IMPLEMENTACIÓN DE LA SOLUCIÓN

Hemos llegado a la parte de la implementación de la solución. En esta etapa del ciclo de vida de nuestro sistema, hay que proceder a la creación de las aplicaciones correspondientes, someterlas a pruebas, crear la documentación pertinente y capacitar a los usuarios[1].

Antes de meternos de lleno en el desarrollo de la solución, debemos exponer qué tecnologías vamos a utilizar y lo más importante, **por qué** vamos a utilizarlas.

1. Análisis tecnológico

Para poder realizar un análisis más ordenado de las tecnologías a utilizar, voy a dividir las tecnologías usadas entre la parte del cliente (aplicación móvil) y la parte del backend.

1.1. Tecnología del cliente

Partimos desde la premisa de que nuestra aplicación debe de ser multiplataforma, es decir, debe funcionar en los dos sistemas operativos principales del mercado: iOS y Android. Con esta restricción, debemos dejar de lado la idea del desarrollo totalmente nativo, por lo que Java y Swift quedan descartados.

Debemos acudir, pues, a tecnologías que nos permitan realizar un desarrollo de una solución multiplataforma. Llegados a este punto tenemos varias opciones, entre las que destacan:

- React Native[2]

  Es un framework de Javascript, creado por Facebook, y que permite crear aplicaciones reales nativas para iOS y Android, basado en la librería de Javascript React. En vez de realizar una aplicación web híbrida o en HTML lo que se obtiene con este framework es una aplicación real nativa.

  Utiliza el paradigma fundamental de construcción de bloques de interfaz de usuario (componentes visuales con los que interacciona el usuario) y gestiona la interacción entre los mismos utilizando las capacidades de Javascript y React.

  Las características principales de este framework son:

[1] https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/ciclo-de-vida-de-un-sistema-de-informacion-fases-y-componentes
[2] https://www2.deloitte.com/es/es/pages/technology/articles/que-es-react-native.html

- Compatibilidad: Permite crear aplicaciones que pueden ser ejecutadas tanto en iOS como en Android con el mismo código fuente.
- Funcionalidad nativa: Las aplicaciones creadas con este framework funciona de la misma manera que una aplicación nativa real sin el uso de un webview.
- Hot-reload: Podemos ver los cambios que ejecutamos al instante, debido a la gran velocidad de compilación.
- Sencillo de aprender: Muy fácil de leer y comprender tanto si conoces Javascript como si eres principiante en ese lenguaje de programación.
- Experiencia positiva para el desarrollador: Funcionalidades como el hot-reload y el flexbox layout engine (nos permite abstraernos de los detalles de cada layout en iOS y Android) hacen que el desarrollo sea más satisfactorio para el programador.

Otra de las cosas a destacar de este framework es el gran soporte que tiene y la extensa documentación que facilita[3].

Su principal desventaja es que utiliza un bridge de Javascript que es el que se encarga de "convertir" nuestro código Javascript en código nativo de la plataforma de ejecución, lo que lo hace un poco menos veloz que otras alternativas.

■ Ionic[4]

Se trata de un framework de código abierto que se utiliza destinado al desarrollo de aplicaciones híbridas, es decir, en la que se utilizan HTML, Javascript y CSS para generarar interfaces y se "envuelven" en una aplicación nativa que la muestra en un WebView. Incluso el despliegue se realiza en plataformas nativas (Play Store y App Store).

Lo bueno, es que, junto a este framework, puedes utilizar otro framework de desarrollo front-end web como es Angular, que agilizará el desarrollo mucho más que si se realiza con Vanila Javascript (Javascript puro).

[3] https://reactnative.dev/
[4] https://www.qualitydevs.com/2019/05/31/que-es-ionic-desarrollador-web/

Como principales desventajas, destaca que su rendimiento puede ser ligeramente menor en comparación a una aplicación nativa[5].

- Kotlin Native

  Se trata de una tecnología que permite compilar el código escrito en Kotlin directamente a archivos binarios nativos, por lo que puede ser utilizado sin la necesidad de una máquina virtual (como sería el caso de Java). Actualmente ofrece soporte para plataformas como iOS, MacOS, Android, Windows, Linux y WebAssembly.

