RecyclerView

1TEL05 - Servicios y Aplicaciones para IoT

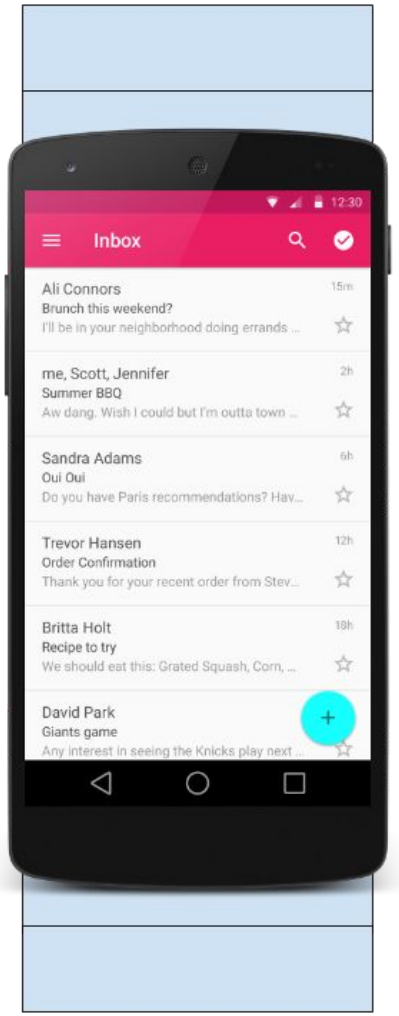
Clase 5.1

Prof. Oscar Díaz

¿Qué es un RecyclerView?

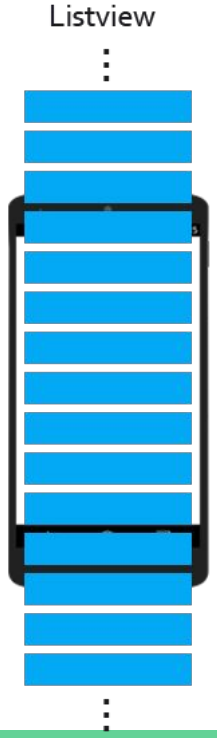
- Es un contenedor scrollable de una larga cantidad de elementos.
- Permite ocultar de la vista y de los recursos de Android los elementos que no puede ver el usuario directamente.
- Es eficiente:
  - Reutiliza una sola vista para mostrar todos los elementos
  - Actualiza los cambios de forma rápida

➜ https://developer.android.com/guide/topics/ui/layout/recyclerview

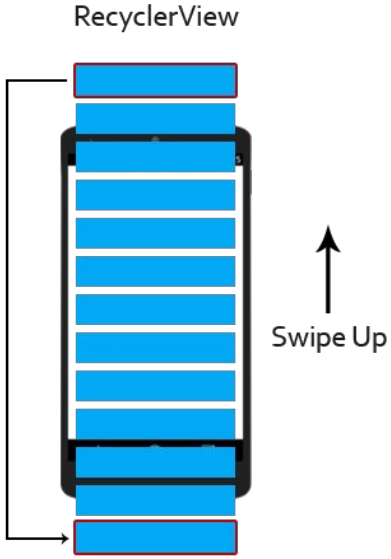


RecyclerView

Por ejemplo, si compara un elemento **Listview** con muchos items, todos se cargan y están presente en la pantalla, aún si no se muestran, ocupando espacio para ser mostrados.

