Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

INTRODUCCIÓN AL DESARROLLO DE PROGRESSIVE WEB APPS: ANÁLISIS DE LA METODOLOGÍA Y EJEMPLO DE APLICACIÓN PRÁCTICA

Alumno: Miguel Ángel Carpio Colomo

Tutor: Prof. D. Víctor Manuel Rivas Santos
Dpto: Informática

Junio, 2021

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Victor Manuel Rivas Santos , tutor del Trabajo Fin de Grado titulado: ”*Introducción al desarrollo de Progressive Web Apps: análisis de la metodología y ejemplo de aplicación práctica*”, que presenta Miguel Ángel Carpio Colomo, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2021

El alumno: El tutor:

Miguel Ángel Carpio Colomo Victor Manuel Rivas Santos

Índice

1. Introducción

Son miles las aplicaciones y sitios web que visitamos y utilizamos en nuestro día a día, pero, ¿qué pasaría si fusionamos las características de uno con las características de otro? El resultado de dicha fusión es lo que conocemos como Aplicaciones Web Progresivas (*Progressive Web Application*, PWA), es decir, una nueva corriente de desarrollo de aplicaciones en la que se combina el funcionamiento de las aplicaciones web que todos conocemos, junto con la compatibilidad en diferentes dispositivos y funcionalidades que aportan las aplicaciones nativas.

En este Trabajo de Fin de Grado abordaremos todo lo relacionado con esta corriente emergente, analizaremos de dónde proviene este concepto, cuáles son sus características, quiénes utilizan este tipo de aplicaciones, cómo las podemos construir, cuál es el desarrollo de cara al futuro de este tipo de aplicaciones y mucho más. Así pues también pasaremos a realizar una pequeña aplicación web y la transformaremos en una aplicación web progresiva.

2. Motivación

Lo que me llevó a coger este Trabajo Fin de Grado fue el hecho de investigar esta nueva corriente de desarrollo ya que durante el trascurso del grado no se ha mencionado este concepto de Aplicaciones Web Progresivas. Además me considero una persona curiosa y que siempre busca el constante aprendizaje, por lo que este TFG se adapta a la perfección a mi mentalidad.

Por otro lado también conocía al tutor que dirige este trabajo de fin de grado de anteriores asignaturas del grado y mi experiencia con él fue sin duda muy satisfactoria, por lo que al contar con él como tutor asumía que sin duda iba a ser una gran elección y que me iba a guiar de la mejor forma posible.

3. Objetivos

En este TFG se pretende realizar un trabajo de documentación en relación a esta metodología o marco de trabajo: *Progressive Web Apps*. Se tratará de definir claramente qué es, cuáles son las alternativas que existen a las que intenta suplantar o complementar y desglosar justificadamente un conjunto de ventajas e inconvenientes de unas sobre otras.

Además, se desarrollará una aplicación a modo de prueba de este marco de trabajo; para ello, se utilizará una metodología adecuada y justificada para el proceso de ingeniería del software de dicha aplicación.

4. Planificación temporal inicial

De forma voluntaria he querido reflejar una planificación temporal en la que estimé tiempos en función de las etapas que tendría este Trabajo Fin de Grado. Esta planificación temporal se lleva a cabo por medio de un Diagrama de *Gantt*.

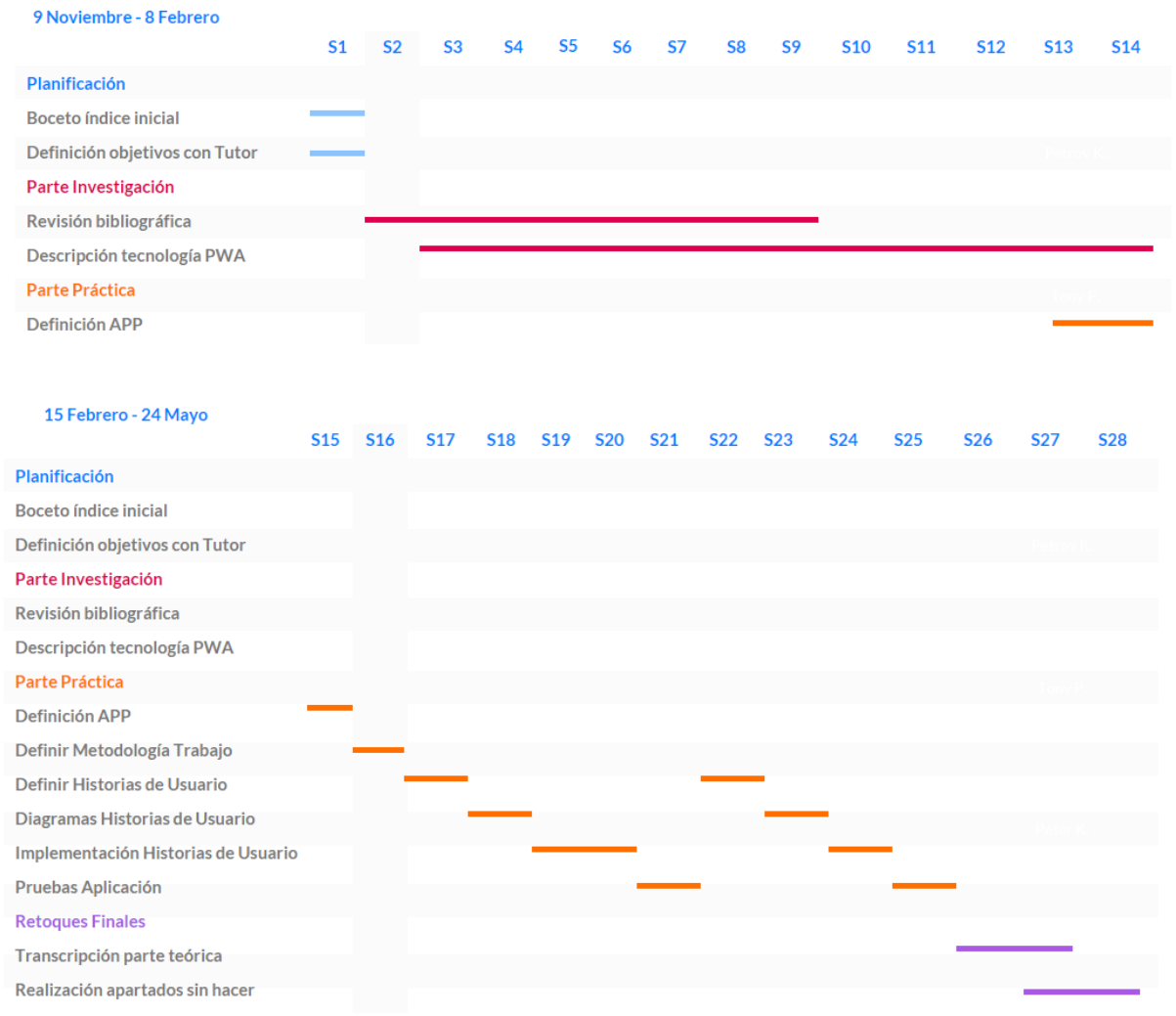

Ilustración 1. Diagrama de Gantt con la planificación inicial del Trabajo de Fin de Grado

5. PWA

5.1. Historia de las PWA

En 2015 fue desarrollado por Google el término aplicación web progresiva, sin embargo la idea de centrarse en la experiencia del usuario móvil no es algo que podamos considerar nuevo. De hecho la definición actual que Google ofrece difiere con respecto a la primera que se dio, es por ello que para poder comprender todo el concepto del estándar PWA, debemos analizar su origen [1, 2].

Los orígenes de las PWA podríamos situarlos a principios de la década del 2000 con la creación de las tecnologías *XML*, *HTTP*, *Request*. Estas tecnologías permiten recuperar datos de una URL sin tener que actualizar toda la página. Cinco

años después tenemos a *AJAX*. *AJAX* es una tecnología que utiliza *XML*, *HTML*, *CSS* y *JavaScript* para crear aplicaciones web mejores, más rápidas y más interactivas. *AJAX* permite que las aplicaciones web envíen y reciban datos de forma asíncrona, es decir, que se ejecuten en segundo plano, sin afectar la funcionalidad o el rendimiento de las páginas que conocemos hoy en día.

Estas nuevas tecnologías, se basan en algunos principios comunes, como conectividad, disponibilidad del sistema tanto para motores como para personas, una interfaz de usuario que complementa la estructura funcional y la libertad de administrar y distribuir sin privilegios ni licencias adicionales.

Ni a Google ni a Apple se les ocurrió la idea de las PWA, sino que fue Steve Jobs quien anunció la idea cuando presentó el iPhone en 2007. En un principio parecía natural que las aplicaciones externas fueran útiles para conseguir una mayor reputación para ese dispositivo. Steve Jobs en ese momento buscaba que los desarrolladores utilizasen tecnologías estándar de Internet para el desarrollo de las aplicaciones [22].

> *"(...) puedes escribir increíbles aplicaciones Web 2.0 y Ajax que se ven y se comportan exactamente como las aplicaciones del iPhone. Y estas aplicaciones pueden integrarse perfectamente con los servicios de iPhone. ¿Y adivina qué? ¡No necesitas un SDK! Tiene todo lo que necesita si sabe cómo escribir aplicaciones utilizando los estándares web más modernos para escribir aplicaciones asombrosas para el iPhone hoy.*
>
> *Entonces, desarrolladores, creemos que tenemos una historia muy agradable para ustedes. Puede comenzar a crear sus aplicaciones de iPhone hoy "- Steve Jobs, Apple*

La *App Store* no se mencionó en ese momento, pero eso cambió drásticamente con el lanzamiento del SDK de iPhone en octubre del 2007. Fue entonces cuando Apple incorporó la *App Store* en julio del 2008.

Esta idea de las "aplicaciones universales" ya existía, sin embargo la idea permaneció paralizada durante casi diez años, tiempo que fue reinado por las aplicaciones nativas, que cambiaron la forma en que se usa Internet. Las

aplicaciones nativas controlan las redes móviles y ayudaron a mejorar el rendimiento de Google y de Apple en dispositivos móviles. De este modo, los dueños de los sitios web se empezaron a comprometer con el diseño web ***responsive***. Esta característica implica mostrar páginas web correctamente en diferentes dispositivos y resoluciones de pantalla usando una gran variedad de cuadrículas de *CSS* cada una con unas proporciones y medidas concretas. La idea hacia los PWA se vio por tanto ralentizada.

Tal y como narraba Alex Russell, ingeniero de software sénior de Google, en su texto "*Progressive Web Apps: Escaping Tabs Without Losing Our Soul*". Publicado en Infrequently.org en julio del 2015 [23]:

> *"Ocurre en la web de vez en cuando que las tecnologías poderosas llegan a existir sin el beneficio de los departamentos de marketing o aplicaciones sofisticadas. Permanecen y crecen en la periferia, convirtiéndose en algo anticuado para un grupo pequeño mientras permanecen casi invisibles para todos los demás. Hasta que alguien los nombra". - Alex Russell, ingeniero de software senior de Google*

Y como ya dijeron Frances Berriman y Alex Russell, los autores del término PWA, en el prólogo de " *Progressive Web Apps, a book by Jason Grigsby"* [24]:

> *"La idea de las aplicaciones nativas siempre pareció una regresión. Jardines amurallados con una búsqueda terrible, una seguridad dudosa y el impuesto interminable de las actualizaciones, se sentía tan de los noventa". - Frances Berriman y Alex Russell*

Berriman y Russell proponen en 2015, una nueva categoría de sitios web que ofrecían una mejor experiencia de usuario, que las otras aplicaciones web tradicionales. Todos estos programas informáticos tienen una función en común, y es que, desaparecen las pestañas del navegador manteniendo intacto y tanto la versatilidad y como la conectividad.

Esta nueva categoría son precisamente las "**aplicaciones web progresivas**". Al año siguiente, Eric Bidelman, ingeniero sénior de aplicaciones en RR.HH., en la conferencia de Google, ofreció una conferencia sobre las "*Progressive Web Apps*" como el inicio de un nuevo punto de referencia para el desarrollo web.

5.2. ¿Qué son las PWA?

PWA es el acrónimo en inglés de *Progressive Web Apps* (Aplicaciones Web Progresivas). Antes de explicar lo que son las PWA, será necesario explicar el concepto de Aplicación Nativa [3, 4]:

- Aplicaciones que se instalan mediante *App Stores*
- Pantalla Completa en el dispositivo
- Muchas veces se ganan el derecho a estar en nuestro *HomeScreen* o Pantalla de Inicio de nuestro dispositivo

Una vez definido el concepto de Aplicación Nativa podemos decir que las PWA, son aplicaciones Web, que también puede ser una página web que progresivamente va implementando características tales como notificaciones *Push*. En cuanto a sus características principales encontramos:

- La posibilidad de poder ser añadida al *HomeScreen* sin ser diferenciada de una aplicación nativa
- Funciones sin conexión a Internet
- Uso de características nativas de nuestro dispositivo (cámara, localización, micrófono...)
- Se actualizan constantemente
- Atractiva para los usuarios
- Pesan muy poco
- Rápida a la hora de cargar

Una PWA bien hecha es totalmente indiferente de una aplicación Nativa, sin embargo, una PWA, puede cargar más rápido, es confiable y está actualizada.

Como se ha comentado, una aplicación nativa se encuentra en una *App Store*, sin embargo, si queremos que dicha aplicación se mantenga actualizada, tendremos que actualizarla manualmente nosotros, o bien activar las actualizaciones automáticas de dicha tienda, normalmente la mayoría de usuarios no tienen actualizadas sus aplicaciones porque suelen tener algún limitante de espacio, es por ello que las PWA se aprovechan de esto, debido a que estas aplicaciones no preguntan al usuario si quieren actualizar o no, sino que seremos nosotros como diseñadores los que tendremos el control de cuando se quiera actualizar o no.

Pensadas para ***Mobile First***, es decir, el objetivo es diseñar una aplicación pensando en que los diferentes usuarios de la misma, accederán a ella utilizando su dispositivo móvil, luego en tablets, y por último, pantallas de escritorio

5.3. ¿Quién da soporte a las PWA?

El diseñador Francis Berriman y el ingeniero de Google Chrome Alex Russell adoptaron en 2015 el término aplicación web progresiva para definir a aquellos programas o aplicaciones que utilizan nuevas funcionalidades del navegador, como los trabajadores de servicios de Internet (*service worker*) y el manifiesto de aplicación web. Como resultado, Google ha estado realizando grandes esfuerzos para acelerar el desarrollo de aplicaciones PWA para Android. Además, con la introducción en el navegador de Apple conocido como Safari del *Service Worker Support* en 2017, las PWA ahora son compatibles tanto con Android como Apple, los dos sistemas operativos móviles más utilizados.

Tal fue la evolución de este tipo de aplicaciones que para el año 2019 las PWA ya se encontraban disponibles para Google Chrome y Microsoft Edge. Las Aplicaciones Web Progresivas tienen el acceso total de las API de Windows 10 y se pueden instalar en cualquier dispositivo UWP (incluyendo a aquellos dispositivos con la funcionalidad S de Windows 10), mientras se mantiene la compatibilidad con otros navegadores y dispositivos.

A modo de resumen, se podría decir que a Google se le ocurrió el concepto de las PWA, pero Microsoft apostaba por la API de **UWP** [25] (*Universal Windows*

Platform). La plataforma universal de Windows (UWP) fue desarrollada y expuesta por Microsoft y fue de las primeras API incorporadas en Windows 10. Esta plataforma tenía como objetivo crear aplicaciones universales que puedan ejecutarse en Windows 10, Windows 10 Mobile, Xbox One y HoloLens, sin embargo este proyecto por parte de Microsoft acabó fracasando, lo que provoca un claro impulso en las tecnologías PWA. A pesar de este hecho, la cooperación entre estos dispositivos aún no es lo suficientemente fuerte. En cambio, lograron formar una asociación para desarrollar los estándares de las PWA. Después de un largo proceso, Google era el líder en aplicaciones web progresivas y necesitaba un soporte completo, así lo dijo Jeff Bertoft [1]:

> *"Google abrió el camino con las aplicaciones web progresivas y, después de un largo proceso, decidimos que necesitábamos apoyarlo por completo". – Jeff Burtoft .- Director de programas de Microsoft*

Tal y como se puede observar en la *ilustración 2,* el interés por las aplicaciones nativas se mantuvo a un ritmo constante dado que este concepto ya estaba muy consolidado tanto entre los desarrolladores como en los usuarios. Sin embargo, el interés por las PWA es considerablemente superior al de las nativas debido a la novedad que este tipo de aplicaciones supone para el panorama tecnológico y la gran proyección de futuro que conlleva estas aplicaciones [1].

