

Documentación de uso de herramienta de IA (ChatGPT)

Daniel León Ramírez

19 de agosto de 2025

Índice

1. Introducción	2
2. Base de datos	2
2.1. Interacción 1	2
2.2. Interacción 2	4
2.3. Interacción 3	4
3. Backend	6
3.1. Interacción 1	6
3.2. Interacción 2	9
4. Frontend	10
4.1. Interacción 1	10
4.2. Interacción 2	11
4.3. Interacción 3	11
4.4. Interacción 4	12
4.5. Interacción 5	13
4.6. Interacción 6	14
4.7. Interacción 7	15
4.8. Interacción 8	15
4.9. Interacción 9	17
5. Uso de la herramienta de IA en el desarrollo	19

1. Introducción

En este documento se recoge el uso de una herramienta de Inteligencia Artificial (ChatGPT) para el desarrollo de una prueba técnica en software. Se documentan los *prompts* utilizados y las respuestas obtenidas de la IA, de acuerdo con los requisitos de transparencia en el uso de herramientas basadas en IA.

2. Base de datos

2.1. Interacción 1

Prompt del usuario

Voy a realizar una prueba tecnica de software usando php, mysql 8 o superior, react y tailwind. El dominio de la aplicación es el siguiente: Cliente: con un nombre arbitrario y un identificador textual único que no puede repetirse en el sistema para uso interno. Producto de cliente: un nombre que es único por cliente. Relación que hay: Los productos tienen un atributo principal “nombre”. Puede darse la circunstancia de que distintos clientes vendan el mismo producto con nombres diferentes. Por ejemplo, Worten puede vender “Monitor DELL UltraSharp 24” y MediaMarkt “Pantalla UltraSharp Dell 24”, tratándose realmente del mismo producto. De ahí que se quiera tener reflejada una relación entre productos de forma que podamos vincular unos productos con otros, significando “estos productos son el mismo”. Las condiciones son: 1º Un producto no puede estar relacionado con (“ser el mismo que”) otro del mismo cliente. A un cliente sólo le interesa conocer los productos que “son el mismo” en otros clientes. Es decir: no se permite que productos de un mismo cliente estén relacionados entre sí como iguales. 2º Un producto de un cliente puede “ser el mismo” que uno o más de otro(s) cliente(s). Funcionalidad: Primero vamos hacer la base de datos: Crear un schema. Modelo de tablas, relaciones, etc. (el esquema de la BDD) de acuerdo a lo explicado arriba como requisitos mínimos. Script para pre- “rellenar” 10 clientes. El “output” debería ser un archivo SQL con todo lo necesario.

Respuesta de ChatGPT

Tabla clients

```
CREATE TABLE clients (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  name VARCHAR(100) NOT NULL,
```

```

        identifier VARCHAR(50) NOT NULL UNIQUE
    );

```

Tabla client_products

```

CREATE TABLE client_products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    client_id INT NOT NULL,
    name VARCHAR(255) NOT NULL,
    CONSTRAINT fk_client FOREIGN KEY (client_id) REFERENCES
        ↪ clients(id) ON DELETE CASCADE,
    CONSTRAINT uq_client_product UNIQUE (client_id, name)
);

```

Tabla product_relation

```

CREATE TABLE product_relations (
    product_a INT NOT NULL,
    product_b INT NOT NULL,
    PRIMARY KEY (product_a, product_b),
    CONSTRAINT fk_product_a FOREIGN KEY (product_a) REFERENCES
        ↪ client_products(id) ON DELETE CASCADE,
    CONSTRAINT fk_product_b FOREIGN KEY (product_b) REFERENCES
        ↪ client_products(id) ON DELETE CASCADE,
    CONSTRAINT chk_different CHECK (product_a < product_b)
    -- Nota: restricción de "no mismo cliente" se debe validar vía
    ↪ aplicación o trigger
);

```

Tabla Insertar 10 clientes

```

INSERT INTO clients (name, identifier) VALUES
('Worten', 'worten'),
('MediaMarkt', 'mediamarkt'),
('Carrefour', 'carrefour'),
('Amazon', 'amazon'),
('El Corte Inglés', 'eci'),
('PCComponentes', 'pccomponentes'),
('Fnac', 'fnac'),
('Alcampo', 'alcampo'),
('Eroski', 'eroski'),
('Sprinter', 'sprinter');

```

2.2. Interacción 2

Prompt del usuario

La relación de product_relations la prefiero con un identificador único propio.

