Universidad Nacional Autónoma de México

Facultad de Ingeniería

Metodología de pruebas en aplicaciones móviles

PROYECTO DE TESINA PARA OBTENER
EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

PRESENTA:
Marco Antonio Salcedo Sansón

ASESOR:
Luis Sergio Valencia Castro

México, D.F. **2014**

Índice

I. Introducción

Los paradigmas referentes al uso de la tecnología de la información están en constante cambio, nos encontramos en la etapa de transición de la era de las computadoras de escritorio a la de la tecnología móvil. Así como cambian los modos de uso, en consecuencia también deben de cambiar los métodos de desarrollo, así que el desarrollo de aplicaciones para dispositivos móviles (coloquialmente llamadas “apps”) está cobrando más importancia en estos días. Como todo proceso de desarrollo de software, al desarrollar apps eventualmente se tendrá que pasar por uno o varios períodos de pruebas, pero realizar pruebas para aplicaciones móviles es más complicado que para otro tipo de software, por razones inherentes a las plataformas. Lo que se busca en este trabajo es plantear una metodología rápida, eficaz y confiable para realizar pruebas de calidad y usabilidad en aplicaciones móviles. Para llegar a ello, se analizan los obstáculos que comúnmente se presentan en este tipo de software, su alcance y magnitud y se propondrán soluciones analíticas y caminos para llegar a una metodología robusta de pruebas.

Este trabajo tiene como objetivo ofrecer una metodología rápida, eficaz y contundente para realizar pruebas de calidad y usabilidad en aplicaciones móviles.
Primero que nada se parte de una metodología de pruebas de software estándar, propuesta por la IEEE, y con el trabajo propuesto por UNAM mobile se adapta a aplicaciones para móviles. Para lograrlo, se estudian las recurrentes dificultades que se presentan en este tipo de software, su alcance y magnitud; y se propondrán soluciones analíticas otorgando herramientas al usuario para llegar a una metodología vasta de pruebas. Debido a que actualmente es necesario contar con aplicaciones para móviles, es importante mantener los costos de desarrollo lo más bajo que se pueda para garantizar que el desarrollo de apps sea costeable para una mayor variedad de clientes.

Al ser una metodología que parte de un estándar es sencilla de incorporar en el proceso de desarrollo de cualquier aplicación, y la adaptación para el caso específico de aplicaciones para móviles que se expone en este trabajo quita algunos pasos que en el ambiente móvil no son necesarios, ahorrando tiempo valioso para el desarrollo.

Cabe mencionar que lo expuesto anteriormente, es el resultado de un proyecto elaborado por UNAM Mobile, solicitado por la empresa médica HDS. UNAM Mobile realizó el diseño y pruebas de las aplicaciones, dichas aplicaciones tenían como objetivo funcionar en las tres plataformas más importantes del mercado actual, Android, iOS y Windows Phone.

El resultado de la investigación culminó en la Metodología de pruebas para aplicaciones móviles y que más adelante se expone a detalle.

II. Marco teórico

Las pruebas son un proceso esencial para el desarrollo de software, ya que es la única forma de asegurarse de que la calidad y la experiencia de uso son todo lo que se desea para el usuario. Dentro del proceso de desarrollo de software hay uno o varios periodos de pruebas, el número de iteraciones es dictado por los resultados de las mismas pruebas, estas pruebas consisten en encontrar errores y descuidos de programación (bugs), pues es el último momento seguro para rectificar errores o el diseño de la aplicación si es el caso, también es el momento para poner a prueba el diseño de una aplicación en un entorno controlado, planteando las siguientes preguntas:

- ¿el flujo de la aplicación es congruente?
- ¿la interfaz es adecuada?
- ¿el usuario se siente cómodo usando la aplicación?

Los usuarios de pruebas van con la mentalidad de que van a toparse con una aplicación que no funcione al 100%, a diferencia de los usuarios finales, sobre todo si la aplicación será de pago, puesto que esperarán un producto de calidad y no van a sobrellevar lo contrario debido a que han pagado por un servicio.

Hasta este punto todo lo referido podría aplicarse al proceso de pruebas de cualquier tipo de aplicación, pero hay una serie de características que poseen las aplicaciones móviles que no tiene el software para PC, las cuales obligan a todo desarrollador a usar un enfoque distinto para todo el proceso de desarrollo, incluidas las pruebas. Para poder entrar en detalle es necesario contar con una definición concreta de “aplicación móvil”.

Una aplicación móvil a rasgos generales, es una pieza del software que corre en el dispositivo móvil, como puede ser un smartphone (teléfono inteligente), o una tablet. Un smartphone es un dispositivo electrónico portátil que permite a su usuario acceder a datos e información idealmente desde cualquier lugar. A diferencia de los teléfonos celulares tradicionales, que cumplen únicamente con las funciones de llamadas y mensajes SMS, los smartphones tienen un sin fin de funciones, como lectura y edición de documentos, juegos, agenda, despertador, etc. Estas funciones son expandibles,

debido a la posibilidad de instalarles aplicaciones de terceros. Otro punto importante en el que un smartphone se separa de los teléfonos tradicionales es su posibilidad de conectarse a Internet, ya sea por medio de la red del proveedor de telefonía celular o por Wifi. Los smartphones tienen las funciones de una computadora portátil, adicionales a las de un teléfono convencional. Las funciones más recurrentes de los smartphones incluyen un navegador de Internet, correo electrónico, VoIP, cámara de fotos y videos, acceso a redes sociales, reproductor de música y un amplísimo etcétera. El sistema operativo varía ya que son fabricados por diferentes empresas, los más importantes o usuales son Android, Blackberry, iOS, Symbian y Windows Phone.

Dicho lo anterior es posible dar una definición más satisfactoria de “aplicación móvil”. Una aplicación móvil, coloquialmente conocida como “app”, es una pieza de software concebida para ser ejecutada en un dispositivo móvil, como puede ser un smartphone o una tablet. Normalmente las apps son una unidad pequeña e individual de software, que tiene un uso muy particular y limitado, a diferencia del software para computadora, que suele tener implementadas muchas funcionalidades. Un buen ejemplo es un tipo de software muy común, un procesador de palabras, el Word de Microsoft está lleno de funcionalidades, algunas de las cuales son incluso muy avanzadas para un procesador de texto, como la posibilidad de insertar gráficas, en cambio los procesadores de palabras de los paquetes de oficina de Android (Officesuite, Quickoffice, Google Docs) son en general más limitados, le dan al usuario las herramientas básicas que requieren para crear documentos de texto.

Teniendo las anteriores definiciones en cuenta es posible empezar a ver algunas complicaciones que conlleva por definición el proceso de pruebas de aplicaciones móviles, la mayor de ellas radica en el hardware. Cada fabricante de este tipo de hardware soporta uno o varios sistemas operativos, en consecuencia son muchos sistemas operativos los que siguen vigentes y que vale la pena soportar, y para la mayoría de ellos hay cientos (si no es que miles) de dispositivos en los que corren, entonces se vuelve una tarea muy complicada probar la aplicación en cuestión en cada uno de los dispositivos y ver cómo se comporta con ese hardware en particular. Los propietarios de estas tecnologías están al tanto de este gran problema con el que se topan los desarrolladores, y es por eso que una herramienta, o conjunto de herramientas para solucionar este problema fueron desarrollados: los simuladores.

Los simuladores son de gran ayuda, cuentan con una representación en software del dispositivo real (un emulador), la cual se puede configurar de forma que se parezca lo más posible a un cierto dispositivo, en algunos casos incluso hay perfiles predefinidos con el nombre de algún dispositivo. También cuentan con otras herramientas, como perfiles de uso de hardware (procesador, memoria y procesador gráfico), bitácora de bugs y crashes e incluso herramientas de análisis de red.

Los dispositivos reales tienen la ventaja de ser exactamente la experiencia que el usuario final tendrá, tienen todas las limitaciones y caprichos a los cuales estarán expuestos los usuarios finales, y cuentan con funcionalidades que son difíciles de emular, por ejemplo lo que respecta a la localización, pero probar en dispositivos reales

también tiene muchas desventajas. Para empezar, es caro, conseguir una muestra representativa de dispositivos para pruebas es una inversión bastante considerable. Además, en caso de ser necesario para las pruebas, también hay que considerar los costos que conlleva la funcionalidad de red 3G, llamadas y mensajes. Otra desventaja es que probar en dispositivos reales tiende a ser más desorganizado, simplemente tener los dispositivos regados por todo el escritorio, o exponerse al tedio de estarlos guardando y sacando cada vez, asegurarse siempre de que la versión que se está probando es la que está instalada en todos los dispositivos, y claro, el hecho de que todo esto se lleva bastante más tiempo que en el caso de los dispositivos virtuales, tiempo que sumado en un proceso largo puede complicar llegar al fin del proyecto en el tiempo establecido.

Hasta ahora parecería que la respuesta al problema es ir con simuladores y listo, pero desafortunadamente la respuesta no es tan sencilla, probar únicamente en simuladores también tiene un gran número de problemas. Les faltan todas esas pequeñas características y caprichos que usualmente sólo tienen los dispositivos reales y que pueden hacer toda la diferencia para una experiencia de uso y sobre todo, el gran problema que tienen es que el poder de procesamiento que tiene una computadora de escritorio, sobre todo una que se usa para desarrollar cómodamente, probablemente es mucho mayor que el de cualquier dispositivo móvil que se esté emulando, y eso puede causar mediciones erróneas sobre el perfil de hardware de la aplicación, puede correr suavemente en el dispositivo emulado, mientras que en el real ni siquiera correr. Algunas funciones de hardware del dispositivo no funcionan, o no de forma confiable en un simulador, por ejemplo el multi-touch, o los servicios de localización.

Si hacer pruebas en dispositivos físicos es muy difícil y conlleva una inversión importante en equipo, la cual en ocasiones no es posible hacer, pero hacerlas en simuladores da resultados ideales, los cuales podrían no presentarse en un dispositivo real, ¿entonces cómo se debe proceder para probar una aplicación móvil? La respuesta es que no hay una respuesta absoluta, esto es algo que depende de la aplicación que se está desarrollando. Si en la aplicación en cuestión no se usa nada del hardware que causa conflictos en los simuladores y no necesita una gran cantidad de recursos puede que con el simulador sea suficiente, aunque conformarse sólo con los simuladores puede ser riesgoso, siempre es recomendable probar en al menos un dispositivo físico. A pesar de que no hay sólo una manera de hacer esto correctamente, el esquema que da un panorama más completo es uno mixto, probar en la mayor cantidad de dispositivos posible y hacer uso de las herramientas que tiene el simulador y no están disponibles para los dispositivos físicos.

Además del dilema de los dispositivos hay más retos específicos cuando se prueba una aplicación móvil, uno de ellos es el de las redes. Las redes son un problema puesto que son en extremo variantes geográficamente, tanto redes de datos de operadores de telefonía (EDGE, GSM, HSPA+, LTE, entre otros) como las redes TCP/IP de ISP’s. Y esto también depende de factores ambientales que no se pueden prever, como por ejemplo la presencia de una estructura metálica que genere

interferencia. Además de la calidad, que es un problema más bien menor, cada proveedor de servicios de red tiene su propia configuración de red que hace que cada red se comporte un poco diferente, debido a diferentes implementaciones de protocolos. Muchos proveedores incluso tienen implementados proxys que obstaculizan el flujo de información de una aplicación si requiere hacer peticiones HTTP. Hacer viajes con todo el equipo de pruebas para certificar que la aplicación funciona bien en cualquier red en la mayoría de los casos no es una solución viable. Aquí de nuevo entra en juego la utilidad de los simuladores, como el de Android, tienen la capacidad de emular redes, siempre y cuando la configuración del APN de la red en cuestión sea pública. De esta forma se puede hacer una prueba realista de cómo se comportará el dispositivo en esa red en particular y se pueden hacer cambios en el diseño si se llega a presentar algún problema.

