DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE VERACRUZ

RESIDENCIAS PROFESIONALES

ASUNTO:
PRIMER REPORTE

CARRERA:
INGENIERÍA EN SISTEMAS COMPUTACIONALES

PROYECTO:
ARTESANÍAS MATEMÁTICAS

EMPRESA:
CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS, A.C.

PRESENTA:
GUERRERO MONTERO FERNANDO

NÚMERO CONTROL:
E16020823

PERIODO:
FEBRERO - JULIO 2021

Asesor externo
Mtro. Maximino Tapia Rodríguez

Asesor interno
Lsca. Jesús Cruzado Calleja

ÍNDICE

ÍNDICE DE FIGURAS

PRIMER REPORTE DE ACTIVIDADES

En este informe se presentan las actividades realizadas en las primeras semanas del proyecto de residencias profesionales, apegándose a lo plasmado en el cronograma.

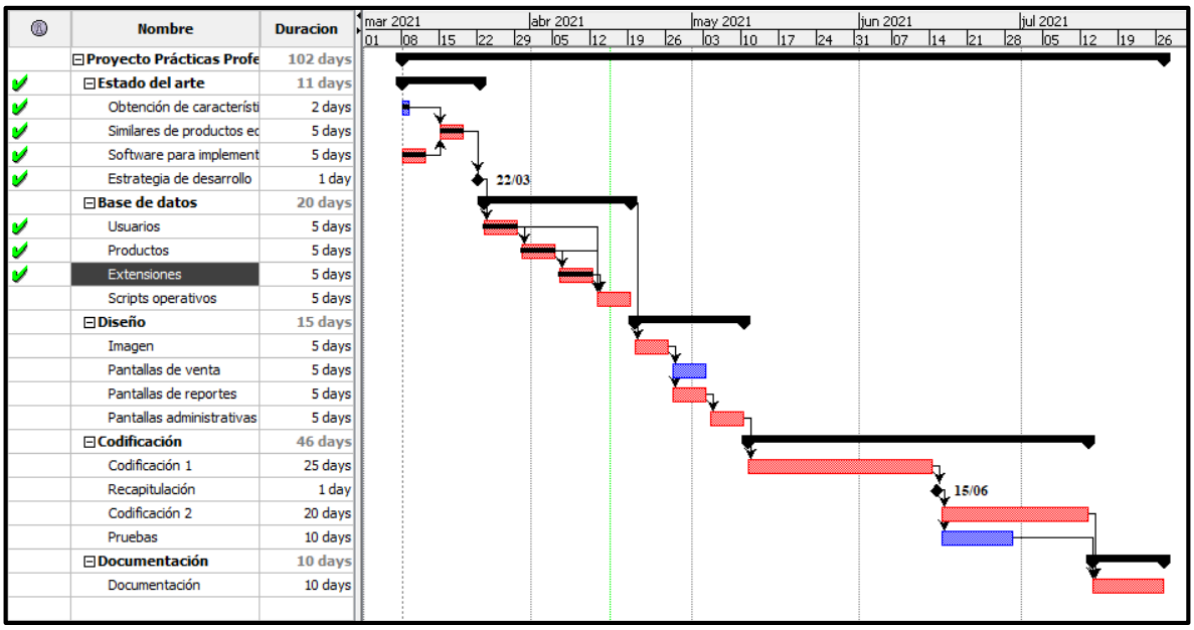


Figura 1. Cronograma de actividades.

Como se puede apreciar en el Cronograma los plazos marcados se han cumplido hasta la fecha y se ha completado la etapa de “Estado de arte” y gran parte de la etapa “Base de datos”. Posterior a este informe, se empezará a realizar la etapa de “Diseño” y la etapa “Base de datos” se puede retomar para finalizarla después.

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 2. Captura 1 de Aprendiendo Matemáticas.

Pantalla de productos:

Figura 3. Captura 2 de Aprendiendo Matemáticas.

Pantalla de carrito:

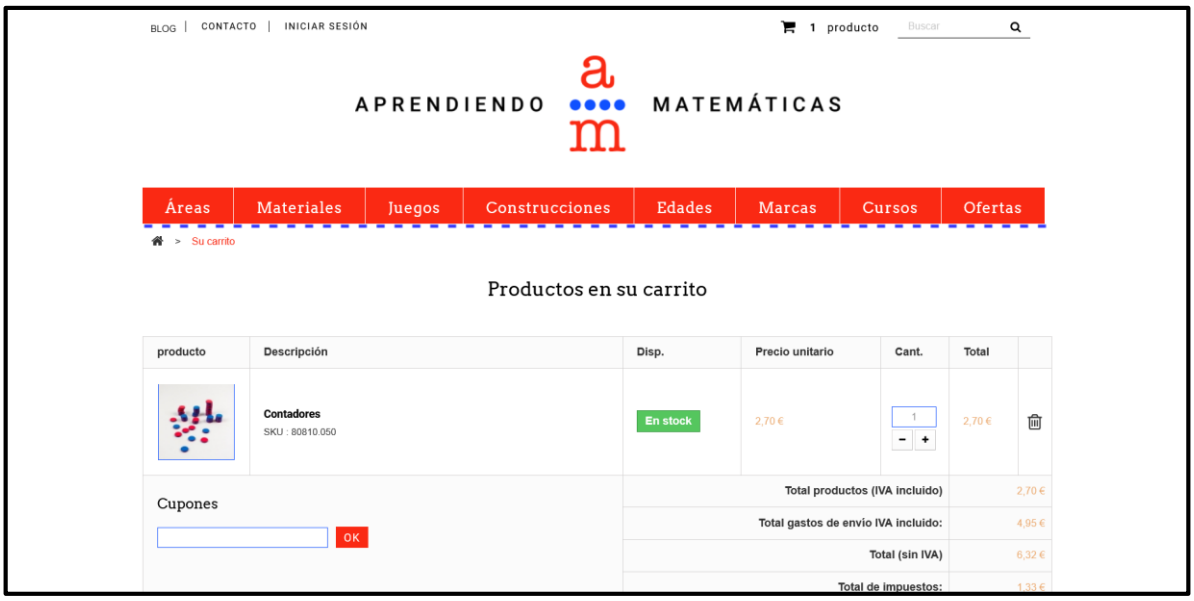


Figura 4. Captura 3 de Aprendiendo Matemáticas.

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

Pantalla de inicio:

Figura 5. Captura 1 de ideashands.

Pantalla de productos:

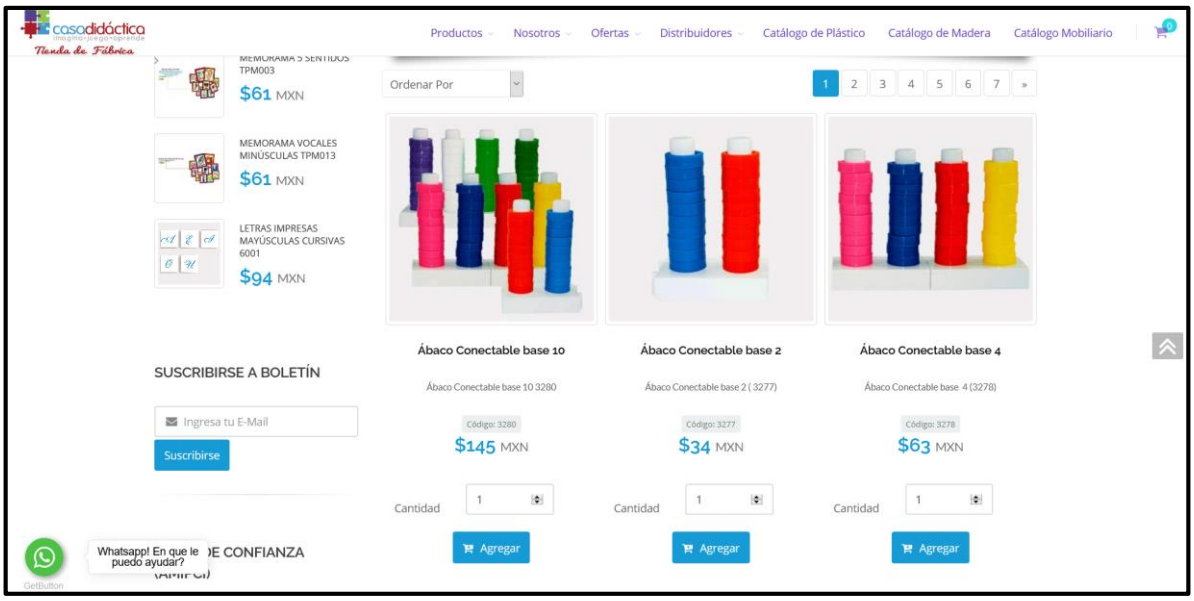


Figura 6. Captura 2 de ideashands.

Pantalla de carrito:

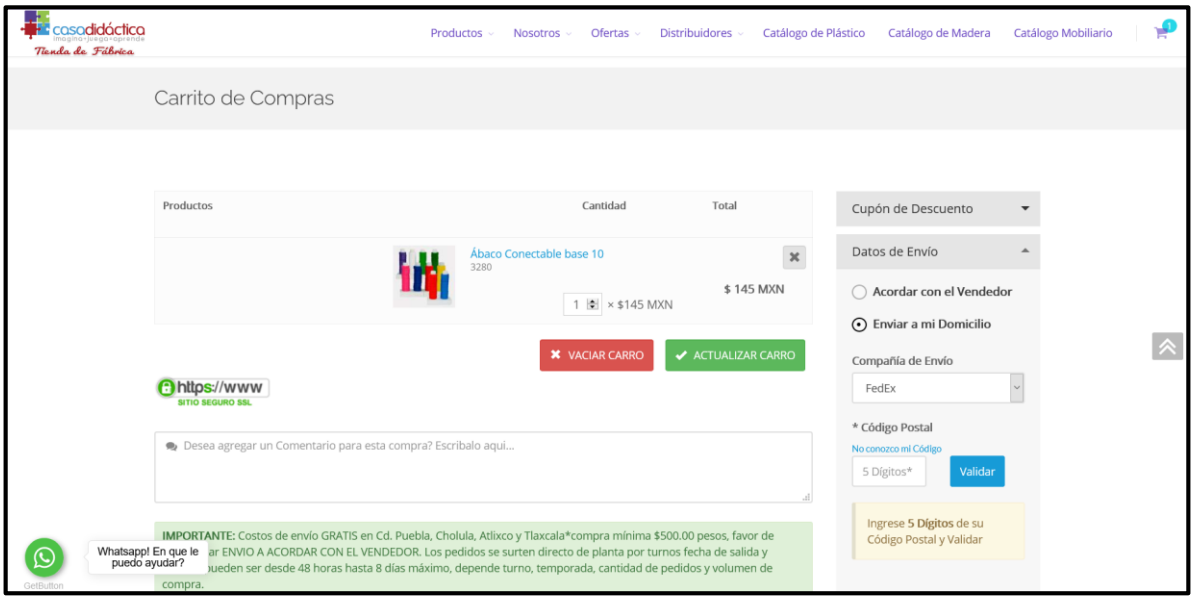


Figura 7. Captura 3 de ideashands.

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

Pantalla de inicio:

Figura 8. Captura 1 de Grupo Educar.

Pantalla de productos:

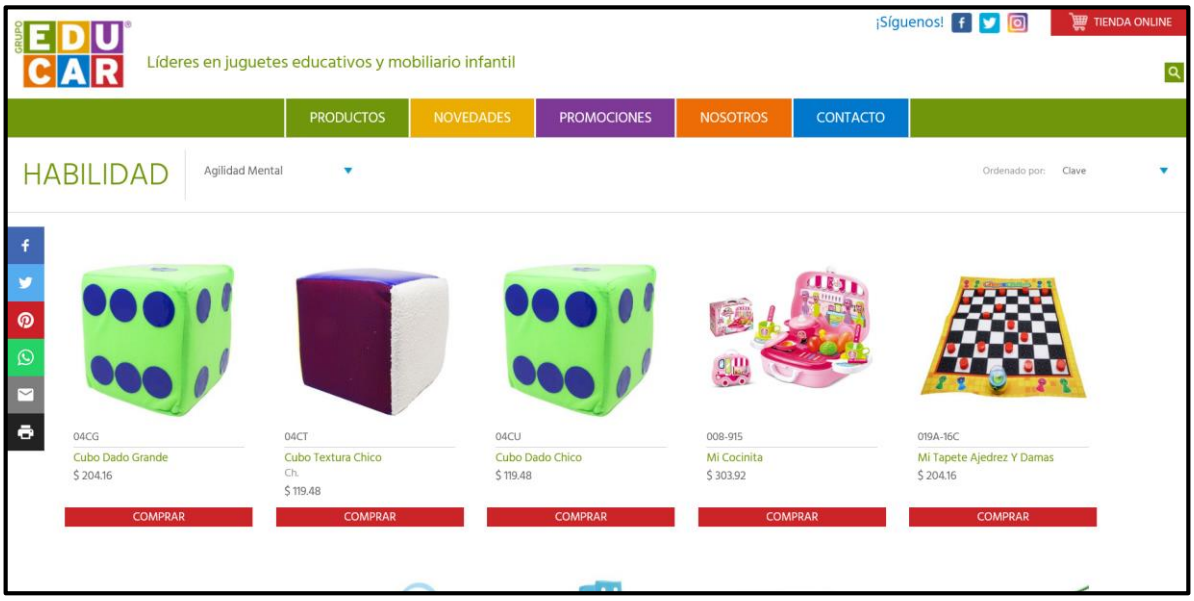


Figura 9. Captura 2 de Grupo Educar.

Pantalla de “Nosotros”:

Figura 10. Captura 3 de Grupo Educar.

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

Pantalla de inicio 1:

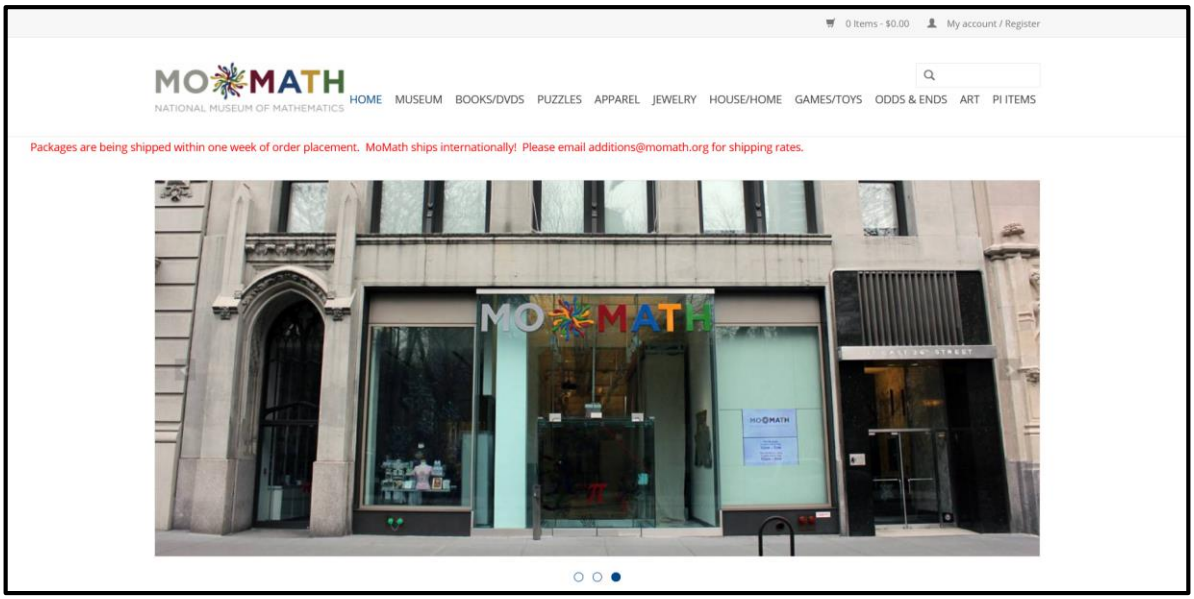


Figura 11. Captura 1 de National Museum of Mathematics.

Pantalla de inicio 2:

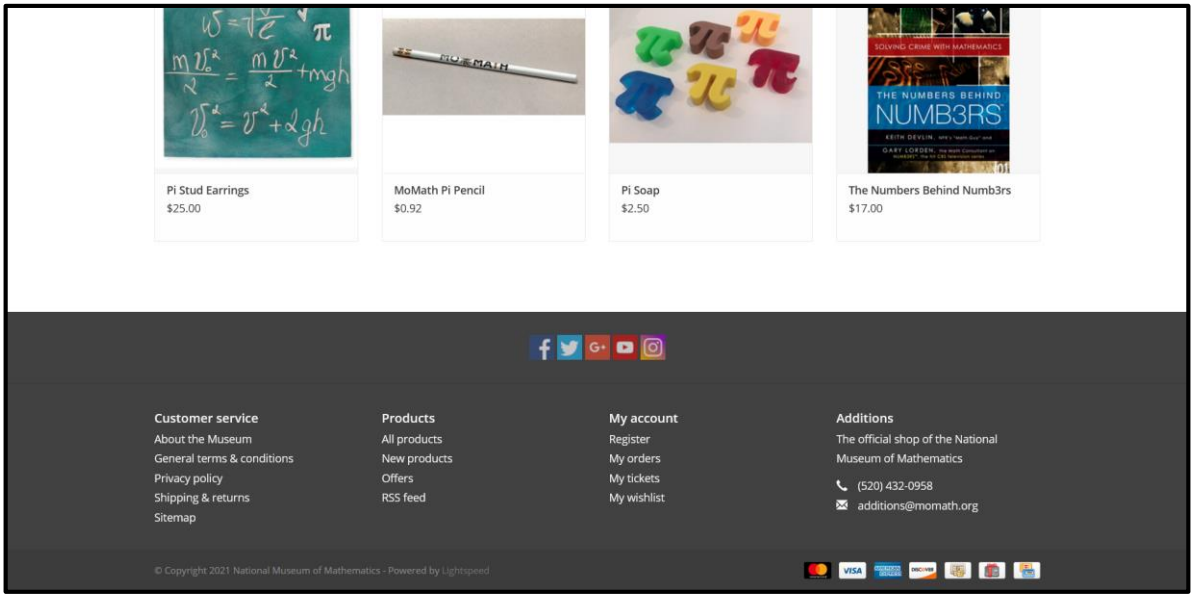


Figura 12. Captura 2 de National Museum of Mathematics.

Pantalla de productos:

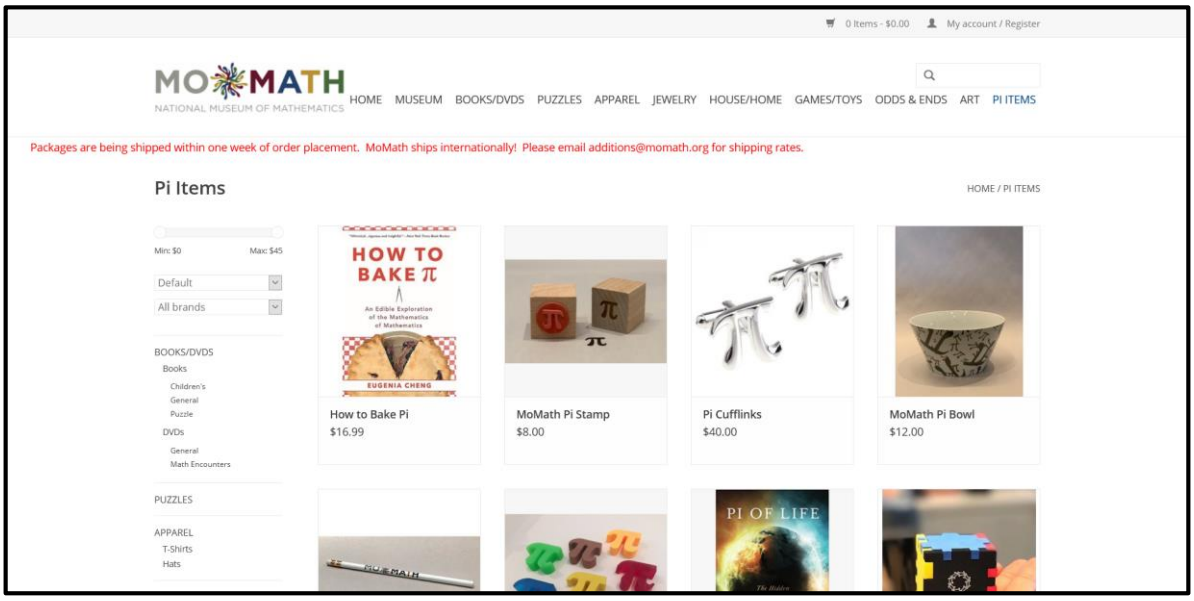


Figura 13. Captura 3 de National Museum of Mathematics.

