## Consideraciones antes de iniciar el desarrollo

- 1. Requisitos del proyecto: Incluir una descripción detallada de lo que se espera lograr.
- 2. Especificaciones técnicas: Establecer documentación sobre frameworks, SDK, bibliotecas, servicios, etc. que se fueran a utilizar en el desarrollo.
- 3. Diseños y maquetas: Si hay diseños de interfaz de usuario o maquetas disponibles se deberá compartir con el fin de entender como tiene que verse y funcionar la aplicación.
- 4. Documentación existente: Si hay algún código o documentación existente relacionada con el proyecto, proporcionará una comprensión de la arquitectura y la lógica de la aplicación.
- 5. Flujo de trabajo y proceso de desarrollo: Esto puede incluir el uso de control de versiones, sistemas de seguimiento de problemas, prácticas de pruebas, entre otros.
- 6.Restricciones y limitaciones: Tener identificado cualquier restricción o limitación que pueda afectar el desarrollo, como restricciones de tiempo, presupuesto, o cualquier otro.
- 7. Planes de pruebas y casos de prueba: Si hay planes de pruebas existentes, es importante revisarlos para comprender qué se espera que funcione y cómo se probará la aplicación.
- 8. Recursos y apoyo disponibles: Conocer qué recursos y apoyo tiene el desarrollador a su disposición, como la disponibilidad de otros miembros del equipo, documentación adicional, o herramientas específicas.



## Buenas Prácticas para Desarrollo de Aplicaciones Nativas iOS:

- 1. Diseño Orientado a la Experiencia del Usuario (UX/UI): Seguir las pautas de diseño de Apple y los componentes nativos para asegurarse de que la aplicación ofrezca una experiencia de usuario intuitiva y atractiva.
- 2. Utilizar Auto Layout: Utilizar Auto Layout para que la aplicación se vea bien en diferentes tamaños de pantalla y orientaciones.
- 3. Gestionar Memoria de Manera Eficiente: Usar ARC (Conteo de Referencias Automático) para administrar la memoria y evitar fugas de memoria.
- 4. Seguir el Principio de Separación de Responsabilidades: Dividir el código en módulos y clases para mantener un código limpio y organizado.
- 5. Utilizar Patrones de Diseño de Software: Emplea patrones de diseño como MVC (Model-View-Controller) y VIPER (View, Interactor, Presenter, Entity y Router) para facilitar la escalabilidad y mantenibilidad del código.
- 6. Optimizar el Rendimiento: Realizar pruebas de rendimiento y optimización para asegurarte de que la aplicación funcione sin problemas y con fluidez.
- 7. Gestionar los Ciclos de Vida de la Aplicación: Manejar adecuadamente los eventos del ciclo de vida de la aplicación (como viewDidLoad, viewWillAppear, etc.).
- 8. Gestionar la Persistencia de Datos: Usa Core Data o UserDefaults para almacenar y recuperar datos de manera eficiente.

