

2025



DISPOSITIVOS MOVILES

Integrantes:

- Kelly Ledesma
- Ariel Elizalde
- Edison Enríquez
- Ángelo Silva
- Alexis Troya
- Stalin Acurio
- Mauricio López

Contenido

Documentación Técnica - ProducTrack.....	4
Información General	4
Descripción del Proyecto	4
Características Principales:	4
Arquitectura del Proyecto.....	5
Patrón de Diseño: MVC (Modelo-Vista-Controlador).....	5
Componentes del Proyecto.....	6
1 MODELO (Model)	6
2 CONTROLADOR (Controller)	7
3 VISTA (View)	9
4 NAVEGACIÓN	22
Tema y Estilos	23
Color.kt	23
Colores en Recursos (colors.xml)	23
Tema Material Design 3	23
AndroidManifest.xml	24
Dependencias (build.gradle.kts).....	24
Plugins:	24
Configuración Android:	25
Dependencias Principales:	25
Flujo de Uso de la Aplicación.....	25
1. Inicio de la App	25

2. Flujo de Login	26
3. Gestión de Productos	26
4. Formulario de Producto	26
Datos de Prueba	26
Usuarios Predefinidos:	26
Productos Predefinidos:	27
<input checked="" type="checkbox"/> Validaciones Implementadas	27
Login:	27
Registro:	27
Productos:	27
Teclado Virtual:	27
Casos de Uso Detallados	28
Caso 1: Usuario Nuevo se Registra	28
Caso 2: Login con Credenciales Existentes	28
Caso 3: Agregar Producto Nuevo	28
Caso 4: Editar Producto Existente	29
Caso 5: Eliminar Producto	29
Caso 6: Cerrar Sesión	29
Características Técnicas Destacadas	29
Reactividad con Compose:	29
Gestión de Estado:	30
Material Design 3:	30
Experiencia de Usuario:	30
Estadísticas del Proyecto	30
Archivos de Código:	30

Pantallas Implementadas:	30
Componentes Reutilizables:	31
⚠ Limitaciones Conocidas	31
Almacenamiento:	31
Seguridad:	31
Validaciones:	31
Navegación:	32
📚 Recursos y Referencias.....	32
Documentación Oficial:	32
Bibliotecas Utilizadas:	32
📄 Licencia	32
📝 Notas Finales.....	32

Documentación Técnica - ProducTrack

Información General

Nombre del Proyecto: ProducTrack

Versión: 1.0

Plataforma: Android

Lenguaje: Kotlin

Framework UI: Jetpack Compose

Arquitectura: MVC (Modelo-Vista-Controlador)

SDK Mínimo: Android 7.0 (API 24)

SDK Target: Android 14+ (API 36)

Descripción del Proyecto

ProducTrack es una aplicación móvil Android diseñada para la gestión y control de inventarios de productos. Permite a los usuarios registrarse, iniciar sesión y administrar un catálogo completo de productos con funcionalidades CRUD (Crear, Leer, Actualizar, Eliminar).

Características Principales:

- Sistema de autenticación (Login/Registro)
- Gestión completa de productos (CRUD)
- Interfaz moderna con Material Design 3
- Validación de formularios
- Selector de fechas interactivo
- Almacenamiento en memoria (runtime)
- Navegación entre pantallas

Arquitectura del Proyecto

Patrón de Diseño: MVC (Modelo-Vista-Controlador)

```
src/main/java/com/example/tarea07_u2_g2/
|
|   └── model/          # MODELO - Clases de datos
|       ├── Usuario.kt    # Estructura de datos de usuario
|       └── Producto.kt    # Estructura de datos de producto
|
|   └── view/           # VISTA - Interfaces de usuario
|       ├── LoginScreen.kt  # Pantalla de inicio de sesión
|       ├── RegistroScreen.kt # Pantalla de registro
|       ├── HomeScreen.kt    # Pantalla principal (lista)
|       ├── ProductoFormScreen.kt # Formulario agregar/editar
|       ├── DatePicker.kt     # Componente selector de fecha
|       └── Logo.kt          # Componentes de marca
|
|   └── controller/      # CONTROLADOR - Lógica de negocio
|       └── Controlador.kt  # Gestor central de datos
|
|   └── ui/theme/        # Tema visual
|       ├── Color.kt
|       ├── Theme.kt
|       └── Type.kt
|
└── MainActivity.kt      # Punto de entrada y navegación
```

