E1. Fundamentos y ciclo de vida de apps

Sitio: Agencia de Habilidades para el Futuro
Curso: Desarrollo de Aplicaciones para Dispositivos 2° D
Libro: E1. Fundamentos y ciclo de vida de apps

Imprimido por: Eduardo Moreno
Día: jueves, 14 de agosto de 2025, 22:52

Descripción

Tabla de contenidos

Introducción

Este primer libro es la puerta de entrada al universo de las aplicaciones móviles. Contiene las bases conceptuales de este tipo de desarrollo, presentadas en torno a dos temas:

1. **Conceptos generales de aplicaciones móviles**
2. **Ciclo de vida de una aplicación móvil**

- Además... ¡tenemos un extra!

Un glosario de términos técnicos que te permitirá familiarizarte con el vocabulario del área.

¡Comencemos!

Conceptos generales de aplicaciones móviles

¡Comenzamos nuestro recorrido sobre aplicaciones móviles ! Nuestra vida cotidiana y profesional está atravesada por el uso de este tipo de desarrollos... desde pedir comida por el celular o utilizar el GPS para llegar a un lugar, estamos usando aplicaciones móviles.

Definición Una aplicación móvil es el *software* en un dispositivo portátil (similar a los programas en una computadora de escritorio). Desarrollar una aplicación de este tipo supone un largo camino, que consta de varias etapas de desarrollo, desde su conceptualización hasta su planificación.

A lo largo de este tema, estudiaremos en detalle cada etapa.

Vamos a empezar por ahondar en la definición de desarrollo móvil, los elementos que se usan para el desarrollo, como la plataforma -de *software* y de *hardware*- y las aplicaciones móviles, clasificándolas y presentando sus ventajas y desventajas. También, aprenderemos cómo elegir los tipos de app que varía según nuestro público, el tiempo, el rendimiento y acceso al dispositivo y, muy importante, según el presupuesto.
Entramos en un nuevo universo de conocimiento, y es necesario empezar a familiarizarnos con los términos técnicos de los dispositivos móviles: para eso, tendrás un glosario con los conceptos más importantes.

Algunas preguntas clave que trataremos de responder son:

- ¿Qué tipos de aplicaciones móviles hay?
- ¿En qué se parecen y en qué se diferencian?
- ¿Qué es necesario considerar para elegir una aplicación para nuestro proyecto?
- ¿Qué se realiza en cada etapa de su desarrollo?

¡Adelante!

¿Qué es una aplicación móvil?

El teléfono celular revolucionó los formatos de comunicación y a medida que la tecnología avanza cambia la perspectiva en cuanto a su uso.

Podemos decir que el *software* transita una nueva era, ya que las distintas actividades que se realizaban por medio de una computadora; ahora pueden ser ejecutadas en pequeños dispositivos portátiles. Es así como surgen las populares aplicaciones móviles.

Evolución de las aplicaciones móviles

Aunque parecen un fenómeno reciente, la realidad es que las aplicaciones hace tiempo que están entre nosotros. Últimamente, no sólo se han vuelto más populares y atractivas para los/las usuarios/as, sino también para los/las diseñadores/as y desarrolladores/as que están sacando provecho de las posibilidades que ofrecen las nuevas pantallas de teléfono de mayor calidad; por ello deben trabajar codo a codo para lograr aplicaciones que cuiden todos y cada uno de los detalles.

En esencia, una aplicación no deja de ser un *software*. Para entender un poco mejor el concepto, podemos decir que las aplicaciones son para los móviles lo que los programas son para las computadoras de escritorio.

Las aplicaciones comparten la pantalla del teléfono con las webs móviles, pero mientras las primeras tienen que ser descargadas e instaladas antes de usar, a una web puede accederse simplemente usando internet y un navegador; sin embargo, no todas pueden verse correctamente desde una pantalla generalmente más pequeña que la de una computadora de escritorio.

Pero, ¿cuáles son las etapas del desarrollo de una app? **¡Te lo contamos!**

¿Cuáles son las etapas del desarrollo de una app?

Cuando estamos frente al desarrollo de una app puede ocurrir que ya exista una web como antecedente o que el diseño comienza desde cero, que es cuando todavía no hay ni web ni aplicación y hay que decidir por cuál de ellas empezar. Si estamos en este último caso se puede considerar un nuevo concepto conocido como mobile first.

Mobile first sugiere plantear el proceso de diseño teniendo en cuenta el móvil en primer lugar.

