



MiPIZZERÍA 2.0

DISEÑO DE BASE DE DATOS

PRÁCTICA TERCER TRIMESTRE 1º DAW
ALBERTO MOLEIRO SÁNCHEZ

OBJETIVO

La siguiente documentación describe las modificaciones añadidas al diseño de una base de datos del servicio **online** de una pizzería. Así mismo, se ha añadido un ejemplo sencillo de una aplicación web que interacciona con la base de datos.

INTRODUCCIÓN

Este proyecto parte de una base de datos, diseñada previamente, para un servicio de gestión online de pedidos. Dicha base de datos recoge la información de los pedidos realizados, clientes, pizzas e ingredientes.

Debido al crecimiento del negocio se ha hecho necesario añadir modificaciones que recojan nuevos aspectos del negocio. Aprovechando esta tesitura, a petición del cliente, también se añaden nuevas funcionalidades a la base de datos para hacerla más dinámica.

MODIFICACIONES Y FUNCIONALIDADES

- Debido a la apertura de una nueva sucursal es necesario añadir una tabla que recoja la información de las sucursales del negocio y además debiendo permitir poder solicitar pedidos indiferentemente.
- Los tipos de pizza deben guardar la relación con los ingredientes que los forman y además debe automatizarse la grabación de esta información cuando seleccionemos un tipo.
- Se deberá llevar un registro con la fecha, hora y número de pedidos enviados.
- Como funcionalidades propias de la base de datos se deberá crear funciones o procedimientos que calculen el precio de las pizzas y pedidos, inserten ingredientes únicos y en función del tipo, liquiden pedidos a domicilio y recoger.
- Se automatizará el cálculo del precio de las pizzas y pedidos así como la inserción de los ingredientes en función del tipo.

ÍNDICE

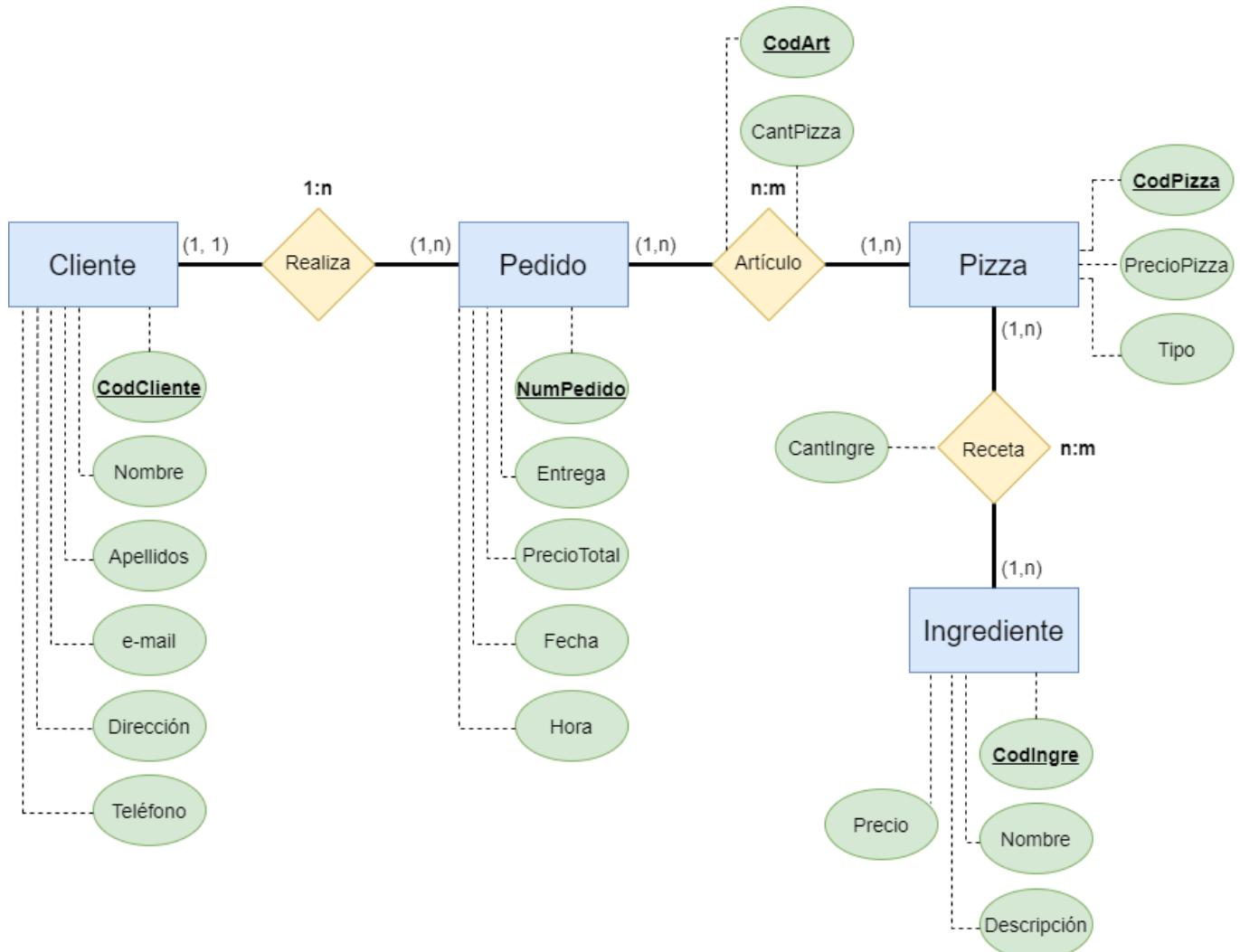
1. DISEÑO MODELO ENTIDAD/RELACIÓN	4
1.1 DISEÑO PREVIO	4
1.2 DISEÑO ACTUAL	5
2. MODELO RELACIONAL	6
2.1. TABLA MODELO RELACIONAL	6
2.2. MODELO RELACIONAL GRÁFICO	7
2.3. DESCRIPCIÓN TABLAS Y COLUMNAS	8
2.3.1. Tabla Cliente	8
2.3.2. Tabla Direcciones	9
2.3.3. Tabla Teléfonos	9
2.3.4. Tabla Sucursal	9
2.3.5. Tabla Pedido	9
2.3.6. Tabla Pizza	10
2.3.7. Tabla Precio_Pizza	10
2.3.8. Tabla Tipo	10
2.3.9. Tabla Ingrediente	10
2.3.10. Tabla Pizza_Ingrediente	10
2.3.11. Tabla Tipo_Ingrediente	11
2.3.12. Tabla PrecioVigen	11
2.3.13. Tabla ListaPrecios	11
2.3.14. Tabla Envío	11
3. FUNCIONES Y PROCEDIMIENTOS	11
3.1 FUNCIÓN PrecioPizza	12
3.2 FUNCIÓN PrecioPedido	12
3.3 PROCEDIMIENTO Insertalngrediente	13
3.4 PROCEDIMIENTO CrearPizzaBase	13
3.5 PROCEDIMIENTO EnvioPedidos	14
3.6 PROCEDIMIENTO PedidoRecogido	14
3.7 TRIGGER nuevaPizza_after_insert	15
3.8 PROCEDIMIENTO ActualizaPrecioPizza	15

3.9 TRIGGER nuevoIngre_after_insert	15
3.10 TRIGGER nuevoIngre_after_update	15
3.11 PROCEDIMIENTO ActualizaPrecioPedido	15
3.12 TRIGGER pizzaPedido_after_insert	16
3.13 TRIGGER pizzaPedido_after_delete	16
4. APPLICACIÓN WEB	16
4.1 TECNOLOGÍAS UTILIZADAS	16
4.2 DISEÑO UI/UX DE LA WEB	17
4.2.1 Vista Landing Page	18
4.2.2 Vista Pizzas	19
4.2.3 Vista Login	20
4.2.4 Vista Pedido	21
4.3 DISEÑO ESTRUCTURA WEB BACKEND	22
4.4 FLUJO DE EJECUCIÓN DE LA APLICACIÓN	23
4.5 CONCLUSIONES DE LA APLICACIÓN	29

