**UT1. Análisis de tecnologías para aplicaciones en dispositivos móviles.**

1. **Identifica las necesidades de cada una de las cinco tecnologías de desarrollo de aplicaciones para dispositivos móviles (Aplicaciones Web Móviles, Aplicaciones Web Progresivas, Aplicaciones Móviles Nativas, Aplicaciones Móviles Híbridas Multiplataforma basadas en tecnologías Web y Aplicaciones Móviles Multiplataforma basadas en un Lenguaje Común) en cuanto a: complejidad, tiempo de desarrollo, sistemas operativos soportados, herramientas de desarrollo utilizadas, lenguajes de programación y seguridad.**

* **Aplicaciones Web Móviles:**

Una aplicación Web es un sitio web optimizado correctamente para su visualización y ejecución en un dispositivo móvil.

Al ser un sitio web, es accesible desde cualquier sistema operativo que disponga de un navegador web. Esto hace que es desarrollo sea más ágil ya que no hay que enfocarse en el desarrollo y optimización para cada uno de los sistemas operativos que lo van a consumir.

Al tratarse de un sitio web, no es necesario que el programador conozca el lenguaje de programación de cada uno de los sistemas (Swift, Java, Xamarin, etc.), ya que con conocimientos de desarrollo web (JavaScript, HTML y CSS), y una adaptación para dispositivos móviles a través de diseños responsive, se puede usar desde cualquier dispositivo. Esto hace que el desarrollo de esta aplicación sea el más reducido de todos.

Así mismo, el editor para el desarrollo puede ser el elegido por el usuario para un desarrollo web normal, sin ser necesario instalar ningún IDE propio de cada sistema operativo.

En cuanto a seguridad, este contenido no tiene acceso a nuestro dispositivo, y los permisos a los que puede acceder son los propios que tiene el navegador web. De esta forma el usuario sólo debe preocuparse de los datos introducidos dentro de la aplicación web, y que serán tratados por ésta.

* **Aplicaciones Web Progresivas:**

Las aplicaciones Web Progresivas son una versión de una aplicación web, que se pueden consultar a través del navegador de internet, aunque también es posible instalarlas en el mismo equipo (aunque no suelen estar disponibles a través de las tiendas oficiales).

Esto permite a la aplicación realizar tareas que generalmente sólo las aplicaciones nativas pueden llevar a cabo (notificaciones, funcionamiento sin conexión a internet, etc.).

El desarrollo de estas aplicaciones añade un Service Worker (separado de la página web), un fichero Javascript que contiene una secuencia de comandos que el sistema ejecuta en segundo plano.

Al tratarse de una versión extendida de una aplicación web, el desarrollo de ésta sólo se alarga un poco más en el tiempo que éstas, debido al desarrollo del Service Worker. Para ello, los lenguajes de programación utilizados siguen siendo JS, HTML y CSS. Y los editores de código quedan a elección del usuario, sin tener que usar ningún IDE propio de cada sistema.

En cuanto a la seguridad, aunque se ejecutan scripts en segundo plano, el protocolo de comunicación con la aplicación es HTTPS para mantener seguros los datos del usuario.

* **Aplicaciones Móviles Nativas:**

Una aplicación web nativa es una aplicación que se desarrolla específicamente para un sistema operativo completo.

Esto hace necesario que el programador conozca el lenguaje y para el sistema operativo que se va a desarrollar, y es aconsejable usar el IDE que facilita éste sistema para agilizar el desarrollo.

Al tener que desarrollar la aplicación varias veces (una para cada sistema operativo), se requiere de más tiempo para finalizar su desarrollo, pero la optimización de ésta es mejor.

Por tanto, el lenguaje de programación y el editor en cada desarrollo será el propio del sistema operativo a utilizar (Android Studio - Java, Visual Studio - .Net, SwiftStudio - Swift, etc).

Dado que son aplicaciones nativas se pueden encontrar en la propia tienda de aplicaciones del sistema operativo, y se instalan en el propio dispositivo. Tienen acceso total a los recursos del terminal. Por este motivo, es necesario revisar los permisos que requiere cada aplicación.

