA LIBRERÍA ONLINE" above a stylized book icon. A navigation bar below the header includes links for "Inicio", "Libros", "Editoriales", and "Autores". The main content area has a dark red background with white text. It greets the user with "Hola, mikel9" and displays the message "¡Las últimas novedades de las editoriales!". Two book covers are shown side-by-side: "La balada de Mulán" by Planeta del Libro and "Las brujas" by Erein. The Planeta del Libro cover features two characters in traditional Chinese attire. The Erein cover features a green, skeletal character with arms raised.

Cuando movemos el cursor por encima de los libros, nos aparece que es la última publicación de la editorial correspondiente. Así:



Al contrario, si hacemos click en libros, nos aparecerá el listado de todos ellos, lo mismo ocurrirá con autores y editoriales. En las imágenes de a continuación, os mostramos las 3 listas.

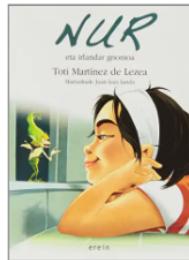
Listado de los libros

¡Echa un vistazo a todos los libros que tenemos!



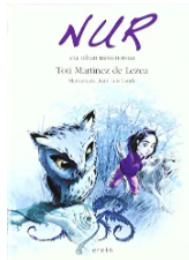
Harry Potter y la piedra filosofal

4,11 ★ / 5 ★



Nur 1

3,00 ★ / 5 ★



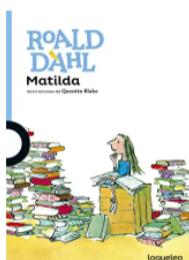
Nur 2

3,33 ★ / 5 ★



Charlie y la fábrica de chocolate

4,00 ★ / 5 ★



Matilda

3,50 ★ / 5 ★



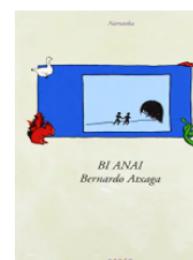
Los 5

3,50 ★ / 5 ★



La casa de los espíritus

3,25 ★ / 5 ★



Bi anai

4,75 ★ / 5 ★

Listado de las editoriales

¿Por cuál de las editoriales te decantas?



PlanetaDeLibro



Erein

Listado de los autores

¡Echa un vistazo a todos los autores!



J. K. Rowling



Toti Martínez de Lezea



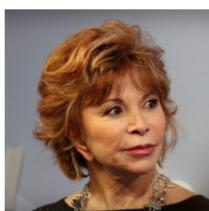
Roald Dahl



Enid Blyton



Bernardo Atxaga



Isabel Allende

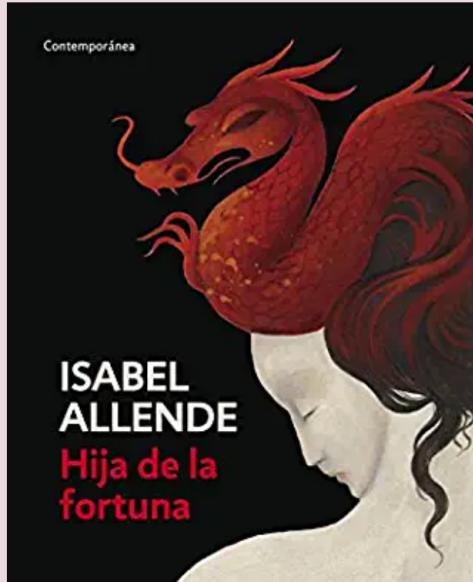


Claude Voilier



Eileen Soper

En la página **detalles libros**, cada libro dispondrá del título, la fecha en el que se publicó, el/los autor/es, una breve sinopsis, la media de valoraciones que han hecho los usuarios sobre este libro en estrellas, el número de valoraciones y los comentarios. Los usuarios tendrán la opción de valorar cada libro del 0-5 introduciendo también una breve valoración de texto. Para hacer esa valoración hemos introducido un formulario al final de la página a la que podemos ir de una manera rápida mediante el botón valorar.



Hija de la fortuna

Número de libro:
Este es el libro número 10 de la biblioteca Bookstore.

Fecha de publicación:
Fue publicado el 15-02-2018.

Autor/es:
• Isabel Allende

Sinopsis:
Un retrato palpitante de una época marcada por la violencia y la codicia, con entrañables personajes. Eliza Sommers es una joven chilena que vive en Valparaíso en 1849, el año en que se descubre oro en California. Su amante, Joaquín Andieta, parte hacia el norte decidido a encontrar fortuna, y ella decide seguirlo.

Valorar

A continuación, podemos leer los diferentes comentarios que tiene el correspondiente libro. En el mismo comentario aparecerá el nombre del usuario que ha hecho el comentario, el número de estrellas con la que ha valorado el libro dicho usuario y el texto del comentario, es decir, la opinión que tiene acerca del libro.

¡Echa un vistazo a los comentarios!

pablo3 5 ★ / 5 ★

Le doy un diez. Maravilloso.

nora7 2 ★ / 5 ★

Libro entretenido pero no mucho más. Hubo momentos en que se me hizo pesado. Recomendable si buscas una historia entretenida y larga.

jon10 2 ★ / 5 ★

Con mucho esfuerzo llegué a la segunda parte de esta novela, pero no pude continuar, me rendí, no me enganchó.

Para finalizar con la página que muestra los detalles de cada libro, existe la opción de dejar una valoración. Dicha valoración se realiza mediante un formulario que hemos mencionado anteriormente y lo podemos observar en la siguiente captura de pantalla. El título tiene un efecto de blink, por lo que parpadea, para que de este modo destaque el la vista del usuario.

¡victor8, anímate a dejar tu comentario!

Puntuacion:

0

Texto:

CALIFICAR



En lo que respecta a los **autores**, estos dispondrán del nombre, la biografía y la lista de libros que han escrito y que están disponibles en la librería Book Store. Se podrá clicar en cada libro e iremos a la página detalles libro de dicho libro.

D e t a l l e d e l a u t o r



Totí Martínez de Lezea

Número de identificación:
Este es el autor número 2 de la biblioteca Bookstore.

Biografía:
Esperanza Martínez de Lezea nació en 1949 en Vitoria. Una vez finalizados sus estudios de bachillerato se desplazó a la comarca guipuzcoana de Goierri a aprender euskera durante un año. Estudió cuatro años de francés en Francia, para acto seguido viajar a Inglaterra durante tres años para estudiar inglés, así como alemán en Alemania durante dos años más.

Libros escritos:

Para finalizar, y en lo que respecta a las editoriales, estas dispondrán del nombre de la editorial, una breve explicación de la editorial y la lista de libros que pertenecen a la misma.

D e t a l l e d e l a e d i t o r i a l



PlanetaDeLibro

Número de la editorial:
Esta es la editorial número 1 de la biblioteca Bookstore.

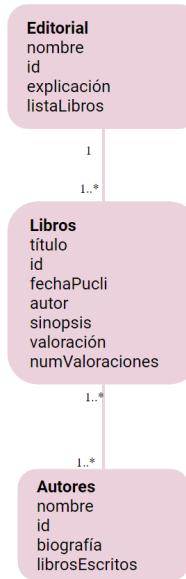
Explicación:
Editorial Planeta, embrío de lo que es hoy el Grupo Planeta, se fundó en 1945, hace más de sesenta años. Es la editorial de prestigio con mayor influencia en el mundo de habla hispana. Anualmente convoca el Premio Planeta, el más destacado de los certámenes españoles, junto a otros de gran relevancia literaria.

Libros de la editorial:

1.2. DESCRIPCIÓN DEL MODELO DE DATOS DISEÑADO

Una vez conocemos cómo funciona nuestro proyecto, también sabemos que disponemos de tres clases importantes dentro de la librería de Book Store. Las tres clases serían: editoriales, libros y autores.

Cada editorial dispone de X libros, pero cada libro pertenece a una editorial. Por otro lado, cada libro puede estar escrito por uno o varios autores y cada autor escribe X libros.



