GUÍA BÁSICA DE RECOMENDACIONES PARA EL DESARROLLO DE *APPS* DE SALUD Y ATENCIÓN SOCIAL

Cómo crear una aplicación móvil segura, accesible, usable e interoperable.

Versión 1.0

TIC Salut Social
Tecnologia, innovació i transformació digital.

Generalitat de Catalunya

Esta guía está destinada a todas aquellas personas que deseen iniciarse en el entorno mHealth y quieran tener una visión general de cuáles son los aspectos a tener en cuenta a la hora de desarrollar una aplicación móvil del entorno de la salud y/o del bienestar social.

GUÍA BÁSICA DE RECOMENDACIONES PARA EL DESARROLLO DE *APPS* DE SALUD Y ATENCIÓN SOCIAL

TIC Salut Social
Tecnologia, innovació i transformació digital.
Generalitat de Catalunya

Primera edición: agosto de 2018

Presentación

En el marco de la adopción de las políticas mHealth, el gobierno catalán aprobó a principios del 2015 el Plan Estratégico de Movilidad instando a las instituciones del sector a acercar a los usuarios los servicios sanitarios y de atención a la persona a través de las tecnologías de la movilidad y, a su vez, facilitar la transformación de los distintos procesos asistenciales y sociales para conseguir una atención integrada y próxima al paciente. De acuerdo con este encargo y bajo el paraguas de la CE, la Fundació TIC Salut Social, organismo del Departamento de Salud de la Generalitat de Catalunya, ha trabajado en la puesta en marcha de un modelo de acreditación de aplicaciones móviles del sector de la salud y de los servicios sociales que permite certificar aquellas aplicaciones que son aptas para facilitar y mejorar el seguimiento de la salud de los ciudadanos, aportando garantía de calidad y seguridad.

Fruto de los conocimientos generados, del trabajo en red con otros agentes del sector y de una experiencia piloto enriquecedora, nace esta guía de recomendaciones para el desarrollo de apps. La Fundació TIC Salut Social apuesta por un proceso de acreditación abierto y transparente a todo tipo de iniciativas cuyo objetivo es transmitir confianza a la ciudadanía y a los profesionales. En esta línea, la guía recoge una serie de recomendaciones para acompañar al desarrollador en la programación de aplicaciones móviles seguras, accesibles, usables e interoperables.

Esperamos que los pasos y recomendaciones que incluimos en la guía sean de utilidad, tanto para las administraciones como para las empresas, universidades y start-ups que trabajan para compartir su conocimiento con el ciudadano, y que esta se convierta en una herramienta de consulta al alcance de todas las personas.

Agradecimientos

Entidades colaboradoras

La Fundació TIC Salut Social quiere agradecer muy especialmente a CSV Experts, iSalus, m4social, la Taula d'Entitats del Tercer Sector, Pasiona y la UPC (Universitat Politècnica de Catalunya) su plena disposición para el desarrollo de esta guía y por aportar su amplio conocimiento sobre las distintas temáticas en las que han participado.

Sumario

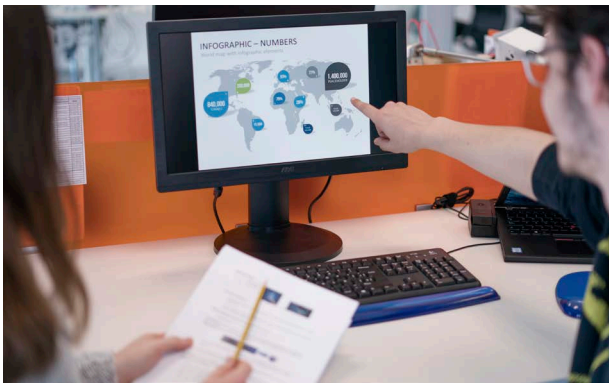
1. Introducción

Las tecnologías de la movilidad en el ámbito de la salud son utilizadas de forma cotidiana por los ciudadanos a nivel mundial, ya que a través del teléfono móvil hacen uso de aplicaciones de todos los ámbitos, incluidos el de la salud, el del bienestar y el social. Actualmente se pueden encontrar en el mercado más de 325.000 apps de salud y bienestar, lo que determina una demanda creciente y hace necesario el establecimiento de marcos reguladores que permitan adecuar esta tecnología en las fases iniciales de los proyectos, aportando garantías a los usuarios y estableciendo así verdaderas herramientas de valor y de confianza.

Durante los últimos años, la Fundació TIC Salut Social, junto con el Departamento de Salud y el Departamento de Trabajo, Asuntos Sociales y Familias de la Generalitat de Catalunya, ha trabajado en el sector de las aplicaciones móviles desempeñando funciones de observatorio de las nuevas tendencias en movilidad y definiendo un proceso de acreditación de aplicaciones móviles del sector de la salud y la atención a la persona.

Objetivos

A partir de estos modelos de acreditación y de la participación activa de distintas entidades y colaboradores externos, se ha desarrollado esta guía, que recoge las recomendaciones que deberían tenerse en cuenta a la hora de desarrollar cualquier aplicación móvil destinada a la atención a la persona, especialmente en los entornos de la salud y social.



Partes de la guía

Esta guía aborda los aspectos tanto técnicos como funcionales que cualquier aplicación móvil ha de incorporar en las fases iniciales del proyecto de desarrollo. Este contenido está estructurado en los siguientes apartados:

- **Introducción:** presentación del objetivo y relación de las partes de la guía.
- **Tipos de aplicaciones:** *apps* que existen, sus principales características y casos en los que son más adecuadas.
- ***Markets* de aplicaciones:** entornos de distribución de las aplicaciones móviles y consejos para su publicación en los mismos.
- **Proceso de desarrollo de aplicaciones móviles:** en este apartado se describen las metodologías de trabajo más utilizadas actualmente para desarrollar aplicaciones y cuáles son los roles profesionales que debe haber en el equipo de un proyecto de *software* de aplicaciones móviles.
- **Lenguajes y entornos de desarrollo:** en este apartado se describen las diferentes tecnologías para cada tipo de aplicación y sistema operativo. También se listan los *softwares* de desarrollo que se pueden utilizar en cada caso.
- **Consejos para el desarrollo y la publicación de aplicaciones:** en este apartado se listan algunos consejos y recomendaciones a tener en cuenta desde la fase de planificación del proyecto hasta la publicación de la aplicación.
- **Seguridad y protección de datos:** en este apartado se listan recomendaciones de seguridad y protección de datos que deben incluir las aplicaciones a desarrollar.
- **Accesibilidad:** métodos y técnicas para garantizar que una aplicación pueda ser utilizada por el mayor número posible de personas.
- **Usabilidad:** criterios a tener en cuenta a la hora de diseñar y desarrollar una aplicación para garantizar su facilidad de uso.
- **Experiencia de usuario:** relación de consejos destinados a mejorar la percepción del usuario al utilizar una app.
- **Estándares de comunicación y representación de información:** presentación del concepto de interoperabilidad y de los principales estándares para garantizarla en el entorno móvil.
- ***Medical device*, la aplicación móvil como producto sanitario:** relación de puntos clave para ayudar a identificar si una app es un producto sanitario, así como los aspectos a tener en cuenta en caso de serlo.
- **Entidades participantes:** relación de entidades que han elaborado esta guía.
- **Autores:** detalle de los autores que han participado en la redacción del presente documento.

2. Área mHealth de la Fundació TIC Salut Social

El área mHealth nace a partir del Plan Maestro de Movilidad, un proyecto impulsado por el Departamento de Trabajo, Asuntos Sociales y Familias de la Generalitat de Catalunya y por la Fundació TIC Salut Social, con la colaboración de la Fundación Mobile World Capital Barcelona.

Esta área pretende establecer un marco de referencia común que incluya las estrategias de movilidad de estos departamentos para permitir una visión completa de la influencia de la movilidad en la prevención y promoción de la salud y en la atención social a fin de aportar soluciones y servicios a través de sistemas móviles y con el objetivo principal de mejorar y personalizar la salud y el bienestar de las personas.

En esta línea, mHealth aproxima los servicios sanitarios y sociales a la ciudadanía por medio de tecnologías de movilidad como, por ejemplo, los smartphones, las tabletas o los ordenadores portátiles, herramientas que permiten interactuar de forma ágil y personalizada.

Uno de los servicios que se prestan es la acreditación de aplicaciones móviles del entorno de la salud y social.

Proceso de acreditación de aplicaciones móviles

La Fundació TIC Salut Social ha creado un modelo de acreditación de aplicaciones móviles del sector de la salud y de los servicios sociales que permite certificar si una aplicación es apta para facilitar y mejorar el seguimiento de la salud de los ciudadanos, aportando garantía de calidad y seguridad.

Las aplicaciones son sometidas a un conjunto de criterios de dominio público, que pueden ser consultados en la web de la Fundació. Promoviendo el uso seguro de aplicaciones de calidad y dando a conocer aquellas que tienen un alto valor añadido, se contribuye a mejorar el empoderamiento del ciudadano, haciéndole consciente y corresponsable de su propia salud.

El diseño del modelo de acreditación se ha basado en un consenso transversal entre distintos profesionales y agentes del sector: expertos del ámbito tecnológico, profesionales sanitarios (médicos, enfermeras, psicólogos y trabajadores sociales), profesionales de la comunicación en salud, pacientes expertos, representantes institucionales y ciudadanos.

Esta diversidad ha facilitado la adopción de un modelo que engloba todas las visiones y desde distintos ámbitos, lo que ha permitido obtener un conjunto de criterios agrupados en cuatro áreas de experiencia: usabilidad y experiencia de usuario; tecnología y fiabilidad de la aplicación; seguridad y confidencialidad de datos; y funcionalidad y evaluación de contenidos.

Estas áreas de experiencia son evaluadas por separado, siendo, cada una de ellas, independientes y adaptables a las distintas demandas del sector.

El Col·legi Oficial d'Infermeres i Infermers de Barcelona (COIB), el Col·legi Oficial de Metges de Barcelona (COMB), el Col·legi Oficial de Llicenciats en Educació Física i Ciències de l'Activitat Física i l'Esport de Catalunya (COPLEFC), la Associació d'Infermeria Familiar i Comunitària (AIFiCC), la Societat Catalana d'Especialistes en Psicologia Clínica (SCEPC) y la Societat Catalana de Medicina Familiar i Comunitària (CAMFIC) son algunas de las entidades que colaboran en la evaluación de estas áreas.

Las aplicaciones que superan el proceso de acreditación reciben un sello de confianza y son publicadas en el catálogo de la web de la Fundació para darles visibilidad y ponerlas a disposición del sector a fin de que se puedan consultar y utilizar.

La Fundació TIC Salut Social apuesta por un proceso de acreditación abierto y transparente a todo tipo de iniciativas con el objetivo de transmitir confianza a la ciudadanía y a los profesionales.

Toda la documentación relativa al proceso de acreditación, así como los criterios a los que son sometidas las apps, se puede consultar en la web de la Fundació TIC Salut Social.

3. Tipos de aplicaciones

Uno de los principales elementos a tener en cuenta a la hora de planificar el desarrollo de una aplicación es elegir qué tipo se quiere desarrollar. Esta decisión tiene implicaciones en muchos aspectos, como los conocimientos necesarios del equipo, el coste del proyecto o el resultado final obtenido. A continuación se presentan los diferentes tipos de aplicaciones existentes y sus principales características.

Nativas

Las aplicaciones nativas se desarrollan específicamente para cada sistema operativo con un lenguaje de programación compatible. Este tipo de *apps* permiten explotar todos los recursos del dispositivo (como la cámara o el GPS), así como utilizar las APIs propias del sistema operativo.
Las aplicaciones nativas permiten optimizar mejor el rendimiento, un aspecto clave en las aplicaciones que requieren de una gran cantidad de recursos, como los juegos 3D.
Este tipo de *apps* se distribuyen en los *markets* oficiales y se pueden utilizar sin conexión a internet, aunque la mayoría de ellas tienen funcionalidades que requieren de conectividad para ejecutar determinadas funcionalidades o, directamente, no se pueden usar sin conexión.

