**¿Qué es una aplicación multiplataforma?**

Es el desarrollo de una aplicación que puede ser ejecutada en más de una arquitectura de hardware. Las aplicaciones multiplatafor

**¿Por qué una aplicación multiplataforma?**

**Diferencias entre una aplicacion nativa y una aplicacion multiplataforma**

Today there are differences between native applications and cross-platform applications that lead some companies to choose some solutions and not others. This decision depends, for example, on the features that are to be incorporated into the application and the business goals:

Native applications:

* When a project requires extensive data processing and, therefore, intensive use of the device's memory, the most obvious option is for the members of the team to choose a 100% native development. The higher performance of native applications is a significant aspect. A design and development for Android, another one for iOS and another for Windows Phone, if the idea is finally to have an app in this operating system.
* Scaling: Native applications are supported by companies such as Apple (developer of iOS) and Google (Android). Therefore, each new device or solution launched in the market by the two companies would have the advantage of the approval and all the facilities when scaling the native application to these new scenarios: whether Apple's new smart watch (Apple Watch 2) or Google's new smartphone (Pixel).
* If the choice is a native application, this will have implications in the design and development, which will take the project to specific usability standards for each platform. The native elements of Android and iOS are completely different and this conditions the entire process.
* Native applications make the most of all the possibilities afforded by the hardware. Each device has its standard operating system and, obviously, developing native applications not only has advantages on the operating system side, but also on the side of the hardware itself.
* Native applications usually have greater visibility at app stores and often get better user ratings and recommendations. This is essential for [a good ASO positioning](https://bbvaopen4u.com/es/actualidad/algunos-consejos-seo-de-ultima-generacion-para-principiantes).

Cross-platform applications:

* Compared with the obvious benefits offered by a native application, web project encapsulating solutions or cross-platform applications considerably shorten development and deployment times at app stores. There are therefore fewer costs and the risk of opting for projects that will fail is lower.
* Moreover, we should not forget that cross-platform applications not only cut costs, but they also maximize the profit, either through exposure to a larger number of users (the company will have an application in most environments) or on the revenue side.
* The development team can centralize all the launch effort on a single source code. The fragmentation when implementing an MVP and also in subsequent upgrades does not exist.
* This also has the advantage of not only avoiding fragmentation, but also when it comes to saving error resolution efforts. When the code produces a bug, one single solution can be used for all the environments.
* It is not necessary to incorporate new profiles to the team other than those that can already be serving a desktop version: having a team knowledgeable in HTML, CSS and JavaScript would be enough.

**Herramientas para una aplicación multiplataforma.**

1. Cordova

[Apache Cordova](http://cordova.apache.org/) is first on this list because **it powers the build process for most free cross-platform mobile application development tools, and many that aren't free, for that matter**. Formerly known as PhoneGap, Adobe released Cordova as an open-source project which, as of this writing, has received contributions from BlackBerry, Google, IBM, Intel and Microsoft.

The majority of cross-platform mobile application development tools on this list, both free and paid, use Cordova to package a single codebase into a native executable. Cordova first bundles HTML, CSS and JavaScript into a client-side package. Then, the software executes and renders the custom code within a native WebView — this is known as the “hybrid” application technique. The hybrid approach provides a Write-Once-Run-Anywhere solution (WORA).

1. PhoneGap

PhoneGap was originally the basis for Cordova, but is now a standalone product powered by Cordova—times they are a changin’. The primary benefit Adobe provides with PhoneGap is a GUI build system that abstracts the command line messiness of Apache Cordova.

By providing an interface to build Cordova applications, PhoneGap simplifies the process and gets developers working sooner. PhoneGap also offers an application server you can run from your machine to host your apps over the network. The server accelerates the build→run→revise loop so developers can spend more time coding and less time running, compiling, and deploying manual updates.

1. Ionic

In the previous section, I revealed that PhoneGap is a thin layer above Apache Cordova that improves the hybrid development experience without forcing an application paradigm or structure onto the developer. For a similar yet more opinionated tool that provides an excellent development experience, consider the [Ionic Framework](https://ionicframework.com/).

Ionic combines Angular with its own UI library to provide a cross-platform mobile application development experience that many web developers will find familiar. They find it so familiar in fact, that Ionic has become synonymous with cross-platform hybrid development. More so than PhoneGap, actually. However, Ionic’s mobile build process relies on PhoneGap and therefore also inherits all of Apache Cordova’s plugins.

1. Apache Weex

Yes, this is another one of the Apache-sponsored free cross-platform mobile application development tools. The developers of [Weex](https://weex.apache.org/) combined the idea behind Cordova with a runtime JavaScript interpreter. Their tool generates native experiences from a single source code whereas Cordova executes JavaScript and renders HTML in a WebView. Weex interprets HTML and renders it dynamically using native view elements.

Like Framework7, Weex provides UI components out of the box. But these elements translate to their native counterparts during interpretation, thereby boosting performance considerably. And like Cordova, Weex has a strong community of independent developers that contribute new design and interface plugins to keep the platform current.