DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE VERACRUZ

RESIDENCIAS PROFESIONALES

REPORTE FINAL

PROYECTO:

ARTESANÍAS MATEMÁTICAS

LUGAR:

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS, A.C.

PRESENTA:

GUERRERO MONTERO FERNANDO

NÚMERO DE CONTROL:

E16020823

CARRERA:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

PERÍODO:

FEBRERO – JULIO 2021

Asesor externo	Asesor interno
Mtro. Maximino Tapia Rodríguez	Lsca. Jesús Cruzado Calleja

Agradecimientos

Agradezco a mi familia; mi mamá Marisela, mi papá Jorge, mis hermanos Eduardo y Jorge Luis, y demás familiares por apoyarme siempre. Y a mi perrita Boney que tranquiliza mi alma.

Agradezco a mis amigos y compañeros que he tenido a lo largo de mi vida. A mis compañeros de carrera Sergio, Israel, Adal, Arciniega, Irving, Omar, Lalo, Jimmy, Adrián, y muchos amigos que me han ayudado a divertirme en la universidad y a disfrutar aprender.

Agradezco al Mtro. Darío por enseñarme la belleza de las matemáticas durante la preparatoria, que en paz descanse.

Agradezco a los tutores y profesores de bien durante mi carrera académica que me han enseñado y preparado para ser un buen Ingeniero en Sistemas Computacionales. A la Comunidad Estudiantil de Programación Competitiva formada por el Dr. Torreblanca y el Dr. Estudillo por aceptarme sin experiencia previa de programación, y fortalecer mis habilidades durante los primeros semestres de la carrera. Al Dr. Rivera por enseñarme todo durante sus clases.

Agradezco a mi asesor el Mtro. Maximino por apoyarme durante el desarrollo de este proyecto. Y a la Mtra. Judith por aceptarme en las residencias.

Por último, agradezco a mi mejor amiga Maritza por haber estado a mi lado apoyándome durante 11 años.

Resumen

El contenido de este documento es una descripción de las actividades realizadas para Residencias Profesionales durante el período Febrero – Julio 2021 bajo la modalidad *home office*, las cuales incluyen diseñar y desarrollar una tienda en línea, como aplicación Web, para la venta de productos y talleres de la empresa CIMAT.

Como preámbulo a este proyecto, se aportó un apoyo a otro proyecto que se conoce como Red IRAG (Infección Respiratoria Aguda Grave). La aportación fue realizar una Base de Datos para importar registros diarios de la Red IRAG desde archivos *csv*.

Ya como parte del proyecto de *Artesanías Matemáticas*, se realizó un *Estado del Arte* para analizar tiendas Web similares, y se estudió las posibilidades de utilizar softwares especiales para crear tiendas Web o, finalmente lo que se decidió, empezar desde cero. De igual manera se decidieron las herramientas a utilizar para el desarrollo de la aplicación Web de manera local.

Después se empezó la etapa de desarrollo, que incluye: el modelado de una Base de Datos para la tienda en línea, realizar el diseño de las páginas Web y programar la operatividad de la aplicación web; todo bajo los alcances y límites planteados en la fase inicial del proyecto. Finalmente se realizaron pruebas de manera local para corroborar el funcionamiento de la aplicación Web.

El primer capítulo contiene las generalidades del proyecto; como la información de la empresa, objetivos, justificación y otros apartados.

El segundo capítulo es el marco teórico; donde se detallan las herramientas, tecnologías y software a utilizar para el desarrollo del proyecto.

El tercer capítulo redacta el desarrollo de las actividades realizadas durante el proyecto.

El cuarto capítulo desglosa los resultados obtenidos, y se muestra los archivos, páginas y procedimientos realizados.

Por último, se tiene en los capítulos finales las conclusiones, competencias desarrolladas a lo largo del proyecto, las fuentes de información y un apartado de anexos.

Índice

Índice de Figuras

Índice de Tablas

CAPÍTULO 1. GENERALIDADES DEL PROYECTO

1.1 Introducción

En tiempos recientes el CIMAT, como parte de su estrategia para promover el entendimiento y difusión de las matemáticas entre los estudiantes de educación Básica, Media y Superior; desarrolló e implementó para tal fin el CL-MaT (Centro y Laboratorio de Materiales didácticos).

A la fecha ya se han desarrollado e impartido diferentes talleres y atendido diversos grupos de estudiantes.

Ahora se pretende que algunos de los materiales que se han fabricado en estos talleres, pueden ser objeto de venta, es decir se pretende: Fabricar y poner a la venta productos desarrollados en los talleres, con actividades y eventos que refuercen la participación de talleres como una forma continua de promover las matemáticas entre la población en general.

1.2 Información de la empresa

- **Nombre**: Centro de Investigación en Matemáticas A.C. (CIMAT)
- **Domicilio**: Jalisco S/N, Col. Valenciana CP: 36023 Guanajuato, Gto, México, Apartado Postal 402, CP 36000
- **Giro**: Investigación y docencia matemática
- **Contacto**: Tel. + 52 473 732 7155 / 735 0800 / Responsable de la información laura@cimat.mx
- **Misión**: “El **Centro de Investigación en Matemáticas AC** es un centro público de investigación integrado al Sistema de Centros Públicos CONACYT, dedicado a la generación, transmisión y aplicación de conocimientos especializados en las áreas de matemáticas, estadística y ciencias de la computación. Orientado hacia la investigación científica, la formación de recursos humanos de alto nivel, el mejoramiento de la competencia matemática de la sociedad, así como al apoyo en la solución de problemas que competen a sus áreas de interés, el **CIMAT** busca contribuir al desarrollo científico y tecnológico de México.”
- **Visión**: “Ser un centro de investigación de excelencia y polo de desarrollo científico en progresiva consolidación, reconocido a nivel nacional e internacional en sus áreas de especialización; fortalecido en su capacidad de convocatoria y en la integración de una masa crítica en grupos de alto rendimiento científico, y ser modelo de eficiencia y crecimiento e impacto social para otros centros de investigación.”

- **Organigrama general**:

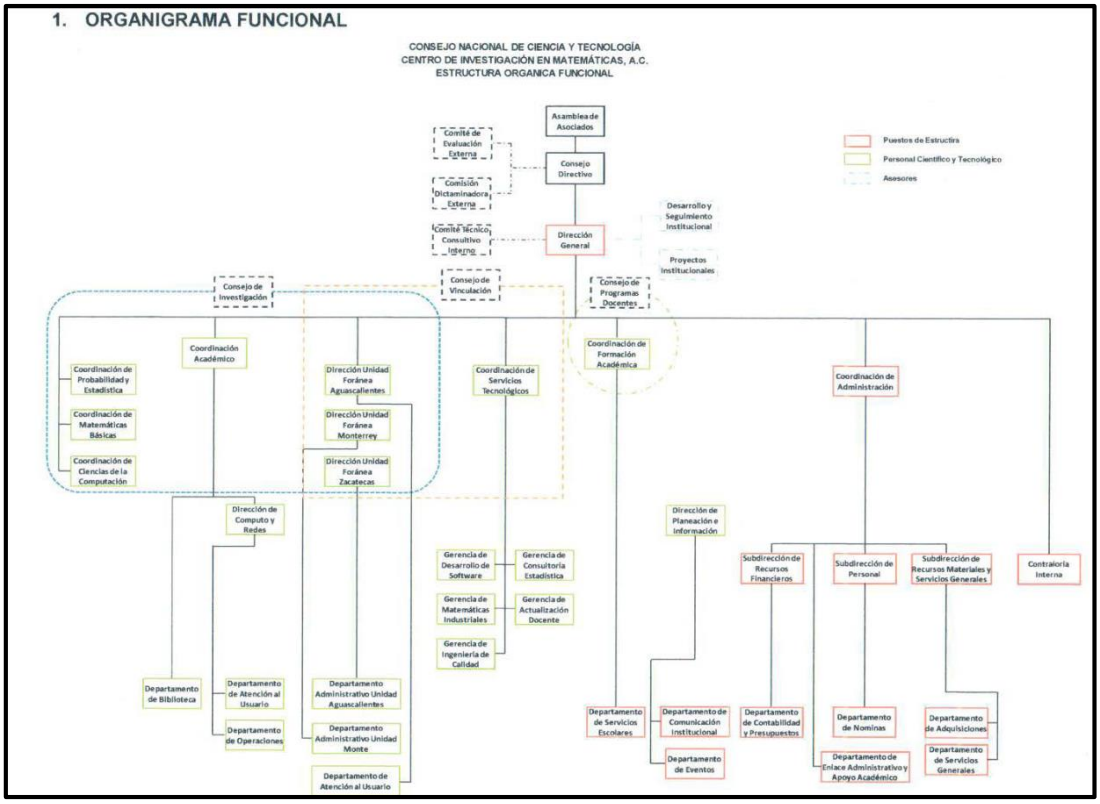

Figura 1.1 Organigrama general del CIMAT.

- **Organigrama específico del área**:

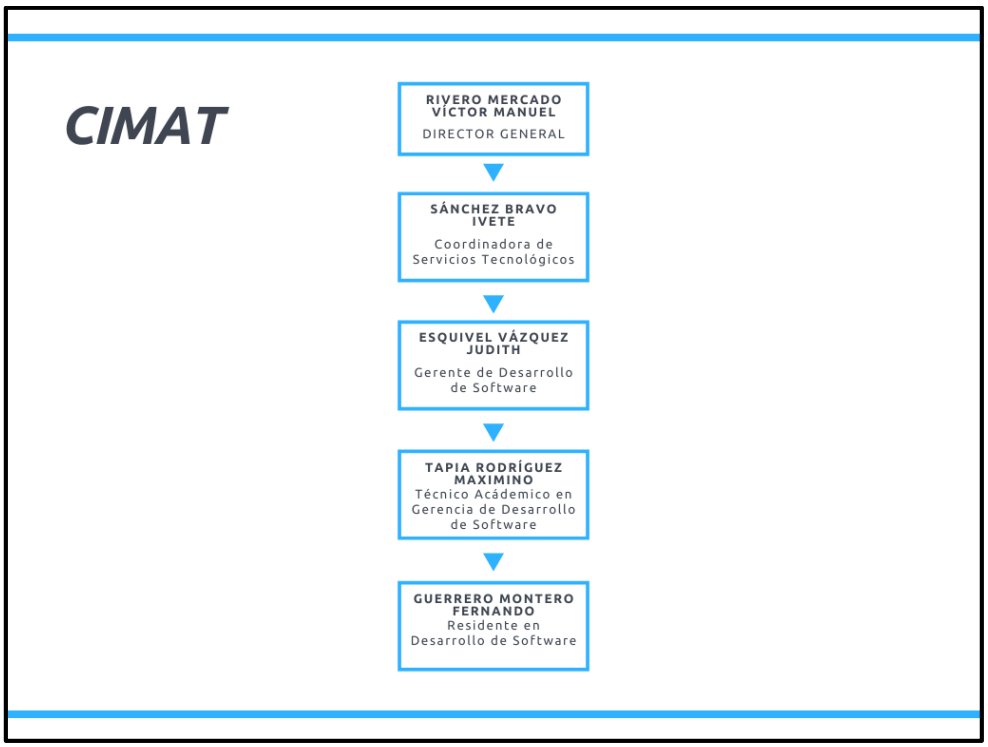

Figura 1.2 Organigrama específico del área de trabajo del CIMAT.

1.3 Problemática

Implementar una tienda virtual (Clásica) para vender objetos de CL-MaT.

Características generales de una tienda online

1. Apariencia y diseño.
2. Catálogo de productos.
3. Sistema interno de búsqueda de productos.
4. Sistema de recomendaciones.
5. Carrito de compra.
6. Proceso de registro.
7. Métodos de pago y pasarelas de pago.
8. Certificado de seguridad.
9. Proceso de venta y embudo de conversión.
10. Sistemas para la atención al cliente.
11. Gestión de stocks.
12. Integración de otros sistemas de gestión.
13. Consideraciones.

En la primera versión no se realizarán compras en línea, solo depósito bancario y solo dentro de la república Mexicana.

Características especiales

- Venderá artículos didácticos de matemáticas.
- Cada producto tiene un pequeño texto-video-audio explicativo de que matemáticas entran en la fabricación del producto o en su uso.
- Sin anuncios.
- Muy simple.
- Con videos de los productos.
- Con actividades para los productos, la idea es que después que lo compran tienen acceso a actividades para utilizarlo mejor, como clases de utilización.
- También vender los talleres, que entran como producto.

1.4 Objetivos

1.4.1 Objetivo General

Difundir las matemáticas a partir de la venta de productos didácticos de matemáticas.

1.4.2 Objetivos Específicos

- Tener una tienda en línea para vender productos CL-MaT.
- Buscar la sustentabilidad del CL-MaT.

1.5 Justificación

En la era actual de la tecnología y el Internet, han surgido diversos catálogos de diferentes productos que uno puede pedir y comprar en línea. Desde un comercio popular como Amazon, a algo más tradicional como la zapatería de la ciudad; se ha demostrado que para una empresa es muy relevante tener un sitio web donde se pueda desplegar y comprar los productos y servicios que ofrece.

En el CIMAT se ofrecen talleres para promover las matemáticas entre estudiantes de todos los niveles donde se realizaban materiales especiales como objetivo del taller. El CIMAT no sólo quiere ofrecer más directamente los talleres, también estos materiales especiales con los que cualquiera puede aprender algún concepto de matemáticas. Para ello se necesita un sitio web donde se muestre a un cliente potencial lo que el CIMAT tiene para ofrecer: una tienda en línea. Y en el futuro se pueden agregar más talleres, más productos y más actividades especiales que se vayan desarrollando. En las siguientes imágenes se muestran algunos de los productos que se venderían, realizados en talleres:

Figura 1.3 Porta lápices de CL-Mat.

Figura 1.4 Lámparas de CL-Mat.

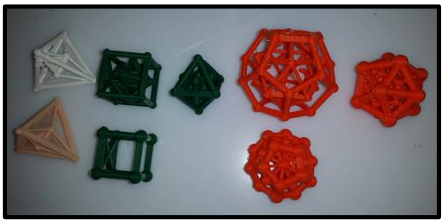

Figura 1.5 Sólidos platónicos de CL-Mat,

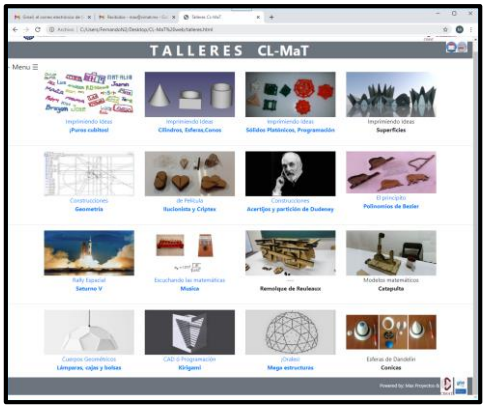

Figura 1.6 Talleres de CL-Mat.

Figura 1.7 Maquetas de CL-Mat.

Figura 1.8 Visor estereoscópico de CL-Mat.

En una tienda web es importante tener ciertos elementos para una mejor experiencia de usuario y mejores resultados en ventas como lo son: el carrito de compras, el registro del cliente, el despliegue estructurado de productos, el costo de envío, el poder marcar como favoritos ciertos elementos, el pagar con una tarjeta bancaria, etcétera.

Aparte, se quiere ofrecer a los clientes: ejercicios, ejemplos y experimentos; al momento de comprar algún producto. Estos documentos estarán relacionados con las matemáticas, lo que fomentará su aprendizaje y práctica en mayor medida; de igual manera, se espera que satisfagan los deseos del cliente.

Para atraer e incentivar a los clientes, los productos tendrán texto-vídeo-audio explicativo sobre las matemáticas que se utilizan y así puedan escoger sus compras. Además, se quiere que la vista sea muy simple, que la página sea muy sencilla de usar y que no tenga anuncios para eliminar distracciones sobre el cliente.

El objetivo es el aprendizaje y utilización de las matemáticas, y con una herramienta potente como lo es una tienda en línea, es un objetivo que se puede conseguir con los procesos adecuados.

1.6 Alcances y Limitaciones

1.6.1 Alcances

Desarrollar un prototipo de la tienda web con la funcionabilidad básica.

1.6.2 Limitaciones

- El proyecto funcionará sobre un servidor del CIMAT.
- El uso de la plataforma lo dictará el CIMAT.
- Se planea enfocarse más en las características especiales de la tienda.

CAPÍTULO 2.
MARCO TEÓRICO

2.1 Herramientas y Tecnologías

2.1.1 Visual Studio Code

"Visual Studio Code es un ligero pero poderoso editor de código fuente el cuál corre en tu escritorio y está disponible para Windows, macOS y Linux. Viene incluido con soporte para JavaScript, [...] y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, PHP, GO)" (Microsoft, 2021).

Visual Studio Code es una interfaz para programar muy fácil y rápida de instalar, además de tener compatibilidad con todos los lenguajes que se utilizan para una aplicación web (por ejemplo, HTML, CSS, PHP, e incluso SQL). También tiene un buen manejo de directorio de archivos, lo que hace a la programación de muchos archivos muy cómoda y rápida.

Figura 2.1 Logotipo de Visual Studio Code.

2.1.2 HTML

HTML (Lenguaje de Marcado de Hipertexto, del inglés *HyperText Markup Language*) es el lenguaje básico para realizar páginas web. Con HTML se define la estructura y los elementos que contendrá una página web, y combinándolo con otras tecnologías para darle estilo (CSS) y funcionalidad (PHP y JavaScript) se construyen las aplicaciones web (Mozilla, 2021).

Los elementos de un código HTML se escriben entre '<' y '>', y se conocen como etiquetas. Así, podemos definir los diferentes componentes de una página web por ejemplo: contenedores, imágenes, texto y listas.

Figura 2.2 Logotipo de HTML5, su versión más reciente.

2.1.3 CSS

"Hojas de Estilo en Cascada (del inglés *Cascading Style Sheets*) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML […]. CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios" (Mozilla, 2021).

CSS es lo que le da forma y estilo a los elementos de una página web. Es usual querer escoger un color único o un tamaño específico para ciertos elementos, de

igual manera es útil para manejar los espaciados y la división del contenido. Lo normal es redactar todas las reglas CSS en su propio archivo y agregarlo como referencia desde HTML para que se apliquen las reglas. También es posible agregar reglas CSS a una etiqueta HTML agregando la propiedad *style*, puede ser muy útil para definir un cambio para un elemento muy específico (por ejemplo, el color de un texto).

Figura 2.3 Logotipo de CSS 3.

2.1.4 Bootstrap

Bootstrap es un framework de código abierto para aplicaciones web cuya utilidad es construir rápidamente diseños responsivos. Su mayor ventaja es su sistema responsivo *grid* o de cuadrículas que sirve para acomodar el contenido de la página web de forma sencilla e ideal, aunque también tiene elementos rápidos de diseño como botones (Bootstrap team, 2021).

Para poder utilizarlo, se debe añadir un enlace al repositorio que contiene Bootstrap. También está la opción de descargarlo en su totalidad y manejar el enlace de manera local.

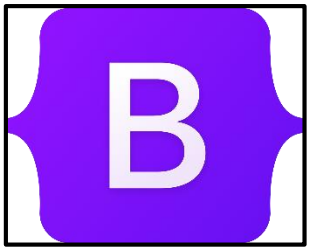

Figura 2.4 Logotipo de Bootstrap.

2.1.5 JavaScript

"JavaScript (JS) es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional)" (Mozilla, 2021).

Se puede agregar JS a códigos HTML con la etiqueta <*script*>, de igual manera Bootstrap contiene cierto funcionamiento de JS. Sirve para agregar funcionalidad desde el lado del cliente, a diferencia de PHP que es desde el lado del servidor.

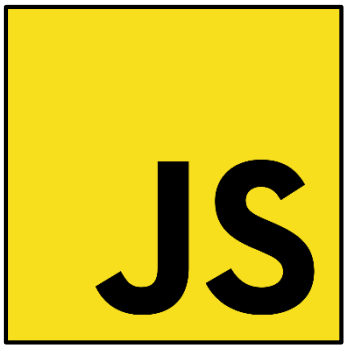

Figura 2.5 Logotipo de JavaScript.

2.1.6 HTTP

"*Hypertext Transfer Protocol (HTTP)* (o Protocolo de Transferencia de Hipertexto en español) es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML […]. Sigue el clásico modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta del mismo" (Mozilla, 2021).

HTTP define un conjunto de métodos de petición para el manejo de recursos. Lo más relevante es que se puede mandar información entre diferentes páginas web usando varios de estos métodos. Los más importantes son:

- Método GET: "El método GET solicita una representación de un recurso específico. Las peticiones que unan el método GET sólo deben recuperar datos" (Mozilla, 2021).
- Método POST: "El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor" (Mozilla, 2021).

2.2 XAMPP

2.2.1 Apache

"Apache es el servidor Web más usado típicamente en sistemas Linux. Servidores Web son usados para servir páginas Web solicitadas por clientes de computadoras […]. Los usuarios introducen un Localizador Uniforme de Recursos (URL) para apuntar a un servidor Web mediante su Nombre de Dominio Completamente Calificado (FQDN) y una ruta al recurso requerido" (Ubuntu, 2021).

Los servidores Web con Apache son usados en combinación con un motor de Base de Datos MySQL, el lenguaje de scripting PHP, entre otros lenguajes populares. Está configuración es definida como LAMP (Linux, Apache, MySQL, PHP) (Ubuntu, 2021).

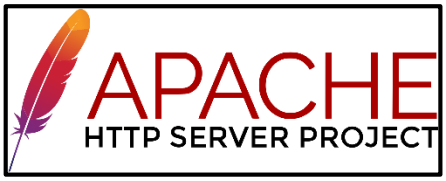

Figura 2.6 Logotipo del servidor web Apache.

2.2.2 PHP

"PHP, acrónimo de 'PHP: Hypertext Preprocessor', es un lenguaje de 'scripting' de propósito general y de código abierto que está especialmente pensado para el desarrollo web y que puede ser embebido en páginas HTML. Su sintaxis recurre a C, Java y Perl, siendo así sencillo de aprender. El objetivo principal de este lenguaje es permitir a los desarrolladores web escribir dinámica y rápidamente páginas web generadas; aunque se puede hacer mucho más con PHP" (PHP Group, 2021).

Figura 2.7 Logotipo de PHP.

2.2.3 MariaDB

"MariaDB Server es una de las bases de datos relacionales de código abierto más populares. Fue creado por los desarrolladores originales de MySQL y está garantizado de permanecer como código abierto. Es parte de la mayoría de ofrecimientos de la nube y está por defecto en varias distribuciones de Linux" (MariaDB Foundation, 2021).

Figura 2.8 Logotipo de MariaDB.

2.2.4 phpMyAdmin

"phpMyAdmin es una herramienta de software gratis escrito en PHP, con la intención de manejar la administración de MySQL a través de la Web. phpMyAdmin soporta un vasto rango de operaciones en MySQL y MariaDB. Operaciones usadas frecuentemente (manejo de base de datos, tablas, columnas, relaciones, índices, usuarios, permisos, etc.) pueden ser realizadas a través de la interfaz de usuario, mientras todavía tienes la habilidad de ejecutar directamente cualquier línea de SQL" (phpMyAdmin contribuitors, 2021).

Figura 2.9 Logotipo de phpMyAdmin.