El coste de desarrollar una aplicación nativa, en comparación con el de otro tipo de *apps*, normalmente es mayor, ya que, si bien en caso de usar como *back-end* un servidor externo este sí será compartido por las distintas apps nativas, el *front-end* deberá desarrollarse desde cero (o casi) para los diferentes sistemas operativos. Ello también implica tener que mantener las distintas aplicaciones en fase de producción.

Se recomienda el desarrollo de aplicaciones nativas en caso de *apps* que requieran de un alto rendimiento o consumo de recursos de *hardware*.

Web

Una aplicación web es aquella que se utiliza accediendo a una URL a través de un navegador (como Chrome, IE o Safari). Este tipo de *apps* se pueden ejecutar en cualquier sistema operativo, tanto en dispositivos móviles como en ordenadores de sobremesa, y requieren de conexión a internet. **Es importante destacar que las aplicaciones web no permiten utilizar los recursos del dispositivo, no es necesario que cumplan las condiciones de los *markets*, ya que no se publicarán, y el usuario siempre dispone de la última actualización de forma transparente.**
Debido a que son multiplataforma, el desarrollo de aplicaciones web suele ser más económico, ya que no necesitan diferentes versiones o adaptaciones para cada sistema operativo del dispositivo. No obstante, sí cabe considerar la fragmentación entre los distintos navegadores (y versiones de cada uno), así como las diferencias de resolución de las pantallas o las capacidades del *hardware* como la memoria o el procesador (especialmente importantes en aplicaciones pesadas).

Se recomienda desarrollar este tipo de apps cuando se desee tener una única aplicación, incluso entre dispositivos móviles y PC. Pero deberá tenerse en cuenta que puede ser necesaria la realización de modificaciones, en algunos casos bastante importantes, a nivel de interfaz gráfica, para que la aplicación se adapte y sea usable en las diferentes plataformas. Uno de sus principales inconvenientes es la limitación para usar los elementos de *hardware* del dispositivo u otras opciones como las notificaciones nativas.

Híbridas

La aplicación híbrida está a medio camino entre la aplicación nativa y la aplicación web. Está construida con los mismos lenguajes y estructura que la *app* web, pero se distribuye del mismo modo que la *app* nativa. **Las aplicaciones híbridas tienen los mismos requerimientos de conectividad a internet que las nativas, así como la misma capacidad para acceder a los recursos de *hardware* del dispositivo.**

Este tipo de aplicaciones generalmente implican un menor coste, en comparación con el de las nativas, debido a que se realiza un único desarrollo (y, por lo tanto, también un único mantenimiento) que sirve para los diferentes sistemas operativos. Se puede dar el caso de que se tengan que realizar ajustes menores por problemas puntuales con alguno de los sistemas operativos, pero el esfuerzo siempre será mucho menor que el que implica desarrollar diferentes aplicaciones nativas.

Las aplicaciones híbridas son recomendables para desarrollar herramientas de gestión o de visualización de información en las que se desee reducir el coste de desarrollo y mantenimiento. Ejemplos adecuados de apps de este tipo pueden ser las que permiten pedir cita en línea, realizar *e-consultas* o conformar repositorios de consejos prácticos de salud.

Generadas

Las aplicaciones generadas también se denominan «falsas nativas», dado que el proceso consiste en escribir una app en un lenguaje de programación no nativo, como, por ejemplo, C#, y después generar las aplicaciones nativas automáticamente con el lenguaje específico de cada plataforma. De este modo, se consiguen aplicaciones similares a las nativas, aunque no lleguen a los mismos niveles de optimización. Esta aproximación permite reducir costes porque gran parte de la aplicación es común a todas las versiones, pero es posible que algunos elementos puntuales deban programarse con lenguaje nativo. Por ello, se recomienda tener nociones de los lenguajes propios de cada sistema operativo y poder desarrollar así un código específico.

Las aplicaciones generadas son recomendables para controlar los costes de desarrollo y mantenimiento, pero hay que valorar bien las ventajas y limitaciones de cada plataforma concreta que permite generarlas. Un ejemplo es el motor gráfico Unity, que permite crear aplicaciones 3D compatibles con PC, móviles y consolas, con un único desarrollo.

4. *Markets* de aplicaciones

Las tiendas de aplicaciones, también conocidas con el anglicismo markets, son plataformas web que permiten vender y/o distribuir aplicaciones móviles. En función del sistema operativo al que se dirijan, las *apps* se publicarán en un *market* específico, siendo Google Play Store para Android y App Store para iOS los más utilizados. También existen otras tiendas de aplicaciones, como Amazon App Store, la principal alternativa a Google Play Store; F-Droid, que tiene la restricción de que todas las aplicaciones que contiene son de código libre y, por lo tanto, se puede acceder a su código y/o contribuir a él; GetJar, una tienda multiplataforma (Android, iOS, Windows Mobile); o Microsoft Store, que es la tienda oficial de Microsoft.

En este apartado se presentan más en detalle los *markets* Google Play Store y App Store, con algunos consejos destacados sobre cómo publicar aplicaciones en ellos.

Google Play Store

Google Play es la tienda oficial de la compañía Google y en ella se pueden encontrar libros digitales, música, películas, noticias y publicaciones, así como *apps* y juegos. El apartado específico para las aplicaciones móviles se denomina Google Play Store y contiene más de 3,7 millones de apps organizadas por diferentes categorías, entre las cuales la de «Salud y *fitness*».

Para publicar una aplicación en el Google Play Store, es imprescindible cumplir las políticas del **Programa para desarrolladores** y aceptar el **Acuerdo de distribución para desarrolladores**. La compañía ofrece consejos a seguir para cumplir estas políticas, relativos a los contenidos, la suplantación de identidad, la propiedad intelectual, la privacidad, la seguridad y el engaño, la obtención de ingresos o la presencia de publicidad. Estos consejos se pueden consultar en el siguiente enlace:
Centro de políticas para desarrolladores.

A modo de resumen, se destacan los siguientes aspectos a tener en cuenta a la hora de publicar una aplicación en el Google Play Store:

1. Realizar un **plan de pruebas** exhaustivo que incluya tanto tests unitarios como de integración, para minimizar los posibles errores que puedan surgir y que la **experiencia de usuario se vea alterada en la menor medida posible.**
2. La aplicación se puede publicar en modo *alpha* o *beta* para asegurar su calidad antes de publicarla en producción.
3. Revisar y garantizar que la *app* **cumpla todas las políticas** del *market*.
4. Asegurar que la **descripción** de la aplicación sea correcta para todos los públicos, y que sea **comprensible y concisa**.
5. Asegurar que se cuente con los **derechos de autor** de todas las imágenes que se utilizan, incluidas las que se muestran en el *market*.
6. Si la aplicación utiliza datos de los usuarios, asegurar que se incluya una **política de privacidad** que exponga los datos que se recogen y su finalidad.
7. Asegurar que la aplicación y los anuncios que se visualizan **no incluyan contenidos inadecuados** (como contenido sexual explícito, de incitación al odio, amenazas o acoso). Además, si se incluyen anuncios en la aplicación estos no pueden ser ni molestos ni engañosos.
8. Incluir toda la **información básica (con traducciones, si se quiere publicar en diferentes dominios lingüísticos) y relevante**

sobre la aplicación: una **descripción** precisa, **imágenes** que muestren el funcionamiento, el etiquetado PEGI correspondiente y, en su caso, un vídeo de demostración.

9. Elegir el icono que siga las guías de recomendación de Android.
10. Asegurar que el **correo electrónico de contacto asociado a la *app* sea el correcto** para que Google pueda comunicar cualquier incidencia con la aplicación.
11. **Firmar la aplicación en modo *release*.**
12. **Elegir adecuadamente los nombres de los *packages* ya que son únicos, permanentes y no se pueden reutilizar en el futuro.**
13. Se pueden publicar distintas aplicaciones (APKs) en la misma ficha de la aplicación para diferentes configuraciones y/o compatibilidad de dispositivos.
14. Optimizar la aplicación: deshabilitar los logs que no sean necesarios, los recursos que no se usen, etc.
15. Verificar que los permisos definidos en el manifiesto sean realmente necesarios.
16. Verificar la versión mínima escogida.
17. **Corregir los errores** que se comuniquen o ponerse en contacto con Google para comunicar cualquier disconformidad o necesidad adicional de información.
18. Emplear herramientas como Crashlytics o Google Analytics para simplificar la gestión del mantenimiento correctivo y evolutivo y analizar cómo se está utilizando la aplicación.

App Store

El App Store también es una plataforma de distribución digital, pero es propiedad de la compañía Apple Inc. En este *market* se pueden comprar y/o descargar más de 2,1 millones de apps, también organizadas por categorías como la de «Salud y *fitness*».

Todas las aplicaciones que se publican en el App Store son revisadas previamente por un experto de Apple, pero los requisitos que hay que cumplir son bastante similares a los indicados para la publicación en el Google Play Store. No obstante, se deben tener en cuenta las siguientes particularidades:

1. Para que los expertos de Apple puedan revisar la aplicación, es necesario crear un perfil en iTunes Connect y proporcionar toda la información relevante relativa a la app, cuyo objetivo es ofrecer una idea clara sobre su funcionalidad.
2. Las capturas de pantalla que se incorporen a iTunes Connect deben coincidir exactamente con las del producto final. De no ser así, no se pasará la validación de seguridad.
3. No se puede incluir ninguna referencia ni logotipo de Google u otras plataformas como Windows o Amazon en las aplicaciones que se deseen publicar.
4. Las aplicaciones *in-house* (de distribución interna) no se pueden desplegar en esta tienda. Este tipo de aplicaciones deben desplegarse en el programa Enterprise de Apple.
5. Deben emplearse herramientas como Crashlytics o Google Analytics para realizar el seguimiento de la utilización de la aplicación.

Per tal de provar versions *beta* de les aplicacions, Apple posa a disposició dels desenvolupadors l'eina TestFlight. Aquesta *App* permet publicar la versió de prova d'una aplicació, i convidar fins a 10.000 usuaris per a que se la descarreguin i la provin, proporcionant el seu feedback. L'eina també permet realitzar validacions internes per part de l'equip de desenvolupament, i amb una limitació de fins a 30 dispositius.

Al següent enllaç es poden consultar les recomanacions i bones pràctiques que s'especifiquen al web de l'App Sotre, relatives a la revisió de les aplicacions, la interfície d'usuari o el màrqueting: https://developer.apple.com/app-store/guidelines/.

5. Proceso de desarrollo de aplicaciones móviles

El proceso o ciclo de vida del desarrollo del *software* tiene características diversas en función del ámbito, las funcionalidades o los riesgos asociados al tipo de aplicación a desarrollar. Según estas variables, aspectos como la complejidad, la duración o el coste pueden variar de forma muy significativa. En este apartado se presentan las metodologías a aplicar para desarrollar *apps*, con sus correspondientes fases y roles.

Fases del proceso de desarrollo

El proceso de desarrollo se puede gestionar siguiendo diferentes metodologías, y la elección de una u otra se realizará según el tipo de proyecto que se esté llevando a cabo. No obstante, existen algunas etapas básicas comunes a casi todos los procesos de desarrollo del *software*, las cuales se presentan a continuación:

- **Toma de requerimientos:** se identifican y definen inicialmente los casos de uso o funcionalidades a cubrir por la *app*, los cuales podrán sufrir modificaciones a lo largo del proceso de desarrollo si se considera oportuno. Las modificaciones pueden darse por varios factores como cambios en las necesidades de los potenciales usuarios, cambios legales o limitaciones de recursos, entre otros.

- **Diseño:** se realiza un análisis funcional y técnico para determinar qué requerimientos se quieren implementar y cómo se hará.

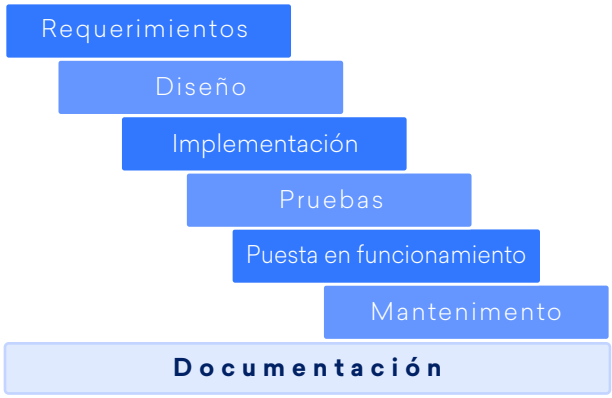
- **Implementación:** en esta etapa se desarrollan las funcionalidades, los elementos gráficos, etc., especificados en el diseño.

