SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE HERMOSILLO

INSTITUTO TECNOLÓGICO DE HERMOSILLO

DEPARTAMENTO DE SISTEMAS COMPUTACIONALES

REPORTE FINAL DE AÑO SABÁTICO

23 DE JULIO DEL 2017 A 22 DE ENERO DEL 2018

MANUAL DE PRÁCTICAS

Técnicas, Herramientas y Metodologías para Desarrollo de Aplicaciones Móviles

MARTHA ALICIA ROMERO DUEÑAS

DICTAMEN No. A5-1-89/2017

HERMOSILLO, SONORA **22 DE Enero 2018**

Í N D I C E

INTRODUCCIÓN

Es un hecho que la economía alrededor de los móviles corresponde con uno de los sectores de mayor crecimiento en la actualidad. Los dispositivos móviles están comenzando a desplazar a los equipos de cómputo tradicionales, convirtiéndolos en uno de los dispositivos preferidos no sólo para la comunicación, sino para el acceso a todo tipo de contenidos y servicios, que van desde noticias, información multimedia o entretenimiento, hasta servicios de pago, monederos electrónicos, entre muchos otros. Es por lo anterior que las aplicaciones para dichos dispositivos se vislumbran como una de las principales áreas de oportunidad en la actualidad y el futuro próximo.

Si bien este rápido crecimiento abre inmensas posibilidades, también trae retos asociados para aquellos que deberán desarrollar las aplicaciones informáticas del futuro próximo, entre los que destacan la variedad de plataformas, lenguajes de programación, sistemas de distribución y comercialización de aplicaciones, así como la cada vez mayor variedad de dispositivos, con características muy diversas que hacen del desarrollo de aplicaciones para dispositivos móviles un área de estudio bastante con bastantes vertientes.

Este manual tiene un carácter eminentemente práctico y fomenta el aprendizaje del estudiante de Ingeniería en Informática mediante la incorporación gradual de funcionalidades a una aplicación.

El objetivo de este trabajo es elaborar un Manual de prácticas que permita a los estudiantes diseñar y desarrollar de manera lógica y coherente con base en el dominio de competencias para abordar cada uno de sus elementos, donde se conocerán la evolución, entorno arquitecturas y metodología para desarrollar aplicaciones móviles

El beneficio para los estudiantes de contar con un Manual de Prácticas de Desarrollo de Aplicaciones Móviles, es que podrán tener una idea más precisa de lo que deben hacer para elaborar dichas aplicaciones. También será útil para los docentes que imparten esta materia, ya que, se podrán uniformar los criterios y las competencias para su elaboración, tomando en cuenta que es una materia que se imparte en todas las carreras de los institutos del Tecnológico Nacional de México.

El manual contiene una breve descripción de cada una de las cuatro unidades y de los temas que conforman el programa de la materia de Desarrollo de aplicaciones para dispositivos móviles, incluye ejercicios por tema y al final de cada unidad se incluyen las prácticas programadas para este manual, las cuales se resolverán en forma individual y en trabajo de equipo. En total son 12 prácticas

Cada práctica cumple con el formato establecido de 10 puntos: número de práctica, título, objetivo, introducción, correlación con los temas y subtemas del programa de estudio, material y equipo necesario, metodología, sugerencias didácticas, reporte del alumno (resultados) y bibliografía preliminar.

OBJETIVO GENERAL

El principal objetivo de esta asignatura es que al final del curso los estudiantes sean capaces de diseñar y construir aplicaciones para dispositivos móviles usando la plataforma Android. Para alcanzar este objetivo, los estudiantes tienen que desarrollar paso a paso una aplicación Android completa. Esta asignatura ayuda a adquirir las siguientes competencias de acuerdo con el currículo:

TI3. Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.

TI6. Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.

S3. Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.

CC3. Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.

Elaborar un Manual de prácticas que permita a los estudiantes de Ingeniería en Informática, con base en el dominio de competencias:

- Identificar las implicaciones actuales de la programación móvil.
- Identificar las características de los diferentes emuladores para dispositivos móviles.
- Identificar los problemas de comunicación entre sistemas.
- Utilizar técnicas de modelado para la solución de problemas.
- Aplicar la sintaxis de un lenguaje para aplicaciones móviles.
- Aplicar un lenguaje para la solución de problemas para dispositivos móviles.

Además:

1. Presentar el panorama actual de hardware y software en lo referente a dispositivos móviles.
2. Describir en sus aspectos básicos los entornos iOS y Android, así como sus aplicaciones nativas
3. Mostrar las posibilidades del uso de los dispositivos móviles en la actividad docente, tanto en lo referido a los aspectos tutoriales como académicos.
4. Dar a conocer buenas prácticas llevadas a cabo con dispositivos móviles,

5. Establecer la situación actual y las perspectivas de futuro en el uso de los dispositivos móviles y los entornos que generan en educación.

ESQUEMA DE PRÁCTICAS POR UNIDAD

UNIDAD, TEMAS Y SUBTEMAS	NÚMERO, NOMBRE Y OBJETIVO ESPECÍFICO DE CADA PRÁCTICA
UNIDAD 1. Aplicaciones móviles contra software tradicional 1.1. Requerimientos funcionales y no funcionales de las aplicaciones móviles 1.2. Tendencias en la ingeniería de software para aplicaciones móviles 1.3. Metodologías de desarrollo de aplicaciones móviles	Práctica No. 1 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES DE LAS APLICACIONES Objetivo: Para hacer una correcta identificación de los clientes y poder realizar un análisis de manera asertiva se pueden implementar una serie de técnicas de acuerdo al cliente con el que se esté tratando. Para tener una buena definición de requerimientos es necesario realizar una buena identificación de los mismos, posterior a esto es importante definirlos de manera detallada, donde se involucre la información aportada por los usuarios Para realizar una correcta definición de los requerimientos del proyecto y que éstos satisfagan las necesidades verdaderas del cliente, se deben tener en cuenta las siguientes actividades Conocerá las diferencias del desarrollo de aplicaciones móviles contra el desarrollo del software tradicional y sus coincidencias
UNIDAD 2. Patrón de Desarrollo de Dispositivos Móviles 2.1 Las tres grandes fases en la espiral del patrón de desarrollo de dispositivos móviles. 2.1.1. Educación 2.1.2. Fortalecimiento 2.1.3. Independencia (Liberación Final) 2.2.- Elementos de la espiral de patrón del desarrollo de dispositivos móviles 2.2.1.- Determinación del Alcance y Planificación 2.2.2.- Análisis y Diseño Implementación y Verificación 2.2.3.- Despliegue y Soporte 2.3.- Elección de la Metodologías de desarrollo adecuada 2.3.1.- Metodologías Ágiles 2.3.2.- Metodología en Cascada 2.3.3.- Programación Extrema 2.3.4.- Desarrollo Rápido (RAD)	**Práctica No. 2** ESQUEMA DE LAS TRES GRANDES FASES EN LA ESPIRAL DEL PATRÓN DE DESARROLLO DE DISPOSITIVOS MÓVILES EJEMPLIFICANDO CON UN PROYECTO REAL. Objetivo: El propósito es que los estudiantes tengan una idea clara de las diferencias de desarrollo de aplicaciones móviles contra el desarrollo del software tradicional y sus coincidencias Conocerá el patrón de desarrollo de dispositivos móviles.

UNIDAD 3. Software embebido 3.1. Definición 3.2. Diferencias con el software tradicional 3.3. Consideraciones de diseño impuestas por el hardware 3.4. Diseñar para todo tipo de pantallas 3.5. Baja velocidad y alta latencia 3.6. Desconexiones de red y otros problemas	Práctica No. 3 ENSAYO SOBRE SOFTWARE EMBEBIDO Objetivo: Conocerá las características principales del desarrollo del software embebido relacionado con los dispositivos móviles. Práctica No 4 CUADRO SIPNOPTICO DE LAS CONSIDERACIONES DE DISEÑO Objetivo: Conocerá las características principales del desarrollo del software embebido relacionado con los dispositivos móviles Práctica No. 5 VIDEO DOCUEMENTAL DE LOS PROCESOS EN LO QUE IMPLICA LA METODOLOGIA DE DESARROLLO ADECUADA Objetivo; Conocerá el patrón de desarrollo de dispositivos móviles**.** Práctica No. 6 DESARROLLO DE UNA APLICACION NATIVA Objetivo: Conocerá cuando debe utilizarse código nativo en lugar de código Java Describir los elementos necesarios para el desarrollo de una aplicación nativa en Android Práctica No. 7 DESARROLLO DE APLICACIONES WEB O WEBAPP Objetivo: Conocerá las fases de seguimiento, retroalimentación y actualización del desarrollo de una aplicación de dispositivos móviles**.** Práctica No. 8 DESARROLLO DE APLICACIONES HIBRIDAS Objetivo: Conocerá las características principales del desarrollo del software embebido relacionado con los dispositivos móviles Práctica No. 9 TABLA COMPARATIVO DE LAS PRÁCTICAS 6, 7, 8 Objetivo: Conocerá el patrón de desarrollo de dispositivos móviles**.**

UNIDAD 4. Seguimiento, Retroalimentación y Actualización Móviles 4.1. Diferencias de seguimiento, de un producto del mercado entre las plataformas líderes de dispositivos 4.2. Diferencias de retroalimentación de la experiencia del usuario de un producto del mercado entre las plataformas líderes de dispositivos móviles 4.3. Diferencias de actualización y políticas, de un producto del mercado entre las plataformas líderes de dispositivos móviles 4.4. Desarrollo de un proyecto de aplicaciones móviles implementando la técnicas, herramientas y metodologías.	Práctica No 10 FASES DE SEGUIMIENTO, RETROALIMENTACION Y ACTUALIZACION Objetivo: Conocerá las fases de seguimiento, retroalimentación y actualización del desarrollo de una aplicación de dispositivos móviles. Práctica No 11 DESARROLLO DE UNA APLICACICON MOVIL QUE INTEGRE BASES DE DATOS Objetivo: Conocerá las fases de seguimiento, retroalimentación y actualización del desarrollo de una aplicación de dispositivos móviles PRÁCTICA #12 PROYECTO INTEGRADOR Objetivo: Comprender y aplicar nuevas formas de desarrollo de software para valorar: individuos, clientes, software, interacciones y respuestas al cambio.

Práctica No. 1

Nombre: REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES DE LAS APLICACIONES

Competencia(s) a desarrollar.

Para hacer una correcta identificación de los clientes y poder realizar un análisis de manera asertiva se pueden implementar una serie de técnicas de acuerdo al cliente con el que se esté tratando.

Para tener una buena definición de requerimientos es necesario realizar una buena identificación de los mismos, posterior a esto es importante definirlos de manera detallada, donde se involucre la información aportada por los usuarios

Para realizar una correcta definición de los requerimientos del proyecto y que éstos satisfagan las necesidades verdaderas del cliente, se deben tener en cuenta las siguientes actividades

Introducción

Si los requerimientos se enfocan a describir las necesidades del cliente, entonces es lógico que para recabarlos haya que obtener la información de primera mano. Esto es, mediante entrevistas con el cliente u obteniendo la documentación que describa la manera que el cliente desea como funcione el sistema de software. Las necesidades y / o requerimientos del cliente evolucionan con el tiempo y cada cambio involucra un costo. Por eso es necesario tener archivada una copia de la documentación original del cliente, así como cada revisión o cambio que se haga a esta documentación.

Para que la metodología sea efectiva en los puntos descritos se definieron las siguientes actividades que se deben desarrollar para la correcta identificación de necesidades de los clientes:

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionada con los temas de la unidad 1

Material y equipo necesario

Acceso a Internet
Laboratorio de Computo´

Metodología

Requerimiento: Describe los servicios que deben ofrecer la aplicación y sus restricciones, según el proyecto que vaya a presentar en el formato de proyecto genérico deberá identificar los requerimientos funcionales y no funcionales. Reconocerlos es muy sencillo.

Un **requerimiento funcional** define lo que esperamos que deba hacer una app. Aquí se describe la interacción entre la app y el entorno se detallan los servicios o funciones que proveerá la aplicación por ejemplo: Un requerimiento funcional que detectamos con nuestro cliente *ALGO QUE NECESITAMOS ES QUE LA APP DEBERA GENERAR LSO REPORTES DE LOS LUGARES CON MAYOR RIESGO DE ACCIDENTE*

Un **requerimiento no-funcional** define como debe ser la app, describe restricciones que limitan las selecciones para construir una solución por ende son atributos relacionados con la calidad como rendimiento, escalabilidad, viabilidad, disponibilidad, mantenimiento, seguridad, etc. Un ejemplo seria el lenguaje de programación debe ser java, alta velocidad de procesamiento de datos o una interfaz gráfica de fácil lectura.

A la hora de obtener los requerimientos es muy importante tener en cuenta estos factores:

- Revisar las necesidades del os clientes, usuarios y otros interesados
- Revisar la situación actual de la app
- Revisar la organización actual
- Conocer la versión actual del sistema
- Entrevistar desarrolladores de varias versiones anteriores
- Revisar documentos existentes y sus antecedentes entre otros

Descripción del Problema.

Se requiere una app:

- Para vender música a través de Internet,
- Los usuarios compraran a crédito para adquirir canciones

- Los usuarios buscaran las canciones que deseen y las pagaran con tarjetas de crédito
- Los usuarios tendrán algunos días para descargar las canciones que hayan adquirido
- Se harán ofertas generales (afectaran a todos los usuarios) y particulares (afectaran a usuarios concretos)
- La solución es una aplicación móvil

REQUERIMIENTOS FUNCIONALES:

- **Los usuarios compraran a crédito para adquirir canciones**

  El sistema debe:
  - registrar la información de los usuarios y las tarjetas de créditos que poseen
  - permitir que los usuarios registrados compren a créditos y proporciona las herramientas para que los usuarios paguen
- **Los usuarios buscaran las canciones que deseen y las pagaran con tarjetas de crédito**

  El sistema debe:

  - Almacenar información sobre las canciones que se pueden adquirir y su precio en créditos.
  - Permitir a los usuarios buscar y consultar la información sobre las canciones.
  - Permitir a un usuario adquirir una canción a cambio de una cantidad con su tarjeta de crédito.