  De momento es algo experimental pero que vale la pena mencionar ya que se está apostando mucho por ello y está en constante cambio[6].

- Native Script

  Se trata de una alternativa bastante similar a Ionic, que permite construir aplicaciones para Android y iOS utilizando lenguajes de programación independientes del dispositivo (Javascript o Typescript). También soporta desarrollo directamente con Angular y con Vue.js mediante un complemento desarrollado por la comunidad. Además, se pueden utilizar bibliotecas de terceros como Maven y npm en las aplicaciones móviles[7].

  Como desventajas de esta herramienta encontramos[8]:

  - No tiene una curva de aprendizaje sencilla
  - No se puede reutilizar buena parte del código de la web
  - No tiene un conjunto de componentes de interfaz muy destacable, siendo superado por el de Ionic.

- Xamarin[9]

  Es una plataforma de código abierto para crear aplicaciones multiplataforma (iOS, Android y Windows) con .NET. Se ejecuta en un entorno administrado que

[5] https://openwebinars.net/blog/ionic-framework-ventajas-desventajas/
[6] https://www.bbvanexttechnologies.com/kotlin-native-ayer-hoy-y-manana/
[7] https://es.wikipedia.org/wiki/NativeScript
[8] https://desarrolloweb.com/articulos/ionic-vs-nativescript.html
[9] https://docs.microsoft.com/es-es/xamarin/get-started/what-is-xamarin

proporciona ventajas como la asignación de memoria y la recolección de elementos no utilizados.

Permite a los desarrolladores compartir un promedio de un 90% del código entre plataformas.

Entre sus principales desventajas (que son unas cuantas) es su coste para uso profesional empresarial. Por ejemplo, es necesario comprar una licencia de Visual Studio Professional que cuesta 1.199 dólares el primer año y 799 por renovación[10].

Además, personalmente, pienso que cualquiera de las alternativas aquí mencionadas es mejor.

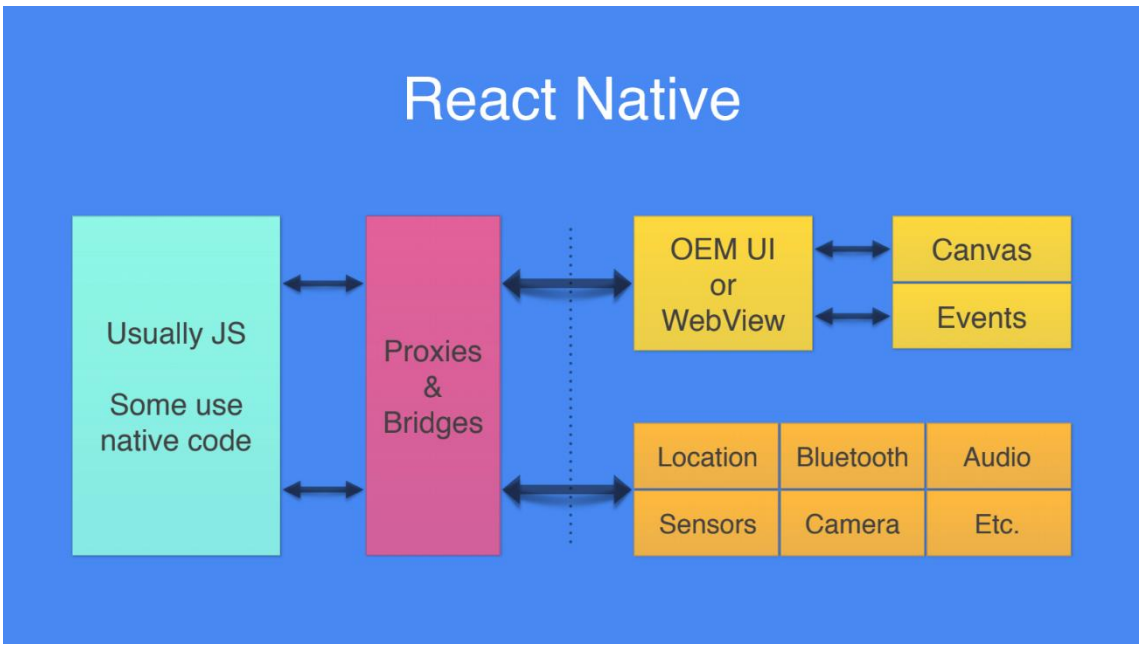
- Flutter[11]

