Investiga lenguajes o frameworks para desarrollar aplicaciones nativas y sus características

- **Java**:

Simple: ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos.

Orientado a objetos: trabaja con sus datos como objetos y con interfaces a esos objetos**.**

Robusto: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error.

Seguro: La seguridad en Java tiene dos facetas. En el lenguaje, características como los punteros o el casting implícito que hacen los compiladores de C y C++ se eliminan para prevenir el acceso ilegal a la memoria. Cuando se usa Java para crear un navegador, se combinan las características del lenguaje con protecciones de sentido común aplicadas al propio navegador.

Portable: implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre enteros y además, enteros de 32 bits en complemento a 2. Además, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos Unix, Pc o Mac**.**

Multithreaded: permite muchas actividades simultáneas en un programa. Los threads (a veces llamados, procesos ligeros), son básicamente pequeños procesos o piezas independientes de un gran proceso. Al estar los threads contruidos en el lenguaje, son más fáciles de usar y más robustos que sus homólogos en C o C++.

Dinamico: Java se beneficia todo lo posible de la tecnología orientada a objetos. Java no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Las librería nuevas o actualizadas no paralizarán las aplicaciones actuales (siempre que mantengan el API anterior)

- **Swift**

Seguridad: se basa en la menor probabilidad de cometer errores durante su escritura. Al basarse en un código más limpio, con una estructura de variables menos propensa a incorrecciones y con gestiones automáticas, la existencia de errores o problemas es menor.

Limpia: Un lenguaje de programación sin errores o con menos probabilidades de que aparezcan, tiene como ventaja consecuente que el desarrollo digital basado en este código, también sea más estable. En consecuencia, las apps en Swift son más

seguras que las creadas con otros lenguajes de programación. Incluso ya no es necesario la instrucción public antes de la clase, ya que por defecto todo es público lo que genera un gran ahorro de tipeo.

Rápido:

- **Kotlin**

Importación de diseño estático: permite importar todas las referencias a las vistas desde el diseño con una sola importación

Clases pojo: el poder definir nuestras clases POJO (Plain Old Java Object) usadas para almacenar datos.

La función with(): Se puede usar para guardar algunos tipeos si necesita acceder a muchas propiedades de un objeto.

Investiga lenguajes o frameworks para desarrollar aplicaciones no nativas y sus características

- **Ruby**

Sencillo de aprender para programadores experimentados:es necesario que aprender PHP, no es necesario dominar todo el lenguaje de programación, sólo hace falta un conocimiento básico.

Mejora la productividad: Una aplicación realizada en Ruby on Rails se puede modificar fácilmente. Añadir nuevas características, aplicar cambios a plantillas y otro tipo de modificaciones pueden realizarse de manera rápida de modo que ahorras tiempo valioso. Es decir, Ruby on Rails es una framework flexible que te permite realizar cambios rápidamente.

- **Php**

Orientada a objetos: podréis dividir vuestros scripts en métodos, clases, etc. para hacer más ágil de cara al servidor el procesamiento de los datos.

Estructura separada: Se puede tener de manera independiente del código que se encarga de mover los datos del que se enlaza a la interfaz. Esto se conoce como Modelo Vista/Controlador (MVC). Gracias a esta característica, no “contaminaremos” código con líneas innecesarias y lo mantendremos limpio y ordenado.

Fácil de aprender: se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas, ejemplo que se hace evidente en el uso de php arrays.

- **Perl**

Fácil de usar

Se le considera como un lenguaje que no tiene fronteras

Es rápido

Se puede utilizar en varios entornos

Tiene variedad de características como estructural, funcional y orientado a objetos

Investiga lenguajes o frameworks para desarrollar aplicaciones hibridas y sus características

- **jQuery Mobile**

Selección de elementos DOM.

Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3 y un plugin básico de XPath.

Eventos.

Manipulación de la hoja de estilos CSS.

Efectos y animaciones.

Animaciones personalizadas.