- La **ventaja** de esta forma de trabajar es que el pensar en el móvil como punto de partida, obliga a concentrarse en lo esencial de un producto y a hacer foco sólo en lo que tiene sentido para este dispositivo.
- Una vez que la aplicación está diseñada, puede preguntarse **cuál es la mejor forma de llevar lo hecho para el teléfono a una pantalla de computadora o a otros dispositivos,** extendiendo y escalando el contenido y repensando la diagramación.
- Todos los dispositivos tienen usos diferentes, y en el momento de adaptar el diseño, **hay que tener en cuenta las características particulares de cada uno de ellos.**

El proceso de diseño y desarrollo de una aplicación, abarca desde la concepción de la idea hasta el análisis posterior a su publicación en las tiendas. Durante las diferentes etapas, diseñadores/as y desarrolladores/as trabajan la mayor parte del tiempo de manera simultánea y coordinada.

Las etapas son las siguientes:

Hacé clic en el ícono expandir pantalla del visor y podrás ampliar la imagen

Veamos cómo trabajan solapados los/las diseñadores/as y desarrolladores/as:

Hacé clic en el ícono expandir pantalla del visor y podrás ampliar la imagen

¡Estudiaremos los elementos de desarrollo!

Elementos del desarrollo

Muchos términos son nuevos en tu lenguaje, por eso en el "glosario de términos técnicos" se explica el significado para que la lectura del material pueda ser comprendida.

¡No dudes en consultarlo haciendo click aquí!

Para introducirnos en el mundo de las aplicaciones móviles debemos conocer los elementos que se usan para el desarrollo.

Analicemos cuáles son:

Plataforma: Una plataforma es una combinación de *hardware* y *software* utilizado para ejecutar aplicaciones de software. Una plataforma puede ser descrita simplemente como un sistema operativo, la arquitectura de una computadora o la combinación de ambos (un ejemplo de una plataforma común es Microsoft Windows que se ejecuta en la arquitectura x86).

Teniendo en cuenta la definición anterior, a continuación te presentamos los distintos tipos de plataformas:

1. Una plataforma de *hardware* se refiere a la arquitectura de la computadora o del procesador. Las más comunes son x86 o x64 (para computadoras) o arquitectura ARM para dispositivos móviles.
2. Una plataforma de *software* se refiere al sistema operativo, entorno de programación o la combinación de ambos. Una notable excepción a esto es Java, que utiliza un sistema operativo independiente de la máquina virtual para cada código compilado, conocido en el mundo de Java como *bytecode* (es por esto que se dice que Java es multiplataforma).

   Ejemplos de plataformas de *software* incluyen:

- Android (sistema operativo móvil)
- iOS (sistema operativo móvil)
- Java
- Linux (x86, x86-64, PowerPC, y otras arquitecturas)
- Mac OS X (x86, x86-64)
- Microsoft Windows (x86, x86-64)

Aplicaciones móviles: Deben correr en un sistema operativo y en la actualidad existen 2 grandes sistemas operativos móviles: Android y iOS. Esto quiere decir que tenemos que sortear de alguna manera el problema de la incompatibilidad entre ambas plataformas.

Tenemos hoy en día 3 posibles soluciones a esto, resultando en 3 tipos de aplicaciones diferentes:

- Aplicaciones móviles nativas (Native App): se desarrollan específicamente para una plataforma. Las aplicaciones nativas sólo pueden ser ejecutadas en la plataforma para la que se desarrollaron, y si se desea que se ejecuten en otra plataforma debe hacerse una aplicación distinta para la misma. Para una aplicación en iOS programamos en el lenguaje propietario de Apple, Swift (también podemos usar Objective-C), y para una aplicación nativa de Android programamos en Java/Kotlin usando Android Studio. En el mundo de los videojuegos encontramos muchísimas aplicaciones nativas que solo existen para la plataforma Windows, si se tiene una computadora con Mac OS o con Linux no se podrá jugar.

- Aplicaciones móviles híbridas o multiplataforma: son aquellas aplicaciones que desarrollamos para múltiples plataformas a la vez, haciendo sólo una aplicación. Para esto podemos usar lenguajes como Flutter o React Native.
- Aplicaciones móviles web: se realiza un único desarrollo para todas las plataformas, pero se acceden desde el navegador web. Se les llama también "aplicaciones web *responsive*" ya que deben programarse teniendo en cuenta que se utilizarán en dispositivos móviles. Cualquier lenguaje web sirve para este tipo de desarrollo, como por ejemplo React.

Te mostramos un cuadro con las ventajas y desventajas de cada tipo de aplicación para que al momento de analizar el desarrollo puedas elegir cual utilizar.