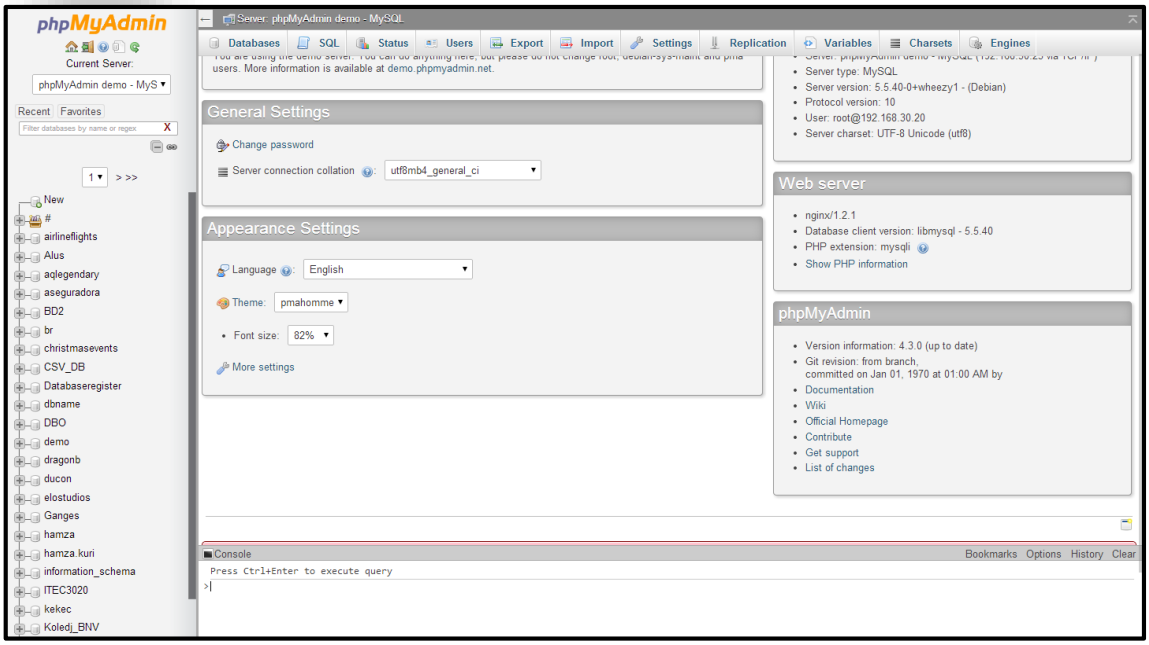

Figura 2.10 Interfaz de Usuario de phpMyAdmin.

2.2.5 XAMPP

"XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl. El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar" (Apache Friends, 2021).

Usar XAMPP nos permite, con una sola instalación, empezar a trabajar en una aplicación Web. Contiene el servidor de Apache, MariaDB y phpMyAdmin para manejar bases de datos, PHP para la codificación de las páginas web; además se combina con las demás herramientas descritas como HTML, CSS, JS y Bootstrap para poder crear una aplicación Web completa de manera local, y el trabajo hecho sobre XAMPP se puede copiar para un servidor.

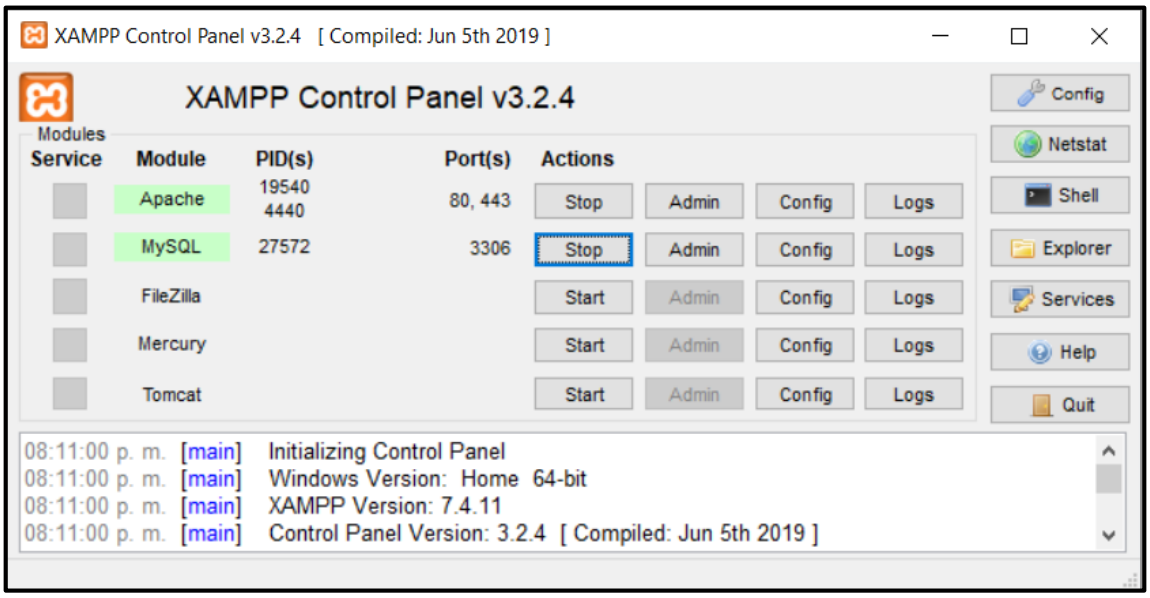

Figura 2.11 Interfaz de XAMPP.

2.3 htaccess

“Los ficheros *.htaccess* (o ‘ficheros de configuración distribuida’) facilitan una forma de realizar cambios en la configuración en contexto directorio. Un fichero, que contiene una o más directivas, se coloca en un documento específico de un directorio, y estas directivas aplican a ese directorio y todos sus subdirectorios” (The Apache Software Foundation, 2021).

Los ficheros *.htaccess* se pueden crear desde Visual Studio Code, y son útiles para funcionamientos adicionales de los directorios del servidor. Interesa su funcionamiento de redireccionamiento básico. Los ficheros *.htaccess* están escritos en la variante de Directivas Apache del lenguaje de las Expresiones Regulares Compatibles con Perl (PCRE). La PCRE es una librería de C de expresiones regulares inspirada por Perl.

CAPÍTULO 3.
DESARROLLO

3.1 Desarrollo de la Base de Datos

3.1.1 Modelo Entidad-Relación

Se realizó el siguiente modelo Entidad-Relación para la tienda CL-MaT:

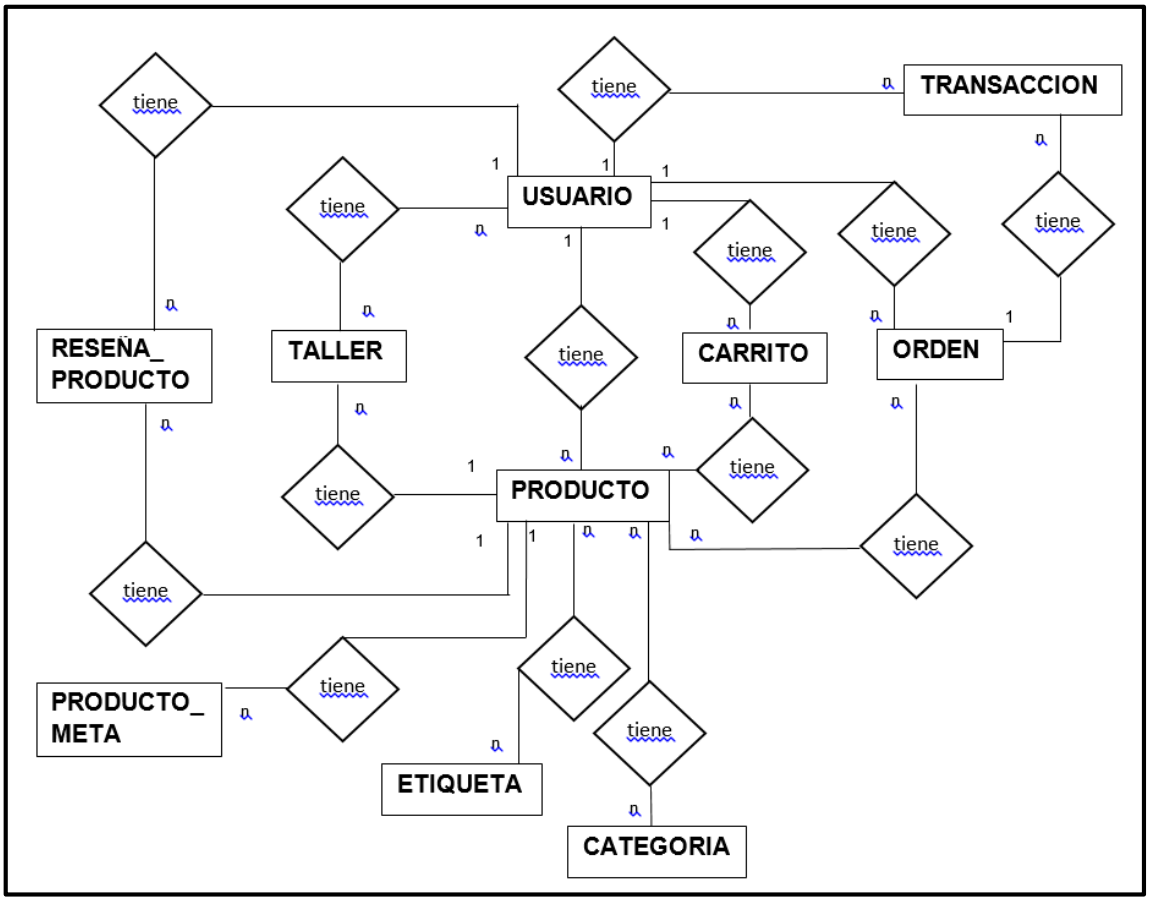

Figura 3.1 Modelo Entidad-Relación de la Base de Datos.

El centro del modelo es la tabla Producto; que puede pertenecer a diferentes Categorías y Etiquetas para su filtrado de búsqueda, tiene reseñas y se pueden desglosar a Talleres. Para manejar los diferentes recursos de los productos (imágenes, vídeos, archivos PDF, etc.) se maneja la tabla Producto_Meta para guardar las rutas de los recursos. El otro centro del modelo es la tabla Usuario, que contiene usuarios administradores que registran productos y usuarios normales. Los usuarios pueden dejar reseñas en los productos y pueden acceder a los talleres que han comprado. Los usuarios tendrán su Carrito, controlado por sesión, para guardar productos que desean comprar, en cuyo caso los usuarios podrán realizar una Orden con los productos a comprar y su respectiva Transacción.

3.1.2 Diccionario de Datos

A continuación se presenta el Diccionario de Datos. Las llaves primarias están resaltadas en negritas, mientras que las llaves foráneas están subrayadas.

Diccionario de Datos			
Columna	Tipo	Validaciones	Descripción
Tabla Usuario			
id	**bigint**	**not null, auto_increment**	**El id único para identificar al usuario.**
nombre	varchar(100)	default null	El nombre del usuario.
apellidos	varchar(100)	default null	Los apellidos del usuario.
telefono	varchar(15)	unique index	El número telefónico del usuario. Puede usarse para login y el registro.
email	varchar(50)	unique index	El correo electrónico del usuario. Puede usarse para login y el registro.
linea_a	varchar(50)	default null	La primera línea de la dirección.
linea_b	varchar(50)	default null	La segunda línea de la dirección.
ciudad	varchar(50)	default null	La ciudad de la dirección.
estado	varchar(50)	default null	La provincia o estado de la dirección.
pais	varchar(50)	default null	El país de la dirección.
contraseña	varchar(32)	not null	El hash de la contraseña generada por el algoritmo apropiado. Debemos evitar guardar las contraseñas reales o encriptadas.
admin	tinyint(1)	not null, default 0	La bandera para identificar si el usuario es administrador.
registrado	datetime	not null	Esta columna puede usarse para calcular el tiempo del usuario con la aplicación.
ultimo_login	datetime	default null	Puede usarse para identificar el último login del usuario.
Tabla Producto			
id	**bigint**	**not null, auto_increment**	**El id único para identificar al producto.**
id_usuario	bigint	foreign key, not null	El id de usuario para identificar al admin.
titulo	varchar(75)	unique index, not null	El título del producto a mostrarse en la página de la tienda y del producto.
titulo_meta	varchar(100)	default null	El título meta a usarse para el título del navegador y el SEO.
slug	varchar(100)	unique index, not null	El slug para formar el URL.
resumen	tinytext	default null	El resumen para mencionar los puntos importantes.
sku	varchar(100)	not null	La Unidad de Mantenimiento de Almacén para rastrear el inventario del producto.

precio	float	not null, default 0	El precio del producto.
descuento	float	not null, default 0	El descuento del producto.
cantidad	smallint(6)	not null, default 0	La cantidad disponible del producto.
en_venta	tinyint(1)	not null, default 0	Puede usarse para identificar si el producto está públicamente disponible para venderse.
es_taller	tinyint(1)	not null, default 0	Bandera para identificar si el producto es un taller o no.
creado	datetime	not null	Guarda la fecha y hora cuando el producto es creado.
actualizado	datetime	default null	Guarda la fecha y hora cuando el producto es actualizado.
publicado	datetime	default null	Guarda la fecha y hora cuando el producto es publicado en la tienda.
rebaja_inicio	datetime	default null	Guarda la fecha y hora cuando la venta con rebaja del producto empieza.
rebaja_fin	datetime	default null	Guarda la fecha y hora cuando la venta con rebaja del producto termina.
contenido	text	default null	La columna usada para guardar detalles adicionales del producto.
Tabla Taller			
id	**bigint**	**not null, auto_increment**	**El id único para identificar al taller.**
id_producto	bigint	foreign key, not null	El id del producto para identificar a cuál producto pertenece el taller.
duracion	smallint(6)	not null, default 0	El número de horas que tiene de duración el taller.
url	varchar(100)	not null	La URL para apuntar al taller.
Tabla Taller_Usuario			
id_usuario	**bigint**	**foreign key, not null**	**El id del usuario para identificar los talleres que ha comprado el usuario.**
id_taller	**bigint**	**foreign key, not null**	**El id del taller para identificar los talleres que ha comprado el usuario.**
Tabla Producto_Meta			
id	**bigint**	**not null, auto_increment**	**El id único para identificar el meta del producto.**
id_producto	bigint	foreign key, not null	El id del producto para identificar el producto padre.
tipo	enum	default null	El tipo del meta del producto puede ser imagen (0), video (1), actividad (2).
url	varchar(100)	not null	La URL para apuntar al recurso.

contenido	text	default null	La columna usada para guardar detalles adicionales al meta del producto.
Tabla Reseña_Producto			
id	**bigint**	**not null, auto_increment**	**El id único para identificar la reseña del producto.**
id_producto	bigint	foreign key, not null	El id del producto para identificar el producto padre.
id_padre	bigint	foreign key, default null	El id para identificar la reseña padre.
id_usuario	bigint	foreign key, default null	El id para identificar el usuario quien escribe la reseña.
titulo	varchar(100)	not null	El título de la reseña.
valuacion	smallint(6)	not null, default 0	La valuación de la reseña.
disponible	tinyint(1)	not null, default 0	Puede usarse para identificar si la reseña está públicamente disponible.
creado	datetime	not null	Guarda la fecha y hora cuando la reseña es presentada.
publicado	datetime	default null	Guarda la fecha y hora cuando la reseña es publicada.
contenido	text	default null	La columna usada para guardar los detalles de la reseña.
Tabla Categoria			
id	**bigint**	**not null, auto_increment**	**El id único para identificar la categoría.**
id_padre	bigint	foreign key, default null	El id del padre para identificar la categoría padre.
titulo	varchar(75)	unique index, not null	El título de la categoría.
titulo_meta	varchar(100)	default null	El título meta para usarse con el título del navegador y el SEO.
slug	varchar(100)	not null	El slug de la categoría para formar el URL.
activa	tinyint(1)	not null, default 0	Puede usarse para activar o desactivar categorías sin tener que eliminarlas.
contenido	text	default null	La columa usada para guardar los detalles de la categoría.
icono	varchar(100)	default null	La columna usada para guardar la ruta al recurso del ícono de la categoría.
Tabla Categoria_Producto			
id_producto	**bigint**	**foreign key, not null**	**El id del producto para identificar el producto.**
id_categoria	**bigint**	**foreign key, not null**	**El id de la categoría para identificar la categoría.**

Tabla Etiqueta			
id	**bigint**	**not null, auto_increment**	**El id único para identificar la etiqueta.**
titulo	varchar(75)	unique index, not null	El título de la etiqueta.
titulo_meta	varchar(100)	default null	El título meta para usarse con el título del navegador y el SEO.
slug	varchar(100)	not null	El slug de la etiqueta para formar el URL.
contenido	text	default null	La columna usada para guardar los detalles de la etiqueta.
Tabla Etiqueta_Producto			
id_producto	**bigint**	**foreign key, not null**	**El id del producto para identificar el producto.**
id_etiqueta	**bigint**	**foreign key, not null**	**El id de la etiqueta para identificar la etiqueta.**
Tabla Carrito			
id	**bigint**	**not null, auto_increment**	**El id único para identificar el carrito.**
id_usuario	bigint	foreign key, default null	El id del usuario para identificar el usuario o comprador asociado al carrito.
sesion	varchar(100)	not null	El id único de sesión asociado al carrito.
token	varchar(100)	not null	El token único asociado al carrito para identificar el carrito sobre múltiples sesiones. El mismo token puede pasarse a la Pasarela de Pago si se requiere.
estado	enum	not null, default 'nuevo'	El estado del carrito puede ser nuevo (0), carrito (1), checkout (2), pagado (3), completado (4), y abandonado (5).
creado	datetime	not null	Guarda la fecha y hora cuando el carrito es creado.
actualizado	datetime	default null	Guarda la fecha y hora cuando el carrito es actualizado.
contenido	text	default null	La columna usada para guardar los detalles adicionales del carrito.
Tabla Item_Carrito			
id	**bigint**	**not null, auto_increment**	**El id único para identificar el item del carrito.**
id_producto	bigint	foreign key, not null	El id del producto para identificar el producto asociado con el item del carrito.
id_carrito	bigint	foreign key, not null	El id del carrito para identificar el carrito asociado con el item del carrito.
sku	varchar(100)	not null	El SKU del producto mientras se está comprando.

precio	float	not null, default 0	El precio del producto mientras se está comprando.
descuento	float	not null, default 0	El descuento del producto mientras se está comprando.
cantidad	smallint(6)	not null, default 0	La cantidad del producto seleccionada por el usuario.
activo	tinyint(1)	not null, default 0	La bandera para identificar si el producto está activo en el carrito. Puede usarse para evitar que el mismo producto se añada al mismo carrito varias veces.
creado	datetime	not null	Guarda la fecha y hora cuando el item del carrito es creado.
actualizado	datetime	default null	Guarda la fecha y hora cuando el item del carrito es actualizado.
contenido	text	default null	La columna usada para guardar los detalles adicionales del item del carrito.
Tabla Orden			
id	**bigint**	**not null, auto_increment**	**El id único para identificar la orden.**
id_usuario	bigint	foreign key, default null	El id del usuario para identificar el usuario o comprador asociado a la orden.
sesion	varchar(100)	not null	El id único de sesión asociado a la orden.
token	varchar(100)	not null	El token único asociado a la orden para identificarla sobre múltiples sesiones. El mismo token puede pasarse a la Pasarela de Pago si se requiere.
estado_orden	enum	not null, default 'nueva	El estado de la orden puede ser nueva (0), checkout (1), pagada (2), fallida (3), enviada (4), entregada (5), regresada (6), y completada (7).
subtotal	float	not null, default 0	El precio total de los items de la orden.
descuento	float	not null, default 0	El descuento total de los items de la orden.
impuestos	float	not null, default 0	El impuesto sobre los items de la orden.
envio	float	not null, default 0	El costo de envío de los items de la orden.
total	float	not null, default 0	El precio total de la orden incluyendo impuestos y costos de envío. Excluye los descuentos.
codigo_promo	varchar(50)	default null	El código promo de la orden.
promocion	float	not null, default 0	El descuento total de la orden basada en el código promo y/o el descuento de la tienda.
total_general	float	not null, default 0	El total general de la orden a ser pagado por el comprador.

nombre	varchar(100)	default null	El nombre del usuario.
apellidos	varchar(100)	default null	Los apellidos del usuario.
telefono	varchar(15)		El número telefónico del usuario.
email	varchar(50)		El correo electrónico del usuario.
linea_a	varchar(50)	default null	La primera línea de la dirección.
linea_b	varchar(50)	default null	La segunda línea de la dirección.
ciudad	varchar(50)	default null	La ciudad de la dirección.
estado	varchar(50)	default null	La provincia o estado de la dirección.
pais	varchar(50)	default null	El país de la dirección.
creado	datetime	not null	Guarda la fecha y hora cuando la orden es creada.
actualizado	datetime	default null	Guarda la fecha y hora cuando la orden es actualizada.
contenido	text	default null	La columa usada para guardar los detalles adicionales de la orden.
Tabla Item_Orden			
id	**bigint**	**not null, auto_increment**	**El id único para identificar el item de la orden.**
id_producto	bigint	foreign key, not null	El id del producto para identificar el producto asociado con el item de la orden.
id_orden	bigint	foreign key, not null	El id de la orden para identificar la orden asociada con el item de la orden.
sku	varchar(100)	not null	El SKU del producto mientras se está comprando.
precio	float	not null, default 0	El precio del producto mientras se está comprando.
descuento	float	not null, default 0	El descuento del producto mientras se está comprando.
cantidad	smallint(6)	not null, default 0	La cantidad del producto seleccionada por el usuario.
creado	datetime	not null	Guarda la fecha y hora cuando el item de la orden es creado.
actualizado	datetime	default null	Guarda la fecha y hora cuando el item de la orden es actualizado.
contenido	text	default null	La columna usada para guardar los detalles adicionales del item de la orden.
Tabla Transaccion			
id	**bigint**	**not null, auto_increment**	**El id único para identificar la transacción.**

id_usuario	bigint	foreign key, not null	El id del usuario para identificar el usuario asociado con la transacción.
id_orden	bigint	foreign key, not null	El id de la orden para identificar la orden asociada con la transacción.
codigo	varchar(100)	not null	El id de pago proporcionado por la Pasarela de Pago.
tipo	enum	not null	El tipo de la transacción de la orden puede ser credito (0) o debito (1).
modo	enum	not null	El modo de la transacción de la orden puede ser offline (0), envio_contra_reembolso (1), cheque (2), draft (3), wired (4), y online (5).
estado	enum	not null, default 'nueva'	El estado de la transacción de la orden puede ser nueva (0), cancelada (1), fallida (2), pendiente (3), declinada (4), rechazada (5), y exitosa (6).
creado	datetime	not null	Guarda la fecha y hora cuando la transacción de la orden es creada.
actualizado	datetime	default null	Guarda la fecha y hora cuando la transacción de la orden es actualizada.
contenido	text	default null	La columna usada para guardar los detalles adicionales de la transacción.

Tabla 1. Diccionario de Datos.

3.1.3 Creación de la Base de Datos

Para crear la Base de Datos, es mejor manejar comandos SQL en un archivo para que se pueda compartir, cargar de nuevo y modificar directamente.

Al momento de la creación de la Base de Datos, se indica el conjunto de caracteres a usar: *utf8mb4*, cotejamiento *utf8mb4_general_ci* (este conjunto y cotejamiento permiten todo tipo de caracteres y operaciones entre ellos como comparación).

```
CREATE SCHEMA IF NOT EXISTS `artesanias_matematicas` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
```

Figura 3.2 Comando SQL para crear la Base de Datos.

A continuación se muestra la creación de la tabla 'Taller' con comandos SQL. Es importante cuidar los tipos de datos de cada columna, indicar la llave primaria, los índices y las llaves foráneas. Comúnmente, el motor de almacenamiento por defecto es *InnoDB* pero no está de más indicarlo en cada tabla.

```
CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Taller` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_producto` BIGINT NOT NULL,
  `duracion` SMALLINT(6) NOT NULL DEFAULT 0,
  `url` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_taller_producto` (`id_producto` ASC),
  CONSTRAINT `fk_taller_producto`
    FOREIGN KEY (`id_producto`)
    REFERENCES `artesanias_matematicas`.`Producto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 3.3 Ejemplo de creación de una tabla con comandos SQL.

Ya que está creada la base de datos y sus tablas, se realiza una inserción directa de algunos datos para poder trabajar sobre la página web. Se inserta: mínimo un usuario administrador, productos, talleres, recursos de imágenes en Producto_Meta, e información de categorías.

