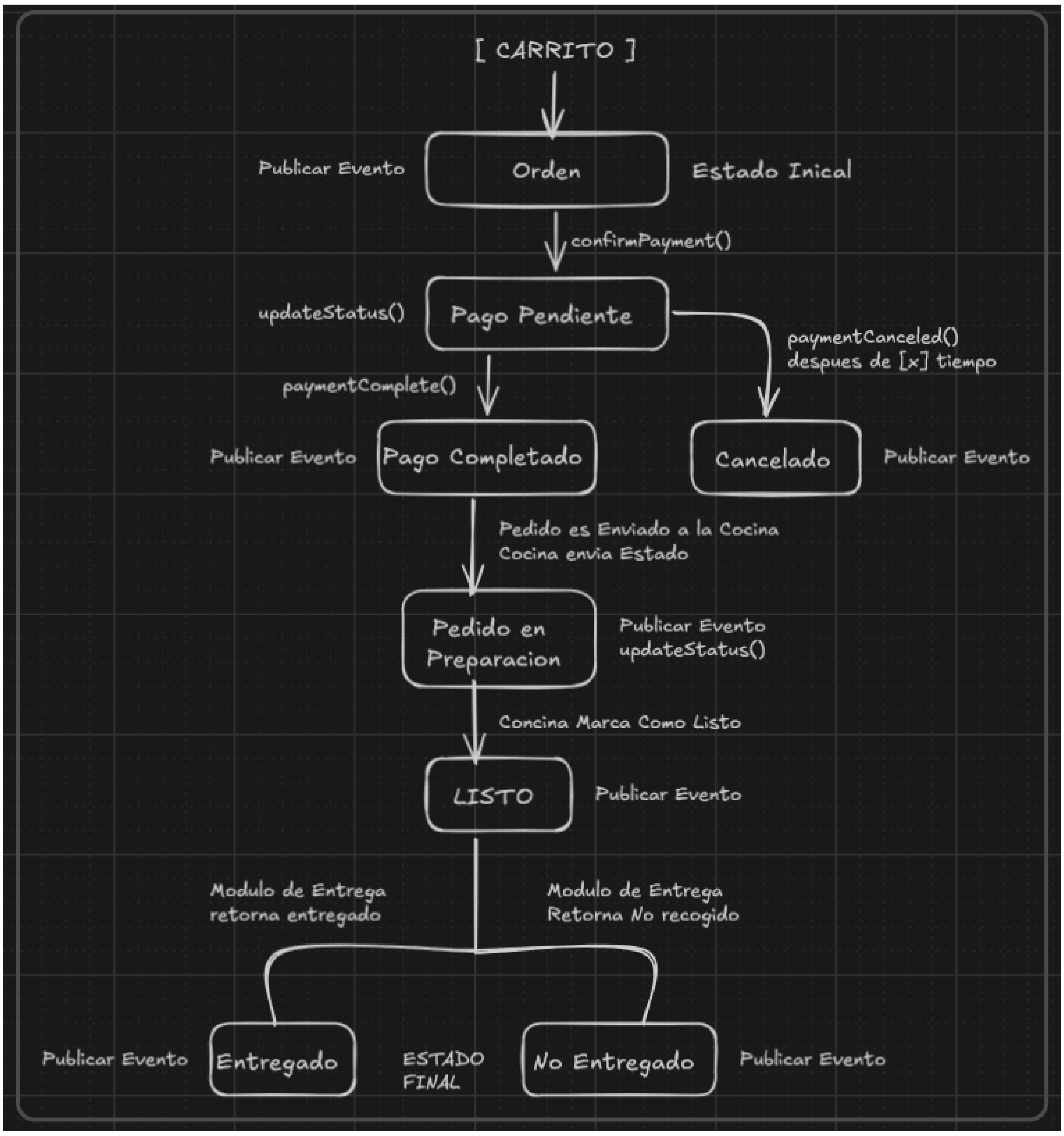
Actividad 03

pseudo-codigo

```
public class Order {
    UUID id;
    String number;
    List<OrderItem> items;
    BigDecimal subtotal;
    BigDecimal tax;
    BigDecimal total;
    OrderStatus status;
    OrderOrigin origin;
        LocalDateTime createdAt;
    String paymentRef;
}

OrderStatus {
    PENDING, PAID, IN_PREPARATION, READY, DELIVERED, CANCELLED
};
```

```
public class OrderService {

      private OrderRepository repository;
    private EventPublisher publisher;

    public Order createOrder(List<OrderItem> cartItems, OrderOrigin origi
n) {
        validateAvailability(cartItems);

        Order order = new Order();
        order.id = UUID.randomUUID();
        order.items = cartItems;
        order.origin = origin;
        order.number = generateOrderNumber();
                order.status = OrderStatus.PENDING;
        order.createdAt = LocalDateTime.now();

        repository.save(order);
        return order;
    }

    public void confirmPayment(UUID orderId, String paymentRef) {
        Order order = findOrder(orderId);
        assertStatus(order, OrderStatus.PENDING);
        order.status = OrderStatus.PAID;
        order.paymentRef = paymentRef;

        repository.save(order);
        publisher.publish("order.id", order.status);
    }

    public void cancelOrder(UUID orderId) {
        Order order = repository.findOrder(orderId);

        if (order.status != OrderStatus.PENDING) {
            "Only cancellable before preparation";
        }

        order.status = OrderStatus.CANCELLED;
        repository.save(order);
        publisher.publish(order.id, order.status);
    }

    public void startPreparation(UUID orderId) {
        Order order = repository.findOrder(orderId);
        assertStatus(order, OrderStatus.PAID);

        order.status = OrderStatus.IN_PREPARATION;
        repository.save(order);
        publisher.publish(order.id, order.status);
    }

    public void markReady(UUID orderId) {
        Order order = repository.findOrder(orderId);
        assertStatus(order, OrderStatus.IN_PREPARATION);

        order.status = OrderStatus.READY;
        repository.save(order);
        publisher.publish(order.id, order.status);
    }

    public void markDelivered(UUID orderId) {
        Order order = repository.findOrder(orderId);
        assertStatus(order, OrderStatus.READY);

        order.status = OrderStatus.DELIVERED;
        repository.save(order);
        publisher.publish(order.id, order.status);
    }


    public void assertStatus(Order order, OrderStatus expected) {
        if (order.status != expected) {
            throw new InvalidTransitionException(order.status, expected);
        }
    }

    public String generateOrderNumber() {
        return prefix + dailyCounter.incrementAndGet();
    }
}
```


[CARRITO]
Publicar Evento
Orden
Estado Inical
confirmPayment()
updateStatus()
Pago Pendiente
paymentCanceled()
despues de [x] tiempo
paymentComplete()
Publicar Evento
Pago Completado
Cancelado
Publicar Evento
Pedido es Enviado a la Cocina
Cocina envia Estado
Pedido en Preparacion
Publicar Evento
updateStatus()
Concina Marca Como Listo
LISTO
Publicar Evento
Modulo de Entrega
retorna entregado
Modulo de Entrega
Retorna No recogido
Publicar Evento
Entregado
ESTADO FINAL
No Entregado
Publicar Evento