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

Pantalla de inicio:

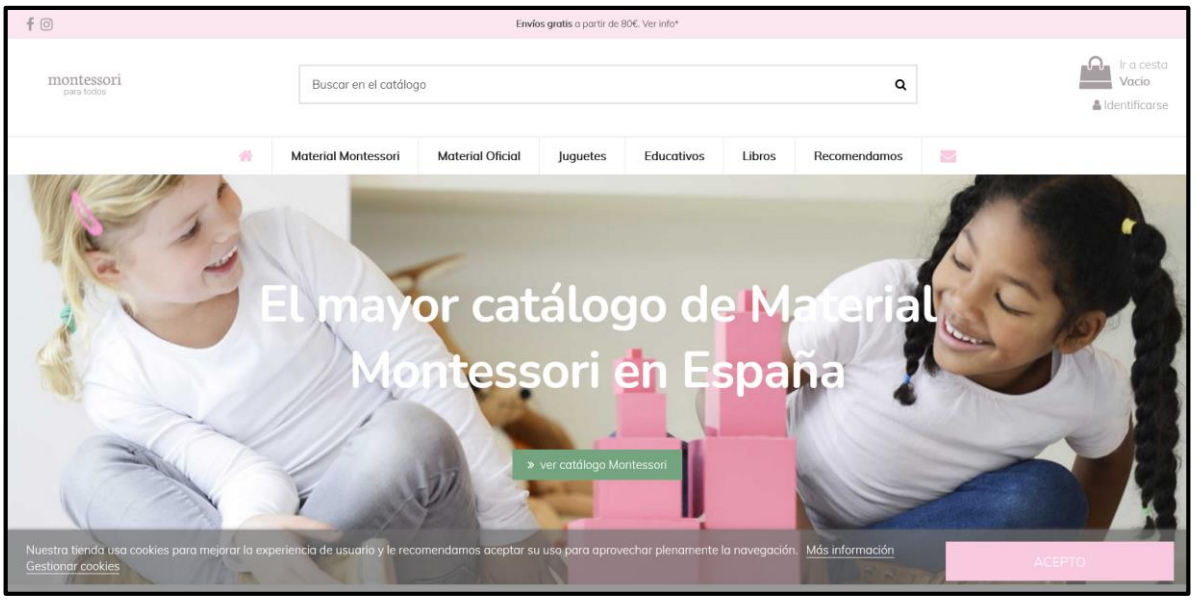


Figura 14. Captura 1 de Montessori Para Todos.

Pantalla de productos:

Figura 15. Captura 2 de Montessori Para Todos.

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

Pantalla de inicio:

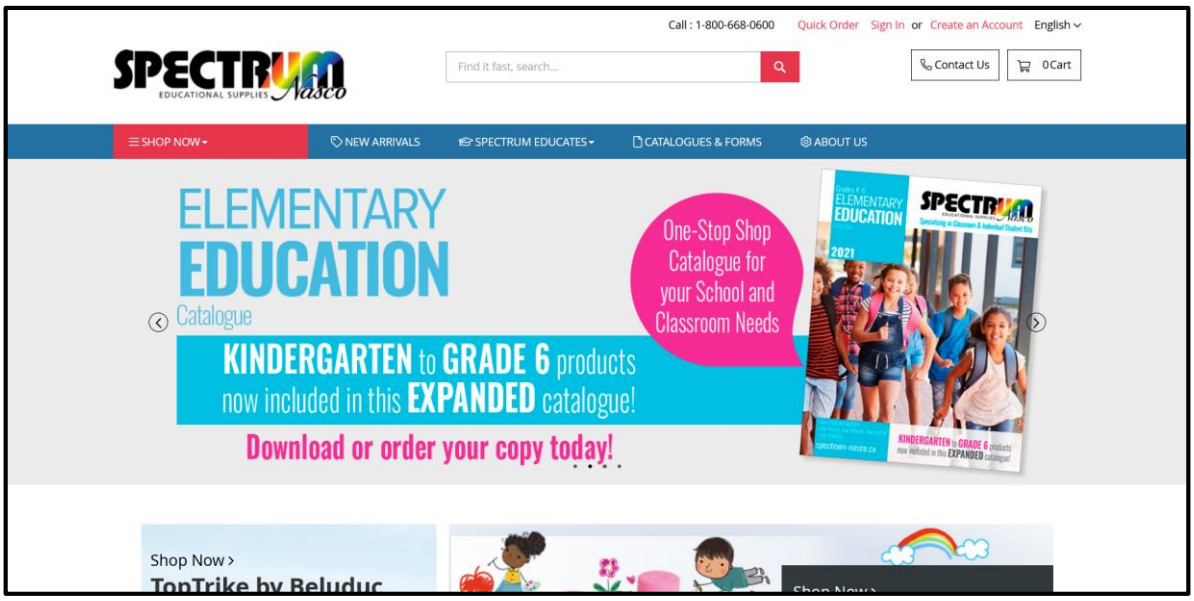


Figura 16. Captura 1 de Spectrum.

Pantalla de productos:

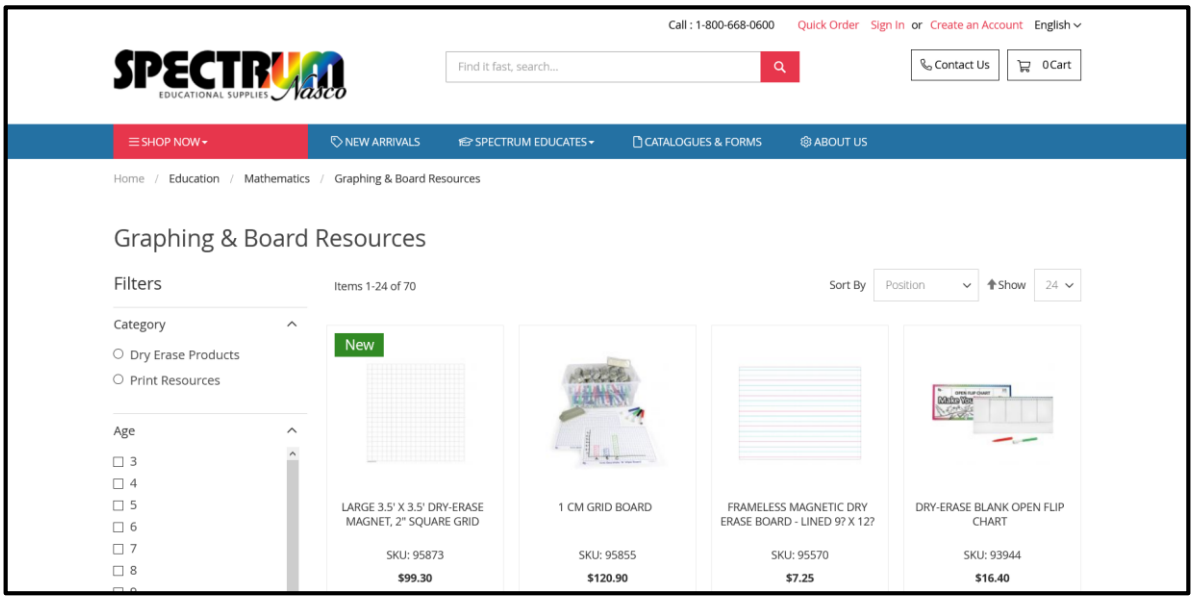


Figura 17. Captura 2 de Spectrum.

Pantalla de login:

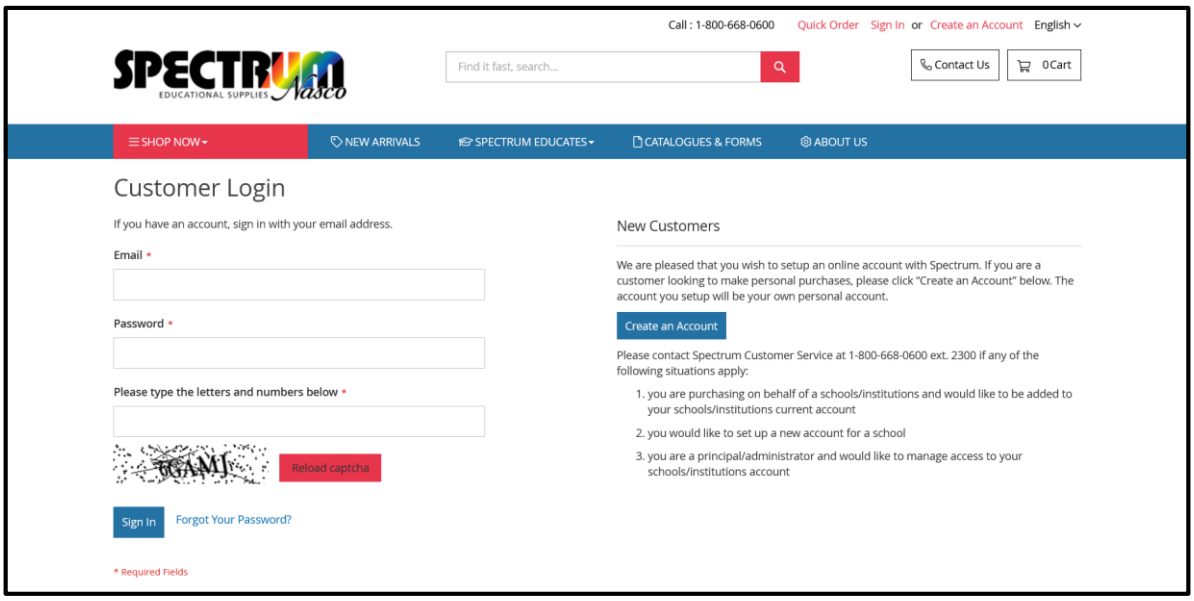


Figura 18. Captura 3 de Spectrum.

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	Sin referencias de producto

	Vender de todo Hasta 20 ventas al día Anuncio poco intrusivo Buenas opciones para pagos	Sin opciones SEO Sólo en inglés 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	Muy técnico Necesitas usar WordPress No tiene soporte Buscar propio alojamiento web

	Elegante presentación	
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

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 Múltiples opciones de pago	Mayoría de integraciones a través de aplicaciones que ofrece no son gratuitas
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

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

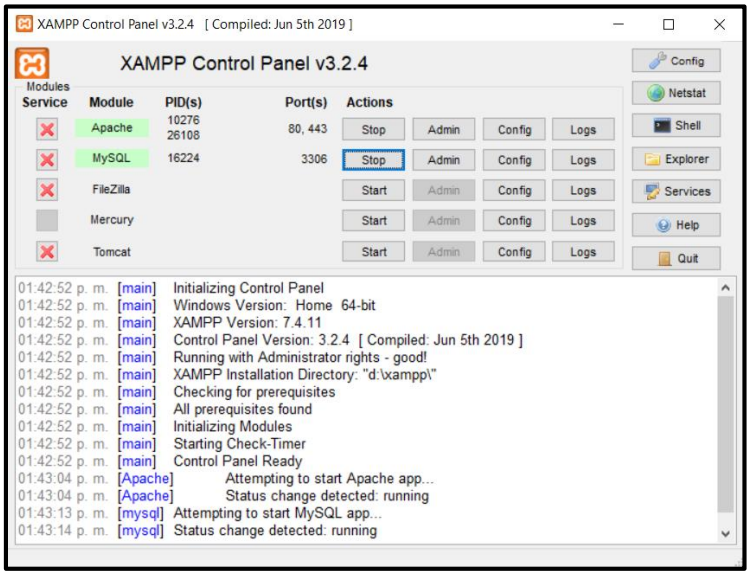


Figura 19. Panel de Control de XAMPP

BASE DE DATOS

Diseño de la Base de Datos

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

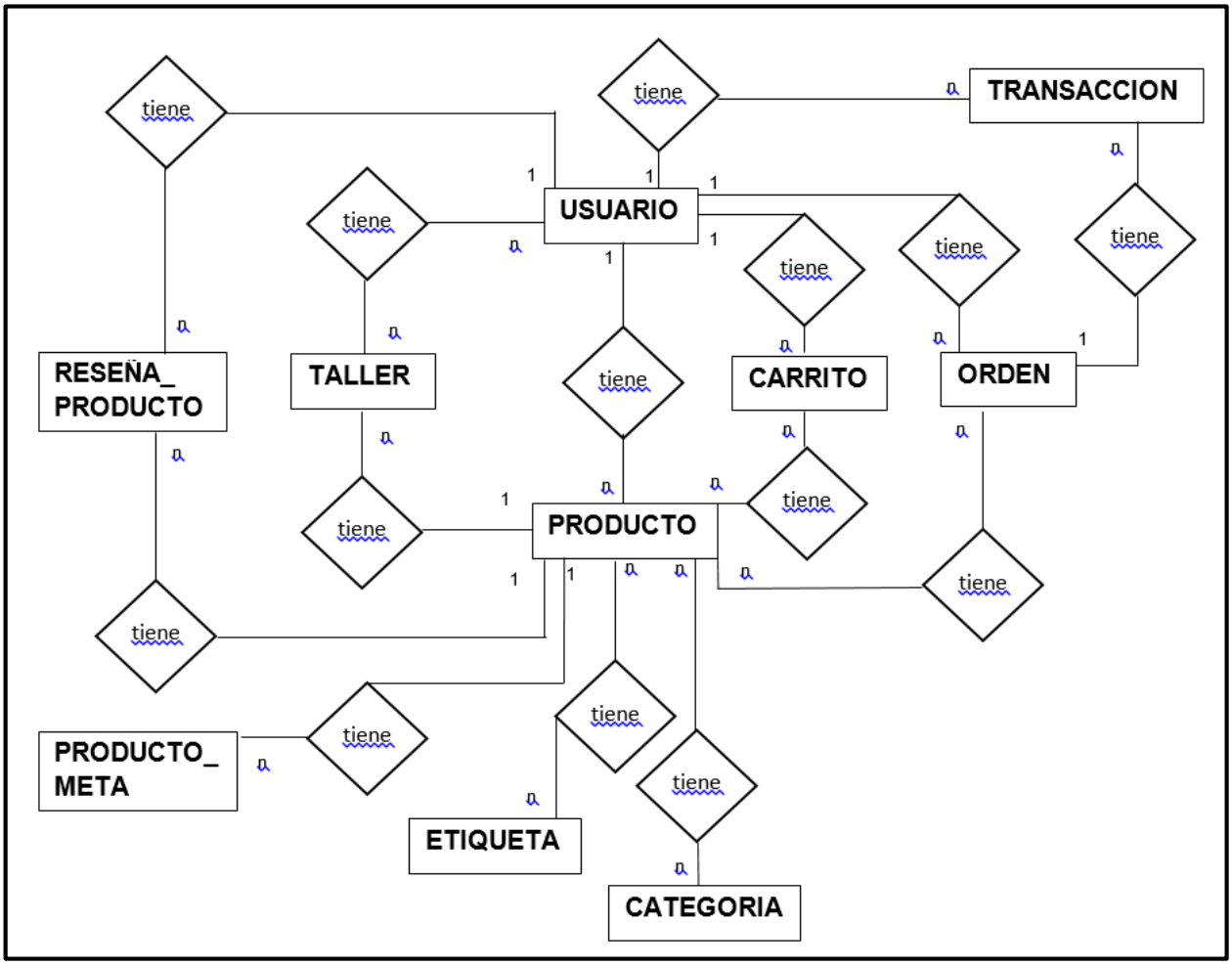


Figura 20. Modelo Entidad-Relación.

De igual forma, se complementó con el siguiente Modelo Relacional usando la herramienta en línea SqlDBM:

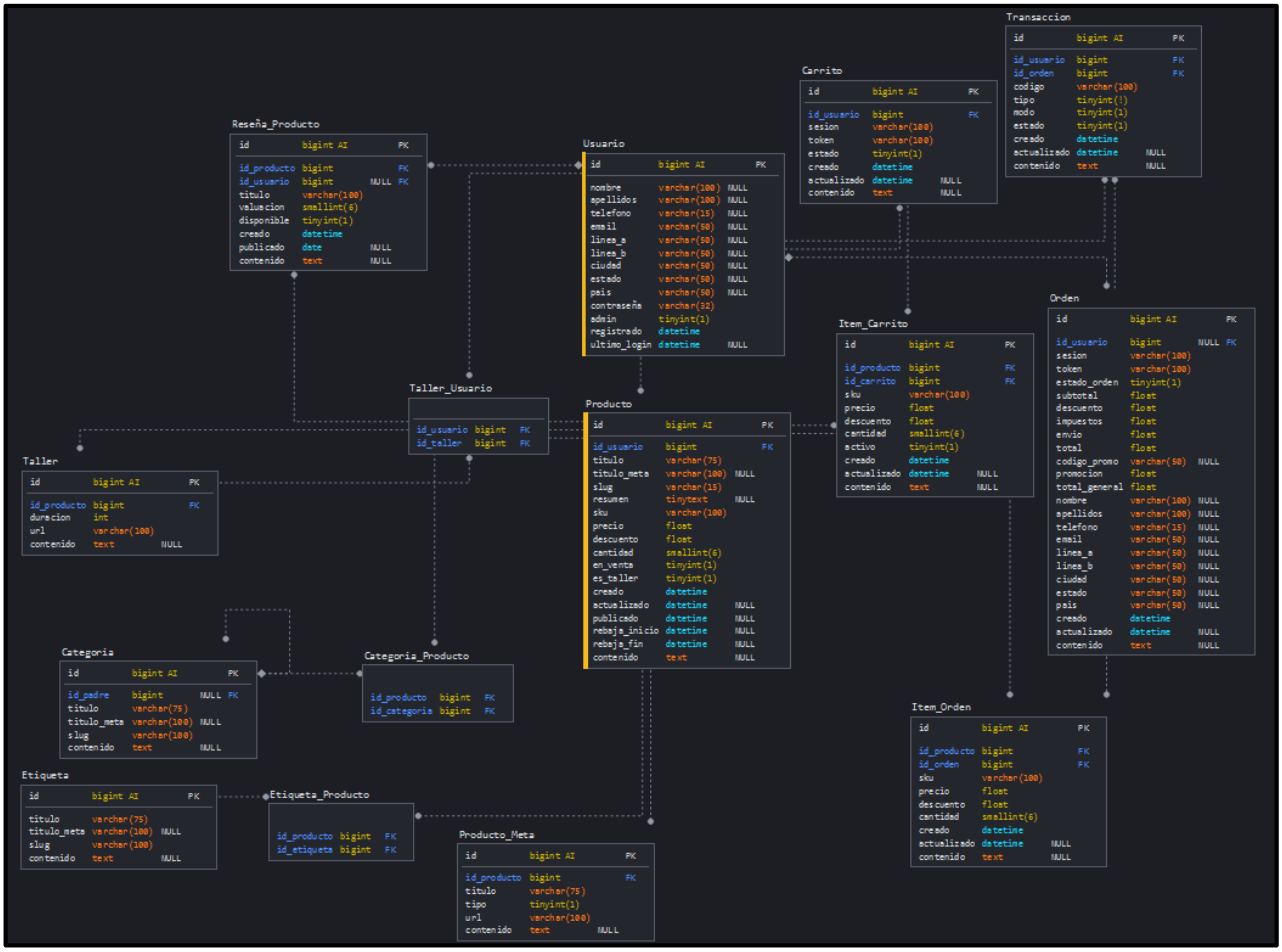


Figura 21. Modelo Relacional.