```
INSERT INTO `Categoria` (`id`, `id_padre`, `titulo`, `titulo_meta`, `slug`, `activa`, `contenido`, `icono`) VALUES
(1, NULL, 'Grado escolar', 'Escolaridad', 'Grado_escolar', 1, 'Nivel escolar recomendable para la utilización del producto', './img/logo1.png'),
(2, 1, 'Publico en general', 'Publico en general', 'Publico', 1, 'Cualquier persona', NULL),
(3, 1, 'Preescolar', 'Dirigido a Preescolar', 'Preescolar', 1, 'Preescolar', NULL),
(4, 1, 'Educación básica 1, 2 3 Primaria', '1o, 2o 3ero Primaria', 'Primero_segundo_tercero', 1, '1o, 2o y 3ro grado de educación Primaria', NULL),
(5, 1, 'Educación básica 4, 5 6 Primaria', '4o, 5o 6exto Primaria', 'Cuarto_Quinto_sexto', 1, '4o, 5o y 6o grado de educación Primaria', NULL),
(6, 1, 'Educación Secundaria', 'Educación Secundaria', 'Secundaria', 1, 'Secundaria', NULL),
(7, 1, 'Media superior', 'Dirigido a Bachillerato', 'Media_superior', 1, 'Preparatoria', NULL),
(8, 1, 'Superior', 'Superior', 'Superior', 1, 'Nivel licenciatura', NULL),
(9, 1, 'Posgrado', 'Posgrado', 'Posgrado', 1, NULL, NULL),
(10, NULL, 'Mas vendidos', 'Mas vendidos', 'Vendidos', 1, 'Productos que son los mas populares', NULL),
(11, NULL, 'Nuevos productos', 'Nuevos productos', 'Nuevos', 1, 'Productos agregados en el ultimo mes', NULL),
(12, NULL, 'Edición de colección', 'Edición de colección', 'Colección', 1, 'Productos especiales que no se volverán a producir en este formato ', NULL),
(13, NULL, 'Areas', 'Clasificación por areas', 'Clasificación_por_areas', 1, NULL, NULL),
(14, 13, 'Razonamiento Lógico', 'Razonamiento Lógico', 'Razonamiento_Lógico', 1, NULL, NULL),
(15, 13, 'Números', 'Números', 'Números', 1, NULL, NULL),
(16, 13, 'Geometría', 'Geometría', 'Geometría', 1, NULL, NULL),
(17, 13, 'Medidas', 'Medidas', 'Medidas', 1, NULL, NULL),
(18, NULL, 'Ofertas', 'Ofertas', 'Ofertas', 1, NULL, NULL);
```

Figura 3.4 Ejemplo de inserción de datos a la tabla Categoría con comandos SQL.

Todo lo anterior se puede manejar de igual manera desde la interfaz *phpMyAdmin*, si se tiene que hacer cambios se puede hacer en cualquiera de las dos partes mientras se actualicé la otra.

Finalmente, se debe crear un nuevo usuario para manejar la Base de Datos ya que no se debe manejar operaciones desde las aplicaciones web con *root*. En *phpMyAdmin* podemos crear un nuevo usuario '*webexterno*' al cual le daremos los privilegios básicos y una contraseña.

Figura 3.5 Cuentas de usuario desde phpMyAdmin.

3.1.4 Procedimientos Almacenados

Para realizar operaciones sobre la Base de Datos se van a utilizar Procedimientos Almacenados. Con estos, podemos enviar parámetros a la Base de Datos, realizar un conjunto de instrucciones y regresar a la aplicación web el resultado de la consulta o *query*. Es práctico crearlas desde *phpMyAdmin* y luego exportarlas para juntar los comandos SQL en el mismo archivo de la creación de la Base de Datos.

En *phpMyAdmin*, el menú tiene la opción llamada 'Rutinas'. Ahí podemos crear un Procedimiento Almacenado o una Función (Consulta rápida para regresar un valor) para la Base de Datos específica, y luego aparecerán en la jerarquía de la izquierda antes de las tablas.

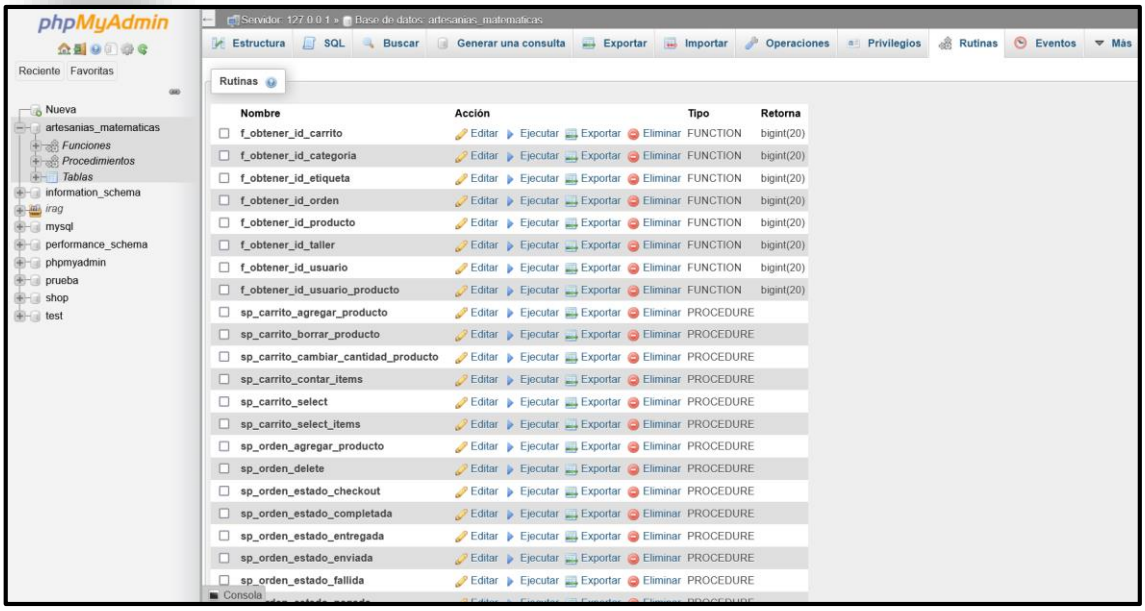

Figura 3.6 Interfaz de Rutinas en phpMyAdmin.

Aquí se puede crear todos los procedimientos necesarios para el funcionamiento de la página. Se ocupará las funciones para retornar las columnas *id* de las tablas, proceso que se aplicará bastante en los procedimientos almacenados. Las funciones se llamarán: 'f_obtener_*columna_tabla*'.

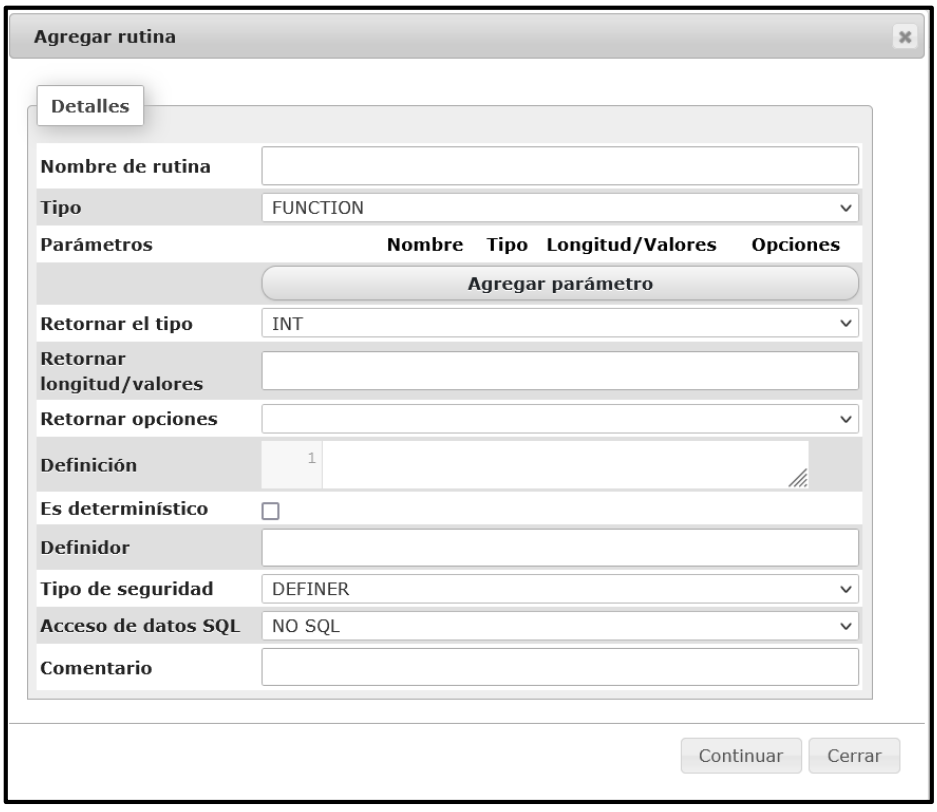

Figura 3.7 Interfaz para creación de una Función en phpMyAdmin.

En el campo de 'Definición', se debe escribir los comandos SQL que se desea realizar con la Función o Procedimiento Almacenado; y encerrarlos en un bloque BEGIN - END.

```sql
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `f_obtener_id_carrito`(`email` VARCHAR(50) CHARSET utf8mb4,
 `sesion` VARCHAR(100) CHARSET utf8mb4, `token` VARCHAR(100) CHARSET utf8mb4) RETURNS bigint(20)
    READS SQL DATA
    COMMENT 'Retornar el id_carrito usando las columnas sesion y token.'
BEGIN
RETURN (
    SELECT id
    FROM `Carrito`
    WHERE
        `Carrito`.`id_usuario` = f_obtener_id_usuario(email) AND
        `Carrito`.`sesion` = sesion AND
        `Carrito`.`token` = token
);
END$$
DELIMITER ;
```

Figura 3.8 Ejemplo de una Función exportada en comandos SQL.

Exportar la Función o Procedimiento Almacenado muestra los comandos SQL en su totalidad. Ya que se puede realizar varias sentencias, la exportación automáticamente agrega la línea 'DELIMITER $$' para poder cerrar la rutina con '$$' en vez del usual punto y coma.

Para mejor comprensión, se ha separado los Procedimientos Almacenados en diferentes tipos (usuario, producto, carrito, orden, transacción), basados en el tipo de operaciones que manejan. Sus nombres serán: 'sp_*tipo*_*nombre*'.

```sql
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_actualizar_informacion`(IN `nombre` VARCHAR(100) CHARSET utf8mb4,
 IN `apellidos` VARCHAR(100) CHARSET utf8mb4, IN `linea_a` VARCHAR(50) CHARSET utf8mb4, IN `linea_b` VARCHAR(50) CHARSET utf8mb4,
 IN `ciudad` VARCHAR(50) CHARSET utf8mb4, IN `estado` VARCHAR(50) CHARSET utf8mb4, IN `pais` VARCHAR(50) CHARSET utf8mb4,
 IN `email` VARCHAR(50) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Actualizar la información personal de un usuario.'
BEGIN
UPDATE `Usuario`
SET
    `Usuario`.nombre = nombre,
    `Usuario`.apellidos = apellidos,
    `Usuario`.linea_a = linea_a,
    `Usuario`.linea_b = linea_b,
    `Usuario`.ciudad = ciudad,
    `Usuario`.estado = estado,
    `Usuario`.pais = pais
WHERE `Usuario`.email = email;
END$$
DELIMITER ;
```

Figura 3.9 Ejemplo de un Procedimiento Almacenado de usuario con sentencia UPDATE.

```sql
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_select`(IN `inicio` INT, IN `limite` INT)
    READS SQL DATA
    COMMENT 'Obtener información de todos los productos.'
BEGIN
SELECT id_usuario, titulo, titulo_meta, slug, resumen, sku, precio, descuento, cantidad, en_venta,
       es_taller, creado, actualizado, publicado, rebaja_inicio, rebaja_fin, contenido
FROM `Producto`
ORDER BY creado DESC, titulo ASC
LIMIT inicio, limite;
END$$
DELIMITER ;
```

Figura 3.10 Ejemplo de un Procedimiento Almacenado de producto con sentencia SELECT.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_carrito_borrar_producto`(IN `titulo_producto` VARCHAR(75) CHARSET utf8mb4,
IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Eliminar un producto de un carrito.'
BEGIN
DELETE FROM `Item_Carrito` WHERE
    `Item_Carrito`.`id_producto` = f_obtener_id_producto(titulo_producto) AND
    `Item_Carrito`.`id_carrito` = f_obtener_id_carrito(email, sesion, token);
END$$
DELIMITER ;
```

Figura 3.11 Ejemplo de un Procedimiento Almacenado de carrito con sentencia DELETE.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_agregar_producto`(
    IN `titulo_producto` VARCHAR(75) CHARSET utf8mb4, IN `email` VARCHAR(50) CHARSET utf8mb4,
    IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4,
    IN `cantidad` SMALLINT(6), IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Agregar un producto a la orden.'
BEGIN
DECLARE id_producto BIGINT;
DECLARE id_orden BIGINT;
DECLARE sku VARCHAR(100) CHARSET utf8mb4 DEFAULT '';
DECLARE precio FLOAT;
DECLARE descuento FLOAT;
SET id_producto = f_obtener_id_producto(titulo_producto);
SET id_orden = f_obtener_id_orden(email, sesion, token);
SELECT `Producto`.`sku` INTO sku
FROM `Producto` WHERE `Producto`.`id` = id_producto;
SELECT `Producto`.`precio` INTO precio
FROM `Producto` WHERE `Producto`.`id` = id_producto;
SELECT `Producto`.`descuento` INTO descuento
FROM `Producto` WHERE `Producto`.`id` = id_producto;
INSERT INTO `Item_Orden`(id_producto,id_orden,sku,precio,descuento,cantidad,creado,contenido)
VALUES (id_producto,id_orden,sku,precio,descuento,cantidad,NOW(),contenido);
END$$
DELIMITER ;
```

Figura 3.12 Ejemplo de un Procedimiento Almacenado de orden con sentencia INSERT.

3.2 Diseño de las páginas web

3.2.1 Ambiente de Trabajo

Primero, se debe conocer como empezar a trabajar con XAMPP. XAMPP se instala en la carpeta *xampp* directamente en el Disco Duro (acceder desde *Mi Equipo*), y dentro de *xampp* se encuentra la carpeta *htdocs*. Dentro de *htdocs* se colocará el ambiente de trabajo, es decir todos los directorios y recursos necesarios para el funcionamiento local de la aplicación web. La ruta será: '*D:\xampp\htdocs\cimat\artesanias-matematicas\diseño*', y la carpeta *diseño* se manejará como la raíz de la aplicación web. Aquí se colocarán los archivos y directorios necesarios como se muestra en la siguiente imagen:

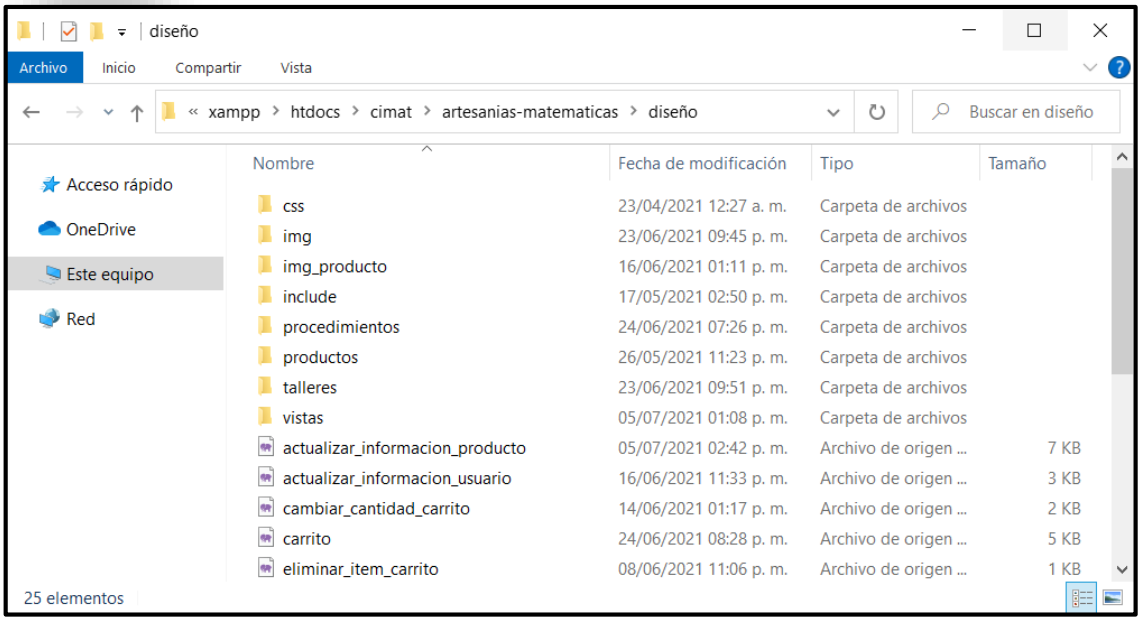

Figura 3.13 Directorio de la aplicación web dentro de XAMPP.

Cuando se inician los servicios de *Apache* y *MySQL* (que realmente es *MariaDB*) desde el panel de control de XAMPP, se podrá visualizar nuestra aplicación web si se entra a un navegador web y se escribe la dirección: 'localhost/*directorios_aplicacion_web*'. En este caso la dirección URL es: '*http://localhost/cimat/artesanias-matematicas/diseño*' y desde aquí se puede acceder a los archivos de la aplicación web que se encuentran en el directorio explicado anteriormente.

De igual manera, con esos servicios prendidos desde XAMPP, se puede acceder a la interfaz *phpMyAdmin* escribiendo la dirección URL '*localhost/phpmyadmin*' en un navegador web.

Finalmente, para empezar a trabajar se recomienda abrir el directorio indicado con Visual Studio Code usando click derecho, y desde ahí se puede manejar rápidamente todos los archivos del directorio.

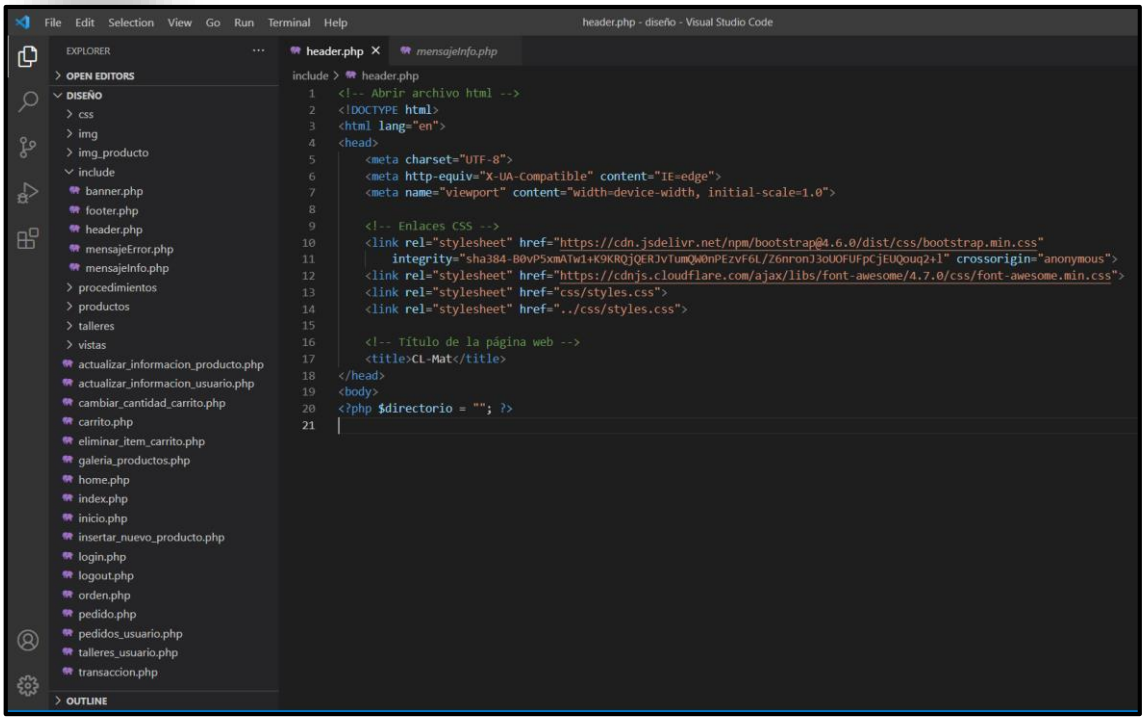

Figura 3.14 Interfaz de Visual Studio Code y el ambiente de trabajo.

3.2.2 Directorio 'include'

La carpeta "*include*" contendrá archivos *PHP* que todas (o la mayoría) de las páginas de la tienda online deberá incluir para su funcionamiento.

- **header**: el encabezado de la página contiene la sección <*head*> de una página *html* donde se añaden todos los enlaces necesarios para el funcionamiento y el diseño de la página (como el link a Bootstrap).

```
<!-- Abrir archivo html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Enlaces CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css"
        integrity="sha384-B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l" crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="css/styles.css">
    <link rel="stylesheet" href="../css/styles.css">

    <!-- Título de la página web -->
    <title>CL-Mat</title>
</head>
<body>
```

Figura 3.15 Archivo header.

- **footer**: el pie de página que contiene información de contacto y derechos de autor, además de cerrar un archivo *html* correctamente y añadir scripts *JavaScript* de Bootstrap para el funcionamiento de la página.

```
    <!-- Espacio en blanco para separar contenido -->
    <div class="espacio-blanco"></div>

    <!-- Pie de Página -->
    <div class="container-fluid" id="footer">
        <p> Centro de Investigación en Matemáticas, A.C., Jalisco S/N, Col. Valenciana CP:
            36023 Guanajuato, Gto, México, Apartado Postal 402, CP 36000
            <br>
            Tel. + 52 473 732 7155 / 735 0800 / Responsable de la información laura@cimat.mx
        </p>
        <br>
        <a href="https://www.cimat.mx/es/politica-de-privacidad-y-manejo-de-datos-personales-0">
            Política de Privacidad y Manejo de Datos Personales
        </a>
        <br>
        <p> © CIMAT 2021</p>
    </div>

    <!-- Scripts JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-Piv4xVNRyMGpqkS2by6br4gNJ7DXjqk09RmUpJ8jgGtD7zP9yug3goQfGII0yAns" crossorigin="anonymous"></script>

<!-- Cerrar archivo html -->
</body>
</html>
```

Figura 3.16. Archivo footer.

- **banner**: en este archivo colocaremos la barra de navegación que se situará en la parte superior de todas las páginas de manera fija. Este *navbar* sirve para poner enlaces rápidos en la mano del usuario para: visitar las páginas de productos, acceder a su cuenta, verificar su Carrito, entre otras cosas.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light mb-4" style="position:fixed;width:100%;">
  <a class="navbar-brand" href="https://www.conacyt.mx/">
    <img src="img/logoCONACYT.jpg" alt="logoCONACYT" style="width:200px;height:50px;">
  </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse pl-4 mr-4" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Inicio <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown"
          aria-haspopup="true" aria-expanded="false">
        Productos
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Talleres</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Productos</a>
        </div>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown"
          aria-haspopup="true" aria-expanded="false">
        Talleres
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Talleres</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Productos</a>
        </div>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown"
          aria-haspopup="true" aria-expanded="false">
        Nosotros
        </a>
```

Figura 3.17. Fragmento del archivo banner que contiene el navbar.

Figura 3.18 Ejemplo de Vista del navbar y footer.

- **mensajeError y mensajeInfo**: estos archivos contienen *modals* (ventanas que se colocan encima de las demás) para mostrar mensajes de errores, advertencias, de éxito, información; para lo que se requiera. Para mostrarlos en cualquier momento se manda a llamar una función *php* que contiene código *JavaScript* para mostrar *modals*.

```
<!-- Modal Mensaje Error -->
<div id="mensajeErr" class="modal fade" role="dialog">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h4 class="modal-title" id="cancelModal">Error</h4>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                <div id="mensajeErrTexto">texto</div>
            </div>
        </div>
    </div>
</div>

<?php
    //Enseñar Modal Error
    function mensaje_error($mensajeErr){ ?>
        <script type='text/javascript'>
            $('#mensajeErr').on('show.bs.modal', function(event) {
                var mensaje = "<?php echo $mensajeErr; ?>";
                $('#mensajeErrTexto').html(mensaje);
            });
            $('#mensajeErr').modal('show');
        </script> <?php
    }
?>
```

Figura 3.19. Modal para mensajes de error.