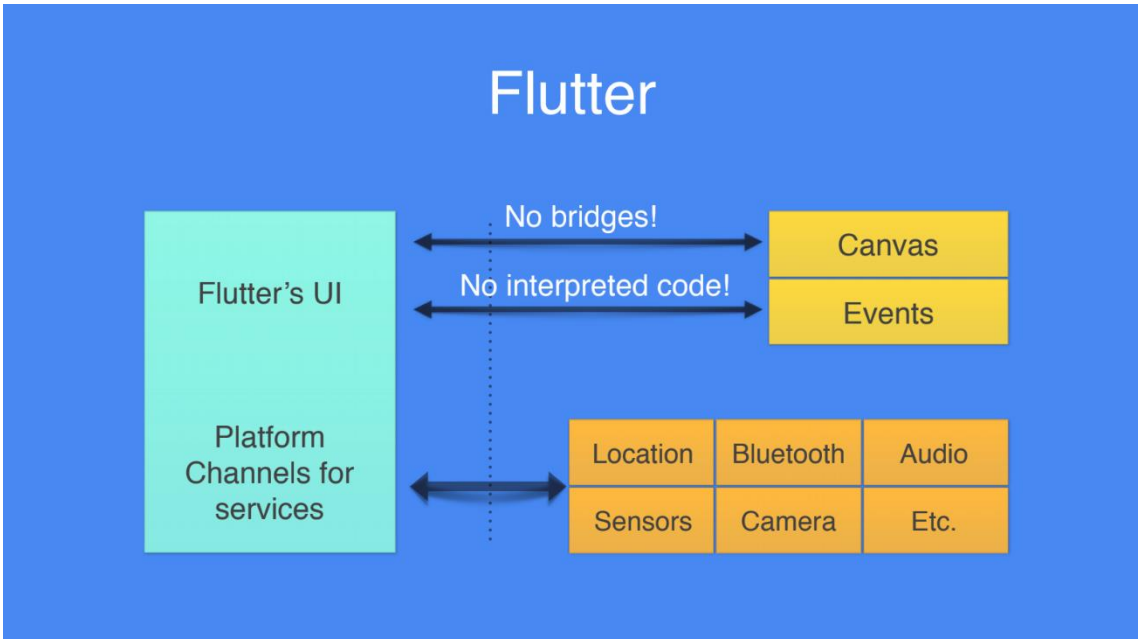
  Es un framework de código abierto desarrollado por Google parar crear aplicaciones nativas de forma fácil, rápida y sencilla. Su principal ventaja radica en que el código que genera es 100% nativo para cada plataforma, consiguiendo un rendimiento y una UI idénticos a las aplicaciones nativas tradicionales. Pese a las promesas de otros frameworks con el mismo propósito, se trata del único en generar código 100% nativo y ejecutable en ambas plataformas.

  Otras ventajas:

  - Experiencia de usuario. Flutter incluye Material Design de Google y Cupertino de Apple por lo que la experiencia de usuario es óptima y las interfaces idénticas a las de las aplicaciones desarrolladas de forma nativa.
  - Tiempo de carga. Con Flutter se experimentan tiempos de carga por debajo de un segundo en cualquiera de las plataformas.
  - Desarrollo ágil y rápido: Gracias al hot-reload.
  - Se puede programar desde cualquier sistema operativo.
  - Rendimiento superior a su máximo competidor, React Native, ya que no utiliza un bridge[12].

[10] https://inmediatum.com/blog/ingenieria/ventajas-y-desventajas-de-apps-desarrolladas-en-xamarin/
[11] https://www.qualitydevs.com/2019/07/05/que-es-flutter/
[12] https://www.bbvanexttechnologies.com/es-flutter-el-framework-del-futuro/





Además, se está viendo que este Framework tiene un gran futuro por delante, debido a que Google está desarrollando su sistema operativo Fucsia, que si finalmente ve la luz sería un salto enorme para Flutter.

