**Aplicación nativa**

Se denomina aplicación nativa a las aplicaciones que se desarrollan específicamente para cada sistema operativo: *Android, iOS o Windows Phone.*

Estas aplicaciones deben estar realizadas en un lenguaje u otro dependiendo de para qué sistema operativo se haya realizado. En el caso de *Android*, la aplicación se desarrolla usando el lenguaje *Java,* en el caso de *iOS*, la aplicación se desarrollo usando el lenguaje *Objetive-C* o *Swift* y en el caso de *Windows Phone*, la aplicación se desarrolla usando el lenguaje *.NET*.

**Ventajas**

Las ventajas de realizar aplicaciones nativas son:

* **Máximo rendimiento en el dispositivo.** Al conocer todas las funciones posibles del sistema operativo es posible optimizar el rendimiento para la aplicación
* **Mejor experiencia de usuario.** La optimización de las funcionalidades de la aplicación puede implicar una mejor navegación y un uso más sencillo y agradable para el usuario.
* **Disponibilidad en las App Store.** Una de las normas para que una app pueda estar en su correspondiente app store es que sea nativa. Es cierto que también se permite subir aplicaciones que contienen únicamente una parte nativa y otra web, pero la app store proporciona mayor visibilidad a las aplicaciones nativas.

**Desventajas**

Las desventajas de realizar aplicaciones nativas son:

* **Variedad de funcionalidades.** Dependiendo del sistema operativo del dispositivo, la aplicación contará con unas u otras funcionalidades. Esto puede suponer un problema a la hora de desarrollar la misma aplicación para distintos sistemas operativos, ya que es probable que no compartan todas las funcionalidades.
* **Más costosas.** El precio que se debe pagar para el desarrollo de aplicaciones nativas es mayor que para otros tipos de aplicaciones.
* **Código personalizado y exclusivo.** Todo el código generado para la aplicación no podrá ser reutilizable para otros sistemas operativos.

**Aplicación web**

Las aplicaciones web son la opción más sencilla y económica para crear una aplicación para móvil debido a que solo es necesario crear una sola aplicación.

Estas aplicaciones se basan en la creación de una página web *responsive*, esto es una página web que se adapta al dispositivo donde se muestra.

**Ventajas**

Las ventajas de las aplicaciones web son las siguientes:

* **Baratas.** Realizar aplicaciones web es la alternativa más barata ya que únicamente es necesario desarrollar la aplicación una vez.
* **Menor complejidad.** Para el desarrollador es más fácil de realizar ya que no necesita aprender distintos lenguajes de programación.
* **Multiplataforma.** La aplicación se podrá usar en cualquier dispositivo independientemente del sistema operativo de este.

**Desventajas**

Las desventajas de las aplicaciones web son las siguientes:

* **Peor rendimiento.** Las aplicaciones web tienen un peor rendimiento debido a que no aprovechan el potencial del sistema operativo del dispositivo.
* **Peor experiencia de usuario.** El desconocimiento de las características del dispositivo implica una peor experiencia en el uso de este.
* **Menor seguridad.** La seguridad de la aplicación depende de la seguridad del navegador.

**Aplicación híbrida**

Una aplicación híbrida, como su propio nombre indica, se trata de una aplicación que reúne características tanto de las aplicaciones nativas como de las aplicaciones web.

Este tipo de aplicación permite utilizar los estándares para el desarrollo de las aplicaciones web y aprovecha funcionalidades nativas del dispositivo.

**Ventajas**

Las ventajas de las aplicaciones híbridas son las siguientes:

* **Menor coste**. Las aplicaciones híbridas implican un coste menor que las aplicaciones nativas.
* **Mejor experiencia de usuario**. Las aplicaciones híbridas proporcionan una mejor experiencia de uso al usuario debido a que contiene funcionalidades nativas de los dispositivos donde se usa.
* **Multiplataforma.** Las aplicaciones híbridas se pueden usar en cualquier dispositivo, aprovechando funcionalidades específicas de cada uno de ellos dependiendo de su sistema operativo.

**Desventajas**

Las desventajas de las aplicaciones híbridas son las siguientes:

* **Mayor complejidad.** Para el desarrollador resultan más difíciles de desarrollar debido a las diferentes funcionalidades nativas de cada sistema operativo que tiene que tener en cuenta.
* **Rendimiento ligeramente inferior.** Las aplicaciones híbridas tienen un rendimiento peor que las aplicaciones nativas debido a que cada página debe ser renderizada desde el servidor.

