

QHA_PMDM - (API Rest)

Proyecto desarrollado para la asignatura de **Programación Multimedia y Dispositivos Móviles (PMDM)**. La aplicación consume datos de una API pública de productos y los muestra en una interfaz moderna utilizando tecnologías de vanguardia en Android.

💡 Características

- **Consumo de API REST:** Petición única al endpoint de productos.
- **Arquitectura MVVM:** Separación clara entre datos, lógica de negocio y vista.
- **Jetpack Compose:** Interfaz 100% declarativa y reactiva.
- **Diseño Dinámico:** Cards con colores personalizados según la categoría del producto.
- **Gestión de Imágenes:** Carga asíncrona y eficiente mediante Coil.

🛠 Stack Tecnológico

- **Lenguaje:** [Kotlin](#)
- **UI:** [Jetpack Compose](#)
- **Red:** [Retrofit](#)
- **JSON Parsing:** [Moshi](#) (con Kotlin Json Adapter)
- **Carga de Imágenes:** [Coil](#)
- **Concurrencia:** Kotlin Coroutines & Flow

📁 Estructura del Proyecto

El proyecto sigue la estructura de **Clean Architecture** sugerida:

```
com.example.api_pmdm_t02/
  └── data/
      └── model/
          ├── ProductoDto.kt      # Clase de datos para el objeto Producto
          └── PagedResponse.kt    # Clase envoltorio para la respuesta de la API
  (results)
  └── remote/
      ├── ProductosApi.kt     # Interfaz de Retrofit con los endpoints (@GET)
      └── ProductosNetwork.kt # Cliente Retrofit configurado con Moshi
  └── repository/
      └── ProductosRepository.kt # Mediador entre la API y el ViewModel
  └── presentation/
      └── viewmodel/
          ├── ProductosViewModel.kt # Gestión del estado de la UI y llamadas al
          │   repository
          └── UiState.kt           # Sealed interface para estados: Loading,
          Success y Error
```

```

ui/
  └── screens/
    └── ProductosScreen.kt      # UI principal y diseño de la Card (Composables)

  └── theme/
    ├── Color.kt                # Definición de la paleta de colores del proyecto
    ├── Theme.kt                 # Configuración del tema Material 3 de la app
    └── Type.kt                  # Configuración de tipografías

  └── MainActivity.kt          # Punto de entrada: Configura EdgeToEdge y llama
a la Screen

```

🔗 API Reference

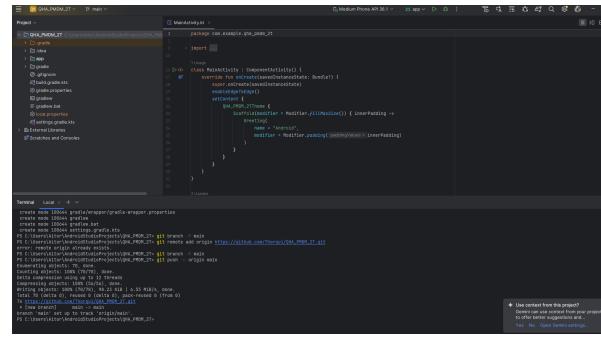
La aplicación consume el siguiente endpoint para obtener el listado de productos:

GET <https://peticiones.online/api/products>

Nota importante: Aunque la API devuelve metadatos de paginación (`page`, `per_page`, `total`, `total_pages`), este proyecto cumple estrictamente con el requisito de realizar una **única petición simple** al endpoint indicado, trabajando exclusivamente con el array de objetos contenido en el campo `results`.

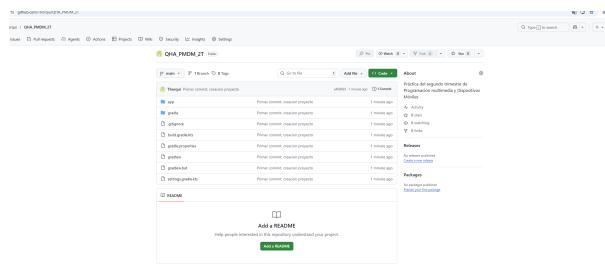
📸 Capturas de Pantalla

📸 Vista General de la Aplicación



Creacion del proyecto

📸 Detalle de Producto



Generacion archivos



Generacion archivos



Generacion archivos



```
package com.example.java_20.data.repositories;

import com.example.java_20.data.model.ResponseBody;
import com.example.java_20.data.model.ResponseStatus;
import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Query;

public interface ProductRepository {
    @GET("products")
    Call<ResponseBody> getProducts(@Query("page") int page);
}

interface ResponseBody {
    ResponseStatus getStatus();
    String getMessage();
}

class ResponseStatus {
    private String status;
    private String message;
}
```

Generacion archivos



Generacion archivos



Proyecto funcionando



- **Aitor Quilez Herrero** - Desarrollo de Software Android -
[https://github.com/Thorqui/QHA_PMDM_2T.git]