Caso de Estudio "Delivery & Takeout"

¡Excelente elección! El modelo de Delivery & Takeout es perfecto para este propósito. Vamos a abordar esto de una manera práctica y ágil, tal como se haría en un equipo de desarrollo real. Olvídate de documentos kilométricos; nos enfocaremos en lo esencial para empezar a programar.

Como experto, te guiaré a través de los pasos iniciales, poniendo énfasis en definir lo que realmente importa para tu proyecto Spring Boot.

El inicio de la implementación en un entorno ágil no comienza con un diseño monolítico, sino con la identificación del flujo de valor principal y las reglas de negocio mínimas viables.

🚀 Inicio Ágil de Implementación: Delivery & Takeout

El primer paso es entender el core del negocio y traducirlo en funcionalidades concretas.

Fase 1: Definición del "Happy Path" (Flujo Principal Feliz)

Nos centraremos en la acción más importante: que un cliente pida una pizza y la reciba. Todo lo demás (gestión de inventario detallada, promociones complejas) vendrá después.

1. Identificar el Flujo de Valor Principal

El flujo de valor principal es simple:

Cliente Navega → Cliente Selecciona Pizzas → Cliente Paga → Restaurante Recibe Pedido → Restaurante Prepara y Entrega → Cliente Recibe.

2. Definición de Entidades Clave (Modelos de Datos - Entities)

Antes de los casos de uso, necesitamos saber de qué cosas vamos a hablar. Estas serán tus clases en Spring Boot:

- **Cliente (Customer):** Datos básicos (Nombre, Email, Dirección, Teléfono).
- **Producto (Product/Pizza):** Lo que vendemos (Nombre, Descripción, Precio base, Tamaño).
- ❌ **Ingrediente (Ingredient):** Materia prima (Nombre, Stock, Costo unitario). *Inicialmente opcional.*
- **Pedido (Order):** El contenedor principal de la transacción (ID, Fecha/Hora, Estado, Dirección de Entrega, Total).
- **ItemPedido (OrderItem):** La línea de detalle que conecta un Pedido con un Producto y cantidades.

3. Definición de Reglas de Negocio Mínimas Viables (MVBR)

Aquí es donde está la "jugosidad" que buscas, más allá del CRUD básico. Estas reglas dictan cómo se comportan tus servicios en Spring Boot:

- **RN1:** El precio del pedido es dinámico. El sistema debe calcular el total del Pedido sumando los precios de todos sus ItemPedido en el momento de finalizar la compra. (No se almacena el total en la base de datos hasta confirmarse, o se recalcula siempre).

- **RN2:** El estado del pedido sigue una secuencia estricta. Un pedido solo puede pasar de PENDIENTE a EN_PREPARACION, luego a LISTO_PARA_ENTREGA, y finalmente a ENTREGADO. No puede pasar de ENTREGADO a EN_PREPARACION.
- **RN3:** Validación de datos de entrega. Un Pedido solo puede ser confirmado si el Cliente ha proporcionado una dirección válida y un número de teléfono.
- **RN4** (Opcional pero recomendable): Validación de inventario. Antes de aceptar un Pedido, el sistema debe verificar que hay suficiente stock de ingredientes clave (ej. masa base, queso, pepperoni). Si no hay stock, el pedido se rechaza o el artículo se marca como no disponible.

Fase 2: Definición de Casos de Uso y Epopeyas (User Stories)

Ahora, traducimos ese flujo y esas reglas a tareas concretas para tu proyecto. En un enfoque ágil (Scrum/Kanban), a esto le llamamos Historias de Usuario.

Aquí tienes las 3 "Epopeyas" (grandes bloques de trabajo) para empezar:

Epopeya 1: Gestión del Menú y Productos

- **HU1:** Como administrador, quiero poder añadir un nuevo tipo de pizza (ej. "Hawaiana", precio $15) al menú para que los clientes puedan verla.
- **HU2:** Como administrador, quiero poder actualizar el precio o marcar una pizza como "Agotada" (ej. si nos quedamos sin piña).

