✅ 1. item.xml (layout de cada fila)

`res/layout/item.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tvNombre"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp"
    android:textSize="20sp"/>
```

Este es el “molde” que se repetirá por cada nombre.

✅ 2. fragment_fragmento1.xml (el layout del fragmento)

`res/layout/fragment_fragmento1.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recicler"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

✅ 3. Adaptador completo

`NombresAdapter.kt`

```
package com.dam.ejercicio_practica

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class NombresAdapter(
    private val nombres: List<String>
```

```
) : RecyclerView.Adapter<NombresAdapter.NombreViewHolder>() {

    // ViewHolder
    class NombreViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val tvNombre: TextView = itemView.findViewById(R.id.tvNombre)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
NombreViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.item, parent, false)
        return NombreViewHolder(view)
    }

    override fun onBindViewHolder(holder: NombreViewHolder, position: Int) {
        val nombre = nombres[position]
        holder.tvNombre.text = nombre
    }

    override fun getItemCount(): Int = nombres.size
}
```

🔍 Este adapter:

- recibe una lista de Strings
- infla `item.xml`
- pone cada nombre en el TextView

✅ 4. El fragmento completo: fragmento1

`fragmento1.kt`

```
package com.dam.ejercicio_practica

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class fragmento1 : Fragment() {

    // Lista inventada
    private val listaNombres = listOf(
        "lucia", "alvaro", "jorge", "milena",
        "dani", "tomas", "hugo", "jose juan"
    )

    private lateinit var recyclerView: RecyclerView
    private lateinit var adapter: NombresAdapter

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
```

```
    ): View? {

        // inflamos el layout
        val view = inflater.inflate(R.layout.fragment_fragmento1, container,
false)

        // pillamos el RecyclerView
        recyclerView = view.findViewById(R.id.recicler)

        // la lista va de arriba hacia abajo
        recyclerView.layoutManager = LinearLayoutManager(requireContext())

        // creamos el adapter con la lista de nombres
        adapter = NombresAdapter(listaNombres)

        // lo conectamos
        recyclerView.adapter = adapter

        return view
    }
}
```

🧠 RESUMEN MENTAL EXPRESS

Para que el RecyclerView funcione necesitas:

1. **item.xml** → diseño de cada fila
2. **fragment_fragmento1.xml** → layout con el RecyclerView
3. **NombresAdapter** → conecta cada nombre con item.xml
4. **fragmento1** → infla el layout, configura RecyclerView y le mete el adapter

Nada más.

TABLAYOUT

🔹 Paso 1: Asegúrate de tener el ViewPager2 y el TabLayout en el layout de la Activity

En tu `activity_main.xml` (o como se llame) debería haber algo así:

```
<com.google.android.material.tabs.TabLayout
    android:id="@+id/tabLayout"
    ... />

<androidx.viewpager2.widget.ViewPager2
    android:id="@+id/viewPager"
    ... />
```

IDs importantes:

- `R.id.tabLayout`

- `R.id.viewPager`

🔹 Paso 2: En tu MainActivity, consigue las referencias

En `onCreate` de tu `MainActivity`:

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val viewPager: ViewPager2 = findViewById(R.id.viewPager)
        val tabLayout: TabLayout = findViewById(R.id.tabLayout)
    }
}
```

🔹 Paso 3: Conecta el adapter al ViewPager2

Antes de usar el `TabLayoutMediator`, tu `ViewPager2` tiene que tener un adapter, por ejemplo `MiPagerAdapter`:

```
val adapter = MiPagerAdapter(this)
viewPager.adapter = adapter
```

`MiPagerAdapter` es el que devuelve `FragmentInfo` y `FragmentRutinas` según la posición.

🔹 Paso 4: Crear el TabLayoutMediator

Aquí viene **el método** que manda:

```
TabLayoutMediator(tabLayout, viewPager) { tab, position ->
    when (position) {
        0 -> tab.text = "INFO"
        1 -> tab.text = "RUTINAS"
    }
}.attach()
```

Qué está pasando aquí:

- `TabLayoutMediator(tabLayout, viewPager) { tab, position -> ... }`
    - Crea el enlace entre las pestañas y las páginas del ViewPager.
- Dentro del bloque (`{ tab, position -> ... }`) decides:
    - Qué texto (o icono) tiene cada pestaña según la `position`.
- `.attach()`
    - **Importantísimo**: sin esto no hace nada. Es lo que “activa” el mediador.

