

# Factura de Venta

Datos del Cliente

Cédula del Cliente

Buscar

Nombre del Cliente

Datos del Producto

Código del Producto

Buscar

Nombre del Producto

Agregar

\$/ 0,000.00

Factura No:

Nro	CODIGO	DESCRIPCION	PRECIO	CANTIDAD	SUBTOTAL
<div>Generar Factura</div> <div>Cancelar Registro</div>					
Subtotal:					<div>\$/ 0,000.00</div>
Valor Iva:					<div>\$/ 0,000.00</div>
Total con Iva:					<div>\$/ 0,000.00</div>

Buscar Cliente: Microservicio de Clientes

Buscar Producto: Microservicio de Productos

Generar Factura: Microservicio de Ventas

Backend: spring web - devTools - MongoDB - swagger

- Agregamos a la clase principal el swagger con el método adicional
- Modelo con pojo tipo MongoDB @Document(collection = "") y @Id ajustar el nombre del atributo sin guión de piso
- DAO (Repositorio)
- Servicio (Interface - Clase de Implementación)
- Controlador (Se define los endpoint a utilizar get-post-put-delete)
- Excepcion

MongoDB con Atlas Cloud AWS

Métodos de consulta personalizados.

Podemos agregar algunos métodos a nuestro Repositorio para tener alguna funcionalidad adicional basada en nuestros requisitos funcionales, en la siguiente tabla se pueden relacionar los nombres de los métodos que podemos agregar al repositorio.

<https://docs.spring.io/spring-data/mongodb/docs/current/reference/html/#mongo.repositories>

Table 15. Supported keywords for query methods

Keyword	Sample	Logical result
After	<code>findByBirthdateAfter(Date date)</code>	<code>{"birthdate" : {"\$gt" : date}}</code>
GreaterThan	<code>findByAgeGreaterThan(int age)</code>	<code>{"age" : {"\$gt" : age}}</code>
GreaterThanEqual 1	<code>findByAgeGreaterThanEqual(int age)</code>	<code>{"age" : {"\$gte" : age}}</code>
Before	<code>findByBirthdateBefore(Date date)</code>	<code>{"birthdate" : {"\$lt" : date}}</code>
LessThan	<code>findByAgeLessThan(int age)</code>	<code>{"age" : {"\$lt" : age}}</code>
LessThanEqual	<code>findByAgeLessThanEqual(int age)</code>	<code>{"age" : {"\$lte" : age}}</code>
Between	<code>findByAgeBetween(int from, int to)</code> <code>findByAgeBetween(Range&lt;Integer&gt; range)</code>	<code>{"age" : {"\$gt" : from, "\$lt" : to}}</code> lower / upper bounds ( <code>\$gt</code> / <code>\$gte</code> & <code>\$lt</code> / <code>\$lte</code> ) according to <code>Range</code>
In	<code>findByAgeIn(Collection ages)</code>	<code>{"age" : {"\$in" : [ages...]}}</code>

NotIn	findByAgeNotIn(Collection ages)	{"age" : {"\$nin" : [ages_]}}
IsNotNull , NotNull	findByFirstnameNotNull()	{"firstname" : {"\$ne" : null}}
IsNull , Null	findByFirstnameNull()	{"firstname" : null}
Like , StartingWith , EndingWith	findByFirstnameLike(String name)	{"firstname" : name} (name as regex)
NotLike , IsNotLike	findByFirstnameNotLike(String name)	{"firstname" : { "\$not" : name }} (name as regex)
Containing ON String	findByFirstnameContaining(String name)	{"firstname" : name} (name as regex)
NotContaining on String	findByFirstnameNotContaining(String name)	{"firstname" : { "\$not" : name }} (name as regex)
Containing ON Collection	findByAddressesContaining(Address address)	{"addresses" : { "\$in" : address }}

NotContaining on Collection	findByAddressesNotContaining(Address address)	{"addresses" : { "\$not" : { "\$in" : address}}}
Regex	findByFirstnameRegex(String firstname)	{"firstname" : {"\$regex" : firstname }}
(No keyword)	findByFirstname(String name)	{"firstname" : name}
Not	findByFirstnameNot(String name)	{"firstname" : {"\$ne" : name}}
Near	findByLocationNear(Point point)	{"location" : {"\$near" : [x,y]}}
Near	findByLocationNear(Point point, Distance max)	{"location" : {"\$near" : [x,y], "\$maxDistance" : max}}
Near	findByLocationNear(Point point, Distance min, Distance max)	{"location" : {"\$near" : [x,y], "\$minDistance" : min, "\$maxDistance" : max}}
Within	findByLocationWithin(Circle circle)	{"location" : {"\$geoWithin" : {"\$center" : [ [x, y], distance]}}}
Within	findByLocationWithin(Box box)	{"location" : {"\$geoWithin" : {"\$box" : [ [x1, y1], x2, y2]}}}
IsTrue , True	findByActiveIsTrue()	{"active" : true}
IsFalse , False	findByActiveIsFalse()	{"active" : false}
Exists	findByLocationExists(boolean exists)	{"location" : {"\$exists" : exists }}



Backend de Ventas

Modelo de Datos

Ventas: Encabezado - Pie (Cuerpo Arreglo detalleVenta)

DetalleVenta: Clase de Java normal (pojo)

Ventas Collection de MongoDB

# Factura de Venta

Seleccione la Ciudad de la sucursal de Ventas:

Bogotá

Datos del Cliente

Cédula del Cliente

Buscar

Nombre del Cliente

Datos del Producto

Código Producto

Buscar

Nombre del Producto

+

Agregar Producto

1

\$ 0,000.00

BuscarCliente: Recibe la cédula del cliente llama al endpoint: 8082  
/api/clientes/buscar/{cedula}  
Con el objeto cliente imprimimos el nombre

Buscar: 8083 Producto: Recibe el código del producto y llama al endpoint /api/productos/buscar/{codigo}  
Con el objeto Producto escribimos el nombre y el precio

Escribimos cantidad y presionamos Agregar Producto y se agrega al arreglo del detalleventa

Factura No:

Item	CODIGO	DESCRIPCION	PRECIO VENTA	CANTIDAD	TOTAL
			\$	\$	\$

Generar Factura

Cancelar Factura

Subtotal:

\$ 0,000.00

Total Iva:

\$ 0,000.00

Total con Iva:

\$ 0,000.00

El arreglo de detalle venta lleva un consecutivo, código del producto, nombre del producto, precio venta, cantidad y total

Guardar en la colección de ventas solo se guarda: cantidad, codigo, valor total, valor venta y valor iva