

Guatemala 25 de enero de 2026.

Universidad De San Carlos De Guatemala.

Laboratorio software avanzado sección "A".

Auxiliar: Juan Pablo Samayoa Ruiz.



Practica #1: Diseño De Arquitectura y Base De Datos Para Delivereats.

Ciclo Escolar 2026

Estudiante: Juan Carlos Maldonado Solorzano.

Carnet: 2012-226-87

1) Introducción

Delivereats es una plataforma de delivery basada en una arquitectura orientada a microservicios. En esta práctica no se implementan aún microservicios ni componentes backend/frontend, sino que se construye la documentación base y se implementa una base de datos funcional en un motor relacional (MySQL), la cual será consumida posteriormente durante el desarrollo completo del sistema.

2) Objetivos

2.1 Objetivo general

Aplicar de manera efectiva la metodología ágil **SCRUM** para la construcción de una aplicación mediante la entrega continua y progresiva de incrementos funcionales. A través de este enfoque iterativo, se busca planificar, diseñar, documentar y validar componentes del sistema en ciclos cortos, garantizando una evolución constante del proyecto y una adecuada adaptación a cambios.

2.2 Objetivos específicos

- Definir requerimientos funcionales y no funcionales del sistema.
- Diseñar arquitectura a nivel de microservicios, comunicación y responsabilidades.
- Diseñar el esquema de base de datos alineado con servicios Auth, Catálogo, Órdenes y Delivery.
- Proponer un diagrama de despliegue (mínimo entorno local).
- Entregar una base de datos funcional en MySQL.

3) Alcance

Incluye

- Documentación completa del sistema.
- Diseño arquitectónico propuesto para microservicios.
- Implementación de la base de datos funcional en MySQL con datos de prueba.

No incluye

- Desarrollo de backend.
- Desarrollo de frontend.
- Implementación real de microservicios.

Diseño:

Auth

- Usuarios
- Roles
- Relación muchos a muchos usuario_rol

Catálogo

- Restaurantes
- Items del menú por restaurante

Órdenes

- Ordenes con estados requeridos
- Detalle de orden por items y cantidades

Delivery

- Entregas con estados requeridos
- Un repartidor puede ser NULL si no está asignado

1) Requerimientos Funcionales (RF)

RF-01 Registro de usuario

El sistema debe permitir registrar usuarios con correo, contraseña, nombre y teléfono.

RF-02 Inicio de sesión

El sistema debe permitir autenticación mediante correo y contraseña.

RF-03 Gestión de roles

El sistema debe manejar roles: **ADMIN, CLIENTE, RESTAURANTE, REPARTIDOR.**

RF-04 Asignación de roles a usuarios

El sistema debe permitir asignar uno o más roles a un usuario.

RF-05 Gestión de restaurantes

El sistema debe permitir registrar restaurantes con nombre, descripción, dirección, teléfono y estado activo/inactivo.

RF-06 Gestión de menú

El sistema debe permitir registrar ítems de menú por restaurante, con nombre, descripción, precio y disponibilidad.

RF-07 Creación de orden

El cliente debe poder crear órdenes indicando restaurante, dirección de entrega y lista de ítems con cantidades.

RF-08 Gestión de estados de orden

El sistema debe permitir cambiar estados de orden:

- **CREADA**
- **EN_PROCESO**
- **FINALIZADA**
- **RECHAZADA**

RF-09 Registro de detalle de orden

Cada orden debe guardar los ítems comprados con cantidad, precio unitario y subtotal.

RF-10 Cálculo del total de la orden

El total debe representar la suma de subtotales del detalle.

RF-11 Gestión de entregas

El sistema debe permitir registrar entregas asociadas a una orden.

RF-12 Gestión de repartidor

El sistema debe permitir asignar un repartidor (usuario con rol REPARTIDOR) a una entrega.

