Conjunto de requisitos de la empresa

DeliverUS

Trabajo realizado por: Avilés Carrera, Francisco
Molins López, Alexis
Pizzano Cerrillos, Andrés
Portela Huerta, José María
Torralba Lanzas, Lorenzo

Índice

1.Introducción

Una empresa de distribución desea desarrollar un sistema de información, para la gestión de los pedidos de clientes a distintos restaurantes de Sevilla.

Los responsables del negocio necesitan un buen sistema que resuelva diversos problemas, como la falta de claridad en los pedidos del restaurante, la escasez de stock, la dificultad que existe al encontrar al cliente, el retraso de muchos pedidos, etc. Por otro lado, existen fallos de reparto, pues algunos restaurantes necesitan mayor cantidad de repartidores que otros.

La empresa busca un medio que facilite y agilice el sistema de envíos de los productos. Una vez planteado el problema, habiendo analizado el negocio y entrevistado a los diferentes responsables, planteamos el glosario de términos, los requisitos, la matriz de trazabilidad y los modelos conceptual, relacional en 3FN y tecnológico.

2. Glosario de términos

Término	Descripción
Área de reparto	Radio de distancia que limita el listado de restaurantes disponibles para un cliente concreto.
Cesta	Conjunto de productos almacenados, previos a la realización del pedido. Contiene el producto y su cantidad.
Código de identificación	Secuencia de caracteres que identifica de forma única un pedido
Estado del pedido	Proceso en el que se encuentra el pedido. Distinguimos entre los siguientes estados: recibido, preparado, en reparto o entregado.
Horario de restaurante	Franja horaria en la que se encuentra abierto el establecimiento.
Incidencia	Perturbación en el proceso de reparto de un pedido. Pueden existir distintos tipos de incidencias: retrasos, fallos de entrega, accidentes durante el transporte, etc.
Método de pago	Forma en la que el cliente abona el precio del pedido, bien por tarjeta de crédito o débito, Paypal o en efectivo.
Pedido	Conjunto de productos demandados por el cliente y enviados a su dirección.
Valoración	Número de estrellas que le da un cliente a cierto restaurante, acompañado de un comentario.

3. Visión general del sistema

3.1. Requisitos generales

OBJ-001 Gestión de usuarios
Como gerente de la empresa DeliverUS. **Quiero** asignar a cada cliente una cuenta de usuario. **Para** dar un mejor servicio a los clientes, dándole un trato personalizado.

OBJ-002 Gestión de Restaurantes
Como gerente de la empresa DeliverUS. **Quiero** que los propietarios registren los datos de un restaurante. **Para** ponerlos a disposición del cliente.

OBJ-003 Gestión de productos
Como gerente de la empresa DeliverUS. **Quiero** que se lleve un control del producto. **Para** que los restaurantes informen de los productos que poseen.

OBJ-004 Gestión de pedidos
Como gerente de la empresa DeliverUS. **Quiero** que se asignen los pedidos a un repartidor. **Para** que se coordinen los pedidos y haya un seguimiento de este.

3.2. Usuarios del sistema

Cliente
Usuario registrado que encarga productos a los restaurantes a través de la empresa DeliverUS.

Propietario del restaurante
Dueño de un establecimiento que gestiona su funcionamiento y se comunica con la empresa encargada de los repartos.

Repartidor
Trabajador de la empresa DeliverUs encargado de la recogida y entrega segura de los pedidos.

4. Catálogo de requisitos

4.1. Requisitos de información

RI-001 Información sobre usuarios
Como gerente de la empresa DeliverUs. **Quiero** disponer de la información correspondiente a los usuarios. Específicamente: nombre de usuario, nombre y apellidos, DNI, correo electrónico, teléfono, dirección, código postal y método de pago. Se diferenciará entre usuario propietario de restaurante, usuario cliente y repartidor, mediante una propiedad que distinguirá el tipo de usuario. **Para** hacer estadísticas dentro de la aplicación, enviar publicidad personalizada y para asegurar que el producto llega al usuario correcto.

RI-002 Información sobre restaurantes
Como gerente de la empresa DeliverUs. **Quiero** disponer de la información correspondiente a los restaurantes. Específicamente: nombre, descripción, dirección, código postal, URL, correo electrónico, número de teléfono, tipo de comida, gastos de envío (por defecto para los pedidos realizados a cada restaurante), tiempo medio de servicio en minutos (que se calculará a partir del registro de pedidos) y estado. El estado es la disponibilidad del restaurante (abierto y cerrado). El usuario propietario gestiona su restaurante y su categoría (elegida entre predefinidos del sistema, por el tipo de comida). **Para** facilitar información al cliente.

RI-003 Información sobre productos
Como gerente de la empresa DeliverUs. **Quiero** disponer de la información correspondiente a los productos. Específicamente: nombre, descripción, precio, imagen, disponibilidad y orden de los productos. Además, se busca conocer la categoría del producto. **Para** poder identificar y conocer el estado de los productos.

RI-004 Información sobre pedidos
Como gerente de la empresa DeliverUs. **Quiero** disponer de la información correspondiente a los pedidos. Específicamente: fecha de creación (cuando el cliente realiza el pedido), fecha de inicio (cuando un restaurante acepta el pedido), fecha de envío (cuando el pedido sale del restaurante) y fecha de entrega (cuando el cliente recibe el pedido), precio total de los productos incluidos, la dirección donde debe ser entregado y los gastos de envío. **Para** conocer datos útiles sobre el pedido y saber los lugares y tramos horarios más frecuentes y su tiempo de entrega.

4.2. Requisitos funcionales

RF-001 Filtrado de horarios
Como cliente. **Quiero** saber qué restaurantes están disponibles. **Para** saber en cuales se pueden realizar pedidos.
PRUEBAS DE ACEPTACIÓN • Cuando el usuario inicia la aplicación aparecerán solo los restaurantes disponibles por el horario. • Si un usuario intenta realizar un pedido a un restaurante no disponible saltará por pantalla un aviso. • Si el restaurante está disponible se le permitirá al usuario realizar el pedido con total normalidad. • Si un restaurante pasa a estar dentro de horario se incorpora a final de la lista de restaurantes disponibles.

RF-002 Filtrado por tipos de comida
Como cliente. **Quiero** que los restaurantes se puedan filtrar por el tipo de comida que ofrecen. **Para** ver en qué restaurantes pedir.
PRUEBAS DE ACEPTACIÓN: • El usuario puede filtrar los restaurantes por los distintos tipos de comida. • Al cerrar y abrir la aplicación tendrá que volver a establecer el filtro. • Los restaurantes solo pueden tener asignados un tipo de comida. • El filtro solo se puede realizar sobre un tipo de comida.

RF-003 Listado de pedidos
Como propietario del restaurante. **Quiero** consultar el número de pedidos realizados al mes. **Para** estudiar la rentabilidad del restaurante.
PRUEBAS DE ACEPTACIÓN: • El usuario propietario al acceder al sistema se le muestra un balance de los pedidos realizados en su restaurante. • Dicho balance se renueva mensualmente. • El balance solo tendrá en cuenta los pedidos realizados satisfactoriamente.

RF-004 Ayuda al repartidor

Como repartidor.

Quiero que se me envíe la dirección de recogida y la entrega, además del identificador del pedido, uno por uno.

Para ubicar el origen y el destino del pedido.

PRUEBAS DE ACEPTACIÓN:

- El usuario repartidor al acceder al sistema se le proporciona en una ventana la información del pedido asignado.
- Si no hay pedido asignado la pantalla se muestra vacía.
- Si el repartidor entra y sale del sistema se restaura la información.
- El repartidor no puede modificar dicha información.

RF-005 Zona de trabajo

Como repartidor.

Quiero elegir la zona en la que voy a trabajar ese día.

Para que se me asignen restaurantes cercanos.

PRUEBAS DE ACEPTACIÓN:

- Al acceder como usuario repartidor por primera deberá elegir su zona.
- Hasta que no se asigne la zona de trabajo el repartidor no podrá empezar a realizar pedidos.
- Si el repartidor está fuera de la zona asignada saltará una pestaña emergente que le impedirá recibir pedidos.
- Al usuario repartidor no se le permitirá cambiar dicha zona.
- Estando dentro de la zona recibirá pedidos con total normalidad.