1.3. DESCRIPCIÓN DE LAS PLANTILLAS UTILIZADAS PARA LAS VISTAS (DESCRIPCIÓN DE LA ESTRUCTURA HTML Y ESTILOS CSS)

En cuanto a las plantillas utilizadas para las vistas, en nuestro proyecto no hemos hecho uso de plantillas externas de HTML y CSS, por lo que no disponíamos de una estructura de ficheros con archivos HTML, archivos CSS, JS, fuentes o imágenes. Por el contrario, nuestro proyecto ha sido desarrollado por elaboración propia. A excepción de que hemos introducido una librería llamada bootstrap-social.css sobre todo para los iconos del footer.

Comenzaremos explicando base.html, que es el archivo donde hemos decidido meter el header y el footer que será común en todas las páginas. Las demás páginas heredarán de base.html. El archivo base.html comienza con la parte de head. En ella se declaran los metadatos y continúa con título y los stylesheets.

```

[|] DOCTYPE html
<html>
  <head>
    <meta charset="UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="descripción" content="Página inicial de nuestra biblioteca online">
    <meta name="author" content="IW-Grupo1">
    <meta name="robots" content="index">
    <title>Book Store</title>
    {% load static %}
    {% load i18n %}
    <link rel="stylesheet" type="text/css" media="screen" href="{% static 'css/main.css' %}">
    <link rel="stylesheet" type="text/css" media="screen" href="{% static 'css/index.css' %}">
    <link rel="stylesheet" type="text/css" media="screen" href="{% static 'css/detalles.css' %}">
    <link rel="stylesheet" type="text/css" media="screen" href="{% static 'css/inicio.css' %}">
    <link rel="stylesheet" type="text/css" media="screen" href="{% static 'css/bootstrap-social.css' %}">
    <link rel="apple-touch-icon" sizes="57x57" href="{% static 'favicon/apple-icon-57x57.png' %}">
    <link rel="apple-touch-icon" sizes="60x60" href="{% static 'favicon/apple-icon-60x60.png' %}">
    <link rel="apple-touch-icon" sizes="72x72" href="{% static 'favicon/apple-icon-72x72.png' %}">
    <link rel="apple-touch-icon" sizes="76x76" href="{% static 'favicon/apple-icon-76x76.png' %}">
    <link rel="apple-touch-icon" sizes="114x114" href="{% static 'favicon/apple-icon-114x114.png' %}">
    <link rel="apple-touch-icon" sizes="120x120" href="{% static 'favicon/apple-icon-120x120.png' %}">
    <link rel="apple-touch-icon" sizes="144x144" href="{% static 'favicon/apple-icon-144x144.png' %}">
    <link rel="apple-touch-icon" sizes="152x152" href="{% static 'favicon/apple-icon-152x152.png' %}">
    <link rel="apple-touch-icon" sizes="180x180" href="{% static 'favicon/apple-icon-180x180.png' %}">
    <link rel="icon" type="image/png" sizes="192x192" href="{% static 'favicon/android-icon-192x192.png' %}">
    <link rel="icon" type="image/png" sizes="32x32" href="{% static 'favicon/favicon-32x32.png' %}">
    <link rel="icon" type="image/png" sizes="96x96" href="{% static 'favicon/favicon-96x96.png' %}">
    <link rel="icon" type="image/png" sizes="16x16" href="{% static 'favicon/favicon-16x16.png' %}">
    <link rel="manifest" href="{% static 'favicon/manifest.json' %}">
    <meta name="msapplication-TileColor" content="#ffffff">
    <meta name="msapplication-TileImage" content="/ms-icon-144x144.png">
    <!-- Font Awesome icons (free version) -->
    <script src="https://use.fontawesome.com/releases/v5.1.1/js/all.js" crossorigin="anonymous"></script>
    <!-- Google fonts -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css?family=Poppins:wght@100,400&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Poppins:400,700" rel="stylesheet" type="text/css" />
    <link href="https://fonts.googleapis.com/css?family=Poppins:400,700,400italic,700italic" rel="stylesheet" type="text/css" />
    <link href="https://fonts.googleapis.com/css?family=Poppins:400,100,300,700" rel="stylesheet" type="text/css" />
    <!-- Core theme CSS (includes Bootstrap) http://127.0.0.1:8000/appBookStore-->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-E4BQHlFJAWO1zjgA3CTCgGGTtG7El+qf4fFfLj+VWZLqfXfKq6PwQ&lt;!-->">
    <link rel="icon" href="icono.ico" />

```

A continuación, empieza el `<body>`. Lo primero que hemos hecho ha sido insertar el código necesario para emplear el multilingüismo y dar la opción de poder cambiar de idioma al usuario. Esto es lo que se visualiza en la web:

[español \(es\)](#) ▾ [Change](#)

Y este es el código para crearlo:

```

<body>
<body class="fondo">
  <form action= "{% url 'set_language' %}" method="post"> {% csrf_token %}>
    <input name="next" type="hidden" value= "{{redirect_to}} ">
    <select name = "language">
      {% get_current_language as LANGUAGE_CODE %}
      {% get_available_languages as LANGUAGES %}
      {% get_language_info_list for LANGUAGES as languages%}
      {% for language in languages%}
        <option value="{{language.code}}" {% if language.code == LANGUAGE_CODE %} selected {%endif%}>
          | {{ language.name_local }} ({{ language.code }})
        </option>
      {% endfor%}
    </select>
    <input type="submit" value="Change" />
  </form>

```

Después, empieza el `<header>`. Esta sección del header contiene el título de la web junto al logo del mismo y el menú de navegación. Hemos introducido id-s con nombres de op1, op2, op3 y op4, para después poder marcar en las páginas que heredan en qué página se encuentra el usuario mediante el css. Este es el ejemplo:



Y este es el código de base.html donde se ven las diferentes opciones.

```
<header>
    <section>
        <div>

            <h1 class="estiloTitulo" ><a href="#">{% url 'index' %}>BOOK STORE</a></h1>
            <h2 class="estiloSubtitulo"><a href="#">{% url 'index' %}>LA <b>LIBRERÍA</b> ONLINE</a></h2>
            <a href="#">{% url 'index' %}>{% static 'irudiak/logoBS.png' %}" alt="" Title="logo"></a>

            <ul class="navegacion menu">
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="#">{% url 'index' %}" id="op1">{% trans "Inicio" %}</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">{% url 'listaLibros' %}" id="op2">Libros</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">{% url 'listaEditoriales' %}" id="op3">Editoriales</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">{% url 'listaAutores' %}" id="op4">Autores</a>
                </li>
            </ul>

        </div>
    </section>
</header>
```

A continuación, comienza el contenido de la página, que varía según el archivo. Para ello, hemos empleado el siguiente código. Todo el contenido que queremos que aparezca en la parte del contenido, debe ir entre el código en blanco.

```
<div id="subtitulo">
    {% block titulo %}{% endblock %}
</div>
    {% block contenido %}{% endblock %}

    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
```

Para finalizar, disponemos del `<footer>`. Nuestro footer cuenta con dos bloques: el bloque principal dividido en 3 columnas y el bloque de abajo que incluye el copyright. La primera columna nos da la información necesaria de la librería, la segunda contiene los enlaces a las diferentes redes sociales y la tercera las opciones de contacto que dispone la librería.