RF-13 Gestión de estados de entrega

La entrega debe manejar estados:

- **EN_CAMINO**
- **ENTREGADO**
- **CANCELADO**

RF-14 Consultas operativas

El sistema debe permitir consultar:

- usuarios y roles
- menú por restaurante
- órdenes por estado
- entregas por estado

2) Requerimientos No Funcionales (RNF)

RNF-01 Seguridad

Las contraseñas deben almacenarse como **hash**, nunca en texto plano.

RNF-02 Integridad de datos

La base de datos debe garantizar consistencia mediante:

- PK/FK
- restricciones
- relaciones entre tablas

RNF-03 Disponibilidad

La base debe estar disponible para uso posterior por backend y frontend.

RNF-04 Escalabilidad

El diseño debe soportar crecimiento de restaurantes, usuarios, órdenes y entregas.

RNF-05 Mantenibilidad

El esquema debe estar normalizado y ser comprensible para facilitar cambios.

RNF-06 Rendimiento

Consultas frecuentes deben responder eficientemente (por ejemplo: órdenes por cliente, menú por restaurante).

RNF-07 Trazabilidad

Las tablas deben permitir auditoría básica con timestamps como fecha_creacion.

RNF-08 Compatibilidad

La base debe funcionar en motor relacional **MySQL 8+**.

3) Diagrama de Arquitectura de Alto Nivel (Microservicios)

Aunque aún no se implementen microservicios, este es el diseño propuesto.

Servicios propuestos

1) Auth Service

Responsabilidad:

- Registro/login
- roles y permisos
- gestión de usuarios

Tablas relacionadas:

- usuarios
- roles
- usuario_roles

2) Catálogo Service

Responsabilidad:

- restaurantes
- menú

Tablas relacionadas:

- restaurantes
- menu_items

3) Órdenes Service

Responsabilidad:

- creación y seguimiento de órdenes
- estados de orden
- detalle de órdenes

Tablas relacionadas:

- ordenes
- orden_detalle

4) Delivery Service

Responsabilidad:

- asignación de repartidor

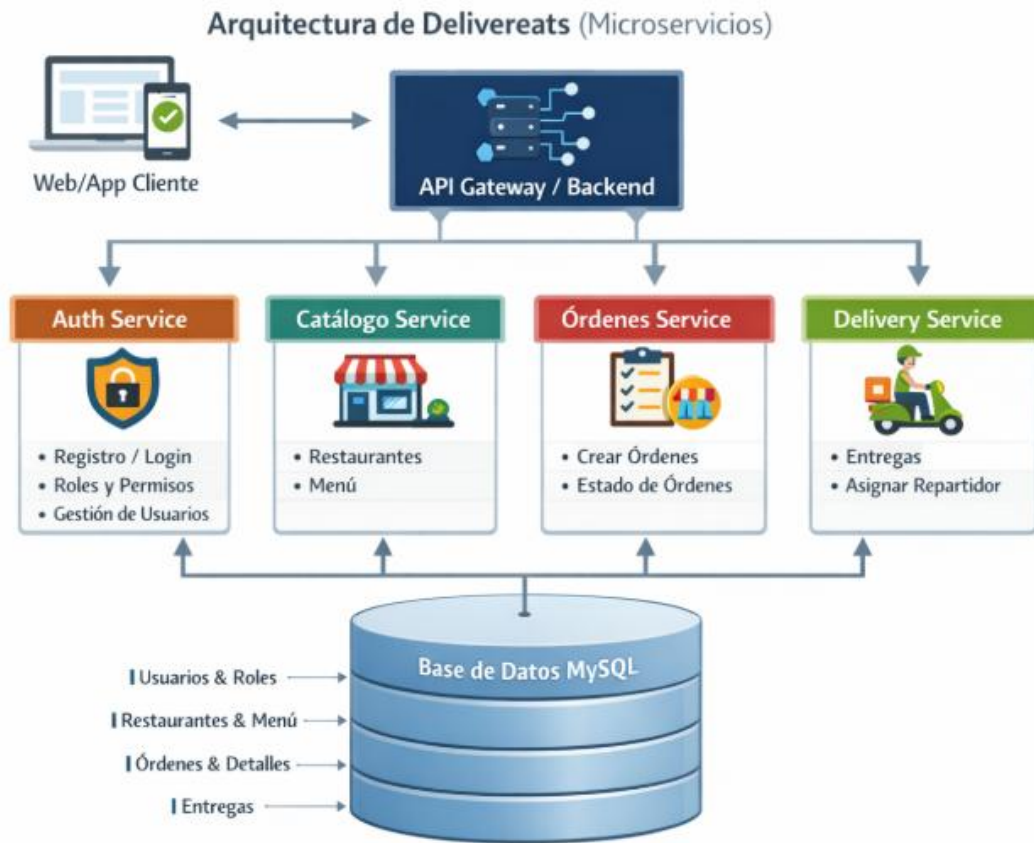
- control de entregas y estados

Tablas relacionadas:

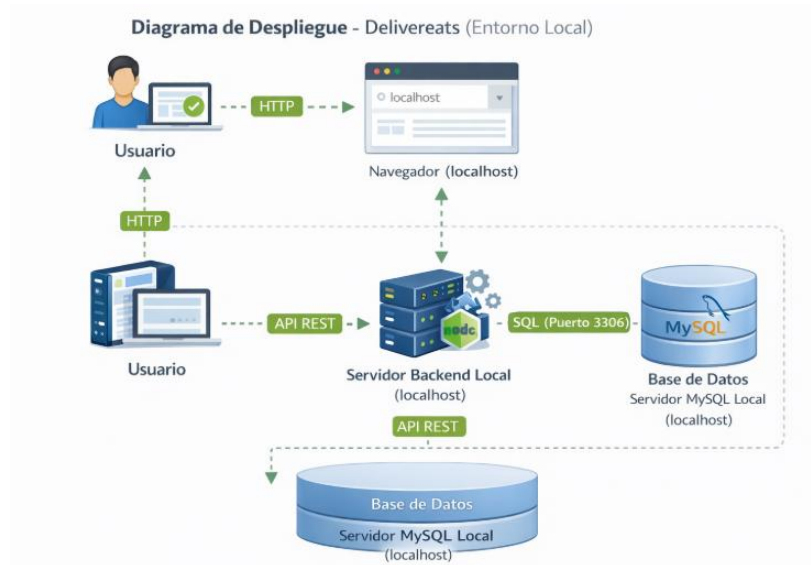
- entregas

Comunicación entre servicios (propuesta)

- Auth → valida tokens / roles
- Órdenes → consulta Catálogo (items y restaurante)
- Delivery → consulta Órdenes (id_orden, estado)
- Órdenes → notifica Delivery cuando orden pasa a EN_PROCESO



4) Diagrama de Despliegue (Deployment)



Opción mínima requerida: entorno local

Cliente Web / App

- Navegador (Front-End)

Backend

- API Gateway (o servidor principal)
- Auth Service
- Catálogo Service
- Órdenes Service
- Delivery Service

Base de Datos

- MySQL Server

Entorno local

- Todo puede correr en una PC usando Docker Compose o instalación local.

Proyección a nube (opcional recomendado)

- Frontend: hosting estático (S3 / Firebase Hosting / Netlify)
- Backend: contenedores (Docker + Kubernetes o Cloud Run)
- Base de datos: MySQL administrado (Cloud SQL / RDS)

5) Esquema de Base de Datos (Descripción)

```
-- =====
-- BASE DE DATOS: Delivereats
-- =====

DROP DATABASE IF EXISTS delivereats;
CREATE DATABASE delivereats CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE delivereats;

-- =====
-- TABLAS AUTH (Usuarios y Roles)
-- =====

CREATE TABLE roles (
  id_role INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE usuarios (
  id_usuario INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  correo VARCHAR(150) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  telefono VARCHAR(20),
  direccion VARCHAR(200),
  activo BOOLEAN NOT NULL DEFAULT TRUE,
  fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE usuario_roles (  
    id_usuario INT NOT NULL,  
    id_role INT NOT NULL,  
    PRIMARY KEY (id_usuario, id_role),  
    FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (id_role) REFERENCES roles(id_role)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
-- =====  
-- TABLAS CATALOGO (Restaurantes y Menú)  
-- =====
```

```
CREATE TABLE restaurantes (  
    id_restaurante INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(150) NOT NULL,  
    descripcion VARCHAR(255),  
    direccion VARCHAR(200) NOT NULL,  
    telefono VARCHAR(20),  
    activo BOOLEAN NOT NULL DEFAULT TRUE,  
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