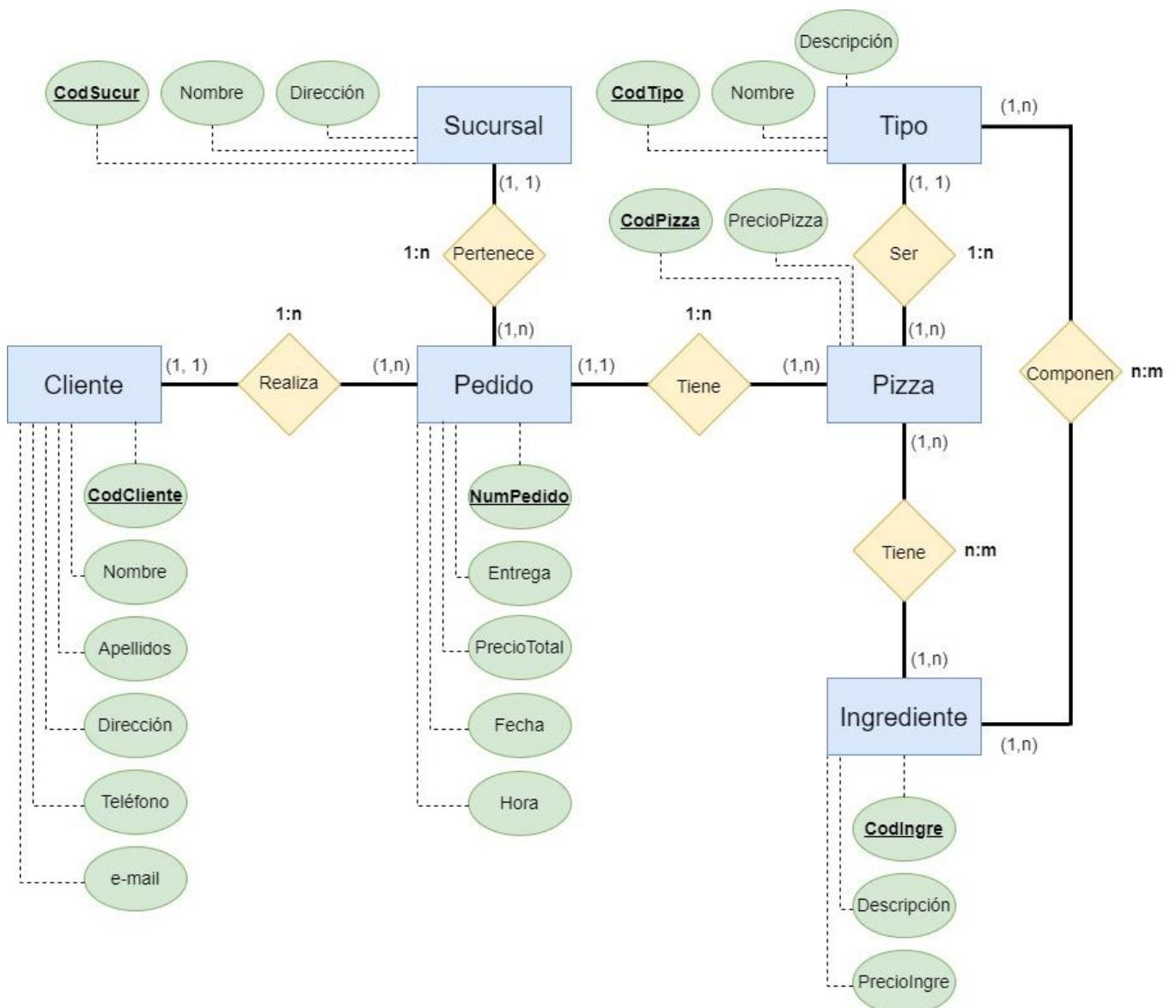
1. DISEÑO MODELO ENTIDAD/RELACIÓN

A continuación se muestra la evolución del modelo de entidad relación.

1.1 DISEÑO PREVIO



1.2 DISEÑO ACTUAL



2. MODELO RELACIONAL

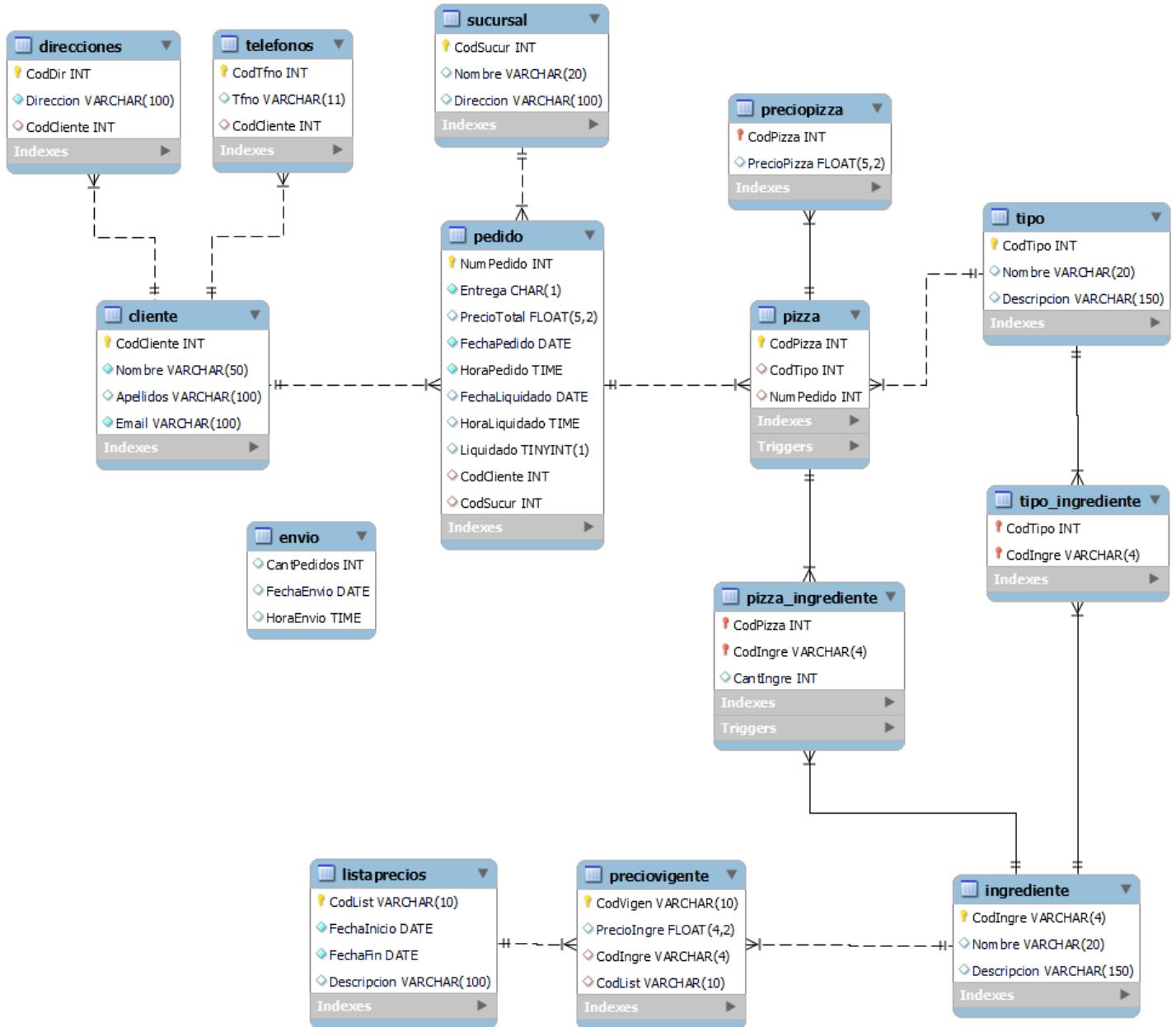
A continuación, se detallan las tablas que se generan tras el desarrollo del modelo entidad/relación.

