



**Facultad de Economía,  
Empresa y Turismo**  
Universidad de La Laguna

Proyecto final de la asignatura:

Product-E-Match

Administración y Diseño de

Base de Datos

Roberto Jesús Pérez Medina



# ÍNDICE

<b>ÍNDICE.....</b>	<b>1</b>
<b>Objetivos del proyecto.....</b>	<b>3</b>
Gestión Centralizada de Inventarios:.....	3
Flexibilidad en la Personalización:.....	3
Interfaz Intuitiva para Usuarios:.....	3
Optimización de Procesos de Gestión:.....	3
Escalabilidad y Adaptación:.....	3
Accesibilidad desde Múltiples Dispositivos:.....	4
Optimización de la Interacción con la Base de Datos:.....	4
Pruebas y Fiabilidad del Sistema:.....	4
<b>Especificación de los Requisitos para la Base de Datos.....</b>	<b>4</b>
Requisitos funcionales.....	4
Requisitos de Modelado.....	5
<b>Diseño conceptual.....</b>	<b>6</b>
Modelo Entidad-Relación (E-R).....	6
Entidades.....	6
Relaciones.....	8
Supuestos Semánticos Complementarios.....	9
<b>Modelo Relacional.....</b>	<b>10</b>
<b>Base de datos.....</b>	<b>11</b>
<b>Carga de datos de las tablas.....</b>	<b>17</b>
<b>Consultas de ejemplo.....</b>	<b>18</b>
1. Consultas Básicas.....	18
1.1. Obtener todas las camisetas con su equipo asociado.....	18
1.2. Mostrar los clientes y el número de pedidos realizados.....	18
2. Consultas con Filtros.....	18
2.1. Buscar camisetas de un equipo específico.....	18
2.2. Mostrar pedidos realizados en un rango de fechas.....	19
3. Consultas con Agregaciones.....	19
3.1. Obtener el stock total de camisetas por equipo.....	19
3.2. Calcular el precio total de un pedido.....	19
4. Consultas con Relaciones Complejas.....	19
4.1. Mostrar todas las camisetas personalizadas con sus parches.....	19
5. Consultas de Integridad y Pruebas de Restricciones.....	20
5.1. Camisetas sin personalización.....	20



5.2. Clientes sin pedidos.....	20
6. Consultas de Unión y Subconsultas.....	20
6.1. Lista de todos los equipos (clubes y selecciones).....	20
6.2. Subconsulta: Pedidos con más de 2 camisetas.....	20
<b>Diseño de la API REST.....</b>	<b>21</b>
Propósito del API REST.....	21
Tecnologías Utilizadas.....	21
Características del API REST.....	22
Estructura del API.....	22
Justificación del Desarrollo.....	22
Presupuesto del Proyecto.....	27
1. Desarrollo.....	27
2. Infraestructura.....	27
3. Mantenimiento y Soporte.....	28
4. Presupuesto Total.....	28
Conclusión.....	29
Logros del Proyecto.....	29
Aprendizajes y Desafíos.....	29
Futuras Extensiones.....	30
Conclusión Final.....	30



## Objetivos del proyecto

El propósito principal del proyecto es desarrollar un sistema de gestión de camisetas de fútbol que facilite el manejo de inventarios, personalización y visualización de productos para su venta o distribución. Este sistema tiene como meta satisfacer las siguientes necesidades:

### Gestión Centralizada de Inventarios:

Permitir el registro, visualización, actualización y eliminación de camisetas deportivas en un entorno centralizado.

Facilitar el almacenamiento de información detallada, como temporada, categoría, talla, equipo asociado y personalización opcional.

### Flexibilidad en la Personalización:

Proporcionar la opción de agregar personalizaciones a las camisetas, como parches o nombres de jugadores, sin hacer obligatorio este paso, permitiendo la venta de camisetas genéricas o personalizadas.

### Interfaz Intuitiva para Usuarios:

Diseñar una interfaz web simple y fácil de usar que permita a los usuarios interactuar con el sistema mediante opciones como añadir, editar, visualizar y eliminar camisetas.

Ofrecer una experiencia visual atractiva con tablas, formularios y botones claramente identificados para cada acción.

### Optimización de Procesos de Gestión:

Simplificar el flujo de trabajo de los administradores al automatizar tareas como la validación de entradas y la asociación de camisetas con equipos o personalizaciones existentes en la base de datos.

### Escalabilidad y Adaptación:

Diseñar la base de datos y la aplicación para que puedan escalar en caso de que se requiera manejar un volumen mayor de camisetas, equipos o configuraciones personalizadas.

Asegurar que las relaciones entre tablas sean claras y manejables, permitiendo futuras integraciones con otros sistemas o módulos.



## Accesibilidad desde Múltiples Dispositivos:

Implementar el proyecto como una aplicación web accesible desde cualquier dispositivo con conexión a internet, facilitando la gestión remota del inventario.

## Optimización de la Interacción con la Base de Datos:

Proveer consultas eficientes para buscar, filtrar y manipular datos relacionados con camisetas, equipos y personalizaciones, asegurando un rendimiento adecuado del sistema.

## Pruebas y Fiabilidad del Sistema:

Garantizar que el sistema sea robusto mediante pruebas exhaustivas en las operaciones CRUD (Crear, Leer, Actualizar y Eliminar).

Minimizar errores como conflictos en claves foráneas, valores nulos no deseados o entradas inválidas.

# Especificación de los Requisitos para la Base de Datos

El contexto de esta base de datos está basado en un sistema de gestión integral para la venta, distribución y personalización de camisetas deportivas. El sistema busca satisfacer las necesidades de un negocio dedicado a la comercialización de camisetas asociadas a equipos de fútbol, tanto clubes como selecciones nacionales, permitiendo el manejo de pedidos de clientes y distribuidores, así como la personalización de las prendas.

## Requisitos funcionales

1. Gestión de Equipos y Camisetas:
  - Registrar información de los equipos, diferenciando entre clubes y selecciones nacionales.
  - Cada equipo debe tener al menos un escudo y una categoría de equipación (primera, segunda, tercera).
  - Relación 1:N entre los equipos y las camisetas que producen.
  - Incorporar datos detallados de las camisetas como temporada, categoría, talla, y stock.
2. Gestión de Jugadores:
  - Los jugadores deben estar asociados a un equipo y registrar su dorsal y nombre.
  - Existe una relación N:M entre los equipos y los jugadores, ya que un jugador puede haber pertenecido a varios equipos a lo largo de su carrera.



3. Personalización de Camisetas:

Cada camiseta puede ser personalizada con parches opcionales.

Relación débil entre personalización y parches, ya que una personalización depende de la existencia de al menos un parche.

Restricción de hasta 4 parches por camiseta, permitiendo personalización parcial o sin ella.

