Universidad Don Bosco

Desarrollo de Software para Móviles DSM941

Tema: Trabajo de investigación - Proyecto de Carrito de Compras en Kotlin con Facturación

Docente: Alexander Alberto Siguenza Campos

Integrantes:

Villalta Reinoza, Luis Edgardo VR181981

Fecha de entrega: 3 de marzo de 2024

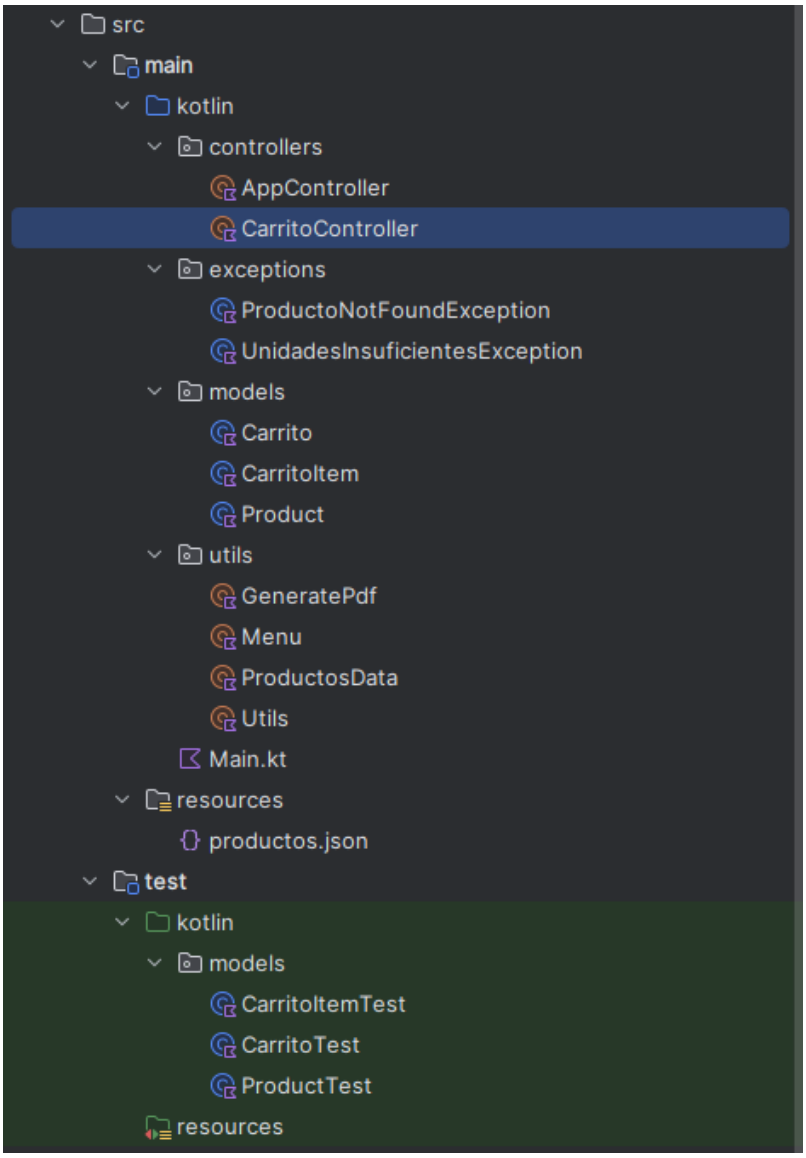
Ciclo: 01 – 2024

Contenido

No se encontraron entradas de tabla de contenido.

Estructura del código

La estructura del proyecto podemos localizarla dentro de la carpeta **src** del mismo. Dentro de esta carpeta se encuentran los archivos main y test, dentro de la carpeta main se encuentra subdivido el proyecto en las siguientes carpetas: controllers, exceptions, models y utils, además, también se encuentra el archivo main.kt. De igual forma en main se encuentra la carpeta resources con el archivo productos.json (utilizado para brindar información a nuestro aplicativo).