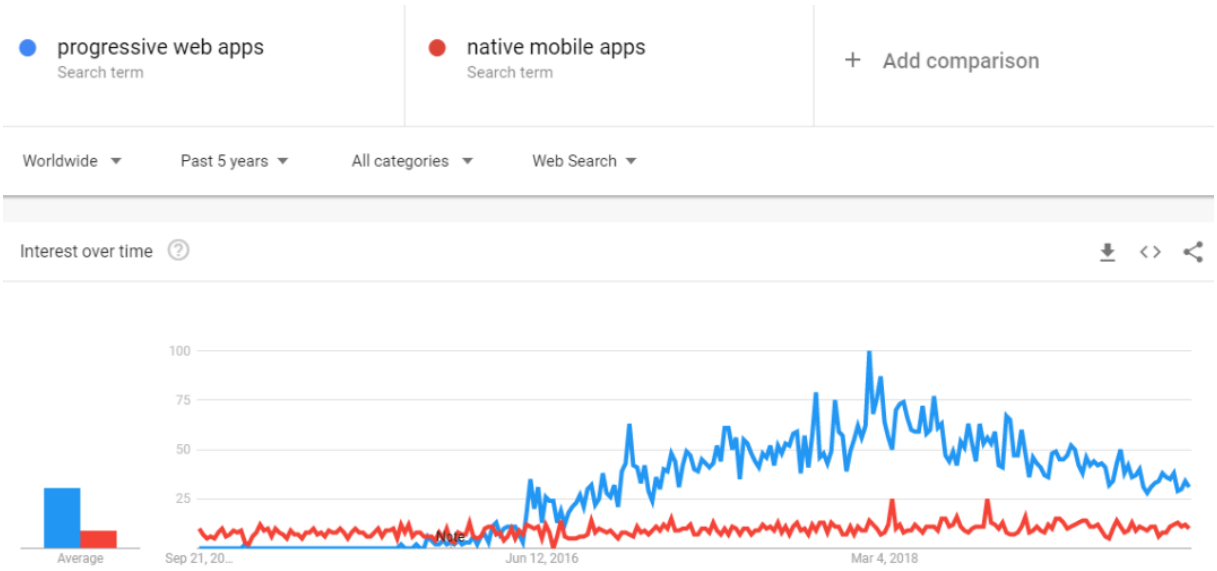

Ilustración 2. Gráfico de interés mostrado por las aplicaciones nativas y aplicaciones progresivas a lo largo del tiempo

A continuación, se pasará a analizar algunas de las herramientas que hoy en día favorecen el soporte y el desarrollo de las PWA [5].

5.3.1. Extensión de código en Visual Studio

Existe un extensión para el conocido IDE de desarrollo *Visual Studio Code* que añade fragmentos de código al esquema JSON del archivo *manifest.json* para crear una aplicación de web progresiva (PWA), además de facilitar el desarrollo y la implementación del *service worker*. En la ilustración 3 se puede observar cuál es la extensión junto a una pequeña visualización de lo que permite.

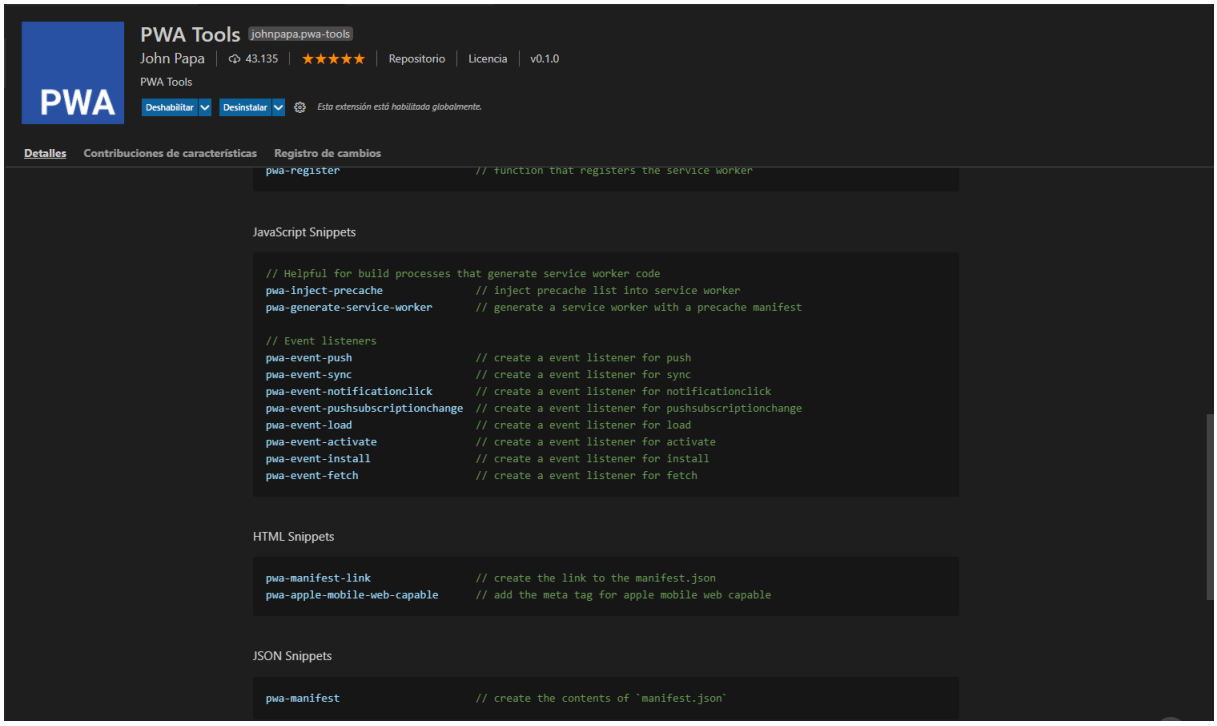

Ilustración 3. Extensión de Visual Studio Code para facilitar el desarrollo de PWA.

5.3.2. Depuración de PWA con Microsoft Edge

Las Dev Tools de Microsoft Edge actualmente son compatibles con la depuración y la inspección de las tecnologías PWA básicas, entre las cuales podemos incluir el *service worker* tal y como se puede ver en la ilustración 4, la API de caché e IndexedDB

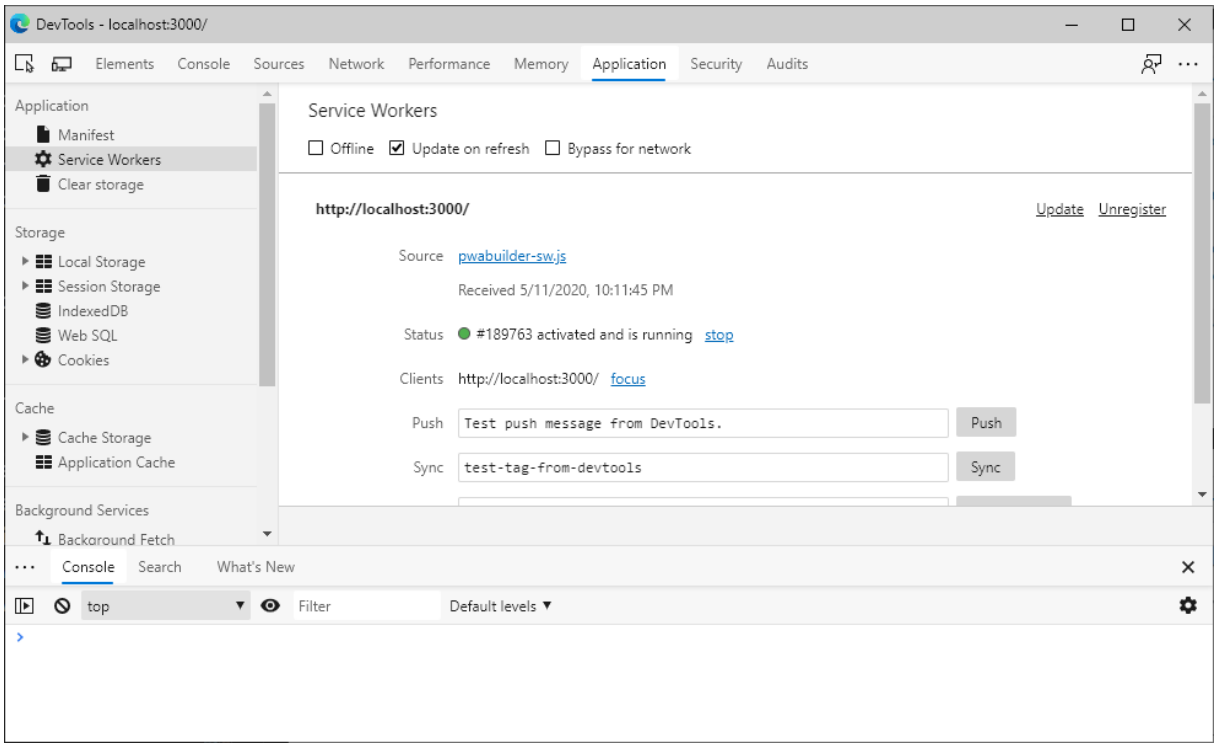

Ilustración 4. Dev Tools de Microsoft Edge para Service Worker.

5.3.3. PWA Builder

Esta herramienta es capaz de analizar un sitio Web concreto para crear un manifiesto completo. El usuario tendrá que gestionar los diferentes tamaños de imágenes y realizar un seguimiento de las especificaciones para asegurar que el tanto el manifiesto como el *service worker* están generados correctamente a partir de la introducción de la URL de la PWA; esto lo podemos contemplar en la ilustración 5. Este tipo de herramienta se utiliza para transformar una aplicación existente en una PWA.

Ilustración 5. Interfaz principal de la herramienta PWA Builder.

5.3.4. Sonarwhal

Es una herramienta de código abierto totalmente personalizable, y puede ser utilizado para ayudar a las distintas aplicaciones web a comprobar distintos parámetros como accesibilidad, interoperabilidad, seguridad, funcionamiento a la hora de establecer los estándares y los componentes web de las PWA. A continuación en la ilustración 6 se observa cual es la interfaz principal de esta herramienta junto con los análisis principales que proporciona.

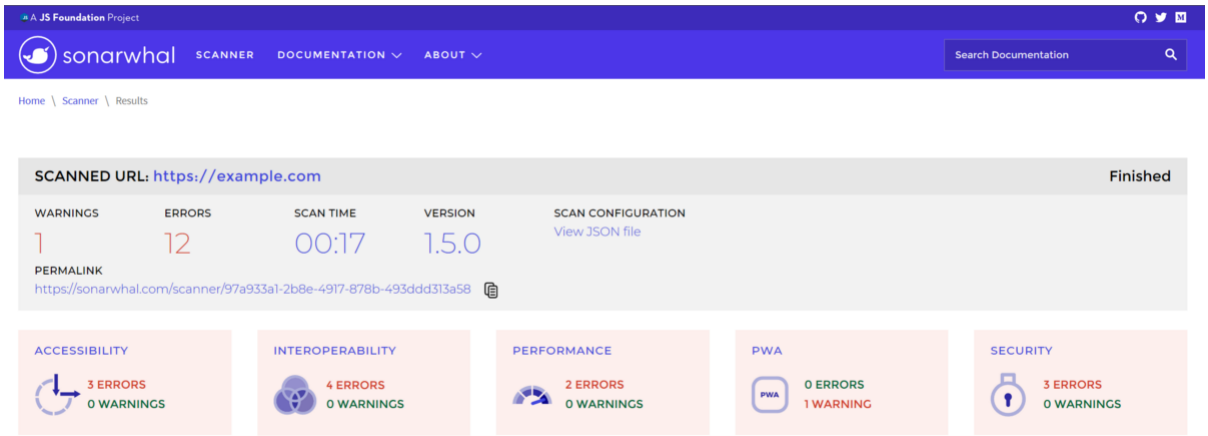

Ilustración 6. Interfaz principal de la herramienta Sonarwhal.

5.4. ¿Quién usa las PWA?

5.4.1. Alibaba

Alibaba opera en más de 200 países y se trata del mercado B2B más grande del mundo. Maneja una base de clientes tan grande que precisa de un puente más eficiente entre las aplicaciones y su web; es por ello que decidieron brindar una mejor experiencia móvil para clientes nuevos y para clientes que ya hayan utilizado sus servicios. De hecho una parte importante del éxito de Alibaba es proporcionar excelentes servicios móviles para ambos tipos de usuarios [6].

Como resultado de la implementación de la PWA, Alibaba aumentó en hasta un 76% el número de conversiones totales, con un aumento de usuario de Android de un 30% y un 14% en iOs. Estos números nos llevan a la conclusión de que la opción de “Añadir al escritorio” provoque un buen *feedback* y una mayor atracción por parte de los usuarios

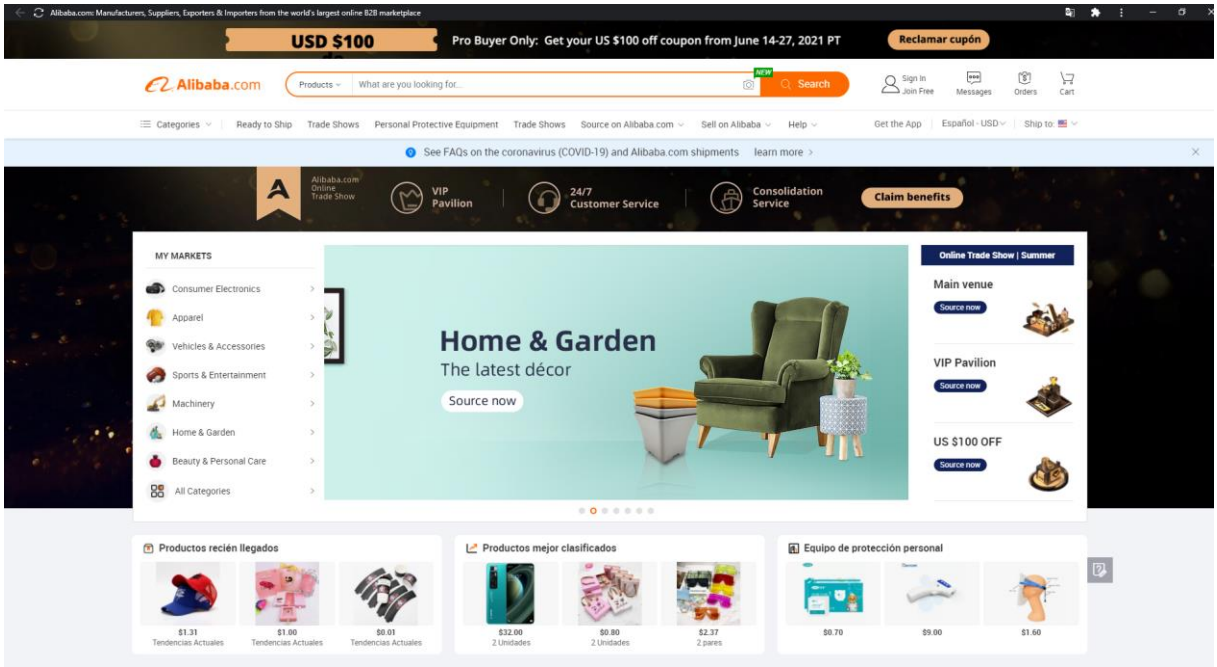

Ilustración 7. Captura de pantalla de la PWA de Alibaba.

5.4.2. The Washington Post

El 50% de las visitas al prestigioso periódico The Washington Post provienen de dispositivos, es por ello que entienden que la clave del éxito a largo plazo es una buena experiencia de lectura móvil. A partir de la observación de estos análisis The Washington Post decidió junto a un grupo de empresas la fundación del "*Accelerated Mobile Pages Project*". Este proyecto fue creado como una nueva plataforma que te permite enviar contenido directamente a todo tipo de dispositivos [6].

La plataforma PWA permite al Washington Post publicar más de 1.000 artículos al día, con un tiempo de carga promedio de 400 ms (88% mejor que los sitios móviles tradicionales). Este cambio implicó que la mayoría de sus consumidores, concretamente un 63% de ellos, instalasen la PWA en un plazo de 7 días. Lo que demuestra que las tecnologías PWA es una herramienta de fidelización completamente válida.

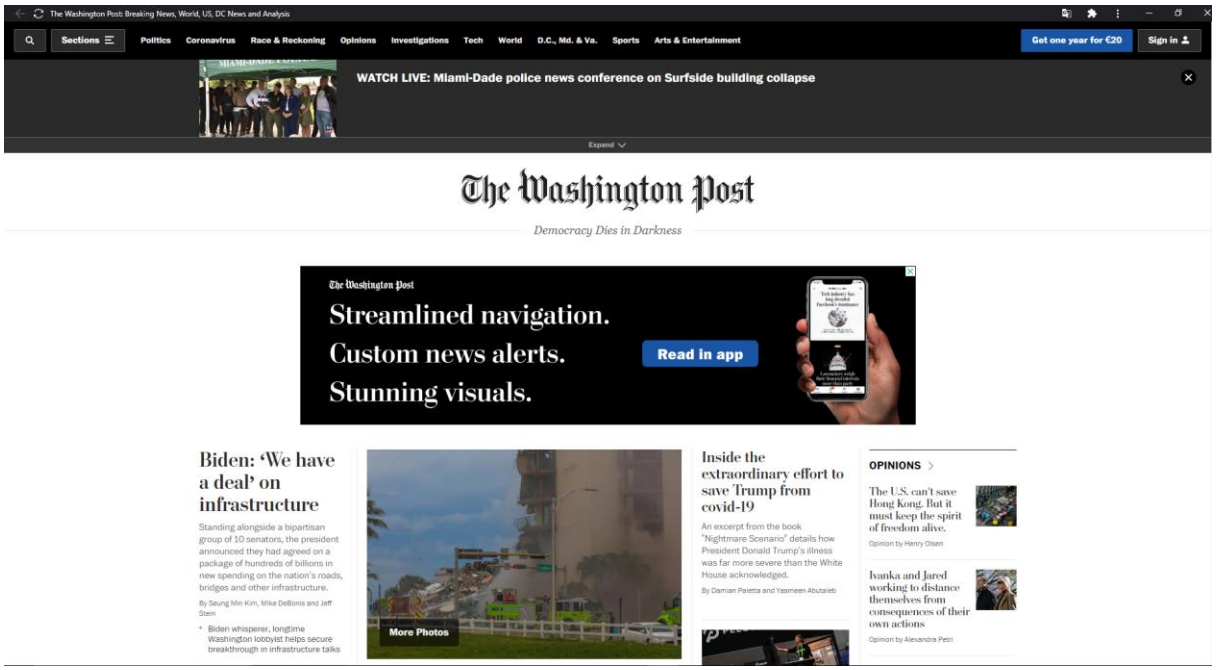

Ilustración 8. Captura de pantalla de la PWA de The Washington Post.