- **Pruebas:** se realizan todas las pruebas necesarias para garantizar que el software cumple con las necesidades identificadas y con el diseño:
  - Las **pruebas unitarias** son aquellas que evalúan cada funcionalidad de forma aislada del resto del sistema, como puede ser un método o una función.
  - Las **pruebas de integración** consisten en probar cómo funcionan conjuntamente dos o más elementos a los que previamente se les debería de haber aplicado las pruebas unitarias pertinentes.
  - Las **pruebas de estrés** se centran en el rendimiento del *software* cuando este es llevado al límite, para ver hasta qué punto sigue funcionando con normalidad y qué sucede cuando dicho umbral se sobrepasa.
  - Las **pruebas de penetración** son aquellas en las que se simula que hay un atacante malintencionado que, mediante todas las técnicas a su alcance, tratará de vulnerar la seguridad de la aplicación para extraer datos, corromper su funcionamiento, etc.

- **Puesta en funcionamiento:** el *software* se configura, se despliega y se distribuye poniéndolo al alcance de los usuarios para que empiecen a usarlo

- **Mantenimiento:** proceso durante el cual se van aplicando medidas para corregir posibles errores de la aplicación, así como mejoras para satisfacer nuevos requerimientos que puedan surgir a lo largo del tiempo.

- **Documentación:** se trata de una tarea transversal a realizar durante todo el ciclo de vida del producto (incluida la etapa de mantenimiento), en la que deben documentarse las funcionalidades y los casos de uso a los que dar respuesta, el diseño interno del *software*, las modificaciones que se realicen a lo largo del proceso, etc.



Metodologías de desarrollo

Como se ha indicado anteriormente, existen diferentes metodologías para el desarrollo de aplicaciones, las cuales se clasifican en paradigmas en función de sus características principales. Dos de los paradigmas más utilizados actualmente son:

- La **metodología clásica, en cascada**, en la que las fases se suceden de forma lineal. Este enfoque es poco flexible, dado que, una vez finalizada una fase, se pasa a la siguiente con la idea de no volver a abordarla posteriormente. Las metodologías en cascada resultan adecuadas para proyectos en los que los requerimientos y el alcance son muy claros y cuando la tecnología que se empleará es muy conocida por parte del equipo de desarrolladores.

- Las **metodologías ágiles** buscan ofrecer una gran flexibilidad, permitiendo adaptar el desarrollo a cualquier cambio que se requiera. En estas aproximaciones, la modificación de los requerimientos es constante y habitual, y se percibe como un medio para aumentar las posibilidades de éxito de un proyecto y priorizar el *software* que funciona. De este modo, no es necesario tener todos los requerimientos definidos ni implantados para poder poner la aplicación al alcance de los usuarios, lo que reduce el *time-to-market*.

Metodologías ágiles

Las metodologías ágiles se basan en los principios del llamado «manifiesto ágil», que se pueden agrupar en los cuatro puntos siguientes:

- Priorizar los individuos y sus interacciones por encima de los procesos y sus herramientas.
- Priorizar el *software* que funciona por encima de la documentación exhaustiva.
- Priorizar la colaboración con el cliente por encima de la negociación contractual.
- Priorizar la respuesta al cambio por encima del seguimiento de un plan previamente definido y fijo.

Es muy importante tener en cuenta que priorizar no quiere decir que se deban descuidar aspectos como la documentación o la negociación contractual, dado que, por ejemplo, si el contrato es demasiado ambiguo o la documentación realizada está incompleta, es muy probable que se acaben produciendo muchas incidencias durante el ciclo de vida del *software*.

En un proyecto ágil el coste también puede variar, ya que depende de los requerimientos que se vayan redefiniendo.
Otra de las características principales de estas metodologías es el papel que tienen los clientes: hay que establecer una relación de colaboración y confianza para poder definir los requerimientos y realizar las validaciones de las funcionalidades desarrolladas.

Las metodologías de este tipo son adecuadas para proyectos poco definidos, bastante cambiantes y en los que el cliente está muy implicado y dispuesto a invertir esfuerzo.

Algunas de las metodologías ágiles más conocidas son:

Scrum

Es un marco que define etapas incrementales, flexibles, y que permite responder a necesidades cambiantes. Resulta adecuado para proyectos con una cierta complejidad en los que no se dispone de todos los requerimientos definidos completamente al inicio, y en los que estos son susceptibles de ser modificados o ampliados. En cuanto a la composición del equipo, se recomienda que este tenga un volumen de tres a diez personas.

Scrum trabaja sobre iteraciones que denomina ***sprints***, al finalizar las cuales debe haberse desarrollado un incremento del producto que sea potencialmente entregable. La duración de un *sprint* normalmente comprende de una a cuatro semanas.

Un equipo Scrum lo conforman tres roles muy diferenciados:

- ***Product Owner:*** rol cuyas funciones son mediar entre el cliente y el equipo, buscar financiación y decidir qué características y funcionalidades son indispensables para que el producto pueda ser entregado. En relación con las tareas no indispensables, este las ordena asignándoles una prioridad.

- ***Scrum master:*** rol cuya función es orientar al equipo en los principios y conceptos Scrum, además de facilitar el proceso.

- **Equipo de desarrollo:** equipo relativamente pequeño, multifuncional y autoorganizado. Se considera que si el equipo es muy grande, se hace difícil lograr la autogestión necesaria, ya que además no existe jerarquía dentro del mismo. Por otro lado, si es muy pequeño, es difícil que se tengan todos los conocimientos necesarios para el proyecto. El equipo se encarga de negociar con el *product owner* qué desarrollos se realizarán en cada *sprint* y de llevarlos a cabo.

El ciclo de vida Scrum se inicia con la creación, por el *product owner*, de una lista de requerimientos ordenados por prioridad llamada «***backlog*** de producto». A continuación, el *product owner* negocia con el equipo de desarrollo qué requerimientos se desarrollarán en el próximo *sprint*. Cuando empieza el sprint, el equipo de desarrollo se encarga de gestionarse para desarrollar las tareas incluidas en el *sprint*. Diariamente se hace una reunión rápida de pie entre el equipo de desarrollo, el *Scrum master* y el *product owner* para aclarar dudas, solucionar problemas, comentar temas, etc.

Una vez finalizado el *sprint*, se lleva a cabo la revisión del mismo durante la cual el equipo presenta el desarrollo realizado. El ciclo de la iteración finaliza con la realización de la retrospectiva del *sprint*, en la que se valoran posibles mejoras a partir de la experiencia vivida. Tras un *sprint* se inicia otro, y el proceso se repite hasta tener desarrollados todos los requerimientos.

Programación extrema (XP)

Esta metodología tiene un doble objetivo: por un lado, mejorar la calidad del software y, por el otro, la de las condiciones del equipo de desarrollo. Para lograr este doble hito, se definen los cinco valores básicos siguientes:

- **Comunicación:** el desarrollo del *software* se considera un trabajo de equipo, por lo que es indispensable la buena comunicación entre los miembros que lo conforman, que debe ser, preferentemente, cara a cara.
- **Simplicidad:** hay que desarrollar únicamente lo que sea absolutamente necesario, y el diseño de la solución tiene que ser lo más simple posible para facilitar el mantenimiento, el soporte y la revisión del producto.
- ***Feedback:*** deben usarse las grandes cantidades de *feedback* que se reciben de varios orígenes para identificar las áreas de mejora y revisar las prácticas realizadas.
- **Coraje:** se considera importante actuar con valentía y no dejarse llevar por el miedo. Por ejemplo, atreverse a aceptar que algún proceso no se hace correctamente y probar algo nuevo o detectar e informar de problemas organizacionales.
- **Respeto:** es básico que los miembros de un equipo se respeten entre sí para poder trabajar juntos, colaborar y permitir que fluya la comunicación.

Kanban

Esta metodología tiene por objetivo conseguir un producto de calidad y reducir al mínimo los cuellos de botella que se puedan producir a lo largo del proyecto. Las reglas básicas que sigue son:

- **Aceptar el cambio:** como ocurre comúnmente en las metodologías ágiles, se quiere motivar a las personas para que vean el cambio como algo positivo y alentarlas a modificar aquello que no funciona bien o que es mejorable.
- **Liderar:** hay que tener iniciativa y gestionar correctamente el equipo o la tarea asignada.
- **Respetar el proceso activo y las responsabilidades**: cada miembro del equipo debe tener claro cuáles son sus funciones en cada momento del proyecto, pero para ello estas deben haberse definido previamente.
- **Kanban no establece cómo hacer una tarea**, sino que ayuda a decidir si se está haciendo de la mejor manera posible o si se puede mejorar en algo.

Selección de una metodología

Para decidir qué metodología se ajusta más a las necesidades del proyecto de desarrollo concreto que debe llevarse a cabo, es necesario tener en cuenta factores como el tipo de empresa, las características del proyecto de *software* concreto o el volumen y las características del equipo que se empleará. Esta decisión debe tomarse específicamente para cada proyecto, y se puede dar el caso de que, dentro de una misma empresa, se utilicen metodologías distintas en función de cada uno.

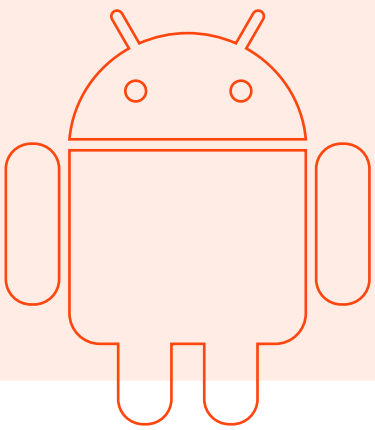
Las características propias del entorno móvil hacen que los requisitos iniciales de las aplicaciones sean cambiantes y que, en muchos casos, resulte prioritario reducir al máximo el *time-to-market* para conseguir un producto funcional lo antes posible. Por dichos motivos, las metodologías ágiles van muy ligadas a los desarrollos de este tipo de aplicaciones, ya que permiten adaptar el ciclo de vida del *software* a este entorno altamente cambiante.

6. Lenguajes y entornos de desarrollo

En función del sistema operativo al que se quiera destinar la aplicación (principalmente, Android o iOS), y del tipo de aplicación a desarrollar (nativa, web, híbrida o generada), se pueden utilizar unos determinados lenguajes de programación u otros, presentados a continuación.

Aplicaciones Android

El lenguaje de programación a usar en el desarrollo de aplicaciones Android es **Java**. Con este lenguaje se implementan las funcionalidades de la aplicación, la gestión del almacenamiento y el acceso a los datos, y la interacción con el *hardware* (como la cámara o el GPS). En cuanto a la interfaz de usuario en las **aplicaciones nativas**, esta se crea con un asistente que ofrece el entorno de desarrollo, y que se encarga de generar el correspondiente **XML** de diseño y configuración. A través del asistente (y/o de este XML) se pueden organizar y personalizar todos los elementos de la aplicación como la posición, el aspecto o la distribución de los botones, formularios e imágenes.



En el caso de las **aplicaciones híbridas**, se utilizan lenguajes web como **HTML** (o **HTML5**), **JavaScript** o **CSS**. La aplicación resultante se ejecuta sobre un componente llamado *webview*, que es como un navegador con motor de JavaScript (compatible con Chrome). Existen multitud de *plugins* que permiten acceder a recursos de bajo nivel en el *hardware* (p. ej. sensores) y siempre existe la opción de implementar un *plugin ad hoc*. Estos tipos de aplicaciones minimizan el *time-to-market*, dado que con un único desarrollo, sobre un alto nivel de abstracción (HTML5), se pueden generar productos para las principales plataformas. Sin embargo, esta abstracción añade el inconveniente de que no puede usarse en aplicaciones que precisen de un alto rendimiento o de un uso exhaustivo del *hardware* (como en el caso de juegos o de realidad aumentada).

El entorno de desarrollo oficial para aplicaciones Android, tanto nativas como híbridas, es **Android Studio**. Este *software* ofrece un editor de código, las librerías (SDK) de Android y Java, un visor de la aplicación en tiempo real y una máquina virtual que permite ejecutar la app para probarla. También se pueden utilizar otros entornos, como, por ejemplo, **Eclipse**, previa instalación de los componentes específicos para poder compilar y ejecutar Android.