Modelos

Carrito

Propiedades:

- items (private val items: MutableList<CarritoItem>):
    - Tipo: MutableList<CarritoItem>
    - Descripción: Lista mutable que almacena los elementos del carrito.
- isEmpty (val isEmpty: Boolean):
    - Tipo: Boolean
    - Descripción: Propiedad de solo lectura que indica si el carrito está vacío, verificando si la lista de elementos cartItems está vacía.
- cartItems (val cartItems: MutableList<CarritoItem>):
    - Tipo: MutableList<CarritoItem>
    - Descripción: Propiedad de solo lectura que filtra los elementos del carrito para incluir solo aquellos con una cantidad mayor a 0.
- total (val total: Double):
    - Tipo: Double
    - Descripción: Propiedad de solo lectura que calcula el total del carrito sumando el precio total de cada elemento, teniendo en cuenta la cantidad de unidades.

Funciones:

- items (private val items: MutableList<CarritoItem>):
    - Tipo: MutableList<CarritoItem>
    - Descripción: Lista mutable que almacena los elementos del carrito.
- isEmpty (val isEmpty: Boolean):
    - Tipo: Boolean
    - Descripción: Propiedad de solo lectura que indica si el carrito está vacío, verificando si la lista de elementos cartItems está vacía.
- cartItems (val cartItems: MutableList<CarritoItem>):
    - Tipo: MutableList<CarritoItem>

    - Descripción: Propiedad de solo lectura que filtra los elementos del carrito para incluir solo aquellos con una cantidad mayor a 0.
- total (val total: Double):
    - Tipo: Double
    - Descripción: Propiedad de solo lectura que calcula el total del carrito sumando el precio total de cada elemento, teniendo en cuenta la cantidad de unidades.

Carrito Item

Propieades:

- id (var id: Int):
    - Tipo: Int
    - Descripción: Identificador único del elemento en el carrito.
- product (val product: Product):
    - Tipo: Product
    - Descripción: Representa el producto asociado al elemento del carrito.
- cantidad (var cantidad: Int):
    - Tipo: Int
    - Descripción: La cantidad de unidades de este producto en el carrito.
- subtotal (val subtotal: Double):
    - Tipo: Double
    - Descripción: Propiedad de solo lectura que calcula el subtotal multiplicando el precio del producto por la cantidad de unidades en el carrito.

Funciones:

- agregarUnidades(qty: Int): Boolean:
    - Parámetros:
        - qty (Int): La cantidad de unidades a agregar al carrito.
    - Retorno: Boolean
    - Descripción: Agrega la cantidad especificada al total de unidades en el carrito, verificando si la cantidad total no supera la cantidad

disponible del producto. Lanza una excepción si la cantidad ingresada es negativa o si excede la cantidad disponible.

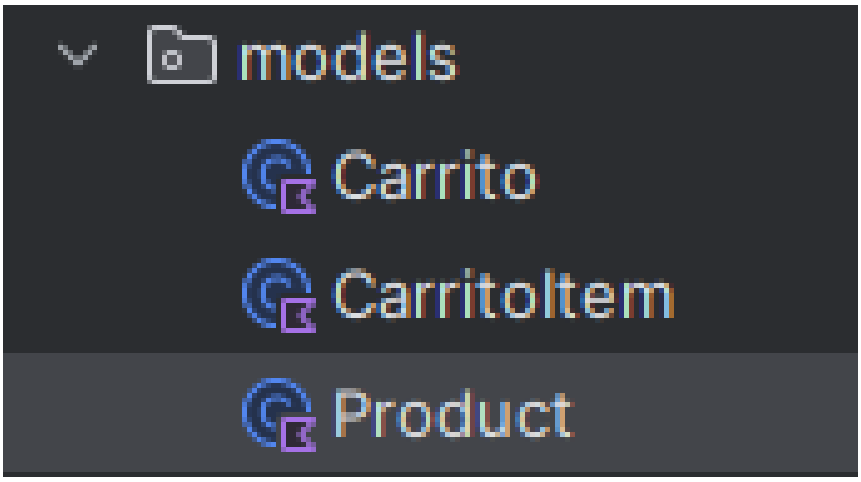
- restarUnidades(qty: Int): Boolean:
  - Parámetros:
    - qty (Int): La cantidad de unidades a restar del carrito.
  - Retorno: Boolean
  - Descripción: Resta la cantidad especificada del total de unidades en el carrito, verificando si la cantidad a restar no es negativa ni mayor que la cantidad actual en el carrito. Lanza una excepción si la cantidad a restar es menor que 0 o si es mayor que la cantidad actual en el carrito.