Este es el código del footer:

```
<footer class = "Footer">
    <div class="todoFooter">
        <div class="cuerpo">
            <div class="columna1">
                <h2> Más información </h2>
                <p>La librería Book Store dispone de información de distintos libros, autores y editoriales.</p>
                <p>Si desea saber más contacta con nosotros mediante correo electrónico, redes sociales o el teléfono de atención al cliente.</p>
            </div>
            <div class="columna2">
                <h2>Redes sociales</h2>
                <div class="iconos">
                    <a class="btn btn-social mx-2" href="https://mobile.twitter.com/casadellibro" target="_blank"><i class="fab fa-twitter"></i></a>
                    <label> <a href="https://mobile.twitter.com/casadellibro" target="_blank">Síguenos en Twitter </a></label>
                </div>
                <div class="iconos">
                    <a class="btn btn-social mx-2" href="https://es-la.facebook.com/casadellibro/" target="_blank"><i class="fab fa-facebook"></i></a>
                    <label> <a href="https://es-la.facebook.com/casadellibro/" target="_blank">Síguenos en Facebook </a></label>
                </div>
                <div class="iconos">
                    <a class="btn btn-social mx-2" href="https://www.instagram.com/casadellibro/" target="_blank"><i class="fab fa-instagram"></i></a>
                    <label> <a href="https://www.instagram.com/casadellibro/" target="_blank">Síguenos en Instagram </a></label>
                </div>
            </div>
            <div class="columna3">
                <h2> Información Contactos </h2>
                <div class="fila2">
                    <a class="btn btn-social mx-2" href="tel:+34911793463" target="_blank"><i class="fa fa-phone"></i></a>
                    <label> <a href="tel:+34911793463" target="_blank">911 79 34 63</a>
                    </label>
                </div>
                <div class="fila2">
                    <a class="btn btn-social mx-2" href="mailto:alameda@casadellibro.com" target="_blank"><i class="fa fa-envelope"></i></a>
                    <label> <a href="mailto:alameda@casadellibro.com" target="_blank">alameda@casadellibro.com</a>
                    </label>
                </div>
                <div class="fila2">
                    <a class="btn btn-social mx-2" href="https://www.google.com/maps?q=casa+del+libro+bilbao&um=1&ie=UTF-8&s=X&ved=2ahUKEwitjprIsd70AhU4AWMBHSKeCbIQ_AUoAxoECAIQAw" target="_blank"><i class="fa fa-home"></i>
                    <label> <a href="https://www.google.com/maps?q=casa+del+libro+bilbao&um=1&ie=UTF-8&s=X&ved=2ahUKEwitjprIsd70AhU4AWMBHSKeCbIQ_AUoAxoECAIQAw" target="_blank">Urkiixo Zumarkalea, 9, 48009 Bilbo, Bizkaia</a>
                    </label>
                </div>
            </div>
        </div>
        <div class = "cuerpoDeabajo">
            <div class="copyright">
                &copy; Todos los derechos reservados | By Ainhoa, Lorea, Naroa y Nora
            </div>
        </div>
    </div>
</footer>
```

Así es como ha quedado el footer:



1.4. DESCRIPCIÓN DE LA ESTRUCTURA DEL PROYECTO WEB (FICHEROS CORRESPONDIENTES A CADA COMPONENTE DEL PATRÓN MVC)

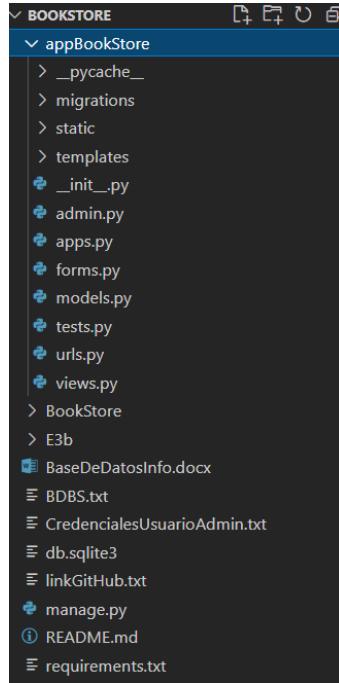
En este siguiente apartado, procederemos a explicar todo lo referido a la estructura de Book Store y también los ficheros correspondientes a cada componente del patrón MVC. Book Store se ubica en el directorio de trabajo donde los proyectos son creados, la cual está definida en PROJECT_HOME, una variable del sistema. El proyecto, de la misma manera está incluido dentro del entorno virtual “entornoBS”:

Si observamos el Explorer en Visual Studio Code en la imagen inferior podemos distinguir las siguientes secciones:

Por un lado, la carpeta “appBookStore” la cual hace referencia a la carpeta de la aplicación. Aquí, podemos encontrar todo lo que tiene que ver con la aplicación como tal, es decir, los archivos HTML -incluidos en la carpeta llamada “templates”-, los archivos CSS de estilos, los archivos JS, los iconos *favicon* e imágenes -incluidos en la carpeta llamada “static”-. Además, como vemos en la imagen los archivos python

también se encuentran dentro, que son `__init__.py`, `admin.py`, `apps.py`, `forms.py`, `models.py`, `tests.py`, `urls.py` y `views.py`.

Por otro lado, la carpeta “BookStore” es la que hace referencia al proyecto.



Si nos centramos en el patrón MVC, los ficheros se clasifican de la siguiente manera:

- El Modelo de los datos lo podemos encontrar en el directorio de la app, concretamente en el archivo python `models.py`.
- Las Vistas también las podemos encontrar en el directorio de la app, en los archivos HTML, dentro de la carpeta “templates”.
- El Controlador, lo podemos encontrar en el directorio de la app, concretamente en el archivo python `views.py`. Este sirve para relacionar los modelos de datos y las vistas.

1.5. DESCRIPCIÓN DE LAS FUNCIONALIDADES AÑADIDAS IMPLEMENTADAS

En esta siguiente sección mencionaremos las funcionalidades añadidas que hemos implementado en Book Store. Primero de todo hemos optado por la interacción enriquecida en el cliente con JavaScript (eventos, efectos, ajax) y sus librerías o frameworks. No solo eso, también hemos implementado , el uso de vistas basadas en clases, el uso de formularios en la aplicación pública, la personalización de la aplicación de administración y el multilingüismo.

1.5.1. Interacción enriquecida en el cliente con JavaScript

Este apartado corresponde con la primera funcionalidad añadida que hemos implementado en nuestro proyecto.

Hemos añadido un efecto para que el título de añadir un comentario parpadee. Para ello, hemos creado una función dentro de script, también se podría haber hecho en un .js. Esta es la función:

```
<script>
  window.setInterval (BlinkIt, 1000);
  var color = "rgba(136, 66, 95, 0.965)";
  function BlinkIt () {
    var blink = document.getElementById ("blink");
    color = (color == "#ffffff")? "rgba(136, 66, 95, 0.965)" : "#ffffff";
    blink.style.color = color;
    blink.style.fontSize='36px';
  }
</script>
<div id="blink"><h3 style="text-align: center;">¡{{usuarioRegistrado}}, anímate a dejar tu comentario!
```

La parte “¡victor8, anímate a dejar tu comentario!” será la parpadeará.



1.5.2. Uso de vistas basadas en clases

Hemos hecho uso de las vistas basadas en clase. Hemos creado un fichero forms.py donde se encuentran diversas clases: UsuarioForm, UsuariosExistentesForm y ValoracionCrear:

```

class UsuarioForm(forms.ModelForm):
    class Meta:
        model = Usuario
        fields = ("__all__")
        exclude = ['notificar']

class UsuariosExistentesForm(forms.ModelForm):
    class Meta:
        model = Usuario
        fields = ("__all__")
        exclude = ['notificar', 'nombre']

class ValoracionCrear(forms.ModelForm):
    class Meta:
        model = Valoracion
        fields = ['puntuacion', 'texto']
        db_constraints = {
            'puntuacion': 'check (puntuacion<6)'
        }
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['puntuacion'].widget.attrs.update({
            'class': 'form-puntuacion'
        })
        self.fields['texto'].widget.attrs.update({
            'class': 'form-texto'
        })

```

Accederemos a estas clases desde views.py. Al recargar la página registrarse, por ejemplo, llamará a UsuarioForm para pasarle todos los campos de la clase Usuario menos la de notificar.

```

def registrarse(request):
    form = UsuarioForm()
    context = {'form' : form}
    return render(request, 'registrarse.html', context)

```

Así es como se verá en la página web:

The screenshot shows a user registration form. It consists of three input fields: 'Nombre:' (Name) with a placeholder, 'Usuario:' (User) with a placeholder, and 'Password:' (Password) with a placeholder. Below these fields is a large red button labeled 'CREAR' (Create). At the bottom of the form, there is a link '¿Ya tienes una cuenta? Iniciar sesión' (Do you have an account? Log in).

Haremos esto mismo con otras clases.

1.5.3. Uso de formularios en la aplicación pública

Hemos insertado tres formularios en la página web: iniciar sesión, registrar usuario y valorar el libro. En el formulario de iniciar sesión, cualquier usuario que esté registrado previamente podrá introducir sus datos y si son correctos entrar en la página web Book Store. Si no dispone de una cuenta podrá crear una cuenta accediendo desde un link que se encuentra en el formulario de iniciar sesión y que te lleva al formulario de registro. Por último, mediante el formulario de valorar el libro, cualquier usuario que accede a la página web de Book Store podrá valorar el libro que desee puntuando el libro de 0 a 5 y añadiendo un comentario.

