Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Nombre del Centro

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

Alumno: José Carlos Mena Expósito

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

Septiembre, 2021

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Víctor M. 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 José Carlos Mena Expósito, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2021

El alumno:

Jose Carlos Mena Exposito
Firmado digitalmente por Jose Carlos Mena Exposito
Fecha: 2021.09.10 14:06:41 +02'00'

José Carlos Mena Expósito

El tutor:

Firmado por RIVAS SANTOS VICTOR MANUEL - 29798183G el día 10/09/2021 con un certificado emitido por AC FNMT Usuarios

Víctor M. Rivas Santos

Agradecimientos: Hoy echo la vista hacia atrás y soy más consciente que nunca de que todo el esfuerzo realizado para llegar hasta aquí ha merecido la pena. En este largo camino ha habido momentos buenos, momentos muy malos y días en los que era imposible seguir compatibilizando mis estudios con el trabajo. Pero sin duda, todas estas dificultades me han hecho ser una persona más madura que cuando comencé mi etapa universitaria.

Es por eso que me gustaría agradecer en primer lugar a mi tutor Víctor M. Rivas Santos por apoyarme y ser mi guía en cada una de las etapas de este proyecto. Gracias mamá por enseñarme lo realmente importante y darme tu cariño en los momentos más difíciles. Gracias a mi padre y a mis hermanas por preocuparos por mí y estar siempre ahí. Gracias Ana Mari por ser mi apoyo en este duro camino, por darme tu amor y hacer que encontrara los errores del código con una simple charla y una bolsa de chuches cualquiera. Gracias a la Escuela Politécnica Superior y a todos sus profesores sin excepción, por motivarme y alimentar mi curiosidad y conocimiento durante todos estos años.

A todos y cada uno de vosotros, ¡gracias!.

Índice

1. Introducción 11
1.1. Evolución telefonía móvil 11
1.2. Objetivo principal 12
1.3. La World Wide Web y su protocolo HTTP 12
1.3.1. Versiones protocolo HTTP 13
1.2.2. Métodos de petición más importantes 15
1.2.3. Códigos de respuesta HTTP 15
1.4. Evolución de usuarios en internet 16
2. Tipos de aplicaciones móviles 17
2.1. Aplicación nativa 18
2.2. Aplicación híbrida 18
2.3. Progressive Web Apps 19
2.3.1. Definición Progressive Web Apps 19
2.3.2. Características Progressive Web Apps 19
2.3.3. Elementos de una Progressive Web App 21
2.3.4. Algunos ejemplos de Progressive Web Apps 23
2.3.5. Marketplace de Progressive Web Apps 27
2.4. Comparativa de aplicaciones 29
3. Metodologías de desarrollo 31
3.1. Introducción 31
3.2. Definición 31
3.3. Tipos de metodologías 31
3.3.1. Metodologías tradicionales 31
3.3.2. Metodologías Ágiles 32
3.4. Metodología utilizada 36
4. SEO y Progressive Web App 36
4.1. Origen SEO 37
4.2. Definición de SEO 37
4.3. Cómo funcionan los motores de búsqueda 38
4.4. SEO On-site y SEO Off-site 39
4.5. Black Hat SEO vs White Hat SEO 40
4.6. Beneficios SEO de una Progressive Web App 40
4.7. Lighthouse 41
5. Del CMS a la PWA 42
5.1. ¿Qué es un CMS? 42

5.2. CMS más populares 43
5.3. ¿Qué son los plugins? 45
5.4. Plugins para convertir una web en una PWA 45
5.4.1. Plugins para Shopify 46
5.4.2. Plugins para WordPress 47
5.4.3. Plugins para Magento 49
6. Tecnologías utilizadas 49
6.1. Phaser 49
6.1.1. Crear un proyecto en Phaser desde cero 51
6.1.2. Convertir aplicación Phaser en PWA 53
6.1.3. Mostrar botón para instalar aplicación a2hs 57
6.2. Jira para la administración de tareas 59
6.3. GIT para el control de versiones 60
6.4. Otras tecnologías 60
7. Desarrollo del software 60
7.1. Tareas a realizar 61
7.2. Entregable 1 62
7.2.1. Tareas entregable 63
7.2.2. Storyboards entregable 63
7.2.3. Paleta de colores 64
7.2.4. Implementación entregable 65
7.2.5. Resultados 69
7.3. Entregable 2 70
7.3.1. Tareas entregable 70
7.3.2. Storyboards entregable 70
7.3.3. Implementación entregable 71
7.3.4. Resultados 79
7.4. Entregable 3 80
7.4.1. Tareas entregable 81
7.4.2. Implementación entregable 81
7.4.3. Resultados 84
7.5. Entregable 4 85
7.5.1. Tareas entregable 85
7.5.2. Implementación entregable 85
7.5.3. Resultados 88
7.6. Entregable 5 89

7.6.1. Tareas entregable 89
7.6.2. Storyboards entregable 89
7.6.3. Implementación entregable 90
7.6.4. Resultados 95
7.6.5. Pruebas usabilidad del sistema 96
8. Conclusión 98
9. Trabajos futuros 99
10. Apéndice 100
10.1. Instalación y configuración del sistema 100
10.2. Cuestionario 104
11. Bibliografía 104

Índice de ilustraciones

Ilustración 1.1 Ley de Moore 11
Ilustración 1.2 Versiones HTTP 13
Ilustración 1.3 Campos de una petición HTTP 14
Ilustración 1.4 Números de usuarios de internet (millones) 17
Ilustración 2.1 Ejemplo service worker 22
Ilustración 2.2 Ejemplo manifest 23
Ilustración 2.3 PWA Starbucks 24
Ilustración 2.4 PWA Trivago 25
Ilustración 2.5 PWA Twitter 26
Ilustración 2.6 PWA Aliexpress 27
Ilustración 2.7 Marketplace Appscore 28
Ilustración 2.8 Instalar PWA Appscore 29
Ilustración 2.9 PWA Pinterest 30
Ilustración 2.10 App nativa Pinterest 30
Ilustración 3.1 Planificación, ejecución y entrega final 32
Ilustración 3.2 Eventos de un sprint 34
Ilustración 3.3 Tablero básico Kanban 36
Ilustración 4.1 Componentes SEO 38
Ilustración 4.2 Resultado Lighthouse web starbucks.es 42
Ilustración 5.1 CMS más utilizados 43
Ilustración 5.2 Top 10 CMS para e-commerces 44
Ilustración 5.3 WooCommerce 45
Ilustración 5.4 Litefy 46
Ilustración 5.5 PWA, Mobile App & Web Push 47
Ilustración 5.6 PWA for WP & AMP 47
Ilustración 5.7 Super Progressive Web App 48
Ilustración 5.8 PWA 48
Ilustración 5.9 Magento PWA Studio 49
Ilustración 6.1 Icono framework Phaser 50
Ilustración 6.2 Estructura clase Game de Phaser 50
Ilustración 6.3 Estructura clase State de Phaser 51
Ilustración 6.4 Código base fichero index.html 52
Ilustración 6.5 Código ejemplo configuración juego Phaser 52
Ilustración 6.6 Aplicación de ejemplo Phaser 53
Ilustración 6.7 Código carga service worker (fichero load-sw.js) 54
Ilustración 6.8 Código para guardar ficheros en caché (sw.js) 55
Ilustración 6.9 Código para utilizar ficheros en caché (sw.js) 55
Ilustración 6.10 Código para limpiar caché (sw.js) 56
Ilustración 6.11 Código ejemplo manifest (manifest.webmanifest) 56
Ilustración 6.12 Enlaza el fichero manifest (index.html) 57
Ilustración 6.13 Código html para implementar botón instalar pwa (index.html) 57
Ilustración 6.14 Ejemplo botón instalar pwa 57
Ilustración 6.15 Código javascript implementación instalar pwa (installable.js) 58
Ilustración 6.16 Tablero Jira 59
Ilustración 7.1 Storyboards Busca las parejas 64
Ilustración 7.2 Precarga de recursos (Bootloader.js) 65

Ilustración 7.3 Agregar título y sprite de carga animado (Preload.js) 66
Ilustración 7.4 Agrupar elementos a un contenedor (Menu.js) 66
Ilustración 7.5 Agregar animaciones a un contenedor (Menu.js) 67
Ilustración 7.6 Alinear elementos (Menu.js) 67
Ilustración 7.7 Eventos botones menú principal (Menu.js) 68
Ilustración 7.8 Crear cartas en la escena Play (Play.js) 68
Ilustración 7.9 Cargar iconos aplicación (manifest.webmanifest) 69
Ilustración 7.10 Interfaz aplicación a la finalización del primer entregable 69
Ilustración 7.11 Estado del repositorio github a la finalización del primer entregable 70
Ilustración 7.12 Storyboards Busca las parejas 71
Ilustración 7.13 Configuración compartida (options.js) 71
Ilustración 7.14 Actualizar dificultad (Menu.js) 72
Ilustración 7.15 Importar clase options (Menu.js) 72
Ilustración 7.16 Método reparteCartas (Play.js) 75
Ilustración 7.17 Métodos necesarios para barajar y colocar cartas sobre la mesa (Play.js) . 76
Ilustración 7.18 Escucha los eventos realizados sobre las cartas (Play.js) 77
Ilustración 7.19 Comprueba si se han seleccionado las mismas imagenes (Play.js) 78
Ilustración 7.20 Comprueba si se han encontrado todas las parejas (Play.js) 79
Ilustración 7.21 Marcadores y venta resultado partida implementados en el segundo entregable 80
Ilustración 7.22 Estado del repositorio github a la finalización del segundo entregable 80
Ilustración 7.23 Ranking de jugadores (ranking.json) 81
Ilustración 7.24 Carga json con ranking de jugadores (ranking.js) 82
Ilustración 7.25 Carga datos json ranking de jugadores (ranking.js) 82
Ilustración 7.26 Método pintaRanking (Ranking.js) 82
Ilustración 7.27 Método checkRanking (Play.js) 83
Ilustración 7.28 Fichero ranking.php 84
Ilustración 7.29 Estado del repositorio github a la finalización del tercer entregable 84
Ilustración 7.30 Ranking implementado en el tercer entregable 85
Ilustración 7.31 Lanzar audio flipcard (Play.js) 86
Ilustración 7.32 Precarga de audio flipcard (Bootloader.js) 86
Ilustración 7.33 Inicializa audio flipcard (Play.js) 86
Ilustración 7.34 Lanzar audio flipcard (Play.js) 87
Ilustración 7.35 Lanzar audio de victoria (Play.js) 87
Ilustración 7.36 Lanzar audio de repetir partida (Play.js) 88
Ilustración 7.37 Estado del repositorio github a la finalización del cuarto entregable 88
Ilustración 7.38 Storyboards ventanas emergentes 90
Ilustración 7.39 Icono de opciones del menú principal 90
Ilustración 7.40 Ventana de opciones 91
Ilustración 7.41 Evento para lanzar ventana de opciones (Menu.js) 91
Ilustración 7.42 Cargar imágenes de audio, barajas de cartas y botón ok (Menu.js) 92
Ilustración 7.43 Código para alinear elementos en el menú de opciones (Menu.js) 92
Ilustración 7.44 Código para agregar elementos al contenedor de opciones y centrarlo (Menu.js) 92
Ilustración 7.45 Código para activar/desactivar audio del juego (Menu.js) 93
Ilustración 7.46 Código para seleccionar tipo de baraja (Menu.js) 93
Ilustración 7.47 Baraja seleccionada 93
Ilustración 7.48 Baraja no seleccionada 93

Ilustración 7.49 Código para cerrar ventana cuando se pulsa el botón ok (Menu.js) 94
Ilustración 7.50 Icono info del menú principal 94
Ilustración 7.51 Icono cerrar ventana info del menú principal 94
Ilustración 7.52 Código para abrir y cerrar ventana información sobre el juego (Menu.js) 95
Ilustración 7.53 Estado del repositorio github a la finalización del quinto entregable 95
Ilustración 7.54 Ventanas de información sobre el juego (reglas) y ventana opciones 96
Ilustración 9.1 Resultados Lighthouse 99
Ilustración 9.2 Consejos apartado performance Lighthouse 100
Ilustración 10.1 Instalar PWA desde el navegador Google Chrome 101
Ilustración 10.2 PWA instalada sin conexión a internet 102
Ilustración 10.3 Instalar PWA en IPhone 103
Ilustración 10.4 Prueba de PWA en IPhone 103

Índice de tablas

Tabla 1.1 Métodos de petición más importantes 15
Tabla 7.1 Tareas Proyecto Busca las parejas 62
Tabla 7.2 Tareas Proyecto Busca las parejas 63
Tabla 7.3 Paleta de colores 65
Tabla 7.4 Tareas Proyecto Busca las parejas 70
Tabla 7.5 Tareas Proyecto Busca las parejas 81
Tabla 7.6 Tareas entregable 4 Proyecto Busca las parejas 85
Tabla 7.7 Tareas entregable 4 Proyecto Busca las parejas 89
Tabla 7.8 Resultado por edades de los usuarios pruebas de usabilidad 97
Tabla 7.9 Respuestas de los usuarios a las pruebas de usabilidad 97
Tabla 7.10 Desviación típica respuestas a las pruebas de usabilidad 98

1. Introducción

1.1. Evolución telefonía móvil

Al igual que pasa con la sociedad, las tecnologías están en constante evolución con la intención o finalidad de mejorar los servicios y el valor añadido que estos aportan a los usuarios finales.