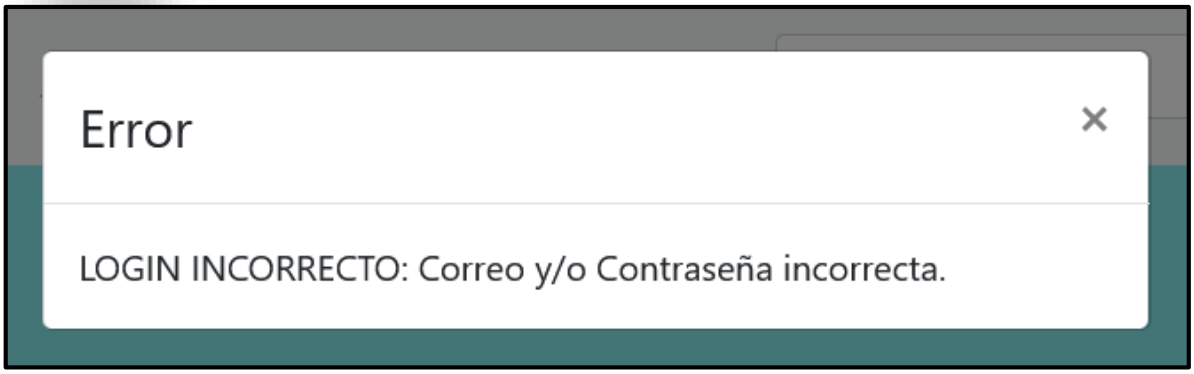

Figura 3.20. Diseño del Modal para mensajes de error.

En estos archivos se puede ver que la extensión es *.php* pero en su mayoría se maneja código HTML. Es útil crear todos los códigos con extensión *.php* ya que se puede agregar HTML y JavaScript con la etiqueta *<script>*. La combinación de todos estos elementos permiten tener un buen control del funcionamiento de las páginas web (por ejemplo, manejar variables de PHP en las etiquetas de HTML). Para juntar todos estos archivos, y más, se utilizan las funciones de PHP *include()*, *include_once()*, *require()* o *require_once()*. Así se puede mandar a llamar otros códigos PHP en la página actual.

```php
<?php
    // Inicializar la sesión
    session_start();

    // Agregar Vistas
    include_once('include/header.php');
    include_once('include/banner.php');
    include_once('vistas/vista_inicio.php');
    include_once('include/footer.php');
?>
```

Figura 3.21 Archivo inicio.php.

3.2.3 Vistas

Primero se debe pensar en los elementos que debe tener las páginas web, y por eso se realiza el diseño de las vistas antes que el funcionamiento. Con el atributo *class* indicamos a que clases de CSS pertenecen los elementos, es aquí donde se indica los diseños de Bootstrap como los colores o los contenedores. Si se requiere elementos con contenido dependiente o que simplemente no aparezcan bajo ciertas circunstancias, se puede pasar todo ese código HTML al código PHP que controla el funcionamiento de la página y en las vistas solamente imprimir la variable (aunque hay que asegurarse que siempre esté inicializada para evitar errores).

```
<div class="container"><div class="row"><div class="col text-center">
    <h2>Talleres comprados <a href="home.php" class="btn btn-warning btn-sm float-right">Regresar</a></h2>
    <table id="tabla-talleres">
        <thead>
            <tr>
                <th>Producto</th>
                <th>Descripción</th>
                <th>Duración</th>
                <th>Resumen</th>
                <th>Creado</th>
                <th>URL</th>
            </tr>
        </thead>
        <tbody>
            <?php
                echo $tabla;
            ?>
        </tbody>
    </table>
</div></div></div>
```

Figura 3.22 Ejemplo de un archivo Vista.

Además de las etiquetas básicas de HTML, hay algunos elementos complejos que se han usado para realizar las páginas más complicadas:

- *Carousel*: Un Carrusel de imágenes sirve para pasar las imágenes como diapositivas en la página web.

```
<!--Carousel-->
<div class="container">
    <div class="row">
        <div class="col">
            <div id="demo" class="carousel slide" data-ride="carousel">

                <!-- Indicadores -->
                <ul class="carousel-indicators">
                <li data-target="#demo" data-slide-to="0" class="active"></li>
                <li data-target="#demo" data-slide-to="1"></li>
                <li data-target="#demo" data-slide-to="2"></li>
                </ul>

                <!-- Presentación de imágenes -->
                <div class="carousel-inner">
                <div class="carousel-item active">
                    <img src="img/inicio-1.png" alt="Experimentos">
                </div>
                <div class="carousel-item">
                    <img src="img/inicio-2.png" alt="Productos">
                </div>
                <div class="carousel-item">
                    <img src="img/inicio-3.png" alt="Talleres">
                </div>
                </div>

                <!-- Controles de derecha e izquierda -->
                <a class="carousel-control-prev carousel-flechas" href="#demo" data-slide="prev">
                    <span class="carousel-control-prev-icon"></span>
                </a>
                <a class="carousel-control-next carousel-flechas" href="#demo" data-slide="next">
                    <span class="carousel-control-next-icon"></span>
                </a>

            </div>
        </div>
    </div>
</div>
<!--Fin Carousel-->
```

Figura 3.23 Código para realizar un carrusel de imágenes.

- *Form*: La etiqueta <*form*> sirve para realizar formularios en la página web, es para solicitar información al usuario. Puede servir para realizar el Login a la página, para actualizar la información de usuario, o en casos específicos que se necesite mandar una información introducida por el usuario al código PHP.

```
<!-- Columna Registro -->
<div class="col border border-light flex-fill pb-2">
    <h4 class="text-center">Crear una cuenta</h4>
    <hr class="separador-blanco">
    <!-- Código php manda la acción del form a la propia página y limpia de caractéres "maliciosos" -->
    <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>" method="post">
        <div class="form-group">
            <label for="email-registro" class="font-weight-bold">Correo Electrónico:</label>
            <input type="email" class="form-control" placeholder="Correo Electrónico" id="email-registro" name="email-registro">
            <!-- Mostrar contenido de una variable php para funcionamiento de las páginas -->
            <span class="error"><?php echo $correoRegErr;?></span>
        </div>
        <div class="form-group">
            <label for="telefono-registro" class="font-weight-bold">Número de Celular:</label>
            <input type="tel" class="form-control" placeholder="Número de Celular" id="telefono-registro" name="telefono-registro"
                pattern="[0-9]{10}">
            <span class="error"><?php echo $telefonoRegErr;?></span>
        </div>
        <div class="form-group">
            <label for="pwd-registro" class="font-weight-bold">Contraseña:</label>
            <input type="password" class="form-control" placeholder="Contraseña" id="pwd-registro" name="pwd-registro">
            <span class="error"><?php echo $contraseñaRegErr;?></span>
        </div>
        <button type="submit" class="btn btn-success" name="boton-registro">CREAR</button>
    </form>
</div>
```

Figura 3.24 Código para realizar un formulario.

- *Pagination*: Paginar sirve para separar contenido de una misma página en diferentes subpáginas. En este caso es útil para dividir las consultas de los productos en un número por página específico; para evitar que la enorme cantidad de información aparezca de golpe en la misma página.

```
<nav aria-label="Page navigation example">
  <ul class="pagination justify-content-center">
    <li class="page-item disabled"><a class="page-link" href="#">Anterior</a></li>
    <li class="page-item active"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item"><a class="page-link" href="#">Siguiente</a></li>
  </ul>
</nav>
```

Figura 3.25 Ejemplo de código para paginación.

- *Card-deck*: La clase de Cartas son contenedores flexibles que pueden manejar contenidos, encabezados, imágenes, entre otras cosas. Usar un *card-deck* te permite agrupar varias Cartas para tener varios recuadros similares y mostrar información. Por ejemplo, las opciones de usuario en el menú de *home*.

```php
<!-- Recuadros de opciones -->
<div id="card-deck-perfil" class="card-deck">
    <a href="pedidos_usuario.php" class="card-perfil">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Mis Pedidos</h5>
                <p class="card-text">Revisa tu historial de pedidos.</p>
            </div>
        </div>
    </a>
    <a href="talleres_usuario.php" class="card-perfil">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Mis Talleres</h5>
                <p class="card-text">Accede a tus talleres comprados.</p>
            </div>
        </div>
    </a>
    <a href="actualizar_informacion_usuario.php" class="card-perfil">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Mi Información</h5>
                <p class="card-text">Modifica tu información para realizar pedidos.</p>
            </div>
        </div>
    </a>
    <a href="carrito.php" class="card-perfil">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Su carrito</h5>
                <p class="card-text">Accede a su carrito virtual para proceder al pago.</p>
            </div>
        </div>
    </a>
    <?php echo $opciones_admin; ?>
</div>
```

Figura 3.26 Código para realizar un card-deck.

3.2.4 Archivo CSS

Aparte de utilizar los diseños de Bootstrap, se crea un propio archivo *CSS* para manejar estilos de diseño específicos en las vistas. De esta manera se puede manejar distintos colores, cuidar la colocación de contenido, quitar o poner márgenes, entre otras muchas cosas. Los diseños de Bootstrap son muy útiles pero juntado a esto se le puede dar una identidad a la página.

Desde CSS se puede modificar las reglas de las etiquetas HTML, pero también se pueden crear clases propias (como lo maneja Bootstrap). Los nombres de las clases van precedidas por un punto, e incluso se puede modificar reglas de Bootstrap (en especial si se utiliza *!important*). Las clases son para un conjunto de elementos HTML, pero usando identificadores las reglas solamente aplicarían a un elemento por página. Para usar identificadores hay que agregar la propiedad *id* a las etiquetas HTML, y para llamarlas en CSS se preceden con un '#'. Además de todo esto, se puede especificar reglas para elementos dentro de un identificador, clase o etiqueta; aparte de propiedades especiales como enlaces visitados.

```css
html, body {
    margin:0;
    padding: 0;
    height: 100vh;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
}

body {
    background-color: cyan;
}

a {
    text-decoration: underline;
}
```

Figura 3.27 Ejemplo de reglas CSS para etiquetas HTML.

```css
.espacio-blanco{
    min-height: 100px;
}

.separador-blanco{
    color: white;
    background-color: white;
    border: double white;
}

.error {
    color: #FF0000;
}

.navbar {
    z-index: 10;
    min-height: 80px;
    max-height: 80px;
}

.carousel-inner img {
    width: 100%;
    height: 100%;
}
```

Figura 3.28 Ejemplo de reglas CSS para clases.

```css
#footer{
    margin-top: auto;
    background-color: black;
    color: white;
    text-align: center;
    border: 1px solid whitesmoke;
    font-weight: bold;
    border-top: 5px solid #EEEEEE !important;
}

#demo-1{
    width: 100%;
    height: 100%;
    padding: 0;
    margin: 0;
}

#demo-2{
    background-color: #010101;
    border: 2px double black;
}

#introduccion{
    margin-top: 32px;
    background-color: indigo;
    color: lightcyan;
    border: 4px dotted black;
    text-align: justify;
}
```

Figura 3.29 Ejemplo de reglas CSS usando identificadores.

3.3 Funcionalidad de las páginas web

Como ya se ha mencionado, la operatividad de la página web se codificará en PHP. En estos archivos se manejará la conexión con la Base de Datos y consultas, manejo de la información de formularios y operaciones de la página web como mostrar información o realizar un pedido.

3.3.1 Modelos

En la carpeta '*procedimientos*' se colocarán todos los modelos que manejarán la conexión con la Base de Datos y su procesamiento. De esta forma se puede mandar a llamar todos los Procedimientos Almacenados creados anteriormente desde cualquier parte de la operatividad y la codificación es más ordenada.

Se puede manejar estos modelos como clases, como lenguaje de programación orientado a objetos. En la clase *Conexion* se maneja la función para conectarse a la Base de Datos. Y se usará este modelo para los demás. Aquí es importante haber creado un nuevo usuario para no utilizar *root*.

```php
<?php
    class conexion {
        //CONECTAR A LA BD
        protected function conectar(){
            $dbhost = 'localhost';
            $dbuser = 'webexterno';
            $dbpass = '[redacted]';
            $dbdata = 'artesanias_matematicas';
            try{
                $conn = new mysqli($dbhost,$dbuser,$dbpass,$dbdata);
                $conn->set_charset("utf8mb4");
                return $conn;
            } catch(Exception $ex){
                die('Conexión fallida: '.$conn->connect_error);
            }
        }
    }
?>
```

Figura 3.30 Clase para la conexión con la Base de Datos.

Se añade este archivo en los otros modelos para poder usar la función *conectar()* y se hereda la clase con la sentencia *extends*. Se crea un modelo para cada tipo de Procedimiento Almacenado que se definió anteriormente (usuario, producto, carrito, orden, transacción), y en estos modelos se usa una función para llamar a cada Procedimiento Almacenado definido (realmente no es necesario separar por tipos pero separa mejor el código). Todos los parámetros definidos en los procedimientos almacenados se deben pasar desde los códigos PHP como parámetros de las funciones de los modelos. Todas las funciones siguen esta sintaxis:

```php
<?php
    // Añadir archivo conexion.php para usar su modelo
    require_once('conexion.php');

    class transaccion extends conexion {

        // Función para crear una nueva orden
        function nueva_transaccion($correo, $sesion, $token, $codigo, $tipo, $modo, $contenido){
            $conn = $this->conectar();
            $sql = "call `artesanias_matematicas`.sp_transaccion_nueva('$correo','$sesion',
                                                '$token','$codigo','$tipo','$modo','$contenido');";
            if ($conn->query($sql) === TRUE) {
                $conn->close();
                return 1;
            } else {
                $resultado = $conn->error;
                $conn->close();
                return $resultado;
            }
        }

    }
?>
```

Figura 3.31 Ejemplo de modelo con función no SELECT.

Dependiendo de la consulta que se realice en el Procedimiento Almacenado, va a retornar diferentes tipos de datos. Si el Procedimiento Almacenado no contiene un SELECT (es decir, contiene UPDATE, DELETE o INSERT) se usa la sintaxis anterior. Si contiene un SELECT, se usa la siguiente sintaxis. Si el SELECT regresará más de una fila, se debe agregar corchetes '*[]*' a la línea '*$data = $row;*' como se aprecia a continuación:

```php
// Función para obtener los pedidos comprados por el usuario
function obtener_pedidos($correo) {
    $conn = $this->conectar();
    $sql = "call `artesanias_matematicas`.sp_usuario_obtener_pedidos('$correo');";
    if ($resultado=$conn->query($sql)) {
        $data = array();
        while($row=mysqli_fetch_array($resultado)) {
            $data[] = $row;
        }
        $conn->close();
        return $data;
    } else {
        die("Error: ".$conn->error);
    }
}

// Función para verificar si el Usuario es administrador
function es_admin($correo) {
    $conn = $this->conectar();
    $sql = "call `artesanias_matematicas`.sp_usuario_es_admin('$correo');";
    if ($resultado=$conn->query($sql)) {
        $data = array();
        while($row=mysqli_fetch_array($resultado)) {
            $data = $row;
        }
        $conn->close();
        return $data;
    } else {
        die("Error: ".$conn->error);
    }
}
```

Figura 3.32 Ejemplo de modelo con función SELECT.

3.3.2 Códigos PHP

Cuando se está codificando es importante manejar toda la operatividad antes de añadir las vistas. Por ejemplo, la función para redirigir *header()* no funciona si se ha impreso contenido en la página. Y más importante aún es inicializar la sesión con la función *session_start()* para poder utilizar el arreglo global *$_SESSION[]* en el código, este arreglo sirve para guardar información durante diferentes páginas. Y si el usuario no debería estar en una página específica dependiendo de las circunstancias, se debe verificar al inicio del código para en ese caso mandarlo a otra página ideal.

```php
<?php
    // Inicializa la sesión
    session_start();

    // Verificar que el usuario esté loggeado
    if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){
        header("location: login.php");
        exit;
    }
```

Figura 3.33 Verificación inicial de los códigos PHP.

Para obtener la información de un formulario de la vista de una página, se utiliza el método POST de HTTP. Cuando se da clic a un botón de tipo *submit*, se manda la información del formulario a la variable global *$_POST[]* si así se ha indicado en la vista. Se verifica si hay una petición POST en el servidor, se checa cuál botón fue oprimido y se recupera la información introducida por el usuario.

```php
//Declarar variables
$correoErr = $contraseñaErr = $correoRegErr = $contraseñaRegErr = $telefonoRegErr ="";
$correo = $contraseña = $correoReg = $contraseñaReg = $telefonoReg = "";

//Obtener variables del formulario
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if(isset($_POST["boton-login"])){
        if (empty($_POST["email"])) {
            $correoErr = "Campo obligatorio";
        } else {
            $correo = test_input($_POST["email"]);
        }
        if (empty($_POST["pwd"])) {
            $contraseñaErr = "Campo obligatorio";
        } else {
            $contraseña = test_input($_POST["pwd"]);
        }
    } else {
        if (empty($_POST["email-registro"])) {
            $correoRegErr = "Campo obligatorio";
        } else {
            $correoReg = test_input($_POST["email-registro"]);
        }
        if (empty($_POST["pwd-registro"])) {
            $contraseñaRegErr = "Campo obligatorio";
        } else {
            $contraseñaReg = test_input($_POST["pwd-registro"]);
        }
        if (empty($_POST["telefono-registro"])) {
            $telefonoRegErr = "Campo obligatorio";
        } else {
            $telefonoReg = test_input($_POST["telefono-registro"]);
        }
    }
}
```

Figura 3.34 Recuperar información de un formulario.

Para poder mandar a llamar algún Procedimiento Almacenado, se debe añadir el modelo correcto al código y crear un nuevo objeto de la clase del modelo. Así se puede mandar a llamar la función correspondiente y pasarle todos los parámetros necesarios. Se recibe el resultado, se procesa para saber si hubo algún tipo de error o si se quiere recuperar información de un SELECT. Todas las conexiones a la Base de Datos se deben manejar de esta forma.

```php
// Verificar correo y contraseña Login
if(!empty($correo) && !empty($contraseña)){
    $usuario = new Usuario();
    $login = $usuario->login($correo, $contraseña);
    if($login["0"]["resultado"] == '1') {

        // Verificar si el usuario es administrador
        $admin = $usuario->es_admin($correo);
        //var_dump($admin);

        // Guardar datos de sesión
        $_SESSION["loggedin"] = true;
        $_SESSION["correo"] = $correo;
        $_SESSION["admin"] = $admin["admin"];

        // Actualizar último login
        $ultimo_login = $usuario->last_login($correo);

        // Redireccionar a la página de home
        header("location: home.php");
    }
    else {
        $mensajeError = 'LOGIN INCORRECTO: Correo y/o Contraseña incorrecta.';
    }
}
```

Figura 3.35 Procedimiento de utilizar un modelo y procesar el resultado de una consulta.

Como se puede ver en la imagen anterior, se puede usar el modelo para más funciones (más Procedimientos Almacenados). También se puede apreciar cómo utilizar la variable $_SESSION. La función *var_dump()* es muy útil para debuggear, imprime en la página web el tipo y contenido de una variable. Al procesar un resultado de una consulta, puede interesar mandar mensajes de error o mensajes de éxito y para ello ya se había mostrado esos archivos correspondientes. Ahora el último paso para mostrarlos en pantalla desde el código PHP: después de las funciones nuevas y de añadir las vistas, se checa si hay algún mensaje por mostrar y se manda a llamar la función declarada anteriormente.

```php
// Función para limpiar campos
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

// Añadir Vistas
require_once('include/header.php');
require_once('include/banner.php');
require_once('vistas/vista_login.php');
require_once('include/mensajeError.php');
require_once('include/mensajeInfo.php');
require_once('include/footer.php');

//Mandar mensaje de error si existe
if(isset($mensajeError) && !empty($mensajeError)){
    mensaje_error($mensajeError);
}

//Mandar mensaje de éxito si existe
if(isset($mensajeInfo) && !empty($mensajeInfo)){
    mensaje_info($mensajeInfo);
}
```

Figura 3.36 Procedimiento para mandar mensajes de error o éxito con modals.

A veces se desea mostrar diferente contenido en la vista dependiendo de los procesos que se manejen. Para ello en las vistas solamente se colocó la sentencia de imprimir una variable PHP. Ahora hay que inicializar esa variable en el código PHP y asignarle código HTML en forma de String.

```php
// Opciones administrativas para administrador
$opciones_admin = '';
if(isset($_SESSION["admin"]) && $_SESSION["admin"] == "1"){
    $opciones_admin = "
                    <a href='insertar_nuevo_producto.php' class='card-perfil'>
                        <div class='card bg-dark text-white'>
                            <div class='card-body'>
                                <h5 class='card-title'>Nuevo Producto</h5>
                                <p class='card-text'>Insertar un nuevo producto a la base de datos.</p>
                            </div>
                        </div>
                    </a>
                    <a href='actualizar_informacion_producto.php' class='card-perfil'>
                        <div class='card bg-dark text-white'>
                            <div class='card-body'>
                                <h5 class='card-title'>Actualizar Producto</h5>
                                <p class='card-text'>Modificar la información de un producto de la base de datos.</p>
                            </div>
                        </div>
                    </a>
    ";
}
```

Figura 3.37 Procedimiento para utilizar código HTML en una variable de PHP.

La sesión es un aspecto importante porque permite utilizar la variable $_SESSION. Cuando se necesite de dejar de usar por ciertos procesos, hay que quitar la información de $_SESSION con *unset()* o directamente destruir la sesión para renovarla cuando sea el proceso oportuno.

```php
<?php
    // Inicia la sesión
    session_start();

    // Elimina todas las variables de la sesión
    $_SESSION = array();

    // Destruye la sesión
    session_destroy();

    // Redirige a la página de Login
    header("location: login.php");
    exit;
?>
```

Figura 3.38 Procedimiento para destruir una sesión.

Hay varios procesos, en especial los procesos del carrito y orden, que utilizan la sesión para identificar registros en la Base de Datos. Se puede usar la función *session_id()* para obtener el identificador único de la sesión actual (hasta que se destruya o renueve).

```php
// Obtener productos de carrito
$correo = $_SESSION["correo"];
$sesion = session_id();
$token = $sesion.$correo;
if(isset($_SESSION["token"])) $token = $_SESSION["token"];
$carrito = new Carrito();
$items = $carrito->select_items($correo, $sesion, $token);
//var_dump($items);
$pedido = "<h1>Realizar el pedido: <a href='pedido.php' class='btn btn-primary btn-lg mt-2'>Orden</a></h1>";
if(empty($items)) {
    $tabla = "<tr><td colspan='7'>Su carrito está vacío</td></tr>";
    $pedido = "";
}
```

Figura 3.39 Procedimiento para utilizar el identificador de la sesión en los procesos.

Cuando se recuperan muchos registros de la Base de Datos, una forma sencilla de manejar las filas es usando el ciclo *foreach*. Así, en cada iteración se pasa al siguiente registro y se puede recuperar información más fácil. Cada registro de un SELECT acomoda la información directamente con el nombre de la columna, por lo que no se vuelve complejo manejar la información de un SELECT.

```php
// Variable para guardar talleres
$talleres = array();

// Agregar productos a la orden
foreach($items as $item){
    $producto_titulo = $item["titulo"];
    $producto_cantidad = $item["cantidad"];
    $producto_contenido = $item["contenido"];
    $resultado = $orden->insert_item($producto_titulo, $correo, $sesion, $token, $producto_cantidad, $producto_contenido);
    if($resultado != 1) {
        $resultado = $orden->eliminar_orden($correo, $sesion, $token);
        echo "<h1>ERROR EN AGREGAR '$producto_titulo' A LA ORDEN. INTENTAR DE NUEVO O COMUNICARSE CON UN ADMINISTRADOR.
            <br> $resultado</h1>";
        exit;
    }
    if($item["es_taller"] == "1"){
        $talleres[] = $producto_titulo;
    }
}

// Añadir talleres a cuenta de Usuario
$usuario = new Usuario();
foreach($talleres as $taller){
    $resultado = $usuario->agregar_taller($correo, $taller);
    if($resultado != 1){
        $resultado = $orden->eliminar_orden($correo, $sesion, $token);
        echo "<h1>ERROR EN AGREGAR '$taller' A LA CUENTA DE USUARIO. INTENTAR DE NUEVO O COMUNICARSE CON UN ADMINISTRADOR.
            <br> $resultado</h1>";
        exit;
    }
}
```

Figura 3.40 Procedimiento para utilizar foreach con consultas SELECT.