Respuesta de ChatGPT

Tabla product_relation

```
CREATE TABLE product_relations (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  product_a INT NOT NULL,  
  product_b INT NOT NULL,  
  CONSTRAINT fk_product_a FOREIGN KEY (product_a) REFERENCES  
    ↳ client_products(id) ON DELETE CASCADE,  
  CONSTRAINT fk_product_b FOREIGN KEY (product_b) REFERENCES  
    ↳ client_products(id) ON DELETE CASCADE,  
  CONSTRAINT uq_relation UNIQUE (product_a, product_b),  
  CONSTRAINT chk_different CHECK (product_a < product_b)  
  -- Nota: la validación de "no mismo cliente" se controla en  
    ↳ aplicación o trigger  
);
```

2.3. Interacción 3

Prompt del usuario

Vale, vamos a crear el trigger para product_relations para validar que no es el mismo cliente.

Respuesta de ChatGPT

Tabla Triggers para product_relations

```
DELIMITER $$  
  
-- Trigger BEFORE INSERT  
CREATE TRIGGER trg_product_relations_before_insert  
BEFORE INSERT ON product_relations  
FOR EACH ROW  
BEGIN
```

```

DECLARE client_a INT;
DECLARE client_b INT;

SELECT client_id INTO client_a
FROM client_products
WHERE id = NEW.product_a;

SELECT client_id INTO client_b
FROM client_products
WHERE id = NEW.product_b;

IF client_a = client_b THEN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se permite relacionar productos
        ↳ del mismo cliente';
END IF;
END$$

-- Trigger BEFORE UPDATE
CREATE TRIGGER trg_product_relations_before_update
BEFORE UPDATE ON product_relations
FOR EACH ROW
BEGIN
    DECLARE client_a INT;
    DECLARE client_b INT;

    SELECT client_id INTO client_a
    FROM client_products
    WHERE id = NEW.product_a;

    SELECT client_id INTO client_b
    FROM client_products
    WHERE id = NEW.product_b;

    IF client_a = client_b THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'No se permite relacionar productos
            ↳ del mismo cliente';
    END IF;
END$$

DELIMITER ;

```

3. Backend

3.1. Interacción 1

Prompt del usuario

Vale ahora vamos a ir con la segunda parte parte que es crear los ficheros php:

Web PHP:

- Un formulario para introducir productos (finalidad: tener productos en la BBDD) para un cliente seleccionable. Un POST clásico en interacción formulario web. Que contemple los requisitos expuestos también en el apartado del Modelo de Dominio.
- Una página que permita seleccionar un cliente y un producto (en ese orden) y, para ese producto, establecer nuevas relaciones con “mismos productos” en otros clientes.
 - Una vez seleccionado el producto, con un cuadro de texto o similar podremos introducir texto arbitrario.
 - Al introducir 3 o más caracteres se ha de lanzar una búsqueda por productos que puedan contener este texto en el nombre.
 - Los resultados han de mostrarse en una vista tipo tabla o lista en la que como mínimo aparezca el nombre del producto y el cliente al que pertenecen.
 - Junto con el display de cada uno de los productos “candidatos” existe un botón o similar con el que puedo confirmar que, efectivamente, el producto propuesto “es el mismo” que el que he seleccionado de partida, estableciendo así una relación de igualdad entre los productos.

Ten en cuenta que voy a usar PHP + React.