Componentes del Proyecto

1 MODELO (Model)

Usuario.kt

```
data class Usuario(  
    val nombre: String,  
    val apellido: String,  
    val clave: String  
)
```

Descripción: - Representa un usuario del sistema - Contiene credenciales para autenticación - Inmutable gracias a data class

Campos: - nombre: Nombre del usuario - apellido: Apellido del usuario - clave: Contraseña (almacenada sin cifrar - solo para demo)

Producto.kt

```
data class Producto(  
    val codigo: String,      // Identificador único  
    val descripcion: String, // Nombre del producto  
    val precio: Double,     // Costo del producto  
    val fechaFabricacion: String, // Fecha en formato dd/MM/yyyy  
    val disponibilidad: Boolean // Estado de stock  
)
```

Descripción: - Representa un producto en el inventario - Contiene toda la información necesaria para su gestión

Campos: - codigo: Identificador numérico único (no editable) - descripcion: Nombre descriptivo del producto - precio: Valor monetario en dólares - fechaFabricacion: Fecha de producción - disponibilidad: true = Disponible, false = Agotado

2 CONTROLADOR (Controller)

Controlador.kt

Objeto Singleton que gestiona toda la lógica de negocio

Almacenamiento de Datos:

```
val usuarios = mutableStateListOf<Usuario>(  
    Usuario("Edison", "Enriquez", "1234"),  
    Usuario("Kelly", "Ledesma", "1234"),  
    Usuario("Stalin", "Acurio", "1234"),  
    Usuario("Ariel", "Elizalde", "1234"),  
    Usuario("Mauricio", "Lopez", "1234"),  
    Usuario("Angelo", "Silva", "1234"),  
    Usuario("Alexis", "Troya", "1234")  
)
```

```
val productos = mutableStateListOf<Producto>(  
    Producto("001", "Laptop HP", 850.00, "01/01/2024", true),  
    Producto("002", "Mouse Logitech", 25.50, "15/02/2024", true),  
    Producto("003", "Teclado Gamer", 45.00, "10/03/2024", false)  
)
```

Características: - Usa mutableListOf para reactividad en Compose - Los cambios se reflejan automáticamente en la UI - Datos precargados para pruebas

Funciones de Usuario:

1. Validar Login

```
fun validarLogin(nombre: String, apellido: String, clave: String): Boolean
```

- Compara credenciales con usuarios registrados
- Ignora mayúsculas/minúsculas en nombre y apellido
- Contraseña debe coincidir exactamente

- **Retorna:** true si las credenciales son válidas

2. Registrar Usuario

```
fun registrarUsuario(nombre: String, apellido: String, clave: String)
```

- Agrega un nuevo usuario a la lista
 - No valida duplicados (simplificación)
 - Los datos persisten durante la sesión de la app
-

Funciones de Productos:

1. Agregar Producto

```
fun agregarProducto(producto: Producto)
```

- Añade un nuevo producto al inventario
- El código debe ser único (validado en UI)

2. Editar Producto

```
fun editarProducto(productoEditado: Producto)
```

- Busca el producto por código
- Reemplaza todos sus datos excepto el código
- Si no encuentra el producto, no hace nada

3. Eliminar Producto

```
fun eliminarProducto(producto: Producto)
```

- Remueve el producto de la lista
 - Actualización inmediata en UI
-

3 VISTA (View)

LoginScreen.kt

Pantalla de inicio de sesión

Funcionalidad:

- Solicita Nombre, Apellido y Contraseña
- Valida credenciales contra el controlador
- Muestra mensajes Toast de éxito/error
- Navegación a HomeScreen o RegistroScreen

Componentes UI:

@Composable

```
fun LoginScreen(  
    onLoginSuccess: () -> Unit,  
    onNavigateToRegister: () -> Unit  
)
```

Elementos: - AppLogo(): Logo circular de la app - AppName(): Nombre “ProducTrack” y tagline - 3 OutlinedTextField: - Nombre (capitalización automática) - Apellido (capitalización automática) - Contraseña (oculta con asteriscos) - Botón “INGRESAR” - TextButton “¿Nuevo usuario? Regístrate aquí”

Navegación del Teclado: - Enter en Nombre → Foco a Apellido - Enter en Apellido → Foco a Contraseña - Enter en Contraseña → Intenta login

Validación:

Controlador.validarLogin(nombre, apellido, clave)

- Éxito → Toast “¡Bienvenido {nombre}!” → HomeScreen
- Error → Toast “Nombre, Apellido o Clave incorrectos”



RegistroScreen.kt

Pantalla de creación de nuevos usuarios

Funcionalidad:

- Formulario con 3 campos (Nombre, Apellido, Contraseña)
- Validación de campos no vacíos
- Registra usuario y regresa a Login

Componentes UI:

@Composable

fun RegistroScreen(onRegistroSuccess: () -> Unit)

Elementos: - Título “REGISTRO DE USUARIO” - 3 campos de texto con las mismas características que Login - Botón “GUARDAR Y VOLVER” - TextButton “Cancelar”

Flujo de Registro: 1. Usuario completa los campos 2. Presiona “GUARDAR Y VOLVER” 3. Validación de campos completos 4. Si válido → Controlador.registrarUsuario() 5. Toast “Registrado con éxito” 6. Regresa a LoginScreen

Validación:

```
if (nombre.isNotEmpty() && apellido.isNotEmpty() && clave.isNotEmpty()) {  
    Controlador.registrarUsuario(nombre, apellido, clave)  
}
```



HomeScreen.kt

Pantalla principal - Lista de productos

Funcionalidad:

- Muestra lista completa de productos
- Operaciones: Agregar, Editar, Eliminar
- Cerrar sesión (vuelve a Login)

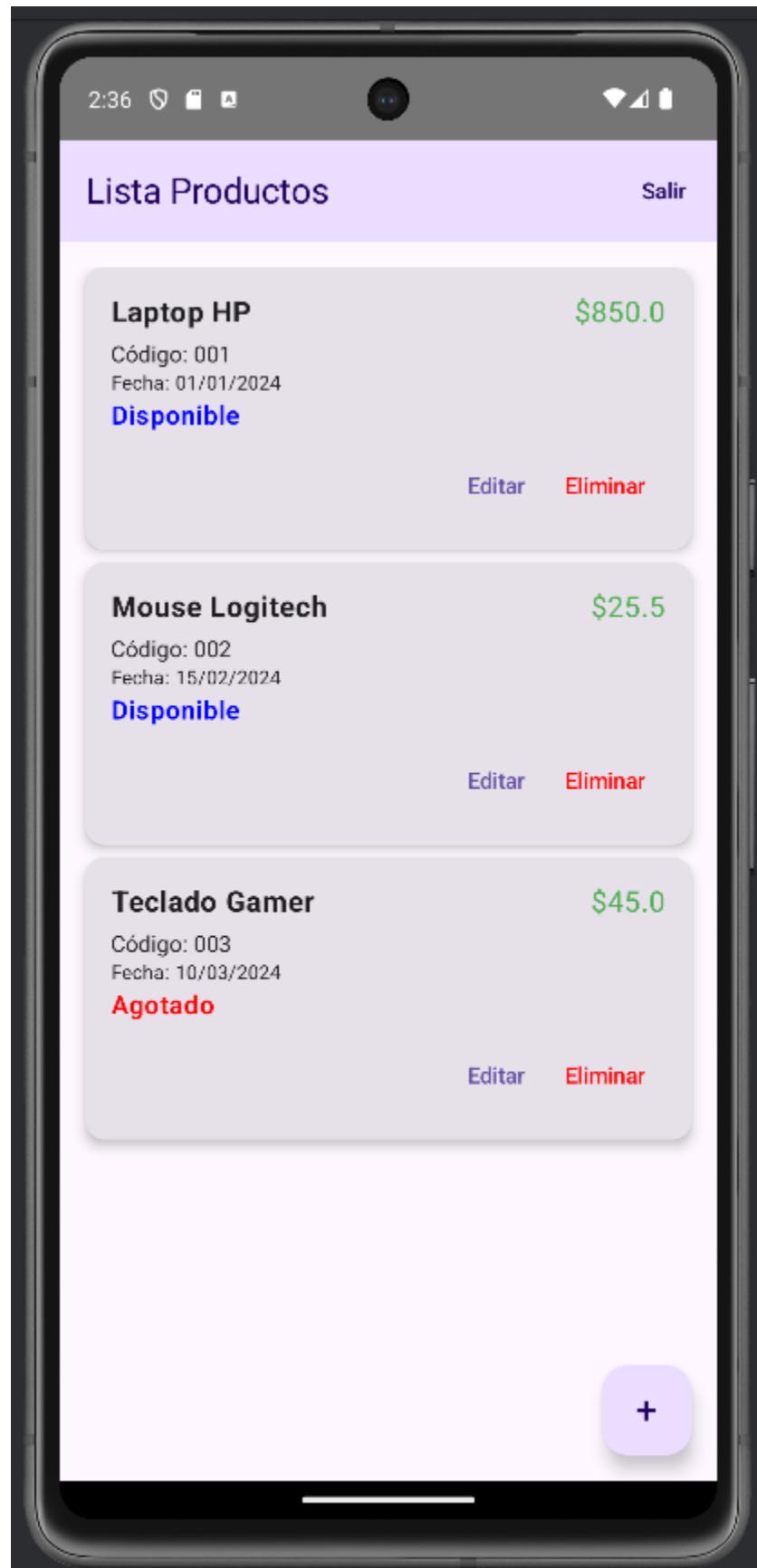
Componentes UI:

@Composable

```
fun HomeScreen(  
    onLogout: () -> Unit,  
    onAddProduct: () -> Unit,  
    onEditProduct: (Producto) -> Unit  
)
```

Estructura: - TopAppBar: - Título: “Lista Productos” - Botón “Salir” (TextButton)

- **FloatingActionButton:**
 - Ícono “+” para agregar producto
- **LazyColumn:**
 - Lista desplazable de ProductoCard
 - Espaciado de 8dp entre elementos



ProductoCard.kt (Componente)

Tarjeta individual de producto

```
@Composable
```

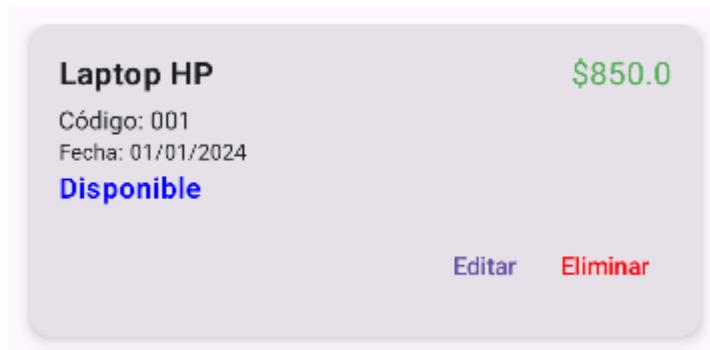
```
fun ProductoCard(  
    producto: Producto,  
    onEdit: () -> Unit,  
    onDelete: () -> Unit  
)
```

Elementos Visuales: - **Encabezado:** Descripción (bold) | Precio (verde) -

Detalles: - Código del producto - Fecha de fabricación - Estado: “Disponible” (azul) o “Agotado” (rojo) - **Acciones:** - Botón “Editar” - Botón “Eliminar” (rojo)

Eliminación: - Click en “Eliminar” → Controlador.eliminarProducto(producto) -

Actualización inmediata (reactividad)



ProductoFormScreen.kt

Formulario para agregar/editar productos

Funcionalidad:

- **Modo Creación:** Campos vacíos, código editable
- **Modo Edición:** Campos prellenados, código bloqueado

Componentes UI:

```
@Composable  
fun ProductoFormScreen(  
    productoAEellar: Producto?,  
    onSaveSuccess: () -> Unit,  
    onCancel: () -> Unit  
)
```

Campos del Formulario:

1. Código (OutlinedTextField)

- Tipo: Numérico (solo dígitos)
- Validación: Solo números
- Estado: Deshabilitado en edición

2. Descripción (OutlinedTextField)

- Tipo: Texto con capitalización automática
- Requerido

3. Precio (OutlinedTextField)

- Tipo: Numérico decimal
- Formato: Double

4. Fecha de Fabricación (DatePickerField)

- Componente personalizado
- Selector visual de calendario
- Formato: dd/MM/yyyy

5. Disponibilidad (Switch)

- Toggle on/off
- Default: true (disponible)

Botones: - “GUARDAR” → Valida y guarda/edita - “CANCELAR” → Descarta cambios y regresa





Validación y Guardado:

```
val precio = precioStr.toDoubleOrNull()
if (codigo.isNotEmpty() && descripcion.isNotEmpty() &&
    precio != null && fecha.isNotEmpty()) {

    val nuevoProd = Producto(codigo, descripcion, precio, fecha, disponible)

    if (esEdicion) {
        Controlador.editarProducto(nuevoProd)
        Toast.makeText(context, "Producto editado", Toast.LENGTH_SHORT).show()
    } else {
        Controlador.agregarProducto(nuevoProd)
        Toast.makeText(context, "Producto creado", Toast.LENGTH_SHORT).show()
    }
    onSaveSuccess()
} else {
    Toast.makeText(context, "Revise los datos", Toast.LENGTH_SHORT).show()
}
```

Validaciones: - Código no vacío - Descripción no vacía - Precio válido (número decimal) - Fecha no vacía

DatePicker.kt

Componente personalizado para selección de fechas

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DatePickerField(
    label: String,
    selectedDate: String,
```

```
onDateSelected: (String) -> Unit  
)
```

Funcionalidad:

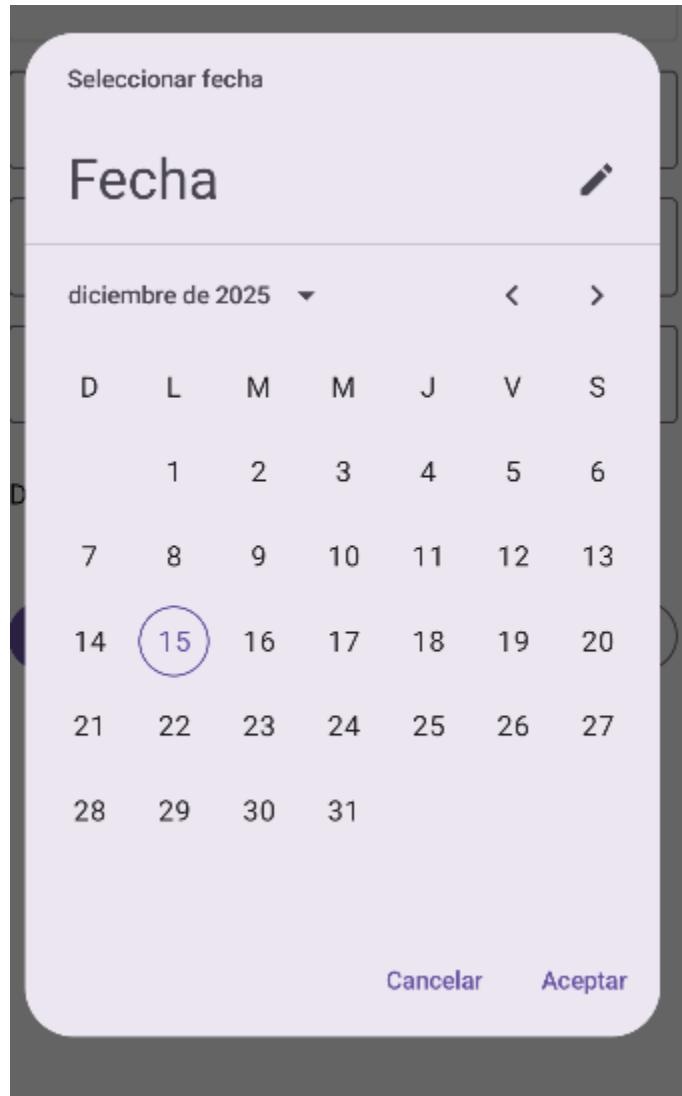
- Campo de texto con ícono de calendario
- Click en ícono → Abre diálogo de fecha
- Formato de salida: dd/MM/yyyy
- Zona horaria: UTC (evita problemas de conversión)

Componentes:

- OutlinedTextField (editable manualmente)
- IconButton con Icons.Filled.DateRange
- DatePickerDialog con calendario visual
- Botones “Aceptar” / “Cancelar”

Conversión de Fecha:

```
val sdf = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault())  
sdf.timeZone = TimeZone.getTimeZone("UTC")  
val date = Date(datePickerState.selectedDateMillis ?: System.currentTimeMillis())  
onDateSelected(sdf.format(date))
```



Logo.kt

Componentes de marca de la aplicación

AppLogo()

- Logo circular con imagen de Android
- Fondo azul (#2196F3)
- Tamaño: 120dp
- Ícono interno: 80dp

AppName()

- Texto “ProducTrack” (32sp, bold)
 - Subtítulo “Tu inventario bajo control” (16sp, gris)
 - Alineación centrada
-

4 NAVEGACIÓN

MainActivity.kt

Punto de entrada y gestor de navegación

```
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            MaterialTheme {  
                AppNavegacion()  
            }  
        }  
    }  
}
```

AppNavegacion() - Sistema de Rutas

@Composable

```
fun AppNavegacion() {  
    var pantallaActual by remember { mutableStateOf("login") }  
    var productoSeleccionado by remember { mutableStateOf<Producto?>(null) }  
  
    when (pantallaActual) {  
        "login" -> LoginScreen(...)  
        "registro" -> RegistroScreen(...)  
        "home" -> HomeScreen(...)  
    }  
}
```

```

    "formulario" -> ProductoFormScreen(...)
}

}

```

Estados de Navegación:

- pantallaActual: String que controla qué pantalla mostrar
- productoSeleccionado: Producto a editar (null = crear nuevo)

Rutas Disponibles:

Ruta	Pantalla	Acceso desde
"login"	LoginScreen	Inicio, Logout, Registro exitoso
"registro"	RegistroScreen	Link en Login
"home"	HomeScreen	Login exitoso, Guardar producto
"formulario"	ProductoFormScreen	Botón +, Editar producto

Navegación Manual (sin Navigation Compose):

- Uso de callbacks (onLoginSuccess, onNavigateToRegister, etc.)
 - Cambio de estado reactivo con remember + mutableStateOf
 - Recomposición automática al cambiar pantallaActual
-

⌚ Tema y Estilos

Color.kt

```

val Purple80 = Color(0xFFD0BCFF)
val Purple40 = Color(0xFF6650a4)
// ... paleta Material 3

```

Colores en Recursos (colors.xml)

```
<color name="ic_launcher_background_blue">#2196F3</color>
```

Tema Material Design 3

- Uso de MaterialTheme genérico

- Componentes: Material 3 (Compose BOM)
 - Iconos extendidos para DateRange
-

AndroidManifest.xml

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/Theme.Tarea07U2G2">

    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:theme="@style/Theme.Tarea07U2G2">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Configuración: - Actividad principal: MainActivity - Exportada para launcher - Tema Material personalizado - Backup habilitado

Dependencias (build.gradle.kts)

Plugins:

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
```

```
    alias(libs.plugins.kotlin.compose)
}
```

Configuración Android:

```
android {
    namespace = "com.example.tarea07_u2_g2"
    compileSdk = 36
    minSdk = 24
    targetSdk = 36
    buildFeatures { compose = true }
}
```

