Perfecte! Ara que tens un `ViewModel` que encapsula l'estat (`NotificacioConfiguracio`), podem afegir **validació reactiva** a mesura que l'usuari escriu. Et proposo començar pas a pas validant:

1. Que `limitNotifications` sigui enter positiu.
2. Que `kmsInicial` i `kmsAvis` siguin floats positius.
3. Que `kmsAvis` sigui més gran que `kmsInicial` (si cal lògica de negoci).

🧩 Pas 1: Ampliar el `ViewModel` amb un estat de validació

Afegirem un altre `StateFlow` per representar si la configuració és vàlida.

```
class SettingsViewModel : ViewModel() {

    private val _estatNotificacio = MutableStateFlow(NotificacioConfiguracio())
    val estatNotificacio: StateFlow<NotificacioConfiguracio> = _estatNotificacio

    private val _formulariValid = MutableStateFlow(false)
    val formulariValid: StateFlow<Boolean> = _formulariValid

    fun onLimitNotificationsChange(value: TextFieldValue) {
    _estatNotificacio.value = _estatNotificacio.value.copy(limitNotifications = value)
    validarFormulari()
    }

    fun onKmsInicialChange(value: TextFieldValue) {
    _estatNotificacio.value = _estatNotificacio.value.copy(kmsInicial = value)
    validarFormulari()
    }

    fun onKmsAvisChange(value: TextFieldValue) {
    _estatNotificacio.value = _estatNotificacio.value.copy(kmsAvis = value)
    validarFormulari()
    }

    private fun validarFormulari() {
    val estat = _estatNotificacio.value
    val limitOk = estat.limitNotifications.text.toIntOrNull()?.let { it > 0 } ?: false
    val kmsInicialOk = estat.kmsInicial.text.toFloatOrNull()?.let { it >= 0f } ?: false
    val kmsAvisOk = estat.kmsAvis.text.toFloatOrNull()?.let { it > 0f } ?: false
```

```
        // Exemple de lògica: kmsAvis ha de ser major que kmsInicial
        val relacioOk = estat.kmsInicial.text.toFloatOrNull() != null &&
                estat.kmsAvis.text.toFloatOrNull() != null &&
                estat.kmsAvis.text.toFloat() > estat.kmsInicial.text.toFloat()

        _formulariValid.value = limitOk && kmsInicialOk && kmsAvisOk && relacioOk
        }
}
```

Pas 2: Activar o desactivar el botó segons la validació

Al teu `AfegirBtn`, rep el `formulariValid` com a paràmetre:

```
@Composable
fun AfegirBtn(
        state: NotificacioConfiguracio,
        formulariValid: Boolean,
        modifier: Modifier = Modifier
) {
        Row {
        SmallFloatingActionButton(
        onClick = {
                Log.d("Info", state.kmsAvis.text + "," +
                state.kmsInicial.text + "," + state.limitNotifications.text)
        },
        enabled = formulariValid,
        modifier = Modifier,
        content = {
                Icon(Icons.Filled.Add, "Add configuration button")
        }
        )
        }
}
```

🧩 Pas 3: Passar el `formulariValid` a `AfegirBtn`

Modifica la crida des de `SettingsContent`:

```
val state by viewModel.estatNotificacio.collectAsState()
val formulariValid by viewModel.formulariValid.collectAsState()

ParametritzacioNotificacio(state, viewModel)
AfegirBtn(state, formulariValid)
```

Resultat

Ara tens un sistema reactiu on cada canvi actualitza l'estat i recalcula la validesa del formulari, activant o desactivant el botó dinàmicament.