La evolución de la tecnología ha ido ligada con el cumplimiento de la conocida ley de Moore [1] (la ilustración 1.1 muestra un ejemplo de la la ley de Moore) según la cual la capacidad de procesamiento se duplicará de forma anual; es decir, de un año para otro la capacidad de procesamiento hardware se multiplicará por dos.

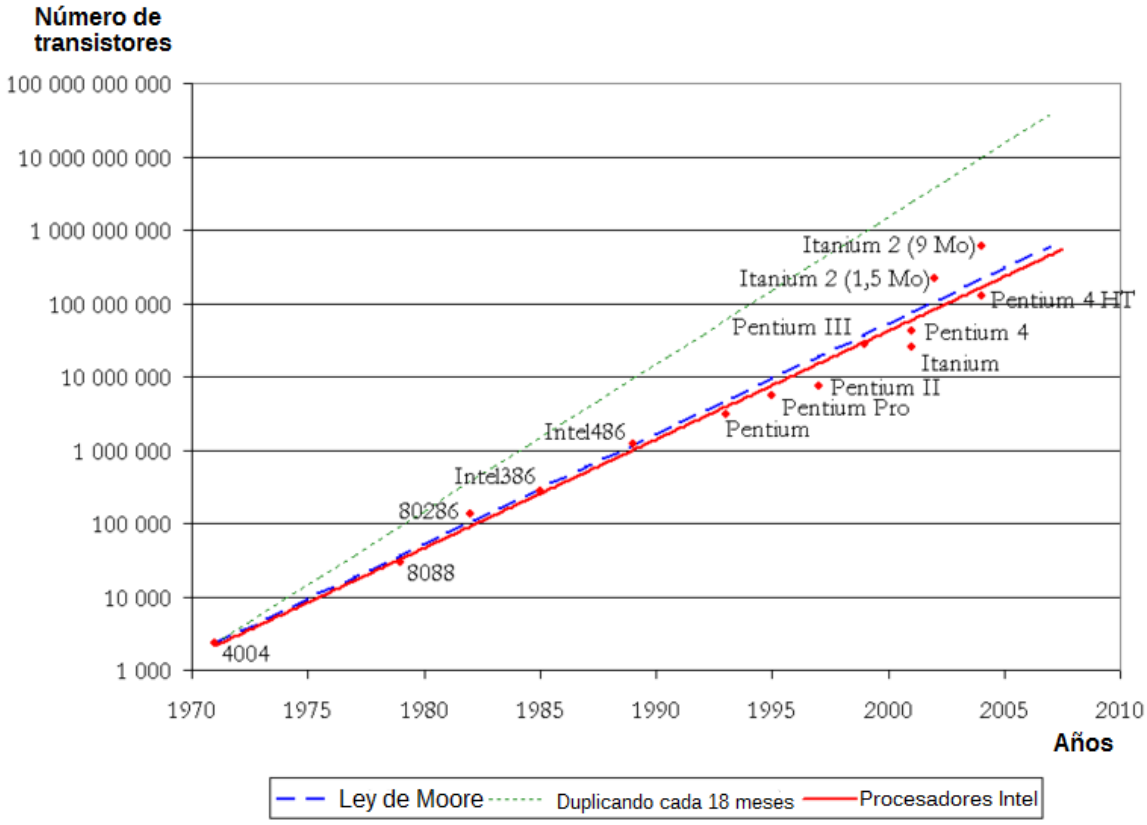


Ilustración 1.1 Ley de Moore

Si hablamos de software, existe una incompatibilidad entre plataformas desde sus comienzos, como puede ser diferencias entre software para ordenadores, móviles y otros dispositivos.

Para intentar unificar el software lo máximo posible y facilitar la adaptación de este al mayor número de dispositivos posibles se observó que un punto en común

que tienen todos los dispositivos era el entorno web, transportado sobre el protocolo de comunicación web HTTP.

Haciendo uso de este protocolo surgieron nuevas tecnologías como son las aplicaciones híbridas o las aplicaciones PWA (Progressive Web Apps).

1.2. Objetivo principal

Como objetivo principal este Trabajo Fin de Grado titulado “Introducción al desarrollo de Progressive web apps – Análisis de la metodología y ejemplo de aplicación práctica” pretende demostrar lo fácil que es transformar una aplicación desarrollada con lenguajes de programación para el desarrollo web, como es Javascript junto con el lenguaje de marcas HTML y CSS para los estilos, en una aplicación instalable en cualquier dispositivo como si de una aplicación nativa se tratase.

Para esta demostración se realizará un videojuego, haciendo uso del Framework Phaser, basado en el tradicional juego de cartas de buscar las parejas. Con este juego se intenta ayudar a personas que necesiten un estímulo mental para reforzar su memoria como pueden ser personas con síndrome de Down, personas con Alzheimer o personas mayores de 65 años. Por supuesto, cualquier persona que quiera ejercitar la memoria podrá utilizar este software.

1.3. La World Wide Web y su protocolo HTTP

El protocolo de comunicación HTTP no se puede entender sin lo que se conoce comúnmente como “internet” o “World Wide Web”. Ya que HTTP es el protocolo utilizado para la comunicación a través de archivos XML o HTML entre otros.

HTTP al igual que internet nació el en año 1989 en el CERN, desarrollados por Tim Berners-Lee. [2] Aunque la primera conexión entre computadoras se realizó en el año 1969 entre 3 Universidades de California (Estados Unidos), en lo que se conoce como proyecto ARPANET.

En un principio el nombre que se le dio a internet fue el de “Mesh”, que significa malla en inglés, para después modificar su nombre a como se conoce actualmente “World Wide Web” (red mundial).

La World Wide Web se basa en cuatro bloques importantes:

- HTML (HyperText Markup Lenguage). Es una forma de texto utilizado para representar documentos de hipertexto.
- Protocolo HTTP (HyperText Transfer Protocol). Es un protocolo sencillo para el intercambio de documentos en hipertexto.
- Cliente o navegador web. Puede mostrar o incluso editar los documentos intercambiados. El primer navegador web se llamaba “WorldWideWeb”.
- Servidor. Es el encargado de dar acceso a los documentos.

Para los primeros meses del año 1991 ya estaban funcionando los primeros servidores fuera del CERN bajo el protocolo HTTP/0.9, que fue la primera versión de este protocolo (a veces conocida como el protocolo de una sola línea).

Inicialmente a esta versión del protocolo HTTP no se le había asignado ningún nombre pero para poder diferenciarla de las posteriores versiones se le dio el nombre de HTTP/0.9.

Realmente era un protocolo tan sencillo como se le conocía ya que las peticiones se realizaban a través de una petición GET /nombreWeb.html y la respuesta era un documento HTML. En esta primera versión únicamente se podían transmitir archivos html ya que aún no se utilizaban las cabeceras.

A continuación veremos las diferentes versiones del protocolo HTTP y sus métodos de petición más importantes.

1.3.1. Versiones protocolo HTTP

Desde su creación en el año 1989, el protocolo HTTP [3] ha sufrido varias actualizaciones y mejoras, tal y como se puede observar en la ilustración 1.2. En la ilustración 1.3 puede verse un ejemplo de los campos que contiene una petición HTTP.

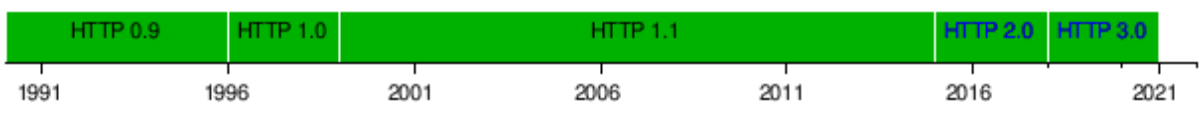


Ilustración 1.2 Versiones HTTP

- HTTP/0.9. Lanzada en 1991 es una versión obsoleta. Soporta únicamente el comando GET y no soporta el uso de cabeceras.
- HTTP/1.0. Lanzada en el mes de Mayo de 1996. Fue la primera revisión del protocolo y también la primera que especifica su versión. Los métodos permitidos son GET, HEAD y POST. Actualmente se sigue utilizando ampliamente en servidores proxy.
- HTTP/1.1. Lanzada en el mes de Junio de 1999. Es la versión más utilizada en la actualidad y permite al cliente enviar varias solicitudes a la vez por la misma conexión (lo que se conoce como pipelining) lo que permite eliminar el tiempo de Round-Trip delay o tiempo de ida y vuelta [4] por cada petición.
- HTTP/1.2. Versión experimental lanzada en el mes de Febrero del año 2000.
- HTTP/2. Versión lanzada en el mes de Mayo de 2015. En esta versión los conceptos básicos continúan igual, sin cambios. Sus novedades están enfocadas en la forma de empaquetar los datos y su transporte.
- HTTP/3. Es la versión más reciente del protocolo HTTP y fue lanzada en Octubre de 2018. Su principal novedad es el uso del protocolo UDP en lugar de TCP. Lo que se busca con esta nueva versión es mejorar el tiempo de latencia de la conexión reduciendo el número de intercambios de datos entre el emisor y el receptor.

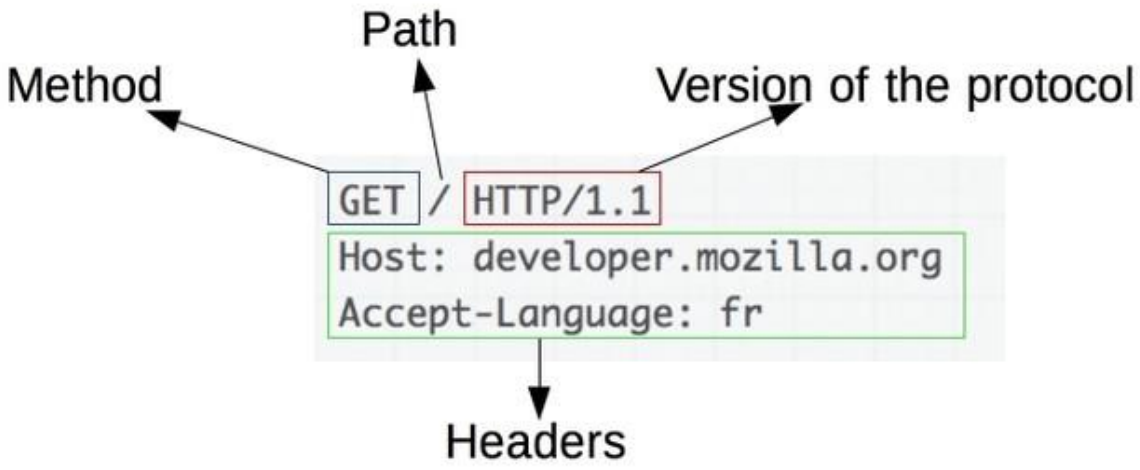


Ilustración 1.3 Campos de una petición HTTP

1.2.2. Métodos de petición más importantes

Según ha ido evolucionando el protocolo HTTP se han ido añadiendo nuevos métodos de petición haciendo uso de su gran flexibilidad, una de las características por las que se sigue utilizando este protocolo hoy en día.

Estos métodos suelen hacer referencia a una acción de un recurso, por lo que se suele decir que deben ser verbos.

Algunos de los métodos más utilizados [5] son los siguientes:

Método	**Definición**
GET	Es una petición de solicitud o lectura de datos. Consiste en obtener información almacenada en el servidor por lo que no modifica el estado interno del servidor.
POST	Envía datos al servidor, de forma que estos sean procesados por la URI a la que se envía. Se utiliza para crear nuevos recursos en el servidor.
PUT	Envía datos al servidor al igual que POST pero con la diferencia de que la URI hace referencia a un dato ya existente en el servidor para su actualización. En resumen, actualiza un recurso ya existente en el servidor.
DELETE	Elimina un recurso existente en el servidor modificando el estado interno del mismo.

Tabla 1.1 Métodos de petición más importantes

1.2.3. Códigos de respuesta HTTP

Una vez realizada la petición al servidor, éste nos envía una respuesta haciendo uso de unos códigos. Cada código, indica lo que ha sucedido con la petición realizada y el resto del mensaje de respuesta dependerá de él.

Al igual que ha sucedido con los métodos los códigos de respuesta también han ido evolucionando y han sido actualizados a lo largo del tiempo. Cada código tiene su significado concreto.

