T1.3. Características y limitaciones en el desarrollo de aplicaciones para dispositivos móviles

Las aplicaciones móviles no son aplicaciones de escritorio adaptadas para dispositivos con pantallas pequeñas, son por el contrario, aplicaciones diferentes por varias razones: (i) la capacidad para comunicarse desde cualquier lugar cambia la interacción del usuario con la aplicación, (ii) la interfaz de usuario para una pantalla y teclados pequeños difiere de forma significativa de la interfaz de una aplicación diseñada para un ordenador de sobremesa o un portátil, (iii) los tipos de canales de comunicación son diferentes, los dispositivos móviles incorporan capacidades de voz, mensajería, información de geolocalización y vídeo conferencia (en algunos teléfonos). Las mejores aplicaciones para móviles integran estas capacidades para optimizar la interacción del usuario con los datos y, por último, (iv) la naturaleza de las redes inalámbricas, aunque las redes ofrecen capacidades de datos de banda ancha, estas pueden variar, dependiendo de la calidad de la señal y de la disponibilidad de conexión de la red, en particular si se trata de usuarios móviles.

Limitaciones en la ejecución de aplicaciones para dispositivos móviles

El desarrollo de aplicaciones para dispositivos móviles requiere tener en cuenta las limitaciones que podemos encontrar a la hora de ejecutarlas. Estas limitaciones están relacionadas, principalmente, con las características hardware y de conexión, asociadas a cada dispositivo móvil.
A pesar de que la capacidad de procesamiento y la memoria de los dispositivos móviles han ido mejorando con el paso del tiempo, las aplicaciones deben diseñarse evitando la sobrecarga de elementos multimedia, que exijan del dispositivo ciclos de procesamiento demasiado largos.

El tamaño de las pantallas y la iluminación también son factores determinantes en el diseño de aplicaciones. No hay que olvidar que la mayoría de los datos de entrada que proporciona el usuario son introducidos con una sola mano y en muchas ocasiones en movimiento. Por lo tanto, en la medida de lo posible se le debe facilitar al usuario la entrada de datos por pantalla, a través de los botones de navegación propios de cada dispositivo, de modo que pueda ir seleccionando opciones (desde un menú desplegable, por ejemplo), y que no tenga que rellenar campos de texto, de esta forma podrá realizar varias operaciones en poco tiempo.

La distribución de los elementos en la interfaz marca la diferencia, no solo en la forma en la que se presenta la información, sino también en cómo navega el usuario a través de ella. Lo fundamental es ofrecer interfaces a través de las cuales la entrada de los datos se realice de la forma más intuitiva y natural. Una buena opción es agrupar la información por funcionalidades o por jerarquías, de modo que el usuario acceda a la información que necesita en cada momento, sin tener que pasar por pantallas o información irrelevante para el tipo de operación que desea realizar.

Si además tenemos en cuenta que cada vez que accedemos desde nuestro teléfono móvil a Internet estamos pagando una tarifa por cantidad de bytes descargados y no por tiempo de uso de la red de datos, es necesario que el envío y la recepción de datos se realicen dentro de un tiempo de espera aceptable. Durante el proceso de envío y/o recepción de datos, debemos mantener informado al usuario sobre el progreso de esta operación y el tiempo estimado para la finalización de la misma. El tiempo de espera entre la petición de los datos y el momento en el que empieza a llegar la respuesta es lo que se conoce como latencia. Hay que tener en cuenta que en el caso de aplicaciones para dispositivos móviles la latencia es mucho mayor que la de una aplicación web normal.

También hay que tener presente que cuando se trata del desarrollo de aplicaciones para teléfonos móviles, la función de teléfono (es decir, realizar y/o recibir llamadas), tiene la prioridad más alta. Por lo tanto, en el momento en el que se reciba una llamada, la aplicación debe proporcionar la forma de mantener el estado en el que ha sido interrumpida, para volver a ella cuando la llamada termine y el usuario confirme que quiere volver al punto en el que se interrumpió su ejecución.

