Especificaciones Aplicación

Vamos a partir del proyecto `Climatización` proporcionado en los recursos para el examen y que contiene ya ciertos módulos y clases que nos van a ser útiles para la realización del examen y entre los cuales **podemos destacar**:

En el paquete `com.climatizacion.data`:

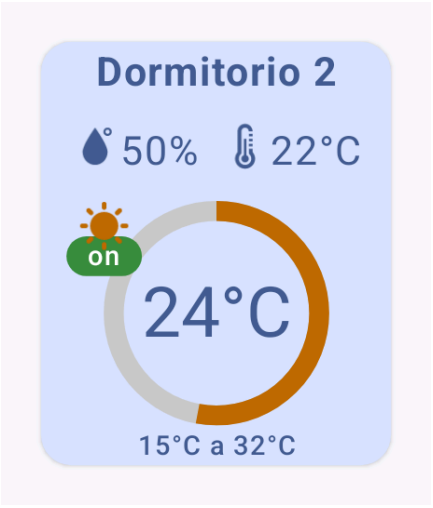
- En el paquete `.termostato` están `TermostatoMock.kt` y `TermostatoDaoMock.kt` que contiene un listado de termostatos que vamos a utilizar para la primera carga de datos en la base de datos de Room.

En el paquete `com.climatizacion.ui.composables`:

- Dispones de las clases utilizadas en los materiales del curso denominas que puedes utilizar para algunos de los apartados del examen.

En el paquete `com.climatizacion.ui.features`:

- Dispones del data class `TermostatoUiState.kt` que vamos a utilizar para gestionar los estados de los termostatos de la casa.
- Em el paquete `.termostato` dispones de `TermostatoEvent.kt` y `TermostatoScreen.kt` que contiene la implementación de la UI de un termostato y que deberás completar según las especificaciones que se indican en la parte 3 del ejercicio.
- Em el paquete `.casa` dispones del enum `FiltroTermostatos.kt` , `CasaEvent.kt` con los eventos de la pantalla y `TermostatoThumb.kt` que contiene la definición de la UI para **ver un termostato** en la lista a modo de galería de termostatos y que puedes ver en las imagen de ejemplo de ejemplo. Por último, también dispones de `CasaScreen.kt` que deberás completar según las especificaciones que se indican en la parte 2 del ejercicio.



En el se indica:

- El nombre de la sala, su humedad relativa y la temperatura de la sala donde está instalado.
- El termostato está encendido o apagado.
- El termostato está en modo frío o calefacción mediante el símbolo correspondiente.
- El progreso circular que indica la temperatura de ajuste del termostato y que irá acompañada de un progreso circular que cuando esté encendido se mostrará con un color.

Dispones de diferentes clases de utilidad utilizadas en los materiales del curso y que puedes utilizar si lo consideras necesario de forma opcional.

Cuando tengas algún tipo de duda sobre el funcionamiento descrito en las especificaciones, puedes consultar el vídeo **ejecucion.mp4** que se te proporciona en los recursos.

Parte 1

Crea un BD con Room para almacenar todos los datos de nuestro modelo denominada "***refrigeracion.db***". La BD debe contener una única tabla denominada ***termostatos*** donde la clave primaria será el campo ***id*** de tipo ***Int*** que tienes en el modelo. Los campos de las tabla deben ir en `snake_casing` y los puedes deducir del modelo.

Las operaciones sobre la tablas serán todas asíncronas y debes definir las siguientes:

- Operaciones básicas `insert`, `delete`, `update` y `count`
- `get()`: Devuelve todos los termostatos de la base de datos.
- `get(Int)`: Devuelve un termostato de la base de datos a partir de su id.
- `getEncendidos()`: Devuelve todos los termostatos de la base de datos que estén encendidos, encendido igual a 1.
- `getApagados()`: Devuelve todos los termostatos de la base de datos que estén apagados, encendido igual a 0.

> **Importante**
>
> Úsalos en **Repository** sustituyendo el `TermostatoDaoMock` por el `TermostatoDao` de Room.

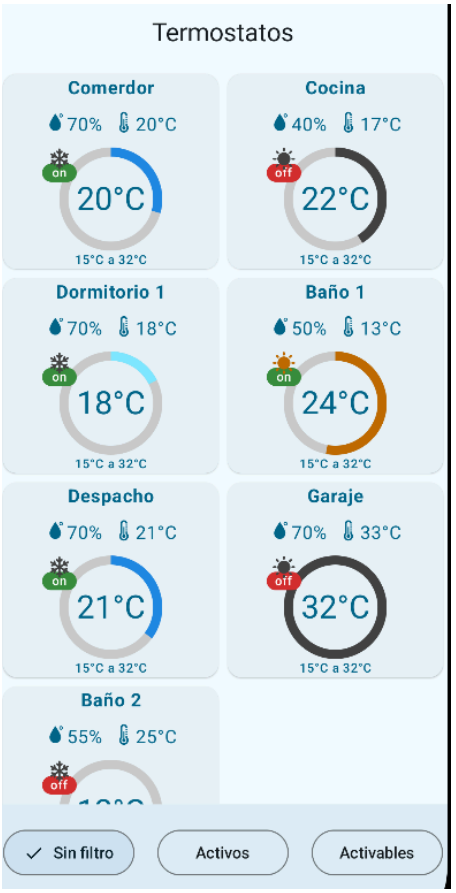
Prepara los '*providers*' que consideres necesarios de las instancias de Room para la inyección de dependencias con **Hilt**. Teniendo en cuenta que se creará una **instancia única** de cada uno de ellos.