```

CREATE TABLE menu_items (
    id_item INT AUTO_INCREMENT PRIMARY KEY,
    id_restaurante INT NOT NULL,
    nombre VARCHAR(150) NOT NULL,
    descripcion VARCHAR(255),
    precio DECIMAL(10,2) NOT NULL CHECK (precio >= 0),
    disponible BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (id_restaurante) REFERENCES restaurantes(id_restaurante)
    ON DELETE CASCADE ON UPDATE CASCADE
);

-- =====
-- TABLAS ORDENES (Órdenes y Detalles)
-- =====

CREATE TABLE ordenes (
    id_orden INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente INT NOT NULL,
    id_restaurante INT NOT NULL,
    estado ENUM('CREADA', 'EN_PROCESO', 'FINALIZADA', 'RECHAZADA') NOT NULL DEFAULT 'CREADA',
    total DECIMAL(10,2) NOT NULL DEFAULT 0 CHECK (total >= 0),
    direccion_entrega VARCHAR(200) NOT NULL,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_cliente) REFERENCES usuarios(id_usuario)
    ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (id_restaurante) REFERENCES restaurantes(id_restaurante)
    ON DELETE RESTRICT ON UPDATE CASCADE
);

```

```

CREATE TABLE orden_detalle (
    id_detalle INT AUTO_INCREMENT PRIMARY KEY,
    id_orden INT NOT NULL,
    id_item INT NOT NULL,
    cantidad INT NOT NULL CHECK (cantidad > 0),
    precio_unitario DECIMAL(10,2) NOT NULL CHECK (precio_unitario >= 0),
    subtotal DECIMAL(10,2) NOT NULL CHECK (subtotal >= 0),
    FOREIGN KEY (id_orden) REFERENCES ordenes(id_orden)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (id_item) REFERENCES menu_items(id_item)
        ON DELETE RESTRICT ON UPDATE CASCADE
);

-- =====
-- TABLAS DELIVERY (Entregas)
-- =====

CREATE TABLE entregas (
    id_entrega INT AUTO_INCREMENT PRIMARY KEY,
    id_orden INT NOT NULL UNIQUE,
    id_repartidor INT,
    estado ENUM('EN_CAMINO', 'ENTREGADO', 'CANCELADO') NOT NULL DEFAULT 'EN_CAMINO',
    fecha_asignacion TIMESTAMP NULL,
    fecha_entrega TIMESTAMP NULL,
    FOREIGN KEY (id_orden) REFERENCES ordenes(id_orden)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (id_repartidor) REFERENCES usuarios(id_usuario)
        ON DELETE SET NULL ON UPDATE CASCADE
);

```

```

-- =====
-- DATOS DE PRUEBA (Roles, Usuarios, Restaurantes, Menú, Órdenes, Entregas)
-- =====

-- Roles
INSERT INTO roles (nombre) VALUES
('ADMIN'),
('CLIENTE'),
('RESTAURANTE'),
('REPARTIDOR');

-- Usuarios
INSERT INTO usuarios (nombre, correo, password_hash, telefono, direccion) VALUES
('Admin Principal', 'admin@delivereats.com', 'hash_admin', '5555-0001', 'Zona 1, Ciudad'),
('Juan Cliente', 'cliente@delivereats.com', 'hash_cliente', '5555-0002', 'Zona 7, Ciudad'),
('Restaurante Owner', 'restaurante@delivereats.com', 'hash_rest', '5555-0003', 'Zona 10, Ciudad'),
('Pedro Repartidor', 'repartidor@delivereats.com', 'hash_repartidor', '5555-0004', 'Zona 5, Ciudad');

