



UNIVERSIDAD DE
COSTA RICA



NORMA PARA EL DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES EN LA UNIVERSIDAD DE COSTA RICA

Fecha de emisión o actualización: 12/09/2018

Código: CI-N02

Versión: 1.0

Tabla de contenidos

1 INTRODUCCIÓN.....	4
2 PROPÓSITO.....	5
3 CONSIDERACIONES GENERALES.....	5
4 LEYES, REGLAMENTOS O DOCUMENTOS DE REFERENCIA.....	6
5 CONTENIDO DE LA NORMA.....	7
5.1 Seguridad.....	7
5.1.1 Gestión de riesgos.....	9
5.1.2 Prácticas básicas de seguridad.....	9
5.2 Capacidad y rendimiento.....	11
5.2.1 Almacenamiento y memoria.....	11
5.2.2 Rendimiento.....	12
5.3 Lineamientos de contenido.....	13
5.4 Accesibilidad y usabilidad.....	14
5.4.1 Lineamientos generales.....	14
5.4.2 Navegación.....	16
5.4.2.1 Tabs.....	16
5.4.2.2 Side Menu.....	18
5.4.2.3 Combinaciones.....	20
5.4.3 Compatibilidad.....	22
5.5 Diseño gráfico.....	22
5.5.1 Ícono.....	22
5.5.1.1 Buenas prácticas y plantillas para iOS.....	23
5.5.1.2 Buenas prácticas y plantillas para Android.....	24
5.5.2 Colores.....	25
5.5.3 Pantallas.....	25
5.5.4 Menús y otros elementos.....	25
5.5.5 Efectos especiales.....	26
5.5.6 Aspectos propios del sistema operativo.....	26
5.6 Formato y contenido de la documentación interna y externa.....	26
5.6.1 Documentación externa mínima.....	26
5.6.1.1 Documentos de análisis y diseño.....	27
5.6.1.2 Documentos de comunicación con aplicaciones externas.....	29
5.6.1.3 Documentación final.....	30
5.6.2 Documentación interna.....	31
5.7 Arquitectura.....	32
5.7.1 Tipos de aplicaciones móviles, según su desarrollo.....	32
5.7.2 Aplicaciones móviles nativas.....	32
5.7.3 Aplicaciones móviles híbridas.....	32

5.7.4 ¿Qué tipo utilizar?.....	33
5.7.5 Arquitectura de Modelo Vista Controlador.....	35
5.7.5.1 Modelo.....	36
5.7.5.2 Vista.....	36
5.7.5.3 Controlador.....	37
5.7.5.4 Comunicación.....	37
5.7.6 Comunicación con aspectos propios del dispositivo.....	37
5.8 Lenguajes, APIs y convenciones de programación.....	39
5.8.1 IOS.....	39
5.8.1.1 Objective-C.....	39
5.8.1.2 Swift.....	40
5.8.1.3 Conceptos clave del API.....	41
5.8.2 Android.....	42
5.8.2.1 Java.....	42
5.8.2.2 Kotlin.....	43
5.8.2.3 Conceptos clave del API.....	43
5.8.3 Aplicaciones híbridas.....	44
5.8.3.1 HTML, CSS, Javascript.....	44
5.8.3.2 Cordova.....	45
5.8.3.3 Frameworks.....	45
5.9 Estructuras e integración de datos.....	46
5.9.1 Recomendaciones para datos locales.....	46
5.9.2 Recomendaciones para datos remotos.....	47
5.9.3 Requisitos específicos de la Institución y aspectos legales.....	47
5.10 Publicación.....	49
5.10.1 Contenidos del paquete de publicación.....	51
5.10.1.1 Contenidos para la publicación en App Store (iOS).....	52
5.10.1.2 Contenidos para la publicación en Play Store (Android).....	53
5.10.2 Otros lineamientos de publicación.....	54
5.10.3 Reportes.....	54
6 CONCEPTOS Y ABREVIATURAS.....	55
7 APROBACIÓN.....	57

1 INTRODUCCIÓN

El presente documento constituye una norma que se requiere aplicar en el proceso de análisis, diseño, desarrollo e implementación de aplicaciones móviles (apps) en la Universidad de Costa Rica. Es, por lo tanto, un conjunto de responsabilidades técnicas y administrativas que se deben seguir para asegurar la calidad del desarrollo dichas aplicaciones.

La finalidad de este documento es que las aplicaciones móviles creadas por las dependencias de la Universidad de Costa Rica sigan un mismo patrón de diseño tanto arquitectural como de diseño gráfico, de tal forma que se uniformen aspectos de calidad, funcionalidad y diseño.

Durante la creación de este documento se recolectaron y revisaron referencias relacionadas con el tema del desarrollo de aplicaciones móviles. Además, se llevaron a cabo sesiones de trabajo con diversos expertos del Centro de Informática en temas de desarrollo, seguridad, usabilidad, contenido, diseño gráfico, publicación y estructuras de datos, con el fin de aplicar y adaptar los aspectos teóricos del desarrollo de apps al contexto de la Universidad de Costa Rica.

El desarrollo de las normas se divide en diez secciones:

- Normas de seguridad, identificación de riesgos y buenas prácticas para mitigar dichos riesgos en el ámbito de desarrollo móvil.
- Normas de capacidad y rendimiento que incluyen el manejo de memoria y rendimiento general de la aplicación.
- Normas relacionadas con los contenidos creados por miembros de la Universidad de Costa Rica para las aplicaciones móviles.
- Buenas prácticas para la accesibilidad y usabilidad, donde se busca mejorar la experiencia del usuario con las funcionalidades de la aplicación y la compatibilidad de la misma con diversos sistemas y dispositivos.
- Diseño gráfico, donde se definen los lineamientos de calidad y uniformidad gráfica de las aplicaciones.
- Documentación necesaria para presentar y desarrollar una aplicación móvil de la Universidad de Costa Rica. Incluye la documentación interna y externa.
- Arquitectura que presentan los diferentes tipos de desarrollo y propuesta de las mejores prácticas para elegir el tipo de desarrollo y creación la arquitectura de la aplicación.

- Uso de lenguajes, APIs y convenciones de programación para el desarrollo de aplicaciones móviles así como las buenas prácticas para seguir la arquitectura de la aplicación.
- Lineamientos de estructuras de datos, tanto locales como remotas, así como los procesos de integración e integridad con los datos de los usuarios de la Universidad de Costa Rica.
- Proceso de publicación y de solicitud de reportes que las Unidades de la Universidad de Costa Rica pueden pedir a los encargados de las cuentas de publicación de las aplicaciones en el Centro de Informática.

Es importante destacar que este es un documento dinámico y se debe actualizar constantemente con el cambio y avance de las tecnologías relacionadas al desarrollo de aplicaciones móviles y de las diversas plataformas de la Universidad de Costa Rica.

2 PROPÓSITO

El propósito principal de este documento es uniformar el desarrollo de las aplicaciones móviles de la Universidad de Costa Rica, para lo cual se enfoca en los siguientes ámbitos de calidad:

- Seguridad
- Capacidad y rendimiento
- Contenido
- Accesibilidad y usabilidad
- Diseño gráfico
- Documentación
- Arquitectura
- Desarrollo
- Estructura e integración de datos
- Publicación

3 CONSIDERACIONES GENERALES

Este documento es el único conjunto de normas oficial aprobado para el modelado, diseño, desarrollo y publicación de aplicaciones móviles en la Universidad de Costa Rica. Por lo tanto, solamente serán consideradas como aplicaciones móviles oficiales de la Institución aquellas apps que cumplan con los requerimientos técnicos y con los procedimientos de publicación descritos en este documento.

Aunado a ello, se debe enfatizar el hecho de que las apps que se publiquen, deben estar relacionadas intrínsecamente con el quehacer de la Universidad de Costa Rica.

4 LEYES, REGLAMENTOS O DOCUMENTOS DE REFERENCIA

- [1] Mobile App Security Guide [Infográfico]. (n.d.). Tomado de <http://autosend.io/mobile-app-security-guide/>
- [2] Security Tips [Artículo]. (n.d.). Tomado de <https://developer.android.com/training/articles/security-tips.html>
- [3] Secure Coding Guide [Manual]. (2014, Febrero 12). Tomado de <https://developer.apple.com/library/mac/documentation/Security/Conceptual/SecureCodingGuide/Introduction.html>
- [4] Managing Your App's Memory [Artículo]. (n.d.). Tomado de <http://developer.android.com/training/articles/memory.html>
- [5] Jandial, M., Somasundara, A., Karthikeyan. (2012, Marzo). Enhance mobile application performance with network testing [Artículo]. Tomado de <http://searchsoftwarequality.techtarget.com/tip/Enhance-mobile-application-performance-with-network-testing>
- [6] Shema, M. (2011, Mayo 31). Web Security: Why You Should Always Use HTTPS [Artículo]. Tomado de <http://mashable.com/2011/05/31/https-web-security/#eoQgnh3zksqf>
- [7] Smartface Get Everything Done About Memory Management For You [Artículo]. (n.d.) Tomado de <https://www.smartface.io/smartface-memory-management/>
- [8] Nielsen, J. (2010, Junio 21). Website Response Times [Artículo]. Tomado de <https://www.nngroup.com/articles/website-response-times/>
- [9] Cerejo, L. (2012, Julio 12). The Elements of Mobile User Experience [Artículo]. Tomado de <http://www.smashingmagazine.com/2012/07/elements-mobile-user-experience/>
- [10] Nielsen, J. (2005, Julio 11). Scrolling and Scrollbars [Artículo]. Tomado de <https://www.nngroup.com/articles/scrolling-and-scrollbars/>
- [11] Best Practices for Designing Mobile Applications [Artículo]. (2009, Agosto 26). Tomado de <http://blog.gfk.com/2009/08/best-practices-for-designing-mobile-applications/>
- [12] Iphone side navigation vs. Tab bar Navigation [Foro]. (2013, Octubre 22). Tomado de <http://ux.stackexchange.com/questions/40204/iphone-side-navigation-vs-tab-bar-navigation>
- [13] Rose, A. (2014, Abril 8). UX designers: Side drawer navigation could be costing you half your user engagement [Artículo]. Tomado de <http://thenextweb.com/dd/2014/04/08/ux-designers-side-drawer-navigation-costing-half-user-engagement/>
- [14] Iphone App Icon Design: Best Practices [Artículo]. (n.d.). Tomado de

- <http://www.pixelresort.com/blog/iphone-app-icon-design-best-practises/>
- [15] App Icon [Manual]. (2015, Noviembre 5). Tomado de https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/AppleCons.html#//apple_ref/doc/uid/TP40006556-CH19-SW1
- [16] Flarup, M. (2015, Julio 2). Designing Android Product Icons [Artículo]. Tomado de <http://appicontemplate.com/android-product-icons/>
- [17] Simon, G. (2013, Setiembre 5). The Differences Between iOS and Android Icon Design [Video]. Tomado de <http://design.tutsplus.com/courses/designing-icons-for-ios-and-android/lessons/the-differences-between-ios-and-android-icon-design>
- [18] Sasa, W., Aguilar, E. (2015, Mayo). Manual de identidad visual. San José, Costa Rica: Universidad de Costa Rica.
- [19] Sample Test Case Template With Examples [Archivo]. (2015, Noviembre 3). Tomado de <http://www.softwaretestinghelp.com/test-case-template-examples/>
- [20] Web Services API Specification Doc Template [Archivo]. (2013, Enero 5). Tomado de <http://rebugged.com/web-service-api-specification-doc-template/>
- [21] Documentation Guidelines [Artículo]. (n.d.). Tomado de <http://web.cs.wpi.edu/~cs1005/common/documentation.html>
- [22] Korf, M., Oksman, E. (2015, Abril). Native, HTML5 or Hybrid: Understanding Your Mobile Application Development Options [Artículo]. Tomado de https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options
- [23] The App Life Cycle [Manual]. (2015, Octubre 21). Tomado de https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html#//apple_ref/doc/uid/TP40007072-CH2-SW2
- [24] Application Fundamentals [Artículo]. (n.d.). Tomado de <http://developer.android.com/guide/components/fundamentals.html>

5 CONTENIDO DE LA NORMA

5.1 Seguridad

La rápida expansión y popularidad de los dispositivos móviles inteligentes, ha fomentado que el desarrollo y la publicación de las apps sea muy acelerado. Esto se traduce en que los

desarrolladores den mayor prioridad a la programación de los requerimientos funcionales, en detrimento de los requerimientos relacionados con la seguridad. Por ello, es necesario abordar el diseño de la seguridad, tomando en cuenta que las aplicaciones móviles con vulnerabilidades de seguridad tienen el potencial de generar [1]:

- Acceso no autorizado a datos sensibles
- Robo de propiedad intelectual
- Fraude
- Experiencia del usuario alterada
- Daño a la marca
- Robo de identidad

Los frameworks de desarrollo de aplicaciones móviles y las habilidades nativas de los sistemas operativos están diseñados de forma que en su estándar la mayoría de las medidas de seguridad estén ya tomadas a la hora de desarrollar las aplicaciones [2, 3]. Sin embargo, con el fin de minimizar los riesgos, existen aspectos de seguridad que deben ser tomados en cuenta durante el desarrollo.

Desde el punto de vista de riesgos, se deben tomar en cuenta los dos aspectos siguientes:

- **Las aplicaciones móviles normalmente utilizan redes públicas no seguras:** Normalmente los usuarios utilizan las aplicaciones en ambientes y redes públicas, las cuales son poco seguras, como por ejemplo las redes de celular o redes inalámbricas públicas, como las que se utilizan en centros comerciales, restaurantes y comercios dentro y fuera del país. Debido a esto, los mensajes entre la aplicación y los servidores de datos que utiliza pueden ser interceptados fácilmente por visores externos, si no se toman las medidas de seguridad necesarias.
- **Los dispositivos móviles pueden ser robados y utilizados por otras personas:** Muchos usuarios tienen las sesiones de sus aplicaciones móviles permanentemente activas en sus celulares. Esto ayuda a que su usabilidad sea mejor y más cómoda para el usuario. Sin embargo, en caso de robo, los usuarios pueden ser víctimas de

robo de identidad o pérdida de datos sensibles.

