# **BOCETO PROYECTO FINAL 1ºDAM**

El tema del proyecto será una librería, encargada de vender libros. Estará más enfocada al sector del manga.

***1 CLASES POJO***

**Libro**

int isbn

String titulo

String serie

String tipo {Manga, Cómic, Libro europeo, Manhwa} **(VER POSIBILIDAD DE PONERLA COMO ENUM)**

String genero {Aventuras, Ciencia-Ficción, Romance, Misterio, Terror, Comedia, Acción, Policíaco, Fantasia} **(VER POSIBILIDAD DE PONERLA COMO ENUM)**

double precio

int existencias

String autor

String editorial

String descripcion

LocalDate fechaPublicacion

int numPaginas

String cubierta {Tapa blanda, tapa dura, Sobrecubierta} **(VER POSIBILIDAD DE PONERLA COMO ENUM)**

String publico {Infantil, Juvenil, Adulto} **(VER POSIBILIDAD DE PONERLA COMO ENUM)**

int puntosPopularidad **//ESTO VA SER UN MÉTODO: calcularPuntosPopularidad(){}**

¿boolean favorito?

**Reservas (Preguntar a Luismi por las asociaciones)**

**Almacen**

**LineaVenta**

Libro libro

int cantidad

**Venta**

ArrayList de LineaVenta

double gastosEnvio

LocalDate fechaVenta

double precioFinal

**¿Favoritos?**

**Usuario**

int id

String nombre

String apellidos

String correo

String nombreUsuario

String contrasenha

String direccion

int numPuntos

**Admin**

***2 ESTRUCTURA DEL SITIO WEB***

Todas las páginas tendrán el mismo *header* y el mismo *footer*.

* El *header* contendrá:
  + El logo del sitio web.
  + Una navbar.
  + El botón de inicio de sesión.
  + Un botón que te llevará.
  + Un menú desplegable.
* El *footer* contendrá:
  + Los iconos de redes sociales.
  + La política de copyright.
  + ¿Quiénes somos?
  + Política de privacidad.

Páginas del sitio web:

**PagInicio** → Página de inicio del sitio web. Tendrá:

* Un *banner* con fotos de los productos para promocionarlos.
* Debajo habrá un menú con 3 opciones: Novedades, Próximamente y Merchandising.
* Un pequeño filtro para mostrar los productos que desee el cliente.
* Los productos estarán expuestos mediante tarjetas (que contendrán una imagen del producto, el título del producto, un botón de “marcar como favorito”, el precio y el botón de “añadir a la cesta”).

**PagUsuario** → Página con la información del usuario.

**PagInfoProd** →

**PagSerie** →

**PagFavoritos** →

**PagCesta** →

***APUNTES PARA EL UML***

Roadmap de Spring Boot: <https://roadmap.sh/spring-boot>

**Tutorial UsingThymeleaf (Puntos de interés):**

6 (bucles) - 7 (if) - 8 - 18 (apéndice A) - 19 (apéndice B).

VÍDEO TUTORIAL DE THYMELEAF (MUY BUENO)

<https://www.youtube.com/watch?v=GRsBkK48NsI>

**Producto**

* id
* nombre: String
* precio: double
* descuento: double
* especial: boolean
* categoria: Categoria

[Agregación]

**Categoria**

* id
* nombre: String

SELECT \*

LIMIT(3)

**3 productos más baratos.**

public List<Producto> obtenerTresProductosMasBaratos(double precio){

return repositorio.findTop3PrecioOrderByPrecioDesc(precio);

}

**Los productos especiales de la categoría con nombre “Apple” que tengan descuento”.**

List<Producto> findByEspecialTrueAndCategoriaNombreAndDescuentoIsNull(String categoriaNombre);

**Los productos donde el nombre del mismo o el nombre de la categoría sea ‘ordenador’. Dan igual mayus/minus.**

public List<Producto> findByNombreOrCategoriaNombreAllIgnoreCase(String nombre, String categoriaNombre);

public List<Producto> buscarPorNombre(String nombre){

return repositorio.findBy…(nombre, nombre);