2.1. TABLA MODELO RELACIONAL

Tablas	Columnas	Claves Primarias	Claves Foráneas
Cliente	<u>CodCliente</u> , Nombre, Apellidos, email.	<u>Código Cliente.</u>	
Direcciones	<u>CodDir</u> , Dirección, <u>CodCliente</u> .	<u>CodDir.</u>	<u>CodCliente</u> (ref. Cliente).
Teléfonos	<u>CodTfno</u> , Tfno, <u>CodCliente</u> .	<u>CodTfno.</u>	<u>CodCliente</u> (ref. Cliente).
Sucursal	<u>CodSucur</u> , Nombre, Dirección.	<u>CodSucur</u>	
Pedido	<u>NumPedido</u> , Entrega, PrecioTotal, FechaPedido, HoraPedido, FechaLiquidado, HoraLiquidado, Liquidado <u>CodCliente</u> , <u>CodSucur</u> .	<u>NumPedido.</u>	<u>CodCliente</u> (ref. Cliente), <u>CodSucur</u> (ref. Sucursal).
Tipo	<u>CodTipo</u> , Nombre, Descripción.	<u>CodTipo.</u>	
Pizza	<u>CodPizza</u> , <u>CodTipo</u> , <u>NumPedido</u> .	<u>Código Pizza.</u>	<u>CodTipo</u> (ref. Tipo), <u>NumPedido</u> (ref. Pedido).
PrecioPizza	<u>CodPizza</u> , PrecioPizza.	<u>CodPizza</u>	<u>CodPizza</u> (ref. Pizza).
Ingrediente	<u>CodIngre</u> , Nombre, Descripción.	<u>CodIngre.</u>	
Pizza_Ingrediente	<u>CodPizza</u> , <u>CodIngre</u> , CantIngre.	<u>CodPizza</u> , <u>CodIngre.</u>	<u>CodPizza</u> (ref. Pizza), <u>CodIngre</u> (ref. Ingrediente).
Tipo_Ingrediente	<u>CodTipo</u> , <u>CodIngre</u> .	<u>CodTipo.</u> <u>CodIngre.</u>	<u>CodTipo</u> , <u>CodIngre</u> .
ListaPrecios	<u>CodList</u> , FechaInicio, FechaFin, Descripción.	<u>CodList.</u>	
PrecioVigente	<u>CodVigen</u> , PrecioIngre, <u>CodIngre</u> , <u>CodList</u> .	<u>CodVigen.</u>	<u>CodIngre</u> (ref. Ingrediente), <u>CodList</u> (ref. ListaPrecios).
Envío	CantPedidos, FechaEnvio, HoraPedido.		

2.2. MODELO RELACIONAL GRÁFICO

En este gráfico podemos ver como quedarían definidas las tablas y sus columnas tras normalizar.



2.3. DESCRIPCIÓN TABLAS Y COLUMNAS

2.3.1. Tabla Cliente

Es la tabla que almacenará los datos del cliente. Sus columnas son:

- **CodCliente:** Es el código identificativo del cliente. Es un código numérico auto incrementado.
- **Nombre:** Es el nombre del cliente. Será obligatorio llenar este campo.
- **Apellidos:** Son los apellidos del cliente.
- **Email:** Es el correo electrónico del cliente. Será obligatorio llenar este campo.

2.3.2. Tabla Direcciones

Es la tabla dónde se almacenarán las direcciones de los clientes. Sus columnas son:

- **CodDir:** Es el código identificativo de la dirección. Es un código numérico auto incrementado.
- **Dirección:** Aquí se almacenará la dirección de cada cliente. Será obligatorio llenar este campo.
- **CodCliente:** Es el código de cliente y hará referencia a la clave primaria de la tabla cliente.

2.3.3. Tabla Teléfonos

Es la tabla dónde se guardarán los teléfonos desde los que se realicen pedidos. Sus columnas son:

- **CodTfno:** Es el código identificativo del teléfono. Es un código numérico auto incrementado.
- **Tfno:** Es el número de teléfono desde el que se realiza el pedido. Será obligatorio llenar este campo.
- **CodCliente:** Es el código de cliente y hará referencia a la clave primaria de la tabla cliente.

2.3.4. Tabla Sucursal

Es la tabla donde se guarda la información referente a las sucursales.

- **CodSucur:** Es el código identificativo de la sucursal. Es un código numérico auto incrementado.
- **Nombre:** Es el nombre de la sucursal. Será obligatorio llenar este campo.
- **Dirección:** Dirección de la sucursal. Será obligatorio llenar este campo.

2.3.5. Tabla Pedido

Es la tabla dónde se almacenará toda la información referente al pedido. Sus columnas son:

- **NumPedido:** Es el código identificativo del pedido. Es un código numérico auto incrementado.
- **Entrega:** Es la columna dónde se almacenará si el pedido ha de entregarse en local o domicilio. Los campos podrán tener 2 valores “R” (recoger) y “D” (domicilio). Será obligatorio llenar este campo.
- **PrecioTotal:** Es el precio total del pedido. Será resultado de la suma de todas las pizzas que contenga el pedido.
- **FechaPedido:** Será la fecha en la que se realiza el pedido. El formato con el que se guarda la información es “AAAA-MM-DD”.

- **HoraPedido:** Es la hora a la que se realiza el pedido y el formato en el que se guardará la información será de hora y minutos “HH:MM”.
- **FechaLiquidado:** Será la fecha en la que se realiza el envío o recogida.. El formato con el que se guarda la información es “AAAA-MM-DD”.
- **HoraLiquidado:** Es la hora a la que se realiza el envío o recogida. El formato en el que se guardará la información será de hora y minutos “HH:MM”.
- **Liquidado:** Es un campo booleano que marcará si el pedido ha sido enviado o recogido.
- **CodCliente:** Es el código de cliente y hará referencia a la clave primaria de la tabla cliente.
- **CodSucur:** Es el código de sucursal y hará referencia a la clave primaria de la tabla sucursal.

2.3.6. Tabla Pizza

Es la tabla dónde se almacenarán las pizzas. Sus columnas son:

- **CodPizza:** Es el código identificativo de cada pizza. Es un código numérico auto incrementado.
- **CodTipo:** Es el código identificativo del tipo de pizza y hará referencia a la tabla tipo.

2.3.7. Tabla Precio_Pizza

Es la tabla dónde se almacenarán los precios de cada pizza. Esta tabla surge tras normalizar el precio ya que es un campo que dependía funcionalmente de la pizza y del tipo de pizza. Sus columnas son:

- **CodPizza:** Es el código identificativo de cada pizza que hace referencia a la clave primaria de la tabla Pizza.
- **PrecioPizza:** Es el precio de cada pizza.

2.3.8. Tabla Tipo

Es la tabla que almacena los tipos de pizza y una descripción breve de cada tipo. Sus columnas son:

- **CodTipo:** Es el código identificativo de cada tipo de pizza.
- **Nombre:** Es el nombre del tipo de pizza.
- **Descripción:** Será una breve descripción de cada tipo de pizza y de los ingredientes que la forman.

2.3.9. Tabla Ingrediente

Es la tabla que almacena la información referente a los ingredientes. Sus columnas son:

- **CodIngre:** Es el código identificativo de cada ingrediente.
El código se generará tomando la letra o letras iniciales del nombre de cada ingrediente. En el caso de nombres de ingredientes que compartan la misma letra inicial se irán tomando letras en orden de escritura hasta generar un código único.
La primera letra del código irá siempre en mayúsculas y las siguientes en minúsculas.
- **Nombre:** Es el nombre de cada ingrediente.
- **Descripción:** Es una breve descripción de cada ingrediente.

2.3.10. Tabla Pizza_Ingrediente

Es la tabla que almacena la relación de ingredientes que lleva cada pizza. Sus columnas son:

- **CodPizza:** Es el código identificativo de cada pizza y hará referencia a la clave primaria de la tabla Pizza. Será parte de la clave primaria de esta tabla.
- **CodIngre:** Es el código identificativo de cada ingrediente y hará referencia a la clave primaria de la tabla Ingrediente. Será parte de la clave primaria de esta tabla.
- **CantIngre:** Será la cantidad de un ingrediente que llevará cada pizza. Por defecto el valor de este campo será 1.