- **Los usuarios tendrán algunos días para descargar las canciones que hayan adquirido**

  El sistema debe:

  - Almacenar las canciones adquiridas por un usuario y la fecha, para saber durante cuánto tiempo puede descargar dichas canciones.
  - Permitir descargar las canciones que un usuario ha adquirido mientras tenga tiempo.

REQUERIMIENTOS NO FUNCIONALES:

El sistema debe:

- Visualizarse y funcionar correctamente en cualquier navegador, especialmente en Internet Explorer, Chrome, Firebird, Mozilla y Nautilus.

- Cumplir las disposiciones recogidas en la Ley Orgánica de Datos Personales y en el Reglamento de medidas de seguridad
- NO debe tardar más de cinco segundos en mostrar los resultados de una búsqueda. Si se supera este plazo, el sistema detiene la búsqueda y muestra los resultados encontrados

Sugerencias didácticas.

Listar los Requerimientos funcionales y no funcionales

1. Un sistema de parqueadero requiere que por cada 10 veces que un cliente utilice el servicio le genere una descuento del 20%, pero teniendo en cuenta que ese periodo de tiempo de uso debe ser mayor a dos horas
2. El sistema de asignación de un hotel, debe permitir asignar habitación solo cuando el usuario ya este registrado, y por cliente solo podrá hacer uso de 3 habitaciones.
3.

Reporte del alumno.

Criterio para calificar	Sí	No
Identificó los Requerimientos Funcionales	1 punto	0 punto
Identificó los Requerimientos No Funcionales	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

Caro, Paola y Cruz, Juliana (2012). Recuperado de: https://sites.google.com/site/metodologiareq/capitulo-i

Gutiérrez, Javier. (2006) Seminario Solo Requisitos 2006. De los casos de uso a los casos de Prueba(PDF). Recuperado de http://www.lsi.us.es/~javierj/cursos_ficheros/02. Un ejemplo de requisitos.pdf

Práctica No. 2

Nombre: ESQUEMA DE LAS TRES GRANDES FASES EN LA ESPIRAL DEL PATRÓN DE DESARROLLO DE DISPOSITIVOS MÓVILES EJEMPLIFICANDO CON UN PROYECTO REAL.

Competencia(s) a desarrollar

Conocerá el patrón de desarrollo de dispositivos móviles.

Introducción

METODOLOGÍAS PARA EL DESARROLLO DE APLICACIONES MÓVILES

Actualmente una metodología se ha popularizado especialmente para diseñar aplicaciones móviles, recibe el nombre de Mobile-D y fue propuesta por P. Abrahamsson y su equipo de trabajo en la ciudad de Finlandia. El método se basa en dos de las metodologías ágiles más populares, la XP y la Crystal, incluye desarrollo basado en pruebas, la programación en parejas, integración continua y refactorización, así como las tareas de mejora de procesos de software; según Abrahamsson, "Mobile-D debe ser utilizado por un equipo de no más de diez desarrolladores, trabajando en conjunto para suministrar un producto listo en un plazo máximo de diez semanas".

Otra propuesta importante y actual es la de Rahimian y Ramsin, denominada HMD (Hybrid Methodology Design), que se apoya en dos diseños la cual, el desarrollo adaptativo de software (Adaptive Software Development, ASD) y el diseño de nuevos productos (New Product Development, NPD), parte del ciclo de vida tradicional (análisis, diseño, implementación, pruebas y desarrollo) e incluyen además una fase de comercialización.

Correlación con los temas y subtemas del programa de estudio

Los ejercicios que se realizan en esta práctica están directamente relacionados con los temas 2.1 de la unidad 2.

El propósito es que los estudiantes tengan una idea clara de las diferencias de desarrollo de aplicaciones móviles contra el desarrollo del software tradicional y sus coincidencias

Material y equipo necesario

Acceso a Internet

Laboratorio de Computo

Metodología

Todos los modelos de procesos están compuestos en su mayoría por distintas fases que varían, aunque ligeramente, de modelo en modelo.

Fase de definición

Planificación del proyecto de desarrollo software

Ingeniería de requisitos / Extracción de información

Análisis (estudio) de esos requisitos

Fase de desarrollo

Diseño del software

Generación del código

Pruebas del software

Fase de mantenimiento

Corrección de errores y reajustes que a veces provienen de nuevos requisitos e implican repetir las actividades de fases anteriores

La más reciente propuesta de metodologías diseñadas específicamente para aplicaciones móviles y que aún se encuentra en etapa experimental se denomina Mobile Development Process Spiral, el cual es un modelo impulsado por la usabilidad y toma como base el modelo espiral.

Mobile Development Process Spiral Este modelo utiliza como base el modelo de desarrollo en espiral, e incorpora procesos de evaluación de la usabilidad, priorizando la participación del usuario en todos los procesos del ciclo de vida de diseño, con el fin de garantizar un diseño centrado en el usuario, aun cuando se trata de un modelo de proceso orientado a proyectos grandes y costosos, ya que está destinado a ser un modelo de reducción de riesgos.

Teóricamente el proceso de desarrollo de aplicaciones móviles en espiral contempla cinco iteraciones, cada una de ellas contiene a su vez tres tareas (determinación de requisitos, diseño y prueba) y cada iteración finaliza con la planificación de la siguiente. A continuación se explica más detalladamente cada una de las iteraciones, según (8):

En la primera iteración se determinan los requisitos del sistema y se identifican usuarios, tareas y contextos en los que se utilizará la aplicación. Luego, se definen y priorizan los atributos de facilidad de uso y se identifican métricas para cada atributo; se dibuja un prototipo de la interfaz de aplicación y se realiza la prueba del prototipo, los desarrolladores podrán utilizar diferentes técnicas de usabilidad para medir el valor de cada atributo.

En la segunda iteración el equipo de desarrollo recogerá más datos y requisitos, explorará si hay más usuarios potenciales, tareas y contextos en los que se utilizará la aplicación. A continuación, los atributos de usabilidad se redefinen y son 34 priorizados, como resultado, los desarrolladores alterarán las métricas para acomodar los requisitos añadidos; en el diseño se realiza un prototipo de alta fidelidad de la interfaz y se realizan las pruebas, utilizando técnicas de usabilidad para cada atributo, la calificación se compara con los resultados de la iteración anterior.

En la tercera iteración los desarrolladores pueden identificar y priorizar los atributos de usabilidad con mayor claridad utilizando los resultados de la iteración anterior; se desarrolla el diseño de todo el sistema y se realiza la versión alfa con sus respectivas pruebas, el equipo de desarrollo compara los resultados con la calificación de la iteración anterior.

En la cuarta iteración, son tomados los resultados de la iteración anterior para identificar y dar prioridad a los atributos de facilidad de uso; se desarrolla la versión beta y se libera para su evaluación por parte del cliente.

En la quinta iteración se desarrolla el producto final; se realiza una evaluación de facilidad de uso, la calificación de cada atributo se calcula y se compara con la calificación de la fase anterior. Una alteración en el producto final se realiza sobre la base de los resultados y se libera al producto.

En la figura se expresa mejor este ciclo de desarrollo:

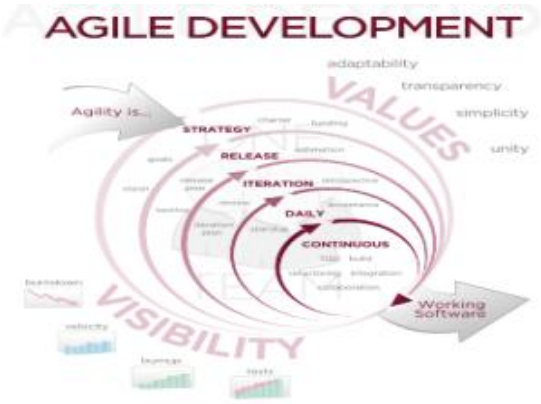


Figura Modelo Mobile Development Process Spiral

Fases de este modelo de desarrollo:

1. Planificación: Se elabora en función del estudio de riesgos de los resultados previos.
2. Análisis: Estudia los casos de uso y los escenarios a realizar. Se descubren nuevas clases y asociaciones.
3. Diseño: Se estudian las opciones necesarias para realizar la iteración. Si se necesita se retoca la arquitectura.
4. Codificación y pruebas: Se codifica el nuevo código y se integra con el resultante de iteraciones anteriores.
5. Evaluación del prototipo parcial: Los resultados se evalúan respecto a los criterios definidos para la iteración.
6. Documentación del prototipo: Se congela y documenta el conjunto de elementos del prototipo obtenido.

Ejemplo:

El Modelo Espiral es particularmente apto para el desarrollo de Sistemas Operativos (complejos); también en sistemas de altos riesgos o críticos (Ej. navegadores y controladores aeronáuticos) y en todos aquellos en que sea necesaria una fuerte gestión del proyecto y sus riesgos, técnicos o de gestión.

CICLO DE VIDA

- **PLANIFICACIÓN:** Revelación de requerimientos iniciales o luego de una iteración.
- **ANALISIS Y RIESGO:** De acuerdo con el relevamiento de requerimientos decidimos si continuamos con el desarrollo.
- **IMPLEMENTACIÓN:** Desarrollamos un prototipo basados en los requerimientos.
- **EVALUACIÓN:** El cliente evalúa el prototipo, si da su conformidad, termina el proyecto. En caso contrario, incluimos los nuevos requerimientos solicitados por el cliente en la siguiente interacción.

CONCEPTO

Es un modelo de proceso de software evolutivo. Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a priori, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior.

CONTEXTO DE LA APLICACIÓN ESPIRAL EN PROGRAMACIÓN

El modelo en espiral es un enfoque realista del desarrollo de sistemas. El modelo en espiral puede aplicarse a lo largo de la vida del software.

FASES

- **Determinar o fijar objetivos**
  - Fijar también los productos definidos a obtener: requerimientos, especificación, manual de usuario.
  - Fijar las restricciones.
  - Identificación de riesgos del proyecto y estrategias alternativas para evitarlos.
  - Hay una cosa que solo se hace una vez: planificación inicial.
- **Análisis del riesgo**
  - Se lleva a cabo el estudio de las causas de las posibles amenazas y probables eventos no deseados y los daños y consecuencias que éstas puedan producir.
- **Desarrollar, verificar y validar (probar)**
  - Tareas de la actividad propia y de prueba.

- Análisis de alternativas e identificación resolución de riesgos.

- **Planificar**
  - Revisamos todo lo hecho, evaluándolo, y con ello decidimos si continuamos con las fases siguientes y planificamos la próxima actividad.

VENTAJAS Y DESVENTAJAS

VENTAJAS

El modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software de computadora.

Como el software evoluciona a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos.

El modelo en espiral permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto.

El modelo en espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemas.

DESVENTAJAS

Resulta difícil convencer a grandes clientes de que el enfoque evolutivo es controlable.

Debido a su elevada complejidad no se aconseja utilizarlo en pequeños sistemas.

Genera mucho tiempo en el desarrollo de sistemas.

Sugerencias didácticas

1. ¿Cuáles son las fases de todo proceso de desarrollo software?

   ___ Análisis, diseño, codificación y pruebas

   ___ Especificación, iteración, desarrollo y mantenimiento

   ___ Definición, desarrollo, análisis, diseño y pruebas

   ___ Definición, desarrollo y mantenimiento

2. ¿Cuáles son los principios básicos del desarrollo en espiral?

   ___ Determinar tus opciones, valorar sus riesgos, desarrollar una cierta cantidad de trabajo y planificar la siguiente... en cada iteración

   ___ Determinar tus opciones y luego iterar valorando sus riesgos, desarrollando una cierta cantidad de trabajo y planificando la siguiente

   ___ Iterar hasta determinar tus opciones, iterar hasta valorar sus riesgos e iterar hasta desarrollar todo el trabajo, de manera planificada

   ___ Ninguna de las anteriores es correcta

Reporte del alumno (resultados)

Criterio para calificar	Sí	No
Contestó la pregunta 1 en forma correcta	1 punto	0 punto
Contestó la pregunta 2 en forma correcta	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía preliminar

Calle Marulanda, Ismael, Palacio Arias Fabian Arturo. (2014). DISEÑO E IMPLEMENTACION DE UN APLICATIVO MÓVIL PARA LA CONSULTA DE SERVICIOS POR PARTE DE LOS CLIENTES DE PEQUEÑA Y MEDIANA EMPRESA DE UNE-TELEFONICA DE PEREIRA. Trabajo presentado para optar al título de: Ingeniero Electrónico. Universidad Tecnológica de Pereira Facultad de Ingenierías, Ingeniería Electrónica Pereira 2014, http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/5135/62138456C157.pdf?sequence=1

Escribano Romero, Pedro. (2011). Creación y uso de patrones de producto en el marco del proceso de gestión de proyectos software. Descargado de e-Archivo, repositorio institucional de la Universidad Carlos III de Madrid. http://hdl.handle.net/10016/13181

Boehm, B. 1986. "Un Modelo Espiral de Desarrollo de Software y Mejora", ACM SIGSOFT Software Ingeniería Notas

Boehm B, 1988 . "Un Modelo Espiral de Desarrollo de Software y Mejora", IEEE Computer, IEEE, 21 (5): 61-72

Boehm, B, 2000. "Espiral de Desarrollo: Experiencia, principios y matices", Informe Especial CMU / SEI-2000-SR-008

Pressman, R., Ingeniería de software, un enfoque práctico, sexta edición. Editorial McGrawHill, México, año 2010 (Libro digital). Capítulo 1

Práctica No. 3

Nombre: ENSAYO SOBRE SOFTWARE EMBEBIDO

Competencia(s) a desarrollar.

Conocerá las características principales del desarrollo del software embebido relacionado con los dispositivos móviles.