A continuación se presenta el Diccionario de Datos realizado para complementar el diseño de la Base de Datos, lo cual todo fue revisado y aceptado:

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)		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		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.
titulo	varchar(75)	not null	El título del recurso. Puede usarse como texto alterno al recurso si falla la carga.
tipo	tinyint(1)	not null, default 0	El tipo del meta del producto puede ser Imagen (0), Vídeo (1), y 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.

Figura 22. Diccionario de Datos 1.

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_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.
contenido	text	default null	La columna usada para guardar los detalles 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	tinyint(1)	not null, default 0	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.

Figura 23. Diccionario de Datos 2.

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	tinyint(1)	not null, default 0	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	tinyint(1)	not null	El tipo de la transacción de la orden puede ser Crédito (0) o Débito (1).
modo	tinyint(1)	not null, default 0	El modo de la transacción de la orden puede ser Offline (0), Envío Contra Reembolso (1), Cheque (2), "Draft" (3), "Wired" (4), y Online (5).
estado	tinyint(1)	not null, default 0	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.

Figura 24. Diccionario de Datos 3.

Creación de la Base de Datos

Se crea la Base de Datos y las tablas explicadas anteriormente, ya sea con comandos SQL o usando la interfaz de phpMyAdmin. Una vez creada se puede exportar a un archivo *.sql* para respaldar.

```sql
CREATE SCHEMA IF NOT EXISTS `artesanias_matematicas` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Usuario` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(100) NULL DEFAULT NULL,
  `apellidos` VARCHAR(100) NULL DEFAULT NULL,
  `telefono` VARCHAR(15) NULL,
  `email` VARCHAR(50) NULL,
  `linea_a` VARCHAR(50) NULL DEFAULT NULL,
  `linea_b` VARCHAR(50) NULL DEFAULT NULL,
  `ciudad` VARCHAR(50) NULL DEFAULT NULL,
  `estado` VARCHAR(50) NULL DEFAULT NULL,
  `pais` VARCHAR(50) NULL DEFAULT NULL,
  `contraseña` VARCHAR(32) NOT NULL,
  `admin` TINYINT(1) NOT NULL DEFAULT 0,
  `registrado` DATETIME NOT NULL,
  `ultimo_login` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `uq_telefono` (`telefono` ASC),
  UNIQUE INDEX `uq_email` (`email` ASC) )
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Producto` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_usuario` BIGINT NOT NULL,
  `titulo` VARCHAR(75) NOT NULL,
  `titulo_meta` VARCHAR(100) NULL,
  `slug` VARCHAR(100) NOT NULL,
  `resumen` TINYTEXT NULL,
  `sku` VARCHAR(100) NOT NULL,
  `precio` FLOAT NOT NULL DEFAULT 0,
  `descuento` FLOAT NOT NULL DEFAULT 0,
  `cantidad` SMALLINT(6) NOT NULL DEFAULT 0,
  `en_venta` TINYINT(1) NOT NULL DEFAULT 0,
  `es_taller` TINYINT(1) NOT NULL DEFAULT 0,
  `creado` DATETIME NOT NULL,
  `actualizado` DATETIME NULL DEFAULT NULL,
  `publicado` DATETIME NULL DEFAULT NULL,
  `rebaja_inicio` DATETIME NULL DEFAULT NULL,
  `rebaja_fin` DATETIME NULL DEFAULT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX 'uq_titulo' (`titulo` ASC),
  UNIQUE INDEX `uq_slug` (`slug` ASC),
  INDEX `idx_producto_usuario` (`id_usuario` ASC),
  CONSTRAINT `fk_producto_usuario`
    FOREIGN KEY (`id_usuario`)
    REFERENCES `artesanias_matematicas`.`Usuario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 25. Creación de tablas de la Base de Datos 1.

```
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;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Taller_Usuario` (
  `id_usuario` BIGINT NOT NULL,
  `id_taller` BIGINT NOT NULL,
  PRIMARY KEY (`id_usuario`, `id_taller`),
  INDEX `idx_tu_taller` (`id_taller` ASC),
  INDEX `idx_tu_usuario` (`id_usuario` ASC),
  CONSTRAINT `fk_tu_usuario`
    FOREIGN KEY (`id_usuario`)
    REFERENCES `artesanias_matematicas`.`Usuario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_tu_taller`
    FOREIGN KEY (`id_taller`)
    REFERENCES `artesanias_matematicas`.`Taller` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Producto_Meta` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_producto` BIGINT NOT NULL,
  `tipo` TINYINT(1) NOT NULL DEFAULT 0,
  `url` VARCHAR(100) NOT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_pm_producto` (`id_producto` ASC),
  UNIQUE INDEX `uq_pm_producto` (`id_producto` ASC),
  CONSTRAINT `fk_pm_producto`
    FOREIGN KEY (`id_producto`)
    REFERENCES `artesanias_matematicas`.`Producto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Reseña_Producto` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_producto` BIGINT NOT NULL,
  `id_usuario` BIGINT NULL DEFAULT NULL,
  `titulo` VARCHAR(100) NOT NULL,
  `valuacion` SMALLINT(6) NOT NULL DEFAULT 0,
```

Figura 26. Creación de tablas de la Base de Datos 2.

```
  `disponible` TINYINT(1) NOT NULL DEFAULT 0,
  `creado` DATETIME NOT NULL,
  `publicado` DATETIME NULL DEFAULT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_rp_producto` (`id_producto` ASC),
  INDEX `idx_rp_padre` (`id_usuario` ASC),
  CONSTRAINT `fk_rp_producto`
    FOREIGN KEY (`id_producto`)
    REFERENCES `artesanias_matematicas`.`Producto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_rp_usuario`
    FOREIGN KEY (`id_usuario`)
    REFERENCES `artesanias_matematicas`.`Usuario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Categoria` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_padre` BIGINT NULL DEFAULT NULL,
  `titulo` VARCHAR(75) NOT NULL,
  `titulo_meta` VARCHAR(100) NULL DEFAULT NULL,
  `slug` VARCHAR(100) NOT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_categoria_padre` (`id_padre` ASC),
  CONSTRAINT `fk_categoria_padre`
    FOREIGN KEY (`id_padre`)
    REFERENCES `artesanias_matematicas`.`Categoria` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Categoria_Producto` (
  `id_producto` BIGINT NOT NULL,
  `id_categoria` BIGINT NOT NULL,
  PRIMARY KEY (`id_producto`, `id_categoria`),
  INDEX `idx_cp_categoria` (`id_categoria` ASC),
  INDEX `idx_cp_producto` (`id_producto` ASC),
  CONSTRAINT `fk_cp_producto`
    FOREIGN KEY (`id_producto`)
    REFERENCES `artesanias_matematicas`.`Producto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_cp_categoria`
    FOREIGN KEY (`id_categoria`)
    REFERENCES `artesanias_matematicas`.`Categoria` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 27. Creación de tablas de la Base de Datos 3.

```sql
CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Etiqueta` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `titulo` VARCHAR(75) NOT NULL,
  `titulo_meta` VARCHAR(100) NULL DEFAULT NULL,
  `slug` VARCHAR(100) NOT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Etiqueta_Producto` (
  `id_producto` BIGINT NOT NULL,
  `id_etiqueta` BIGINT NOT NULL,
  PRIMARY KEY (`id_producto`, `id_etiqueta`),
  INDEX `idx_ep_etiqueta` (`id_etiqueta` ASC),
  INDEX `idx_ep_producto` (`id_producto` ASC),
  CONSTRAINT `fk_ep_producto`
    FOREIGN KEY (`id_producto`)
    REFERENCES `artesanias_matematicas`.`Producto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_ep_etiqueta`
    FOREIGN KEY (`id_etiqueta`)
    REFERENCES `artesanias_matematicas`.`Etiqueta` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Carrito` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_usuario` BIGINT NULL DEFAULT NULL,
  `sesion` VARCHAR(100) NOT NULL,
  `token` VARCHAR(100) NOT NULL,
  `estado` TINYINT(1) NOT NULL DEFAULT 0,
  `creado` DATETIME NOT NULL,
  `actualizado` DATETIME NULL DEFAULT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_carrito_usuario` (`id_usuario` ASC),
  CONSTRAINT `fk_carrito_usuario`
    FOREIGN KEY (`id_usuario`)
    REFERENCES `artesanias_matematicas`.`Usuario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 28. Creación de tablas de la Base de Datos 4.

```
CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Item_Carrito` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_producto` BIGINT NOT NULL,
  `id_carrito` BIGINT NOT NULL,
  `sku` VARCHAR(100) NOT NULL,
  `precio` FLOAT NOT NULL DEFAULT 0,
  `descuento` FLOAT NOT NULL DEFAULT 0,
  `cantidad` SMALLINT(6) NOT NULL DEFAULT 0,
  `activo` TINYINT(1) NOT NULL DEFAULT 0,
  `creado` DATETIME NOT NULL,
  `actualizado` DATETIME NULL DEFAULT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_ic_producto` (`id_producto` ASC),
  INDEX `idx_ic_carrito` (`id_carrito` ASC),
  CONSTRAINT `fk_ic_producto`
    FOREIGN KEY (`id_producto`)
    REFERENCES `artesanias_matematicas`.`Producto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_ic_carrito`
    FOREIGN KEY (`id_carrito`)
    REFERENCES `artesanias_matematicas`.`Carrito` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 29. Creación de tablas de la Base de Datos 5.

```
CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Orden` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_usuario` BIGINT NULL DEFAULT NULL,
  `sesion` VARCHAR(100) NOT NULL,
  `token` VARCHAR(100) NOT NULL,
  `estado_orden` SMALLINT(6) NOT NULL DEFAULT 0,
  `subtotal` FLOAT NOT NULL DEFAULT 0,
  `descuento` FLOAT NOT NULL DEFAULT 0,
  `impuestos` FLOAT NOT NULL DEFAULT 0,
  `envio` FLOAT NOT NULL DEFAULT 0,
  `total` FLOAT NOT NULL DEFAULT 0,
  `codigo_promo` VARCHAR(50) NULL DEFAULT NULL,
  `promocion` FLOAT NOT NULL DEFAULT 0,
  `total_general` FLOAT NOT NULL DEFAULT 0,
  `nombre` VARCHAR(50) NULL DEFAULT NULL,
  `apellidos` VARCHAR(50) NULL DEFAULT NULL,
  `telefono` VARCHAR(15) NULL,
  `email` VARCHAR(50) NULL,
  `linea_a` VARCHAR(50) NULL DEFAULT NULL,
  `linea_b` VARCHAR(50) NULL DEFAULT NULL,
  `ciudad` VARCHAR(50) NULL DEFAULT NULL,
  `estado` VARCHAR(50) NULL DEFAULT NULL,
  `pais` VARCHAR(50) NULL DEFAULT NULL,
  `creado` DATETIME NOT NULL,
  `actualizado` DATETIME NULL DEFAULT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_orden_usuario` (`id_usuario` ASC),
  CONSTRAINT `fk_orden_usuario`
    FOREIGN KEY (`id_usuario`)
    REFERENCES `artesanias_matematicas`.`Usuario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 30. Creación de tablas de la Base de Datos 6.

```sql
CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Item_Orden` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_producto` BIGINT NOT NULL,
  `id_orden` BIGINT NOT NULL,
  `sku` VARCHAR(100) NOT NULL,
  `precio` FLOAT NOT NULL DEFAULT 0,
  `descuento` FLOAT NOT NULL DEFAULT 0,
  `cantidad` SMALLINT(6) NOT NULL DEFAULT 0,
  `creado` DATETIME NOT NULL,
  `actualizado` DATETIME NULL DEFAULT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_io_producto` (`id_producto` ASC),
  INDEX `idx_io_orden` (`id_orden` ASC),
  CONSTRAINT `fk_io_producto`
    FOREIGN KEY (`id_producto`)
    REFERENCES `artesanias_matematicas`.`Producto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_io_orden`
    FOREIGN KEY (`id_orden`)
    REFERENCES `artesanias_matematicas`.`Orden` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `artesanias_matematicas`.`Transaccion` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `id_usuario` BIGINT NOT NULL,
  `id_orden` BIGINT NOT NULL,
  `codigo` VARCHAR(100) NOT NULL,
  `tipo` TINYINT(1) NOT NULL DEFAULT 0,
  `modo` TINYINT(1) NOT NULL DEFAULT 0,
  `estado` TINYINT(1) NOT NULL DEFAULT 0,
  `creado` DATETIME NOT NULL,
  `actualizado` DATETIME NULL DEFAULT NULL,
  `contenido` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_transaccion_usuario` (`id_usuario` ASC),
  INDEX `idx_transaccion_orden` (`id_orden` ASC),
  CONSTRAINT `fk_transaccion_usuario`
    FOREIGN KEY (`id_usuario`)
    REFERENCES `artesanias_matematicas`.`Usuario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_transaccion_orden`
    FOREIGN KEY (`id_orden`)
    REFERENCES `artesanias_matematicas`.`Orden` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 31. Creación de tablas de la Base de Datos 7.

Procedimientos Almacenados

Para realizar las consultas y movimientos correspondientes en la Base de Datos se utilizará Procedimientos Almacenados. Estos *Stored Procedures* están guardados en el servidor de la Base de Datos y contienen comandos SQL para realizar algún procedimiento. Se podrán llamar después cuando se esté realizando la codificación de la página web.

```
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 ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `f_obtener_id_usuario_producto`(`titulo` VARCHAR(75) CHARSET utf8mb4) RETURNS bigint(20)
    READS SQL DATA
    COMMENT 'Retornar el id_usuario de un producto con la columna titulo.'
BEGIN
RETURN (
    SELECT id_usuario
    FROM `producto`
    WHERE `producto`.`titulo` = titulo);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `f_obtener_id_etiqueta`(`titulo` VARCHAR(75) CHARSET utf8mb4) RETURNS bigint(20)
    READS SQL DATA
    COMMENT 'Retornar el id_etiqueta usando la columna titulo.'
BEGIN
RETURN (
    SELECT id
    FROM `etiqueta`
    WHERE `etiqueta`.`titulo` = titulo);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `f_obtener_id_categoria`(`titulo` VARCHAR(75) CHARSET utf8mb4) RETURNS bigint(20)
    READS SQL DATA
    COMMENT 'Retornar el id_categoria usando la columna de titulo.'
BEGIN
RETURN (
    SELECT id
    FROM `categoria`
    WHERE `categoria`.`titulo` = titulo);
END$$
DELIMITER ;
```

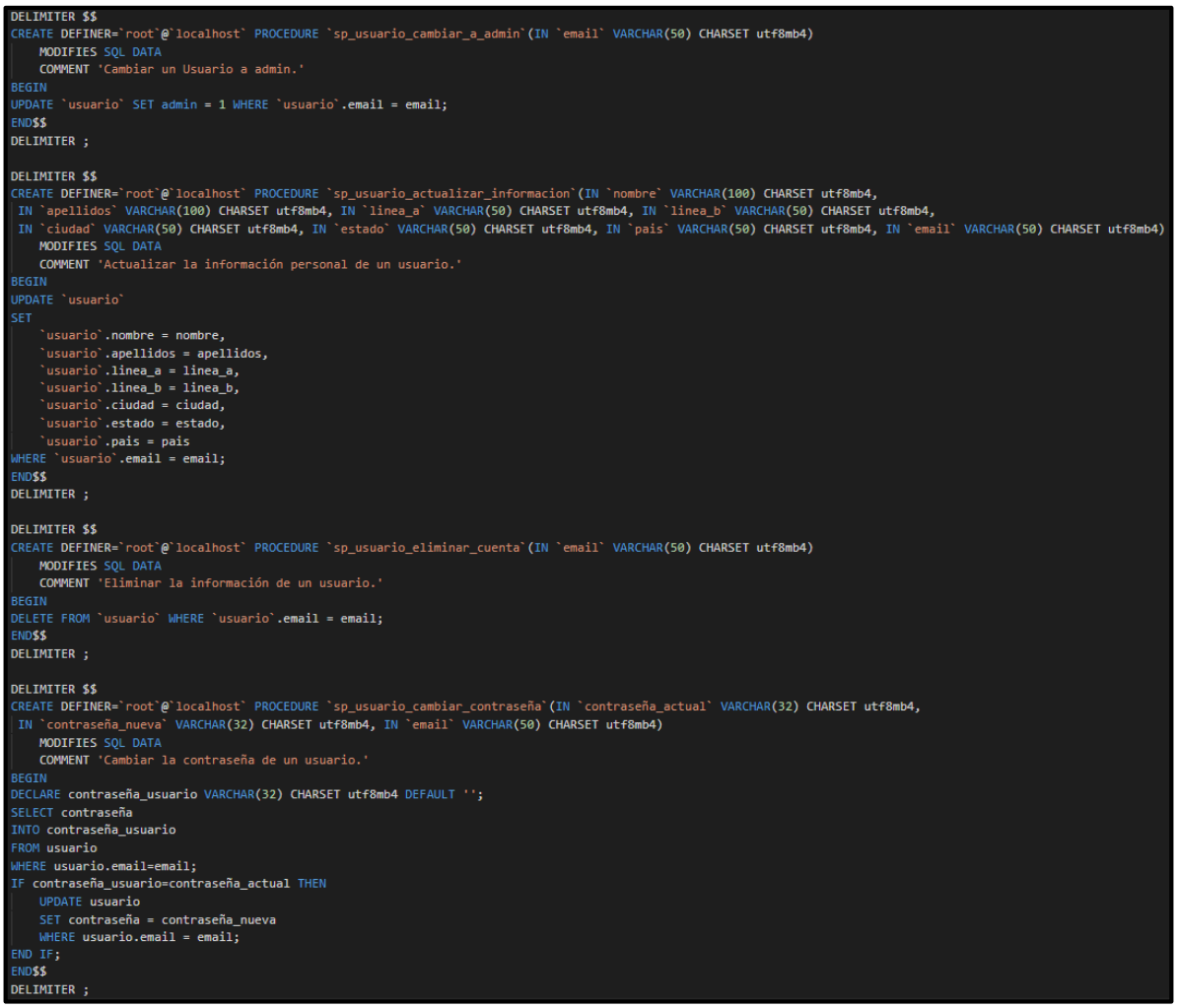
Figura 32. Funciones de la Base de Datos 1.

```sql
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `f_obtener_id_usuario`(`email` VARCHAR(50) CHARSET utf8mb4) RETURNS bigint(20)
    READS SQL DATA
    COMMENT 'Retornar el id_usuario con la columna email.'
BEGIN
RETURN (
    SELECT id
    FROM `usuario`
    WHERE `usuario`.`email` = email);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `f_obtener_id_orden`(`email` VARCHAR(50) CHARSET utf8mb4,
`sesion` VARCHAR(100) CHARSET utf8mb4, `token` VARCHAR(100) CHARSET utf8mb4) RETURNS bigint(20)
    READS SQL DATA
    COMMENT 'Retornar el id_orden con la columna sesion y token.'
BEGIN
RETURN (
    SELECT id
    FROM `orden`
    WHERE
        `orden`.`id_usuario` = f_obtener_id_usuario(email) AND
        `orden`.`sesion` = sesion AND
        `orden`.`token` = token
);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `f_obtener_id_producto`(`titulo` VARCHAR(75) CHARSET utf8mb4) RETURNS bigint(20)
    READS SQL DATA
    COMMENT 'Retornar el id_producto con la columna titulo.'
BEGIN
RETURN (
    SELECT id
    FROM `producto`
    WHERE `producto`.`titulo` = titulo);
END$$
DELIMITER ;
```

Figura 33. Funciones de la Base de Datos 2.



```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_cambiar_a_admin`(IN `email` VARCHAR(50) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar un Usuario a admin.'
BEGIN
UPDATE `usuario` SET admin = 1 WHERE `usuario`.email = email;
END$$
DELIMITER ;

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 ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_eliminar_cuenta`(IN `email` VARCHAR(50) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Eliminar la información de un usuario.'
BEGIN
DELETE FROM `usuario` WHERE `usuario`.email = email;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_cambiar_contraseña`(IN `contraseña_actual` VARCHAR(32) CHARSET utf8mb4,
 IN `contraseña_nueva` VARCHAR(32) CHARSET utf8mb4, IN `email` VARCHAR(50) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar la contraseña de un usuario.'
BEGIN
DECLARE contraseña_usuario VARCHAR(32) CHARSET utf8mb4 DEFAULT '';
SELECT contraseña
INTO contraseña_usuario
FROM usuario
WHERE usuario.email=email;
IF contraseña_usuario=contraseña_actual THEN
    UPDATE usuario
    SET contraseña = contraseña_nueva
    WHERE usuario.email = email;
END IF;
END$$
DELIMITER ;
```