En el proceso de pruebas de aplicaciones móviles no todo son problemas, también se tienen algunas herramientas útiles para facilitar el proceso. Una de ellas es la facilidad de automatizar el proceso de pruebas. En Mac OSX existe la aplicación llamada Automator, la cual facilita el proceso, puesto que se pueden hacer scripts para realizar pruebas automáticas para la aplicación. Por decir algo, si se conocen los pasos para reproducir un bug, se pueden programar dichos pasos, así al hacer un cambio en el código, una posible solución al bug, se puede correr el script y en segundos ya se tiene la respuesta de si el bug fue resuelto o no. Dentro de las herramientas de desarrollo del SDK de Android hay una llamada UIAutomator Viewer, la cual tiene funciones muy parecidas, programar un cierto caso de uso y que se pruebe automáticamente con solo correr un script. Además de éstas existen herramientas de terceros, algunas de las cuales incluso se anuncian como capaces de poder automatizar el proceso de pruebas completamente, de forma que un ser humano únicamente tiene que leer reportes y darle un resumen al desarrollador. Cómo uno se podría esperar estas soluciones son de paga, por lo que de quererlas incorporar al proceso de pruebas se tendría que analizar su viabilidad. Estas herramientas pueden ser muy útiles, sobre todo para eliminar muchos pasos intermedios que después de repetirse muchas veces se vuelven tediosos, pero sin duda es muy recomendable probar la aplicación exhaustivamente a mano al menos una vez, para asegurarse de que a la máquina no se le pasó de largo ningún problema, y también cuidar el hacer esta prueba en un dispositivo físico, esta es la forma de tener la mayor credibilidad y no dejar nada al azar.

III. Desarrollo

III.1 Planteamiento

La Metodología Inclusiva de Software y Hardware para pruebas en aplicaciones móviles que en este trabajo se plantea tiene su origen en el plan de pruebas estándar que la IEEE propone en su formato 829 - 2008, creando los documentos de diseño de la aplicación a probar acatando las recomendaciones propuestas en el documento de la IEEE. Los elementos esenciales son los siguientes:

- Un identificador del plan de pruebas: Este preferentemente debe ser creado artificialmente, no ser una palabra de algún lenguaje natural, de esta forma se puede garantizar más fácilmente que sea único.

- Referencias a documentos de diseño de la aplicación en sí: Con el afán de dar fé de que la aplicación que se está probando es congruente con la que se planteó originalmente, una de las pruebas que se realiza es la de flujo, que parte del diseño original.

- Definir un objetivo y alcance de las pruebas: ¿Qué es lo que se busca de este proceso? y ¿hasta dónde se debe llegar para poder decir que las pruebas dieron resultados satisfactorios?, aquí también se define claramente qué características del software se están evaluando y cuáles no, esto es algo importante en las plataformas móviles, puesto que hay cosas que se deben programar de una cierta forma por la misma arquitectura de la plataforma y un desarrollador no puede hacer nada al respecto.

- Requisitos mínimos del sistema: Para que el software pueda correr de manera óptima son necesarios ciertos recursos. Con la finalidad de evitar falsos positivos de errores o problemas de rendimiento en hardware no soportado se acota a partir de qué especificaciones de dispositivo pueden correr la aplicación sin problemas.

- Criterio o escala de aprobación: Evidentemente todo lo que anteriormente se ha definido debe servir para emitir un juicio sobre la calidad y/o funcionalidad del software, y los juicios siempre son subjetivos, pero con la idea de aterrizarlos en algo más objetivo se define un criterio o escala de aprobación o desaprobación, los cuales deben estar bien documentados, debe de estar explícita la escala o criterio y exactamente qué significa.

- Entregables: El proceso de pruebas debe rendir algún fruto, y es común que se den diferencias de opinión entre desarrolladores y el equipo de pruebas sobre qué información vale la pena rescatar. Los resultados, calificaciones y/o juicios de valor se plasman en los documentos entregables, que son lo que esperan recibir los desarrolladores de todo este proceso, dichos entregables deben ser un común acuerdo entre los equipos de pruebas y desarrollo, de forma que ambos tengan la información y el formato que el equipo de pruebas pueda ofrecer y el de desarrollo pueda entender.

- Recursos necesarios: Tanto humanos como materiales que se necesitan para llevar a cabo el proyecto. Este punto es vital, ya que desde este momento el equipo se puede dar la idea de si el proceso de pruebas es viable o no, si se dispondrá de la gente necesaria para realizarlo y/o de los recursos materiales, como hardware, software o en su defecto los recursos económicos para conseguirlos. Esta lista también debe tener en cuenta la demografía del público meta, por ejemplo, si se tiene una aplicación que hace uso extensivo de redes 4G y en el lugar donde se realizarán las pruebas no se dispone de acceso a una red 4G ahí hay un problema, puesto que la prueba podría no estar dando resultados realistas de uso.

- Responsable: En el plan de pruebas debe estar designado un responsable, y esta asignación debe quedar constatada por escrito en el plan de pruebas. El responsable del proceso de pruebas es la persona encargada de que se lleve a cabo todo el proceso en tiempo y forma, desde la realización del plan de pruebas hasta la entrega de informes al equipo de desarrollo. El responsable es también el primer canal de información entre el equipo de pruebas y el de desarrollo.

- Manejo de riesgos: El plan también debe contemplar el manejo de riesgos, esta es una actividad fundamental en la planificación de pruebas de software y seguimiento. Incluye la identificación, priorización, análisis y tratamiento de los riesgos que puede enfrentar el equipo. Los riesgos se derivan de una variedad de puntos de vista como el fracaso del proyecto, la seguridad, responsabilidades legales y los incumplimientos de estándares. Una cosa importante a considerar es que los riesgos son cosas que podrían presentarse, pero aún no ocurrieron. Un problema que ya se ha producido es un problema y se trata de manera diferente en la planeación de pruebas de software. El manejo de riesgos se divide en diferentes actividades. Primero que nada, la identificación de los riesgos. Los riesgos son identificados dentro del alcance del proyecto, pueden ser identificados utilizando varios recursos, por ejemplo, los objetivos del proyecto, listas de riesgos de proyectos anteriores, cosas inherentes a las plataformas o la comprensión del sistema antes del uso del sistema. La identificación de riesgos usualmente es un proceso iterativo, a medida que se avanza en el proyecto

algunos de los nuevos riesgos aparecen y algunos riesgos antiguos desaparecen. Una vez que se tienen identificados los riesgos es necesario para fines de la planeación el proponer un tratamiento para cada uno, como puede ser, en orden de prioridad, una solución, una postergación, una transferencia o una aceptación. En otras palabras, primero se busca una solución, si no la hay, se analiza si el riesgo es producto de una característica en particular y por lo tanto susceptible a ser postergado, en caso negativo se busca transferirlo a otra área, si alguien más especializado puede lidiar con él de una mejor forma, y en el caso extremo de que ninguna de las opciones haya resultado viable, simplemente se considera ese riesgo y se acepta en la arquitectura del sistema.

- Descripción de las pruebas a realizar: Esta descripción debe detallar exactamente qué se va a probar, darle un identificador a la prueba, a fin de poderse referir a ella con más facilidad, se debe decir también lo que necesita el equipo de pruebas para realizarla, en cuanto a estado del proyecto, recursos de software y de hardware; y una estimación del tiempo que llevará completarla una vez que se tiene todo lo necesario para comenzar. Así como se definió un criterio de aprobación o una escala para el proceso de pruebas en su totalidad, también se debe definir para cada prueba, o si es pertinente, usar el mismo que se usa para el proyecto en su totalidad, lo importante es que se tenga claro qué representa la escala o juicio y la causa por la que la prueba dio ese resultado. Una última cosa importante a definir para el conjunto de pruebas es su criticidad para el proyecto de software en sí, es de criticidad alta si es una prueba de la que depende en gran medida la concepción general de la calidad del software o si se trata únicamente de una funcionalidad superficial la que se está probando, se trata como una prueba de baja criticidad.

- Cronograma: Como todo elemento dentro del desarrollo de software, para tener los pies en la tierra es necesario un cronograma, una tabla de tiempos en la que quede constatada la duración de las pruebas y estimaciones del fin de estas. Este cronograma es susceptible a cambios, puesto que el proceso de pruebas es iterativo, y uno no puede saber de antemano que tan bien o mal viene cada versión del software que manda el equipo de desarrollo a pruebas. En cuanto al formato, cualquier herramienta que sea sencilla y fácil de entender para los involucrados en el proyecto está bien, puede ser un diagrama de Gantt, una hoja de Scrum o un simple calendario.

Esto resume el proceso de planeación general del plan de pruebas, y todo esto queda constatado en uno o varios documentos, pero para cada prueba en particular es también necesario hacer un plan de prueba, este no es tan exhaustivo como el plan general, porque se considera que varios elementos se heredan del plan general y sería redundante manejarlos en los planes particulares también. Los planes particulares de cada prueba de cualquier forma tienen que llevar los siguientes elementos particulares:

- Describir con suficiente detalle el proceso a seguir para cada prueba, los pasos deben estar explicados con suficiente detalle y en términos que una persona que no posea mucho conocimiento del tema pueda entender.

- Lo que se espera conseguir en la prueba en cuestión, un objetivo claro y concreto de a dónde debe llevar la prueba.

- El enfoque que se le va a dar, como las aplicaciones tienen diferentes ángulos es fácil malinterpretar el rumbo que debe tomar la prueba. Lo mejor es que se pueda resumir en unas pocas palabras, por ejemplo, funcional, de flujo, estético.

- Los requisitos de software, hardware, si la aplicación no ha alcanzado un cierto nivel de avance en el desarrollo en ocasiones es difícil, o incluso imposible hacer la prueba, y si no se cuenta con el equipo necesario, sean dispositivos o computadoras para correr las simulaciones, la prueba tampoco se podrá realizar.

- Personal que se necesitan para poder comenzar con la prueba, tanto integrantes del equipo de pruebas como usuarios. Es importante definir una muestra significativa en el grupo de usuarios, es decir, cantidad suficiente de usuarios de diferentes tipos en cuanto al nivel de uso de tecnología.

- Tiempo estimado para completar la prueba, para poder asegurar que las pruebas no representarán un atraso en el desarrollo en general.

- Responsable de la prueba, al igual que en la totalidad del ciclo de pruebas, cada una tiene su representante, que no necesariamente tiene que ser el mismo que el del ciclo completo.

- Escala o juicio de evaluación que se va a seguir para la prueba en particular, la cual debe quedar bien explicada para evitar ambigüedades. Puede ser numérica, basada en adjetivos, o una combinación de ambas cosas.

Con todo lo anteriormente expuesto se puede considerar trazada toda la fase de planeación de las pruebas, y si la redacción de los documentos es correcta se pueden construir los entregables muy fácilmente a partir de ellos.

III.2 Caso Tipo: HDS

La empresa de salud HDS encargó a la UNAM que supervisara el desarrollo de un conjunto de aplicaciones que deseaban desarrollar. Desde el principio el desarrollo de estas aplicaciones fue complicado debido a las limitaciones de tiempo, puesto que sólo se contaba con 3 meses para todo el proceso. Se hizo un levantamiento de requerimientos para poder llegar a propuestas de diseño de las aplicaciones, y lo que resultó fue lo siguiente:

- Las aplicaciones debían ser desarrolladas para tres plataformas, Android, iOS y Windows Phone.
- El usuario debe ser capaz de cerrar la aplicación y que ésta siga corriendo en segundo plano, en una forma mínima que no consuma muchos recursos.
- Debe de existir una implementación de notificaciones push, esto porque los usuarios deben poder configurar recordatorios en las aplicaciones y las aplicaciones deben alertarlos a la hora estipulada, sin importar el estado de la aplicación en ese momento.