Introducción

Hace varias décadas los dispositivos móviles eran exclusivos solo para un grupo selecto de personas, quienes poseían uno, como un teléfono móvil, en ese tiempo eran admirados y envidiados por el resto y ahora pensar que ya casi todos tienen un dispositivo móvil, quien no tiene un dispositivo móvil se consideraría una persona extraña, o un tanto inusual. Hace no mucho tiempo las personas podían lograr sus vidas sin los dispositivos, hoy en día tan útiles e indispensables que son que se pueden utilizar ya sea en una profesión, como lo puede ser un trabajo, un muy importante medio de comunicación, estar informado de lo que pasa alrededor de nosotros, eso y muchas más cosas hacen que los dispositivos sean un herramienta que se utilice cotidianamente.

El medio cambiante de la tecnología se esparce a todas las áreas donde el ser humano tiene que ver. El análisis que gira en torno a la aplicación de las TIC´s en la educación es un tema ya muy trabajado y no deja de ser importante en la misma medida en que las tecnologías se reinventan. La tecnología ocupa su lugar en este debate y es importante tomar consciencia al respecto ya que es una necesidad y una mejora radical a los sistemas educativos. Hoy en la actualidad se emplean los dispositivos en la educación como una herramienta de trabajo, es muy útil y fácil ya que es rápido y efectivo y pueden ampliar tu conocimiento del tema del cual se está investigando.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.

Esta práctica está directamente relacionada con la unidad 3.

Material y equipo necesario

Acceso a Internet

Laboratorio de Computo

Metodología

Los **sistemas operativos para ordenadores personales**, muchos nos hablaréis de Windows, OS X y una docena de distribuciones de Linux; si hacemos lo mismo para móviles quizás no levantéis tantos la mano, pero saldrán varios nombres: iOS, Android, Symbian, Bada…

Pero… ¿y si os preguntamos por **cajeros automáticos**? ¿O por **lavadoras** o **microondas**? Al fin y al cabo también son dispositivos electrónicos y llevan integradas ciertas instrucciones y maneras de comunicarse con el usuario para saber qué quiere hacer y cómo ofrecérselo.

Hoy vamos a hablaros acerca de los **sistemas operativos embebidos**, aquellos que controlan aparatos más limitados pero que también gozan de un "cerebro". ¿Te sonará alguno?

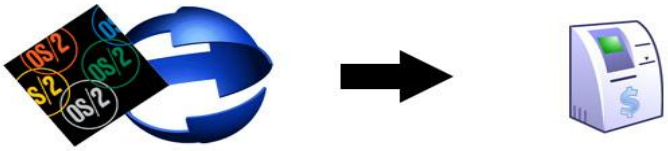
¿Qué es un sistema operativo embebido?

Por definición, un sistema operativo embebido es aquel que ha sido creado para un sistema embebido, es decir, un sistema de computación limitado a un **número fijo y escaso de tareas**.

Aquí no entran, evidentemente, los ordenadores personales ni los móviles, *tablets* o dispositivos avanzados que conoces.

Una peculiaridad de los sistemas embebidos es que son un todo con las aplicaciones que ejecutan, lo que quiere decir que en muchos casos no es posible instalar en ellos ningún tipo de software adicional.

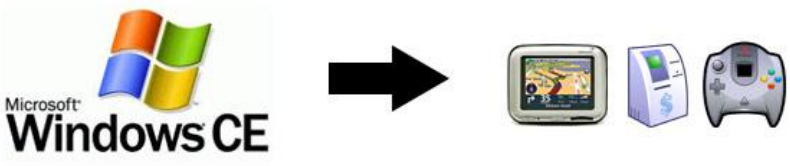
OS/2 (eComStation)



¿Dónde puedo encontrarlo? Ahora mismo está en desuso, pero en su momento podías hacerlo en cualquier cajero automático.

Desarrollado a medias entre Microsoft e IBM como posible sucesor de PC DOS, el OS/2 sufrió un batacazo comercial en favor de la versión 3.0 de Windows. No obstante, su orientación a sistemas embebidos fue muy popular durante los 90 y hoy en día aún sigue implementado en expendedores públicos en su versión original o en la derivada, de nombre eComStation.

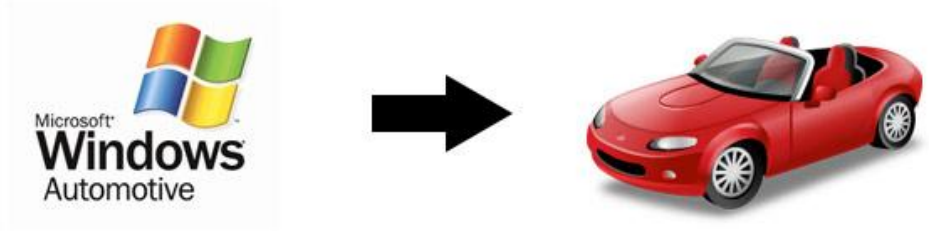
Windows CE



¿Dónde puedo encontrarlo? Cajeros automáticos (en desuso), sistemas de navegación para el coche o videoconsolas como Dreamcast.

Windows CE fue diseñado con un núcleo totalmente nuevo, distinto al del resto de sus sistemas y optimizado para dispositivos con evidentes limitaciones técnicas. Aspectos como la interfaz gráfica quedaban en un segundo plano y podían ser modificados por las empresas que hacían uso de él. Como desarrollo interno, dio lugar a sistemas operativos para Pocket PC y fue el precursor del de Zune y de Windows Phone. Actualmente ha quedado en desuso en favor de **Windows XP Embedded** y de **Windows Embedded Standard**, usado en la mayoría de cajeros y terminales públicos hoy en día.

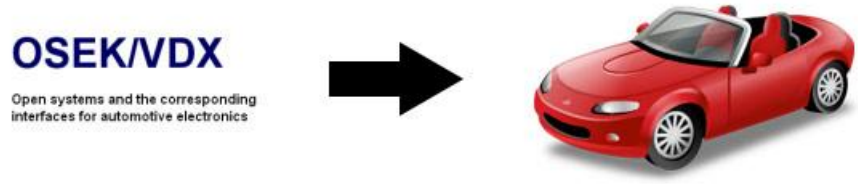
Windows Embedded Automotive



¿Dónde puedo encontrarlo? En varios coches de marcas como FIAT, Nissan o Ford.

Pensar en sistemas operativos para automóviles es cada vez más frecuente, pero a Microsoft ya le vino esta idea a mediados de los noventa, cuando comenzó a idear un sistema derivado de Windows CE que funcionaría en los paneles de navegación de todo tipo de vehículos. Hoy en día, muchos sistemas de comunicación por Bluetooth integrados (manos libres, GPS, reproducción de música…) llevan detrás esta tecnología. Su última versión se basa Windows 7.

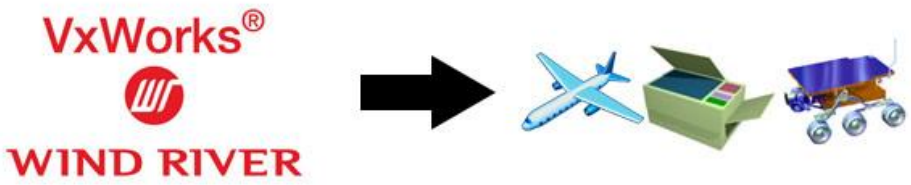
OSEK



¿Dónde puedo encontrarlo? En los automóviles de las constructoras que forman parte de este consorcio, como BMW, Chrysler, Opel o Renault.

Las siglas de OSEK hacen referencia tanto a un consorcio de empresas como a un estándar abierto de sistema operativo e interfaz de comunicaciones básicos que rige más de la mitad de la industria del automóvil. La portabilidad de OSEK hace posible que pueda ser llevado incluso a sistemas con un microprocesador de 8 bits.

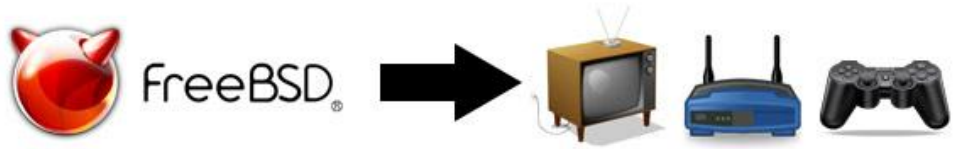
vxWorks



¿Dónde puedo encontrarlo? En una gran variedad de dispositivos: desde aviones a fotocopiadoras; desde navegadores GPS a routers.

Desarrollado por la empresa Wind River Systems, este sistema operativo en tiempo real (RTOS) ha sido llevado a infinidad de dispositivos. Incluso ha controlado el cerebro de varios vehículos espaciales como el *Sojourner*, el rastreador enviado a Marte, convirtiéndose en el SO que más lejos ha viajado en la historia de la Informática.

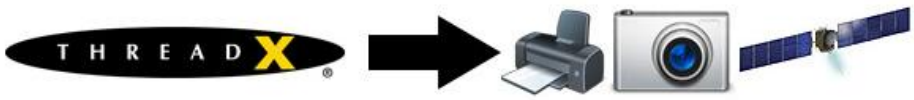
FreeBSD



¿Dónde puedo encontrarlo? Televisores, routers y sistemas de seguridad entre otros.

Aunque FreeBSD es un sistema operativo completo, son varios los proyectos que han derivado para portarse a sistemas embebidos. Como curiosidad, cabe destacar que es la base sobre la que se apoya CellOS, el sistema que rige Playstation 3 y su Cross Media Bar (XMB)

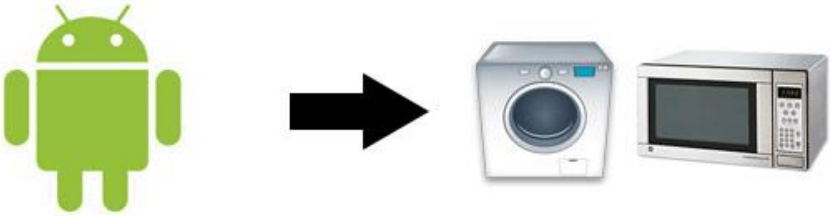
ThreadX



¿Dónde puedo encontrarlo? En impresoras, cámaras digitales, módems y sondas espaciales.

Como vxWorks, es un sistema operativo en tiempo real, es decir, que ha sido diseñado específicamente para trabajar en condiciones de rápida respuesta. Una de las empresas que ha apostado recientemente por su uso es HP, que lo incluye para gestionar la mayoría de modelos de sus impresoras de tinta y láser.

Android



¿Dónde puedo encontrarlo? Además de en los móviles… en microondas y lavadoras.

Por extraño que parezca, la aparición de Android en dispositivos poco comunes es un paso natural en su evolución como sistema operativo libre. La empresa Touch Revolution desarrolló hace unos meses el panel Nimble NIM1000, que puede incrustarse en todo tipo de electrodomésticos: desde teléfonos fijos a microondas y lavadoras. Así podrás saber qué tiempo hace antes de poner la colada.

Sugerencias didácticas.

Elaborará un ensayo donde se analicen las problemáticas de baja velocidad, alta latencia, desconexiones de red y otros problemas y propondrá soluciones e implicaciones en la aplicación a desarrollar.

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Analizó las problemática de baja velocidad	1 punto	0 punto
Analizó las problemática de alta latencia	1 punto	0 punto
Analizó las problemática de desconexiones de red	1 punto	0 punto
Propone soluciones en las aplicaciones a desarrollar	1 punto	0 punto
Propone implicaciones en las aplicaciones a desarrollar	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

Bueno, Abel. (2011). **Así son los sistemas operativos de cajeros, automóviles…** https://www.softonic.com/articulos/sistemas-embebidos-y-usos-cotidianos

Perspectiva de los sistemas embebidos 2013. ftp://ftp.ni.com/pub/branches/latam/outlook/12775_ESO_Outlook_IA.pdf

Práctica No. 4

Nombre: CUADRO SIPNOPTICO DE LAS CONSIDERACIONES DE DISEÑO

Competencia(s) a desarrollar.

Conocerá las características principales del desarrollo del software embebido relacionado con los dispositivos móviles**.**

Introducción

El factor de la “movilidad” ya es un elemento prioritario. Los expertos no olvidan que el futuro del Marketing pasa obligatoriamente por **asistir y alimentar los canales incipientes de recepción de datos.** Sus ponencias se han enriquecido con la llegada de los dispositivos móviles y lo harán todavía más cuando el llamado “internet de las cosas” (neveras, cepillos de dientes, zapatos, relojes inteligentes…) forme parte de la rutina diaria de los consumidores, como ya lo hacen los smartphones y las tabletas. Aún es pronto para asegurar que las estrategias de Marketing destinadas a los dispositivos móviles están bien consolidadas entre la clase gerente y los núcleos de toma de decisión de las empresas. Aunque crece anualmente, apenas una minoría sí tiene en cuenta los elementos de geolocalización (detectores de la cercanía o lejanía de nuestros clientes potenciales) o el diseño web adaptativo (también llamado responsive). Y nos centramos en esta última funcionalidad para realizarnos la siguiente pregunta…

¿Qué es el Diseño Web Responsive y por qué tu web debería tenerlo?

Las ventajas son múltiples. Brevemente, el Diseño Web Responsivo adapta la visualización del contenido web a cualquier resolución de pantalla y garantiza la correcta visualización tanto en ordenadores de sobremesa y portátiles como en dispositivos móviles. Es tarea de los programadores y maquetadores web que la página construida “en responsive” mute en función del dispositivo que se dispone a abrirla. **Y dicha “mutación” consiste en conseguir que los módulos de la web de escritorio se reordenen automáticamente**, teniendo en cuenta que las dimensiones en mobile son más reducidas.

El diseño se gestiona y optimiza conservando el look&feel de la compañía, pero, ahora, se trata de sacar el máximo partido al escaparate que (no lo olvide) constituye su página web a ojos de sus clientes. Sobre todo, teniendo en cuenta que, según un estudio de Comscore, la venta de smartphones y tabletas superará al de computadores en los próximos años.

Y, como queda claro que el futuro avanza hacia esta clase de dispositivos, apréndase las siguientes ventajas. Le ayudarán a introducir el responsive de manera avanzada para optimizar la estrategia de Marketing que demanda la sociedad tecnológica.