AJAX.

Soporta extensiones.

Utilidades varias como obtener información del navegador, operar con objetos y vectores, funciones para rutinas comunes, etc.

Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+

- **Ionic**

Alto rendimiento La velocidad es importante. Tan importante que sólo se nota cuando no está en tu app. Ionic está construido para ser rápido gracias a la mínima manipulación del DOM, con cero jQuery y con aceleraciones de transiciones por hardware.

AngularJS & Ionic Ionic utiliza AngularJS con el fin de crear un marco más adecuado para desarrollar aplicaciones ricas y robustas. Ionic no sólo se ve bien, sino que su arquitectura central es robusta y seria para el desarrollo de aplicaciones. Trabaja perfectamente con AngularJS.

Centro nativo Ionic se inspira en las SDK de desarrollo móviles nativos más populares, por lo que es fácil de entender para cualquier persona que ha construido una aplicación nativa para iOS o Android. Lo interesante, como sabéis, es que desarrollas una vez, y compilas para varios.

Bonito diseño Limpio, sencillo y funcional. Ionic ha sido diseñado para poder trabajar con todos los dispositivos móviles actuales. Con muchos componentes usados en móviles, tipografía, elementos interactivos, etc.

Un potente CLI Con un sólo comando podrás crear, construir, probar y compilar tus aplicaciones en cualquier plataforma.

- **React Native**

Compatibilidad Cross-Platform: ya que la mayoría de las APIs de React Native lo son de por sí, lo cual ayuda a los propios desarrolladores a crear aplicaciones que puede ser ejecutados tanto en iOS como Android simultáneamente con el mismo código base.

Funcionalidad nativa: las aplicaciones creadas mediante React Native funcionan de la misma manera que una aplicación nativa real creada para cada uno de los sistemas usando su lenguaje nativo propio. La unión de React Native junto con JavaScript permite la ejecución de aplicaciones más complejas de manera suave, mejorando incluso el rendimiento de las apps nativas y sin el uso de un WebView.

Actualizaciones instantáneas (para desarrollo y/o test): con la extensión de JavaScript, los desarrolladores tienen la flexibilidad de subir los cambios contenidos en la actualización directamente al dispositivo del usuario sin tener que pasar por las tiendas de aplicaciones propias de cada sistema y sus tediosos ciclos de procesos obligatorios previos. Hay que aclarar que este uno es exclusivo de versiones de desarrollo o para test, es ilegal, y puede llegar a conllevar castigos que llegan hasta la retirada definitiva de la aplicación si se realizan cambios directos sobre código con aplicaciones ya publicados y en producción. La tienda de Apple lleva un control muy exhaustivo sobre este tipo de prácticas.

Sencilla curva de aprendizaje: React Native es extremadamente fácil de leer y sencillo de aprender ya que se basa en los conceptos fundamentales del lenguaje JavaScript, siendo especialmente intuitivo tanto para los ya expertos en dicho lenguaje o incluso para las personas sin experiencia en él, ya que nos provee de un rango muy amplio de componentes, incluyendo ejemplo como los maps y filters típicos que se han usado siempre.

Experiencia positiva para el desarrollador: si bien la curva de aprendizaje hemos dicho que es sencilla, también el propio lenguaje nos motiva y ayuda a la hora de la evolución según aumentamos nuestro conocimiento y dominio del mismo. Nos ofrece varias características importantes como, por ejemplo, el Hot reloading que nos refresca la app en el momento en que guardamos cambios, y nos ofrece una

gran ventaja para el desarrollo y testing de nuevas versiones, como hemos comentado arriba. O el uso del flexbox layout engine gracias al cual nos permite abstraernos de muchos de los tediosos detalles de la generación de cada uno de los layouts correspondientes a iOS y Android. Así como el uso del debugger de las herramientas de desarrollados del navegador Google Chrome, facilitando de sobre manera la tarea de depuración de código.

Angel Nail Tun