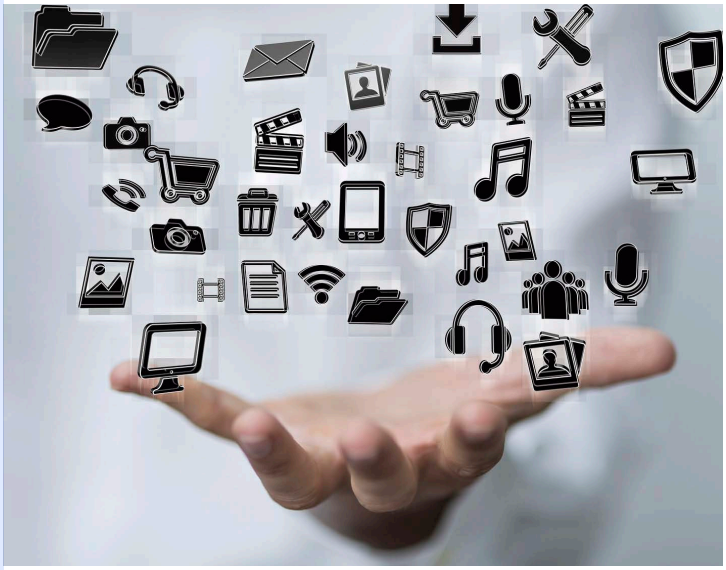
También se pueden desarrollar aplicaciones Android con otros lenguajes, como **Kotlin** o **GoLang** y/o C++, y con Android Native Development Kit (NDK) si se quieren desarrollar funcionalidades de bajo nivel.

Aplicaciones iOS

En el caso del sistema operativo iOS, las *apps* se pueden programar utilizando **Objective-C** o **Swift.** Objective-C es el lenguaje que se utilizó inicialmente en el desarrollo de aplicaciones nativas o híbridas para iOS, pero en el 2014 Apple anunció el lanzamiento de un nuevo lenguaje más fácil, potente e intuitivo: Swift. Sin embargo, en la actualidad sigue siendo posible desarrollar *apps* utilizando solo Objective-C y, de hecho, las principales librerías para iOS están escritas en este lenguaje, incluyendo la API específica Cocoa Touch. Swift también se basa en esta misma API, de forma que resulta fácil migrar una aplicación de Objective-C a Swift, y ambos lenguajes se pueden complementar utilizándose para el desarrollo de una misma *app*.

En las **aplicaciones nativas**, tanto Objective-C como Swift se utilizan para el desarrollo de las funcionalidades, la gestión de datos y el acceso al *hardware,* pero también para implementar la capa de presentación (interfaz de usuario). En las **aplicaciones híbridas** también se utilizan estos lenguajes para implementar las funcionalidades, pero la interfaz de usuario se desarrolla con los lenguajes web citados previamente (HTML/HTML5, JavaScript o CSS). Phone Gap o Apache Cordova son ejemplos de SDKs que permiten crear *webapps* multiplataforma y que generan productos para las distintas plataformas existentes.

El entorno de desarrollo oficial para iOS es XCode (para el sistema operativo MacOS), que ofrece un editor de código con control de cambios, así como herramientas de testeo y simulación.



Aplicaciones generadas

Para desarrollar aplicaciones generadas se utilizan entornos específicos como **Xamarin Studio** o **Genexus,** que permiten realizar implementaciones multiplataforma. En el desarrollo se usa un lenguaje único en función de la herramienta utilizada (p. ej. **C#, .NET** o JavaScript), que será compilado y transformado automáticamente al lenguaje nativo deseado. Como resultado de este proceso, se obtendrán los ejecutables específicos para cada sistema operativo soportado (como iOS o Android).

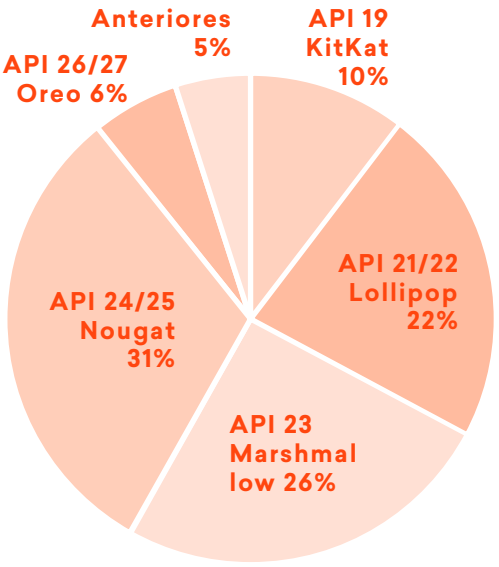
7. Consejos para el desarrollo y la publicación de aplicaciones

En este apartado se presentan una serie de recomendaciones a tener en cuenta desde la fase de planificación del proyecto hasta la publicación de la aplicación, pasando por el desarrollo de la misma.

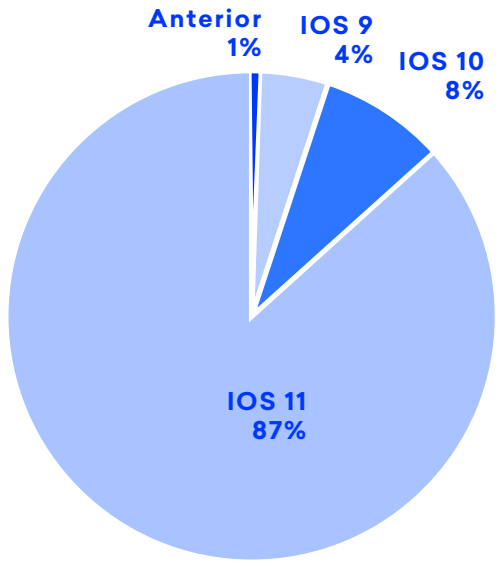
Planificación y diseño

Para lograr una buena planificación, se recomienda **tener presente el diseño** de la interfaz de usuario desde el principio del proyecto, ya que dejarlo para más adelante puede suponer tener que adaptar las funcionalidades a *posteriori*. Por ello, es necesario que exista una sincronización constante entre el equipo de desarrolladores y el de diseño, y que trabajen con una visión conjunta desde el inicio.

Otro factor importante a considerar antes de empezar a desarrollar la aplicación es **escoger la versión (API) en la que se realizará la programación.** Optar por la última versión puede suponer descartar a un gran número de usuarios que utilizan versiones inferiores por cuestiones de incompatibilidad. Pero, por otro lado, escoger una versión muy inferior no permitirá utilizar las funcionalidades más nuevas, lo que puede provocar que la app quede desfasada. En el caso de Android, habrá que estudiar qué versión utilizar para llegar al mayor número de usuarios posible, siempre que sea una versión suficientemente nueva para disfrutar de las últimas funcionalidades. En iOS basta con programar dos versiones anteriores a la última, ya que, como se ve en el gráfico a continuación, se incluirá prácticamente a todo el mercado:



Distribución de las versiones de Android en junio de 2018.



Distribución de las versiones de iOS en junio de 2018.

Desarrollo

Es recomendable utilizar las **guías oficiales de desarrolladores de Android y de iOS** para descubrir las funcionalidades que presenta cada sistema operativo y resolver las dudas de programación que puedan surgir durante el desarrollo. Además, también puede resultar útil unirse a una **comunidad de desarrolladores**, como, por ejemplo, **Stack Overflow**, donde se podrán compartir dudas con otros programadores y consultar las que hayan sido resueltas con anterioridad.

Además, se recomienda que se incorporen los siguientes aspectos al modelo de desarrollo:

- Utilización de un sistema de control de versiones (p. ej. basados en GIT).
- Generación de productos de forma automatizada y en un solo paso utilizando sistemas de integración continua.
- Usar una base de datos de incidencias o *bugs*.
- Realizar el *testing* (incluido el de usabilidad), lo que implica contar con personas formadas en este ámbito.
- Utilizar herramientas de análisis de la calidad del *software*, como Android Lint o SonarQube, para identificar posibles problemas o vulnerabilidades.

Publicación

Una aplicación puede contener **errores (*bugs*) que pueden pasar desapercibidos** porque solo se den en ciertos dispositivos o en versiones específicas de los mismos. Como es muy difícil probar la aplicación en todos los dispositivos y versiones del mercado, se recomienda lanzar una publicación inicial a modo de piloto, en una **región delimitada.** Por ejemplo, si el objetivo es publicar una aplicación a nivel internacional, es aconsejable que durante unos meses se publique solo en el propio país, lo que permite asegurar que la aplicación no contiene errores. De este modo, se evitará que la aplicación reciba valoraciones negativas que podrían perjudicar su posicionamiento en los *markets* y que pierda usuarios. Una vez corregidos todos los errores, se procederá a publicar la *app* en una región de mayor tamaño para que los usuarios finales puedan valorar la aplicación sin *bugs*.

Por otro lado, también es importante **garantizar que las traducciones de la aplicación son correctas** en todas las lenguas, ya que una mala traducción podría proporcionar información incorrecta de la misma y generar comentarios negativos que perjudicarían su posicionamiento.

Cómo dar visibilidad a la *app*

A continuación se proporcionan una serie de consejos para que la aplicación ocupe una posición destacada en los *markets* y obtenga el mayor número posible de descargas:

1. **El icono** de la aplicación es lo primero que los usuarios ven. Tiene que destacar, ser **agradable** y estar **relacionado** con la búsqueda.
2. **Las imágenes** que se ven en el *market* deben **mostrar la principal funcionalidad** de la aplicación, ser comprensibles en el idioma del usuario y **estar ordenadas** por prioridad.
3. Se puede aprovechar el **vídeo para mostrar las funcionalidades clave** de la aplicación, y así acabar de convencer al usuario de que se satisfarán sus necesidades.
4. Es importante **escoger bien las palabras clave** (keywords) de la aplicación para aparecer en las primeras posiciones de las búsquedas.
   a. **Investigar a la competencia** y asegurar que la descripción **destaca funcionalidades de la app que la diferencian** del resto.
   b. El App Store permite incluir palabras clave que pueden ayudar a encontrar la aplicación. Es importante **escoger las más adecuadas.**
5. Si se publica la aplicación en varias regiones, es necesario garantizar que está **bien traducida a los idiomas principales** de cada localización.
6. **Las reseñas y valoraciones son importantes.** Si incluyen palabras clave de la aplicación y son positivas, harán subir el posicionamiento de la aplicación en las búsquedas del *market*.
7. También se pueden utilizar **herramientas de posicionamiento** que ayudarán a ganar descargas:
   a. **Google Play Devoloper Console** es una herramienta gratuita que proporciona Google y que permite definir y optimizar de forma fácil e intuitiva todos los elementos de la ficha de la app, así como ver sus estadísticas.
   b. **App Annie** tiene características gratuitas y de pago. Permite seguir los rankings de aplicaciones por país y calcular los ingresos que obtendrá la *app*.
   c. **The Tool** es una herramienta de pago que permite monitorizar rankings de *keywords*, consultar las valoraciones de los usuarios y gestionar la publicidad.
   d. **Sensor Tower** es una herramienta de pago que ayuda a mejorar el posicionamiento de la aplicación en las búsquedas, gracias al estudio de las *keywords*.

8. Seguridad y protección de datos

La seguridad en el mundo del *software* a menudo pasa erróneamente a un segundo plano, priorizando el número de funcionalidades o la facilidad de uso. La seguridad de las *apps* está relacionada con la seguridad de los datos que estas gestionan y, por lo tanto, es fundamental que se haga todo lo posible por garantizar tanto su seguridad como la privacidad de los datos. También es importante entender que la seguridad no se ha de tener en cuenta solo durante el desarrollo del *software*, sino que representa un proceso continuo de mejora que se tendrá que llevar a cabo durante toda la vida de la *app*.

A continuación se destacan aspectos relevantes a tener en cuenta para garantizar que la *app* a desarrollar es segura:

- Identificar los **datos sensibles** que se tratarán y determinar el **tratamiento** que se hará de los mismos, así como aplicar, para cada tipo, las **garantías de seguridad adecuadas.**

- Valorar los aspectos de seguridad desde las **primeras fases del proyecto**. Se recomienda contar con un **experto en seguridad**, ya sea interno o externo, que valore las necesidades de seguridad del proyecto.