El método GET de HTTP sirve para compartir parámetros entre páginas desde la dirección URL. Al final de la dirección URL se puede agregar los parámetros de la siguiente forma: '?*parametro1=valor&parametro2=valor*'. Esto indica que se tiene dos parámetros con sus valores respectivos. Para recuperarlos en la siguiente página a donde accedimos solo hay que verificar que existan los parámetros en la variable global *$_GET*.

```php
// Cambiar cantidad producto
$cantidad_nueva = intval($cantidad_producto)-1;
$boton_menos = "<a class='enlace-normal' href='cambiar_cantidad_carrito.php?titulo=$titulo_producto
            &correo=$correo&sesion=$sesion&token=$token&cantidad=$cantidad_nueva&taller=$esTaller_producto'>
                <button class='btn btn-warning'><i class='fa fa-minus'></i></button>
            </a>";
if($cantidad_nueva == 0 || $esTaller_producto == "1") $boton_menos = '';
$cantidad_nueva = intval($cantidad_producto)+1;
$boton_mas = "<a class='enlace-normal' href='cambiar_cantidad_carrito.php?titulo=$titulo_producto
            &correo=$correo&sesion=$sesion&token=$token&cantidad=$cantidad_nueva&taller=$esTaller_producto'>
                <button class='btn btn-success'><i class='fa fa-plus'></i></button>
            </a>";
if($cantidad_nueva == 10 || $esTaller_producto == "1") $boton_mas = '';
```

Figura 3.41 Procedimiento para utilizar el método GET.

```php
// Variables GET
if(isset($_GET["titulo"]) && isset($_GET["correo"]) && isset($_GET["sesion"]) &&
    isset($_GET["token"]) && isset($_GET["cantidad"]) && isset($_GET["taller"])){
    $titulo = $_GET["titulo"];
    $correo = $_GET["correo"];
    $sesion = $_GET["sesion"];
    $token = $_GET["token"];
    $cantidad = intval($_GET["cantidad"]);
    $taller = $_GET["taller"];
    if($correo == $_SESSION["correo"] && $sesion == session_id() &&
        $cantidad > 0 && $cantidad < 10 && $taller == "0"){
        $carrito = new Carrito();
        $resultado = $carrito->cambiar_cantidad_producto($titulo,$correo,$sesion,$token,$cantidad);
    }
}
```

Figura 3.42 Procedimiento para recuperar parámetros del método GET.

3.3.3 Páginas de productos y talleres

Para el manejo de las páginas de los productos y talleres, no se puede crear un archivo PHP por cada producto; es inviable. Una solución es utilizar el método GET para enviar como parámetro el nombre del producto seleccionado o algo similar, y así en una página general para productos se puede recuperar la información del producto seleccionado. Una desventaja del método GET es que es algo visible para todos en la dirección URL de la página. Para manejar cuestiones de redireccionamiento se puede usar los ficheros *.htaccess* como una solución. Así, en vez de mandar parámetros con el método GET; se puede escribir directamente el *slug* del producto (un *slug* es la parte final de una dirección URL para identificar un recurso específico) y el fichero *.htaccess* del directorio correspondiente se haría cargo.

```php
// Generar ruta correcta para producto o taller
$ruta = "productos/$producto_slug";
//$ruta = "productos/producto.php?slug=$producto_slug";
if($producto_es_taller == '1'){
    $ruta = "talleres/$producto_slug";
    //$ruta = "talleres/taller.php?slug=$producto_slug";
}
```

Figura 3.43 Procedimiento para utilizar slugs en una dirección URL.

El fichero *.htaccess* debe redireccionar las peticiones (es decir, las direcciones URL) a un solo archivo dónde se manejará las solicitudes, se recupera el *slug* de la dirección URL y se obtiene la información del producto de la Base de Datos.

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ index.php [NC,L]
```

Figura 3.44 Fichero .htaccess para manejar el uso de slugs.

```php
// Obtener slug del url
if(isset($_GET["slug"])) $slug = $_GET["slug"];
else $slug = basename($_SERVER["REQUEST_URI"]);

// Obtener información de producto
$producto = new Producto();
$resultado = $producto->select_slug($slug);
//var_dump($resultado);