Los códigos de respuesta están elegidos de forma que según la centena a la que pertenezcan pueda identificarse el tipo de respuesta. Esto es:

- 1xx. Códigos de respuesta informativos. Petición recibida y está siendo procesada.
- 2xx. Códigos de respuesta correcta. La solicitud ha sido procesada correctamente.
- 200. Ok
- 201. Creado
- 202. Aceptado
- 3xx. Códigos de respuesta de redirección. Acciones adicionales deben ser tomadas por el cliente para finalizar la petición.
- 4xx. Códigos de error causados por el cliente. Error producido por un error del cliente mientras se procesaba la solicitud.
- 400. Bad request
- 401. Unauthorized
- 403. Forbidden
- 404. Not found
- 5xx. Códigos de error causados por el servidor. Error producido por el servidor mientras se procesaba la solicitud.
- 500. Internal Server Error

1.4. Evolución de usuarios en internet

A finales de los años 90 el número de usuarios de internet no era superior a los 300 millones [6]. En la década de los dos mil el número de usuarios siguió creciendo y ganando popularidad.

Como podemos observar en la siguiente ilustración 1.4, la evolución del número de usuarios en los últimos 15 años ha aumentado hasta alcanzar el valor de 4.538 millones de usuarios, lo que supone un 59% de la población mundial actual.

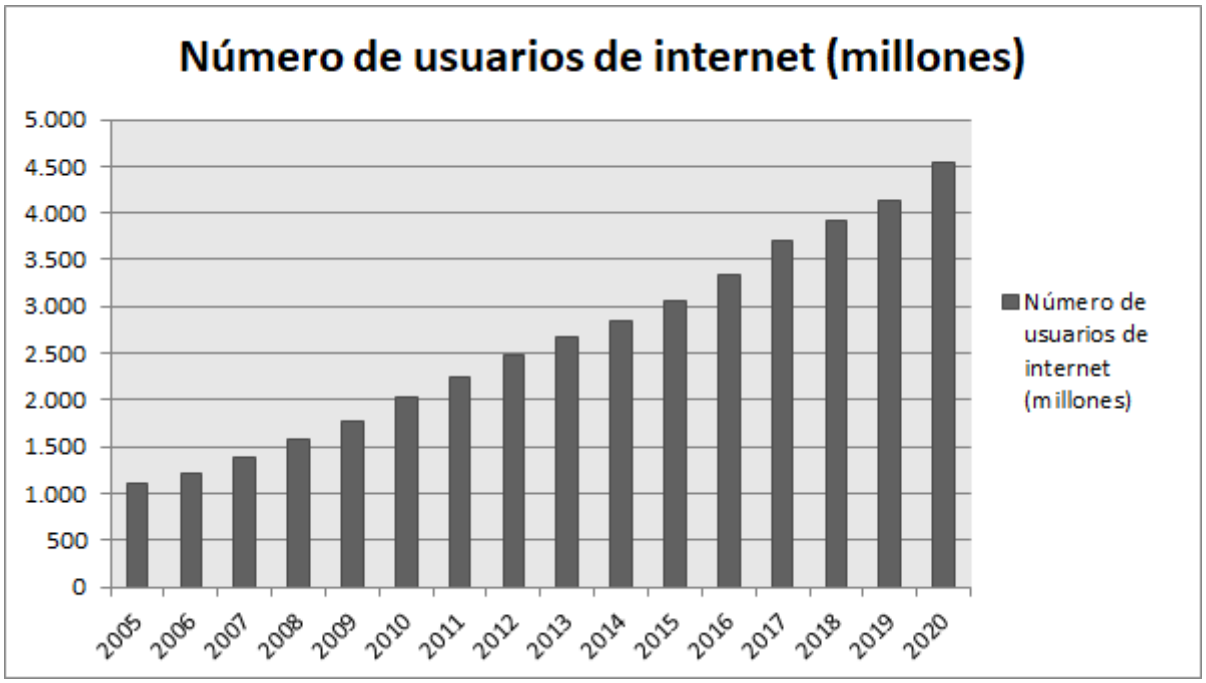


Ilustración 1.4 Números de usuarios de internet (millones)

Este aumento ha permitido que cada vez sea más fácil para las empresas llegar a sus usuarios objetivos a través de internet.

Una forma fácil y directa de conectar con los usuarios son las aplicaciones móviles, ya que el uso del teléfono móvil (Smartphone) es cada vez mayor, el porcentaje de usuarios que acceden a internet a través de un dispositivo móvil ha ascendido hasta el 92,6%.

Según un estudio[1] realizado por la empresa HootSuite[2] el usuario promedio pasa unas 6 horas y 54 minutos al día conectado a internet. Siendo de 3 horas y 39 minutos, sin tener en cuenta el tiempo en redes sociales, si hablamos de usuarios de dispositivos móviles.

2. Tipos de aplicaciones móviles

Hoy en día antes de comenzar a desarrollar una aplicación es necesario conocer las características del proyecto para así decidir qué tipo de aplicación se adapta mejor.

[1] https://wearesocial.com/es/digital-2021-espana
[2] https://www.hootsuite.com/es/

Con la evolución de los dispositivos móviles han ido apareciendo más alternativas a las tradicionales aplicaciones nativas, como son las aplicaciones híbridas y las PWA o Progressive Web Apps.

A continuación, se definirá el concepto de aplicación nativa y aplicación híbrida para después dejar paso a las PWA.

2.1. Aplicación nativa

En pocas palabras, una aplicación nativa es aquella que está desarrollada para el sistema operativo en el que correrá, como puede ser IOS o Android.

La principal ventaja de este tipo de aplicaciones es que cuentan con la posibilidad de acceder al hardware específico del dispositivo, lo que permite a este tipo de aplicaciones obtener un mayor rendimiento.

Por el contrario, su mayor desventaja es que si queremos que nuestra aplicación pueda utilizarse tanto en IOS como en Android es necesario desarrollar dos aplicaciones por separado: una para el sistema operativo Android y otra para el sistema operativo IOS.

2.2. Aplicación híbrida

El funcionamiento de una aplicación híbrida es diferente al de una aplicación nativa, ya que una aplicación híbrida ejecuta una aplicación web a través de un contenedor o WebView. Esto es, un navegador web que puede estar contenido dentro de una aplicación móvil.

Este tipo de aplicaciones están basadas en aplicaciones web por lo que comparten sus mismos elementos de navegación y funcionan únicamente si cuentan con conexión a internet.

Pese a no ser una aplicación nativa se instala igualmente como si lo fuera y además tienen acceso a componentes hardware del dispositivo.

2.3. Progressive Web Apps

2.3.1. Definición Progressive Web Apps

Las PWA son básicamente aplicaciones web que se aprovechan del aumento de nuevas capacidades en los navegadores modernos. [7] En los últimos años, la definición de las Progressive Web Apps se ha visto modificada en varias ocasiones, lo que puede ser un poco confuso.

Las constantes evoluciones en las PWA han permitido que algunas características que antes no estaban disponibles para este tipo de aplicaciones ahora sí lo estén, como puede ser el acceso a ciertos aspectos del hardware del dispositivo.

Frances Berriman y Alex Russell, fueron los impulsores del nombre "Progressive Web Apps" y definieron nueve características [8]:

2.3.2. Características Progressive Web Apps

- **Responsive**

Haciendo uso de las técnicas responsive más modernas, una aplicación PWA es capaz de adaptarse a cualquier dispositivo. Pese a que las PWA van más allá y no se quedan en un simple diseño responsive, esta característica se puede seguir considerando como una de sus principales características.

- **Offline**

Las PWA pueden seguir funcionando de forma parcial pese a no tener conexión a internet o contar con una red deficiente. Esto permite conseguir una mejor experiencia de usuario y evitar la frustración por la imposibilidad de acceso a la aplicación.

- **Apariencia nativa**

Pese a no ser una aplicación nativa, la interfaz de usuario, apariencia, la forma de interactuar y navegar son muy similares a las de las aplicaciones nativas.

- **Actualizada**

Gracias a las actualizaciones automáticas, que no es necesario descargar, la aplicación siempre está actualizada.

- **Segura**

Se utiliza el protocolo seguro HTTPS y tecnologías como TLS para el cifrado, lo que posibilita que todo el tráfico de datos de la aplicación esté seguro.

- **Indexable**

El contenido de una PWA es rastreable e indexable por los buscadores, de forma que pueda aparecer como resultado en un buscador

Utilizando un simple archivo de texto plano llamado “manifest” es posible indicar a los navegadores y buscadores que es una PWA.

- **Notificaciones**

Para llamar la atención a los usuarios y hacer que regresen a la aplicación las PWA cuentan con la posibilidad de utilizar notificaciones para enviar notificaciones.

- **Instalable**

Una PWA no requiere una descarga como tal, sino que cuenta con la posibilidad de crear un icono de la aplicación en la página de inicio del dispositivo móvil, simulando así estar instalada.

Este icono se muestra como un icono más, siendo idéntico al de cualquier otra aplicación nativa.

- **Enlazable**

Las PWA pueden compartirse haciendo uso de una URL, por lo que se puede acceder fácilmente sin necesidad de descargar o instalar nada.

2.3.3. Elementos de una Progressive Web App

Una definición más técnica para las aplicaciones PWA consiste en considerar que una aplicación es una PWA si incluye los siguientes tres elementos:

- **HTTPS**

Una PWA no puede definirse como tal si no hace uso del protocolo HTTPS, ya que esto ofrece una mayor seguridad y es necesario para que los service worker trabajen correctamente.

- **Service Worker**

La tecnología de service worker permite a los desarrolladores crear aplicaciones rápidas y fiables que funcionen aun cuando no se tiene conexión a internet. Puede observarse un ejemplo de service worker en la ilustración 2.1.

```
var myCache = 'testApp-v1';
var files = [
    './',
    './index.html',
    './manifest.webmanifest',
    './def/phaser.d.ts',
    './src/load-sw.js',
    './src/main.js',
    './src/phaser.min.js',
    './assets/icon-192.png',
    './assets/icon-256.png',
    './assets/icon-512.png'
];

self.addEventListener('install', function(event) {
    console.log('sw instalado');
    event.waitUntil(
        caches.open(myCache).then(function(cache) {
            console.log('sw cacheando ficheros');
            return cache.addAll(files);
        }).catch(function(error) {
            console.log(error)
        })
    )
});

self.addEventListener('fetch', (event) => {
    console.log('sw fetch');
```

```
    console.log(event.request.url);
    event.respondWith(
        caches.match(event.request).then(function(response) {
            return response || fetch(event.request);
        }).catch(function(error) {
            console.log(error);
        })
    );
});

self.addEventListener('activate', function(event) {
    console.log('sw activate');
    event.waitUntil(
        caches.keys().then(function(keyList) {
            return Promise.all(keyList.map(function(key) {
                if(key !== myCache) {
                    console.log('sw removing old cache', key);
                    return caches.delete(key);
                }
            }));
        })
    );
});
```

Ilustración 2.1 Ejemplo service worker

La ventaja del service worker es que le permite al desarrollador entender cómo trabaja el navegador; es decir, cómo procesa o maneja las solicitudes y el cacheado.

- **Archivo manifest**

El objetivo del archivo manifest es hacer que la aplicación sea detectable e indexable. En el manifest se indica el nombre de la aplicación, la URL de inicio, los iconos y todos los demás detalles necesarios para transformar el sitio web en una aplicación. En la ilustración 2.2 puede verse un ejemplo de archivo manifest.

```
{
  "short_name": "Phaser ejemplo",
  "name": "Aplicacion de ejemplo - Framework Phaser",
  "icons": [
    {
      "src": "assets/icon-192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "assets/icon-256.png",
      "type": "image/png",
```

```
      "sizes": "256x256"
    },
    {
      "src": "assets/icon-512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "description": "Aplicacion de ejemplo con el framework Phaser en HTML5."
}
```

Ilustración 2.2 Ejemplo manifest

Pese a que una PWA tiene más elementos aparte de estos tres, no puede considerarse una PWA si no incluye todos estos elementos. Es decir, estos tres elementos es lo mínimo que debe incluir una aplicación para que se pueda considerar Progressive Web App.

2.3.4. Algunos ejemplos de Progressive Web Apps

- **Starbucks**

Starbucks[3] estaba en la necesidad de ofrecer una aplicación accesible y funcional para clientes con una mala conexión a internet o una conexión inestable, es por este motivo que decidieron desarrollar su propia PWA [9]. En la ilustración 2.3 se muestra la PWA de Starbucks. Gracias a esto la PWA de Starbucks ahora permite navegar a través del menú, personalizar y agregar productos a un pedido sin conexión a la red, siendo únicamente necesaria la conexión a internet en los pasos finales para confirmar el pedido.