1.5.4. Personalizar la aplicación de administración

En la siguiente funcionalidad añadida, hemos procedido a personalizar la aplicación de administración, dando soporte a varios roles de usuarios administradores con diferentes permisos. Para ello hemos accedido a <http://127.0.0.1:8000/admin/auth/user/> donde hemos podido observar que tenemos dos usuarios: *Benny96* y *adminDjango*.

NOMBRE DE USUARIO	DIRECCIÓN DE CORREO ELECTRÓNICO	NOMBRE	APELLIDOS	ES STAFF
Benny96	benny96@gmail.com			✓
adminDjango	ainhoaillarramendi@opendeusto.es			✓

A continuación, detallaremos los permisos de cada uno de ellos. En cuanto a *adminDjango* será el superusuario, por lo que se le adjudicarán todos los permisos, podrá visualizar, modificar, añadir, borrar, etc.

Sitio administrativo

The screenshot shows the administrative interface for the APPBOOKSTORE application. It features two main sections:

- AUTENTICACIÓN Y AUTORIZACIÓN** (Authentication and Authorization):
 - Grupos**: Añadir, Modificar
 - Usuarios**: Añadir, Modificar
- Permisos** (Permissions):
 - Activo**: Descripción: Indica si el usuario debe ser tratado como activo. Desmarque esta opción en lugar de borrar la cuenta.
 - Es staff**: Descripción: Indica si el usuario puede entrar en este sitio de administración.
 - Es superusuario**: Descripción: Indica que este usuario tiene todos los permisos sin asignárselos explícitamente.

En lo que respecta a *Benny96*, este será un usuario activo y sólo tendrá permiso para entrar y visualizar los datos del sitio de administración Book Store, como podemos observar en las siguientes imágenes:

The screenshot shows the 'Permisos' (Permissions) section of the administrative interface. It contains three checkboxes:

- Activo**: Descripción: Indica si el usuario debe ser tratado como activo. Desmarque esta opción en lugar de borrar la cuenta.
- Es staff**: Descripción: Indica si el usuario puede entrar en este sitio de administración.
- Es superusuario**: Descripción: Indica que este usuario tiene todos los permisos sin asignárselos explícitamente.

permisos de usuario Disponibles ⓘ

Filtro

permisos de usuario elegidos ⓘ

admin | entrada de registro | Can add log entry
 admin | entrada de registro | Can change log entry
 admin | entrada de registro | Can delete log entry
 appBookStore | autor | Can add autor
 appBookStore | autor | Can change autor
 appBookStore | autor | Can delete autor
 appBookStore | editorial | Can add editorial
 appBookStore | editorial | Can change editorial
 appBookStore | editorial | Can delete editorial
 appBookStore | genero | Can add genero
 appBookStore | genero | Can change genero
 appBookStore | genero | Can delete genero
 appBookStore | idioma | Can add idioma

Selecciona todos ⓘ Eliminar todos ⓘ

Permisos específicos para este usuario. Mantenga presionado "Control", o "Command" en un Mac, para seleccionar más de una opción.

Sitio administrativo

APPBOOKSTORE	
Autors	👁 Vista
Editorials	👁 Vista
Géneros	👁 Vista
Idiomas	👁 Vista
Libros	👁 Vista
Usuarios	👁 Vista
Valoraciones	👁 Vista

AUTENTICACIÓN Y AUTORIZACIÓN	
Grupos	👁 Vista
Usuarios	👁 Vista

1.5.5. Multilingüismo - i18n

Gracias a la funcionalidad añadida del multilingüismo, nuestra web dispone la opción de poder seleccionar el idioma que más se ajuste a tus necesidades. Todos sabemos que para poder avanzar en esta sociedad es obligatorio ajustarse a sus necesidades. Con la variedad de idiomas que se hablan en el mundo ocurre justo esto mismo. Dicho esto, hemos decidido traducir la web en 2 idiomas: en castellano y en inglés.

Para empezar a emplear el multilingüismo lo primero que tuvimos que hacer fue descargar `gettext0.21-iconv1.16-static-64` en nuestros ordenadores. Por otro lado, modificamos el archivo `settings.py` añadiendo el siguiente código.

```
from pathlib import Path
from django.utils.translation import ugettext_lazy as _
import os
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django.middleware.locale.LocaleMiddleware',
]
```

Por otro lado, y como hemos podido aprender durante el trabajo, el multilingüismo funciona gracias a i18n. Por lo tanto, es necesario habilitar el siguiente código especificando el lenguaje original (Español) y los demás idiomas disponibles (Inglés). Todos los idiomas se guardarán en el array de LENGUAJES. Para finalizar, es necesario crear una carpeta LOCALE_PATHS.

```
# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

USE_I18N = True
LANGUAGE_CODE = 'es'
_ = lambda s:s
LANGUAGES = [
    ('es', _("Español")),
    ('en', _("English")),
]
TIME_ZONE = 'UTC'

USE_L10N = True

USE_TZ = True

LOCALE_PATHS = (BASE_DIR, 'locale')
```

En lo que respecta a urls.py, hemos tenido que incluir el siguiente código.

```
from django.contrib import admin
from django.urls import include, path
from django.conf.urls import url
from django.conf.urls.i18n import i18n_patterns
from django.utils.translation import ugettext_lazy as _

urlpatterns = [
    path('appBookStore/', include('appBookStore.urls')),
    path('admin/', admin.site.urls),
    url(r'^i18n/', include('django.conf.urls.i18n')),
]
```

Una vez realizados los cambios anteriores, ya podemos hacer los cambios necesarios para las traducciones en las páginas HTML que queremos traducir. Lo primero que hemos hecho es añadir un formulario en *base.html* donde el usuario podrá escoger el idioma de la página: español o inglés. Por otro lado, en todas las páginas HTML que queramos traducir, debemos incluir `{% load i18n %}` al principio de la página.

```
{% load i18n %}

<form action= "{% url 'set_language' %}" method="post"> {% csrf_token %}
    <input name="next" type="hidden" value= "{{redirect_to}}" />
    <select name = "language">
        {% get_current_language as LANGUAGE_CODE %}
        {% get_available_languages as LANGUAGES %}
        {% get_language_info_list for LANGUAGES as languages%}
        {% for language in languages%}
            <option value="{{language.code}}"{% if language.code == LANGUAGE_CODE %} selected {%endif%}>
                {{language.name_local }} ({{language.code}})
            </option>
        {% endfor%}
    </select>
    <input type="submit" value="Change" />
</form>
```

Una vez estamos en este punto, sólo queda seleccionar el texto que queremos traducir. Para ellos, incluiremos `{% trans "texto" %}` en las partes correspondientes como se muestra en la siguiente imagen.

```
url 'index' %}" id="op1">{% trans "Inicio" %}</a>

p2">{% trans "Libros" %}</a>

id="op3">{% trans "Editoriales" %}</a>

op4">{% trans "Autores" %}</a>
```

Después de haber incluido las estructuras `{% trans %}` se debe ejecutar el siguiente comando, con el idioma indicado en *settings.py*, en nuestro caso el inglés:

```
python manage.py makemessages -l en
```