* **Aplicaciones Móviles Híbridas:**

Las aplicaciones móviles híbridas son aplicaciones que están desarrolladas en lenguajes como HTLM5, CSS3 y JavaScript. Y son Works como PhoneGap o Ionic, que ejecutaran la aplicación para trasladar ese código HTML5, CSS3 y JavaScript y añadiendo funcionalidades a la aplicación, dándole acceso a los elementos del dispositivo (cámaras, sensores, etc.).

Esto implica que lo que se hace es ejecutar una especie máquina virtual sobre la que se ejecuta el código de nuestra aplicación. Por este motivo hay casos en los que estos Works están limitados y no pueden acceder a todos las funcionalidades hardware del dispositivo.

El lenguaje para desarrollar estas aplicaciones que debemos conocer son tanto HTML5, CSS3 y JavaScript, como el lenguaje reservado del Work elegido. Para el desarrollo de estas aplicaciones suele bastar con un simple editor de texto, como Visual Studio Code o Sublime Text.

Por lo tanto, el desarrollo de estas aplicaciones es mayor que el de una aplicación web pero inferior a una aplicación nativa, ya que un mismo desarrollo puede exportarse a varios sistemas operativos.

La seguridad de estas aplicaciones es inferior al resto de aplicaciones, ya que aúna las vulnerabilidades de una página web con las de una aplicación con acceso a nuestro dispositivo y sensores, cámaras, etc.

* **Aplicaciones Multiplataforma:**

Las aplicaciones Multiplataforma son aplicaciones que están desarrolladas en un único lenguaje, pero son capaces de exportarse a varios sistemas operativos.

El IDE utilizado serán el que requiera el lenguaje utilizado para este fin, dejando así de lado la necesidad de conocer los lenguajes de programación propios de cada sistema, y el uso de los IDEs reservados. Y agilizando enormemente el desarrollo de la aplicación con respecto a desarrollarla por separado para cada sistema.

Aunque al haberse desarrollado de una forma genérica para todas las plataformas no es tan óptima como una aplicación nativa, ofrece una sensación muy cercana a esta, ya que cada experiencia de usuario está optimizada para cada uno de los sistemas a través del compilador.

Esto hace que el tiempo de desarrollo sea mucho menor que desarrollando para cada sistema operativo individualmente.

En cuanto a seguridad, ésta es homónima a la seguridad de una aplicación nativa, y será el propio usuario quien deberá gestionar los permisos a los que da acceso a la aplicación y a cuáles no.

**UT2. Programación de aplicaciones para dispositivos móviles.**

1. **Realiza un análisis de los diferentes servicios disponibles en los dispositivos móviles y su utilidad en el desarrollo de las aplicaciones considerando la plataforma Xamarin.**

Un servicio es una funcionalidad desempeñada por un sistema y que nosotros, como desarrolladores, podemos explotar para nuestros desarrollos.

La forma que Xamarin tiene de acceder a estos servicios es a través de Xamarin.Essentials, el cuál brinda a los desarrolladores una API multiplataforma (Android, iOS y UWP) que funciona con cualquier aplicación Xamarin.Forms.

Listados de servicios disponibles desde Xamarin.Essentials:

* **Acelerómetro**: recupera los datos de aceleración del dispositivo en un espacio tridimensional.
* **Información de la App**: conozca información sobre la aplicación.
* **Tema de la aplicación**: detecta el tema actual solicitado para la aplicación.
* **Barómetro**: supervise los cambios de presión con el barómetro.
* **Batería**: detecte fácilmente el nivel, origen y estado de la batería.
* **Portapapeles**: establezca o lea fácil y rápidamente texto en el Portapapeles.
* **Conversor de Colores**: métodos del asistente para System.Drawing.Color.
* **Brújula**: supervise los cambios en la brújula.
* **Conectividad**: compruebe el estado de la conectividad y detecte cambios.
* **Detector de movimiento**: detecte movimientos de agitación en el dispositivo.
* **Información de orientación**: obtenga la orientación y las métricas de la pantalla del dispositivo.
* **Información del dispositivo**: conozca información sobre el dispositivo de manera sencilla.
* **Email**: envíe fácilmente mensajes de correo electrónico.
* **Almacenamiento**: guarde fácilmente archivos en los datos de la aplicación.
* **Linterna**: una manera sencilla de encender y apagar la linterna.
* **Geocodificación**: coordenadas y direcciones de código geográfico y de código geográfico inverso.
* **Geolocalización**: recupere la ubicación de GPS del dispositivo.
* **Giroscopio**: haga seguimiento de la rotación alrededor de tres ejes primarios del dispositivo.
* **Launcher**: permite que una aplicación abra un URI por el sistema.
* **Magnetómetro**: detecte la orientación del dispositivo respectivo del campo magnético de la Tierra.
* **MainThread**: ejecute código en el subproceso principal de la aplicación.
* **Maps**: abra la aplicación de mapas en una ubicación específica.
* **Navegador**: abra rápidamente y sin problemas un explorador en un sitio web específico.
* **Orientazión**: Recupera la orientación del dispositivo en un espacio tridimensional.
* **Permisos**: compruebe y solicite permisos a los usuarios.
* **Llamadas**: abra el marcador telefónico.
* **Platform Extensions**: métodos del asistente para convertir Rect, Size y Point.
* **Preferencias**: Agregue rápida y sencillamente las preferencias persistentes.
* **Almacenamiento Seguro**: almacene datos de manera segura.
* **Share**: envíe texto y URI de sitio web a otras aplicaciones.
* **SMS**: Cree mensajes SMS para enviarlo.
* **Text-to-Speech**: Vocalice texto en el dispositivo.
* **Conversor de Unidades**: Métodos del asistente para convertir unidades.
* **Versión Tracking**: Permite hacer seguimiento de las versiones de las aplicaciones y los números de compilación.
* **Vibración**: Permite que el dispositivo vibre.
* **Autenticador web**: Inicie los flujos de autenticación web y escuche una devolución de llamada.

**UT3. Utilización de librerías multimedia integradas.**

1. **Explica las distintas posibilidades de trabajar con datos multimedia desde la plataforma Xamarin.Forms.**

**Imágenes**

El uso de imágenes planas o gifs es el uso más básico de archivos multimedia.

Las imágenes son una parte fundamental de la navegación de las aplicaciones, la facilidad de uso y la personalización.

La utilización de éstas consiste en elegir el origen del archivo, y especificarlo para mostrarlo en la aplicación.

Se deberá tener en cuenta que cada plataforma accederá a estas imágenes de una forma diferente.

**Transformaciones**

Con la ayuda de elementos propios de Xamarin, se podrá transformar las imágenes para obtener un redimensionado y posicionamiento personalizado. Estos cambios en la ubicación, el tamaño o la orientación se realizan a través de la propiedad transformación de Xamarin. Dentro de estas transformaciones podemos encontrar tres tipos básicos:

* Traslación: Desplazamiento de un elemento horizontal o verticalmente, o ambos.
* Escalado: Redimensión de una imagen o elemento.
* Rotación: Permite girar un elemento en función a un eje determinado.

**Animaciones**

Las animaciones son recursos que requieren de cualquier cambio visual dinámico, y que se utilizan para generar transaccionalidad de un estado a otro (una pantalla de carga, una animación para que aparezca una lista, una animación después de pulsar un botón, etc).

Una animación típica cambia progresivamente el valor de una propiedad a otro durante un periodo de tiempo, y esto genera ese cambio en la apariencia que nosotros percibimos como animación.

Los tipos de animaciones son:

* **Animaciones simples**: Son aquellas en las que se cambian valores a través de propiedades predefinidas, y que generan una animación simple:
* **TranslateTo**: Traslada un elemento a otra posición
* **ScaleTo**: Cambia la escala general de un elemento.
* **RotateTo**: Rota el elemento.
* **FadeTo**: Cambia la opacidad del elemento, haciéndolo aparece o desaparecer
* **Animaciones asíncronas**: Estas animaciones se ejecutan en segundo plano, y se llama al método encargado de realizar la animación de una manera más rápida que a través de una animación simple.
* **Animaciones compuestas:** Estas animaciones mezclan animaciones simples y asíncronas para crear una animación doble. Por ejemplo, se puede llamar a dos métodos a la vez, creando una animación doble (un botón que cambia su tamaño a la vez que desaparece).
* **Animaciones de entrada:** Son animaciones que se muestran cuando una página se hace visible por primera vez. Por ejemplo, cuando entramos por primera vez a una aplicación, o a un apartado de la aplicación
* **Animaciones infinitas:** Estas animaciones son animaciones que se prolongan en el tiempo de manera que no tienen un fin, y se encuentran en un bucle. Aunque su uso es poco común, hay sistemas de animación que las utilizan.