5.1.1 Gestión de riesgos

La gestión de riesgos en un proyecto de desarrollo de aplicaciones móviles debe efectuarse tanto para la fase de desarrollo (riesgos asociados a la gestión del proyecto), como para la puesta en producción.

Una respuesta de “Sí” a alguna de las siguientes preguntas identifica una aplicación móvil expuesta a más riesgos [1]:

- ¿La aplicación se comunica con uno o varios servidores externos?
- ¿La aplicación guarda información del usuario?
- ¿La aplicación puede requerir en algún momento saber la geo-localización del usuario?

Aplicaciones que no necesitan comunicación con la web o que son meramente informativas y no guardan información del usuario por lo general están expuestas a menos riesgos.

Para llevar a cabo la gestión de riesgos, puede consultar los documentos **Metodología de Análisis de Riesgos** y el **Catálogo de Riesgos y Controles para el Desarrollo de Aplicaciones Móviles**, desarrollados por la Unidad de Riesgo y Seguridad (URS) del Centro de Informática.

5.1.2 Prácticas básicas de seguridad

En esta sección se detallan las prácticas mínimas de seguridad que debe tener una aplicación móvil [1, 6]:

- Mantener bibliotecas actualizadas: Independientemente del sistema operativo para el que se desarrolle la aplicación, el lenguaje de programación y los frameworks utilizados en cada versión, se debe revisar y comprobar que se están utilizando las bibliotecas más actualizadas. Los desarrolladores de las bibliotecas normalmente realizan mejoras en la seguridad, funcionalidad y rendimiento de sus bibliotecas, por lo que mantenerlas actualizadas brindará esos beneficios a la aplicación móvil.

- Utilizar conexiones y protocolos seguros para comunicarse con los servidores: Se deben utilizar solicitudes GET y POST en HTTPS para imágenes, documentos, credenciales y en cualquier dato importante que se tome de un servidor. El protocolo HTTPS cifra los datos de tal manera que sólo la aplicación y el servidor con el que se está comunicando puedan leerlos. De esta forma evita ataques conocidos como man-in-the-middle y eavesdropping que pueden comprometer los datos. HTTPS es lo mínimo que se debería usar, sin embargo se puede complementar con otros protocolos que incrementen la seguridad de las comunicaciones.
- Cifrar los datos que se guardan localmente en el dispositivo: Así como se deben cifrar los datos que viajan por las conexiones inseguras, se deben cifrar los datos que se guarden en el dispositivo. Tanto Android y iOS tienen bibliotecas para cifrar los datos que se guardan para una aplicación, y también hacen que estos datos puedan ser accedidos solamente por esta. No se deben guardar datos sensibles en memorias externas como SD Cards, ya que estos pueden ser accedidos por otras aplicaciones.
- Filtrar inputs para evitar inyección de SQL: Todos los inputs de la aplicación deben ser filtrados y validados para evitar ataques de inyección de SQL. No se debe obviar incluso aunque se sepa que el servidor no utiliza bases de datos en SQL o se sepa que el servidor filtrará los datos, ya que el servidor puede cambiar en cualquier momento manteniendo el servicio web que la aplicación utiliza.
- Revisiones de código fuente: Se recomienda programar revisiones de código fuente especialmente si se trabaja en un equipo. En cualquier caso, se deberá coordinar con el Centro de Informática para designar a un funcionario que pueda apoyar en las revisiones del código fuente.
- Solicitar contraseñas fuertes al usuario: En el caso de las aplicaciones de la Universidad de Costa Rica, se debe utilizar el sistema para autenticar a los usuarios que son estudiantes o funcionarios. Si por alguna razón se necesita tener usuarios que no son parte de la Universidad, sus contraseñas deben ser fuertes. Las contraseñas deben tener al menos 8 caracteres, contener letras en minúscula y mayúscula, números y caracteres especiales.
- No guardar contraseñas: Por ningún motivo se deben guardar contraseñas en el dispositivo. Las contraseñas deben ir cifradas en cualquier mensaje que se envíe al

servidor y en los servidores se deben guardar en forma cifrada.

- Poder desactivar sesiones desde algún sistema en el exterior: En toda aplicación se debe implementar una forma de desactivar cualquier sesión desde algún sistema externo a la aplicación móvil y al dispositivo. Esta práctica se necesita ante la eventualidad de que el usuario pierda su teléfono y necesite bloquear el acceso a la aplicación por medio de su cuenta.

5.2 Capacidad y rendimiento

Los dispositivos móviles tienen limitaciones en cuanto a capacidad de la memoria y almacenamiento, que deben ser consideradas por los desarrolladores. Asimismo, muchos de los datos deben ser descargados o enviados a servidores externos lo que impactará en los tiempos de carga y el rendimiento general de la aplicación móvil. Esta sección describe las prácticas mínimas por realizar con respecto a la capacidad y rendimiento de las aplicaciones móviles de la Universidad de Costa Rica.

5.2.1 Almacenamiento y memoria

- El almacenamiento interno debe ser utilizado mínimamente y de forma temporal. En la sección de seguridad se detalló que no es una práctica recomendable guardar datos en el dispositivo, además de esto se debe tener en cuenta que el almacenamiento de los dispositivos móviles es más limitado que en otros tipos de dispositivos. Así que el almacenamiento interno se debe utilizar solo para guardar pocos datos que deban ser utilizados continua y rápidamente [4, 7]. En la sección 11 de Estructuras e integración de datos se detallan las mejores maneras de utilizar el almacenamiento interno en dispositivos móviles.
- Si la aplicación necesita crear algún servicio para realizar trabajos en el segundo plano, estos no se deben mantener corriendo mientras no estén activamente realizando alguna tarea. Los servicios en segundo plano toman mucha RAM del dispositivo, que no puede ser utilizada luego por otras aplicaciones [4, 7].
- Se deben tener controles apropiados para la administración de memoria. Se debe liberar memoria mientras la aplicación móvil no esté activa o cuando la memoria

disponible del dispositivo sea poca [4, 7].

- Se debe minimizar el uso de memoria al cargar imágenes, además de liberar esa memoria tan pronto como la imagen se deje de utilizar [4, 7].
- Como regla general: Evitar asignar memoria que no se vaya a utilizar y evitarle al dispositivo trabajos que no se vayan a necesitar [4, 7].

5.2.2 Rendimiento

Es difícil predecir el comportamiento y rendimiento de los servidores externos a la aplicación, sin embargo los desarrolladores deben implementar las prácticas necesarias para que la aplicación móvil se comporte correctamente ante cualquier situación externa.

- Se debe probar la aplicación en distintos tipos de conexiones y velocidades disponibles en el país. Se debe probar tanto con conexiones WiFi como con conexiones de celular [5]. Adicionalmente, se deberán realizar pruebas en dispositivos sin conexión, para analizar el comportamiento de la aplicación móvil y sus respuestas al usuario.
- Cuando se esperen datos de servidores externos siempre se debe mostrar al usuario un spinner o algún texto que indique que los datos se están cargando.
- La aplicación no se debe congelar, cerrar o mostrar algún dato inconsistente mientras espera algún dato externo [5].
- El usuario debe poder salirse de cualquier pantalla aunque los datos no hayan sido descargados. En tal caso, se debe garantizar que los datos o porciones de datos que hayan sido descargados, sean eliminados de la memoria.
- Si una transacción toma más de diez segundos, se debe desplegar una notificación para que el usuario pueda decidir si esperar o cancelar dicha transacción [8].
- La aplicación debe comportarse consistentemente ante cualquier retardo o pérdida de los datos que se esperan de servidores externos e informar debidamente al usuario.

5.3 Lineamientos de contenido

Si bien es cierto, las aplicaciones móviles son un medio un poco más informal, se deben mantener los lineamientos de contenido generales que reflejen el nivel de educación superior de la Universidad de Costa Rica. Además, es importante que los contenidos permitan un entorno de alta usabilidad y buena experiencia a los usuarios. En esta sección se listan los lineamientos generales de contenido de las aplicaciones móviles:

- Al diseñar una aplicación, es importante considerar que su creación responde a un propósito específico y delimitar bien su funcionalidad. Si la aplicación tiene muchas funciones es importante dividirlas en módulos claros o incluso dividirla en varias aplicaciones más pequeñas y delimitadas (atomicidad). Lo importante es no abrumar al usuario y aumentar la usabilidad de cada una de las aplicaciones. Esto se ha visto en los últimos años con aplicaciones como Foursquare (dividida en Foursquare y Swarn) y Facebook (dividida en Facebook, Messenger, Pages, entre otras) que han dividido su aplicación principal en aplicaciones más pequeñas y más delimitadas con funcionalidades específicas y claras.
- El texto debe estar adecuadamente redactado, si es posible, revisado por una persona competente en el campo (como un comunicador, community manager o especialista del campo), con una adecuada ortografía y lenguaje apropiado que refleje el nivel de educación superior.
- Además, para cumplir con las políticas institucionales de información y comunicación, se recomienda hacer uso del lenguaje inclusivo de género.
- Cada aplicación móvil debe tener una sección de “Acerca de”. Es recomendable que se incluya en esta sección los medios disponibles para contactar a los responsables de la aplicación, por ejemplo, dirección de correo electrónico, número de teléfono, redes sociales, entre otros.
- Cada aplicación móvil debe desplegar el nombre completo de la Universidad de Costa Rica. Se debe utilizar cualquiera de las firmas oficiales de la Universidad de Costa Rica, las cuales se encuentran disponibles en el archivo digital que la Oficina de Divulgación e Información (ODI) facilita por medio de su sitio web odi.ucr.ac.cr y respetando los lineamientos estipulados en el manual de identidad visual institucional.

5.4 Accesibilidad y usabilidad

5.4.1 Lineamientos generales

- Los enlaces y botones deben contener textos o imágenes (íconos) representativos del contenido al que apuntan [11]. Además, deben ser intuitivos y fáciles de presionar con el dedo (hacer tap) en la pantalla táctil. Dichos enlaces necesitan tener un tamaño de al menos treinta píxeles (en tamaño de pantalla, no tamaño de resolución) de alto y de ancho que no dificulte hacer tap sobre ellos [9]. Si se usan imágenes en lugar de textos o botones, deben ser claramente representativas de su acción.
- Los campos de entrada de textos, los botones y otros tipos de elementos en formularios deben estar, apropiadamente, etiquetados. Dichos elementos deben tener un tamaño de al menos 30 píxeles (en tamaño de pantalla, no tamaño de resolución) de alto y de ancho que no dificulte hacer tap sobre ellos [9].
- Utilizar el tipo de teclado adecuado para los diferentes tipos de campos de entrada [9]. La siguiente ilustración muestra algunos de los distintos tipos de teclado disponibles en los dispositivos móviles, según el tipo de dato.

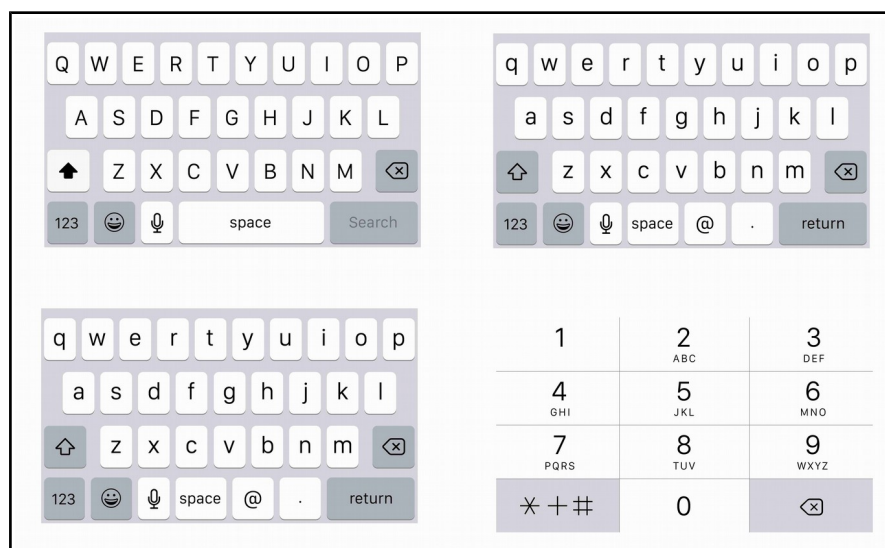


Ilustración 1. Tipos de teclado en dispositivos móviles.