A pesar de que las aplicaciones en sí son sencillas y de una funcionalidad muy puntual, como se vio en el levantamiento de requerimientos usan ciertas funciones inherentes del sistema operativo, lo cual no representa un problema en sí, salvo por el detalle de que las aplicaciones fueron solicitadas agnósticas, es decir, que un mismo desarrollo pudiera funcionar en las diferentes plataformas objetivo. En vista de que hacer un desarrollo nativo para cada plataforma se llevaría más tiempo del que se disponía, se procedió a una evaluación de posibles marcos de trabajo con los que se pudiera llevar a cabo un desarrollo de naturaleza agnóstica y poder realizar un solo desarrollo que se pudiera trasladar a todas las plataformas, los candidatos fueron PhoneGap, JQuery Mobile, Cordova, Sencha, Unity3D y MonoGame. Eventualmente se decidieron por PhoneGap, debido a diversos factores, como su licencia, el hecho de que es gratuito, que tiene buena documentación y ofrece acceso a varias funciones de hardware. La desventaja es que ninguno de los marcos de trabajo propuestos incluye soporte a ciertas funciones de sistema operativo, como notificaciones push y comportamiento en segundo plano, que eran requisitos para el desarrollo. Sin embargo PhoneGap tiene la ventaja de admitir plug-ins, es decir, pequeños bloques de funcionalidad realizados con código nativo de la plataforma en cuestión (Java en el caso de Android y Objective-C en el caso de iOS, lenguajes de .NET para Windows Phone), lo cual le da la facultad de cumplir con todos los requisitos necesarios para las aplicaciones de HDS, aunque eso también estaría a merced de la capacidad de sus desarrolladores en el manejo e integración de código nativo con el proyecto de PhoneGap.

Al ser alrededor de 10 aplicaciones, 10 desarrollos, una sola persona no podía ser responsable de las pruebas de todas ellas, hubiera sido demasiado trabajo considerando las restricciones de tiempo. Así que se hizo una selección inicial de una persona por aplicación, y a medida que se fueron terminando se asignaban más

aplicaciones a los participantes. En mi caso me tocaron un total de 2 aplicaciones a probar, una llamada Asistente Nutricional y otra Epoc.

Aplicación 1: Asistente Nutricional

Asistente Nutricional es una aplicación bastante sencilla, pregunta al usuario algunos datos sobre su físico, edad, sexo, peso y estatura, en base a eso calcula su Índice de Masa Corporal (IMC) y da un juicio de la situación de peso de la persona, si está baja, alta o bien de peso, y qué nivel de sobrepeso tiene en su caso. Además de eso le permite al usuario programar recordatorios de comida, desayuno, almuerzo, comida, cena y/o merienda; así como establecer rutinas de ejercicios dados en una lista (correr, pesas, yoga, fútbol, etc.) y también programar un recordatorio.

Los requerimientos puntuales para esta aplicación son los siguientes:

- Plataformas Android y iOS
- Indispensable que funcione en smartphones, tablets no es necesario
- Notificaciones push para recordatorios de comida y/o ejercicio

A continuación se presenta la propuesta de diseño final para la aplicación asistente nutricional:

Diseño de “Asistente Nutricional”

Control de versiones

Fecha	Versión - Comentarios	Emisor/Revisor
21/09/2012	1.0	Fernando Robles Muñoz
25/10/12	1.1-Informacion General	Fernando Robles Muñoz

Información general

Nombre propuesto para la aplicación
Asistente Nutricional

Palabras clave
Recordatorios, ejercicio, comida, peso, salud, avance

Ícono propuesto
{Poner aquí propuestas}

Idiomas soportados

{Poner aquí listados numéricamente los lenguajes con código de 2 letras}

1. Español (ES)

Plataformas soportadas

1.	iPhone	SI
2.	Android SmartPhone	SI
3.	iPad	NO
4.	Android Tablet	NO

Plataforma de desarrollo base

a. Desarrollo nativo para Android e iOS

Con base en los casos de uso descritos por el cliente, se propone como punto de partida para el desarrollo de la aplicación el análisis de la interfaz de usuario que a continuación se detalla en la figura 3.1:

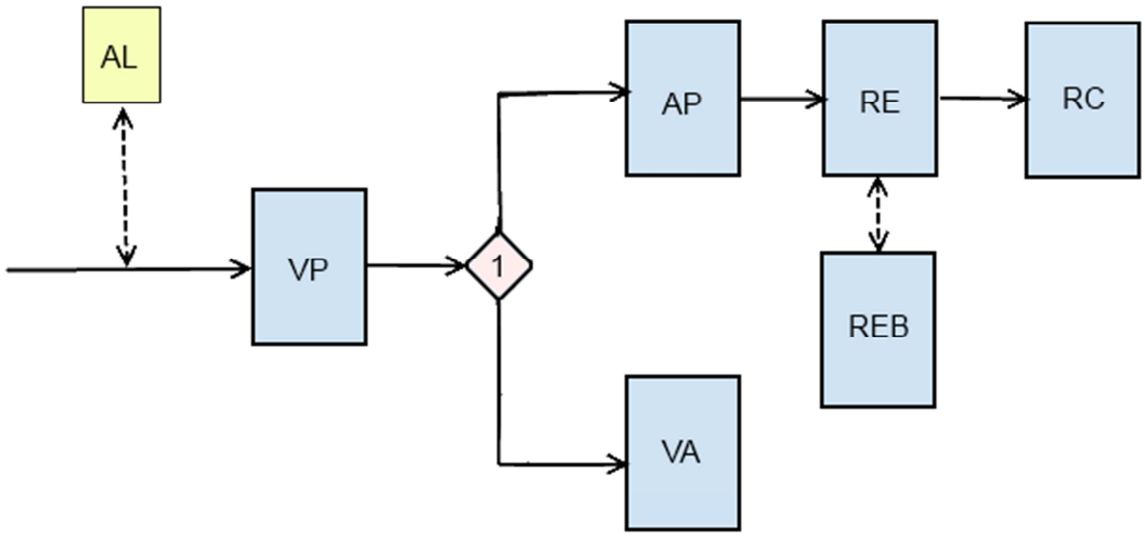


Figura 3.1 Flujo de aplicación para Asistente Nutricional

Documentos de referencia:

- **A** = ARM_CU1_IniciarControlNuticional.doc
- **B** = ARM_CU2_AdministrarAplicacion.doc
- **C** = ARM_CU3_VisualizarAvance.doc
- **D** = ARM_CU4_ConfigurarRecordatoriosEjercicio.doc
- **E** = ARM_CU5_ConfigurarRecordatoriosComidas.doc

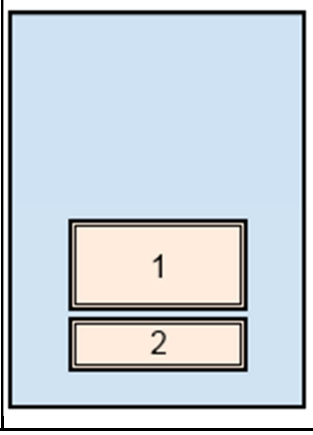
- **F** = ARM_CU6_EjecutarRecordatoriosEjercicios.doc
- **G** = ARM_CU7_EjecutarRecordatoriosComidas.doc

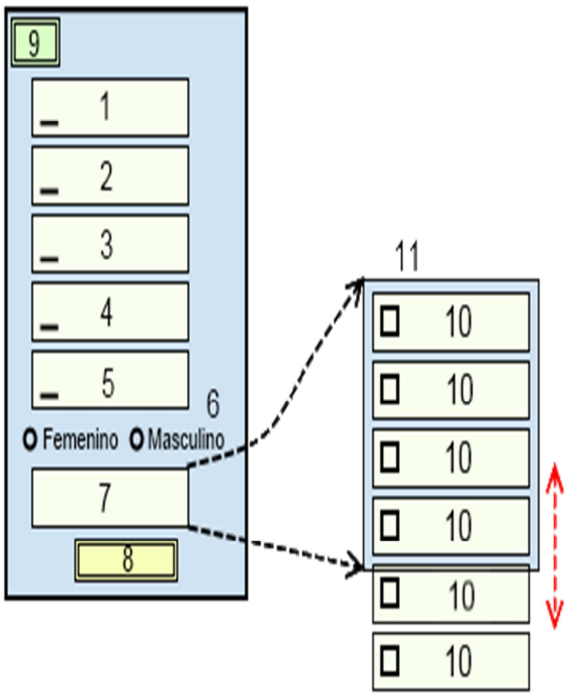
Vistas

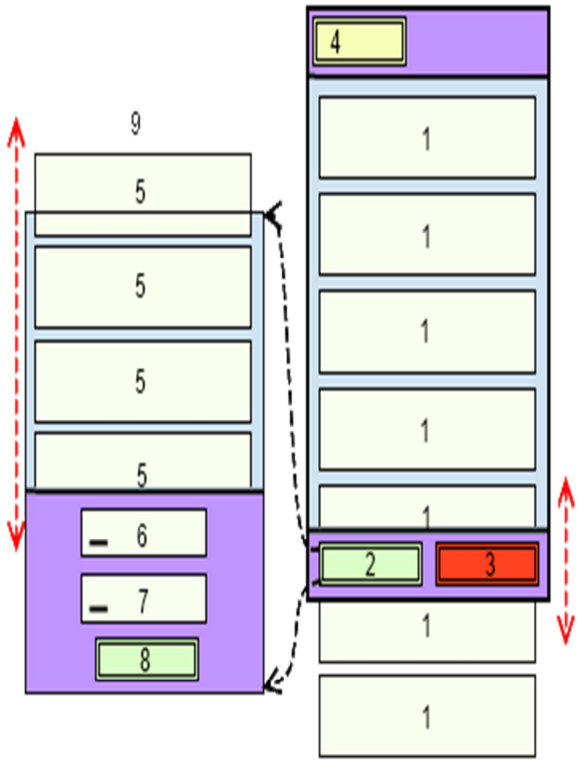
- VP = Vista Principal
- AP = Administrar Aplicación
- RE = Configurar Recordatorios Ejercicio
- REB = Modo Borrar Ejercicio de Configurar Recordatorios Ejercicio
- RC = Configurar Recordatorios Comidas
- VA = Visualizar Avance
- AL = Alarma de Ejercicio o Comida

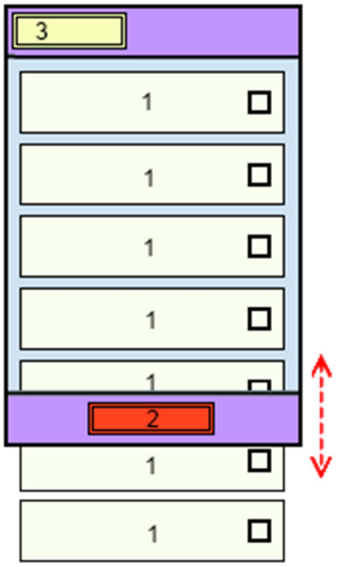
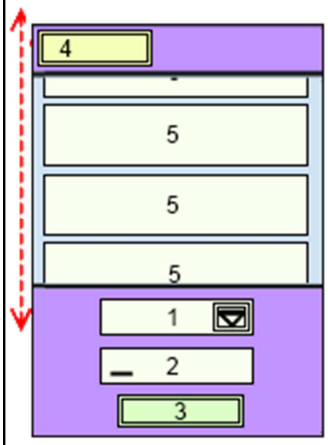
A continuación, en la tabla 1 se presenta el detalle da cada pantalla y su funcionalidad:

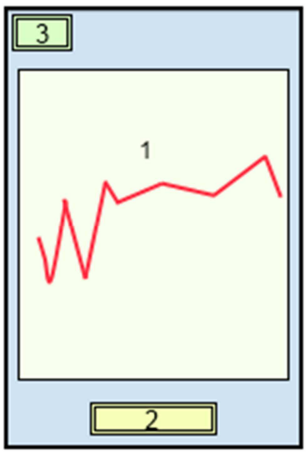
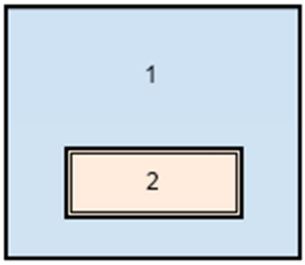
Tabla 3.1. Desglose de funcionalidad