RF-006 Cantidad de cada producto por pedido

Como cliente.

Quiero tener una lista temporal con los productos seleccionados.

Para poder pedir más de un producto a un restaurante.

PRUEBAS DE ACEPTACIÓN:

- Los productos se añaden uno a uno a la cesta.
- Los productos pueden retirarse de la cesta.
- Si cierras y abres la aplicación tu cesta no se conserva.
- Si accedes a la cesta el cliente puede ver los productos añadidos hasta el momento.

RF-007 Historial del cliente

Como cliente.

Quiero que se recojan todos mis pedidos y se guarden en mi historial.

Para tener acceso a los pedidos realizados.

PRUEBAS DE ACEPTACIÓN:

- El usuario accede al apartado historial y se le muestran los últimos establecimientos a los que accedió y lo que pidió.
- El usuario accede al buscador de establecimientos y se le recomiendan sus establecimientos más recientes en base al historial.
- Solo añaden pedidos al historial si han sido realizados satisfactoriamente.
- El usuario puede borrar pedidos de su historial o restablecerlo completamente.

RF-008 Estado de pedido

Como cliente.

Quiero saber en qué proceso se encuentra mi pedido, entre los distintos estados de pedido, así como si ha habido algún tipo de incidencia.

Para conocer cómo va mi pedido.

PRUEBAS DE ACEPTACIÓN:

- El usuario cliente tras realizar un pedido se le proporciona una barra que guarda los diferentes estados definidos en lo que se puede encontrar un pedido.
- Una vez entregado dicha barra desaparece.
- Si cierras y abres la aplicación la barra se conserva mientras el pedido esté activo.
- Si ocurre una incidencia en el pedido y tiene que ser cancelado la barra desaparece.

4.3. Reglas de negocio

RN-001: Gastos de envío gratuito

Como director de marketing.

Quiero que se cumpla la siguiente regla de negocio: si el precio total del pedido es superior a 10€ los gastos de envío serán 0€ (envío gratuito).

Para fomentar la solicitud de pedidos que además serán de mayor cantidad.

PRUEBAS DE ACEPTACIÓN:

- Una vez añadidos los productos a la cesta con un precio total de 10€ o más la aplicación elimina automáticamente los gastos de envío.
- Si el precio de los productos es inferior a 10€ se añaden de forma predeterminada los gastos de envío.

RN-002: Restricción de pedidos

Como director del departamento de gestión de establecimientos.

Quiero que se cumpla la siguiente regla de negocio: un pedido sólo puede incluir productos de un restaurante.

Para facilitar el reparto de los pedidos.

PRUEBAS DE ACEPTACIÓN:

- Si el cliente intenta solicitar productos de dos o más restaurantes distintos, recibirá un aviso que le impedirá añadir dicho producto.
- Cuando el pedido sea de un único establecimiento se procederá a la tramitación del pedido.
- Los repartidores solo podrán llevar pedidos de un único restaurante.

RN-003: Valoración del restaurante

Como director de marketing.

Quiero que se cumpla la siguiente regla de negocio: Un cliente solo puede tener una valoración por restaurante, como máximo.

Para fomentar la frecuencia de pedidos y hacer mejoras a partir de los comentarios.

PRUEBAS DE ACEPTACIÓN:

- Un cliente solo puede hacer una valoración si ha hecho un pedido a dicho restaurante.
- Si no ha hecho ningún pedido, al intentar introducir la valoración salta un error.
- Si el cliente ya tiene una valoración e intenta añadir una nueva salta error, indicándole que solo puede actualizar su anterior comentario.
- Si el cliente intenta actualizar una valoración sin tener ninguno previo, salta error indicándole que debe añadir uno para poder llegar a actualizar una valoración.
- Si no eres cliente salta error indicando que solo los usuarios cliente pueden hacer una valoración

RN-004: Horario de pedidos

Como director del departamento de coordinación.

Quiero que se cumpla la siguiente regla de negocio: Los pedidos solo se podrán realizar dentro del horario del establecimiento.

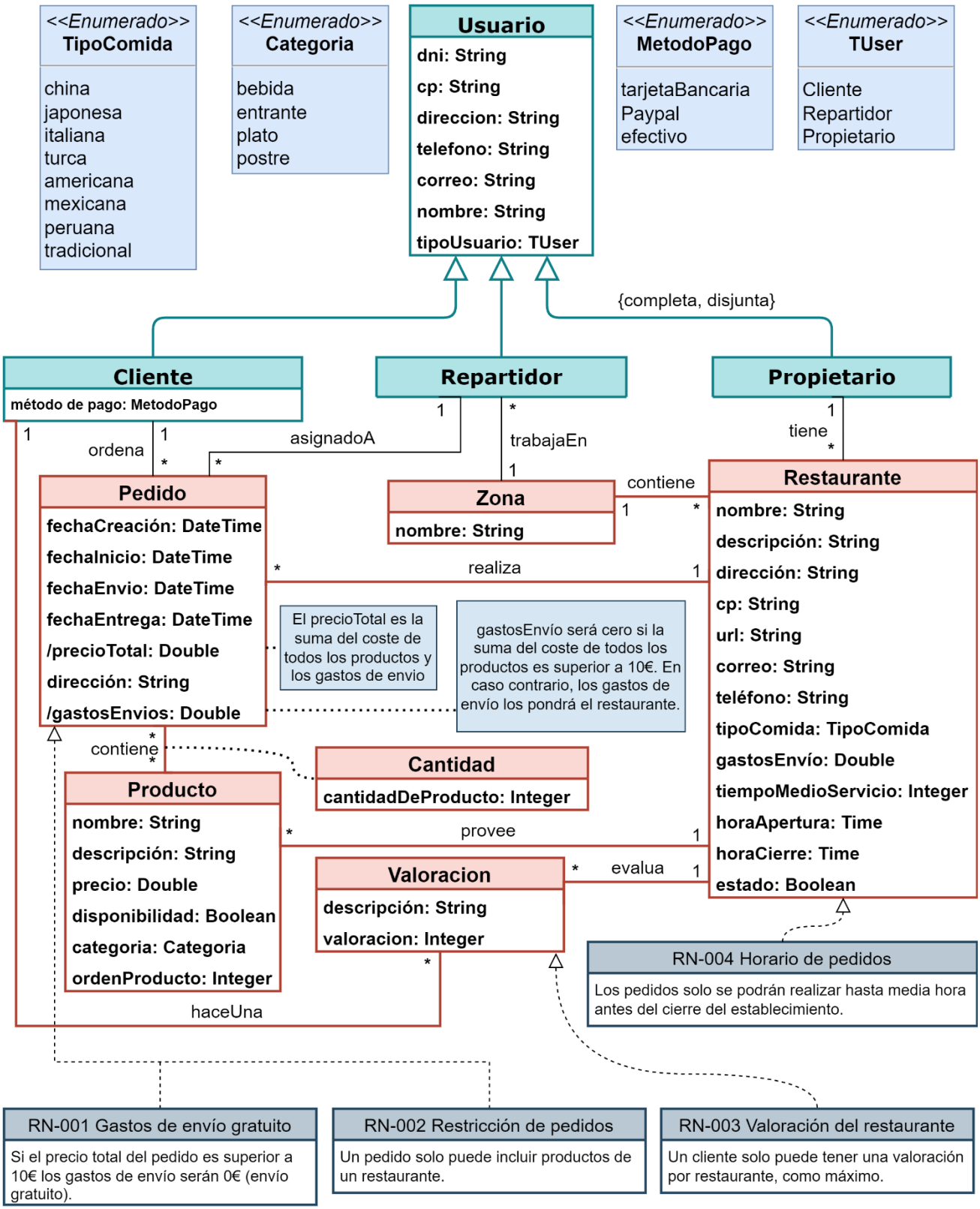
Para facilitar el trabajo de establecimientos y repartidores.

PRUEBAS DE ACEPTACIÓN:

- Hasta la hora de cierre del restaurante, este estará disponible para ordenar pedidos.
- Una vez abierto el restaurante empezará a estar disponible.
- El restaurante debe facilitar el horario al sistema si quiere obtener sus servicios.