- Debe existir un contraste adecuado entre el color del texto y el color del fondo, para así asegurarse de que los elementos del sitio son legibles para las personas con visión reducida o con daltonismo.
- Al entrar a un módulo específico de la aplicación, se debe evitar tener muchas pantallas anidadas dentro del mismo. Se debe tratar de mantener un máximo de tres pantallas anidadas [9] dentro de lo posible y permitir al usuario regresar fácilmente en la jerarquía de pantallas y a la pantalla principal del módulo. Siempre se debe informar claramente al usuario, por medio de títulos u otros medios de información, el propósito de cada pantalla y especialmente cuando se deben tener varias pantallas anidadas.
- Se debe minimizar el scrolling horizontal ya que no es intuitivo para el usuario [10]. Si se utiliza el scrolling horizontal como parte de una experiencia diferente para el usuario dentro de la aplicación debe ser muy claro.
- La información más importante de cada pantalla debe aparecer sin tener que hacer scroll. Cuando haya que hacer scroll debe ser claro para el usuario que hay más información fuera de la pantalla [10].
- En aplicaciones móviles, cuando se debe agregar una pantalla con un formulario de toma de datos, es más efectivo utilizar pantallas con scroll vertical que dividirlo en un diseño tipo wizard [10, 11].
- Cada aplicación móvil debe tener un diseño totalmente responsivo. Esto quiere decir que se debe ajustar automáticamente a cualquier tamaño de pantalla. Se deben realizar pruebas de la aplicación en varios dispositivos de distintos tamaños tanto en modo vertical (portrait) como en modo horizontal (landscape). Si la aplicación funciona en solo uno de los modos, se debe verificar que no permita el giro, es decir, que se mantenga la posición correcta al girar el dispositivo [9].
- Los números de teléfono, direcciones de correo electrónico y enlaces a sitios web que aparezcan en la aplicación deben permitir abrir las aplicaciones respectivas al hacer tap sobre ellos [9]. Es altamente recomendable que la aplicación móvil detecte si el usuario tiene algún cliente de correo configurado antes de abrir la acción de envío de correo. De lo contrario, deberá informar la situación e indicar la no posibilidad de

realizar la operación, o bien, brindar la posibilidad de instalar y configurar una aplicación.

5.4.2 Navegación

Existen varios tipos de navegación comprobados para mejorar la usabilidad en aplicaciones móviles. No hay una regla única para utilizar un tipo sobre otro o alguna combinación, sin embargo esta sección detalla las ventajas y desventajas de cada estilo, así como las situaciones en las que se recomienda utilizar cada tipo.

5.4.2.1 Tabs

Este tipo de navegación consiste en una barra de botones estática, sin scroll, y que se mantiene durante toda la navegación. Es común que dichos botones contengan un ícono y un texto que describen claramente su acción como se muestra en la siguiente figura.



Ilustración 2. Captura de pantalla de aplicación con tabs.

Ventajas de la navegación con tabs

- Permite que el usuario pueda ver todas las posibles rutas todo el tiempo [12].
- Acceso directo y rápido [12].
- Es una buena forma de incluir notificaciones y que el usuario tenga buena visibilidad de ellas [12]. La siguiente figura presenta cómo la aplicación móvil de Facebook muestra las notificaciones en sus tabs.



Ilustración 3. Captura de pantalla de tabs de aplicación de Facebook.

Desventajas de la navegación con tabs

- Se debe hacer una buena y clara combinación entre íconos y textos [12].
- Reduce el tamaño usable de la pantalla [12].
- Se debe mantener en un máximo de entre 3 y 5 tabs [12].

¿Cuándo utilizar la navegación con tabs?

- Cuando las opciones de navegación son pocas (no más de 5 tabs)
- Cuando se requiere que el usuario esté cambiando de pantallas rápidamente.
- Para pantallas que son muy importantes y muy usadas dentro de la navegación.
- Para módulos que requieren informar al usuario directamente por medio de notificaciones.

5.4.2.2 Side Menu

Este tipo de navegación consiste en un botón lateral que normalmente se encuentra en alguno de los extremos de la barra de título, el cual al ser presionado muestra un menú con una lista de opciones. Este menú comúnmente se puede volver a cerrar al abrir alguna de las opciones, al presionar de nuevo el botón con el que se abrió el menú o al hacer swipe de la pantalla principal de la aplicación. La siguiente Ilustración muestra el side menu abierto de la aplicación de Gmail.

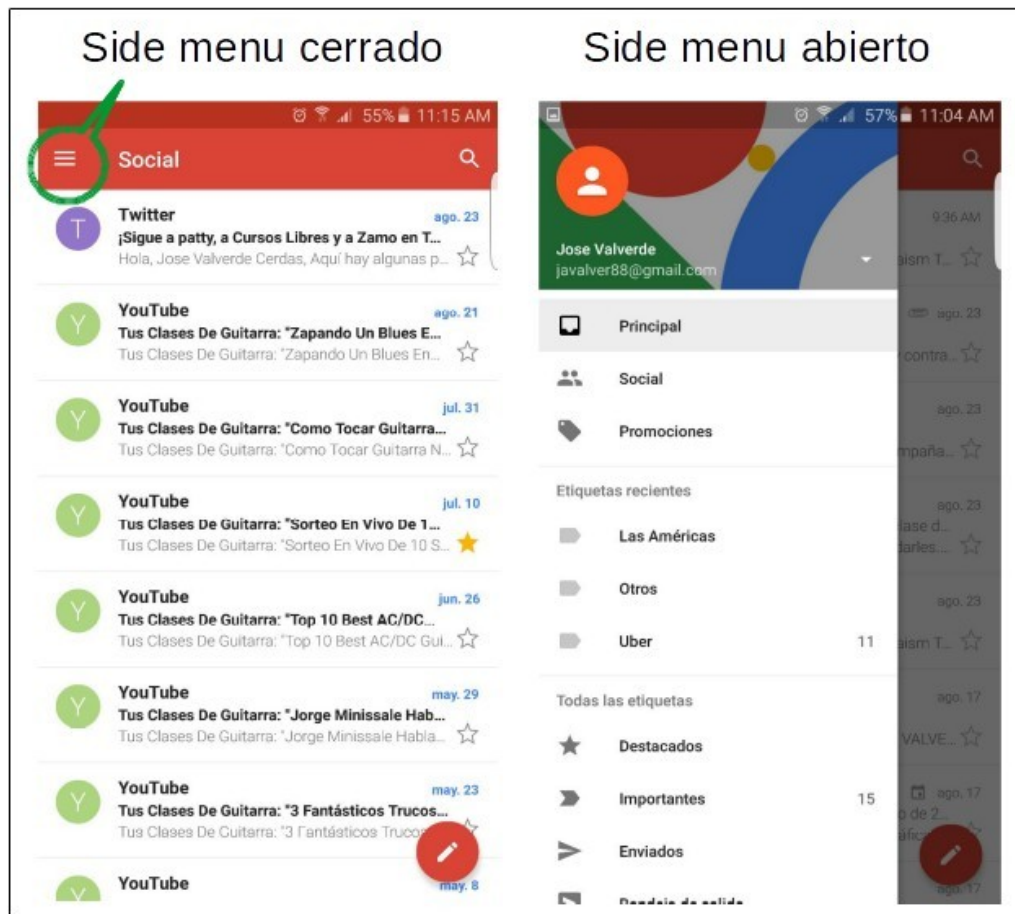


Ilustración 4. Captura de pantalla de side menu de aplicación de Gmail.

Ventajas de la navegación con side menu

- Funciona para muchas opciones de navegación [12].
- Brinda la opción de incluir menús anidados [12].
- Al estar cerrado brinda mucho más espacio en la pantalla para cada módulo [12].

Desventajas de la navegación con side menu

- Fuerza al usuario a hacer al menos un tap más para navegar a cada opción [12].

- No es visible hasta que el usuario lo abra [12].

¿Cuándo utilizar la navegación con side menu?

- Cuando las opciones de navegación son muchas (más de 3)
- Funciona muy bien para módulos que el usuario no debe utilizar continuamente [13].
- Cuando el espacio para contenido es importante.

5.4.2.3 Combinaciones

Este tipo de navegación consiste en una combinación de los dos anteriores. Una barra estática con botones para las acciones más importantes de la aplicación móvil y un menú que se puede abrir y cerrar para las acciones menos utilizadas. Es común en aplicaciones con muchas funcionalidades como Facebook, la cual se muestra en la siguiente ilustración.

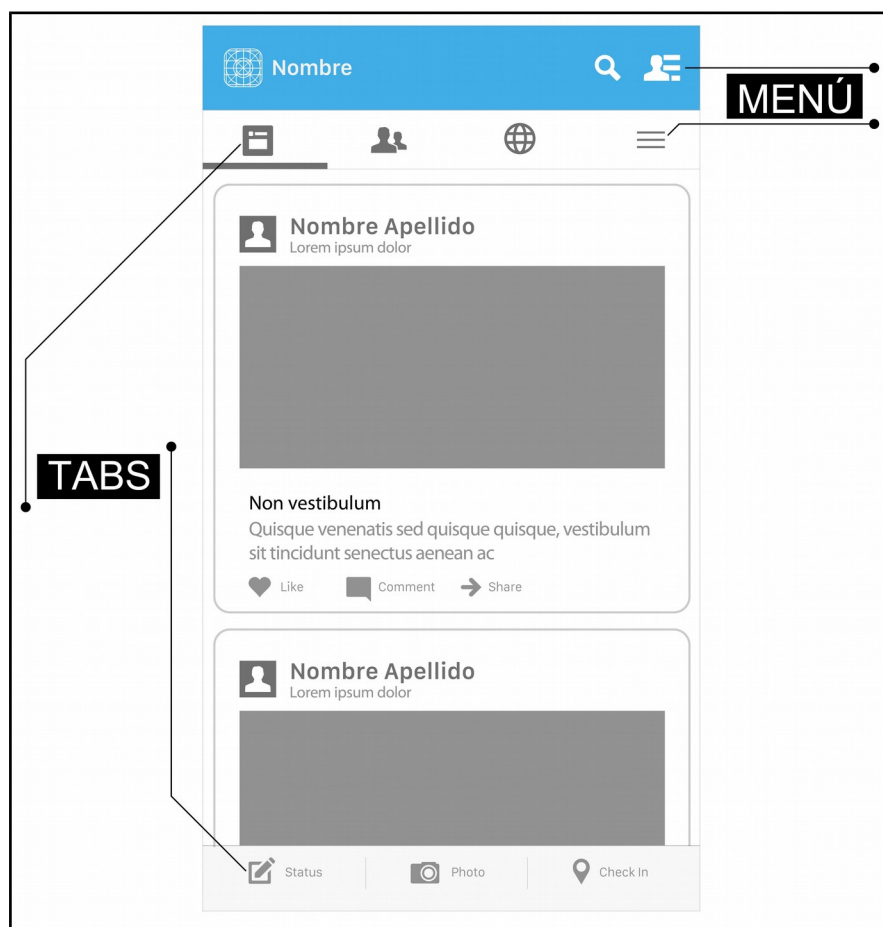


Ilustración 5. Captura de pantalla de aplicación de Facebook.

Si la aplicación es grande y tiene muchas opciones, una forma de mantener la navegación usable es combinar ambos tipos de navegación, utilizando los siguientes lineamientos:

- Utilizar barras de pestañas para las tres a cinco opciones más importantes y más utilizadas dentro de la aplicación.
- Utilizar la barra de pestañas para mostrar notificaciones.
- Utilizar el side menu para opciones menos utilizadas dentro de la aplicación como contacto, términos de uso, editar perfil, entre otras.

5.4.3 Compatibilidad

Es indispensable que las aplicaciones de la Universidad de Costa Rica sean totalmente compatibles con los sistemas operativos iOS y Android. Es deseable que sean compatibles incluso con otras plataformas como Windows.

Cada aplicación móvil debe ser compatible con versiones anteriores de cada sistema operativo. Debido al cambio acelerado de versiones de sistema operativo, para la publicación y actualización de cada aplicación se deben investigar las versiones de más uso en cada sistema operativo¹. En cualquier caso, el desarrollador debe documentar el listado de versiones de sistemas operativos con los que la aplicación es compatible.

Es responsabilidad de los desarrolladores comprobar el funcionamiento correcto de la aplicación en nuevas versiones de los sistemas operativos, según vayan surgiendo. En caso de que se requiera desarrollar una actualización, se debe notificar al Centro de Informática que dicha actualización está siendo desarrollada.

Si el Centro de Informática detecta que una aplicación móvil no funciona correctamente con una actualización de sistema operativo y no se recibe ninguna notificación de actualización de los desarrolladores, se procederá a retirar la aplicación de las tiendas de Play Store y App Store.

La notificación de actualización se debe enviar a moviles@ucr.ac.cr con el asunto "UCR @ Movil Notificación de actualización app <Nombre del app> para versión <Versión del sistema operativo> del SO <nombre del sistema operativo>".

5.5 Diseño gráfico

Como se indicó anteriormente, si bien es cierto las aplicaciones móviles son un medio un poco más informal, también se debe respetar la identidad visual de la Universidad de Costa Rica.

5.5.1 Ícono

- Los íconos deben ser imágenes simples que describan de una forma clara la funcionalidad principal de la aplicación [18].

¹ Para más información visitar los sitios oficiales de Android y iOS en los siguientes enlaces: <https://developer.android.com/about/dashboards/index.html> y <https://developer.apple.com/support/app-store/>

- Es altamente recomendable no incluir textos ni utilizar fotos en los íconos [14].
- Se debe crear el diseño del ícono a una escala de 1024x1024 pixeles ya que es el tamaño máximo requerido para iOS [15]. El tamaño máximo requerido en Android es de 512x512 pixeles [16], sin embargo es buena práctica diseñar el ícono de Android también a 1024x1024 pixeles y luego exportarlo al tamaño requerido.
- Todo elemento del ícono debe ser legible al reducirse a los tamaños necesarios para cada sistema operativo y dispositivo [14]. Se debe tener en cuenta que el ícono debe ser claro a tamaños de hasta 48x48 pixeles.
- Se debe mantener la consistencia entre el ícono y el diseño de la aplicación. Esto mejora altamente la experiencia del usuario [14, 15].
- Se debe probar el ícono con varios colores de fondo de pantalla del dispositivo [15, 16].
- Se debe mantener consistencia en los íconos de la aplicación en los diferentes sistemas operativos para que la aplicación sea fácilmente reconocible [14, 17].

5.5.1.1 Buenas prácticas y plantillas para iOS

En el siguiente enlace se puede encontrar una tabla actualizada de los tamaños de íconos (y otras imágenes) requeridos según las versiones de iOS para las que se desarrolle la aplicación móvil:

https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html#//apple_ref/doc/uid/TP40006556-CH27-SW1

Es importante tomar en cuenta que:

- iOS aplica una máscara a los íconos que hace que las orillas queden redondeadas [17].
- iOS no permite transparencias en los íconos [17].