}

**SQL →** SELECT \* FROM {Country}->tabla

**JPQL →** SELECT FROM {Country}->entidad {c}->alias

WHERE c.population > 50M

En el repositorio

@Query(“””SELECT a FROM Alumno a

WHERE a.fecha\_nac >= ? AND a.curso.id = 72

“””)

List<Country>elQueYoQuiera (LocalDate f, Long id)

**TE VA A SERVIR PARA EL PROYECTO:**

@Query(“””

SELECT v

FROM Venta v

LEFT JOIN v.listadoLineasVenta lv

WHERE lv.producto = ?1

“””)

List<Venta>ventasConProducto(Producto p);

***CARRITO***

Añadir atributo **“private boolean finalizada”** a la clase **“Venta”**.

**//Calcular subtotal (Subtotal = Precio Final \* Cantidad)**

public double getPrecioLineaDeVenta(){

return producto.getPvp() \* cantidad;

}

SERVICIOS

**Producto Usuario →** Nada.

**Venta →**

public boolean existeVentaSinFinalizar(Usuario u){

return this.repositorio.existsByFinalizadaAndUsuario(false, u);

}

public boolean existeVentaSinFinalizar(Usuario u){

return this.repositorio.existsByFinalizadaUsuario();

}

public boolean hayProductosEnElCarrito (){

return this.repository.

}

public void addProducto(Usuario u, Producto p, int cantidad){

Venta carrito = getCarrito(u);

if(! ventaServicio.hayProductoEnCarrito(u, p)){

carrito.addLineaDeVenta(

LineaDeVenta.builder()

.producto(p)

.cantidad(cantidad)

.build()

);

}else{

Optional<LineaDeVenta> lv = buscarPorProducto(u, p);

if(lv.isPresent()){

modificarCantidad(u, p, lv.get().getCantidad()+1);

}

}

ventaServicio

}

public Optional <LineaDeVenta> buscarPorProducto(Usuario u, Producto p){

}

Venta carrito = getCarrito(usuario);

if(cantidad <= 0)

eliminarProducto(usuario, producto){

else{

Optional<LineaDeVenta> aModificar = buscarPorProducto(usuario, producto);

if(aModificar.isPresent())

public void eliminarProducto(Usuario usuario, Producto producto){

Venta carrito =

private Optional<LineaDeVenta> buscarPorProducto(Usuario usuario, Producto producto){

Venta carrito = getCarrito

public void finalizarCompra(Usuario usuario){

Venta carrito = getCarrito(usuario);

carrito.setFinalizada(true);

carrito.set

**public Map<Producto, Integer> getProductosEnCarritos(Usuario usuario){**

**return getCarrito(usuario).getLineasVenta().stream().collect(Collectors.toMap(lv -> lv.getProducto(), lv -> lv.getCantidad()));**

public double getImporteTotal(Usuario usuario){

}

public String showCarrito(“/carrito”){

@AuthenticationPrincipal Usuario usuario, Model model){

if (carritoServicio.hayCarritoCreado

@GetMapping(“/productoACarrito/{id}”)

public String productoACarrito{

@AuthenticationPrincipal

@PostMapping(“/carrito/checkout”)

public String finalizarCompra(@AuthenticationPrincipal

@ GetMapping(“/carrito/modificar/{id}/cantidad/{cant}”)

public String modificarCantidadProducto(@AuthenticationPrincipal Usuario usuario){

${#number.formalCurrency(producto.getPrecioFinal())}

SERVICIOS

VentaServicio → métodos sólo de comprobar

boolean existeVentaSinFinalizar(Usuario u)

Optional<Venta> getVentaSinFinalizar(Usuario u)

public boolean hayProductoEnCarrito(Usuario u, Producto p){

return

}

—--------------------------------------------------------------------------------------------

Public void addProducto(Usuario u, Producto p, int cantidad){

}