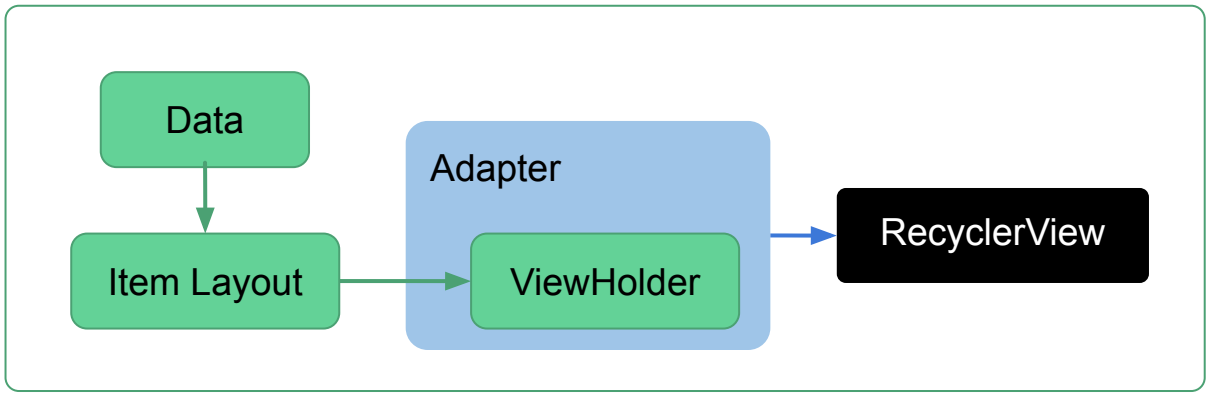


Por otro lado, **RecyclerView** mantiene todos los elementos cargados pero solo muestra en pantalla los que entren en el dispositivo y los inmediatamente arriba y abajo.



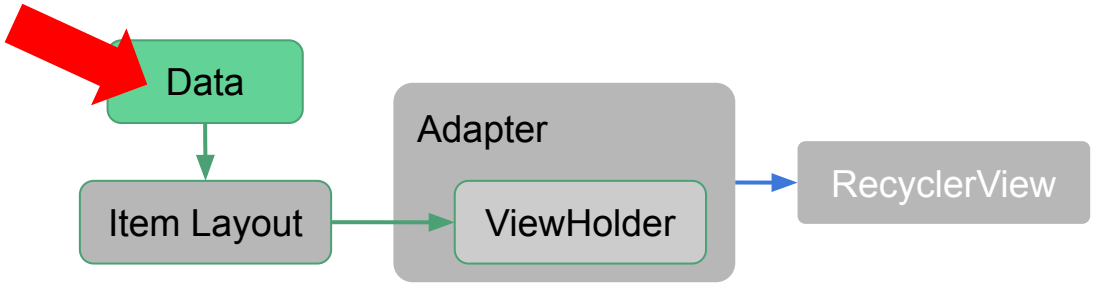
Componentes de un RecyclerView

- **Data**: información a mostrar
- **Item Layout**: Diseño que un elemento de la lista (como un layout separado)
- **Adapter**: conecta la **Data** con el **RecyclerView**. Debe ser una clase que herede de RecyclerView.Adapter.
- **ViewHolder**: Contenedor de la **data** a ser usado por el **Adapter**. Debe ser una clase que herede de RecyclerView.ViewHolder.
- **RecyclerView**: Elemento que contendrá la lista



Implementar un RecyclerView

Definir la data



En este ejemplo, la **data** será obtenida desde un webservice, el cual muestra una lista de 107 empleados:

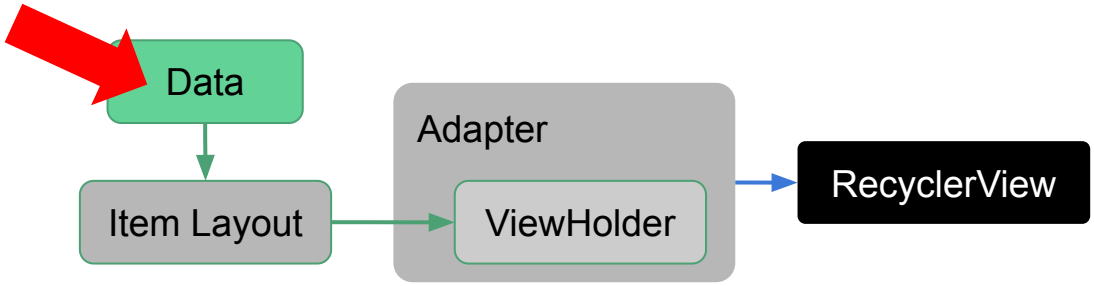
- El **endpoint** es: https://3dkvh9b90.execute-api.us-east-1.amazonaws.com/prod/
- **Método**: GET
- **Headers**: api-key : EaQiblyUgcoCAyelLnDwUAxR1OX6AH

Si el webservice no funciona, utilice la aplicación:

https://github.com/2022-2-1TEL05-Servicios-y-Apps-IoT/clase6ws.git


```
            "id": 205,
            "firstName": "Shelley",
            "lastName": "Higgins",
            "email": "SHIGGINS",
            "phoneNumber": "515.123.8080",
            "hireDate": "1994-06-07T05:00:00Z",
            "jobId": "AC_MGR",
            "salary": 12000.00,
            "commissionPct": null,
            "managerId": 101,
            "departmentId": 110
        },
        {
            "id": 206,
            "firstName": "William",
            "lastName": "Gietz",
            "email": "WGIETZ",
            "phoneNumber": "515.123.8181",
            "hireDate": "1994-06-07T05:00:00Z",
            "jobId": "AC_ACCOUNT",
            "salary": 8300.00,
            "commissionPct": null,
            "managerId": 205,
            "departmentId": 110
        }
    ],
    "estado": "ok"
}
```

Clases necesarias



Para gestionar la respuesta desde el webservice y mapear los resultados a objetos en java, se deben crear dos clases.