4. Pedidos y Clientes:

Gestionar pedidos realizados por clientes, incluyendo información de fecha, estado del pedido y duración estimada del envío.

Un cliente puede realizar múltiples pedidos (relación 1:N).

Cada pedido puede contener una o más camisetas, y debe registrar la cantidad y el precio de cada producto.

Los pedidos de clientes pueden estar vinculados a descuentos opcionales y registrar un número total de compras realizadas por el cliente.

5. Distribuidores y Pedidos Mayoristas:

Registrar distribuidores con información de contacto, nombre de la compañía, y su IBAN para transacciones.

Los distribuidores pueden realizar pedidos de camisetas en grandes cantidades.

Estos pedidos deben incluir información sobre fecha de pago, fecha estimada de llegada y duración del envío.

6. Gestión de Imágenes:

Asociar imágenes con camisetas para mostrar una vista previa del producto.

Cada camiseta puede tener una o más imágenes relacionadas.

## Requisitos de Modelado

1. Entidades Débiles:

La entidad Personalización es débil y depende de la existencia de una camiseta.

Una personalización no puede existir sin estar asociada a una camiseta específica.

2. Relaciones M:N:

Relación entre jugadores y equipos, ya que un jugador puede pertenecer a varios equipos en diferentes periodos.

Relación entre camisetas y pedidos, ya que un pedido puede contener varias camisetas y cada camiseta puede estar en múltiples pedidos.

3. Relaciones Triples:

Relación entre el cliente, el pedido y la fecha en que el pedido fue realizado.



4. Jerarquías de Especialización (IS-A):  
Los equipos se dividen en Clubes y Selecciones Nacionales, heredando propiedades comunes como nombre y escudo, pero diferenciándose por atributos específicos como estadio y liga (en el caso de los clubes) o ranking FIFA (en el caso de las selecciones).
5. Restricciones de Inclusión, Exclusión y Exclusividad:  
Exclusión entre Club y Selección Nacional: Un equipo debe ser exclusivamente un club o una selección nacional.  
Inclusividad en los pedidos: Un pedido puede contener tanto camisetas personalizadas como genéricas, pero no se requiere que todas las camisetas sean personalizadas.
6. Relaciones 1:N:  
Entre equipos y camisetas, ya que un equipo puede producir múltiples camisetas.  
Entre clientes y pedidos, ya que un cliente puede realizar varios pedidos.

## Diseño conceptual

### Modelo Entidad-Relación (E-R)

#### Entidades

1. **Equipo**
  - Atributos:
    - id (Clave primaria)
    - nombre (Único)
    - escudo (Ruta a la imagen del escudo)
    - equipacion (Primera, Segunda, Tercera)
  - Supuesto semántico: Los equipos se dividen en **Club** y **Selección Nacional**, formando una jerarquía IS-A.
2. **Club**
  - Atributos:
    - estadio
    - liga
  - Relación **IS-A** con Equipo.
3. **Selección Nacional**
  - Atributos:
    - ranking\_fifa
  - Relación **IS-A** con Equipo.
4. **Jugador**
  - Atributos:



- id (Clave primaria)
- nombre
- dorsal
- Supuesto semántico: Un jugador puede pertenecer a varios equipos a lo largo de su carrera (relación N:M).

#### 5. Camiseta

- Atributos:
  - id (Clave primaria)
  - temporada
  - categoria (Retro, Jugador, Fan, Conjunto)
  - talla (S, M, L, XL, etc.)
  - stock
  - precio
- Relación 1:N con Equipo, ya que un equipo puede tener múltiples camisetas.

#### 6. Personalización

- Atributos:
  - id (Clave primaria)
  - precio
- Entidad débil relacionada con Camiseta (una personalización depende de la existencia de una camiseta).

#### 7. Parche

- Atributos:
  - id (Clave primaria)
  - descripcion
- Relación N:M entre Personalización y Parche.

#### 8. Cliente

- Atributos:
  - id (Clave primaria)
  - nombre
  - direccion
  - telefono
  - descuento (Opcional)
  - num\_compras (Total de compras realizadas)
- Supuesto semántico: Un cliente puede realizar múltiples pedidos (relación 1:N).

#### 9. Pedido Cliente

- Atributos:
  - id (Clave primaria)
  - estado (Enviado, Procesado, Cancelado)
  - fecha (Fecha de realización)
- Relación N:M con Camiseta (un pedido puede incluir múltiples camisetas y una camiseta puede estar en varios pedidos).

#### 10. Pedido Distribuidor





- Atributos:
  - id (Clave primaria)
  - fecha\_pago
  - fecha\_llegada
  - duracion\_envio
- Relación 1:N con Distribuidor.

#### 11. Distribuidor

- Atributos:
  - id (Clave primaria)
  - nombre
  - nombre\_compania
  - IBAN
  - contacto

#### 12. Imagen

- Atributos:
  - id (Clave primaria)
  - ruta (Ruta a la imagen)
- Relación 1:N con Camiseta.

## Relaciones

### 1. Equipo - Camiseta

- Relación: 1:N
- Un equipo puede tener múltiples camisetas asociadas a diferentes temporadas, categorías o tallas.

### 2. Jugador - Equipo

- Relación: N:M
- Un jugador puede haber estado en varios equipos, y un equipo puede tener múltiples jugadores.

### 3. Camiseta - Personalización

- Relación: 1:1
- Una camiseta puede tener una única personalización o ninguna.

### 4. Personalización - Parche

- Relación: N:M
- Una personalización puede incluir múltiples parches, y un parche puede estar en varias personalizaciones.

### 5. Cliente - Pedido Cliente

- Relación: 1:N
- Un cliente puede realizar múltiples pedidos.

### 6. Pedido Cliente - Camiseta

- Relación: N:M



- Un pedido puede incluir múltiples camisetas, y una camiseta puede estar en varios pedidos.
- 7. **Distribuidor - Pedido Distribuidor**
  - Relación: 1:N
  - Un distribuidor puede gestionar múltiples pedidos de camisetas.
- 8. **Camiseta - Imagen**
  - Relación: 1:N
  - Una camiseta puede tener múltiples imágenes asociadas (frontal, trasera, detalles).

## Supuestos Semánticos Complementarios

1. **Jerarquía IS-A:**
  - Los equipos están especializados en **Club** y **Selección Nacional**, lo que permite gestionar atributos específicos de cada tipo.
  - Supuesto: Un equipo no puede ser simultáneamente un Club y una Selección Nacional (exclusión).
2. **Pedidos y Personalización:**
  - Una camiseta puede o no tener personalización asociada, dependiendo de la preferencia del cliente.
  - Supuesto: Si un cliente no selecciona personalización, el precio base de la camiseta es suficiente.
3. **Restricción de Parches:**
  - Cada personalización puede incluir hasta 4 parches.
  - Supuesto: Si el cliente no selecciona parches, la personalización no se genera.
4. **Exclusividad entre Relaciones:**
  - Un cliente puede realizar pedidos que contengan camisetas genéricas, personalizadas, o ambas.
5. **Dependencias de Entidades Débiles:**
  - La entidad débil **Personalización** depende de la existencia de una camiseta y no puede existir de manera independiente.
6. **Imágenes:**
  - Cada camiseta puede tener múltiples imágenes asociadas, pero una imagen no puede pertenecer a más de una camiseta.
7. **Stock y Precio:**
  - Cada camiseta tiene un stock asociado que debe actualizarse automáticamente después de cada pedido procesado.



