🧩 MODELO A — GESTIÓN DE CLIENTES (CRM BÁSICO)

🕵️ Entrevista simulada con el cliente

Cliente (consultoría / gestoría):
“Necesitamos una aplicación sencilla para gestionar nuestros clientes. Queremos un **CRM básico** donde podamos:

- Añadir, editar y borrar fichas de cliente.
- Guardar nombre, email, teléfono, dirección y una **etiqueta** (por ejemplo: *activo*, *prospecto*, *inactivo*).
- Subir una **imagen** del cliente (por ejemplo su logotipo o foto).
- La aplicación debe guardar el **nombre original** del fichero y otro nombre físico generado con un **UUID o hash** para evitar colisiones, y el archivo debe almacenarse en disco.
- Clasificar clientes por **tipo de cliente** (particular, pyme, SA, asociación, entidad pública...).
- Poder buscar clientes por nombre o email, y filtrar por tipo o etiqueta.
- Validar los datos básicos: nombre obligatorio, email válido y la imagen solo si es JPG o PNG y pesa menos de 3 MB.
- Cuando se elimine un cliente, debe eliminarse también su fichero físico asociado.”

Objetivo: Tener un CRUD funcional para clientes con subida segura de imágenes y una tabla de tipos de cliente para clasificar.

🎯 Objetivos del ejercicio

- Comprender cómo se pasa de una descripción textual (requisitos del cliente) a un **modelo UML con clases**.
- Construir una **base de datos relacional 1:N** usando claves foráneas.
- Usar **PDO** para las operaciones CRUD (Create, Read, Update, Delete).
- Aprender a manejar **subida de ficheros desde formularios** de forma segura.
- Validar y sanear datos antes de guardarlos en la base de datos.

📘 Modelo UML

Clase: Cliente

- id: int
- nombre: string
- email: string
- telefono: string
- direccion: string
- etiqueta: string
- imagen: string // nombre original del fichero
- nombre_fisico_imagen: string // nombre único (UUID o hash)
- tipo_id: int // FK → TipoCliente.id
- comentarios: string

Clase: TipoCliente

- id: int
- tipo: string // particular, pyme, SA, asociación, entidad pública...
- notas: string

Relación: 1 TipoCliente → N Cliente

📦 Esquema de Base de Datos (MySQL)

```
CREATE TABLE tipo_cliente (
 id INT AUTO_INCREMENT PRIMARY KEY,
 tipo VARCHAR(100) NOT NULL,
 notas TEXT,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE cliente (
 id INT AUTO_INCREMENT PRIMARY KEY,
```

```
 nombre VARCHAR(150) NOT NULL,
 email VARCHAR(150),
 telefono VARCHAR(30),
 direccion VARCHAR(255),
 etiqueta VARCHAR(50) DEFAULT 'activo',
 imagen VARCHAR(255),
 nombre_fisico_imagen VARCHAR(255),
 tipo_id INT,
 comentarios TEXT,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 FOREIGN KEY (tipo_id) REFERENCES tipo_cliente(id) ON DELETE SET NULL
);
```

📁 **Estructura recomendada del proyecto**

```
/crm/
 ├── config.php            // conexión PDO
 ├── models/
 │   ├── Cliente.php
 │   └── TipoCliente.php
 ├── public/
 │   ├── index.php         // lista de clientes con búsqueda y filtro
 │   ├── cliente_crear.php
 │   ├── cliente_editar.php
 │   ├── cliente_borrar.php
 │   └── uploads/clients/     // imágenes subidas
 └── templates/
     ├── header.php
     └── footer.php
```

💡 Recordatorio teórico — Subida de ficheros

Pasos para subir un fichero en PHP:

1. Formulario HTML con enctype="multipart/form-data".
2. Revisar $_FILES['campo']['error'].
3. Validar tamaño y tipo MIME real (finfo).
4. Generar un nombre físico único (random_bytes, uniqid, etc.).
5. Mover el fichero a su destino con move_uploaded_file().
6. Guardar los metadatos en BD (nombre original + físico).
7. Al borrar el registro, eliminar también el fichero físico con unlink().

