**Universidad Don Bosco**


Dirección de Educación a Distancia

Desarrollo de Software para Móviles

Estudiantes:

José Roberto Ochoa Medrano OM172014

Adriana Guadalupe Martínez Soto MS211714

Grupo teórico:

01T

Profesor:

Alexander Alberto Siguenza Campos

Contenido:

# Foro 1: Proyecto de Carrito de Compras en Kotlin con Facturación

03 de marzo del 2024

## Kotlin

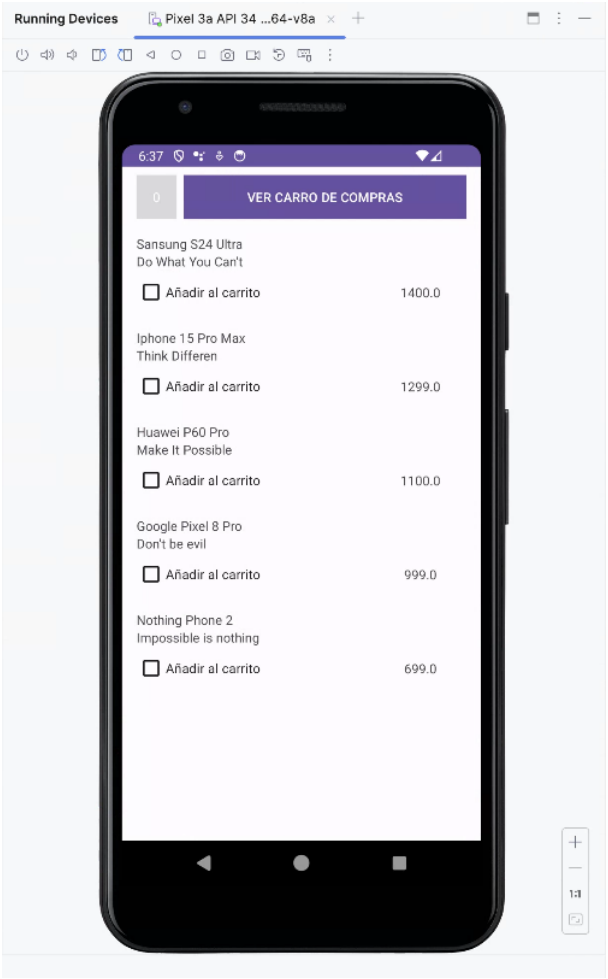
El lenguaje de programación Kotlin, desarrollado por JetBrains, se ha ganado popularidad como una alternativa de código abierto y tipado estático para el desarrollo de aplicaciones, especialmente en el contexto de Android.

Entre las características y ventajas destacadas de Kotlin, se incluyen:

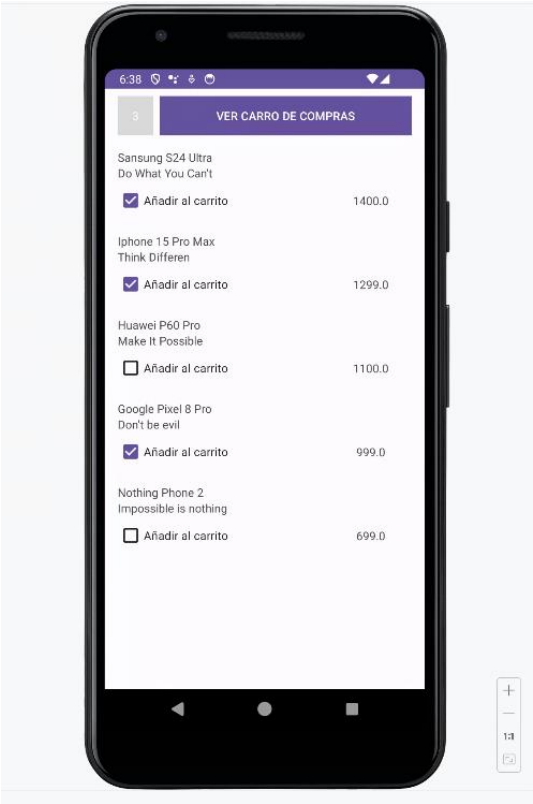
- Interoperabilidad con código Java: Kotlin está diseñado para integrarse sin problemas con el código Java existente, permitiendo una transición suave y efectiva entre ambos lenguajes.
- Curva de aprendizaje sencilla: Gracias a su sintaxis clara y concisa, Kotlin ofrece una curva de aprendizaje más accesible, lo que lo convierte en una excelente opción para quienes están comenzando en la programación.
- Menor tiempo de programación: Kotlin elimina la redundancia en el código y ofrece una sintaxis más compacta y legible, lo que agiliza el proceso de desarrollo y reduce la posibilidad de errores.
- Orientación a objetos y programación funcional: Aunque Kotlin se basa en la orientación a objetos, también ofrece soporte para la programación funcional, lo que brinda a los desarrolladores flexibilidad en la elección del paradigma de programación.
- Corrutinas: Kotlin incluye corrutinas, que simplifican la programación asíncrona al evitar el uso de callbacks, lo que facilita el manejo de tareas como llamadas de red y acceso a bases de datos.
- Desarrollo multiplataforma: Kotlin es adecuado para el desarrollo en una variedad de plataformas, incluyendo web, Android e iOS, lo que permite compartir código entre diferentes sistemas de forma eficiente.
- Flexibilidad: Kotlin ofrece a los desarrolladores la libertad de trabajar con diferentes estilos de programación, ya sea funcional u orientado a objetos, lo que lo convierte en un lenguaje altamente adaptable a las necesidades del proyecto.