2.3.11. Tabla Tipo_Ingrediente

Es la tabla que guarda la relación entre ingredientes y tipo. Sus columnas son:

- **CodTipo:** Es el código identificativo de cada tipo que hace referencia a la tabla Tipo y forma parte de la clave primaria de la tabla.
- **CodIngre:** Es el código identificativo de cada ingrediente que hace referencia a la tabla Ingredientes y forma parte de la clave primaria de la tabla.

2.3.12. Tabla PrecioVigen

Es la tabla que guardará la información referente a la relación entre la tabla Ingrediente y la lista de precios.

- **CodVigen:** Es el código identificativo de la tabla, compuesto de la letra "V" en mayúscula más un número que se incrementará con cada nuevo registro.
- **PrecioIngre:** Es el precio de cada ingrediente en función de la temporada.
- **CodIngre:** Es el código identificativo de cada ingrediente y hará referencia a la clave primaria de la tabla Ingrediente.
- **CodList:** Es el código identificativo de cada lista de precios y hará referencia a la clave primaria de la tabla ListaPrecios.

2.3.13. Tabla ListaPrecios

Es la tabla que almacena la información referente a las listas de precios que pudiera haber en función de la temporada.

- **CodList:** Es el código identificativo de la tabla.
- **FechaInicio:** Es la fecha que indica cuando comienza una temporada. El formato con el que se guarda la información es "AAAA-MM-DD". Será obligatorio llenar este campo.
- **FechaFin:** Es la fecha que indica cuando termina una temporada. El formato con el que se guarda la información es "AAAA-MM-DD". Será obligatorio llenar este campo.
- **Descripción:** Es una breve descripción de cada temporada y las alteraciones que estas conllevan.

2.3.14. Tabla Envío

Es la tabla que guarda la información de los envíos a domicilio que han sido liquidados.

- **CantPedidos:** Es el número de pedidos liquidados.
- **FechaEnvío:** Es la fecha de liquidación de los envíos.
- **HoraEnvío:** Es la hora de liquidación de los envíos.

3. FUNCIONES Y PROCEDIMIENTOS

A continuación se detallan las funciones y procedimientos implementados en la base de datos.

3.1 FUNCIÓN PrecioPizza

Está función calcula el importe de una pizza en función de cada ingrediente y su cantidad, teniendo en cuenta la variación de los precios de los ingredientes según la temporada en la que nos encontremos.

La función **PrecioPizza** recibe como parámetro un **código de pizza**.

Mediante este **código de pizza** se declara un cursor que rescata el **código de cada ingrediente**, que lleva la pizza y la **cantidad**, y por otra parte se almacena la **fecha del pedido** al que pertenece la pizza.

Con la **fecha del pedido** se averigua el **código de la lista de precios** que corresponde.

Con estos datos se abre el **cursor** que devuelve el **ingrediente y su cantidad** y se inicia un bucle para recorrerlo. A cada vuelta de bucle, se **acumula** en la variable **precio** el precio de cada ingrediente, que rescatamos de la tabla **PrecioVigente** gracias al **código de la lista de precios** y el **código de ingrediente**, y lo multiplicamos por la **cantidad de ingrediente** que lleva nuestra pizza.

Finalmente se devuelve la variable **precio**.

Variables	Descripción
codigo (parámetro)	Variable que hace referencia a la clave primaria de la pizza.
ingreCod	Variable que guarda la clave primaria del ingrediente.
listCod	Variable que guarda la clave primaria de la lista de precios.
ingreCant	Variable que guarda la cantidad de ingredientes.
fin	Variable booleana para terminar la ejecución del bucle while.
precio	Variable que guarda el sumatorio del precio.
pedidoFecha	Variable que almacena la fecha del pedido al que pertenece la pizza.
PizIngCursor	Cursor que selecciona la clave primaria de ingrediente y la cantidad de ingrediente de la tabla Pizza_Ingrediente.

3.2 FUNCIÓN PrecioPedido

Esta función calcula el precio de los pedidos en función de las pizzas que tiene asociadas.

La función **PrecioPizza** recibe como parámetro el **código de pedido** del que se quiere calcular el precio.

Mediante un cursor implícito se calcula el precio del pedido y se almacena en la variable **precio**.

Finalmente se devuelve el **precio**.

Variables	Descripción
codigo (parámetro)	Variable que hace referencia a la clave primaria del pedido.
precio	Variable que guarda el sumatorio de precios de las pizzas.

3.3 PROCEDIMIENTO InsertaIngrediente

Este procedimiento recibe como parámetros de entrada el **código de pizza**, el **código de ingrediente** y la **cantidad de ingrediente**.

Con estos datos ejecuta una sentencia de inserción en la tabla **Pizza_Ingredient**.

Variables	Descripción
pizzaCod (parámetro)	Variable que hace referencia a la clave primaria de la pizza.
ingreCod (parámetro)	Variable que hace referencia a la clave primaria del ingrediente.
ingreCant (parámetro)	Variable que guarda la cantidad de ingrediente que se quiere insertar.

3.4 PROCEDIMIENTO CrearPizzaBase

Este procedimiento recibe como parámetro un **código de pizza**.

Se declara un cursor que rescata los **códigos de ingredientes** de la tabla **Tipo_Ingrediente** mediante el **código de pizza** pasado como parámetro.

Se recorre el cursor con un bucle y a cada vuelta se llama al procedimiento **Insertalngrediente** y se le pasa como parámetros el **código de pizza**, el **código de ingrediente** y como cantidad de ingrediente se pasa “1”.

Variables	Descripción
codigo (parámetro)	Variable que hace referencia a la clave primaria de la pizza.
ingreCod	Variable que guarda el código de ingrediente.
fin	Variable booleana para terminar la ejecución del bucle while.
TipIngCursor	Cursor que devuelve los códigos de ingredientes asociados al tipo de la tabla Tipo_Ingrediente .

3.5 PROCEDIMIENTO EnvioPedidos

Este procedimiento cuenta el número de pedidos a domicilio que están pendientes de ser enviados, los marca como liquidados, almacena la fecha y hora en la que se liquidan e inserta en la tabla Envíos el número de pedidos que se han enviado, la fecha y la hora.

Mediante un cursor implícito se almacena el **número de pedidos** a domicilio que están sin liquidar.

Con una sentencia de actualización se cambia el valor de la columna **Liquidado** de la tabla **Pedido**, y se registra la fecha y hora en la que se realiza.

Con una sentencia de inserción se almacena en la tabla **Envío el número de pedidos**, y la fecha y hora actual.

Variables	Descripción
numeroPedidos	Variable que almacena los pedidos a domicilio que están pendientes de ser liquidados.

3.6 PROCEDIMIENTO PedidoRecogido

Este procedimiento recibe un **código de pedido** y ejecuta una sentencia de actualización en la tabla **Pedido**, si el pedido es para recoger, cambia el valor de la columna **Liquidado**.

Variables	Descripción
codigo (parámetro)	Variable que hace referencia a la clave primaria de la tabla Pedido.

3.7 TRIGGER nuevaPizza_after_insert

Este trigger se dispara después de la inserción de una nueva pizza en la tabla **Pizza** y ejecuta el procedimiento **CrearPizzaBase** y le pasa como parámetro el **nuevo código de pizza**.

3.8 PROCEDIMIENTO ActualizaPrecioPizza

Este procedimiento recibe como parámetro un **código de pizza** y si existe un registro en la tabla **Precio_Pizza** realiza una sentencia de actualización asignando como valor a la columna **PrecioPizza** la función **PrecioPizza** pasándole como parámetro el **código de pizza** para el registro con el mismo código.

Si no existe un registro con clave primaria igual al **código de pizza** realiza una sentencia de inserción en los mismos términos que la sentencia de actualización antes descrita.

Variables	Descripción
codigo (parámetro)	Variable que hace referencia a la clave primaria de la tabla Pizza.