Figura 34. Procedimientos Almacenados para Usuario 1.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_obtener_talleres`(IN `email` VARCHAR(50) CHARSET utf8mb4)
    READS SQL DATA
    COMMENT 'Obtener información de los talleres que pertenecen al usuario.'
BEGIN
SELECT t.id, t.duracion, t.url
FROM usuario u, taller t, taller_usuario tu
WHERE u.`email` = email AND u.id = tu.id_usuario AND t.id = tu.id_taller
ORDER BY t.id ASC;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_insert`(IN `nombre` VARCHAR(100) CHARSET utf8mb4,
 IN `apellidos` VARCHAR(100) CHARSET utf8mb4, IN `telefono` VARCHAR(15) CHARSET utf8mb4, IN `email` VARCHAR(50) CHARSET utf8mb4,
 IN `linea_a` VARCHAR(50) CHARSET utf8mb4, IN `linea_b` VARCHAR(50) CHARSET utf8mb4, IN `ciudad` VARCHAR(50) CHARSET utf8mb4,
 IN `estado` VARCHAR(50) CHARSET utf8mb4, IN `pais` VARCHAR(50) CHARSET utf8mb4, IN `contraseña` VARCHAR(32) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Insertar un Usuario con toda la información correspondiente.'
BEGIN
INSERT INTO usuario (nombre,apellidos,telefono,email,linea_a,linea_b,ciudad,estado,pais,contraseña,registrado)
VALUES (nombre,apellidos,telefono,email,linea_a,linea_b,ciudad,estado,pais,contraseña,NOW() );
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_cambiar_correo`(IN `email_viejo` VARCHAR(50) CHARSET utf8mb4,
 IN `email_nuevo` VARCHAR(50) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar el correo relacionado a la cuenta del usuario.'
BEGIN
UPDATE `usuario` SET `usuario`.email = email_nuevo WHERE `usuario`.email = email_viejo;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_insert_admin`(IN `nombre` VARCHAR(100) CHARSET utf8mb4,
 IN `apellidos` VARCHAR(100) CHARSET utf8mb4, IN `telefono` VARCHAR(15) CHARSET utf8mb4, IN `email` VARCHAR(50) CHARSET utf8mb4,
 IN `linea_a` VARCHAR(50) CHARSET utf8mb4, IN `linea_b` VARCHAR(50) CHARSET utf8mb4, IN `ciudad` VARCHAR(50) CHARSET utf8mb4,
 IN `estado` VARCHAR(50) CHARSET utf8mb4, IN `pais` VARCHAR(50) CHARSET utf8mb4, IN `contraseña` VARCHAR(32) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Crear Usuario admin en la base de datos.'
BEGIN
INSERT INTO usuario (nombre,apellidos,telefono,email,linea_a,linea_b,ciudad,estado,pais,contraseña,admin,registrado)
VALUES (nombre,apellidos,telefono,email,linea_a,linea_b,ciudad,estado,pais,contraseña,1,NOW() );
END$$
DELIMITER ;
```

Figura 35. Procedimientos Almacenados para Usuario 2.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_select_email`(IN `email` VARCHAR(50) CHARSET utf8)
    READS SQL DATA
    COMMENT 'Obtener información del usuario con la columna email.'
BEGIN
SELECT id,nombre,apellidos,telefono,email,linea_a,linea_b,ciudad,estado,pais,registrado
FROM
    usuario
WHERE
    usuario.email = email;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=```root`@`localhost` PROCEDURE `sp_usuario_actualizar_ultimo_login`(IN `email` VARCHAR(50) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Actualizar la fecha y hora del ultimo login del usuario.'
BEGIN
UPDATE `usuario`
SET ultimo_login = NOW()
WHERE `usuario`.`email` = email;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_login_telefono`(IN `telefono` VARCHAR(15) CHARSET utf8mb4,
 IN `contraseña` VARCHAR(32) CHARSET utf8mb4)
    READS SQL DATA
    COMMENT 'Verifica la información del usuario para login con teléfono.'
BEGIN
DECLARE contraseña_usuario VARCHAR(32) CHARSET utf8mb4 DEFAULT '';
SELECT contraseña
INTO contraseña_usuario
FROM usuario
WHERE usuario.telefono=telefono;
IF contraseña_usuario=contraseña THEN
    SELECT 1;
ELSE
    SELECT 0;
END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_cambiar_telefono`(IN `email` VARCHAR(50) CHARSET utf8mb4,
 IN `telefono` VARCHAR(15) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar un teléfono registrado por otro.'
BEGIN
UPDATE `usuario`
SET `usuario`.telefono = telefono
WHERE `usuario`.email = email;
END$$
DELIMITER ;
```

Figura 36. Procedimientos Almacenados para Usuario 3.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_login_correo`(IN `email` VARCHAR(50) CHARSET utf8mb4,
 IN `contraseña` VARCHAR(32) CHARSET utf8mb4)
    READS SQL DATA
    COMMENT 'Verifica la información del usuario para login con email.'
BEGIN
DECLARE contraseña_usuario VARCHAR(32) CHARSET utf8mb4 DEFAULT '';
SELECT contraseña
INTO contraseña_usuario
FROM usuario
WHERE usuario.email=email;
IF contraseña_usuario=contraseña THEN
    SELECT 1;
ELSE
    SELECT 0;
END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_usuario_crear_carrito`(IN `email` VARCHAR(50) CHARSET utf8mb4,
 IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Crear el carrito asociado al usuario.'
BEGIN
DECLARE id_usuario BIGINT;
SET id_usuario = f_obtener_id_usuario(email);
INSERT INTO carrito (id_usuario,sesion,token,creado)
VALUES (id_usuario,sesion,token,NOW());
END$$
DELIMITER ;
```

Figura 37. Procedimientos Almacenados para Usuario 4.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_agregar_descuento`(IN `titulo` VARCHAR(75) CHARSET utf8mb4,
 IN `descuento` FLOAT, IN `rebaja_inicio` DATETIME, IN `rebaja_fin` DATETIME)
    MODIFIES SQL DATA
    COMMENT 'Agregar descuento a un producto por el período especificado.'
BEGIN
UPDATE `producto`
SET
    `producto`.`descuento` = descuento,
    `producto`.`actualizado` = NOW(),
    `producto`.`rebaja_inicio` = rebaja_inicio,
    `producto`.`rebaja_fin` = rebaja_fin
WHERE `producto`.`titulo` = titulo;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_asignar_categoria`(IN `titulo_producto` VARCHAR(75) CHARSET utf8mb4,
 IN `titulo_categoria` VARCHAR(75) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Asignar una categoría a un producto.'
BEGIN
INSERT INTO `categoria_producto` (id_producto,id_categoria)
VALUES (f_obtener_id_producto(titulo_producto),f_obtener_id_categoria(titulo_categoria));
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_asignar_etiqueta`(IN `titulo_producto` VARCHAR(75) CHARSET utf8mb4,
 IN `titulo_etiqueta` VARCHAR(75) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Asignar una etiqueta a un producto.'
BEGIN
INSERT INTO `etiqueta_producto` (id_producto,id_etiqueta)
VALUES (f_obtener_id_producto(titulo_producto),f_obtener_id_etiqueta(titulo_etiqueta));
END$$
DELIMITER ;
```

Figura 38. Procedimientos Almacenados para Producto 1.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_categoria_delete`(IN `titulo` VARCHAR(75) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Eliminar una categoría de productos usando la columna titulo.'
BEGIN
DELETE FROM `categoria` WHERE `categoria`.`titulo` = titulo;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_categoria_insert`(IN `titulo_padre` VARCHAR(75) CHARSET utf8mb4,
 IN `titulo` VARCHAR(75) CHARSET utf8mb4, IN `titulo_meta` VARCHAR(100) CHARSET utf8mb4, IN `slug` VARCHAR(100) CHARSET utf8mb4, IN `contenido` TEXT)
    MODIFIES SQL DATA
    COMMENT 'Insertar una nueva categoría de productos.'
BEGIN
INSERT INTO `categoria` (id_padre,titulo,titulo_meta,slug,contenido)
VALUES (f_obtener_id_categoria(titulo_padre),titulo,titulo_meta,slug,contenido);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_categoria_select`(IN `titulo` VARCHAR(75) CHARSET utf8mb4)
    READS SQL DATA
    COMMENT 'Select de una categoría de producto.'
BEGIN
SELECT id_padre, titulo, titulo_meta, slug, contenido
FROM `categoria`
WHERE `categoria`.`titulo` = titulo;
END$$
DELIMITER ;
```

Figura 39. Procedimientos Almacenados para Producto 2.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_categoria_update`(IN `titulo_viejo` VARCHAR(75) CHARSET utf8mb4,
 IN `titulo` VARCHAR(75) CHARSET utf8mb4, IN `titulo_meta` VARCHAR(100) CHARSET utf8mb4, IN `slug` VARCHAR(100) CHARSET utf8mb4, IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Modificar la información de una categoría de un producto.'
BEGIN
UPDATE `categoria`
SET
    `categoria`.`titulo` = titulo,
    `categoria`.`titulo_meta` = titulo_meta,
    `categoria`.`slug` = slug,
    `categoria`.`contenido` = contenido
WHERE `categoria`.`titulo` = titulo_viejo;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_categorias_asignadas`(IN `titulo_producto` VARCHAR(75) CHARSET utf8mb4)
    READS SQL DATA
    COMMENT 'Checar cuáles categorías están asignadas a un producto.'
BEGIN
SELECT
    id_categoria
FROM `categoria_producto`
WHERE `categoria_producto`.`id_producto` = f_obtener_id_producto(titulo_producto);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_eliminar_etiqueta_asignada`(IN `titulo_producto` VARCHAR(75) CHARSET utf8mb4,
 IN `titulo_etiqueta` VARCHAR(75) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Eliminar una etiqueta asignada a un producto.'
BEGIN
DELETE FROM `etiqueta_producto` WHERE
    `etiqueta_producto`.`id_producto` = f_obtener_id_producto(titulo_producto) AND
    `etiqueta_producto`.`id_etiqueta` = f_obtener_id_etiqueta(titulo_etiqueta);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_en_venta`(IN `titulo` VARCHAR(75) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Publicar un producto en la tienda.'
BEGIN
UPDATE `producto`
SET
    `producto`.`en_venta` = 1,
    `producto`.`publicado` = NOW()
WHERE `producto`.`titulo` = titulo;
END$$
DELIMITER ;
```

Figura 40. Procedimientos Almacenados para Producto 3.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_en_venta`(IN `titulo` VARCHAR(75) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Publicar un producto en la tienda.'
BEGIN
UPDATE `producto`
SET
    `producto`.`en_venta` = 1,
    `producto`.`publicado` = NOW()
WHERE `producto`.`titulo` = titulo;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_etiqueta_delete`(IN `titulo` VARCHAR(75) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Eliminar una categoría de productos usando la columna titulo.'
BEGIN
DELETE FROM `etiqueta` WHERE `etiqueta`.`titulo` = titulo;
END$$
DELIMITER ;
```

Figura 41. Procedimientos Almacenados para Producto 4.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_etiqueta_insert`(IN `titulo` VARCHAR(75) CHARSET utf8mb4,
 IN `titulo_meta` VARCHAR(100) CHARSET utf8mb4, IN `slug` VARCHAR(100) CHARSET utf8mb4, IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Insertar una nueva etiqueta de producto.'
BEGIN
INSERT INTO `etiqueta` (titulo,titulo_meta,slug,contenido)
VALUES (titulo,titulo_meta,slug,contenido);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_etiqueta_select`(IN `titulo` VARCHAR(75) CHARSET utf8mb4)
    READS SQL DATA
    COMMENT 'Select de una etiqueta de producto.'
BEGIN
SELECT titulo, titulo_meta, slug, contenido
FROM `etiqueta`
WHERE `etiqueta`.`titulo` = titulo;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_etiqueta_update`(IN `titulo_viejo` VARCHAR(75) CHARSET utf8mb4,
 IN `titulo` VARCHAR(75) CHARSET utf8mb4, IN `titulo_meta` VARCHAR(100) CHARSET utf8mb4, IN `slug` VARCHAR(100) CHARSET utf8mb4, IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Modificar la información de una etiqueta de producto.'
BEGIN
UPDATE `etiqueta`
SET
    `etiqueta`.`titulo` = titulo,
    `etiqueta`.`titulo_meta` = titulo_meta,
    `etiqueta`.`slug` = slug,
    `etiqueta`.`contenido` = contenido
WHERE `etiqueta`.`titulo` = titulo_viejo;
END$$
DELIMITER ;
```

Figura 42. Procedimientos Almacenados para Producto 5.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_etiquetas_asignadas`(IN `titulo_producto` VARCHAR(75) CHARSET utf8mb4)
    READS SQL DATA
    COMMENT 'Checar cuáles etiquetas están asignadas a un producto.'
BEGIN
SELECT
    id_etiqueta
FROM `etiqueta_producto`
WHERE `etiqueta_producto`.`id_producto` = f_obtener_id_etiqueta(titulo_producto);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_etiquetas_select`()
    READS SQL DATA
    COMMENT 'Select de todas las etiquetas de producto.'
BEGIN
SELECT titulo, titulo_meta, slug, contenido
FROM `etiqueta`;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_insert_taller`(IN `email` VARCHAR(50) CHARSET utf8mb4,
 IN `titulo` VARCHAR(75) CHARSET utf8mb4, IN `titulo_meta` VARCHAR(100) CHARSET utf8mb4, IN `slug` VARCHAR(100) CHARSET utf8mb4,
 IN `resumen` TINYTEXT CHARSET utf8mb4, IN `sku` VARCHAR(100) CHARSET utf8mb4, IN `precio` FLOAT, IN `contenido` TEXT CHARSET utf8mb4,
 IN `duracion` SMALLINT(6), IN `url` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Admin registra un nuevo taller en la Base de Datos.'
BEGIN
INSERT INTO producto (id_usuario,titulo,titulo_meta,slug,resumen,sku,precio,es_taller,creado,contenido)
VALUES (f_obtener_id_usuario(email),titulo,titulo_meta,slug,resumen,sku,precio,1,NOW(),contenido);
INSERT INTO taller (id_producto,duracion,url)
VALUES (f_obtener_id_producto(titulo),duracion,url);
END$$
DELIMITER ;
```

Figura 43. Procedimientos Almacenados para Producto 6.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_insert`(IN `email` VARCHAR(50) CHARSET utf8mb4,
 IN `titulo` VARCHAR(75) CHARSET utf8mb4, IN `titulo_meta` VARCHAR(100) CHARSET utf8mb4, IN `slug` VARCHAR(100) CHARSET utf8mb4,
 IN `resumen` TINYTEXT CHARSET utf8mb4, IN `sku` VARCHAR(100) CHARSET utf8mb4, IN `precio` FLOAT, IN `cantidad` SMALLINT(6), IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Admin registra un nuevo producto en la Base de Datos.'
BEGIN
INSERT INTO producto (id_usuario,titulo,titulo_meta,slug,resumen,sku,precio,cantidad,creado,contenido)
VALUES (f_obtener_id_usuario(email),titulo,titulo_meta,slug,resumen,sku,precio,cantidad,NOW(),contenido);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_meta_delete`(IN `url` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Borrar un recurso meta de un producto usando la columna url.'
BEGIN
DELETE FROM `producto_meta` WHERE `producto_meta`.`url` = url;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_meta_insert`(IN `titulo` VARCHAR(75) CHARSET utf8mb4,
 IN `tipo` TINYINT(1), IN `url` VARCHAR(100) CHARSET utf8mb4, IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Insertar un recurso meta para un producto.'
BEGIN
INSERT INTO `producto_meta` (id_producto,tipo,url,contenido)
VALUES (f_obtener_id_producto(titulo), tipo, url, contenido);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_meta_update`(IN `titulo` VARCHAR(75) CHARSET utf8mb4,
 IN `tipo` TINYINT(1), IN `url` VARCHAR(100) CHARSET utf8mb4, IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Modificar la información de un resurso meta de un producto.'
BEGIN
UPDATE `producto_meta`
SET
    `producto_meta`.`tipo` = tipo,
    `producto_meta`.`url` = url,
    `producto_meta`.`contenido` = contenido
WHERE `producto_meta`.`id_producto` = f_obtener_id_producto(titulo);
END$$
DELIMITER ;
```

Figura 44. Procedimientos Almacenados para Producto 7.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_no_en_venta`(IN `titulo` VARCHAR(75) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Remover un producto de la tienda.'
BEGIN
UPDATE `producto`
SET
    `producto`.`en_venta` = 0
WHERE `producto`.`titulo` = titulo;
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_update_taller`(IN `titulo_viejo` VARCHAR(75) CHARSET utf8mb4,
 IN `titulo` VARCHAR(75) CHARSET utf8mb4, IN `titulo_meta` VARCHAR(100) CHARSET utf8mb4, IN `slug` VARCHAR(100) CHARSET utf8mb4,
 IN `resumen` TINYTEXT CHARSET utf8mb4, IN `sku` VARCHAR(100) CHARSET utf8mb4, IN `precio` FLOAT, IN `contenido` TEXT CHARSET utf8mb4,
 IN `duracion` SMALLINT(6), IN `url` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Actualizar la información de un taller.'
BEGIN
UPDATE `producto`
SET
    `producto`.`titulo` = titulo,
    `producto`.`titulo_meta` = titulo_meta,
    `producto`.`slug` = slug,
    `producto`.`resumen` = resumen,
    `producto`.`sku` = sku,
    `producto`.`precio` = precio,
    `producto`.`actualizado` = NOW(),
    `producto`.`contenido` = contenido
WHERE `producto`.`titulo` = titulo_viejo;
UPDATE `taller`
SET
    `taller`.`duracion` = duracion,
    `taller`.`url` = url
WHERE `taller`.`id_producto` = f_obtener_id_producto(titulo);
END$$
DELIMITER ;
```

Figura 45. Procedimientos Almacenados para Producto 8.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_producto_update`(IN `titulo_viejo` VARCHAR(75) CHARSET utf8mb4,
 IN `titulo` VARCHAR(75) CHARSET utf8mb4, IN `titulo_meta` VARCHAR(100) CHARSET utf8mb4, IN `slug` VARCHAR(100) CHARSET utf8mb4,
 IN `resumen` TINYTEXT CHARSET utf8mb4, IN `sku` VARCHAR(100) CHARSET utf8mb4, IN `precio` FLOAT, IN `cantidad` SMALLINT(6),
 IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Actualizar la información de un producto.'
BEGIN
UPDATE `producto`
SET
    `producto`.`titulo` = titulo,
    `producto`.`titulo_meta` = titulo_meta,
    `producto`.`slug` = slug,
    `producto`.`resumen` = resumen,
    `producto`.`sku` = sku,
    `producto`.`precio` = precio,
    `producto`.`cantidad` = cantidad,
    `producto`.`actualizado` = NOW(),
    `producto`.`contenido` = contenido
WHERE `producto`.`titulo` = titulo_viejo;
END$$
DELIMITER ;
```