**Alternativas**

A la hora de realizar la aplicación se han estudiado aplicaciones cuya funcionalidad fuese similar y se han encontrado las siguientes alternativas:

**Easy Referee**

Se trata una aplicación con una interfaz muy simple donde aparecen las siguientes opciones:

* Tirar una moneda al aire para sortear el campo al inicio del partido.
* Mostrar una tarjeta amarilla donde se puede anotar un registro de todas estas.
* Mostrar una tarjeta roja donde se puede anotar un registro de todas estas.
* Un registro para anotar cualquier suceso del partido.
* Un silbato para reproducir un sonido similar al de un silbato al pulsar sobre él.
* Un cronómetro para cronometrar el partido.

En general, es una app bastante fácil de usar, algo que es necesario para los usuarios de este tipo de apps.

Por otra parte, se trata de una aplicación con muchas carencias, por ejemplo, todos los registros que se guardan en la aplicación deben ser introducidos mediante el teclado, esto implica una pérdida de tiempo bastante alta para el colegiado.

Otra característica que se echa en falta es la generación de un acta al finalizar el arbitraje de un partido.

En resumen, *Easy Referee* es una aplicación bastante intuitiva pero muy limitada en funcionalidades.

Falta imagen

**Árbitro del fútbol**

*Árbitro del fútbol* es una aplicación que nos ofrece una interfaz gráfica más agradable para el usuario que la anterior aplicación.

Permite realizar el arbitraje de un partido y visualizar los partidos que han sido ya arbitrados.

La funcionalidad que nos permite esta aplicación a la hora de arbitrar un partido es demasiado escasa. Permite mostrar tarjetas amarillas o tarjetas rojas al equipo local o visitante, añadir goles a ambos equipos y muestra la cuenta mediante un cronómetro.

A pesar de que visualmente es una aplicación más lograda que *Easy Referee*, su funcionalidad es insuficiente para cualquier árbitro ya que, por ejemplo, no puedes indicar a qué jugador muestras las tarjetas ni, como en la anterior aplicación, tampoco puedes generar un acta una vez se finaliza el partido.

Falta imagen

**Referee Pro**

Esta aplicación es, sin duda alguna, la mejor aplicación de arbitraje que existe actualmente en el mercado. Además, la razón por la que se desarrolló en el año 2012 es exactamente la misma razón que ha hecho que se desarrolle nuestra aplicación:

*“Referee PRO nace en 2012 como el Primer DRSS (Sistema de Señalización Digital, por sus siglas en inglés), la idea de digitalizar los datos oficiales del deporte desde la raíz. Así hemos ayudado a facilitar el trabajo de árbitros y organizadores de ligas con resultados increíbles.*

*Con Referee PRO ya no es necesario administrar datos de la manera tradicional; imprimir identificaciones de jugadores, tomar notas en papel y adivinar el contenido de las Actas Arbitrales, transcribir información a una computadora y perder información crucial en el proceso, así como la falta de transparencia... son cosas del pasado.”*

Según aparece en la página web de la app.

Esta aplicación se encuentra en el mercado desde 2012 y ha ido actualizándose con el paso del tiempo por lo que actualmente es una aplicación muy completa.

Las funcionalidades que esta aplicación proporciona son muy extensas y cubren totalmente cualquier necesidad de un árbitro en un partido.

Algunas desventajas que he encontrado en esta aplicación son que, al proporcionar tanta funcionalidad, la usabilidad se reduce en comparación con las anteriores apps y que, para poder usar la aplicación, es necesario realizar un registro anterior mediante el envío de un e-mail a un particular, la app mejoraría mucho si esto se pudiese realizar directamente desde la app.

**Front-end**

Se denomina *front-end* a la parte del software que interactúa directamente con el usuario.

El *front-end* se encarga de recolectar datos de parte del usuario y darles el formato necesario para enviar estos al *back-end*, a su vez, también se encarga de recibir respuestas del *back-end* y exponer estas de manera entendible para el usuario.

La parte *front-end* de nuestro proyecto sería la propia *app* con la que interactúa el árbitro.