La siguiente figura muestra como lucen los íconos en iOS.



Ilustración 6. Íconos en iOS.

En el siguiente enlace se puede encontrar una plantilla .psd que facilita el diseño del ícono de iOS: <http://appicontemplate.com/ios9/>

5.5.1.2 Buenas prácticas y plantillas para Android

En el siguiente enlace se encuentra la guía oficial de Android para el diseño de íconos: <https://www.google.com/design/spec/style/icons.html>

Es importante tomar en cuenta que Android permite cualquier tipo de formas y transparencias en sus íconos [17].

La siguiente figura muestra como lucen los íconos en Android.

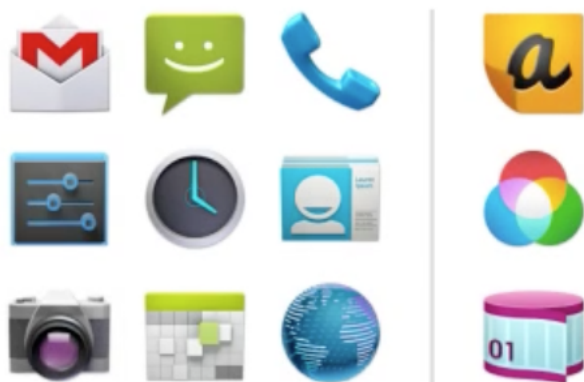


Ilustración 7. Íconos en Android.

En el siguiente enlace se puede encontrar una plantilla .psd que facilita el diseño del ícono de Android: <http://appicontemplate.com/android/>

5.5.2 Colores

Deben prevalecer los colores de la paleta oficial de la Universidad de Costa Rica la cual se encuentra disponible en el archivo digital que la Oficina de Divulgación e Información (ODI) facilita por medio de su sitio web odi.ucr.ac.cr.

5.5.3 Pantallas

- La pantalla que se muestra en el dispositivo cuando la aplicación está cargándose, debe presentar la firma promocional con texto en la parte inferior. El fondo debe ser, preferiblemente, “celeste UCR”; sin embargo, se permiten otros colores de la paleta [18].
- En el resto de pantallas de la aplicación, se sigue el estándar de la plataforma en la cual se está ejecutando. Se recomienda incorporar la firma promocional en la parte izquierda de la aplicación, ya sea en el área superior o inferior [18].

5.5.4 Menús y otros elementos

Los botones, gráficos, íconos, fondos u otros elementos que forman parte de las aplicaciones, deben respetar los lineamientos gráficos institucionales [18].

5.5.5 Efectos especiales

Las aplicaciones institucionales deben tener una apariencia limpia y hacer un uso mesurado de los efectos especiales (gradientes, sombras, embozados y texturas fotorrealistas). Se recomienda crear interfaces con diseños planos que enfatizen los contenidos [18].

5.5.6 Aspectos propios del sistema operativo

Cada sistema operativo tiene sus propias recomendaciones y frameworks de diseño para las aplicaciones móviles. Estas son utilizadas durante el desarrollo de la aplicación móvil al realizar el desarrollo en cada plataforma.

Android hace uso de Material Design. La información, recomendaciones y aspectos generales del Material Design pueden ser consultados en el siguiente enlace: <https://design.google.com/resources/>.

IOS, por su parte, brinda información de las recomendaciones de diseño de interfaces de usuario, a través del manual que se encuentra disponible ingresando en el siguiente enlace: <https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/>.

5.6 Formato y contenido de la documentación interna y externa

Es indispensable que cada aplicación móvil cuente con una completa documentación tanto interna como externa. Estos documentos deben facilitar la actualización continua y normal de las aplicaciones móviles.

5.6.1 Documentación externa mínima

La documentación externa es un proceso vital, principalmente en las etapas de desarrollo e implementación de una aplicación. Parte de esa documentación se debe levantar antes de empezar el desarrollo, mientras se realizan las etapas de análisis y diseño, mientras que otra parte se confecciona al final de la etapa de implementación. Según la metodología que se utilice, cada documento será actualizado en cada iteración. Se debe detallar cada actualización de la documentación externa en una tabla al inicio de cada documento con campos de fecha, descripción y responsable.

En esta sección se da una breve descripción de los documentos mínimos para tener una documentación externa suficiente para asegurar el apropiado mantenimiento de cada

aplicación móvil.

Es importante mencionar que todos los documentos presentados en esta sección deben ser actualizados en cada versión de la aplicación móvil y se debe detallar claramente las novedades de cada versión.

5.6.1.1 Documentos de análisis y diseño

- Documento de requerimientos: En este documento se detallarán los alcances y requerimientos del sistema. Debe incluir una lista no ambigua, completa y consistente de los requerimientos de dominio, funcionales y no funcionales. Se recomienda utilizar el formato de casos de uso e incluir diagramas para los requerimientos más complejos (diagramas de secuencia, actividades, colaboración, entre otros). Se puede encontrar el formato en **CI-AID-PL16-2016 Formato de Documentación de Requisitos**.
- Detalle del diseño de pantallas (mockups): En este documento y como regla general, se le ha puesto mucho énfasis a la accesibilidad, usabilidad y diseño gráfico de las aplicaciones de la Universidad de Costa Rica. Por eso es importante que en la etapa de diseño se incluya un documento de mockups que muestre el diseño y flujo de las pantallas en la aplicación móvil. La siguiente figura muestra un ejemplo de mockups para una aplicación móvil simple.

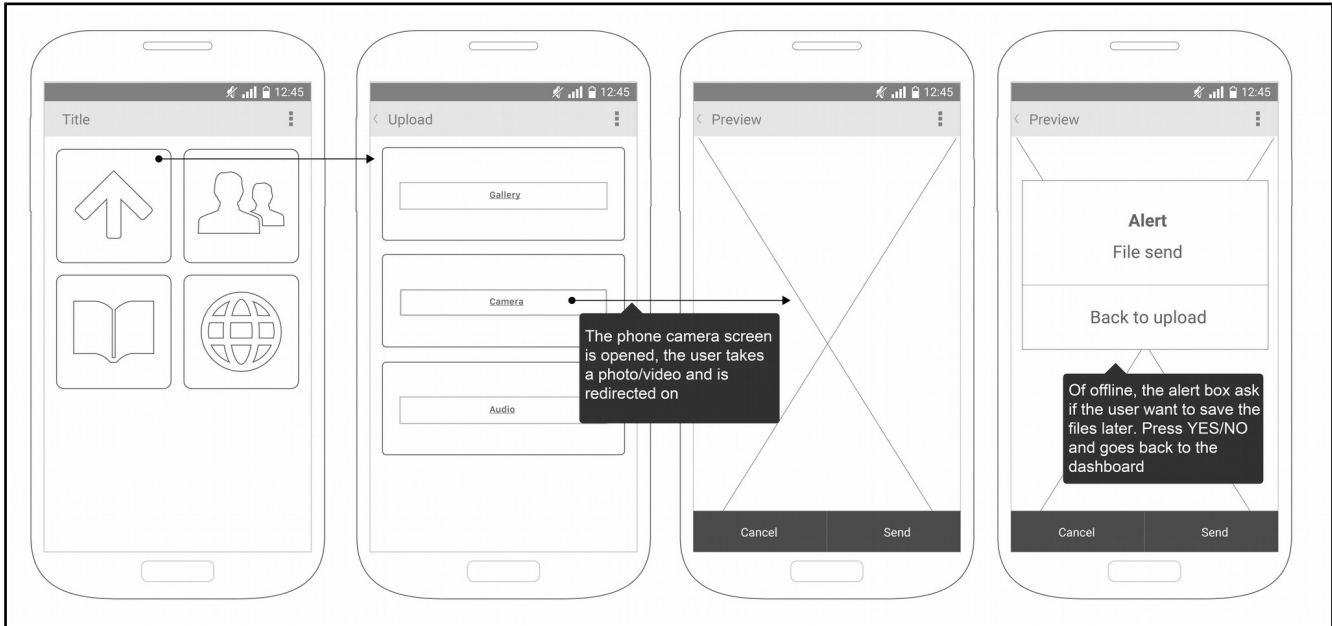


Ilustración 8a. Ejemplo de mockups de una aplicación móvil Android.

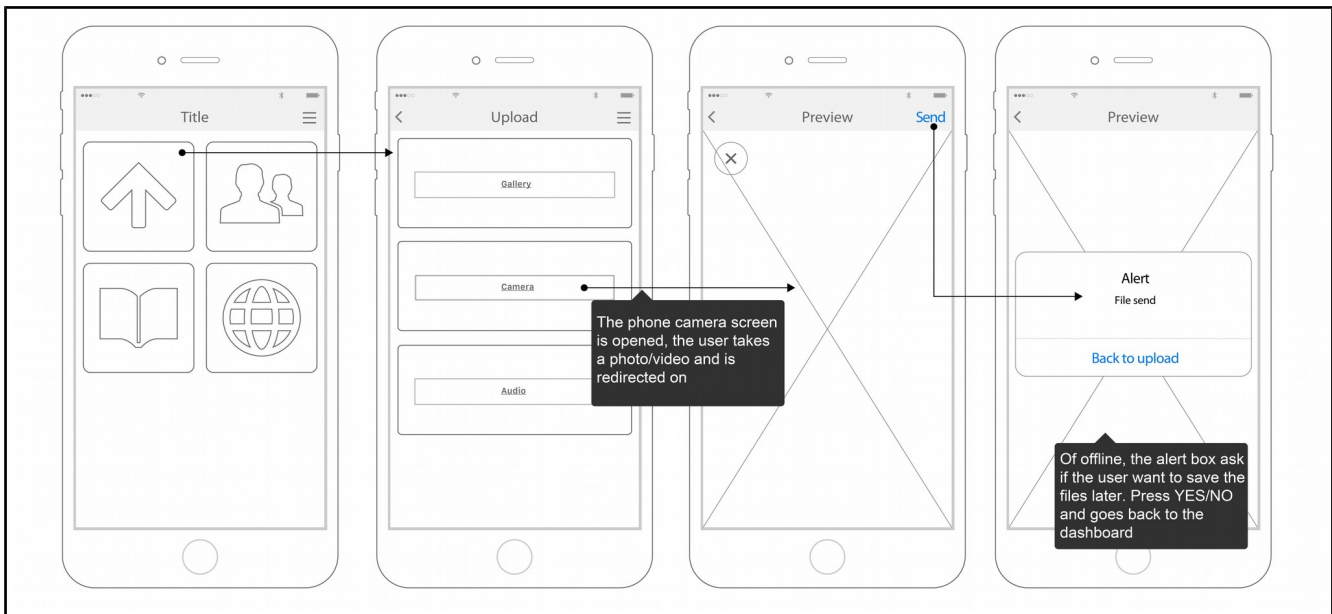


Ilustración 8b. Ejemplo de mockups de una aplicación móvil iOS.

- Diseño de casos de prueba: Es altamente recomendable usar una metodología de

desarrollo guiado por pruebas. En esta metodología las pruebas se detallan en la etapa de diseño. Estas guían la implementación pensando en que la aplicación responda satisfactoriamente a todos los casos de prueba que se detallan en la etapa de diseño, minimizando los errores al final del desarrollo. Al final de la(s) etapas de implementación se debe ejecutar todos los casos de prueba documentados. Se puede encontrar el formato de la documentación de los casos de prueba en [19] **CI-AID-PL14-2016 Formato de documentación de casos de prueba**. Además de documentar los casos de pruebas funcionales, se deben contemplar los siguientes aspectos fundamentales:

- Pruebas en distintos tipos de conexiones (WiFi y redes celulares) y velocidades de conexión disponibles.
- Pruebas sin conexión a red alguna, con el fin de analizar el comportamiento de la aplicación.
- Pruebas con distintos tamaños, resoluciones de pantalla (tabletas y teléfonos) y posición del dispositivo (horizontal o vertical). Además, pruebas de iconografía en distintos colores de fondo de pantalla.

5.6.1.2 Documentos de comunicación con aplicaciones externas

Documentación de servicios web utilizados y mensajes específicos: Este documento se debe iniciar durante las etapas de diseño y se debe actualizar durante todo el desarrollo.

- En la etapa de análisis y diseño se debe detallar en el documento todos los servicios externos que la aplicación va a utilizar. Esto incluye todos los repositorios de datos, aplicaciones externas y otros sistemas con los que la aplicación móvil debe interactuar. Esto con la idea de tener claridad y de poder tramitar todos los recursos pertinentes dentro de la Universidad de Costa Rica para los diferentes accesos que se necesiten.
- Durante la etapa de análisis y diseño y durante la implementación se debe agregar al documento el detalle y diseño de los servicios web que se utilicen. El formato de este documento dependerá de la tecnología que se utilice para el desarrollo del servicio web, sin embargo como regla general se debe incluir [20]:

- Descripción, URL, Método (POST, GET, otros), parámetros de cada solicitud y de cada respuesta (descripción detallada de cada parámetro no trivial) de cada mensaje.
- Código y detalle de cada posible respuesta de cada mensaje.
- Detalle de convenciones generales del servicio web.
- Códigos de status generales del servicio web.