Dependencias Principales:

```
implementation(libs.androidx.core.ktx)
implementation(libs.androidx.lifecycle.runtime.ktx)
implementation(libs.androidx.activity.compose)
implementation(platform(libs.androidx.compose.bom))
implementation(libs.androidx.compose.ui)
implementation(libs.androidx.compose.material3)
implementation(libs.androidx.navigation.compose)
implementation("androidx.compose.material:material-icons-extended:1.6.8")
```

Tecnologías Clave: - **Jetpack Compose:** UI declarativa moderna - **Material 3:**

Componentes de diseño actualizados - **Kotlin Coroutines:** Para operaciones asíncronas (lifecycle) - **Material Icons Extended:** Iconos adicionales (DateRange)

Flujo de Uso de la Aplicación

1. Inicio de la App

```
MainActivity.onCreate()
    → MaterialTheme
    → AppNavegacion()
    → LoginScreen
```

2. Flujo de Login

LoginScreen

- └─ Usuario nuevo → RegistroScreen → Registro exitoso → LoginScreen
- └─ Credenciales válidas → HomeScreen

3. Gestión de Productos

HomeScreen

- └─ Botón "+" → ProductoFormScreen (modo crear) → Guardar → HomeScreen
- └─ "Editar" → ProductoFormScreen (modo editar) → Guardar → HomeScreen
- └─ "Eliminar" → Confirmación Toast → Actualización lista
- └─ "Salir" → LoginScreen

4. Formulario de Producto

ProductoFormScreen

- └─ Rellenar campos → "GUARDAR"
 - └─ Validación exitosa → Controlador.agregar/editar → Toast → HomeScreen
 - └─ Validación fallida → Toast de error
 - └─ "CANCELAR" → HomeScreen (sin guardar)
-

Datos de Prueba

Usuarios Predefinidos:

```
Usuario("Edison", "Enriquez", "1234")
Usuario("Kelly", "Ledesma", "1234")
Usuario("Stalin", "Acurio", "1234")
Usuario("Ariel", "Elizalde", "1234")
Usuario("Mauricio", "Lopez", "1234")
Usuario("Angelo", "Silva", "1234")
Usuario("Alexis", "Troya", "1234")
```

Productos Predefinidos:

```
Producto("001", "Laptop HP", 850.00, "01/01/2024", true)  
Producto("002", "Mouse Logitech", 25.50, "15/02/2024", true)  
Producto("003", "Teclado Gamer", 45.00, "10/03/2024", false)
```

Validaciones Implementadas

Login:

- Nombre no vacío
- Apellido no vacío
- Contraseña no vacía
- Coincidencia con usuarios registrados
- Ignore case en nombre y apellido

Registro:

- Todos los campos requeridos
- Sin validación de duplicados (simplificación)

Productos:

- Código numérico y no vacío
- Descripción no vacía
- Precio numérico válido (Double)
- Fecha no vacía
- Código único (visual, no forzado en backend)

Teclado Virtual:

- IME Action (Next/Done) en todos los campos
- Navegación automática entre campos

- Submit con Enter en último campo
-

⌚ Casos de Uso Detallados

Caso 1: Usuario Nuevo se Registra

1. Abre la app → LoginScreen
2. Click “¿Nuevo usuario? Regístrate aquí”
3. Completa formulario (Nombre, Apellido, Contraseña)
4. Click “GUARDAR Y VOLVER”
5. Validación exitosa → Usuario agregado al Controlador
6. Regresa a Login → Puede iniciar sesión

Caso 2: Login con Credenciales Existentes

1. Ingresa “Edison”, “Enriquez”, “1234”
2. Click “INGRESAR”
3. Validación exitosa → Toast “¡Bienvenido Edison!”
4. Navega a HomeScreen con lista de productos

Caso 3: Agregar Producto Nuevo

1. En HomeScreen → Click botón “+”
2. ProductoFormScreen en modo creación
3. Completa:
 - Código: 004
 - Descripción: Monitor Samsung
 - Precio: 320.50
 - Fecha: 10/12/2024 (desde DatePicker)
 - Disponible: ON
4. Click “GUARDAR”
5. Validación exitosa → Controlador.agregarProducto()