3.9 TRIGGER nuevoIngre_after_insert

Este trigger se dispara tras la inserción de un nuevo ingrediente asociado a una pizza en la tabla **Pizza_Ingrediente** ejecutando el procedimiento **ActualizaPrecioPizza** pasándole como parámetro el **nuevo código de pizza**.

3.10 TRIGGER nuevoIngre_after_update

Este trigger se dispara tras la actualización de la **cantidad de ingrediente** en la tabla **Pizza_Ingrediente** ejecutando el procedimiento **ActualizaPrecioPizza** pasándole como parámetro el **nuevo código de pizza**.

3.11 PROCEDIMIENTO ActualizaPrecioPedido

Este procedimiento recibe como parámetro un **código de pedido** y ejecuta una sentencia de actualización sobre la tabla **Pedido** asignando como valor a la columna **PrecioTotal** la función **PrecioPedido** pasándole como parámetro el **código de pedido** para el registro con clave primaria igual al **código de pedido**.

Variables	Descripción
codigo (parámetro)	Variable que hace referencia a la clave primaria de la tabla Pedido.

3.12 TRIGGER pizzaPedido_after_insert

Este trigger se dispara tras la inserción de una nueva pizza en la tabla **Pizza** asociada a un pedido y ejecuta el procedimiento **ActualizaPrecioPedido** pasándole como parámetro el **nuevo código de pedido**.

3.13 TRIGGER pizzaPedido_after_delete

Este trigger se dispara tras el borrado de una nueva pizza en la tabla **Pizza** asociada a un pedido y ejecuta el procedimiento **ActualizaPrecioPedido** pasándole como parámetro el **viejo código de pedido**.

4. APPLICACIÓN WEB

Con objetivo de probar la interacción real con la base de datos se ha diseñado una aplicación sencilla que permite el registro de usuarios y la realización de pedidos.

Al no tratarse de objeto de esta materia el diseño de esta aplicación dista mucho de ser perfecto, sin embargo sirve de ejemplo para probar los mecanismos implementados en la base de datos.

4.1 TECNOLOGÍAS UTILIZADAS

Para la realización frontend del diseño se han utilizado los siguientes lenguajes:

- **HTML5**
- **CSS3**
- **JavaScript ***

Para la realización backend de la estructura del servidor se han utilizado los siguientes lenguajes:

- **Node.js** *
- **Express.js** (Framework de Node.js) *
- **Body-parser** (Librería de express.js) *

* Nótese que estos lenguajes eran desconocidos por el desarrollador hasta la realización de esta aplicación, por ello, se pide no tener en cuenta el código ya que está faltó de buenas prácticas y lejos de ser óptimo.

Como entorno de servidor se ha utilizado el paquete de software **XAMPP**. En específico como servidor web se ha utilizado **Apache** y como gestor de base de datos **MySQL**.

MySQL en **XAMPP** incluye como cliente del gestor de base de datos **phpMyAdmin** con motor **MariaDB**.

4.2 DISEÑO UI/UX DE LA WEB

Al no tratarse del objetivo de la práctica, el diseño de la parte del cliente es sencillo. La página consta de cuatro vistas principales, dos de ellas sirven para dar cuerpo a la aplicación pero su única funcionalidad es hacer de puente hasta las otras dos páginas, las cuales contienen los **formularios** de **registro** y de **realización del pedido**. La aplicación es **full responsive** aunque en este documento solo mostraremos las vistas para el ancho del viewport de 1200px.

- **Landing page**, nombrada como **Ofertas** en la web, alojada en el archivo “*index.html*” es una de las páginas de relleno/puente de la aplicación, nos muestra, según el flujo del documento, el header de la página con el logo, un menú de navegación, como cuerpo una serie de ofertas ficticias y el pie de página con información básica.
Los enlaces del menú de navegación nos dirigen a **Ofertas** que es la vista en la que nos encontramos, a la vista de **Pizzas** (*pizzas.html*) y **Login** (*login.html*), los enlaces del cuerpo de esta vista nos dirigen a la página de **Pedido** (*pedido.html*).
- **Pizzas** alojada en el archivo “*pizzas.html*” es la segunda vista de relleno/puente, la estructura es similar al **index** y los enlaces dirigen a las mismas vistas.
- **Login** alojada en el archivo “*login.html*” es la vista que nos muestra el formulario de registro de los clientes. Es un formulario **html** que envía la información mediante método **POST** y que está compuesto de los campos:
 - **Nombre** es un input de tipo texto que es requerido para validar el formulario.
 - **Apellidos** es un input de tipo texto.
 - **Dirección** es un input de tipo texto que es requerido para validar el formulario.
 - **Email** es un input de tipo email que es requerido para validar el formulario.
 - **Teléfono** es un input de tipo teléfono que es requerido para validar el formulario.

Cuando hacemos click en el botón “Registrar” hace una validación de los datos en el propio browser y si los datos son correctos nos redirige a una vista de confirmación de registro.

- **Pedido** alojada en el archivo “*pedido.html*” es la vista que nos muestra el formulario de realización de pedidos. Es un formulario gestionado por el archivo “*pedido.js*” con los siguientes campos:
 - **Usuario** es un input de tipo email que podemos llenar con el email de un usuario registrado.
 - **Sucursal** es un elemento select que nos proporciona como opciones de selección las sucursales del negocio.
 - **Entrega** es un elemento select que nos proporciona como opciones de selección los tipos de entrega.
 - **Tipo de pizza** es un elemento select que nos proporciona como opciones de selección los tipos de pizza predefinidas que tiene el negocio.
 - **Ingredientes** son varios input de tipo número manejados por dos botones de up/down que nos permiten elegir la cantidad de cada ingrediente en un rango de 0 a 2.

Además en esta vista tenemos un carrito de compra, muy sencillo, el cual nos muestra las pizzas y su tipo que vamos añadiendo, al pedido, al hacer click en el botón “Añadir al carrito”. Cuando terminamos de configurar nuestro pedido podemos hacer click en el botón “Comprar” para realizar el pedido, esta acción nos redirige a una vista de confirmación de pedido.

4.2.1 Vista Landing Page

The landing page features a large banner at the top with a pizza slice and the word "PIZZA". Below the banner is a navigation bar with links for "Ofertas", "Pizzas", and "Login". The main content area is titled "Ofertas" and displays four promotional boxes, each containing a pizza image, a price offer ("Oferta de varias pizzas por 7,95€"), and a green "AÑADIR AL CARRITO" button. Below these boxes is a large image of a pizza with the text "Las mejores pizzas". The footer contains sections for "Ayuda", "Nutrición y calidad", and "Acerca de MiPizza", along with links for "Política de privacidad", "Política de cookies", and "Aviso legal". Social media icons for Facebook, Instagram, and Twitter are also present.

Ofertas

Oferta de varias pizzas por 7,95€

AÑADIR AL CARRITO

Oferta de varias pizzas por 7,95€

AÑADIR AL CARRITO

Oferta de varias pizzas por 7,95€

AÑADIR AL CARRITO

Oferta de varias pizzas por 7,95€

AÑADIR AL CARRITO

Las mejores pizzas

Oferta de varias pizzas por 7,95€

AÑADIR AL CARRITO

Oferta de varias pizzas por 7,95€

AÑADIR AL CARRITO

Oferta de varias pizzas por 7,95€

AÑADIR AL CARRITO

Oferta de varias pizzas por 7,95€

AÑADIR AL CARRITO

Ayuda

- Preguntas frecuentes
- Localizador de pizzerías
- Estado del pedido
- COVID-19

Nutrición y calidad

- Listado de alérgenos
- Valores nutricionales
- Pizzas sin gluten

Acerca de MiPizza

- Quiénes somos
- Contacto
- Trabaja con nosotros

Política de privacidad Política de cookies Aviso legal

4.2.2 Vista Pizzas

The screenshot shows the MiPizza website's pizza menu page. At the top, there is a banner with a pizza image and a logo that says "PIZZA SINCE 2008". Below the banner, a red navigation bar contains the links "Ofertas", "Pizzas", and "Login". The main title "Nuestras Pizzas" is displayed in a green gradient box. A large image of pizzas cooking in a wood-fired oven is shown with the text "Las mejores pizzas". Below this, four pizza options are listed in boxes:

- Carbonara**
Masa, salsa carbonara, queso, cebolla, bacon y champiñón.
[PEDIR](#)
- Barbacoa**
Masa, salsa barbacoa, queso y carne.
[PEDIR](#)
- Margarita**
Masa, tomate y queso.
[PEDIR](#)
- A tu gusto!**
Elige la combinación de ingredientes que más te guste.
[PEDIR](#)

At the bottom, there is a footer with sections for "Ayuda", "Nutrición y calidad", and "Acerca de MiPizza", each with several links. Social media icons for Facebook, Instagram, and Twitter are also present.