Finalmente, es necesario hacer mención del lenguaje que utiliza Flutter. Se trata de Dart, un lenguaje creado también por Google, sencillo de aprender ya que guarda similitudes con C++, C# o Java.

Lo bueno de Dart es que funciona de forma similar a Java, ejecutándose sobre una máquina virtual (Dart VM) y permite a Flutter evitar la necesidad de tener otra capa de

lenguaje declarativo como XML para realizar las interfaces, porque el propio layout es definido en Dart y es fácil de entender y rápido de desarrollar[13].

Conclusión tecnológica para la parte del cliente

Para desarrollar Instadroid, voy a utilizar Flutter. Me ha costado decidirme entre Flutter y React Native, pero finalmente me convenció Flutter porque parece ser que ofrece una mejor performance y tiene detrás una gran comunidad y al mismísimo Google[14].

Además, estos son otros de los motivos:

- No necesito utilizar ningún lenguaje para definir las vistas, ni HTML, ni XML, utilizando el propio Dart y de forma sencilla podré hacerlo.
- Ofrece muchas facilidades para aplicaciones que realizan cargas de datos en tiempo real como los Widget Future
- Muestra unas interfaces increíbles, totalmente iguales a las de aplicaciones nativas
- Al ser un producto de Google como Firebase tiene una buenísima integración con esta última tecnología
- Pienso que mi desarrollo va a ser mucho más ágil y rápido
- Puedo utilizar el Hot Reload para ver cambios y realizar pruebas al momento
- Por gusto personal de la tecnología
- Porque al estar en la parte de móviles en mis prácticas me gustaría saber y entender las tecnologías que invadirán el desarrollo móvil en los próximos años

[13] https://alexmarket.medium.com/flutter-o-nativescript-707b7ffdbceb
[14] https://flutter.dev/

1.2. Tecnología del lado del servidor

Del lado del servidor serían muchas las tecnologías que podríamos utilizar para construir una API REST, como:

- SpringBoot
- Django
- Laravel
- Node js con express
- Deno
- .NET

Incluso podríamos utilizar sockets de cualquier lenguaje de programación que trabaje con ellos como Java o Python.

Además, estas tecnologías podrían conectar con cualquier tipo de base de datos tanto SQL como NoSQL, como MySQL, Oracle o Mongo.

Nuestra principal limitación a la hora de elegir una tecnología para el backend viene marcada por la necesidad de la aplicación de mostrar datos en tiempo real, lo que nos hace acudir a distintas soluciones que ofrecen un backend ya realizado en vez de realizar nosotros el nuestro propio.

Entre estas alternativas destacan[15]:

- Back4App

  Una solución sencilla y diseñada para aplicaciones móviles y sitios web. Ofrece una manera más rápida y sencilla de alojar y administrar aplicaciones. Es una solución de código abierto que ofrece múltiples herramientas e integraciones que nos ayudan como desarrolladores a construir aplicaciones de una forma mucho más rápida. Algunas de sus características son:

  - Modelo de datos
  - APIs GraphQL
  - Base de datos en tiempo real

[15] https://blog.back4app.com/es/las-mejores-alternativas-a-firebase/

- Autenticación de dos factores
- Copias de seguridad automatizadas
- Posibilidad de utilizarlo directamente como una API Rest

- Parse

  Se trata de una plataforma que ofrece funcionalidades similares a la anterior, pero es más fácil de utilizar en comparación con otras opciones similares. De tal manera que ofrece características y herramientas para desarrollar y administrar servicios de backend:

  - Base de datos en tiempo real
  - Integración con redes sociales
  - Autenticación a través de correo electrónico y contraseña
  - Funciones de seguridad
  - Alojamiento SSL

- AWS Amplify

  De código abierto, Amazon ha creado un gran entorno que permite a los desarrolladores a crear aplicaciones sin servidor, fácilmente integrable con cualquier frontend.