5.6.1.3 Documentación final

- Documentación técnica: Este es el documento oficial para la documentación general de la plataforma que va a ser complementado con todos los otros documentos que se describen en esta sección de Documentación externa. Para este documento se debe utilizar el formato de documentación técnica del Centro de Informática de la Universidad de Costa Rica que incluye muchos de los puntos que ya se han detallado en otros documentos (se recomienda referenciar dichos documentos) así como detalles del modelo de datos y diseño de arquitectura, entre otros. Se puede encontrar el formato de la documentación técnica en **CI-AID-PL15-2016 Formato de Documentación Técnica de Software**.
- Informe de casos de prueba: Durante la etapa de pruebas se debe actualizar el documento de diseño de casos de prueba con todos los resultados de ejecución de los mismos.
- Manual de usuario: Se debe realizar un documento que detalle la interacción de los usuarios con la aplicación móvil. Como mínimo debe incluir:
 - Detalles de instalación y configuración.
 - Requisitos del dispositivo: versiones de sistema operativo soportadas, sensores (cámara, GPS, etc.) requeridos.
 - Conocimientos mínimos del usuario.
 - Funciones básicas del sistema: Detalle paso a paso de las funciones con capturas de pantalla.

- Detalle de posibles problemas o errores.
- Detalle de contacto en caso de necesitar más ayuda.

5.6.2 Documentación interna

Como se ha explicado previamente en esta sección, debido a la continua actualización que requieren las aplicaciones móviles, es muy importante que el código fuente sea fácil de mantener y de extender. Esto hace que sea de mucha importancia que el código fuente tenga una buena documentación interna. Se deben seguir los estándares normales de ingeniería de software que incluyen [21]:

- Encabezado en cada archivo con el título y descripción, datos del autor, versión del archivo, cambios en la nueva versión (principalmente en métodos).
- Encabezado de cada función, con su nombre, detalle de parámetros, precondiciones y postcondiciones.
- Los nombres de variables deben ser representativos del significado de la variable.
- Se debe utilizar notación lowerCamelCase (Esto según el lenguaje utilizado). El lowerCamelCase consiste en iniciar el nombre de la variable o método con letra minúscula y cada cambio de palabra se empieza con una letra mayúscula por ejemplo: setLandscapeMode().
- Los nombres de los métodos deben ser representativos de la funcionalidad del método.
- Los métodos que retornan valores booleanos deben ser nombrados en función a la condición de cuando se devuelve el valor verdadero (true) por ejemplo: isFull().
- Se debe comentar claramente cualquier porción compleja de código.
- Se debe utilizar una indentación consistente.
- Los nombres de variables y funciones pueden estar escritos en inglés o en español, sin embargo, debe existir uniformidad, es decir, todo el código fuente debe estar escrito en un único idioma.

5.7 Arquitectura

5.7.1 Tipos de aplicaciones móviles, según su desarrollo

Actualmente las herramientas y frameworks disponibles para el desarrollo de aplicaciones móviles dividen a las aplicaciones en dos tipos nativas e híbridas. En las siguientes secciones se explica cada una de los tipos y se dan las recomendaciones sobre que tipo utilizar en cada caso.



Ilustración 9. Diagrama de interacción de aplicaciones nativas e híbridas con los APIs del dispositivo.

5.7.2 Aplicaciones móviles nativas

Las aplicaciones nativas son específicas a un solo sistema operativo y se desarrollan haciendo uso del lenguaje y herramientas específicas de dicho sistema. Pueden utilizar el potencial completo de cada dispositivo tan pronto como las plataformas desarrollen y publiquen las bibliotecas necesarias. Son más costosas y llevan mucho más tiempo de desarrollo debido que se debe programar una aplicación completamente nueva para cada sistema operativo [22]. Ejemplos de aplicaciones móviles nativas incluyen Angry Birds y Shazam.

5.7.3 Aplicaciones móviles híbridas

Las aplicaciones híbridas son básicamente aplicaciones web envueltas en un contenedor nativo que les brinda acceso a las funcionalidades nativas del dispositivo. La principal ventaja

es la rapidez y consistencia del desarrollo, ya que de un mismo código fuente se pueden generar ejecutables para múltiples plataformas. Las principales desventajas son que las bibliotecas para acceder a nuevas funcionalidades nativas que se vayan incluyendo en nuevos modelos de teléfonos duran un poco más en ser desarrolladas (o deben ser accesadas con código nativo y se debe modificar directamente el código del contenedor nativo) y que utilizan Canvas y WebGL para los gráficos en vez de las APIs nativas [22]. Ejemplos de aplicaciones móviles híbridas incluyen Khan Academy y Sworkit.

5.7.4 ¿Qué tipo utilizar?



Ilustración 10. Capacidades de los tipos de desarrollo de aplicaciones móviles [22].

La siguiente tabla muestra las diferencias y similitudes entre las capacidades de las aplicaciones móviles nativas y las híbridas.

	Nativas	Híbridas
Funcionalidades		
Gráficos	APIs Nativos	HTML, Canvas, SVG, WebGL
Rendimiento	Alto	Normal
Look and Feel Nativo	Nativo	Emulado
Distribución	App Store / Play Store	App Store / Play Store
Acceso al dispositivo		
Cámara	Sí	Sí
Notificaciones	Sí	Sí
Contactos, Calendario	Sí	Sí
Datos locales	Sistema de archivos nativo, SQL	Sistema de archivos nativo, SQL, HTML5 Local Storage
Geo localización	Sí	Sí
Gestures		
Swipe	Sí	Sí
Pinch, spread	Sí	Sí
Lenguajes	Objective-C, Swift, Java	HTML5, CSS, Javascript

Tabla 1. Diferencias y similitudes entre las capacidades de las aplicaciones nativas y las híbridas [22].

Para decidir que tipo de aplicación desarrollar se puede realizar las siguientes preguntas [22]:

- ¿Qué tanto se utilizarán las funcionalidades nativas del dispositivo?

Tanto las aplicaciones nativas como las híbridas pueden utilizar perfectamente las funcionalidades nativas del dispositivo como la cámara y el GPS. Sin embargo, si la funcionalidad principal de la aplicación móvil requiere utilizar alguna de las funcionalidades nativas del dispositivo de manera muy continua y muy pesada, es altamente recomendable desarrollar una aplicación nativa. Esto es especialmente importante en aplicaciones que hacen uso intenso de las funcionalidades gráficas nativas del dispositivo, por ejemplo juegos en 3D.

- ¿Qué tan pronto se quiere publicar la aplicación? ¿Se tiene presupuesto separado para una equipo de desarrollo para cada sistema operativo?

Estas dos preguntas van de la mano, ya que si el presupuesto es limitado y se necesita ingresar rápido al mercado la mejor opción es una aplicación híbrida. Si no se tienen restricciones limitadas de tiempo y presupuesto (incluso pensando en el mantenimiento) lo mejor es realizar las diferentes aplicaciones nativas.

- ¿Qué tan a menudo se debe actualizar la aplicación?

Actualizar una aplicación nativa es un trabajo complejo ya que se deben actualizar los códigos fuente (totalmente distintos) de cada plataforma. Con las aplicaciones híbridas es mucho más sencillo ya que es el mismo código fuente para todas las plataformas y los lenguajes de programación son muy conocidos. Incluso nuevas tecnologías en aplicaciones híbridas permiten actualizaciones de la aplicación móvil sin tener que volver a hacer una publicación en las tiendas App Store y Play Store.

5.7.5 Arquitectura de Modelo Vista Controlador

Las bibliotecas principales que se utilizan para cualquier tipo de desarrollo de aplicación móvil, ya sea híbrida o nativa, para iOS o Android en la actualidad están definidas para utilizar la arquitectura de Modelo Vista Controlador [23, 24]. En esta sección se describe el patrón de Modelo Vista Controlador. En una siguiente sección de Lenguajes y Convenciones se detallará el mapeo de clases y componentes de cada plataforma y tipo de desarrollo con este patrón.

En el patrón de Modelo Vista Controlador, se le asigna a cada objeto de la aplicación uno de estos tres roles. El patrón no solo define los roles si no que también la comunicación entre estos. Entre las ventajas de este patrón se encuentran la reusabilidad de los objetos, la definición clara de las interfaces y la extensibilidad y fácil mantenimiento de las aplicaciones [23]. La siguiente ilustración muestra la comunicación típica de los elementos del patrón de Modelo Vista Controlador.

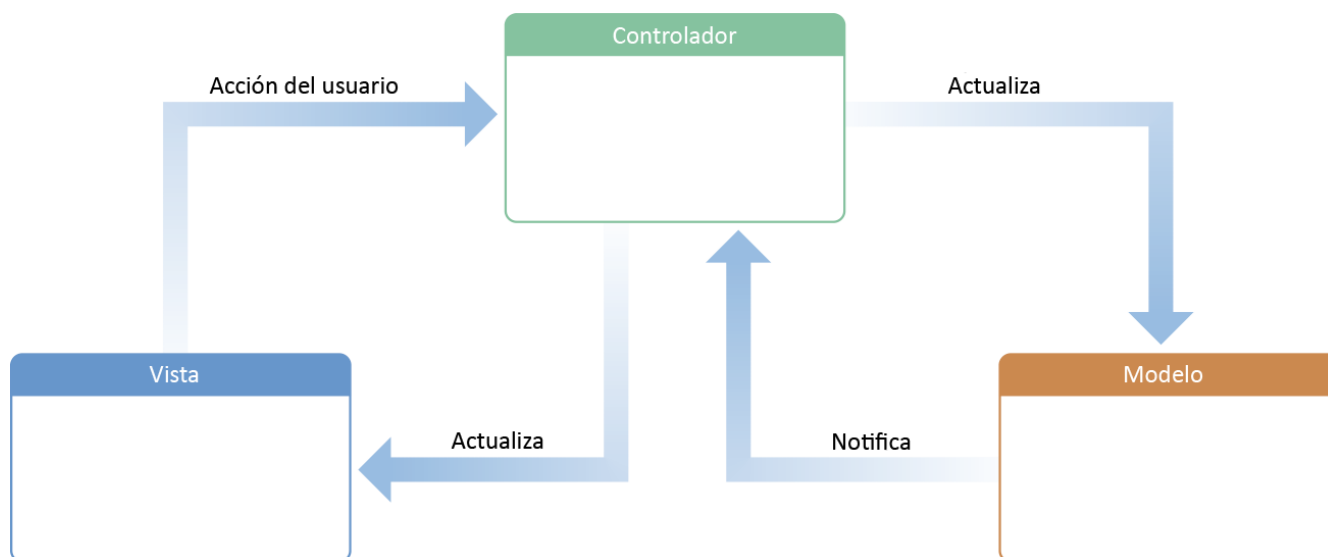


Ilustración 11. Diagrama del patrón Modelo Vista Controlador [23].

5.7.5.1 Modelo

Los objetos con rol de Modelo encapsulan datos específicos de la aplicación y la lógica para procesar dichos datos. Normalmente mapean datos que se encuentran en algún almacenamiento persistente [23].

Estos objetos pueden tener relaciones de uno a uno, uno a muchos o muchos a muchos con otros objetos de tipo Modelo, así que la capa Modelo de la aplicación consiste en una red de varios objetos [23].

Debido a que estos objetos representan datos específicos de un dominio, los mismos pueden ser reutilizados en distintos ámbitos de la aplicación. Idealmente estos objetos no deben tener comunicación directa ni saber nada de las vistas y deben ser modificados y manipulados por medio de los controladores [23].

5.7.5.2 Vista

Los objetos con rol Vista son los que el usuario puede ver y manipular. El objeto de tipo Vista debe saber como mostrarse en pantalla y permitir la interacción del usuario. Una de las funcionalidades más importantes de las Vistas es permitir al usuario manipular los objetos de tipo Modelo, sin embargo, los objetos de tipo Modelo y los de tipo Vista deben estar

totalmente no acoplados, las Vistas solo deben interactuar con los Controladores [23].

5.7.5.3 Controlador

Los objetos con rol Controlador actúan de intermediarios entre una Vista y uno o más Modelos. Estos forman un conducto por medio el cual las Vistas conocen los cambios en los Modelos y los Modelos son modificados por medio de las interacciones del usuario con las Vistas. Los Controladores también pueden realizar otras tareas como el manejo del ciclo de vida de otros objetos o coordinación de tareas generales de la aplicación [23].

5.7.5.4 Comunicación

El flujo de comunicación en una arquitectura de Modelo Vista Controlador es el siguiente: la Vista recibe una interacción del usuario que llama a alguna funcionalidad del Controlador. El Controlador pide a los Modelos de datos realizar los cambios que el usuario requiere. El Modelo notifica al Controlador de los cambios que se han realizado. El Controlador actualiza la Vista con los cambios correspondientes [23].

5.7.6 Comunicación con aspectos propios del dispositivo

Varios de los aspectos propios del dispositivo son sensores que generan eventos los cuales deben ser procesados en el loop principal de la aplicación. Es importante conocer la naturaleza asíncrona de estos eventos al momento de implementar funcionalidades que los utilicen [23, 24]. En la siguiente ilustración se muestra como estas funcionalidades, en este ejemplo un swipe en la pantalla táctil, interactúan de manera asíncrona con el loop principal de la aplicación. El swipe se registra en la pantalla la cual envía un mensaje al sistema operativo que lo encola en una cola de eventos. El loop principal de la aplicación, entre sus tareas, revisa cada cierto tiempo la cola de eventos y los notifica al objeto principal de la aplicación, que a su vez lo envía a los objetos de la aplicación que estén escuchando asíncronamente el evento de swipe.

