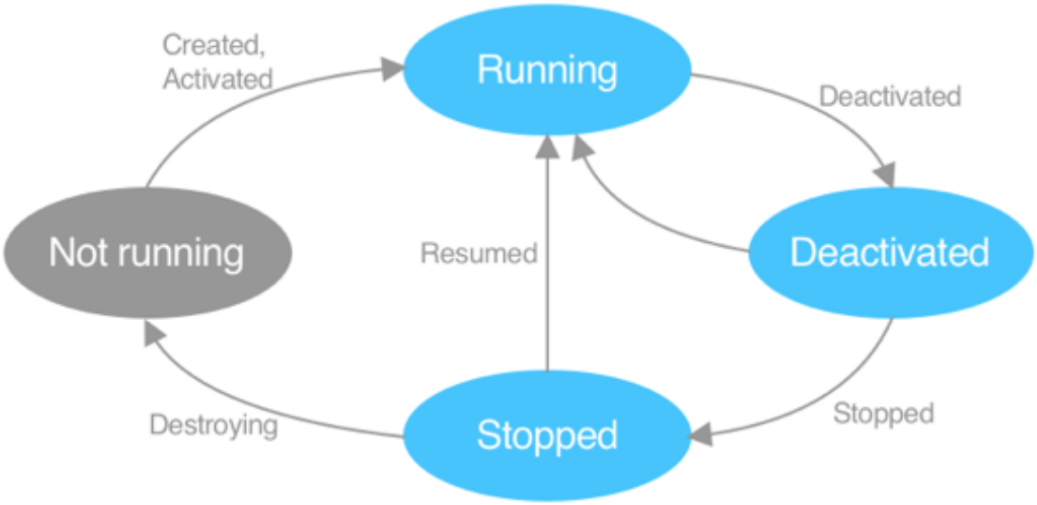
.NET Multi-platform App UI (.NET MAUI) apps generally have four execution states: *not running*, *running*, *deactivated*, and *stopped*. .NET MAUI raises cross-platform lifecycle events on the `Window` class when an app transitions from the not running state to the running state, the running state to the deactivated state, the deactivated state to the stopped state, the stopped state to the running state, and the stopped state to the not running state.

The following diagram shows an overview of the .NET MAUI app lifecycle:



In the diagram, the gray oval indicates that the app isn't loaded into memory. The light blue ovals indicate that the app is in memory. Text on arcs indicates events that are raised by .NET MAUI, that provide notifications to the running app.

The execution state of an app depends on the app's history. For example, when an app is installed for the first time, or a device is started, the app can be considered to be *not running*. When the app is started, the `Created` and `Activated` events are raised and the app is *running*. If a different app window gains focus, the `Deactivated` event is raised and the app is *deactivated*. If the user switches to a different app or returns to the device's Home screen, so that the app window is no longer visible, the `Deactivated` and `Stopped` events are raised and the app is *stopped*. If the user returns to the app, the `Resuming` event is raised and app is *running*. Alternatively, an app might be terminated by a user while it's running. In this situation the app is *deactivated* then *stopped*, the `Destroying` event is raised, and the app is *not running*. Similarly, a device might terminate an app while it's stopped, due to resource restrictions, and the `Destroying` event is raised and the app is *not running*.

In addition, .NET MAUI enables apps to be notified when platform lifecycle events are raised. For more information, see Platform lifecycle events.