-- Asignar roles a usuarios
INSERT INTO usuario_rol (id_usuario, id_rol) VALUES
(1, 1), -- Admin -> ADMIN
(2, 2), -- Cliente -> CLIENTE
(3, 3), -- Owner -> RESTAURANTE
(4, 4); -- Repartidor -> REPARTIDOR

-- Restaurantes
INSERT INTO restaurantes (nombre, descripcion, direccion, telefono) VALUES
('Pizza Express', 'Pizzas artesanales y bebidas', 'Zona 10, Ciudad', '2222-1111'),
('Burger House', 'Hamburguesas premium', 'Zona 4, Ciudad', '2222-2222');

-- Menú
INSERT INTO menu_items (id_restaurante, nombre, descripcion, precio, disponible) VALUES
(1, 'Pizza Pepperoni', 'Pizza grande con pepperoni', 65.00, TRUE),
(1, 'Pizza Hawaiana', 'Pizza grande con piña y jamón', 60.00, TRUE),
(1, 'Coca Cola 600ml', 'Bebida gaseosa', 10.00, TRUE),
(2, 'Hamburguesa Clásica', 'Carne, queso y vegetales', 45.00, TRUE),
(2, 'Papas Fritas', 'Papas medianas', 20.00, TRUE),
(2, 'Refresco Natural', 'Bebida natural', 12.00, TRUE);

-- Órdenes
INSERT INTO ordenes (id_cliente, id_restaurante, estado, total, direccion_entrega) VALUES
(2, 1, 'CREADA', 75.00, 'Zona 7, Calle Principal'),
(2, 2, 'EN_PROCESO', 65.00, 'Zona 7, Avenida Central'),
(2, 1, 'FINALIZADA', 70.00, 'Zona 7, Colonia A'),
(2, 2, 'RECHAZADA', 45.00, 'Zona 7, Colonia B');

```

```

-- Detalles de órdenes
INSERT INTO orden_detalle (id_orden, id_item, cantidad, precio_unitario, subtotal) VALUES
(1, 1, 1, 65.00, 65.00),
(1, 3, 1, 10.00, 10.00),

(2, 4, 1, 45.00, 45.00),
(2, 5, 1, 20.00, 20.00),

(3, 2, 1, 60.00, 60.00),
(3, 3, 1, 10.00, 10.00),

(4, 4, 1, 45.00, 45.00);

-- Entregas
INSERT INTO entregas (id_orden, id_repartidor, estado, fecha_asignacion, fecha_entrega) VALUES
(2, 4, 'EN_CAMINO', NOW(), NULL),
(3, 4, 'ENTREGADO', NOW(), NOW()),
(4, NULL, 'CANCELADO', NULL, NULL);

