# Coordinación de Ingeniería en Sistemas Computacionales

| | | | |
|---|---|---|---|
| **Materia:** | Programación Móvil I | | |
| **Práctica:** | Estado en Jetpack Compose | | |
| **Alumno(s):** | Arnold Javier Reyes Garcia | **Fecha:** | 19-09 –2025 |

## Objetivo

Comprender y aplicar los conceptos fundamentales del manejo de estado en Jetpack Compose. El objetivo es desarrollar una aplicación simple que demuestre cómo el estado de la aplicación determina la interfaz de usuario y cómo Compose reacciona a los cambios de estado para actualizar la UI de manera eficiente.

## Temas del plan de estudios

- Programación en Kotlin para Android.
- Uso de Android Studio.
- Conceptos de Jetpack Compose: Estado y recomposición.

## Material

- Android Studio
- Documentación oficial de Android Developers.
- Codelab: "Estado en Jetpack Compose".

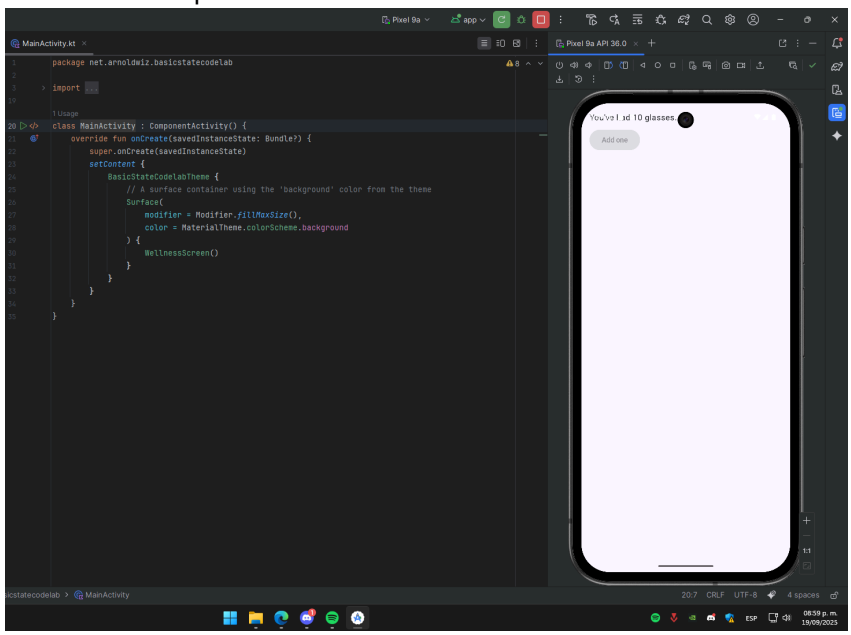
## Marco Teórico

El estado en una aplicación es cualquier valor que puede cambiar con el tiempo. En Jetpack Compose, la interfaz de usuario es una función del estado. Cuando el estado cambia, las funciones Composable que leen ese estado se recomponen para describir la nueva UI. Los eventos son las acciones que modifican el estado. Un patrón clave es la elevación de estado, que consiste en mover la gestión del estado a un componente padre para hacer que los Composables hijos no tengan estado, sean más reutilizables y fáciles de probar.

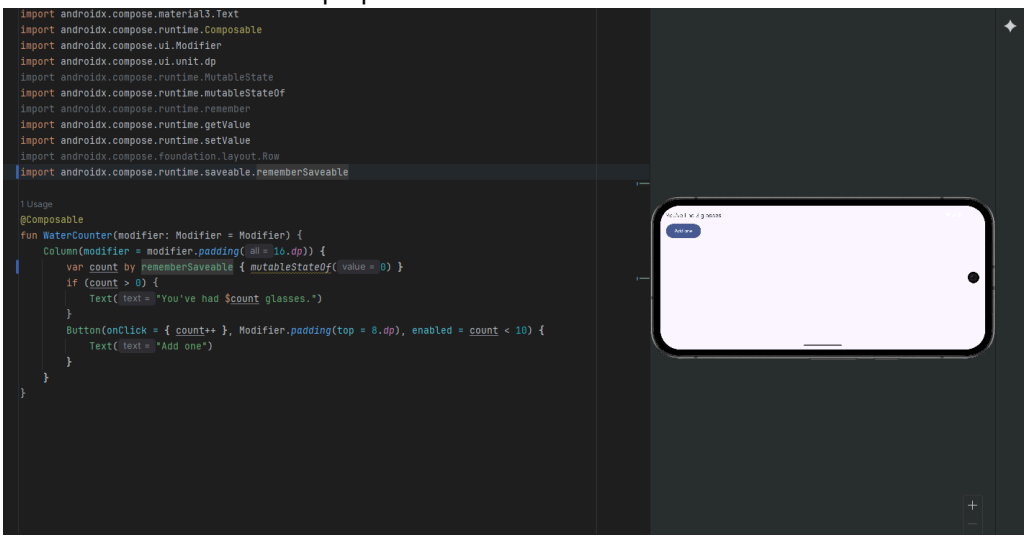
## Desarrollo

Para esta práctica, se siguió el codelab "Estado en Jetpack Compose", creando una aplicación de bienestar.