## Funcionamiento de la aplicación

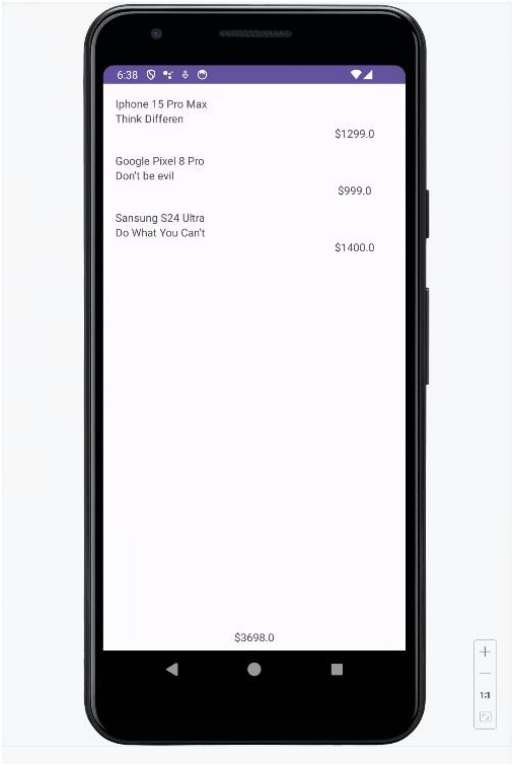
En la primera pantalla se puede observar el catalogo de los productos que ofrece la aplicación, además del botón que permite ver el carrito de compras y la cantidad de ítems seleccionados.




Una vez que se seleccionan ítems del catalogo la pantalla pasa a mirarse de la siguiente forma:




Si se presiona el botón “Ver carrito” se muestra el total a pagar por los ítems que se han seleccionado:



El código incluye la visualización de productos, la gestión del carrito de compras y la navegación entre diferentes pantallas:

“AdaptadorCarroCompras”

Este adaptador se encarga de mostrar los elementos del carrito de compras en un RecyclerView y calcular el total de la compra en función de los precios de los productos en el carrito.

```kotlin
package com.example.carrocompras_kotlin

import ...

class AdaptadorCarroCompras(
    var tvTotal: TextView,
    var carroCompras: ArrayList<Producto>
): RecyclerView.Adapter<AdaptadorCarroCompras.ViewHolder>() {

    var total: Double = 0.0

    class ViewHolder(itemView: View): RecyclerView.ViewHolder(itemView) {
        val tvNomProducto = itemView.findViewById(R.id.tvNomProducto) as TextView
        val tvDescripcion = itemView.findViewById(R.id.tvDescripcion) as TextView
        val tvPrecio = itemView.findViewById(R.id.tvPrecio) as TextView
    }

    override fun onCreateViewHolder(
        parent: ViewGroup,
        viewType: Int
    ): ViewHolder {
        val vista = LayoutInflater.from(parent.context).inflate(R.layout.item_rv_carro_compras, parent, false)

        total = 0.0

        carroCompras.forEach {
            total += it.precio
        }

        tvTotal.text = "$$total"

        return ViewHolder(vista)
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val producto = carroCompras[position]

        holder.tvNomProducto.text = producto.nomProducto
        holder.tvDescripcion.text = producto.descripcion
        holder.tvPrecio.text = "$${producto.precio}"
    }

    override fun getItemCount(): Int {
        return carroCompras.size
    }

}
```

“CarroComprasActivity”

Esta actividad muestra la lista de productos del carrito de compras utilizando RecyclerView y View Binding. Se encarga de configurar el adaptador y manejar la lógica para mostrar los datos correctamente.

```kotlin
package com.example.carrocompras_kotlin

import ...

class CarroComprasActivity : AppCompatActivity() {

    private lateinit var binding: ActivityCarroComprasBinding
    private lateinit var adapter: AdaptadorCarroCompras

    var carritodeCompras = ArrayList<Producto>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityCarroComprasBinding.inflate(layoutInflater)
        setContentView(binding.root)

        carritodeCompras = intent.getSerializableExtra("carro_compras") as ArrayList<Producto>

        setupRecyclerView()
    }

    fun setupRecyclerView() {
        binding.rvListaCarro.layoutManager = LinearLayoutManager(this)
        adapter = AdaptadorCarroCompras(binding.tvTotal, carritodeCompras)
        binding.rvListaCarro.adapter = adapter
    }
}
```