ID	Vista	Descripción	Ref
VP		**Vista principal de la aplicación:** Pantalla principal donde se puede elegir administrar la aplicación y visualizar avance. **Elementos:** 1) Acceso a registrar Administrar Aplicación 2) Acceso a Visualizar Avance	A-4.1

AP		**Administrar Aplicación:** Pantalla para configurar parámetros fundamentales para la aplicación. **Elementos:** 1) Campo de texto para nombre del usuario. 2) Campo de texto para la edad del usuario. 3) Campo de texto para peso del usuario (kg). 4) Campo de texto para la estatura del usuario (cm). 5) Campo de texto para la circunferencia de cintura del usuario (cm). 6) Botón tipo radio para el sexo del usuario. 7) Botón para elegir enfermedades. 8) Botón de guardar información. 9) Botón de regresar. 10) Campos tipo check para elegir las enfermedades. 11) Sub ventana para elegir una o más enfermedades.	B-4.1

RE		**Configurar Recordatorios** **Ejercicio:** Lista de los ejercicios guardados, con posibilidad de agregar o eliminar ejercicios. **Elementos:** 1) Datos del ejercicio guardado (Ejercicio, rango de horas). 2) Botón de agregar ejercicio. 3) Botón de borrar ejercicio (lleva REB). 4) Botón de finalizar. 5) Tipo de ejercicio del catálogo. 6) Campo para la hora de inicio del ejercicio. 7) Campo para la hora de fin del ejercicio. 8) Botón de guardar ejercicio. 9) Sub ventana de agregar ejercicio.	D-4.1 D-4.2.1-FA01

REB		**Modo de Borrado de Ejercicios de RE.** Vista para seleccionar uno o más ejercicios a ser borrados. **Elementos:** 1) Datos del ejercicio guardado (Ejercicio, rango de horas). Con un cuadro tipo check para seleccionar los ejercicios. 2) Botón de borrar ejercicios. 3) Botón de cancelar.	D-4.2.1-FA02
RC		**Configurar Recordatorios Comidas:** Vista con la lista de comidas almacenadas, con sus respectivas horas. Con opción de agregar nueva comida. **Elementos:** 1) Lista seleccionable de la comida (desayuno, colación matutina, comida, colación vespertina y cena). 2) Campo para la hora de la comida. 3) Botón de agregar comida. 4) Botón de finalizar. 5) Comida con su respectiva hora.	E-4.1

VA		**Visualizar Avance:** Vista con la gráfica con el valor de todos los registros de peso contra sus fechas. **Elementos:** 1) Gráfica peso vs fecha. 2) Botón de aceptar. 3) Botón de regresar.	C-4.1
AL		**Alarma de Recordatorio:** Recordatorio generado cuando el sistema encuentra que la hora de una comida o un ejercicio es igual a la del sistema. **Elementos:** 1) Mensaje de la alarma. 2) Botón de aceptar.	F-4.1 G-4.1

Epoc es una aplicación orientada a un problema muy específico, es un asistente para las personas que padecen de la Enfermedad Pulmonar Obstructiva Crónica, que es una enfermedad que puede producir paros respiratorios, llamados en la aplicación crisis de Epoc, además de que la condición hace necesario el uso constante de medicamentos y oxígeno para el que lo padece. La aplicación de HDS está 100% enfocada a esta enfermedad, le da al usuario la capacidad de configurar un botiquín con recordatorios de toma para cada medicamento, permite también configurar un tanque de oxígeno, con todos los parámetros que eso involucra, dicha configuración avisa cuando el tanque se está a punto de acabar, si es que todo funciona bien. Lo más destacado de la aplicación es que tiene un botón de pánico, el usuario configura una lista de contactos de emergencia dando los números telefónicos y correos electrónicos de gente de su confianza. Cuando el usuario esté teniendo un paro respiratorio puede usar el botón de pánico y la aplicación le mandará a los contactos que se hayan configurado un mensaje de texto SMS y/o un correo electrónico para que acudan a auxiliarlo a la brevedad.

Los requerimientos puntuales para Epoc requeridos por el cliente son:

- Plataformas Android, iOS y Windows Phone
- Notificaciones push para recordatorios de oxígeno y medicamentos

- Acceso a la lista de contactos del usuario para la función de pánico
- Envío de SMS y correo electrónico automáticamente

Una vez conocida la funcionalidad de las aplicaciones a detalle en la UNAM se realizaron propuestas de diseño, diferentes flujos y propuestas gráficas para las aplicaciones. El cliente los fue revisando y dando retroalimentación hasta que vieron los requerimientos cubiertos a su satisfacción, y entonces tomaron las propuestas de la UNAM y las canalizaron a su propio departamento de desarrollo para iniciar la programación de las aplicaciones. A continuación se presenta la propuesta de diseño final de la aplicación Epoc:

Diseño de "Epoc"

Control de versiones

Fecha	Versión - Comentarios	Emisor/Revisor
29/09/2012	1.0	Yésica Hernandez
30/10/12	1.1-Informacion General	Yésica Hernandez

Información general

Nombre propuesto para la aplicación
Epoc

Palabras clave
Recordatorios, botiquín, paro respiratorio, salud

Ícono propuesto
{Poner aquí propuestas}

Idiomas soportados
{Poner aquí listados numéricamente los lenguajes con código de 2 letras}

2. Español (ES)

Plataformas soportadas

5. iPhone SI
6. Android SmartPhone SI
7. Windows Phone SI
8. Tablets NO

Plataforma de desarrollo base

b. Desarrollo nativo para Android, iOS y Windows Phone

Con base en los casos de uso descritos por el cliente, se propone como punto de partida para el desarrollo de la aplicación el análisis de la interfaz de usuario que a continuación se detalla en la figura 3.2:

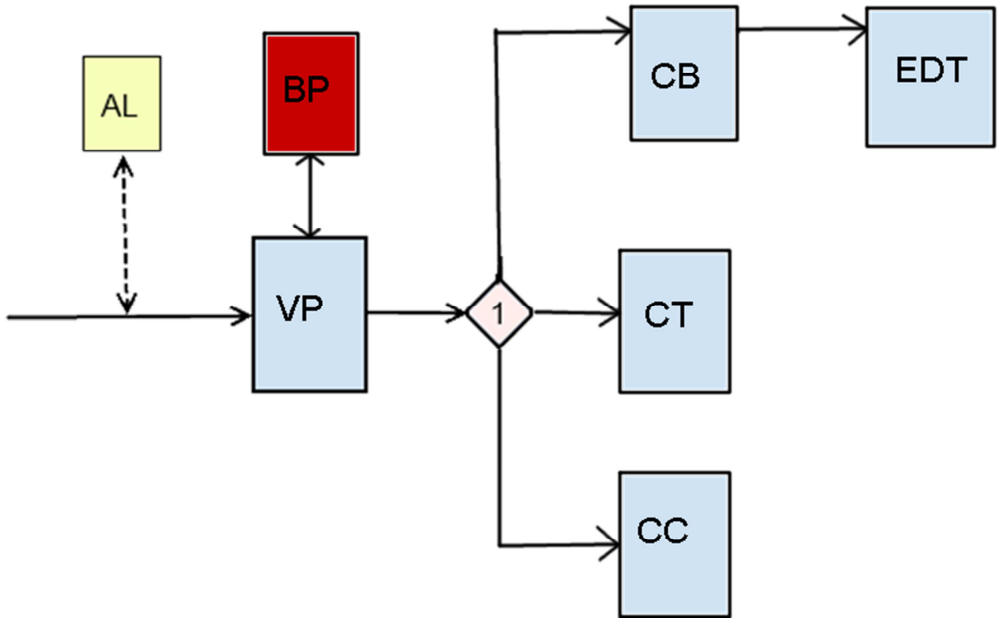


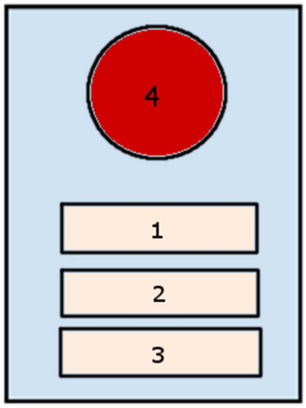
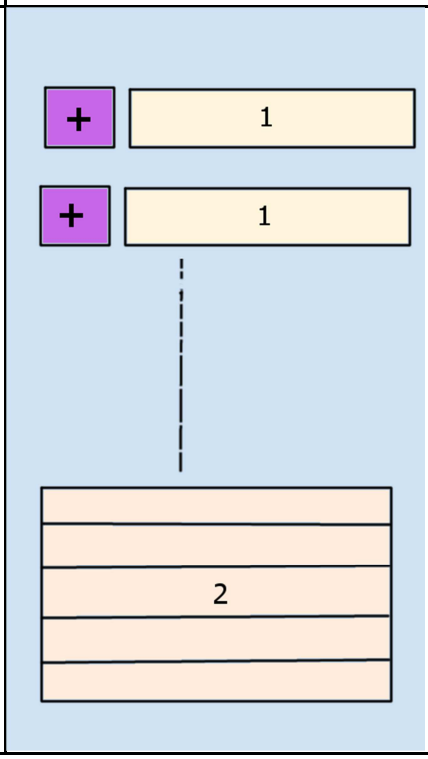
Figura 3.2 Flujo de aplicación para Epoc

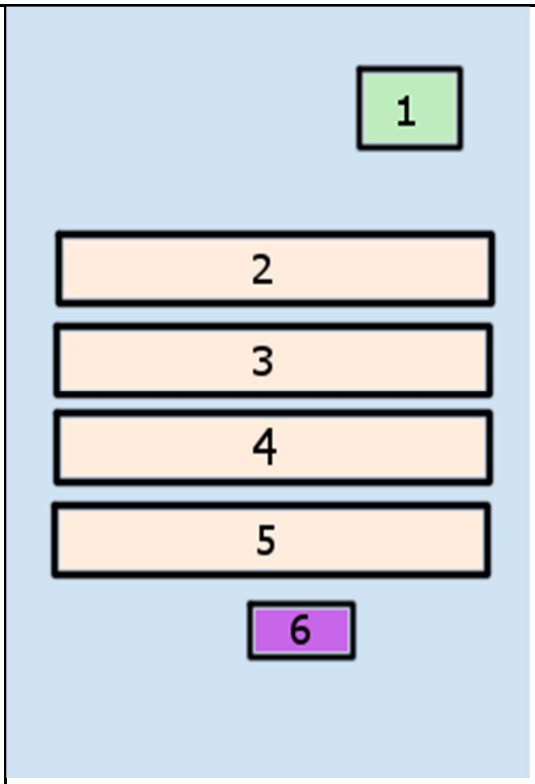
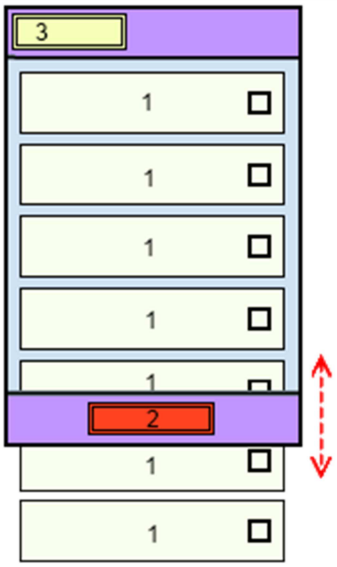
Vistas

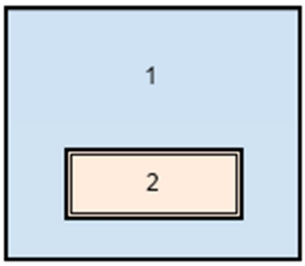
- VP = Vista Principal
- CB = Configurar botiquín
- CT = Configurar tanque de oxígeno
- EDT = Modo Editar botiquín
- CC = Configurar contactos para modo pánico
- BP = Vista de pánico
- AL = Alarma de Medicamento u Oxígeno