En cuanto a las conexiones, debemos tener presente que pueden fallar y de hecho es algo que sucede con relativa frecuencia, por ejemplo, por la falta de cobertura en determinadas áreas. Esto

implica que no podemos dar por sentado que tendremos el acceso a Internet garantizado y que podremos obtener los datos necesarios para la ejecución de la aplicación.

Aunque el uso de emuladores es útil para simular el comportamiento de la aplicación, hay que tener presente que se ejecutan en equipos cuya capacidad de procesamiento es superior a la del dispositivo. Por lo cual, es muy recomendable probarla cuanto antes en el dispositivo para el cual ha sido diseñada y así evitar comportamientos indeseados.

Enfoques para el desarrollo de aplicaciones móviles

Para el diseño de aplicaciones para dispositivos móviles se consideran cuatro enfoques principales:

- **Clientes nativos**: las aplicaciones se escriben en lenguajes de bajo nivel como C o ensamblador y compilados en un lenguaje de máquina para un grupo específico de procesadores y configuraciones hardware, y luego se ejecuta como código nativo en esos dispositivos móviles. El principal beneficio de este enfoque es la capacidad para utilizar al completo todas las características de un hardware determinado. Este enfoque se debe utilizar si el dispositivo móvil tiene un hardware especializado al que solo se puede acceder utilizando una API de C. Otro beneficio de desarrollar en clientes nativos es que se pueden ajustar los bucles, la gestión de memoria y el acceso a datos para lograr un alto nivel de rendimiento de la aplicación. Sin embargo esta potencia y control tiene un precio. Los desarrollos se hacen para dispositivos que tienen hardware similar, por tanto es necesario mantener diferentes versiones del código fuente para todos los clientes. Además, los lenguajes de bajo nivel no son tan productivos y por tanto la cantidad de código que hay que escribir para que la aplicación realice una determinada operación es bastante grande. Por ejemplo, si se utiliza C, las asignaciones de memoria se deben hacer manualmente, lo cual añade no solo complejidad y líneas de código, sino también la posibilidad de errores. Resumiendo, se puede utilizar este enfoque cuando la utilización completa del hardware y el alto desempeño son primordiales. Cuando hay que dar soporte a un gran número de clientes móviles o cuando el tiempo de desarrollo es clave, este enfoque no es la mejor opción, a menos que sea la única opción.

- **Clientes JME** (Java Platform Micro Edition, Plataforma Java Micro Edición): las aplicaciones se escriben en Java y se compilan para ejecutarse contra una máquina virtual Java (JVM), diseñada específicamente para computadoras de mano y clientes móviles. Este enfoque proporciona dos beneficios principales: tiempo de desarrollo rápido y la posibilidad de utilizar el mismo código base en un gran número de dispositivos. La cantidad de código necesario escrito en Java es por lo general menor que si se escribe en un lenguaje de bajo nivel como C. Esto se debe a que la máquina virtual se encarga de manejar automáticamente muchas de las operaciones tediosas, entre ellas el manejo de memoria. Esto significa que para realizar la misma operación se necesitan pocas líneas de código y menos errores. Además, la JVM está disponible en varios dispositivos, por lo que la aplicación cliente trabajará en diferentes dispositivos sin necesidad de mantener múltiples versiones. Sin embargo, debido a las diferencias en las implementaciones, será necesario probar la aplicación sobre cada plataforma. Los principales inconvenientes de este enfoque son el desempeño y la flexibilidad. Si la aplicación debe caber en un pequeño espacio de memoria o realizar operaciones que requieran mucha CPU (Central Processing Unit, Unidad Central de Procesamiento), entonces Java no es adecuado. Además, es posible que uno o más dispositivos no tengan la JVM. Resumiendo, si la aplicación a desarrollar tiene una interfaz estándar y no requiere acceso especial al hardware, entonces Java es una buena elección.