Product

Propiedades

- id (val id: Int):
  - Tipo: Int
  - Descripción: Identificador único del producto.
- nombre (val nombre: String):
  - Tipo: String
  - Descripción: Nombre del producto.
- precio (val precio: Double):
  - Tipo: Double
  - Descripción: Precio del producto.
- cantidadDisponible (var cantidadDisponible: Int):
  - Tipo: Int
  - Descripción: Cantidad de unidades disponibles del producto.
- valorTotal (val valorTotal: Double):
  - Tipo: Double

    - Descripción: Propiedad de solo lectura que calcula el valor total del inventario del producto multiplicando la cantidad disponible por el precio.
- disponible (val disponible: Boolean):
    - Tipo: Boolean
    - Descripción: Propiedad de solo lectura que indica si hay unidades disponibles del producto (cantidadDisponible > 0).



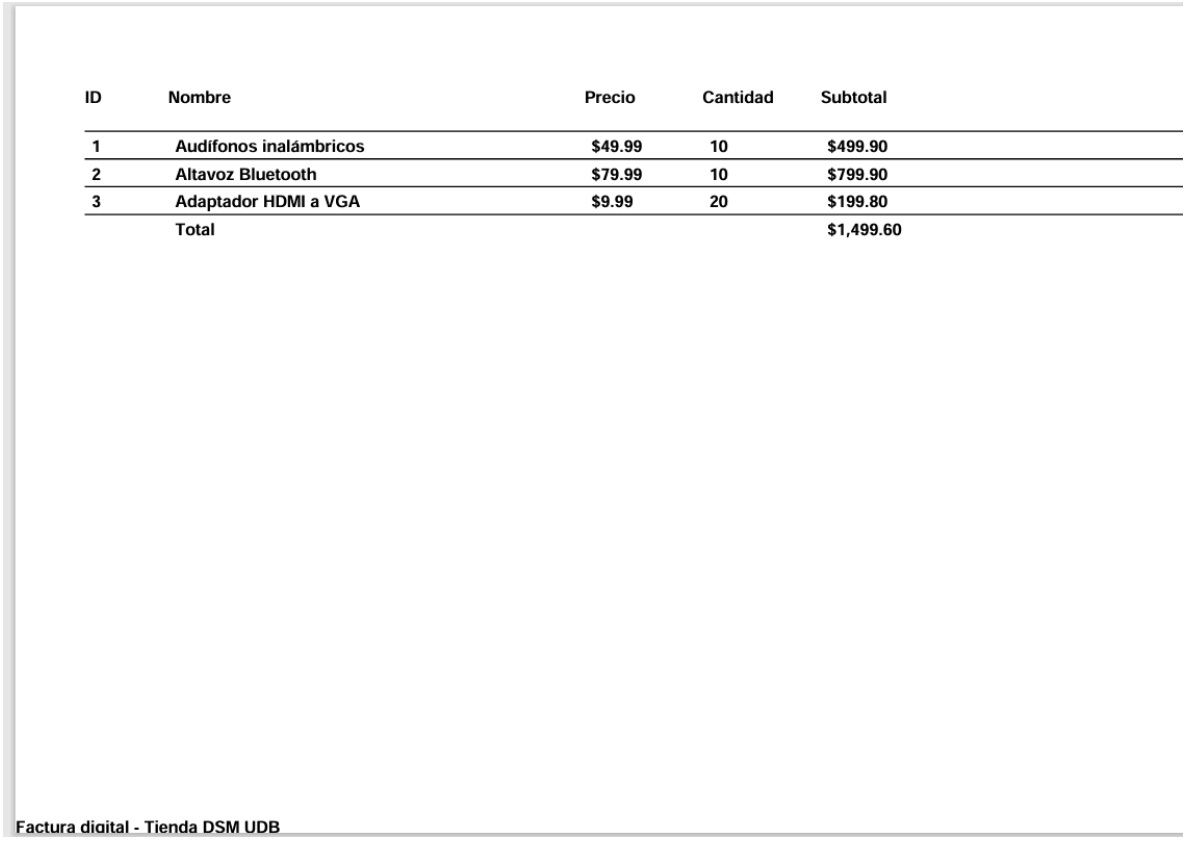
Utils

Generate PDF

Función generate(carrito: Carrito):