// Resultado vacío = Producto Inexistente
if(empty($resultado)){
    header("location: producto_no_disponible.php");
}
```

Figura 3.45 Procedimiento para recuperar el slug de la dirección URL.

CAPÍTULO 4.
RESULTADOS

4.1 Archivos

A continuación se muestra la estructura final de los archivos y directorios en los que se trabajó:

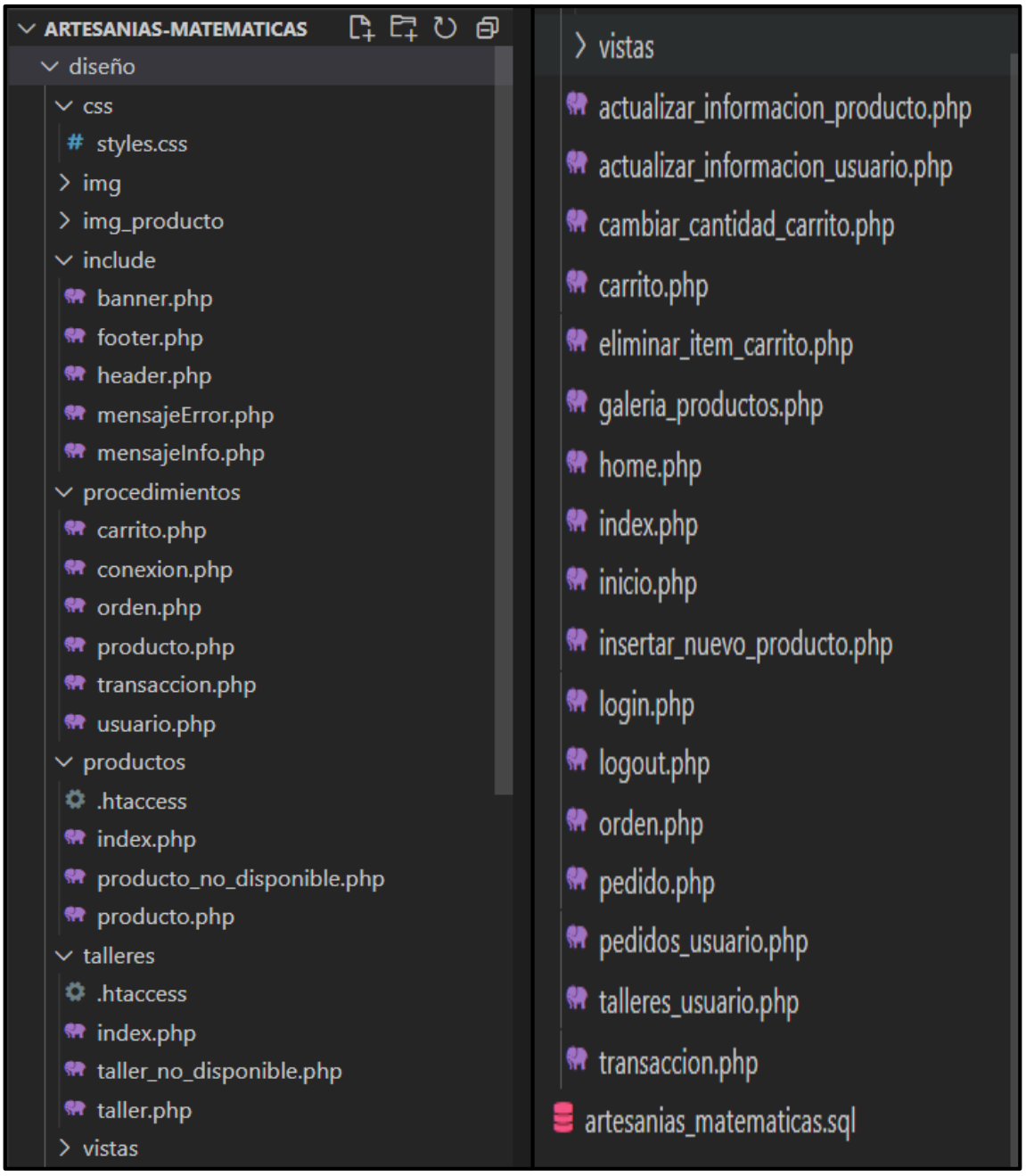

Figura 4.1 Estructura de los Archivos y Directorios.

El Directorio *img* contiene algunas imágenes usadas de prueba y otras imágenes que corresponden a contenido de la página web. El Directorio *img_producto* contiene imágenes de los productos como recursos las cuáles se llaman durante el proceso de la página.

4.2 Modelos

A continuación se describen las clases de los modelos y sus métodos que conectan a la Base de Datos.

4.2.1 Modelo Conexion

Método	Descripción
conectar()	Función para conectarse a la Base de Datos con el usuario ‘webexterno’.

Tabla 2. Modelo Conexión.

4.2.2 Modelo Usuario

Método	Descripción
login()	Función para realizar login
registro()	Función para realizar registro con solo correo y contraseña
select_info()	Función para obtener información del usuario
update_info()	Función para modificar información de usuario
last_login()	Función para actualizar la fecha del último login del usuario
crear_carrito()	Función para crear el carrito del usuario
agregar_taller()	Función para agregar un taller comprado a la cuenta de usuario
obtener_talleres()	Función para obtener los talleres comprados por el usuario
obtener_pedidos()	Función para obtener los pedidos comprados por el usuario
es_admin()	Función para verificar si el Usuario es administrador

Tabla 3. Modelo Usuario.

4.2.3 Modelo Producto

Método	Descripción
select()	Función para realizar select
total_productos()	Función para contar total de productos
select_slug()	Función para obtener información de un producto usando su slug
primera_imagen()	Función para obtener la primera imagen de un producto

obtener_imagenes()	Función para obtener todas las imágenes de un producto
select_slug_taller()	Función para obtener información de un taller usando su slug
insertar_producto()	Función para agregar un producto a la base de datos
insertar_taller()	Función para agregar un taller a la base de datos
actualizar_producto()	Función para actualizar la información de un producto de la base de datos
actualizar_taller()	Función para actualizar la información de un taller de la base de datos

Tabla 4. Modelo Producto.

4.2.4 Modelo Carrito

Método	Descripción
select_info()	Función para obtener información del carrito de un usuario
select_items()	Función para obtener información de los productos
insert_item()	Función para agregar un producto al carrito
eliminar_item()	Función para eliminar un producto del carrito
contar_items	Función para contar productos distintos en un carrito
cambiar_cantidad_producto()	Función para cambiar cantidad de un producto en un carrito

Tabla 5. Modelo Carrito.

4.2.5 Modelo Orden

Método	Descripción
insert_orden()	Función para crear una nueva orden
insert_item()	Función para agregar un producto a la orden
orden_estado_fallida()	Función para modificar el estado de la orden a FALLIDA

eliminar_orden()	Función para eliminar una orden y sus productos, en caso de un error en el proceso de transacción

Tabla 6. Modelo Orden.

4.2.6 Modelo Transacción

Método	Descripción
nueva_transaccion	Función para crear una nueva transacción.

Tabla 7. Modelo Transacción.

4.3 Páginas

A continuación se muestran las vistas de las páginas y de los procesos.

4.3.1 Vista Inicio

Figura 4.2 Vista de inicio.

La página de inicio contiene un carrusel para atraer la atención de los usuarios. Después tiene contenedores para colocar información que de momento es información no verídica.

4.3.2 Vista Login

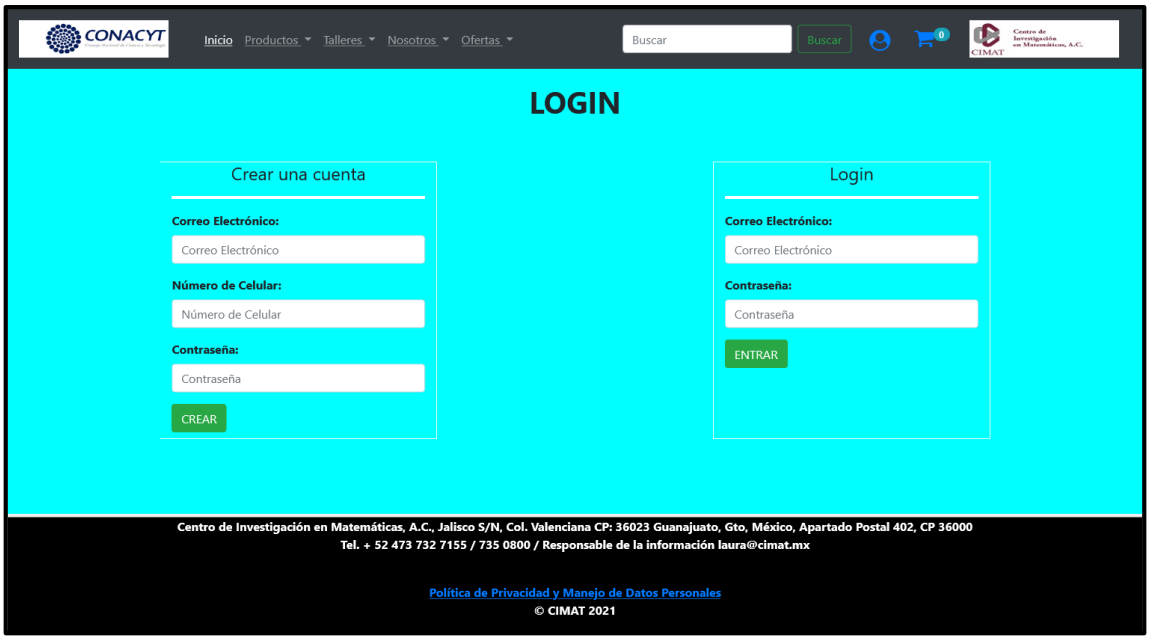

Figura 4.3 Vista de login.

La página de login contiene dos formularios: para crear una cuenta con sólo 3 datos y para hacer login. Los correos y teléfonos no pueden repetirse en la Base de Datos. Por el momento la contraseña no es encriptada. Se puede acceder con el ícono de perfil del navbar.

4.3.3 Vista Home

Figura 4.4 Vista de home para usuarios normales.

La página de home contiene un menú de opciones simples para revisar compras, modificar información y checar el carrito de usuario. Cuando se realiza el login, se asigna un nuevo carrito a esta sesión. Si eres administrador, tendrás dos

opciones más: para registrar un producto nuevo y para modificar la información de un producto. También contiene un botón para hacer logout.

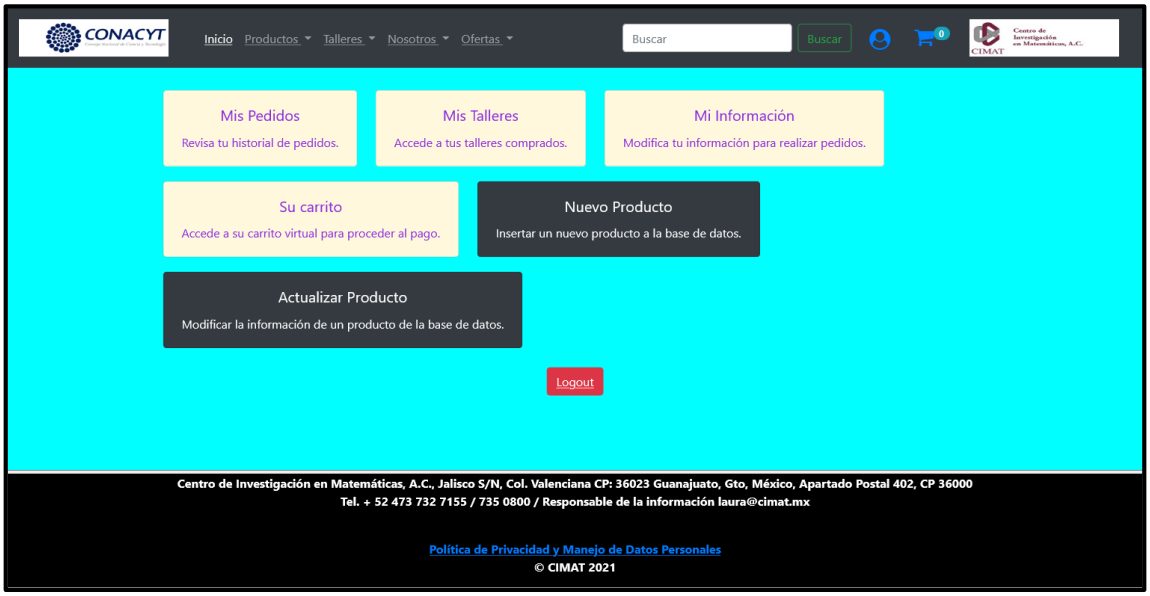

Figura 4.5 Vista de home para usuarios administradores.

4.3.4 Vista Actualizar información de usuario

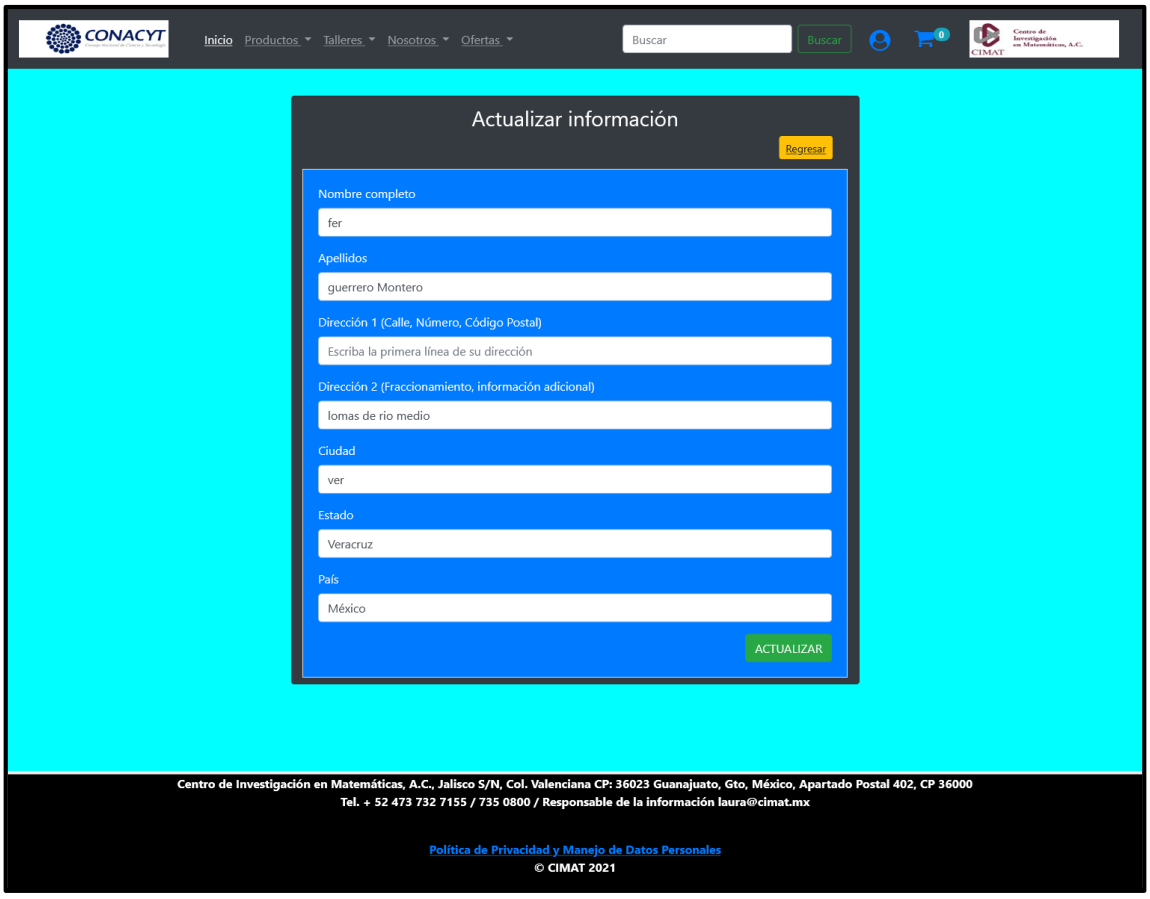

Figura 4.6 Vista de actualizar la información de usuario.

La página para actualizar la información de un usuario contiene un formulario que muestra la información actual y permite modificarla. Solo se maneja la información de contacto, para realizar pedidos (una nueva cuenta tiene todos los

campos vacíos y debe registrar su información antes de poder realizar un pedido). No sé puede cambiar aún campos como correo, teléfono y contraseña.

4.3.5 Vista Insertar nuevo producto

Figura 4.7 Vista para agregar un nuevo producto.

La página para insertar un nuevo producto es un formulario parecido al anterior. La opción solo está disponible para usuarios administradores. También permite agregar talleres si se marca la opción.

4.3.6 Vista Actualizar información de un producto

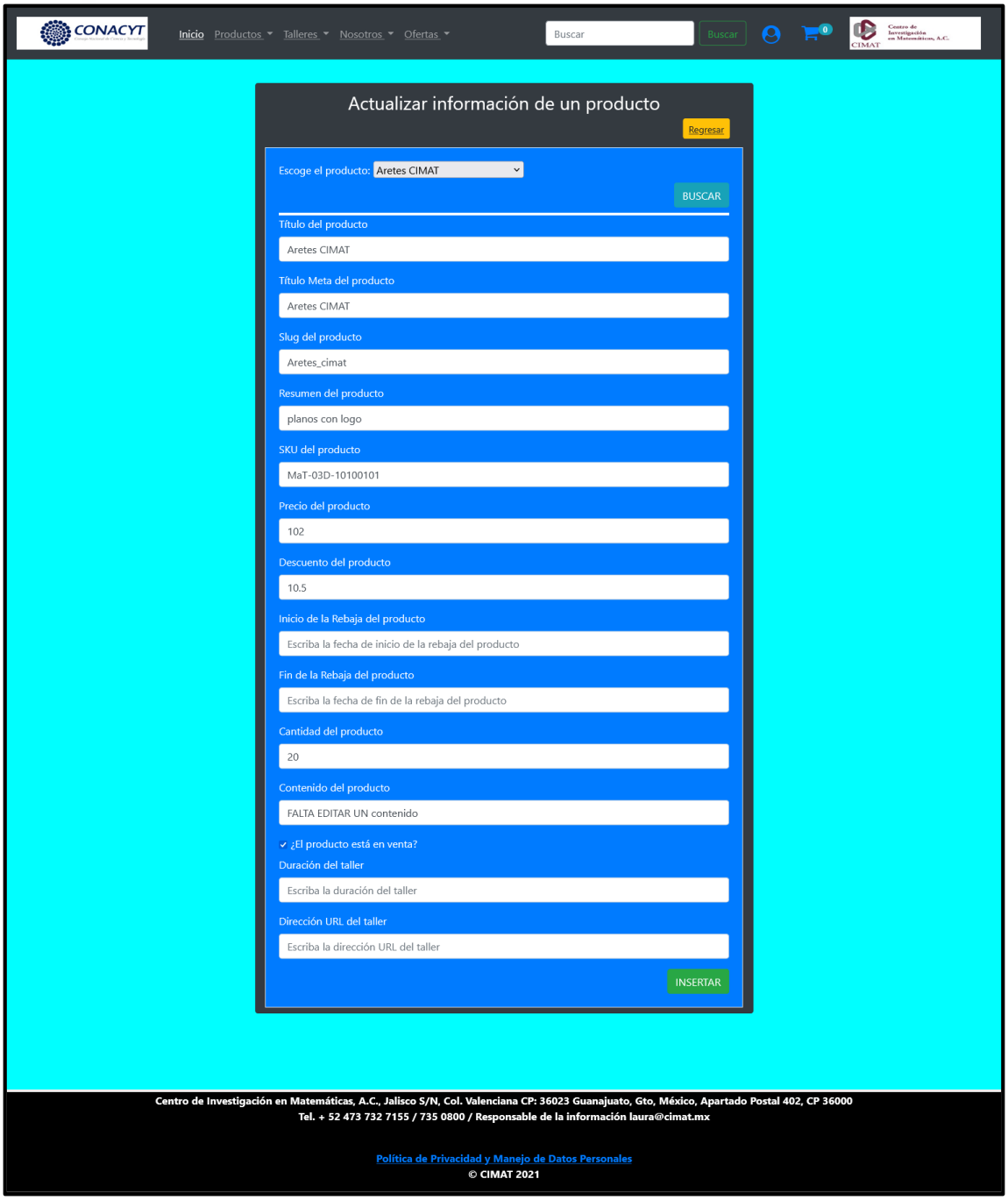

Figura 4.8 Vista para actualizar información de un producto.

La página para actualizar la información de un producto es nuevamente un formulario muy similar al anterior. Puedes seleccionar el producto a modificar desde una lista y entonces su información se desplegará en los campos. Permite modificar productos y talleres.

4.3.7 Vista Pedidos de usuario

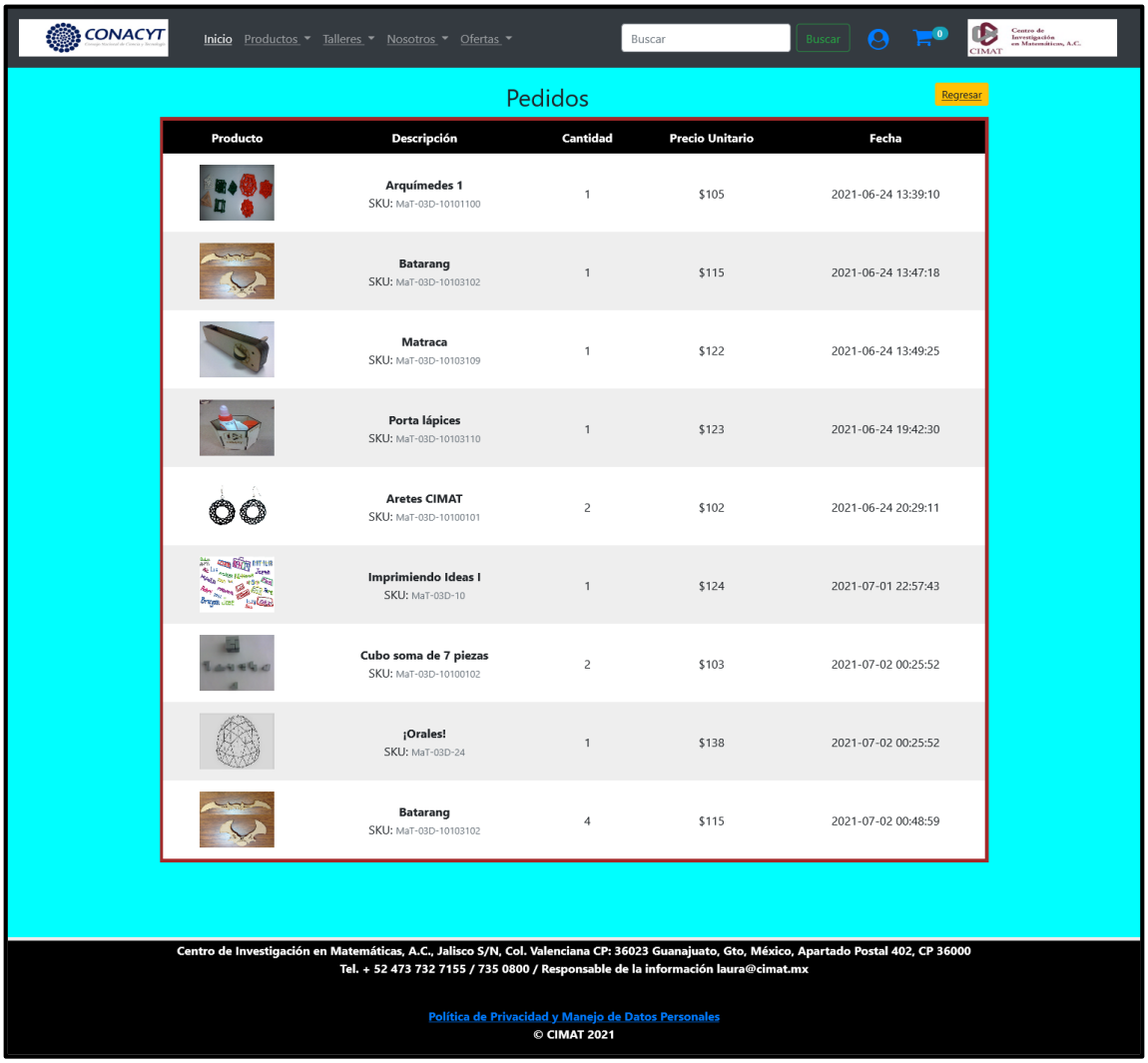

Figura 4.9 Vista de pedidos de un usuario.

La página de los pedidos de un usuario muestra en una tabla los productos y talleres comprados por un usuario. Solo muestra información muy básica. Las imágenes y los nombres de los productos funcionan como enlaces hacia las páginas de venta de productos y talleres.

4.3.8 Vista Talleres de usuario

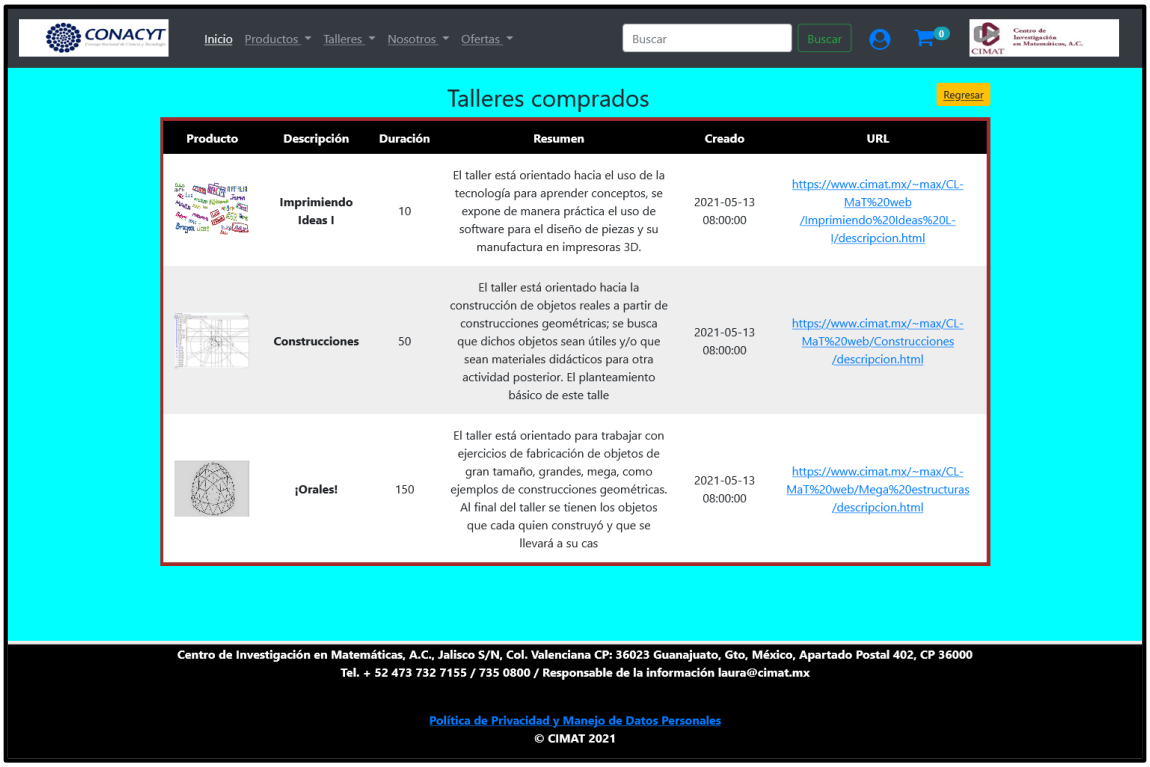

Figura 4.10 Vista de talleres de un usuario.

En la página de los talleres de usuario, se puede revisar los talleres comprados en una tabla similar a la anterior. En este caso se podría acceder a los talleres, y sus actividades.

4.3.9 Vista Galería Productos

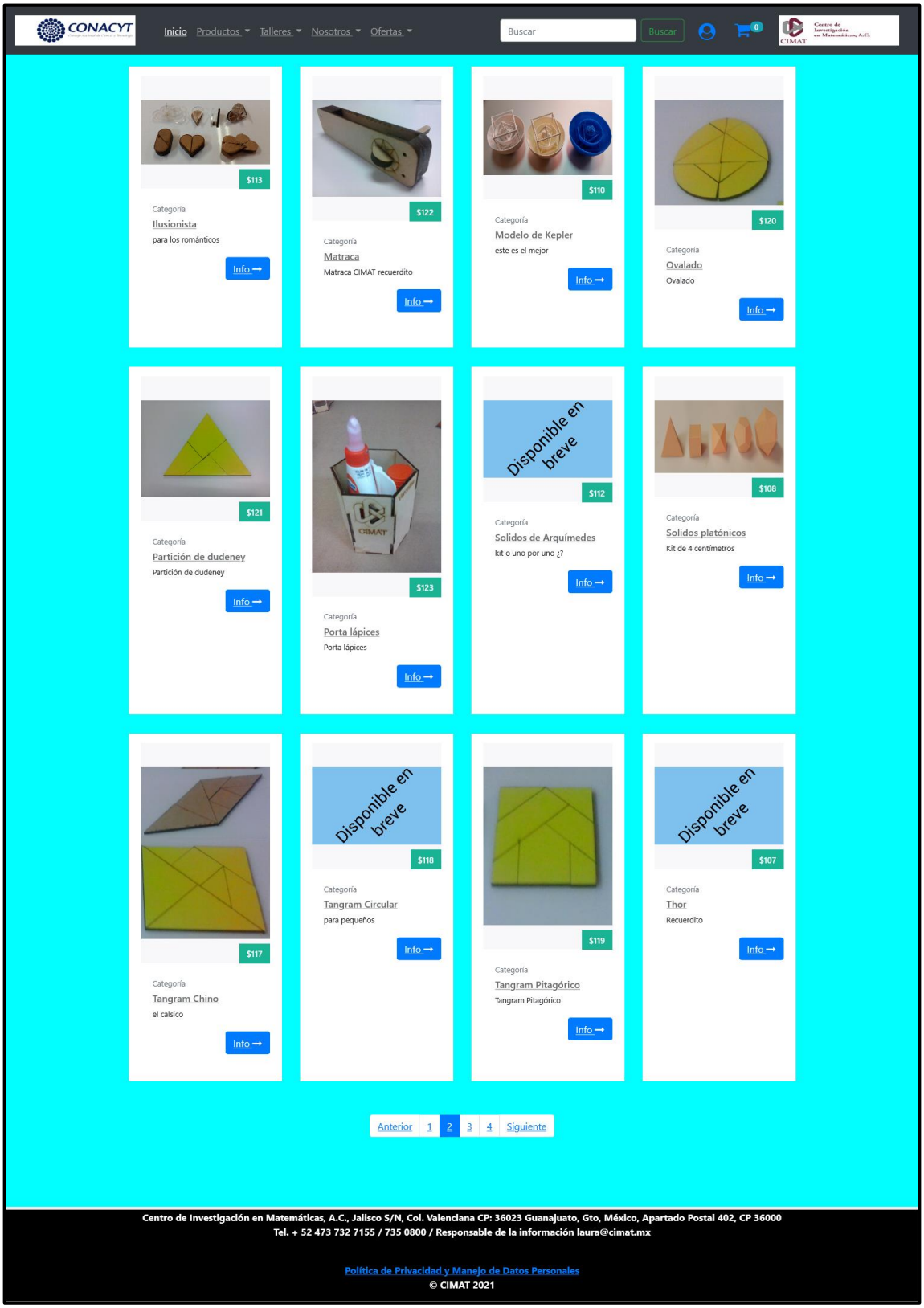

Figura 4.11 Vista de la página 2 de la galería de productos.

Esta página es la “tienda”, dónde podrás buscar productos y talleres. Por el momento se regresa todos los productos, no hay funcionalidad para filtrar o búsquedas específicas. El límite por página son 12 productos. Cada producto te

llevará a su propia página de venta usando su *slug* (ya sea con fichero *.htaccess* o método GET).

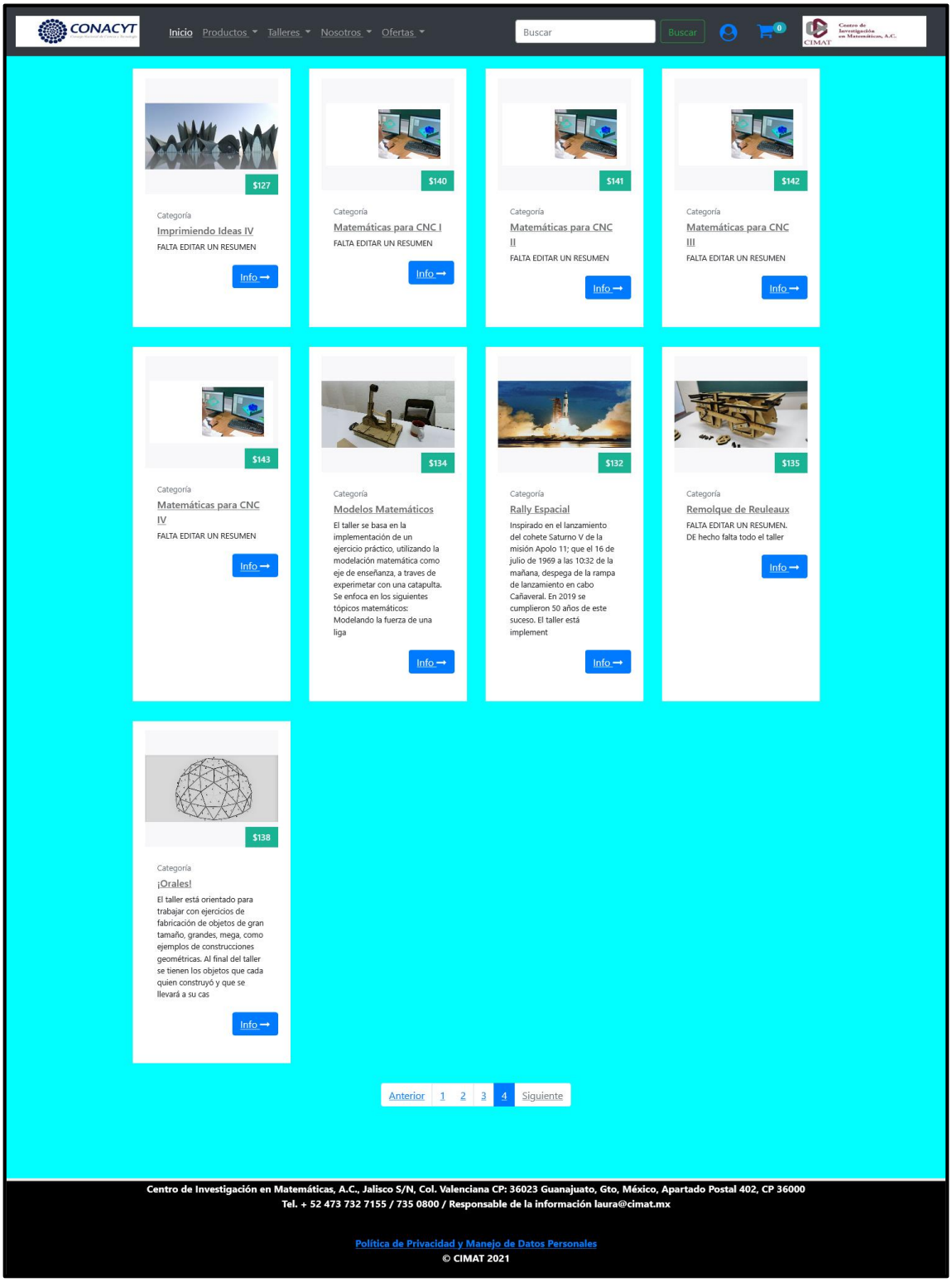

Figura 4.12 Vista de la página 4 de la galería de productos.

4.3.10 Vista de Producto en venta

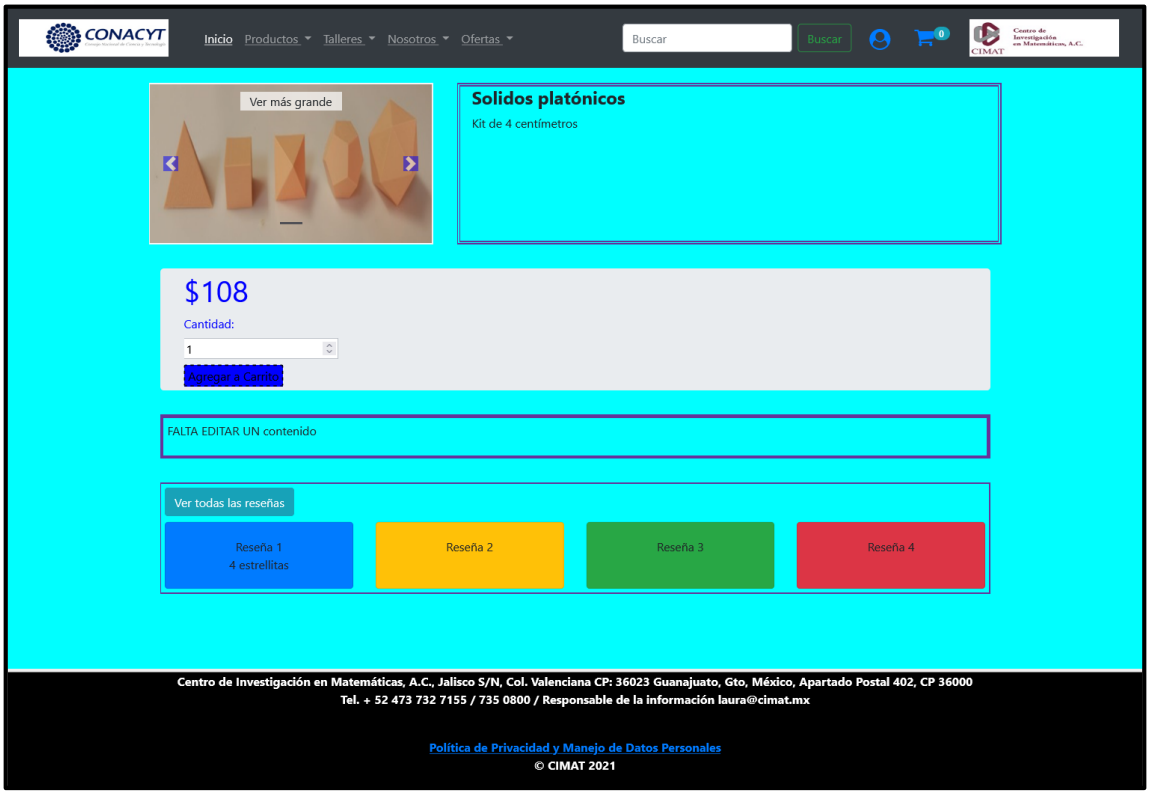

Figura 4.13 Vista de página de venta de un producto.

La página especial para cada producto. Se puede agregar el producto al carrito (mientras te hayas logeado antes). No hay funcionalidad para reseñas del producto.

4.3.11 Vista de Taller en venta

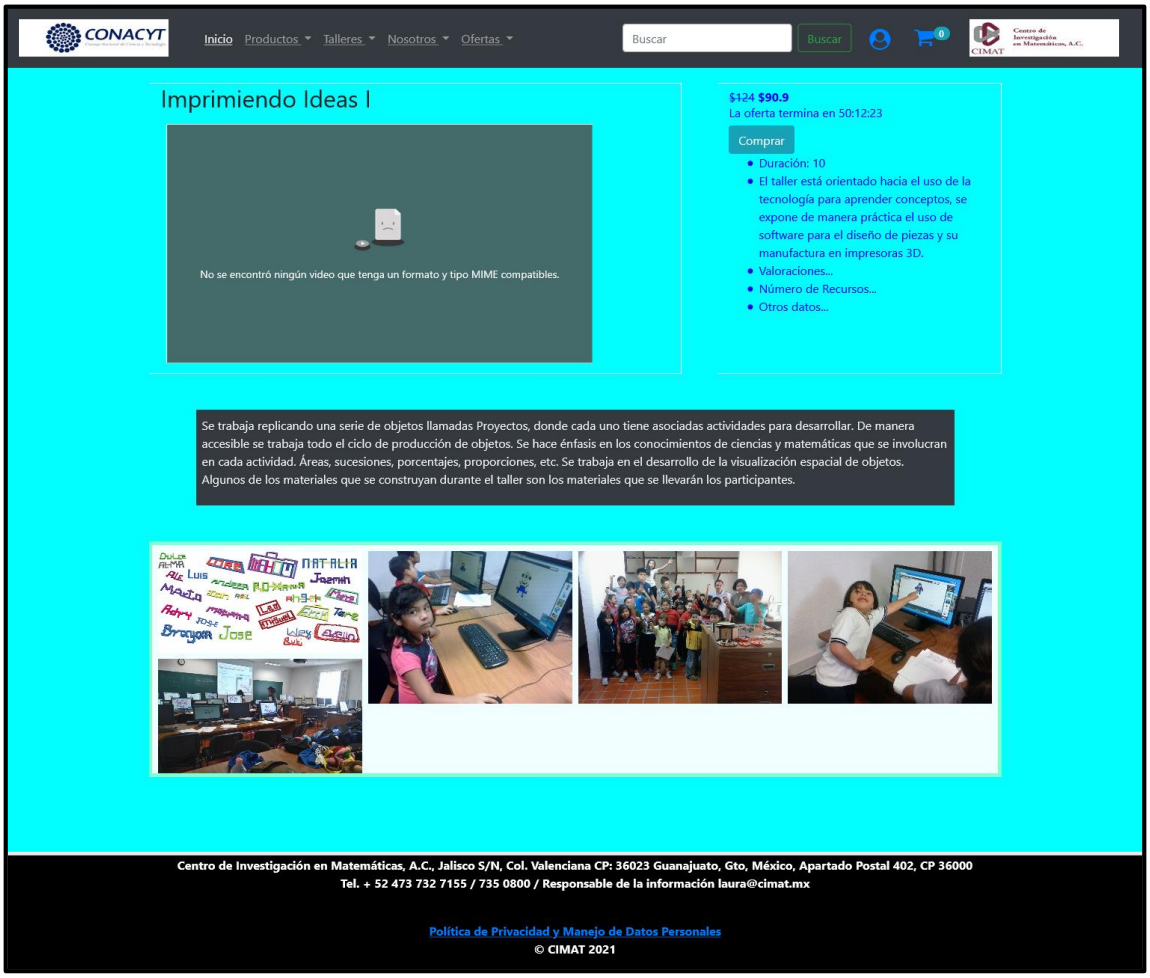

Figura 4.14 Vista de página de venta de un taller.

Parecido a la página de producto en venta, pero ésta es para talleres. Se pone en primer plano un vídeo del taller, y abajo coloca una galería de imágenes.

4.3.12 Vista de Carrito

Figura 4.15 Vista de carrito de un usuario.

La página del carrito se puede acceder desde el menú de home, o desde el icono del carrito en el navbar. Muestra los productos en una tabla, y sus precios. Las imágenes y los nombres funcionan como enlaces a las páginas de venta. Se puede aumentar o disminuir la cantidad de los productos desde esta página. También se pueden eliminar del carrito. Cuando haya productos en el carrito, el botón para realizar el pedido aparecerá.

4.3.13 Vista de Pedido

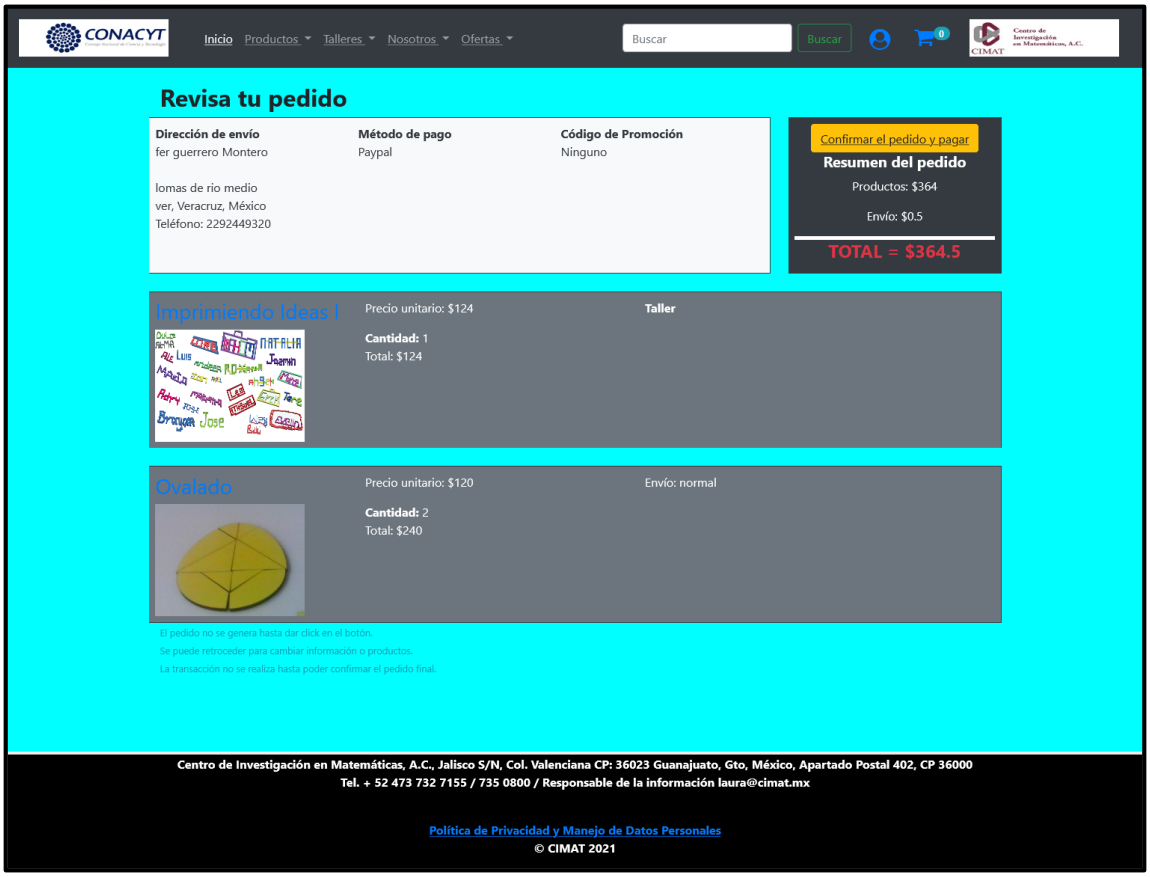

Figura 4.16 Vista de pedido

A esta página solo se puede acceder si la información de usuario está completa y si hay productos en el carrito. Muestra la lista de los productos a comprar, como también información incompleta. Contiene el botón para confirmar la orden y realizar la transacción. La orden y la transacción no se crean hasta que se le de clic a ese botón. Si no se puede completar el proceso (crear orden, agregar productos a la orden, agregar talleres al usuario si es necesario y realizar la transacción) se borra la orden y la transacción. Se maneja información incompleta y no verídica por el momento. Se puede completar una orden y transacción pero es falsa.