🧩 Fragmentos de ejemplo (para guiar al alumno)

Formulario HTML

```
<form action="cliente_crear.php" method="post" enctype="multipart/form-data">
 <label>Nombre: <input type="text" name="nombre" required></label>
 <label>Email: <input type="email" name="email"></label>
 <label>Tipo de Cliente:
  <select name="tipo_id">
   <!-- Opciones cargadas desde la BD -->
  </select>
 </label>
 <label>Imagen: <input type="file" name="imagen" accept=".jpg,.png"></label>
 <textarea name="comentarios" placeholder="Comentarios..."></textarea>
 <button type="submit">Guardar</button>
</form>
```

Procesado de subida (esquema)

```
$tmp = $_FILES['imagen']['tmp_name'];
$original = $_FILES['imagen']['name'];
$finfo = new finfo(FILEINFO_MIME_TYPE);
$mime = $finfo->file($tmp);
```

```
// Validaciones de tipo y tamaño
$hash = bin2hex(random_bytes(16));
$ext = pathinfo($original, PATHINFO_EXTENSION);
$nombreFisico = $hash . '.' . $ext;

$dest = __DIR__ . '/../uploads/clients/' . $nombreFisico;
move_uploaded_file($tmp, $dest);
```

Inserción con PDO (fragmento)

```
$sql = "INSERT INTO cliente (nombre, email, telefono, direccion, etiqueta, imagen,
nombre_fisico_imagen, tipo_id, comentarios)
    VALUES (:nombre, :email, :telefono, :direccion, :etiqueta, :imagen,
:nombre_fisico_imagen, :tipo_id, :comentarios)";
$stmt = $pdo->prepare($sql);
$stmt->execute([
 ':nombre' => $nombre,
 ':email' => $email,
 ':telefono' => $telefono,
 ':direccion' => $direccion,
 ':etiqueta' => $etiqueta,
 ':imagen' => $original,
 ':nombre_fisico_imagen' => $nombreFisico,
 ':tipo_id' => $tipo_id,
 ':comentarios' => $comentarios
]);
```

Borrado con eliminación del fichero

```
$cliente = obtenerClientePorId($id);
$path = __DIR__ . '/../uploads/clients/' . $cliente['nombre_fisico_imagen'];
if (file_exists($path)) unlink($path);
$borrar = $pdo->prepare("DELETE FROM cliente WHERE id=:id");
$borrar->execute([':id' => $id]);
```

🧠 **Conceptos teóricos clave**

- **Relación 1:N:** un tipo de cliente puede tener muchos clientes.
- **Integridad referencial:** la FK asegura consistencia entre tablas.
- **Seguridad:** validar MIME real y tamaño antes de mover ficheros.
- **UUID/hash:** evita que dos imágenes distintas tengan el mismo nombre.
- **Transacciones:** garantizan que los datos y archivos se mantengan sincronizados.

🏀 MODELO B — CATÁLOGO DE PRODUCTOS DEPORTIVOS

🕵️ Entrevista simulada con el cliente

Cliente (tienda de artículos deportivos):
“Queremos un pequeño sistema para administrar nuestro catálogo de productos deportivos.
Cada producto debe tener un **nombre**, **SKU único**, **precio**, **talla**, **peso**, **descripción** y una **imagen** principal que subiremos desde el formulario.
Además, necesitamos clasificar los productos por **categoría** (calzado, ropa, accesorios, máquinas, etc.), y poder crear y editar esas categorías.

- Queremos validar que el SKU no se repita y que el precio sea mayor o igual a 0.
- Si se sube una imagen, debe ser JPG o PNG y no superar los 2 MB.
- Al eliminar un producto, debe borrarse también su imagen.
- Más adelante añadiremos más imágenes por producto, pero por ahora solo una.
- Nos gustaría ver una miniatura de la imagen en el listado general (no hace falta implementarlo completamente, solo explicarlo).”

Objetivo: Crear un CRUD para productos con imágenes y relación 1:N con categorías.

🎯 Objetivos del ejercicio

- Diseñar una base de datos con dos tablas relacionadas 1:N (Categoria → Producto).
- Implementar un CRUD con **PDO** para ambas tablas.
- Aplicar correctamente la **subida de imágenes con validación**.
- Comprender cómo generar **miniaturas** de imágenes (explicación conceptual).
- Asegurar consistencia entre ficheros físicos y registros de BD.

