



EVIMERCE

7 Pedidos Hoy [Ver pedidos](#)

24347.30€ Ingresos Totales [Ver ventas](#)

19 Productos Activos [Ver productos](#)

1 Bajo Stock [Ver stock](#)

Ventas Mensuales [Últimos 30 días](#)

Categorías de Productos

Categoría	Cantidad
Iluminación Técnica	3
Cableado y Conectores	3
Herramientas Eléctricas	3
Automatización y PLC	3
Material Eléctrico	3
Iluminación Decorativa	2
Componentes Electrónicos	2

Productos Más Vendidos

Producto	Ventas	Stock	Estado
PLC Siemens LOGO! 8 12/24RCE 189.00€	140	-85	Agujado
Foco LED Industrial 50W	8	40	En stock

Pedidos Recientes 1 pendientes

Pedido #	Cliente	Total	Estado	Fecha
#000009	Alvaro	17771.00€	Pagado	26/1, 09:56
#000008	Alvaro	308.75€	Pagado	26/1, 09:48

Índice

EviMerce

1. Descripción General del Proyecto

2. Tecnologías Utilizadas

 2.1 Frontend

 2.2 Backend

3. Arquitectura del Proyecto

4. Instalación y Ejecución en Local

 4.1 Requisitos Previos

 4.2 Clonar el Proyecto

 4.3 Configuración de PHP

 4.4 Inicio del Servidor

 Exposición en red local

5. Documentación de Uso

 5.1 Usuario Cliente

 5.2 Usuario Administrador

6. Ejemplo de uso

7. Ejemplo de uso:

 ElectricMer:

8. Proceso técnico de creación

 Fase: Análisis y diseño

 Diagrama ER

 Fase: Arquitectura del Proyecto

 Fase: Backend (PHP + MySQL)

 Fase: Frontend (HTML + CSS + JS)

 Fase: Interacción Frontend–Backend

 Fase de Gestión de Archivos

 Dockerizacion

9. Licencia

EviMerce

EviMerce es una aplicación web de comercio electrónico desarrollada como proyecto de prácticas para **1º de Desarrollo de Aplicaciones Web (DAW)**. El objetivo principal del proyecto es aplicar y consolidar conocimientos de desarrollo web frontend y backend utilizando tecnologías clásicas del entorno web.

1. Descripción General del Proyecto

EviMerce es un eCommerce básico que permite la gestión completa de productos, categorías y pedidos, así como la interacción de los clientes con la tienda mediante un sistema de compra, carrito y seguimiento de pedidos.

El proyecto separa claramente la interfaz de **administración** y la de **clientes**, permitiendo una gestión segura y organizada de la tienda.

Este proyecto ha sido desarrollado con fines educativos para la empresa **Evirom**, ubicada en Cañada Rosal (Sevilla, España).

2. Tecnologías Utilizadas

2.1 Frontend

- HTML5
- CSS3
- JavaScript
- jQuery 3.7.1
- jQuery UI
- jQuery Knob
- Bootstrap 5
- AdminLTE
- Font Awesome
- JVectorMap
- DateTimePicker
- DateRangePicker
- DataTables
- Chart.js
- Moment.js

El frontend está enfocado en ofrecer una interfaz clara, visual y funcional, utilizando plantillas administrativas modernas y componentes reutilizables.

2.2 Backend

- PHP 8.5.1
- PDO (PHP Data Objects)
- MySQL 8.0.44

El backend se encarga de la lógica de negocio, la gestión de usuarios, productos, pedidos y la comunicación segura con la base de datos mediante PDO.

3. Arquitectura del Proyecto

- **Separación de vistas:**
 - Interfaz de cliente
 - Panel de administración
- **Router personalizado en PHP** para gestionar las rutas de la aplicación.
- **Base de datos SQL completa**, diseñada para cubrir productos, categorías, usuarios, pedidos y devoluciones.

4. Instalación y Ejecución en Local

4.1 Requisitos Previos

- Sistema operativo Windows
- PHP instalado en: `C:/php/php.exe`
- Servidor MySQL activo
- Usuario MySQL: `root`
- Contraseña MySQL: `root`

4.2 Clonar el Proyecto

```
git clone https://github.com/arugerdev/1_DAW_ECOMERCE.git  
cd 1_DAW_ECOMERCE
```

4.3 Configuración de PHP

Es necesario habilitar la extensión **pdo_mysql** en el archivo *php.ini*:

```
;extension = pdo_mysql
```

Debe quedar como:

```
extension = pdo_mysql
```

También es posible copiar el archivo *php.ini* personalizado incluido en el repositorio y reemplazarlo en la carpeta de instalación de PHP.

4.4 Inicio del Servidor

Ejecutar el archivo:

```
./start.bat
```

O hacer doble clic desde el explorador de archivos.

Exposición en red local

Editar el archivo *start.bat*:

Expuesto en red local:

```
C:/php/php.exe -S 192.168.2.175:80
```

Solo local:

```
C:/php/php.exe -S localhost:80
```

5. Documentación de Uso

5.1 Usuario Cliente

El cliente puede:

- Visualizar productos
- Filtrar productos por categorías
- Buscar productos
- Añadir productos al carrito
- Realizar compras
- Gestionar envíos
- Solicitar reembolsos

La interfaz está pensada para ser intuitiva y accesible, con un diseño responsive básico.