5.4.3. The Weather Channel

The Weather Channel es una empresa cuyo objetivo se basa en proporcionar a los consumidores la información meteorológica más precisa en tiempo real. Uno de sus valores principales que van implícitos en su filosofía es la continua innovación tecnológica, es por ello que siempre están en busca de nuevas tecnologías. La compañía quiso ofrecer el mismo diseño para usuarios con malas conexiones y por tanto decidieron apostar por la implementación de su propia PWA. Con la introducción de PWA, la velocidad de carga ha aumentado hasta en un 80%. Además, gracias a las novedosas tecnologías les permitió habilitar soporte para más de 60 idiomas diferentes con un único código fuente, aumentando considerablemente la eficiencia de la aplicación [6].

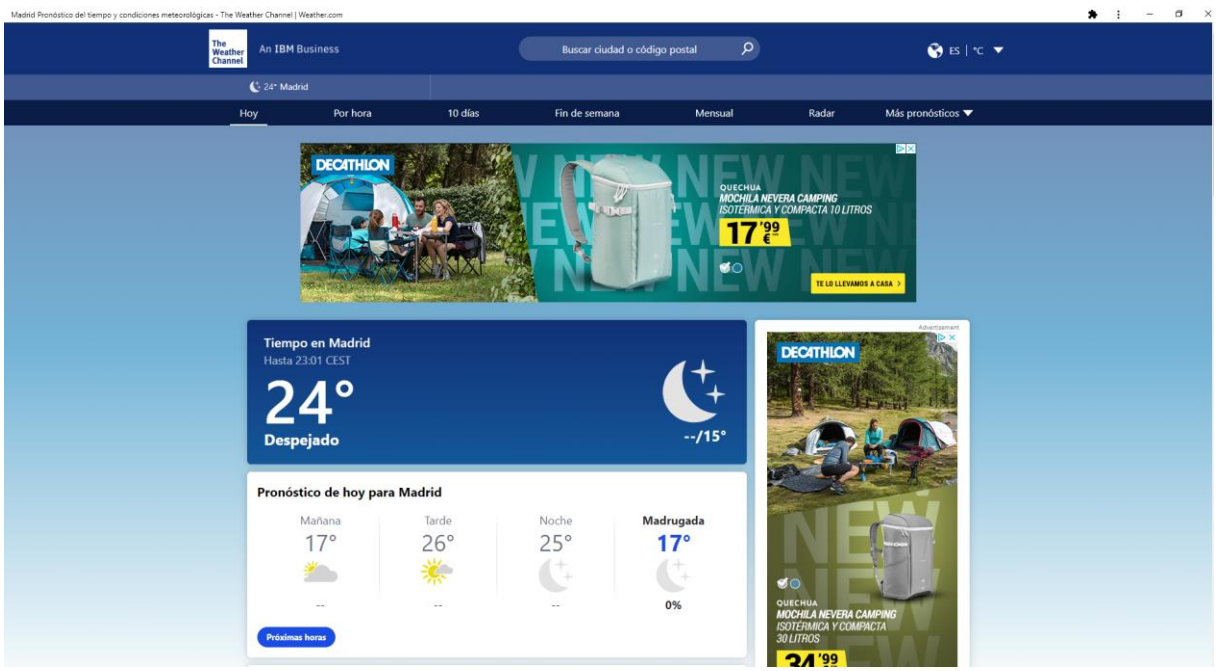

Ilustración 9. Captura de pantalla de la PWA de The Weather Channel.

5.4.4. Google Drive

Últimamente la compañía ha estado ocupada enfocándose en el desarrollo de las PWA para sus propios servicios que ofrece. Durante el pasado año 2020 Google lanzó las PWA respectivas Google Fotos y YouTube Music. A partir de febrero de este año 2021 los usuarios de Google Drive ya pueden disfrutar también de su PWA, la cual permite acceder fácilmente a todos los archivos y documentos alojados en Drive. Cuando se completa la instalación, el programa se abre en una nueva ventana con acceso a todos los archivos. Sin embargo al no estar completamente implementadas las PWA de todos los servicios de Google provoca que algunas funcionalidades no funcionen tan bien del todo, como por ejemplo los documentos de Google, que si los intentamos abrir desde la PWA de Google Drive, se abre en una pestaña de Chrome [7, 8].

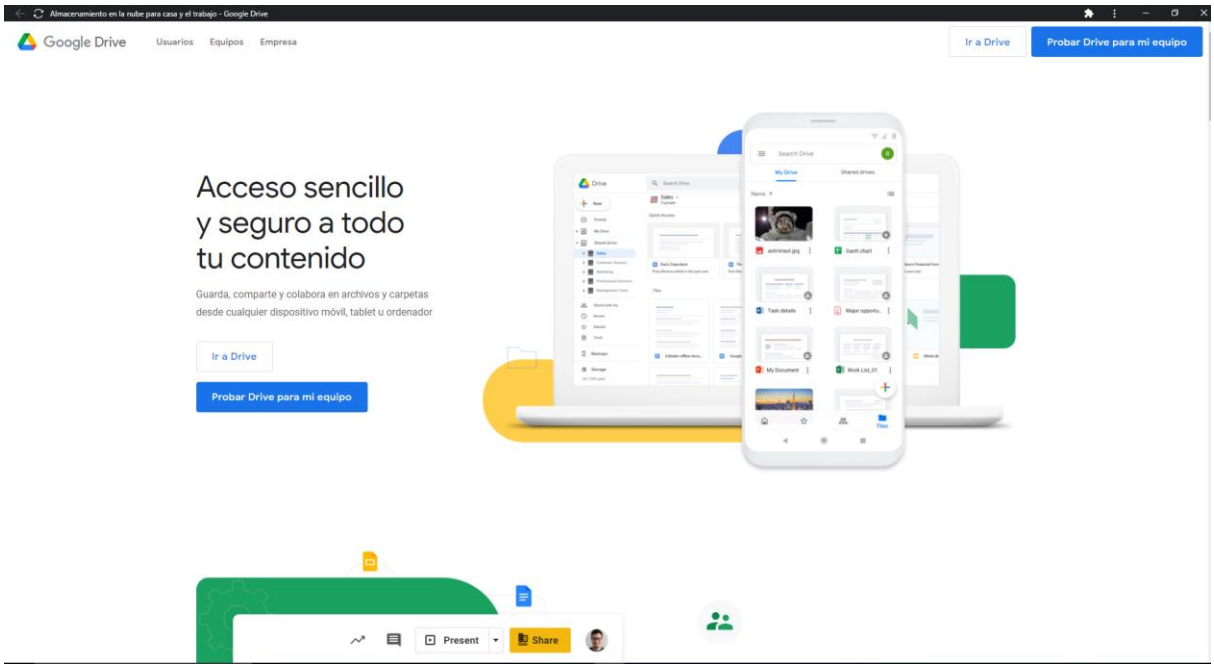

Ilustración 10. Captura de pantalla de la PWA de Google Drive.

5.5. ¿Por qué construir una PWA?

Tal y como se puede observar en la ilustración 11, en 2016 el acceso a las aplicaciones web tuvo lugar en un 57% desde dispositivos móviles o *tablets*, en 2017 este número se incrementó hasta un 63%, por lo que se puede llegar a deducir que la tendencia es creciente hacia los dispositivos móviles. Es por ello por lo que las aplicaciones progresivas son una ideal alternativa para mejorar la experiencia de usuario, pues como se ha comentado anteriormente este tipo de aplicaciones siguen la filosofía de ***Mobile First*** [9]

Ilustración 11. Gráfico de interés mostrado por el diferente acceso en los diferentes dispositivos a las aplicaciones web

En la conferencia de Google de 2017 se mostró el siguiente gráfico, correspondiente a la ilustración 12 de este documento [10].

Ilustración 12. Gráfico de alcance de visitas entre las aplicaciones móviles y las aplicaciones web

Con la ilustración 12 podemos deducir que el alcance de las páginas web visitadas desde el móvil casi triplica a las visitas desde las aplicaciones móviles nativas.

Sin embargo con la ilustración 13 obtenemos que las aplicaciones nativas, es decir, aquellas que instalamos en nuestros dispositivos, tienen una media de duración para cada visita de usuario mucho mayor que las aplicaciones Web.

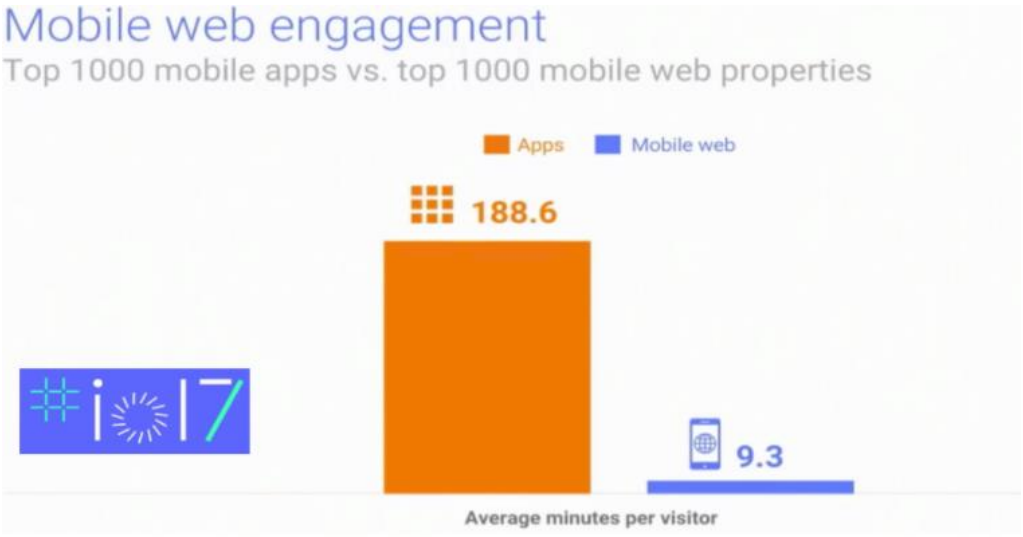

Ilustración 13. Gráfico de la media de duración para las visitas entre las aplicaciones móviles y las aplicaciones web

Esto se debe principalmente a las notificaciones *push*, pero también se debe a otros factores como por ejemplo la posibilidad de utilizarla sin conexión, la seguridad que implican, el buen diseño y las animaciones que estas tienen o las

actualizaciones continuas de las mismas. Sin embargo, esto llevado a la práctica resulta no ser tan idílico como parece.

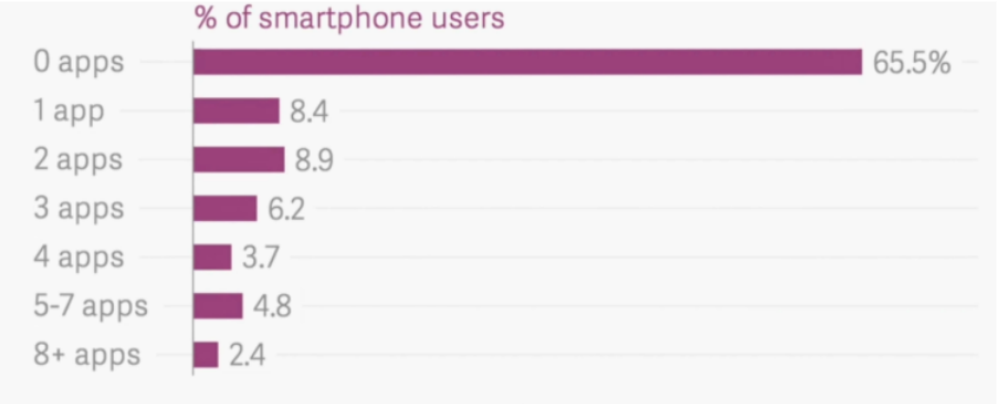

Ilustración 14. Gráfico de la media de aplicaciones instaladas por usuario al mes en US

Con la ilustración 14 se puede observar que la gran mayoría de personas al final de mes no ha instalado ninguna aplicación nueva en su dispositivo, sino que principalmente instalan aquellas aplicaciones que utilizan en su día a día, tales como por ejemplo WhatsApp, Twitter, Facebook etc…

Por regla general, un usuario va a buscar una aplicación nueva, normalmente se busca primero en Google antes de acceder a dicha aplicación, es aquí donde entra en juego el papel de las PWA, de tal forma que al acceder a una web que es una PWA, esta le dará la opción de poder añadirla al dispositivo, **aumentando así los accesos y la regularidad del usuario**

Con este simple gesto, en cuestión de unos pocos segundos ya estaría instalada nuestra PWA en el dispositivo, actuando de igual forma que una aplicación nativa.

Normalmente una aplicación web tradicional no es lo suficientemente atractiva como para que los usuarios regresen a ella en un dispositivo móvil. La principal causa de esto es que no pueden acceder a las funcionalidades nativas del dispositivo y además requieren siempre una continua conexión a internet.

Por tanto se puede concluir que las PWA combinan todas las principales características que hemos comentado favorables tanto a las aplicaciones nativas

como a las aplicaciones web, y lo mejor de todo es que no hay que volver a crear toda la aplicación de nuevo para que estas se conviertan en una PWA.

5.6. Alternativas a las PWA: Aplicaciones híbridas

Aunque las dos principales formas que hemos tratado en este documento de crear aplicaciones en diferentes plataformas son las PWA y las aplicaciones nativas, no significa que no haya otras. Recientemente han salido novedosas tecnologías como *Native Scripts*, *Flutter*, *React Native* o *Ionic* capaces de implementar aplicaciones con un comportamiento muy similar al que ofrece una aplicación nativa pero el código fuente solamente se realiza una única vez y es válido para los 3 tipos de dispositivos que encontramos en la actualidad (Windows Phone, iOs, Android), este tipo de aplicaciones reciben el nombre de **aplicaciones híbridas** [12]**.**

En este tipo de aplicaciones una vez está totalmente desarrollada ya se encontraría disponible y totalmente funcional en los tres sistemas operativos, mientras que para realizar aplicaciones nativas es necesario realizar 3 códigos independientes como por ejemplo utilizar java para Android, Objetive-C para iOs o C# para Windows Phone. Está claro por tanto que con el enfoque que brinda las aplicaciones híbridas ahorramos de considerable en tiempo y esfuerzo.

Una aplicación híbrida bien diseñada y optimizada no debe diferir mucho de la aplicación nativa, pero sin embargo para proyectos más complejos, como juegos o aplicaciones avanzadas, pueden ralentizarse significativamente. Un problema que encontramos en este tipo de aplicaciones es que si buscamos utilizar algunas funciones específicas de Android o iPhone resultaría complejo hibridar dichas funcionalidades.

Las aplicaciones híbridas son una buena opción, especialmente si el proyecto es relativamente pequeño y su presupuesto es limitado; de lo contrario, será más interesante realizar aplicaciones nativas ya que al estar codificadas en lenguajes para esos sistemas operativos en concreto proporciona un mejor rendimiento y compatibilidad con las funciones de los diferentes dispositivos móviles, lo que

implica que el desarrollo también requiere más tiempo y dinero. Por supuesto, es más difícil instalar y actualizar estas aplicaciones nativas.

Estas aplicaciones se desarrollan por norma general con lenguajes que proporcionan un marco de trabajo (*framework*) para personalizar las diferentes vistas en *Tablet* o *Smartphone*. De esta manera, se puede personalizar una aplicación híbrida en plataformas móviles sin escribir código nuevo, pero debe adaptarse a cada sistema operativo.

Si recurrimos a los conceptos de versatilidad y polivalencia, las aplicaciones híbridas tienen una gran cantidad de ventajas frente a otras aplicaciones como las aplicaciones Web o las nativas, entre ellas podemos destacar [11]:

- Ahorro en costes y en tiempo
- Capacidad de implementar aplicaciones con las mismas características, funcionalidades y calidad que las aplicaciones nativas
- No es necesario el uso de un navegador al contar con un acceso directo de la aplicación
- Se cuenta con un grado bastante elevado de integración entre el software desarrollado y el hardware del dispositivo donde se instale

Por otro lado si nos basamos en términos de usabilidad, ambas son bastante similares, pero el rendimiento de la aplicación híbrida es significativamente menor que la aplicación nativa porque las aplicaciones nativas aprovechan mucho mejor los recursos hardware que ofrece el dispositivo, al aprovechar recursos como cámara, sensores específicos del dispositivo, GPS, etc. Sin embargo, las aplicaciones híbridas también pueden usar estas funciones de hardware, pero no de forma tan óptima y en profundidad como lo pueden hacer las nativas

A pesar de estas diferencias es difícil distinguir entre aplicaciones híbridas y nativas. Una forma de visualizar esto es comparar el diseño visual en diferentes dispositivos con diferentes sistemas operativos. Si el comportamiento y la estructura son similares visualmente, se puede decir que muy probablemente es una aplicación híbrida, pero si los elementos están distribuidos en diferentes lugares y tienen diferentes características, estaremos ante una aplicación nativa.

Las PWA son la última tendencia en el mundo de las tecnologías de la información y pueden llegar a reemplazar a las aplicaciones híbridas e incluso a las aplicaciones nativas. A comparación con las aplicaciones híbridas, son mucho más rápidas y utilizan, a diferencia de las aplicaciones nativas, el mismo código fuente. Otra ventaja importante de las PWA es que reducen significativamente los costos de desarrollo y el tiempo de comercialización. En resumen, se puede decir que las PWA son una combinación entre de las aplicaciones nativas y las aplicaciones híbridas.