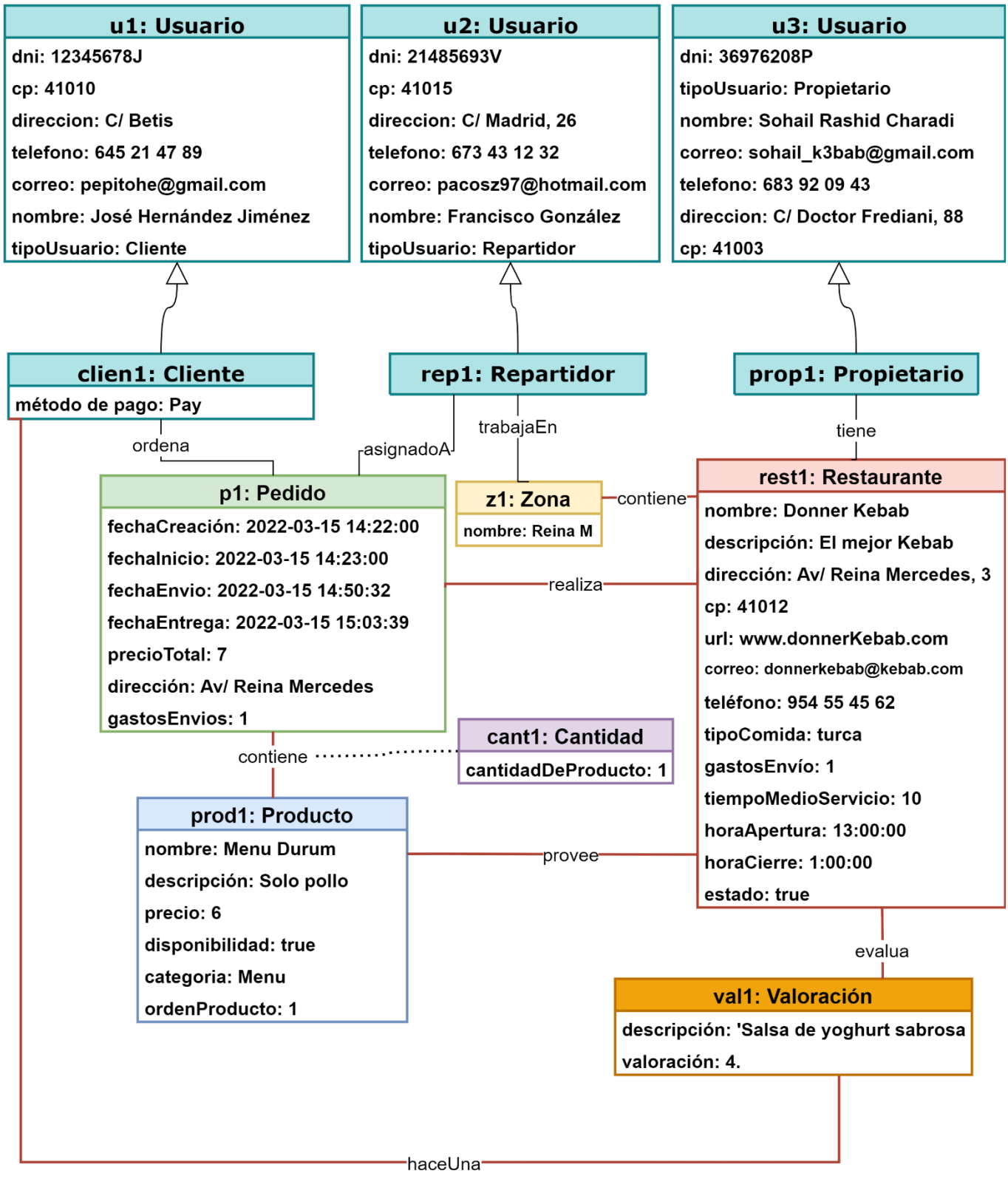
5. Modelo Conceptual

5.1. Modelo

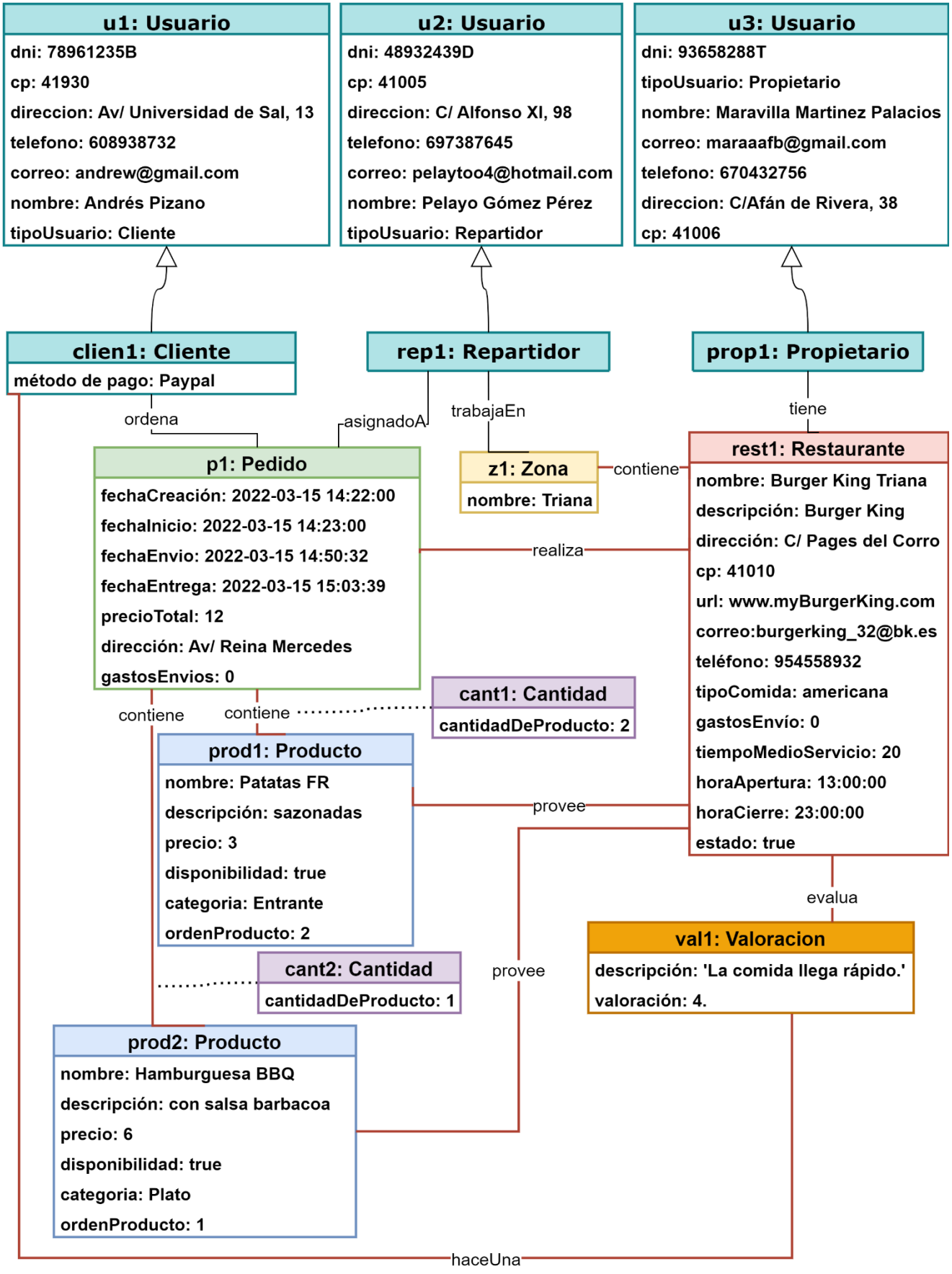


5.2. Escenas

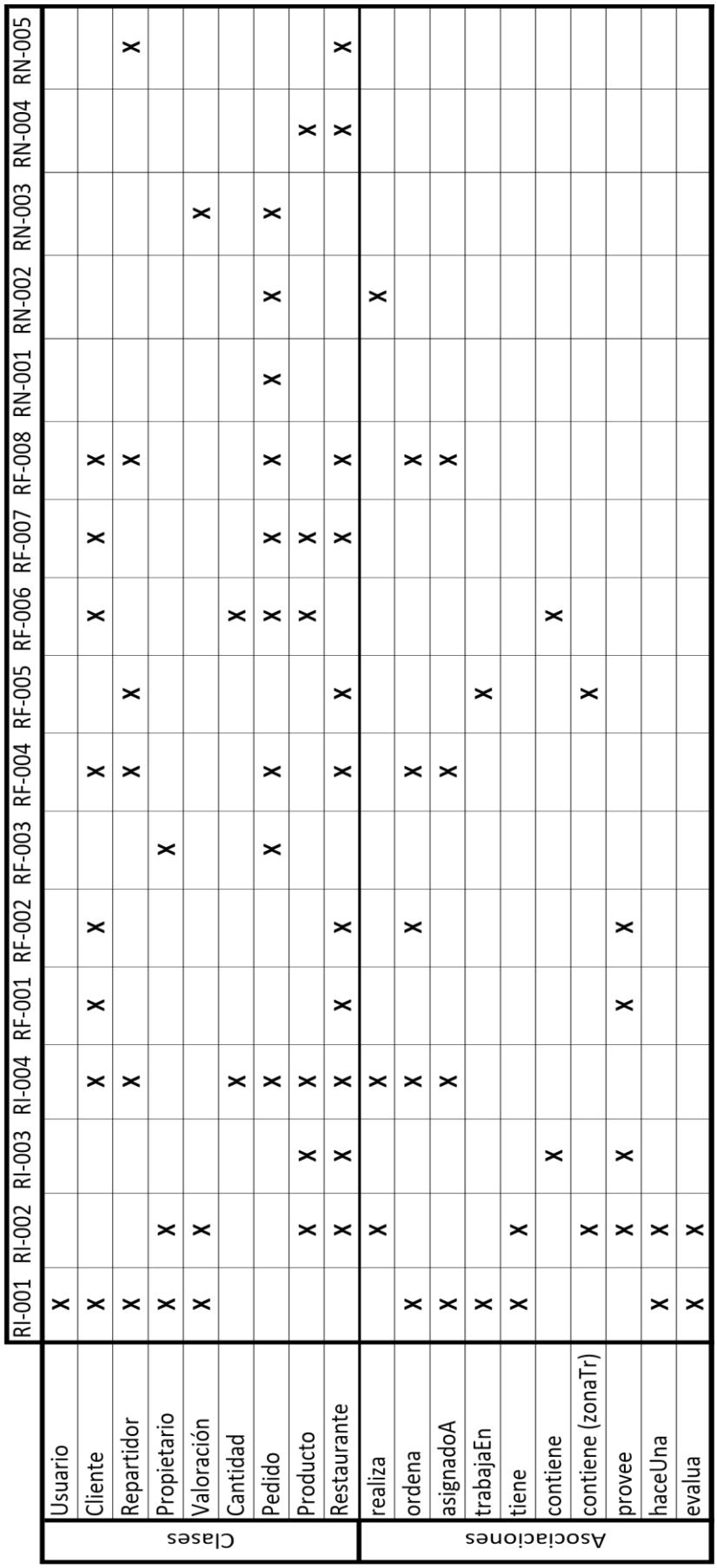
Escenario de un usuario cliente que hace un pedido a un restaurante turco.



Escenario de un usuario que hace un pedido a un Burger King en Triana.

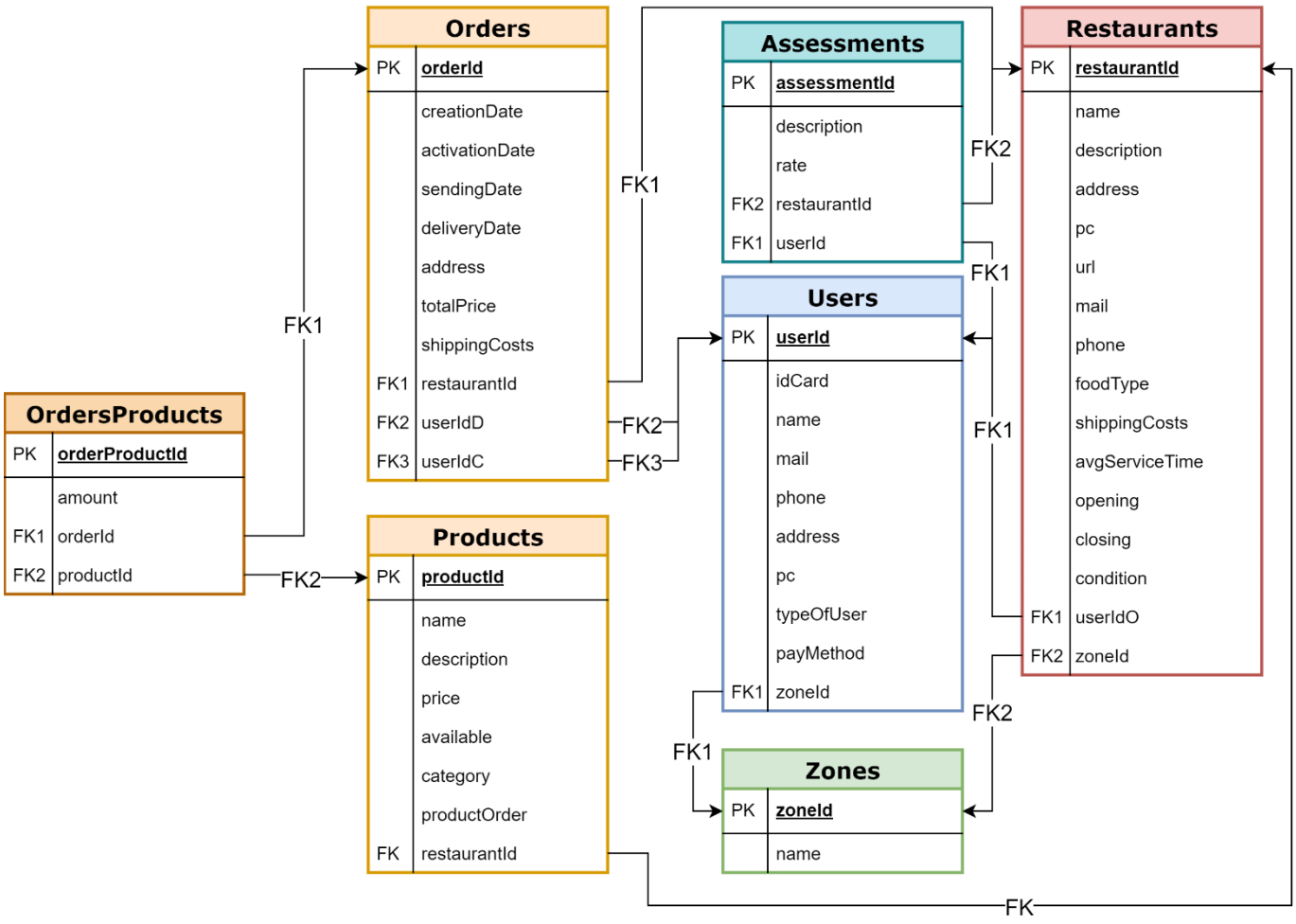


6. Matriz de trazabilidad



		RI-001	RI-002	RI-003	RI-004	RF-001	RF-002	RF-003	RF-004	RF-005	RF-006	RF-007	RF-008	RN-001	RN-002	RN-003	RN-004	RN-005
Clases	Usuario	X																
	Cliente	X			X	X	X		X		X	X	X					
	Repartidor	X			X				X	X			X					X
	Propietario	X	X					X										
	Valoración	X	X													X		
	Cantidad				X						X							
	Pedido				X			X	X		X	X	X	X	X	X		
	Producto		X	X	X						X	X					X	
	Restaurante		X	X	X	X	X		X	X		X	X				X	X
Asociaciones	realiza		X		X										X			
	ordena	X			X		X		X				X					
	asignadoA	X			X				X				X					
	trabajaEn	X								X								
	tiene	X	X															
	contiene			X							X							
	contiene (zonaTr)		X							X								
	provee		X	X		X	X											
	haceUna	X	X															
	evalua	X	X															

7. Modelo Relacional 3FN



Para el Modelo Relacional hemos seguido la estrategia de una superclase, porque a fin de cuentas así tenemos un menor número de tablas con las que trabajar, al salirnos pocos nulls por las pocas propiedades que distinguen a los distintos usuarios.