- Parámetros:
  - carrito (Carrito): El carrito de compras del cual se generará la factura.
- Acciones:
  - Creación del Documento PDF:
    1. Se crea un nuevo documento PDF (PDDocument).
    2. Se añade una página al documento (PDPage).
  - Configuración del Contenido:
    1. Se configura un stream de contenido de página (PDPageContentStream).
    2. Se establece la fuente para el contenido como "Helvetica Bold" con un tamaño de 12 puntos.
  - Configuración de la Tabla y Encabezados:
    1. Se definen variables y dimensiones para la tabla, columnas y márgenes.
    2. Se dibujan los encabezados de la tabla, incluyendo "ID", "Nombre", "Precio", "Cantidad", y "Subtotal".
  - Iteración sobre los Elementos del Carrito:
    1. Se itera sobre los elementos del carrito (carrito.cartItems).
    2. Para cada elemento, se dibuja una fila en la tabla con información como ID, nombre del producto, precio, cantidad y subtotal.
  - Cálculo y Dibujo del Total:
    1. Se calcula el total del carrito y se añade una fila adicional en la tabla con el total.
  - Cierre del Contenido de la Página:
    1. Se cierra el stream de contenido de página.

- Guardado y Cierre del Documento:
    1. Se guarda el documento PDF con un nombre que incluye la fecha y hora actual.
    2. Se cierra el documento.



ID	Nombre	Precio	Cantidad	Subtotal
1	Audífonos inalámbricos	$49.99	10	$499.90
2	Altavoz Bluetooth	$79.99	10	$799.90
3	Adaptador HDMI a VGA	$9.99	20	$199.80
	Total			$1,499.60

Factura digital - Tienda DSM UDB

ProductosData

readProductos(): List<Product>

- Descripción: Lee y carga la lista de productos desde un archivo JSON.
- Acciones:
    - Utiliza la biblioteca Gson para realizar la deserialización del contenido del archivo JSON.
    - Devuelve la lista de productos leída del archivo.
- Retorno:
    - List<Product>: Lista de productos cargada desde el archivo JSON.

listarProductos(listProductos: MutableList<Product>)

- Parámetros:
    - listProductos (MutableList<Product>): Lista de productos a listar.
- Descripción: Imprime en la consola una tabla con la información básica de cada producto en la lista.
- Acciones:
    - Imprime una tabla en la consola con columnas para ID, Nombre, Precio y Cantidad Disponible.
    - Utiliza formatos específicos para alinear la información en la tabla.
- Salida:
    - Muestra en la consola la tabla formateada con la información de los productos.

Utils

generateString(input: String, char: Char = '-', length: Int = 32): String

- Parámetros:
  - input (String): La cadena original.
  - char (Char): El carácter utilizado para el relleno, por defecto es '-'.
  - length (Int): La longitud total de la cadena generada, por defecto es 32.
- Descripción: Genera una nueva cadena añadiendo relleno a ambos lados de la cadena original.
- Retorno:
  - String: La cadena original con relleno añadido.

limpiarConsola()

- Descripción: Limpia la consola imprimiendo líneas en blanco.
- Acciones:
  - Imprime 50 líneas en blanco en la consola, proporcionando una forma de "limpiar" visualmente la consola.

formatCurrency(qty: Double): String

- Parámetros:
  - qty (Double): Cantidad numérica que se formateará como moneda.
- Descripción: Formatea una cantidad numérica como una cadena de moneda en dólares (USD).
- Retorno:
  - String: La cantidad formateada como una cadena de moneda en dólares.

Menu

mostrarMenu()

- Descripción: Muestra el menú principal de la aplicación.
- Acciones:
    - Imprime opciones para seleccionar productos, ver el carrito y salir.
    - Solicita al usuario que ingrese una opción.

verCarrito(carrito: Carrito)

- Parámetros:
    - carrito (Carrito): El carrito de compras a visualizar.
- Descripción: Muestra el contenido del carrito de compras.
- Acciones:
    - Imprime una tabla con información detallada de cada elemento en el carrito.
    - Si el carrito está vacío, imprime un mensaje indicando que no hay productos en el carrito.

carritoMenu()

- Descripción: Muestra un submenú para realizar operaciones en el carrito.
- Acciones:
    - Imprime opciones para proceder al pago, añadir más unidades, eliminar un producto, vaciar el carrito y regresar al menú principal.