5.7. Ventajas e Inconvenientes de las PWA

A continuación, identificamos las ventajas e inconvenientes del uso de PWA, las cuales describiremos con más detalle posteriormente [13, 14].

5.7.1. Ventajas

Rápidas: Con las PWA tanto el tiempo de carga como el de navegación disminuyendo provocando así una mejor experiencia de usuario

Coste bajo: El desarrollo, el mantenimiento y las actualizaciones son más baratos que las aplicaciones nativas dado que son más sencillas de programar y se adapta a todo tipo de dispositivos y navegadores.

Indexables: Al encontrarse en la Web los motores de búsqueda la detectan, mejorando así el posicionamiento Web.

Parecida a una App Nativa: En cuanto a la interfaz se refiere es similar a una aplicación nativa, lo que le da al usuario confianza. Como se mencionó anteriormente, también puede usar varias funciones del dispositivo y las funcionalidades de este.

Actualizadas: Esto les da a todos los usuarios acceso a la última versión de la aplicación sin tener que descargar nada.

Funcionalidad sin conexión: Pueden utilizar el contenido recopilado en la memoria caché en situaciones donde la conexión es débil o directamente sin ella.

Independiente de las tiendas de aplicaciones: las PWA son accesibles a través de enlaces, búsqueda Web o incluso mediante QR, es por ello que no es necesario desplegarla en tiendas de aplicaciones, ahorrando tiempo y costes.

Enlazables: En la línea de lo anterior al ser accesibles mediante enlaces, se pueden compartir de forma muy sencilla y rápida con otros usuarios.

Accesibles desde la pantalla de inicio: Consiguiendo así una visibilidad buena para el usuario que le permitirá acceder a ella de forma sencilla y rápida sin la necesidad de pasar por *markets* de aplicaciones que impliquen una instalación que requiera más almacenamiento y una inversión de tiempo mayor por parte del usuario final.

Uso de menos recursos: Las PWA requieren menos almacenamiento que las aplicaciones nativas, a pesar de que estéticamente sean muy similares. Por esta razón, el uso de estos programas elimina una desventaja muy significativa para muchos usuarios.

Uso de notificaciones: Dan la posibilidad de enviar notificaciones *push*. Esta funcionalidad provoca que los usuarios entren en la aplicación con más facilidad lo que deriva en una mejor comunicación, fidelización del usuario, y un mayor *engagement.*

Seguras: Utilizan protocolos seguros como HTTPS o TLS.

***Responsive*:** El diseño de las PWA se adapta a cualquier dispositivo o navegador, lo cual es algo fundamental hoy en día en el mundo de las aplicaciones.

Además de las ventajas mencionadas se da una serie de ventajas para todos los actores involucrados en el desarrollo de una PWA (actores, usuarios y desarrolladores)

5.7.2. Desventajas

Consumo mayor de batería: Debido a que el código utilizado es un código de Internet, consume más batería que la que consume por regla general una aplicación nativa que utiliza un código diseñado específicamente para su dispositivo. Este punto es una desventaja grave que puede marcar que el usuario mantenga la aplicación en su dispositivo o la desinstale.

Más limitadas que las Apps Nativas: Las aplicaciones web progresivas pueden usar algunas características del dispositivo, lo cual es un aspecto positivo. Sin embargo, si la funcionalidad de la aplicación implica el uso de demasiadas tareas nativas, el uso de PWA en algunas aplicaciones para este propósito es limitado. Si está estrechamente relacionado con el uso de funciones como NFC, GPS o telefonía, se debe abandonar esta opción y evaluar la opción de diseñar una aplicación nativa.

No están en los mercados de aplicaciones: El hecho de que los motores de búsqueda encuentren PWA ha demostrado ser bueno para un gran número de usuarios. Pero muchos buscan las aplicaciones en la tienda de aplicaciones de su sistema operativo. Cuando un usuario busca en Internet, es posible que no desee agregar algo a su teléfono, sino que con la Web ya le satisfaga sus necesidades. Además al tratarse de una tecnología muy reciente, los usuarios no están muy familiarizados con las PWA y pueden preferir a las aplicaciones habituales en algunos casos.

Tendencia aún no consolidada en el público: A mucha gente le resulta difícil aceptar cambios. No todo el mundo quiere probar nuevas opciones. Además, la ignorancia general de estas tecnologías junto al no conocer las ventajas y la utilidad de este tipo de aplicaciones puede provocar que no se le dé una oportunidad a las PWA. Por tanto es aconsejable informar al usuario de lo que ofrece una Aplicación Web Progresiva antes de ofrecer la opción de instalarla.

5.8. ¿Cómo funciona una PWA?

Entre los elementos fundamentales dentro de una PWA destacan: el *service worker*, el manifiesto de la aplicación, protocolo de seguridad HTTPS [15, 16, 17]

5.8.1. Service Worker

El *Service Worker* es un script que se ejecuta en segundo plano independiente del navegador y que da acceso a funcionalidades que no requieren un sitio web o la interacción del usuario como pueden ser las notificaciones *push* o la sincronización en segundo plano. Una de las funcionalidades que hacen tan interesante al *Service Worker* es la posibilidad de otorgar un mayor control a los desarrolladores al permitir algunas experiencias **sin la necesidad de conexión a internet** [19].

El *Service Worker* cuenta con un ciclo de vida independiente del sitio web y este dura mientras se requieran los servicios que este ofrece.

Para poder ser instalado en primer lugar debe de ser registrado, lo cual se lleva a cabo por medio del JavaScript de la página web. Una vez esté registrado el *service worker*, el navegador dará paso a la instalación de este en segundo plano [18].

Durante la fase de instalación, normalmente es necesario guardar algunos objetos estáticos en la caché y finalmente si estos archivos se guardan correctamente, el *service worker* estará instalado. Si esos archivos no se descargan ni se guardan en la caché, el proceso de instalación abortará y el *service worker* no se activará. Si la instalación ha fallado se intentará de nuevo la próxima vez que se acceda. En cambio si la instalación ha sido satisfactoria, los archivos estáticos descargados se mantendrán en caché y no será necesario volver a instalar el *service worker* la próxima vez que se acceda.

La siguiente fase en caso de que la instalación haya sido exitosa sería la activación. Es el momento en el que el *service worker* empieza a ejecutar sus funciones, además es en esta fase el momento idóneo para gestionar las cachés en las que se han guardado ciertos elementos en la fase anterior de instalación.

Una vez el *service worker* esté activado, éste comprueba todas las páginas que se encuentren dentro de su alcance. Sin embargo no toma el control de la página en

la cual se registró el *service worker* hasta que esta se vuelva a recargar. Cuando el *service worker* controla la página, éste puede tomar dos posibles acciones; o bien ahorra memoria o bien entra al modo de procesamiento de mensajes que consiste en el control y extracción de aquellos eventos de mensaje después de recibir un mensaje o una solicitud de red.

En la ilustración 15 de puede observar un breve resumen gráfico acerca del ciclo de vida que los *service worker* siguen al instalarse.

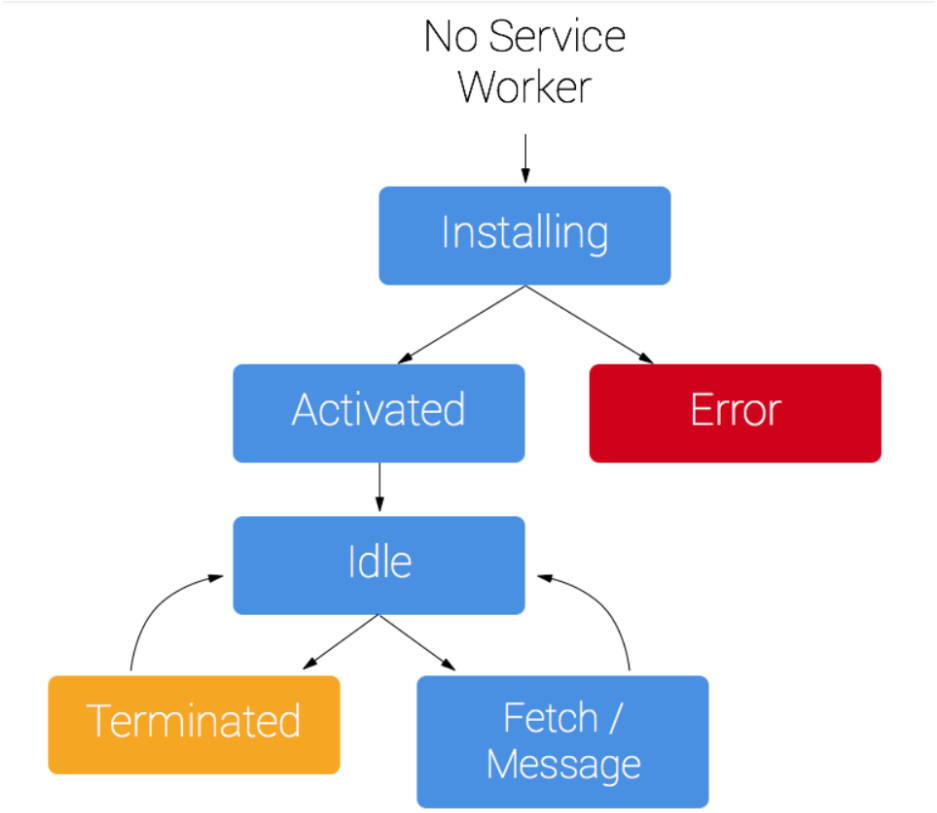

Ilustración 15. Ciclo de vida de un Service Worker

5.8.2. Manifiesto de la aplicación

El manifiesto de la aplicación es uno de las partes más importantes que componen una PWA y este se define en formato ".json". El manifiesto nos permite definir aspectos visuales y básicos de la aplicación, entre ellos podemos destacar [15]:

Name: Nombre con el que se identificará la aplicación. Corresponderá también al nombre que aparecerá debajo del icono una vez esté instalada la aplicación.

Description: No es un parámetro visual como tal pero es adecuado añadir una pequeña descripción de la aplicación.

Icons: Definir iconos en diferentes tamaños para una correcta visualización en todos los tipos de dispositivos.

Start url: Se trata de la dirección URL inicial de la aplicación.

Orientation: Corresponde al modo de orientación en el que se ejecutará la aplicación, es decir, si queremos que la aplicación sea vertical u horizontal.

Display: Diferentes tipos de visualizaciones, el desarrollador tiene varias opciones a elegir como por ejemplo: *standalone*, *fullscreen*, *minimal-ui* etc.

Background_color: Color de fondo de la pantalla de carga inicial de la aplicación

Theme_color: Se trata del color predominante de la aplicación, este parámetro define entre otros el color de la barra superior de la aplicación

5.8.3. HTTPS

Si estamos desarrollando una PWA el *service worker* se podrá utilizar con normalidad a través de un servidor local (*localhost*), sin embargo para poder desplegar una PWA en un servidor en internet a nivel global será necesario el protocolo a seguir sea el protocolo HTTPS.

HTTPS (*HyperText Transfer Protocol Secure*) se trata de un protocolo de conexión a Internet que protege la privacidad e integridad de los datos del usuario entre la plataforma web y el equipo. Este protocolo se encuentra protegido a su vez con el protocolo criptográfico SSL y más reciente TLS para codificar toda la información que se pase [20].

Estos protocolos utilizan una encriptación simétrica a través de una clave que se genera para cada sesión. Sin embargo el intercambio de esa clave, se lleva a cabo mediante clave pública y clave privada, es decir, mediante lo que se conoce como encriptación asimétrica. En la ilustración 16 se puede observar un breve esquema conceptual del funcionamiento del protocolo HTTPS

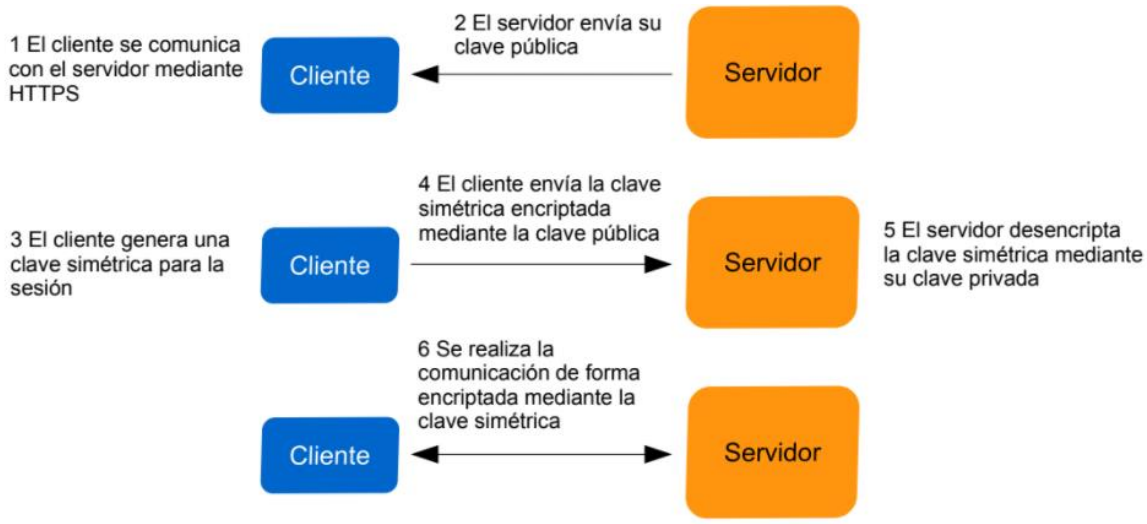

Ilustración 16. Esquema del funcionamiento del protocolo HTTPS

Con el uso de este protocolo conseguimos que nuestros datos estén a salvos durante el tráfico de estos, impidiendo así que cualquier agente que busque interceptarnos esos datos puede hacerse con ellos.

Se puede utilizar el *service worker* para manejar una conexión en concreto y con ello filtrar o crear nuevas respuestas, por tanto no cabe duda de que se trata de elemento poderoso que en las manos equivocadas puede suponer un peligro, es por ello que, solo puede *service workers* en sitios web compatibles con HTTPS anteriormente detallado, por tanto los *service worker* que se registran en el navegador no se ven comprometidos cuando estos viajan por la red.

5.8.4. Soporte para navegadores

Ya hemos expuesto muchas de las ventajas que ofrece las PWA, así como los requisitos para poder implementarlas y desplegarlas, sin embargo un requisito fundamental al tratarse de una nueva tecnología es la de los navegadores al soportar estas tecnologías, es por ello que en este apartado trataremos diferentes funcionalidades de las PWA y mostraremos una ilustración que nos muestre la compatibilidad entre los principales navegadores en los diferentes sistemas operativos y esa funcionalidad en concreto. Recalcar que los datos de las ilustraciones que se mostrarán a continuación son recogidos del año 2018, por lo que está sujeto a cambios en la actualidad.

5.8.4.1. Experiencia del usuario mejorada y funcionamiento offline

La característica más destacada que distinguimos entre las PWA y una aplicación Web tradicional es la experiencia de usuario que las aplicaciones progresivas ofrecen

Gracias al *Service worker* los desarrolladores de aplicaciones web progresivas son capaces de implementar funcionalidades especiales más avanzadas. Como ya se ha mencionado, el *service worker* se instala en el navegador y una vez esté instalado se ejecuta en segundo plano. A continuación, en la ilustración 17 se puede observar una tabla con las diferentes compatibilidades entre los navegadores y los *service workers* [21].

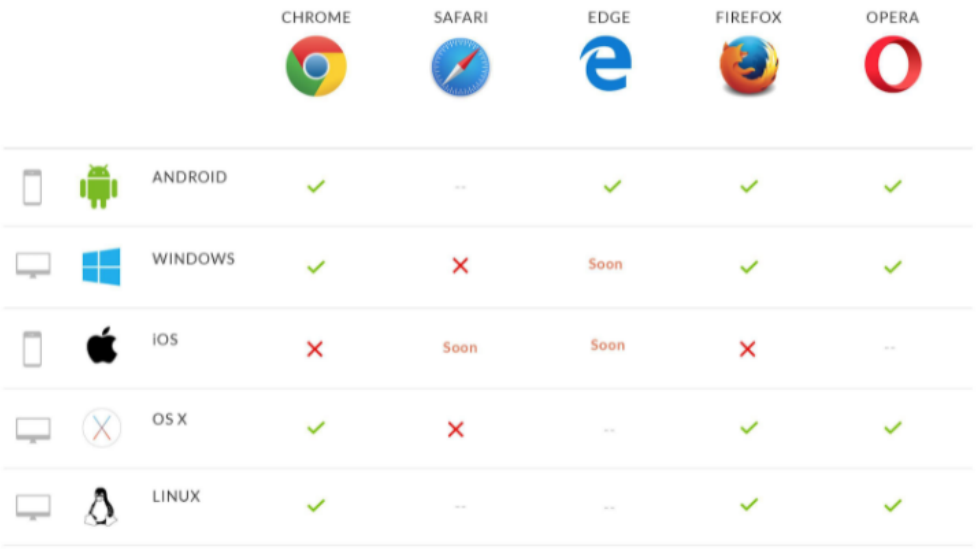

Ilustración 17. Compatibilidad de los principales navegadores en los diferentes sistemas operativos con los Service Workers

Una de las funcionalidades avanzadas que comentábamos es la sincronización. Esto lo que permite es que el usuario pueda visualizar los datos directamente cuando llegan gracias a que continuamente se están realizando actualizaciones en busca de nuevos datos a mostrar o procesar. De esta forma el usuario percibe y disfruta de una mejor experiencia de usuario al ganar bastante fluidez la aplicación y con ellos la velocidad de ejecución de la misma.