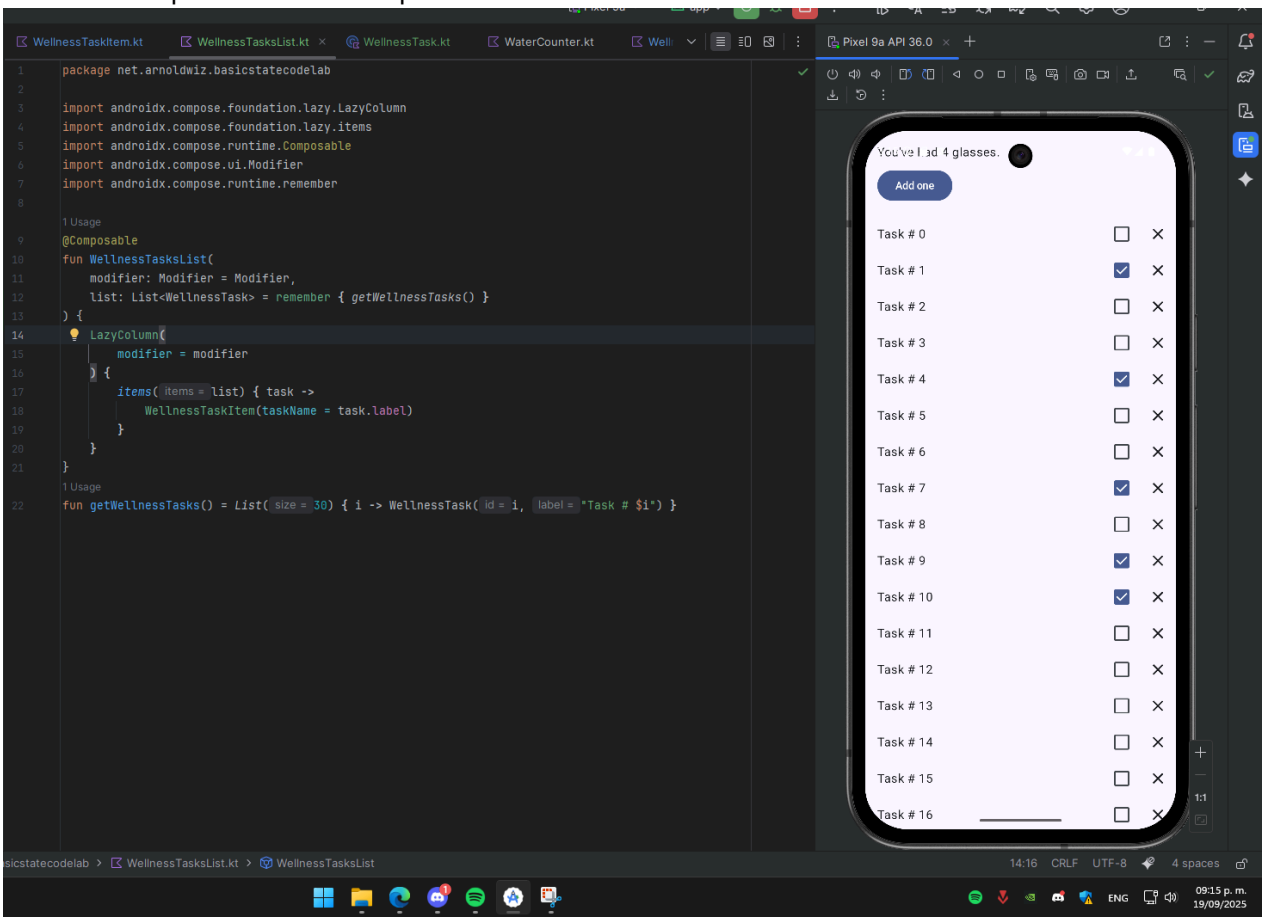
Se implementó un contador simple para registrar los vasos de agua consumidos. Se utilizó remember y mutableStateOf para almacenar el estado del contador dentro de un Composable.




Para hacer el contador más reutilizable, se aplicó el patrón de "elevación de estado". El estado (el conteo de vasos) y la lógica para modificarlo se movieron a un Composable padre, haciendo que el Composable del contador no tuviera estado propio.



Se agregó una lista de tareas que el usuario puede marcar como completadas. Se utilizó rememberSaveable para que el estado de la lista persista a través de cambios de configuración, como la rotación de la pantalla. También se implementó un botón para eliminar tareas de la lista.




Finalmente, toda la lógica de negocio y la gestión del estado se trasladaron a un ViewModel. Esto asegura que el estado sobreviva a los cambios de configuración y separa las responsabilidades, siguiendo las mejores prácticas de arquitectura de Android.

## Resultado

La aplicación final permite al usuario:

- Llevar un conteo de los vasos de agua bebidos.
- Visualizar una lista de tareas de bienestar.
- Marcar tareas como completadas.
- Eliminar tareas de la lista.

- Mantener el estado de la aplicación incluso después de rotar el dispositivo.

## Conclusiones

Esta práctica permitió comprender en profundidad cómo funciona el estado en Jetpack Compose. Se aprendió a utilizar las APIs remember, rememberSaveable y mutableStateOf para manejar el estado localmente. Además, se aplicó el patrón de elevación de estado para crear componentes sin estado, mejorando la reutilización del código. La integración con ViewModel demostró ser fundamental para desarrollar aplicaciones robustas y escalables que siguen una arquitectura limpia.

## Bibliografía


- Android Developers