	PK	AK	FK
Zones(zoneId, name)	zoneId	name (1*)	
Users (userId, idCard, name, mail, phone, address, pc, typeOfUser, payMethod, zoneId)	userId	idCard mail phone	zoneId/Zones
Orders(orderId, creationDate, activationDate, sendingDate, deliveryDate, address, shippingCosts, userIdD, userIdC, restaurantId)	orderId	(creationDate, activationDate, sendingDate, deliveryDate) (2*)	userIdD/ Users(userId)
Products (productId, name, description, price, available, category, productsOrders, restaurantId)	productId	(name, description) (3*)	restaurantId/ Restaurants
OrdersProducts(orderProductId, amount, orderId, productId)	orderProductId	(orderId, productId)	orderId/ Orders productId/ Products
Restaurants(restaurantId, name, description, adress, pc, url, mail, phone, foodType, shippingCosts, avgServiceTime, opening, closing, condition, userIdO, zoneId)	restaurantId	url mail phone (name, description, address)	userIdO/ Users(userId) zoneId/ Zones
Assessments(assessmentId, description, rate, userIdC, restaurantId)	assessmentId	(userId, restaurantId)	userIdC/ Users(userId) restaurantId/ Restaurants

(1*): Es razonable pensar que en una ciudad dos barriadas no van a tener el mismo nombre.

(2*): Es razonable pensar que 2 pedidos no van a tener las mismas fechas de todo, hasta el último segundo.

(3*): Es razonable pensar que 2 productos no van a compartir descripción y nombre, puesto que puede cambiar un poco el nombre o cómo se presenta el plato.

8. Modelo tecnológico

PASOS PARA LA PRUEBA DE SQL

1. Crear la base de datos deliver_us y seleccionarla
2. Ejecutar los scripts en el siguiente orden:

   TABLES, CURSOR, TRIGGERS, POPULATE

3. Las funciones y procedimientos se pueden ejecutar en el orden que se desee

SCRIPTS

Script 0: TABLES

```
DROP TABLE IF EXISTS assessments;
DROP TABLE IF EXISTS inactiveClients;
DROP TABLE IF EXISTS orderProducts;
DROP TABLE IF EXISTS products;
DROP TABLE IF EXISTS orders;
DROP TABLE IF EXISTS restaurants;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS zones;

CREATE TABLE zones(
	zoneId INT NOT NULL AUTO_INCREMENT,
	`name` VARCHAR(100) NOT NULL UNIQUE,
	PRIMARY KEY (zoneId)
);

CREATE TABLE users(
	userId INT NOT NULL AUTO_INCREMENT,
	idCard CHAR(9) NOT NULL UNIQUE,
	`name` VARCHAR(100) NOT NULL,
	mail VARCHAR(100) NOT NULL UNIQUE,
	phone CHAR(9) NOT NULL UNIQUE,
	address VARCHAR(100) NOT NULL,
	pc CHAR(5) NOT NULL,
	typeOfUser VARCHAR(100) NOT NULL,
	payMethod VARCHAR(100),
	zoneId INT,
	FOREIGN KEY(zoneId) REFERENCES zones (zoneId),
	CONSTRAINT invalidPayMethod CHECK (payMethod IN ('PayPal','Efectivo','TarjetaBancaria')),
	CONSTRAINT invalidTypeOfUser CHECK (typeOfUser IN ('Cliente','Repartidor','Propietario')),
	PRIMARY KEY(userId)
);

CREATE TABLE restaurants(
	restaurantId INT NOT NULL AUTO_INCREMENT,
	`name` VARCHAR(100) NOT NULL,
	description VARCHAR(200) NOT NULL,
	address VARCHAR(100) NOT NULL,
	pc CHAR(5) NOT NULL,
	url VARCHAR(100) NOT NULL UNIQUE,
	mail VARCHAR(100) NOT NULL UNIQUE,
	phone CHAR(9) NOT NULL UNIQUE,
	foodType VARCHAR(100) NOT NULL,
	shippingCosts DOUBLE NOT NULL,
	avgServiceTime INT NOT NULL,
	opening TIME NOT NULL,
	closing TIME NOT NULL,
	`condition` BOOLEAN NOT NULL,
	userIdO INT NOT NULL,
	zoneId INT NOT NULL,
	PRIMARY KEY (restaurantId),
	FOREIGN KEY (userIdO) REFERENCES users (userId),
	FOREIGN KEY (zoneId) REFERENCES zones(zoneId),
	CONSTRAINT invalidAvgServiceTime CHECK (avgServiceTime>0),
	CONSTRAINT invalidShippingCosts CHECK (shippingCosts>=0),
```

```
	CONSTRAINT invalidTime CHECK (closing>opening),
	CONSTRAINT invalidFoodType CHECK (foodType IN ('China','Japonesa','Italiana','Turca','Americana','Peruana',
                                                   'Mejicana','Tradicional'))
);

CREATE TABLE orders(
	orderId INT NOT NULL AUTO_INCREMENT,
	creationDate DATETIME NOT NULL,
	activationDate DATETIME NOT NULL,
	sendingDate DATETIME NOT NULL,
	deliveryDate DATETIME NOT NULL,
	address VARCHAR(100) NOT NULL,
	totalPrice DOUBLE,
	shippingCosts DOUBLE,
	userIdD INT NOT NULL,
	userIdC INT NOT NULL,
	restaurantId INT NOT NULL,
	PRIMARY KEY(orderId),
	FOREIGN KEY (userIdD) REFERENCES users (userId),
	FOREIGN KEY (userIdC) REFERENCES users (userId),
	FOREIGN KEY (restaurantId) REFERENCES restaurants (restaurantId),
	CONSTRAINT invalidShippingCosts CHECK (shippingCosts>=0),
	CONSTRAINT invalidTime CHECK (creationDate<activationDate AND
                                  activationDate<sendingDate AND sendingDate<deliveryDate)
);

CREATE TABLE products(
	productId INT NOT NULL AUTO_INCREMENT,
	`name` VARCHAR(100) NOT NULL,
	description VARCHAR(100) NOT NULL,
	price DOUBLE NOT NULL,
	available BOOLEAN NOT NULL,
	category VARCHAR(100) NOT NULL,
	productOrder INT NOT NULL,
	restaurantId INT NOT NULL,
	PRIMARY KEY (productId),
	FOREIGN KEY (restaurantId) REFERENCES restaurants (restaurantId),
	CONSTRAINT invalidPrice CHECK (price>0),
	CONSTRAINT invalidProductOrder CHECK (productOrder>=0),
	CONSTRAINT invalidCategory CHECK(category IN ('Bebida','Entrante','Plato','Postre'))
);

CREATE TABLE orderProducts(
	orderProductId INT NOT NULL AUTO_INCREMENT,
	amount INT NOT NULL,
	orderId INT NOT NULL,
	productId INT NOT NULL,
	PRIMARY KEY (orderProductId),
	FOREIGN KEY (orderId) REFERENCES orders (orderId),
	FOREIGN KEY (productId) REFERENCES products (productId),
	CONSTRAINT invalidAmount CHECK (amount>0)
);

CREATE TABLE inactiveClients(
	userIdC INT NOT NULL,
	FOREIGN KEY (userIdC) REFERENCES users (userId)
);

CREATE TABLE assessments(
	assessmentId INT NOT NULL AUTO_INCREMENT,
	description VARCHAR(500) NOT NULL,
	rate INT NOT NULL,
	userIdC INT NOT NULL,
	restaurantId INT NOT NULL,
	PRIMARY KEY(assessmentId),
	FOREIGN KEY (userIdC) REFERENCES users (userId),
	FOREIGN KEY (restaurantId) REFERENCES restaurants (restaurantId),
	CONSTRAINT invalidRate CHECK (rate>=1 AND rate<=5)
);
```