Para la funcionalidad de poder realizar algunas funciones sin necesidad de conexión a internet también se requiere el *service worker*. Éste una vez instalado, pasa a gestionar ciertos ficheros propios de la aplicación en la caché y este intercepta las solicitudes de Red siendo capaz de detectar si la red se encuentra activa o no y actuar en función de esta disponibilidad. Es así como la experiencia del usuario que esté utilizando la aplicación se ve incrementada nuevamente al poder acceder a cierto contenido sin necesidad de conexión ya que habrá algunos ficheros que no tengan que ser solicitados al servidor web, sino que ya se encontrarán descargados gracias al *service worker*. En la ilustración 18 de puede observar qué navegadores de los más utilizados son compatibles con esta interesante funcionalidad.

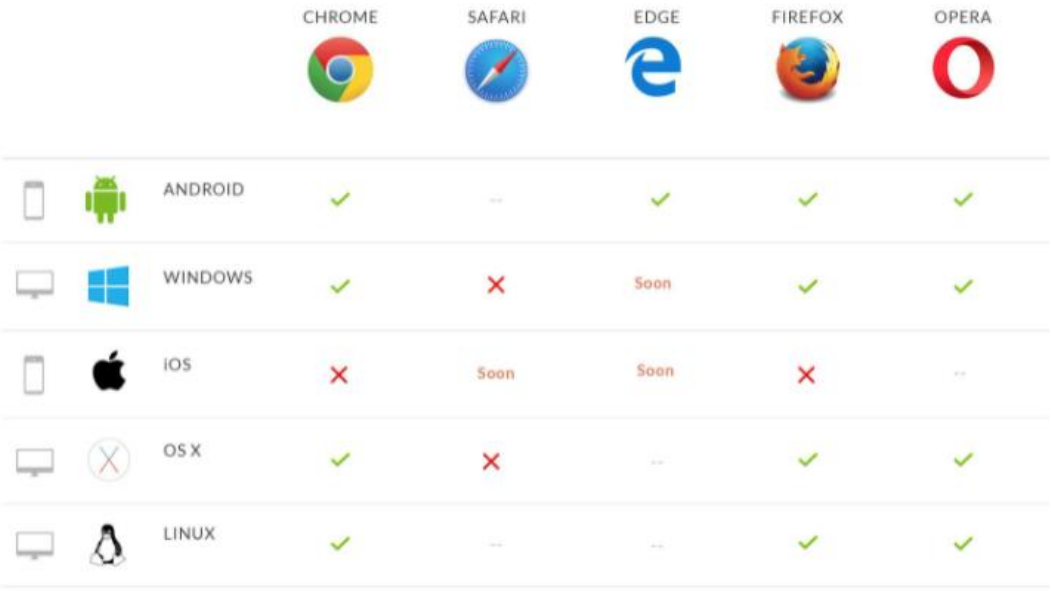

Ilustración 18. Compatibilidad de los principales navegadores en los diferentes sistemas operativos con el funcionamiento sin conexión

5.8.4.2. Notificaciones Push

Esta es una de las características más interesantes que nos ofrece las PWA, gracias también al *service worker*. Para esta funcionalidad se requiere el uso de dos APIs que funcionarán de forma complementaría entre ellas.

Por un lado encontramos la *API Push* que se encarga de notificar al *service worker* de la existencia de una notificación nueva. Por otro lado tenemos la *API Notification* que será la responsable de la activación de la visualización de dicha notificación en el navegador donde se esté ejecutando.

En la ilustración 19 podemos observar la compatibilidad de los principales navegadores para esta funcionalidad de gestión y visualización de notificaciones *push*. Cabe destacar que todos los navegadores salvo el navegador nativo de Apple se basan en los *service workers* para la implementación de esta funcionalidad. Éste utiliza un sistema de gestión desarrollado por Apple que recibe el nombre de APNs (Apple *Push Notification Service*). Este sistema permite la gestión y recepción de notificaciones en OS X, sin embargo no lo permite para el sistema operativo iOS

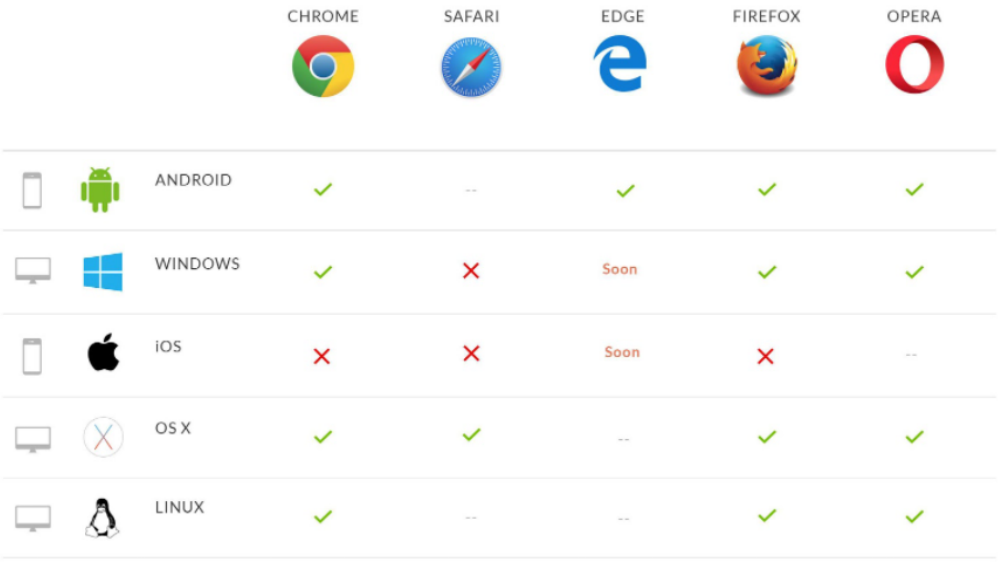

Ilustración 19. Compatibilidad de los principales navegadores en los diferentes sistemas operativos con las notificaciones Push

5.8.4.3. HomeScreen y SplashScreen

Una de las funcionalidades que permiten las Aplicaciones Web Progresivas es, que a pesar de que se ejecuten en un navegador, estas pueden estar instaladas y encontrarse accesibles desde el escritorio del dispositivo donde se encuentre instalada, provocando que al abrirla desde ahí se ejecute en una ventana completa propia de la aplicación dando al usuario una experiencia satisfactoria.

Estas funcionalidades son posibles principalmente por el manifiesto de la aplicación, donde los desarrolladores de la aplicación describen cómo se va a comportar la PWA. Este se encargará de guardar la aplicación y añadirla en la pantalla de inicio desde el navegador. Finalmente en la pantalla de inicio se mostrará tanto el icono como el nombre de la aplicación que hayamos definido previamente en el manifiesto.

Con el manifiesto también se habilita una pantalla de inicio de la aplicación que el usuario visualizará cuando ésta se ejecute y tenga todos los datos necesarios, bien a partir de la caché o descargarlos.

A continuación en la ilustración 20 se refleja las compatibilidades entre navegadores y esta funcionalidad.

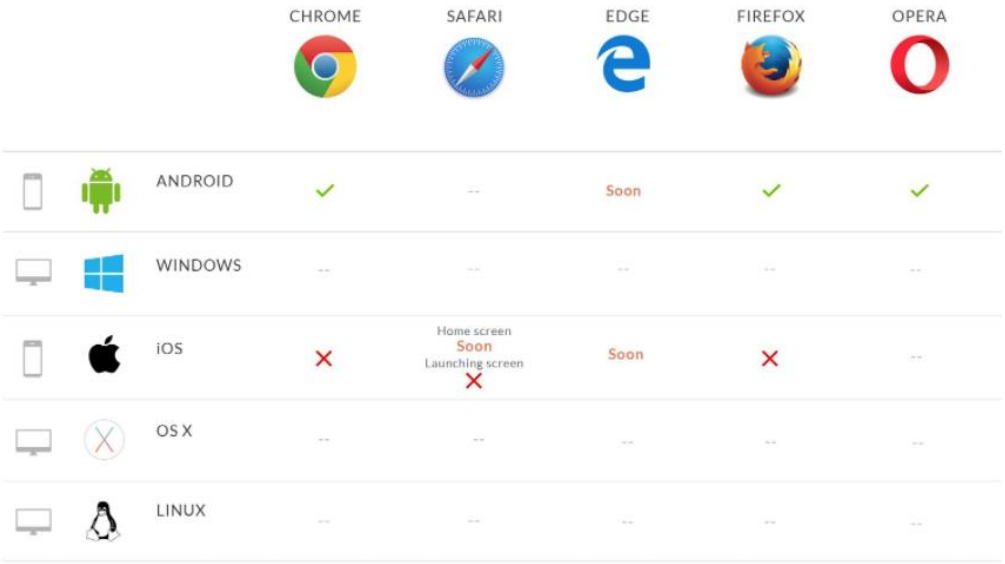

Ilustración 20. Compatibilidad de los principales navegadores en los diferentes sistemas operativos con las HomeScreens y SplashScreens

5.8.4.4. Geolocalización

Esta funcionalidad se lleva a cabo por medio de la API *Geolocation*, la cual se encarga, siempre con el consentimiento del usuario de la aplicación, de recuperar la geolocalización del usuario e identificarla, así pues es capaz de llevar un seguimiento de las localizaciones de éste, incluso de notificarle en el momento en la que la posición del individuo cambie.

Para el correcto uso de la API, se requiere el protocolo HTTPS para obtener estos datos de una forma segura y que no se pueda interceptar por individuos maliciosos. Debido a esta funcionalidad ahora es posible llevar a cabo estrategias de fidelización y compromiso que nunca antes se había concebido en el ámbito Web.

Tal y como se puede observar en la posterior ilustración 21, para esta interesante funcionalidad los navegadores más conocido y utilizados actualmente son compatibles con ella, por lo que podemos concluir que los desarrolladores de los navegadores, van siendo conscientes de que las PWA, son sin duda, una tendencia en pleno auge de crecimiento.

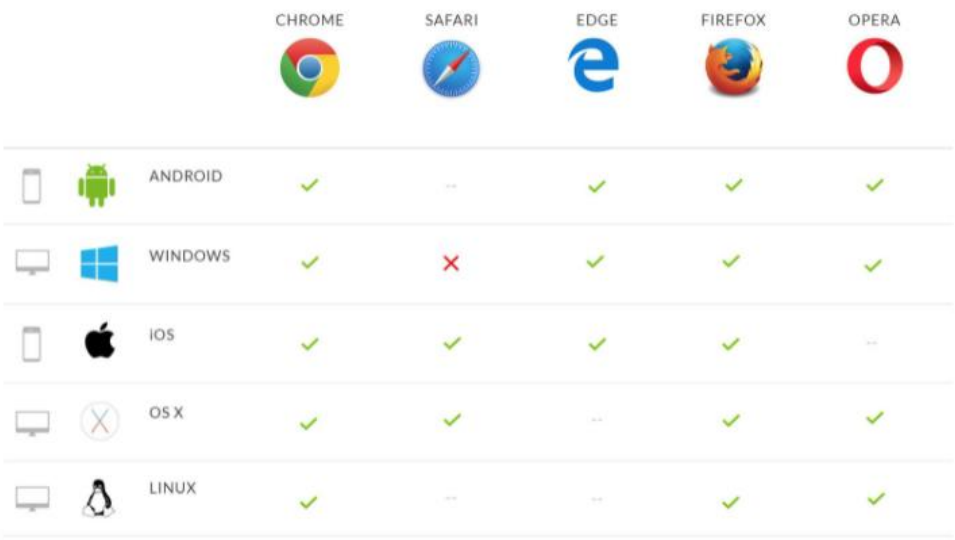

Ilustración 21. Compatibilidad de los principales navegadores en los diferentes sistemas operativos con la Geolocalización

5.8.4.5. Vídeo y captura de imagen

Hasta hace poco, solamente se utilizaba algún elemento HTML para lanzar aplicaciones proveedoras de imágenes como puede ser la cámara del dispositivo con el fin de tratar con imágenes y vídeos. Sin embargo con la API *Media Capture*, es posible que las aplicaciones web tengan directamente acceso a los medios de entrada de audio y video del dispositivo en concreto. A través de ella ya una aplicación web es capaz de tomar una foto o vídeo sin necesidad de salir del propio navegador.

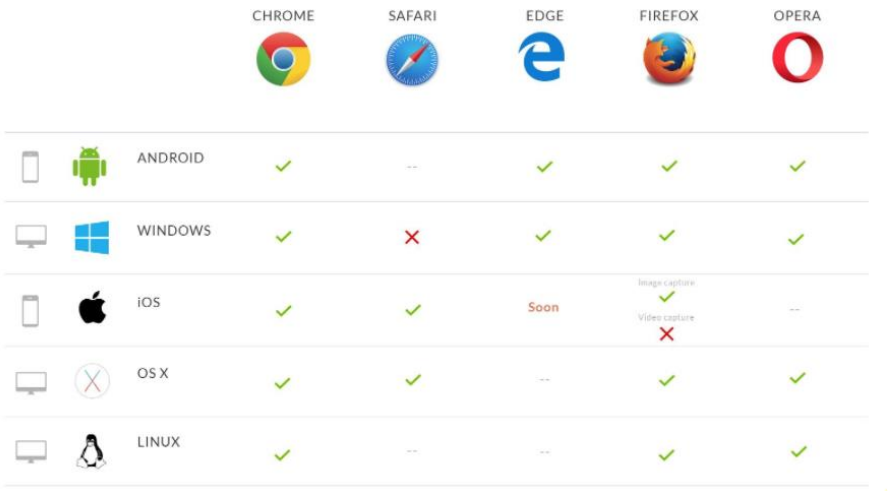

Ilustración 22. Compatibilidad de los principales navegadores en los diferentes sistemas operativos con las capturas de imágenes y vídeos

5.8.4.6. Bluetooth

Anteriormente la única forma de conectar una aplicación con los dispositivos Bluetooth era por medio de una aplicación nativa. Esto ha cambiado en los últimos años con la aparición de la API Web Bluetooth, que ofrece la posibilidad de establecer una comunicación entre la aplicación web y dichos dispositivos Bluetooth de forma segura.

En cuanto al funcionamiento la aplicación web solicita al navegador una lista de los dispositivos Bluetooth dentro del alcance de este, esa lista se le muestra al usuario y éste elige el dispositivo con el que le gustaría conectarse; o en caso de no querer conectarse a ninguno, simplemente cierra la lista de dispositivos disponibles. Nuevamente para disponer de esta funcionalidad, la API requiere el uso del protocolo HTTPS, para de esta forma evitar riesgos en cuanto a seguridad se refiere.

En la ilustración 23 se mostrará el listado de navegadores y la compatibilidad entre ellos y esta funcionalidad.

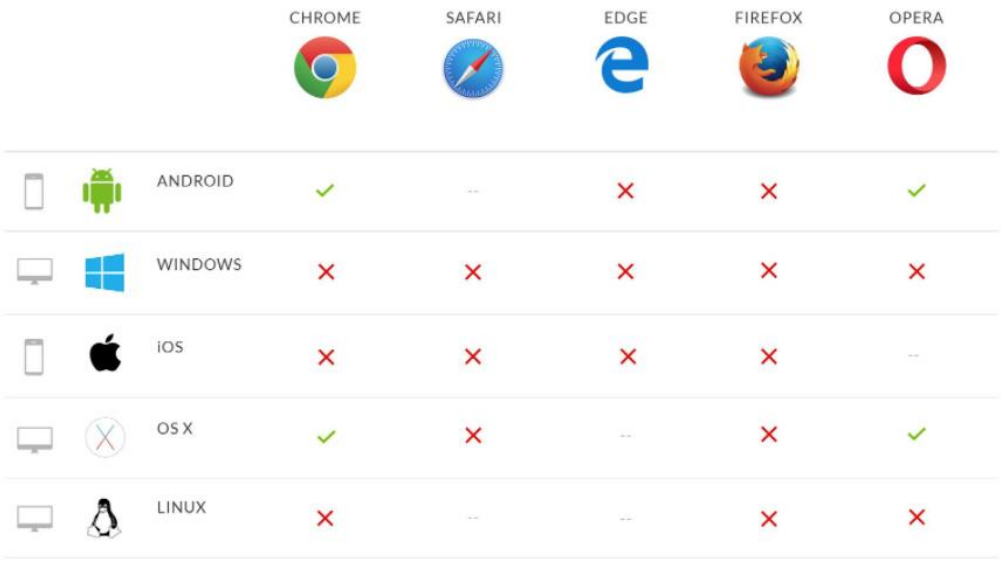

Ilustración 23. Compatibilidad de los principales navegadores en los diferentes sistemas operativos con el uso de Bluetooth

5.9. El futuro de las PWA

Las herramientas y métodos existentes para el desarrollo web aún no son compatibles con las PWA, no obstante muchos sistemas y servicios de desarrollo están empezando a permitir el desarrollo de aplicaciones avanzadas. En concreto, el sistema del lado del cliente con diseño de servidor personalizado funciona bien con el segundo modelo para el enrutamiento de carga en el lado del cliente que las PWAs adoptan debido a la implementación de experiencias sin conexión.