-- =====
-- CONSULTAS DE VALIDACIÓN (Opcional)
-- =====

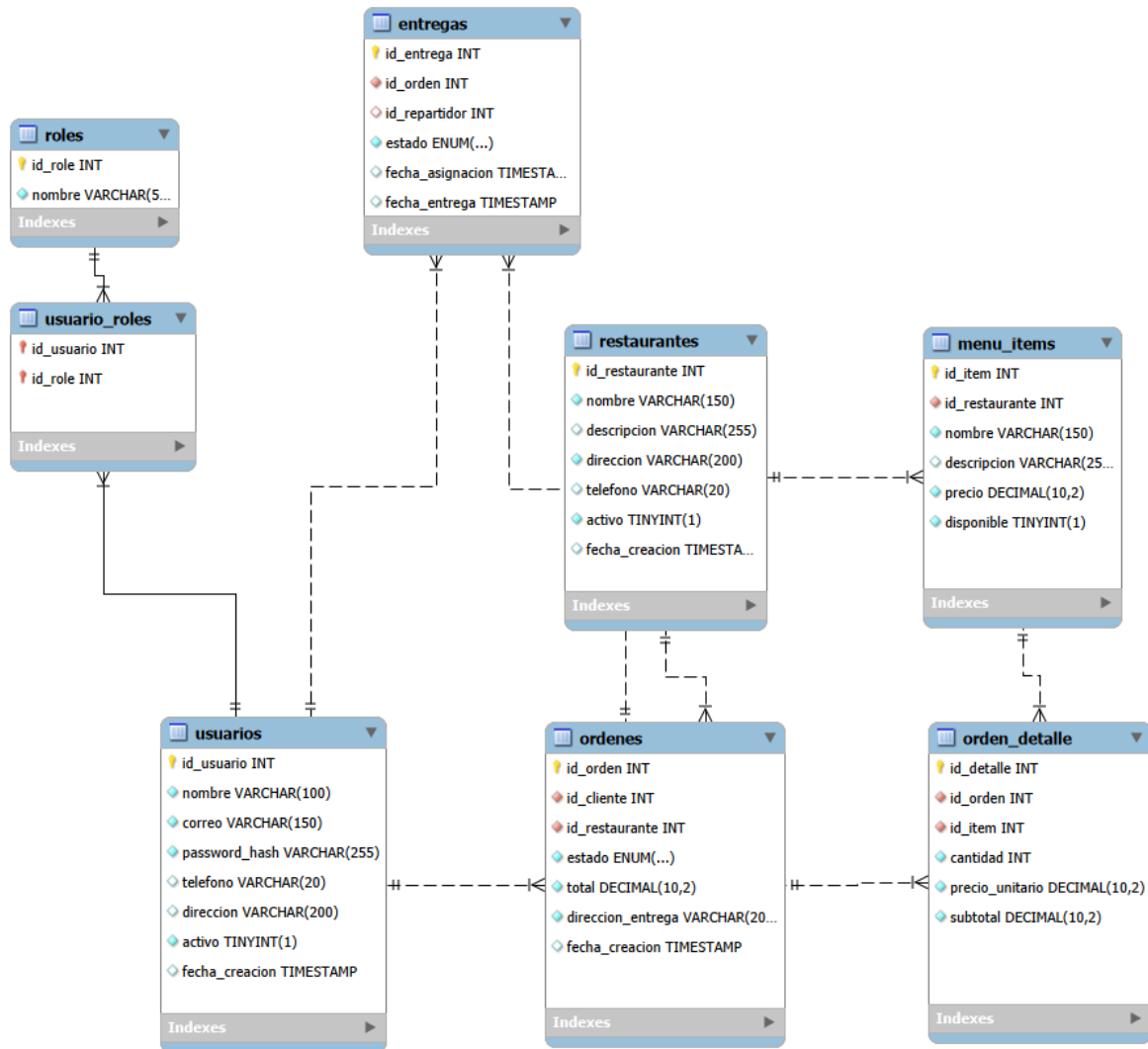
-- Ver usuarios con roles
SELECT u.id_usuario, u.nombre, r.nombre AS rol
FROM usuarios u
JOIN usuario_rols ur ON u.id_usuario = ur.id_usuario
JOIN roles r ON ur.id_role = r.id_role
ORDER BY u.id_usuario;

-- Ver menú completo
SELECT res.nombre AS restaurante, mi.nombre AS item, mi.precio, mi.disponible
FROM menu_items mi
JOIN restaurantes res ON mi.id_restaurante = res.id_restaurante
ORDER BY res.id_restaurante;

-- Ver órdenes con estado
SELECT o.id_orden, u.nombre AS cliente, r.nombre AS restaurante, o.estado, o.total
FROM ordenes o
JOIN usuarios u ON o.id_cliente = u.id_usuario
JOIN restaurantes r ON o.id_restaurante = r.id_restaurante
ORDER BY o.id_orden;