Script 1: CURSOR

```
CREATE OR REPLACE VIEW viewTotalPrice AS
```

```
        SELECT orderproducts.amount, products.price FROM orders
                        JOIN orderproducts ON (orders.orderId=orderproducts.orderId)
                        JOIN products ON (orderproducts.productId=products.productId);

DELIMITER //
CREATE OR REPLACE FUNCTION PT(orderId INT) RETURNS DOUBLE
BEGIN
        DECLARE done BOOLEAN;
        DECLARE price DOUBLE;
        DECLARE register ROW TYPE OF viewTotalPrice;
        DECLARE productsPrice CURSOR FOR SELECT orderproducts.amount, products.price FROM orders
                        JOIN orderproducts ON (orders.orderId=orderproducts.orderId)
                        JOIN products ON (orderproducts.productId=products.productId)
                        WHERE (orders.orderId=orderId);

        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done := TRUE;
        SET price= 0;
        OPEN productsPrice;
        readLoop: LOOP
                FETCH productsPrice INTO register;
                IF done THEN
                        LEAVE readLoop;
                END IF;
                        SET price= price + register.amount*register.price;
        END LOOP;
        CLOSE productsPrice;
        RETURN price;
END //
DELIMITER ;

SELECT PT(4);
```

Script 2: TRIGGERS

```
DELIMITER //
CREATE OR REPLACE TRIGGER triggerProductsSameRestaurant
BEFORE INSERT ON orderProducts
FOR EACH ROW
BEGIN
        DECLARE newResId INT;
        DECLARE resId INT;

        SET newResId= (SELECT restaurants.restaurantId  FROM products
                JOIN restaurants ON (products.restaurantId= restaurants.restaurantId) WHERE (products.productId = new.productId));

        SET resId= (SELECT DISTINCT(restaurants.restaurantId) FROM orders
                JOIN orderProducts ON (orders.orderId=orderProducts.orderId)
                JOIN products ON (orderProducts.productId=products.productId)
                JOIN restaurants ON (products.restaurantId= restaurants.restaurantId)
                WHERE (orders.orderId= new.orderId));

        IF (newResId!=resId) THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT= 'It is not allowed to add products FROM another restaurant
                                                                                                to the order.';
        END IF;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE TRIGGER triggerOrderOutOfHours
AFTER INSERT ON orderProducts
FOR EACH ROW
BEGIN
        DECLARE idRestaurantOrder INT;
        DECLARE hourIntroducing TIME;
        DECLARE hourOpeningRes TIME;
        DECLARE hourClosingRes TIME;

        SET idRestaurantOrder=(SELECT restaurantId FROM orderProducts
                        JOIN products ON (orderProducts.productId=products.productId)
                        WHERE orderProducts.orderProductId=NEW.orderProductId);

        SET hourIntroducing=(SELECT TIME(creationDate) FROM orders JOIN orderProducts ON
```

```
                                                                                (orders.orderId=orderProducts.orderId)
        WHERE orderProducts.orderProductId=NEW.orderProductId);

        SET hourOpeningRes=(SELECT opening FROM restaurants WHERE idRestaurantOrder=restaurants.restaurantId);
        SET hourClosingRes=(SELECT closing FROM restaurants WHERE idRestaurantOrder=restaurants.restaurantId);

        IF(hourIntroducing<hourOpeningRes OR hourIntroducing>hourClosingRes) THEN
                SET @error_message=CONCAT(idRestaurantOrder,'The restaurant you are trying to order is closed right now,
                                                                        opening: ',hourOpeningRes,' closing: ',hourClosingRes);
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =@error_message;
        END IF;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE TRIGGER triggerCalculateTotalPriceAndShCosts
AFTER INSERT ON orderproducts
FOR EACH ROW
BEGIN
        DECLARE realTotalPrice DOUBLE;
        DECLARE realShCosts DOUBLE;

        SET realShCosts=(SELECT restaurants.shippingCosts FROM orders
                        JOIN restaurants ON(restaurants.restaurantId=orders.restaurantId) WHERE orders.orderId
                                                                                =NEW.orderId);
        SET realTotalPrice=PT(NEW.orderId);
        UPDATE orders SET shippingCosts=realShCosts WHERE orderId=NEW.orderId;
        UPDATE orders SET totalPrice=realTotalPrice WHERE orderId=NEW.orderId;
        IF(realTotalPrice>10)THEN
                UPDATE orders SET shippingCosts=0 WHERE orderId=NEW.orderId;
        END IF;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE TRIGGER triggerOwnerId
BEFORE INSERT ON restaurants
FOR EACH ROW
BEGIN
        DECLARE typeOfUser VARCHAR(100);
        SET typeOfUser= (SELECT typeOfUser FROM users WHERE(users.userId=NEW.userIdO));
        IF (typeOfUser!='Propietario') THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The user must be an owner.';
        END IF;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE TRIGGER triggerClientId
BEFORE INSERT ON orders
FOR EACH ROW
BEGIN
        DECLARE typeOfUser VARCHAR(100);
        SET typeOfUser= (SELECT typeOfUser FROM users WHERE(users.userId=NEW.userIdC));
        IF (typeOfUser!='Cliente') THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The user must be a client.';
        END IF;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE TRIGGER triggerDeliverymanId
BEFORE INSERT ON orders
FOR EACH ROW
BEGIN
        DECLARE typeOfUser VARCHAR(100);
        SET typeOfUser= (SELECT typeOfUser FROM users WHERE(users.userId=NEW.userIdD));
        IF (typeOfUser!='Repartidor') THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The user must be a deliveryman.';
        END IF;
END //
DELIMITER ;

DELIMITER //
```

```
CREATE OR REPLACE TRIGGER triggerAssessment
BEFORE INSERT ON assessments
FOR EACH ROW
BEGIN
	DECLARE typeOfUser VARCHAR(100);
	DECLARE numberOrders INT;
	SET typeOfUser= (SELECT typeOfUser FROM users WHERE(users.userId=NEW.userIdC));
	SET numberOrders= (SELECT COUNT(*) FROM orders WHERE(orders.restaurantId=NEW.restaurantId AND orders.userIdC=
                                                                                                NEW.userIdC));
	IF (typeOfUser!='Cliente') THEN
		SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The user must be a client.';
	END IF;
	IF (numberOrders<1) THEN
		SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The client has not placed an order with that restaurant.';
	END IF;
END //
DELIMITER ;
```