Ayuda			Nutrición y calidad			Acerca de MiPizza		
Preguntas frecuentes	Localizador de pizzerías	Estado del pedido	Listado de alérgenos	Valores nutricionales	Pizzas sin gluten	Quiénes somos	Contacto	Trabaja con nosotros
COVID-19								

[Política de privacidad](#) [Política de cookies](#) [Aviso legal](#)

4.2.3 Vista Login

PIZZA
SINCE 2008

Ofertas Pizzas Login

Únete!

REGÍSTRATE

Nombre **Apellidos**

Nombre Apellidos

Dirección

Dirección

Email

e-mail

Teléfono

Teléfono

REGISTRAR

Ayuda

- Preguntas frecuentes
- Localizador de pizzerías
- Estado del pedido
- COVID-19

Nutrición y calidad

- Listado de alérgenos
- Valores nutricionales
- Pizzas sin gluten

Acerca de MiPizza

- Quiénes somos
- Contacto
- Trabaja con nosotros

 Política de privacidad Política de cookies Aviso legal   

4.2.4 Vista Pedido

The screenshot shows a mobile application for ordering pizza. At the top, there is a banner featuring a large pizza with various toppings like olives, bell peppers, and onions. A red ribbon across the banner has the word "PIZZA" in white. Below the banner is a dark red header bar with three white text links: "Ofertas", "Pizzas", and "Login".

The main content area is divided into four sections:

- 1. Identificate**: A form field labeled "Usuario" with the placeholder "usuario@email.com".
- 2. Elige tu tienda**: A dropdown menu labeled "Sucursal" with "Guadarrama" selected.
- 3. Elige el tipo de entrega**: A dropdown menu labeled "Entrega" with "Recoger" selected.
- 4. Configura tu pizza**:
 - A dropdown menu labeled "Elige el tipo de pizza" with "A Tu Gusto" selected.
 - A section titled "Elige los ingredientes" with a list of toppings and their counts:

Masa	1	-	+
Tomate	0	-	+
Queso	0	-	+
Cebolla	0	-	+
Carne	0	-	+
Atún	0	-	+
Bacon	0	-	+
Champiñón	0	-	+
Anchoas	0	-	+
Pimiento	0	-	+
Salsa Barbacoa	0	-	+
Salsa Carbonara	0	-	+
 - A green button at the bottom of this section labeled "AÑADIR AL CARRITO".

To the right of the configuration section is a "Carrito" (Cart) sidebar containing a list of items with quantities:

- Pizza Margarita x1
- Pizza Barbacoa x1
- Pizza Carbonara x1
- Pizza A Tu Gusto x1

Below the cart list is a green "COMPRAR" (Buy) button.

4.3 DISEÑO ESTRUCTURA WEB BACKEND

La parte backend de la aplicación consta de tres archivos encargados de gestionar la información recibida del cliente, la conexión con la base de datos y generar la conexión al servidor y los endpoints. Estos tres archivos son:

- **index.js** es archivo main de la aplicación y en él declaramos:
 - La escucha en un puerto específico del localhost.
 - Las rutas a los archivos estáticos de la aplicación.
 - Los endpoints de las solicitudes.
- **mysql_conector.js** es el archivo encargado de la conexión con la base de datos y donde declaramos las sentencias SQL.
- **pedido.js** es un script que se carga en el browser conjuntamente con la vista **Pedido** y que se encarga de la gestión del formulario de realización de pedido y de generar la solicitud cuando el usuario pulsa en el botón “comprar”.

4.4 FLUJO DE EJECUCIÓN DE LA APLICACIÓN

A continuación se describe el flujo de ejecución de la aplicación, como se sirven las vistas desde el servidor al cliente y cómo interacciona este con la base de datos a través de los archivos del backend.

- En primer lugar iniciamos el servidor desde el archivo **index.js** y le indicamos la ruta donde debe buscar los archivos estáticos haciendo uso de los recursos de **express.js**.

```
const app = express() //Iniciar express

//Iniciamos servidor
app.listen('8000', () => {
    console.log('aplicacion iniciada puerto 8000')
})

//Configurando archivos estáticos
app.use(express.static('.'))
```

Cuando generamos la escucha en el puerto indicado nos imprime en consola un mensaje de confirmación.

En este punto, si nos conectamos al puerto 8000 de nuestro **localhost**, el browser nos sirve la vista de *index.html*.

- Creamos la conexión con la base de datos en el archivo **mysql_conector.js**. Previamente debemos haber iniciado **XAMPP** y sus servicios de **Apache** y **MySQL**.

The screenshot shows the XAMPP Control Panel interface. At the top, it says "XAMPP Control Panel v3.3.0". Below that is a table with columns: Service, Module, PID(s), Port(s), and Actions. There are two rows: one for Apache (PID 15188, Ports 80, 443, Actions: Stop, Admin, Config, Logs) and one for MySQL (PID 15428, Port 33066, Actions: Stop, Admin, Config, Logs). Both services have green checkmarks in the "Service" column, indicating they are running.

```
//crear conexion
const conector = mysql.createConnection(
  {
    host: 'localhost',
    user: 'Administrador',
    password: 'Admin.1234',
    database: 'pizzeria',
    port: '33066'
  }
)
```