A continuación, en la tabla 2 se presenta el detalle da cada pantalla y su funcionalidad:

Tabla 3.2. Desglose de funcionalidad

ID	Vista	Descripción	Ref
VP		**Vista principal de la aplicación:** Pantalla principal donde se puede elegir administrar la aplicación y hacer uso de la función de pánico. **Elementos:** 1) Acceso a configurar botiquín 2) Acceso a configurar botón de pánico 3) Acceso a configurar tanque 4) Activar botón de pánico	A-4.1
CB		**Configurar botiquín:** Pantalla para agregar medicamentos al botiquín **Elementos:** 1) Campo para agregar medicamento con el botón + 2) Lista de medicamentos recomendados para el padecimiento, con las dosis normales del mercado.	B-4.1

CT		**Configurar Tanque de oxígeno:** Parámetros para definir la duración de un tanque de oxígeno en base a su consumo. **Elementos:** 1) Botón para configurar hora del recordatorio 2) Campo de texto para capacidad (l). 3) Campo de texto para consumo (l/h). 4) Campo de texto para consumo diario (h). 5) Campo de texto para duración (m). 6) Botón guardar.	D-4.1 D-4.2.1-FA01
CC		**Configurar contactos** Vista para seleccionar uno o varios contactos para una crisis. **Elementos:** 1) Agregar contacto con el botón + 2) Botón de guardar. 3) Drop down para tipo de notificación (SMS o email).	D-4.2.1-FA02

AL		**Alarma de Recordatorio:** Recordatorio generado cuando el sistema encuentra que la hora de tomar un medicamento u oxígeno ha llegado. **Elementos:** 1) Mensaje de la alarma. 2) Botón de aceptar.	F-4.1 G-4.1

Para cuando se hicieron las primeras entregas de las aplicaciones, los planes de pruebas ya estaban listos, entonces se procedió directamente a las pruebas en sí. Esto comenzó reuniendo el grupo que estaría encargado de hacer las pruebas, puesto que fue estipulado que cada responsable de cada aplicación tendría que integrar su propio grupo. Esta parte es crucial para todo el proceso de pruebas, ya que se necesitan usuarios de todos perfiles, que estén dentro del target y también algunos que no, que tengan conocimientos técnicos y que no estén tan familiarizados con la tecnología, que sean usuarios avanzados de las plataformas y que apenas las conozcan, y sobre todo, que incluya otros desarrolladores y gente que se dedique a algo distinto. Esto por varias razones, primero porque esta es una muestra significativa del ecosistema al que se va a lanzar la aplicación a final de cuentas, usuarios de todos tipos la tendrán en sus manos, pero en el proceso de pruebas es especialmente útil tener usuarios avanzados y programadores, porque ellos son los que pueden dar información más rica y útil a la hora de reportar bugs, y otros problemas de funcionamiento. Esto no quiere decir para nada que las opiniones y reportes de los usuarios sin conocimientos técnicos son inservibles, por el contrario, la exposición de la aplicación a este tipo de usuarios es la que da la información más fructífera en cuanto a facilidad de uso y diseño de la aplicación, tanto gráfico como funcional. Esto se explica porque una aplicación debe estar hecha de la forma más intuitiva posible, y los usuarios que no están acostumbrados a este tipo de tecnología son los que más reaccionan ante comportamientos no intuitivos, y como los que desarrollamos aplicaciones normalmente ya estamos muy embebidos en las tecnologías damos por hecho que todo el mundo conoce ciertas prácticas comunes, cuando podría no ser el caso para todo mundo. Una última observación respecto al grupo de testers para este caso en particular, para el tipo de aplicaciones que busca HDS, aplicaciones de apoyo médico, fue importante para conformar un grupo dentro del target que fuera gente que tuviera interés o necesidad de este tipo de aplicaciones, esto no fue difícil para Asistente Nutricional, puesto que sólo es un apoyo para cuidar la salud, la alimentación y los hábitos de ejercicio, así que a muchas personas le puede interesar y pueden hacer uso de ella, pero para Epoc no resultó tan sencillo, porque es una aplicación orientada a gente con un padecimiento muy específico, y desafortunadamente no nos fue posible encontrar a

alguien que lo tuviera, pero compensamos esta falla pensando en otro trastorno que conlleva a fallos respiratorios, que es la parte delicada y que puede salvar vidas en esta aplicación en particular.

A final de cuentas el equipo de testers fue integrado de la siguiente manera:

Tabla 3.3: Relación de Testers

Perfil	Cantidad	Equipos
Con conocimientos técnicos (desarrolladores)	6	iOS, Android y WP
Sin conocimientos técnicos	5	iOS, Android y WP
Que hacen ejercicio y/o se preocupan por su salud	4	iOS, Android
Que no hacen ejercicio y/o no se preocupan por su salud	4	iOS, Android y WP

A continuación se presenta el plan de pruebas completo de las dos aplicaciones:

Plan de pruebas para "Asistente Nutricional"
Elaboró:

- Edgar García Cano
- Luis Valencia
- Marco Salcedo

Identificador del plan

ASISTENTENUTRICIONAL-PPG1

Alcance

Este documento describe las pruebas mínimas necesarias para asegurar la calidad del software desde el punto de vista de los requisitos de las plataformas móviles siguientes a través de una estrategia simplificada de validación para un tiempo no máximo a 1 semana laboral:

- Android
- iOS

Configuraciones y requisitos mínimos

1. Ambiente de desarrollo configurado correctamente para soportar las versiones del software establecidas en el alcance del proyecto.
2. Código fuente debidamente documentado para efectuar los niveles de pruebas.

Estrategia

1. Se elaborarán las pruebas descritas en la tabla inferior para ejecutarse en las distintas etapas del software indicadas en la misma. Con la información recabada se podrá generar un entregable que resuma el resultado de las validaciones aplicadas.
2. Así mismo se tomará en cuenta la criticidad de cada prueba dependiendo de la escala propuesta para detener, proseguir o pausar el ciclo de pruebas.

Tabla 3.4: Niveles de criticidad

Etiqueta	Definición
BAJA	El fallo en esta prueba no constituye un impedimento para la liberación del software y se puede corregir en tiempo y forma dentro del ciclo de desarrollo.
MEDIANA	El fallo en esta prueba podría ser un impedimento para la liberación del software, al no cumplir con un punto importante en la correcta ejecución de la misma y se deberá evaluar el costo en tiempo y recursos de su corrección. Y se puede continuar con las pruebas de otro tipo.
ALTA	El fallo en esta prueba se considera una falta grave de diseño, la cual impide la ejecución de las pruebas subsecuentes hasta que no se evalúe el fallo y se defina una solicitud de cambio para su corrección.

Entregables

Documento que resuma los aspectos evaluados en las pruebas con el resumen de pruebas efectuadas, las pruebas con criticidad alta que no fueron aprobadas y el número de veces en que se tuvieron que volver a efectuar dichas pruebas.

Recursos

Simuladores/emuladores capaces de ejecutar el código en las versiones de las plataformas establecidas en el proyecto.
No necesario: Dispositivos físicos para probar las resoluciones descritas en la arquitectura del proyecto y configurados para proveer información de depuración.

Tabla 3.5: Recursos del proyecto con los que se cuenta para diseñar las pruebas

Documento o fuente	SI/NO
Requerimientos funcionales y no funcionales	si
Plan de trabajo	no
Diagrama de arquitectura	si
Código fuente comentado	no
Ejecutable funcional	si
Otros: _____	

Manejo de riesgos

Con el fin de minimizar los riesgos en la ejecución del plan de pruebas es imperante que el ambiente de desarrollo esté bien configurado y se tengan en cuenta los errores comunes que pudieran impedir la ejecución de las pruebas.

Amenaza	Vulnerabilidad	Solución
Fallo del ambiente de pruebas	Mala configuración para depurar la ejecución	Efectuar una revisión general de la configuración del ambiente antes de iniciar las pruebas.
	Error en el software (IDE, Simuladores, plugins, librerías, etc) que impide la depuración.	Buscar en las páginas oficiales la información relacionada al fallo y reiniciar o reinstalar servicios, procesos o programas cuando sea necesario.
	Error en la conexión de dispositivos físicos.	Buscar en las páginas de los fabricantes antes de la ejecución de pruebas la lista de requisitos para el modo de depuración en el sistema operativo huésped. (Uso de drivers especiales, configuraciones de montaje, fiabilidad de los periféricos de conexión, etc...).

La incapacidad de manejar otras amenazas y vulnerabilidades que no estén contempladas en este plan deberá ser informada a la brevedad al líder de proyecto para que se tomen las acciones adecuadas a las políticas de la organización.

Responsables

Nombre (s)	Responsabilidad
Marco Salcedo	Pruebas Android
Marco Salcedo	Pruebas iOS
Marco Salcedo	Pruebas Windows Phone*

* Solo si aplica la entrega a tiempo del código fuente.

Pruebas a aplicar

A continuación se enlistan los bloques de pruebas generales a aplicar, de los cuales se desglosan pruebas puntuales y un calendario para dichas pruebas basados en la calendarización de las fases de construcción de software que se lleven a cabo.
Para cada prueba individualmente se debe especificar:

1. Duración de la prueba
2. Alcance
3. Restricciones
4. Responsable de reportar resultados
5. Fecha de inicio
6. Reporte final de resultados
7. Elementos objetivo de la elaboración de la prueba (personas, proceso automatizados, testers)

Tabla 3.5: Flujo de aplicación asistente nutricional

ID	Ámbito	Tipo de prueba	Fecha o fase de aplicación	Descripción	Criterios de Aprobación	Criticidad
CH1	Navegación en la interfaz de usuario	Validación contra checklist	Cuando la interfaz esté implementada.	Revisar contra los casos de uso que se cumplan todas las características requeridas a través de las vistas navegables. El checklist se elabora de manera personalizada de acuerdo al proyecto y a la arquitectura provista.	Todos los puntos del checklist aprobados	BAJA

CE1	Intuitividad de la interfaz de usuario	Validación contra cuestionario por escalas	Cuando la interfaz esté implementada e integrada con el diseño gráfico.	El cuestionario por escalas se elabora de acuerdo a los puntos más importantes en las guías de diseño de cada plataforma y de acuerdo a los componentes usados en el proyecto. Se necesitan sujetos de prueba pertenecientes al mercado objetivo del proyecto.	Se definirá un rango mínimo de aceptación del 90% de satisfacción en la escala definida.	BAJA
CH2	Nivel de calidad de la interfaz de usuario a nivel diseño gráfico.	Validación contra checklist	Cuando la interfaz esté implementada e integrada con el diseño gráfico	El cuestionario por escalas se elabora de acuerdo a los puntos más importantes en las guías de diseño de cada plataforma y de acuerdo a los componentes usados en el proyecto. Se necesitan expertos en diseño gráfico	Se definirá un rango mínimo de aceptación del 90% de satisfacción en la escala definida.	BAJA
CH3	Tolerancia a patrones anormales de uso de los elementos en la interfaz	Validación contra checklist	Cuando la aplicación esté en beta o bien implementada más del 90%	De acuerdo a los componentes y al flujo de vistas de la aplicación se crea una serie de pruebas puntuales cuyo objetivo es forzar el bloqueo de la aplicación producto de un	Todos los puntos del checklist aprobados	MEDIA

				uso exhaustivo y desmedido de los eventos de interfáz		
CH4	Tolerancia a errores de acceso a medios	Validación contra checklist	Cuando la aplicación esté en beta o bien implementada más del 90%	De acuerdo a los componentes y al flujo de vistas de la aplicación se crea una serie de pruebas puntuales cuyo objetivo es generar los escenarios restrictivos de acceso a medios.	Todos los puntos del checklist aprobados	MEDIA
CH5	Compatibilidad con las plataformas objetivo	Validación contra checklist	En cualquier etapa, y cuando la aplicación esté en beta	De acuerdo a las plataformas soportadas de la aplicación se intentará instalar el producto en todos las versiones soportadas y en las que no a fin de garantizar que el software tiene las configuraciones pertinentes para impedir que el sistema la instale en versiones no soportadas.	Todos los puntos del checklist aprobados	ALTA
CH6	Pruebas unitarias	Validación contra checklist	Al implementar el diseño de arquitectura.	Comprobar que todos los métodos devuelven los resultados esperados dados un conjunto iterativo de datos que abarcan los	Todos los puntos del checklist aprobados	ALTA

				mejores y peores escenarios así como datos erróneos o indefinidos.		