4.4 Base de datos

4.4.1 Resumen de la Base de Datos

Base de Datos	Cantidad
Tablas	15
Funciones	8
Procedimientos Almacenados	64
Procedimientos de Usuario	17

Procedimientos de Producto	30
Procedimientos de Carrito	6
Procedimientos de Orden	10
Procedimientos de Transacción	1

Tabla 8. Resumen de la Base de Datos.

No todos los procedimientos se han implementado en los modelos, ya que aún no se llegaba a esa parte de la implementación.

4.4.2 Modelo Relacional

Usando el modo Diseñador de *phpMyAdmin* se puede mostrar este Modelo Relacional:

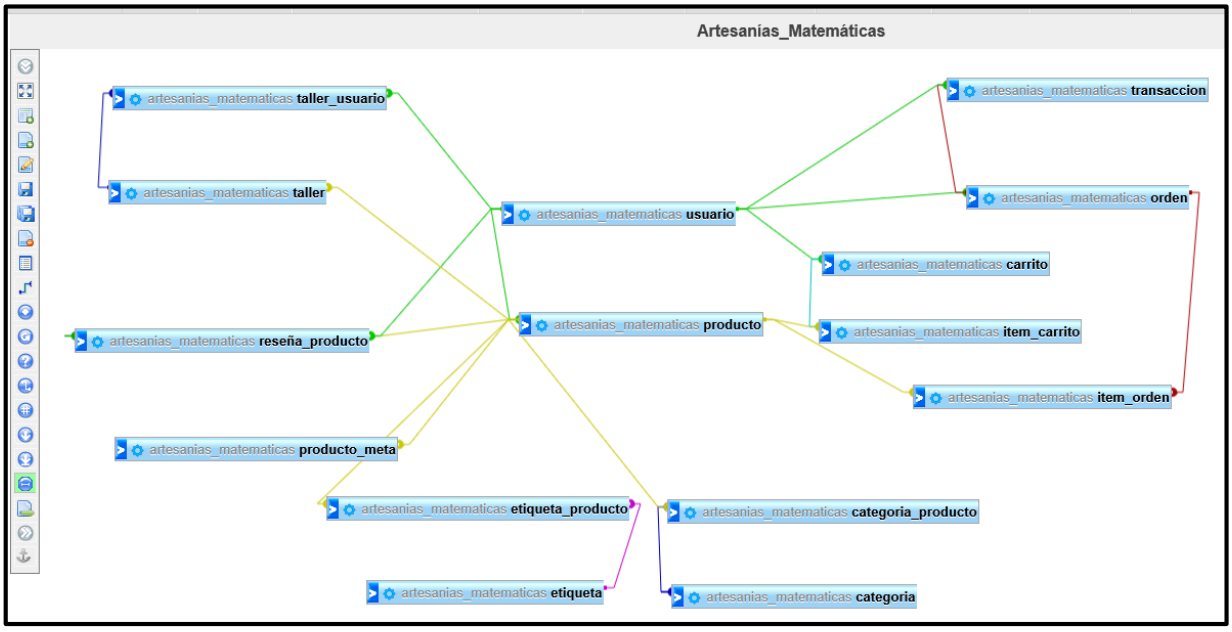

Figura 4.17 Modelo Relacional de la Base de Datos.

CAPÍTULO 5.
CONCLUSIONES

El proyecto realizado era un objetivo de la empresa muy especial, ya que se trata de vender sus propios productos y talleres para fomentar las matemáticas, que es uno de los objetivos del CIMAT. Como los resultados lo indican, se realizó un prototipo básico de la tienda en línea, en la cual ya se puede realizar (con información temporal de prueba) el proceso primordial de una tienda: comprar productos.

A pesar de que el proceso se puede completar, la aplicación no está absenta de fallos y carece de opciones. Con el tiempo dado, se enfocó en desarrollar una buena Base de Datos, usar un diseño muy simple y terminar el proceso de compra en específico. Pero las bases ya están asentadas para las siguientes fases de desarrollo de la aplicación Web para quien le corresponda.

Con las competencias adquiridas durante el desarrollo del proyecto, me siento más capacitado y con mayor conocimiento en los temas de base de datos y programación web con PHP. El ser autodidacta y saber encontrar soluciones en Internet es realmente una competencia importante de tener.

Realizar mis Residencias Profesionales es la primera vez que tuve un trabajo, y sobre todo en la industria de mi carrera. Desarrollar el proyecto fue una experiencia enriquecedora personal y profesionalmente. Es un lindo sentimiento darse cuenta que todo lo aprendido durante tus estudios se puede aplicar en el mundo laboral, y siempre es bueno obtener nuevas experiencias. Estoy satisfecho con mi trabajo realizado.

CAPÍTULO 6. COMPETENCIAS DESARROLLADAS

Durante mi estancia de Residencias Profesionales y el desarrollo del proyecto logré reforzar y obtener nuevas habilidades en el ámbito personal y profesional. Las competencias desarrolladas más importantes fueron las siguientes:

- Aprendizaje autodidacta.
- Habilidades de investigación.
- Análisis de problemas e implementación de soluciones.
- Trabajo bajo una guía.
- Mejora de planificación y organización.
- Diseño de base de datos.
- Procedimientos almacenados de base de datos.
- Uso de CSS y Bootstrap para diseños de páginas web.
- Aplicar scripts JavaScript para programación de diseños de páginas web.
- Uso extensivo de PHP para la operatividad de páginas web.
- Conexión a la base de datos con PHP.
- Clases y objetos en el lenguaje PHP.
- Utilizar métodos HTTP para manejo de información en páginas web.
- Aplicaciones de ficheros *.htaccess* para redireccionar páginas web.

CAPÍTULO 7.
FUENTES DE INFORMACIÓN

Desconocido. (N/A). *Elementos de una tienda online para vender en Internet*. Febrero 20, 2021, de IngenioVirtual Sitio web: https://www.ingeniovirtual.com/elementos-de-una-tienda-online/

Gustavo B. (2021). *Las 15 mejores plataformas de venta online para crear tu tienda online en 2021*. Febrero 27, 2021, de Hostinger Sitio web: https://www.hostinger.mx/tutoriales/mejores-plataformas-ecommerce/

Garcia, J. (2020). *8 soluciones para crear una tienda online gratis*. Febrero 20, 2021, de WebsiteToolTester Sitio web: https://www.websitetooltester.com/es/blog/crear-tienda-online-gratis/

Barakat, R. (2020). *12 buenas prácticas para tu e-commerce*. Febrero 22, 2021, de ids.agency Sitio web: https://www.ids.agency/es/blog/12-buenas-pr%C3%A1cticas-para-tu-e-commerce

Bolina, L. (2018). *Mejores prácticas de contenidos para ecommerce*. Febrero 22, 2021, de rockcontent Sitio web: https://rockcontent.com/es/blog/mejores-practicas-de-contenidos-para-ecommerce/

Microsoft (2021). *Visual Studio Code*. Recuperado de: https://code.visualstudio.com/docs

Mozilla (2021). *HTML*. Recuperado de: https://developer.mozilla.org/es/docs/Web/HTML

Mozilla (2021). *CSS*. Recuperado de: https://developer.mozilla.org/es/docs/Web/CSS

Bootstrap team (2021). *Bootstrap*. Recuperado de: https://getbootstrap.com/

Mozilla (2021). *HTTP*. Recuperado de: https://developer.mozilla.org/es/docs/Web/HTTP

Mozilla (2021). *Métodos HTTP*. Recuperado de: https://developer.mozilla.org/es/docs/Web/HTTP/Methods

Mozilla (2021) *JavaScript*. Recuperado de: https://developer.mozilla.org/es/docs/Web/JavaScript

Ubuntu (2021). *Apache*. Recuperado de: https://ubuntu.com/server/docs/web-servers-apache

PHP Group (2021). *PHP*. Recuperado de: https://www.php.net/manual/es/preface.php

MariaDB Foundation (2021). *MariaDB*. Recuperado de: https://mariadb.org/

phpMyAdmin contribuitors (2021). *phpMyAdmin*. Recuperado de: https://www.phpmyadmin.net/

Apache Friends (2021). *XAMPP*. Recuperado de: https://www.apachefriends.org/es/index.html

The Apache Software Foundation (2021). *Ficheros .htaccess*. Recuperado de: https://httpd.apache.org/docs/current/howto/htaccess.html

CAPÍTULO 8.
ANEXOS

ESTADO DEL ARTE

Obtención de Características

Como se menciona en el anteproyecto, las características generales de una tienda en línea son:

1. Apariencia y diseño: Los colores predominantes, las formas y los elementos gráficos deberían hacer distinguibles los valores de la empresa o marca para diferenciarla del resto dentro de su sector.
2. Catálogo de productos: Habrá que tener en cuenta varios factores: Imágenes o fotografías de producto, Productos disponibles o en venta, Taxonomías según las características de los productos, Descripción o ficha del producto.
3. Sistema interno de búsqueda de productos: La organización y gestión de los productos gracias a una base de datos hace que el usuario pueda encontrarlos de forma dinámica a través de campos de búsqueda con menor esfuerzo. Para aumentar la eficacia del sistema, es importante que esta estructura refleje también diferentes características y atributos con los que el usuario pueda estar familiarizado.
4. Sistema de recomendaciones: Entre los elementos de una tienda online, una práctica muy recomendable es implementar un sistema que muestre otros productos cuyas características puedan estar relacionadas o sirvan para complementar a aquellos que resultan más interesantes para el usuario.
5. Carrito de compra: El carrito de la compra debe estar bien localizado, aportando información clara sobre la cantidad de productos que se incluyen en el proceso. Es uno de los elementos funcionales más relevantes durante el proceso de adquisición de productos, ya que debe mostrar datos asociados también a los impuestos, gastos de envío, posibles descuentos, y la actualización de todos ellos en su conjunto.
6. Proceso de registro: Para llegar a comprar lo normal es registrarse antes, y el proceso de registro suele ser uno de los procedimientos a los que el usuario se muestra más reacio, pero a la vez imprescindible para realizar un pedido. La solicitud de los datos y sus diferentes fases con campos para rellenarlos, contribuyen a que el usuario se desanime en muchos casos y no llegue a completar el proceso.
7. Métodos de pago y pasarelas de pago: Las alternativas más habituales van desde el pago contra reembolso, la opción de pagar a través de Paypal, o realizando transferencias por medio de pasarelas de pago bancarias, entre las que hay a la vez variedad de opciones a elegir según las condiciones impuestas por cada entidad.
8. Certificado de seguridad: la seguridad debe ser instaurada para evitar que terceros puedan acceder a esta información. Los certificados SSL (Secure Sockets Layer) añaden una capa de seguridad para evitar en la medida de lo posible situaciones comprometedoras. Lo hacen utilizando una transcripción cifrada de los datos, impidiendo así que metodologías de hacking puedan interpretarlos. Disponer de uno de estos certificados es otro plus gracias al cual el visitante de la página depositará más confianza para decidirse a realizar la compra.
9. Proceso de venta y embudo de conversión: Existen variedad de herramientas analíticas, tanto gratuitas como con un coste directamente

relacionado con el plan y el volumen de negocio. Estas captan información desde que el usuario accede a la tienda hasta que adquiere el producto o disiente abandonando el proceso en algún punto.

10. Sistemas para la atención al cliente: Entre la variedad de sistemas para atender las dudas o incidencias derivadas de posibles errores en los procesos se encuentran los siguientes: Contacto por email, Contacto telefónico, Chat online en tiempo real, Generadores de tickets.