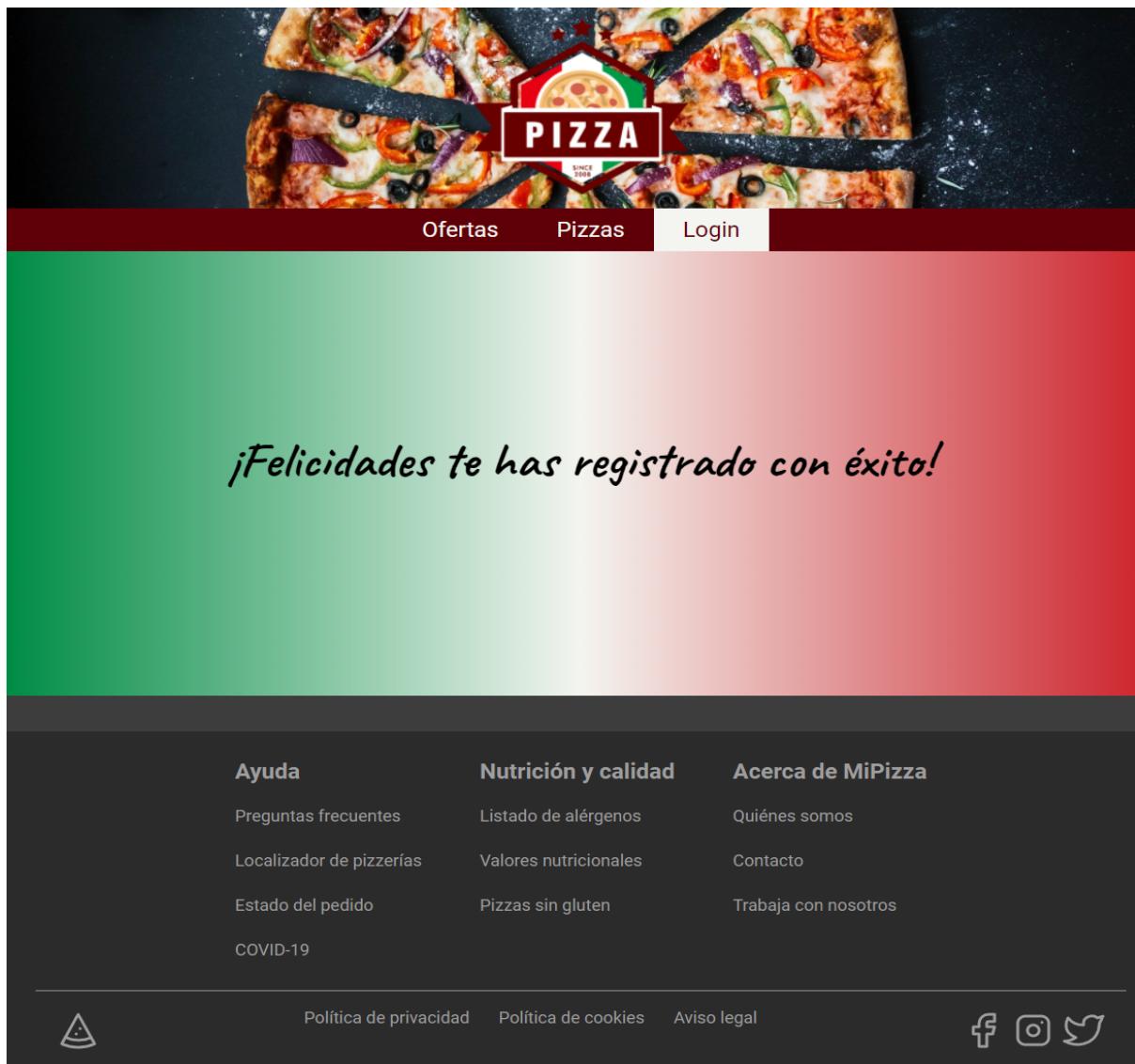
- Generamos un endpoint en el archivo **index.js** para recoger el formulario enviado desde *login.html* por el método de direccionamiento **POST**. Almacenamos los datos recibidos en variables y se los pasamos como parámetros a la función **agregarCliente()** definida en el archivo **mysql_conector.js**.

```
/* Recojo el formulario de login enviado por método post */
app.post('/login', (req, res) => {
  let nombre = req.body.nombre
  let apellidos = req.body.apellidos
  let direccion = req.body.direccion
  let email = req.body.email
  let telefono = req.body.telefono

  agregarCliente(nombre, apellidos, email, direccion, telefono)

  res.redirect('/registered.html')
})
```

Como respuesta devolvemos la vista *registered.html* que contiene un mensaje de confirmación de registro. Al tratarse de una aplicación sencilla no se ha gestionado las excepciones que pueden resultar, por ejemplo, al duplicar datos.



Como hemos dicho la función **agregarCliente()** está definida en el archivo **mysql_conector.js** y se compone de tres sentencias SQL de inserción que insertan datos en las tablas de **Cliente**, **Direcciones** y **Teléfonos**, y una sentencia de consulta que recupera el valor de la clave primaria de la tabla **Cliente** para poder usarla en las sentencias de inserción como clave foránea.

```
const agregarCliente = (nombre, apellidos, email, direccion, telefono) => {
  const sqlCliente = `INSERT INTO Cliente (Nombre,Apellidos,Email) VALUES ("${nombre}", "${apellidos}", "${email}")`;
  const idCliente = `SELECT CodCliente FROM Cliente WHERE Nombre = "${nombre}" AND Apellidos = "${apellidos}" AND Email="${email}"`;
  const sqlDireccion = `INSERT INTO Direcciones (Direccion,CodCliente) VALUES ("${direccion}",(${idCliente}))`;
  const sqlTelefono = `INSERT INTO Telefonos (Tfno,CodCliente) VALUES ("${telefono}",(${idCliente}))`;

  conector.query(sqlCliente, (err, result, field) => {
    if (err) throw err
  })

  conector.query(sqlDireccion, (err, result, field) => [
    if (err) throw err
  ])
  conector.query(sqlTelefono, (err, result, field) => {
    if (err) throw err
  })
}
```

- Gestionamos el formulario de pedido a través del archivo **pedido.js** que se carga conjuntamente con la vista **Pedido**.

La primera parte de este archivo es la declaración de las variables con la información referente al pedido y la creación de la función **configuracion()** que asigna valor a estas variables, se implementa en el archivo “*pedido.html*” ante un evento “*onchange*” de alguno de los inputs declarados.

```
const usuario = document.getElementById('usuario')

let sucursal = document.querySelector('#sucursal').options[document.querySelector('#sucursal').selectedIndex].value

let entrega = document.querySelector('#entrega').options[document.querySelector('#entrega').selectedIndex].value

let tipo = document.querySelector('#tipo').options[document.querySelector('#tipo').selectedIndex].value

let masa = document.querySelector('#masa').value //etc...
```

```
/* Asignación de valores */
function configuracion() {
    sucursal = document.querySelector('#sucursal').options[document.querySelector('#sucursal').selectedIndex].value

    entrega = document.querySelector('#entrega').options[document.querySelector('#entrega').selectedIndex].value

    document.querySelector('#tipo').options[document.querySelector('#tipo').selectedIndex].value

    tipo = document.querySelector('#tipo').options[document.querySelector('#tipo').selectedIndex].value

    masa = document.querySelector('#masa').value //etc...
```

También se crea el array de arrays de objetos “pedido” y el array de objetos “pizza”. El objeto embebido en “pedido” guarda la información en relación con la tabla **Pedido** de la base de datos (usuario, sucursal, entrega).

Los objetos embebidos en “pizza” guardan los ingredientes de la pizza y el tipo. Creamos la función **configuraPizza()** que asigna valores a la cantidad de ingrediente y se ejecuta ante el evento de hacer click en el botón “Añadir al carrito”.

```
/* Objetos que almacenan la información del pedido */
const pedido = [[{
    "usuario": "",
    "sucursal": sucursal,
    "entrega": entrega
}]]

let pizza //Variable global para el objeto pizza

function configuraPizza() { //Función de asignación
    pizza = [
        {
            "tipo": tipo
        },
        {
            "ingre": "M",
            "cant": masa
        },
        {
            "ingre": "T",
            "cant": tomate
        }, //etc... |
```

Por otra parte capturamos el evento del click en el botón “Añadir al Carrito” y declaramos las instrucciones que se ejecutan ante este evento.

- Primero se asignan los valores a las claves de los objetos de pedido, y pizza.
- Se añade al final del array pedido la pizza que se ha configurado.
- Se guarda en la variable “pedidoJSON” el objeto “pedido” convertido a formato JSON.
- Se bloquea la modificación de los inputs de usuario, entrega y sucursal.
- Se genera una nueva línea en el apartado carrito que nos indica la pizza y su tipo que hemos añadido al carrito.

```
/* Boton añadir para añadir una pizza al pedido */
const añadir = document.querySelector('#añadir')

/* Evento click sobre boton añadir */
añadir.addEventListener('click', () => {

    /* Construyo el objeto pedido y el objeto pizza con los valores actualizados*/
    pedido[0][0].usuario = usuario.value
    pedido[0][0].sucursal = sucursal
    pedido[0][0].entrega = entrega
    configuraPizza()

    /* Añadiendo una pizza al pedido y convirtiendo el objeto pedido a JSON */
    pedido.push(pizza)
    pedidoJSON = JSON.stringify(pedido)

    /* Desabilitando las opciones que no deben cambiar una vez se empieza el pedido */
    document.getElementById('usuario').readOnly = true
    document.getElementById('entrega').disabled = true
    document.getElementById('sucursal').disabled = true

    /* Mostrar en pantalla el contenido del pedido (Carrito de la compra)*/
    const tipoNombre = document.querySelector('#tipo').options[document.querySelector('#tipo').selectedIndex].text
    const pizzaPedido = document.querySelector(".pedido_carrito_pizzas")
    pizzaPedido.insertAdjacentHTML("beforeend", '<li class="pedido_carrito_pizzas-pizza">Pizza ' + tipoNombre + ' x1 </li>')
})
```