“Adaptadorproducto”

Este adaptador se encarga de mostrar los productos en un RecyclerView y manejar la lógica para añadir o quitar productos del carrito de compras cuando se marca/desmarca el CheckBox correspondiente. Además, permite al usuario ver el contenido del carrito de compras al hacer clic en el botón btnVerCarro.

```kotlin
package com.example.carrocompras_kotlin

import ...

class AdaptadorProducto(
    var context: Context,
    var tvCantProductos: TextView,
    var btnVerCarro: Button,
    var listproducts: ArrayList<Producto>,
    var carroCompras: ArrayList<Producto>
): RecyclerView.Adapter<AdaptadorProducto.ViewHolder>() {

    class ViewHolder(itemView: View): RecyclerView.ViewHolder(itemView) {
        val tvNomProducto = itemView.findViewById(R.id.tvNomProducto) as TextView
        val tvDescripcion = itemView.findViewById(R.id.tvDescripcion) as TextView
        val cbCarro = itemView.findViewById(R.id.cbCarro) as CheckBox
        val tvPrecio = itemView.findViewById(R.id.tvPrecio) as TextView
    }

    override fun onCreateViewHolder(
        parent: ViewGroup,
        viewType: Int
    ): ViewHolder {
        var vista = LayoutInflater.from(parent.context).inflate(R.layout.item_rv_productos, parent, false)
        return ViewHolder(vista)
    }
```

```kotlin
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val producto = listproducts[position]

        holder.tvNomProducto.text = producto.nomProducto
        holder.tvDescripcion.text = producto.descripcion
        holder.tvPrecio.text = producto.precio.toString()

        holder.cbCarro.setOnCheckedChangeListener { compoundButton, b ->
            if (holder.cbCarro.isChecked) {
                tvCantProductos.text = "${Integer.parseInt(tvCantProductos.text.toString().trim()) + 1}"
                carroCompras.add(listproducts[position]) ^setOnCheckedChangeListener
            } else {
                tvCantProductos.text = "${Integer.parseInt(tvCantProductos.text.toString().trim()) - 1}"
                carroCompras.remove(listproducts[position]) ^setOnCheckedChangeListener
            }
        }

        btnVerCarro.setOnClickListener {
            val intent = Intent(context, CarroComprasActivity::class.java)
            intent.putExtra("carro_compras", carroCompras)
            context.startActivity(intent) ^setOnClickListener
        }
    }

    override fun getItemCount(): Int {
        return listproducts.size
    }
}
```

“MainActivity”

Esta actividad es la pantalla principal de la aplicación y muestra una lista de productos disponibles en un RecyclerView. Los usuarios pueden agregar productos al carrito de compras haciendo clic en el botón correspondiente.

```kotlin
package com.example.carrocompras_kotlin
import ...
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding
    private lateinit var adapter: AdaptadorProducto
    var productlist = ArrayList<Producto>()
    var carrito = ArrayList<Producto>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        addproducts()
        setupRecyclerView()
    }
    fun setupRecyclerView() {
        binding.rvListaProductos.layoutManager = LinearLayoutManager(this)
        adapter = AdaptadorProducto(this, binding.tvCantProductos, binding.btnVerCarro, productlist, carrito)
        binding.rvListaProductos.adapter = adapter
    }
    fun addproducts() {
        productlist.add(Producto("101", "Product No 1", "Made with natural ingredients", 40.00))
        productlist.add(Producto("102", "Product No 2", "Made with premium materials", 30.00))
        productlist.add(Producto("103", "Product No 3", "Wooden chair made with natural materials", 90.00))
        productlist.add(Producto("104", "Product No 4", "Made with natural ingredients", 50.00))
        productlist.add(Producto("105", "Product No 5", "Brand new high heels shoes", 80.00))
    }
}
```

“Producto”

Aquí se encuentran las que clases proporcionan una estructura de datos básica para representar productos y elementos de carrito de compras en la aplicación. La implementación de Serializable permite que estos objetos se pasen entre componentes de la aplicación, como actividades, fragmentos o servicios, utilizando Intents u otros mecanismos de comunicación en Android

```kotlin
package com.example.carrocompras_kotlin

import java.io.Serializable

data class Producto(
    var idProducto: String,
    var nomProducto: String,
    var descripcion: String,
    var precio: Double
): Serializable
```