6. Toast “Producto creado”
7. Regresa a HomeScreen → Nuevo producto visible

Caso 4: Editar Producto Existente

1. En lista, selecciona “Mouse Logitech”
2. Click “Editar”
3. ProductoFormScreen en modo edición
4. Código bloqueado: “002”
5. Modifica precio: 30.00 → 28.99
6. Click “GUARDAR”
7. Controlador.editarProducto() reemplaza datos
8. Toast “Producto editado”
9. Lista actualizada con nuevo precio

Caso 5: Eliminar Producto

1. En lista, selecciona “Teclado Gamer”
2. Click “Eliminar”
3. Controlador.eliminarProducto() lo remueve
4. Recomposición automática → Ya no aparece en lista

Caso 6: Cerrar Sesión

1. En HomeScreen → Click “Salir”
 2. Regresa a LoginScreen
 3. Datos permanecen en memoria (usuarios y productos)
-

Características Técnicas Destacadas

Reactividad con Compose:

- mutableStateListOf en Controlador
- Cambios instantáneos en UI sin notifyDataSetChanged()

- Recomposición automática de LazyColumn

Gestión de Estado:

- remember + mutableStateOf para variables locales
- Estado compartido a través de callbacks
- Sin ViewModel (simplificación MVC básico)

Material Design 3:

- OutlinedTextField con labels flotantes
- TopAppBar con Material You
- FloatingActionButton con elevación
- Card con elevación y sombras
- Switch moderno

Experiencia de Usuario:

- Toast informativos en todas las acciones
 - Navegación del teclado con IME Actions
 - DatePicker visual (no escritura manual obligatoria)
 - Campos con placeholder y autocompletado
 - Contraseñas ocultas con asteriscos
-

Estadísticas del Proyecto

Archivos de Código:

- **Kotlin:** 13 archivos
- **XML:** 10 archivos (recursos y manifest)
- **Total líneas de código:** ~1200+ líneas

Pantallas Implementadas:

- LoginScreen

- RegistroScreen
- HomeScreen
- ProductoFormScreen

Componentes Reutilizables:

- AppLogo
 - AppName
 - DatePickerField
 - ProductoCard
-

⚠ Limitaciones Conocidas

Almacenamiento:

- ✗ Sin persistencia (datos se pierden al cerrar app) los datos se guardan en memoria
- ✗ No usa Room Database ni Shared Preferences
- ✓ Solución: Implementar Room o DataStore

Seguridad:

- ✗ Contraseñas sin cifrar
- ✗ Sin validación de fortaleza de contraseña
- ✓ Solución: Hash con BCrypt o similar

Validaciones:

- ✗ No verifica códigos duplicados al crear
- ✗ No limita longitud de campos
- ✓ Solución: Agregar validaciones en Controlador

Navegación:

- No usa Navigation Compose oficial
 - Sin back stack management
 - Solución: Migrar a NavController
-

Recursos y Referencias

Documentación Oficial:

- Jetpack Compose
- Material Design 3
- Kotlin

Bibliotecas Utilizadas:

- androidx.compose.material3 - Componentes UI
 - androidx.navigation.compose - (Declarada pero no usada)
 - material-icons-extended - Iconos DateRange
-

Licencia

Proyecto académico - Universidad (Tarea 07 Unidad 2)

Notas Finales

Este proyecto demuestra una implementación sólida de los conceptos fundamentales de desarrollo Android moderno:

- Arquitectura MVC bien estructurada
- UI declarativa con Jetpack Compose

- Material Design 3 consistente
- Gestión de estado reactiva
- Validaciones funcionales
- Código limpio y organizado

Es una base excelente que puede evolucionar hacia una aplicación profesional con las mejoras sugeridas.

Fecha de Documentación: 14 de diciembre de 2024

Versión del Documento: 1.0

Estado del Proyecto: Funcional y Completo