Script 3: POPULATE

```
DELETE FROM orderProducts;
DELETE FROM products;
DELETE FROM orders;
DELETE FROM restaurants;
DELETE FROM users;
DELETE FROM zones;

INSERT INTO zones(zoneId,`name`) VALUES
(1,'Sevilla Capital'),
(2,'Aljarafe'),
(3,'Pino Montano'),
(4,'Triana'),
(5,'Los Remedios');

INSERT INTO users(userId,idCard,`name`,mail,phone,address,pc,typeOfUser,payMethod,zoneId) VALUES
(1, '67898765R', 'Kevin Martínez Monedero', 'er_keviiin@hotmail.es', '608958742', 'Avda Reina Mercedes 41', '41012','Repartidor',NULL,1),
(2, '78961235B', 'Andrés Pizzano Cerrillos', 'andrew@gmail.com', '608938732', 'Avda Universidad de Salamanca 13',
                                                                                                '41930','Repartidor',NULL,4),
(3, '67845625Z', 'Lorenzo Torralba Lanzas', 'ellorenso21@gmail.com', '608958312', 'Avda Reina Mercedes 3', '41012','Repartidor',NULL,3),
(4, '67898765A', 'Francisco Avilés Carrera', 'gemeliersss@outlook.com', '609968742', 'Avda Al Margen 5', '41930','Repartidor',NULL,2),
(5, '67893333V', 'Alexis Molins López', 'biris_norte55@hotmail.es', '651583225', 'Avda de la Inspiración', '41014','Repartidor',NULL,5),
(6, '67895566J', 'Jose María Portela Huerta', 'josemaria86@gmail.com', '608998745', 'Calle Piodoce 15', '41234','Propietario',NULL,NULL),
(7, '67894567A', 'Sohail Rashid Charadi', 'sohashisha2@kebab.com', '608555444', 'Avda Los Remedios', '41015','Propietario',NULL,NULL),
(8, '54321678N', 'Pedro Jiménez Pérez', 'ximenezjp@hotmail.es', '652345675', 'Avda Reina Mercedes 8', '41013','Propietario',NULL,NULL),
(9, '67898866P', 'William Olankawa Iniesta', 'wiolaninies@hotmail.es', '693214567', 'Calle Sierpes 3', '41012','Propietario',NULL,NULL),
(10, '67888962W', 'Pedro Sánchez Pérez-Castejón', 'pedritosanche5@hotmail.es', '608665522', 'Avda Moncloa s/n',
                                                                                                '41010','Cliente','PayPal',NULL),
(11, '67888999A', 'Pablo Iglesias Turrión', 'coletaforever5@outlook.es', '608689522', 'Avda Moncloa s/n', '41010','Cliente','PayPal',NULL),
(12, '67877777M', 'Mariano Rajoy Brey', 'elpuebloespañol@hotmail.es', '608555666', 'Avda Moncloa s/n', '41010','Cliente','Efectivo',NULL),
(13, '67888332D', 'Linus Benedict Torvalds', 'linuxbestso@hotmail.es', '608875612', 'Calle Betis 12', '41010','Cliente','TarjetaBancaria',NULL),
(14, '67882424X', 'William Henry Gates', 'windowsbestso@hotmail.es', '608857891', 'Calle Betis 13', '41930','Cliente','TarjetaBancaria',NULL),
(15, '67888962U', 'Steven Paul Jobs', 'appleisbetter@hotmail.es', '608785412', 'Calle Betis 14', '41010','Cliente','TarjetaBancaria',NULL),
(16, '65897412M', 'Juan Díaz Álvarez', 'jdiazdiazdias@hotmail.es', '608785419', 'Calle Betis 22', '41930','Propietario',NULL,NULL),
(17, '78459632N', 'Cristiano Ronaldo dos Santos', 'cr7suu@cr.com', '457893612', 'Calle margarita 40',
                                                                                                '41930','Cliente','TarjetaBancaria',NULL),
(18, '78789855M', 'Pau Gasol Sáez', 'paupaupivot@gmail.es', '605231454', 'Calle Béquer 7', '41010','Repartidor',NULL,3),
(19, '88896325L', 'Peter Benjamin Parker', 'gafitas33@hotmail.es', '666321255', 'Calle Lope de Vega 1',
                                                                                                '41925','Cliente','TarjetaBancaria',NULL),
(20, '98741236C', 'Lionel Andrés Messi', 'messinashee33@gmail.es', '321456987', 'Calle Béquer 5', '41010','Propietario',NULL,NULL),
(21, '45457878R', 'Luis Enrique Martínez García', 'luispadrique@hotmail.es', '608885413', 'Calle Dos Hermanas 5',
                                                                                                '41907','Propietario',NULL,NULL);

INSERT INTO restaurants(restaurantId,`name`,description,address,pc,url,mail,phone,foodType,
                                                shippingCosts,avgServiceTime,opening,closing,`condition`,userIdO,zoneId) VALUES
(1, 'BurgenKing', 'Restaurante de comida rápida especializado en hamburguesas', 'C/Pages del Corro','41930',
                        'www.myBurgerKing.com','burgerking_32@bk.es', '954558932', 'Americana', 1.6, 20, '13:00:00', '23:00:00', TRUE, 6, 1),
(2, 'Kebab Reina Mercedes', 'Establecimiento especializado en kebabs', 'Av/ Reina Mercedes','41930',
                        'www.KebabTurco.com','kebabturk@gmail.com', '967823456', 'Turca', 1.3, 10, '13:00:00', '23:00:00', TRUE, 7 , 2),
(3, 'Papa Johns', 'Pizzas y comida rápida a muy buen precio', 'Av/ Reina Mercedes','41930', 'www.papajohns.com','papajhons_76@ppj.es',
                                                '659342576', 'Americana', 2.5, 30, '12:00:00', '22:30:00', TRUE, 8, 3),
(4, 'Goiko Grill', 'Hamburguesas deluxe calidad precio', 'C/ Albareda 14', '41930','www.GoikoGrill.com','goikoalbareda@gk.es','734098342',
                                                'Americana', 2.2, 30, '20:00:00', '23:00:00', TRUE, 9, 4),
(5, 'Casa Paco', 'Comida tradicional española para toda la familia', 'C/ Albareda  24',
```

```
                    '41930','www.casaPaco.com','casapaco33@gmail.com','966322145', 'Tradicional', 3.3, 30, '20:00:00', '23:00:00', TRUE, 21, 4),
(6, 'Iguanas Ranas', 'Comida mejicana tradicional de buena calidad', 'C/ Reyes Católicos 18','41930',
                    'www.iguanasranas.com','iguanasreyesc@gmail.com', '955512067','Mejicana',3.5, 25, '12:00:00', '23:30:00', TRUE , 21,1),
(7, 'Asador Argentino Tradicional', 'Carne Argentina en abundancia', 'C/ Alférez 20','41930',
                    'www.argentinoss5.com','argentinoss5@gmail.com', '954121245','Tradicional',2.1, 30, '13:00:00', '23:30:00', TRUE , 14,4),
(8, 'El tortellini', 'Comida italiana tradicional de buena calidad', 'C/ Reyes Católicos 35','41930',
                    'www.eltortini4545.com','eltortini4545@gmail.com', '955122132','Italiana',1.6, 40, '12:30:00', '23:30:00', TRUE , 16,5);

INSERT INTO products(productId,`name`,description,price,available,category,productOrder,restaurantId) VALUES
(1,'CocaCola','Bebida refrescante de frutas',1.2,TRUE,'Bebida',1,1),
(2,'Fanta','Bebida refrescante de naranja',1.2,TRUE,'Bebida',1,6),
(3,'Nestea','Bebida refrescante de te frío',1.2,TRUE,'Bebida',1,5),
(4,'Patatas Fritas','Patatas fritas en aceite de oliva',3.2,TRUE,'Entrante',2,1),
(5,'Nachos','Nachos mejicanos con guacamole',3.5,TRUE,'Entrante',2,6),
(6,'Ensaladilla','Ensaladilla de patata mahonesa y gambas',3.7,TRUE,'Entrante',2,5),
(7,'Hamburguesa','Hamburguesa de ternera completa',8.2,TRUE,'Plato',2,1),
(8,'Quesadillas','Quesadillas clásicas mejicanas',8.3,TRUE,'Plato',2,6),
(9,'Tortilla de patatas','Tortilla española con cebolla',8.9,TRUE,'Plato',2,5),
(10,'Tortitas','Tortitas de huevo y harina',5.3,TRUE,'Postre',2,1),
(11,'Flan','Flan de huevo con caramelo dulce',5.2,TRUE,'Postre',2,6),
(12,'Arroz con leche','Arroz con leche y canela de la abuela',5.1,TRUE,'Postre',2,5),
(13,'Casera','Bebida refrescante con gas y pocas calorías',1.2,TRUE,'Bebida',1,8),
(14,'Pizza Barbacoa','Pizza con carne y salsa barbacoa',8.6,TRUE,'Plato',2,8),
(15,'Coulant de chocolate','Postre de chocolate y helado de vainilla',6.7,TRUE,'Postre',3,8);

INSERT INTO orders(orderId,creationDate,activationDate,sendingDate,deliveryDate,address,totalPrice,shippingCosts,userIdC,
                                                                                    userIdD,restaurantId) VALUES
(1,'2022-03-02 13:55:05','2022-03-02 13:56:05','2022-03-02 13:59:05','2022-03-02 14:10:00','Calle Piodoce 15',0,0,11,1,1),
(2,'2022-03-05 22:03:55','2022-03-05 22:04:55','2022-03-05 22:07:55','2022-03-05 22:30:55','Avda de la Inspiración',0,0,19,2,6),
(3,'2022-03-04 20:00:00','2022-03-04 20:01:00','2022-03-04 20:04:00','2022-03-04 20:26:00','Calle Betis 13',0,0,12,3,5),
(4,'2022-03-04 15:00:00','2022-03-04 15:01:00','2022-03-04 15:04:00','2022-03-04 15:26:00','Calle Alfonso XII',0,0,17,4,8);

INSERT INTO orderProducts(orderProductId,amount,orderId,productId) VALUES
(1,2,1,1),
(2,2,1,4),
(3,3,1,7),
(4,2,2,2),
(5,3,2,5),
(6,4,2,8),
(7,1,3,3),
(8,1,3,6),
(9,2,4,13),
(10,1,4,14),
(11,2,4,15);

INSERT INTO assessments(assessmentId,description,rate,userIdC,restaurantId) VALUES
(1,'No es un BurgerKing se hace llamar burgenking y es el mayor robo que he visto, hamburguesas nefastas y establecimiento sucio',1,11,1),
(2,'Comida demasiado picante y tardan mucho en atenderte',1,19,6),
(3,'la mejor hambourguesa q mecomío',5,12,5);
```