Modifica el '*repositorio*' para usar los métodos definidos.

Por último, realiza la primera carga de datos de la base de datos de Room a partir de los datos `TermostatoDaoMock`. **Solo si la BD está vacía**. Para ello, realizarás la carga de datos en la BD al iniciar la aplicación, utilizando el método onCreate de la clase `ClimatizacionApplication` e inyectando los repositorios con Hilt.

Parte 2

Basándote en el componente `TermostatoThumb.kt` proporcionado, define la pantalla con la lista de termostatos `CasaScreen.kt` y su correspondiente `CasaViewModel.kt` que gestione el **filtrado usando Room**, así como manteniendo los estados y gestionando los eventos de `CasaScreen` en el paquete `.casa`.



La lista de termostatos debe estar contenida en un Scaffold con un `CenterAlignedTopAppBar` que se colapse al hacer scroll en la lista y un `BottomAppBar` que cambiará dependiendo de si tenemos seleccionado Sin filtro, Activos, Activables.

- **Sin filtro**, mostrará todos los termostatos de la base de datos.
- **Activos**, mostrará todos los que estén activos.
- **Activables**, mostrará de aquellos termostatos que no estén activos, los que sean activables (en caso de frio, aquellos en la que la temperatura del termostato sea menor que la de la estancia y en caso de calor aquellos en la que la temperatura de termostato sea mayor que la de la estancia).

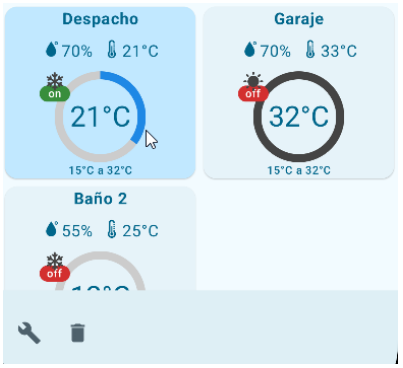
Por ejemplo, para termostatos activables se mostraría como en el ejemplo de la derecha.

Si no hay ningún termostato seleccionado la `BottomAppBar` debe ofrecer los tres componentes `SegmentedButton` que deberemos incluir en el contenedor `SingleChoiceSegmentedButtonRow` y con la funcionalidad que se ha explicado en el párrafo anterior *(puedes ver un ejemplo de uso en el código de TermostatoScreen.kt).*

> **Nota**
>
> A la propiedad space de `SingleChoiceSegmentedButtonRow` se le asignará `-20.dp`. Mientras que a la propiedad `shape` de SegmentedButton se le asignará `SegmentedButtonDefaults.baseShape`.

Si tenemos un termostato seleccionado la `BottomAppBar` debe ofrecer dos botones de acción:

- Uno para **editar el termostato seleccionado** con el icono `Icons.Filled.Edit`.
- Otro para **eliminar el termostato seleccionado** con el icono `Icons.Filled.Delete`.



Tanto el filtrado como las modificaciones y eliminaciones se harán en `CasaViewModel.kt` llamando a los métodos correspondientes del repositorio que usarán las operaciones definidas en la BD con Room.

Parte 3

Completa el código de `TermostatoScreen.kt` para que en la parte inferior del termostato muestre un Slider con los botones de incremento y decremento como en la imagen (*el componente está implementado en composables*). El Slider podrá modificar su valor desde el propio componente o desde alguno de los botones que lo acompañan, en todo momento se reflejará el cambio que hagamos de la temperatura en el componente `CircularProgressIndicator`

Además se deberá añadir el `FloatingActionButton` con la funcionalidad de guardar los cambios de esta pantalla.

A parte de añadir los componentes anteriores para que el screen quede como en la imagen, se deberá añadir el código necesario para que las funcionalidades del interruptor de encendido apagado y del selector del modo (frio/calefacción), sean correctas. Para ver el funcionamiento de esta pantalla, te puedes remitir al video que se adjunto como recurso.

> **Nota**
>
> La gestión de funcionamiento del Slider y el resto de botones de este componente se realizará en `TermostatoViewModel.kt`, además podrás crear una clase sellada para gestionar los eventos si lo ves necesario.

Parte 4

Implementación de la **navegación** para que desde la primera pantalla `CasaScreen` se pueda navegar a `TermostatoScreen` al pulsar la acción de modificar la temperatura del termostato, y la gestión de atrás al guardar.

Debes usar los patrones vistos en clase para implementar la navegación y evitar pasar el `NavController` dentro de los Screen.