5.2 Usuario Administrador

El administrador puede:

- Iniciar sesión en el panel de administración
- Crear, editar y eliminar productos
- Subir y gestionar imágenes de productos
- Crear y editar categorías
- Gestionar pedidos
- Editar estados de pedidos
- Gestionar reembolsos
- Visualizar estadísticas mediante gráficos

6. Ejemplo de uso

Caso: Creación y Venta de un Producto

1. El administrador accede al panel de administración.
2. Crea una nueva categoría llamada "**Iluminación**".
3. Añade un producto:
 - **Nombre:** Bombilla LED 12W
 - **Precio:** 4,99 €
 - **Categoría:** Iluminación
 - **Imagen:** bombilla_led.jpg
4. El producto se guarda y queda visible para los clientes.
5. Un cliente accede a la tienda y busca "bombilla".
6. Añade el producto al carrito.
7. Finaliza la compra y selecciona el método de envío.
8. El pedido queda registrado en el sistema.
9. El administrador edita el estado del pedido a "Enviado".

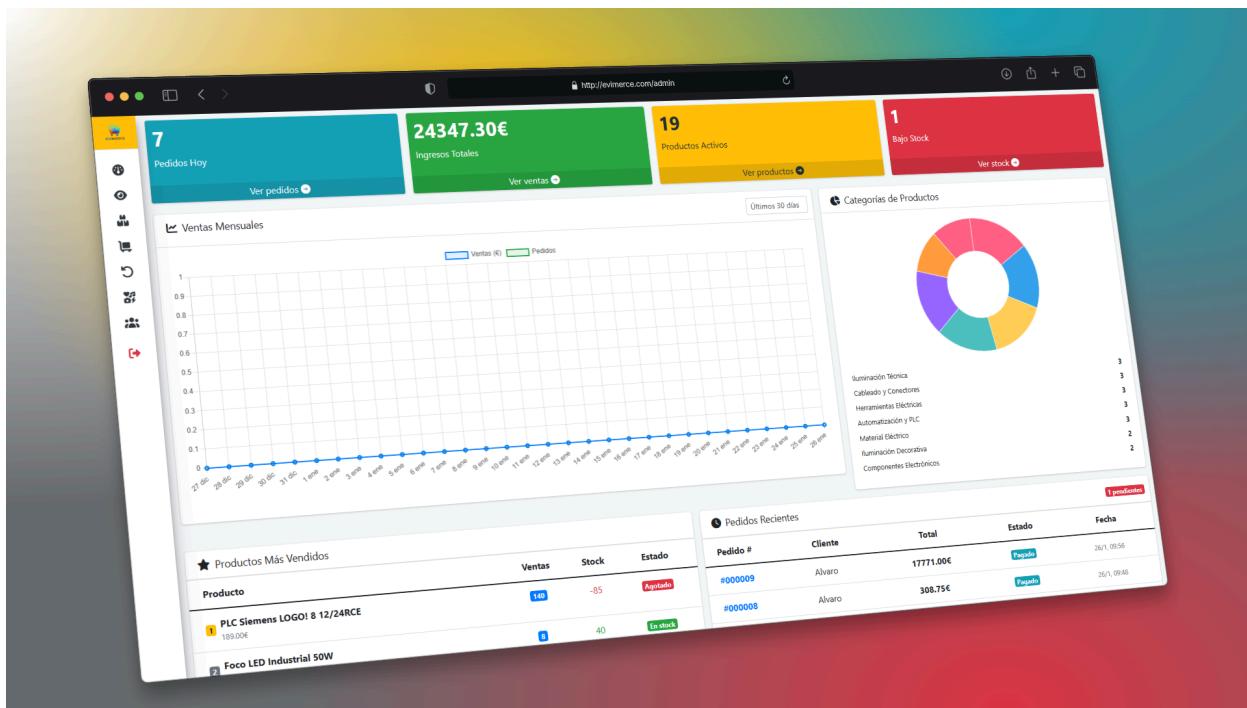
Este flujo representa el funcionamiento básico del eCommerce.

7. Ejemplo de uso:

ElectricMer:



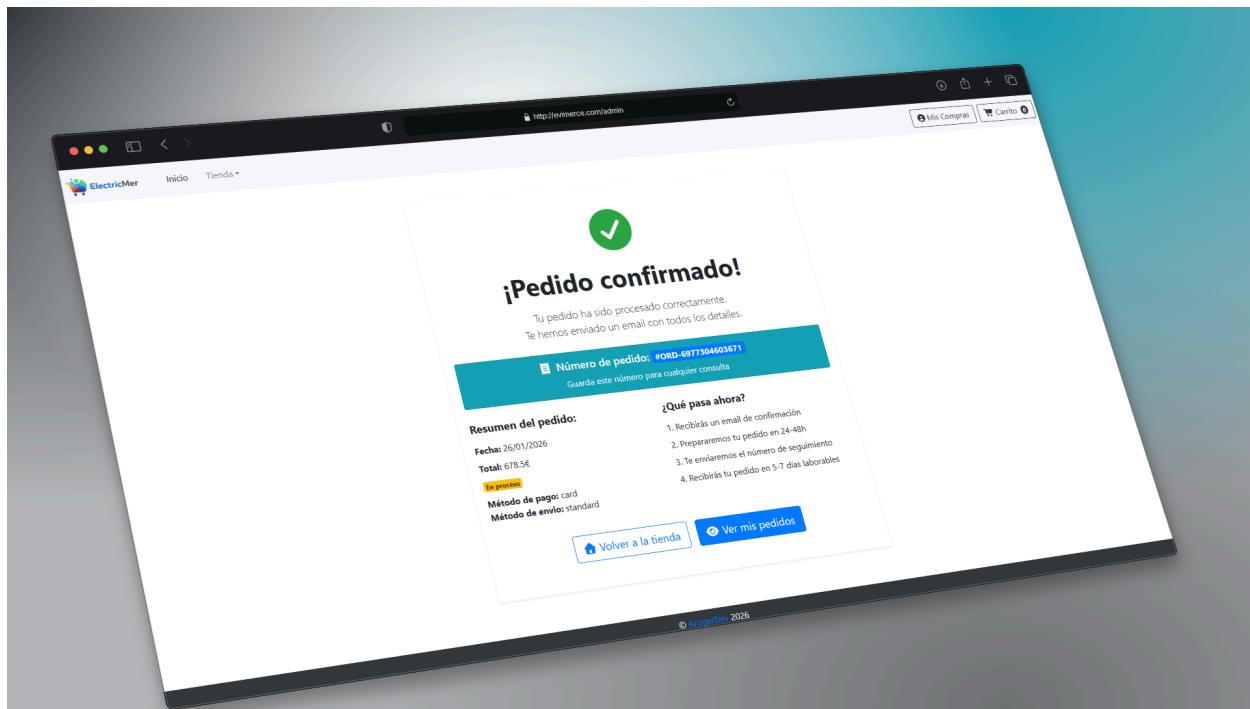
ElectricMer es un ejemplo creado para usarse en **Evimerce**, se trata de una empresa de venta electrónica y de componentes electrónicos simulada.



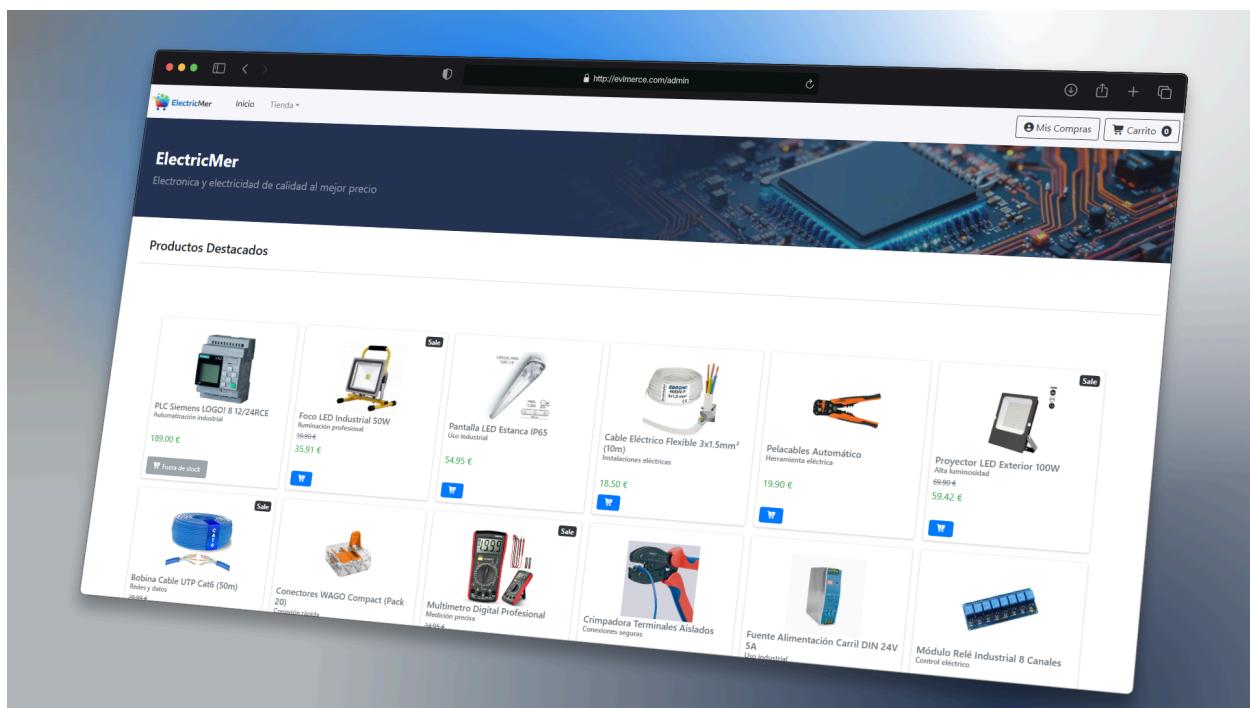
El ejemplo viene con 1 usuario administrador, 7 categorías nuevas y 19 productos, los cuales vienen asociados a sus correspondientes categorías.

Las imágenes de los productos, al no ser guardadas directamente en la base de datos, sino que en la carpeta del proyecto en una carpeta separada, no se pueden subir en este ejemplo, pero asociar las imágenes a cada producto y conseguirlas de internet o haciendo fotos a los productos es muy sencillo.