Respuesta de ChatGPT

Tabla Fichero PHP de configuración de BD

config.php

```
<?php
$host = 'localhost';
$db   = 'product_matching';
$user = 'root';
```

```

$pass = '';
$charset = 'utf8mb4';

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
];

try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e->getCode());
}
?>

```

Tabla Fichero PHP para add_product

```

api/add_product.php
<?php
require_once '../config.php';
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405);
    exit('Method not allowed');
}
$data = json_decode(file_get_contents('php://input'), true);
$client_id = $data['client_id'] ?? null;
$product_name = trim($data['name'] ?? '');
if (!$client_id || !$product_name) {
    http_response_code(400);
    exit('Missing parameters');
}
try {
    $stmt = $pdo->prepare("INSERT INTO client_products (client_id, name)
VALUES (?, ?)");
    $stmt->execute([$client_id, $product_name]);
    echo json_encode(['success' => true, 'id' => $pdo->lastInsertId()]);
} catch (\PDOException $e) {
    http_response_code(500);
    echo json_encode(['success' => false, 'error' => $e->getMessage()]);
}
?>

```

Tabla Fichero PHP para search_product

```
api/search_products.php
<?php
require_once '../config.php';
$query = $_GET['query'] ?? '';
$exclude_client = $_GET['exclude_client'] ?? 0;
if (strlen($query) < 3) {
    echo json_encode([]);
    exit;
}
$stmt = $pdo->prepare("
    SELECT cp.id, cp.name, c.name AS client_name
    FROM client_products cp
    JOIN clients c ON c.id = cp.client_id
    WHERE cp.name LIKE ? AND cp.client_id != ?
    ORDER BY cp.name
    LIMIT 20
");
$stmt->execute(['%' . $query . '%', $exclude_client]);
$results = $stmt->fetchAll();
echo json_encode($results);
?>
```

Tabla Fichero PHP para add_relation

```
api/add_relation.php
<?php
require_once '../config.php';
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    http_response_code(405);
    exit('Method not allowed');
}
$data = json_decode(file_get_contents('php://input'), true);
$product_a = $data['product_a'] ?? null;
$product_b = $data['product_b'] ?? null;
if (!$product_a || !$product_b) {
    http_response_code(400);
    exit('Missing parameters');
}
try {
    $stmt = $pdo->prepare("INSERT INTO product_relations (product_a, product_b)
    VALUES (?, ?)");
    $stmt->execute([$product_a, $product_b]);
}
```



```

        echo json_encode(['success' => true]);
    } catch (PDOException $e) {
        http_response_code(500);
        echo json_encode(['success' => false, 'error' => $e->getMessage()]);
    }
?>

```

3.2. Interacción 2

Prompt del usuario

Vale además de esto también necesito otras dos consultas para hacer un `getClient`s y luego según un cliente quiero todos sus productos `getProducts`, los nombres los dejo a tu elección.

Respuesta de ChatGPT

Tabla Fichero PHP para `get_clients`

```

api/get_clients.php
<?php
require_once '../config.php';

$stmt = $pdo->query("SELECT id, name, unique_identifier FROM clients
ORDER BY name");
$clients = $stmt->fetchAll();

echo json_encode($clients);
?>

```

Tabla Fichero PHP para `get_client_products`

```

api/get_client_products.php
<?php
require_once '../config.php';

$client_id = $_GET['client_id'] ?? null;

if (!$client_id) {
    http_response_code(400);
    exit('Missing client_id');
}

```

```
$stmt = $pdo->prepare("SELECT id, name FROM client_products WHERE client_id = ?  
ORDER BY name");  
$stmt->execute([$client_id]);  
$products = $stmt->fetchAll();  
  
echo json_encode($products);  
?>
```

4. Frontend

4.1. Interacción 1

Prompt del usuario

Vale a continuación vamos hacer el diseño del frontend utilizando react + tailwind, para esto tengo un diseño predefinido de figma. Como tengo el diseño de figma tengo todos los iconos descargados utilizados para esto. Te voy a ir pasando las imagenes de los componentes y tu traducelo a react + tailwind.