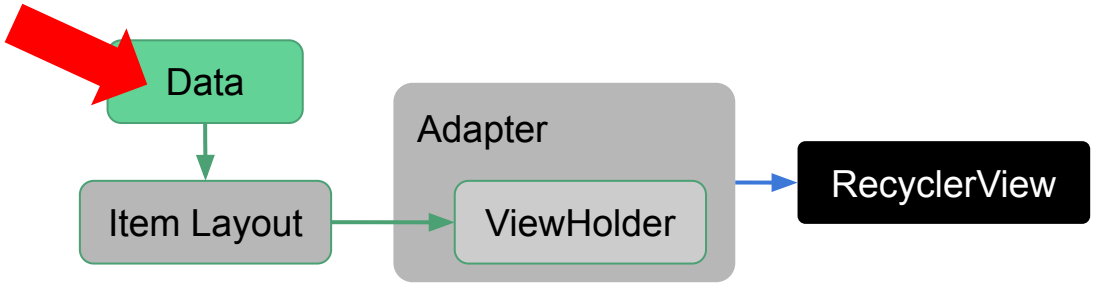
Puede utilizar: https://www.jsonschema2pojo.org/

```
public class Employee implements Serializabl

    private String id;
    private String firstName;
    private String lastName;
    private String email;
    private String phoneNumber;
    private String hireDate;
    private String jobId;
```

```
public class EmpleadoDto {

    private Empleado[] lista;
    private String estado;
```

```
{
    "id": "205_AC",
    "firstName": "Shelley",
    "lastName": "Higgins",
    "email": "SHIGGINS",
    "phoneNumber": "515.123.8080",
    "hireDate": "1994-06-07T05:00:00Z",
    "jobId": "AC_MGR",
    "salary": 12000.00,
    "commissionPct": null,
    "managerId": "101_EX",
    "departmentId": 110
},
{
    "id": "206_AC",
    "firstName": "William",
    "lastName": "Gietz",
    "email": "WGIETZ",
    "phoneNumber": "515.123.8181",
    "hireDate": "1994-06-07T05:00:00Z",
    "jobId": "AC_ACCOUNT",
    "salary": 8300.00,
    "commissionPct": null,
    "managerId": "205_AC",
    "departmentId": 110
}
],
"estado": "ok"
}
```

Obtención de la data

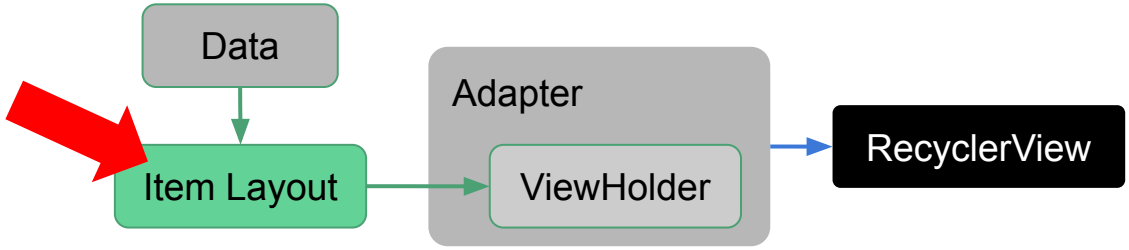


Usando Retrofit se puede obtener la data:

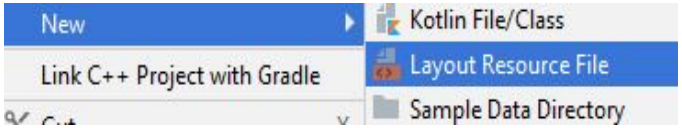
```
public void obtenerData() {
    EmployeesRepo employeesRepo = new Retrofit.Builder()
            .baseUrl("http://10.0.2.2:8080")
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(EmployeesRepo.class);

    employeesRepo.listEmployees().enqueue(new Callback<EmpleadoDto>() {
        @Override
        public void onResponse(@NonNull Call<EmpleadoDto> call, Response<EmpleadoDto>
response) {
            if (response.isSuccessful()) {
                EmpleadoDto empleadoDto = response.body();
                Log.d("msg-test", empleadoDto.getEstado());

            } else {
                Log.d("msg-test", "error en la respuesta del webservice");
            }
        }

        @Override
        public void onFailure(Call<EmpleadoDto> call, Throwable t) {
            t.printStackTrace();
        }
    });
}
```

En las siguientes diapositivas se completa

Crear el Item Layout

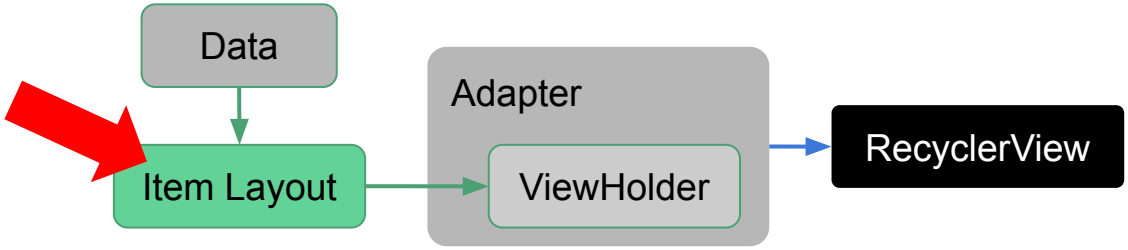


El elemento que usará el RecyclerView para llenar su lista debe ser creado como un elemento aparte dentro de la carpeta layout.

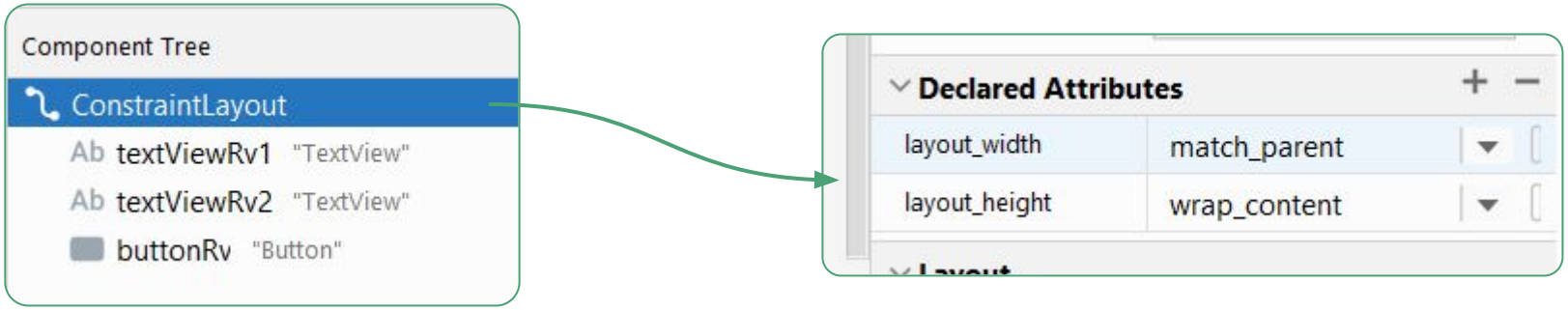


Para el ejemplo se ha creado uno de nombre **`item_rv.xml`**

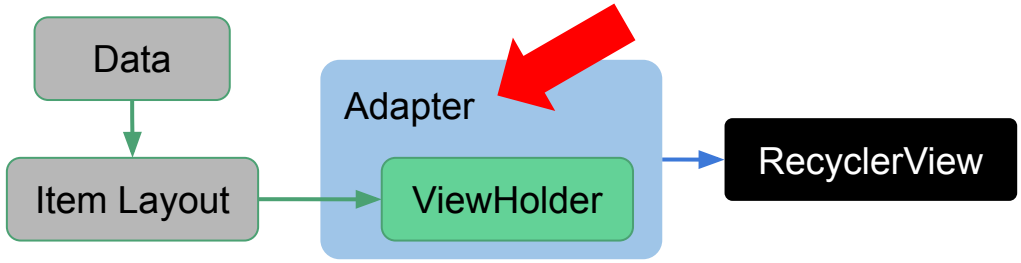
Crear el Item Layout



En este layout, es donde se define el elemento.