**UT4. Análisis de motores de juegos.**

1. **Identifica los distintos elementos que componen la arquitectura de un juego 2D y 3D.**

**Características de los juegos 2D**: Estos utilizan gráficos planos, llamados sprites, y no tienen geometría tridimensional. Esto hace que la realización de ellos sea menos compleja. Se dibujan en la pantalla como imágenes planas, y la cámara no tiene perspectiva.

**Características de los juegos 3D**: Los juegos 3D utilizan geometría tridimensional, con texturas y materiales renderizados en la superficie de GameObjects para que aparezcan como los entornos sólidos, personajes y objetos que conforman el mundo del juego.

Los juegos 3D sueles renderizar la escena con perspectiva, por lo que los objetos se ven más grandes en la pantalla cuando se acercan a la cámara.

La elección del tipo de juego que querremos desarrollar (2D o 3D) se nos presentará desde el momento en que abrimos el editor de Unity, y se podrá intercambiar entre los dos modos en cualquier momento.

Al elegir entre uno de los dos modos en Unity, se establecen algunos ajustes como por ejemplo, se determinará si las imágenes se importarán como texturas o sprites, y si la cámara es ortográfica o de perspectiva.

Además, en función de la opción que elijamos también tendremos acceso a unas herramientas u otras:

Herramientas para **el desarrollo 3D:**

* ProBuilder, para la creación de escenarios
* Polybrush para esculpir las mallas

Herramientas para el **desarrollo 2D:**

* Sprite Editor 2D
* Renderizador de máscaras de sprites
* Herramientas de construcción de mundos como editores Tilemap para mosaicos
* Animación basada en huesos
* Posibilidad de crear fácilmente Shaders y luces 2D.

Además de esto, también tenemos acceso al Asset Store, en el cual podremos encontrar plantillas y elementos que podrán ayudarnos en nuestros desarrollos ahorrándonos tiempo a la hora de crear estos elementos.

1. **Analiza detalladamente los componentes de los motores de juegos: Unity, Cryengine y Unreal Engine.**

Todos los motores de juegos citados a continuación tienen componentes fundamentales, como el sistema de renderizado gráfico, el sistema de detección de colisiones, escenarios virtuales o el sistema de audio. Por este motivo, en lugar de hablar de ellos, hablaré sobre el resto de características que diferencia a cada uno.

* **Unity**: Unity es considerado uno de los mejores motores de juegos, ya que ofrece una gran gama de herramientas a las que los desarrolladores pueden acceder fácilmente.

Una de las mejores características que ofrece la plataforma es su integración multiplataforma, que permite exportar el desarrollo a la plataforma Android, Blackberry, iOS, Windows, etc. Mientras que Unity admite hasta 25 plataformas para exportar sus desarrollos, Unreal y CryEngine sólo admiten 10 y 5 respectivamente.

Otra característica es su compatibilidad con diferentes formatos, como 3D Max, Blender, CINEMA, Maya, Softimage, etc.

Los desarrolladores también tienen acceso a la tienda de recursos, la cual tiene más de 15.000 modelos gratuitos y de pago, audio, animaciones, extensiones para el editor, materiales, guiones, sombras, etc.

En el caso en que creásemos un juego con Unity que nos generase un beneficio superior a 100.000$, deberemos comprar la licencia Professional de Unity, la cual puede tener un precio de hasta 150$/año.

Una de las principales ventajas de Unity es su curva de aprendizaje. Debido a lo intuitivo que resulta, los desarrolladores se sienten cómodos con la plataforma.

También cuenta con servicios como el Unity Cloud Build (permite almacenar un proyecto en la nube), Unity Everplay (permite grabar y compartir archivos audiovisuales por parte de los jugadores), Unity Analytics (permite ver la información de los jugadores), Unity Ads (permite insertar publicidad en los videojuegos), etc.