	Ventajas	Desventajas
Nativas	• Tienen el mejor rendimiento posible de todas las aplicaciones móviles. • Tenemos acceso completo a todas las funciones nativas del dispositivo (como cámara, GPS, almacenamiento, sensores, etc.) • Cada aplicación puede ser subida a su respectiva App Store.	• Debemos aprender más de un lenguaje de programación (si es que queremos cubrir todos los dispositivos). • Debemos desarrollar varias aplicaciones (al menos 2), lo cual aumenta considerablemente el costo.
Híbridas	• Al ser multiplataforma se puede subir a la App Store. • Solo se necesita desarrollar una sola aplicación, disminuyendo el tiempo y los costos.	• Tienen menos rendimiento que las aplicaciones nativas. El rendimiento que requiere una aplicación depende del tipo de experiencia que se quiera generar, por ejemplo, hay juegos que requieren una interfaz visual muy optimizada para brindar una mejor experiencia, por lo que el rendimiento es un punto clave. • El acceso al dispositivo es menos permisivo que en las apps desarrolladas en nativo.
Web	• El tiempo y costo de desarrollo son más bajos ya que es más simple hacer una aplicación web *responsive* que una aplicación móvil. • Es más simple encontrar desarrolladores/as con el conocimiento necesario para hacer este tipo de aplicación.	• No puede subirse a Apps Stores. • Requiere acceso a internet para funcionar. • El acceso a los permisos del dispositivo es significativamente menor al de las aplicaciones híbridas.

Cómo elegir tipos de app

El tipo de aplicación a elegir dependerá de la situación de partida, los recursos y el público al que nos queramos dirigir.

- **Según nuestro público:** tenemos que tener en cuenta el sistema operativo que utiliza mayoritariamente nuestro público objetivo.
- **Según el tiempo:** si necesitás contar con una aplicación en el menor tiempo posible, las aplicaciones web y las híbridas serán una gran opción.
- **Según el rendimiento y acceso al dispositivo:** si lo que necesitás es un gran rendimiento y acceso a funciones específicas del dispositivo, requerís una app nativa.
- **Según el presupuesto:** si tu presupuesto es limitado, las aplicaciones web e híbridas son las opciones más económicas. Luego, podés elegir en función de si deseás tener presencia en las tiendas de apps o no, entre otros factores.

- En resumen, antes de desarrollar una app, debés sentarte a pensar qué funcionalidades querés que tenga, a quién te dirigirás con ella y ser realista en cuanto a presupuesto.
- Solo a partir de ahí podrás elegir el tipo de aplicación más conveniente para tu negocio.
- Sin embargo, esto es solo el principio si realmente querés triunfar con tu app.

Ciclo de vida de una aplicación móvil

Con este tema nos adentraremos en el estudio del ciclo de vida de un desarrollo de *software* móvil, desde la selección del proyecto, pasando por su diseño y desarrollo de *software*, luego por sus pruebas y calidad, hasta la implementación y puesta en producción.

Definición | Entendemos al ciclo de vida como el tiempo de duración de una aplicación de software, desde que se concibe la idea hasta que se retira del uso.

Analizaremos las consideraciones sobre el desarrollo móvil, que comprenden: las consideraciones comunes, las consideraciones de iOS y las consideraciones de Android. También, aprenderemos cómo documentar una app, siguiendo un orden de planificación, con su determinada plantilla, y la historia de usuario.

Hay preguntas claves que vamos a respondernos:

- ¿Cómo seleccionamos un proyecto?
- ¿De qué manera lo diseñamos y luego lo desarrollamos?
- ¿Qué método de prueba llevamos a cabo y cómo evaluamos su calidad?
- ¿Cuándo es preciso su implementación y cuándo su puesta en producción?

La respuesta a estas preguntas será el punto de partida de nuestra búsqueda de conocimiento. ¡Hacía allí vamos!

Inicio/selección del proyecto

La fase de inicio consiste en definir y perfeccionar la idea de una aplicación. Para crear una aplicación correctamente, es importante hacerse algunas preguntas fundamentales.

Antes de publicar una aplicación en una de las App Store públicas debemos observar los siguientes puntos:

- **Ventaja competitiva:** ¿ya existen aplicaciones similares? Si es así, ¿cómo se diferencia esta aplicación de las otras?
- **Integración de la infraestructura:** ¿con qué infraestructura existente se integrará o cuál extenderá?
- **Valor:** ¿qué valor aporta esta aplicación a los usuarios? ¿Cómo la usarán?
- **Forma/movilidad:** ¿cómo funcionará esta aplicación en un factor de forma móvil? ¿Cómo puedo agregar valor mediante tecnologías móviles, como el reconocimiento de ubicación, la cámara, etc.?