-- Ver entregas con estado
SELECT e.id_entrega, e.id_orden, e.estado, u.nombre AS repartidor
FROM entregas e
LEFT JOIN usuarios u ON e.id_repartidor = u.id_usuario
ORDER BY e.id_entrega;

```



Tablas y propósito

- roles: catálogo de roles del sistema
- usuarios: usuarios registrados
- usuario_roles: relación N:M usuarios ↔ roles
- restaurantes: restaurantes disponibles
- menu_items: items por restaurante
- ordenes: órdenes realizadas por clientes
- orden_detalle: items incluidos en la orden
- entregas: delivery asociado a una orden

Relaciones principales

- Un usuario puede tener varios roles (N:M)
- Un restaurante tiene muchos items (1:N)
- Un cliente puede tener muchas órdenes (1:N)
- Una orden tiene muchos detalles (1:N)

- Una orden tiene 0 o 1 entrega (1:1)
- Un repartidor puede tener muchas entregas (1:N)

6) Diagrama de Actividades (Flujo principal)

Actividad: Crear orden y entrega

1. Cliente inicia sesión
2. Cliente selecciona restaurante
3. Cliente selecciona ítems del menú y cantidades
4. Sistema crea orden en estado **CREADA**
5. Sistema registra detalles de la orden
6. Restaurante acepta la orden → cambia estado a **EN_PROCESO**
7. Sistema crea registro de entrega (estado **EN_CAMINO**)
8. Sistema asigna repartidor
9. Repartidor entrega → estado entrega **ENTREGADO**
10. Sistema marca orden como **FINALIZADA**

7) SCRUM + Product Backlog

Historias de usuario

HU-01 Registro

Como usuario
quiero registrarme
para usar la plataforma.

Criterios de aceptación

- El correo debe ser único.
- Se almacena hash de contraseña.
- Usuario queda activo.

HU-02 Ver restaurantes

Como cliente
quiero ver restaurantes activos
para elegir dónde pedir.

Criterios

- Solo restaurantes activos.
- Mostrar nombre, dirección y teléfono.

HU-03 Ver menú

Como cliente
quiero ver menú por restaurante
para seleccionar productos.

Criterios

- **Mostrar items disponibles.**
- **Mostrar precio.**

HU-04 Crear orden

Como cliente
quiero crear una orden
para solicitar delivery.

Criterios

- **Estado inicial: CREADA**
- **Orden debe tener al menos 1 item**

HU-05 Cambiar estado de orden

Como restaurante
quiero cambiar estado de la orden
para procesarla.

Criterios

- **CREADA → EN_PROCESO**
- **EN_PROCESO → FINALIZADA**
- **CREADA → RECHAZADA**

HU-06 Crear entrega

Como sistema
quiero crear una entrega
para asignar repartidor.

Criterios

- **Solo para órdenes EN_PROCESO**
- **Estado inicial: EN_CAMINO**

HU-07 Finalizar entrega

Como repartidor
quiero finalizar entrega
para completar el pedido.

Criterios

- EN_CAMINO → ENTREGADO
- Si se cancela → CANCELADO

11) Product Backlog (Tabla)

| ID | Historia | Prioridad | Estimación |
|-------|--------------------|-----------|------------|
| HU-01 | Registro/Login | Alta | 3 pts |
| HU-02 | Restaurantes | Alta | 2 pts |
| HU-03 | Menú | Alta | 3 pts |
| HU-04 | Crear orden | Alta | 5 pts |
| HU-05 | Estados de orden | Media | 3 pts |
| HU-06 | Crear entrega | Alta | 3 pts |
| HU-07 | Estados de entrega | Media | 2 pts |

8) Planificación de Sprints

Sprint 1 (Documentación + BD)

- Requerimientos
- Arquitectura
- Despliegue
- Modelo BD
- Script MySQL + datos de prueba

Sprint 2 (Backend base)

- Auth Service
- CRUD Catálogo
-

Sprint 3 (Órdenes)

- Crear orden
- Cambios de estado
- Detalle

Sprint 4 (Delivery)

- **Asignación repartidor**
- **Estados de entrega**
- **Seguimiento**

9) Conclusiones

- **Se definieron requerimientos funcionales y no funcionales del sistema Delivereats.**
- **Se propuso arquitectura de microservicios para facilitar escalabilidad y mantenimiento.**
- **Se diseñó e implementó una base de datos funcional en MySQL con restricciones, relaciones y datos de prueba.**
- **Se aplicó SCRUM como marco de trabajo, definiendo backlog y planificación incremental.**