Figura 46. Procedimientos Almacenados para Producto 9.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_carrito_agregar_producto`(IN `titulo_producto` VARCHAR(75) CHARSET utf8mb4, IN `email` VARCHAR(50) CHARSET utf8mb4,
IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4, IN `cantidad` INT, IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Agregar un producto al carrito.'
BEGIN
DECLARE id_producto BIGINT;
DECLARE id_carrito BIGINT;
DECLARE sku VARCHAR(100) CHARSET utf8mb4 DEFAULT '';
DECLARE precio FLOAT;
DECLARE descuento FLOAT;
SET id_producto = f_obtener_id_producto(titulo_producto);
SET id_carrito = f_obtener_id_carrito(email, sesion, token);
SELECT `producto`.`sku` INTO sku
FROM `producto` WHERE `producto`.`id` = id_producto;
SELECT `producto`.`precio` INTO precio
FROM `producto` WHERE `producto`.`id` = id_producto;
SELECT `producto`.`descuento` INTO descuento
FROM `producto` WHERE `producto`.`id` = id_producto;
INSERT INTO `item_carrito`(id_producto,id_carrito,sku,precio,descuento,cantidad,activo,creado,contenido)
VALUES (id_producto,id_carrito,sku,precio,descuento,cantidad,1,NOW(),contenido);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_carrito_select`(IN `email` VARCHAR(50) CHARSET utf8mb4)
    READS SQL DATA
    COMMENT 'Select de la información de un carrito de un usuario.'
BEGIN
SELECT
    sesion,token,estado,creado,actualizado,contenido
FROM `carrito`
WHERE `carrito`.`id_usuario` = f_obtener_id_usuario(email);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_carrito_select_items`(IN `email` VARCHAR(50) CHARSET utf8mb4,
IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    READS SQL DATA
    COMMENT 'Select de todos los ítems en un carrito.'
BEGIN
SELECT
    id_producto,id_carrito,sku,precio,descuento,cantidad,activo,creado,actualizado,contenido
FROM `item_carrito`
WHERE `item_carrito`.`id_carrito` = f_obtener_id_carrito(email, sesion, token);
END$$
DELIMITER ;
```

Figura 47. Procedimientos Almacenados para Carrito 1.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_carrito_cambiar_cantidad_producto`(IN `titulo_producto` VARCHAR(75) CHARSET utf8mb4,
IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4, IN `cantidad_nueva` SMALLINT(6))
    MODIFIES SQL DATA
    COMMENT 'Cambiar la cantidad de un producto en un carrito.'
BEGIN
DECLARE id_producto BIGINT;
DECLARE id_carrito BIGINT;
SET id_producto = f_obtener_id_producto(titulo_producto);
SET id_carrito = f_obtener_id_carrito(email, sesion, token);
UPDATE `item_carrito`
SET
    `item_carrito`.`cantidad` = cantidad_nueva,
    `item_carrito`.`actualizado` = NOW()
WHERE
    `item_carrito`.`id_producto` = f_obtener_id_producto(titulo_producto) AND
    `item_carrito`.`id_carrito` = f_obtener_id_carrito(email, sesion, token);
END$$
DELIMITER ;

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 48. Procedimientos Almacenados para Carrito 2.

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

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_estado_completada`(IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar el estado de la orden a Completada.'
BEGIN
UPDATE `orden`
SET
    `orden`.`estado_orden` = 7,
    `orden`.`actualizado` = NOW()
WHERE
    `orden`.`id` = f_obtener_id_orden(email, sesion, token);
END$$
DELIMITER ;
```

Figura 49. Procedimientos Almacenados para Orden 1.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_estado_regresada`(IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar el estado de la orden a Regresada.'
BEGIN
UPDATE `orden`
SET
    `orden`.`estado_orden` = 6,
    `orden`.`actualizado` = NOW()
WHERE
    `orden`.`id` = f_obtener_id_orden(email, sesion, token);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_estado_enviada`(IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar el estado de la orden a Enviada.'
BEGIN
UPDATE `orden`
SET
    `orden`.`estado_orden` = 4,
    `orden`.`actualizado` = NOW()
WHERE
    `orden`.`id` = f_obtener_id_orden(email, sesion, token);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_estado_fallida`(IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar el estado de la orden a Fallida.'
BEGIN
UPDATE `orden`
SET
    `orden`.`estado_orden` = 3,
    `orden`.`actualizado` = NOW()
WHERE
    `orden`.`id` = f_obtener_id_orden(email, sesion, token);
END$$
DELIMITER ;
```

Figura 50. Procedimientos Almacenados para Orden 2.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_estado_entregada`(IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar el estado de la orden a Entregada.'
BEGIN
UPDATE `orden`
SET
    `orden`.`estado_orden` = 5,
    `orden`.`actualizado` = NOW()
WHERE
    `orden`.`id` = f_obtener_id_orden(email, sesion, token);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_insert`(IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4,
 IN `token` VARCHAR(100) CHARSET utf8mb4, IN `nombre` VARCHAR(100) CHARSET utf8mb4, IN `apellidos` VARCHAR(100) CHARSET utf8mb4,
 IN `telefono` VARCHAR(15) CHARSET utf8mb4, IN `linea_a` VARCHAR(50) CHARSET utf8mb4, IN `linea_b` VARCHAR(50) CHARSET utf8mb4,
 IN `ciudad` VARCHAR(50) CHARSET utf8mb4, IN `estado` VARCHAR(50) CHARSET utf8mb4, IN `pais` VARCHAR(50) CHARSET utf8mb4, IN `contenido` TEXT CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Crear una nueva orden, los productos se agregan posteriormente.'
BEGIN
INSERT INTO `orden` (id_usuario,sesion,token,nombre,apellidos,telefono,email,linea_a,linea_b,ciudad,estado,pais,creado,contenido)
VALUES (f_obtener_id_usuario(email),sesion,token,nombre,apellidos,telefono,email,linea_a,linea_b,ciudad,estado,pais,NOW(),contenido);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_estado_pagada`(IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar el estado de la orden a Pagada.'
BEGIN
UPDATE `orden`
SET
    `orden`.`estado_orden` = 2,
    `orden`.`actualizado` = NOW()
WHERE
    `orden`.`id` = f_obtener_id_orden(email, sesion, token);
END$$
DELIMITER ;

DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_estado_checkout`(IN `email` VARCHAR(50) CHARSET utf8mb4, IN `sesion` VARCHAR(100) CHARSET utf8mb4, IN `token` VARCHAR(100) CHARSET utf8mb4)
    MODIFIES SQL DATA
    COMMENT 'Cambiar el estado de la orden a Checkout.'
BEGIN
UPDATE `orden`
SET
    `orden`.`estado_orden` = 1,
    `orden`.`actualizado` = NOW()
WHERE
    `orden`.`id` = f_obtener_id_orden(email, sesion, token);
END$$
DELIMITER ;
```

Figura 51. Procedimientos Almacenados para Orden 3.

Resumen Base de Datos

Para resumir, se detalla a continuación como se encuentra la Base de Datos por el momento:

Base de Datos	Cantidad
Tablas	15
Funciones	7
Procedimientos Almacenados	54
Procedimientos de Usuario	14
Procedimientos de Producto	26
Procedimientos de Carrito	5
Procedimientos de Orden	9

La Base de Datos ha avanzado bien, todavía faltan algunos procesos más como la unión con la interfaz o los procedimientos de Transacción. Esto se dejará pendiente para empezar a trabajar con el diseño de la Página Web y se retomará cuando se comience a realizar la codificación de la Página Web.

ANEXOS – Informe Red IRAG

Durante el mes de Febrero, se trabajó en el proyecto de la Red IRAG; cuyo fin fue desarrollar una Base de Datos que manejará los datos de hospitales de México y analizar estos datos diariamente. A continuación se presenta el informe que se realizó.

Introducción

Antecedentes

El Centro de Investigación en Matemáticas es un Centro Público de Investigación que desarrolla proyectos innovadores de Inteligencia Artificial y soluciones a problemas complejos con manejo de información mediante desarrollos confiables y fundamentos científicos.

A raíz de la reciente pandemia de COVID-19 que ataca al mundo, se requiere estudiar los efectos en el país y contribuir en el análisis de información de este nuevo virus. Actualmente se realizan esfuerzos del Centro de Investigación en Matemáticas para utilizar técnicas y modelos de ciencia de datos e inteligencia artificial para colaborar en modelos de predicción, de riesgo y de exploración de los datos que se emiten día con día sobre la evolución de la epidemia en México.

El CIMAT, en apoyo a la secretaría de salud, requiere unificar esfuerzos para contribuir en el desarrollo de la arquitectura para la obtención de datos relacionados con el riesgo que presenta el covid-19 en el país. A raíz de ellos se propone la siguiente colaboración.

Objetivos

General.

- Análisis de archivos de ocupación hospitalaria de la RED IRAG (enfermedades respiratorias agudas graves) para crear una base de datos que permita actualizar de forma diaria la información de ocupación hospitalaria por COVID-19.

Específicos.

- Desarrollo de una base de datos para la RED IRAG de ocupación hospitalaria por COVID-19.

Herramientas

Los avances se realizaron en una Base de Datos montada en XAMPP:

- XAMPP Versión: 7.4.11
- PHP 7.4.11
- Apache 2.4.46
- MariaDB 10.4.14
- phpMyAdmin 5.0.3

Fuente de Datos

Los siguientes archivos fueron proporcionados como Fuente de Datos para realizar el proyecto de la Base de Datos de la Red IRAG, colocados en una carpeta que llamaremos Fuente-Datos:

- **202006-CLUES**. Archivo en Excel que contiene registros de los Hospitales (CLUEs) del país y sus respectivos municipios y estados. Fue transformado a un archivo *.csv* para su mejor manipulación.
- **estados_pob**. Archivo *.csv* que contiene registros de los Estados de la República Mexicana y su proyección de población para el año 2020.
- **ZM_2015_pob**. Archivo *.csv* que contiene registros de las Zonas Metropolitanas de la República Mexicana, los municipios que contienen, a cuáles Estados pertenecen y su proyección de población para el año 2020.
- **BD_Estados_historico**. Archivo *.csv* que contiene los registros diarios reportados por las CLUEs de la Red IRAG desde aproximadamente la fecha 1/04/2020 hasta la fecha 14/02/2021.
- **Archivos-Diarios**. Directorio que contiene los archivos diarios que reportan las CLUEs. Contiene dos tipos de archivos, aunque se puede presentar variaciones en los nombres:
    - **Infraestructura**. Archivo *.csv* que contiene registros reportados del día anterior sobre la ocupación y disponibilidad de los hospitales. Debe contener la fecha del día en el nombre del archivo con formato ‘yymmdd’.
    - **Pacientes**. Archivo *.csv* que contiene registros reportados del día anterior sobre los pacientes en los hospitales. Debe contener la fecha del día en el nombre del archivo con formato ‘yymmdd’.

Base de Datos

Modelo Entidad-Relación

Con base en las necesidades del proyecto de la Red IRAG, se decidió en el siguiente Modelo Entidad-Relación:

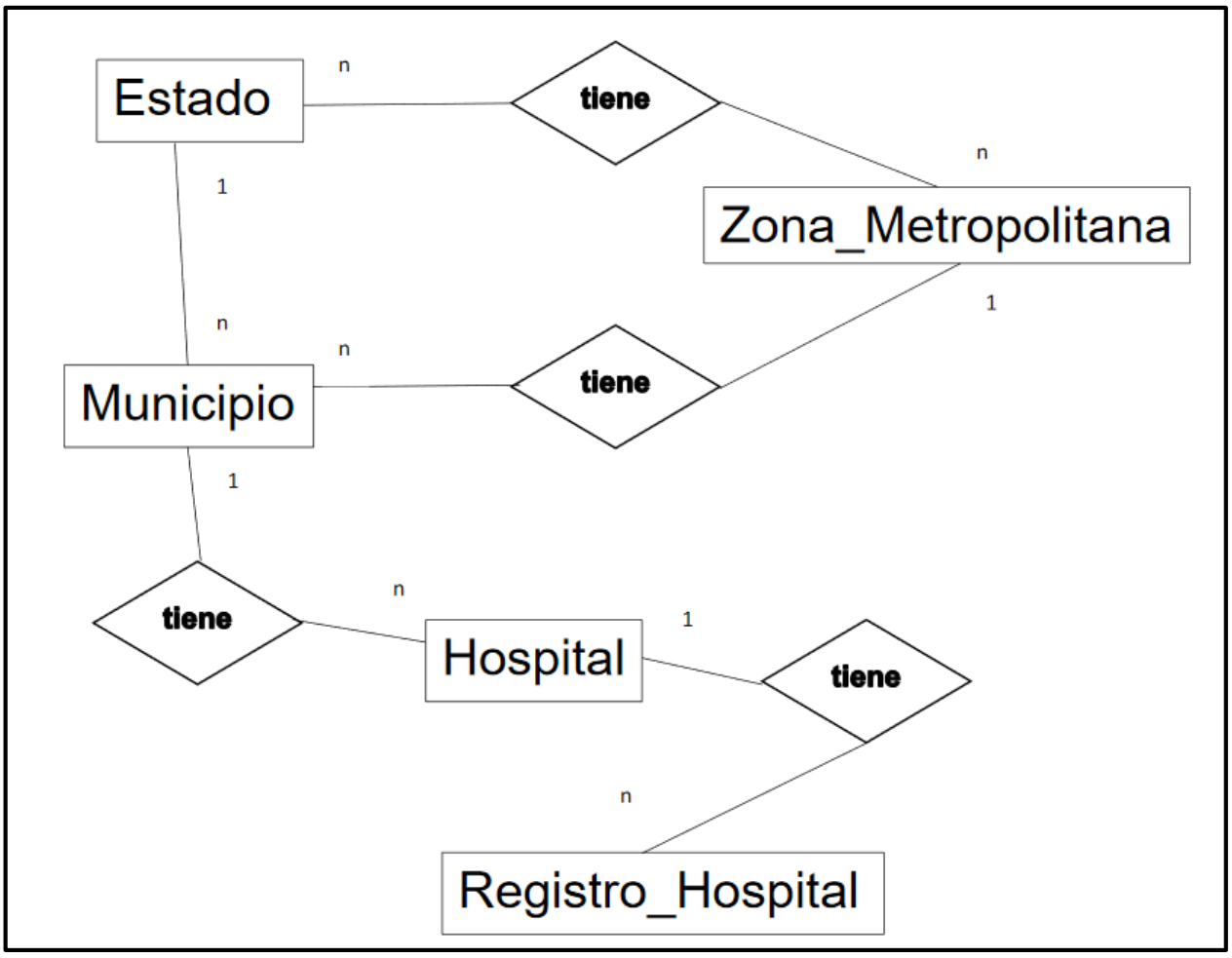


Figura 52. Modelo Entidad-Relación Red IRAG,

Modelo Relacional

Con base en el modelo anterior, se realizó el siguiente Modelo Relacional:

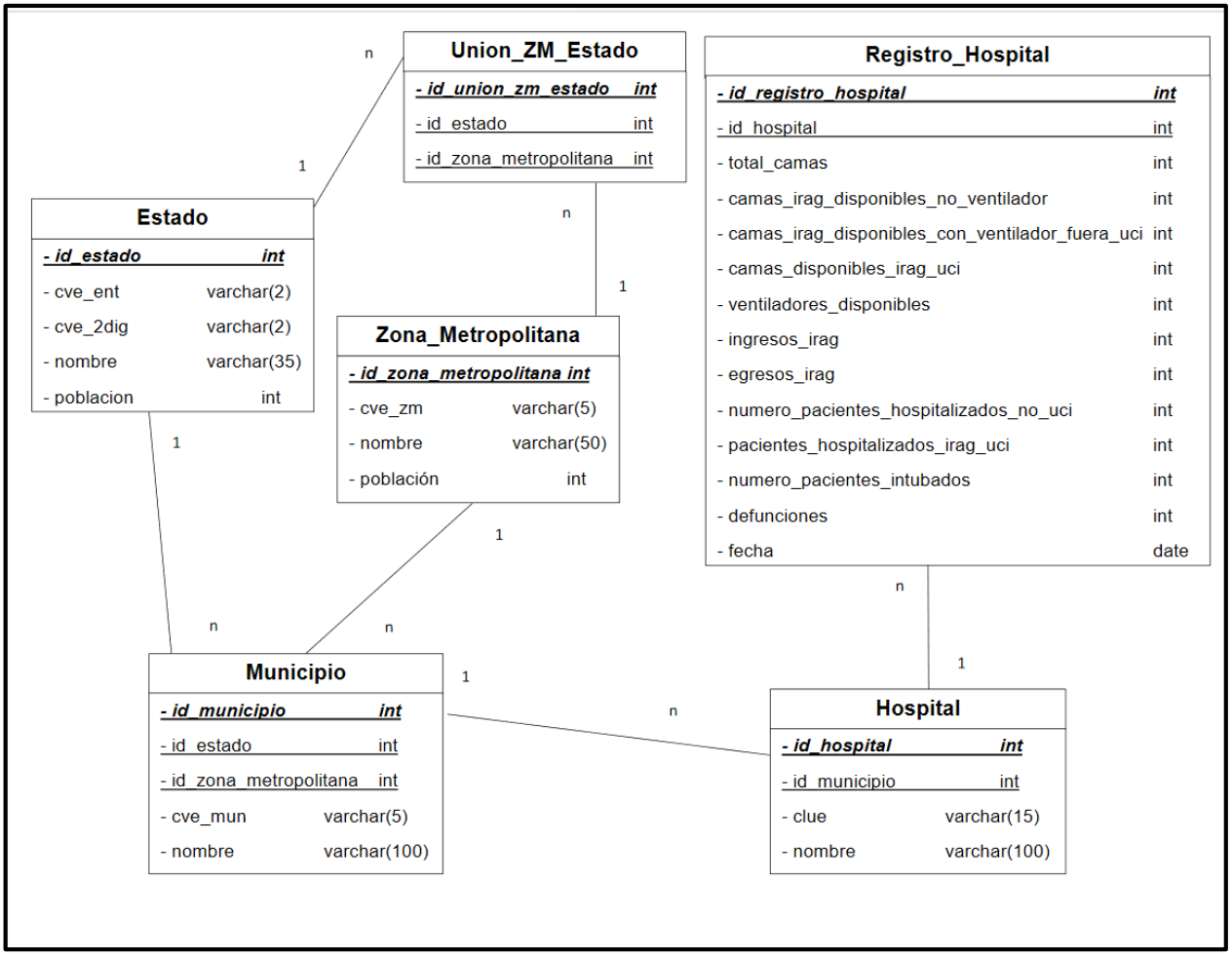


Figura 53. Modelo Relacional Red IRAG.

Las respectivas llaves de las tablas están subrayadas, y la llave primaria además se encuentra escrita en negritas. Se agregó la tabla Union_ZM_Estado por la relación *n* a *n* que tiene el modelo E-R. Los nombres de las columnas son los mismos dentro de los archivos *.csv* de *Fuente-Datos*.

Diccionario de Datos

Se presenta el siguiente Diccionario de Datos con la información de las columnas de la Base de Datos:

Nombre	Tipo	Formato	Descripción
id_estado	int	Auto_Increment	Identificador de la Tabla Estado
cve_ent	varchar(2)	'01' – '32'	Clave de la Entidad
cve_2dig	varchar(2)	'AS' – 'ZS'	Clave de la Entidad a 2 dígitos alfanuméricos

nombre	varchar(35)		Nombre completo de la Entidad
población	int		Proyección de la población para el 2020 del Estado
id_zona_metropolitana	int	Auto_Increment	Identificador de la Tabla Zona Metropolitana
cve_zm	varchar(5)	'1.01' – '32.01'	Clave de la Zona Metropolitana
nombre	varchar(50)		Nombre de la Zona Metropolitana
poblacion	int		Proyección de la población para el 2020 de la Zona Metropolitana
id_union_zm_estado	int	Auto_Increment	Identificador de la Tabla Union_ZM_estado
id_estado	int	Foreign_Key	Llave foránea para la Tabla Estado
id_zona_metropolitana	int	Foreign_Key	Llave foránea para la tabla Zona_Metropolitana
id_municipio	int	Auto_Increment	Identificador de la Tabla Municipio
id_estado	int	Foreign_Key	Llave foránea para la Tabla Estado
id_zona_metropolitana	int	Foreign_Key SÍ NULL	Llave foránea para la Tabla Zona_Metropolitana
cve_mun	varchar(5)	'01001' – '32999'	Clave del Municipio. Primeros 2 dígitos son la clave del estado
nombre	varchar(100)		Nombre del Municipio
id_hospital	int	Auto_Increment	Identificador de la Tabla Hospital
id_municipio	int	Foreign_Key SÍ NULL	Llave foránea para la Tabla Municipio
clue	varchar(15)	'ASDIF000011' – 'ZSSSA013213'	Identificador único del Hospital llamado CLUE, de 15 dígitos

nombre	varchar(100)		Nombre del Hospital
id_registro_hospital	int	Auto_Increment	Identificador de la Tabla Registro_Hospital
id_hospital	int	Foreign_Key	Llave foránea para la Tabla Hospital
total_camas	int		Total de Camas
camas_irag_disponibles_no_ventilador	int		Camas IRAG Sin Ventilador Disponibles
camas_irag_disponibles_con_ventilador_fuera_uci	int		Camas IRAG con Ventilador Fuera de UCI Disponibles
camas_disponibles_irag_uci	int		Camas IRAG en UCI Disponibles
ventiladores_disponibles	int		Ventiladores Disponibles de Respaldo
ingresos_irag	int		Ingresos IRAG
egresos_irag	int		Egresos IRAG
numero_pacientes_hospitalizados_no_uci	int		Pac Hospitalizados No UCI
pacientes_hospitalizados_irag_uci	int		Pac Hospitalizados IRAG UCI
numero_pacientes_intubados	int		Pacientes intubados
defunciones	int		Defunciones IRAG
fecha	date	'2020-04-01' – '2021-02-14'	Fecha de los registros

Script SQL

Utilizando la opción 'Exportar' de phpMyAdmin obtenemos un Script SQL de la Base de Datos de la Red IRAG con las tablas anteriormente explicadas y sus configuraciones, como también todos los registros insertados en ellas (del que se hablará en el siguiente apartado). El nombre del archivo es ***irag.sql*** y se puede volver a generar la Base de Datos desde este archivo, además que también sirve como respaldo.