* **Unreal**: La principal característica de Unreal Engine es la calidad de sus gráficos y sus iluminaciones dinámicas. Esto también hace que el desarrollo sea más pesado.

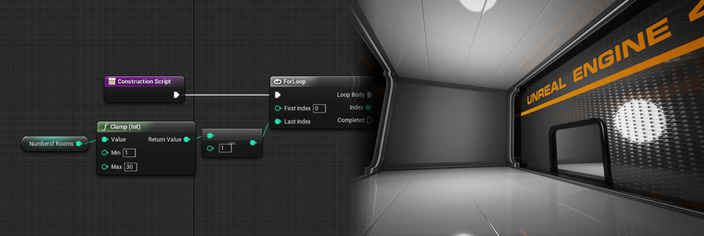
Una de sus características principales es su sistema de partículas, que tiene la capacidad de manejar hasta un millón de partículas en una sola escena.

Aunque Unreal también permite exportar los juegos a diferentes plataformas, tiene menos opciones de exportación que Unity (sólo 18 con respecto a las 25 de Unity).

El principal problema que nos podemos encontrar al empezar a acercarnos a Unreal Engine es la dificultad de la curva de aprendizaje. Unreal está basado en C++ y tanto el aprendizaje del lenguaje como el del editor puede ser una primera barrera para los no instruidos que se acerquen a este motor para aprender a realizar videojuegos.

Otro de sus características es que utiliza la tecnología Blueprint Visual Scripting, que permite generar juegos usando sólo Blueprint (unos diagramas de flujo que no requieren programar sobre un fichero plano). Aunque estos juegos serán funcionales, tendrán ciertas limitaciones.

Ejemplo:



Aunque Unreal Engine es gratuito, se debe pagar una tarifa del 5% sobre el beneficio total del juego (ya sean compras en la aplicación, anuncios en el juego, o propio dinero que cuesta el juego). Esta tarifa sólo se aplicará cuando el juego genere más de 3000$/trimestre.

También nos ofrece una gran calidad y potencia en el apartado de la iluminación global y en el de la creación de shaders y materiales. Con un sistema de nodos que nos puede recordar a otros softwares más especializados en este ámbito y que nos permiten crear un resultado final espectacular.

Epic también ofrece una gran cantidad de documentación y tutoriales para el uso de su motor gráfico y, además, Unreal Engine también cuenta con una comunidad muy activa y muy colaborativa.

* **Cryengine**: Es un motor de juego que llegó a la fama por tratarse del motor de juego de la saga Far Cry y Crysis, los cuales sorprendían por su impresionante potencial medioambietal.

Lo que hace de Cryengine uno de los principales motores gráficos son sus habilidades gráficas, que son muy superiores a las de Unity y se equiparan con las de Unreal.

Uno de sus principales componentes es la compatibilidad con Realidad Virtual, y la cantidad de recursos y usuarios que hay tras este objetivo. Este componente incluye un sistema de representación volumétrica de niebla y nubes, que brinda tanto a la niebla como a las nubes una representación espacial 3D completa, y una visualización realista de los efectos del clima.

Una de las principales desventajas de Cryengine es que los desarrolladores les lleva algo de tiempo el poder usar la plataforma de manera efectiva, y es más complejo comprender que otros motores. Aparte de esto, la documentación es bastante más pobre que la de Unity o Unreal. Debido a este motivo, y a su curva de aprendizaje, se recomienda haber trabajado antes con otros motores de juegos.

Aunque Cryengine no exige una cuantía sobre los beneficios que se generan sobre el juego, sí es necesario pagar una cuantía de 9.90$/mes para obtener acceso a su plataforma. Además de esto, también se tiene acceso a un foro que ayuda a los usuarios a aclarar sus dudas y consultas.

El desarrollo con CryEngine es también más complejo que con Unity, aunque la interfaz es más amigable que la de Unreal. Es especialmente sencilla la creación de escenarios exteriores con terrenos de gran amplitud. La documentación existente es extensa y hay foros oficiales para desarrolladores.