Ciclo de vida de un desarrollo móvil

Una buena gestión del ciclo de vida de las aplicaciones de cualquier tipo, es fundamental para el éxito de cualquier desarrollador/ra de *software* en el mercado actual.

El ciclo de vida de una aplicación es la **vida útil** completa de una aplicación de *software*, desde el concepto hasta el final de su vida útil. Es decir, el término se refiere a todo el proceso desde el momento en que se concibe la idea de la aplicación hasta el momento en que se retira del uso.

Si bien el ciclo de vida de la app la podemos resumir en los pasos de

Hacé clic en el ícono expandir pantalla del visor y podrás ampliar la imagen

A estas etapas las podemos descomponer en otras más detalladas y específicas al desarrollar una aplicación móvil, es decir que, cuando nos referimos a este ciclo de vida, agregamos 5 partes importantes en el proceso:

1) Inicio/selección del proyecto: todas las aplicaciones se inician con una idea. Normalmente, la idea se perfecciona en una base sólida para una aplicación.

2) Diseño: la fase de diseño consiste en definir la experiencia del usuario (UX) de la aplicación (como cuál es el diseño general, cómo funciona, etc.), así como convertir esa experiencia del usuario en un diseño de interfaz de usuario (UI) adecuado, normalmente con la ayuda de un diseñador gráfico.

3) Desarrollo: normalmente, es la fase con un uso más intensivo de recursos, esta es la creación real de la aplicación. Principalmente, esto implica dos partes:

a. **Desarrollo *front-end***: se trata del desarrollo de la parte que verá el/la cliente, la capa con la que interactuará el/la usuario/a.

b. **Desarrollo** *back-end* : es la parte de desarrollo en conexión con un servidor o base de datos, que conecta el *front-end* de la aplicación móvil con los datos.

4) Pruebas y calidad: cuando el desarrollo ha avanzado lo suficiente, normalmente el control de calidad empieza a probar la aplicación y se corrigen los errores. A veces, una aplicación pasará a una fase *beta* limitada en la que una audiencia de usuarios más amplia tiene la oportunidad de usarla, enviar comentarios y notificar cambios.

5) Implementación/puesta en producción: a menudo, muchas de estas partes se superponen, por ejemplo, es común que el desarrollo siga mientras se finaliza la interfaz de usuario e incluso puede afectar al diseño de la interfaz de usuario. Además, una aplicación puede estar en una fase de pruebas al mismo tiempo que se agregan nuevas características a una nueva versión.

¿Pero qué implica cada fase?

Diseño

Una vez que se hayan determinado las características y funcionalidades de la aplicación, el siguiente paso es intentar resolver la experiencia del usuario o UX.

El UX está formado por la experiencia del usuario y por la interfaz del usuario, veamos de que se trata:

Diseño de la experiencia del usuario

- La experiencia del usuario se efectúa normalmente a través de prototipos con uno de los muchos conjuntos de herramientas de diseño.
- Los prototipos de experiencia del usuario permiten diseñar esta experiencia sin tener que preocuparse por el diseño real de la interfaz de usuario.
- Al crear prototipos de experiencia del usuario, es importante tener en cuenta las instrucciones de la interfaz para las diferentes plataformas a las que se dirigirá la aplicación.
- La aplicación debería "sentirse cómoda" en todas las plataformas.
- Además, el propio *hardware* también impone decisiones de la experiencia del usuario.
- Asimismo, el factor de forma también influye en las decisiones de la experiencia del usuario.
- Una tableta tiene mucho más espacio y, por tanto, puede mostrar más información. A menudo, lo que necesita varias pantallas en un teléfono, se comprime en una para una tableta.

Diseño de la interfaz de usuario (UI)

- Una vez determinada la experiencia del usuario, el siguiente paso es crear el diseño de la interfaz de usuario.
- Mientras que la experiencia del usuario suele componerse de prototipos en blanco y negro, la fase de diseño de la interfaz de usuario es donde se introducen y finalizan los colores, gráficos, etc.
- Es importante dedicar tiempo a un buen diseño de la interfaz de usuario y, por lo general, las aplicaciones más populares tienen un diseño profesional.
- Al igual que con la experiencia del usuario, es importante comprender que cada plataforma tiene su propio lenguaje de diseño, por lo que una aplicación bien diseñada puede tener un aspecto diferente en cada plataforma.