Epopeya 2: Realizar un Pedido (El Core)

- **HU3:** Como cliente, quiero ver la lista de pizzas disponibles y sus precios.
- **HU4:** Como cliente, quiero añadir múltiples pizzas de diferentes tipos y cantidades a mi "carrito de compras".
- **HU5:** Como cliente, quiero proporcionar mi dirección y datos de contacto al finalizar la compra.
- **HU6:** Como sistema, quiero calcular el precio total final del carrito aplicando la RN1.
- **HU7:** Como cliente, quiero confirmar mi pedido y recibir un número de seguimiento.
- **HU8:** Como sistema, quiero validar los datos del cliente aplicando la RN3.

Epopeya 3: Gestión Interna del Pedido (Backoffice)

- **HU9:** Como empleado de cocina, quiero ver una lista de nuevos pedidos PENDIENTES.
- **HU10:** Como empleado, quiero cambiar el estado de un pedido a EN_PREPARACION y luego a LISTO_PARA_ENTREGA aplicando la RN2.
- **HU11:** Como repartidor, quiero ver los pedidos LISTOS_PARA_ENTREGA y marcar como ENTREGADO una vez completados.

Fase 3: Tu Enfoque en Spring Boot

Para tu proyecto, te sugiero empezar por la **Epopeya 2 (Realizar un Pedido)**, ya que es la que más lógica de negocio real tiene.

1. **Define tus Entidades:** Crea tus clases `@Entity` (Cliente, Pedido, ItemPedido, Producto).
2. **Crea tus Repositorios:** Interfaces `JpaRepository` para acceder a los datos.

3. **Implementa un Servicio (OrderService):** Aquí es donde pones las Reglas de Negocio (RN1 y RN3). La magia ocurre aquí, no en el controlador.
4. **Crea tus Controladores (OrderController, ProductController):** Endpoints REST (`/api/pedidos`, `/api/productos`).

¡Este es el camino ágil! Empieza con el flujo principal, define tus reglas y usa las historias de usuario para organizar tus tareas en Spring Boot.

🚀 1. Core del Negocio (Alcance Mínimo Viable)\

El objetivo de este proyecto académico es implementar la lógica de negocio de un sistema de Delivery & Takeout (Entrega y Recogida), manteniendo la complejidad al mínimo necesario para ir más allá de un CRUD simple.

Aspecto	Definición Explícita del Core de Negocio	Impacto en el Diseño
Modelo	Pizzería de Un Solo Local.	No se implementa lógica de múltiples tiendas, geolocalización avanzada, o balanceo de carga.
Menú	Pizzas Predefinidas de un Único Tamaño Estándar.	No se permite la personalización (añadir/quitar ingredientes). No se necesita la tabla Ingrediente.
Inventario	Inventario Básico (Solo Disponibilidad).	No hay disminución de stock por pedido. Solo se verifica un flag disponible en el producto (RN4).
Pago	Pago Físico y Simulado (Efectivo o Tarjeta POS al recibir).	No se requiere integración real con pasarelas de pago (Stripe, etc.). El pago se asume "validado" para efectos de la lógica.
Entregas	Se gestionan pedidos con Entrega a Domicilio o Recogida en Tienda (será un campo simple en Pedido).	El cálculo de tiempos de entrega es una lógica simple (no georreferenciada).

2. Reglas de Negocio Clave (RN)

Estas cuatro reglas son la base de la lógica que implementarás en tu capa de `Service` (ej. `OrderService`) en Spring Boot, donde ocurre la "magia" del negocio.

ID	Regla de Negocio (RN)	Descripción Funcional	Clases Afectadas
RN1	**Cálculo de Total Dinámico**	El precio total del `Pedido` es la suma de los subtotales de sus `ItemPedido`: ∑(Producto.precioBase × ItemPedido.cantidad)	Pedido, ItemPedido, Producto

ID	Regla de Negocio (RN)	Descripción Funcional	Clases Afectadas
RN2	**Secuencia de Estados**	El `Pedido` debe pasar por una secuencia estricta e irreversible de estados (PENDIENTE → EN_PREPARACION → LISTO_PARA_ENTREGA → ENTREGADO / CANCELADO).	Pedido (Campo estado usando un Enum)
RN3	**Validación de Entrega**	Un `Pedido` solo puede ser confirmado (pasar de carrito a PENDIENTE) si el `Cliente` ha proporcionado datos de contacto y dirección válidos (no nulos/vacíos).	Cliente, Pedido
RN4	**Validación de Disponibilidad**	Solo se pueden añadir al `Pedido` los `Productos` que estén marcados como `disponible = true`.	Producto, ItemPedido

3. Flujo de Operaciones (Diagrama de Secuencia Implícito)

Este diagrama describe el proceso clave para la gestión de un pedido, desde su creación hasta su finalización.