## Modelo Relacional

```
# Modelo Relacional Product-E-Match

- **equipo** (`NOMBRE`, `escudo`, `equipacion`)

- **seleccion_nacional** (`nombre_equipo`, `ranking_fifa`)
- `nombre_equipo`: **FOREIGN KEY** `equipo` (`NOMBRE`)

- **club** (`nombre_equipo`, `estadio`, `liga`)
- `nombre_equipo`: **FOREIGN KEY** `equipo` (`NOMBRE`)

- **jugador** (`ID`, `nombre`, `dorsal`, `nombre_equipo`)
- `nombre_equipo`: **FOREIGN KEY** `equipo` (`NOMBRE`)

- **parches** (`ID`, `parche`)

- **personalizacion** (`ID`, `num_parches`, `id_parches`, `id_jugador`) //PRECIO
ATRIBUTO CALCULADO
- `id_parches`: **FOREIGN KEY** `parches` (`ID`)
- `id_jugador`: **FOREIGN KEY** `jugador` (`ID`)

- **imagen_camiseta** (`ID`, `ruta`)

- **camiseta** (`ID`, `temporada`, `categoria`, `talla`, `nombre_equipo`,
`id_personalizacion`, `id_imagen`) //STOCK Y PRECIO ATRIBUTOS CALCULADOS
- `nombre_equipo`: **FOREIGN KEY** `equipo` (`NOMBRE`)
- `id_personalizacion`: **FOREIGN KEY** `personalizacion` (`ID`)
- `id_imagen`: **FOREIGN KEY** `imagen_camiseta` (`ID`)

- **distribuidor** (`ID`, `nombre`, `nombre_compania`, `iban`, `contacto`)

- **cliente** (`ID`, `nombre`, `telefono`, `direccion`) //DESCUENTO Y NUM COMPRAS
ATRIBUTOS CALCULADOS

- **pedido_cliente** (`ID`, `id_camiseta`, `id_cliente`) //DURACION ENVIO, PRECIO Y
ESTADO ATRIBUTOS CALCULADOS
- `id_camiseta`: **FOREIGN KEY** `camiseta` (`ID`)
- `id_cliente`: **FOREIGN KEY** `cliente` (`ID`)
```



```
- **pedido_distribuidor** (`ID`, `fecha_pago`, `fecha_llegada`, `id_distribuidor`,  
`id_camiseta`, `id_pedido_cliente`) //PRECIO Y DURACION ENVIO ATRIBUTOS CALCULADOS  
- `id_distribuidor`: **FOREIGN KEY** `distribuidor` (`ID`)  
- `id_camiseta`: **FOREIGN KEY** `camiseta` (`ID`)  
- `id_pedido_cliente`: **FOREIGN KEY** `pedido_cliente` (`ID`)  
  
- **cliente_realiza_pedido** (`id_cliente`, `id_pedido_cliente`, `fecha_realiza`)  
- `id_cliente`: **FOREIGN KEY** `cliente` (`ID`)  
- `id_pedido_cliente`: **FOREIGN KEY** `pedido_cliente` (`ID`)  
  
- **cliente_recibe_pedido** (`id_cliente`, `id_pedido_cliente`, `fecha_envia`)  
- `id_cliente`: **FOREIGN KEY** `cliente` (`ID`)  
- `id_pedido_cliente`: **FOREIGN KEY** `pedido_cliente` (`ID`)
```