Desarrollo

Normalmente, la fase de desarrollo se inicia muy pronto.

Una vez que la idea ha madurado en la fase conceptual o de inspiración:

- Se desarrolla un prototipo de trabajo que valida la funcionalidad, las suposiciones y ayuda a comprender el ámbito del trabajo.

Pruebas y calidad

Definición Una prueba es el proceso de solucionar los errores de la aplicación. No debe entenderse sólo desde un punto de vista funcional (por ejemplo: "Se bloquea al hacer clic en este botón"), sino también de facilidad de uso y rendimiento. Es mejor empezar la estabilización muy pronto en el proceso de desarrollo para que se puedan realizar correcciones antes de que sean costosas.

Normalmente, las aplicaciones pasan por las fases Prototipo, Alfa, Beta y Versión candidata para lanzamiento. Hay personas que las definen de forma diferente, pero, generalmente, siguen el modelo siguiente:

a) **Prototipo**: la aplicación aún está en fase de prueba de concepto y sólo funcionan la funcionalidad principal o determinadas partes de la aplicación. Hay errores principales.

b) **Alfa**: la funcionalidad principal tiene normalmente el código completado (compilado, pero no probado por completo). Aún hay errores principales, puede que no haya funcionalidades aisladas.

c) **Beta**: la mayoría de la funcionalidad está completa y ha pasado al menos una corrección de errores y una prueba ligera. Puede que aún haya problemas conocidos principales.

d) **Versión candidata para lanzamiento**: toda la funcionalidad está completa y probada. Salvo errores nuevos, la aplicación es una versión candidata para lanzamiento.

Nunca es demasiado pronto para empezar a probar una aplicación.

Por ejemplo Si se encuentra un problema importante en la fase de prototipo, aún se puede modificar la experiencia del usuario de la aplicación para darle cabida. Si se encuentra un problema de rendimiento en la fase *alfa*, es lo suficientemente pronto para modificar la arquitectura antes de que se haya generado mucho código basado en suposiciones falsas.

Normalmente, según una aplicación avanza en el ciclo de vida, se abre a más personas para que la prueben, envíen comentarios, etc.

Por ejemplo Las aplicaciones prototipo pueden mostrarse sólo a partes interesadas clave o ponerse a disposición de estas partes, mientras que las aplicaciones de versión candidata para lanzamiento pueden distribuirse a los clientes que se hayan registrado para obtener acceso anticipado.

Para realizar primeras pruebas y la implementación en pocos dispositivos, normalmente es suficiente implementar directamente desde un equipo de desarrollo. En cambio, si se va ampliando el público, se puede volver más complicado rápidamente. Por tanto, hay una serie de **opciones de implementación de pruebas** que facilitan este proceso al permitirle invitar a usuarios a un grupo de pruebas, lanzar compilaciones en la web y proporcionar herramientas que permiten que los usuarios envíen comentarios.

Implementación / puesta en producción

Una vez que la aplicación se ha estabilizado o probado, es el momento de **publicarla.** Hay una serie de opciones de distribución diferentes, dependiendo de la plataforma.

Consideraciones sobre el desarrollo móvil

Aunque desarrollar aplicaciones móviles no es fundamentalmente diferente que el desarrollo web o de escritorio tradicional en términos de proceso o arquitectura, existen algunas consideraciones que se deben tener en cuenta, veamos cuales son:

1. **Consideraciones comunes**
2. **Consideraciones de IOS**
3. **Consideraciones de Android**

A continuación profundizarás más en cada una de ellas.

Comunes

Estas consideraciones comunes abordan aspectos fundamentales que deben tenerse en cuenta independientemente del tipo de dispositivo o plataforma de desarrollo.

Consideraciones	Descripción
Multitarea	• Implica la gestión eficiente del rendimiento y la energía cuando varias aplicaciones se ejecutan simultáneamente. • Por ejemplo, en dispositivos móviles, solo una aplicación puede estar en primer plano a la vez para maximizar el espacio en pantalla y reducir el consumo de batería. • La multitarea debe abordarse considerando las diferencias en la gestión entre plataformas como iOS y Android.
Factor de forma	• Refiere a adaptar el diseño de la interfaz de usuario según el tamaño y las características del dispositivo, ya sea teléfono o tableta. • Por ejemplo, el espacio limitado en la pantalla de un teléfono requiere un diseño más compacto y simplificado, mientras que las tabletas permiten una disposición más espaciosa de los elementos de la interfaz. • La efectividad en el diseño de la interfaz varía según el factor de forma del dispositivo.