- Utilizar sistemas alternativos de autenticación de usuarios como **Oauth** para facilitar el acceso con credenciales de otros servicios como Google o Facebook. En caso de utilizar credenciales propias, es obligatorio que estas cumplan unos **requisitos mínimos de longitud y complejidad**, y es imprescindible guardar estos **datos encriptados** en el servidor.
  También se puede valorar la posibilidad de validación utilizando elementos biométricos como las huellas digitales.

- Si la aplicación se comunica con sistemas externos, como, por ejemplo, servicios web, es sumamente importante que toda la información enviada esté cifrada con protocolos de transporte seguros como **TLS**. También es conveniente que todas las comunicaciones seguras se realicen con servidores que estén identificados por **certificados digitales válidos.**

- Si se almacenan datos sensibles en el dispositivo, hay que garantizar que estos estén **encriptados.**

- La aplicación debe solicitar sobre el terminal **solo aquellos permisos que sean imprescindibles.** De este modo, en caso de que la aplicación se vea comprometida, se limitará el acceso al dispositivo. Además, algunos usuarios, conscientes de los riesgos de seguridad existentes, no estarán dispuestos a conceder determinados permisos a la aplicación si no los consideran realmente necesarios.

- Es necesario **verificar** todas las **vías de entrada de información** y validar que solo se pueda introducir aquello que se espera, incluido el tipo de datos introducidos y la longitud de los distintos campos. En aplicaciones cliente-servidor, cuando se trata de seguridad, debe ponerse especial énfasis en las validaciones en el servidor. Las validaciones en el cliente mejoran la experiencia del usuario, pero son fácilmente eludibles, de forma que hay que replicarlas en el servidor.

- Si la aplicación trabaja con una base de datos es susceptible de sufrir un ataque de **SQL Injection**, que es uno de los ataques más comunes y que puede implicar la pérdida, corrupción o robo de la totalidad de los datos. Este ataque consiste en introducir código SQL en parámetros de entrada de datos, como puede ser un campo donde poner la contraseña en un *login*, de forma que se modifica la consulta realizada en la base de datos permitiendo la realización de las acciones que quiera el atacante. Para evitarlo es importante validar todas las cadenas de texto que se usan en acciones contra la base de datos, lo que impide la introducción de código SQL en las mismas.

- En las fases de test de la *app*, no solo deben realizarse **pruebas de las funcionalidades**, sino también **tests de penetración** (*pentesting*) para verificar que la misma no es vulnerable a las tipologías de ataques conocidos. Para realizar un *pentesting* completo se pueden seguir distintas guías o *frameworks*, como, por ejemplo, el realizado por la **OWASP** (Open Web Application Security Project), organización mundial sin ánimo de lucro centrada en mejorar la seguridad del *software*.

- Verificar que se cumple con todo lo establecido en las leyes aplicables en materia de protección de datos. En el ámbito catalán resultan de aplicación la ley española LOPD (Ley Orgánica de Protección de Datos) y el reglamento europeo **RGPD** (Reglamento General de Protección de Datos), que se pueden consultar en la web de la **AEPD** (Agencia Española de Protección de Datos).
  Estas leyes no solo regulan aspectos técnicos relativos a la protección de datos, sino que también incluyen cuestiones relacionadas con la garantía de los derechos de los usuarios o con los consentimientos que deben solicitarse para realizar el tratamiento de datos personales.

Reglamento General de Protección de Datos

Este reglamento es de aplicación para todas las empresas que traten datos de ciudadanos europeos y regula la seguridad de los datos personales. A continuación se presentan sus aspectos principales a modo de resumen.

Datos personales

El RGPD define el dato personal como cualquier información relacionada con una persona identificable, que puede ser identificada directa o indirectamente por referencia de un identificador. Ejemplos de datos personales se consideran el nombre, la dirección, la localización GPS, la información sanitaria o los ingresos monetarios de una persona.

Dentro de los datos personales, el RGPD distingue aquellos que se consideran sensibles y, por lo tanto, requieren las máximas medidas de seguridad.
Este tipo de datos incluyen los relativos a:

- Opiniones políticas.
- Afiliación sindical.
- Convicciones religiosas.
- Convicciones filosóficas.
- Origen racial o étnico.
- Datos relativos a la salud.
- Vida sexual.
- Datos genéticos.
- Datos biométricos.
- Orientación sexual.

El RGPD, en términos generales, prohíbe el tratamiento de estos datos excepto en determinados casos, como cuando se obtiene el consentimiento explícito para su tratamiento.

Procesamiento de datos

El procesamiento de datos (o **tratamiento**) cubre un amplio rango de operaciones sobre datos personales, ya sean manuales o automáticas, incluidas la **recogida, grabación, organización, estructuración, almacenamiento, adaptación o alteración, recuperación, consulta, uso, transmisión o divulgación**. Algunos ejemplos considerados como procesamiento de datos son: enviar correos electrónicos promocionales, colgar fotos de una persona en una web, almacenar IPs o MACs de los usuarios o consultar una base de datos de contactos.

Derechos del sujeto de los datos

Cuando se realiza el procesamiento de los datos de un sujeto, deben garantizarse los siguientes aspectos clave:

- **Consentimiento claro y explícito**: se necesita un consentimiento explícito para tratar datos personales. El formulario de consentimiento no puede ser ambiguo y tiene que usar un lenguaje sencillo y comprensible. Es necesario, por ejemplo, añadir un *check* para consentir cada uno de los tratamientos que se realizarán de los datos y no sirve incluir un «Acepto los términos y las condiciones de uso» genérico. Los consentimientos previos al reglamento que no cumplan este punto **deben volverse a solicitar.** En el caso de menores de 16 años, el consentimiento lo tiene que proporcionar un tutor legal -esta edad se reduce hasta los 13 años en algunos estados-.

- **Derecho de acceso:** el sujeto tendrá acceso a todos sus datos en un formato amigable y comprensible, junto con la información adicional detallada en el artículo 15 del RGPD (como los fines del procesamiento).

- **Derecho de rectificación:** el sujeto podrá modificar todos los datos de su perfil, incluidos los que se han obtenido a través de otras fuentes.

- **Derecho al olvido:** se garantizará que se eliminan físicamente todos los datos de una persona si esta así lo pide, y cesará su transmisión a terceros sin demora injustificada. Este derecho se aplicará siempre que no comprometa otros como la libertad de expresión.

- **Derecho a la restricción del procesamiento de datos:** en este caso se eliminan todos los datos del sujeto de forma lógica, haciendo que no sean visibles, de ninguna manera, desde la aplicación.

- **Derecho a la portabilidad de datos:** es necesario poder exportar todos los datos en un formato estructurado.

- **Notificación de las actividades de procesamiento a las autoridades de protección de datos locales** de cada país o región.

- **Minimización de datos:** se trata de solicitar solo aquellos datos que son estrictamente necesarios para el tratamiento concreto que se hace de los mismos.

- **Mantenimiento de los datos solo el tiempo necesario:** cuando ya no sean necesarios los datos para el fin para el cual han sido recogidos, habrá que eliminarlos, aunque se tendrá en cuenta si hubiera que mantenerlos por otros motivos legales.

- **Consentimiento independiente para cada tipo de tratamiento:** cada tipo de tratamiento que se realice de los datos, o si se decide posteriormente tratar los datos para una nueva finalidad, requiere de un consentimiento específico.

- **Notificación a los usuarios en caso de que sus datos sean transferidos fuera de la Unión Europea.**

- **Notificación de intrusiones:** deben notificarse a los afectados y a la autoridad supervisora las intrusiones que puedan afectar a los derechos o libertades de los individuos. La notificación se realizará lo antes posible, preferiblemente antes de transcurridas **72 horas.**

Medidas relativas a los datos personales

El reglamento establece que para los datos personales hay que aplicar medidas apropiadas, teniendo en cuenta la tecnología, los costes de implementación, la naturaleza, el alcance, el contexto y el objetivo del procesamiento, así como los riesgos y la severidad de los mismos en relación con los derechos y las libertades de las personas.

En cuanto a los datos personales sensibles, deben aplicarse una serie de medidas que van más allá y que se resumen a continuación:

- Disponer de un registro actualizado de todos los tratamientos que se realicen en la entidad, documentándolos y aportando un conjunto de información específica.

- Si se realiza un tratamiento de datos personales de las categorías especiales (datos sensibles) a gran escala, se llevará a cabo también una evaluación del impacto. Esta evaluación describirá los procesamientos realizados y su proporcionalidad respecto a los objetivos. Asimismo, detallará todos los riesgos respecto a los derechos y las libertades de los sujetos relacionados con dichos tratamientos, así como las medidas tomadas para minimizar estos riesgos.

- También se tomarán medidas de seguridad pertinentes, como:
  - Disponer de un listado de personas autorizadas para el tratamiento de los datos.
  - Disponer de un registro de accesos que incluya la fecha y el nombre de la persona que ha accedido.
  - Cifrar los datos en las comunicaciones y en el almacenamiento.
  - Utilizar la seudonimización de los datos si estos se usan, por ejemplo, en entornos de test o con fines de investigación.
  - No incluir datos personales en los ficheros de *log*.
  - Si la aplicación interacciona con terceros para compartir datos, hay que garantizar que estos también cumplen la normativa.

9. Accesibilidad

La accesibilidad en el ámbito digital consiste en velar porque toda la información disponible, tanto en la red como en las aplicaciones, así como el propio uso de los dispositivos tecnológicos, estén al alcance de todas las personas, independientemente de sus condiciones, características o capacidades. Este acceso ha de poderse realizar utilizando o no sistemas de soporte, como los lectores de pantalla **VoiceOver** para iOS o **Talk-Back** para Android. Los sistemas operativos como iOS o Android actualmente ya ofrecen muchas opciones de accesibilidad para mejorar la experiencia del usuario, y es importante garantizar que la *app* a desarrollar sea compatible con ellos.

A menudo, la adaptación de los programas, webs y *apps* para que sean accesibles se considera un proceso caro y complejo. Esta complejidad, y el coste derivado, quedan extremadamente minimizados o completamente anulados si, desde el primer momento, se tienen en cuenta las directrices básicas de accesibilidad. De este modo, el elemento principal a tener en cuenta en esta materia es comprender que la accesibilidad debe contemplarse desde el momento de elaboración de los *wireframes* y en el análisis funcional de la *app*.

El organismo internacional encargado de promover la accesibilidad en internet es el W3C (World Wide Web Consortium), a través de su grupo de trabajo **WAI** (Web Accessibility Initiative). Esta organización publica una serie de estándares de accesibilidad a tener en cuenta que quedan recogidos en las guías WCAG (**Web Content Accessibility Guidelines**). Hay herramientas que permiten comprobar el nivel de desempeño de estos estándares, así como testar apps con diferentes tipos de filtros para dar respuesta a todas las necesidades de visualización. También existen otras para testar la calidad de la accesibilidad de las aplicaciones y que ofrecen propuestas de cambio. A la hora de desarrollar una *app*, es importante considerar los complementos que están específicamente diseñados para ayudar a utilizar las aplicaciones, tales como lectores de pantalla, herramientas de dictado, o aquellos que permiten aumentar el tamaño del texto.

En octubre de 2017, la Fundació TIC Salut i Social colaboró en la edición de la guía ***Accesibilidad digital: las TIC para todo el mundo***, elaborada por la Taula d'Entitats del Tercer Sector Social de Catalunya a través de su proyecto **m4Social**. Esta guía recoge buenas prácticas y recomendaciones para elaborar webs, *apps* y contenidos accesibles, así como un listado de referencias a herramientas y complementos existentes.

10. Usabilidad

Se define «usabilidad» como la capacidad de una persona para utilizar cualquier objeto o herramienta artificial con un objetivo concreto. En este ámbito, el objeto o herramienta es una *app*, que tendrá que seguir unos criterios determinados para que el usuario la pueda utilizar de forma cómoda e intuitiva. En el presente apartado se destacan una serie de recomendaciones y consideraciones dirigidas a garantizar que la app a desarrollar sea usable, y relativas a la navegación, la legibilidad y la coherencia, el contenido y la previsión de la misma.