## Base de datos

A continuación, se proporciona una descripción de la base de datos en SQL, diseñada para implementarse en PostgreSQL. Esta descripción incluye la creación de tablas, claves primarias, claves ajenas, dominios, restricciones y valores permitidos.

```
-- Tabla equipo  
CREATE TABLE equipo (  
  id SERIAL PRIMARY KEY,  
  nombre VARCHAR(50) NOT NULL,  
  escudo VARCHAR(150) UNIQUE, --ruta a la imagen  
  equipacion VARCHAR(50) NOT NULL,  
  CONSTRAINT chk_escudo CHECK (escudo ~ '^(.*)?[a-zA-Z0-9_-]+\.\.(jpg|jpeg|png)$'),  
  CONSTRAINT chk_equipacion CHECK (  
    equipacion LIKE 'primera' OR  
    equipacion LIKE 'segunda' OR  
    equipacion LIKE 'tercera'  
  )  
);  
  
-- Tabla seleccion_nacional  
CREATE TABLE seleccion_nacional (  
  ranking_fifa INT NOT NULL,  
  id_equipo SERIAL NOT NULL,
```



```
FOREIGN KEY (id_equipo) REFERENCES equipo (id) ON DELETE CASCADE,
CONSTRAINT chk_ranking_fifa CHECK (ranking_fifa >= 1 AND ranking_fifa <= 210)
);

-- Tabla club
CREATE TABLE club (
    estadio VARCHAR(50),
    liga VARCHAR(50) NOT NULL,
    id_equipo SERIAL NOT NULL,
    FOREIGN KEY (id_equipo) REFERENCES equipo (id) ON DELETE CASCADE
);

-- Tabla jugador
CREATE TABLE jugador (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    dorsal INT NOT NULL,
    id_equipo SERIAL NOT NULL,
    FOREIGN KEY (id_equipo) REFERENCES equipo(id) ON DELETE CASCADE,
    CONSTRAINT chk_nombre_jugador CHECK (nombre ~ '^[A-Z]+(\.?[ ]?[A-Z]+)*\.?$'),
    CONSTRAINT chk_dorsal CHECK (dorsal >= 1 AND dorsal <= 99)
);

-- Tabla parches
CREATE TABLE parches (
    id SERIAL PRIMARY KEY,
    parche VARCHAR(150) NOT NULL,
    CONSTRAINT chk_parche CHECK (parche ~ '^(.*/)?[a-zA-Z0-9_-]+\.\.(jpg|jpeg|png)$')
);

-- Tabla personalizacion
CREATE TABLE personalizacion (
    id SERIAL PRIMARY KEY,
    num_parches INT NOT NULL,
    id_jugador SERIAL,
    id_parches SERIAL,
    FOREIGN KEY (id_jugador) REFERENCES jugador(id) ON DELETE CASCADE,
    FOREIGN KEY (id_parches) REFERENCES parches(id) ON DELETE CASCADE,
    CONSTRAINT chk_num_parches CHECK (num_parches >= 0 AND num_parches <= 4)
```



```
);

CREATE TABLE imagen_camiseta (
  id SERIAL PRIMARY KEY,
  imagen VARCHAR(150) NOT NULL --ruta a la imagen
  CONSTRAINT chk_imagen_camiseta CHECK (imagen ~
'^(.*/)?[a-zA-Z0-9_-]+\.\.(jpg|jpeg|png)$')
);

-- Tabla camiseta
CREATE TABLE camiseta (
  id SERIAL PRIMARY KEY,
  temporada VARCHAR(7) NOT NULL,
  categoria VARCHAR(25) NOT NULL,
  talla VARCHAR(10) NOT NULL,
  id_equipo SERIAL NOT NULL,
  id_personalizacion SERIAL,
  id_imagen_camiseta SERIAL,
  FOREIGN KEY (id_equipo) REFERENCES equipo(id) ON DELETE CASCADE,
  FOREIGN KEY (id_personalizacion) REFERENCES personalizacion(id) ON DELETE CASCADE,
  FOREIGN KEY (id_imagen_camiseta) REFERENCES imagen_camiseta(id) ON DELETE CASCADE,
  CONSTRAINT chk_temporada CHECK (temporada ~ '^[0-9]{4}/[0-9]{2}$'),
  CONSTRAINT chk_categoria CHECK (categoria IN ('retro', 'jugador', 'fan',
'conjunto')),
  CONSTRAINT chk_talla CHECK (
  talla IN ('16', '18', '20', '22', '24', '26', '28', 'S', 'M', 'L', 'XL', '2XL',
'3XL')
)
);

-- Tabla distribuidor
CREATE TABLE distribuidor (
  id SERIAL PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  nombre_compania VARCHAR(50) NOT NULL,
  IBAN VARCHAR(34) UNIQUE NOT NULL,
  contacto VARCHAR(50) NOT NULL,
```



```
CONSTRAINT chk_nombre_distribuidor CHECK (nombre ~ '^[A-Z][a-zA-Z ]*$'),
CONSTRAINT chk_iban CHECK (
    IBAN ~ '^[A-Z]{2}[0-9A-Z]{13,32}$'
)
);

-- Tabla cliente
CREATE TABLE cliente (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    telefono VARCHAR(15),
    direccion VARCHAR(255),
    CONSTRAINT chk_nombre_cliente CHECK (nombre ~ '^[A-Z][a-zA-Z ]*$'),
    CONSTRAINT chk_telefono CHECK (
        telefono ~ '^\+?[0-9]{9,15}$'
    )
);

-- Tabla pedido_cliente
CREATE TABLE pedido_cliente (
    id SERIAL PRIMARY KEY,
    id_cliente SERIAL NOT NULL,
    id_camiseta SERIAL NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES cliente(id) ON DELETE CASCADE,
    FOREIGN KEY (id_camiseta) REFERENCES camiseta(id) ON DELETE CASCADE
);

-- Tabla pedido_distribuidor
CREATE TABLE pedido_distribuidor (
    id SERIAL PRIMARY KEY,
    fecha_pago DATE NOT NULL,
    fecha_llegada DATE,
    id_distribuidor SERIAL NOT NULL,
    id_camiseta SERIAL NOT NULL,
    id_pedido_cliente SERIAL NOT NULL,
    FOREIGN KEY (id_distribuidor) REFERENCES distribuidor(id) ON DELETE CASCADE,
    FOREIGN KEY (id_camiseta) REFERENCES camiseta(id) ON DELETE CASCADE,
    FOREIGN KEY (id_pedido_cliente) REFERENCES pedido_cliente(id) ON DELETE CASCADE,
    CONSTRAINT chk_fecha_pago CHECK (fecha_pago <= CURRENT_DATE),
```



```
CONSTRAINT chk_fecha_llegada CHECK (fecha_llegada IS NULL OR fecha_llegada <=
CURRENT_DATE),
    CONSTRAINT chk_fecha_llegada_posterior_pago CHECK (fecha_llegada IS NULL OR
fecha_llegada >= fecha_pago)
);

-- Tabla relacion cliente realiza pedido
CREATE TABLE cliente_realiza_pedido (
    id_cliente SERIAL NOT NULL,
    id_pedido_cliente SERIAL NOT NULL,
    fecha_realiza DATE NOT NULL,
    PRIMARY KEY (id_cliente, id_pedido_cliente),
    FOREIGN KEY (id_cliente) REFERENCES cliente(id) ON DELETE CASCADE,
    FOREIGN KEY (id_pedido_cliente) REFERENCES pedido_cliente(id) ON DELETE CASCADE,
    CONSTRAINT chk_fecha_realiza CHECK (fecha_realiza <= CURRENT_DATE)
);

-- Tabla relacion cliente realiza pedido
CREATE TABLE cliente_recibe_pedido (
    id_cliente SERIAL NOT NULL,
    id_pedido_cliente SERIAL NOT NULL,
    fecha_recibe DATE,
    PRIMARY KEY (id_cliente, id_pedido_cliente),
    FOREIGN KEY (id_cliente) REFERENCES cliente(id) ON DELETE CASCADE,
    FOREIGN KEY (id_pedido_cliente) REFERENCES pedido_cliente(id) ON DELETE CASCADE,
    CONSTRAINT chk_fecha_recibe CHECK (fecha_recibe <= CURRENT_DATE)
);
```

Se muestra a continuación el listado de tablas implementadas en la base de datos que se ha creado “product-e-match”:

```
postgres=# \c "product-e-match"
You are now connected to database "product-e-match" as user "postgres".
product-e-match=# \dt
               List of relations
Schema |      Name      | Type  | Owner
-----+-----+-----+-----
public | camiseta       | table | postgres
public | cliente        | table | postgres
public | cliente_realiza_pedido | table | postgres
public | cliente_recibe_pedido | table | postgres
public | club           | table | postgres
public | distribuidor   | table | postgres
public | equipo         | table | postgres
public | imagen_camiseta | table | postgres
public | jugador        | table | postgres
public | parches        | table | postgres
public | pedido_cliente | table | postgres
public | pedido_distribuidor | table | postgres
public | personalizacion | table | postgres
public | seleccion_nacional | table | postgres
(14 rows)
```





Y los atributos de las tablas más relevantes de la base de datos:

## Entidad camisetas:

```
product-e-match=# \d camiseta
Table "public.camiseta"
  Column      | Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
id            | integer       |           | not null | nextval('camiseta_id_seq'::regclass)
temporada    | character varying(7) |           | not null |
categoria    | character varying(25) |           | not null |
talla        | character varying(10) |           | not null |
id_equipo    | integer       |           | not null | nextval('camiseta_id_equipo_seq'::regclass)
id_personalizacion | integer       |           | not null | nextval('camiseta_id_personalizacion_seq'::regclass)
id_imagen_camiseta | integer       |           | not null | nextval('camiseta_id_imagen_camiseta_seq'::regclass)
Indexes:
  "camiseta_pkey" PRIMARY KEY, btree (id)
Check constraints:
  "chk_categoria" CHECK (categoria::text = ANY (ARRAY['retro'::character varying, 'jugador'::character varying, 'fan'::character varying, 'conjunto'::character varying::text[]]))
  "chk_talla" CHECK (talla::text = ANY (ARRAY['16'::character varying, '18'::character varying, '20'::character varying, '22'::character varying, '24'::character varying, '26'::character varying, '28'::character varying, 'S'::character varying, 'M'::character varying, 'L'::character varying, 'XL'::character varying, '2XL'::character varying, '3XL'::character varying::text[]]))
  "chk_temporada" CHECK (temporada::text ~ '^(0-9)(0-9)(25)::text')
Foreign-key constraints:
  "camiseta_id_equipo_fkey" FOREIGN KEY (id_equipo) REFERENCES equipo(id) ON DELETE CASCADE
  "camiseta_id_imagen_camiseta_fkey" FOREIGN KEY (id_imagen_camiseta) REFERENCES imagen_camiseta(id) ON DELETE CASCADE
  "camiseta_id_personalizacion_fkey" FOREIGN KEY (id_personalizacion) REFERENCES personalizacion(id) ON DELETE CASCADE
Referenced by:
  TABLE "pedido_cliente" CONSTRAINT "pedido_cliente_id_camiseta_fkey" FOREIGN KEY (id_camiseta) REFERENCES camiseta(id) ON DELETE CASCADE
  TABLE "pedido_distribuidor" CONSTRAINT "pedido_distribuidor_id_camiseta_fkey" FOREIGN KEY (id_camiseta) REFERENCES camiseta(id) ON DELETE CASCADE
```

## Entidad pedido distribuidor:

```
product-e-match=# \d pedido_distribuidor
Table "public.pedido_distribuidor"
  Column      | Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
id            | integer       |           | not null | nextval('pedido_distribuidor_id_seq'::regclass)
fecha_pago    | date          |           | not null |
fecha_llegada | date          |           | not null |
id_distribuidor | integer       |           | not null | nextval('pedido_distribuidor_id_distribuidor_seq'::regclass)
id_camiseta    | integer       |           | not null | nextval('pedido_distribuidor_id_camiseta_seq'::regclass)
id_pedido_cliente | integer       |           | not null | nextval('pedido_distribuidor_id_pedido_cliente_seq'::regclass)
Indexes:
  "pedido_distribuidor_pkey" PRIMARY KEY, btree (id)
Check constraints:
  "chk_fecha_llegada" CHECK (fecha_llegada IS NULL OR fecha_llegada <= CURRENT_DATE)
  "chk_fecha_llegada_posterior_pago" CHECK (fecha_llegada IS NULL OR fecha_llegada >= fecha_pago)
  "chk_fecha_pago" CHECK (fecha_pago <= CURRENT_DATE)
Foreign-key constraints:
  "pedido_distribuidor_id_camiseta_fkey" FOREIGN KEY (id_camiseta) REFERENCES camiseta(id) ON DELETE CASCADE
  "pedido_distribuidor_id_distribuidor_fkey" FOREIGN KEY (id_distribuidor) REFERENCES distribuidor(id) ON DELETE CASCADE
  "pedido_distribuidor_id_pedido_cliente_fkey" FOREIGN KEY (id_pedido_cliente) REFERENCES pedido_cliente(id) ON DELETE CASCADE
```

## Entidad pedido cliente:

```
product-e-match=# \d pedido_cliente
Table "public.pedido_cliente"
  Column      | Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
id            | integer       |           | not null | nextval('pedido_cliente_id_seq'::regclass)
id_cliente    | integer       |           | not null | nextval('pedido_cliente_id_cliente_seq'::regclass)
id_camiseta    | integer       |           | not null | nextval('pedido_cliente_id_camiseta_seq'::regclass)
Indexes:
  "pedido_cliente_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
  "pedido_cliente_id_camiseta_fkey" FOREIGN KEY (id_camiseta) REFERENCES camiseta(id) ON DELETE CASCADE
  "pedido_cliente_id_cliente_fkey" FOREIGN KEY (id_cliente) REFERENCES cliente(id) ON DELETE CASCADE
Referenced by:
  TABLE "cliente_realiza_pedido" CONSTRAINT "cliente_realiza_pedido_id_pedido_cliente_fkey" FOREIGN KEY (id_pedido_cliente) REFERENCES pedido_cliente(id) ON DELETE CASCADE
  TABLE "cliente_recibe_pedido" CONSTRAINT "cliente_recibe_pedido_id_pedido_cliente_fkey" FOREIGN KEY (id_pedido_cliente) REFERENCES pedido_cliente(id) ON DELETE CASCADE
  TABLE "pedido_distribuidor" CONSTRAINT "pedido_distribuidor_id_pedido_cliente_fkey" FOREIGN KEY (id_pedido_cliente) REFERENCES pedido_cliente(id) ON DELETE CASCADE
```

## Entidad equipo:

```
product-e-match=# \d equipo
Table "public.equipo"
  Column      | Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
id            | integer       |           | not null | nextval('equipo_id_seq'::regclass)
nombre        | character varying(50) |           | not null |
escudo        | character varying(150) |           | not null |
equipacion    | character varying(50) |           | not null |
Indexes:
  "equipo_pkey" PRIMARY KEY, btree (id)
  "equipo_escudo_key" UNIQUE CONSTRAINT, btree (escudo)
Check constraints:
  "chk_equipacion" CHECK (equipacion::text ~ 'primera'::text OR equipacion::text ~ 'segunda'::text OR equipacion::text ~ 'tercera'::text)
  "chk_escudo" CHECK (escudo::text ~ '^(\.\/)?[a-zA-Z0-9_-]+\.\.(jpg|jpeg|png)$'::text)
Referenced by:
  TABLE "camiseta" CONSTRAINT "camiseta_id_equipo_fkey" FOREIGN KEY (id_equipo) REFERENCES equipo(id) ON DELETE CASCADE
  TABLE "club" CONSTRAINT "club_id_equipo_fkey" FOREIGN KEY (id_equipo) REFERENCES equipo(id) ON DELETE CASCADE
  TABLE "jugador" CONSTRAINT "jugador_id_equipo_fkey" FOREIGN KEY (id_equipo) REFERENCES equipo(id) ON DELETE CASCADE
  TABLE "seleccion_nacional" CONSTRAINT "seleccion_nacional_id_equipo_fkey" FOREIGN KEY (id_equipo) REFERENCES equipo(id) ON DELETE CASCADE
```



