AquaSmart

Autor	**José Morillo Almazán**
Asignatura	PMDM
Curso	2º DAM (A)

Versión 1.1. Descripción General

La aplicación utiliza un `RecyclerView` para mostrar una lista de reportes. La arquitectura se basa en un **Controller**, un **Adapter**, y un **ViewHolder** para gestionar la interacción y visualización de los datos.

Principales Componentes

1. **Controller**: Controla la lógica de la aplicación, como el manejo de los datos y la interacción con el `RecyclerView`.
2. **Adapter**: Administra la creación y vinculación de vistas individuales en el `RecyclerView`.
3. **ViewHolder**: Representa y renderiza cada elemento en la lista del `RecyclerView`.

Clases

Clase `Controller`

```
class Controller(private val context: Context) {

    private lateinit var listReports: MutableList<Reports>
    private lateinit var adapter: AdapterReports

    init {
        initData()
    }

    /**
     * Método que inicializa la lista de reportes y e inicializa el adaptador
     * del recyclerView
     */
    private fun initData() {

        listReports = ReportsDaoImpl.myDao.getReports().toMutableList()

        adapter = AdapterReports(listReports,
            { position ->
                deleteReport(position)
            })
    }

    /**
     * Método que setea el adaptar del RecyclerView asociado a la vista
     */
    fun setAdapter() {
        val myActivity = context as MainActivity
        myActivity.binding.rvReports.adapter = adapter
    }

    /**
     * Método para borrar un Reporte.
     * También notifica de que un item ha sido eliminado.
     */
    private fun deleteReport(position: Int) {
        listReports.removeAt(position)
        adapter.notifyItemRemoved(position)
        adapter.notifyItemRangeChanged(position, listReports.size)
    }
```

```
}
```

Descripción

Esta clase gestiona la lógica principal de la aplicación. Su objetivo es inicializar los datos, configurar el adaptador del `RecyclerView` y manejar la eliminación de elementos en la lista.

Métodos

- `initData()`
  Inicializa la lista de reportes obtenida de la capa DAO y configura el adaptador.
- `setAdapter()`
  Asigna el adaptador al `RecyclerView` asociado a la vista principal.
- `deleteReport(position: Int)`
- Elimina un reporte de la lista en una posición específica y actualiza el adaptador para reflejar los cambios.

  Nota: Este método también notifica al adaptador sobre el cambio en el rango de los datos.

Uso del `Controller`

```
val controller = Controller(context)
controller.setAdapter()
```

Clase `AdapterReports`

Descripción

El adaptador de RecyclerView gestiona la creación y vinculación de vistas para cada elemento de la lista de reportes.

Métodos

- `onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolderReports` Infla la vista para un elemento y crea un ViewHolderReports asociado.

```
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolderReports {

        val binding =
ItemReportBinding.inflate(LayoutInflater.from(parent.context), parent, false)
```

```
        return ViewHolderReports(binding, deleteOnClick)
    }
```

- `getItemCount(): Int` Devuelve el tamaño de la lista de reportes.

```
override fun getItemCount(): Int = listReports.size
```

- `onBindViewHolder(holder: ViewHolderReports, position: Int)` Vincula un ViewHolderReports con los datos de un reporte específico en la posición dada.

```
override fun onBindViewHolder(holder: ViewHolderReports, position: Int) {
        holder.render(listReports[position])
    }
```

Clase `ViewHolderReports`

Descripción

Esta clase extiende RecyclerView.ViewHolder y se encarga de renderizar un reporte individual en el RecyclerView.

Métodos

- `render(report: Reports)` Muestra la información de un reporte en la vista asociada al ViewHolder.

```
fun render(report: Reports) {

        binding.textViewTitle.text = report.name
        binding.textViewSubtitle.text = report.date.toString()
        binding.textViewBody.text = report.clientName

        binding.ivReport.post {
            Glide.with(itemView.context)
                .load(report.image)
                .centerCrop()
                .into(binding.ivReport)
        }

        binding.floatButtonDelete.setOnClickListener {
            deleteOnClick(adapterPosition)
        }
    }
```

Resumen de la lógica

1. `Inicialización:` El Controller inicializa los datos llamando al DAO (ReportsDaoImpl) y crea una list mutable de reportes. Configura el AdapterReports para manejar la interacción.

2. `Configuración del Adaptador:` El método setAdapter() conecta el adaptador al RecyclerView definido en el MainActivity.

3. `Interacción del Usuario:` El usuario puede eliminar reportes. Esta acción activa el deleteOnClick definido en el AdapterReports, que llama al método deleteReport() en el Controller.