Navegación

La aplicación ha de proporcionar al usuario una forma **intuitiva, rápida y sencilla** de interacción, cumpliendo, siempre que sea posible, con la **regla de los tres clics.** Con esta aproximación, se garantiza que el acceso a toda la información se pueda realizar en un máximo de tres pasos. También es importante **reducir el proceso de aprendizaje** del usuario al mínimo, y ofrecerle **recursos de ayuda** que lo guíen paso a paso en la utilización de las funcionalidades de la *app*, como un tutorial, por ejemplo. Determinar si será posible visualizar la interfaz tanto en horizontal como en vertical implicará unos requerimientos específicos de desarrollo, por lo que será necesario tenerlo en cuenta en el diseño de la misma.

En cuanto al área de clic, no se incorporarán botones inferiores a 9 mm de ancho y alto (el equivalente a unos 44px), ya que serían demasiado pequeños en relación con las proporciones de los dedos de los usuarios. Es igualmente importante respetar la separación entre los botones para evitar interacciones no deseadas, como presionar dos a la vez, así como asegurar que cada elemento tiene la apariencia de aquello que representa. De este modo, todo botón, enlace o zona a clicar será **identificable** por el usuario, ya sea por la forma que espera que tenga o por un color que sea coherente con el código establecido dentro del universo de la plataforma.

Legibilidad y coherencia

El **diseño** de la interfaz tiene que ser lo más **sencillo y práctico** posible, sin elementos que no aporten nada, lo que no está reñido con presentar un diseño atractivo. Es más fácil para el usuario recordar e interactuar con una estructura sencilla, que con una compleja que presente muchas opciones e información que puede generar estrés, ruido en la pantalla e, incluso, una toma de decisiones errónea. No se puede perder de vista que todas las acciones y elementos interactivos tienen que ser fáciles de encontrar.

El uso del color es un factor importante para ayudar a jerarquizar la información y ordenar el contenido dentro de las aplicaciones. Es recomendable usar la **regla de 60-30-10**, en la que el 60 % corresponde al color principal (p. ej. un color de fondo que ayude a la lectura), el 30 % es para un color secundario que ayude a complementar y dar contraste a la parte visual, y el 10 % final se reserva a los colores de acento (enlaces, botones, elementos importantes, etc.). Los colores han de formar parte de una gama cromática que proporcione un alto contraste y defina claramente los elementos principales. Se pueden emplear recursos cromáticos que sigan el código de colores universal, como el rojo para el error o la cancelación y el verde para denotar correctitud o confirmación.

Resulta igualmente importante mantener una legibilidad alta para que los mensajes se puedan transmitir fácilmente y de forma clara. Es recomendable usar las tipografías del propio sistema operativo, ya que estas están especialmente diseñadas a tal efecto. Además, es aconsejable usar entre **uno y tres tipos de tipografía** con un **máximo de tres tamaños** o estilos diferentes, así como aplicar **diferentes estilos y jerarquías** para ayudar a ordenar la información (como el uso de títulos, subtítulos o campos de texto).

Si hay que usar iconografía para representar las funciones habituales de una app, lo más recomendable es no hacer un diseño nuevo, sino **imitar el diseño estándar** que usan para lo mismo otras aplicaciones que el usuario seguramente ya conoce. Estos iconos deberían tener una misma coherencia gráfica y funcional, con el objetivo de que el usuario los pueda identificar y relacionar fácilmente.

En cuanto al diseño, es especialmente relevante **tener en cuenta el sistema operativo** para el cual se está desarrollando la aplicación, ya que algunos criterios de usabilidad se han adaptado específicamente para ello. En este sentido, si el sistema operativo dispone de botones para navegar, por ejemplo, conviene que la app sea compatible con ellos. Por ejemplo, en Android siempre ha de funcionar el botón de «Atrás» del sistema operativo, aunque ya se tenga un botón específico para llevar a cabo esta función dentro de la aplicación.

Contenido

El contenido de una *app* se puede estructurar de diferentes maneras. En función de cual sea su objetivo final, la interacción con el usuario puede ser:

- **Jerárquica:** existe una pantalla principal (o índice) desde donde se accede al resto de páginas, las cuales también pueden enlazar con otras (a modo de subpáginas), creando una jerarquía. Este tipo de organización requiere de un menú que permita la navegación.
- **Lineal:** desde una pantalla se puede acceder a la siguiente y a la anterior, como si fueran las páginas de un libro. No es recomendable si el número de páginas es muy elevado, pero podría servir, por ejemplo, para un tutorial.
- **Lineal con jerarquía:** es una estructura híbrida, en la que la organización es jerárquica, pero que permite navegar de manera lineal por páginas que están en el mismo nivel.
- **Red:** no existe orden aparente. No es una estructura recomendada cuando hay un gran volumen de páginas, ya que desorienta al usuario.

En cualquiera de estos casos, el contenido debe incluir **descripciones concisas, precisas y claras,** y evitar la redundancia. Este criterio es especialmente importante para la visualización de contenidos en dispositivos móviles, debido al espacio limitado que tienen en comparación con, por ejemplo, los ordenadores de sobremesa.

Previsión

Es recomendable generar las especificaciones que detallen todas las funcionalidades que incluirá la *app* para satisfacer las necesidades del usuario, desde el punto de vista funcional (y no solo técnico). Estas especificaciones funcionales deben incorporar los siguientes elementos:

- Detallar las situaciones críticas que puedan implicar las funcionalidades de la *app*.
- Describir las funciones que se quieren implementar.
- Evitar detalles técnicos, excepto si es estrictamente necesario.
- Incluir descripciones concisas, precisas y claras para evitar la redundancia.

Para minimizar los posibles errores de usuario, es imprescindible procurar detectarlos con anterioridad y ofrecer un sistema de prevención para evitarlos. Un ejemplo es la herramienta de autocompletado del buscador Google, que ofrece una propuesta de búsqueda corregida. También hay que tener en cuenta que el usuario puede cometer errores propios, y no de sistema, en cuyo caso este tiene que poder encontrar siempre de forma rápida la opción que deshace la acción que acaba de realizar.

También es importante que una app pueda ser usada por todo tipo de usuarios. Si esta contiene funciones avanzadas, no se debería obligar al usuario inexperto a usarlas. Siguiendo con el ejemplo anterior, el buscador Google incorpora operadores para filtrar mejor las búsquedas, pero no es necesario que el usuario los conozca para utilizarlo.

La aplicación no debería necesitar documentación para que el usuario sea capaz de utilizarla, ya que uno de los conceptos básicos de usabilidad es que esta sea intuitiva. No obstante, es necesario ofrecer ayuda a través de la web de la aplicación o de la aplicación misma, mediante una sección de preguntas frecuentes o de iconos de interrogación junto a ciertas funciones, por lo que es recomendable disponer de documentación de las funcionalidades de la *app* y de cómo se utilizan.

11. Experiencia de usuario / UX

La experiencia de usuario es el conjunto de elementos relacionados con el modo en el que este interactúa con la aplicación y que hacen que tenga una percepción positiva o negativa de la misma. No es un factor que dependa solo de la usabilidad, la accesibilidad o el diseño visual, sino que también incorpora aspectos relativos a las emociones, las preferencias o la sensación de fiabilidad que tiene del producto.

A continuación se destacan aspectos que pueden influir en la percepción final que el usuario tendrá de la aplicación:

Tener en cuenta el primer uso: debe definirse la información que se mostrará cuando el usuario todavía no haya introducido datos. Normalmente no es buena idea dejar las partes de las pantallas simplemente «vacías», sin información previa o indicaciones relacionadas. Algunos ejemplos serían la visualización de la pestaña de «Preferidos» cuando el usuario todavía no ha definido ninguno, la apariencia de la cesta de la compra cuando está vacía o la ausencia de resultados de búsqueda. Estos casos se pueden solucionar con mensajes informativos del tipo: «La lista de Preferidos está vacía» o «No se han añadido productos a la cesta», pero resulta más útil aprovechar también el espacio para enseñar al usuario cómo llenarlo.

Incorporar mensajes de error: deben preverse el máximo de errores que se puedan producir en la aplicación y no mostrar descripciones incomprensibles para el usuario. En este sentido, es necesario identificar aquellos elementos que puedan desencadenar dichos errores, como los campos de informado obligatorio, las restricciones de tipos de datos aceptados o el tamaño máximo de los ficheros; e implementar sistemas de prevención de los mismos. Caso que se produzca un error en la aplicación, habrá que informar de ello al usuario, indicando cómo lo puede solucionar. El usuario ha de saber en todo momento cuál es el estado del sistema para evitar que se sienta confundido o perdido.

Dar la opción de deshacer cambios: tanto si se elimina como si se modifica cualquier elemento, siempre debe tenerse en cuenta que el usuario se puede equivocar y/o puede querer deshacer los cambios. Por ello, es una buena opción incorporar el botón de «Deshacer» y/o una papelera adonde vayan a parar los elementos eliminados y desde la que se puedan restaurar. Informar al usuario de que tiene la opción de deshacer las acciones que lleva a cabo permite que se muestre más confiado a la hora de utilizar la app y que no tenga miedo de explorarla.

Incorporar mensajes de confirmación: antes de llevar a cabo acciones permanentes, es mejor abrir un mensaje para confirmar que el usuario quiere hacer lo que realmente está indicando. A menudo, se muestra un simple botón de «Aceptar» o «Guardar» del que no se conocen las consecuencias que comportará. De este modo, recordar al usuario que la acción que está realizando es permanente es una buena opción para que se sienta más cómodo con el uso de la *app*.

Diseñar botones y menús familiares: se trata de diseñar los botones y menús de forma que sean intuitivos y conocidos por el usuario. Por ejemplo, utilizar los mismos iconos que usan las aplicaciones más conocidas o las del mismo ámbito es una buena opción. Del mismo modo, usar una interfaz gráfica parecida a la de la mayoría de aplicaciones o seguir un mismo patrón dentro de la aplicación permitirán que el usuario utilice la herramienta más fácil e intuitivamente.

Gestionar los tiempos de espera: la inacción es uno de los elementos que más molesta a los usuarios, ya que no les resulta agradable tener que esperar para hacer lo que han indicado. Hay que notificar de alguna manera que la aplicación está realizando acciones que pueden tardar cierto tiempo (para evitar que el usuario llegue a la conclusión de que la app se ha colgado, por ejemplo). Lo más habitual es utilizar un elemento que muestre el progreso en forma de barra, reloj o ruedita de carga, pero también se pueden amenizar estos ratos perdidos con contenidos más atractivos como bromas, consejos de utilización o curiosidades de la *app*.

12. Estándares de comunicación y representación de información

En el entorno mHealth y mSocial coexisten aplicaciones que permiten recoger distintos tipos de información, de forma que los datos generados por los ciudadanos quedan repartidos en un conjunto de *apps* sin que se puedan intercambiar entre sí. Cada aplicación utiliza su formato interno de almacenamiento y representación, así como interfaces propias de comunicación con otros sistemas (normalmente, limitado a plataformas propiedad de la misma compañía). En este escenario, un ciudadano puede estar utilizando dos aplicaciones que permiten recoger el mismo dato (como el peso), sin que los valores de una y otra se puedan intercambiar, comparar ni explotar globalmente. Para superar estas limitaciones, es necesario ponerse de acuerdo en cómo se tiene que producir el intercambio. Los estándares representan estos acuerdos, que se producen también a nivel internacional, y que permiten lograr la interoperabilidad entre sistemas, dispositivos, aplicaciones, etc.

En el sector de la salud en general, y en el SISCAT (Sistema Sanitari Integral d'Utilització Pública de Catalunya) en particular, se lleva años trabajando en la interoperabilidad de sistemas, dispositivos y servicios, en respuesta a la necesidad de compartir información entre centros, incluso de diferentes niveles asistenciales. El logro de este intercambio implica un trabajo exhaustivo de estandarización que, en el caso del SISCAT, ha sido liderado desde la **OFSTI** (Oficina d'Estàndards i Interoperabilitat) de la Fundació TIC Salut Social, y que se está reproduciendo en otros ámbitos como el social o el de la movilidad.