Cronograma de pruebas por bloques

Fecha de inicio del ciclo de pruebas	30/11/2012
Deadline de aplicación de pruebas (al menos un ciclo)	21/12/2012

Tabla 3.6: Profundidad de pruebas

Nivel de pruebas	Pruebas asociadas	Responsable	Tiempo estimado de aplicación de la prueba
Pruebas de caja negra, unitarias y de integración	CH6	Fuera del alcance del Departamento de computación por no contar con el código fuente	Depende del proveedor del código.
Pruebas de estrés e integridad (compatibilidad, ejecución y fallos en la navegación)	CH1-3, CH5-6	UNAM Mobile + Facultad de Ingeniería	1 día, se pueden aplicar en paralelo a la misma revisión de código (versión)

Pruebas de experiencia de usuario y satisfacción	CH4, CE1	UNAM Mobile + Facultad de Ingeniería	1 día, se pueden aplicar en paralelo a la misma revisión de código (versión)

Las pruebas se repiten una vez enviada alguna corrección o propuesta de cambio cíclicamente hasta alcanzar el resultado establecido para cada una, por ejemplo, en un rango de calificación un 70% del valor máximo alcanzable en la prueba.

Plan de pruebas para EPOC

Elaboró:

- Edgar García Cano
- Luis Valencia
- Marco Salcedo

Identificador del plan

EPOC-PPG1

Alcance

Este documento describe las pruebas mínimas necesarias para asegurar la calidad del software desde el punto de vista de los requisitos de las plataformas móviles siguientes a través de una estrategia simplificada de validación para un tiempo no máximo a 1 semana laboral:

- Android
- iOS
- Windows Phone

Configuraciones y requisitos mínimos

1. Ambiente de desarrollo configurado correctamente para soportar las versiones del software establecidas en el alcance del proyecto.
2. Código fuente debidamente documentado para efectuar los niveles de pruebas.

Estrategia

1. Se elaborarán las pruebas descritas en la tabla inferior para ejecutarse en las distintas etapas del software indicadas en la misma. Con la información recabada se podrá generar un entregable que resuma el resultado de las validaciones aplicadas.
2. Así mismo se tomará en cuenta la criticidad de cada prueba dependiendo de la escala propuesta para detener, proseguir o pausar el ciclo de pruebas.

Niveles de criticidad

Etiqueta	Definición
BAJA	El fallo en esta prueba no constituye un impedimento para la liberación del software y se puede corregir en tiempo y forma dentro del ciclo de desarrollo.
MEDIANA	El fallo en esta prueba podría ser un impedimento para la liberación del software al no cumplir con un punto importante en la correcta ejecución de la misma y se deberá evaluar el coste en tiempo y recursos de su corrección. Y se puede continuar con las pruebas de otro tipo.
ALTA	El fallo en esta prueba se considera una falta grave de diseño la cual impide la ejecución de las pruebas subsecuentes hasta que no se evalúe el fallo y se defina una solicitud de cambio para su corrección.

Entregables

Documento que resuma los aspectos evaluados en las pruebas con el resumen de pruebas efectuadas, las pruebas con criticidad alta que no fueron aprobadas y el número de veces en que se re-efectuaron dichas pruebas.

Recursos

Simuladores/emuladores capaces de ejecutar el código en las versiones de las plataformas establecidas en el proyecto.
No necesario: Dispositivos físicos para probar las resoluciones descritas en la arquitectura del proyecto y configurados para proveer información de depuración.

Recursos del proyecto con los que se cuenta para diseñar las pruebas

Documento o fuente	SI/NO
Requerimientos funcionales y no funcionales	si
Plan de trabajo	no
Diagrama de arquitectura	si
Código fuente comentado	no
Ejecutable funcional	si
Otros: _____	

Manejo de riesgos

Con el fin de minimizar los riesgos en la ejecución del plan de pruebas es imperante que el ambiente de desarrollo esté bien configurado y se tengan en cuenta los errores comunes que pudieran impedir la ejecución de las pruebas.

Amenaza	Vulnerabilidad	Solución
Fallo del ambiente de pruebas	Mala configuración para depurar la ejecución	Efectuar una revisión general de la configuración del ambiente antes de iniciar las pruebas.
	Error en el software (IDE, Simuladores, plugins, librerías, etc) que impide la depuración.	Buscar en las páginas oficiales la información relacionada al fallo y reiniciar o reinstalar servicios, procesos o programas cuando sea necesario.
	Error en la conexión de dispositivos físicos.	Buscar en las páginas de los fabricantes previo a la ejecución de pruebas la lista de requisitos para el modo de depuración en el sistema operativo huésped. (Uso de drivers especiales, configuraciones de montaje, fiabilidad de los periféricos de conexión, etc...).

La incapacidad de manejar otras amenazas y vulnerabilidades que no estén contempladas en este plan deberá ser informada a la brevedad al líder de proyecto para que se tomen las acciones adecuadas a las políticas de la organización.

Responsables

Nombre (s)	Responsabilidad
Marco Salcedo	Pruebas Android
Marco Salcedo	Pruebas iOS
Marco Salcedo	Pruebas Windows Phone*

* Solo si aplica la entrega a tiempo del código fuente.

Pruebas a aplicar

A continuación se enlistan los bloques de pruebas generales a aplicar, de los cuales se desglosan pruebas puntuales y un calendario para dichas pruebas basados en la calendarización de las fases de construcción de software que se lleven a cabo.
Para cada prueba individualmente se debe especificar:

1. Duración de la prueba
2. Alcance
3. Restricciones

4. Responsable de reportar resultados
5. Fecha de inicio
6. Reporte final de resultados
7. Elementos objetivo de la elaboración de la prueba (personas, proceso automatizados, testers)

Tabla 3.7: Flujo de aplicación Epoc

ID	Ámbito	Tipo de prueba	Fecha o fase de aplicación	Descripción	Criterios de Aprobación	Criticidad
CH1	Navegación en la interfaz de usuario	Validación contra checklist	12/12/12	Revisar contra los casos de uso que se cumplan todas las características requeridas a través de las vistas navegables. El checklist se elabora de manera personalizada de acuerdo al proyecto y a la arquitectura provista.	Todos los puntos del checklist aprobados	BAJA
CE1	Intuitividad de la interfaz de usuario	Validación contra cuestionario por escalas	12/12/12	El cuestionario por escalas se elabora de acuerdo a los puntos más importantes en las guías de diseño de cada plataforma y de acuerdo a los componentes usados en el proyecto.	Se definirá un rango mínimo de aceptación del 90% de satisfacción en la escala definida.	BAJA

				Se necesitan sujetos de prueba pertenecientes al mercado objetivo del proyecto.		
CH2	Nivel de calidad de la interfaz de usuario a nivel diseño gráfico.	Validación contra checklist	12/12/12	El cuestionario por escalas se elabora de acuerdo a los puntos más importantes en las guías de diseño de cada plataforma y de acuerdo a los componentes usados en el proyecto. Se necesitan expertos en diseño gráfico	Se definirá un rango mínimo de aceptación del 90% de satisfacción en la escala definida.	BAJA
CH3	Tolerancia a patrones anormales de uso de los elementos en la interfaz	Validación contra checklist	12/12/12	De acuerdo a los componentes y al flujo de vistas de la aplicación se crea una serie de pruebas puntuales cuyo objetivo es forzar el bloqueo de la aplicación producto de un uso exhaustivo y desmedido de los eventos de interfáz	Todos los puntos del checklist aprobados	MEDIA
CH4	Tolerancia a errores de acceso a medios	Validación contra checklist	12/12/12	De acuerdo a los componentes y al flujo de vistas	Todos los puntos del checklist aprobados	MEDIA

				de la aplicación se crea una serie de pruebas puntuales cuyo objetivo es generar los escenarios restrictivos de acceso a medios.		
CH5	Compatibilidad con las plataformas objetivo	Validación contra checklist	12/12/12	De acuerdo a las plataformas soportadas de la aplicación se intentará instalar el producto en todos las versiones soportadas y en las que no a fin de garantizar que el software tiene las configuraciones pertinentes para impedir que el sistema la instale en versiones no soportadas.	Todos los puntos del checklist aprobados	ALTA
CH6	Pruebas unitarias	Validación contra checklist	12/12/12.	Comprobar que todos los métodos devuelven los resultados esperados dados un conjunto iterativo de datos que abarcan los mejores y peores escenarios así	Todos los puntos del checklist aprobados	ALTA

				como datos erróneos o indefinidos.		

Cronograma de pruebas por bloques

Fecha de inicio del ciclo de pruebas	11/12/2012
Deadline de aplicación de pruebas (al menos un ciclo)	21/12/2012

Tabla 3.8: Profundidad de pruebas

Nivel de pruebas	Pruebas asociadas	Responsable	Tiempo estimado de aplicación de la prueba
Pruebas de caja negra, unitarias y de integración	CH6	Fuera del alcance del Departamento de Computación por no contar con el código fuente	Depende del proveedor del código.
Pruebas de estrés e integridad (compatibilidad, ejecución y fallos en la navegación)	CH1-3, CH5-6	UNAM Mobile + Departamento de Computación de Facultad de Ingeniería	1 dia, se pueden aplicar en paralelo a la misma revisión de código (versión)
Pruebas de experiencia de usuario y satisfacción	CH4, CE1	UNAM Mobile + Departamento de Computación de Facultad de Ingeniería	1 dia, se pueden aplicar en paralelo a la misma revisión de código (versión)

Las pruebas se repiten una vez enviada alguna corrección o propuesta de cambio cíclicamente hasta alcanzar el resultado establecido para cada una. por ejemplo, en un rango de calificación un 70% del valor máximo alcanzable en la prueba.

Un punto importante sobre el equipo de pruebas fue que se les pedía como único requisito tener un equipo que pudiera correr alguna versión de la aplicación, se usaron algunos equipos personales para los que no cumplieran este requisito, pero en

general era importante este punto, no sólo porque no se contó con presupuesto para adquirir equipos para las pruebas, sino también porque es importante para el fin de las pruebas verificar el funcionamiento en el mayor número posible de equipos, evidentemente es imposible probar en todos, pero por lo menos juntar una muestra significativa, agrupando los dispositivos por cualidades. Para iOS y Windows Phone esto es sencillo, puesto que existe en el mercado una cantidad bastante pequeña de dispositivos con estas características, el gran problema es Android. Para el caso de Android lo que se hizo fue agrupar los dispositivos bajo dos criterios, versión del sistema operativo (actualmente Google solicita que las aplicaciones corran en las versiones 2.1 y subsecuentes) y resolución de pantalla, de acuerdo a los lineamientos establecidos por Google hay cuatro posibilidades en este grupo, pequeña, normal, grande y extra grande, determinada por la densidad de píxeles de la pantalla del dispositivo.