**Back-end**

Se denomina *back-end* a la parte del software que procesa los datos desde el *front-end*.

El *back-end* suele encargarse de la comunicación con las bases de datos de la arquitectura, de las sesiones de los usuarios en la aplicación y, en el caso del desarrollo web, de servir la página web desde un servidor. En definitiva, la parte *back-end* se encarga de la manipulación de los datos.

La parte *back-end* de nuestro proyecto sería la API REST con la que interactúa la *app*.

**Framework**

Un *framework*, entorno de trabajo o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

En el aspecto del software, un *framework* se trata de una estructura conceptual y tecnológica que ofrece recursos al programador a la hora de la organización y el desarrollo del software. Resumiendo, podríamos definir un *framework* como el esqueleto en un desarrollo software. En ocasiones, los *frameworks* van ligados a lenguajes de programación concretos, pero existen *frameworks* que no van asociados a ningún lenguaje de programación en particular.

El uso de *frameworks* suele proporcionar varias ventajas al programador, como no ncesitar el planteamiento de una estructura global ya que esta ya se encuentra marcada por el *framework* o facilitar la colaboración y su mantenimiento en el futuro gracias a la estandarización servida por el *framework*. Además, existen componentes creados explícitamente para *frameworks* concretos, lo que facilita el desarrollo.

En nuestro proyecto hemos utilizado distintos *frameworks* como *Ionic, Spring o Hibernate.*

**Ionic Framework**

*Ionic* es una framework gratuito y open source, para el desarrollo de aplicaciones híbridas basadas en HTML5, CSS y JS. Está construido con Sass y optimizado con Angular4. Esta tecnología trabaja una capa por encima de *Cordova.*

Las principales características de *Ionic* son las siguientes:

* **Alto rendimiento:** *Ionic* está construido para ser rápido gracias a la mínima manipulación que realiza sobre el DOM, sin usar *jQuery* y con aceleraciones de transición por hardware.
* **Uso de Angular4:** *Ionic* utiliza *Angular4* 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 *Angular4*.
* **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 desarrollado una aplicación nativa para *iOS* o *Android*.
* **Diseño atractivo:** *Ionic* está pensado para trabajar con cualquier dispositivo móvil actual por lo que contiene muchos componentes usados en móviles como tipografías o elementos interactivos entre otros.
* **Uso de *CLI****:* Gracias a *CLI* mediante pocos comandos es posible crear un proyecto, compilarlo para distintas plataformas y probarlo.
* **Ionic Creator:** *Ionic* proporciona una interfaz gráfica para poder crear visualmente aplicaciones sin necesidad de escribir ninguna línea de código. De esta manera, Ionic nos permite diseñar nuestras aplicaciones solo con arrastrar y soltar botones, *inputs* y otros elementos.
* **Ionic DevApp:** *Ionic* creó esta aplicación móvil para ayudar a los desarrolladores. Esta aplicación puede previsualizar los proyectos mediante un simple comando si tanto el proyecto como el dispositivo se encuentran conectados en la misma red. Esta aplicación es muy útil para probar las aplicaciones ya que es mucho más rápido que emular en un ordenador o emular en el dispositivo mediante una conexión física con el ordenador.

Falta imagen

**Metodología de desarrollo software**

Se denomina metodología del desarrollo software al marco de trabajo utilizado para estructurar, planificar y controlar el proceso de desarrollo de software.

El objetivo de estas es presentar un conjunto de técnicas tradicionales y modernas de modelado de sistemas que permitan desarrollar software de calidad.

Tras esta definición es posible percatarse de que existen muchas metodologías de desarrollo software y muy diversas entre sí.

Las metodologías pueden clasificarse en dos grandes grupos: Metodologías tradicionales y metodologías ágiles

**Metodologías de desarrollo software tradicionales**

Las metodologías de desarrollo software tradicionales tienen este nombre debido a que son las metodologías más antiguas del sector, pero, aún así, son las que aún en la actualidad tienen mayor peso en el sector.

Dentro de esta clasificación podemos encontrar distintas metodologías.

**Metodología en cascada**

Al hablar de la metodología de desarrollo en cascada podemos estar hablando de la metodología de desarrollo software más antigua.

Este modelo se basa fundamentalmente en que no se puede avanzar en el proceso hasta que el paso anterior se haya completado.

