En base al modelo y persistencia de los T.P. 2 y T.P. 2-B, consideremos el modelo final la combinación del T.P. 2 y T.P. 3, es decir que al producto lo clasificamos como en el T.P. 2-B, de modo que nos quedan las entidades persistentes:
Cliente, Cuenta, Proveedor, Precio y Producto (General, Alimento, Frio, Congelado y Gondola).
El proyecto Spring Boot creado en clase ya cuenta con la persistencia de las entidades: Cliente y Cuenta utilizando REST y ClienteDTO, ahora bien se pide lo siguiente:

1 - Al proyecto realizado en clase agregar todo el modelo (con sus respectivas annotations de persistencia).

2 - Realizar altas de la entidad Proveedor a través de Test JUnit invocando a un ProveedorService, para esto no es necesario hacer un DTO para el Proveedor, como tampoco un Controller, solamente crearemos los proveedores desde el Test, invocando al Service correspondiente, y este al Repository correspondiente.

3 - Realizar altas de la entidad Producto, los mismos siempre se dan de alta con un Precio y un Proveedor (existente, creado en el punto 2), a pesar del ManyToMany entre Producto y Proveedor agregaremos un solo Proveedor a la Lista para simplificar; observar que tampoco hacen falta DTO ni Controller ya que los productos se darán de alta a través del Test (invocando a Service).

4 - Bueno, aquí haremos el circuito completo al igual que en Cliente, mediante una petición REST (POST) daremos de alta una factura, para ello necesitaremos un Controller que reciba la petición y aquí si utilizaremos los DTO, tendremos que definir las clases de los mismos (FacturaDTO y DetalleDTO, es una opción, resuelvan como les parezca) y luego Spring instanciará estos DTO con el JSON enviado, como ya vimos en el caso del Cliente, luego persistiremos la entidad Factura de forma transaccional, ¿dónde armamos la misma en base a los DTO? ¿Controller, Service o Repository?.
Una estructura de DTO podría ser:

```
class DetalleDTO
        private Long idProducto;

class FacturaDTO
        private Long id;
        private Long idCliente;
        private LocalDateTime fecha;
        private String numero;
        private List<DetalleDTO> detallesDTO;
        private Integer cantidad;
```

Observar que desde la UI ya me vienen los id de Cliente y los id de los productos comprados como también sus respectivas cantidades.

Nota: Conviene que los Test armen la estructura de la base de datos, y el deploy del web utilice la existente sin necesidad de regenerarla.