Fragmentación del sistema operativo y del dispositivo	• Considera las diferencias en hardware y características entre dispositivos, lo que puede afectar el rendimiento y la compatibilidad de la aplicación. • Por ejemplo, una aplicación de videollamadas puede no ser compatible con todos los dispositivos si no todos tienen cámaras funcionales. • Es esencial adaptar el desarrollo, diseño y pruebas para garantizar la compatibilidad con una amplia gama de dispositivos: Planeación y conceptualización: hay que tener en cuenta que el hardware y las características varían de un dispositivo a otro, puede que una aplicación que se basa en determinadas características no funcione correctamente en algunos dispositivos. Por ejemplo, no todos los dispositivos tienen cámaras, por lo que, si se está creando una aplicación de mensajería de vídeo, puede que algunos dispositivos reproduzcan vídeos, pero no los puedan grabar. Diseño: al diseñar la experiencia del usuario (UX) de una aplicación, hay que prestar atención a las diferentes relaciones y tamaños de pantalla de los dispositivos. Además, al diseñar la interfaz de usuario (UI) de una aplicación, se deben tener en cuenta diferentes resoluciones de pantalla. Desarrollo: al usar una característica del código, siempre se debe probar primero la presencia de esa característica. Por ejemplo, antes de usar una característica de dispositivo, como una cámara, confirmar siempre primero que el sistema operativo tenga esa característica. Después, al inicializar el dispositivo o característica, hay que asegurarse de solicitar la compatibilidad actual del sistema operativo sobre el dispositivo y después usar esas opciones de configuración. Pruebas: es muy importante probar la aplicación al principio y con frecuencia en dispositivos reales. Puede haber incluso dispositivos con las mismas especificaciones de hardware en que varíe mucho su comportamiento.
Recursos limitados	• Reconoce las limitaciones de memoria, procesamiento y rendimiento en dispositivos móviles en comparación con equipos de escritorio. • Por ejemplo, cargar imágenes de alta calidad puede agotar rápidamente la memoria disponible, y las aplicaciones que requieren mucho procesamiento pueden impactar negativamente en el rendimiento del dispositivo.

Debido a consideraciones como estas, **es importante crear el código de manera inteligente y realizar la implementación pronto y con frecuencia** en dispositivos reales para validar la capacidad de respuesta.

Según el tipo de dispositivo y la plataforma de desarrollo debemos tener en cuenta diferentes consideraciones

De IOS

Consideraciones	Descripción
Multitarea	• La multitarea está muy controlada en iOS y hay una serie de reglas y comportamientos que la aplicación debe cumplir cuando otra aplicación está en primer plano, de lo contrario, iOS cancelará la aplicación.
Recursos específicos del dispositivo	• En un factor de forma determinado, el hardware puede variar considerablemente entre diferentes modelos. • Por ejemplo, algunos dispositivos tienen una cámara trasera, algunos también tienen una cámara frontal y otros no tienen ninguna. • Algunos dispositivos antiguos (iPhone 3G y anteriores) ni siquiera permiten la multitarea.
Restricciones específicas del sistema operativo	• Para asegurarse de que las aplicaciones respondan correctamente y sean seguras, iOS impone una serie de reglas que deben cumplir las aplicaciones. • Además de las reglas con respecto a la multitarea, hay una serie de métodos de evento de los que la aplicación debe volver en una determinada cantidad de tiempo, de lo contrario, iOS la cancelará. • También las aplicaciones se ejecutan en lo que se conoce como espacio aislado, un entorno que aplica restricciones de seguridad que restringen a qué puede tener acceso la aplicación. • Por ejemplo, una aplicación puede leer y escribir en su propio directorio, pero si intenta escribir en otro directorio de aplicaciones, se cancelará.