  Destaca su interfaz de usuario fácil de usar y navegar y sus extensas librerías que tienen la capacidad de permitir que los desarrolladores conecten sus soluciones backend con interfaces móviles.

  Se trata de la alternativa y competidora más fuerte de Firebase.

  Algunos de los servicios que ofrece:

  - Notificaciones push
  - Marketing vía correo electrónico
  - Base de datos en tiempo real
  - Geolocalización

- Firebase[16]

  Se trata de una plataforma móvil creada por Google cuya principal función es desarrollar y facilitar la creación de apps de elevada calidad de forma rápida. La plataforma se encuentra en la nube y está disponible para distintas plataformas como iOS, Android y Web. Contiene diversas funciones para que cualquier desarrollador pueda combinar y adaptar la plataforma a sus necesidades.

  Características principales:

  - Desarrollo: Firebase permite la creación de mejores apps minimizando el tiempo de optimización y desarrollo mediante diferentes funciones, entre las que destacan la detección de errores y el testeo. Poder almacenar todo en la nube o poder configurarla de forma remota son características destacables de la plataforma.
  - Analítica: Tener un control máximo del rendimiento de la app mediante métricas analíticas desde un único panel y de forma gratuita.
  - Poder de crecimiento: Permite gestionar de manera fácil todos los usuarios de las aplicaciones, con el añadido de captar nuevos usuarios mediante invitaciones o notificaciones.
  - Rapidez: Implementar Firebase puede ser fácil y rápido gracias a su API, que es muy intuitiva. De esta manera, podremos invertir más tiempo en resolver los problemas de las apps cliente y evitar perder tiempo en la creación de una infraestructura compleja.
  - Agilidad: Firebase ofrece apps multiplataforma con APIs integradas a SDK individuales como iOS, Android y Javascript
  - Extensa documentación. La documentación[17] de Firebase es muy extensa y sencilla de entender y aplicar. Además, cuenta con canal de YouTube[18] y sus desarrolladores participan activamente en GitHub y StackOverflow dando soporte al que lo necesite.

[16] https://www.iebschool.com/blog/firebase-que-es-para-que-sirve-la-plataforma-desarroladores-google-seo-sem/
[17] https://firebase.google.com/docs/auth/android/start?hl=es-419
[18] https://www.youtube.com/user/Firebase

¿Qué **servicios** nos ofrece Firebase?[19]. Varios. Entre los que destacan:

- Realtime Database / Cloud Firestore

  Son las bases de datos en tiempo real, se alojan en la nube y son NoSQL. Cloud Firestore es la nueva opción, que poco a poco irá sustituyendo a Realtime Database. El funcionamiento es similar, pero Cloud Firestore admite tipos de datos mientras que Realtime Database solo almacena JSON.

  Firebase envía automáticamente eventos a las aplicaciones cuando los datos cambian.

- Autenticación de usuarios

  Firebase ofrece un sistema de autenticación que permite el registro, como el acceso utilizando otras plataformas externas (como Google o Twitter).

- Almacenamiento en la nube

  Firebase cuenta con un sistema de almacenamiento, donde los desarrolladores pueden guardar ficheros y sincronizarlos. Es personalizable mediante reglas. Es muy útil para guardar fotografías, por ejemplo.

- Cloud Messaging

  Servicio de envío de notificaciones y mensajes a diversos usuarios en tiempo real y a través de varias plataformas.

Conclusión tecnológica para la parte del servidor

Para el backend de Instadroid, como bien ya he venido exponiendo, voy a utilizar Firebase. Me parece la mejor alternativa y además debemos cumplir el requisito de carga de datos en tiempo real.

De igual manera, estos son los servicios a utilizar en Firebase:

[19] https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/

- Servicio de autenticación de usuarios: Nuestros usuarios tendrán la opción de autenticarse con Google y Twitter, algo que ya nos ofrece Firebase.
- Cloud Firestore: Será nuestra base de datos NoSQL, en la que almacenaremos los datos de la aplicación.
- Cloud Storage: Lo utilizaremos para guardar las distintas fotos que suban los usuarios.
- Cloud Messaging: Para enviar las notificaciones.