Una vez completado el equipo de pruebas se procedió a aplicar las pruebas establecidas en el plan. Las primeras dos de ellas fueron básicamente una prueba de consistencia entre el producto y el diseño propuesto, estas no fueron hechas a público, puesto que sólo consistieron en comprobar que el flujo de la aplicación y los casos de uso coincidieran en la aplicación entregada y el diseño propuesto, en caso negativo se informa y se pregunta si este cambio fue intencional y por qué. La siguiente prueba, esta sí fue hecha al grupo de testers, es una prueba de percepción para verificar que la opinión del usuario promedio sobre la calidad de la interfaz gráfica y la usabilidad de la aplicación están al nivel de los estándares de la industria, son unas sencillas preguntas que se contestan con un simple sí o no. La siguiente prueba es la más importante de todas, en esta se evalúa la aplicación a nivel específico, se hacen preguntas tanto de percepción como de funcionalidad de las diferentes características de la aplicación a las cuales se les asigna una calificación según la escala de valor establecida. Al tiempo que se realizan estas pruebas sale de manera natural el llenado de la última prueba, el cómo se comporta la aplicación a través de diferentes dispositivos, incluyendo diferentes sistemas operativos, diferentes versiones de los mismos y diferentes resoluciones de pantalla dentro de un mismo sistema operativo. Los resultados de las pruebas se presentan a continuación:

Prueba CH1 para “Asistente Nutricional”
Tipo de prueba: Funcional
Enfoque: Evaluación del cumplimento de los casos de uso
Objetivo de la prueba
Revisar contra los casos de uso que se cumplan todas las características requeridas a través de las vistas navegables.
Requisitos de aplicación
Contar con un producto de software en fase beta o bien con la interfaz perfectamente desarrollada.
Público objetivo: Personas (testers) con conocimientos suficientes para interpretar la información contenida en los casos de uso y evaluar la interfaz

Tiempo máximo estimado de aplicación por usuario de prueba: 1 hora

Método de aplicación

A cada usuario de prueba se debe entregar una copia de este listado para revisar que en las vistas diseñadas se satisfagan todos los casos de uso, la información debe alimentarse en la tabla indicada.

Escala

Sí o no

Cuestionario

L1 - Casos de uso implementados en la aplicación

Haga una inspección de los casos de uso incluidos en la documentación e indique en qué vistas se cumplen o satisfacen en L1; si no se han incluido en el diseño o no se cumplen con el objetivo requerido, indicar en los casos de uso en la lista L2

Documentos de referencia:
https://docs.google.com/document/d/1tQBj5basNWtB2oZVscNvYKZ4AKz0KKi3Jx-33EoOiaI/edit

Tabla 3.9: CH1 asistente nutricional

ID	Vista	Caso de uso	Requerimiento/Referencia
1	CU_1	Se cumple con el flujo básico: 4.1.1 -> 4.1.2	ARM_CU1_IniciarControlNutricional.doc

2	CU_2	Los datos del usuario se están almacenando	ARM_CU2_AdministrarAplicacion.doc
3		El cálculo de IMC es correcto	
4	CU_3	Se cumple con el flujo básico: 4.1.1 -> 4.1.2 -> -> 4.1.3 -> -> 4.1.4	ARM_CU3_VisualizarAvance.doc
5		La gráfica es legible	
6	CU_4	Se cumple con el flujo básico: 4.1.1 -> 4.1.2 -> -> 4.1.3 -> -> 4.1.4	ARM_CU4_ConfigurarRecordatoriosEjercicio.doc
7		Se cumple con el flujo alterno de agregar ejercicio: 4.2.1 -> 4.2.2 -> -> 4.2.3 -> -> 4.2.4 -> -> 4.2.5 -> -> 4.2.6 -> -> 4.2.7 -> -> 4.2.8 -> -> 4.2.9 -> -> 4.2.10 -> -> 4.2.11	
8		Se cumple con el flujo alterno de eliminar ejercicio: 4.2.1 -> 4.2.2 -> -> 4.2.3 -> -> 4.2.4 -> -> 4.2.5 -> -> 4.2.6	

9		Los ejercicios se están almacenando y borrando correctamente	
10	CU_5	Las horas se almacenan correctamente	ARM_CU5_ConfigurarRecordatoriosComidas.doc
11		La diferencia de horas se calcula bien y muestra las advertencias correspondientes	
12	CU_6	Los recordatorios aparecen a las horas indicadas	ARM_CU6_EjecutarRecordatoriosEjercicios.doc
13	CU_7	Los recordatorios aparecen a las horas indicadas	ARM_CU7_EjecutarRecordatoriosComidas.doc

3.10: Resultados de la prueba CH1

Se entregará un reporte con las estadísticas generales en el siguiente formato:

ID de caso de uso evaluado / comentarios	**Implementado? (Sí o No)**
1, este flujo está ausente, favor de informar si este cambio de diseño estaba planeado	No
2,	Sí
3, según esta fuente (http://www.indicemasacorporal.org/index.php) es correcto	Sí

4,	Sí
5,	Sí
6,	Sí
7,	Sí
8,	Sí
9,	Sí
10,	Sí
11,	Sí
12,	Sí
13,	Sí

Total de casos de uso aplicados
___12___ de 13

Criterio de aceptación
Más del 90% del total de requerimientos/casos de uso presentes en la interfaz

Resultado de la prueba:
Esta aplicación ha *APROBADO* el criterio de aceptación.

Prueba CE1 para "Asistente Nutricional"

Tipo de prueba: Funcional
Enfoque: Evaluación del diseño gráfico de la interfaz de usuario
Objetivo de la prueba
Intuitividad de la interfaz de usuario
Requisitos de aplicación
Contar con un producto de software en fase beta o bien con la interfaz perfectamente desarrollada.
Público objeto: Usuarios finales de prueba o evaluación del diseño de arquitectura por medio de analistas. Al menos 10 usuarios de prueba.
Tiempo máximo estimado de aplicación por usuario de prueba: 1 hora

Método de aplicación

A cada usuario de prueba se debe entregar una copia de este cuestionario con escalas para medir nivel de satisfacción basado en la intuitividad de la interfaz de usuario.
Los puntos que tengan una calificación menor a buena deberán ser entregados al equipo de diseño de interfaces para que evalúen otras opciones de presentación con el fin de elevar la calificación en la prueba si la aplicación no obtiene el

Escala

Descripción	Valor numérico
Totalmente de acuerdo	5
Bastante de acuerdo	4
De acuerdo	3
En desacuerdo	2
Totalmente en desacuerdo	1

Cuestionario

Instrucciones

Elija el valor numérico que piense adecuado para la pregunta

No	Pregunta	Valor
1	Al entrar a la aplicación, ¿Cómo calificaría los íconos que se muestran en la primera vista, dan una buena idea de su funcionalidad?	
2	En la vista de configuración, ¿Cómo calificaría los texto descriptivos de todas las opciones?¿Queda claro su funcionamiento?	
3	¿La gráfica le parece legible?	
4	Si no logra encontrar la opción deseada, ¿Qué tan fácil es buscar ayuda dentro de la aplicación?	
5	Si deseara salir de alguna vista, ¿qué tan simple fue hallar la forma de hacerlo?	
6	Al añadir un nuevo recordatorio de ejercicio o comida, ¿fue sencillo realizarlo?	
7	Al momento de que suene la alarma de horario para el ejercicio programado, ¿Qué tan efectiva es dicha alarma?	

8	Al momento de que suene un recordatorio de comida, ¿Qué tan efectiva es dicha alarma? ¿Le quedó claro el propósito de la misma?	

3.11: Resultados de la prueba CE1

A continuación se muestran los resultados promedio

Pregunta	Valor Promedio Android	Valor Promedio iOS	Comentarios
1	4	4	-
2	5	4	-
3	4	3	-
4	1	4	Ahora desapareció la ayuda que ya estaba en Android
5	3	5	Como podemos ver, la forma de cambiar de vistas está muy en sintonía con la experiencia de uso de iOS, pero a los usuario android los desconcierta un poco
6	4	4	-
7	4	4	
8	4	4	

Valor promedio total de la aplicación

4

Valor promedio total superior al 90% del total de puntos.

Resultado de la prueba:

Esta aplicación ha *APROBADO* el criterio de aceptación para la versión iOS.*

Esta aplicación ha *APROBADO* el criterio de aceptación para la versión Android.**

*A pesar de tener visto bueno, hay todavía algunas sugerencias que se deben tomar en cuenta, ya que podrían llevar a un rechazo de Apple

https://docs.google.com/document/d/1pAbZSkv8j6JoDTlpl3WSJE4r30XursoAngV5jCycOS0/edit

**Sólo hay que regresar la ayuda

Al ir haciendo las pruebas de percepción los usuarios de prueba también expresaron algunas opiniones sobre cómo piensan que se podrían mejorar las aplicaciones. Como no estaba amparado dentro del plan de pruebas original pero algunas opiniones parecían valiosas, se creó un documento adicional que no estaba amparado en el diseño original del plan de pruebas, un documento para reportar bugs y sugerencias de los usuarios. Esto probó ser un gran acierto más adelante, puesto que le dio a los desarrolladores una guía rápida y concreta de dónde estaban los errores y situaciones que tenían que atender con más urgencia. A continuación se muestra un ejemplo de cómo se plasmaron en los documentos las observaciones de los usuarios:

Reportes de Usuarios para "Asistente Nutricional"

Elaboró:

- Marco Salcedo

Los siguientes son reportes de bugs y sugerencias generales del grupo de usuarios de prueba:

BUGS

Plataforma	Pasos para reproducir	Descripción
Todas	Introducir recordatorio de una comida tardía (por ejemplo cena) a una hora temprana y después introducir recordatorio de una comida temprana (ejemplo desayuno) antes de la anterior	Esto no se debería de poder hacer, el sistema debería poder identificar el orden de las comidas y no permitir al usuario poner la cena antes del desayuno

SUGERENCIAS

- Tener mayor variedad de Clasificaciones, porque los usuarios que presentaron sobrepeso les interesó saber cuánto sobrepeso tenían, y la aplicación no da esos datos. Podría haber clasificaciones de Sobrepeso 1-5kg, Sobrepeso 6-10 kg, y así sucesivamente
- Hacer más grandes las leyendas en la gráfica de avance

- En iOS, que el teclado que aparece para introducir valores en la pantalla de configuración sea un teclado numérico únicamente, Apple es muy quisquilloso con estos detalles, podría llevar a un rechazo de la app en la tienda
- En la última versión de Android volvió además a desaparecer la ayuda, hay que volver a ponerla

Como se mencionó anteriormente el proceso de pruebas es iterativo, a menos que se dé un caso ideal en el cual el primer periodo de aplicación del plan de pruebas no detecte ningún problema. En este caso en concreto eso no ocurrió, las pruebas se tuvieron que aplicar varias veces. En el caso de asistente nutricional la nota aprobatoria fue alcanzada hasta la tercera iteración. En la primera iteración falló en dos pruebas, parecía que había funcionalidad que no estaba implementada, y si lo estaba no funcionaba, y además los usuarios reportaban que era una aplicación muy confusa, que para ser algo muy sencillo tenía una curva de aprendizaje muy empinada. En la segunda versión entregada se implementó toda la funcionalidad y se añadió una vista de ayuda, pero desafortunadamente tenía bugs en funcionalidad clave de la aplicación, en ocasiones los recordatorios no sonaban, entonces se tuvo que regresar y pasar por un tercer período de pruebas, el cuál fue lo suficientemente satisfactorio.

En Epoc sólo se tuvieron que aplicar dos veces las pruebas, en la primera versión que llegó a manos del equipo de pruebas todo parecía estar muy bien, salvo por el detalle del botón de pánico, no mandaba ni mensajes SMS ni emails, no es posible hacer suficiente énfasis en lo grave que esto era, si la aplicación salía así al público podría incluso costar vidas, puesto que el usuario estaría confiado a que si tiene una crisis respiratoria con presionar un botón alguien sería informado y llegaría en su ayuda, pero eso no pasaría. Debido a esta falta grave, que independientemente de la parte humana, para fines de la ingeniería de software también es funcionalidad no implementada, se tuvo que solicitar que se corrigiera ese módulo, en la segunda versión ya funcionaba y se dio la nota aprobatoria a esa aplicación.

IV. Conclusiones

En general la metodología completa funcionó bastante bien, los clientes de HDS tuvieron comentarios mayormente positivos, y el proceso ayudó significativamente a que las aplicaciones se entregaran en tiempo y forma, esto es, con todos los requerimientos que pidieron y en la fecha acordada. Además al finalizar el proyecto terminaron con aplicaciones mejores de las que los clientes en un inicio habían planeado, puesto que se tomó en cuenta la retroalimentación de los usuarios, lo cual sirvió para mejorar el diseño propuesto.