El área de las aplicaciones progresivas es un área en la que el diseño y la construcción de éstas ofrecerán una gran ventaja a quien apueste por ellas. Con el desarrollo de aplicaciones progresivas se ofrece a los usuarios mejores experiencias en todos los dispositivos y contextos sin necesidad de realizar códigos independientes, sino que se encuentra todo en el mismo código fuente. Sin embargo para poder consolidar el diseño de estas aplicaciones se requiere un cambio considerable tanto en las herramientas como en la forma de entender las aplicaciones web de hoy en día.

Para acceder fácilmente a la PWA, ésta una vez esté instalada se añade a la página de inicio del dispositivo. Además de la velocidad y la adaptabilidad a diferentes dispositivos, las aplicaciones progresivas aumentan por regla general el tiempo que el usuario pasa dentro del sitio web así como la tasa de conversión en tiendas online.

Por ello las aplicaciones progresivas pueden llegar a ser el futuro de las aplicaciones para los dispositivos móviles. Además el tiempo que se tarda en desarrollar una PWA es notablemente inferior al de una aplicación nativa y por tanto el mantenimiento de la misma también se ve reducido.

Una vez llegado a este punto, nos surge la pregunta de **¿Las PWA pueden sustituir a las aplicaciones nativas?** La respuesta sería que al menos de momento no. Esto se debe a que las aplicaciones progresivas no están muy extendidas como para poder sustituir a las aplicaciones nativas que llevan entre nosotros muchos años. Las PWA actualmente se enfocan principalmente en replicar sitios ya

existentes, por lo que no pueden reemplazar a las aplicaciones nativas que se construyen desde cero.

No obstante, con los beneficios que las aplicaciones progresivas ofrecen en cuanto a rendimiento, compatibilidad, sencillez a la hora de compartir, notificaciones *push* o la rápida instalación sin ser necesario descargarlas y sin ocupar prácticamente nada de almacenamiento del dispositivo; podemos decir que a las PWA les espera un buen futuro en el momento en el que la mayoría de sitios web desarrollen su propia aplicación progresiva y los usuarios poco a poco las vayan utilizando en mayor medida.

6. Ejemplo de desarrollo de una PWA

6.1. Proceso de Diseño

El diseño de las historias de usuario con las que contará la aplicación se llevará a cabo por medio de la principal herramienta que hemos utilizado a lo largo del grado en diferentes asignaturas de diseño software para realizar este tipo de actividades, se trata de **Visual Paradigm**. Este software nos permite realizar una gran variedad de diagramas, pudiendo especificar y modelar a nuestro gusto el diseño que vayamos haciendo de nuestro proyecto.

6.1.1. Casos de uso

En primer lugar, trasladaremos las historias de usuario a un diagrama de casos de uso (ilustración 24), en el cual se podrá observar de una forma más visual lo que podrá realizar cada uno de los diferentes tipos de usuarios que habrá en el sistema.

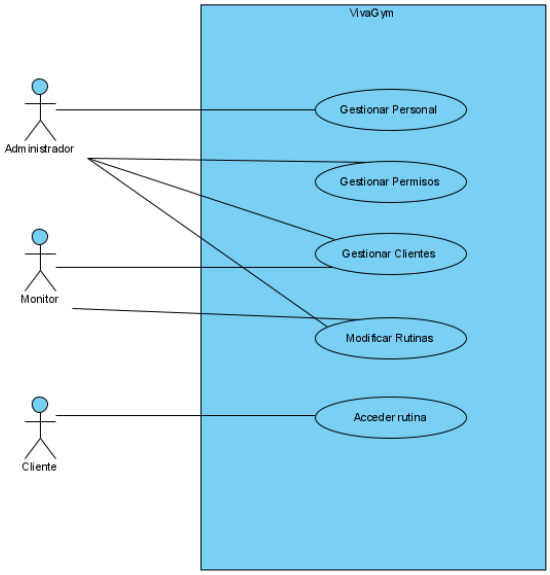

Ilustración 24. Diagrama de Casos de Uso inicial de la Aplicación

6.1.2. Modelo de Dominio

Una vez tengamos definido los casos de uso pasaremos a modelar el modelo de dominio, es decir una primera concepción que se da durante el análisis del problema al que nos enfrentamos, esta representación la vemos en la ilustración 25.

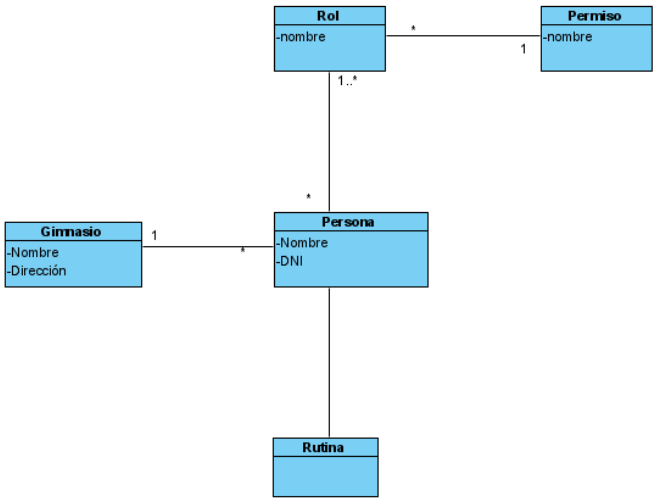

Ilustración 25. Modelo de Dominio inicial de la Aplicación

6.2. Metodología de Trabajo

La metodología de trabajo que se llevará a cabo en este proyecto será por medio de metodologías ágiles, en concreto nos basaremos en las bases de **KANBAN**, ya que esta metodología ágil se adapta bastante bien a las necesidades que en mi caso requiero para un desarrollo de una aplicación gestionada con metodologías ágiles. Normalmente las metodologías ágiles están preparadas para grupos de trabajo de en torno 4-10 personas, sin embargo, al ser un Trabajo de Fin de Grado, no existen estos equipos de trabajo, por lo que el uso la mayoría de estas metodologías ágiles no son la mejor opción. Sin embargo Kanban sí que es una metodología ágil válida para este tipo de situaciones ya que es una técnica sencilla y eficaz para lo que vamos buscando.

En mi caso he decidido emplear metodologías ágiles en lugar de la metodología clásica debido a que con el método clásico, se puede llegar a perder demasiado tiempo en la parte del diseño, pudiendo llegar a provocar ciertos retrasos, errores en el propio diseño o cambios en los requisitos que te obligan a provocar un cambio en el diseño. En este aspecto con esta metodología nos podemos permitir especificar aquellos primeros requisitos que consideramos necesarios, realizar el diseño de esos requisitos y a partir de ahí pasar al desarrollo de esos requisitos. Una vez estén desarrollados esos requisitos y nos sobra tiempo, podemos seguir ampliando nuestro ***backlog*** con nuevos requisitos que consideremos interesantes y realizar el mismo proceso, lo que nos aporta un gran valor y evitamos posibles retrasos de nuestro proyecto.

El funcionamiento será el siguiente; en primer lugar seguiremos la priorización **MoSCoW** (*Must-Should-Could-Wont*), que se basa en facilitarnos la labor en la fase de especificación de requisitos, con ello podremos especificar todas aquellas Historias de Usuario que veamos conveniente en función de su prioridad, distribuyéndolas desde aquellas que son obligatorias para el correcto funcionamiento de la aplicación, hasta llegar a todas esas Historias de Usuario que se nos ocurra para nuestra aplicación pero que el desarrollo de las mismas no tiene por qué ser "obligatorio" para su funcionamiento, sino que son funcionalidades extras que estarían bien desarrollar pero para que eso ocurra, primeramente deberemos de haber desarrollado todas las funcionalidades cuya prioridad es más alta que esta.

Todas aquellas historias de usuario que se nos ocurran la iremos colocando dentro de nuestro ***backlog***, esto no es más que una lista con todas aquellas funcionalidades que buscamos en nuestra aplicación, independientemente de su prioridad y de si se acaben desarrollando o no.

Del ***backlog*** escogeremos aquellas historias de usuario que estamos convencidos que vamos a implementar y las pasaremos a "**seleccionadas para desarrollo**", es decir, que vamos a incluir en nuestro proyecto, esto no implica que cuando saquemos la siguiente versión de la aplicación se encuentren todas estas implementadas, simplemente nos indica que esas historias de usuario se acabarán implementando tarde o temprano.

De esta columna elegiremos aquellas que estemos implementando en ese momento dado a la columna de "**en desarrollo"**, y permanecerán en esta hasta que estén implementadas. Normalmente en esta sección solo se encontrarán 1 o 2 historias de usuario ya que en un mismo momento dado no estaremos realizando más de 2 historias de forma simultánea.

Por último, una vez finalizadas estas tareas que estamos realizando, se pasarán dichas historias de usuario a la columna de "**Listo**" lo que indicará que esas historias de usuario ya están implementadas y funcionando en la aplicación final, si es cierto que en caso de que no estén 100% funcional, debido a algún fallo o algo similar, podemos utilizar anotaciones indicando qué es lo que nos ocurre.

6.2.1. Incrementos

Como hemos especificado anteriormente, el desarrollo de la aplicación lo gestionaremos por medio de la metodología Kanban, ya que es la que consideramos más conveniente para la realización de una aplicación con un equipo de trabajo formado por un único integrante.

Dividiremos el tiempo de desarrollo en diferentes incrementos, es decir, espacios de tiempo, **irregulares** ya que al utilizar esta metodología ágil, no nos

tenemos que preocupar por ceñirnos a realizar algo funcional en un determinado periodo de tiempo fijado.

Es por ello que gracias a esta libertad que nos ofrece esta metodología, podemos ir desarrollando aquello que sea de más prioridad, sin la preocupación de tener que tenerlo finalizado para ese periodo de tiempo fijado. A continuación se irán exponiendo los diferentes incrementos que se irá realizando para el desarrollo de este proyecto.

A continuación por cada uno de los incrementos en los que se ha dividido el desarrollo de la aplicación se pasará a mostrar el *backlog* del que se parte en dicho incremento, junto con las historias de usuario que se van a implementar en ese incremento y el diagrama entidad/relación que implica el desarrollo de dichas historias de usuario.

6.2.1.1. Incremento 1 (18/02/2021 – 04/03/2021)

6.2.1.1.1. Backlog

Para este incremento partiremos del siguiente estado del *backlog*, tal y como se puede observar en la ilustración 26, se han seleccionado algunas de ellas para desarrollo, sin embargo esto no significa que todas las vayamos a implementar en este incremento.

Ilustración 26. Backlog al inicio del incremento 1.

6.2.1.1.2. Historia de Usuario Implementadas

En este primer incremento se procedió a la creación de nuestro proyecto **Laravel**, además de ellos se realizaron las siguientes Historias de Usuario.

- **COMO administrador y usuario QUIERO contar con una interfaz visual de la aplicación PARA que el uso de la misma sea agradable**
    - Para la realización de esta Historia de Usuario se ha utilizado el *template* AdminLTE 3, un tema de código abierto desarrollado por medio de **Bootstrap**, el *Framework* más conocido de CSS, este tema lo utilizaremos para todo lo relacionado con la parte visual, utilizando tablas, zona de menús, y demás funcionalidades que nos proporciona este conocido tema.
- **COMO administrador QUIERO tener diferentes roles PARA restringir el acceso a las secciones de la aplicación**
    - Una vez tengamos nuestro *template* en funcionamiento se procede a realizar tanto la creación de la tabla de la Base de Datos de nuestra aplicación, como el **CRUD** (*Create-Read-Update-Delete*) para los roles de la aplicación que buscamos poder crear tantos como el administrador desee. Una vez tengamos toda la lógica necesaria y la tabla, se pasará a realizar las vistas que reflejará toda esa lógica y con la que finalmente trabajará el usuario que utilice esta aplicación.

En primer lugar, para empezar con la implementación de la aplicación, es necesario tener una interfaz visual sobre la que poder apoyar el resto de funcionalidades, es por ello que el desarrollo de la historia de usuario que implica la implantación de esta interfaz gráfica es de suma prioridad. De forma adicional el siguiente paso era poder distinguir entre los diferentes roles que tendrá la aplicación, de tal forma que se implementó dicha historia de usuario.

6.2.1.1.3. Esquema de la Base de Datos

En la ilustración 27 se muestra el diagrama entidad relación previo a la implementación de las historias de usuario seleccionadas.

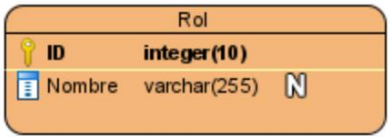

Ilustración 27. Esquema de la Base de Datos del incremento 1.

6.2.1.2. Incremento 2 (04/03/2021 – 18/03/2021)

6.2.1.2.1. Backlog

En este incremento se han introducido nuevas historias de usuario en la selección de historias para desarrollo, y además se han marcado las 2 historias de usuario que se implementaron en el anterior incremento como finalizadas, por tanto para este incremento partiremos del siguiente *backlog* que se muestra en la ilustración 28.

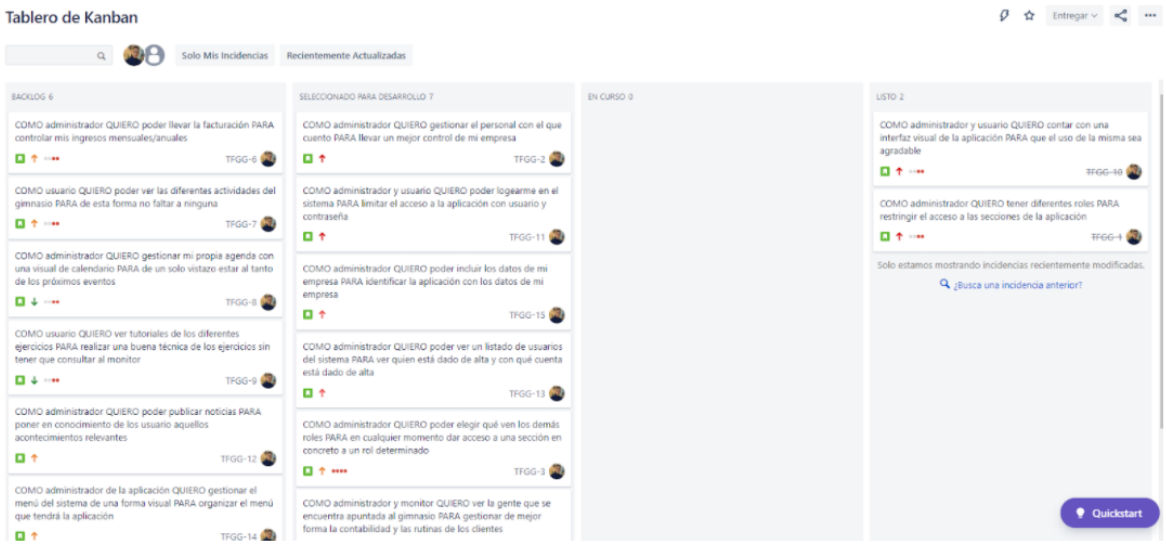

Ilustración 28. Backlog al inicio del incremento 2.

6.2.1.2.2. Historia de Usuario Implementadas

En este incremento se pasó a implementar las siguientes historias de usuario:

- **COMO administrador y usuario QUIERO poder iniciar sesión en el sistema PARA limitar el acceso a la aplicación con usuario y contraseña**
    - Para este punto se utilizará la biblioteca que nos proporciona laravel llamada "**Laravel UI**" por lo que simplemente siguiendo los pasos de instalación podremos contar con el funcionamiento y unas vistas iniciales suficientes para poder desarrollar un *login*, registro y reseteo de contraseña de una forma fácil y no excesivamente compleja. Para finalizar modificaremos la visual de estas vistas, adaptándola a lo que vamos buscando y estaría listo.

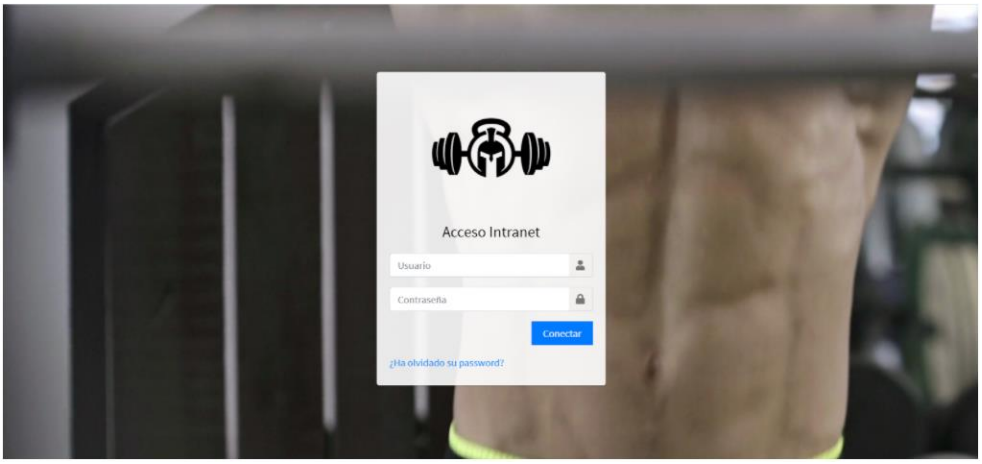

Ilustración 29. Login tras la implementación de la historia de usuario.

- **COMO administrador QUIERO poder incluir los datos de mi empresa PARA identificar la aplicación con los datos de mi empresa**
    - En este caso se pasará a crear la respectiva tabla en nuestra base de datos que será la que se encargará de conservar los datos de la empresa para la que haremos la aplicación. Al tratarse de una empresa única, simplemente tendrá una fila y por tanto solo se podrá crear una única empresa. Constará de un formulario para rellenar los diferentes datos y almacenarlos.