Script 4: FUNCTIONS

```
DELIMITER //
CREATE OR REPLACE FUNCTION funcRestaurantsOpen(resName VARCHAR(100),consult TIME) returns VARCHAR(100)
BEGIN
        DECLARE c VARCHAR(100);
        DECLARE hO TIME;
        DECLARE hC TIME;

        SET c='CLOSED';
        SET hO=(SELECT opening FROM restaurants WHERE restaurants.name=resName);
        SET hC=(SELECT closing FROM restaurants WHERE restaurants.name=resName);
        IF(hO<=consult AND consult<=hC) THEN
                SET c='OPEN';
        END IF;
        RETURN c;
END //
DELIMITER ;
DELIMITER //
CREATE OR REPLACE FUNCTION funcAvgOrders(nombreRepartidor VARCHAR(100)) RETURNS DOUBLE
BEGIN
        DECLARE avgHour DOUBLE;
        SET avgHour=(SELECT AVG(hour(orders.creationDate)) FROM orders
```

```
                                JOIN users ON(users.userId=orders.userIdD));
        RETURN avgHour;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE FUNCTION funcAmountOrdersPerMonth(url VARCHAR(100), m INT) RETURNS INT
BEGIN
        DECLARE ordersPerMonth INT;
        SET ordersPerMonth= (SELECT COUNT(*) FROM orders
                                JOIN restaurants ON (orders.restaurantId=restaurants.restaurantId)
                                WHERE(restaurants.url=url AND MONTH(orders.deliveryDate)=m));
        RETURN ordersPerMonth;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE FUNCTION funcDeliveryHelp(idOrder INT) RETURNS VARCHAR(200)
BEGIN
        DECLARE addressRestaurant VARCHAR(100);
        DECLARE addressUser VARCHAR(100);

        SET addressRestaurant =(SELECT restaurants.address FROM restaurants
                                JOIN orders ON (restaurants.restaurantId=orders.restaurantId)
                                WHERE (orders.orderId=idOrder));
        SET addressUser = (SELECT users.address FROM orders
                                JOIN users ON (orders.userIdC=users.userId)
                                WHERE orders.orderId=idOrder);
        RETURN CONCAT("La dirección del restaurante es: ",addressRestaurant,
                                                        " y la del cliente es: ",addressUser, " para el pedido con id: ", idOrder);
END//
DELIMITER ;

DELIMITER //
CREATE OR REPLACE FUNCTION funcSumUpDelivery(idOrder INT) RETURNS VARCHAR(200)
BEGIN
        DECLARE cDate DATETIME;
        DECLARE dDate DATETIME;
        DECLARE resName VARCHAR(100);
        DECLARE minuteSpent INT;
        DECLARE priceOrder DOUBLE;
        DECLARE clientOrder VARCHAR(100);
        DECLARE deliverManOrder VARCHAR(100);

        SET clientOrder=(SELECT users.name FROM orders JOIN users ON (orders.userIdC=users.userId) WHERE
                                                                                orders.orderId=idOrder);
        SET deliverManOrder=(SELECT users.name FROM orders
                                JOIN users ON (orders.userIdD=users.userId)
                                WHERE orders.orderId=idOrder);
        SET resName=(SELECT restaurants.name FROM orders
                                JOIN restaurants ON (orders.restaurantId=restaurants.restaurantId)
                                WHERE orders.orderId=idOrder);
        SET priceOrder=PT(idOrder)+(SELECT shippingCosts FROM orders WHERE orderId=idOrder);
        SET cDate=(SELECT creationDate FROM orders WHERE orderId=idOrder);
        SET dDate=(SELECT deliveryDate FROM orders WHERE orderId=idOrder);
        SET minuteSpent=(TIMESTAMPDIFF(MINUTE,cDate,dDate));

        RETURN CONCAT('PEDIDO ',idOrder,'-> DURACIÓN=',minuteSpent,' minutos | PRECIO=',priceOrder,'€ |
                                RESTAURANTE=',resName, ' | REPARTIDOR ASIGNADO=',deliverManOrder,' | CLIENTE=',clientOrder);
END //
DELIMITER ;
```

Script 5: PROCEDURE

```
DELIMITER //
CREATE OR REPLACE PROCEDURE procInsertUserDeliver(id CHAR(9),nameD VARCHAR(100),mailD VARCHAR(100),phoneD
                                                        CHAR(9),addressD VARCHAR(100),pcD CHAR(5), zoneD INT)
BEGIN
        INSERT INTO users(userId,idCard,`name`,mail,phone,address,pc,typeOfUser,payMethod,zoneId) VALUES
                                                        (0,id,nameD,mailD,phoneD,addressD,pcD,'Repartidor',NULL,zoneD);
END //
DELIMITER ;
```

```
DELIMITER //
CREATE OR REPLACE PROCEDURE procChangeZoneDeliverman(id CHAR(9) ,nameD VARCHAR (100))
BEGIN
	DECLARE newZoneId INT;
	DECLARE newUserId INT;
	SET newZoneId= (SELECT zoneId FROM zones WHERE (zones.name=nameD));
	SET newUserId= (SELECT userId FROM users WHERE (users.idCard=id));
	UPDATE users SET zoneId=newZoneId WHERE (users.userId=newUserId);
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE PROCEDURE procChangeAmountProduct(idOrder INT,productName VARCHAR(100),amountProduct INT)
BEGIN
	DECLARE idProduct INT;
	SET idProduct=(SELECT productId FROM products WHERE products.name=productName);
	UPDATE orderProducts SET amount=amountProduct WHERE (orderId=idOrder AND productId=idProduct);
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE PROCEDURE procInactiveClient(months INT)
BEGIN
	DECLARE oldestOrder DATETIME;
	DECLARE diffTime INT;
	DECLARE done BOOLEAN;
	DECLARE clien ROW TYPE OF users;
	DECLARE clientsCursor CURSOR FOR SELECT * FROM users WHERE (users.typeOfUser='Cliente');
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET done := TRUE;
	DELETE FROM inactiveClients;

	OPEN clientsCursor;
		readLoop: LOOP
			FETCH clientsCursor INTO clien;
			IF done THEN
				LEAVE readLoop;
			END IF;
			SET oldestOrder= oldestOrder(clien.userId);
			IF (oldestOrder IS NOT NULL) THEN
				SET diffTime= timeHelp(oldestOrder);
				IF (diffTime>months) THEN
					INSERT INTO inactiveClients(userIdC) VALUES (clien.userId);
				END IF;
			END IF;
		END LOOP;
	CLOSE clientsCursor;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE FUNCTION timeHelp(dateStart DATETIME) RETURNS INT
BEGIN
	DECLARE months INT;
	SET months= TIMESTAMPDIFF(MONTH, dateStart, NOW());
	RETURN months;
END //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE FUNCTION oldestOrder(clientId INT) RETURNS DATETIME
BEGIN
	DECLARE maxDeliveryTime DATETIME;
	SET maxDeliveryTime= (SELECT MIN(deliveryDate) FROM orders WHERE (orders.userIdC=clientId));
	RETURN maxDeliveryTime;
END //
DELIMITER ;
```

Todo el código y el sql lo puedes también hallar en mi repositorio de GitHub:

Enlace a mi repositorio