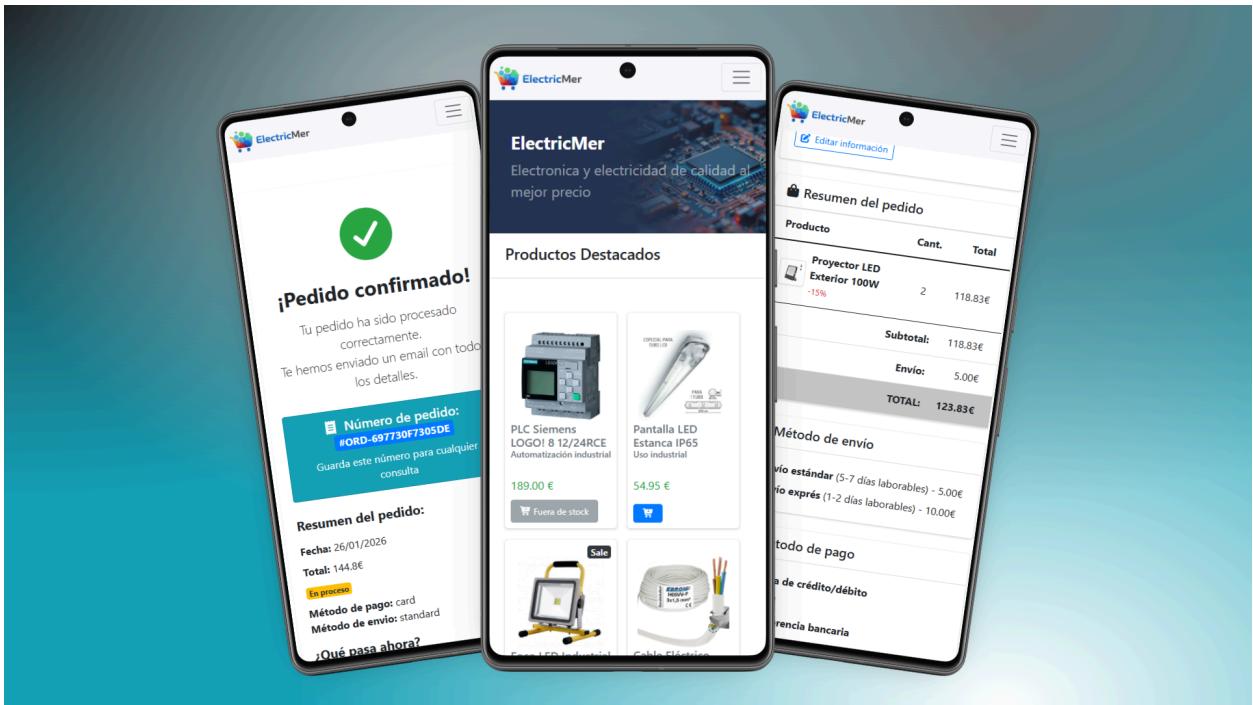
Los procesos de creación de cliente, compra, confirmación de compra, etc son muy sencillos y orientados a poder ser usados por cualquier persona.



Los productos pueden ser buscados por nombre, precio, descripción o filtrados por categorías.



Móviles:



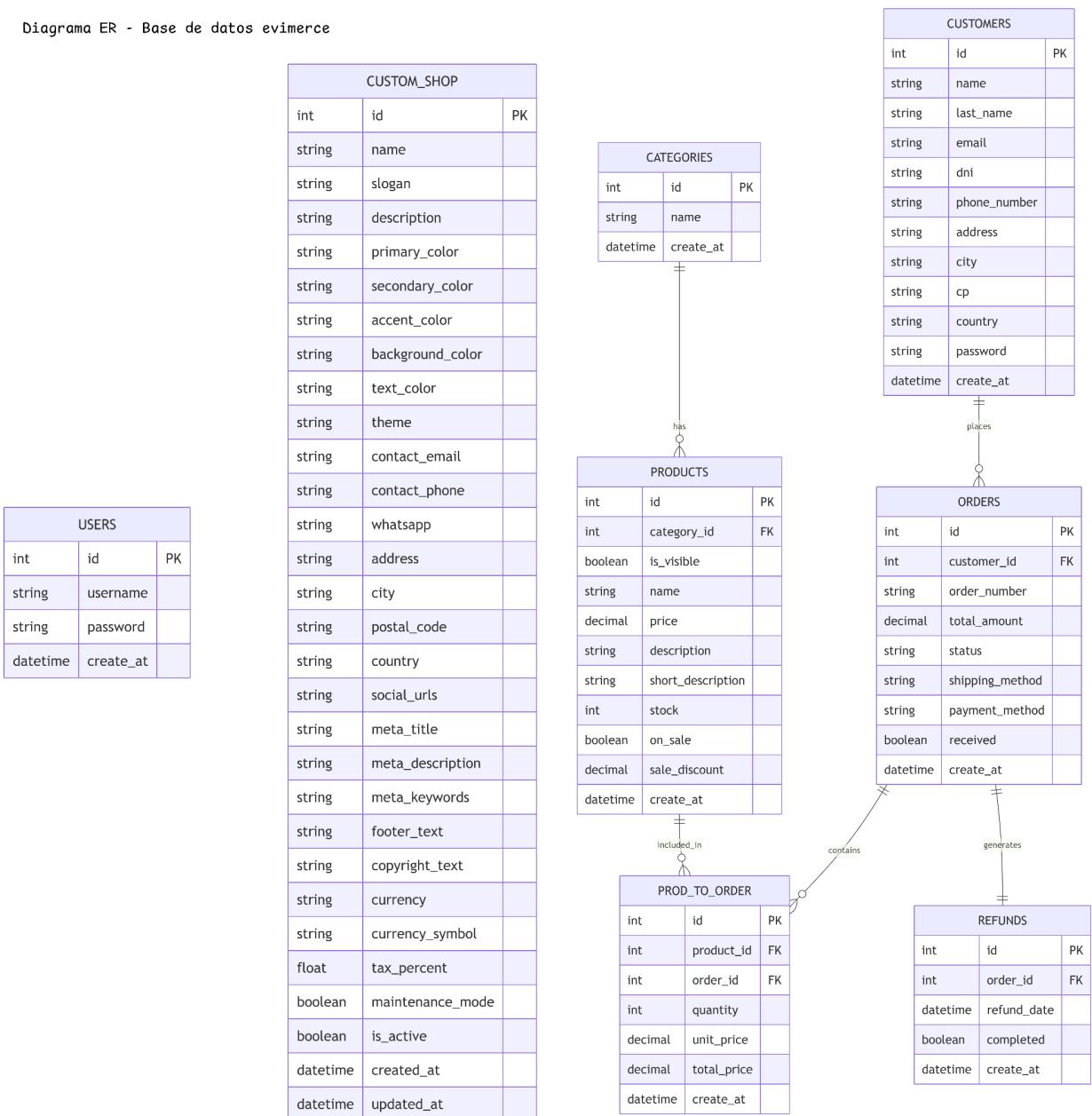
8. Proceso técnico de creación

Fase: Análisis y diseño

1. Definición de requisitos:
 - Roles:
 - Cliente (Usuario, info pública, sin permisos especiales)
 - Administrador
 - Funcionalidades clave:
 - Productos
 - Categorías
 - Carrito
 - Pedidos
 - Reembolsos
 - Personalización de tienda
 - Usuarios
 - Clientes guardados para ver pedidos propios
 - Separación de interfaces:
 - Tienda pública
 - Panel admin (usuario/contraseña, por defecto pero modificable)
2. Diseño de la base de datos
 - Tablas principales:
 - users
 - products
 - categories
 - orders
 - prodToOrder
 - refunds
 - customers
 - customShop
 - Relaciones:
 - products → categories
 - orders → customers
 - orders → prodToOrder ← products
 - refunds → orders

Diagrama ER

Diagrama ER - Base de datos evimerce



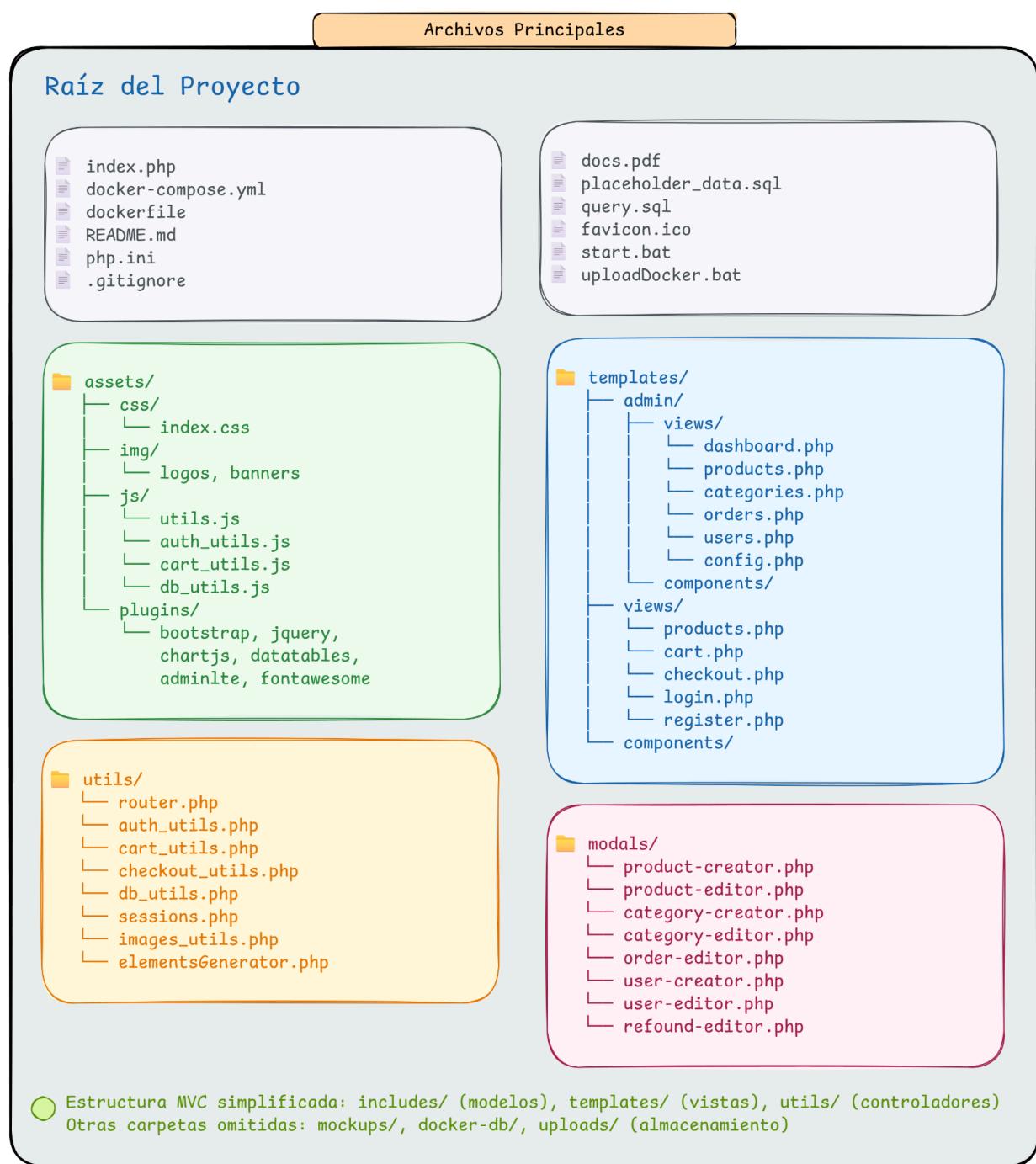
Fase: Arquitectura del Proyecto

1. Estructura de carpetas

Separar responsabilidades desde el inicio:

Estructura del Proyecto EviMerce

☞ Carpetas y archivos importantes resaltados (sin repetitivos)



2. Router en PHP

- Un único punto de entrada (index.php)
- index.php → routes.php
- Router manual:
 - Detecta URL
 - Llama al controlador correcto
 - Evita múltiples archivos PHP accesibles directamente

```
$url = get_current_url();
if (is_admin_route())
    go to admin router;
switch ($url) {
    case "/":
        include "./templates/index.php";
    case "/terms":
        include "./templates/views/terms.php";
    case "/privacy":
        include "./templates/views/privacy.php";
    case "/products":
        include "./templates/views/products.php";
    case "/products/category":
        include "./templates/views/products_category.php";
    case "/product":
        include "./templates/views/product.php";
    case "/cart":
        include "./templates/views/cart.php";
    case "/orders":
        include "./templates/views/orders.php";
    case "/login":
        include "./templates/views/login.php";
    case "/register":
        include "./templates/views/register.php";
    case "/checkout":
        include "./templates/views/checkout.php";
    case "/checkout/confirm":
        include "./templates/views/checkout_confirm.php";
    case "/checkout/success":
        include "./templates/views/checkout_success.php";
default:
    http_response_code(404);
    include "./templates/views/404.php";
}
```

Fase: Backend (PHP + MySQL)

1. Conexión a base de datos
 - Uso exclusivo de PDO
 - Conexión centralizada:

```
$bd_host = getenv("DB_HOST") ?: "localhost";
$bd_base = getenv("DB_NAME") ?: "evimerce";
$bd_usuario = getenv("DB_USER") ?: "root";
$bd_password = getenv("DB_PASS") ?: "root";

// Conexión a BD usando PDO
global $pdo;
$pdo = new PDO('host='.$bd_host.';dbname='.$bd_base.';',
    $bd_usuario, $bd_password,
    [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        Pdo\MySql::ATTR_INIT_COMMAND => "SET NAMES utf8mb4"
    ]);

```

- Prepared Statements siempre

```
$select = $_REQUEST["select"] ?? "";
$table = $_REQUEST["table"] ?? "";
$extra = $_REQUEST["extra"] ?? "";

$query = $pdo->prepare("SELECT $select FROM $table $extra");
$query->execute();
$result = $query->fetchAll();
```

2. Lógica de negocio

- Separar por módulos:
 - Productos
 - Categorías
 - Usuarios
 - Pedidos
 - Reembolsos
- Cada módulo:
 - Crear
 - Leer
 - Actualizar
 - Eliminar (CRUD)

JavaScript:

```
/* ##### */
function selectData(select, table, extra = "", callback = () => { }) {
    $.ajax({
        url: "/utils/db_utils.php",
        type: "POST",
        data: {
            "action": "select",
            "select": select,
            "table": table,
            "extra": extra
        },
        success: (data) => {
            callback(JSON.parse(data))
        }
    });
}
```

```

<?php

require $_SERVER['DOCUMENT_ROOT'] . '/utils/sessions.php';

$option = $_REQUEST["action"];

switch ($option) {
    case "select":
        echo (selectData());
        break;
    case "delete":
        ...
}

function selectData()
{
    try {
        require $_SERVER['DOCUMENT_ROOT'] . "/includes/database.php";
        $select = $_REQUEST["select"] ?? "";
        $table = $_REQUEST["table"] ?? "";
        $extra = $_REQUEST["extra"] ?? "";

        $query = $pdo->prepare("SELECT $select FROM $table $extra");
        $query->execute();
        $result = $query->fetchALL();

        http_response_code(200);
        return json_encode([
            "success" => true,
            "data" => $result
        ]);
    } catch (PDOException $e) {
        http_response_code(500);
        return json_encode([
            "success" => false,
            "error" => $e->getMessage()
        ]);
        exit;
    }
}

```

3. Gestión de sesiones y roles
 - PHP Sessions
 - Verificación:
 - Usuario logueado
 - Rol administrador
 - Protección del panel admin