Tras ejecutarlos se genera `LC_MESSAGES`, donde se encuentran `django.mo` y `django.po`. En `django.po`, se añadirá el código correspondiente de traducción de la app.

```
#: .\appBookStore\templates\base.html:73
msgid "Inicio"
msgstr "Home"

#: .\appBookStore\templates\base.html:76
msgid "Libros"
msgstr "Books"

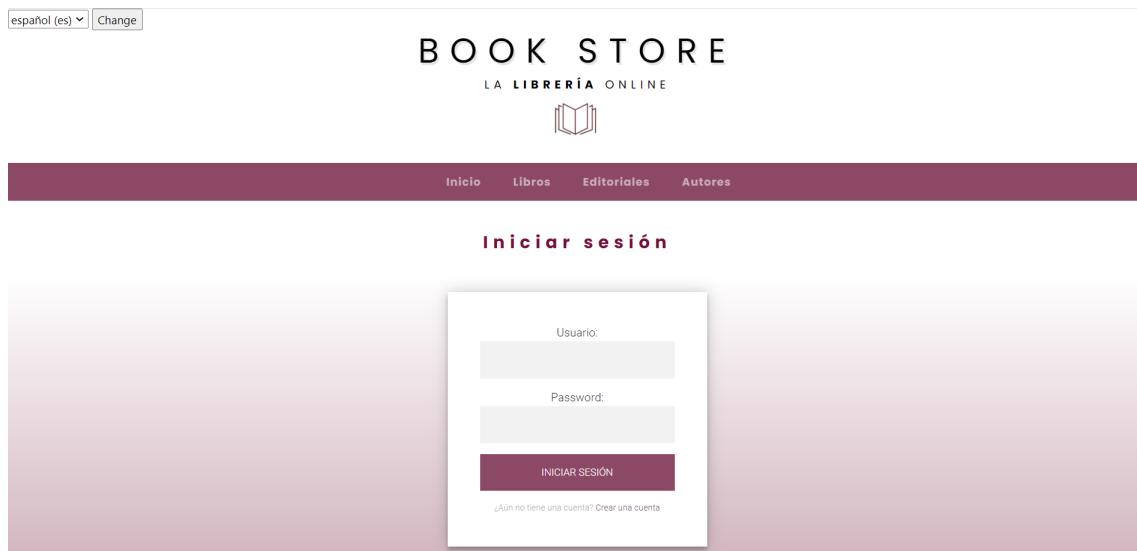
#: .\appBookStore\templates\base.html:79
msgid "Editoriales"
msgstr "Editorials"

#: .\appBookStore\templates\base.html:82
msgid "Autores"
msgstr "Authors"

#: .\appBookStore\templates\base.html:106
msgid "Más información"
msgstr "More information"
```

Una vez añadidas todas las traducciones deseadas, se debe ejecutar el siguiente comando con el fin de que los mensajes compilen y lo visualicemos en Book Store en los dos idiomas:

```
python manage.py compilemessages
```



English (en) ▾ Change

BOOK STORE
LA LIBRERÍA ONLINE


Home Books Editorials Authors

Login

Usuario:

Password:

LOGIN

Don't have an account yet? [Create new account](#)

More information
The Book Store has information on different books, authors and publishers.
If you want to know more, contact us by email, social networks or the customer service phone.

Social networks
 Follow us on Twitter
 Follow us on Facebook
 Follow us on Instagram

Contact us
 911 79 34 63
 alameda@casadelibro.com
 Urkijo Zumarkalea, 9, 48009 Bilbo, Bizkaia

© All rights reserved | By Ainhoa, Lorea, Naroa and Nora

Más información
La librería Book Store dispone de información de distintos libros, autores y editoriales.
Si desea saber más contacta con nosotros mediante correo electrónico, redes sociales o el teléfono de atención al cliente.

Redes sociales
 Síguenos en Twitter
 Síguenos en Facebook
 Síguenos en Instagram

Información Contactos
 911 79 34 63
 alameda@casadelibro.com
 Urkijo Zumarkalea, 9, 48009 Bilbo, Bizkaia

© Todos los derechos reservados | By Ainhoa, Lorea, Naroa y Nora

2. EXPLICACIÓN GENERAL DEL PROYECTO

VUE

2.1. DESCRIPCIÓN DE LA ESTRUCTURA DEL PROYECTO

El proyecto de Vue tiene como objetivo la creación de una agenda donde se puede añadir contacto, eliminar contacto o consultar la lista de los ya existentes.

Para la realización de este proyecto, hemos utilizado la plantilla proporcionada en Alud. Una vez disponíamos de la plantilla, hemos modificado y añadido parte del código que lo detallaremos a continuación.

En primer lugar, hay que añadir VUE en index.html porque este es un framework de Javascript.

```
<!-- Vue -->
<script src="js/vue.js"></script>
```

En segundo lugar, hemos modificado la forma en la que se muestran los contactos. Hemos añadido un mensaje de “No se ha encontrado ningún contacto” para los casos en los que no haya ningún contacto. En caso contrario, los contactos existentes se mostrarán en la página web con el nombre, el email y el número de teléfono. Para ello, se recorrerá un array de los contactos. Por otro lado, nos ha parecido interesante guardar un índice para facilitar el proceso de borrado.

```
<!-- Contactos Grid Section -->
<section id="contactos">
  <div class="container">
    <div class="row">
      <div class="col-lg-12 text-center">
        <h2>Contactos</h2>
        <hr class="star-primary">
      </div>
    </div>

    <div v-if="contacts.length !== 0">
      <div class="row">
        <div class="col-xs-4">
          <h4>Nombre</h4>
        </div>
        <div class="col-xs-4">
          <h4>Email</h4>
        </div>
        <div class="col-xs-3">
          <h4>Teléfono</h4>
        </div>
      </div>

      <div class="row" v-for="item in contacts" v-bind:key="item.id">
        <div class="col-xs-4">
          <p>{{item.data.nombre}}</p>
        </div>
        <div class="col-xs-4">
          <p>{{item.data.email}}</p>
        </div>
        <div class="col-xs-3">
          <p>{{item.data.telefono}}</p>
        </div>
        <div class="col-xs-1">
          <button class="btn btn-danger btn-sm" @click="borrarContacto(item.id)">
            <span class="glyphicon glyphicon-remove"></span>
          </button>
        </div>
      </div>
    </div>
    <div v-else>
      <div class="row">
        <div class="col-xs-12">
          <p>No se ha encontrado ningún contacto.</p>
        </div>
      </div>
    </div>
  </div>
</section>
```

Los nuevos contactos se añadirán mediante un formulario donde el usuario deberá introducir el nombre, el email y el número de teléfono. Una vez se han llenado todos los campos, se habilitará el botón de “Añadir”, y una vez lo pulses se añadirá el nuevo contacto en el modelo.

```
<!-- Añadir nuevo Section -->
<section id="nuevo">
  <div class="container">
    <div class="row">
      <div class="col-lg-12 text-center">
        <h2>Añadir contacto</h2>
        <hr class="star-primary">
      </div>
    </div>
    <div class="row">
      <div class="col-lg-8 col-lg-offset-2">
        <form name="sentMessage" id="contactForm" novalidate>
          <div class="row control-group">
            <div class="form-group col-xs-12 floating-label-form-group controls">
              <label>Nombre</label>
              <!-- hacemos que se asocie lo introducido por el usuario a uno de los campos del objeto que se crea en vue -->
              <input type="text" class="form-control" placeholder="Nombre" id="name" v-model="nombre">
              <p class="help-block text-danger"></p>
            </div>
          </div>
          <div class="row control-group">
            <div class="form-group col-xs-12 floating-label-form-group controls">
              <label>Email</label>
              <input type="email" class="form-control" placeholder="Email" id="email" v-model="email">
              <p class="help-block text-danger"></p>
            </div>
          </div>
          <div class="row control-group">
            <div class="form-group col-xs-12 floating-label-form-group controls">
              <label>Teléfono</label>
              <input type="tel" class="form-control" placeholder="Teléfono" id="phone" v-model="telefono">
              <p class="help-block text-danger"></p>
            </div>
          </div>
          <br>
          <div id="success"></div>
          <div class="row">
            <div class="form-group col-xs-12">
              <button type="button" class="btn btn-success btn-lg" :disabled="nombre==null || email==null || telefono==null" @click="guardarContacto" @keyup.enter="guardarContacto">Añadir</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</section>
```

Si entramos a analizar agendaVue.js, y como podemos observar en la siguiente imagen, el array de contactos sólo dispone del contacto de Naroa Jauregi por ahora.

```
var appAgendaVue= new Vue ({

    el:'#appAgendaVue',
    data:{

        nuevoContacto: { nombre: '', email: '', telefono:'' },
        contactos:[
            {
                nombre: 'Naroa Jauregi',
                email: 'naroa.jauregi@opendeusto.es',
                telefono: 688866657
            }
        ],
    },
    computed:{

        reversedContacts(){
            return this.contactos.slice(0).reverse();
        }
    },
})
```