- **Mejor UX (User Experience)**. La experiencia del usuario repercute directamente en la experiencia de navegación del consumidor online. Conseguir clientes satisfechos es una de las máximas principales para fidelizar y crear vínculos afectivos entre la empresa y sus compradores. Al hablar de "experiencia móvil", es imprescindible que los usuarios accedan de manera clara y directa a los contenidos de su página web. Evitar que utilicen sus dedos para visualizarlos o acceder a través de zoom a determinadas funcionalidades mejora la comodidad de navegación.
- **Mejora la percepción de la empresa**: dejar patente que ha destinado los medios y recursos necesarios para presentarse ante los consumidores de manera simplificada, pero sin perder su esencia, deja patente una cosa: la compañía se ha preocupado por gestionar su imagen y construir la marca en todos los canales donde se hace visible. Tenga en cuenta que los usuarios adquieren cada día más nociones tecnológicas. El acervo de la cultura web crece socialmente.
- **Evita la duplicidad de contenidos**: Google penaliza gravemente a las webs que se nutren de idénticos contenidos. Estos deben ser únicos, exclusivos y novedosos para que cada URL aspire a posicionarse de manera destacada. El Diseño Web Responsivo ataja de raíz este problema, ya que realiza en una única programación la construcción web para todos los dispositivos capaces de abrirla, contemplando las complejidades de cada caso.
- **Mejor SEO y notoriedad**: al hilo del anterior, en principal buscador de internet detecta y considera positivamente los sitios web realizados "en responsive". Introduce, así, un elemento que le hará más fácil escalar hacia los primeros resultados de búsqueda en Google.
- **Mayor presencia en Redes Sociales**: hay estudios que cifran en que un 70% del acceso a perfiles sociales se realiza desde dispositivos móviles. Adaptar una web a smartphones y tabletas, otorgando un lugar destacado a las Redes Sociales, hará más fácil y natural que sus contenidos se compartan y viralicen. Cuente con que una aplastante mayoría de sus clientes ya tienen abiertas sus Redes Sociales en el momento de visitar su página web.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionada con los temas de la unidad 3

Material y equipo necesario

Acceso a Internet

Laboratorio de Computo

Metodología

El diseño en una aplicación web es muy importante para obtener buenos resultado. El usuario no debe tener la impresión de que es una pagina web.

Si quieres desarrollar una aplicación web para Android o rediseñar una existente, hay que tener en cuenta cómo se van a mostrar en diferentes tipos de pantallas. Cuando diseñamos páginas web para dispositivos móviles lo haremos con HTML5. Soporta funcionalidades creadas específicamente para adaptar contenido web a dispositivos móviles. Son dos los factores fundamentales a tener en cuenta el tamaño del área de visualizaciones (*viewport*) y el escalado y la densidad de pantalla del dispositivo.

Área de visualización y escalado

Área de visualización (*viewport*) es el área en el cual una página web es mostrada. Aunque el área visible del *viewport* coincide con el tamaño de la pantalla *viewport* tiene sus propias dimensiones que determinan el número de pixeles disponible para la página. Es decir, los móviles hacen un escalado para mostrar más pixeles de los reales para que una página web diseñada para escritorio se vea completa. El ancho del *viewport* por defecto en Andriod es de 800 px.

Por eso cuando abrimos una página web con el navegador de un dispositivo Android, sin o está adaptado, lo veremos más pequeño y encajonada en pantalla

Viewport por defecto 800 px

En la figura anterior vemos una página web con una imagen de ancho de 320 px donde nos se ha establecido ninguna propiedad *viewport,* por lo tanto el ancho de *viewport* es de 800 px.

Podemos controlar el *viewport* usando la propiedad *viewport* en HTML5 con una etiqueta *<meta name= "viewport">.* Debe ser emplazada dentro de *<head>* en el documento HTML. Podemos establecer *viewport* en HTML5 con una etiqueta *viewport* múltiples propiedades a mediante el atributo *content*. Cada par propiedad-valor en el *content* debe ser separa da por una coma.

La siguiente sintaxis muestra todas las prpi9edadeaes soportadas por *viewport* y los tipos general de valores aceptados por cada uno:

```
<meta name="viewport" content="
    width = [pixels | device-width ],
    height = [pixels | device-height],
    initial-scale = float,
    minimum-scale = float,
    maximum-scale = float,
    user-scalable = [yes | no]
    target-densitydpi = [dpi_value | device-dpi| high-dpi | medium-dpi
| low-dpi]
">
```

Podemos establecer el ancho *(width)* y el alto (*height*) del *viewport* en un tamaño concreto en pixeles. Sera el ancho que dispondrá la pagina web para ser mostrada. Por ejemplo, si establecemos el ancho del *viewport* de la pagina web de la figura siguiente en 400 px

```
<meta name="viewport" content="width = 400”/>
```

Viewport = 400 px

En la figura anterior vemos la página web con un *viewport* de 400 pixeles y la imagen tiene un ancho de 320 pixeles.

Para el ancho, los valores mayores de 10,000 y menor de 320 son ignorados. Para el alto, valores mayores de 10,000 y menores de 200 son ignorados.

Como alternativa a especificar el ancho y el alto de *viewport* tenemos la posibilidad de establecer que su valor encaja con el tamaño de la pantalla con los valores *device-width* y *device-height*. Es apropiado utilizarlo cuando desarrollamos una aplicación web con un ancho fluido (ancho no fijo). Utilícelo si desea que la pagina se visualice como si tuviera un ancho fijo. Se adapta perfectamente a todas las pantallas con si el ancho estuviera configurado para que coincida con daca pantalla.

Viewport = device-width

En la figura anterior vemos la página web con una imagen de 320 pixeles que se adapta al ancho de la pantalla del dispositivo ya que hemos establecido el *viewport* de la siguiente forma:

```
<meta name="viewport" content="width = device-width”/>
```

Tiene que tener en cuenta que la imagen se ha escalado para que coincida con la anchura de la pantalla del dispositivo si la pantalla no es de densidad media.

Escalado

La escala *viewport* del define el nivel del zoom aplicado a la página web. Podemos especificar la escala en las propiedades del *viewport* con:

initial-scale: Escala inicial de la pagina. Un valor tipo *float* que indica un multiplicado del tamaño de la pagina web con respecto al tamaño de la pantalla. Por ejemplo, si establecemos la escala inicial en 1.0 entonces la pagina web es mostrada para encajar con la resolución de la densidad objetivo 1:1. Si se establece a 2.0 la página se agranda con un factor de 2.

minimum-scale: La escala mínima permitida. El valor que indica el multiplicador mínimo para el tamaño de la página web.

maximum-scale: La escala máxima permitida. El valor que indica el multiplicador máximo para el tamaño de la página web relativo al tamaño de la pantalla.

user-scalable: Indica si el usuario puede cambiar la escala de la pagina zoom al usuario, o no, para no permitirlo. Por defecto su valor es *yes*. Si se establece a no, entonces *minimum-scale* y *maximum-scale* son ignorados.

Los tangos de valores de las escalas deben estar comprendidos entre 0.01 y 10

Densidad de pantalla del dispositivo.

La densidad de pantalla de un dispositivo es la relación entre la resolución de la pantalla y su tamaño. Definida como el número de puntos por pulgadas (*dpi*). Una pantalla con baja densidad es la que tiene disponibles más menos pixeles por pulgada; sin embargo, una pantalla de alta densidad tiene más pixeles por pulgada. La densidad de una pantalla es importante porque entre otras cosas, un elemento

del interfaz de usuario (como un botón) cuya altura y anchura son definida en pixeles aparecerá mas grande en una pantalla de baja densidad y más pequeño en un de alta densidad. Para simplificar, Android engloba todas las densidades de pantalla en tres densidades generala: alta, media y baja.

Por defecto, el navegador de Android y *WebView* escalan las pagina web como si esta se mostraran en una pantalla de densidad media. Así, aplica un escalado de 1.5x en ya una pantall de alta densidad (porque los pixeles son más pequeños) y un escalado 0.75x en una pantalla de baja densidad (porque los pixeles son mayores).

Controlar la densidad de patnalla mediante HTML

Utiliza propiedad *target-densitydpi* de *viewport* para especificar la densidad para la cual la página web ha sido diseñada. Usa los siguientes valores:

device-dpi: Usa la alta densidad nativo del dispositivo. Pordefecto nunca ocurre el escaleado.

high-dpi: Usa la alta densidad como densidad destino. Las pantallas densidad media y baja se reducen proporcionalmente tanto como sea apropiado.

medium-dpi: Usa la densidad media como densidad destino. Las pantallas de alta densidad aumentan el escalado y las de baja lo reducen. Este es el comportamiento por defecto.

low-dpi: Usa la baja densidad como densidad destino. Las pantallas densidad media y alta aumentan el escalado tanto como sea apropiado.

Especifica un valor para usar densidad destino. Los valores aceptados son 70-400.

Ejemplo de una etiqueta meta que específica la densidad objetivo y el ancho:

```
<meta name="viewport" content="wtarget-densitydpi = device-dpi, width=
device-width”/>
```

Viewport = device-width y *target-densitydpi= device-dpi*

En la figura anterior vemos como se ve la imagen con la densidad del dispositivo y el ancho del dispositivo.

Sugerencias didácticas.

Realizar un cuadro sinóptico en el cual enumerará las consideraciones de diseño impuestas por el hardware y las consideraciones de diseño para todo tipo de pantallas.

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Enumeró al menos 3 consideraciones del diseño del HW	1 punto	0 punto
Enumeró al menos 3 consideraciones del diseño para todo tipo de pantalla	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

Tomas, Jesús; Carbonell, Vicente; Garcia, Miguel; Vogt, Carsten; Bataller, Jordi. (2016). El gran libro de Android Avanzado. 3ª edición. Capitulo 6. Editorial Alfaomega

Práctica No. 5

Nombre: VIDEO DOCUEMENTAL DE LOS PROCESOS EN LO QUE IMPLICA LA METODOLOGIA DE DESARROLLO ADECUADA

Competencia(s) a desarrollar.

Conocerá el patrón de desarrollo de dispositivos móviles**.**

Introducción

Tipos de aplicaciones según su desarrollo

A nivel de programación, existen varias formas de desarrollar una aplicación. Cada una de ellas tiene diferentes características y limitaciones, especialmente desde el punto de vista técnico.

Aunque a primera vista esto no parezca incumbencia del diseñador, la realidad es que el tipo de aplicación que se elija, condicionará el diseño visual y la interacción.

. APLICACIONES NATIVAS

Las aplicaciones nativas son aquellas que han sido desarrolladas con el software que ofrece cada sistema operativo a los programadores, llamado genéricamente *Software Development Kit* o SDK. Así, Android, iOS y Windows Phone tienen uno diferente y las aplicaciones nativas se diseñan y programan específicamente para cada plataforma, en el lenguaje utilizado por el SDK.

Este tipo de apps se descarga e instala desde las tiendas de aplicaciones —con ciertas excepciones en el caso de Android, que veremos en el capítulo «Lanzando la app»— sacando buen partido de las diferentes herramientas de promoción y marketing de cada una de ellas.

Las aplicaciones nativas se actualizan frecuentemente y en esos casos, el usuario debe volver a descargarlas para obtener la última versión, que a veces corrige errores o añade mejoras.

Una característica generalmente menospreciada de las apps nativas, es que pueden hacer uso de las notificaciones del sistema operativo para mostrar avisos importantes al usuario, aun cuando no se esté usando la aplicación, como los mensajes de Whatsapp, por ejemplo.

Las aplicaciones nativas permiten aprovechar el sistema de notificaciones.

Además, no requieren Internet para funcionar, por lo que ofrecen una experiencia de uso más fluida y están realmente integradas al teléfono, lo cual les permite utilizar todas las características de hardware del terminal, como la cámara y los sensores (GPS, acelerómetro, giróscopo, entre otros).

A nivel de diseño, esta clase de aplicaciones tiene una interfaz basada en las guías de cada sistema operativo, logrando mayor coherencia y consistencia con el resto de aplicaciones y con el propio SO. Esto favorece la usabilidad y beneficia directamente al usuario que encuentra interfaces familiares.

APLICACIONES WEB

La base de programación de las aplicaciones web —también llamadas *webapps*— es el HTML, conjuntamente con JavaScript y CSS, herramientas ya conocidas para los programadores web.

En este caso no se emplea un SDK, lo cual permite programar de forma independiente al sistema operativo en el cual se usará la aplicación. Por eso, estas aplicaciones pueden ser fácilmente utilizadas en diferentes plataformas sin mayores inconvenientes y sin necesidad de desarrollar un código diferente para cada caso particular.

Las aplicaciones web no necesitan instalarse, ya que se visualizan usando el navegador del teléfono como un sitio web normal. Por esta misma razón, no se distribuyen en una tienda de aplicaciones, sino que se comercializan y promocionan de forma independiente.

Al tratarse de aplicaciones que funcionan sobre la web, no es necesario que el usuario reciba actualizaciones, ya que siempre va a estar viendo la última versión. Pero, a diferencia de las apps nativas, requieren de una conexión a Internet para funcionar correctamente.

Facebook cuenta tanto con una webapp como con una app nativa.

Adicionalmente, tienen algunas restricciones e inconvenientes en factores importantes como gestión de memoria y no permiten aprovechar al máximo la potencia de los diferentes componentes de hardware del teléfono.

Las aplicaciones web suelen tener una interfaz más genérica e independiente de la apariencia del sistema operativo, por lo que la experiencia de identificación del usuario con los elementos de navegación e interacción, suele ser menor que en el caso de las nativas.

APLICACIONES HÍBRIDAS

Este tipo de aplicaciones es una especie de combinación entre las dos anteriores. La forma de desarrollarlas es parecida a la de una aplicación web —usando HTML, CSS y JavaScript—, y una vez que la aplicación está terminada, se compila o empaqueta de forma tal, que el resultado final es como si se tratara de una aplicación nativa.