De Android

Consideraciones	Descripción
Multitarea	• La multitarea en Android tiene dos componentes; el primero es el ciclo de vida de la actividad: Cada pantalla de una aplicación de Android se representa mediante una actividad y hay un conjunto específico de eventos que se producen cuando una aplicación se coloca en segundo plano o vuelve al primer plano. Las aplicaciones deben respetar este ciclo de vida para responder y comportarse correctamente. • El segundo componente de la multitarea de Android es el uso de servicios. Los servicios son procesos de ejecución prolongada que existen independientes de una aplicación y se usan para ejecutar procesos mientras la aplicación está en segundo plano. • La multitarea está muy controlada en iOS y hay una serie de reglas y comportamientos que la aplicación debe cumplir cuando otra aplicación está en primer plano, de lo contrario, iOS cancelará la aplicación.
Muchos dispositivos y muchos factores de forma	• Google no impone ningún límite en lo que concierne a los dispositivos que pueden ejecutar el sistema operativo Android. • Este paradigma abierto da como resultado un entorno de productos lleno de una gran variedad de dispositivos diferentes con hardware, resoluciones y relaciones de pantalla, características del dispositivo y capacidades muy diferentes. • Debido a la fragmentación extrema de los dispositivos Android, la mayoría de los usuarios elige los 5 o 6 dispositivos más populares para los que diseñar y probar y les da prioridad a estos.
Consideraciones de seguridad	• Todas las aplicaciones en el sistema operativo Android se ejecutan bajo una identidad distinta y aislada con permisos limitados. • De manera predeterminada, las aplicaciones pueden hacer muy poco. • Por ejemplo, sin permisos especiales, una aplicación no puede enviar un mensaje de texto, determinar el estado del teléfono, ni siquiera obtener acceso a Internet. • Para tener acceso a estas características, las aplicaciones deben especificar en su archivo de manifiesto de la aplicación qué permisos quieren y cuándo se instalan. • El sistema operativo lee los permisos, notifica al usuario que la aplicación pide esos permisos y, después, permite al usuario continuar o cancelar la instalación. • Este es un paso esencial en el modelo de distribución de Android, debido al modelo de tienda de aplicaciones abierto, ya que las aplicaciones no se mantienen de la misma forma que en iOS.

Documentar una app

Uno de los elementos más importantes en el ciclo de vida del desarrollo de un *software* es el de su documentación, pues esta ayuda entender de manera sencilla los programas y procesos, evitando así que el conocimiento se pierda.

Generalmente **se siguen pasos de alguna metodología** para ajustarse a normas y reglas entre los participantes de la tarea de documentación.

Proponemos documentar el desarrollo de la app usando metodologías ágiles, dentro del proceso de desarrollo brinda mayor flexibilidad y rapidez en cuanto a las modificaciones que requiera el proyecto, mediante la inclusión constante del usuario en las actividades requeridas; por otro lado, beneficia la aplicación móvil al asegurar la calidad, usabilidad y demás requerimientos especiales que posee el *software* móvil.

Veamos entonces como aplicar esta metodología.

Pasemos a describir de manera concisa las etapas:

Hacé clic en el ícono expandir pantalla del visor y podrás ampliar la imagen

El modelo propone en primer lugar **realizar una correcta planificación de las principales dimensiones del proyecto**, de esta manera se conoce el entorno y modelo de negocio que posee la empresa y con ello dar paso al planeamiento de soluciones.

Cuando la planificación esté lista se procede **al modelado de la base de datos**, así como la **elaboración de prototipos** que permitirán a todos los interesados tener una noción del entregable o producto final.

La ejecución o codificación es la parte más extensa del proyecto, los/las desarrolladores/as se limitan a realizar lo que está planificado para dicho *sprint*. Para que una aplicación pueda ser lanzada a los usuarios finales, esta debe pasar por una serie de **pruebas** que permitan garantizar la calidad de la misma, donde **en caso de identificar errores** o no alcanzar un determinado requerimiento, se debe llevar un **registro de cambios.**

Fases

Cada una de las fases posee actividades organizadas con el objetivo de obtener un producto funcional como entregable al terminar cada *sprint*, donde además se obtendrá los cambios necesarios a realizarse lo cuáles serán discutidos antes de empezar un nuevo *sprint*, esto con la finalidad de ofrecer una aplicación de calidad.

Vamos a documentar la primera etapa.

Plantilla de planificación

Esta planilla nos da la primera aproximación al desarrollo, una vez analizada podemos completar la planilla de constitución del proyecto y pasar a las historias de usuario.

Hacé clic en el ícono expandir pantalla del visor y podrás ampliar la imagen

Historias de usuario

Se pueden definir como cartillas que poseen los pasos a seguir para efectuar una determinada actividad, estas indicaciones deben ser entendibles para los demás interesados, debido a que deben pasar por una revisión donde se valide y confirme su contenido.

Debido a las características propias de los dispositivos móviles como sus pantallas es necesario diseñar una aplicación que sea fácil de comprender y navegar para el usuario, pero a su vez tenga todo el potencial para cumplir con su función, por tal razón este modelo recomienda el uso de bosquejos digitales o prototipos, de esta manera se busca complementar las historias de usuario.