Por otro lado, hemos añadido los métodos de “guardar contacto” y “borrar contacto” en agendaVue.js. El primero de ellos, introduce un nuevo contacto al array y deja el formulario mencionado anteriormente vacío, para poder añadir otro nuevo contacto cuando sea necesario. En lo que respecta al segundo método, este elimina el contacto del array desde el índice que hemos facilitado en nuestra agenda.

```
methods:{

    guardarContacto: function(event){

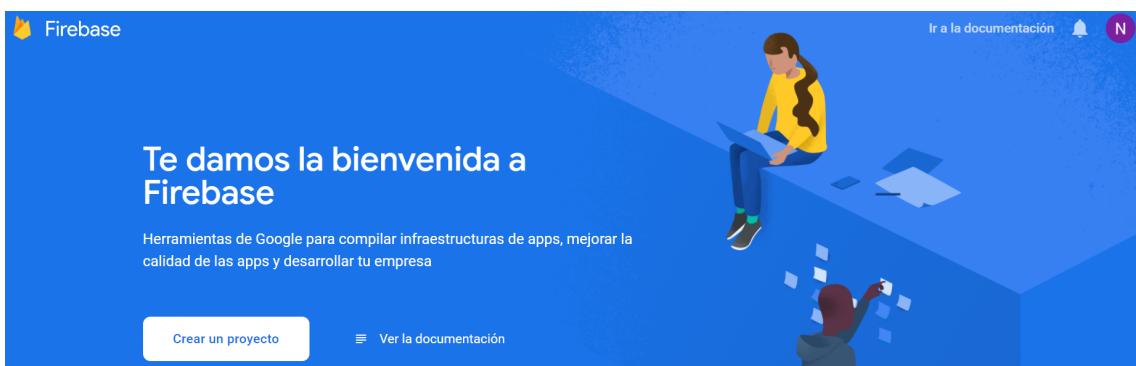
        this.contactos.push ( { nombre: this.nuevoContacto.nombre, email: this.nuevoContacto.email, telefono: this.nuevoContacto.telefono });
        this.nuevoContacto= { nombre: '', email: '', telefono: ''};
    },
    borrarContacto: function (index){

        this.contactos.splice(index, 1);
    }
}
```

2.2. DESCRIPCIÓN DE LAS FUNCIONALIDADES AÑADIDAS IMPLEMENTADAS

2.2.1. Persistir con VueFire los datos de Agenda en Firebase

En este apartado trabajaremos con Vuefire y Firebase. Lo primero que hay que hacer es entrar en la página web de Firebase y crear un nuevo proyecto.



X Crear un proyecto(paso 1 de 3)

nombre de tu proyecto[?]

Nombre del proyecto
Book Store

book-store-7014a Seleccionar recurso superior

Acepto las [condiciones de Firebase](#)

Continuar

En primer lugar, tuvimos que poner a “true” los siguientes campos para poder escribir y leer desde la base de datos.

The screenshot shows the Firebase Realtime Database rules editor. On the left, there's a sidebar with navigation links: Descripción general de, Compilación, Authentication, Firestore Database, Realtime Database (selected), Storage, Hosting, Functions, and Machine Learning. The main area is titled "Realtime Database" and has tabs for Datos, Reglas (selected), Copias de seguridad, and Uso. Below the tabs are buttons for "Editar reglas" and "Supervisar reglas". A blue header bar at the top of the main content area says "Cambios sin publicar" and has "Publicar" and "Descartar" buttons. The code editor contains the following rules:

```
1  rules: {  
2      ".read": true,  
3      ".write": true  
4  }  
5 }  
6 }
```

The screenshot shows the Cloud Firestore rules editor. On the left, there's a sidebar with navigation links: Datos, Reglas (selected), Índices, and Uso. The main area is titled "Cloud Firestore" and has tabs for Datos, Reglas (selected), and Índices. Below the tabs are buttons for "Editar reglas" and "Supervisar reglas". A sidebar on the left shows two entries: "Ahora mismo cambios sin publicar" and "Hoy • 3:44 p. m.". The main content area has a blue header bar at the top saying "cambios no publicados" and has "Publicar" and "Descartar" buttons. The code editor contains the following rules:

```
1  rules_version = '2';  
2  service cloud.firestore {  
3      match /databases/{database}/documents {  
4          match /{document=**} {  
5              allow read, write: if true;  
6          }  
7      }  
8  }
```

Una vez hemos puesto a true los campos de las ilustraciones anteriores, añadimos el siguiente código en index.html. De este modo, podremos crear una colección en la Base de Datos además de guardar los diferentes usuarios. Por otro lado, también modificamos los métodos “guardar contacto” y “borrar contacto” mencionados anteriormente para poder usarlos con Firebase y VueFire.

```
<script>
  const firebaseConfig = {
    apiKey: "AIzaSyBrWs0IokaMHA6qYh8vVT0MrILei7MgJwc",
    authDomain: "book-store-7014a.firebaseio.com",
    databaseURL: "https://book-store-7014a-default-rtbd.firebaseio.com",
    projectId: "book-store-7014a",
    storageBucket: "book-store-7014a.appspot.com",
    messagingSenderId: "143803143024",
    appId: "1:143803143024:web:be578112d8a081f7deb5b5",
    measurementId: "G-6Y995BB9BL"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  const db = firebase.firestore();

  const colección = db.collection('users');

  var agendaVue = new Vue({
    el: '#AppagendaVue',
    data: {
      nombre: null,
      email: null,
      telefono: null,
      contacts: []
    },
  });
</script>
```

```
mounted(){
  this.contacts= []
  colección.get().then( (r) => r.docs.map( (item) => this.contacts.push({id:item.id, data:item.data()})))
},
methods:{
  guardarContacto(){
    colección.add({
      nombre: this.nombre,
      email: this.email,
      telefono: this.telefono
    }).then( ()=> this.$mount())
    this.nombre=null;
    this.email=null;
    this.telefono=null;
  },
  borrarContacto(index){
    colección.doc(index).delete().then( () => this.$mount())
  }
}
</script>
```

Una vez realizado todo lo anterior, es así cómo quedaría nuestra agenda de VUE.

CONTACTOS

NOMBRE	EMAIL	TELÉFONO
Lorea	lorea.intxausti@opendeusto.es	888 888 888
Naroa	naroa.jauregi@opendeusto.es	111 111 111



AÑADIR CONTACTO

Nombre

Email

Teléfono

Añadir

Si procedemos a añadir un nuevo contacto, se mostrará en nuestra lista de contactos:

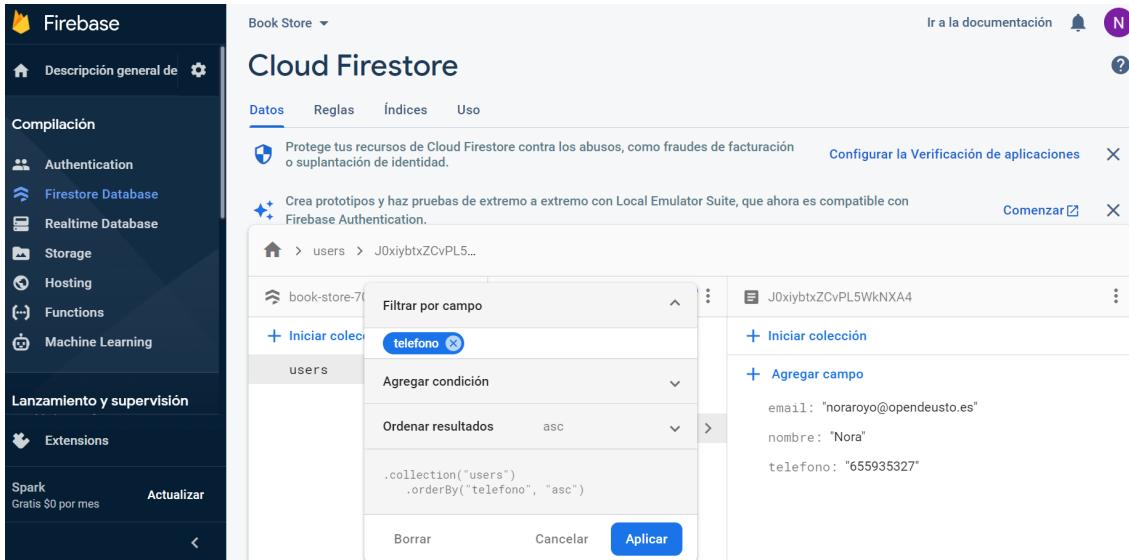
NOMBRE	EMAIL	TELÉFONO
Nora	noraroyo@opendeusto.es	655935327
Naroa	naroa.jauregi@opendeusto.es	688866657
Ainhoa	ainhoailarramendi@opendeusto.es	645738499

Además, también podremos visualizar los mismos datos en la Base de Datos Firestore, dentro del apartado Cloud Firestore, dentro de nuestro proyecto Book Store. Como podemos observar en la siguiente imagen, los contactos añadidos de “Ainhoa” y “Nora” se pueden ver en la lista de “users”.