Respuesta de ChatGPT

Entonces el flujo sería así:

Tú me pasas las imágenes/capturas del Figma (los componentes que quieras implementar).

Yo traduzco ese diseño directamente a React + Tailwind siguiendo las buenas prácticas:

- Uso de componentes reutilizables.
- Flex/Grid según el layout de la maqueta.
- Iconos: si ya tienes SVGs/PNGs descargados del Figma, yo los incorporo como `` o inline-SVG.
- Tipografías, colores, paddings y espaciados reproducidos lo más fielmente posible.

Tú vas montando poco a poco la aplicación hasta completar todas las vistas.

Pásame la primera imagen de un componente del Figma y lo traduzco en código React + Tailwind.

¿Quieres empezar con el formulario de inclusión de productos o con la pantalla de relaciones de productos?

4.2. Interacción 2

Prompt del usuario

Imagen del header de figma. Lo primero es el header de la aplicación.

Respuesta de ChatGPT

```
// header.jsx
import { useState } from "react";
import { Search, Filter, Bell } from "lucide-react";

export default function Header() {
  const [filterOn, setFilterOn] = useState(true);

  return (
    /* Resto del código */
  );
}
```

4.3. Interacción 3

Prompt del usuario

no utilices librerías tengo los iconos

Respuesta de ChatGPT

```
// header.jsx
export default function Header() {
  return (
    /* Search Bar */
    <div className="flex items-center border rounded-lg overflow-hidden">
      <input
        type="text"
        placeholder="Write text..."
      />
    </div>
  );
}
```

```

        className="px-4 py-2 outline-none w-64"
      />
      <button className="...">
        <span>Search</span>
        
      </button>
    </div>

    { /* Resto del código */ }
  );
}

```

4.4. Interacción 4

Prompt del usuario

puedes crearme un sidebar menu vertical usando tailwind con aside, ten en cuenta que tiene un nav con 5 iconos que tengo descargados y los introduzco yo su src simplemente hazme la plantilla. Ten en cuenta estas propiedades: Disposicion: width: 67.51446533203125; height: 4092.051025390625; angle: 0 deg; opacity: 1; border-right-width: 1px; padding-top: 5px; padding-bottom: 5px; Colores: background: var(-GreyScale-White, #FFFFFF); Bordes: border-right: 1px solid var(-color-border-surface-light, #C9C9C9)

Respuesta de ChatGPT

```

// Sidebar.jsx
function SideBar() {
  return (
    <aside
      className="flex flex-col items-center"
      style={{
        width: '67.51446533203125px',
        height: '4092.051025390625px',
        transform: 'rotate(0deg)',
        opacity: 1,
        borderRight: '1px solid #C9C9C9',
        paddingTop: '5px',
        paddingBottom: '5px',
        backgroundColor: '#FFFFFF',
      }}
    >

```

```

>
<nav className="flex flex-col gap-6 mt-5">
  {/* Icono 1 */}
  

  {/* Icono 2 */}
  

  {/* Icono 3 */}
  

  {/* Icono 4 */}
  

  {/* Icono 5 */}
  
</nav>
</aside>
);
}

export default SideBar;

```

4.5. Interacción 5

Prompt del usuario

vale ahora vamos a crear un footer donde tenga el texto de Prueba técnica Minderest 2025 Y tenga el siguiente estilo: font-family: Roboto; font-weight: 400; font-size: 12px; leading-trim: NONE; line-height: 100%; letter-spacing: 0.17px; Utiliza tailwind y react para crear el componente

Respuesta de ChatGPT

```

// Footer.jsx
function Footer() {
  return (
    <footer className="w-full bg-white border-t border-gray-300 py-2
flex justify-center">
      <p className="font-['Roboto'] font-normal text-[12px] leading-[100%]
tracking-[0.17px] text-gray-700">

```

```

        Prueba técnica Minderest 2025
      </p>
    </footer>
  );
}

export default Footer;