11. Gestión de stocks: La variedad y cantidad de productos disponibles debe estar gestionada de forma correcta para prevenir fallos en el proceso de realización y preparación de pedidos. La metodología depende en parte de las alternativas y soluciones tecnológicas para crear una tienda online que se deseen utilizar como base del negocio.
12. Integración de otros sistemas de gestión: La contabilidad, los proveedores, los sistemas de reparto o empresas de mensajería y otros aspectos deben estar controlados. El objetivo es automatizar los diferentes departamentos que necesitan ser gestionados para hacer más flexible y fluido el desempeño de la actividad.
13. Consideraciones: Alinear tareas como la negociación de precios con los proveedores, el stock y almacenaje, la venta y distribución de los productos y la contabilidad en este orden, exige un planteamiento muy meditado y conlleva una buena dosis de organización y disponibilidad, enfocando los esfuerzos siempre para mejorar.

Las características específicas de la tienda CL-MaT son:

- Venderá artículos didácticos de matemáticas.
- Cada producto tiene un pequeño texto-video-audio explicativo de que matemáticas entran en la fabricación del producto o en su uso.
- Sin anuncios.
- Muy simple.
- Con videos de los productos.
- Con actividades para los productos, la idea es que después que lo compran tienen acceso a actividades para utilizarlo mejor, como clases de utilización.
- También vender los talleres, que entran como producto.

Similares de productos educativos

Aprendiendo Matemáticas (https://aprendiendomatematicas.com/tienda/)

Pantalla de inicio:

Figura 8.1. Captura 1 de Aprendiendo Matemáticas.

Pantalla de productos:

Figura 8.2. Captura 2 de Aprendiendo Matemáticas.

Pantalla de carrito:

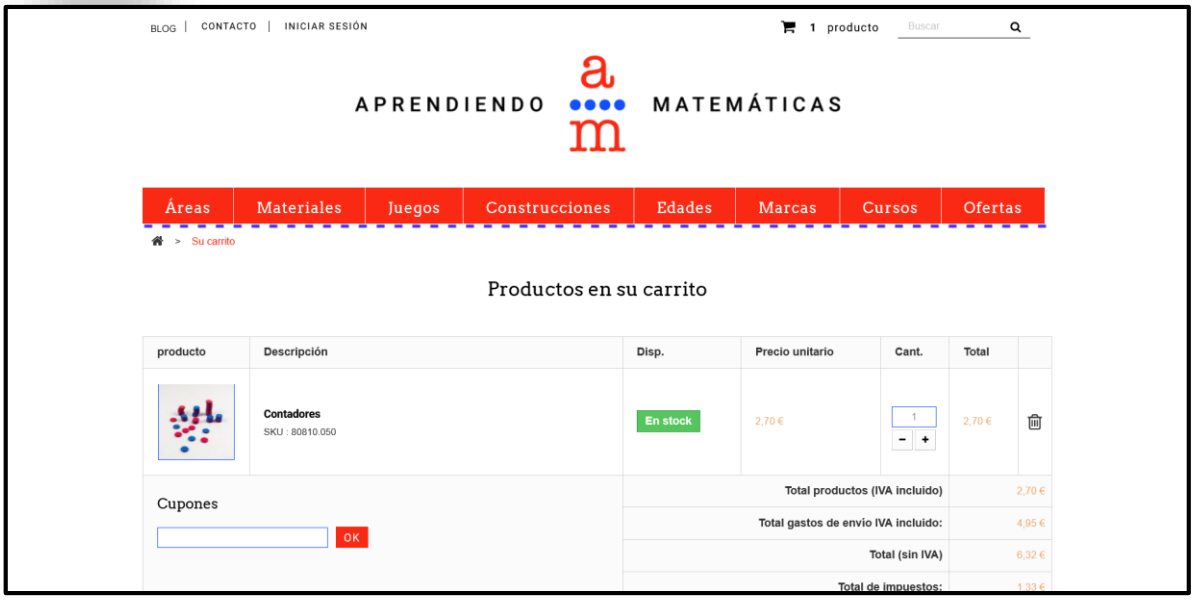

Figura 8.3. Captura 3 de Aprendiendo Matemáticas.

ideashands (https://www.ideashands.com.mx/)

Pantalla de inicio:

Figura 8.4. Captura 1 de ideashands.

Pantalla de productos:

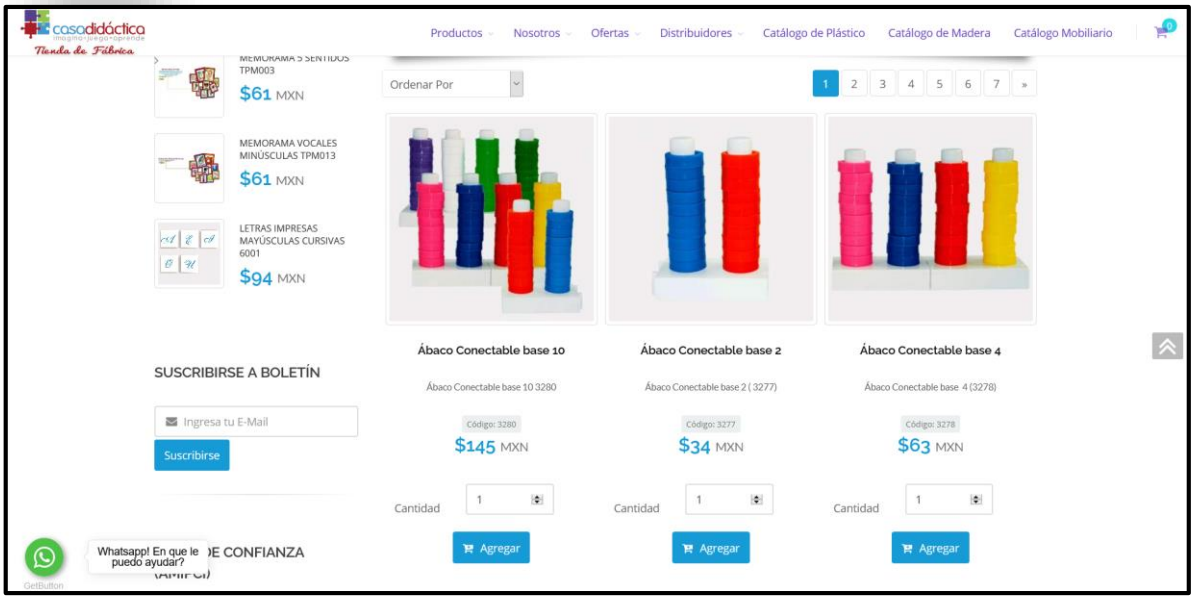

Figura 8.5. Captura 2 de ideashands.

Pantalla de carrito:

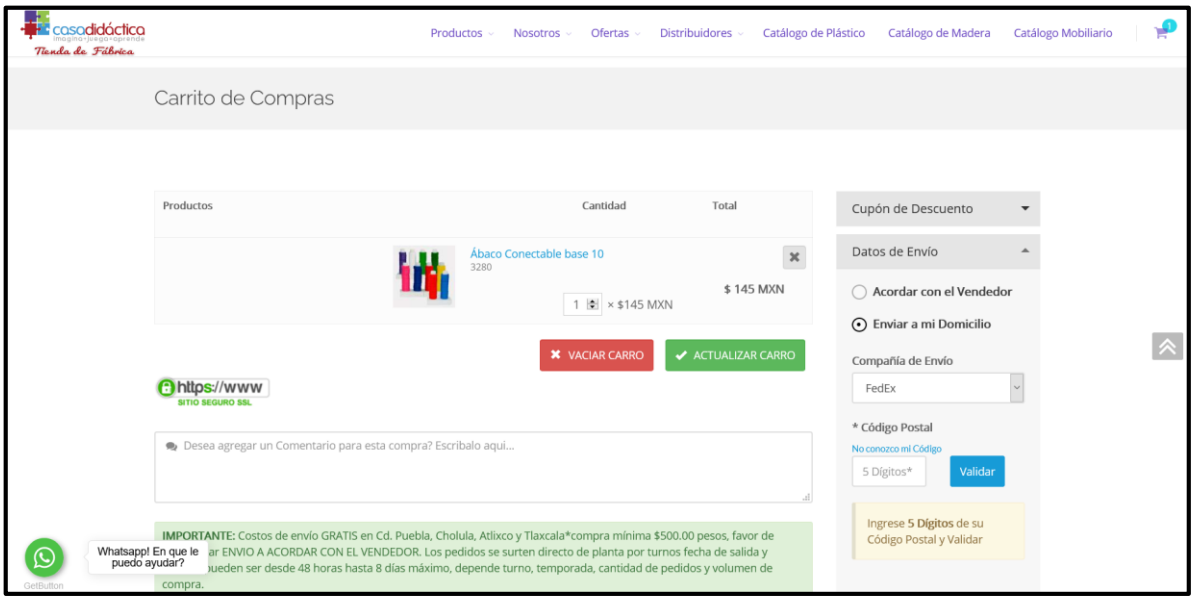

Figura 8.6. Captura 3 de ideashands.

Grupo Educar (https://www.grupoeducar.com.mx/)

Pantalla de inicio:

Figura 8.7. Captura 1 de Grupo Educar.

Pantalla de productos:

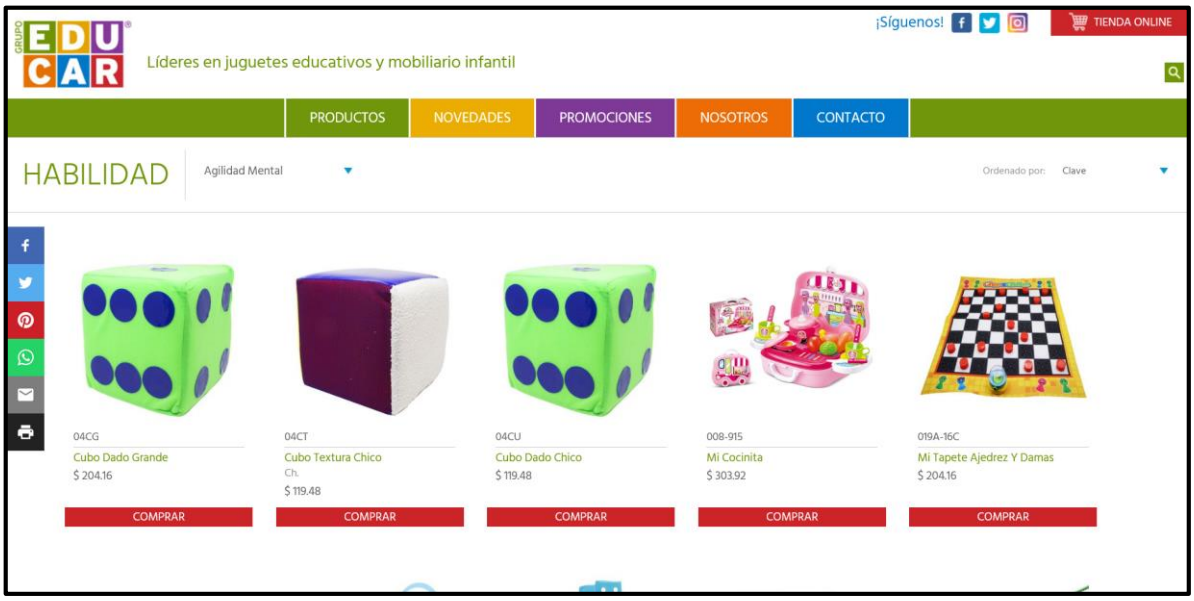

Figura 8.8. Captura 2 de Grupo Educar.

Pantalla de “Nosotros”:

Figura 8.9. Captura 3 de Grupo Educar.

National Museum of Mathematics (https://shop.momath.org/)

Pantalla de inicio 1:

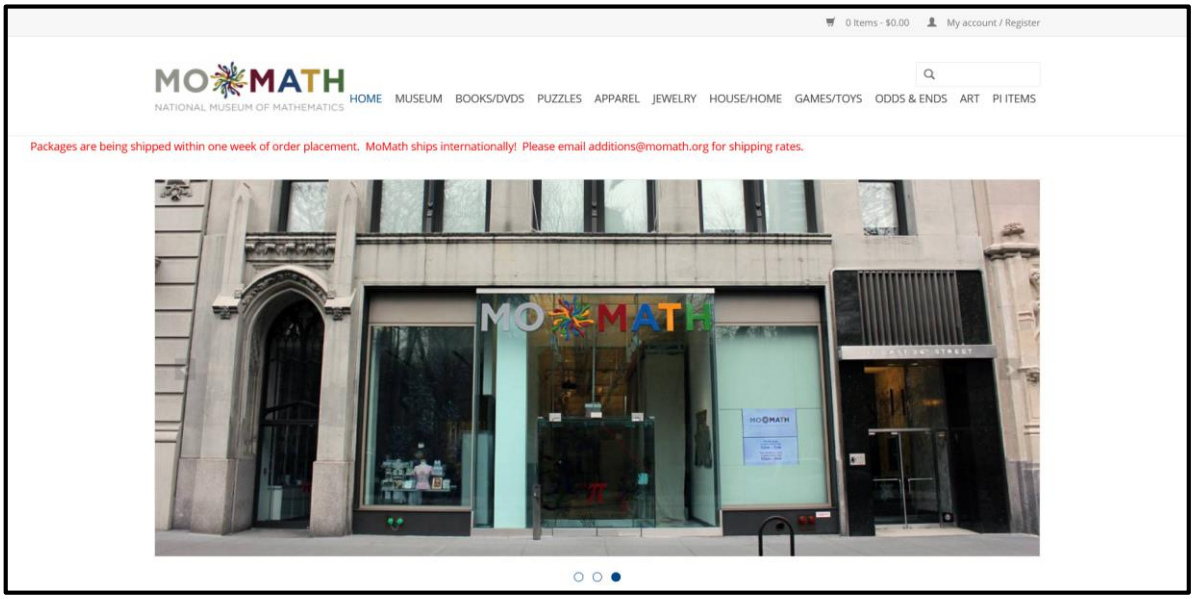

Figura 8.10. Captura 1 de National Museum of Mathematics.

Pantalla de inicio 2:

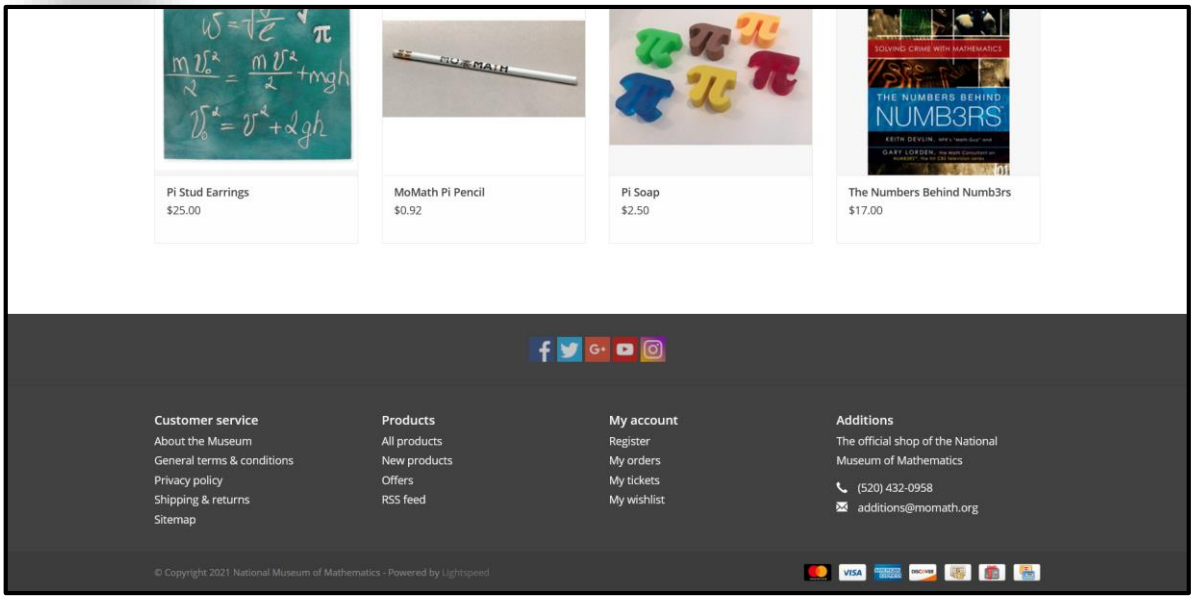

Figura 8.11. Captura 2 de National Museum of Mathematics.

Pantalla de productos:

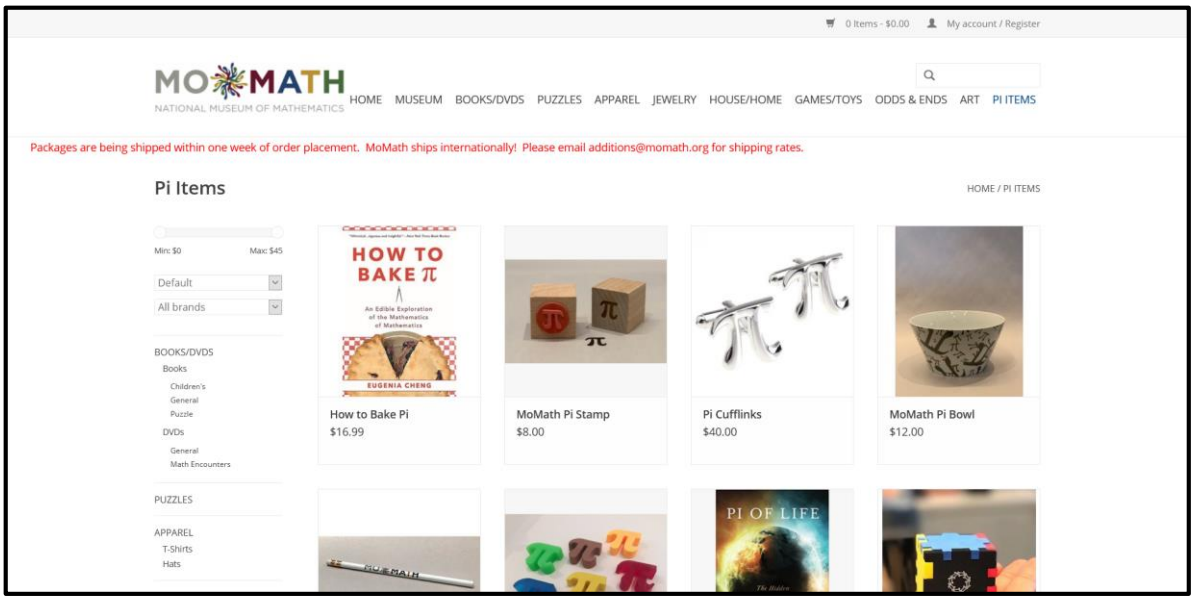

Figura 8.12. Captura 3 de National Museum of Mathematics.

Montessori Para Todos (https://montessoriparatodos.es/)

Pantalla de inicio:

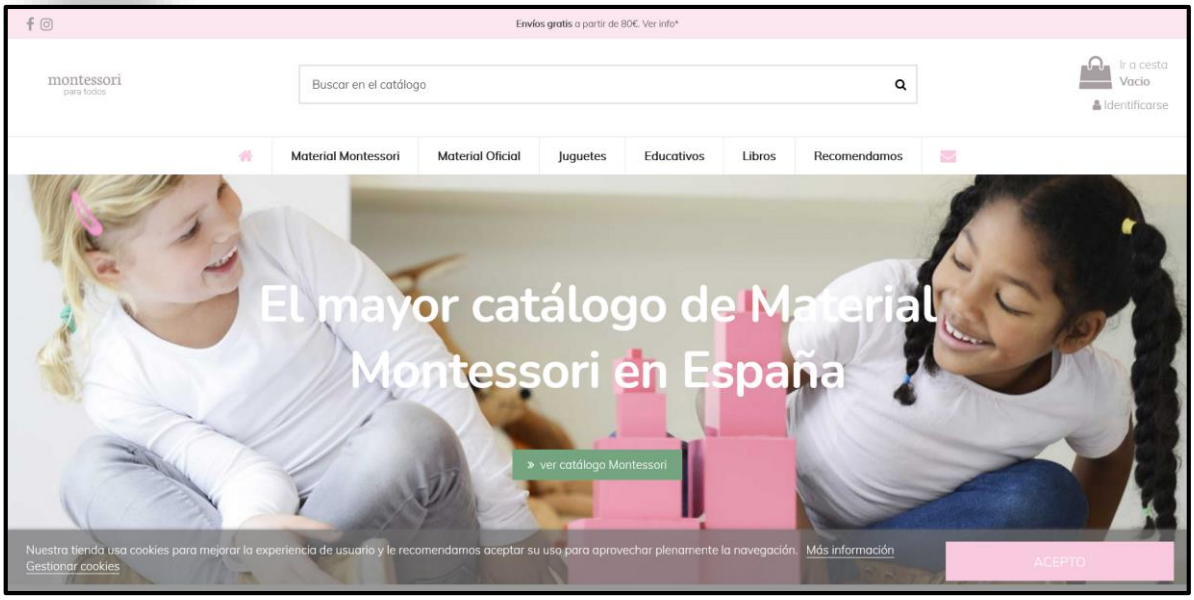

Figura 8.13. Captura 1 de Montessori Para Todos.

Pantalla de productos:

Figura 8.14. Captura 2 de Montessori Para Todos.

Spectrum (https://spectrum-nasco.ca/en/)

Pantalla de inicio:

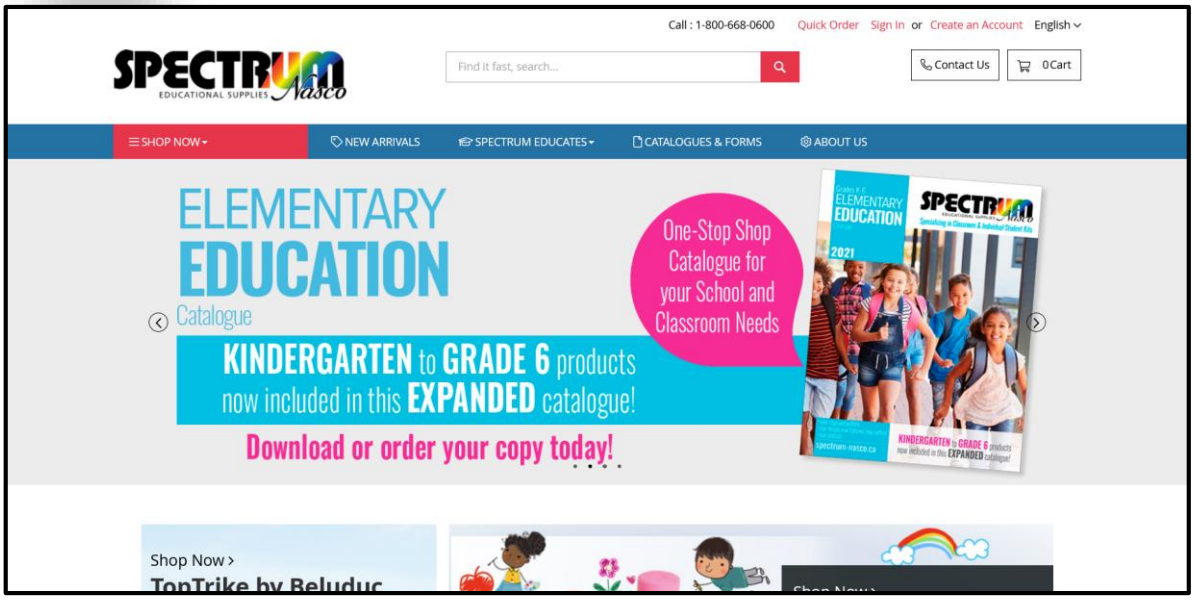

Figura 8.15. Captura 1 de Spectrum.

Pantalla de productos:

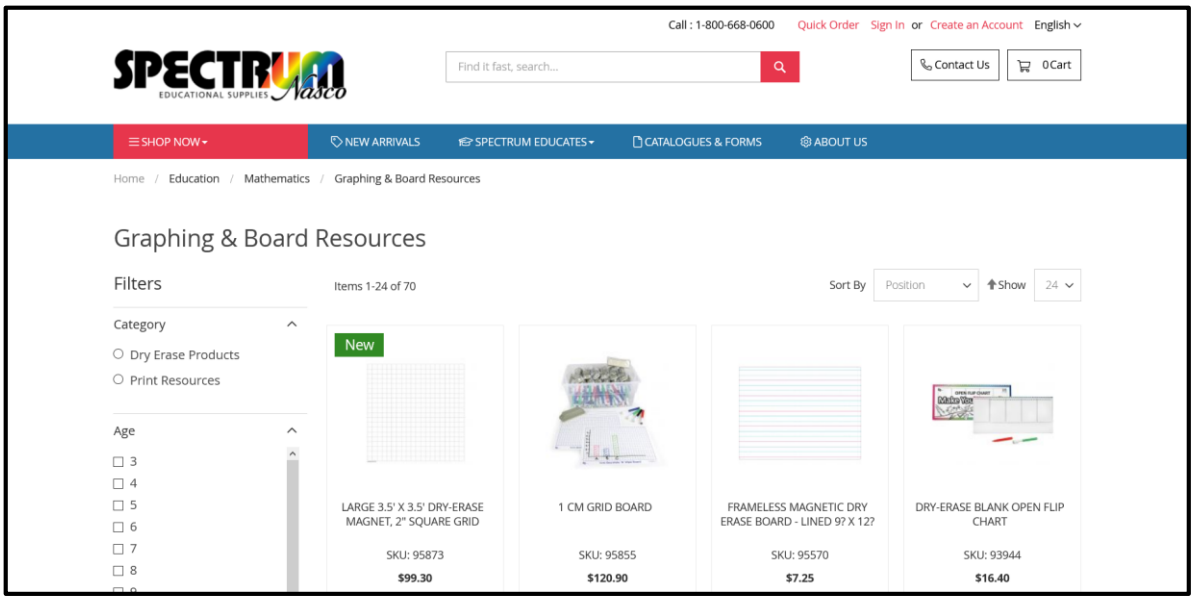

Figura 8.16. Captura 2 de Spectrum.

Pantalla de login:

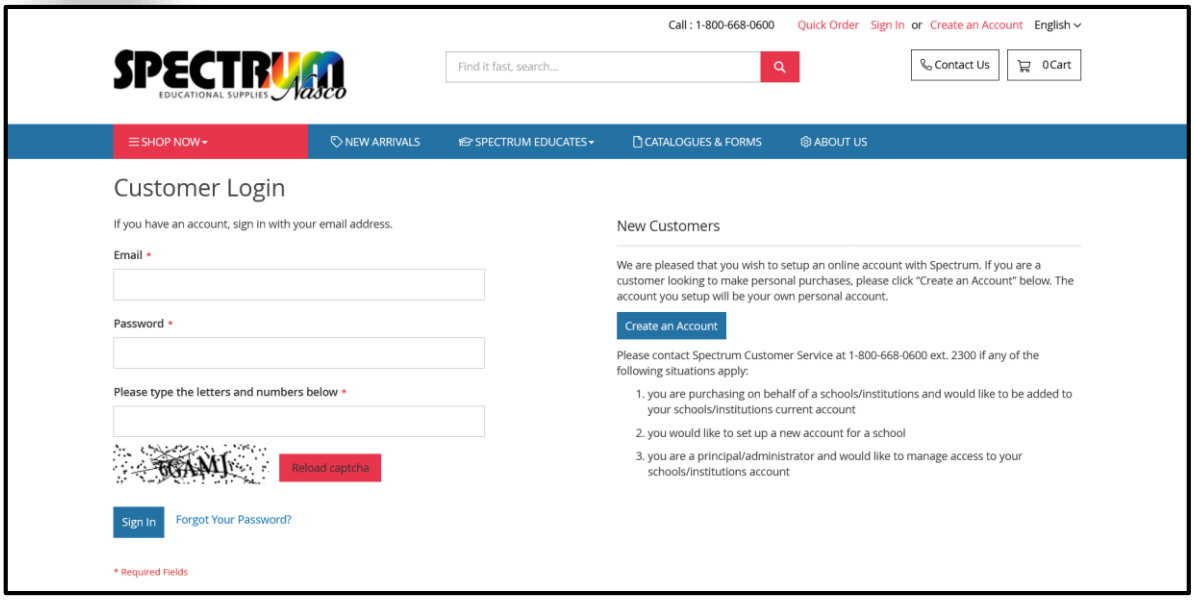

Figura 8.17. Captura 3 de Spectrum.

Software para implementación de tiendas en línea

Tenemos tres opciones:

1. Empezar desde cero.

Usualmente para aplicación web en CIMAT se usa:

- Servidor web
  - centos-release-7.2-2.2004.0.1.el8.x86_64
  - PHP Versión 7.4.7
  - MariaDB -10.3.17
  - Server versión Apache: 2.4.37 (centos)
- Para el cliente
  - Bootstrap v5.0.0-beta2
- Otros
  - phpMyAdmin
  - FileZilla
  - ProjectLibre
  - Visual Studio Code

2. Utilizar un Software libre.

Plataforma	Ventajas	Desventajas
Mozello	500 MB de almacenamiento Varios idiomas Control de inventario Variaciones y opciones para los productos	Sistema básico Sin SSL No productos digitales No pagos con tarjeta 10 productos
WebStarts	1 GB de espacio Vender de todo Hasta 20 ventas al día	Sin referencias de producto Sin opciones SEO Sólo en inglés

	Anuncio poco intrusivo Buenas opciones para pagos	10 productos
Weebly	Productos ilimitados SSL incluido (https) Fácil de usar Blog excelente Escalabilidad	Sin pagos offline
Freewebstore	SSL gratis Numerosos procesadores de pagos Sin comisiones por transacción Funcionalidades avanzadas	Backend complejo y anticuado Solo admite productos físicos Sólo en inglés 20 productos
Strikingly	Buenas plantillas Numerosos procesadores de pago	Sólo un producto Plantillas limitadas Limitaciones de cara al SEO
MyOnlineStore	Buenas plantillas Opciones de pago Sistema en varios idiomas	No incluye SSL Plantillas limitadas Todavía en fase beta No disponible en español
Ecwid	Potentes funcionalidades Numerosos procesadores de pago Intuitivo y fácil de usar Integración con SSL Elegante presentación	Plugin sobre una página web ya creada El SEO podría ser más limpio 10 productos
WooCommerce	Potentes funcionalidades Fantástico para el SEO Funciona bien para tiendas grandes Integración con SSL Elegante presentación	Muy técnico Necesitas usar WordPress No tiene soporte Buscar propio alojamiento web
PrestaShop	Numerosos procesadores de pago Configuración avanzada de administración de productos	Curva de aprendizaje Puede funcionar lento por momentos

OpenCart	Amena para principiantes e intuitiva Muchas opciones de pasarelas y funciones de SEO incorporadas Comunidad activa	No ofrece tantas funciones incorporadas Muchos temas lucen un poco anticuados Necesitas un proveedor para alojar la tienda y comprar dominio
AbanteCart	Permite agregar y manejar nuevos productos sin complicaciones Funcionalidad de SEO incorporada Admite plugins y temas	La comunidad es bastante pequeña Algunos de los temas están desactualizados Necesitas un plan de alojamiento y dominio
osCommerce	Configurar una tienda online es simple y rápido Cuenta con miles de extensiones gratuitas Enorme comunidad	Mala escalabilidad El tablero luce algo desactualizado Necesitas un plan de alojamiento y dominio
CubeCart	Plataforma fácil de usar con muchas funciones adicionales	Regular selección de extensiones Necesitas un plan de alojamiento y dominio
Joomla!	Múltiples extensiones gratuitas Control y flexibilidad total	Algunas extensiones son muy limitadas Carece de funcionalidades dedicadas Necesitas un plan de alojamiento y dominio

Tabla 9. Comparación de software gratis para implementación de tiendas en línea.

3. Comparar Softwares de costo.

Plataforma + Plan Básico	Ventajas	Desventajas
Zyro **€7.64/mes**	Perfecta para principiantes Fácilmente ampliable Múltiples opciones de pago	Carece de funciones avanzadas No hay soporte telefónico
Tiendanube **399 pesos argentinos + 2% transacción mensual**	No requiere conocimientos técnicos previos Diseño intuitivo Admite productos físicos y digitales	Mayoría de integraciones a través de aplicaciones que ofrece no son gratuitas

	Múltiples opciones de pago	
Magento **€7.45/mes en Hostinger**	Muy escalable Buen SEO y seguridad Admite múltiples monedas y tasas de impuestos Se integra con casi cualquier procesador de pago	Curva de aprendizaje empinada Gratis pero sin alojamiento web, dominio, seguridad adicional
Shopify **$29/mes**	Toneladas de herramientas adicionales Excelente soporte al cliente Herramienta de recuperación de Carrito abandonado	Pueden aplicarse comisiones por transacción adicionales Selección limitada de temas gratuitos
BigCommerce **$29.95/mes**	Característica multicanal avanzada Opciones interesantes para el SEO Número ilimitado de cuentas de personal Sin comisiones adicionales	Límite de ventas anuales Pequeña curva de aprendizaje
3dcart **$19/mes**	Admite más de 160 pasarelas de pago de terceros Excelente atención al cliente Cientos de características y herramientas incorporadas	Personalizar el diseño requiere conocimientos de HTML o CSS No hay soporte para aplicaciones móviles
Big Cartel **$9.99/mes**	Plan gratuito para 5 productos Enorme selección de temas gratuitas	Número limitado de productos que se pueden vender Carece de algunas funciones y herramientas
Volusion **$29/mes**	Características enfocadas para dropshipping Impresionantes herramientas de análisis Sistema transparente de inventario y marketing	No hay herramientas específicas para blogs Puede ser un poco lenta a veces

Tabla 10. Comparación de software de costo para implementación de tiendas en línea.

Estrategia de Desarrollo

Se decidió empezar desde cero utilizando la herramienta XAMPP 7.4.11 en Windows ya que contiene en conjunto las siguientes herramientas para desarrollar el proyecto web:

1) Apache 2.4.46
2) PHP 7.4.11
3) MariaDB 10.4.14
4) phpMyAdmin 5.0.3

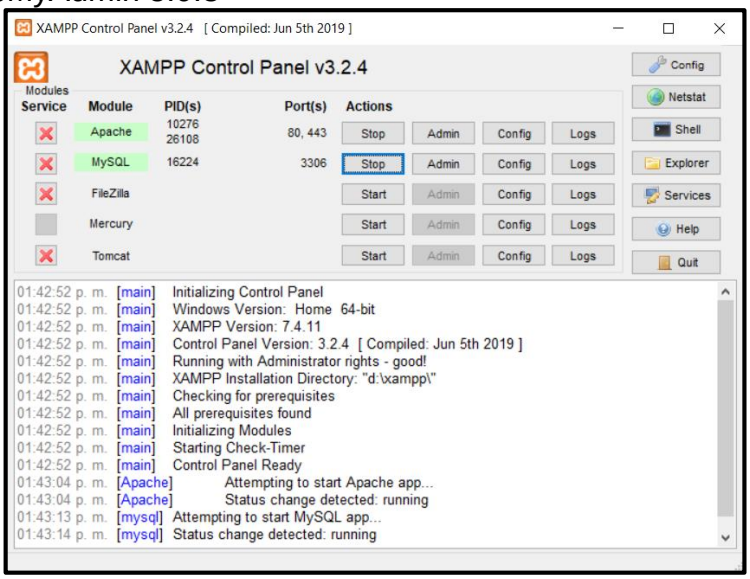

Figura 8.18. Panel de Control de XAMPP