Esta metodología tiene las siguientes fases:

1. **Análisis de requisitos.** En esta fase el equipo de desarrollo documenta todas las peticiones del cliente para su posterior implementación. Es muy importante tener en cuenta que no hay marcha atrás.
2. **Diseño del sistema.** Durante esta fase se elaborará la estructura del sistema y se determinarán las especificaciones de cada módulo del sistema.
3. **Diseño del programa.** En esta fase no se realiza ningún tipo de codificación, simplemente se comienzan a realizar los algoritmos en papel que se codificarán en la siguiente fase.
4. **Codificación.** Se trata de la fase más técnica de todas. Durante esta fase se codificará todo el proyecto en el lenguaje de programación que anteriormente se haya decidido.
5. **Pruebas.** Esta fase final de desarrollo involucra todo tipo de pruebas hacia el proyecto para cerciorarse de que el desarrollo es correcto antes de empezar a utilizarse.
6. **Verificación.** Esta fase es similar a la anterior, la diferencia es que quien realiza las pruebas es ya el cliente. Si se realizó una buena fase anterior, esta fase debería ser poco significante.
7. **Mantenimiento.** Probablemente esta sea la fase más crítica, ya que cualquier desarrollo software necesita evolucionar con el tiempo o sino acabará muriendo. En esta fase se incluyen tanto arreglos en la aplicación como nuevos evolutivos.

**Metodología de prototipos**

Esta metodología consiste básicamente en la recogida de requisitos y necesidades del cliente y, en torno a estas, la realización de crear de manera rápida un prototipo para poder mostrar al cliente y, a partir de este prototipo, continuar con el trabajo.

Las fases de esta metodología son similares a las de metodología en cascada, la diferencia es que estas son mucho más rápidas y que existe una retroalimentación con el cliente para ir trabajando en etapas, algo que la metodología en cascada no permite.

En este tipo de metodología es muy importante la involucración del cliente en el proyecto ya que el desarrollo de este esta constantemente dependiendo de las necesidades del cliente.

**Metodología incremental o iterativa**

Esta metodología es, fundamentalmente, una mezcla entre las dos metodologías anteriores. Básicamente consiste en completar varias iteraciones de todas las fases de la metodología en cascada, pero sin finalizar ninguna de estas, consiguiendo de esta manera la evolución del producto al permitir al cliente añadir nuevas especificaciones al producto después de realizar cada iteración.

**Metodología en espiral**

Esta metodología también se trata de una fusión entre la metodología en cascada y la metodología de prototipos, pero añade a estas una fase de gestión de riesgos, una fase que no existe en ninguna de las metodologías anteriores.

En esta metodología se van realizando las fases del modelo en cascada, pero en este caso no llevan el orden establecido ni son todas de obligado cumplimiento.

El objetivo principal de esta metodología de desarrollo se basa en la minimización de riesgos a lo largo del desarrollo para buscar la máxima seguridad.

Es posible diferenciar cuatro grandes fases en la metodología en espiral:

1. Determinación del objetivo. Se obtiene una planificación inicial.
2. Análisis de riesgos. Se tiene en cuenta todo lo que pueda perjudicar al proyecto.
3. Desarrollar, validar y probar. Esta fase dependerá del análisis de riesgos y se llevarán a cabo las tres tareas.
4. Planificación. Se planifica el siguiente ciclo del proyecto.

Todas estas fases se realizarán tantas veces como sean necesarias en el proyecto.

**Bibliografía**

* <https://www.solbyte.com/blog/2014/07/21/tipos-de-aplicaciones-moviles-nativas-webs-hibridas/>
* <https://okhosting.com/blog/ventajas-desventajas-de-las-aplicaciones-moviles-nativas/#Ventajas_de_crear_aplicaciones_moviles_nativas>
* <https://www.raona.com/aplicacion-nativa-web-hibrida/>
* <https://es.wikipedia.org/wiki/Front-end_y_back-end>
* <https://serprogramador.es/que-es-frontend-y-backend-en-la-programacion-web/>
* <https://es.wikipedia.org/wiki/Framework>
* <https://jordisan.net/blog/2006/que-es-un-framework/>
* <http://www.phonegapspain.com/que-es-y-como-empezar-con-ionic-framework/>
* <https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software#Otros_enfoques_de_desarrollo_de_software>