[3] https://www.starbucks.es/

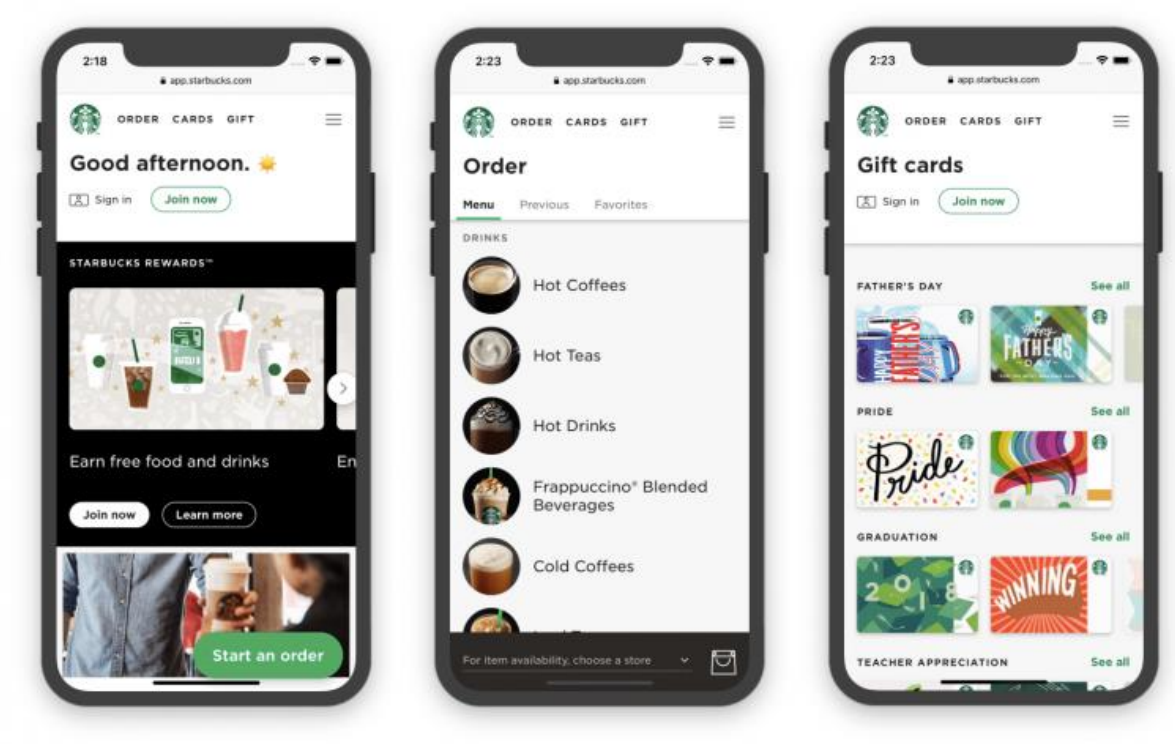


Ilustración 2.3 PWA Starbucks

Las informaciones aportadas por Starbucks indican que han aumentado un 100% la interacción diaria con el usuario, además de que la versión PWA es un 99,84% más ligera que la aplicación de IOS.

- **Trivago**

Trivago[4] es uno de los comparadores de precios para hoteles más famosos y con más tráfico del mundo y cuando le llegó el momento de dar el salto a dispositivos móviles decidieron hacerlo bajo tecnología PWA, ya que ésta le ofrecía una serie de ventajas sobre las demás; como eran las notificaciones push y el acceso sin conexión a internet, entre otras.

Con el desarrollo de su PWA, que puede observarse en la ilustración 2.4, Trivago vio aumentado el engagement en un 97% y las “instalaciones” se dispararon un 150% lo que se traduce en más de medio millón de personas [10].

[4] https://www.trivago.es/

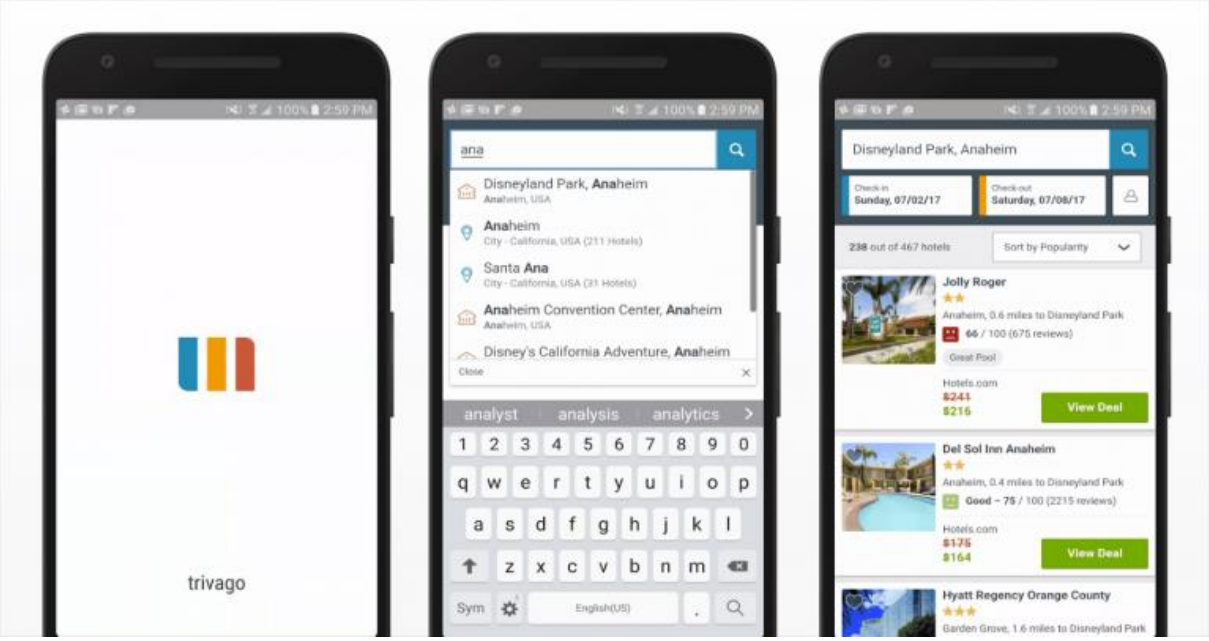


Ilustración 2.4 PWA Trivago

Rolf Schrömgens, cofundador y director general de Trivago resume la decisión de utilizar PWA de la siguiente forma:

> "En trivago, siempre nos hemos sentido orgullosos de lo rápido que crecemos y de lo rápido que aprendemos. Y con todas estas nuevas tecnologías en evolución, queremos estar entre los primeros en hacer uso de su potencial y llevar nuestra misión a más personas".

- **Twitter**

Twitter con Twitter Lite[5] consiguió obtener grandes beneficios utilizando la tecnología PWA. Twitter Lite, en la ilustración 2.5 se muestra su diseño, necesita menos de 1 megabyte de memoria disponible para ser “instalada” en un dispositivo móvil y reduce el consumo de datos móviles en un 70%.

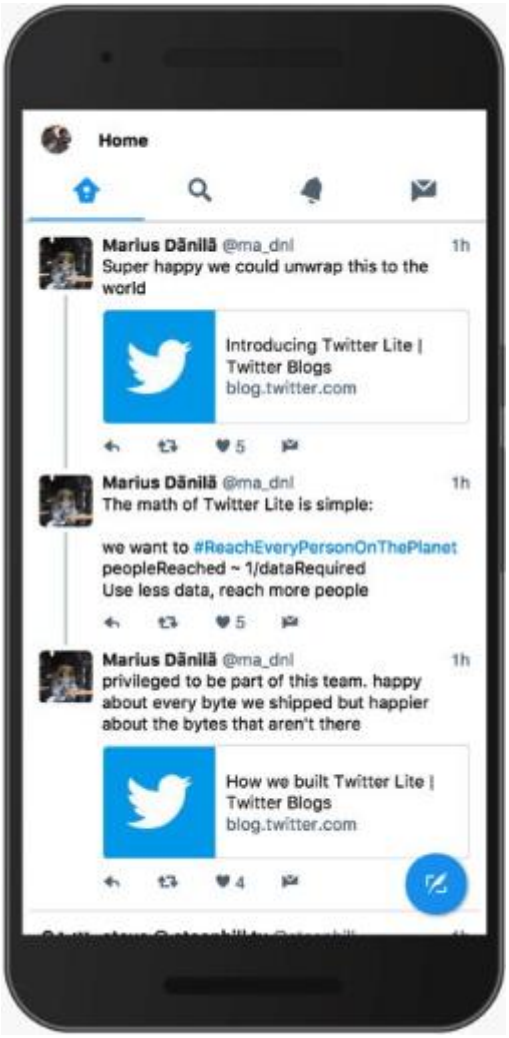


Ilustración 2.5 PWA Twitter

Pero esto no es todo ya que además de la reducción del consumo de datos reduce en un 30% el tiempo de carga de la aplicación. Siendo la primera carga la más lenta con un tiempo inferior a 5 segundos (en redes 3G) y siendo las siguientes cargas casi instantáneas, incluso en las redes más lentas [11].

[5] https://mobile.twitter.com/

- **Aliexpress**

El caso de Aliexpress[6] [12] es un caso donde las PWA han ayudado, y mucho, a atraer y mantener usuarios. Se muestran ejemplos de la PWA de Aliexpress en la ilustración 2.6.

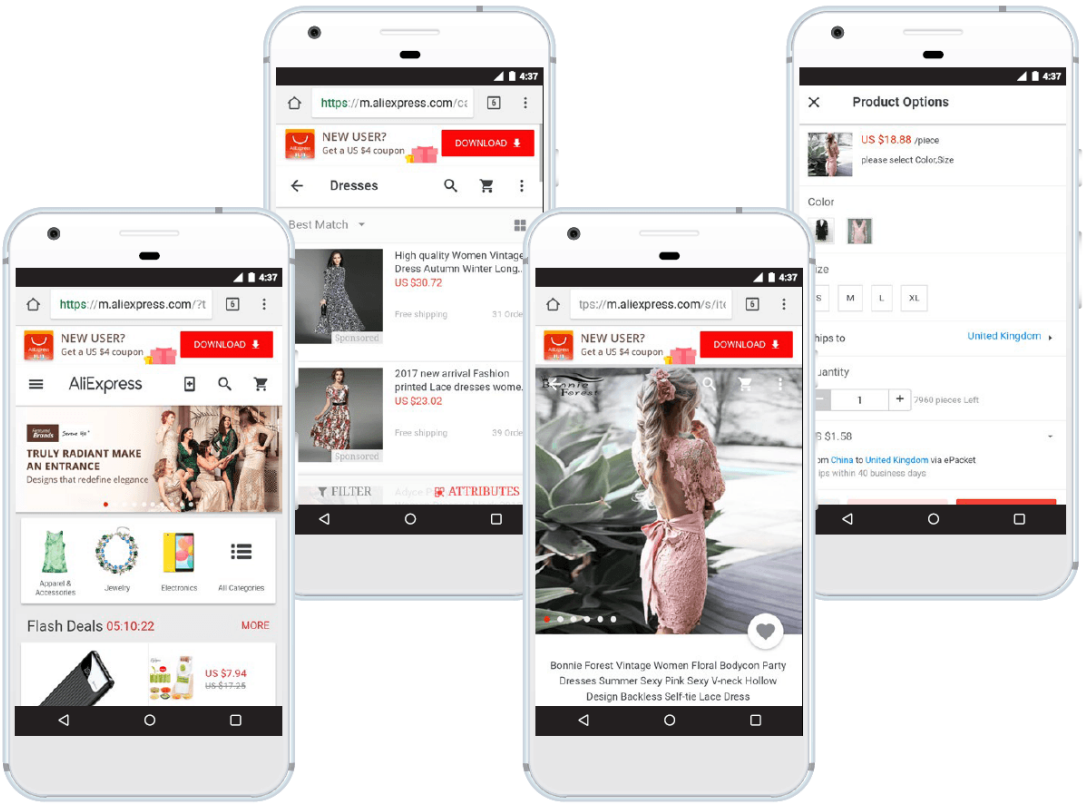


Ilustración 2.6 PWA Aliexpress

La PWA de Aliexpress tras su desarrollo consiguió un tráfico de 666,5 millones de usuarios, lo que se tradujo en un 104% de nuevos usuarios, un aumento del 82% de conversión de compra en los dispositivos con sistema operativo IOS y el doble de páginas visitadas por sesión en todos los navegadores.

Datos que muestran un claro ejemplo de una PWA bien utilizada y los grandes beneficios que ésta puede tener para un negocio o empresa.

2.3.5. Marketplace de Progressive Web Apps

Existe una web donde se puede publicar y descargar aplicaciones PWA como si de una aplicación nativa se tratase, tal y como estamos acostumbrados a hacer en la tienda de Google Play o en la App Store.

[6] https://es.aliexpress.com/

Esta web es appsco.pe[7] y funciona como cualquier otra tienda de aplicaciones nativas comentadas anteriormente. En la ilustración 2.7 se muestran varias imágenes de su funcionamiento. Si accedemos a ella podemos ver que hay multitud de aplicaciones desarrolladas como PWA, hasta aplicaciones tan famosas como Instagram[8] o Twitter[9].