Scripts PHP

Llenado de Tablas

1. **estado.php** – Script para llenar la Tabla Estados. En la Figura 3 se muestra la conexión a la Base de Datos, todos los scripts deben contener este código. Después en la Figura 4 se enseña como abrir el archivo correspondiente para el script; en este caso queremos abrir el archivo de estados de nuestra Fuente de Datos. En la Figura 6 se muestra la última parte del código, donde cerramos el archivo y cerramos la conexión con la Base de Datos; líneas importantes que todos los scripts deben contener. Finalmente, en la Figura 5 se muestra el procedimiento para leer cada línea del archivo, recuperar los datos pertinentes e insertarlos a la Base de Datos con una query SQL, después de verificar si existen o no con otra query SQL. Los scripts siguen este procedimiento: conectar a la BD -> abrir el archivo correspondiente -> realizar el procedimiento de inserción a la BD/recuperar información de la BD -> cerrar el archivo -> cerrar ka conexión con la BD.

```php
<?php
    //CONECTAR A LA BD
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = '';
    $dbdata = 'irag';

    $conn = new mysqli($dbhost,$dbuser,$dbpass,$dbdata);
    $conn->set_charset("utf8");
    if($conn->connect_error){
        die('Hubo un error al conectarse a la base de datos!!!');
    }
```

Figura 54. Script PHP Red IRAG 1.

```php
//LEER ARCHIVO
$ruta = 'Fuente-Datos/';
$myfile = fopen($ruta."estados_pob.csv", "r") or die("Unable to open file!");
```

Figura 55. Script PHP Red IRAG 2.

```php
//LEER PRIMERA LINEA
fgets($myfile);
for($i=0;$i<32;$i++){
    //SEPARAR LINEA EN ARREGLO
    $arreglo = explode(",",fgets($myfile));
    //OBTENER VALORES A INSERTAR
    $cve_ent = $arreglo[0];          //Clave numérica a 2 Dígitos
    $cve_2dig = $arreglo[1];         //Clave alfanumérica a 2 Dígitos
    $nombre = $arreglo[2];           //Nombre completo del estado
    $poblacion = $arreglo[7];        //Proyección Población 2020
    //CHECAR SI EXISTE EL REGISTRO PARA NO REPETIR
    $sql = "SELECT cve_2dig FROM estado WHERE cve_2dig = '".$cve_2dig."'";

    $result = $conn->query($sql);
    if($result->num_rows > 0){
        echo "Error: " .$cve_2dig." ya existe <br>";
        continue;
    }

    //INSERT
    $sql = "INSERT INTO estado (cve_ent, cve_2dig, nombre, poblacion)
            VALUES ('".$cve_ent."', '".$cve_2dig."', '".$nombre."', ".$poblacion.")";

    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    echo $cve_ent." ".$cve_2dig." ".$nombre." ".$poblacion."<br>";
}
```

Figura 56. Script PHP Red IRAG 3.

```php
    //CERRAR EL ARCHIVO
    fclose($myfile);

    //CERRAR LA CONEXIÓN CON LA BD
    $conn->close();
?>
```

Figura 57. Script PHP Red IRAG 4.

2. **zona_metropolitana.php** – Después de conectarse a la BD, abrimos el archivo “ZM_2015_pob.csv” y guardamos las líneas en una variable. Al recorrer estas líneas, debemos de checar todas las claves Estado que pertenecen a una misma Clave ZM, y sumar toda la población de los municipios de la ZM. Después insertamos la ZM y luego insertamos los enlaces entre ZM y Estado en la Tabla Union_ZM_Estado.

```php
//LEER ARCHIVO
$ruta = 'Fuente-Datos/';
$myfile = fopen($ruta."ZM_2015_pob.csv", "r") or die("Unable to open file!");
//LEER PRIMERA LINEA
fgets($myfile);
//GUARDAR ARCHIVO
$archivo = array(fgets($myfile));
while(!feof($myfile)){
    array_push($archivo, fgets($myfile));
}

//RECORRER ARCHIVO
for($i=0;$i<sizeof($archivo);){
    //SEPARAR LINEA EN ARREGLO
    $arreglo = explode(",",$archivo[$i]);
    //OBTENER VALORES A INSERTAR
    $cve_zm = $arreglo[0];          //CLAVE ZONA METROPOLITANA
    $nombre = $arreglo[1];          //NOMBRE ZONA METROPOLITANA
    $cve_ent = array($arreglo[2]);  //ARREGLO CLAVE ESTADOS
    $poblacion = $arreglo[7];       //POBLACIÓN ZONA METROPOLITANA PROYECCIÓN 2020
    $i++;
    //CHECAR LAS CLAVES ESTADO DE LA ZONA METROPOLITANA
    //SUMA ACUMULADA DE LA POBLACIÓN DE LOS MUNICIPIOS DE LA ZM
    do{
        $arreglo = explode(",",$archivo[$i]);
        $aux = $arreglo[0];
        if($aux==$cve_zm){
            array_push($cve_ent, $arreglo[2]);
            $poblacion+= $arreglo[7];       //POBLACIÓN ZONA METROPOLITANA PROYECCIÓN 2020
            $i++;
        }
    }while($aux==$cve_zm && $i<sizeof($archivo));
    //ELIMINAR CLAVES REPETIDAS E ÍNDICES VACÍOS
    $cve_ent = array_values(array_unique($cve_ent));
```

Figura 58. Script PHP Red IRAG 5.

```php
    //CHECAR SI EXISTE EL REGISTRO PARA NO REPETIR
    $sql = "SELECT cve_zm FROM zona_metropolitana WHERE cve_zm = '".$cve_zm."'";

    $result = $conn->query($sql);
    if($result->num_rows > 0){
        echo "Error: " .$cve_zm." ya existe <br>";
        continue;
    }

    //INSERT
    $sql = "INSERT INTO zona_metropolitana (cve_zm, nombre, poblacion)
            VALUES ('".$cve_zm."', '".$nombre."', ".$poblacion.")";

    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    //INSERT en Tabla Union
    //echo var_dump($cve_ent)." <br>";
    for($j=0;$j<sizeof($cve_ent);$j++){
        //CVE_ENT NO TIENE 0 A LA IZQUIERDA EN EL ARCHIVO, POR LO TANTO CVE_ENT = ID_ESTADO
        //CHECAR SI EXISTE EL REGISTRO PARA NO REPETIR
        $sql = 'SELECT cve_zm, cve_ent FROM union_zm_estado WHERE cve_zm = "'.$cve_zm.'" AND cve_ent = "'.$cve_ent.'" ';

        $result = $conn->query($sql);
        if($result->num_rows > 0){
            echo "Error: " .$cve_zm." | ".$cve_ent." ya existe <br>";
            continue;
        }

        $sql = "INSERT INTO union_zm_estado (id_estado, id_zona_metropolitana)
                VALUES (".$cve_ent[$j].",
                        (SELECT id_zona_metropolitana FROM zona_metropolitana WHERE cve_zm='".$cve_zm."'))";
        //echo $cve_ent[$j]." <br>";
        if ($conn->query($sql) === TRUE) {
            echo "New record created successfully";
        } else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }
    }

}
```

Figura 59. Script PHP Red IRAG 6.

3. **municipio.php** – Primero abrimos el archivo “202006-CLUES.csv” para obtener la información de los Munnicipios y la insertamos en la BD. Luego abrimos el archivo “ZM_2015_pob.csv” para obtener la información de los Municipios que pertenecen a las Zonas Metropolitanas y crear el enlace entre las tablas.

```php
//LEER ARCHIVO
$ruta = 'Fuente-Datos/';
$myfile = fopen($ruta."202006-CLUES.csv", "r") or die("Unable to open file!");
//LEER PRIMERA LINEA
fgets($myfile);
while(!feof($myfile)){
    //SEPARAR LINEA EN ARREGLO
    $arreglo = explode(",",fgets($myfile));
    //OBTENER VALORES A INSERTAR
    $cve_mun = $arreglo[5];         //CLAVE MUNICIPIO
    $nombre_mun = $arreglo[4];      //NOMBRE MUNICIPIO
    $cve_ent = $arreglo[3];         //CLAVE ESTADO
    //AGREGAR 0 A LA IZQUIERDA EN CLAVES
    $cve_mun = strlen($cve_mun)==1?"0".$cve_mun:$cve_mun;
    $cve_mun = strlen($cve_mun)==2?"0".$cve_mun:$cve_mun;
    $cve_ent = strlen($cve_ent)==1?"0".$cve_ent:$cve_ent;
    //CONCATENAR CLAVE ENTIDAD A CLAVE MUNICIPIO
    $cve_mun = $cve_ent.$cve_mun;
    //CHECAR SI EXISTE EL REGISTRO PARA NO REPETIR
    $sql = "SELECT cve_mun FROM municipio WHERE cve_mun = '".$cve_mun."'";

    $result = $conn->query($sql);
    if($result->num_rows > 0){
        echo "Error: " .$cve_mun." ya existe <br>";
        continue;
    }

    //INSERT
    $sql = "INSERT INTO municipio (id_estado, cve_mun, nombre)
            VALUES ((SELECT id_estado FROM estado WHERE cve_ent='".$cve_ent."'), '".$cve_mun."', '".$nombre_mun."')";

    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    //echo $cve_ent." ".$cve_mun." ".$nombre_mun." <br>";
}

//CERRAR EL ARCHIVO
fclose($myfile);
```

Figura 60. Script PHP Red IRAG 7.

```php
//AGREGAR MUNICIPIOS A ZONA METROPOLITANAS
//LEER ARCHIVO
$ruta = 'Fuente-Datos/';
$myfile = fopen($ruta."ZM_2015_pob.csv", "r") or die("Unable to open file!");
//LEER PRIMERA LINEA
fgets($myfile);
while(!feof($myfile)){
    //SEPARAR LINEA EN ARREGLO
    $arreglo = explode(",",fgets($myfile));
    //OBTENER VALORES A INSERTAR
    $cve_zm = $arreglo[0];      //CLAVE ZONA METROPOLITANA
    $cve_mun = $arreglo[4];     //CLAVE MUNICIPIO DEL ARCHIVO ZONAS METROPOLITANAS
    //AGREGAR 0 A LA IZQUIERDA EN CLAVE MUNICIPIO
    $cve_mun = strlen($cve_mun)==4?"0".$cve_mun:$cve_mun;
    //CHECAR SI EXISTE EL REGISTRO PARA NO REPETIR
    $sql = "SELECT cve_mun FROM municipio
            WHERE id_zona_metropolitana=(SELECT id_zona_metropolitana FROM zona_metropolitana WHERE cve_zm='".$cve_zm."')
                AND cve_mun = '".$cve_mun."'";

    $result = $conn->query($sql);
    if($result->num_rows > 0){
        echo "Error: " .$cve_mun." con ".$cve_zm." ya existe <br>";
        continue;
    }

    //ACTUALIZAR CLAVES ZM DE LOS MUNICIPIOS, NULL PREDETERMINADO
    $sql = "UPDATE municipio SET id_zona_metropolitana=(SELECT id_zona_metropolitana FROM zona_metropolitana WHERE cve_zm='".$cve_zm."')
            WHERE cve_mun='".$cve_mun."'";

    if ($conn->query($sql) === TRUE) {
        echo "Record updated successfully";
    } else {
        echo "Error updating record: " . $conn->error;
    }

    //echo $cve_zm." ".$cve_mun." <br>";
}

//CERRAR EL ARCHIVO
fclose($myfile);
```

Figura 61. Script PHP Red IRAG 8.

4. **hospital.php** – Abrimos el archivo “202006-CLUES.csv” para obtener la información de las CLUEs e insertamos los datos a la Tabla Hospital.

```php
//LEER ARCHIVO
$ruta = 'Fuente-Datos/';
$myfile = fopen($ruta."202006-CLUES.csv", "r") or die("Unable to open file!");
//LEER PRIMERA LINEA
fgets($myfile);
while(!feof($myfile)){
    //SEPARAR LINEA EN ARREGLO
    $arreglo = explode(",",fgets($myfile));
    //OBTENER VALORES A INSERTAR
    $cve_mun = $arreglo[5];         //CLAVE MUNICIPIO
    $nombre_hospital = $arreglo[6]; //NOMBRE HOSPITAL
    $clue = $arreglo[1];            //CLUE DEL HOSPITAL
    $cve_ent = $arreglo[3];         //CLAVE ESTADO
    //AGREGAR 0 A LA IZQUIERDA EN CLAVES
    $cve_mun = strlen($cve_mun)==1?"0".$cve_mun:$cve_mun;
    $cve_mun = strlen($cve_mun)==2?"0".$cve_mun:$cve_mun;
    $cve_ent = strlen($cve_ent)==1?"0".$cve_ent:$cve_ent;
    //CONCATENAR CLAVE ENTIDAD A CLAVE MUNICIPIO
    $cve_mun = $cve_ent.$cve_mun;
    //CHECAR SI EXISTE EL REGISTRO PARA NO REPETIR
    $sql = "SELECT clue FROM hospital WHERE clue = '".$clue."'";

    $result = $conn->query($sql);
    if($result->num_rows > 0){
        echo "Error: " .$clue." ya existe <br>";
        continue;
    }

    //INSERT
    $sql = "INSERT INTO hospital (id_municipio, clue, nombre)
            VALUES ((SELECT id_municipio FROM municipio WHERE cve_mun='".$cve_mun."'), '".$clue."', '".$nombre_hospital."')";

    if ($conn->query($sql) === TRUE) {
        //echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    //echo $cve_mun." ".$clue." ".$nombre_hospital." <br>";
}

//CERRAR EL ARCHIVO
fclose($myfile);
```

Figura 62. Script PHP Red IRAG 9.

5. **registro_hospital_bd_historico.php** – Al inicio del script debemos agregar unas líneas para aumentar el tamaño de memoria y el tiempo de ejecución (Aún con estas líneas el archivo histórico es tan grande que se necesita correrlo más de 1 vez. También es el que más tarda). Después abrimos el archivo histórico y recuperamos todos los datos pertinentes. Si un registro tiene “NA” en sus columnas de valores, los ignoramos ya que esa CLUEs no reportaron en ese momento. Si los valores de la clave municipio y estado son desconocidos - “NA” – entonces tenemos que insertar las nuevas CLUEs a la Base de Datos, aunque no conocemos a cuál Municipio pertenecen. Finalmente realizamos al inserción de datos a la Tabla Registro_Hospital.

```php
<?php
    //AUMENTAR TAMAÑO DE MEMORIA '-1'=ILIMITADO
    ini_set('memory_limit', '-1');
    //SET TIEMPO DE EJECUCCION DEL SCRIPT A 0=INLIMITADO
    set_time_limit(0);
```

Figura 63. Script PHP Red IRAG 10.

```php
//LEER ARCHIVO
$ruta = 'Fuente-Datos/';
$myfile = fopen($ruta."BD_Estados_historico.csv", "r") or die("Unable to open file!");
//LEER PRIMERA LINEA
fgets($myfile);
while(!feof($myfile)){
    //SEPARAR LINEA EN ARREGLO
    $arreglo = explode(",",fgets($myfile));
    //OBTENER VALORES A INSERTAR
    $clue = $arreglo[0];
    $cve_mun = $arreglo[4];
    $total_camas = $arreglo[5];
    $camas_irag_disponibles_no_ventilador = $arreglo[6];
    $camas_irag_disponibles_con_ventilador_fuera_uci = $arreglo[7];
    $camas_disponibles_irag_uci = $arreglo[8];
    $ventiladores_disponibles = $arreglo[9];
    $ingresos_irag = $arreglo[10];
    $egresos_irag = $arreglo[11];
    $numero_pacientes_hospitalizados_no_uci = $arreglo[12];
    $pacientes_hospitalizados_irag_uci = $arreglo[13];
    $numero_pacientes_intubados = $arreglo[14];
    $defunciones = $arreglo[15];
    $fecha = $arreglo[16];
    //QUITAR COMILLAS A CLUE
    $clue = substr($clue,1,11);
    //CASOS NA
    //ESTOS SON LAS CLUES QUE NO REPORTAN AL DÍA. POR EL MOMENTO LAS SALTAREMOS PORQUE ES EL ARCHIVO HISTÓRICO
    if($fecha == "NA" || $total_camas == "NA" || $ingresos_irag == "NA") continue;
```

Figura 64. Script PHP Red IRAG 11.

```php
//SI LA CLAVE MUNICIPIO ES NA SIGNIFICA QUE LA CLUE NO ESTÁ EN LA BD
if($cve_mun == "NA") {
    //INSERTAR NUEVA CLUE CON ID_MUNICIPIO = NULL Y NOMBRE VACÍO
    //CHECAR SI EXISTE EL REGISTRO PARA NO REPETIR
    $sql = "SELECT id_hospital FROM hospital WHERE clue='".$clue."'";

    $result = $conn->query($sql);
    if($result->num_rows == 0){
        //INSERT
        $sql = "INSERT INTO hospital (id_municipio, clue, nombre)
                VALUES (NULL, '".$clue."', '')";

        if ($conn->query($sql) === TRUE) {
            echo "New record created successfully: ".$clue." <br>";
        } else {
            echo "Error: " . $sql . "<br>" . $conn->error. "<br>";
        }
    }
}
```

Figura 65. Script PHP Red IRAG 12.

```php
    //CHECAR SI EXISTE EL REGISTRO PARA NO REPETIR
    $sql = "SELECT id_registro_hospital FROM registro_hospital WHERE fecha='".$fecha."'
            AND id_hospital = (SELECT id_hospital FROM hospital WHERE clue='".$clue."')";

    $result = $conn->query($sql);
    if($result->num_rows > 0){
        //echo "Error: " .$clue." en ".$fecha." ya existe <br>";
        continue;
    }

    //INSERT
    $sql = "INSERT INTO registro_hospital (id_hospital, total_camas, camas_irag_disponibles_no_ventilador,
                                           camas_irag_disponibles_con_ventilador_fuera_uci, camas_disponibles_irag_uci,
                                           ventiladores_disponibles, ingresos_irag, egresos_irag, numero_pacientes_hospitalizados_no_uci,
                                           pacientes_hospitalizados_irag_uci,numero_pacientes_intubados,defunciones,fecha)
            VALUES ((SELECT id_hospital FROM hospital WHERE clue='".$clue."'), ".$total_camas.",
                    ".$camas_irag_disponibles_no_ventilador.", ".$camas_irag_disponibles_con_ventilador_fuera_uci.",
                    ".$camas_disponibles_irag_uci.", ".$ventiladores_disponibles.", ".$ingresos_irag.", ".$egresos_irag.",
                    ".$numero_pacientes_hospitalizados_no_uci.", ".$pacientes_hospitalizados_irag_uci.",
                    ".$numero_pacientes_intubados.", ".$defunciones.", '".$fecha."')";

    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully: ".$clue." en ".$fecha." <br>";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error. "<br>";
    }
}

//CERRAR EL ARCHIVO
fclose($myfile);
```

Figura 66. Script PHP Red IRAG 13.