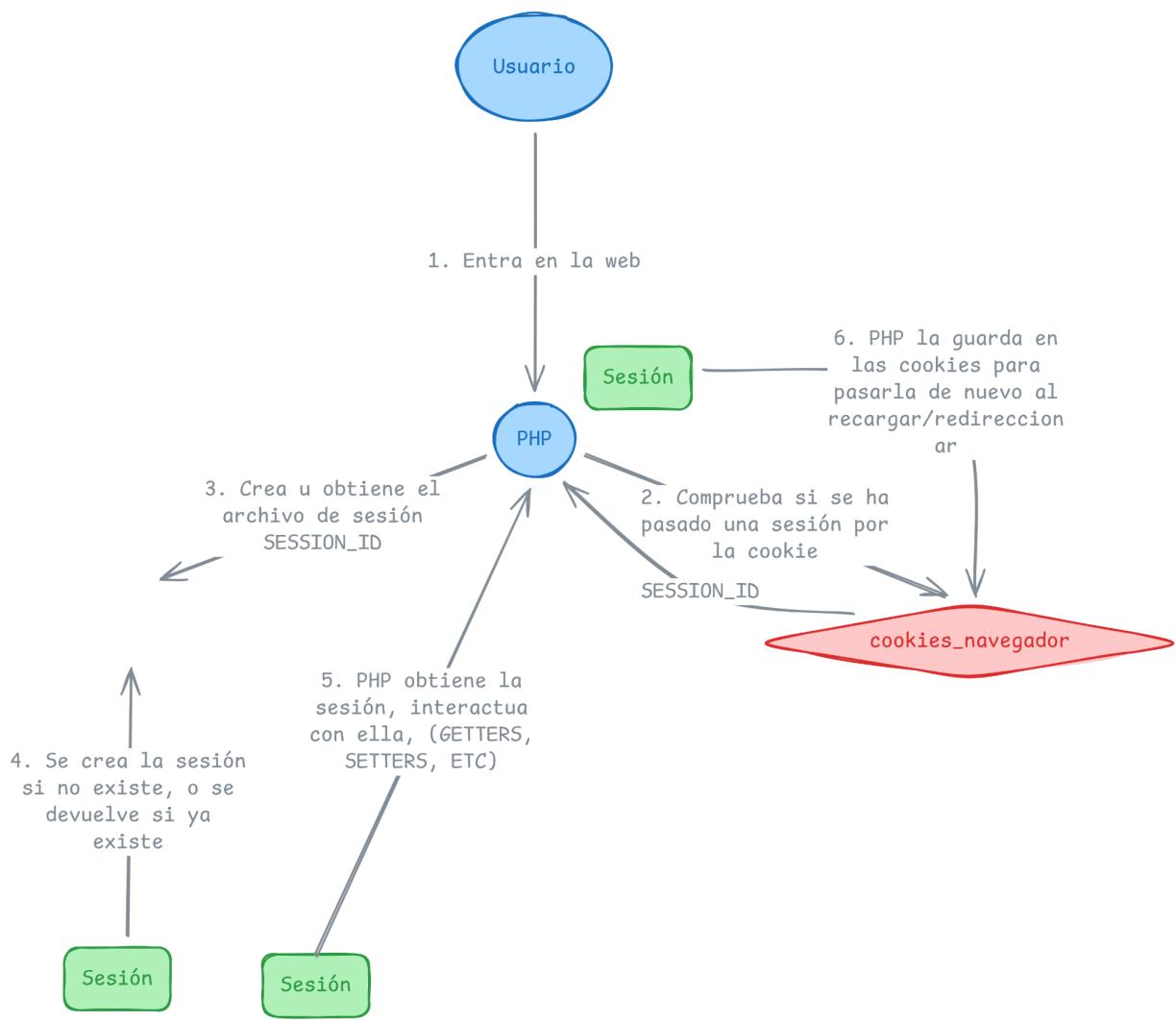
IMPORTAR DONDE SE USE \$_SESSION PARA CONTINUAR CON LA MISMA SESSION ID EN COOKIES

```
<?php
session_start();

$timeout = 7000;

if (isset($_SESSION['last_activity']) && time() - $_SESSION['last_activity'] >
$timeout) {
    session_destroy();
    session_start();
    $_SESSION['state'] = 'expired';
}

$_SESSION['last_activity'] = time();
```



Fase: Frontend (HTML + CSS + JS)

1. Cliente

- Bootstrap 5
- Catálogo de productos
- Filtros y búsqueda
- Carrito con PHP sessions y actualización con jQuery
- Checkout básico

```
<?php

$dir = $_SERVER['DOCUMENT_ROOT'] . "/uploads/img/products/" . $id . "/";
if (!is_dir($dir)) {
    echo '
        <div class="product-image-thumb cursor-pointer">
            
        </div>
    ';
} else {
    $files = array_diff(scandir($dir), array('.', '..'));
    foreach ($files as $fil) {
        echo '
<div class="product-image-thumb cursor-pointer">

</div>
';
    }
}
?>
</div>
</div>
<div class="col-12 col-sm-6">
<h3 class="my-3"><?php echo $data->name; ?></h3>
<p><?php echo $data->short_description ?></p>
<hr>
...

```

```
const mainImg = $(".product-image")
const imgs = $(".product-image-thumb")

imgs.on('click', (event) => {
    mainImg.attr("src", event.target.src);
})

$("#add-to-cart").on("click", (evnt) => {
    const prodData = <?php echo json_encode($data) ?>;
    LoadOrderSummary((res) => {
        if ((res.cart.filter((p) => p.id == prodData.id).length + 1) <=
prodData.stock) {
            addToCart(prodData, (data) => {
                updateCartQuantity()
            })
        }
    })
})
```

2. Panel de administración

- AdminLTE
- DataTables para listados
- Chart.js para estadísticas
- Formularios dinámicos con jQuery