Esto permite casi con un mismo código obtener diferentes aplicaciones, por ejemplo, para Android y iOS, y distribuirlas en cada una de sus tiendas.

A diferencia de las aplicaciones web, estas permiten acceder, usando librerías, a las capacidades del teléfono, tal como lo haría una app nativa.[2]

Netflix tiene una aplicación híbrida que se ve prácticamente igual en iOS y en Android.

Las aplicaciones híbridas, también tienen un diseño visual que no se identifica en gran medida con el del sistema operativo. Sin embargo, hay formas de usar controles y botones nativos de cada plataforma para apegarse más a la estética propia de cada una.

Existen algunas herramientas para desarrollar este tipo de aplicaciones. Apache Cordova es una de las más populares, pero hay otras, como Icenium, que tienen la misma finalidad.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.

Esta práctica está directamente relacionada con los temas de la unidad 3

Material y equipo necesario

Acceso a Internet

Laboratorio de Computo

Metodología

¿CUÁL DEBERÍAS USAR?

Dadas las características de cada una de las aplicaciones, decidirse por una u otra estará determinada por unos pocos factores fundamentales y por la forma en que afectan finalmente la experiencia de uso. Cuando la disponibilidad de la app sin Internet, la posibilidad de usar notificaciones y el acceso a los recursos de hardware del teléfono sean importantes, una aplicación nativa será la opción más indicada.

Si ninguna de estas cosas es realmente importante para la aplicación, quizás sea más fácil diseñar una aplicación web, si es que ya se dispone del conocimiento para ello, heredado del desarrollo de sitios web. En este caso, el costo de desarrollo es más bajo y la forma de trabajar un poco más ágil.

Independientemente de esto, las aplicaciones nativas son las que ofrecen una mejor experiencia de uso y sobre todo, rendimiento. Algunas apps como Facebook o LinkedIn, que antes eran híbridas, han pasado a ser nativas por este motivo. Adicionalmente, ellas responden más a las guías de diseño de cada sistema operativo.

Sugerencias didácticas.

Realizar un video donde investigue los procesos en lo que se implique la metodología de desarrollo adecuada.

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
En el video se plasman los proceso en lo que se implique la metodología de desarrollo adecuada.	10 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

Cuello, Javier; Vittone, Jose. 2013 Diseñando apps para móviles. Capítulo 1.

Práctica No. 6

Nombre: DESARROLLO DE UNA APLICACION NATIVA

Competencia(s) a desarrollar.

Conocerá cuando debe utilizarse código nativo en lugar de código Java

Describir los elementos necesarios para el desarrollo de una aplicación nativa en Android

Introducción

Android NDK es un conjunto de herramientas que permite incorporar los componentes que hacen uso de código nativo en las aplicaciones de Andorid.

Las aplicaciones de Android se suelen ejecutar en la maquina virtual Dalvik. En NDK permite implementar parte de sus aplicaciones utilizando código nativo a partir de lenguajes de programación como son C y C++. Esto puede proporcionar beneficios para ciertas clases de aplicaciones, ya sea por la reutilización de código existente y/o por el aumento de velocidad de algunas aplicaciones. Android NDK es:

Un conjunto de herramientas para crear archivos que se utilizan para generar bibliotecas de código nativo en C y C++.

Una manera de integrar las bibliotecas nativas en un archivo de paquete de aplicaciones (.apk) que posteriormente puede ser ejecutado en dispositivos Android.

Un conjunto de cabecera y bibliotecas nativas del sistema que se apoyaran en todas las futuras versionas de la plataforma, a partir de Andorid 1.5

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.

Esta práctica están directamente relacionados con los temas unidad 3.

Material y equipo necesario

PC o Laptop, Android instalado y configurado e Internet

Instalar el kit de desarrollo de código nativos (Android NDK).

Manejar el Andorid NDK para el procesado de imágenes

Metodología

Cuando utilizar código nativo.

El desarrollo de aplicaciones en Andorid NDK no aporta beneficios en la mayoría de los casos. Como desarrollador, necesitamos encontrar el equilibrio entre beneficios e inconveniente, que el uso de código nativo no se traducen en un aumento de rendimientos automático debido a la mejora en la partencia

de los procesadores y la buena integración del API de Andorid; pero siempre aumenta la complejidad de la aplicación. En general, solo se debe utilizar código nativo si es esencial para su aplicación no solo porque se prefiera programar en C o C++.

Ejemplos típicos y buenos candidatos para el desarrollo mediante Android NDK son aplicaciones con un uso intensivo de CPU, como el procesamiento de la señal, la simulación de la física, etc. El simple hecho de recodificar un método para ejecutarlos en C no siempre da lugar a una gran aumento de rendimiento. Para saber si debemos desarrollar una aplicación en de código nativo o no, lo primero que tenemos que hacer es evaluar las necesidades u comprobar si la API de Andorid tenemos que hacer es evaluar las necesidades y comprobar si la API de Android puede proporcionarnos la funcionalidad que necesitamos. Si es así, lo mejor será seguir la API en caso contrario, podemos ir pensando en desarrollar parte de nuestra aplicación en código nativo. Android NDK, sin embargo, puede ser una forma eficaz de volver a utilizar un gran cuerpo de código existente en C/C++

Existen dos maneras de utilizar el código nativo en Android:

Desarrollar nuestra aplicación utilizando el entorno y SDK en Android, y utilizar la interfaz nativa de Java (JNI) para accederá las API proporcionada por Android NDK. Esta técnica es la más utilizada ya que proporciona la comodidad de utilizar el entorno de Android; pero además presenta la opción de escribir código nativo cuando sea necesario.

Desarrollar una aplicación totalmente nativa. Si trabajamos de esta forma, las devoluciones de llamada (*callback*) de ciclo de vida tendrán que ser en código nativo. El SDK de Android proporciona la clase NativeActivity, que es una clase de conveniencia que notifica el código nativo de cualquier actividad de las devoluciones de llamada de ciclo de vida (onCreate(), OnPause(), OnResume(), etc.). Se pueden implementar los *callbacks* en el código nativo para controlar cuando se producen estos eventos.

No se puede acceder a las funciones como los servicios y proveedores de contenido de forma nativa, por lo que si desea utilizar las API o cualquier otra utilidad de Android, se deberá usar el JNI.

Sugerencias didácticas.

Diferenciación entre programas desarrollados en código nativo y/o Java

Para comparar las diferencias de rendimiento entre desarrollar ciertos algoritmos en Java y en código nativo se le recomienda que realice la siguiente práctica. Vamos a compara dos aplicaciones en Adnroid que utilizan la librería Box2D. Box2D es una librería que permite emular interacciones de objetos

físicos en un espacio de 2D. Esta magnífica librería es de código abierto y ha sido desarrollada por Erin Catto.

Descargar de Android Market la aplicación Box2d Demos. Esta aplicación está basada en la librería JBox2D, una transcripción a Java de la librería de Erin Catto. Pulsa sobre la pantalla para introducir nuevos objetos y observa cómo estos caen atraídos por la gravedad y chocan entre ellos.

Descargar del Market la aplicación *AndEngineExamples.* Selecciona la opción *Physics* y luego *Using Phisics.* Igual que antes, pulsa sobre la pantalla para introducir nuevos objetos. Esta aplicación está basada en el motor de Juegos *AndEngine* que incorpora la librería Box2D; pero esta vez compilada en código nativo.

Compara la velocidad de ejecución de ambas aplicaciones. Para ello introduce un número elevado de objetos en la primera aplicación. Observa cómo a partir de 10 objetos empieza a moverlos con lentitud. Utiliza ahora la aplicación *AndEngineExamples* y observa como la aplicación no pierde prestaciones aunque introduzca 100 objetos.

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Imprimir pantallas de los eventos del inciso a	1 punto	0 punto
Presentar pantalla del resultado de la ejecución del inciso b	1 punto	0 punto
Resultados del inciso c	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

Tomas, Jesús; Carbonell, Vicente; Garcia, Miguel; Vogt, Carsten; Bataller, Jordi. (2016). El gran libro de Android Avanzado. 3ª edición. Capitulo 7. Editorial Alfaomega

Práctica No. 7

Nombre: DESARROLLO DE UNA APLICACIÓN WEB O WEBAPP

Competencia(s) a desarrollar.

Conocerá las fases de seguimiento, retroalimentación y actualización del desarrollo de una aplicación de dispositivos móviles**.**

Introducción

El éxito de desarrollo de aplicaciones móviles es indiscutible: cada día mas personas cuentan común dispositivo móvil con conexión a internet y descubren las ventajas de su uso supone. Por otra parte, cada vez son más los desarrolladores que persiguen llegar a más consumidores, por los que plantean lanzar su aplicación en diferentes plataformas.

Una posibilidad para desarrollara aplicaciones independientes de al plataforma consiste en usar tecnología web; es decir HTML5, CSS3 Y JavaScript. Las aplicaciones creadas con estas tecnologías pueden ser ejecutadas desde gran variedad de plataformas.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.

Esta práctica está directamente relacionada con los temas de la unidad 3

Material y equipo necesario

PC o Laptop, Android instalado y configurado e Internet

Laboratorio de Computo

Metodología

Aplicaciones web optimizadas para móviles: son aplicaciones creadas utilizando principalmente tecnología de aplicaciones pueden ser incrustadas dentro de una aplicación convencional, de forma que el usuario apenas nota la diferencia con respecto a estar ejecutando una aplicación nativa.

Muchas empresas y particulares ya tienen un pagina, un juego o una aplicación empresarial publicado en la web. Pero no disponen de una aplicación para dispositivos móviles. No es necesario crear desde cero una aplicación cada tipo de dispositivos (Andriod, iOS, Windows Phone…) incluso hay veces que no es necesario ni crear una aplicación como tal. Bastara con optimizarla para ser ejecutadas por un navegador web o modulo de ejecución web. Aunque hay que resaltar que en muchos casos el usuario no tendrá que abrir el navegador web para ejecutar la aplicación. Esta podrá ser empaquetada en una aplicación convencional.

Se denomina aplicación web aquella que han sido desarrollada utilizando tecnología web; es decir: HTML5, CSS y JavaScript. En otras palabras, es una aplicación *software* que se codifica en un lenguaje soportado por los

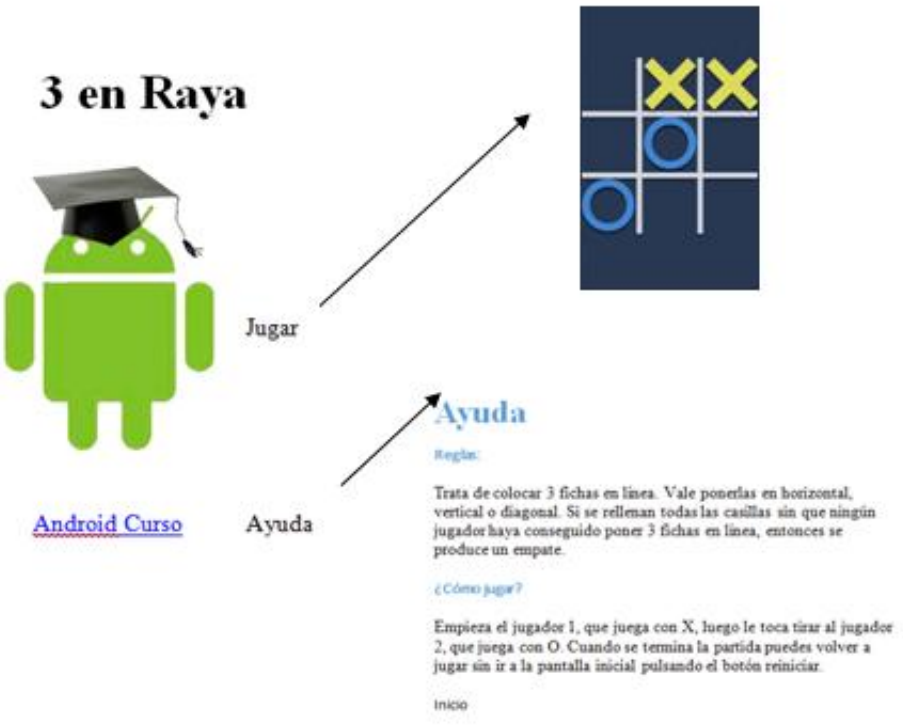
navegadores web y en la que se confía la ejecución al navegador. Esta forma de trabajar tiene la principal ventaja de facilitar la migración de aplicaciones entre sitios web y las diferentes plataformas móviles.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, a la independencia del sistema operativo, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar *software* a miles de usuarios potenciales. Existen aplicaciones como webmails, wikis, weblogs, tiendas en línea o Google Docs son ejemplos bien conocidos de aplicaciones web.

Una aplicación web suele alojarse en un servidor web. Pero también es posible alojarse de forma local en el dispositivo del cliente. Esto nos permitirá acceder a la aplicación de forma más rápida y sin tener acceso a internet. Es importante mencionar que una aplicación web debe contener elementos que permiten una comunicación activa entre el usuario y la información.

Una aplicación web de ejemplo: 3 en raya

Se trata de una aplicación sencilla formada por un archivo HTML con la estructura visual de la aplicación. Un fichero CSS para dar estilo y un fichero JavaScript para otorgar funcionalidad. El juego requiere la participación e dos jugadores humanos y controla quien ha ganado o si se ha producido un empate. Contiene tres pantallas con la siguiente estructura:



Sugerencias didácticas.

Crear una aplicación web 3 en la raya.