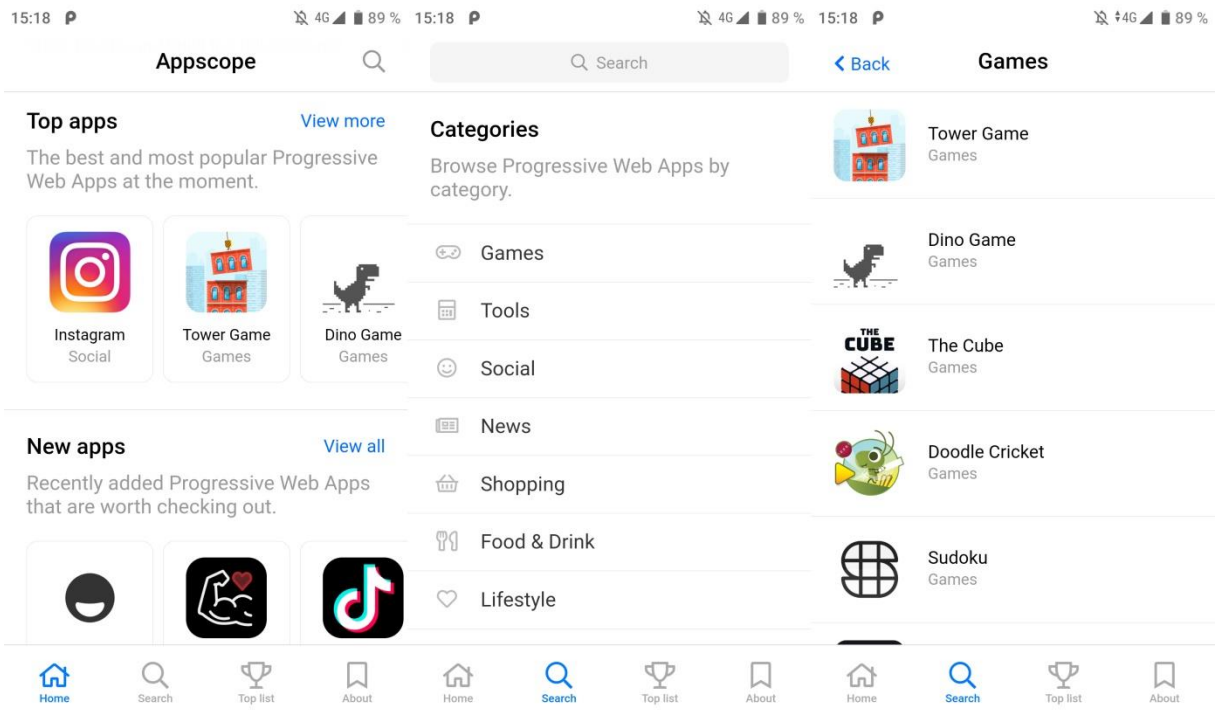


Ilustración 2.7 Marketplace Appscore

La forma de descargar una aplicación es bastante sencilla, ya que basta con entrar en la web, buscar la aplicación que queramos bien por su nombre si ya la conocemos o a través de sus categorías, lanzarla y una vez lanzada podemos instalarla en nuestro dispositivo accediendo a las opciones del navegador y pulsar en “instalar aplicación”. En la ilustración 2.8 se muestra un ejemplo de cómo instalar una PWA en un dispositivo android.

[7] https://appsco.pe/
[8] https://www.instagram.com/
[9] https://mobile.twitter.com/

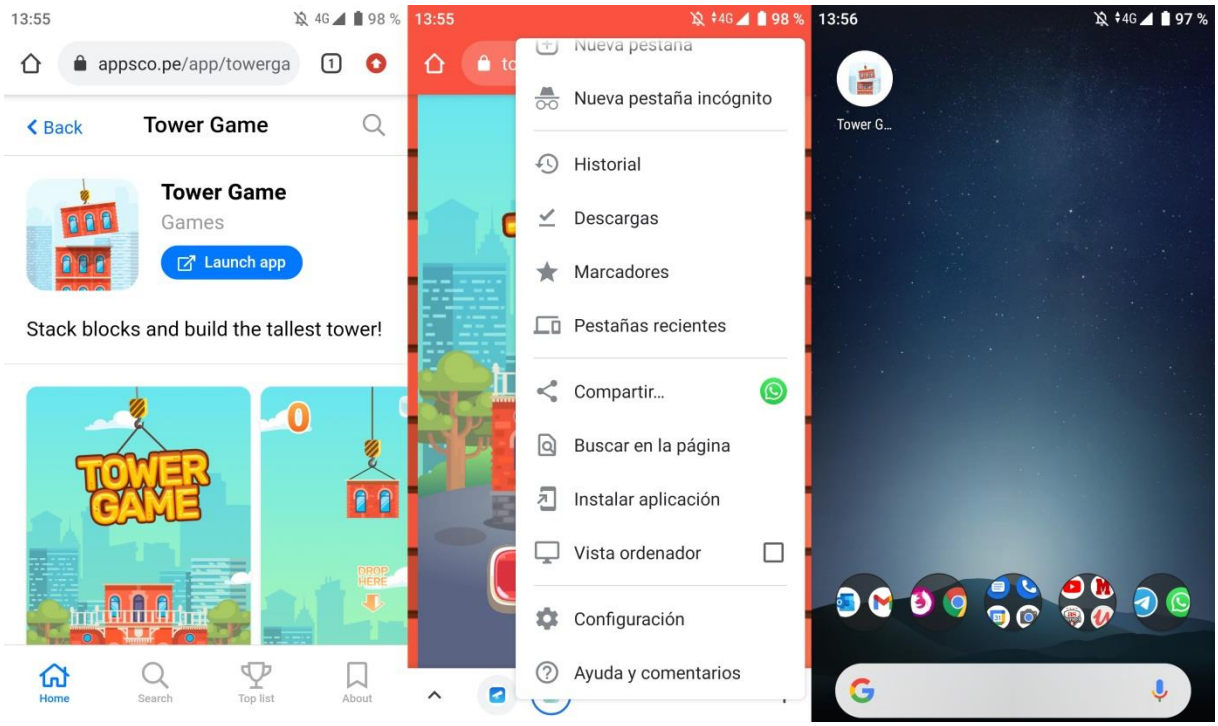


Ilustración 2.8 Instalar PWA Appscore

Una vez instalada la aplicación aparecerá en el escritorio y podremos acceder a ella directamente como una aplicación más [13].

2.4. Comparativa de aplicaciones

A continuación, veremos una comparativa [14] de la aplicación nativa de Pinterest con su PWA. La PWA de Pinterest tiene el siguiente aspecto mostrado en la ilustración 2.9:

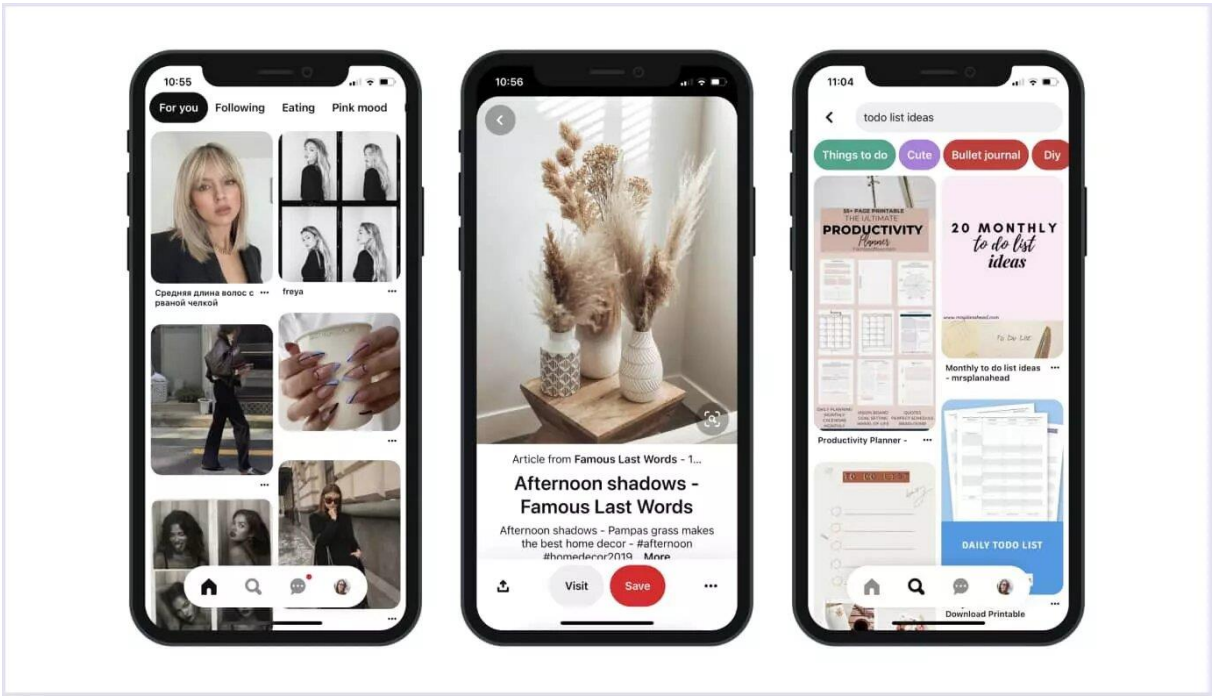


Ilustración 2.9 PWA Pinterest

Mientras que su app nativa tiene el aspecto que se muestra en la ilustración 2.10:

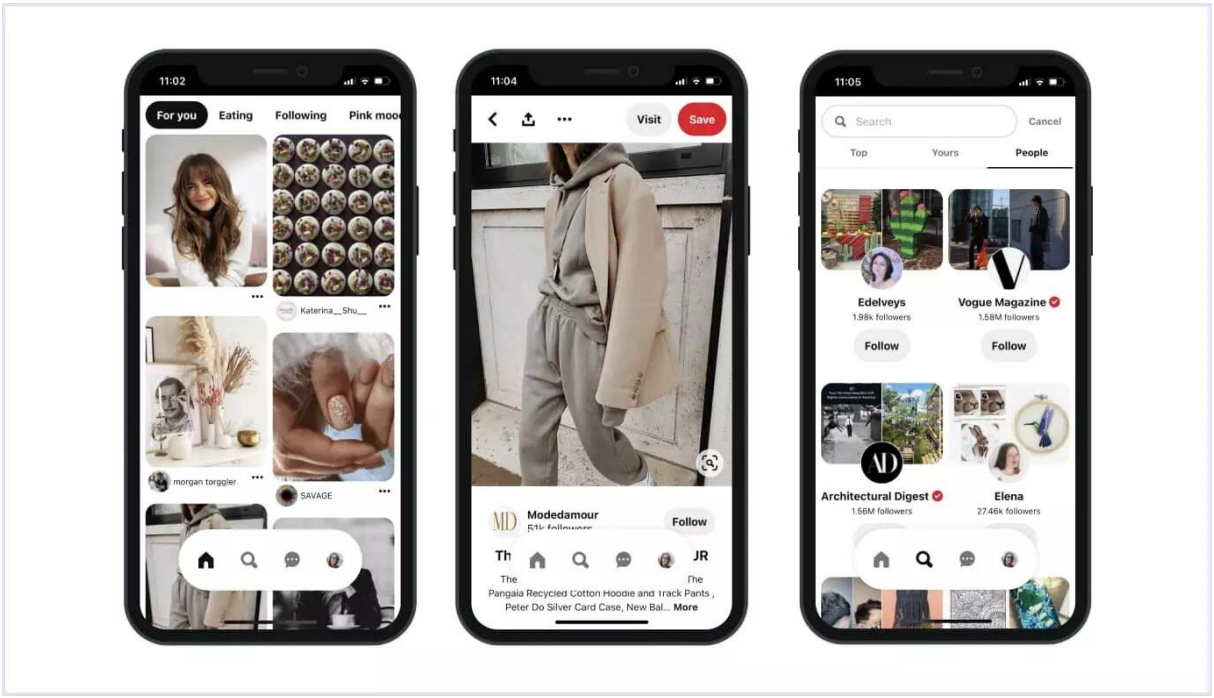


Ilustración 2.10 App nativa Pinterest

Como podemos observar existen pequeñas diferencias pero nada que alejen a ninguna de ellas a su objetivo principal que es proporcionar facilidad a los usuarios para interactuar a través de dispositivos móviles con las marcas.

3. Metodologías de desarrollo

3.1. Introducción

A lo largo de este apartado, incluiré nociones básicas sobre qué son, cómo funcionan y cuando se aplican las diferentes metodologías de desarrollo de software. Explicaré las diferentes alternativas que existen y la que he decidido utilizar para el desarrollo de mi proyecto, con su correspondiente justificación.

3.2. Definición

Las metodologías de desarrollo permiten a los desarrolladores trabajar en equipo de una forma organizada. A lo largo de los últimos años se ha avanzado mucho en la organización de equipos de desarrollo y en el uso de las metodologías.

En un principio las metodologías no eran más que un mero trámite que las empresas llevaban a cabo pero que difícilmente se llevaban a la parte práctica y casi siempre se quedaban en la parte teórica.

Pero las metodologías de desarrollo de software no son ni más ni menos que conjuntos de técnicas y métodos que se utilizan para organizar a los equipos de trabajo cuyo objetivo es el desarrollo de soluciones software.

3.3. Tipos de metodologías

Existen un gran número de metodologías en el desarrollo de software y principalmente podemos agruparlas en dos grandes grupos; las metodologías tradicionales y las metodologías ágiles.