📘 Modelo UML

Clase: **Producto**

- id: int

- nombre: string

- sku: string

- precio: decimal

- talla: string

- peso: string

- descripcion: string

- imagen: string

- nombre_fisico_imagen: string

- categoria_id: int

Clase: **Categoria**

- id: int

- tipo: string

- notas: string

Relación: 1 Categoria → N Producto

📦 **Esquema de Base de Datos (MySQL)**

```
CREATE TABLE categoria (
 id INT AUTO_INCREMENT PRIMARY KEY,
 tipo VARCHAR(100) NOT NULL,
 notas TEXT,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE producto (
 id INT AUTO_INCREMENT PRIMARY KEY,
 nombre VARCHAR(150) NOT NULL,
 sku VARCHAR(50) NOT NULL UNIQUE,
 precio DECIMAL(10,2) NOT NULL DEFAULT 0.00,
 talla VARCHAR(50),
 peso VARCHAR(50),
 descripcion TEXT,
 imagen VARCHAR(255),
```

```
 nombre_fisico_imagen VARCHAR(255),
 categoria_id INT,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 FOREIGN KEY (categoria_id) REFERENCES categoria(id) ON DELETE SET NULL
);
```

🗂️ Estructura recomendada del proyecto

```
/catalogo/
 ├── config.php
 ├── models/
 │   ├── Producto.php
 │   └── Categoria.php
 ├── public/
 │   ├── productos_index.php
 │   ├── producto_crear.php
 │   ├── producto_editar.php
 │   ├── producto_borrar.php
 │   └── uploads/products/
 │       ├── originals/
 │       └── thumbs/
 └── templates/
     ├── header.php
     └── footer.php
```

💡 Recordatorio teórico — Miniaturas (concepto)

1. Cargar la imagen original (imagecreatefromjpeg o imagecreatefrompng).
2. Calcular las nuevas dimensiones manteniendo la proporción.
3. Crear un nuevo lienzo con imagecreatetruecolor().
4. Copiar y redimensionar con imagecopyresampled().

5. Guardar en una carpeta thumbs/.

No es necesario implementarlo completamente en este ejercicio, pero **explicar el proceso** ayuda a comprender cómo optimizar imágenes en aplicaciones reales.

🧩 Fragmentos de código ilustrativos

Formulario HTML

```
<form action="producto_crear.php" method="post" enctype="multipart/form-data">
  <label>Nombre: <input type="text" name="nombre" required></label>
  <label>SKU: <input type="text" name="sku" required></label>
  <label>Precio: <input type="number" step="0.01" name="precio" required></label>
  <label>Talla: <input type="text" name="talla"></label>
  <label>Peso: <input type="text" name="peso"></label>
  <label>Categoría:
    <select name="categoria_id">
      <!-- Opciones desde BD -->
    </select>
  </label>
  <label>Imagen: <input type="file" name="imagen" accept=".jpg,.png"></label>
  <button type="submit">Guardar</button>
</form>
```

Procesamiento de imagen

```
if ($_FILES['imagen']['error'] === UPLOAD_ERR_OK) {
    $tmp = $_FILES['imagen']['tmp_name'];
    $original = $_FILES['imagen']['name'];
    $finfo = new finfo(FILEINFO_MIME_TYPE);
    $mime = $finfo->file($tmp);

    $hash = bin2hex(random_bytes(16));
    $ext = pathinfo($original, PATHINFO_EXTENSION);
```

```
  $nombreFisico = $hash . '.' . $ext;


  move_uploaded_file($tmp, __DIR__ . '/../uploads/products/originals/' . $nombreFisico);
}
```

Inserción con PDO

```
$sql = "INSERT INTO producto (nombre, sku, precio, talla, peso, descripcion, imagen, nombre_fisico_imagen, categoria_id)
    VALUES (:nombre, :sku, :precio, :talla, :peso, :descripcion, :imagen, :nombre_fisico_imagen, :categoria_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute([
 ':nombre' => $nombre,
 ':sku' => $sku,
 ':precio' => $precio,
 ':talla' => $talla,
 ':peso' => $peso,
 ':descripcion' => $descripcion,
 ':imagen' => $original,
 ':nombre_fisico_imagen' => $nombreFisico,
 ':categoria_id' => $categoria_id
]);
```

🧠 **Conceptos teóricos clave**

- **Relación 1:N:** una categoría agrupa varios productos.
- **Claves únicas:** el SKU garantiza que no haya duplicados.
- **Validación de precios y tipos:** el tipo decimal requiere control.
- **Validación MIME:** más fiable que la extensión del fichero.
- **Gestión de imágenes:** guardar en disco, registrar metadatos, y borrar al eliminar.
- **Seguridad:** siempre escapar salidas HTML (htmlspecialchars()) y usar sentencias preparadas (prepare() / execute()).