Formato AMA

Los datos de ocupación se utilizan con los siguientes nombres para las fórmulas de los archivos de formato AMA:

Pac Hospitalizados No UCI -> camasOcupadasNoUCI

Pac Hospitalizados IRAG UCI -> camasOcupadasUCI

Pacientes intubados -> camasOcupadasUCI_NoUCIVent

Defunciones -> defunciones

Las fórmulas para Formato AMA son las siguientes:

camasOcupadasHG = camasOcupadasNoUCI - (camasOcupadasUCI_NoUCIVent – camasOcupadasUCI)

camasOcupadasNoUCIVentiladores = camasOcupadasUCI_NoUCIVent – camasOcupadasUCI

camasOcupadasUCI = camasOcupadasUCI

camasOcupadasUCI_NoUCIVent = camasOcupadasUCI_NoUCIVent

Los directorios para formato AMA son los siguientes:

estados

cveINEGINum_DinHospitales.csv (por ejemplo: 01_DinHospitales.csv)

NOM

cveINEGILetras_DinHospitales.csv (por ejemplo: AS_DinHospitales.csv)

zonas_metropolitanas

cveINEGI_DinHospitales.csv (por ejemplo: 1-01_DinHospitales.csv)

1. **ama_estados.php** – Este script es para generar los archivos de formato AMA para Estados. Primero se debe crear el directorio, si no existe (el formato AMA pide archivos con la clave en el nombre y con la nomenclatura en el nombre). Luego obtenemos todas las claves de los estados para usarlas en los nombres de los archivos. Después, por cada Estado, obtenemos las fechas distintas donde hubo registros diarios de sus hospitales. Con todas las fechas por Estado, podemos recuperar todos los registros pertinentes y escribir la información en el archivo AMA.

```php
//CREAR DIRECTORIOS FORMATO AMA
$directorio_codigo = 'Formato-AMA/datos_'.(new DateTime("now"))->format('Ymd').'/estados';
$directorio_nom = $directorio_codigo.'/NOM';
if(!is_dir($directorio_codigo)){
    mkdir($directorio_codigo, 0777, true);
    mkdir($directorio_nom, 0777, true);
    echo "Directorios creados <br>";
} else {
    //die("La carpeta ya existe");
}
```

Figura 67. Script PHP Red IRAG 14.

```php
//OBTENER TODAS LAS CLAVES DE LOS ESTADOS
$cve_ent = array();
$cve_2dig = array();
$sql = "SELECT cve_ent, cve_2dig FROM estado ORDER BY id_estado ASC";

$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        array_push($cve_ent, $row["cve_ent"]);
        array_push($cve_2dig, $row["cve_2dig"]);
    }
} else {
    echo "0 results";
}
```

Figura 68. Script PHP Red IRAG 15.

```php
for($i=0; $i<sizeof($cve_ent); $i++){
    //CREAR ARCHIVO AMA PARA ESTADOS
    $archivo_codigo = fopen($directorio_codigo."/".$cve_ent[$i]."_DinHospitales.csv", "w") or die("Unable to open file 1 !");
    $archivo_nom = fopen($directorio_nom."/".$cve_2dig[$i]."_DinHospitales.csv", "w") or die("Unable to open file 2 !");
    //ESCRIBIR LÍNEA EN ARCHIVOS
    fwrite($archivo_codigo, "fecha,camasOcupadasHG,camasOcupadasNoUCIVentiladores,camasOcupadasUCI,camasOcupadasUCI_NoUCIVent\r\n");
    fwrite($archivo_nom, "fecha,camasOcupadasHG,camasOcupadasNoUCIVentiladores,camasOcupadasUCI,camasOcupadasUCI_NoUCIVent\r\n");
    //OBTENER TODAS LAS FECHAS DE UN ESTADO
    $fechas = array();
    $sql = "SELECT DISTINCT fecha
                FROM registro_hospital WHERE id_hospital IN (SELECT id_hospital FROM hospital WHERE id_municipio IN
                (SELECT id_municipio FROM municipio WHERE id_estado=(SELECT id_estado FROM estado WHERE cve_ent='".$cve_ent[$i]."'))) ORDER BY fecha ASC";
    $result = $conn->query($sql);
    if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
            array_push($fechas, $row["fecha"]);
        }
    } else {
        echo "0 results";
    }
```

Figura 69. Script PHP Red IRAG 16.

```php
    for($j=0;$j<sizeof($fechas);$j++){
        //SELECT
        $sql = "SELECT fecha,
                    SUM(numero_pacientes_hospitalizados_no_uci)-(SUM(numero_pacientes_intubados)-SUM(pacientes_hospitalizados_irag_uci)) AS camasOcupadasHG,
                    SUM(numero_pacientes_intubados)-SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadasNpUCIVentiladores,
                    SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadasUCI,
                    SUM(numero_pacientes_intubados) AS camasOcupadasUCI_NoUCIVent
                    FROM `registro_hospital` WHERE fecha='".$fechas[$j]."' AND id_hospital IN
                        (SELECT id_hospital FROM hospital WHERE id_municipio IN (SELECT id_municipio FROM municipio WHERE id_estado=(SELECT id_estado FROM estado WHERE cve_ent='".$cve_ent[$i]."')))";
        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
            while($row = $result->fetch_assoc()) {
                $linea = $row["fecha"].','.$row["camasOcupadasHG"].','.$row["camasOcupadasNpUCIVentiladores"].','.$row["camasOcupadasUCI"].','.$row["camasOcupadasUCI_NoUCIVent"];
                fwrite($archivo_codigo, $linea."\r\n");
                fwrite($archivo_nom, $linea."\r\n");
            }
        } else {
            echo "0 results";
        }
    }
    //CERRAR ARCHIVOS
    fclose($archivo_codigo);
    fclose($archivo_nom);
}
```

Figura 70. Script PHP Red IRAG 17.

2. **ama_zm.php** – El procedimiento para generar los archivos AMA para Zonas Metropolitanas es casi el mismo. Crear el Directorio -> Obtener las Claves -> Recorrer Fechas -> Obtener la información -> Escribirla en los archivos. En este caso, en vez de checar las fechas existentes de los registros; se recorrerá todas las fechas desde el 01/04/2020 hasta la actual, y si no hay registros para una fecha en específico se llenan los valores con 0, procedimiento que usaremos para los archivos de Tendencias.

```php
//CREAR DIRECTORIOS FORMATO AMA
$directorio = 'Formato-AMA/datos_'.(new DateTime("now"))->format('Ymd').'/zonas_metropolitanas';
if(!is_dir($directorio)){
    mkdir($directorio, 0777, true);
    echo "Directorio creado <br>";
} else {
    //die("La carpeta ya existe");
}

//OBTENER TODAS LAS CLAVES DE LAS ZONAS METROPOLITANAS
$cve_zm = array();
$sql = "SELECT cve_zm FROM zona_metropolitana ORDER BY id_zona_metropolitana ASC";

$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        array_push($cve_zm, $row["cve_zm"]);
    }
} else {
    echo "0 results cve_zm <br>";
}
```

Figura 71. Script PHP Red IRAG 18.

```php
for($i=0; $i<sizeof($cve_zm); $i++){
    //SEPARAR CLAVE ZM PARA NOMBRE ARCHIVO
    $aux = explode(".",$cve_zm[$i]);
    //CREAR ARCHIVO AMA PARA ZONAS METROPOLITANAS
    $archivo = fopen($directorio."/".$aux[0]."-".$aux[1]."_DinHospitales.csv", "w") or die("Unable to open file!");
    //ESCRIBIR LÍNEA EN ARCHIVOS
    fwrite($archivo, "fecha,camasOcupadasHG,camasOcupadasNoUCIVentiladores,camasOcupadasUCI,camasOcupadasUCI_NoUCIVent\r\n");
    $date = new DateTime('2020-04-01');
    while($date <= new DateTime("now")){
        $fecha = $date->format('Y-m-d');
        $sql = "SELECT fecha,
                    SUM(numero_pacientes_hospitalizados_no_uci)-(SUM(numero_pacientes_intubados)-SUM(pacientes_hospitalizados_irag_uci)) AS camasOcupadasHG,
                    SUM(numero_pacientes_intubados)-SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadasNoUCIVentiladores,
                    SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadasUCI,
                    SUM(numero_pacientes_intubados) AS camasOcupadasUCI_NoUCIVent
                    FROM `registro_hospital` WHERE fecha='".$fecha."' AND id_hospital IN
                        (SELECT id_hospital FROM hospital WHERE id_municipio IN
                        (SELECT id_municipio FROM municipio WHERE id_zona_metropolitana=(SELECT id_zona_metropolitana FROM zona_metropolitana WHERE cve_zm='".$cve_zm[$i]."')))";
        $result = $conn->query($sql);

        if ($result->num_rows > 0) {
            while($row = $result->fetch_assoc()) {
                if(empty($row["fecha"])){
                    $linea = $fecha.',0,0,0,0';
                    fwrite($archivo, $linea."\r\n");
                } else {
                    $linea = $row["fecha"].','.$row["camasOcupadasHG"].','.$row["camasOcupadasNoUCIVentiladores"].','.$row["camasOcupadasUCI"].','.$row["camasOcupadasUCI_NoUCIVent"];
                    fwrite($archivo, $linea."\r\n");
                }
            }
        } else {
            $linea = $fecha.',0,0,0,0';
            fwrite($archivo, $linea."\r\n");
        }
        $date = $date->add(new DateInterval('P1D'));
    }

    //CERRAR ARCHIVOS
    fclose($archivo);
}
```

Figura 72. Script PHP Red IRAG 19.

Formato Tendencias

Los datos de ocupación se utilizan con los siguientes nombres para las fórmulas de los archivos de formato Tendencias:

Pac Hospitalizados No UCI -> camasOcupadasNoUCI

Pac Hospitalizados IRAG UCI -> camasOcupadasUCI

Pacientes intubados -> camasOcupadasUCI_NoUCIVent

Defunciones -> defunciones

Las fórmulas para Formato Tendencias son las siguientes:

camasOcupadas = camasOcupadasNoUCI + camasOcupadasUCI

*camasOcupadas100K = camasOcupadas/población*100000*

camasOcupadasHG = camasOcupadasNoUCI - (camasOcupadasUCI_NoUCIVent – camasOcupadasUCI)

camasOcupadasVentiladores = camasOcupadasUCI_NoUCIVent

defunciones = defunciones

*defunciones100K = defunciones/población*100000*

camasDisponiblesTotales = Total de Camas

disponiblesHG = Camas IRAG Sin Ventilador Disponibles

disponiblesVentiladores = Camas IRAG con Ventilador Fuera de UCI Disponibles + Camas IRAG en UCI Disponibles

camasDisponiblesVentiladorNoUCI = CamasIRAGDisponiblesConVentiladorFueraUCI

camasDisponiblesUCI = CamasDisponiblesIRAGUCI

camasOcupadasUCI = PacientesHospitalizadosIRAGUCI

Se generan dos archivos, con la información desde el 1 de Abril del 2020 a la fecha actual por estado (incluye nivel nacional clave 33) o bien ZM.

Agregados_HospitalizacionZM_AAMMDD.csv : a nivel ZM

Agregados_Hospitalizacion_AAMMDD.csv : a nivel Estatal

1. **tendencias_estados** – Primero debemos definir la zona horaria a “America/Mexico_City” que es la que nos corresponde, para que el manejo de fechas sea correcto respecto a la fecha del día actual. Luego hay que crear un Directorio para los archivos de Tendencias, si no existe. Después hay que obtener las claves de los estados y crear el archivo de Tendencias. Luego hay que recorrer todas las fechas desde el 01/04/2020 hasta el día de hoy; y por cada fecha realizar la query indicada para obtener la información necesitada y escribirla en el archivo de Tendencias. Hay que manejar la Clave Nacional como su caso único y distinto

```php
// DEFINIR ZONA HORARIA
date_default_timezone_set('America/Mexico_City');
```

Figura 73. Script PHP Red IRAG 20.

```php
//CREAR DIRECTORIOS FORMATO TENDENCIAS
$directorio = 'ReporteTendencias';
if(!is_dir($directorio)){
    mkdir($directorio, 0777, true);
    echo "Directorio creado <br>";
} else {
    //die("La carpeta ya existe");
}

//OBTENER TODAS LAS CLAVES DE LOS ESTADOS
$cve_ent = array();
$nombres_ent = array();
$poblacion_nacional = 0;
$sql = "SELECT cve_ent,nombre,poblacion FROM estado ORDER BY id_estado ASC";

$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        array_push($cve_ent, $row["cve_ent"]);
        array_push($nombres_ent, $row["nombre"]);
        $poblacion_nacional+=intval($row["poblacion"]);
    }
} else {
    echo "0 results";
}
//AGREGAR CLAVE NACIONAL
array_push($cve_ent, "33");
array_push($nombres_ent, "NACIONAL");

//CREAR ARCHIVO TENDENCIAS PARA ESTADOS
$archivo = fopen($directorio."/Agregados_Hospitalizacion_".(new DateTime("now"))->format('ymd').".csv", "w") or die("Unable to open file!");

//ESCRIBIR LÍNEA EN ARCHIVOS
$primera_linea = "date,cve,state,camasOcupadas,camasOcupadas100K,camasOcupadasHG,camasOcupadasVentiladores,defunciones,"
                ."defunciones100K,camasDisponiblesTotales,disponiblesHG,disponiblesVentiladores,"
                ."camasDisponiblesVentiladorNoUCI,camasDisponiblesUCI,camasOcupadasUCI";
fwrite($archivo, $primera_linea."\r\n");
```

Figura 74. Script PHP Red IRAG 21.

```php
for($i=0; $i<sizeof($cve_ent); $i++){
    $date = new DateTime('2020-04-01');
    while($date <= new DateTime("now")){
        $fecha = $date->format('Y-m-d');
        //SELECT ESTADO
        if($cve_ent[$i]!='33'){
            $sql = "SELECT fecha,poblacion,
                    SUM(numero_pacientes_hospitalizados_no_uci)+SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadas,
                    (SUM(numero_pacientes_hospitalizados_no_uci)+SUM(pacientes_hospitalizados_irag_uci))*100000/poblacion AS camasOcupadas100K,
                    SUM(numero_pacientes_hospitalizados_no_uci)-(SUM(numero_pacientes_intubados)-SUM(pacientes_hospitalizados_irag_uci)) AS camasOcupadasHG,
                    SUM(numero_pacientes_intubados) AS camasOcupadasVentiladores,
                    SUM(defunciones) AS defunciones,
                    SUM(defunciones)*100000/poblacion AS defunciones100K,
                    SUM(total_camas) AS camasDisponiblesTotales,
                    SUM(camas_irag_disponibles_no_ventilador) AS disponiblesHG,
                    SUM(camas_irag_disponibles_con_ventilador_fuera_uci)+SUM(camas_disponibles_irag_uci) AS disponiblesVentiladores,
                    SUM(camas_irag_disponibles_con_ventilador_fuera_uci) AS camasDisponiblesVentiladorNoUCI,
                    SUM(camas_disponibles_irag_uci) AS camasDisponiblesUCI,
                    SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadasUCI
                    FROM `registro_hospital`,estado WHERE fecha='".$fecha."' AND cve_ent='".$cve_ent[$i]."' AND id_hospital IN
                        (SELECT id_hospital FROM hospital WHERE id_municipio IN
                        (SELECT id_municipio FROM municipio WHERE id_estado =
                        (SELECT id_estado FROM estado WHERE cve_ent='".$cve_ent[$i]."')))";
        } else{     //SELECT CLAVE NACIONAL
            $sql = "SELECT fecha,
                        SUM(numero_pacientes_hospitalizados_no_uci)+SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadas,
                        SUM(numero_pacientes_hospitalizados_no_uci)-(SUM(numero_pacientes_intubados)-SUM(pacientes_hospitalizados_irag_uci)) AS camasOcupadasHG,
                        SUM(numero_pacientes_intubados) AS camasOcupadasVentiladores,
                        SUM(defunciones) AS defunciones,
                        SUM(total_camas) AS camasDisponiblesTotales,
                        SUM(camas_irag_disponibles_no_ventilador) AS disponiblesHG,
                        SUM(camas_irag_disponibles_con_ventilador_fuera_uci)+SUM(camas_disponibles_irag_uci) AS disponiblesVentiladores,
                        SUM(camas_irag_disponibles_con_ventilador_fuera_uci) AS camasDisponiblesVentiladorNoUCI,
                        SUM(camas_disponibles_irag_uci) AS camasDisponiblesUCI,
                        SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadasUCI
                        FROM `registro_hospital` WHERE fecha='".$fecha."'";
        }
        $result = $conn->query($sql);
```

Figura 75. Script PHP Red IRAG 22.

```php
        if ($result->num_rows > 0) {
            while($row = $result->fetch_assoc()) {
                if(empty($row["fecha"])){
                    $linea = $fecha.','.$cve_ent[$i].','.$nombres_ent[$i].',0,0,0,0,0,0,0,0,0,0,0,0';
                    fwrite($archivo, $linea."\r\n");
                } else if($cve_ent[$i]!='33'){
                    $camasOcupadas100K = intval($row["camasOcupadas"])*100000/intval($row["poblacion"]);
                    $defunciones100K = intval($row["defunciones"])*100000/intval($row["poblacion"]);
                    $linea = $row["fecha"].','.$cve_ent[$i].','.strtoupper($nombres_ent[$i]).','.$row["camasOcupadas"].','.$camasOcupadas100K.','.$row["camasOcupadasHG"].','
                            .$row["camasOcupadasVentiladores"].','.$row["defunciones"].','.$defunciones100K.','.$row["camasDisponiblesTotales"].','
                            .$row["disponiblesHG"].','.$row["disponiblesVentiladores"].','.$row["camasDisponiblesVentiladorNoUCI"].','.$row["camasDisponiblesUCI"].','.$row["camasOcupadasUCI"];
                    fwrite($archivo, $linea."\r\n");
                } else {
                    $camasOcupadas100K = intval($row["camasOcupadas"])*100000/$poblacion_nacional;
                    $defunciones100K = intval($row["defunciones"])*100000/$poblacion_nacional;
                    $linea = $row["fecha"].','.$cve_ent[$i].','.strtoupper($nombres_ent[$i]).','.$row["camasOcupadas"].','.$camasOcupadas100K.','.$row["camasOcupadasHG"].','
                            .$row["camasOcupadasVentiladores"].','.$row["defunciones"].','.$defunciones100K.','.$row["camasDisponiblesTotales"].','
                            .$row["disponiblesHG"].','.$row["disponiblesVentiladores"].','.$row["camasDisponiblesVentiladorNoUCI"].','.$row["camasDisponiblesUCI"].','.$row["camasOcupadasUCI"];
                    fwrite($archivo, $linea."\r\n");
                }
            }
        } else {
            echo "0 results";
        }
        $date = $date->add(new DateInterval('P1D'));
    }
}
```

Figura 76. Script PHP Red IRAG 23.