- **Clientes basados en web**: son similares a los clientes web estándar excepto porque se debe tener en cuenta el diseño de la página. Un cliente basado en web se ejecuta dentro del navegador web del dispositivo, desde el cual accede a la página que el servidor web envía utilizando las mismas facilidades que un cliente web de sobremesa. El principal beneficio de este enfoque es que simplifica el mantenimiento del cliente

independientemente de que las características de las versiones cambien. Sin embargo, las aplicaciones deben desarrollarse para que sean compatibles con el navegador del dispositivo. De otra parte, con la proliferación de las características de la Web 2.0, es más sencillo crear clientes más ricos y dinámicos. Los principales inconvenientes para un cliente basado en web son las características, el desempeño y el modelo de conexión. Con un cliente web, el dispositivo requiere una conexión a un servidor web para que la aplicación móvil se pueda ejecutar. Esto significa que si la aplicación cliente necesita realizar trabajo fuera de línea, que se puede procesar por lotes y enviar a los servidores centrales solo unas cuantas veces al día, el enfoque de cliente basado en la web no es la opción adecuada. Además, si la aplicación requiere una interfaz de usuario dinámica o acceso a un hardware I/O (entrada/salida) especial, es necesario considerar una implementación alternativa. En resumen, el cliente basado en web es una buena opción si el cliente software tiene una interfaz de usuario simple, que puede mantener una conexión para realizar trabajo útil.

- **Clientes basados en middleware** (software que ayuda a una aplicación a interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos): el diseño de este tipo de aplicaciones utiliza un conjunto de herramientas y tiempos de ejecución para abstraer la aplicación y las tareas de adquisición de datos lejos de cualquier dispositivo. El principal beneficio de este tipo de implementación es el desarrollo rápido y el mantenimiento del código específico del dispositivo. En el enfoque midleware se crea una aplicación utilizando un conjunto de herramientas de diseño para terceros, en un sistema operativo propietario. Además controla aspectos como lo que se muestra en pantalla, los diálogos, los datos en el lado cliente, la gestión del estado de conexión del dispositivo y la gestión de los datos fuera de línea. El enfoque midleware funciona mejor cuando la aplicación necesita acceder y cambiar datos desde un servidor central. Por lo general, se puede gestionar y desplegar aplicaciones rápidamente. El inconveniente es el enfoque potencialmente estrecho de las capacidades y la incapacidad para utilizar funciones hardware especializadas sobre los dispositivos. Las plataformas midleware generan código que no es adecuado para situaciones de alto rendimiento. Finalmente, el enfoque midleware es muy similar al enfoque JME, pero más especializado y generalmente construido en torno a las bases de datos de las aplicaciones. En conclusión, cada selección tiene sus beneficios e inconvenientes que determinan qué enfoque de desarrollo encaja mejor con las necesidades de desarrollo. Si la aplicación requiere un alto desempeño o un código para funcionalidades específicas del hardware, una aplicación nativa será la mejor opción. Si el cliente necesita ejecutarse sobre una amplia gama de dispositivos y manipulación de datos fuera de línea, el enfoque midleware es el mejor. Si es una aplicación sencilla en la que la operación puede depender del estado de conexión, entonces el cliente basado en web es una buena opción. Finalmente, si la aplicación requiere una

interfaz de usuario especializada o un código del lado cliente especializado, pero no requiere el poder de una aplicación nativa, entonces, el enfoque JME es una buena opción.

Arquitectura	Beneficios	Inconvenientes
Cliente nativo	- Aplicaciones sofisticadas y control sobre el entorno local - Capacidades multitarea sobre muchas plataformas	- El más alto nivel de esfuerzo de desarrollo - Diferente código base para diferentes dispositivos
Java ME	- El mismo código base puede soportar múltiples aplicaciones - Aplicaciones más sofisticadas	- Algunos límites de capacidad (no multitarea) - Requiere prueba y adaptación para las plataformas
Navegador web	- Desarrollo rápido - No requiere mantenimiento del código cliente - Métodos Web 2.0 disponibles - Trabajo con un amplio rango de dispositivos móviles	- Menos sensible y capas que las aplicaciones nativas
Midleware móvil	- Alto nivel de capacidad con reducido esfuerzo de desarrollo	- Tasas adicionales de licencia - Curva de aprendizaje y esfuerzo de integración potencialmente grande