Lo que aquí se presentó fue una metodología eficaz y capaz prevenir problemas en el momento de hacer un lanzamiento público de una aplicación, haciendo una especie de simulacro real, aunque en escala menor, de lo que pasaría si ese fuera en efecto un lanzamiento de la aplicación, debido a los diferentes perfiles de usuarios incorporados en el proceso. Así mismo le sirve a los desarrolladores para evitar pasos redundantes durante el proceso de realimentación, ya que les ayuda a puntualizar el problema por plataforma, familia de dispositivos, versión del sistema operativo, o alguna otra característica que le parezca relevante. Gracias a que dentro de los perfiles de usuario dentro del proceso de prueba también hay gente que tiene conocimientos de desarrollo y sabe cómo llenar apropiadamente un reporte de bug u obtener un vaciado de crash e interpretar los códigos de error. Habiendo partido de una metodología estándar de la IEEE fue sencillo incorporarla al proceso de desarrollo, porque le era familiar a la mayoría de los involucrados, y la decisión de optimizarla para móviles, quitando algunos entregables para la parte específica de las pruebas permite ahorrar un poco más de tiempo, añadiendo más a la eficacia de la metodología.

Al usar simuladores y dispositivos prestados para realizar la pruebas tampoco resulta ser una metodología cara de aplicar, y aunque esto suene limitante, la verdad es que es sorprendentemente fácil encontrar gente dispuesta a prestar sus dispositivos para una prueba rápida, entre amigos, conocidos y familiares, además esta resulta ser una muestra muy significativa de equipos, puesto a que así nos topamos con dispositivos de todas gamas, marcas y modelos.

La metodología probó ser efectiva durante el desarrollo de las aplicaciones de HDS, se pudieron entregar los diagnósticos, reportes de bug y sugerencias en tiempo y forma, conduciendo a rápida respuesta por parte de los desarrolladores y una agilización generalizada del desarrollo.

A mí en lo particular este trabajo me ayudó a revisar detalladamente el proceso de pruebas de software y me di cuenta de lo importante que es este proceso, porque los usuarios tienen el derecho a recibir software de calidad, y los usuarios de teléfonos móviles, que pueden no estar tan familiarizados con la tecnología como los usuarios de computadora, tienen menos paciencia respecto a los problemas que tenga una pieza de software, así que en el caso de las aplicaciones móviles es incluso más importante. También me ayudó a probar mis capacidades profesionales en un entorno profesional, desarrollando un proyecto de la vida real, con un cliente externo y dinero de por medio.

Creo que este fue un proyecto importante para la sociedad porque el mercado de las aplicaciones móviles está en auge, y es importante que México entre a competir en este mercado, puesto que es una oportunidad importante para entrar de lleno en el medio del desarrollo tecnológico en el que desafortunadamente estamos muy atrasados. Cualquier herramienta que se pueda aportar para facilitar esta transición debe ser bienvenida, y esta metodología realmente es un apoyo en ese sentido.

Glosario

Acelerómetro
Un componente de dispositivos que detecta cambios en la orientación física del mismo.

Acción
Una secuencia de interacciones del usuario que lleva a cabo un pequeño paso. Por ejemplo, iniciar una aplicación (desplácese hasta el icono de la aplicación, toca el icono)

Actor
Una especie o grupo de personas que utilizan una aplicación, por ejemplo, los clientes, comerciantes, soporte técnico.

Agnosticismo
En el software, indica que una tecnología o técnica se pueden usar en más de un entorno o plataforma.

Aprobación
Un veredicto de prueba que se da cuando los resultados de la pruebas concuerdan con los resultados esperados y el equipo de pruebas conviene que la aplicación tiene calidad suficiente para una salida pública.

APN
Del inglés Acces Point Name es el nombre de la puerta de enlace que debe configurarse en un dispositivo para poder acceder a redes GPRS, 3G o 4G.

App
Un programa de software, servicio o sitio web interactivo concebido para correr en un dispositivo móvil, como puede ser un smartphone o una Tablet. Normalmente las apps son una unidad pequeña e individual de software, que tiene un uso muy particular y limitado.

Bloqueo

Un veredicto de prueba dado cuando un paso de prueba no se puede realizar debido a un bug o a un estado de terminación de la aplicación aún no alcanzado.

Bug
Una falla observada en la aplicación bajo prueba: salida incorrecta, suspensiones inesperadas, congelamiento, accidentes, respuesta muy lenta, fallos de seguridad son todos ejemplos de bugs. Una característica que falta o está mal diseñada también se puede considerar un bug en ciertas condiciones, como un lanzamiento público.

Caso de uso
Una secuencia de acciones que podría llevar a cabo un usuario promedio.

Campo de texto
Un control que acepta datos alfanuméricos a partir de un teclado físico o virtual

Control
En términos genéricos cualquier elemento gráfico en la pantalla del dispositivo (botón, deslizador de desplazamiento, checkbox, etc.) que se utiliza para mandar señales a la aplicación.

Diagrama de Gantt
Herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

Dispositivo
Una plataforma de hardware y software en el que se ejecuta una app.

Deploy
El lanzamiento de una aplicación para todos sus usuarios y entornos planeados.

Emulador
Un programa de software que proporciona una plataforma de hardware virtual en la que se puede ejecutar una app.

Escala de valor
Los posibles valores que puede dar el resultado de una prueba, los cuales tienen un significado bien definido.

Evento
Un estímulo reconocido por un dispositivo que puede tener un efecto sobre la ejecución de la aplicación bajo prueba. Por ejemplo, la pérdida de la red inalámbrica o una llamada o mensaje de texto entrante.

Fallo

Un veredicto de prueba que se da cuando los resultados reales no coinciden con los resultados esperados o la aplicación está en un estado de terminación inadecuado para poder terminar el ciclo de pruebas.

Flujo alterno
Los pasos adicionales o especiales de un caso de uso que pueden o no presentarse.

Flujo básico
Los pasos de un caso de uso que figuran en la totalidad de sus flujos

Flujo especial
Los pasos adicionales o especiales de un caso de uso que suceden sólo cuando se producen entradas o condiciones atípicas.

Flujo variante
Una secuencia de pasos de prueba para el flujo de un caso de uso causado por una decisión binaria en el flujo básico, por ejemplo, si el usuario decide ir a una pantalla de configuración en vez de saltar ese paso . Un caso de uso variante crea bifurcaciones en el flujo de la aplicación.

GPS
Global Positioning System (Sistema de Posicionamiento Global) es un sistema de posicionamiento a base de geolocalización satelital.

GSM
Global System for Mobile (Sistema Global para telecomunicaciones Móviles) es conjunto de estándares que describen la forma como se comunica un cliente de telefonía celular con el proveedor del servicio. Es el sistema más usado a nivel mundial por los operadores de tecnología celular.

GTP
GPRS Tunnelling Protocol (Protocolo de Tunel del Servicio General de Paquetes via Radio) es un grupo de protocolos de comunicaciones basados en IP que se usan para portar el servicio GPRS dentro de las redes GSM y UMTS.

Háptica
Una entrada o salida táctil o kinestésica en otra forma, por ejemplo, la vibración de un teléfono para indicar una llamada entrante.

IEEE
Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos). Una organización profesional internacional que publica una amplia gama de normas y estándares técnicos de computación y electrónica.

IP

Internet Protocol (Protocolo de Internet), principal protocolo de comunicación de la capa de red del modelo OSI. Es un protocolo bidireccional no orientado a conexión para la comunicación de datos digitales, transfiere paquetes conmutados a través de una o varias redes físicas.

Mutante
Un caso de uso que rompe las reglas de estructura de una variable, por ejemplo, el uso de punto decimal para una variable que se espera sea entera.

Tecla fija o Tecla dura
Un botón físico en un dispositivo o accesorio que puede ser presionado en contraste con uno que aparece bajo ciertas condiciones en una pantalla táctil.

Tecla suave
Un control virtual que aparece en la pantalla táctil de un dispositivo sólo bajo circunstancias especiales.

Pantalla táctil
Un tipo de dispositivo de visualización que además de servir como pantalla también hace las veces de dispositivo de entrada. En dispositivos móviles modernos, la mayoría de las pantallas táctiles aceptan entrada con los dedos además de algunos tipos de plumas.

Perfil de pruebas
Una descripción estructurada y simplificada de la aplicación bajo prueba que hace que sea fácil de producir casos de prueba.

Plan de pruebas
Colección de documentos realizados con el fin de planear y llevar el control de la totalidad del proceso de pruebas.

Protocolo
Conjunto de acciones, métodos, y reglas que constituye un procedimiento planificado y estructurado convencional, destinado a estandarizar un comportamiento o acción.

Pruebas
Uso del software con ojo crítico y con el fin de encontrar bugs u otro tipo de problemas.

Pruebas automáticas
Un estilo de prueba en la que un programa de software ingresa entradas de prueba, observa los resultados y los registra.

Pruebas manuales
Un estilo de prueba de funcionamiento en el que una persona ingresa las entradas de prueba, observa que los resultados reales, y registra veredictos.

Prueba negativa
Un tipo de prueba de stress que tiene como fin probar el comportamiento de la aplicación ante entradas o secuencias de acciones no esperadas.

Prueba típica
Un tipo de prueba que consiste en probar la aplicación bajo los casos de uso establecidos para un usuario promedio.

Scrum
Marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software.

Smartphone
Teléfono móvil construido sobre una plataforma informática móvil, con capacidad de almacenamiento de datos, conexión a Internet e instalación de aplicaciones similares a las de una computadora.

Tablet
Tipo de computadora portátil de tamaño pequeño, usualmente integrada en una pantalla táctil con la cual se puede interactuar sin necesidad de un teclado y/o un ratón. Son similares a los smartphones, pero por lo regular son de mayor tamaño y no incluyen funciones de un teléfono celular convencional.

Bibliografía y Mesografía

STELLMAN, Andrew et al., “Applied Software Project Management”, O’Reilly, 2005

GOLDSTEIN, Neil, “iPhone Application Development”, Wiley Publishing, 2010

JANSSEN, Cory, “What is a mobile application?”, recuperado en Noviembre de 2013 http://www.techopedia.com/definition/2953/mobile-application-mobile-app

BUCKY, James, “Definition of a mobile device”, recuperado en Noviembre de 2013 http://operationstech.about.com/od/glossary/g/Definition-Of-Mobile-Device.htm

PC Magazine, “Smartphone definition”, recuperado en Noviembre de 2013 http://www.pcmag.com/encyclopedia/term/51537/smartphone

Keynote systems, “Testing strategies and tactics for mobile applications”, recuperado en Noviembre de 2013 http://www.keynote.com/docs/whitepapers/WP_Testing_Strategies.pdf

BINDER, Robert, “Course Notes”, recuperado en Noviembre de 2013 http://robertvbinder.com/docs/arts/H2TMA-Course-Notes.pdf

IEEE, “Test plan outline”, recuperado en Noviembre de 2013 http://www.computing.dcu.ie/~davids/courses/CA267/ieee829mtp.pdf

Wikipedia, “Enfermedad pulmonar obstructiva crónica”, recuperado en Abril de 2013 http://es.wikipedia.org/wiki/Enfermedad_pulmonar_obstructiva_cr%C3%B3nica

Android Developer center, “Supporting multiple screens”, recuperado en Mayo de 2013 http://developer.android.com/guide/practices/screens_support.html

IEEE,”IEEE 829”, recuperado en Abril de 2013 https://www.google.com.mx/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CDsQFjAB&url=https%3A%2F%2Fcow.ceng.metu.edu.tr%2FCourses%2Fdownload_courseFile.php%3Fid%3D5148&ei=PLxRUtfYIYH-9gSwpIDQBw&usg=AFQjCNFkTm75PsLFRHxBuYMhgbqCtlxUWw&sig2=4ZY73Ug-jukuRFL5L8Wcqw&bvm=bv.53537100,d.eWU&cad=rja