2. **tendencias_zm** – El procedimiento es idéntico que el script de “tendencias_estados.php”: Crear Directorio -> Obtener Claves ZM -> Crear archivo Tendencias -> Recorrer Fechas -> Query Select -> Escribir información en archivo Tendencias.

```php
//CREAR DIRECTORIOS FORMATO TENDENCIAS
$directorio = 'ReporteTendencias';
if(!is_dir($directorio)){
    mkdir($directorio, 0777, true);
    echo "Directorio creado <br>";
} else {
    echo "La carpeta ya existe <br>";
}

//OBTENER TODAS LAS CLAVES DE LAS ZONAS METROPOLITANAS
$cve_zm = array();
$nombres_zm = array();
$poblacion_zm = array();
$sql = "SELECT cve_zm,nombre,poblacion FROM zona_metropolitana ORDER BY id_zona_metropolitana ASC";

$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        array_push($cve_zm, $row["cve_zm"]);
        array_push($nombres_zm, $row["nombre"]);
        array_push($poblacion_zm, $row["poblacion"]);
    }
} else {
    echo "0 results";
}

//CREAR ARCHIVO TENDENCIAS PARA ZONAS METROPOLITANAS
$archivo = fopen($directorio."/Agregados_HospitalizacionZM_".(new DateTime("now"))->format('ymd').".csv", "w") or die("Unable to open file!");

//ESCRIBIR LÍNEA EN ARCHIVOS
$primera_linea = "date,cve,state,camasOcupadas,camasOcupadas100K,camasOcupadasHG,camasOcupadasVentiladores,defunciones,"
                ."defunciones100K,camasDisponiblesTotales,disponiblesHG,disponiblesVentiladores,"
                ."camasDisponiblesVentiladorNoUCI,camasDisponiblesUCI,camasOcupadasUCI";
fwrite($archivo, $primera_linea."\r\n");
```

Figura 77. Script PHP Red IRAG 24.

```php
for($i=0; $i<sizeof($cve_zm); $i++){
    $date = new DateTime('2020-04-01');
    while($date <= new DateTime("now")){
        $fecha = $date->format('Y-m-d');
        //SELECT
        $sql = "SELECT fecha,
                    SUM(numero_pacientes_hospitalizados_no_uci)+SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadas,
                    SUM(numero_pacientes_hospitalizados_no_uci)-(SUM(numero_pacientes_intubados)-SUM(pacientes_hospitalizados_irag_uci)) AS camasOcupadasHG,
                    SUM(numero_pacientes_intubados) AS camasOcupadasVentiladores,
                    SUM(defunciones) AS defunciones,
                    SUM(total_camas) AS camasDisponiblesTotales,
                    SUM(camas_irag_disponibles_no_ventilador) AS disponiblesHG,
                    SUM(camas_irag_disponibles_con_ventilador_fuera_uci)+SUM(camas_disponibles_irag_uci) AS disponiblesVentiladores,
                    SUM(camas_irag_disponibles_con_ventilador_fuera_uci) AS camasDisponiblesVentiladorNoUCI,
                    SUM(camas_disponibles_irag_uci) AS camasDisponiblesUCI,
                    SUM(pacientes_hospitalizados_irag_uci) AS camasOcupadasUCI
                    FROM `registro_hospital` WHERE fecha='".$fecha."' AND id_hospital IN
                            (SELECT id_hospital FROM hospital WHERE id_municipio IN
                            (SELECT id_municipio FROM municipio WHERE id_zona_metropolitana =
                            (SELECT id_zona_metropolitana FROM zona_metropolitana WHERE cve_zm='".$cve_zm[$i]."')))";
        $result = $conn->query($sql);

        if ($result->num_rows > 0) {
            while($row = $result->fetch_assoc()) {
                if(empty($row["fecha"])){
                    $linea = $fecha.','.$cve_zm[$i].','.$nombres_zm[$i].',0,0,0,0,0,0,0,0,0,0,0,0';
                    fwrite($archivo, $linea."\r\n");
                } else {
                    $camasOcupadas100K = intval($row["camasOcupadas"])*100000/intval($poblacion_zm[$i]);
                    $defunciones100K = intval($row["defunciones"])*100000/intval($poblacion_zm[$i]);
                    $linea = $row["fecha"].','.$cve_zm[$i].','.$nombres_zm[$i].','.$row["camasOcupadas"].','.$camasOcupadas100K.','.$row["camasOcupadasHG"].','
                            .$row["camasOcupadasVentiladores"].','.$row["defunciones"].','.$defunciones100K.','.$row["camasDisponiblesTotales"].','
                            .$row["disponiblesHG"].','.$row["disponiblesVentiladores"].','.$row["camasDisponiblesVentiladorNoUCI"].','.$row["camasDisponiblesUCI"].','.$row["camasOcupadasUCI"];
                    fwrite($archivo, $linea."\r\n");
                }
            }
        } else {
            echo "0 results";
        }
        $date = $date->add(new DateInterval('P1D'));
    }
}
```

Figura 78. Script PHP Red IRAG 25.

Registro Diario

A continuación se muestran las figuras correspondientes al Script de los registros diarios. Primero hay que obtener los nombres de los archivos diarios (los

colocaremos en el directorio Archivos-Diarios dentro de Fuente-Datos) y consideraremos que la fecha en el nombre tiene el formato ‘yymmdd’. Entonces recorremos cada archivo y se obtiene los datos pertinentes que son distintos entre el archivo Infraestructura y archivo Pacientes, por lo que tenemos que distinguir entre los dos casos. Validamos que no exista el registro en la Base de Datos (si existe actualizamos el registro, esto sirve para llenar un registro de una fecha con la información de los dos archivos); y si no, lo insertamos. Si se genera un error en la inserción (preguntamos por ello), entonces significa que la CLUE no está registrada en la Base de Datos. En ese caso insertamos la CLUE sin enlace a la Tabla Municipio y luego insertamos el registro diario.

```php
//OBTENER NOMBRES DE ARCHIVOS EN UNA CARPETA
$ruta = 'Fuente-Datos/Archivos-Diarios/';
//ELIMINAR DIRECTORIOS ., ..; Y OBTENER INDEXADO NORMAL
$files = array_values(array_diff(scandir($ruta), array('.', '..')));

//RECORRER ARCHIVOS
for($i=0;$i<sizeof($files);$i++){
    //OBTENER FECHA EN EL NOMBRE DEL ARCHIVO. RECORDAR FECHA DE LOS REGISTROS = DÍA ANTERIOR A FECHA DEL ARCHIVO
    $fecha_archivo = substr($files[$i],0,6);
    echo "Archivo: ".$files[$i]." <br>";
    //PARSEAR FECHA DEL ARCHIVO AL DÍA ANTERIOR
    $date = DateTime::createFromFormat('ymd',$fecha_archivo);
    $date = $date->sub(new DateInterval('P1D'));
    $fecha = $date->format('Y-m-d');
    //ABRIR ARCHIVO
    $myfile = fopen($ruta.$files[$i], "r") or die("Unable to open file!");
    //PRIMERA LÍNEA
    $primera_linea = (fgetcsv($myfile));
```

Figura 79. Script PHP Red IRAG 26.

```php
if(sizeof($primera_linea)==10){         //ARCHIVO INFRAESTRUCTURA
    while(!feof($myfile)){
        //LEER LÍNEA
        $linea = (fgetcsv($myfile));
        //LÍNEA VACÍA
        if(!$linea) continue;
        //OBTENER VARIABLES
        $clue = $linea[0];
        $total_camas = $linea[5];
        $camas_irag_disponibles_no_ventilador = $linea[6];
        $camas_irag_disponibles_con_ventilador_fuera_uci = $linea[7];
        $camas_disponibles_irag_uci = $linea[8];

        //PRIMERO SELECT, SI EXISTE UPDATE, SI NO INSERT
        $sql = "SELECT id_registro_hospital FROM registro_hospital
                WHERE id_hospital=(SELECT id_hospital FROM hospital WHERE clue='".$clue."') AND fecha='".$fecha."'";

        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
            $sql = "UPDATE registro_hospital SET
                    total_camas=".$total_camas.",
                    camas_irag_disponibles_no_ventilador=".$camas_irag_disponibles_no_ventilador.",
                    camas_irag_disponibles_con_ventilador_fuera_uci=".$camas_irag_disponibles_con_ventilador_fuera_uci.",
                    camas_disponibles_irag_uci=".$camas_disponibles_irag_uci."
                WHERE id_hospital=(SELECT id_hospital FROM hospital WHERE clue='".$clue."') AND fecha='".$fecha."'";

            if ($conn->query($sql) === TRUE) {
                //echo "Record updated successfully: ".$clue." en ".$fecha." <br>";
            } else {
                echo "Error updating record: " . $conn->error;
            }
        } else {
```

Figura 80. Script PHP Red IRAG 27.

```
        //INSERT
        $sql = "INSERT INTO registro_hospital (id_hospital, total_camas, camas_irag_disponibles_no_ventilador,
                    camas_irag_disponibles_con_ventilador_fuera_uci, camas_disponibles_irag_uci,
                    ventiladores_disponibles, ingresos_irag, egresos_irag, numero_pacientes_hospitalizados_no_uci,
                    pacientes_hospitalizados_irag_uci,numero_pacientes_intubados,defunciones,fecha)
                VALUES ((SELECT id_hospital FROM hospital WHERE clue='".$clue."'), ".$total_camas.",
                ".$camas_irag_disponibles_no_ventilador.", ".$camas_irag_disponibles_con_ventilador_fuera_uci.",
                ".$camas_disponibles_irag_uci.", 0, 0, 0, 0, 0, 0, 0, '".$fecha."')";

        if ($conn->query($sql) === TRUE) {
            //echo "New record created successfully: ".$clue." en ".$fecha." <br>";
        } else {
            //INSERT
            $sql = "INSERT INTO hospital (id_municipio, clue, nombre)
                    VALUES (NULL, '".$clue."', '')";

            if ($conn->query($sql) === TRUE) {
                //Nueva Clue creada. Insertar registro
                echo "New record created successfully: ".$clue." <br>";
                $sql = "INSERT INTO registro_hospital (id_hospital, total_camas, camas_irag_disponibles_no_ventilador,
                            camas_irag_disponibles_con_ventilador_fuera_uci, camas_disponibles_irag_uci,
                            ventiladores_disponibles, ingresos_irag, egresos_irag, numero_pacientes_hospitalizados_no_uci,
                            pacientes_hospitalizados_irag_uci,numero_pacientes_intubados,defunciones,fecha)
                        VALUES ((SELECT id_hospital FROM hospital WHERE clue='".$clue."'), ".$total_camas.",
                        ".$camas_irag_disponibles_no_ventilador.", ".$camas_irag_disponibles_con_ventilador_fuera_uci.",
                        ".$camas_disponibles_irag_uci.", 0, 0, 0, 0, 0, 0, 0, '".$fecha."')";

                if ($conn->query($sql) === TRUE) {
                } else {
                    //Error inoportuno al insertar registro de nueva clue
                    echo "Error: " . $sql . "<br>" . $conn->error. "<br>";
                }
            } else {
                //Error inoportuno al insertar nueva clue
                echo "Error: " . $sql . "<br>" . $conn->error. "<br>";
            }
        }
    }
}
```

Figura 81. Script PHP Red IRAG 28.

```php
} else {                        //ARCHIVO PACIENTES
    while(!feof($myfile)){
        //LEER LÍNEA
        $linea = (fgetcsv($myfile));
        //LÍNEA VACÍA
        if(!$linea) continue;
        //OBTENER VARIABLES
        $clue = $linea[0];
        $ingresos_irag = $linea[4];
        $egresos_irag = $linea[5];
        $numero_pacientes_hospitalizados_no_uci = $linea[6];
        $pacientes_hospitalizados_irag_uci = $linea[7];
        $numero_pacientes_intubados = $linea[8];
        $defunciones = $linea[9];

        //PRIMERO SELECT, SI EXISTE UPDATE, SI NO INSERT
        $sql = "SELECT id_registro_hospital FROM registro_hospital
                WHERE id_hospital=(SELECT id_hospital FROM hospital WHERE clue='".$clue."') AND fecha='".$fecha."'";

        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
            $sql = "UPDATE registro_hospital SET
                    ingresos_irag=".$ingresos_irag.",
                    egresos_irag=".$egresos_irag.",
                    numero_pacientes_hospitalizados_no_uci=".$numero_pacientes_hospitalizados_no_uci.",
                    pacientes_hospitalizados_irag_uci=".$pacientes_hospitalizados_irag_uci.",
                    numero_pacientes_intubados=".$numero_pacientes_intubados.",
                    defunciones=".$defunciones."
                WHERE id_hospital=(SELECT id_hospital FROM hospital WHERE clue='".$clue."') AND fecha='".$fecha."'";

            if ($conn->query($sql) === TRUE) {
                //echo "Record updated successfully: ".$clue." en ".$fecha." <br>";
            } else {
                echo "Error updating record: " . $conn->error;
            }
        } else {
```

Figura 82. Script PHP Red IRAG 29.

```
                //INSERT
                $sql = "INSERT INTO registro_hospital (id_hospital, total_camas, camas_irag_disponibles_no_ventilador,
                            camas_irag_disponibles_con_ventilador_fuera_uci, camas_disponibles_irag_uci,
                            ventiladores_disponibles, ingresos_irag, egresos_irag, numero_pacientes_hospitalizados_no_uci,
                            pacientes_hospitalizados_irag_uci,numero_pacientes_intubados,defunciones,fecha)
                        VALUES ((SELECT id_hospital FROM hospital WHERE clue='".$clue."'), 0, 0, 0, 0, 0,
                            ".$ingresos_irag.", ".$egresos_irag.", ".$numero_pacientes_hospitalizados_no_uci.",
                            ".$pacientes_hospitalizados_irag_uci.", ".$numero_pacientes_intubados.", ".$defunciones.", '".$fecha."')";

                if ($conn->query($sql) === TRUE) {
                    //echo "New record created successfully: ".$clue." en ".$fecha." <br>";
                } else {
                    //INSERT
                    $sql = "INSERT INTO hospital (id_municipio, clue, nombre)
                            VALUES (NULL, '".$clue."', '')";

                    if ($conn->query($sql) === TRUE) {
                        //Nueva Clue creada. Insertar registro
                        echo "New record created successfully: ".$clue." <br>";
                        $sql = "INSERT INTO registro_hospital (id_hospital, total_camas, camas_irag_disponibles_no_ventilador,
                                    camas_irag_disponibles_con_ventilador_fuera_uci, camas_disponibles_irag_uci,
                                    ventiladores_disponibles, ingresos_irag, egresos_irag, numero_pacientes_hospitalizados_no_uci,
                                    pacientes_hospitalizados_irag_uci,numero_pacientes_intubados,defunciones,fecha)
                                VALUES ((SELECT id_hospital FROM hospital WHERE clue='".$clue."'), 0, 0, 0, 0, 0,
                                    ".$ingresos_irag.", ".$egresos_irag.", ".$numero_pacientes_hospitalizados_no_uci.",
                                    ".$pacientes_hospitalizados_irag_uci.", ".$numero_pacientes_intubados.", ".$defunciones.", '".$fecha."')";

                        if ($conn->query($sql) === TRUE) {
                        } else {
                            //Error inoportuno al insertar registro de nueva clue
                            echo "Error: " . $sql . "<br>" . $conn->error. "<br>";
                        }
                    } else {
                        //Error inoportuno al insertar nueva clue
                        echo "Error: " . $sql . "<br>" . $conn->error. "<br>";
                    }
                }
            }
        }
    }
    //CERRAR EL ARCHIVO
    fclose($myfile);
}
```

Figura 83. Script PHP Red IRAG 30.

Resultados

Archivos generados

Con lo presentado anteriormente, se logró generar:

1) La Base de Datos, exportada al archivo "**irag.sql**".
2) Los archivos en Formato AMA, en el directorio "**Formato-AMA\datos_20210224**" con fecha hasta el 17/02/2021.
3) Los archivos en Formato Tendencias, en el directorio "**ReporteTendencias**" con fecha hasta el 17/02/2021.

Se compararon el contenido de todos los archivos generados con la Base de Datos con otros archivos generados de otras maneras y se encuentran idénticos en un 99%.

La comparación se realizó igualando celdas en Excel, como también utilizando un comparador de textos online.

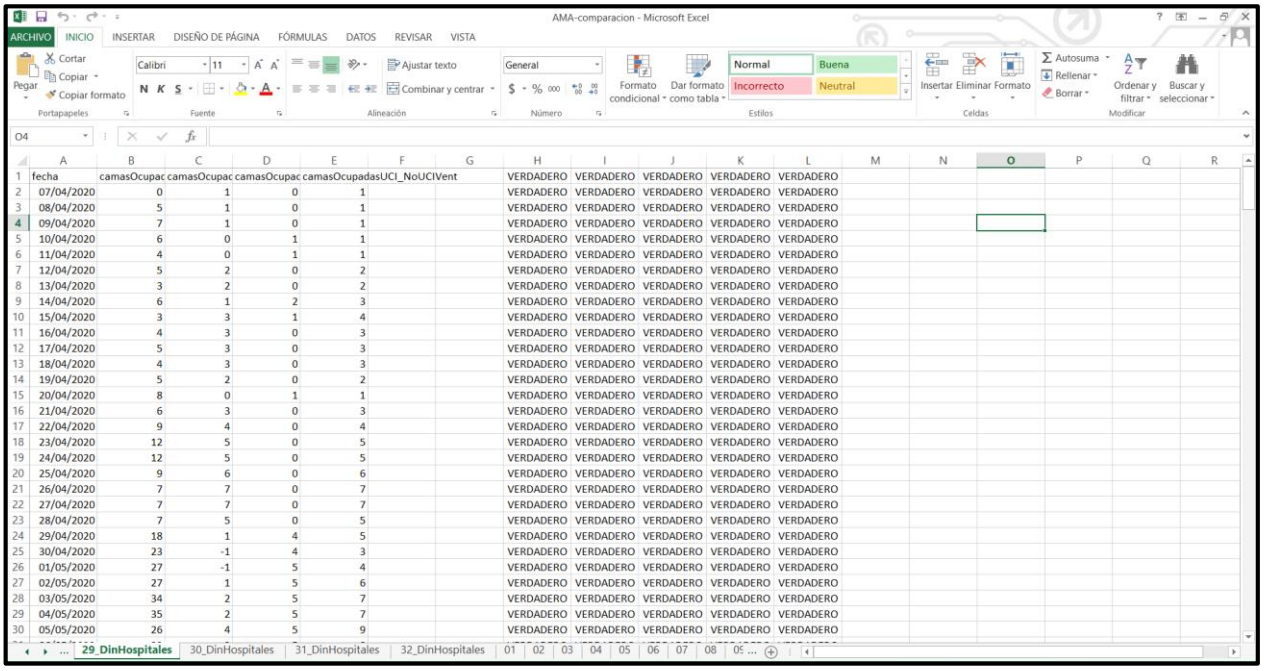


Figura 84. Comparación de resultados Red IRAG.

Problemas

Respecto a la comparación del punto anterior, se encontraron muy pocas discrepancias que se exponen a continuación:

I. Las fechas: 14/09/2020, 28/09/2020, 25/10/2020, 13/11/2020, 21/12/2020 y 24/12/2020; contienen en sus registros los mismos valores a sus respectivas fechas anteriores (14/09/2020 con 13/09/2020 por ejemplo). Esto sucedió porque los archivos diarios correspondientes tuvieron errores y se tuvo que tomar los registros anteriores.

II. La fecha: 28/07/2020, para los Estados con clave 09, 14, 24 y 30; tiene variaciones mínimas en los datos. Existen dos registros diferentes (valores distintos) en el archivo histórico que está generando esta discrepancia.

III. En los archivos de Tendencias, las columnas: camasOcupadas100K y defunciones100K; a generarse de una división por la población correspondiente, crea una diferencia muy pequeña en la precisión de los decimales (es decir los últimos decimales son distintos pero los primeros ~12 son iguales).

Pendientes

Lo que no se logró realizar durante este período:

I. Generar aviso cuando se agregan nuevas CLUEs a la Base de Datos cuando se detectan nuevos registros en los archivos diarios.

II. Detectar CLUEs que no reportaron en cada día y el porcentaje respecto al total.

III. Solucionar los problemas mencionados anteriormente.

BIBLIOGRAFÍA

Desconocido. (N/A). *Elementos de una tienda online para vender en Internet. Marzo 5, 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*. Marzo 5, 2021, de Hostinger Sitio web: https://www.hostinger.mx/tutoriales/mejores-plataformas-ecommerce/

Garcia, J. (2020). *8 soluciones para crear una tienda online gratis*. Marzo 5, 2021, de WebsiteToolTester Sitio web: https://www.websitetooltester.com/es/blog/crear-tienda-online-gratis/

Desnococido. (2021). *XAMPP*. Marzo 15, 2021, de VMware Sitio web: https://www.apachefriends.org/es/index.html