Crear una carpeta en tu computadora llamada 3enraya

Dentro crea un fichero de texto plano con un editor de texto; llamado *index.html* y pega el siguiente código:

```
<html>
   <head>
      <link   href="estilos.css" rel="stylesheet" type="text/css">
      <script src="funciones.js"></script>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   </head>
   <body>
      <!-- Pág. Inicio -->
      <div id="inicio" style="width:100%; position:absolute; top:0;
                    left:0; visibility:visible;">
         <div class="cabecera">
            <h1 align="center">3 en Raya</h1>
         </div>
         <div style="width:50%; margin-left:auto; margin-right:auto;">
            <table align="center">
               <tr>
                  <td><img src=
          "http://www.androidcurso.com/images/certificado_upv.jpg"></td>
                  <td><button type"button" onClick="iniciarJuego();">
                           Jugar</button></td>
               </tr>
               <tr>
                  <td align="center"><a href="http://www.androidcurso.com">
                           Android Curso</a></td>
                  <td><button type"button" onClick="mostrarAyuda();">
                           Ayuda</button></td>
               </tr>
            </table>
         </div>
      </div>
      <!-- /Pág. Inicio -->
      <!-- Pág. Partida -->
      <div id="partida" style=
       "width:100%; position:absolute; top:0; left:0; visibility:hidden;">
         <div  class="cabecera">
            <h1 align="center" id="turno">Turno Jugador 1</h1>
         </div>
         <div id="tabla" align="center">
            <table class="tabla" id="tabla">
               <tbody>
               <tr>
                  <td name="0_0" id="0_0" value="  "
          onClick="click_celda('0_0');"></td>
                  <td name="0_1" id="0_1" value="  "
          onClick="click_celda('0_1');"></td>
                  <td name="0_2" id="0_2" value="  "
          onClick="click_celda('0_2');"></td>
               </tr>
               <tr>
                  <td name="1_0" id="1_0" value="  "
          onClick="click_celda('1_0');"></td>
```

```
            onClick="click_celda('1_1');"></td>
                    <td name="1_2" id="1_2" value="  "
            onClick="click_celda('1_2');"></td>
                </tr>
                <tr>
                    <td name="2_0" id="2_0" value="  "
            onClick="click_celda('2_0');"></td>
                    <td name="2_1" id="2_1" value="  "
            onClick="click_celda('2_1');"></td>
                    <td name="2_2" id="2_2" value="  "
            onClick="click_celda('2_2');"></td>
                </tr>
                </tbody>
              </table>
            </div>
            <div>
               <button type"button"
             onClick="mostrarInicio();">Inicio</button>
               <button type"button"
             onClick="iniciarJuego();">Reiniciar</button>
            </div>
        </div>
        <!-- /Pág. Partida -->
        <!-- Pág. Ayuda -->
        <div id="ayuda" style=
         "width:100%; position:absolute; top:0; left:0; visibility:hidden;">
                <div  class="cabecera">
                <h1 id="turno" align="center">Ayuda</h1>
            </div>
            <div>
               <h3>Reglas:</h3>
               <p>Trata de colocar 3 fichas en línea. Vale ponerlas en
horizontal, vertical o diagonal. Si se rellenan todas las casilla sin que
ningún jugador haya conseguido poner 3 fichas en línea, entonces se
produce un empate.</p>
               <h3>¿Cómo jugar?</h3>
               <p>Empieza el jugador 1, que juega con X, luego le toca tirar
al jugador 2, que juega con O. Cuando se termina la partida puedes volver
a jugar sin ir a la pantalla inicial pulsando el botón reiniciar.</p>
            </div>
            <div>
              <button type"button" onClick="mostrarInicio();">Inicio</button>
            </div>
        </div>
        <!-- /Pág. Ayuda -->
    </body>
</html>
```

Crea también en la carpeta otro fichero con el editor de textos llamado *estilos.css*

```
body {
   font-family: Arial, Helvetica, sans-serif;
}
button {
   background-color:#79bbff;
   -moz-border-radius:6px;
   -webkit-border-radius:6px;
   border-radius:6px;
   border:1px solid #84bbf3;
   display:inline-block;
   color:#ffffff;
   font-family:arial;
   font-size:15px;
   font-weight:bold;
   padding:6px 24px;
   text-decoration:none;
```

```
   text-shadow:1px 1px 0px #528ecc;
}
.cabecera {
   margin-left: auto;
   margin-right: auto;
   margin-bottom: 5px;
   background-color: #006699;
   padding-top: 5px;
   padding-right: 15px;
   padding-bottom: 5px;
   padding-left: 15px;
   font-size: 12pt;
   color:#FFF;
}
#turno{
   text-align: center;
   margin-bottom: 30px;
}
.tabla {
   text-align: center;
   border-collapse: collapse;
   border: 1px solid #03476F;
}
.tabla td{
   border: 1px dotted #03476F;
   text-align: center;
}
.tdX{
   border:1px solid #000000;
   color: #369;
   text-align: center;
}
.tdO{
   border:1px solid #000000;
   color: #DF0101;
   text-align: center;
}
```

d) Por último, crea otro fichero de texto llamado *funciones.js* con el siguiente código JavaScript:

```
var tamano = 3;
var turno = "1";
var numJugadas = 0;
var finDelJuego = false;
var nombreJugador1 = "Jugador 1";
var nombreJugador2 = "Jugador 2";

function mostrarInicio() {
   document.getElementById("inicio").style.visibility = 'visible';
   document.getElementById("partida").style.visibility = 'hidden';
   document.getElementById("ayuda").style.visibility = 'hidden';
}

function mostrarPartida() {
   document.getElementById("inicio").style.visibility = 'hidden';
   document.getElementById("partida").style.visibility = 'visible';
   document.getElementById("ayuda").style.visibility = 'hidden';
}
```

```
function mostrarAyuda() {
   document.getElementById("inicio").style.visibility = 'hidden';
   document.getElementById("partida").style.visibility = 'hidden';
   document.getElementById("ayuda").style.visibility = 'visible';
}

function iniciarJuego() {
   finDelJuego = false;
   numJugadas = 0;
   turno = "1";
   document.getElementById("turno").innerHTML = "Turno " + nombreJugador1;
   iniciarTablero();
   mostrarPartida();
}

function iniciarTablero() {
   if (document.documentElement.clientWidth <
                                 document.documentElement.clientHeight) {
      ancho = document.documentElement.clientWidth / 4;
   } else {
      ancho = document.documentElement.clientHeight / 4;
   }
   for ( var i = 0; i < tamano; i++) {
      for ( var j = 0; j < tamano; j++) {
         var cell = document.getElementById(i + "_" + j);
         cell.innerHTML = "";
         cell.style.width = ancho + "px";
         cell.style.height = ancho + "px";
         cell.value = "  ";
         cell.className = "";
      }
   }
}

function click_celda(elemento) {
   var casilla = document.getElementById(elemento);
   if (finDelJuego) {
      alert("El juego ya ha terminado. Comienza uno nuevo!");
      return;
   }
   if (casilla.innerHTML != "") {
      alert("Casilla ocupada!");
      return;
   }
   numJugadas++;
   if (turno == "1") {
      casilla.className = "tdX";
      casilla.innerHTML = "X";
      casilla.style.fontSize = (ancho * 0.8) + "px";
      if (buscaGanador('X')) {
         document.getElementById("turno").innerHTML =
           "Fin del Juego: Gana " + nombreJugador1 + "!!";
         alert("Fin del Juego: Gana " + nombreJugador1 + "!!");
         finDelJuego = true;
         return;
      }
      turno = "2";
      if (numJugadas < tamano*tamano) {
         document.getElementById("turno").innerHTML =
                        "Turno "+ nombreJugador2;
      }
```

```
   } else {
      casilla.className = "td0";
      casilla.innerHTML = "0";
      casilla.style.fontSize = (ancho * 0.8) + "px";
      if (buscaGanador('0')) {
         document.getElementById("turno").innerHTML =
              "Fin del Juego: Gana " + nombreJugador2 + "!!";
         alert("Fin del Juego: Gana " + nombreJugador2+ "!!");
         finDelJuego = true;
         return;
      }
      turno = "1";
      document.getElementById("turno").innerHTML =
                 "Turno " + nombreJugador1;
   }
   if (numJugadas >= tamano*tamano) {
      document.getElementById("turno").innerHTML =
                    "Fin del Juego: EMPATE!!";
      alert("Fin del Juego: EMPATE!!");
      finDelJuego = true;
      return;
   }
}

function casilla(i, j) {
   return document.getElementById(i + '_' + j).innerHTML;
}

function buscaGanador(turno) {
   //verificamos diagonales
   if (casilla(0,0)==turno && casilla(1,1)==turno && casilla(2,2)==turno)
      return true;
   if (casilla(0,2)==turno && casilla(1,1)==turno && casilla(2,0)==turno)
      return true;
   for (n = 0; n < tamano; n++) {
      //verificamos columnas
      if (casilla(n,0)==turno && casilla(n,1)==turno &&
casilla(n,2)==turno)
         return true;
      //verificamos filas
      if (casilla(0,n)==turno && casilla(1,n)==turno &&
casilla(2,n)==turno)
         return true;
   }
}
```

Abre el archivo *index.html* con un navegador y ya puedes jugar al *3 en raya*

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Funciona el archivo del incico b	1 punto	0 punto
Funciona el archivo *estilos.css*	1 punto	0 punto
Funciona el archivo *funciones.js*	1 punto	0 punto
Funciona el archivo *index.html*	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

Tomas, Jesús; Carbonell, Vicente; Garcia, Miguel; Vogt, Carsten; Bataller, Jordi. (2016). El gran libro de Android Avanzado. 3ª edición. Capitulo 6. Editorial Alfaomega

Práctica No. 8

Nombre: DESARROLLO DE UNA APLICACIÓN HIBRIDA

Competencia(s) a desarrollar.

Conocerá las características principales del desarrollo del software embebido relacionado con los dispositivos móviles

Introducción

Si creamos una aplicación web adaptada a dispositivos móviles alojada en un servidor web, tenemos la ventaja de su fácil portabilidad. Pero también comporta que la aplicación no pueda acceder a los recursos del dispositivo notificaciones del a barra de estado. ¿Por qué no aprovechar ambas ventajas? Para eso necesitamos comunicarla parte nativa e la aplicación con la parte web.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.

Esta práctica están directamente relacionados con los temas unidad 3.

Material y equipo necesario

PC o Laptop, Android instalado y configurado e Internet

Laboratorio de computo

Metodología

Para crear una aplicación hibrida (nativa-web), hemos visto que tenemos que utilizar *WebView* para mostrar la parte web. Para realizar la comunicación debemos permitir la ejecución de JavaScript en el *WebView* estableciendo *setJavaScriptEnabled* a *true.*

La comunicación desde la parte web a la nativa, la realizaremos mediante la interfaz de comunicación. Usaremos el lenguaje Java en la parte nativa y JavaScript en la parte web. Para la comunicación desde la parte nativa a la web, utilizaremos el método *loadUrl* del *WebView.*

Modificaremos el juego 3 en Raya para que muestre los mensajes mediante el *toast* (comunicación desde *WebView* a Android) y añadiremos, al pulsar la tecla menú del dispositivo, una opción de menú que no llevara a la pantalla inicial (comunicación desde Android *WebView*).

Sugerencias didácticas.

Comunicación *WebView-Android* y visceversa

Añade el siguiente código al final de la clase *ActividadPrincipal*

```
public class InterfazComunicacion {
   Context mContext;

    InterfazComunicacion(Context c) {
      mContext = c;
   }
   @JavascriptInterface
   public void mensaje(String contenido){
      Toast.makeText(mContext, contenido, Toast.LENGTH_SHORT).show();
   }
}
```

Hemos creado la interfaz de comunicación entre Andorid y el *WebView* mediante la interfaz *InterfazComunicacion*

Declara el objeto de esta clase añadiendo el siguiente código:

```
final InterfazComunicacion miInterfazJava = new InterfazComunicacion(this);
```

Tenemos que añadir la interfaz que acabamos de crear *WebView* para ello al fina l de *onCreate* añade:

```
navegador.addJavascriptInterface(miInterfazJava, "jsInterfazNativa");
```

Nota: Para la versión Android 4.2 (API 17) y posteriores añade* @SupperssLint("JavascriptInterface") *delante de onCreate en la clase en la clase ActividadPrincipal

Para crear un menú, en la carpeta *res/menu* crea un fichero llamado *inicio.xml* con el siguiente contenido:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
   <item
      android:id="@+id/inicio"
      android:icon="@android:drawable/ic_menu_save"
      android:title="Inicio">
   </item>
</menu>
```

Ahora se tiene que hacer funcional el menú, para ello se añade al final de la clase *ActividadPrincipal:*

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   super.onCreateOptionsMenu(menu);
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.inicio, menu);
   return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.inicio:
         navegador.loadUrl("javascript:mostrarInicio()");
         break;
      }
   return true;
}
```

Por último debemos modificar las alertas del fichero funciones.js en la carpeta assets. Sustituye toda las instrucciones *alert ("------");* por *jsInterfazNativa.mensaje("…");*

Ejecutar el juego. Observe que los mensajes aparecen en un *toast.* Al pulsar la tecla Menú accedemos al menú que nos permitirá ir a la pantalla inicial. Pruébenlos desde una partida o desde la ayuda.

En esta práctica que se ha creado una interfaz de comunicación creando la clase *InterfazComunicacion*. En esta clase tenemos que definir las funciones que podrán ser llamadas desde *JavaScript* y se ejecutaran en Android. En nuestro casi, tenemos el método *mensaje()* que muestra un texto en un *Toast*.

Tamnbien puede que se necesite llamar una función JavaScript desde la parte nativa. Para ello se ha de utilizar método *loadUrl* de *WebView;* pero en vez de pasarle un URL, se le pasara *javascript:nombre_funcion().* Asi, para ir a la pantalla de inicio ejecutamos *navegador.loadUrl("javascript:mostrarInicio()").*

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Añadió **la interfaz de comunicación entre Android y el WebView**	1 punto	0 punto
Crea la carpeta res/menu	1 punto	0 punto
Funciona el juego al correrse	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

Tomas, Jesús; Carbonell, Vicente; Garcia, Miguel; Vogt, Carsten; Bataller, Jordi. (2016). El gran libro de Android Avanzado. 3ª edición. Capitulo 6. Editorial Alfaomega

Practica No.9

Nombre: TABLA COMPARATIVO DE LAS PRÁCTICAS 6, 7, 8

Competencia(s) a desarrollar.