```

4.6. Interacción 6

Prompt del usuario

ahora tengo que crear el componente que es un formulario para añadir productos de cliente que tiene los siguientes apartados un selector de cliente, despues debajo del selector de cliente se encuentra dos input, el primero para meter el nombre del producto y el segundo para el meter el precio pon dos placeholder y debajo esta el boton de añadir este producto

Respuesta de ChatGPT

```

// AddProductForm.jsx
import { useState } from "react";

function AddProductForm() {
  const [cliente, setCliente] = useState("");
  const [producto, setProducto] = useState("");
  const [precio, setPrecio] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault();
    // Aquí puedes manejar el envío del producto
    console.log({ cliente, producto, precio });
    // Limpiar formulario
    setProducto("");
    setPrecio("");
  };

  {/* Resto del código */}
}

```

```
export default AddProductForm;
```

4.7. Interacción 7

Prompt del usuario

vale vamos a conectar la parte de seleccionar cliente con el backend de php la ruta a la api es la siguiente: Es un get a `http://localhost/backend/api/get_clients.php`

Respuesta de ChatGPT

```
// AddProductForm.jsx
const [clientes, setClientes] = useState([]);
// Traer clientes desde la API al cargar el componente
useEffect(() => {
  fetch("http://localhost/backend/api/get_clients.php")
    .then((res) => res.json())
    .then((data) => {
      setClientes(data);
    })
    .catch((err) => console.error("Error fetching clients:", err));
}, []);

<option value="">-- Elige un cliente --</option>
{clientes.map((c) => (
  <option key={c.id} value={c.id}>
    {c.nombre}
  </option>
))}
}
```

4.8. Interacción 8

Prompt del usuario

vale ahora vamos hacer la funcionalidad del boton de añadir producto que va hacer un post a `http://localhost/backend/api/add_product.php` donde tiene el cuerpo de un json: `{ "client_id": 4, "name": "HP Pavilion 15 Notebook" }` donde el `client_id` lo sacas del selector y el `name` es el nombre del producto

Respuesta de ChatGPT

```
// AddProductForm.jsx
const [loading, setLoading] = useState(false);
const [message, setMessage] = useState("");
const handleSubmit = async (e) => {
  e.preventDefault();
  if (!cliente || !producto) {
    setMessage("Selecciona un cliente y escribe un nombre de producto.");
    return;
  }

  setLoading(true);
  setMessage("");

  try {
    const response = await fetch("http://localhost/backend/api/add_product.php", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        client_id: parseInt(cliente),
        name: producto,
      }),
    });

    const data = await response.json();

    if (response.ok) {
      setMessage("Producto añadido correctamente ");
      setProducto("");
      setPrecio(""); // opcional si quieres limpiar también el precio
    } else {
      setMessage("Error al añadir el producto: " + data.message);
    }
  } catch (error) {
    console.error(error);
    setMessage("Error al conectar con el servidor.");
  } finally {
    setLoading(false);
  }
};
{/* Resto del código */}
```



```
}
```

4.9. Interacción 9

Prompt del usuario

Vamos hacer un nuevo componente para hacer la búsqueda y poder hacer las relaciones entre productos. Para ello necesitamos: Primera linea otro selector de clientes y al lado en la misma linea otro selector de productos de ese cliente, debajo va a ver un input para buscar y luego debajo del input saldrán los resultados de búsqueda en un tipo tabla que contiene el nombre del producto, el nombre del cliente y después un botón para crear la relación