Crear el Adapter

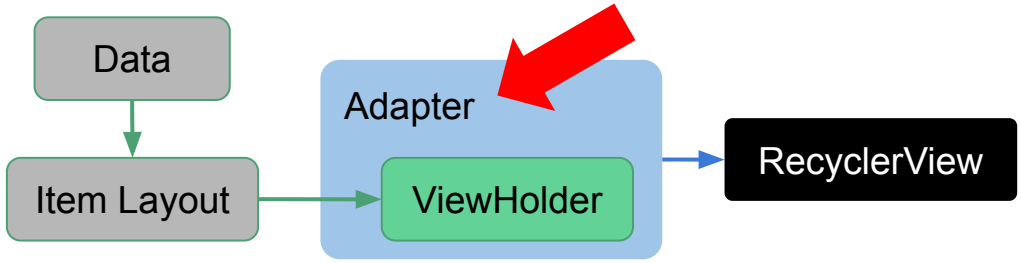


Se crea una clase que contendrá el Adapter que llamaremos:
ListaEmpleadosAdapter, con 2 variables:

- listaEmpleados ➜ para mantener el estado de la lista de elementos
- context ➜ para mantener el Contexto del activity que gestiona la aplicación.

```
public class ListaEmpleadosAdapter {

    private List<Empleado> listaEmpleados;
    private Context context;
```

ViewHolder

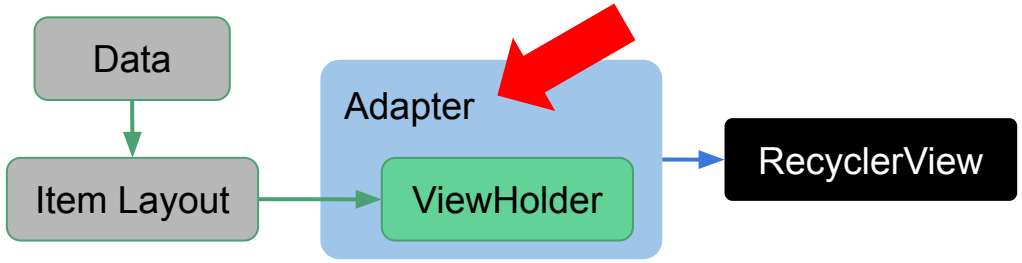


Para gestionar cada elemento desde el adapter, se debe utilizar la clase **ViewHolder** como una clase interna. Aquí se definen los objetos cuyos estados se desean conversar para cada **ViewHolder**.

Debe definir la clase y su constructor con parámetros. Si quiere gestionar un botón, aquí se realiza.

```
public class ListaEmpleadosAdapter {

    private List<Empleado> listaEmpleados;
    private Context context;

    public class EmpleadoViewHolder extends RecyclerView.ViewHolder{

        Empleado empleado;

        public EmpleadoViewHolder(@NonNull View itemView) {
            super(itemView);
        }
    }
```

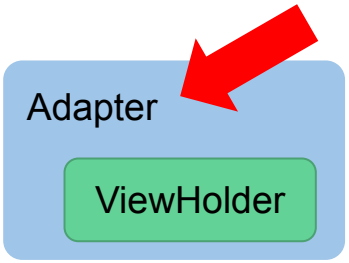
Completar la creación del Adapter



- Se hereda de la clase **RecyclerView.Adapter<> indicando el ViewHolder** que tiene en su interior, y
- Se implementan los **métodos abstractos**.

```
public class ListaEmpleadosAdapter
        extends RecyclerView.Adapter<ListaEmpleadosAdapter.EmpleadoViewHolder> {

    private List<Empleado> listaEmpleados;
    private Context context;

    @NonNull
    @Override
    public EmpleadoViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull EmpleadoViewHolder holder, int position) {

    }

    @Override
    public int getItemCount() {
        return 0;
    }
```

Método `onCreateViewHolder()`

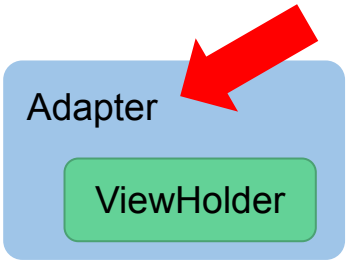


En el método **`onCreateViewHolder()`** se debe :