Controllers

AppController

start()

- Descripción: Inicia la aplicación y llama a la función ejecutarMenu para gestionar el menú principal.
- Acciones:
    - Imprime un mensaje de bienvenida.
    - Llama a ejecutarMenu para gestionar el menú principal.

ejecutarMenu()

- Descripción: Gestiona la ejecución del menú principal de la aplicación.
- Acciones:
    - Utiliza un bucle do-while para permitir al usuario seleccionar opciones del menú.
    - Muestra el menú principal utilizando Menu.mostrarMenu().
    - Captura excepciones de InputMismatchException para manejar entradas no válidas.
    - Ejecuta acciones según la opción seleccionada por el usuario.
    - El bucle continúa hasta que el usuario selecciona la opción para salir (opción 3).

CarritoController

agregarAlCarritoMenu()

- Descripción: Permite al usuario seleccionar productos para agregar al carrito.
- Acciones:
    - Muestra la lista de productos con sus detalles.
    - Solicita al usuario ingresar el ID del producto que desea agregar al carrito.
    - Llama a addToCarrito para procesar la adición del producto al carrito.

verCarrito()

- Descripción: Muestra el contenido del carrito de compras y proporciona un submenú para realizar operaciones en el carrito.
- Acciones:
    - Muestra la información detallada de los productos en el carrito.
    - Presenta opciones para proceder al pago, añadir más unidades, eliminar productos, vaciar el carrito o regresar al menú principal.
    - Llama a funciones correspondientes según la opción seleccionada.

removerProductos()

- Descripción: Permite al usuario especificar el ID de un producto para eliminarlo del carrito.
- Acciones:
    - Solicita al usuario ingresar el ID del producto que desea eliminar.
    - Llama a removeToCarrito para procesar la eliminación del producto del carrito.

addMasProducto()

- Descripción: Permite al usuario especificar el ID de un producto para agregar más unidades al carrito.
- Acciones:
    - Solicita al usuario ingresar el ID del producto que desea agregar.
    - Llama a addToCarrito para procesar la adición de más unidades al carrito.

addToCarrito(productId: Int)

- Parámetros:
    - productId (Int): El ID del producto que se desea agregar al carrito.
- Descripción: Agrega un producto al carrito, solicitando al usuario la cantidad deseada.
- Acciones:
    - Verifica la disponibilidad del producto y solicita la cantidad al usuario.
    - Agrega el producto al carrito y actualiza la disponibilidad del producto.

removeToCarrito(productId: Int)

- Parámetros:
    - productId (Int): El ID del producto que se desea eliminar del carrito.
- Descripción: Elimina un producto del carrito, solicitando al usuario la cantidad a eliminar.
- Acciones:
    - Solicita al usuario la cantidad de unidades del producto a eliminar.
    - Elimina el producto del carrito y actualiza la disponibilidad del producto.

procederPago()

- Descripción: Simula un proceso de pago, generación de factura y actualización del carrito.
- Acciones:
    - Muestra el contenido del carrito.
    - Pregunta al usuario si desea proceder con el pago.
    - Si el usuario elige proceder, simula un proceso de pago y generación de factura.
    - Llama a GeneratePdf.generate para generar la factura en formato PDF.
    - Vacia el carrito y muestra un mensaje de confirmación.

vaciarCarrito()

- Descripción: Elimina todos los productos del carrito y restaura las cantidades disponibles en la lista de productos.
- Acciones:
    - Itera sobre los elementos del carrito y elimina cada producto del carrito.
    - Actualiza la disponibilidad de cada producto en la lista de productos.

Anexos

Video demostrativo

https://drive.google.com/file/d/1ILFBP3M4Jtzqv_1x5Ot-Jz49zw_lT9iz/view?usp=sharing

Repositorio (Github)

https://github.com/LuisVillalta3/shoppingCart-dsm-udb