Conocerá el patrón de desarrollo de dispositivos móviles**.**

Introducción

Si pensamos en el **desarrollo de aplicaciones para móviles**, podemos ver cómo hay una tendencia creciente a pensar que las aplicaciones Web son el futuro al que se dirige este complejo ecosistema, por ser la más sencilla de cuantas alternativas se proponen. Cuando hablemos de aplicaciones web, nos centraremos principalmente en HTML5 (aunque es solo una de las alternativas de desarrollo de éstas), por ser el estándar en boga. En este artículo intentaremos ignorar la barrera levantada entre los defensores de las aplicaciones web (Brad Hill: In Defense of HTML5) y los detarctores de las mismas (Mark Zuckerberg: Our Biggest Mistake Was Betting Too Much On HTML5); y plantearemos una comparativa que quizás nos lleve a pensar que, en este caso, la mejor solución es la más diversificada.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.

Esta práctica están directamente relacionados con los temas unidad 3.

Material y equipo necesario

PC o Laptop, Android instalado y configurado e Internet

Laboratorio de cómputo

Metodología

Aplicaciones nativas, aplicaciones web y aplicaciones híbridas

Hoy día, en función de cómo se aborda su desarrollo, se suele hablar de **tres tipos de aplicaciones móviles: nativas, web e híbridas**, que se definen como:

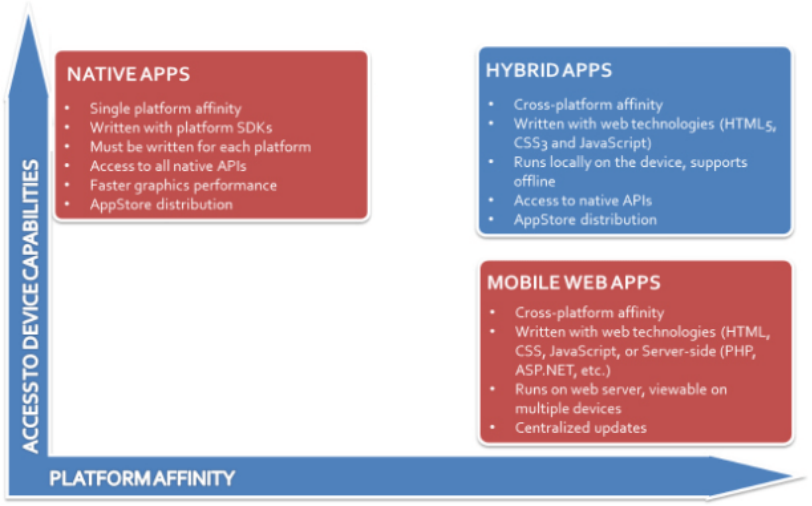
Aplicaciones nativas: aquellas que están íntegramente programadas en el entorno de desarrollo específico para cada sistema operativo.

Aplicaciones web: completamente desarrolladas en HTML 5.

Aplicaciones híbridas: aplicaciones desarrolladas en parte con el entorno de desarrollo nativo y en parte en lenguaje WEB (**HTML 5**).

Cada una de ellas tiene sus aspectos positivos y negativos que pueden y deben influir a la hora de hacer una elección para su desarrollo.

Uno de los puntos fuertes de las aplicaciones web móviles es su funcionamiento en todas las plataformas (solo se debe tener en cuenta la compatibilidad con el motor de *browser* o navegador). Una aplicación web funcionará en (casi) todos los smartphones mientras que las **aplicaciones nativas** requieren un desarrollo para cada uno de los sistemas operativos (e incluso para versiones diferentes de éstos) y las híbridas necesitan por lo menos el desarrollo del "contenedor" nativo que aloje las partes en lenguaje web. Dado que su funcionamiento se basa en recursos web, tanto las aplicaciones híbridas como las web necesitan una buena capacidad de concurrencia en los servidores en donde se albergan y una muy buena conectividad para funcionar de forma óptima.
Obviamente, esto es también uno de sus puntos críticos: algunas de estas aplicaciones no funcionan sin conexión, incluso si a priori parece que no la necesitan (lo que puede generar una cierta frustración en los usuarios). Por el contrario, la gran ventaja está en que no son necesarios unos grandes requisitos de hardware (procesador, memoria) en la parte del dispositivo móvil para poder ejecutar las aplicaciones.



Si pasamos a analizar la integración de las aplicaciones con las funcionalidades que ofrecen los móviles, nos encontramos con que los lenguajes web no tienen todavía compatibilidad con todas las funcionalidades/**APIs nativas** (tales como el GPS, acelerómetro, captura de imágenes, audio y vídeo, 3D, agenda de contactos, calendario, etc.), y eso hace que su integración a veces resulte compleja. En algunos casos, se necesita el desarrollo de un

framework para usar las APIs nativas, lo que convierte el desarrollo de la aplicación web en un proceso tan arduo como el de desarrollar una aplicación nativa. Un ejemplo de esto podemos verlo en el blog de Sencha, en el *post* "The Making of Fastbook: An HTML5 Love Story", Jamie Avins and Jacky Nguyen explican cómo fueron capaces de construir una **aplicación de Facebook en HTML5** con un rendimiento espléndido (desmintiendo las declaraciones de Mark Zuckerberg), pero a costa del desarrollo de un framework que sería muy complejo de hacer para un desarrollador que no sea muy experimentado.
Factores a considerar: complejidad de desarrollo, interfaz de usuario, canal de distribución

Todo lo anterior tiene más o menos relevancia dependiendo de la complejidad de las aplicaciones que queramos desarrollar. La gran ventaja de las aplicaciones web, y en parte de las híbridas, es que requieren una menor inversión inicial, debido a que la mayor parte del desarrollo no se debe repetir para cada sistema operativo. Y es importante resaltar que, cuando se decide hacer múltiples versiones de una aplicación, no solo hay que desarrollarlas sino también probarlas y mantenerlas, lo que puede llegar a disparar los costes. Otra ventaja de las aplicaciones web es que suelen necesitar un **menor tiempo de desarrollo** que sus equivalentes nativas, lo que puede ser crucial en el caso de que exista una ventana de oportunidad para la venta de la aplicación.
Otro aspecto que habría que analizar es la consistencia de la aplicación con la interfaz de usuario de cada sistema operativo. Si se quiere hacer una aplicación única para todos los sistemas operativos, lo que se gana en homogeneidad generalmente se pierde en la riqueza de personalización proporcionada desde la interfaz nativa. Dicho de otro modo, el desarrollador tiene que valorar si compensa tener la misma aplicación (web) en un móvil **Android, Windows Phone, Firefox OS** o iOS, a costa de que dicha aplicación suponga una ruptura con el estilo de la interfaz nativa del dispositivo y se pierdan las opciones específicas de cada plataforma.

Aunque no se trate de un aspecto técnico, también es muy importante considerar cómo es el canal de distribución para los diferentes tipos de aplicaciones, es decir, la tienda. Las aplicaciones web son la más perjudicadas en este aspecto porque su presencia en las *stores* oficiales de los sistemas operativos predominantes, iOS y Android, no está admitida. Esto comporta que su descubrimiento sea más complicado y que para conseguir que la aplicación tenga una buena visibilidad se deba realizar un esfuerzo de marketing mayor que en el caso de las aplicaciones nativas. También **las aplicaciones híbridas están en el punto de mira** de los "dueños" de estos sistemas operativos que en algunos casos, dependiendo de qué parte se haya desarrollado con lenguaje web, no dan el visto bueno para su publicación. En el caso de Firefox OS, el descubrimiento de las aplicaciones web desde el terminal está garantizado, aunque es algo específico de esta plataforma por el momento.

Empezar con aplicaciones web pero seguir una 'estrategia' híbrida

Cualquier empresa que quiera desarrollar aplicaciones para móviles tiene que valorar las **ventajas e inconvenientes de las aplicaciones web y las nativas** antes de comenzar a tirar código. Y la decisión no depende únicamente de lo fácil o difícil que sea desarrollar la propia aplicación, sino también de factores como la ventana de oportunidad, el peso que se quiera dar a la experiencia de usuario, el rendimiento, la agilidad en el mantenimiento y la capacidad de actualización de la propia aplicación o de la información contenida en ella.

Personalmente, no soy partidario de "tirar por la calle de en medio" y apostar exclusivamente por las aplicaciones híbridas ya que, si bien pueden aunar las ventajas de las aplicaciones web y nativas por separado (simplificación, homogeneidad, flexibilidad, adaptación de la interfaz de usuario) **también reúnen los inconvenientes de ambas** (fragmentación, dificultad de mantenimiento, optimización de rendimiento compleja, aumento de inversión inicial y costes). Pero quizás sea buena idea considerar una estrategia híbrida en su lugar, cuando las características del proyecto lo permitan. Por ejemplo: ¿por qué no hacer una versión inicial web que permita lanzar rápido y verificar la aceptación del mercado?. Si la etapa de desarrollo es rápida y barata, siempre es más fácil derivar parte de la inversión inicial a acciones de marketing, que redundarán en sus probabilidades de éxito. Y, si el concepto funciona, se puede **apostar por una segunda oleada de aplicaciones nativas** (o híbridas) con un menor riesgo de fracaso. Además de que sería posible volver a aprovechar la inversión ya hecha en marketing.

Sugerencias didácticas.

Realizar una tabla comparativa entre las Aplicaciones nativas, aplicaciones web y aplicaciones híbridas que se realizaron en las prácticas 6, 7 y 8

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
En la tabla se incluye Aplicaciones Nativa	1 punto	0 punto
En la tabla se incluye Aplicaciones Web	1 punto	0 punto
En la tabla se incluye Aplicaciones Hibridas	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

https://blogthinkbig.com/aplicaciones-web-nativas-hibridas

Tomas, Jesús; Carbonell, Vicente; Garcia, Miguel; Vogt, Carsten; Bataller, Jordi. (2016). El gran libro de Android Avanzado. 3ª edición. Capitulo 6. Editorial Alfaomega

Práctica No. 10

Nombre: FASES DE SEGUIMIENTO, RETROALIMENTACION Y ACTUALIZACION

Competencia(s) a desarrollar.

Conocerá las fases de seguimiento, retroalimentación y actualización del desarrollo de una aplicación de dispositivos móviles.

Introducción

Analizar las aplicaciones

Una vez que la app esté creada, el siguiente paso será el de **realizar un análisis de la aplicación**, para comprobar si se ha desarrollado correctamente y sobre todo ver si tiene algún tipo de fallo que deba ser corregido antes de lanzarla al mercado.

Para conseguirlo, hay que hacer un testeo funcional. En este testeo se analiza si todo funciona correctamente. A continuación, se realiza un testeo de rendimiento. En este apartado se estudia si la app funciona correctamente bajo diferentes condiciones de trabajo. Finalmente, se realizan otras comprobaciones como fuga de memoria, para asegurar que la misma funciona sin ningún tipo de error.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.

Esta práctica está directamente relacionada con los temas de la unidad 4

Material y equipo necesario

Acceso a Internet

Laboratorio de Computo

Metodología

Seguimiento

Esta fase comprende los procesos necesarios para realizar el seguimiento, revisión y monitorización del progreso de proyecto. Se concibe como el medio de detectar desviaciones con la máxima premura posible, para poder identificar las áreas en las que puede ser requerido un cambio en la planificación. La etapa de seguimiento y control se encuentra naturalmente asociada a la de ejecución, de la que no puede concebirse de forma separada, aunque por su importancia y valor crítico.

Configurar el seguimiento de descargas de aplicaciones móviles

Descubra cómo el tráfico que instala su aplicación encuentra su página de mercado.

En este artículo se describe la funcionalidad del SDK de Analytics. Para obtener los últimos informes de aplicaciones móviles en Analytics, utilice el SDK de Firebase..

El seguimiento de descargas le permite recopilar y enviar datos desde sus aplicaciones móviles a su cuenta de Analytics y que se muestren en el informe de Fuentes de aplicaciones móviles.

Debe llevar a cabo los pasos indicados en este artículo para configurar el seguimiento de descargas en sus aplicaciones para móviles.

Secciones de este artículo:

Paso 1: Habilitar el seguimiento de descargas de la aplicación en su cuenta

Para Android

El seguimiento de descargas está habilitado automáticamente en Analytics para las aplicaciones Android. No tiene que hacer nada.

Para iOS

Siga estos pasos para habilitar el seguimiento de descargas en su cuenta de Analytics:

Inicie sesión en Google Analytics. .

Haga clic en **Administrador** y desplácese hasta la **propiedad que desee editar**.

En la columna *PROPIEDAD*, haga clic en **Configuración de la propiedad**.

En la sección *Seguimiento de campañas para iOS*, haga clic en la opción para **activar** la función.

Haga clic en **Guardar**.

Paso 2: Actualizar el SDK de Analytics

Para Android

Los programadores de Android tienen que cambiar algunas líneas en su archivo de manifiesto. Consulte la Guía para desarrolladores de Android para ver ejemplos específicos sobre cómo hacerlo.

Para iOS

Actualice su aplicación para utilizar el SDK de Google Analytics para iOS. Para utilizar el seguimiento de descarga de aplicaciones de Analytics, su aplicación deberá tener acceso al identificador de iOS para publicidad (IDFA). Para ello, deberá agregar la biblioteca libAdIdAccess.a a su proyecto de XCode. Consulte el archivo README del SDK para obtener más información o revise la

Paso 3: Configurar campañas personalizadas

Las campañas personalizadas son una función de Analytics que agrega parámetros a la URL de su página de mercado donde los usuarios descargan su aplicación. Estos parámetros son los que indican a Analytics de qué mercados proviene su tráfico. .