- “inflar” el layout (xml) del elemento que usará el RecyclerView (el ItemLayout ➜ item_rv),
- Luego crear la instancia ViewHolder donde estará la información.

```
@NonNull
@Override
public EmpleadoViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
    View view = LayoutInflater.from(context).inflate(R.layout.item_rv, parent, false);
    return new EmpleadoViewHolder(view);
}
```

Método onBindViewHolder()



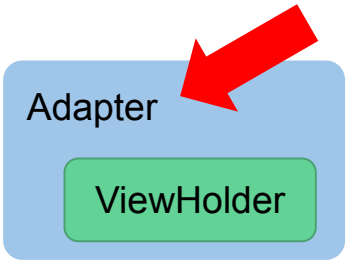
Indica **cómo se llenará cada ViewHolder** cuando se tenga información.

Para llenar el viewHolder, **se puede obtener el índice del elemento** con la variable "position" y **guardar ese empleado en el viewholder** para usarlo posteriormente.

Finalmente, puede cambiar cada elemento del viewHolder mediante **holder.itemView.findViewById**

```
@Override
public void onBindViewHolder(@NonNull EmpleadoViewHolder holder, int position)
{
    Empleado e = listaEmpleados.get(position);
    holder.empleado = e;

    TextView textViewFirstName = holder.itemView.findViewById(R.id.textViewRv1);
    textViewFirstName.setText(e.getFirstName());

    TextView textViewLastName = holder.itemView.findViewById(R.id.textViewRv2);
    textViewLastName.setText(e.getLastName());

}
```

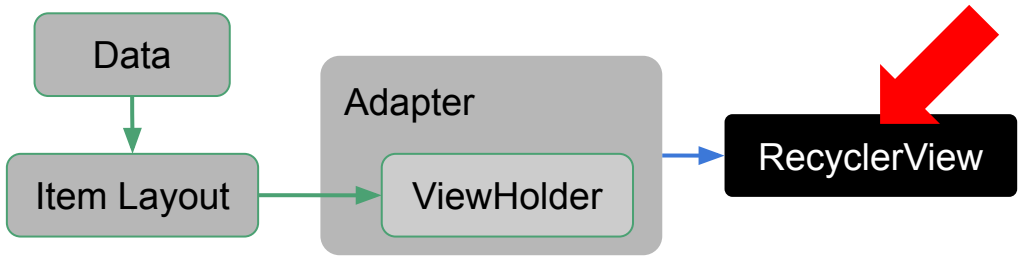
Método `getItemCount()`



Indica la **cantidad total de elementos**, en nuestro caso, del arreglo “data”.

```
@Override
public int getItemCount() {
    return listaEmpleados.size();
}
```

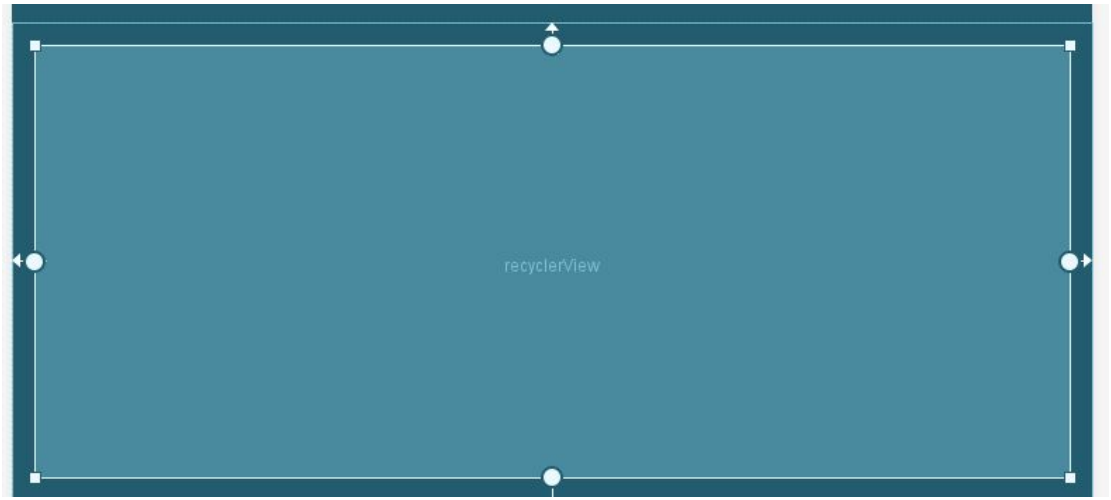
Crear el layout del RecyclerView



El RecyclerView es un componente vacío, pues este se llena de forma dinámica con la información enviada.