Dashboard

5 Pedidos Hoy | 3412.45€ Ingresos Totales | 19 Productos Activos | 1 Bajo Stock

Ventas Mensuales

Categorías de Productos

Producto	Ventas	Stock	Estado	Pedido #	Cliente	Total	Estado	Fecha
★ Productos Más Vendidos								4 pendientes

```

selectData("*", "categories", "", (res) => {
  const data = res.data
  $('#categories-table').DataTable({
    columns: Object.keys(data[0]).map((key) => {
      return title: capitalizeFirstLetter(key)
    })
    columnDefs: [
      {
        targets: [0, 2],
        visible: false,
        searchable: false
      },
      {
        targets: 3,
        render: function(data, type, row) {
          return getRowActions(id, `editCategory(${id})`, `deleteCategory(${id})`);
        }
      },
    ],
  ...
}

```

Fase: Interacción Frontend–Backend

- AJAX con jQuery
 - Añadir al carrito
 - Actualizar cantidades
 - Cambiar estado de pedidos
 - Gestión sin recargar página

```
function addToCart(item, callback = () => { }) {
    $.ajax({
        url: "/utils/cart_utils.php",
        type: "POST",
        data: {
            "action": "add",
            "item": item
        },
        success: (data) => {
            callback(JSON.parse(data))
        }
    });
}

function deleteFromCart(productId, callback = () => { }) {
    $.ajax({
        url: "/utils/cart_utils.php",
        type: "POST",
        data: {
            "action": "delete",
            "product_id": productId
        },
        success: (data) => {
            callback(JSON.parse(data))
        }
    });
}
```

- Validaciones
 - Frontend:
 - Campos obligatorios
 - Regex en inputs
 - Backend:
 - Validación real
 - *prepare* en query sql

```
function Login($pdo)
{
    $email = trim($_POST['email'] ?? '');
    $password = $_POST['password'] ?? '';
    if (!$email || !$password) {
        return json_encode(["success" => false, "message" => "Datos incompletos"]);
    }
    $stmt = $pdo->prepare("SELECT * FROM customers WHERE email = ?");
    $stmt->execute([$email]);
    $customer = $stmt->fetch();

    if (!$customer || !password_verify($password, $customer->password)) {
        return json_encode(["success" => false, "message" => "Cred. incorrectas"]);
    }
    $_SESSION['customer'] = [
        "id" => $customer->id,
        "name" => $customer->name,
        "last_name" => $customer->last_name,
        "email" => $customer->email
    ];
    return json_encode(["success" => true]);
}
```

Fase de Gestión de Archivos

- Imágenes de productos
 - No se guardan en BD
 - Se almacenan en /uploads
 - En BD solo:
 - id → reference to img

```
function deleteImage()
{
    deleteDir(PRODUCT_PATH . '/' . (int)$_POST['id']);
    response(true);
}

function getProductImages()
{
    $id = (int)$_POST['id'];
    response(true, '', ['images' => listImages(PRODUCT_PATH . "/$id")]);
}

function uploadShopImage()
{
    $dir = BASE_PATH . '/img/shop';
    ensureDir($dir);
    $baseName = match ($type) {
        'logo' => 'logo',
        ...
    };
    foreach (glob("$dir/$baseName.*") as $file) {
        if (is_file($file)) {
            unlink($file);
        }
    }
    $filename = "$baseName.$ext";
    move_uploaded_file($_FILES['image']['tmp_name'], "$dir/$filename");

    response(true, 'OK', [
        'url' => "/uploads/img/shop/$filename"
    ]);
}
```

Dockerizacion

Separar productos:

- Front:
 - evimerce:

```
# Imagen origen (PHP)
FROM php:8.5.1
# Extensiones necesarias
RUN docker-php-ext-install pdo pdo_mysql
# Configuración personalizada de PHP
COPY php.ini /usr/local/etc/php/php.ini
# Código de La aplicación (Código del frontend)
COPY . /usr/src/evimerce
WORKDIR /usr/src/evimerce
# Puerto HTTP
EXPOSE 80
# Servidor embebido de PHP escuchando en el puerto 80
CMD ["php", "-S", "0.0.0.0:80", "-t", "."]
```

- BD:
 - evimerce-db:

```
# Imagen origen (MySQL)
FROM mysql:8.0
# Env parameters
ENV MYSQL_DATABASE=evimerce
ENV MYSQL_USER=evimerce
ENV MYSQL_PASSWORD=evimerce
ENV MYSQL_ROOT_PASSWORD=root
# Query al iniciar contenedor por primera vez (Crear base de datos, tablas, relaciones, valores default...)
COPY init.sql /docker-entrypoint-initdb.d/
```

Montar docker container:

```
# Usamos las imágenes personalizadas (Subidas a docker hub)
services:
  app:
    image: aruger/evimerce
    container_name: evimerce
    depends_on:
      - db
    ports:
      - "80:80" # Puerto de escucha linked a host
  environment:
    # Env parameters iguales al de db
    DB_HOST: db
    DB_NAME: evimerce
    DB_USER: evimerce
    DB_PASS: evimerce

  db:
    image: aruger/evimerce-db
    container_name: evimerce-db
    volumes:
      - evimerce_db_data:/var/lib/mysql
  volumes:
    # Volumen no volátil (Guardado de cambios, base de datos, imágenes, etc)
    # persiste entre actualización de imágenes.
    evimerce_db_data:
```

9. Licencia

MIT License

Copyright (c) 2026 ArugerDev

Se concede permiso, de forma gratuita, a cualquier persona que obtenga una copia de este software y archivos de documentación asociados, para usar el software sin restricción, incluyendo sin limitación los derechos de uso, copia, modificación, fusión, publicación, distribución, sublicencia y/o venta.

El software se proporciona "tal cual", sin garantía de ningún tipo.