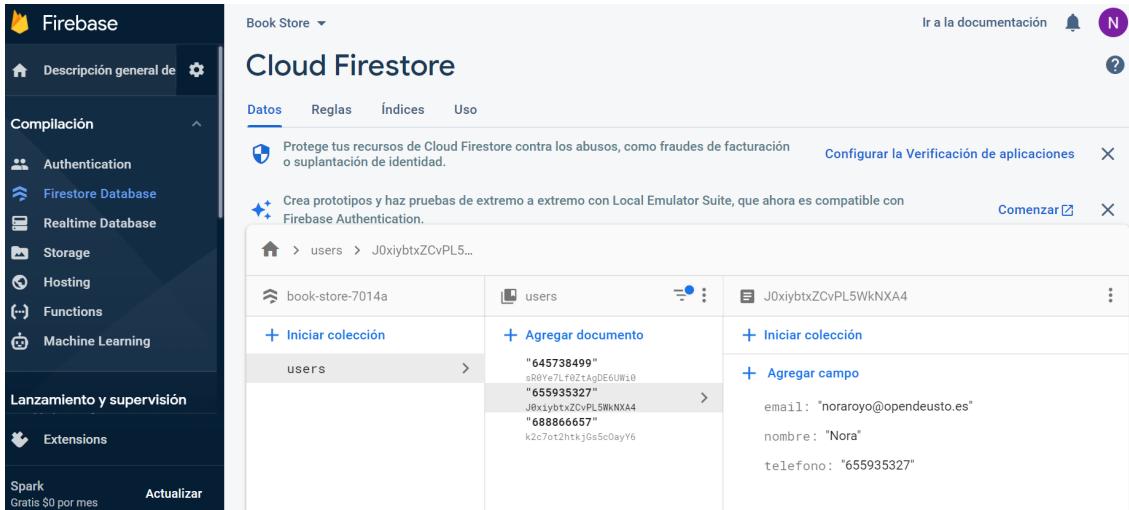
Documento	email	nombre	telefono
"Ainhoa"	ainhoailarramendi@opendeusto.es*	Ainhoa	645738499
"Nora"	noraroyo@opendeusto.es*	Nora	655935327*

Documento	email	nombre	telefono
"Ainhoa"	ainhoailarramendi@opendeusto.es*	Ainhoa	645738499
"Nora"	noraroyo@opendeusto.es*	Nora	655935327*
J0x1ybt...			

Además, podemos filtrarlos por teléfono, por email o por nombre. Por lo que si aplicamos el filtro de teléfono nos quedaría el siguiente resultado:



The screenshot shows the Firebase Cloud Firestore interface. On the left, the navigation sidebar includes 'Authentication', 'Firestore Database', and 'Realtime Database'. The main area displays a query builder for the 'users' collection, filtering by 'telefono'. The results show a single document with fields: email ('noraroyo@opendeusto.es'), nombre ('Nora'), and telefono ('655935327').



This screenshot shows the same interface but with a different view. It lists multiple documents under the 'users' collection, each with a unique ID and three fields: email, nombre, and telefono. The data is identical to the previous screenshot.

2.2.2. Incluir capacidades sociales (Twitter, Facebook)

Este apartado consiste en añadir dos capacidades sociales: Twitter y Facebook.

Si empezamos a analizar el caso de Twitter, lo primero que hemos hecho es crear una cuenta para llevar a cabo esta parte del proyecto.

correo electrónico: ingenieriaweb.bookstore@gmail.com

contraseña: 2021-2022iwbs

nombre de usuario: @iwBookStore

Dicho esto, podemos visualizar los dos tweets que hemos publicado, escribir un nuevo tweet, seguir a @iwBookStore, dar me gusta, responder, etc.

```
<!--aquí empieza el div de las redes sociales -->
</div>

<section id="redesSociales">
  <div class="container">
    <div class="row">
      <div class="col-lg-12 text-center">
        <h2>Redes Sociales</h2>
        <hr class="star-primary">
      </div>
    </div>
    <div class="row">
      <div class="col-lg-8 col-lg-offset-2">
        <div class="form-group col-xs-12" id="followTwitter">
          <a href="https://twitter.com/iwBookStore" class="twitter-follow-button" data-show-count="false" > Follow @iwBookStore</a>
          <a href="https://twitter.com/share?ref_src=twsrctwsrc%5Etfw" class="twitter-share-button" data-show-count="false">Tweet</a>
        </div>

        <div class="fb-like" data-href="https://developers.facebook.com/docs/plugins/" data-width="" data-layout="button_count" data-action="like" data-size="large" data-share="true"></div>

        <blockquote class="twitter-tweet"><p lang="en" dir="ltr">¡Bienvenidos! Book Store (@iwBookStore)</p>
<a href="https://twitter.com/iwBookStore/status/1479871082760912899" >8 de enero, 2022</a></blockquote>
<script async src="https://platform.twitter.com/widgets.js" charset="utf-8"></script>

        <blockquote class="twitter-tweet"><p lang="es" dir="ltr">Ongi etorri! Book Store (@iwBookStore)</p>
<a href="https://twitter.com/iwBookStore/status/1479874082963443718" >8 de enero, 2022</a></blockquote>
<script async src="https://platform.twitter.com/widgets.js" charset="utf-8"></script>
```

The image shows two tweets from the Twitter account @iwBookStore. The first tweet is in Spanish and says '¡Bienvenidos a la librería Book Store! 😊📚'. The second tweet is in English and says 'Ongi etorri Book Store liburu dendaral! 😊📚'. Both tweets have a timestamp of 6:42 p.m. on January 8, 2022, and show standard Twitter interaction buttons.

En lo que respecta a Facebook, también hemos creado una cuenta para el proyecto.

nombre y apellidos: Lann Lann

correo electrónico: ingenieriaweb.bookstore@gmail.com

contraseña: 2021-2022iwbs

Dicho esto, podemos visualizar la publicación que hemos hecho, podemos darle me gusta, comentar y compartir.

```
<div class="fb-post" data-href="https://www.facebook.com/permalink.php?story_fbid=108565938377477&id=100076722140061" data-width="500" data-show-text="true"><blockquote cite="https://www.facebook.com/permalink.php?story_fbid=108565938377477&id=100076722140061/?type=3" class="fb-xfbml-parse-ignore"><p>Me gusta</p>.concat(<p> .concat(<p>JS&#039;));</p>Publicada por<br/><a href="https://www.facebook.com/profile.php?id=100076722140061">Book Store</a> en&ampnbsp<br/><a href="https://www.facebook.com/profile.php?id=100076722140061">Jueves, 13 de enero de 2022</a></blockquote></div>
```



2.2.3. Incluir capacidades micro-semánticas y de la web de datos

En nuestro proyecto, con la implementación de JSON-LD, se han incluido estructuras de datos, teniendo schema.org de referencia. Lo primero que hemos hecho es generar el fichero “db.json”. Esto lo hemos conseguido mediante la ejecución de varios comandos. Primero, ejecutamos el comando que contiene los valores de prueba para la base de datos:

```
python manage.py dumpdata > db.json
```

Después, cargamos esos datos:

```
python manage.py loaddata --app app db.json
```

Los datos estructurados son un formato estandarizado para proporcionar información sobre una página y clasificar el contenido de la página. Como nuestra aplicación es de contactos, hemos aplicado la estructura de persona. Aquí los diferentes atributos serán el nombre, el correo electrónico, la dirección, el código postal o el teléfono.