## Carga de datos de las tablas

### 1. Tabla equipo

```
INSERT INTO equipo (nombre, escudo, equipacion)
VALUES
    ('Real Madrid', 'path/to/real_madrid.png', 'primera'),
    ('FC Barcelona', 'path/to/fc_barcelona.png', 'primera'),
    ('Atlético de Madrid', 'path/to/atletico.png', 'segunda'),
    ('Selección Española', 'path/to/espana.png', 'primera');
```

### 2. Tabla jugador

```
INSERT INTO jugador (nombre, dorsal)
VALUES
    ('Karim Benzema', 9),
    ('Lionel Messi', 10),
    ('Álvaro Morata', 19);
```

### 3. Tabla camiseta

```
INSERT INTO camiseta (temporada, categoria, talla, stock, precio, id_equipo)
VALUES
    ('2024/25', 'jugador', 'M', 50, 79.99, 1),
    ('2024/25', 'retro', 'L', 30, 69.99, 2),
    ('2024/25', 'fan', 'XL', 20, 49.99, 3),
    ('2024/25', 'jugador', 'S', 100, 89.99, 4);
```

### 4. Tabla cliente

```
INSERT INTO cliente (nombre, direccion, telefono, descuento, num_compras)
VALUES
    ('Juan Pérez', 'Calle Falsa 123', '+34123456789', 5.00, 2),
    ('María López', 'Av. Siempre Viva 456', '+34987654321', 10.00, 1);
```

### 5. Tabla pedido\_cliente

```
INSERT INTO pedido_cliente (fecha, estado, id_cliente)
VALUES
    ('2024-12-01', 'Procesado', 1),
    ('2024-12-10', 'Enviado', 2);
```



## 6. Tabla distribuidor

```
INSERT INTO distribuidor (nombre, nombre_compania, IBAN, contacto)
VALUES
('Distribuidor A', 'Deportivos A S.L.', 'ES1234567890123456789012', 'contacto@a.com'),
('Distribuidor B', 'Deportivos B S.L.', 'ES9876543210987654321098', 'contacto@b.com');
```

## 7. Tabla distribuidor

```
INSERT INTO pedido_distribuidor (fecha_pago, fecha_llegada, duracion_envio,
id_distribuidor)
VALUES
('2024-12-01', '2024-12-05', 4, 1),
('2024-12-15', NULL, NULL, 2);
```

# Consultas de ejemplo

A continuación, se proporciona un conjunto de consultas SQL diseñadas para probar y verificar el correcto funcionamiento de la base de datos creada. Estas consultas abordan distintas funcionalidades y relaciones entre las tablas.

## 1. Consultas Básicas

### 1.1. Obtener todas las camisetas con su equipo asociado

```
SELECT c.id, c.temporada, c.categoria, c.talla, e.nombre AS equipo, c.precio, c.stock
FROM camiseta c
JOIN equipo e ON c.id_equipo = e.id;
```

### 1.2. Mostrar los clientes y el número de pedidos realizados

```
SELECT cl.nombre, cl.telefono, COUNT(pc.id) AS num_pedidos
FROM cliente cl
LEFT JOIN pedido_cliente pc ON cl.id = pc.id_cliente
GROUP BY cl.id
ORDER BY num_pedidos DESC;
```

## 2. Consultas con Filtros

### 2.1. Buscar camisetas de un equipo específico

```
SELECT c.id, c.temporada, c.categoria, c.talla, c.precio, c.stock
FROM camiseta c
```



```
JOIN equipo e ON c.id_equipo = e.id  
WHERE e.nombre = 'Real Madrid';
```

## **2.2. Mostrar pedidos realizados en un rango de fechas**

```
SELECT pc.id AS pedido_id, cl.nombre AS cliente, pc.fecha, pc.estado  
FROM pedido_cliente pc  
JOIN cliente cl ON pc.id_cliente = cl.id  
WHERE pc.fecha BETWEEN '2024-12-01' AND '2024-12-31';
```

## **3. Consultas con Agregaciones**

### **3.1. Obtener el stock total de camisetas por equipo**

```
SELECT e.nombre AS equipo, SUM(c.stock) AS total_stock  
FROM camiseta c  
JOIN equipo e ON c.id_equipo = e.id  
GROUP BY e.nombre  
ORDER BY total_stock DESC;
```

### **3.2. Calcular el precio total de un pedido**

```
SELECT p.id AS pedido_id, cl.nombre AS cliente, SUM(c.precio * pc.cantidad) AS  
total_pedido  
FROM pedido_cliente p  
JOIN cliente cl ON p.id_cliente = cl.id  
JOIN pedido_camiseta pc ON p.id = pc.id_pedido  
JOIN camiseta c ON pc.id_camiseta = c.id  
WHERE p.id = 1  
GROUP BY p.id, cl.nombre;
```

## **4. Consultas con Relaciones Complejas**

### **4.1. Mostrar todas las camisetas personalizadas con sus parches**

```
SELECT c.id AS camiseta_id, c.temporada, c.categoria, p.descripcion AS parche  
FROM camiseta c  
JOIN camiseta_personalizacion cp ON c.id = cp.id_camiseta  
JOIN personalizacion pe ON cp.id_personalizacion = pe.id  
JOIN personalizacion_parche pp ON pe.id = pp.id_personalizacion  
JOIN parche p ON pp.id_parche = p.id;
```

### **4.2. Detalles de pedidos realizados por un cliente específico**

```
SELECT p.id AS pedido_id, c.temporada, c.categoria, c.talla, pc.cantidad, c.precio
```



```
FROM pedido_cliente p
JOIN pedido_camiseta pc ON p.id = pc.id_pedido
JOIN camiseta c ON pc.id_camiseta = c.id
WHERE p.id_cliente = (SELECT id FROM cliente WHERE nombre = 'Juan Pérez');
```

## 5. Consultas de Integridad y Pruebas de Restricciones

### 5.1. Camisetas sin personalización

```
SELECT c.id, c.temporada, c.categoria, c.talla, c.precio
FROM camiseta c
LEFT JOIN camiseta_personalizacion cp ON c.id = cp.id_camiseta
WHERE cp.id_camiseta IS NULL;
```

### 5.2. Clientes sin pedidos

```
SELECT cl.id, cl.nombre
FROM cliente cl
LEFT JOIN pedido_cliente pc ON cl.id = pc.id_cliente
WHERE pc.id IS NULL;
```

## 6. Consultas de Unión y Subconsultas

### 6.1. Lista de todos los equipos (clubes y selecciones)

```
SELECT e.nombre AS equipo, 'Club' AS tipo, c.estadio, c.liga
FROM equipo e
JOIN club c ON e.id = c.id_equipo
UNION
SELECT e.nombre AS equipo, 'Selección Nacional' AS tipo, NULL AS estadio, NULL AS liga
FROM equipo e
JOIN seleccion_nacional sn ON e.id = sn.id_equipo;
```

### 6.2. Subconsulta: Pedidos con más de 2 camisetas

```
SELECT p.id, cl.nombre AS cliente, COUNT(pc.id_camiseta) AS total_camisetas
FROM pedido_cliente p
JOIN cliente cl ON p.id_cliente = cl.id
JOIN pedido_camiseta pc ON p.id = pc.id_pedido
GROUP BY p.id, cl.nombre
HAVING COUNT(pc.id_camiseta) > 2;
```



# Diseño de la API REST

En el contexto de la gestión de bases de datos modernas, las aplicaciones requieren una forma eficiente y estructurada para interactuar con los datos almacenados en los sistemas de gestión de bases de datos relacionales (RDBMS). Este proyecto aborda la implementación de un API RESTful para gestionar una base de datos relacional en PostgreSQL, diseñada previamente como parte de un sistema para la gestión de camisetas deportivas y sus personalizaciones.