Definición del concepto de interoperabilidad

La interoperabilidad es la capacidad de compartir información entre componentes (como sistemas o dispositivos) sin que se pierda su significado. Esta comunicación debe garantizar el intercambio coherente de los datos entre departamentos, organizaciones, niveles asistenciales o regiones, como países o continentes, cuyo objetivo principal es proporcionar a los profesionales toda la información relevante de sus pacientes para asegurar que el proceso de toma de decisiones se produce de una forma segura, eficiente y eficaz. La interoperabilidad garantiza el acceso a la información independientemente del lugar en el que se haya registrado, lo que favorece su reaprovechamiento, minimiza puntos ciegos y asegura la continuidad asistencial.

Esta capacidad no es binaria (en términos de lograrse o no), sino que tiene diferentes capas, niveles o dimensiones, las cuales se presentan a continuación:

- **Interoperabilidad técnica:** en este nivel se encuentran las tecnologías y los protocolos que permiten establecer comunicación entre los componentes.
- **Interoperabilidad funcional o sintáctica:** añade intercambio de información a la comunicación. Los estándares de este nivel definen la estructura y el formato de la información a intercambiar.
- **Interoperabilidad semántica:** es el nivel de interoperabilidad que tiene por objetivo garantizar que la información que se intercambia no pierde su significado y se puede utilizar en el componente receptor, como si se hubiera generado ahí.
- **Interoperabilidad legal:** permite garantizar que se cumple la legislación vigente en cada agente implicado en el intercambio. Esta capa es especialmente importante en proyectos transfronterizos donde el intercambio se produce entre países o regiones con marcos legislativos diferentes.
- **Interoperabilidad organizacional:** añade la capa de proceso, de modo que el intercambio y uso de la información esté alineado con los flujos de trabajo de las instituciones involucradas.

Cada una de estas dimensiones se puede garantizar con el uso de estándares explícitamente diseñados a tal efecto. En este apartado se hará hincapié en las capas sintáctica y semántica de interoperabilidad, presentando los principales estándares existentes para su garantía en el entorno móvil.

Interoperabilidad sintáctica y el estándar FHIR

FHIR (Fast Healthcare Interoperability Resources) es un estándar de intercambio de información desarrollado y publicado por la organización internacional **HL7**. Actualmente se encuentra en estado de borrador, concretamente en la fase STU3 (Standard for Trial Use), una versión casi definitiva y que ya cuenta con múltiples experiencias de implementación.

HL7® FHIR®

El estándar FHIR supone un cambio de concepto respecto a los estándares anteriores de HL7, como CDA R2 (Clinical Document Architecture Release 2) o la mensajería 2.x, ya que se ha diseñado específicamente para facilitar a los desarrolladores su aprendizaje, implementación, adaptación y tratamiento. Las piezas a intercambiar (o recursos) también se han simplificado, de forma que permiten enviar solo la información estrictamente necesaria en un formato más ágil y adecuado también para el entorno móvil. Estas características han permitido que FHIR se convierta en el estándar de referencia para el intercambio de información en el entorno mHealth.

FHIR utiliza la arquitectura **REST** (Representational State Transfer) para realizar el intercambio de datos, con estándares estructurales como **XML** o **JSON**. Su implementación permite tener un servidor donde almacenar los datos y una serie de servicios estándar para enviarlos y recuperarlos. Para facilitar la implementación de FHIR y el desarrollo de aplicaciones que lo adopten, **se han creado varias APIs**, cuyas versiones más conocidas son **HAPI** (para el lenguaje Java) y **Firely** (para .Net). Estas APIs también permiten la creación de un servidor basado en FHIR, preparado específicamente para el almacenamiento de recursos.

La seguridad y el control de acceso a los datos generados son esenciales para que la aplicación **cumpla con las normativas correspondientes** y preste un servicio seguro y fiable a los usuarios. Las especificaciones **SMART** on **FHIR** permiten cubrir estas necesidades, habilitando una integración segura con los EHR (Electronical Health Records), portales, etc., y utiliza estándares como OAuth2 para los permisos o OpenID para el inicio de sesión.

Como se ha mencionado anteriormente, los componentes básicos del estándar FHIR son recursos y representan la unidad básica de información que se puede compartir. Cada uno de estos componentes permite representar una entidad concreta (como Paciente, Problema de salud u Observación) y tiene asociadas una serie de variables. A continuación se presenta una parte de la definición del recurso Paciente, a modo de ejemplo:

Structure

Name	Flags	Card.	Type
Patient			DomainResource
identifier	Σ	0..*	Identifier
active	?! Σ	0..1	boolean
name	Σ	0..*	HumanName
telecom	Σ	0..*	ContactPoint
gender	Σ	0..1	code
birthDate	Σ	0..1	date
deceased[x]	?! Σ	0..1	
deceasedBoolean			boolean
deceasedDateTime			dateTime
address	Σ	0..*	Address
maritalStatus		0..1	CodeableConcept
multipleBirth[x]		0..1	
multipleBirthBoolean			boolean
multipleBirthInteger			integer

Para cada una de las variables que forman el recurso, se especifica su cardinalidad (el número mínimo y máximo de veces que puede aparecer dentro del recurso) y su tipo. FHIR contiene muchos tipos diferentes de variables, que pueden ir desde un número entero hasta información codificada utilizando vocabularios controlados como SNOMED CT. El listado de recursos disponibles, así como su definición, se pueden encontrar en el siguiente enlace: Resource Index.

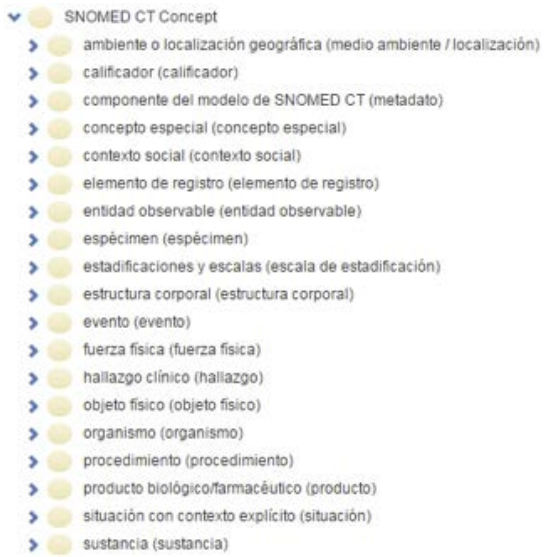
Interoperabilidad semántica y el estándar SNOMED CT

En cuanto a la capa semántica de interoperabilidad, los estándares que permiten lograrla tienen por objetivo que la información intercambiada no pierda su significado, de forma que la normalizan mediante vocabularios controlados. Esta estandarización permite representar los conceptos para que se puedan intercambiar, comparar y explotar, aunque procedan de distintas fuentes.

Existen varios vocabularios controlados para normalizar información, y es importante saber identificar cuál es su propósito para utilizarlos correctamente. Las **clasificaciones**, como CIM-10-MC/SCP (diagnósticos y procedimientos), CIAP-2 (atención primaria), ATC (principios activos) o NANDA (enfermería), se han diseñado para agrupar y explotar la información, de forma que se desaconseja su uso para el registro y la representación amigables y minuciosos. En cambio, las **terminologías**, como LOINC (laboratorio) o SNOMED CT, están específicamente diseñadas para registrar la información con el máximo nivel de detalle y garantizar así la interoperabilidad. En el ámbito de la interoperabilidad semántica, es habitual (y necesario) realizar mapeos (equivalencias) entre vocabularios para poder combinar sus ventajas y utilizarlos cada uno para su propósito concreto.

La terminología clínica SNOMED CT (Systematized Nomenclature of Medicine Clinical Terms) es un estándar semántico **internacional y multilingüe** que permite normalizar conceptos principalmente del ámbito de la salud, pero que ya se está empezando a aplicar a otros ámbitos, como el social o el mHealth. El hecho de que el modelo conceptual de este vocabulario esté diseñado para ofrecer una visión transversal del ciudadano, hace que contenga conceptos que no son estrictamente clínicos. De este modo, SNOMED CT se puede utilizar como una terminología «puente» o «de enlace» entre los diferentes tipos de profesionales.

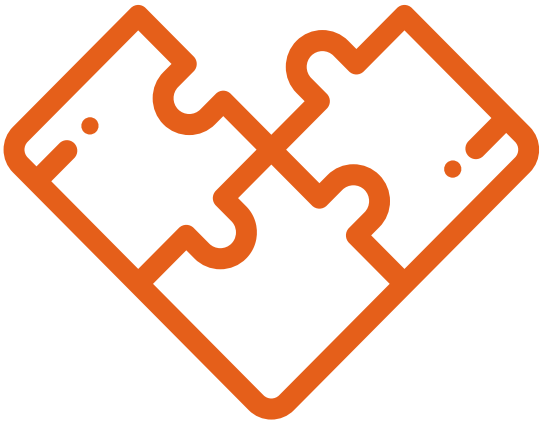
SNOMED International (IHTSDO) es la entidad sin ánimo de lucro que tiene la propiedad de SNOMED CT, así como la responsabilidad de su distribución y actualización a nivel mundial. En el SISCAT, es la OFSTI la que tiene la responsabilidad de su distribución y adaptación a las necesidades del territorio. Esta terminología fue creada en 1965 por el CAP (Colegio Americano de Patólogos) y actualmente ya cuenta con casi medio millón de conceptos de diferentes especialidades. En concreto, SNOMED CT presenta los siguientes 19 ejes de alto nivel o jerarquías:



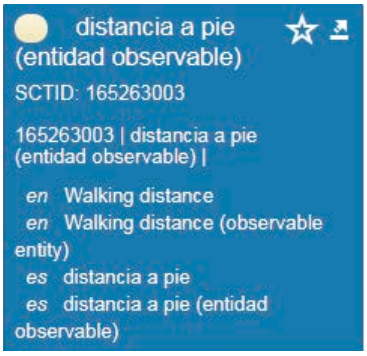
Las ideas en SNOMED CT se representan a través de **conceptos** organizados jerárquicamente, que pueden tener diferentes **descripciones** (con sinónimos) y que están conectados entre sí a través de **relaciones**. Este vocabulario está disponible en inglés y en castellano, y desde la **OFSTI** ya se está trabajando con el Departamento de Planificación Lingüística del Departamento de Salud de la Generalitat de Catalunya y el Termcat (Centre de Terminologia) para irlo traduciendo al catalán (actualmente ya se han traducido cerca de 2.400 descripciones).

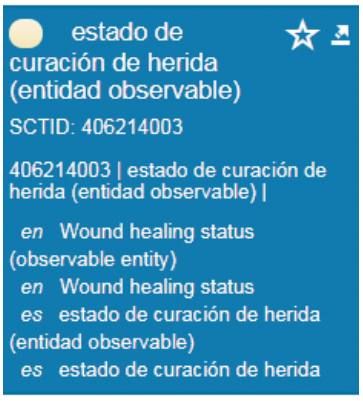
SNOMED CT contempla diferentes mecanismos para facilitar su adopción, entre los que destacan los **subconjuntos** y las **extensiones.** Los primeros están destinados a agrupar componentes (conceptos, descripciones o relaciones) con un propósito determinado y, por lo tanto, permiten utilizar solo aquellos elementos que realmente son necesarios. Las extensiones, en cambio, se pueden ver como versiones locales de la terminología y aportan la flexibilidad necesaria para dar respuesta a las necesidades reales. Dentro de una extensión se pueden crear conceptos, descripciones, subconjuntos, etc., siguiendo las directrices de SNOMED International y sin dañar la interoperabilidad de los contenidos. Desde la OFSTI se mantiene la extensión catalana de SNOMED CT, con casi 3.000 conceptos que no se encuentran en el vocabulario, y que se distribuye de una a dos veces al año según la demanda.

Para facilitar el uso de SNOMED CT en el entorno móvil, se ha creado un subconjunto de variables de movilidad mHealth y mSocial, que cuenta con más de 50 conceptos procedentes de seis *apps*.



A continuación se presentan algunos conceptos que forman parte del subconjunto, a modo de ejemplo:





Desde la **Oficina mhealth**.cat y la **OFSTI** se presta ayuda para normalizar las variables de las aplicaciones móviles y lograr así que estas sean interoperables a nivel internacional.