El siguiente paso para seguir con el desarrollo de la aplicación era tanto poder realizar la funcionalidad de inicio de sesión, registro, recuperación de contraseña para que los diferentes usuarios puedan entrar en el sistema por medio de unas credenciales. Además de ello era importante, al tratarse de una aplicación enfocada a un sistema de gestión de una empresa, la recogida de los datos de esa misma empresa, es por ello que se desarrollaron las historias de usuario anteriormente expuestas.

6.2.1.2.3. Esquema de la Base de Datos

En la ilustración 30 se muestra el diagrama entidad relación previo a la implementación de las historias de usuario seleccionadas.

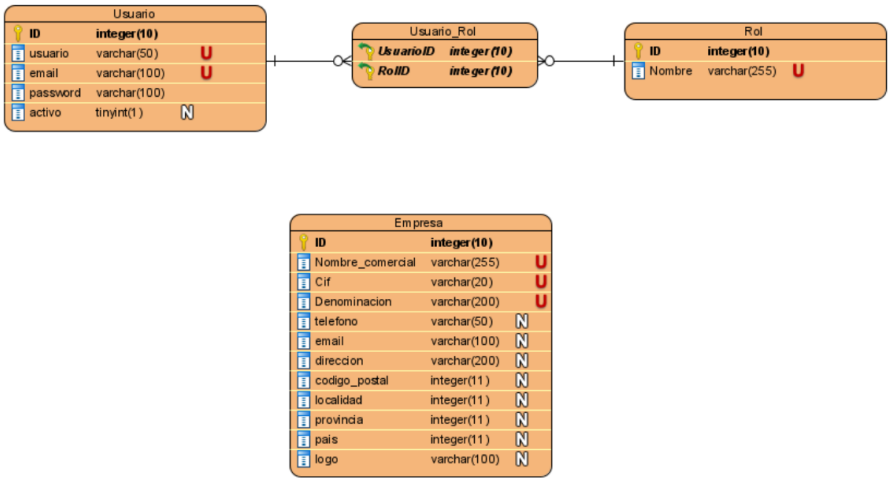

Ilustración 30. Esquema de la Base de Datos del incremento 2.

6.2.1.2.4. Observaciones adicionales del incremento

- **Mejora del código**
  - De forma adicional a las anteriores historias de usuario, se pasó a mejorar el código que ya estaba implementado facilitando sobre todo la inclusión formularios y los diferentes campos de este.

6.2.1.3. Incremento 3 (18/03/2021 – 07/04/2021)

6.2.1.3.1. Backlog

Nuevamente a partir de las historias de usuario implementadas en el anterior incremento, añadimos algunas nuevas para desarrollo y pasamos a implementar algunas de estas que se encuentran en esta sección. Por tanto, el *backlog* del que partimos sería el que se muestra en la ilustración 31:

Ilustración 31. Backlog al inicio del incremento 3.

6.2.1.3.2. Historia de Usuario Implementadas

Con respecto a las historias de usuario que se implementaron en este incremento serían las siguientes:

- **COMO administrador QUIERO poder ver un listado de usuarios del sistema PARA ver quién está dado de alta y con qué cuenta está dado de alta**
    - Para la realización de esta historia de usuario se procederá a la creación de la tabla de la base de datos, así como a la realización de un CRUD completo con las diferentes funcionalidades de creación, actualización, lectura o listado y eliminación de los diferentes usuarios que se crean por medio de la biblioteca que utilizamos en el incremento anterior para la implementación de un registro y un *login*. Además esas operaciones conectarán con las diferentes vistas

necesarias para la visualización de los datos y campos necesarios para las operaciones mencionadas

- **COMO administrador QUIERO poder publicar noticias PARA poner en conocimiento de los usuario aquellos acontecimientos relevantes**
    - De igual forma que en la historia de usuario anterior, habrá que realizar la respectiva tabla en nuestra base de datos donde se almacenarán los diferentes tipos de datos necesarios para confeccionar una noticia con los datos que buscamos. Por otra parte habrá que realizar tanto las operaciones necesarias para listar, guardar, editar y eliminar nuevas noticias como las vistas encargadas de realizar la conexión entre el usuario y nuestro *backend*
- **COMO administrador de la aplicación QUIERO gestionar el menú del sistema de una forma visual PARA organizar el menú que tendrá la aplicación**
    - En primer lugar habrá que crear una visual para poder crear los elementos del menú mediante formularios. Posteriormente, para esta gestión dinámica partiremos de una de las visuales que nos proporciona el tema utilizado, AdminLTE 3, y mediante Ajax haremos ese dinamismo pudiendo personalizar el orden en que se mostrará este menú según nuestras preferencias.

A partir de las historias de usuario implementadas, era necesario dar la posibilidad de listar a los usuarios que se crean en el sistema, así como dar la posibilidad de editar los datos de ese usuario o incluso crearlos desde dentro de la intranet. Por otro lado me parecía bastante interesante la posibilidad de crear noticias que todos los usuarios del sistema puedan ver desde el *feed* principal de la aplicación. Por último, otra funcionalidad importante era la de dar la opción al administrador de gestionar el menú de forma dinámica y visual, de esta forma se pasaron a implementar las historias de usuario anteriores.

6.2.1.3.3. Esquema de la Base de Datos

En la ilustración 32 se muestra el diagrama entidad relación previo a la implementación de las historias de usuario seleccionadas.

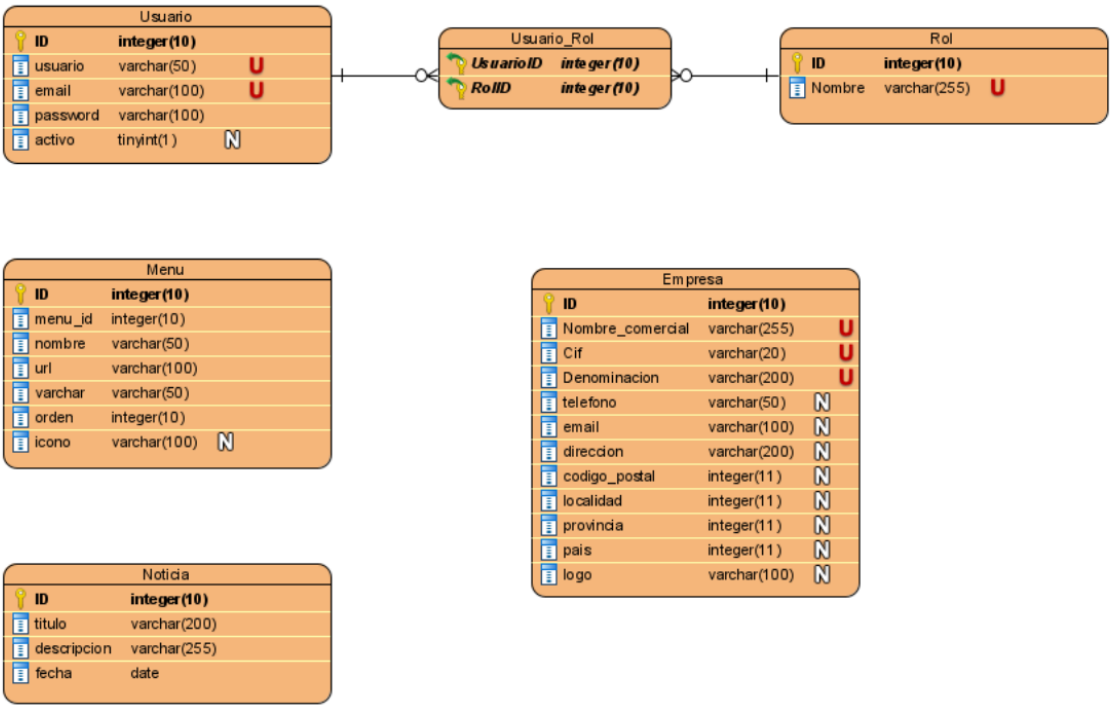

Ilustración 32. Esquema de la Base de Datos del incremento 3.

6.2.1.3.4. Observaciones adicionales del incremento

Además se puede observar la inclusión de una nueva historia de usuario, la cual es bastante interesante, esta corresponde a la historia de usuario:

- **COMO administrador de la aplicación QUIERO poder conectarme con cualquier usuario PARA comprobar si está correcto lo que ve cada tipo de rol de una forma cómoda**
    - Esta historia de usuario consiste en poder acceder a cada una de los usuarios del sistema sin necesidad de conocer la contraseña y así poder corroborar que está todo correcto desde las diferentes perspectivas de los roles que poseemos en la aplicación.

6.2.1.4. Incremento 4 (07/04/2021 – 21/04/2021)

6.2.1.4.1. Backlog

Tras las historias de usuario implementadas en el anterior incremento, estas son trasladadas a la columna de listo y pasamos a coger de nuevo historias de usuario que se encuentren seleccionadas para desarrollo. El *backlog* del que partimos pues es el que se muestra en la ilustración 33.

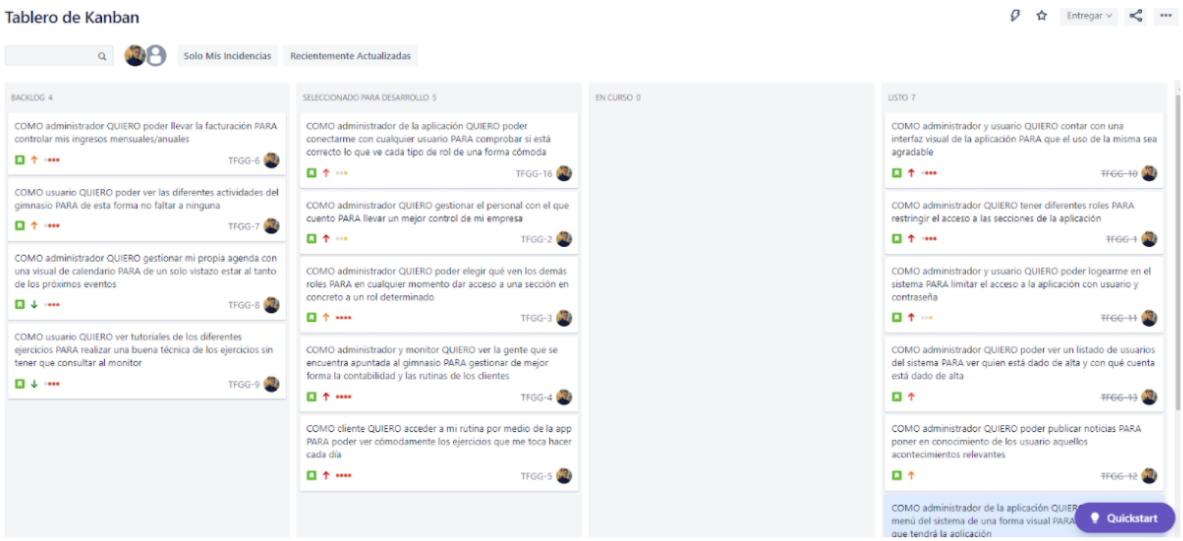

Ilustración 33. Backlog al inicio del incremento 4

6.2.1.4.2. Historia de Usuario Implementadas

En este incremento se implementaron las siguientes funcionalidades:

- **COMO administrador QUIERO gestionar el personal con el que cuento PARA llevar un mejor control de mi empresa**
    - De igual forma que se ha realizado anteriormente para las noticias o los usuarios, se realizará en este caso para el personal de la empresa, es decir, se pasará a crear la organización de los datos en formato de tabla en la base de datos, las diferentes operaciones con las funcionalidades principales que buscamos, en este caso, al tratarse de un software de gestión simplemente bastará con un CRUD, y las vistas donde se refleja y se recopila los datos de la tabla de la base de datos que hemos creado.

- **COMO administrador de la aplicación QUIERO poder conectarme con cualquier usuario PARA comprobar si está correcto lo que ve cada tipo de rol de una forma cómoda**
    - Esta funcionalidad se me ocurrió incorporarla a raíz de que en mi trabajo la utilizamos y tiene gran utilidad si lo que queremos es comprobar lo que ven otros usuarios del sistema y no conocemos las credenciales ya que en la base de datos la contraseña se almacena encriptada. Por tanto se creará esta funcionalidad junto a las anteriores del personal y se incorporará un botón que llame a esta funcionalidad en la ficha de cada personal de la empresa.

En la línea del anterior incremento, no solo era necesario dar acceso a los usuarios, sino que también esos usuarios se corresponden a una persona asociada a algún trabajador de la empresa, por tanto había que reflejar ese personal, además de dar la posibilidad de gestionarlo de forma cómoda e intuitiva. Además ante la necesidad de poder comprobar si es correcto la visualización del resto de usuarios, decidí implementar la funcionalidad de poder conectarse como cada usuario de forma cómoda.

6.2.1.4.3. Esquema de la Base de Datos

En la ilustración 34 se muestra el diagrama entidad relación previo a la implementación de las historias de usuario seleccionadas.

Ilustración 34. Esquema de la Base de Datos del incremento 4.

6.2.1.4.4. Observaciones adicionales del incremento

- **Mapeo de países, provincias y poblaciones**
  - Tal y como se puede observar en el respectivo diagrama entidad relación de este incremento, he decidido almacenar en la base de datos todos los países, provincias y municipios con el fin de agilizar el proceso de elección de dichos campos en los formularios de usuario o personal. De esta forma también evitaremos posibles errores en la escritura de esos datos, para ello buscaremos a través de internet este tipo de tablas y las añadiremos a nuestra base de datos.

6.2.1.5. Incremento 5 (21/04/2021 – 16/05/2021)

6.2.1.5.1. Backlog

Una vez implementadas las historias de usuario anteriores y en función de la fecha en la que se finaliza el incremento anterior, consideramos la necesidad de implementar de añadir e implementar de forma inmediata la historia de usuario siguiente:

- **COMO desarrollador QUIERO poder transformar mi aplicación en una PWA PARA disfrutar de una versión de escritorio mucho más accesible**

Finalmente tras la adición de la historia de usuario en la que convertiremos nuestra aplicación a una PWA el *backlog* resultante quedaría de la manera que observamos en la ilustración 35:

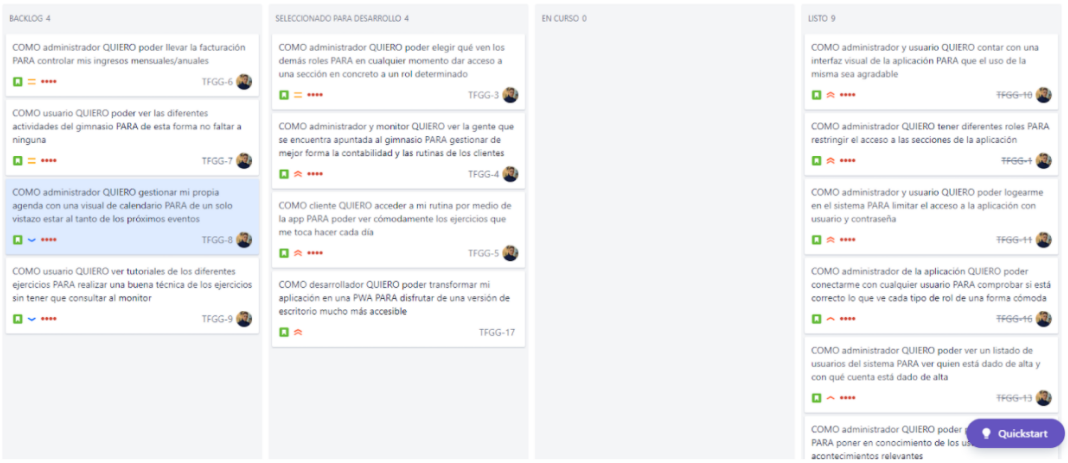

Ilustración 35. Backlog al inicio del incremento 5.

6.2.1.5.2. Historia de Usuario Implementadas

Con respecto a las historias de usuario que se implementaron en este incremento serían las siguientes:

- **COMO desarrollador QUIERO poder transformar mi aplicación en una PWA PARA disfrutar de una versión de escritorio mucho más accesible**
    - Finalmente el objetivo de desarrollar esta aplicación era poder transformarla en una Aplicación Web Progresiva, por lo que una vez que ya tenemos una versión estable y funcional de la aplicación web, lo más conveniente era ponernos manos a la obra en esta transformación y así poner en práctica lo estudiado acerca de este tipo de aplicaciones y comprobar en primera persona ciertas funcionalidades.
- **COMO administrador QUIERO poder elegir qué ven los demás roles PARA en cualquier momento dar acceso a una sección en concreto a un rol determinado**
    - Con esta historia de usuario se terminaría el círculo de gestión en cuanto a permisos y visibilidad de secciones del menú en función del rol que el usuario tenga.

Por último en vista de la fecha en la que estaba y que ya contaba con una aplicación completamente funcional, decidí realizar la transformación de la aplicación en una aplicación web progresiva ya que finalmente es la intención de esta aplicación. Tras esta transformación solamente quedaba para contar con una aplicación de gestión de personal la funcionalidad de decidir qué ve cada tipo de usuario en función de su rol.

6.2.1.5.3. Esquema de la Base de Datos

En la ilustración 36 se muestra el diagrama entidad relación previo a la implementación de las historias de usuario seleccionadas.