Debe configurar las campañas personalizadas para cada una de las plataformas que utilice. Para que esta tarea sea más fácil, hemos creado una herramienta para ayudarle a crear sus URL, de manera que no lo tenga que hacer manualmente. Para obtener más detalles sobre cómo configurar campañas personalizadas, consulte nuestras guías para desarrolladores de cada sistema operativo:

Para Android

Referencia sobre las campañas personalizadas para Android

Herramienta de creación de URLs para Android

Para iOS

Referencia sobre las campañas personalizadas para iOS

Herramienta de creación de URLs para iOS

Pasos siguientes

Crear y administrar campañas de anuncios (en la red publicitaria que prefiera)

Tras finalizar los pasos para configurar el seguimiento de descarga de aplicaciones para móviles con Analytics, puede empezar a pensar en su estrategia y contenidos publicitarios. El modo de crear y administrar sus campañas de anuncios tendrá un impacto importante en el número de usuarios que se descargan su aplicación. Deberá decidir cuál es su audiencia objetivo, el mensaje que desea transmitir y la red publicitaria que va a utilizar.

No importa cómo responda a estas preguntas, lo fundamental es que cree URL personalizadas para cada uno de sus anuncios o campañas que lleven a su página de descarga y que incluyan las URL etiquetadas en cada anuncio. De este modo, podrá saber qué anuncio ha conseguido mejores resultados para dirigir el tráfico a su página de descarga.

Analizar sus datos con el informe Fuentes de aplicaciones para dispositivos móviles

Si finaliza los pasos que se han descrito anteriormente, sus datos se mostrarán en el informe Fuentes de aplicaciones para dispositivos móviles. Utilice este informe para descubrir lo que atrae a los usuarios a su página de descarga en cada mercado.

Probar y obtener retroalimentación del usuario

La falta de pruebas hechas por usuarios fuera de la corporación es una de las grandes fallas cuando se trata de apps móviles.

Se necesita retroalimentación de usuarios externos y determinar así los obstáculos para lograr sus objetivos, necesidades y detalles de usabilidad que pudiera haber.

Se deben hacer estas evaluaciones a través de mediciones cualitativas, como las encuestas. Además, los estudios de rastreo de la vista del usuario, además de los clicks, pueden dar *insights* de cómo los usuarios navegan y tocan en la app para identificar áreas de distracción o atracción.

Combina esos datos cuantitativos con analíticas y prueba las variaciones de la interfaz usando pruebas AB/ y pruebas multivariables del software para determinar el diseño y experiencia óptimos

Por qué se actualizan tanto las aplicaciones

¿Te has preguntado alguna vez por qué se actualizan las aplicaciones del móvil tan a menudo en los dispositivos móviles? Explicamos para qué lo hacen

¿Por qué y para que se actualizan tanto las aplicaciones? ¿Para que sirve? Estamos seguros de que muchos de vosotros os habéis hecho estas mismas pregunta en varias ocasiones.

Nuestros dispositivos nos muestran notificaciones de actualización de aplicaciones prácticamente todos los días, algo que **a veces puede llegar a resultar un tanto molesto**. Y es que dependiendo del número de aplicaciones que tengamos instaladas, podemos pasar un buen rato descargando las nuevas versiones.

Las actualizaciones ¿son necesarias?

Lo primero que debemos tener en cuenta para comprender el motivo de tanta actualización es que las aplicaciones nunca se acaban. Es decir, cuando un desarrollador lanza una aplicación lo hace con la idea de que **su desarrollo continúe toda la vida.**

Como ejemplo práctico podemos hablar de aplicaciones como Facebook, Snapchat o Twitter, las cuales siempre reciben nuevas versiones que debemos instalar. En algunos casos dichas actualizaciones **vienen acompañadas de nuevas funciones**, mientras que en algunas ocasiones no incluyen ninguna novedad aparente de cara a los ojos de los usuarios.

Lo cierto es que los desarrolladores tienen un papel bastante complicado al tener que encontrar un **equilibrio entre actualizar demasiado y no actualizar casi nunca**. La primera de las opciones puede saturar al usuario y este podría acabar cansándose, mientras que la segunda de ellas dará la sensación de que estamos ante una aplicación que está abandonada.

Instala en un clic todas las aplicaciones esenciales en tu Mac

¿Por qué es necesario actualizar mis aplicaciones?

Nos guste o no, la actualización de las aplicaciones es necesaria por **dos razones fundamentales**: incluir nuevas funciones que se adapten a las

necesidades de los usuarios y **corregir errores** que puedan surgir de la versión actual.

¿Cuál es la medida justa a la hora de lanzar actualizaciones? Todo dependerá de lo necesario que sea. Si el equipo de desarrollo **ha prestado atención a los detalles** desde el comienzo seguramente estaremos ante una aplicación mucho más solida y con menos errores que corregir.

En ocasiones, ya sea por errores en la programación o por la llegada de una nueva versión del sistema operativo, será necesario lanzar una actualización que permita a la aplicación **seguir funcionando con total normalidad.**

Webs que se convierten en aplicaciones sin instalar nada, lo último de Google

Puede que para los usuarios resulte un poco molesto recibir actualizaciones demasiado a menudo, pero es un mal necesario si queremos que todo funcione correctamente.

La actualización perfecta para los usuarios

Nunca llueve a gusto de todos y del mismo modo es imposible que los desarrolladores tengan contentos a todos los usuarios de sus aplicaciones. Pero ¿cuál sería el ciclo de actualización ideal?

En mi opinión, las aplicaciones se deberían actualizar **de manera periódica e incluir mejoras de rendimiento y estéticas**. Aquellas actualizaciones que no incluyen ninguna mejora que el usuario puede ver con sus propios ojos no son tan valoradas por los mismos.

Sugerencias didácticas.

Realizar un mapa mental de las fases de seguimiento, retroalimentación y actualización de los dispositivos móviles

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Involucró todas las fases en el mapa mental	3 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

https://support.google.com/analytics/answer/3389142?hl=es

https://marketing4ecommerce.mx/los-8-temas-de-usabilidad-a-cuidar-en-las-apps-moviles/

http://omicrono.elespanol.com/2017/02/por-que-se-actualizan-las-aplicaciones/

Práctica No. 11

Nombre: DESARROLLO DE UN PROYECTO DE APLICAICONES MOVILES

Competencia(s) a desarrollar.

Conocerá las fases de seguimiento, retroalimentación y actualización del desarrollo de una aplicación de dispositivos móviles

Introducción

Es difícil hoy en día no estar vinculada a una red social donde mantener contacto con amigos y conocidos, compartir inquietudes, aficiones, deportes, juegos y opiniones. Con sus matices, Facebook y Twitter nos ofrecen darnos a no conocer y conocer a otros. Por ello, es casi obligatorio que también los programas desarrollemos puedan usar estas redes sociales para obtener o mostrar información de quien los está utilizando.

Por Jordi Bataller Mascarell

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.

Esta práctica está directamente relacionada con los temas de la unidad 4

Material y equipo necesario

Acceso a Internet

Laboratorio de Computo

Conseguir una cuenta Facebook

Aprender a utilizar <<consola>> de gestión de aplicaciones en la red social

Dar de alta la aplicación que queremos desarrollar

Descargar y configurar las bibliotecas que nos servirán para interactuar con las redes sociales.

Configurar y programar una aplicación integrada en Facebook

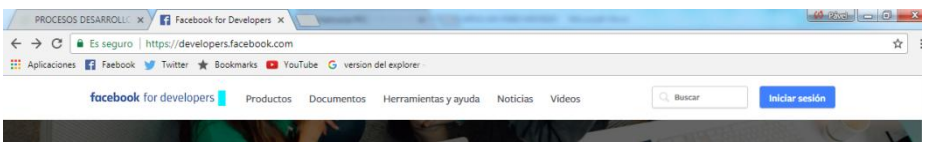
Metodología

Android y Facebook

Preliminares. En primer lugar vamos a ver cómo dar de alta nuestra aplicación, configurarla y descargar el kit de desarrollo del Facebook.

Darse de alta en Facebook como desarrollador

Para escribir aplicaciones para Facebook hay que tener un usuario en la red social, pero además hay que darse de alta como desarrollador. Accedemos a: http://developers.facebook.com



Y vamos a aplicaciones (Apps), donde empezara el proceso de registro.

En él se nos pide un número de teléfono al que se envía una clave, que utilizaremos para completar el proceso. Mas adelantes volveremos a Apps para dar alta nuestra aplicación.

SDK de Facebook para Android

En primer lugar, es necesario instalar la aplicación oficial de Facebook en el teléfono o en el emlulador que vayamos a utilizar.

En el caso de los teléfonos, muchos la tienen pre-instalada o pueden instalarla fácilmente desde "Google Play". Si vamos a utilizar un emulador, debemos conseguir el .apk: podemos hacerlo al dar de alta nuestra aplicación en la consola de desarrollo de Facebook o en https://developers.facebook.com/docs/android/downloads. Dicho .apk se puede instalar escribiendo esta orden en la consola (estando el emulador en ejecución):

adb install Facebook-35.0.0.15.243.apk

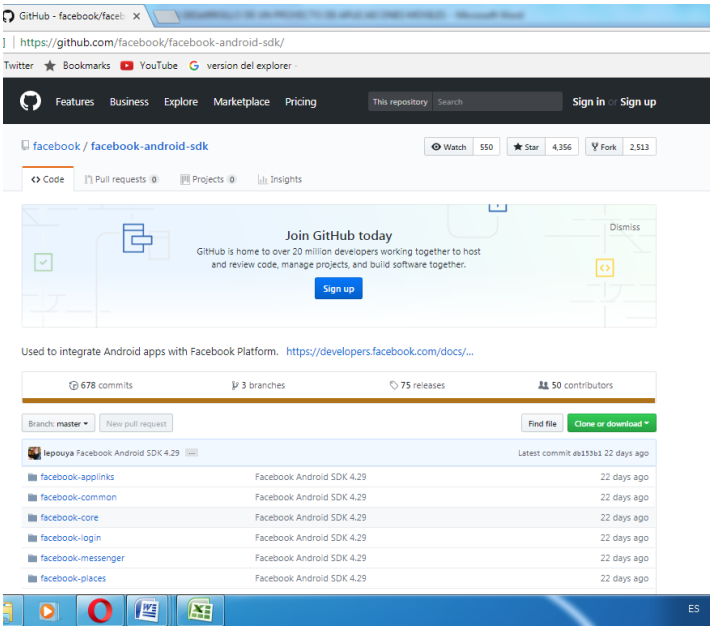
Por otro lado, Facebook proporciona una SDK (kit para desarrollo de sw) para escribir aplicaciones que interactúen con su plataforma. Por tanto, vamos a seguir la guía oficial que nos da por el propio Facebook en

https://developers.facebook.com/docs/android/getting-started/

Anteriormente, pediríamos descargar dicho SDK junto con un conjunto de ejemplos desde la página https://developers.facebook.com/docs/android/

Sin embargo recientemente dicho enlace ("Download the SDK") solo nos ofrece la biblioteca ya compilada en un fichero .aar, y no contiene ejemplos básicos sencillo.

Como decimos, los ejemplos básicos se encuentran el siguiente repositorio de github: https://github.com/facebook/facebook-android-sdk/



Desde ahí podemos configurar Android Studio apra que acceda a los ejemplos, o descargarlos en un zip mediante le botón “Download ZIP”.

Conviene probar el ejemplo “HelloFacebookSample” para ver que todo va correctamente antes de realizar nuestros propios programas. Así, abriremos ese ejemplo de donde lo hayamos descargado. Para poder utilizarlos hay que declarar nuestra clave publica de Android en “Facebook Deveopers”.

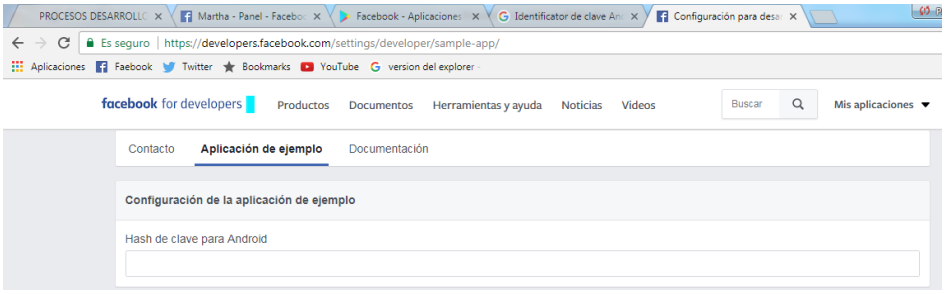
Si no se realiza esta paso, al ejecutar el ejemplo “HelloFacebookSample” veremos un toast mostrándonos el siguiente error:

> (#404) Key hash IpgCWdGg47EaDQy_a0-54rt_TU does not match any stored key hashes.

Para poder ejecutar las aplicaciones dejemplo consultaremos la clave pública que tenemos en “~/android/debug.keystore” (utilizando los comandos keytool y openssl). A continuación, nos dirigimos a la pagina

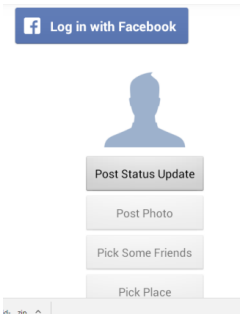
https://developers.facebook.com/settings/developer/sample-app/

Elegimos la pestaña “*Sample App*” y en “*Identificator de clave Android (Andfroid Key Hash)”* copiamos nuestra clave pública, dando a “*Guardar cambios (Save Changes)”*



En general, necesitaremos recompilar las aplicaciones de ejemplo de Facebook para que integren la clave a partir de “debug.keystore”, si la hemos cambiado.

Y ya puedes proceder a probar el ejemplo “HelloFacebookSample”



Sugerencias didácticas.

Darse de alta en Facebook developers y explora el sitio.

Descarga del github el código de las aplicaciones de ejemplos del Facebook

Ejecutar el programa “HelloFacebookSample”

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Se dió de alta en Facebook developers y explora el sitio.	1 punto	0 punto
Descargó del github el código de las aplicaciones de ejemplos del Facebook	1 punto	0 punto
Ejecutó el programa “HelloFacebookSample”	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entregó en la fecha programada	1 punto	0 punto

Bibliografía

Tomas, Jesús; Carbonell, Vicente; Garcia, Miguel; Vogt, Carsten; Bataller, Jordi. (2016). El gran libro de Android Avanzado. 3ª edición. Capitulo 8. Editorial Alfaomega