Proceso	Actividad (Tarea Lógica)	Regla Aplicada	Tablas Involucradas
Creación del Pedido	**1. Seleccionar Productos:** El cliente añade Productos disponibles (RN4) al carrito.	RN4	Producto, ItemPedido
	2. Confirmar Datos: El cliente proporciona o confirma su Cliente (dirección/teléfono).	RN3	Cliente
	3. Cerrar Pedido: El sistema verifica la RN3, calcula el RN1 y crea el Pedido con estado PENDIENTE.	RN1, RN3	Pedido, ItemPedido
Gestión Interna	**4. Procesar:** Empleado cambia el estado de PENDIENTE a EN_PREPARACION.	RN2	Pedido
	5. Finalizar Preparación: Empleado cambia el estado de EN_PREPARACION a LISTO_PARA_ENTREGA.	RN2	Pedido
Finalización	**6. Entregar:** Repartidor cambia el estado de LISTO_PARA_ENTREGA a ENTREGADO (o CANCELADO).	RN2	Pedido

4. Historias de Usuario (Ejemplos)

Estas historias de usuario guían la implementación de las funcionalidades clave desde la perspectiva del administrador del sistema.

Epopeya 1: Gestión del Menú (Admin - CRUD Básico con RN4)

ID	Historia de Usuario	Foco en Spring Boot

ID	Historia de Usuario	Foco en Spring Boot
HU1	Como administrador, quiero añadir, modificar y eliminar Productos (pizzas predefinidas) del menú, incluyendo su nombre, precioBase y descripción.	ProductController, ProductService, ProductRepository
HU2	Como administrador, quiero poder marcar un Producto como disponible o agotado (boolean), para que la RN4 se aplique automáticamente.	Implementación de RN4 y Endpoint de actualización.

Epopeya 2: Realizar Pedido (Cliente - Core de la Lógica)

ID	Historia de Usuario	Foco en Spring Boot
HU3	Como cliente, quiero consultar la lista de Productos que están disponibles = true y ver su precioBase.	ProductController (GET mapping con filtro).
HU4	Como cliente, quiero crear un nuevo Pedido proporcionando mi nombre, teléfono y dirección (o usarlos si ya existo).	CustomerController, OrderService (Creación de Cliente).
HU5	Como cliente, quiero añadir múltiples ItemPedidos con su cantidad a un Pedido en estado PENDIENTE.	OrderController (Endpoint para añadir ítems).
HU6	Como sistema, debo validar que todos los datos de Cliente sean correctos (no vacíos) antes de la confirmación (RN3).	Implementación de RN3 en OrderService.
HU7	Como sistema, al confirmar el pedido, debo calcular y almacenar el total final en el Pedido aplicando la RN1.	Implementación de RN1 en OrderService (Método confirmarPedido).

Epopeya 3: Gestión Interna (Backoffice - Flujo de Estados)

ID	Historia de Usuario	Foco en Spring Boot
HU8	Como empleado, quiero ver una lista de todos los Pedidos que están actualmente en estado PENDIENTE.	OrderController (GET mapping con filtro por estado).
HU9	Como empleado, quiero actualizar el estado de un Pedido de forma secuencial y estricta (RN2), sin poder saltarme pasos.	Implementación del Enum de estados y RN2 en OrderService.
HU10	Como repartidor, quiero marcar un Pedido como ENTREGADO una vez que el ciclo haya finalizado.	OrderController (Método entregarPedido con validación RN2).

Este plan profesional te permite empezar creando las entidades (@Entity), luego los repositorios (JpaRepository), y finalmente el OrderService donde irás implementando las RN1, RN2, RN3, y RN4 con las historias de usuario como guía.