El desarrollo de un API REST (Application Programming Interface) tiene como objetivo proporcionar una interfaz estándar para realizar operaciones de creación, lectura, actualización y eliminación (CRUD) sobre los datos, permitiendo que sistemas externos, como aplicaciones web o móviles, interactúen de manera programática y eficiente con la base de datos. La arquitectura REST, por su naturaleza, sigue los principios de diseño que permiten crear sistemas escalables, mantenibles y reutilizables.

## Propósito del API REST

El API REST desarrollado tiene como finalidad:

1. **Gestionar datos relacionados con camisetas deportivas:** Permitir la creación, actualización, eliminación y consulta de camisetas en la base de datos.
2. **Optimizar la interacción entre cliente y servidor:** Proveer un medio eficiente y estandarizado para que las aplicaciones cliente (por ejemplo, aplicaciones web o móviles) se comuniquen con el servidor.
3. **Abstraer la lógica de la base de datos:** Ocultar la complejidad del sistema relacional a los consumidores del API, proporcionando operaciones sencillas y comprensibles para gestionar los datos.
4. **Facilitar la integración:** Proveer un sistema que pueda integrarse fácilmente con otras aplicaciones o servicios que requieran acceso a los datos.

## Tecnologías Utilizadas

1. **Flask:**
  - Elegido por su simplicidad y flexibilidad, Flask es un microframework para Python que facilita la creación de APIs RESTful ligeros y eficientes.
  - Su diseño minimalista permite integrar extensiones como Flask-CORS para habilitar el acceso cruzado entre dominios.
2. **PostgreSQL:**
  - Base de datos relacional robusta y ampliamente utilizada que proporciona integridad referencial y soporte para operaciones complejas.
  - La base de datos incluye entidades como camisetas, equipos, clientes y pedidos, que se gestionan a través del API.



### 3. Psycopg2:

- Librería para interactuar con PostgreSQL desde Python, utilizada para ejecutar consultas SQL directamente desde el API.

## Características del API REST

El API desarrollado permite las siguientes operaciones:

1. **Creación:** Agregar nuevas camisetas y personalizaciones, asociarlas con equipos existentes y gestionar pedidos de clientes.
2. **Lectura:** Consultar datos sobre camisetas, personalizaciones, clientes y pedidos, tanto en detalle como de manera agregada.
3. **Actualización:** Modificar atributos de camisetas, personalizaciones y pedidos, asegurando la consistencia referencial en la base de datos.
4. **Eliminación:** Borrar registros de camisetas o personalizaciones de manera segura, considerando las dependencias existentes en el modelo relacional.

## Estructura del API

El API sigue una estructura RESTful clara y estandarizada:

- **Endpoints:** Cada recurso (por ejemplo, camisetas, equipos, clientes) tiene un conjunto de rutas dedicadas para realizar operaciones específicas.
- **Métodos HTTP:**
  - **GET:** Para consultar recursos.
  - **POST:** Para crear nuevos recursos.
  - **PUT:** Para actualizar recursos existentes.
  - **DELETE:** Para eliminar recursos.

## Justificación del Desarrollo

El desarrollo de este API RESTful no solo proporciona un acceso estructurado a los datos almacenados en la base de datos, sino que también sirve como un puente entre los datos y los usuarios finales, ya sea a través de aplicaciones web o móviles. Este enfoque garantiza la reutilización del sistema en múltiples plataformas y asegura la escalabilidad del sistema en escenarios futuros, como la incorporación de nuevas funcionalidades o la integración con servicios externos.

Con este diseño, el sistema se posiciona como una solución robusta para la gestión de camisetas y sus personalizaciones, con el potencial de ser adoptado y ampliado en un entorno real.



Se adjuntan a continuación imágenes de los resultados de la API REST, los datos almacenados y las pruebas realizadas sobre la misma:

En esta imagen se muestra la ruta principal en la que se listan las distintas camisetas que existen en la base de datos:

Lista de Camisetas						
<a href="#">Añadir Nueva Camiseta</a>						
Camisetas Disponibles						
ID	TEMPORADA	CATEGORÍA	TALLA	EQUIPO	ACCIONES	
4	2024/25	jugador	XL	Real Madrid	<a href="#">Eliminar</a>	<a href="#">Editar</a>
3	2024/25	jugador	S	Fc Barcelona	<a href="#">Eliminar</a>	<a href="#">Editar</a>

Editamos la camiseta con ID '3' clicando el hipervínculo que se muestra a la derecha del botón de eliminar, y se nos muestra esta pantalla que pertenece a la ruta '/update':

Editar Camiseta
<div>Temporada</div> <div>2024/25</div>
<div>Categoría</div> <div>Jugador</div>
<div>Talla</div> <div>S</div>
<div>Nombre del Equipo</div> <div>Fc Barcelona</div>
<div>Actualizar Camiseta</div>





Se selecciona por ejemplo la talla L para editar la camiseta:

## Editar Camiseta

**Temporada**

2024/25

**Categoría**

Jugador

- S
- M
- ✓ L
- XL
- 2XL

Fc Barcelona

**Actualizar Camiseta**

Posteriormente, actualizamos los cambios en el botón 'Actualizar Camiseta' y observamos los cambios en la ruta principal:

## Lista de Camisetas

[Añadir Nueva Camiseta](#)

**Camisetas Disponibles**

ID	TEMPORADA	CATEGORÍA	TALLA	EQUIPO	ACCIONES
4	2024/25	jugador	XL	Real Madrid	<a href="#">Eliminar</a> <a href="#">Editar</a>
3	2024/25	jugador	L	Fc Barcelona	<a href="#">Eliminar</a> <a href="#">Editar</a>

```
(venv) robertpm@Air-Robert api % flask --app app.py run
* Serving Flask app 'app.py'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [21/Dec/2024 00:00:42] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Dec/2024 00:09:05] "GET /editar/4 HTTP/1.1" 200 -
127.0.0.1 - - [21/Dec/2024 00:09:29] "GET /editar/3 HTTP/1.1" 200 -
127.0.0.1 - - [21/Dec/2024 00:11:04] "POST /editar/3 HTTP/1.1" 302 -
127.0.0.1 - - [21/Dec/2024 00:11:04] "GET / HTTP/1.1" 200 -
```