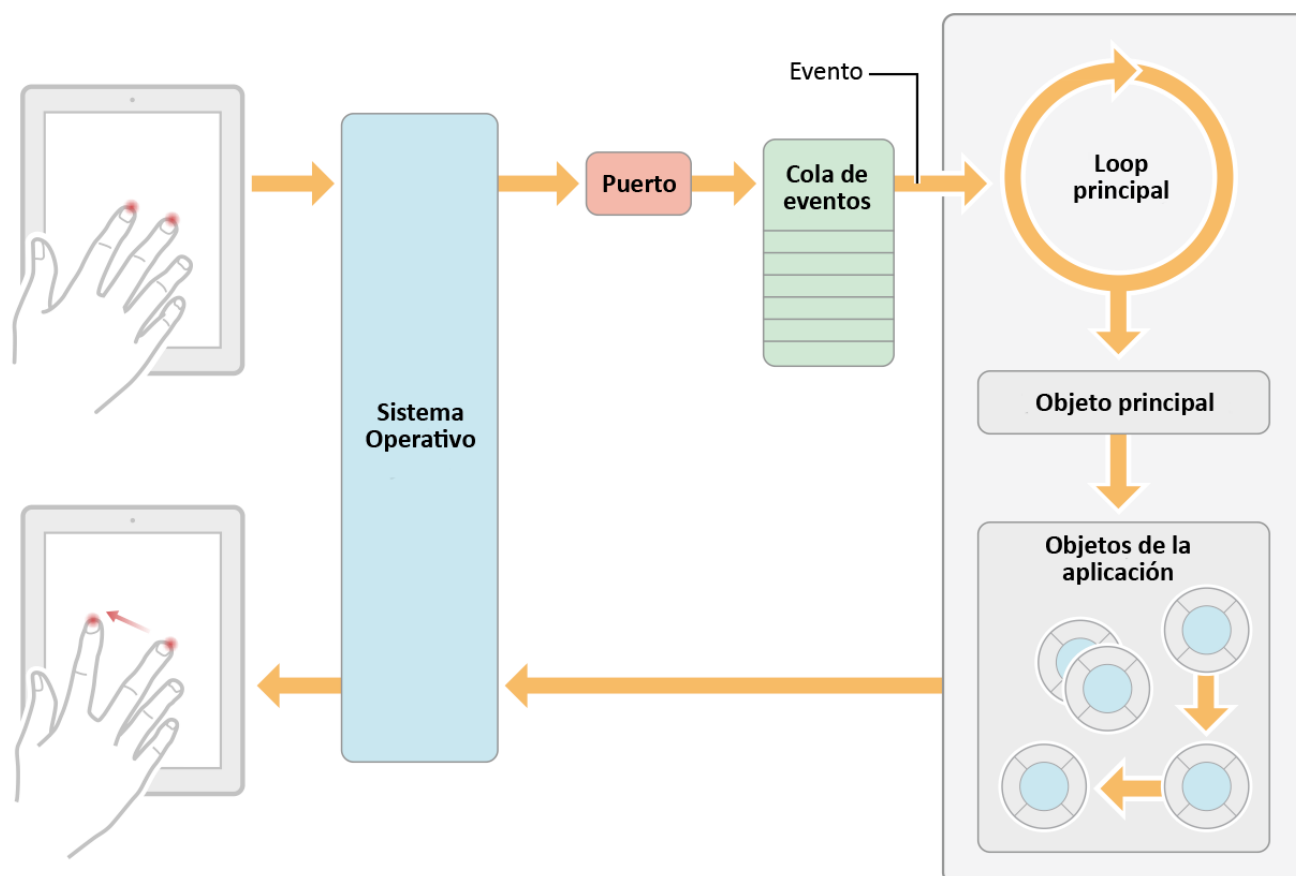


Ilustración 12. Diagrama de comunicación de la aplicación de iOS con el sistema operativo [23].

El tipo de sensores que utilizan este tipo de arquitectura asincrónica incluyen: Geolocalización, Touch, Acelerómetro, Giroscopio, Control Remoto, Sacudida, Magnetómetro, Conectividad (WiFi o Móvil) [23, 24].

Otros aspectos del dispositivo son sincrónicos, por ejemplo la cámara que funciona como un input de archivos [23, 24].

Al momento de implementar funcionalidades que utilicen alguno de estos sensores, en aplicaciones nativas se debe consultar la documentación respectiva de las bibliotecas más actualizadas de cada plataforma. En aplicaciones híbridas se debe consultar los plugins de cada sensor. Es importante actualizarlas a menudo como se detalló en la sección Seguridad.

Los detalles del uso de estas funcionalidades deben ser registrados en la documentación técnica, en la sección de “Detalles Técnicos de la Implementación”.

5.8 Lenguajes, APIs y convenciones de programación

En esta sección se enumeran los lenguajes de programación de cada plataforma y tipo de desarrollo. También se mencionan las convenciones de clases o componentes utilizados para implementar la arquitectura Modelo Vista Controlador en cada caso.

Esta sección se actualizará continuamente según surjan nuevos lenguajes de programación, frameworks y tecnologías para el desarrollo de aplicaciones móviles.

5.8.1 IOS

5.8.1.1 Objective-C

El lenguaje Objective-C fue creado en los 80s y se popularizó cuando la compañía NeXT empezó a utilizarlo en sus productos. Cuando Apple compró NeXT incluyó sus productos de desarrollo y se basó en ellos para la creación de XCode (el IDE para desarrollar aplicaciones Apple) y las Cocoa API (las bibliotecas de interfaces con los sistemas operativos Apple).

Objective-C es una capa sobre C, por ende cualquier compilador de Objective-C puede compilar programas C. Toda la sintaxis de operaciones no orientadas a objetos es idéntica a la de C. Por otro lado, la sintaxis de las operaciones orientadas a objetos es derivada de SmallTalk [25].

Debido a la derivación de Smalltalk, los objetos se comunican por medio de envío de mensajes en lugar de llamadas a métodos. De esta forma los mensajes se resuelven en tiempo de ejecución y son interpretados por los objetos, en oposición a los llamados a métodos que típicamente se resuelven en tiempo de compilación.

El siguiente es un programa de “Hola Mundo” en Objective-C

```
#import <Foundation/Foundation.h>
```



```
int main (int argc, const char * argv[])
```



```
{
```



```

    NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
    NSLog(@"Hola Mundo!");
    [pool drain];
    return 0;
}

```

Para conocer más sobre Objective-C y las especificidades de su sintaxis se recomienda analizar la siguiente guía de Apple, la cual se encuentra disponible en el enlace <https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>

5.8.1.2 Swift

Swift es un lenguaje desarrollado por Apple que apareció por primera vez en el 2014. El objetivo de Swift es ser una forma de programar menos propensa a errores y más concisa que Objective-C. Swift mantiene todos los conceptos centrales de Objective-C, pero introduce conceptos de los lenguajes más modernos para facilitar la sintaxis [26].

El siguiente es un programa de “Hola Mundo” en Swift:

```
println("Hola Mundo")
```

Para conocer más sobre Objective-C y las especificidades de su sintaxis se recomienda analizar la siguiente guía de Apple, la cual se encuentra disponible en el enlace https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/index.html

5.8.1.3 Conceptos clave del API

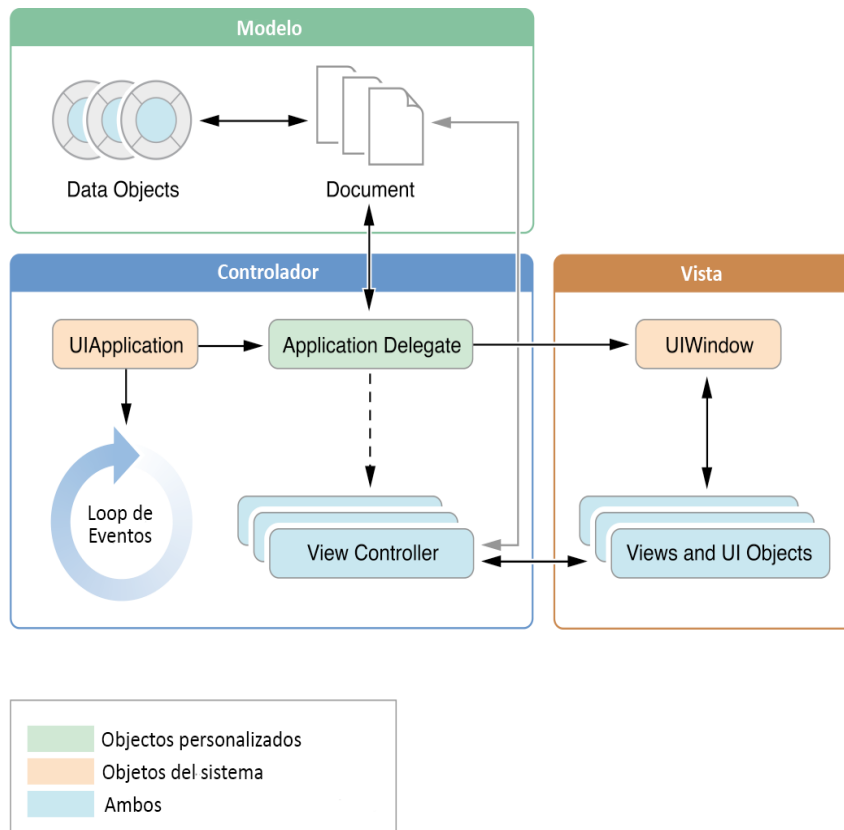


Ilustración 13. Implementación de patrón Modelo Vista Controlador en aplicaciones nativas para iOS [23].

Ya sea que las aplicaciones de iOS se programen en Objective-C o en Swift, ambos utilizan las bibliotecas Cocoa. A continuación se describe brevemente cada uno de los componentes más comunes de dicha biblioteca, al desarrollar la aplicación móvil utilizando el patrón de Modelo Vista Controlador cuya comunicación se gráfica en la ilustración anterior [23].

- **UIApplication:** Este objeto maneja el loop principal de la aplicación y otras tareas de alto nivel. Reporta las transiciones principales y eventos especiales al Application Delegate.
- **Application Delegate:** Este es el objeto principal de la aplicación desde el punto de

vista del desarrollador. Es donde el código especificado por el programador se empieza a ejecutar. Es el único objeto que debe estar en toda aplicación móvil de iOS, por lo tanto normalmente es donde se inicializan los datos principales de la aplicación.

- **Data Objects y Documents:** Estos son los objetos de rol Modelo.
- **View Controllers:** Estos son los objetos de rol Controlador. `UIViewController` es la clase base para todos los controladores en una aplicación de iOS. Esta clase contiene todas las funcionalidades para cargar, presentar y rotar vistas, además de otras funcionalidades específicas. Además, la biblioteca `UIKit` ofrece clases derivadas del `UIViewController` que manejan vistas con funcionalidades más específicas como seleccionadores de imágenes, tabs y otras formas de navegación.
- **UIWindow:** El `UIWindow` es la ventana principal sobre la que se presentan las diferentes vistas de la aplicación móvil.
- **Views and UI Objects:** Estos son los objetos de rol Vista. `UIView` es la clase base para todas las vistas en una aplicación de iOS. `UIKit` ofrece clases derivadas del `UIView` para hacer más simple el uso de vistas estándar.

Para conocer más sobre la programación de aplicaciones móviles nativas en iOS se recomienda analizar la siguiente guía de Apple, la cual se encuentra disponible en el enlace https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html#//apple_ref/doc/uid/TP40007072-CH2-SW2

5.8.2 Android

5.8.2.1 Java

Java es uno de los lenguajes de programación más populares de la actualidad. Fue creado por James Gosling y Sun Microsystems y publicado en 1995.

A pesar de que el kernel de Android es un Linux, y por lo tanto la mayoría de su código fuente es C, Google decidió que el Android SDK (las bibliotecas para desarrollar aplicaciones móviles) utilicen Java. Sin embargo, a diferencia de la gran mayoría de sistemas que ejecutan código en Java, actualmente Android no utiliza una máquina virtual de Java, sino que compila las aplicaciones para producir código máquina cuando la aplicación es instalada

en el dispositivo [27].

El siguiente es un programa de “Hola Mundo” en Java:

```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hola mundo");  
    }  
}
```

5.8.2.2 Kotlin

A partir de Android Studio 3.0 se incluyen por defecto las herramientas de programación en Kotlin y este lenguaje se convierte en uno de los lenguajes oficiales para la programación de aplicaciones nativas de Android.

Kotlin es un lenguaje de programación desarrollado por JetBrains y es interoperable con Java y Javascript. Debido a esta funcionalidad y por mejoras en sintaxis, seguridad y herramientas con respecto a Java, Kotlin se ha convertido en una elección popular entre los desarrolladores de aplicaciones Android.

El siguiente es un programa de “Hola Mundo” en Kotlin:

```
fun main(args : Array<String>) {  
    val scope = "world"  
    println("Hello, $scope!")  
}
```

5.8.2.3 Conceptos clave del API

A continuación se describe brevemente cada uno de los componentes más importantes del Android SDK. A pesar de que estos componentes no siguen roles de Modelo Vista Controlador tan estrictos como otras plataformas, el estándar es programarlos y utilizarlos acoplándose a dicho patrón [24].

- **Activities:** Los Activities son objetos de rol Controlador para una Vista (las Vistas son típicamente representadas en interfaces XML), y representan una pantalla con una interfaz de usuario. Cada Activity es independiente de la otra y una aplicación móvil puede iniciar en cualquiera de ellas.
- **Services:** Los Services no están acoplados a ninguna Vista, estos se utilizan para realizar operaciones de fondo en la aplicación, por ejemplo llamados asíncronos a una base de datos remota.
- **Content Providers:** Estos son objetos de rol Modelo. Esta clase, Content Provider, contiene métodos estándares que deben ser implementados para la actualización y consulta de datos.
- **Broadcast Receivers:** Estos objetos son componentes que reciben mensajes del sistema operativo o de otras aplicaciones. Pueden escuchar a eventos como batería baja, toma de fotos, descargas finalizadas, entre otros.

Para conocer más sobre la programación de aplicaciones móviles nativas en Android se recomienda analizar la siguiente guía de Google, la cual se encuentra disponible en el enlace <http://developer.android.com/guide/components/fundamentals.html>

5.8.3 Aplicaciones híbridas

5.8.3.1 HTML, CSS, Javascript

Las aplicaciones híbridas, como se mencionó en la sección de Arquitectura, se programan en su mayoría con las mismas tecnologías que se programa un sitio web, es decir, HTML, CSS y Javascript.