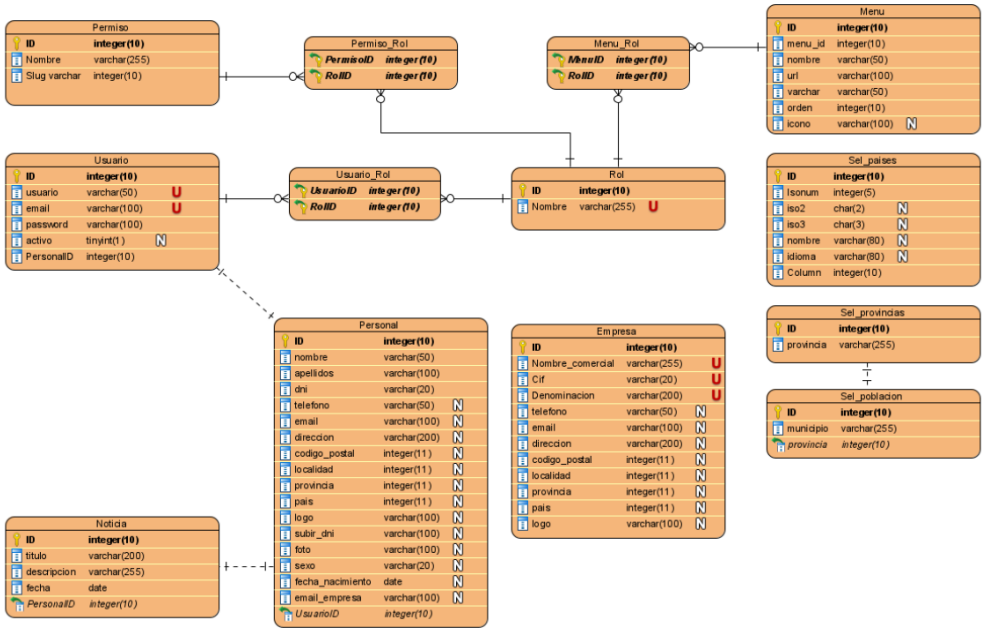

Ilustración 36. Esquema de la Base de Datos del incremento 5.

6.2.1.5.4. Incidencias Conversión a PWA

En mi caso he realizado la aplicación con el *framework* de php **Laravel** y he desplegado la aplicación en un servidor local conocido como **Laragon**.

Sin embargo me he encontrado con incidencias a la hora de realizar la transformación el paso de transformación de la aplicación Web a Aplicación Web Progresiva. Para esta conversión utilicé la biblioteca "**Laravel PWA**". Tras realizar los pasos de configuración me encontré con que no funcionaba el *service worker* dado que no permitía la instalación de la respectiva aplicación.

Tras un tedioso proceso de depuración me encontré con que el problema venía de caché que configura dicha biblioteca, por lo que para que finalmente funcione correctamente hay que descartar la caché local de dicho fichero donde se configura el *service worker*, de esta forma este primer problema quedaría solucionado.

Por otro lado el problema con el que me encontré fue que a pesar de encontrarme en un servidor local en la consola del navegador que utilizo para depurar este tipo de problemas (Google Chrome) aparecía un mensaje que

mencionaba que la seguridad del sitio no era lo suficientemente adecuada para poder utilizar el *service worker*. Después de investigar largo y tendido concluí que al utilizar laragon junto a direcciones de host virtuales para acceder a los proyectos se ve que el navegador no detecta ese tipo de acceso como un servidor local, por lo que lo cataloga como sitio no seguro impidiendo la instalación de la PWA al tratarse de un requerimiento básico la utilización del protocolo HTTPS o el uso de un servidor local.

La solución que acabé dándole a este problema fue en mi caso el de desplegar otro servidor local, a nivel de proyecto, en este caso al tratarse de Laravel esta opción se consigue mediante el uso del comando "*php artisan serve*". Con esta opción se despliega el proyecto en una dirección que nos proporciona la salida de la consola, de tal forma que al acceder de esa directa al proyecto, el navegador sí lo detecta como un servidor local, permitiendo así la ejecución de la funcionalidad implementada en el *service worker* y con ello la instalación de nuestra PWA.

6.3. Resumen final de la aplicación

A pesar de que inicialmente la aplicación se planteó como una aplicación de gestión de un gimnasio, con el paso del tiempo y en función de los requisitos, e incidencias finalmente la aplicación actual solo cuenta con una gestión de personal completa a nivel de roles y permisos, pudiendo elegir en todo momento qué ve cada tipo de rol. Además de esto se permite publicar noticias visibles para todos los usuarios. Esto se debe a que para poder reflejar los clientes del gimnasio, en primer lugar, bajo mi punto de vista es más prioritario tener una gestión de permisos y del personal de la propia empresa y a partir de ahí construir esa gestión y control de personal e incluso aumentarlo cada vez más de forma progresiva según se vaya requiriendo.

7. Conclusión

Una vez hemos analizado todo lo referente a las Aplicaciones Web Progresivas (PWA), podemos concluir que las PWA son un complemento extra para las aplicaciones web, el cual si hacemos balance entre el coste de implementación de este tipo aplicaciones junto con los beneficios que nos aportan, el resultado sin duda es positivo.

Hoy en día las principales aplicaciones web ya cuentan con su versión progresiva, por tanto, con el paso del tiempo será cada vez más conveniente la implementación de la PWA para los diferentes sitios web ya desplegados en internet. De forma adicional a que se trata de una tecnología de desarrollo emergente, el tener la aplicación progresiva va a mejorar también el posicionamiento web de nuestro sitio, aumentando así la visibilidad de la marca.

Las Aplicaciones Web Progresivas se tratan de una corriente en pleno crecimiento, por tanto, de cara a un futuro siempre se podrá hacer mejoras sobre esa PWA ya que poco a poco las herramientas de desarrollo irán profundizando más en este tipo de aplicaciones permitiendo un mayor número de funcionalidades.

Bajo mi punto de vista es cuestión de tiempo que surja la necesidad de implementar estas PWA para adaptarse al desarrollo tecnológico que las diferentes empresas necesitan año tras año y de esta forma no quedarse atrás en el mundo de la tecnología.

8. Temporización Final

De igual forma que al inicio del este Trabajo de Fin de Grado expusimos una planificación temporal de cómo me planteé el desarrollo del mismo en función de las fases que implica este tipo de TFG y el tiempo en el que me gustaría terminarlo, acordé junto con mi tutor el realizar la misma planificación pero después del proceso, con el fin de contrastar lo que se planifica inicialmente con respecto a lo que finalmente es, dado que en el proceso siempre surgen imprevistos que retrasan el proyecto.

El otro motivo por el que fue interesante realizar este proceso de planificación fue la de coger experiencia en cuanto a planificación se refiere. Mi especialización dentro del grado es la de **ingeniería del software** por lo que la planificación de los proyectos es fundamental y empezar a coger soltura en cuanto a estimaciones iniciales y documentación del desarrollo temporal del proyecto siempre será una buena opción de cara a mi futuro laboral. El resultado final de la planificación de este Trabajo de Fin de Grado sería el siguiente:

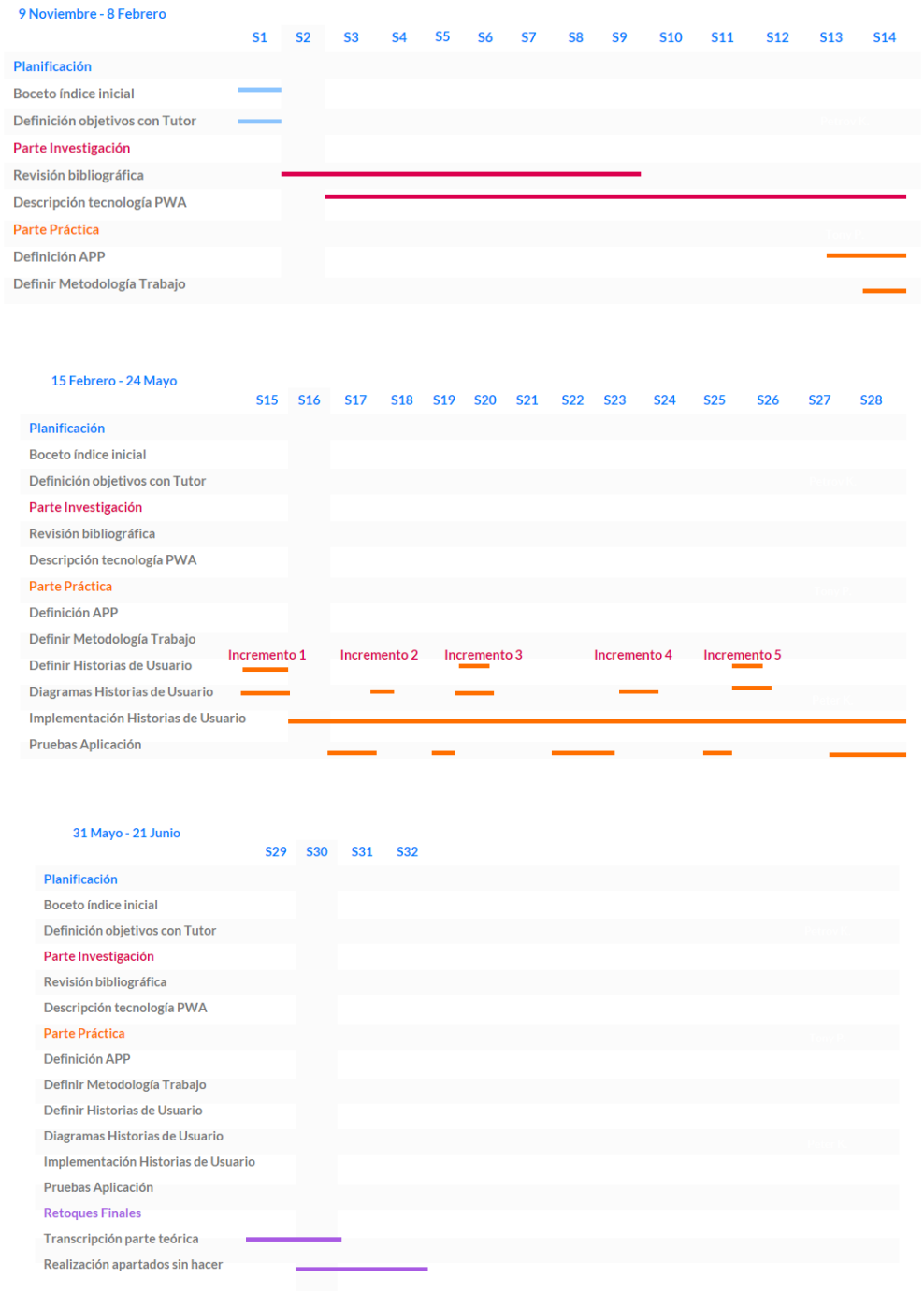

Ilustración 37. Diagrama de Gantt con la temporización final del Trabajo de Fin de Grado

Como principales observaciones podemos decir que en cuanto a la planificación de la primera parte teórica en la que se engloba toda la búsqueda de información así como la extracción de la misma de las fuentes recogidas; se ha cumplido con un gran grado de detalle. Sin embargo donde más discrepancia vemos es en la parte práctica, donde encontramos varianzas tanto en la metodología seguida como en la duración de esta parte, alargándose hasta casi un mes más de lo previsto.

9. Bibliografía

[1] *Grzybowska, K., Karwatka, P., Ratowski, F., & Kwiecien, A. (2019). The history of PWA development: The PWA Book. Divante.com - eCommerce Software House. Recuperado el 28 de noviembre de 2020 de https://www.divante.com/pwabook/chapter/02-the-history-of-pwas#the-external-landscape-of-pwas.*

[2] *Ranchal, J. (2019, September 26). Aplicaciones Web Progresivas: qué son, cómo funcionan y qué tienes que saber. MuyComputerPRO. Recuperado el 17 de enero de 2021 de https://www.muycomputerpro.com/2019/09/26/aplicaciones-web-progresivas-que-son-como-funcionan-y-que-tienes-que-saber.*

[3] *Gustafson, A. (2017, May 24). Progressive Web Apps and the Windows Ecosystem. Aaron Gustafson. Recuperado el 20 de noviembre de 2020 de https://www.aaron-gustafson.com/notebook/progressive-web-apps-and-the-windows-ecosystem/.*

[4] *Osmani, A. (2020, July 24). Getting Started with Progressive Web Apps | Google Developers. Google. Recuperado el 20 de noviembre de 2020 de https://developers.google.com/web/updates/2015/12/getting-started-pwa.*

[5] *MSEdgeTeam. (n.d.). Aplicaciones web progresivas en Windows - Microsoft Edge Development. Aplicaciones web progresivas en Windows - Microsoft Edge Development | Microsoft Docs. Recuperado el 21 de noviembre de 2020 de https://docs.microsoft.com/es-es/microsoft-edge/progressive-web-apps-chromium/.*

[6] *Campos , D. (2018, April 9). 5 grandes compañías que ya han apostado por las Progressive Web Apps. LinkedIn. Recuperado el 22 de noviembre de 2020 de https://www.linkedin.com/pulse/5-grandes-compa%C3%B1%C3%ADas-que-ya-han-apostado-por-las-web-apps-diego-campos/?originalSubdomain=es.*

[7] *Hager, R. (2020, January 22). Google Drive is now available as a Progressive Web App. Android Police. Recuperado el 15 de diciembre de 2020 de https://www.androidpolice.com/2020/01/22/google-drive-is-now-available-as-a-progressive-web-app/.*

[8] *Mehrotra, A. (2020, January 25). Google Drive now has a PWA for Windows 10. MSPoweruser. Recuperado el 15 de diciembre de 2020 de https://mspoweruser.com/google-drive-now-has-a-pwa-for-windows-10/.*

[9] *Enge, E. (n.d.). Mobile vs. Desktop Usage in 2020. / Perficient, Inc. Recuperado el 13 de diciembre de 2020 de https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage.*

[10] *Gómez Chacón, A. (2019, February 19). ¿Qué son las Progressive Web Apps y por qué utilizar esta tecnología en nuestros desarrollos? PWA. Recuperado el 14 de diciembre de 2020 de https://pwaexperts.io/blog/que-son-las-pwa-y-porque-utilizarlas.*

[11] *Deusto. (2014, April 2). Deusto Formación. Recuperado el 17 de enero de 2021 de https://www.deustoformacion.com/blog/apps-moviles/desarrollo-aplicaciones-multiplataforma-claves-principales.*

[12] *Barrera , A. (2017, July 14). APLICACIONES HÍBRIDAS: ¿QUÉ SON Y CÓMO USARLAS? NextU LATAM. Recuperado el 17 de enero de 2021 de https://www.nextu.com/blog/aplicaciones-hibridas-que-son-y-como-usarlas/.*

[13] *López, S. (2020, June 5). Qué es PWA: características, ventajas y desventajas. DIGITAL55. Recuperado el 20 de noviembre de 2020 de https://www.digital55.com/desarrollo-tecnologia/que-es-pwa-ventajas-desventajas/.*

[14] *Progressive Web Apps: Escaping Tabs Without Losing Our Soul. Infrequently Noted. (2015, June 15). Recuperado el 17 de enero de 2021 de https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/.*

[15] Vidal, M. (2019, November 5). PWA: Qué son y cómo funcionan las Progressive Web Apps. IEB School. Recuperado el 22 de noviembre de 2020 de https://www.iebschool.com/blog/progressive-web-apps-analitica-usabilidad/.

[16] López, R. (2020, December 5). Guía Completa sobre Progressive Web Apps. Dosmedia. Recuperado el 18 de enero de 2021 de https://www.dosmedia.com/que-son-progressive-web-apps/#_Aplicaciones_web_progresivas_y_su_soporte_en_navegadores.

[17] Prieto Fernández, R. (n.d.). Qué son y como funcionan las Progressive Web Apps. raulprietofernandez. Recuperado el 19 de enero de 2021 de https://www.raulprietofernandez.net/blog/internet/que-son-y-como-funcionan-las-progressive-web-apps#h3-2-como-funcionan-las-progressive-web-apps.

[18] Morinigo, L. (2020, March 30). PWA Series: Service Workers, los básicos de la experiencia offline. Medium. Recuperado el 19 de enero de 2021 de https://medium.com/samsung-internet-dev/pwa-series-service-workers-los-b%C3%A1sicos-de-la-experiencia-offline-14592542c738.

[19] Gaunt, M. (2020, July 24). Introducción a los service workers | Web Fundamentals. Google Developers. Recuperado el 19 de enero de 2021 de https://developers.google.com/web/fundamentals/primers/service-workers?hl=es.

[20] Google. (n.d.). Proteger Sitios con el protocolo HTTPS. Google Developers. Recuperado el 11 de junio de 2021 de https://developers.google.com/search/docs/advanced/security/https?hl=es.

[21] Santoni, M. (2018, January 10). Las Progressive Web Apps: compatibilidad de las funcionalidades en función de los navegadores. GoodBarber. Recuperado el 11 de febrero de 2021 de https://es.goodbarber.com/blog/las-progressive-web-apps-compatibilidad-de-las-funcionalidades-en-funcion-de-los-navegadores-a592/.

[22] YouTube. (2019). Steve Jobs introducing Pwa in 2007. YouTube. Recuperado el 15 de diciembre de 2020 de https://www.youtube.com/watch?v=QvQ9JNm_qWc

[23] Russell, A. (2015, June 15). Progressive Web Apps: Escaping Tabs Without Losing Our Soul. Infrequently Noted. Recuperado el 15 de diciembre de 2020 de https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/.

[24] Grigsby, J. (2018, November 12). Progressive Web Apps by Jason Grigsby. A Book Apart.. Recuperado el 15 de diciembre de 2020 de https://abookapart.com/products/progressive-web-apps.

[25] Documentación de UWP: desarrollador de aplicaciones para UWP - UWP applications. Documentación de UWP: desarrollador de aplicaciones para UWP - UWP applications | Microsoft Docs. (n.d.). https://docs.microsoft.com/es-es/windows/uwp/.