3.3.1. Metodologías tradicionales

Cuando hablamos de metodologías tradicionales nos referimos a las metodologías que se han utilizado desde el inicio del desarrollo de software y que se basan en el ciclo de vida del desarrollo de software en cascada. Este tipo de metodologías tienen muchos inconvenientes:

- No se adapta a los requisitos cambiantes.
- Demasiado trabajo para la planificación y “papeleo”.
- Se entrega todo el proyecto al finalizar su desarrollo.

- Costo elevado en caso de tener que cambiar o reiniciar parte del proyecto.

Debido a estos inconvenientes y las pérdidas de tiempo y tiempo que conllevan algunos de estos problemas, se dio lugar a lo que se conoce como metodologías ágiles.

3.3.2. Metodologías Ágiles

Puede decirse, que las metodologías ágiles tienen su origen en el año 2001. Año en el que un grupo de desarrolladores definieron un documento conocido como "Manifiesto por el Desarrollo Ágil de software" donde definieron todas las buenas prácticas y principios para el desarrollo ágil.

El manifiesto indicaba lo siguiente:

- **Individuos e interacciones** > procesos y herramientas
- **Software que funcione** > extensa documentación
- **Colaboración del cliente** > negociación del contrato
- **Respuesta a los cambios** > seguimiento de un plan

Estos principios nos indican que sobre todo debemos buscar la satisfacción del cliente haciendo uso de entregas constantes del software, la permanente comunicación con el cliente y tener siempre una buena comunicación entre todos los integrantes del equipo de desarrollo. La ilustración 3.1 representa el proceso continuo de planificación, ejecución y entrega final.

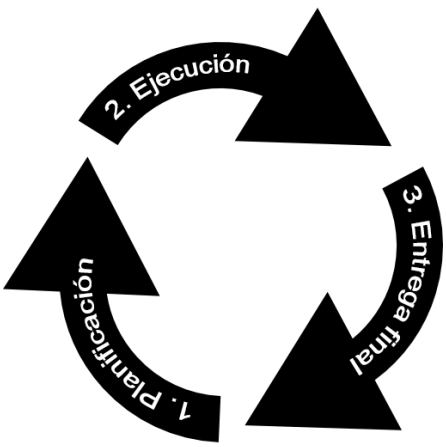


Ilustración 3.1 Planificación, ejecución y entrega final

Las metodologías ágiles se basan en un modelo de desarrollo incremental, respetando el ciclo de desarrollo de software: planificación, ejecución y entrega final. O lo que es lo mismo, en el desarrollo de software por etapas, lo que nos facilita la identificación y resolución de problemas mientras desarrollamos el software.

En este tipo de metodologías, los requisitos y soluciones del software evolucionan de forma continua en el tiempo, según la necesidad del proyecto. El trabajo se realiza mediante la colaboración de equipos auto-organizados y multidisciplinarios.

Existen innumerables metodologías ágiles, de las cuales trataré de explicar las metodologías más utilizadas en la actualidad.

- **SCRUM**

Scrum es sin duda el número uno de las metodologías ágiles, el más utilizado en los equipos de desarrollo. Fue identificado y definido a principios de los años 80 por Ikujiro Nonaka e Hirotaka Takeuchi mientras analizaban el desarrollo por parte de las principales empresas tecnológicas (Fuji-xerox, Canon, Honda, NEX, Epson...) de sus nuevos productos.

Nonaka y Takeuchi compararon la nueva forma de trabajo en equipo con el avance en formación de melé de los jugadores de rugby. Motivo por el que se conoce de esta manera a la metodología Scrum.

En Scrum, las iteraciones son conocidas como “Sprints”, que hace referencia al periodo en el cual se lleva a cabo el trabajo. Es recomendable que su duración sea constante y se defina por el equipo (entre 2 y 4 semanas).

Dentro de la metodología Scrum existen varios componentes:

- **Scrum master.** Es el conocido como facilitador, ya que trabaja principalmente en eliminar los posibles obstáculos que el equipo de desarrollo pueda encontrarse. Además, se asegura de que el proceso Scrum se utilice como es debido y se cumplan las reglas.

- **Product owner.** Representa la voz del cliente y debe asegurarse de que el equipo trabaje de forma adecuada y define los objetivos del proyecto.
- **Equipo de desarrolladores.** Cuenta con la responsabilidad de entregar el producto a tiempo. Se recomienda un equipo pequeño, multifuncional, de 3 a 9 personas con las habilidades y conocimientos necesarios para realizar el trabajo.
- **Eventos de Scrum.** Estos eventos permiten crear hábitos y minimizar el número de reuniones innecesarias. Estos eventos cuentan con una duración máxima establecida.

  Algunos de estos eventos son: Sprint, Sprint planning, Daily Scrum, Sprint review y Spring Retrospective.

A continuación, en la siguiente ilustración 3.2, podemos ver un pequeño resumen de los eventos que debe tener un sprint:

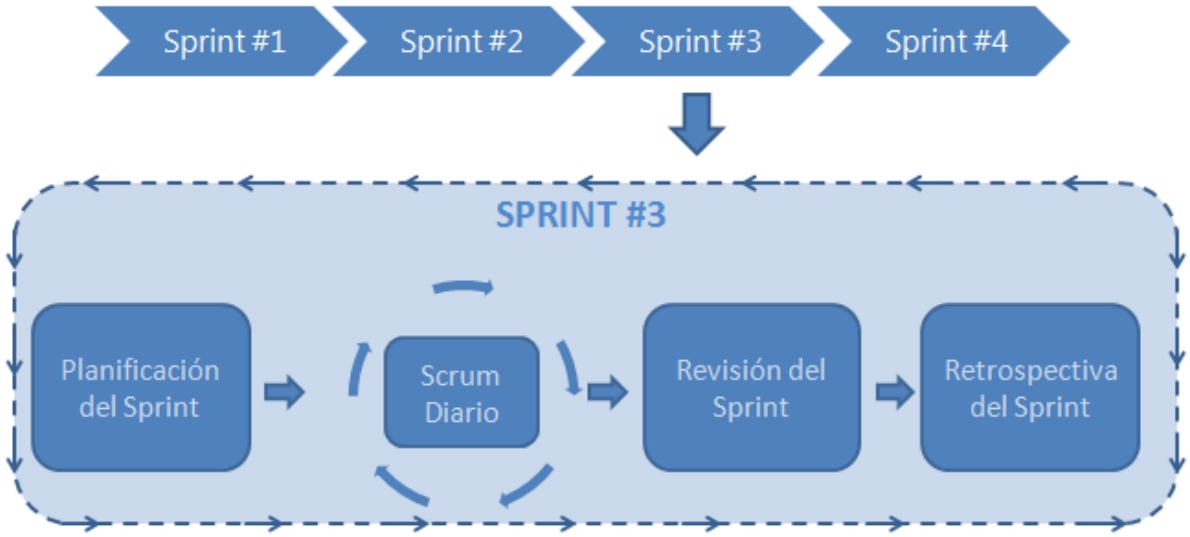


Ilustración 3.2 Eventos de un sprint

Si Scrum es tan conocida y utilizada en todo el mundo es por sus beneficios y su "fácil" implantación. A continuación, se muestran una serie de beneficios de utilizar Scrum en el desarrollo de software:

- Flexibilidad a cambios
- Reducción del "Time to market"
- Mayor calidad del software

- Mayor productividad
- Maximiza el retorno de la inversión (ROI)
- Predicciones de tiempo
- Reducción de riesgos

- **KANBAN**

La metodología Kanban, es un método basado en Lean y fue desarrollado por la empresa de automóviles Toyota con la finalidad de mejorar el desarrollo de procesos. Es una palabra japonesa y si la traducimos literalmente, significa “tarjeta con signos o señal visual”.

Los elementos de trabajo son divididos y representados de forma visual en un tablero (conocido como “tablero de kanban”), lo que permite a los miembros del equipo ver en todo momento el estado actual en que se encuentra cada uno de los elementos.

Los equipos kanban se centran en reducir el tiempo que se tarda en desarrollar un proyecto de inicio a fin, maximizando la eficiencia del equipo.

El tablero kanban más básico está compuesto por tres columnas; “por hacer”, “en proceso” y “hecho”. En la ilustración 3.3 podemos ver un ejemplo de un tablero kanban:

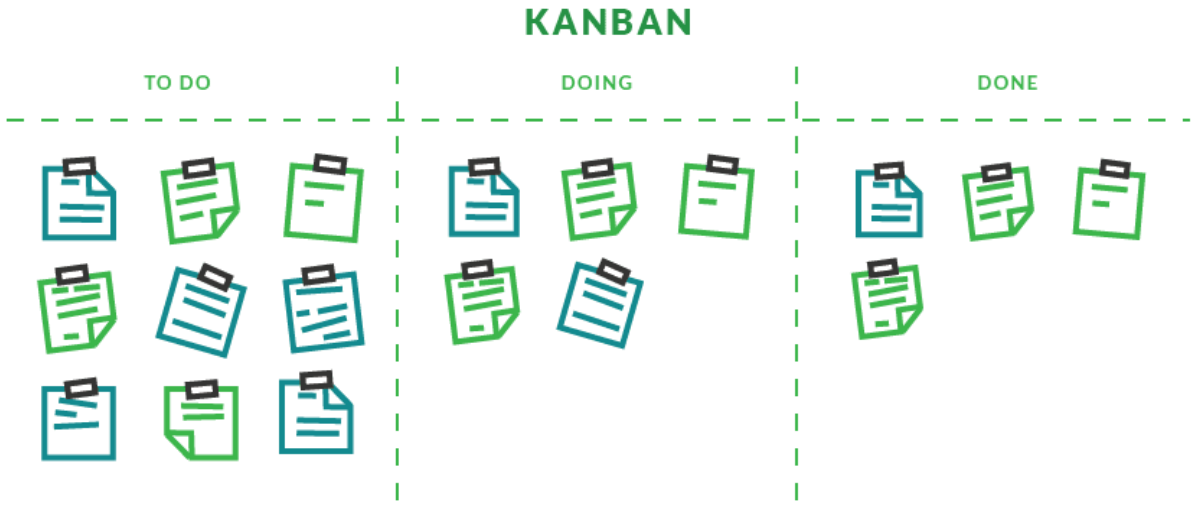


Ilustración 3.3 Tablero básico Kanban

Pero Kanban nos da la libertad absoluta para personalizar la forma de trabajar de nuestro equipo pudiendo crear tantas columnas como queramos.

A diferencia con Scrum, en kanban no existe una planificación para las entregas de hitos. En kanban las entregas se realizan según se van finalizando tareas, sin necesidad de tener que esperar a un hito de publicación.

Las tareas del tablero son modificadas por todos los componentes del equipo, lo que otorga un mayor compromiso a cada uno de los integrantes del equipo de desarrollo, que deben responsabilizarse en la realización y entrega de cada una de las tareas del tablero [15].

3.4. Metodología utilizada

La metodología utilizada para el desarrollo de mi proyecto software será la metodología ágil kanban por los siguientes motivos:

- Kanban no necesita un número mínimo de componentes en el equipo (en mi caso estoy yo solo).
- Evoluciona según las necesidades del proyecto.
- Permite añadir y eliminar tareas según el nivel de prioridad.
- Flexibilidad, en Kanban todo consiste en ser flexible.

4. SEO y Progressive Web App

Las PWA pueden ser indexadas por los buscadores online, lo que le ha dado cierta ventaja frente a las Apps nativas, ya que cuando un usuario realiza una búsqueda en internet se le podrá mostrar la PWA con la opción de instalarla directamente.

Ésto hace que el SEO sea una parte importante durante el proceso de desarrollo de una Progressive Web App.

4.1. Origen SEO

Antes de dar una definición sobre lo que es el SEO es interesante conocer sus orígenes.

No podemos hablar de SEO sin hablar antes de los buscadores web, en concreto de los primeros buscadores que nacieron en los años 90. Década en la que nació el buscador más conocido de todos, Google (1996).

En esta década se produjo el “boom de las webs”, ya que la gente se dio cuenta que se podía ganar dinero con ellas y llegaron a la conclusión de que debería atraer tráfico. Es así como nació el SEO, con la necesidad de los dueños de las páginas web de mejorar el posicionamiento de sus páginas y aparecer en las primeras posiciones.

4.2. Definición de SEO

Las siglas en inglés de la palabra SEO significan “Search Engine Optimization”, lo que en español quiere decir “optimización de motores de búsqueda”. Consiste en mejorar la visibilidad en los resultados orgánicos de los diferentes buscadores de un sitio web.

En sus inicios el SEO era algo que los dueños de las webs podían “manipular” con ciertas técnicas para conseguir que los buscadores les posicionaran en los primeros resultados, aprovechando las “vulnerabilidades” de los buscadores. La ilustración 4.1 muestra algunos de los componentes más importantes del SEO.

Ilustración 4.1 Componentes SEO

Sin embargo, esto ha cambiado con el paso de los años y hoy en día estas técnicas ya no funcionan, es más, una web puede ser penalizada por Google por utilizar estas técnicas (conocidas como Black SEO) hasta el punto de no indexar la web y por consiguiente no mostrar como resultado de búsqueda a los usuarios.