Hemos recurrido a modelos predefinidos para la anotación de las sintaxis de JSON-LD. En ellas, las etiquetas `<script>` de apertura y cierre definen el tipo de script que es “application/ld+json”. Esto significa que los datos que engloba pueden ser interpretados por programas capaces de enlazar datos en formato JSON. En lo que corresponde a las palabras clave `@context` y `@type` con los valores “<http://schema.org>” y “`person`” respectivamente obtiene con ellas la instrucción de que los datos que siguen pertenecen al esquema “Event” según Schema.org. Además, aparecen las propiedades “`name`”, “`address`”, “`email`” y “`telephone`”, en el mismo nivel. A estos se les asignan los datos necesarios sobre la persona.

```

<script type='application/ld+json'>
{
  "@context": "http://www.schema.org",
  "@type": "person",
  "name": "Naroa",
  "address": {
    "@type": "PostalAddress",
    "addressLocality": "Tolosa",
    "addressRegion": "Guipuzcoa",
    "postalCode": "20400",
    "addressCountry": "Spain"
  },
  "email": "naroa.jauregi@opendeusto.es",
  "telephone": "+34 666 666 666"
}
</script>
<script type='application/ld+json'>
{
  "@context": "http://www.schema.org",
  "@type": "person",
  "name": "Lorea",
  "address": {
    "@type": "PostalAddress",
    "addressLocality": "Donostia",
    "addressRegion": "Guipuzcoa",
    "postalCode": "20010",
    "addressCountry": "Spain"
  },
  "email": "lorea.intxausti@opendeusto.es",
  "telephone": "+34 555 555 555"
}
</script>

```

También tenemos la opción de marcar el script con Microdata o RDF, siguiendo el Schema.org, y conseguimos que no se pierda ninguna información:

```

<!-- sintaxis de microdatos según Schema.org-->

<div itemscope itemtype="http://schema.org/Person">
  <span itemprop="name">Jon</span>
  <span itemprop="email">jonleinena@opendeusto.es</span>
  <span itemprop="telephone">+34 222 222 222</span>
</div>

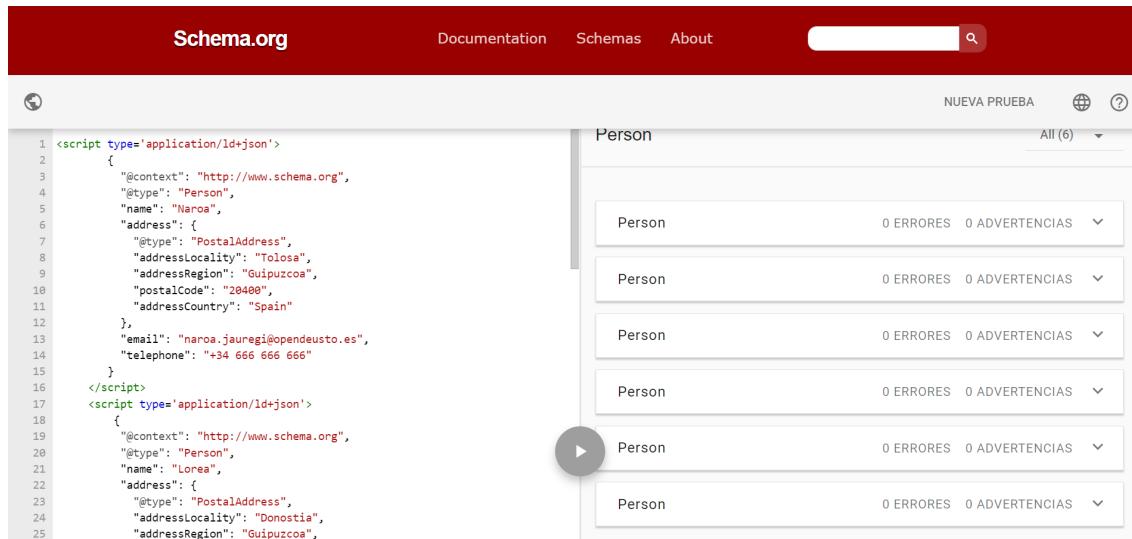
<!-- sintaxis RDFa según Schema.org-->

<div vocab="http://schema.org/" typeof="Person">
  <span property="name">Mikel</span>
  <span property="email">mikelferrer@opendeusto.es</span>
  <span property="telephone">+34 444 444 444</span>
</div>

```

Los datos estructurados, son útiles para habilitar funciones y mejoras de resultados de búsquedas especiales. Debido a que los datos estructurados etiquetan cada elemento individual del contacto, los usuarios pueden buscar un contacto por nombre, dirección, correo electrónico etc. Hemos de mencionar que es necesario superar una verificación la

cual indique que todo está correcto y que no hay errores, ya que sino no se podría incluir en el formato de búsqueda que tiene Internet.



The screenshot shows the Schema.org validation interface. On the left, a code editor displays a JSON-LD script:

```

1 <script type='application/ld+json'>
2   {
3     "@context": "http://www.schema.org",
4     "@type": "Person",
5     "name": "Naroa",
6     "address": {
7       "@type": "PostalAddress",
8       "addressLocality": "Tolosa",
9       "addressRegion": "Guipuzcoa",
10      "postalCode": "20400",
11      "addressCountry": "Spain"
12    },
13    "email": "naroa.jauregi@opendeusto.es",
14    "telephone": "+34 666 666 666"
15  }
16 </script>
17 <script type='application/ld+json'>
18   {
19     "@context": "http://www.schema.org",
20     "@type": "Person",
21     "name": "Lorea",
22     "address": {
23       "@type": "PostalAddress",
24       "addressLocality": "Donostia",
25       "addressRegion": "Guipuzcoa",

```

The main panel shows a list of entities under the category "Person". There are six entries, each with a "Person" label, "0 ERRORES 0 ADVERTENCIAS", and a dropdown arrow. A large play button icon is centered over the list.

Person		0 ERRORES 0 ADVERTENCIAS	▲
@type	Person		
name	Naroa		
email	naroa.jauregi@opendeusto.es		
telephone	+34 666 666 666		
address			
@type	PostalAddress		
addressLocality	Tolosa		
addressRegion	Guipuzcoa		
postalCode	20400		
addressCountry			
@type	Country		
name	Spain		

3. CONCLUSIONES

En conclusión, tras la realización de este proyecto hemos aprendido cómo hacer una página web, y todo lo que eso conlleva. Hoy en día, es esencial que un negocio disponga de una página web y pueda exponer sus servicios de la mejor manera para los usuarios. Por lo que gracias al proyecto de Book Store hemos podido ponernos en la piel del desarrollador y del usuario para llegar al mejor resultado posible.

Aunque en varias ocasiones no nos hayamos atascado, por ejemplo con la conexión a la base de datos en la entrega de JavaScript, hemos trabajado en equipo y cada miembro del grupo ha aportado su grano de arena para seguir adelante. Finalmente creemos que hemos logrado el objetivo deseado.

4. BIBLIOGRAFÍA

Berners-Lee, Tim. “Metadata & Enhanced Semantics · Web Dev Topics · Learn the Web.” Learn the Web, October 15, 2019. <https://learn-the-web.algonquindesign.ca/topics/metadata-enhanced-semantics/>.

“JSON-LD Generator: JSON-LD Schema Generator Tool.” LD, May 6, 2019. <https://jsonld.com/json-ld-generator/>.

“Marcado Con JSON-LD Según Schema.org.” IONOS Digitalguide. Accessed January 11, 2022. <https://www.ionos.es/digitalguide/paginas-web/creacion-de-paginas-web/tutorial-marcado-con-json-ld-segun-schemaorg/>.

“Schema.org.” Schema.org - Schema.org. Accessed January 11, 2022. <https://schema.org/>.

“Understand How Structured Data Works.” Google. Google. Accessed January 11, 2022. <https://developers.google.com/search/docs/advanced/structured-data/intro-structured-data?hl=en&rd=1>.