Respuesta de ChatGPT

```
// ProductRelation.jsx
import { useState, useEffect } from "react";

function ProductRelation() {
  const [clientes, setClientes] = useState([]);
  const [clienteSeleccionado, setClienteSeleccionado] = useState("");
  const [productosCliente, setProductosCliente] = useState([]);
  const [productoSeleccionado, setProductoSeleccionado] = useState("");
  const [search, setSearch] = useState("");
  const [results, setResults] = useState([]);
  // Cargar clientes al inicio
  useEffect(() => {
    fetch("http://localhost/backend/api/get_clients.php")
      .then((res) => res.json())
      .then((data) => setClientes(data))
      .catch((err) => console.error("Error fetching clients:", err));
  }, []);

  // Cargar productos del cliente seleccionado
  useEffect(() => {
    if (!clienteSeleccionado) return;
    fetch(`http://localhost/backend/api/get_products.php?client_id=${clienteSeleccionado}`)
      .then((res) => res.json())
      .then((data) => setProductosCliente(data))
      .catch((err) => console.error("Error fetching products:", err));
  }, [clienteSeleccionado]);
}
```

```

}, [clienteSeleccionado]);

// Simular búsqueda
const handleSearch = async () => {
  if (!search.trim()) return;

  try {
    const res = await fetch(
      `http://localhost/backend/api/search_products.php?query=${encodeURIComponent(search)}`
    );
    const data = await res.json();
    setResults(data);
  } catch (err) {
    console.error("Error en la búsqueda:", err);
  }
};

// Crear relación
const handleCreateRelation = async (targetProductId) => {
  try {
    const res = await fetch("http://localhost/backend/api/create_relation.php", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        client_id: clienteSeleccionado,
        product_id: productoSeleccionado,
        related_product_id: targetProductId,
      }),
    });

    const data = await res.json();
    if (res.ok) {
      alert("Relación creada correctamente ");
    } else {
      alert("Error al crear relación: " + data.message);
    }
  } catch (err) {
    console.error(err);
    alert("Error de conexión con el servidor.");
  }
}

```

```
};  
return (  
    {/* Resto del código */}  
)  
}
```

5. Uso de la herramienta de IA en el desarrollo

Durante el desarrollo de la aplicación, se utilizó una herramienta de inteligencia artificial como asistente en diferentes etapas del proyecto, incluyendo el diseño de la base de datos, la implementación del backend y la creación del frontend. En cada caso, la IA se empleó como apoyo para generar propuestas de código, scripts y estructuras de componentes, mientras que el usuario supervisaba, evaluaba y adaptaba los resultados para garantizar que cumplieran con los requisitos funcionales y el diseño definido.

El proceso fue iterativo: se enviaron múltiples prompts describiendo la funcionalidad deseada o los elementos a implementar, se recibieron respuestas generadas por la IA y se realizaron ajustes o correcciones según las necesidades del proyecto y el conocimiento del desarrollador.

En la fase de diseño de la base de datos, la IA ayudó a generar un modelo en MySQL 8, incluyendo la creación de tablas, definición de restricciones, triggers para validaciones complejas y precarga de datos de ejemplo. Esta colaboración permitió explorar distintas alternativas de diseño y acelerar la generación de un script SQL completo, siempre bajo supervisión para asegurar su coherencia con los requisitos funcionales.

Durante la implementación del backend en PHP, la IA proporcionó ejemplos de endpoints y estructuras de código, incluyendo la conexión a la base de datos mediante PDO, consultas SQL preparadas, manejo de errores y generación de respuestas en formato JSON. Cada propuesta fue evaluada y adaptada por el desarrollador para garantizar consistencia con el modelo de datos y con la interacción prevista desde el frontend. Esto permitió acelerar la creación de los archivos funcionales que forman la base del backend de la aplicación.

En el desarrollo del frontend con React y Tailwind, la IA ayudó a estructurar componentes y generar código adaptado a los requisitos funcionales y al diseño definido en Figma. Se crearon elementos como menús laterales, pies de página y formularios para la gestión de productos de clientes, siguiendo

estilos y medidas establecidos. Además, se implementó la lógica de conexión con el backend, gestionando consultas, envíos de datos en formato JSON y actualización dinámica de la interfaz. Cada propuesta de código fue evaluada, ajustada y validada por el desarrollador para asegurar la coherencia con el modelo de datos y los flujos de interacción previstos.

En todos los casos, la colaboración con la IA permitió acelerar significativamente el desarrollo, explorar alternativas de implementación y mantener un control exhaustivo sobre la calidad y la adecuación de los resultados, asegurando que cada componente funcionara correctamente dentro del sistema completo.