🔹 Paso 5: Import necesario

Asegúrate de tener el import del `TabLayoutMediator`:

```
import com.google.android.material.tabs.TabLayoutMediator
```

🔹 Resumen rápido en contexto

Tu `onCreate` quedaría algo así:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    val viewPager: ViewPager2 = findViewById(R.id.viewPager)
    val tabLayout: TabLayout = findViewById(R.id.tabLayout)

    val adapter = MiPagerAdapter(this)
    viewPager.adapter = adapter

    TabLayoutMediator(tabLayout, viewPager) { tab, position ->
        when (position) {
            0 -> tab.text = "INFO"
            1 -> tab.text = "RUTINAS"
        }
    }.attach()
}
```

1️⃣ Crear la carpeta `layout-land`

1. En Android Studio, en el panel de `res`:
   - Abre: `app > src > main > res`.
2. Haz clic derecho sobre la carpeta `layout`
   → `New > Android Resource Directory`.
3. En **Resource type** elige: `layout`.
4. En **Available qualifiers** pulsa `Orientation` y dale a `>>`.
5. En **Orientation** elige `Landscape`.
6. Nombre del directorio te quedará: `layout-land`. Acepta.

Ahora tienes dos carpetas:

- `res/layout/`
- `res/layout-land/`

2️⃣ Crear el layout landscape de `activity_main`

1. Clic derecho en `res/layout-land`
   → `New > Layout Resource File`.
2. **File name**: `activity_main` (¡Mismo nombre que el de siempre!).
3. Root element: el mismo que uses en portrait (`ConstraintLayout`, `LinearLayout`, lo que tengas).
4. Acepta.

Android ahora tiene:

- `res/layout/activity_main.xml` → para vertical
- `res/layout-land/activity_main.xml` → para horizontal

Tu `MainActivity` seguirá usando `setContentView(R.layout.activity_main)` sin tocar nada de Kotlin. El sistema elige solo el layout según la orientación.

3️⃣ Copiar el contenido del layout portrait

1. Abre `res/layout/activity_main.xml` (el vertical).
2. Copia TODO el XML.
3. Pégalo dentro de `res/layout-land/activity_main.xml`.

Ahora en landscape tienes **la misma UI**, pero puedes recolocarla a tu gusto.

Importante:
👉 **No cambies los `id`** de las vistas (`toolbar`, `img_receta`, `spinner`, `tabLayout`, `ViewPager2`, etc.).
Porque tu `MainActivity` los busca por ID. Solo cambia posiciones, constraints, pesos, etc.

4️⃣ Recolocar elementos para que quede como la captura

La idea de landscape es:

- Izquierda: la **card** grande con la imagen y los datos de la receta.
- Derecha: el **spinner**, el **TabLayout** y debajo el **ViewPager2** con ingredientes/pasos.

INTERNACIONALIZAR

1️⃣ Crear los strings en `strings.xml` (español)

Abre `res/values/strings.xml` y añade (si no los tienes ya):

```
<string name="minutos">minutos</string>
<string name="raciones">raciones</string>
<string name="ingredientes">Ingredientes</string>
<string name="pasos">Pasos</string>
```

Aquí defines la versión en **español**, que es el idioma por defecto.

2️⃣ Usar esos strings en los layouts (quitar hardcodeo)

Busca en tus XML cosas como:

```
android:text="minutos"
android:text="raciones"
android:text="Ingredientes"
android:text="Pasos"
```

y cámbialas por:

```
android:text="@string/minutos"
android:text="@string/raciones"
android:text="@string/ingredientes"
android:text="@string/pasos"
```

Hazlo en **todos** los layouts donde aparezcan (portrait y landscape).

3️⃣ Crear la carpeta de inglés `values-en`

1. Clic derecho en `res` → `New > Android Resource Directory`
2. `Resource type`: **values**
3. En “Available qualifiers” elige **Locale** → botón `>>`
4. Escoge **English** (`en`)
5. Se crea la carpeta: `res/values-en`

4️⃣ Crear `strings.xml` en inglés

Dentro de `res/values-en`, crea/abre `strings.xml` y mete las mismas keys pero traducidas:

```
<string name="minutos">minutes</string>
<string name="raciones">servings</string>
<string name="ingredientes">Ingredients</string>
<string name="pasos">Steps</string>
```

Las claves (`name="..."`) deben ser EXACTAMENTE las mismas que en español.

---

5 Probar que funciona

- En el emulador, cambia el idioma del sistema a **English**.
- Vuelve a abrir la app.
- Donde antes ponía:
    - minutos → ahora pondrá **minutes**
    - raciones → **servings**
    - ingredientes → **Ingredients**
    - pasos → **Steps**