13. *Medical Device:* la aplicación móvil como producto sanitario

El uso cada vez mayor de las aplicaciones de salud ha acrecentado el debate sobre cuándo una *app* debe considerarse un producto sanitario. Identificar en la fase de diseño si la aplicación móvil que se está desarrollando es o no un producto sanitario es fundamental, dado que, de serlo, deberá cumplir la legislación aplicable, cuyo objetivo principal es garantizar la seguridad del paciente.

A nivel legislativo, el producto sanitario está actualmente regulado por la Directiva 93/42/-CEE, transpuesta por el Real Decreto 1591/2009. También hay que tener en cuenta que el producto sanitario para diagnóstico *in vitro* está actualmente regulado por la Directiva 98/79/CE, transpuesta por el Real Decreto 1662/2000 y el Real Decreto 1193/2012.

En este apartado se facilitan una serie de puntos clave para ayudar a identificar si una *app* es o no un *medical device* (o producto sanitario).

Definición de producto sanitario

La Directiva 93/42/CEE define «producto sanitario» como:

«Cualquier instrumento, dispositivo, equipo, *software* informático, material u otro artículo, utilizado solo o en combinación, incluidos los programas informáticos destinados por su fabricante a finalidades específicas de diagnóstico y/o terapia y que intervengan en su buen funcionamiento, destinado por el fabricante a ser utilizado en seres humanos con fines de:

- diagnóstico, prevención, control, tratamiento o alivio de una enfermedad;
- diagnóstico, control, tratamiento, alivio o compensación de una lesión o de una deficiencia;
- investigación, sustitución o modificación de la anatomía o de un proceso fisiológico;
- regulación de la concepción; y que no ejerza la acción principal que se desee obtener en el interior o en la superficie del cuerpo humano por medios farmacológicos, inmunológicos ni metabólicos, pero a cuya función puedan contribuir tales medios.»

Además, la Directiva 98/79/CE define «producto sanitario destinado al diagnóstico *in vitro*» como:

«Cualquier producto sanitario consistente en un reactivo, producto reactivo, calibrador, material de control, estuche de material y materiales, instrumento, aparato, equipo o sistema, utilizado solo o en combinación, destinado por el fabricante a utilizarse in vitro para el estudio de muestras procedentes del cuerpo humano, incluidas las donaciones de sangre y tejidos, solo o principalmente con el fin de proporcionar información:

- relativa a un estado fisiológico o patológico, o
- relativa a una anomalía congénita, o
- para determinar la seguridad y compatibilidad con receptores potenciales, o
- para supervisar las medidas terapéuticas.»

La Unión Europea publicó el 25 de mayo de 2017 los reglamentos europeos EU 2017/745 para producto sanitario y EU 2017/746 para producto sanitario destinado al diagnóstico *in vitro*, que, una vez transcurrido el período establecido de adaptación de tres y cinco años respectivamente, serán aplicables a todos los países de la Unión.

Cómo identificar si una app es un producto sanitario

A continuación se presentan los criterios de decisión y los requisitos reguladores relativos a los productos sanitarios.

Criterios de decisión

La aplicación móvil en salud que se considera producto sanitario a menudo se denomina «app médica» (*mobile medical app*), mientras que el resto se consideran «apps de bienestar» (*wellness apps*). La finalidad de la aplicación (uso previsto o *intended use*) es esencial para determinar si es o no producto sanitario: en caso de que sea **diagnosticar, dar apoyo al diagnóstico o a las decisiones clínicas, hacer cálculos para determinar el diagnóstico o el tratamiento, o utilizarse para cualquier propósito médico**, la aplicación podría ser considerada un producto sanitario. Por dicho motivo, es necesario disponer de una detallada y completa definición de la finalidad prevista de la aplicación, lo que permitirá concluir si es o no producto sanitario.

Las aplicaciones que son producto sanitario han de cumplir la definición de producto sanitario (según las directivas 93/42/CEE y 98/79/CE o los nuevos reglamentos EU 2017/745 y EU 2017/746). Sin embargo, pueden actuar como accesorio de un producto sanitario, o bien convertir directamente la plataforma móvil en un producto sanitario. Por el contrario, en caso de que la aplicación solo **almacene, archive, transmita, realice búsquedas simples de datos o presente resultados de datos sin alterarlos ni manipularlos**, no será considerada un producto sanitario.

A continuación se listan algunas preguntas que pueden ayudar a decidir, junto con los criterios expuestos anteriormente, si la aplicación es o no producto sanitario:

- ¿La aplicación está destinada a interpretar (o facilitar la interpretación) de datos modificando o representando información individual relacionada con la salud?
- ¿La aplicación interpreta o altera datos? ¿O tan solo gestiona información estática?
- ¿La aplicación calcula dosis a tomar/inyectar?
- ¿La aplicación indica condición médica, enfermedad o porcentaje individual de riesgo de sufrir una enfermedad?

En cualquier caso, una misma aplicación podrá ser o no producto sanitario en función de la finalidad prevista. Por ejemplo: una aplicación para monitorizar el ritmo cardíaco, en el supuesto de que esté indicada exclusivamente para uso deportivo, no será un producto sanitario; en cambio, la misma aplicación destinada a un uso médico sí que podrá ser un producto sanitario.

La guía MEDDEV 2.1/6 (*Qualification and Classification of stand-alone software*, de julio de 2016) muestra un diagrama de decisión que aporta orientación sobre los pasos necesarios a seguir para decidir si un *stand-alone software* es o no producto sanitario. Tal y como indica la misma guía, en este contexto se entiende por stand-alone aquel software que no está incorporado a un producto sanitario.
La guía *Guidance on Medical Device stand-alone software including apps (including IVDMDs (In vitro Diagnostic Medical Devices))*, publicada por la MHRA, ofrece información detallada para ayudar a decidir si una *app* es o no producto sanitario. Esta misma guía también clarifica cuándo se considera que una aplicación tiene un propósito médico, lo que la convierte directamente en producto sanitario.

Requisitos reguladores para productos sanitarios

Una vez se ha identificado que la aplicación es un producto sanitario, deberá determinarse **a qué clase pertenece** de entre las indicadas en el anexo IX de la Directiva 93/42/CEE. Esta clasificación se basa en el riesgo potencial que el producto representa para el paciente y/o usuario, y determinará los pasos a seguir para evaluar su conformidad. Los productos sanitarios se pueden clasificar en:

- Clase I ⟶ riesgo bajo.
- Clase IIa ⟶ riesgo moderado.
- Clase IIb ⟶ riesgo severo.
- Clase III ⟶ riesgo alto.

Cabe destacar que para el producto sanitario destinado al diagnóstico in vitro no se aplica esta clasificación, sino que el anexo II de la Directiva 93/42/CE define un listado de productos que requieren de una ruta de evaluación de la conformidad específica. El Reglamento EU 2017/746 indicado anteriormente introduce cambios en este punto, estableciendo una completa clasificación también para los productos sanitarios para el diagnóstico *in vitro*.

Cualquier producto sanitario deberá disponer de la **declaración de conformidad CE** antes de su comercialización, lo que implica que el fabricante (o representante autorizado en caso de que el fabricante sea externo a la Unión Europea) asegura y declara que el producto cumple la legislación vigente correspondiente. Una vez asegurada dicha conformidad, es necesario colocar la marca CE en el producto sanitario.

A grandes rasgos, para poder comercializar en Europa una *app* considerada como producto sanitario, el fabricante deberá:

- Desarrollar un expediente técnico que cubra todos los requisitos de la directiva pertinente.
- Disponer de un sistema de gestión de la calidad: de conformidad con el artículo 100 de la Ley 14/1986 General de Sanidad, de 25 abril, las personas físicas o jurídicas que se dediquen a la fabricación, importación, agrupación o esterilización de productos sanitarios, y las instalaciones en que se lleven a cabo dichas actividades, requieren de licencia previa de funcionamiento, otorgada por la AEMPS (Agencia Española de Medicamentos y Productos Sanitarios).

Los aspectos clave a tener en cuenta cuando se habla de una aplicación móvil como producto sanitario son:

- Estándares de documentación técnica.
- Estándares de evaluación/evidencia clínica.
- Posible necesidad de intervención, en función de la clasificación del producto, de un organismo (notificado) que declara la conformidad del producto adicionalmente a la declaración del fabricante.
- Estándar en el etiquetado del producto.
- Alto nivel de trazabilidad del producto tanto durante el desarrollo como durante las diferentes etapas de fabricación (ciclo de vida del producto).
- Control riguroso de los cambios sobre el producto (versiones del *software*).
- Tratamiento de incidencias y/o quejas.
- Vigilancia y seguimiento poscomercialización.

Además, también deberá tenerse en cuenta que los nuevos reglamentos citados introducen requerimientos adicionales, como la mejora de la transparencia mediante la creación de la EUDAMED (European Database on Medical Devices). Esta base de datos está actualmente en desarrollo y en ella se registrarán todos los productos sanitarios del ámbito europeo.

14. Entidades participantes

A continuación se presenta la relación de entidades que han elaborado esta guía:

- **CSV Experts:**
  La empresa CSV Experts ha desarrollado los apartados correspondientes a la regulación de productos sanitarios en entornos móviles. En concreto, ha desarrollado el siguiente punto:

13. «Medical device»: la aplicación móvil como producto sanitario

- **Fundació TIC Salut Social:**
  La Fundació ha coordinado la edición del contenido con los diferentes colaboradores, ha proporcionado una primera versión de cada apartado a modo de borrador y ha revisado el texto final.
  Desde la Fundació también se han desarrollado los siguientes puntos:

11. Experiencia de usuario
12. Estándares de comunicación y representación de información

- **iSalus:**
  La empresa iSalus ha realizado toda la parte correspondiente al desarrollo de aplicaciones móviles y seguridad de los datos. En concreto, ha desarrollado los siguientes puntos:

3. Tipos de aplicaciones
5. Proceso de desarrollo de aplicaciones móviles
8. Seguridad y protección de datos

- **M4Social, Taula del Tercer Sector:**
  La empresa m4Social, en colaboración con la Taula d'Entitats del Tercer Sector de la Generalitat de Catalunya, han elaborado la parte correspondiente al desarrollo de apps accesibles. En concreto, el punto siguiente:

9. Accesibilidad

- **PASIONA:**
  La empresa tecnológica Pasiona ha realizado la parte correspondiente a usabilidad en entornos móviles en salud. En concreto, el punto siguiente:

10. Usabilidad

- **UPC:**
  La Universitat Politècnica de Catalunya ha participado en la elaboración del apartado sobre lenguajes y entornos de desarrollo de aplicaciones móviles y ha aportado su visión de los distintos mercados actuales.
  En concreto, ha desarrollado los siguientes puntos

4. «Markets» de aplicaciones
7. Consejos para el desarrollo y la publicación de aplicaciones
6. Lenguajes y entornos de desarrollo

15. Autores

A continuación se presenta la relación de autores que han participado en la elaboración de esta guía:

- **CSV Experts:**
  - Josep Hortigüela, consultor sénior y experto en productos sanitarios.

- **Fundació TIC Salut Social:**
  - Carme Pratdepàdua, responsable de la Oficina mhealth.cat.
  - Ariadna Rius, responsable de la Oficina d'Estàndards i Interoperabilitat.
  - Miquel Martí, técnico en interoperabilidad de la Oficina d'Estàndards i Interoperabilitat.
  - Pau Garcia, técnico en interoperabilidad de la Oficina d'Estàndards i Interoperabilitat.

- **iSalus:**
  - Marc Gòrriz, responsable de proyectos y desarrollos.
  - Dylan Rivera, Full-Stack Developer.

- **M4Social, Taula d'Entitats del Tercer Sector:**
  - Jordi Serratosa, coordinador m4social de la Taula d'Entitats del Tercer Sector.

- **PASIONA:**
  - Clara Planet, responsable de UX/UI soluciones de diseño.

- **UPC:**
  - Toni Oller, profesor colaborador en la Universitat Politècnica de Catalunya (Departamento de Ingeniería Telemática).
  - Jesus Alcober, profesor titular en la Universitat Politècnica de Catalunya (Departamento de Ingeniería Telemática).

TIC Salut Social
Tecnologia, innovació i transformació digital

Generalitat de Catalunya