Según palabras de Matt Cutts, ex director del departamento de Spam de Google, se centra cada vez más en la optimización de la experiencia de búsqueda de los usuarios.

4.3. Cómo funcionan los motores de búsqueda

El funcionamiento básico de un motor de búsqueda pueda resumirse en dos pasos importantes:

- **Rastreo.** Consiste en rastrear o explorar páginas web haciendo uso de los rastreadores conocidos como “bots”. Estos bots van recorriendo la web a través de los enlaces de la misma, lo que nos indica que es muy importante tener una buena organización de la estructura de enlaces.

Las páginas utilizarán un archivo llamado “robots.txt” para indicar a los rastreadores si existen páginas que no deben ser rastreadas o indexadas, entre otras opciones.

- **Indexación.** Una vez que una web haya sido rastreada y procesada por uno o varios bots, se incluye en un índice y se ordena según unas características dadas por el motor de búsqueda.

  Una vez indexada la web, el algoritmo del motor de búsqueda será quien decida qué página aparece antes y cuál aparece después para una consulta realizada por el usuario.

  En la actualidad una de las características más importantes para el algoritmo de Google es la calidad del contenido de una web.

4.4. SEO On-site y SEO Off-site

Los ambientes en los que podemos definir el uso del SEO son dos:

- **On-site.** El SEO on-site es todo aquello que podemos realizar dentro de nuestra propia página web con la intención de mejorar el posicionamiento de la misma. Como podría ser:
  - Mejorar el tiempo de carga.
  - Optimizar las Keywords.
  - Optimizar el código fuente.
- **Off-site.** Por el contrario, el SEO off-site es aquel que se realiza fuera de nuestra página web pero que aun así nos ayudará a mejorar el posicionamiento de una página web.

  Una técnica que se puede realizar dentro del SEO off-site y tiene grandes resultados es el linkbuilding, que consiste en colocar enlaces a nuestra web desde otras webs relevantes, como pueden ser revistas, redes sociales o web relacionadas del sector.

4.5. Black Hat SEO vs White Hat SEO

Anteriormente he hablado de que existen ciertas técnicas “desaconsejadas” que son catalogadas como técnicas de Black SEO o Black Hat SEO. A continuación, diferenciaré sobre las técnicas de Black Hat SEO y White Hat SEO.

- **Black Hat SEO.** Consiste en realizar técnicas poco éticas o que contradicen las indicaciones dadas por los motores de búsqueda, como pueden ser Google, Yahoo o Bing.

  Esta técnica puede dar beneficios a corto plazo pero es bastante arriesgada ya que es posible que suframos una penalización a medio plazo que nos mande todo el trabajo realizado al garete.

- **White Hat SEO.** Al contrario que pasaba con el Black Hat el White Hat SEO consiste en utilizar técnicas éticamente correctas y que cumplen con las directrices dadas por los motores de búsqueda en cada momento [16].

4.6. Beneficios SEO de una Progressive Web App

Anteriormente he comentado que una PWA básicamente es un “tipo de página web” que se muestra como una aplicación móvil. Pudiendo realizar las mismas tareas que una aplicación nativa desde un navegador web.

Como sabemos, desde el año 2018 Google (respondiendo a la revolución de los teléfonos móviles) cambió su algoritmo de posicionamiento dando prioridad a las web que estaban adaptadas a dispositivos móviles. Lo que se conoce como “Mobile First Index”.

Esto dio lugar a que todo aquel que tenía una página web (e-commerce, de servicios, personal…) y quería aparecer en los primeros puestos debía tener su web adaptada a dispositivos móviles. Pero, si todas las webs ya están adaptadas a dispositivos móviles la principal ventaja que tienen las PWA es que pueden ser indexadas por los buscadores online y ser instaladas directamente.

Además, es conveniente tener presente que las PWA están basadas en links al igual que las páginas webs, por lo que deben ser optimizadas para los buscadores al igual que una web normal.

Algunas recomendaciones interesantes [17] para obtener un mejor rendimiento en el posicionamiento SEO en una PWA son las siguientes:

- Utilizar Google Search Console y comprobar que los buscadores lean correctamente la información de la PWA.
- Crear un diseño compatible con múltiples dispositivos, adaptando fuentes, imágenes y la estructura.
- Generar contenido indexable por los motores de búsqueda y evitar meter texto en imágenes ya que no podrá ser indexado.
- Las URL deben estar accesibles de forma independiente.
- Evitar contenido duplicado indicando las URL canónicas.
- Comprobar que se ofrece una buena experiencia de usuario midiendo el tiempo de carga.
- Etiquetar y especificar el tipo de contenido que contiene cada una de las páginas de la PWA.
- Comprobar la PWA en diferentes navegadores y dispositivos.

Una herramienta que se puede utilizar para mejorar el SEO de una aplicación PWA, aunque no esté desarrollada específicamente para eso es Lighthouse[10].

4.7. Lighthouse

Lighthouse es una herramienta de auditoría web desarrollada por Google, que viene incorporada en el navegador Google Chrome o que podemos agregar a otro navegador en forma de extensión.

Aunque no es una herramienta pensada para el análisis SEO evalúa varios aspectos que influyen en el posicionamiento SEO de la PWA. Sus resultados nos indicarán el tiempo de carga, la accesibilidad y una serie de recomendaciones de

[10] https://developers.google.com/web/tools/lighthouse?hl=es

mejores prácticas. La ilustración 4.2 muestra los resultados obtenidos de la web de Starbucks.

Algunas cosas que tendrá en cuenta Lighthouse a la hora de analizar nuestro "valor SEO" será la existencia de las meta tags imprescindibles para su correcto posicionamiento y la existencia de un archivo robots.txt válido.

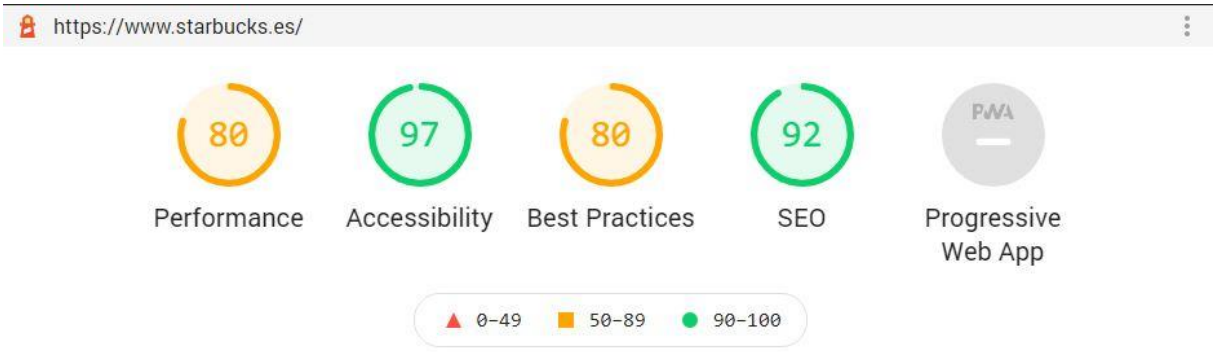


Ilustración 4.2 Resultado Lighthouse web starbucks.es

En cambio, en la sección o apartado de accesibilidad, Lighthouse se centrará más sobre los aspectos que harán de nuestra aplicación una aplicación más accesible para personas discapacitadas. [18]

5. Del CMS a la PWA

Si echamos la vista a atrás, no hace mucho tiempo era necesario tener amplios conocimientos técnicos para ser capaces de crear una web. Sin embargo, con el paso de los años surgieron algunas herramientas conocidas como "Content Management System" o "CMS" que ayudaron y facilitaron la creación de páginas web o tiendas online.

5.1. ¿Qué es un CMS?

Los CMS surgieron con la idea de crear blogs de forma sencilla para los amantes de la escritura y aquellos a los que les gusta escribir sobre uno o varios temas en sus blogs, los conocidos como "bloggers" [19]. Pero con el paso del tiempo estos CMS evolucionaron y se convirtieron en una herramienta sencilla, que mediante el uso de un navegador web permite crear, administrar y modificar todo tipo de sitios web (además de su contenido) sin necesidad de tener conocimientos

sobre programación [20], con la que poder crear todo tipo de páginas webs, desde webs personales, pasando por webs corporativas y llegando hasta tiendas online de gran tamaño.

Los CMS proporcionan al usuario un panel de control desde el que poder personalizar la web de forma muy sencilla, haciendo uso de temas (o themes en inglés) para personalizar el diseño, colores y estilo de la web. En definitiva los CMS nos permiten personalizar casi al 100% una página web de una forma sencilla sin tener, en algunos casos, conocimientos técnicos en programación [21].

Esta democratización del desarrollo web ha abierto un gran abanico de posibilidades y su uso se ha extendido en gran medida, siendo éstos muy utilizados sobre todo en el desarrollo web en empresas de pequeño y/o mediano tamaño.

5.2. CMS más populares

Haciendo uso de la plataforma wappalyzer[11] [22] y sus análisis, se ha obtenido como resultado que los CMS más populares dentro del desarrollo web para e-commerces son los mostrados en la ilustración 5.1.

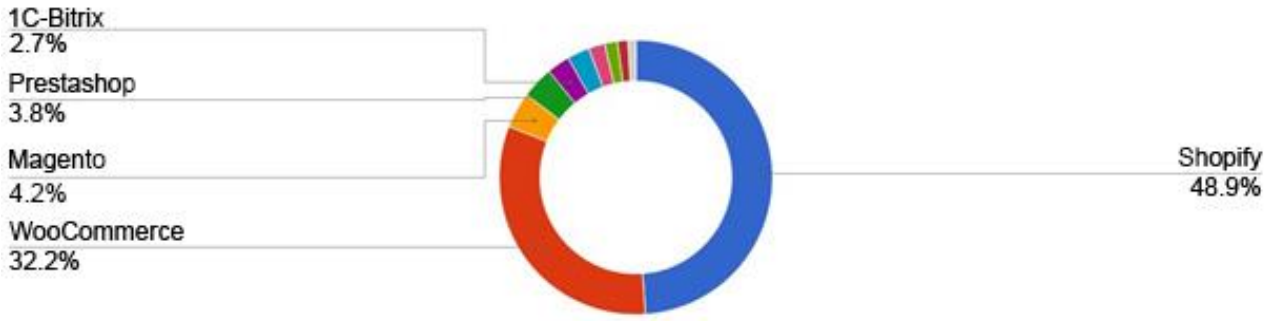
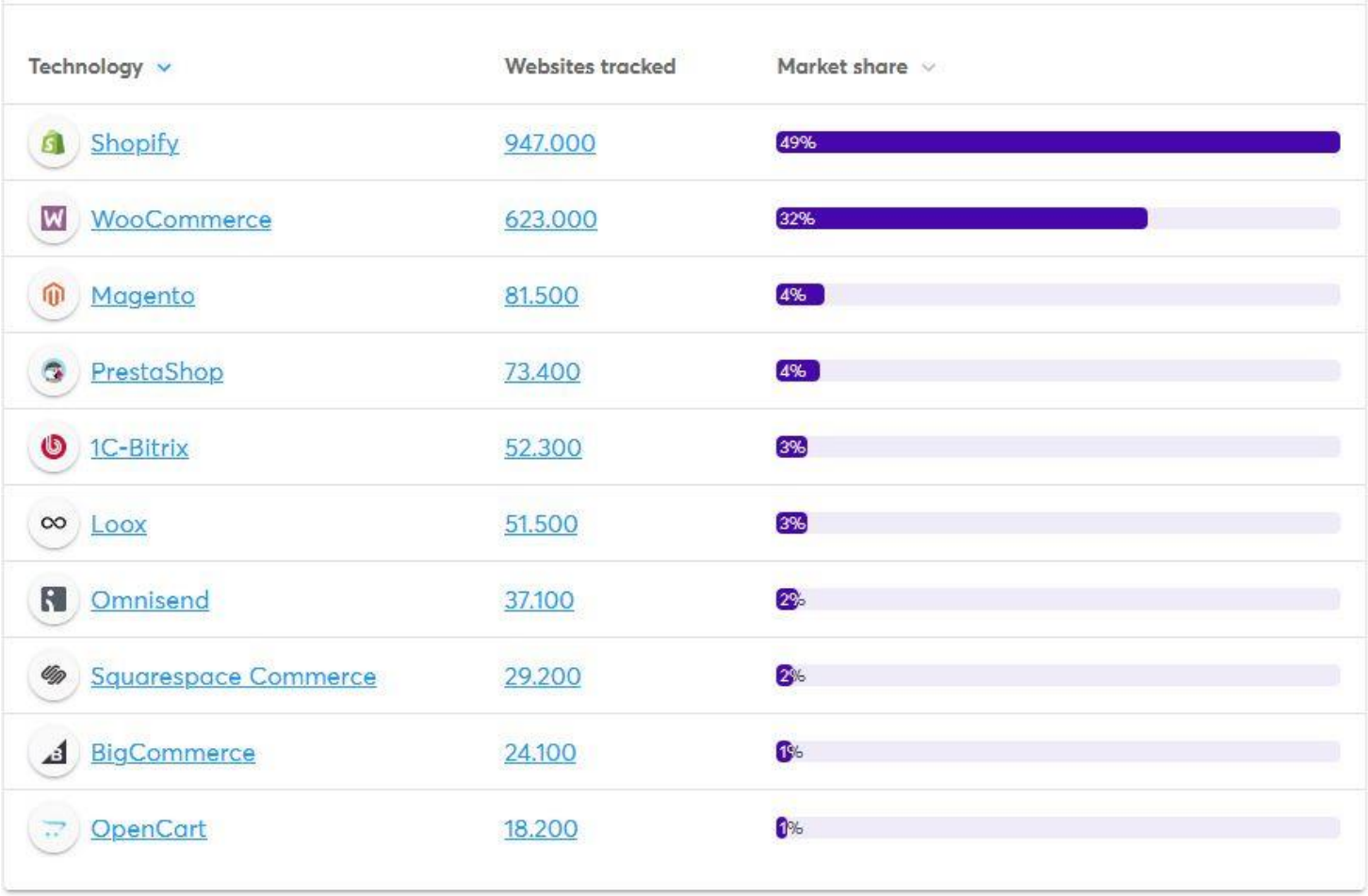


Ilustración 5.1 CMS más utilizados

A continuación, en la ilustración 5.2 se muestra el top 10 de CMS más utilizados actualmente en el año 2021 para el desarrollo de e-commerces.