Por último, capturamos el click del botón “Comprar”. Ante este evento mandamos el objeto “pedidoJSON” junto con la URL mediante una petición **GET**.

```
/* Boton de comprar */

const comprar = document.querySelector('#comprar')

/* Mando al servidor el objeto pedido dentro de la url */
comprar.addEventListener('click', () => {

    window.location.href = `pedir/${pedidoJSON}`

})
```

- La gestión e inserción de los datos del pedido se realiza en el archivo **index.js**. Aquí recojo la petición **GET** realizada desde Pedido y se convierte el objeto JSON de la url en un objeto de JavaScript. Después le paso el objeto como parámetro a la función **agregarPedido()** definida en el archivo **mysql_conector.js**.

```
/* Recojo el objeto pedido enviado desde el cliente y lo convierto en objeto de js */
app.get('/pedir/:pedidoJSON', (req, res) => {

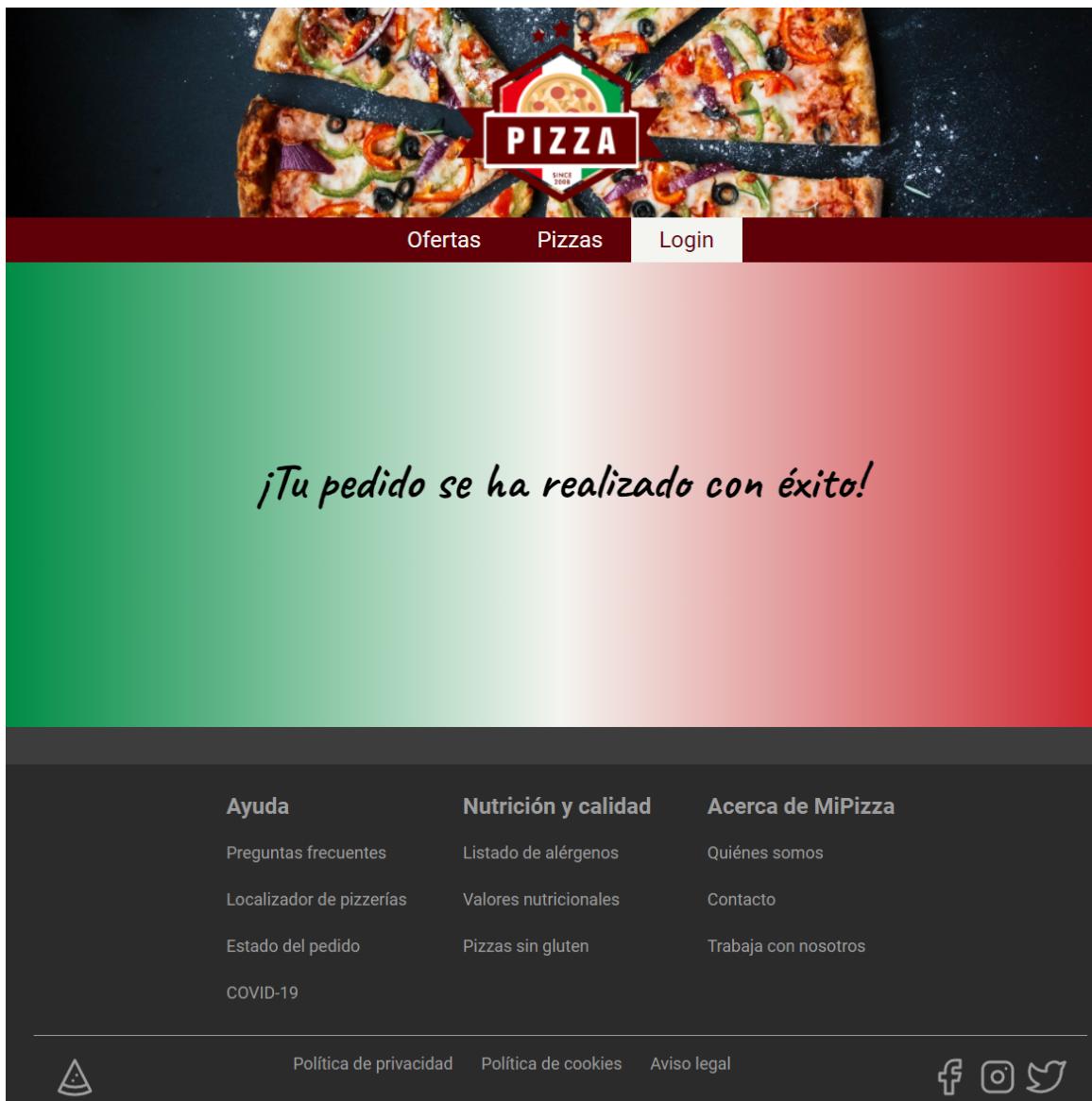
  let pedido = JSON.parse(req.params.pedidoJSON)

  agregarPedido(pedido)//Inserción en la base de datos

  res.redirect('/ordered.html')

})
```

Como respuesta devolvemos la vista *ordered.html* que contiene un mensaje de confirmación del pedido.



- La función **agregarPedido()** está definida en el archivo **mysql_conector.js** y gestiona la inserción de los datos en la tabla **Pedido**, la tabla **Pizza** y la tabla **Pizza_Ingrediente**.

Esta función es algo más compleja y demasiado extensa para incluir en el documento una captura de su código, por ello se detalla a continuación las instrucciones que contiene de forma esquemática.

- Primero se declaran las sentencias SQL para la inserción en la tabla **Pedido**.

Se genera una sentencia de consulta que devuelve la **clave primaria** del cliente en función de su email recibido en el objeto pedido para la clave del objeto “usuario”, y se genera una sentencia de inserción en pedido con los datos de los campos **Entrega**, **FechaPedido**, **HoraPedido**, **CodCliente** y **CodSucur**.

- Se declaran dos sentencias de consulta que devuelven la clave primaria del último pedido realizado y de la última pizza insertada en la tabla **Pizza**.

Se ejecuta la consulta de la **clave primaria** del pedido y se insertan las pizzas, su tipo y como **clave foránea** el número de pedido consultado. Para la inserción del tipo se recorre el objeto pedido, recibido como parámetro, y se inserta, por cada iteración, en los campos, de la tabla **Pizza**, **CodTipo** y **NumPedido**.

- Dentro de cada iteración, justo después de la inserción de una pizza, se consulta el valor de la **clave primaria** de la última pizza insertada, para poder insertar en la tabla **Pizza_Ingrediente** los ingredientes de esa pizza.

Para poder insertar los ingredientes se genera otro bucle para volver a recorrer el objeto pedido, pero en este caso recorremos los ingredientes de la pizza de esta iteración.

Como ya sabemos la propia base de datos inserta ingredientes en la tabla **Pizza_Ingrediente** en función del tipo, por ello debemos comprobar antes de insertar un nuevo ingrediente si este está insertado ya y si además la cantidad del nuevo ingrediente que vamos a insertar es mayor que 0, si se cumplen las condiciones se inserta el ingrediente con su cantidad, en caso contrario (que ya existiese el ingrediente) se ejecuta una sentencia de actualización de la cantidad de ingrediente.

4.5 CONCLUSIONES DE LA APLICACIÓN

Como ya he comentado anteriormente en el documento, la realización de esta aplicación es un complemento a la práctica de la base de datos, por ello, contiene una serie de lagunas bastante importantes que podrían hacer fallar el funcionamiento de la misma.

Personalmente me hubiera gustado optimizar el funcionamiento de la aplicación y parchear los posibles errores de validación de los datos, mejorar el diseño UI/UX para que fuera más funcional y asegurar que se cumplen las características ACID de la base de datos, sin embargo, debido al desconocimiento de algunos de los lenguajes utilizados y de no tratarse de la materia de esta práctica, esto no es posible.

A pesar de los posibles errores que contenga espero que, al menos, la aplicación sirva de humilde ejemplo de cómo podría ser una aplicación real.