Para la eliminación de las camisetas hacemos click en el botón de eliminar, después de ello podemos observar que la camiseta que hemos seleccionado se ha eliminado de la lista:

### Lista de Camisetas

Añadir Nueva Camiseta

Camisetas Disponibles

ID	TEMPORADA	CATEGORÍA	TALLA	EQUIPO	ACCIONES
4	2024/25	jugador	XL	Real Madrid	<div>Eliminar</div> <a href="#">Editar</a>

```
(venv) robertpm@Air-Robert api % flask --app app.py run
* Serving Flask app 'app.py'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [21/Dec/2024 00:00:42] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Dec/2024 00:09:05] "GET /editar/4 HTTP/1.1" 200 -
127.0.0.1 - - [21/Dec/2024 00:09:29] "GET /editar/3 HTTP/1.1" 200 -
127.0.0.1 - - [21/Dec/2024 00:11:04] "POST /editar/3 HTTP/1.1" 302 -
127.0.0.1 - - [21/Dec/2024 00:11:04] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Dec/2024 00:18:13] "POST /eliminar/3 HTTP/1.1" 200 -
127.0.0.1 - - [21/Dec/2024 00:18:45] "GET / HTTP/1.1" 200 -
```

Finalmente, para añadir una camiseta también se ofrece un botón para ello, el cual redirige a la ruta ‘/nueva’:

### Añadir Nueva Camiseta

Temporada

2024/25

Categoría

Jugador

Talla

S

Nombre del Equipo

Real Madrid

URL de la Imagen de la Camiseta

ruta/imagen.jpg

Añadir Camiseta



# Presupuesto del Proyecto

## 1. Desarrollo

Concepto	Descripción	Costo (€)
Análisis y diseño	Requerimientos funcionales, diseño de base de datos, estructura del proyecto	1,200
Desarrollo del back-end	API REST con Flask y PostgreSQL, manejo de rutas, lógica de negocio	2,500
Desarrollo del front-end	Plantillas HTML con Jinja2, diseño de UI/UX	1,500
Integración y pruebas	Pruebas funcionales, unitarias, integración continua	800
Total desarrollo		6,000

---

## 2. Infraestructura

Concepto	Descripción	Costo (€)
Hosting y dominio	Servidor web (por ejemplo, AWS, Azure, o DigitalOcean) y dominio anual	300
Base de datos	Costo del servidor PostgreSQL (en la nube)	200
Certificados SSL	Certificado de seguridad para HTTPS	50
Total infraestructura		550

---



### 3. Mantenimiento y Soporte

Concepto	Descripción	Costo (€)
Mantenimiento mensual	Actualizaciones, monitoreo, soporte técnico (12 meses)	1,800
Ampliación de funcionalidades	Horas dedicadas a nuevas características o mejoras	1,200
Total mantenimiento		3,000

---

### 4. Presupuesto Total

Concepto	Costo (€)
Desarrollo	6,000
Infraestructura	550
Mantenimiento	3,000
Total general	9,550



## Conclusión

El desarrollo de este proyecto ha permitido la implementación de un sistema completo y funcional para la gestión de camisetas deportivas, cubriendo tanto el diseño de la base de datos como la construcción de un API REST que interactúa con esta. A través de este proceso, se han integrado conceptos clave de diseño de bases de datos relacionales, arquitectura RESTful y desarrollo de software, logrando un sistema escalable, flexible y listo para ser utilizado en entornos reales.

## Logros del Proyecto

### 1. Modelado y Diseño Relacional:

- La base de datos diseñada refleja de manera precisa los requisitos funcionales y semánticos del sistema, incorporando relaciones complejas, restricciones de integridad, y entidades débiles y fuertes. Esto asegura un modelo relacional robusto y bien estructurado.

### 2. API REST Funcional:

- Se desarrolló un API REST en Flask que permite realizar operaciones CRUD sobre los datos, proporcionando un acceso seguro y eficiente a través de rutas bien definidas. Este enfoque facilita la integración con aplicaciones cliente, como interfaces web o móviles, que requieren interactuar con el sistema.

### 3. Escalabilidad y Flexibilidad:

- El sistema está diseñado para permitir la adición de nuevas funcionalidades, como el manejo de pedidos más complejos, la integración con pasarelas de pago o la conexión con otros sistemas externos. Esto posiciona el proyecto como una base sólida para futuras ampliaciones.

### 4. Simulación Real de Negocio:

- Los datos y relaciones modelados representan un escenario realista en la gestión de camisetas, incluyendo personalizaciones, asociaciones con equipos y pedidos de clientes, brindando una experiencia que se asemeja a un sistema utilizado en un entorno profesional.

## Aprendizajes y Desafíos

Durante el desarrollo del proyecto, se han abordado y resuelto varios desafíos técnicos:

- **Consistencia Referencial:** Se gestionaron con éxito las dependencias entre tablas mediante claves primarias y ajenas, asegurando la integridad de los datos en todas las operaciones.



- **Optimización de Consultas:** Se desarrollaron consultas SQL eficientes para manejar relaciones complejas, lo que permitió realizar análisis detallados y pruebas de datos.
- **Interacción API-Base de Datos:** La integración entre el API y la base de datos PostgreSQL fue una experiencia enriquecedora, destacando la importancia de abstraer la lógica del sistema subyacente mediante una interfaz sencilla y reutilizable.

### Futuras Extensiones

El proyecto, en su estado actual, proporciona una base funcional sólida, pero también abre camino para futuras extensiones, como:

- **Soporte para Analítica Avanzada:** Incorporar funciones para analizar tendencias en ventas o preferencias de los clientes.
- **Mejoras en la Seguridad:** Implementar autenticación y autorización para proteger el acceso a los datos.
- **Integración con Aplicaciones Cliente:** Desarrollar aplicaciones web o móviles que utilicen el API para gestionar la interacción con los usuarios finales.

### Conclusión Final

Este proyecto no solo cumple con los objetivos iniciales, sino que también demuestra cómo los conceptos teóricos de bases de datos y APIs REST pueden ser aplicados para crear sistemas funcionales y útiles. Además, el enfoque modular y escalable asegura que este sistema puede ser adaptado y ampliado para satisfacer nuevas necesidades o escenarios. En última instancia, el proyecto destaca la importancia de una planificación detallada, un diseño robusto y un desarrollo iterativo para construir soluciones tecnológicas de calidad.