Estas tecnologías, al ser tan utilizadas, aparecen en diversas formas, herramientas y bibliotecas y cualquier combinación de estas puede ser utilizada para el desarrollo de la capa Web de una aplicación híbrida. Hay diversos frameworks que utilizan diferentes combinaciones y brinda porciones reutilizables de código que facilitan el desarrollo según la preferencia del programador. En la capa de Javascript, la cual es la que lleva el peso de la programación de los controladores y los servicios, por ejemplo, es popular utilizar AngularJS, React, o JQuery.

5.8.3.2 Cordova

Cordova es el contenedor nativo más utilizado para las aplicaciones híbridas. Como contenedor nativo se entiende que Cordova encapsula el código HTML5, CSS y Javascript que representan la presentación y lógica de la aplicación, con una capa de código nativa que hace que la aplicación se ejecute nativamente en los diversos sistemas operativos y pueda interactuar con los sensores y aspectos nativos de los mismos.

Al proyecto de Cordova contribuyen Adobe, IBM, Google, Microsoft, Intel, Blackberry, Mozilla entre otros [28]. Cordova puede generar aplicaciones móviles para los sistemas operativos móviles más populares incluyendo: iOS, Android, Windows Phone, BlackBerry, Tizen, Ubuntu Touch, Firefox OS y Amazon-FireOS [29].

En su núcleo, Cordova contiene APIs para utilizar las siguientes funcionalidades nativas del dispositivo: acelerómetro, estado de la batería, cámara, micrófono, brújula, estado de conexiones WiFi o móviles, contactos, información básica del dispositivo (modelo, nombre del sistema operativo, id, etc.), eventos, sistema de archivos, geo-localización, globalización (lenguaje, huso horario, etc.), browser interno, multimedia, notificaciones, splashscreen, barra de estatus, almacenamiento y vibración. De todas maneras existe una gran colección de bibliotecas creadas por terceros para hacer uso de otras APIs específicas de los dispositivos [29].

Al desarrollar una aplicación híbrida de la Universidad de Costa Rica es altamente recomendado utilizar Cordova con contenedor nativo.

5.8.3.3 Frameworks

En la actualidad existe una gran variedad de frameworks muy robustos que utilizan Cordova. Estos frameworks difieren en las bibliotecas que brindan (especialmente para emular la navegación nativa).

La siguiente tabla muestra algunos de los más populares:

Nombre	Implementación de Javascript	Enlace
Ionic	AngularJS o Javascript puro	http://ionicframework.com/
Onsen UI	AngularJS o Javascript puro	https://onsen.io/
Framework 7	Javascript puro	http://www.idangero.us/framework7
React Native	React	http://www.reactnative.com/
jQuery Mobile	jQuery	http://jquerymobile.com/
Native Script	Javascript puro	https://www.nativescript.org/
Famous	WebGL, AngularJS	http://famous.org/

Tabla 2. Frameworks populares para el desarrollo de aplicaciones híbridas [30].

5.9 Estructuras e integración de datos

5.9.1 Recomendaciones para datos locales

Cada plataforma brinda varios servicios que se pueden utilizar para guardar datos locales. Si bien, lo más importante por tener en cuenta a la hora de guardar datos locales es la seguridad de los mismos, en esta sección se enumeran algunas posibilidades y recomendaciones.

- La mayoría de los sistemas operativos móviles, como Android, iOS y Windows Phone, soportan SQLite, la cual es una de las formas más recomendadas de guardar datos locales. SQLite es suficientemente rápida, segura, utiliza poca memoria, tiene alta compatibilidad y es fácil de utilizar.
- Otra opción de manejo de datos locales en iOS es Core Data, más información de cómo utilizarla puede ser vista en el siguiente link: https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/CoreData/index.html#//apple_ref/doc/uid/TP40001075-CH2-SW1
- Android tiene otras opciones de manejo de datos locales como Shared Preferences,

Internal Storage y External Storage que pueden ser utilizadas según el caso. Más información de cómo utilizarlas puede ser vista en el siguiente link: https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/CoreData/index.html#//apple_ref/doc/uid/TP40001075-CH2-SW1

- Como regla principal se debe recordar que por disposiciones de seguridad es importante no guardar datos críticos localmente en el dispositivo y verificar que el método de almacenamiento solo pueda ser accesado por la aplicación móvil que lo necesita.

5.9.2 Recomendaciones para datos remotos

Los datos remotos serán consumidos por la aplicación móvil por medio de servicios web. Esta sección brinda algunas recomendaciones útiles por tomar en cuenta en el diseño de los servicios web.

- Lo más importante es que el diseño de los servicios web sea eficiente y minimice el número de llamados necesarios.
- Los datos se pueden enviar tanto en formato XML como JSON. Para aplicaciones híbridas es recomendado que los datos estén en formato JSON, ya que en JavaScript se facilita mucho la manipulación de datos en este formato.
- Los llamados deben proporcionar información de cada caso de error y la aplicación móvil debe reaccionar acorde al estado de los llamados. Por ende, se debe informar al usuario si los datos se están cargando, si no hay datos disponibles o si hubo un error con la transferencia de datos.

5.9.3 Requisitos específicos de la Institución y aspectos legales

- Toda aplicación que utilice datos de estudiantes o funcionarios de la Universidad de Costa Rica debe permitir la autenticación de usuarios utilizando el Servicio de Directorio Institucional. Posteriormente debe leer los datos que requiera del usuario desde ese servicio.
- Cada aplicación puede luego tener su propia base de datos donde relacione los usuarios con los datos específicos de la aplicación.

- La aplicación también debe estar en comunicación con el Servicio de Directorio Institucional para verificar si algún usuario ha sido inactivado o modificado y realizar los cambios correspondientes en su propia base de datos.
- Es deber y responsabilidad del desarrollador utilizar estas plataformas para lograr la integración y mantener la integridad de los datos de los usuarios de la Universidad de Costa Rica en su aplicación.

La siguiente ilustración muestra la comunicación entre las plataformas y aplicaciones que permite mantener la integración de datos.

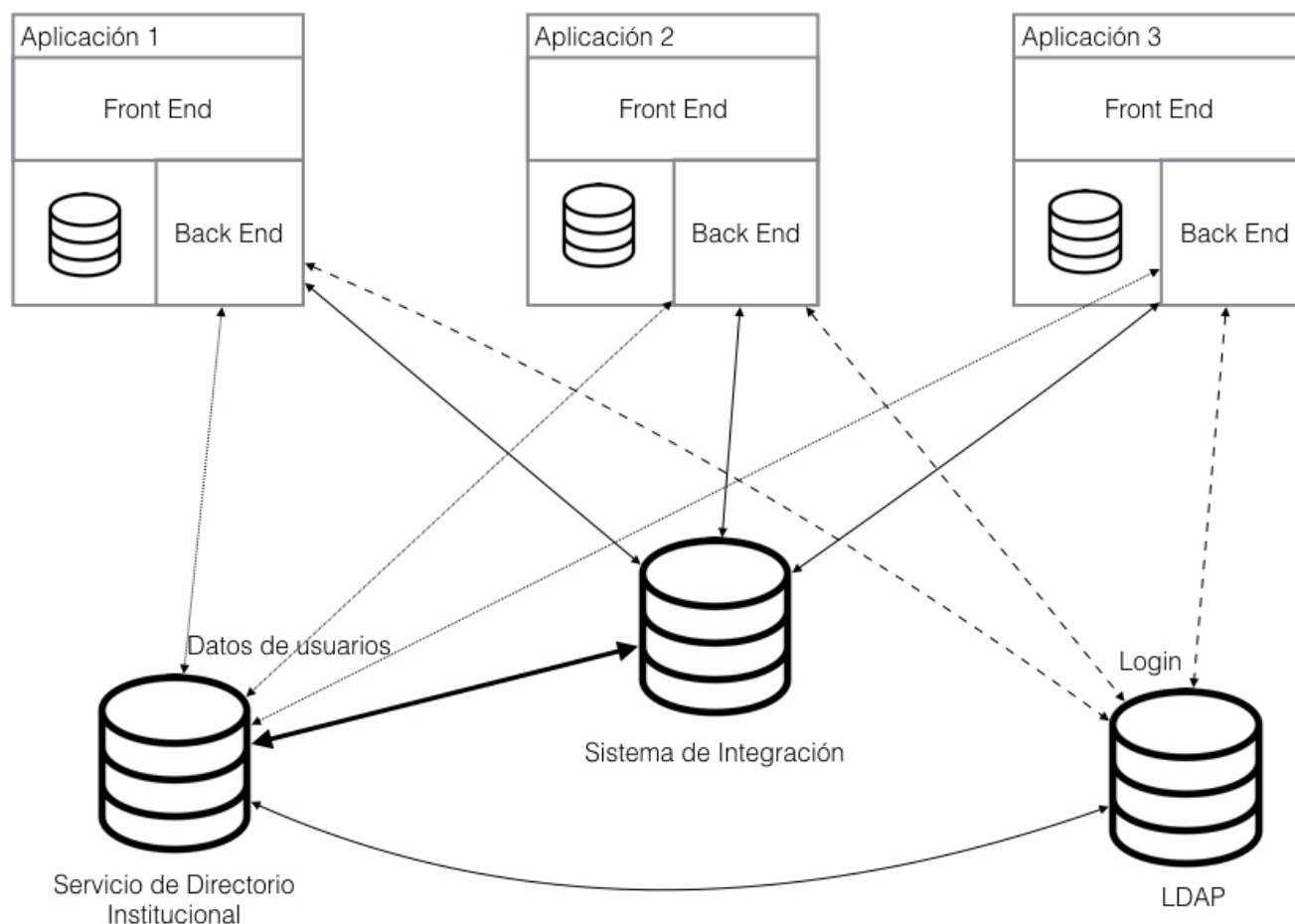


Ilustración 14. Proceso de integración de datos de usuarios de la Universidad de Costa Rica

La legalidad y uso de los datos es responsabilidad de la Unidad desarrolladora de la aplicación en la Universidad de Costa Rica

5.10 Publicación

El registro de nuevas aplicaciones móviles de la Universidad de Costa Rica y sus subsecuentes actualizaciones se realizará únicamente por medio del Centro de Informática siguiendo los lineamientos detallados en este documento. Por tanto, el flujo del proceso de publicación de una aplicación que se describe a continuación, es el único oficial autorizado en la Universidad de Costa Rica.

La siguiente ilustración muestra el proceso de publicación de una aplicación móvil:

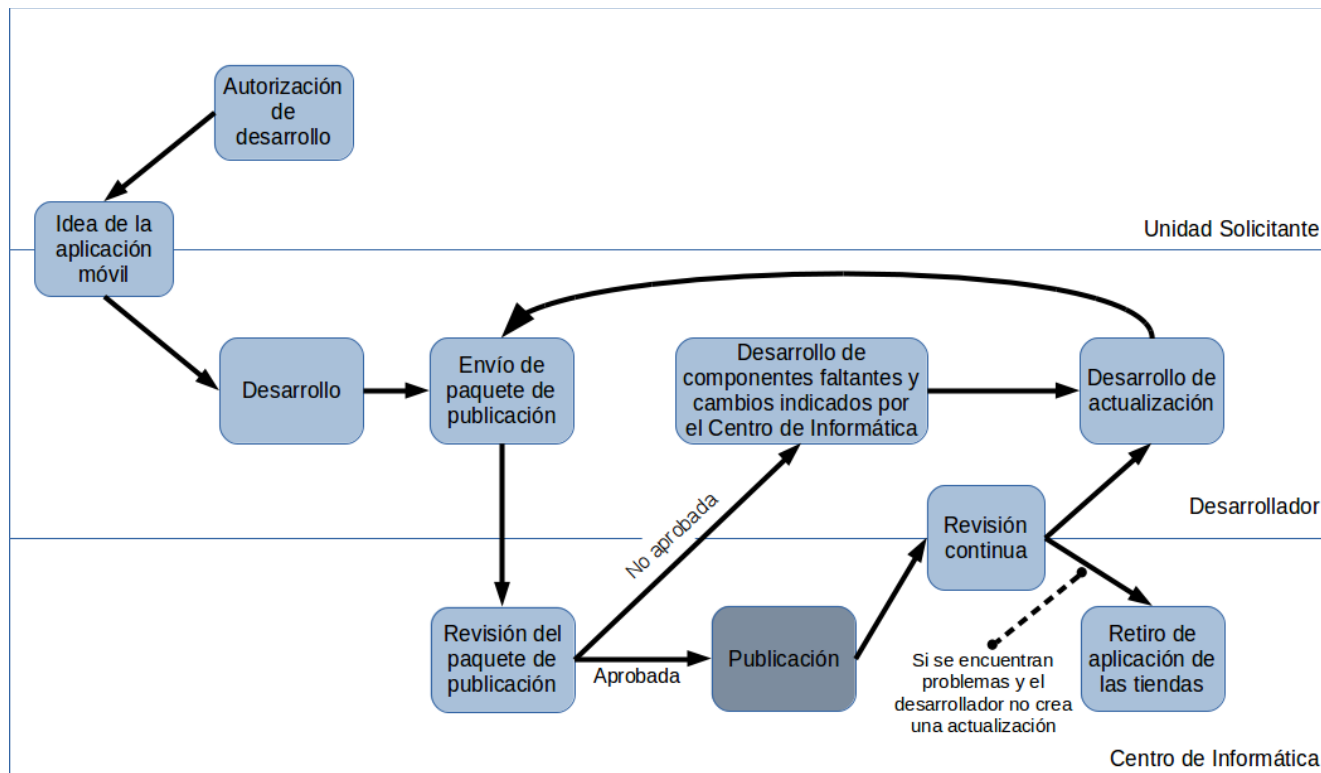


Ilustración 15. Proceso de publicación de una aplicación de la Universidad de Costa Rica