Historias de usuario

Por lo general, el encargado del producto es quien redacta las historias de usuario con base en la investigación de los usuarios y las organiza en una lista para el equipo de desarrollo, también conocida como trabajo pendiente del producto (*backlog*). Aunque técnicamente cualquiera puede redactar historias de usuario, **es responsabilidad del gerente de producto asegurarse de tener toda la información** que el equipo de desarrollo necesita para ejecutar sus iniciativas.

Luego, el equipo de desarrollo priorizará y decidirá qué historias de usuario abordar durante su reunión de planificación de *sprint*.

Para la definición de las historias de usuario se tomó un modelo que presenta los siguientes campos:

1. **ID:** identificador de la historia de usuario.
2. **Rol:** tipo de usuario o función que tiene la persona.
3. **Característica/funcionalidad:** descripción de la historia de usuario, acción que el usuario va a realizar. Acción principal que el usuario realiza en el sistema para cumplir con una necesidad
4. **Razón/resultado:** describe el resultado obtenido al realizar una acción.
5. **Número de escenario:** identifica el número de escenarios asociado con la historia de usuario.
6. **Criterio de aceptación:** condiciones o requisitos que deben cumplirse para que la historia de usuario sea considerada completa y funcional.
7. **Contexto:** describe la situación en la que se ejecuta el escenario y las condiciones bajo las cuales se debe cumplir el criterio de aceptación..
8. **Evento:** acción específica que el usuario lleva a cabo dentro del sistema, activando un proceso o interacción.
9. **Resultado/comportamiento esperado:** comportamiento esperado o el resultado que debe ocurrir una vez que el usuario haya realizado la acción.

ID	Rol	Características/ Funcionalidad	Razón / Resultado	N° de Escenario	Criterio de Aceptación	Contexto	Evento	Resultado /Comportamiento esperado

Ejemplo de una historia de usuario:

Hacé clic en el ícono expandir pantalla del visor y podrás ampliar la imagen

Glosario

Aquí encontrarás una herramienta muy útil para tu inmersión en el mundo del desarrollo móvil: un glosario de términos técnicos.

En este glosario están las definiciones de los principales términos clave de esta materia. Te recomendamos consultarlo cada vez que lo necesites.

Glosario

Capacitor

Capacitor es una herramienta que permite convertir cualquier aplicación web (desarrollada con HTML, CSS y JS) en una aplicación multiplataforma (para iOS, Android, Escritorio o Web), es decir, es el puente de Ionic hacia lo nativo. Capacitor nos permite acceder a los recursos nativos de cada dispositivo (cámara, GPS, multimedia, almacenamiento, etc.) a través de una comunicación sencilla.

Framework

Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular. Su objetivo principal es ofrecer funcionalidad extra a un lenguaje de forma autocontenida para que la programación sea más simple.

Gestor de paquetes

Software que maneja de forma automática los paquetes que podamos necesitar. Se encargan de instalar, actualizar, configurar y eliminar de forma automática los paquetes que se desee utilizar. Los gestores de paquetes más conocidos son npm y yarn.

JavaScript

Lenguaje web por excelencia, todas las páginas web modernas están hechas con JavaScript, aunque la mayoría utiliza algún tipo de framework.

Lenguaje de programación

Lenguaje formal que da a una persona la capacidad de escribir una serie de instrucciones en forma de algoritmos con el fin de controlar el comportamiento de un sistema informático.

Open-Source Software

Es un software que tiene una licencia abierta, esto quiere decir que permite que los/las usuarios/as lo utilicen de forma libre. El código Open-Source es de uso libre y gratuito para cualquiera que desee usarlo, para cualquier propósito.

Paquete

Archivos de software que traen data y metadata necesaria para programar. Pueden incluir funcionalidades puntuales o componentes totalmente nuevos y pre-programados.

Software Development Kit (SDK)

Colección de herramientas que facilitan el desarrollo de software. Estos SDK suelen ser específicos a un lenguaje y una plataforma en particular. Por ejemplo, si queremos desarrollar aplicaciones móviles para Android necesitamos el JDK (Java Development Kit) y si queremos desarrollar aplicaciones para iOS necesitamos el iOS SDK.

Software multiplataforma

Para que el software pueda ser considerado multiplataforma, debe ser capaz de funcionar en más de una arquitectura de computadora o sistema operativo. Esto puede ser una tarea que consume tiempo, ya que los diferentes sistemas operativos tienen diferentes interfaces de programación de aplicaciones o API. Lo que funciona sin problemas en una plataforma puede no compilar en otra, esto aplica no solo para programas de software sino también para lenguajes.