RecyclerView le exige que le defina:

- id
- width
- height

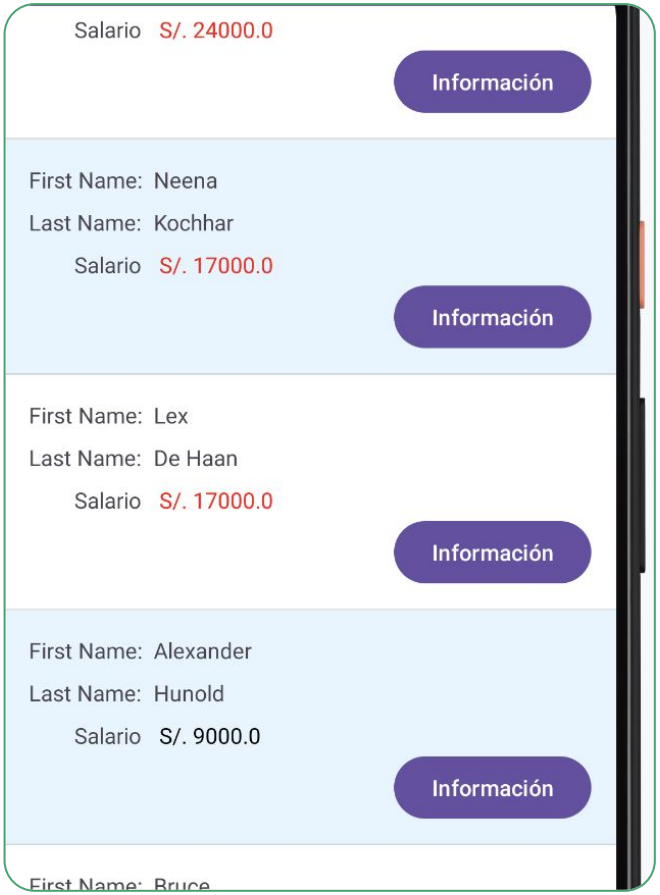


Vincular el RecyclerView con su Adapter y su LayoutManager

- Se crea una instancia del adapter, enviando los empleados y el contexto.
- Se coloca el Adapter al RecyclerView
- Se configura el RecyclerView con un LinearLayoutManager para que disponga los componentes de forma vertical.

```
if (response.isSuccessful()) {
    EmpleadoDto empleadoDto = response.body();
    Empleado[] lista = empleadoDto.getLista();

    ListaEmpleadosAdapter adapter = new ListaEmpleadosAdapter();
    adapter.setContext(MainActivity.this);
    adapter.setListaEmpleados(Arrays.asList(lista));

    binding.recyclerView.setAdapter(adapter);
    binding.recyclerView.setLayoutManager(new LinearLayoutManager(MainActivity.this));

}
```

Probando...


Salario S/. 24000.0
Información
First Name: Neena
Last Name: Kochhar
Salario S/. 17000.0
Información
First Name: Lex
Last Name: De Haan
Salario S/. 17000.0
Información
First Name: Alexander
Last Name: Hunold
Salario S/. 9000.0
Información
First Name: Bruce

Clic en un ViewHolder

Para gestionar acciones en un ViewHolder, debe realizarlo desde el constructor.

```
public class EmpleadoViewHolder extends RecyclerView.ViewHolder {

    Empleado empleado;

    public EmpleadoViewHolder(@NonNull View itemView) {
        super(itemView);
        Button button = itemView.findViewById(R.id. buttonRv);
        button.setOnClickListener(view -> {
            Integer id = empleado.getId();
            Log.d("msg-test", "Presionando el empleado con id: "  + id);
        });

    }
}
```

Actualizar la lista

Al actualizar la lista, lo debe hacer directamente sobre la data del adaptador, esto refrescará la información; sin embargo, para que la parte visual vea estos cambios, debe hacer:

```
adapter.notifyDataSetChanged();
```

➜ Si borra datos de la vista, debe borrarlo de la lista y de la parte visual.

```
list.remove(position);
recycler.removeViewAt(position);
adapter.notifyDataSetChanged();
```

¿Preguntas?

Muchas gracias