- La iniciativa de desarrollo de la aplicación surge desde la Unidad solicitante de la Universidad de Costa Rica. Dicha iniciativa debe estar autorizada por la Jefatura respectiva (responsable administrativo).
- El Centro de Informática brindará los accesos de Play Store y App Store respectivos a la persona o grupo desarrollador para que pueda subir y probar builds durante la fase de desarrollo. Durante esta fase no se puede publicar ningún build públicamente en las tiendas.
- La persona o grupo desarrollador (responsable técnico) deberá aplicar el presente manual de lineamientos durante todo el desarrollo de la aplicación.
- Una vez finalizado el desarrollo, el responsable administrativo debe enviar un oficio al Centro de Informática, mediante el cual solicitará la aprobación y publicación de la aplicación. Paralelamente el responsable técnico enviará un paquete al correo aplicaciones.moviles@ucr.ac.cr con el asunto “UCR@Móvil - Solicitud de publicación app <Nombre de la aplicación móvil>”. El contenido exacto del paquete se detallará más adelante en este documento.
- Una vez que el Centro de Informática reciba tanto el oficio como el paquete de la Unidad solicitante, asignará a dos personas profesionales del Área de Desarrollo de Sistemas y del Área de Captación y Promoción quienes efectuarán la revisión técnica de la aplicación. Dicha revisión técnica involucra la validación de la aplicación desde el punto de vista funcional, así como la documentación técnica y los datos para la publicación en las tiendas respectivas.
- En caso de que la aplicación sea aprobada, esta será enviada a las tiendas App Store y Play Store en un plazo máximo de 10 días hábiles, para que sea publicada utilizando la cuenta oficial de la Universidad de Costa Rica, administrada por el Centro de Informática. Asimismo, el Centro de Informática enviará de vuelta al correo electrónico la notificación de aprobación, junto con las condiciones de servicio de la publicación.
- En caso de que la aplicación no sea aprobada, el Centro de Informática enviará, de vuelta el correo electrónico, la notificación de rechazo, en la cual se deberán indicar los motivos por los cuales no ha sido aprobada. La Unidad efectuará los cambios recomendados y enviará nuevamente el paquete para revisión. Este proceso se

deberá repetir cuantas veces sea necesario para que la aplicación esté completamente lista para su publicación.

- Cuando la Unidad solicitante requiera realizar una actualización de la aplicación, deberá aplicar los pasos descritos anteriormente. En tal caso, el asunto del correo será “UCR@Móvil - Solicitud de actualización app <Nombre de la aplicación móvil>”.
- Una vez publicada la aplicación, como se ha detallado en otras secciones de este documento, el personal encargado del monitoreo de aplicaciones del Centro de Informática deberá efectuar una revisión continua de la aplicación.
- El Centro de Informática se reserva el derecho de retirar la aplicación de las tiendas por las siguientes razones:
 - Riesgos de seguridad
 - Contenido inapropiado
 - Si la aplicación se vuelve obsoleta (por ejemplo en casos de que la aplicación deje de funcionar en un nuevo sistema operativo o que se pierda la comunicación con el servicio web que utilice la aplicación).
 - Controles de calidad
 - Imagen y prestigio de la Universidad de Costa Rica

5.10.1 Contenidos del paquete de publicación

- Código fuente completo que incluya: archivos de código fuente, archivos de proyecto específicos de cada plataforma (iOS, Android, etc.) y globales en caso de que sea una aplicación híbrida, archivos de configuración y de descarga de librerías, paquetes y plugins, assets (archivos multimedia utilizados en la aplicación: imágenes, audio, videos, etc.). En general debe incluir todo lo necesario para poder compilar y ejecutar correctamente la aplicación móvil en todos los sistemas operativos para los que se desarrolló la aplicación. La Universidad de Costa Rica, por medio del Centro de Informática puede solicitar cualquier insumo faltante que considere necesario.
- Archivos de documentación detallados en la sección de “Formato y contenido de la documentación interna y externa”.

- Formulario CI-AID-F26 Solicitud de Publicación de Aplicación para Dispositivos Móviles debidamente completado.

5.10.1.1 Contenidos para la publicación en App Store (iOS)

- Código fuente de la aplicación iOS. El código fuente debe venir con el número de versión correcto para ser publicada sin problemas en la App Store, de lo contrario la aplicación no será aprobada.
- Nombre de la aplicación móvil (máximo 255 caracteres). En la App Store este nombre debe ser único, así que debe ser diferente a cualquier otro nombre de cualquier aplicación ya publicada en la App Store.
- Bundle ID. Este ID debe concordar con el indicado en el archivo Info.plist, de lo contrario la aplicación no será aprobada.
- Descripción de la aplicación (máximo 4000 caracteres).
- Ícono iOS. Debe ser de 1024x1024 pixeles y estar formato JPG, TIFF, PNG.
- Capturas de pantalla y video iOS. En general debe enviarse al menos una captura de pantalla para cada tipo de dispositivo que sea soportado (incluyendo iPhone y iPad). Las cantidades, tamaños y restricciones actuales pueden ser verificadas en el siguiente link:
https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/Appendices/Properties.html#//apple_ref/doc/uid/TP40011225-CH26-SW2
- Categoría primaria y secundaria. Se deben elegir dos de entre las siguientes categorías: Libros, Negocios, Catálogos, Educación, Entretenimiento, Finanzas, Comida y bebida, Juegos, Salud y Actividad Física, Estilo de Vida, Médica, Música, Navegación, Foto y Video, Productividad, Referencia, Red Social, Deportes, Viajes, Utilidades, Clima. Si en la categoría primaria se elige juegos se debe elegir dos de entre las siguientes categorías: Acción, Aventura, Arcade, Tablero, Cartas, Casino, Dados, Educacional, Familia, Niños, Música, Rompecabezas, Carreras, Juego de Rol, Simulación, Deportes, Estrategia, Trivia, Palabras.
- Palabras clave separadas por coma (opcional, máximo 100 caracteres en total).

- URL del sitio web de soporte.
- URL del sitio web de marketing (opcional).
- URL del sitio web de políticas de privacidad (opcional).
- Listado de cambios (si es una actualización)

5.10.1.2 Contenidos para la publicación en Play Store (Android)

- Código fuente de la aplicación Android. El código fuente debe venir con el número de versión correcto para ser publicada sin problemas en la Play Store, de lo contrario la aplicación no será aprobada.
- Nombre de la aplicación móvil (máximo 30 caracteres)
- Descripción corta (máximo 80 caracteres)
- Descripción larga (máximo 4000 caracteres)
- Ícono en tamaño 512x512 pixeles y estar en formato PNG 32bits (con alpha).
- Capturas de pantalla Android. En general debe enviarse al menos una captura de pantalla para teléfono y si la aplicación móvil soporta tabletas, también se debe incluir al menos una captura de pantalla para tableta. Las cantidades, tamaños y restricciones actuales pueden ser verificadas en el siguiente link: <https://support.google.com/googleplay/android-developer/answer/1078870>
- Banner para tienda Play Store de 1024x500 pixeles. Para más información de esta imagen ver el siguiente link: <http://android-developers.blogspot.com/2011/10/android-market-featured-image.html>
- Categoría. Se debe elegir una entre las siguientes categorías: Libros y Referencias, Negocios, Cómic, Comunicaciones, Educación, Entretenimiento, Finanzas, Salud y Bienestar, Bibliotecas y Demostración, Estilo de Vida, Fondos de Pantalla Animados, Medios y Video, Medicina, Música y Audio, Noticias y Revistas, Personalización, Fotografía, Productividad, Compras, Redes Sociales, Deportes, Herramientas, Transporte, Viajes y Local, Clima, Widgets, Juegos. Si en la categoría se elige la categoría de Juegos se debe elegir una entre las siguientes: Acción, Aventura,

Carreras, Cartas, Casino, Casual, Deportes, Educativos, Estrategia, Juegos de mesa, Juegos de palabras, Juegos de rol, Música, Preguntas y respuestas, Rompecabezas, Sala de juegos, Simulación.

- Listado de cambios (si es una actualización)
- URL del sitio web de políticas de privacidad (opcional).
- Archivo APK con un nombre significativo, el cual es único y permanente, no se puede renombrar a futuro. El archivo debe cumplir lo siguiente:
 - El APK debe exportarse sin firmar.
 - Debe tener un tamaño menor a 50MB.
 - Se pueden subir varios archivos para diferentes configuraciones de dispositivos.

5.10.2 Otros lineamientos de publicación

Los siguientes son otros lineamientos por seguir sobre la publicación de las aplicaciones móviles de la Universidad de Costa Rica:

- La aplicación móvil no puede contener anuncios de índole comercial no relacionado con el quehacer universitario.
- Por defecto la aplicación móvil se publicará gratuita y en todas las tiendas a nivel mundial. Si por alguna razón la aplicación se debe restringir los países en los que la aplicación estará disponible, se debe dar el listado de países permitidos y las razones de dicha restricción.
- El Centro de Informática puede retirar la aplicación móvil de las tiendas en cualquier momento por las razones ya mencionadas. Si esto sucede el responsable de la aplicación recibirá el detalle de las razones por las cuales la aplicación fue retirada.

5.10.3 Reportes

La Unidad solicitante puede requerir reportes de la publicación de la aplicación móvil, por ejemplo de número de descargas, clasificación, críticas y comentarios de usuarios entre otros. Para la solicitud de un reporte el responsable técnico debe enviar un correo a aplicaciones.moviles@ucr.ac.cr con el asunto "UCR@Móvil - Solicitud de reporte de app

<Nombre de la aplicación móvil>” y detallar los requerimientos del reporte. El Centro de Informática enviará el reporte en un plazo de 5 días.

Cuando se reciba un reporte de errores por parte de App Store o Play Store, el Centro de Informática lo reenviará al responsable administrativo de la Unidad solicitante presentado en el paquete de publicación.

6 CONCEPTOS Y ABREVIATURAS






- **API:** Interfaz de Programación de aplicaciones. Es un conjunto subrutinas, funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro software como una capa de abstracción.
- **Canvas:** Elemento HTML incorporado en HTML5 que permite la generación de gráficos dinámicamente por medio de scripting.
- **Eavesdropping:** Es una ataque que consiste en capturar paquetes en una red transmitidos por otros dispositivos y leer los datos de dichos paquetes para obtener información sensible o confidencial.
- **Framework:** Estructura conceptual y tecnológica con artefactos y módulos concretos de software que sirve de base para la organización y desarrollo de aplicaciones.
- **Gesture:** Movimiento pregrabado que una pantalla táctil puede reconocer. En dispositivos móviles incluye el tap, pinch, spread, entre otros.
- **GET:** Es un método de solicitud de datos entre un cliente y un servidor.
- **HTTP:** Protocolo de transferencia de hipertexto. Es el protocolo usado en cada transacción de la World Wide Web.
- **HTTPS:** Protocolo seguro de transferencia de hipertexto. Es un protocolo de aplicación basado en HTTP que crea un canal cifrado más apropiado para el tráfico de información sensible.
- **Inyección de SQL:** Es un método de infiltración de código intruso que hace uso de las vulnerabilidades de las entradas de texto no validadas para atacar la base de datos de una aplicación.

- **Landscape:** Vista cuando el dispositivo móvil esta de forma horizontal.
- **Look and Feel:** Conjunto de propiedades y características que le dan una identidad visual a una interfaz gráfica.
- **Man-in-the-middle:** Es un ataque que lee y modifica los mensajes enviados entre dos dispositivos sin que ninguno de ellos lo sepa.
- **Pinch:** Gesture que se da al tocar la pantalla táctil con dos dedos separados e irlos juntando lentamente. Típicamente utilizado para hacer zoom in.
- **Portrait:** Vista cuando el dispositivo móvil esta de forma vertical.
- **POST:** Es un método de envío de datos entre un cliente y un servidor.
- **RAM:** Memoria de acceso aleatorio. Es la memoria de trabajo que utilizan las aplicaciones durante su ejecución.
- **Spinner:** Ícono giratorio que normalmente se utiliza para indicar al usuario que debe esperar a alguna acción de la aplicación.
- **Splash Screen:** Es la imagen que se muestra al abrir una aplicación móvil cuando el usuario la abre y antes de que esta empiece a ejecutarse.
- **Spread:** Gesture que se da al tocar la pantalla táctil con dos dedos juntos e irlos separando lentamente. Típicamente utilizado para hacer zoom out.
- **SQL:** Leguaje de programación popularmente utilizado para la gestión de bases de datos relacionales.
- **SQLite:** Es un sistema de gestión de bases de datos relacionales cuya característica principal es que se utiliza embebido a la aplicación y no utilizando el modelo de cliente-servidor.
- **SVG:** Gráficos vectoriales redimensionables. Es una especificación para describir gráficos vectoriales bidimensionales en formato XML.
- **Swipe:** Gesture que se da al tocar la pantalla táctil con un dedo y deslizarlo en alguna

dirección.

- **Tap:** Gesture que se da al tocar pantalla táctil y quitar el dedo rápidamente.
- **WebGL:** Es una especificación estándar para mostrar gráficos 3D en navegadores web utilizando OpenGL.

7 APROBACIÓN

Actividad	Responsable	Fecha	Firma
Elaboración	Luis Carlos Chacón (contratación externa)	01/02/2016	
Revisión	Jose Valverde Cerdas, AID	05/02/2016	
	Luis Loría Chavarría, AID	15/02/2016	
	Allan Fonseca Calvo, ACP	10/08/2016	
	Jorge Alvarado Zamora, ADS	25/10/2016	
	Luis Jiménez Cordero, Subdirección	20/07/2018	
Aprobación	Alonso Castro Mattei, Dirección	12/09/2018	