[11] https://www.wappalyzer.com/



Technology	Websites tracked	Market share
Shopify	947.000	49%
WooCommerce	623.000	32%
Magento	81.500	4%
PrestaShop	73.400	4%
1C-Bitrix	52.300	3%
Loox	51.500	3%
Omnisend	37.100	2%
Squarespace Commerce	29.200	2%
BigCommerce	24.100	1%
OpenCart	18.200	1%

Ilustración 5.2 Top 10 CMS para e-commerces

Como se ha podido observar en las ilustraciones 5.1 y 5.2, los CMS más utilizados para la creación de tiendas online actualmente son Shopify [12], WooCommerce [13] (plugin de WordPress [14]) y Magento [15], muy seguido por PrestaShop[16].

Ya hemos visto la importancia que tienen los CMS en el desarrollo web pero ahora toca conocer qué relación tiene todo esto con las PWA. Como ya hemos visto en algunos ejemplos anteriormente (Twitter, Trivago o Aliexpress) existe una multitud de casos donde el uso de las PWA, sobre todo en tiendas online, ha mejorado en gran medida los resultados obtenidos por las mismas.

[12] https://www.shopify.es/
[13] https://woocommerce.com/
[14] https://es.wordpress.org/
[15] https://magento.com/
[16] https://www.prestashop.com

Lo que significa que si un gran número de tiendas online o e-commerces han sido desarrolladas bajo el uso de CMS hay que convertirlas en aplicaciones PWA y para ello es necesario el uso de plugins.

5.3. ¿Qué son los plugins?

Los plugins pueden ser definidos como un trozo de código o software que es agregado a un CMS (por ejemplo WordPress) y que amplía o añade una funcionalidad al mismo.

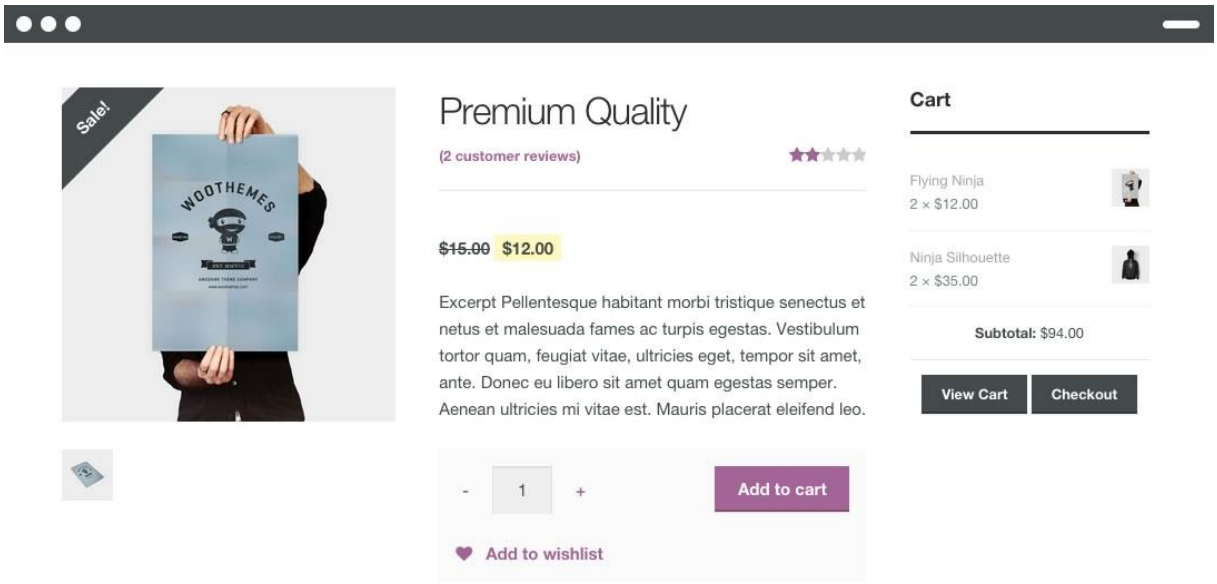


Ilustración 5.3 WooCommerce

Un ejemplo de plugin puede ser WooCommerce, el cual convierte una instalación básica de WordPress en una tienda online con todo lo necesario: carrito, catálogo de productos o la pasarela de pago y sus métodos. En la anterior ilustración 5.3 puede observarse un ejemplo de una web con WooCommerce.

Por tanto, los plugins pueden aumentar en gran medida las posibilidades que puede tener un CMS, tanto como que puede ser capaz de convertir cualquier web en una aplicación PWA.

5.4. Plugins para convertir una web en una PWA

Existe una serie de plugins que al ser instalados en sus CMS correspondientes “transformarán” la web en una aplicación PWA totalmente funcional. A continuación

se mostrarán algunos ejemplos de plugins para cada uno de los CMS más demandados en la actualidad.

5.4.1. Plugins para Shopify

El CMS Shopify fue la primera plataforma de comercio electrónico en ofrecer un plugin para transformar un e-commerce en una PWA, con el plugin Litefy[17], en el año 2017. La ilustración 5.4 muestra la web oficial de este plugin.

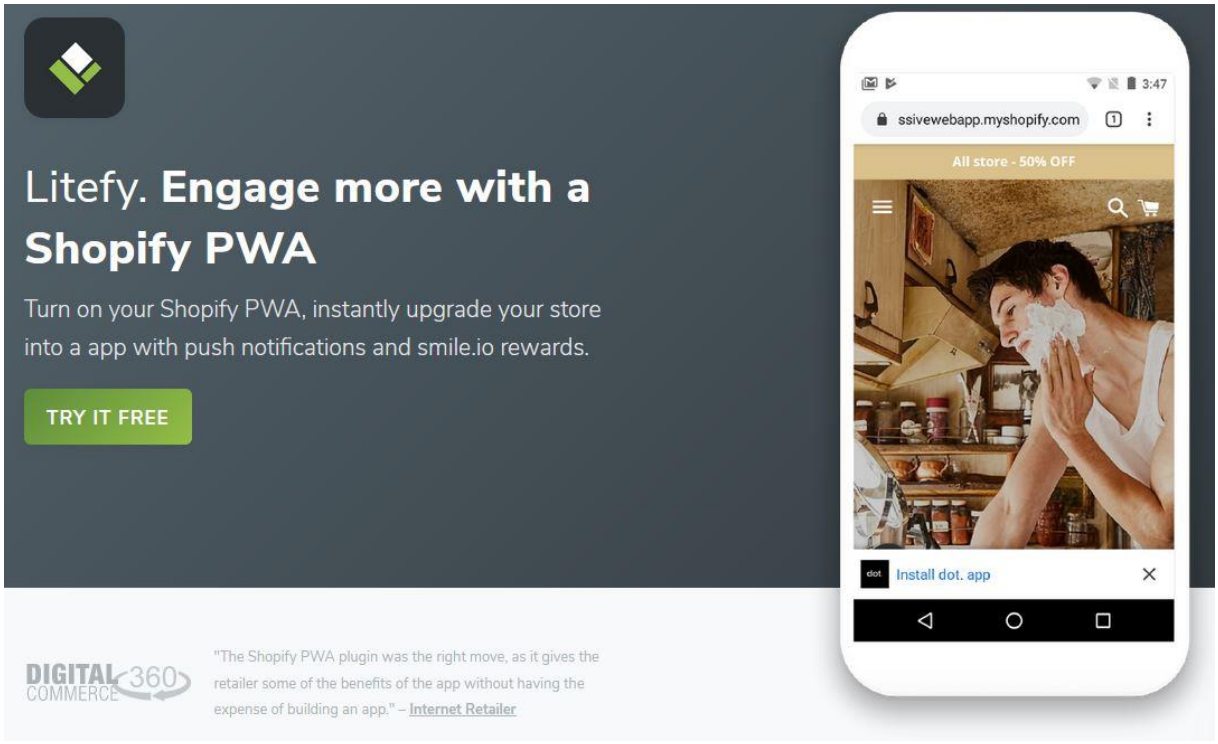


Ilustración 5.4 Litefy

El plugin Litefy es un plugin de pago con tres packs disponibles, pack básico 14,99€/mes, pack shopify 19,99€/mes y pack avanzado 29,99€/mes. Entre sus características más importantes destaca que todas las funciones de la aplicación web son compatibles con el plugin y es compatible con todos los temas de shopify pero aun no está disponible la opción de enviar notificaciones push.

[17] https://litefy.com/

Ilustración 5.5 PWA, Mobile App & Web Push

El plugin PWA, Mobile App & Web Push[18] de shopify cuenta con un plan gratuito de prueba durante 7 días. Además de convertir la aplicación web en una PWA, incluye algunas opciones para mejorar el SEO, implementa Analytics de Google[19] que permite obtener información útil como el número de instalaciones y la forma en que interactúan los usuarios entre otros y ofrece la posibilidad de notificaciones push en dispositivos móviles. En la ilustración 5.5 puede observarse la web de dicho plugin.

5.4.2. Plugins para WordPress

Algunos de los plugins más famosos para convertir un WordPress en una PWA [23] son:

Ilustración 5.6 PWA for WP & AMP

El plugin PWA for WP & AMP[20] es un plugin gratuito que ofrece al usuario de la aplicación web una experiencia muy similar a una aplicación nativa, con la diferencia

[18] https://apps.shopify.com/progressify-me-1
[19] https://analytics.google.com/analytics/web/
[20] https://es.wordpress.org/plugins/pwa-for-wp/

de que la aplicación web puede instalarse y funcionar como una aplicación nativa sin necesidad de conexión a internet. En la anterior ilustración 5.6 puede observarse algunas de las características del plugin.

Ilustración 5.7 Super Progressive Web App

Super PWA[21] es un plugin gratuito de Wordpress que permite instalar la aplicación web en dispositivos móviles, siempre y cuando la aplicación cuente con HTTPS. La ilustración 5.7 muestra el logo del plugin.

Ilustración 5.8 PWA

Por último, el plugin PWA[22] es una aplicación gratuita de Wordpress que no convierte una web en PWA, como los ejemplos anteriores, sino que proporciona una API que permite comprobar si una web basada en Wordpress soporta HTTPS y podría ser "convertida" en una PWA, ya que debe comprobarse si el tema y plugins de la web son compatibles con esta transformación. El logo del plugin puede verse en la ilustración 5.8.

[21] https://es.wordpress.org/plugins/super-progressive-web-apps/
[22] https://es.wordpress.org/plugins/pwa/

5.4.3. Plugins para Magento

Por supuesto, Magento también cuenta con su plugin que ofrece una PWA al servicio de cualquier web basada en su CMS.

Ilustración 5.9 Magento PWA Studio

Adobe no ofrece mucha información sobre el precio ni las estadísticas de aplicaciones que usan el plugin PWA Studio[23]. Sin embargo, podemos observar en su web que el plugin permite personalizar la experiencia de compra en un dispositivo móvil de forma sencilla, algo muy interesante para los sistemas basados en Magento, que es un CMS especializado en tiendas online. En la ilustración 5.9 puede verse un pantallazo de la web oficial de PWA Studio.

6. Tecnologías utilizadas

A lo largo de este apartado mostraré las diferentes tecnologías y software utilizado para el desarrollo del proyecto, tanto para la implementación del software como para el diseño de imágenes, entre otros.

6.1. Phaser

Me he decantado por Phaser[24] (es un lenguaje para videojuegos) por su facilidad de uso y su lenguaje de programación, ya que se puede utilizar JavaScript (que es un lenguaje que ya conozco) junto a código HTML5. La ilustración 6.1 muestra el icono de Phaser.

[23] https://magento.com/products/magento-commerce/pwa
[24] https://phaser.io/

Ilustración 6.1 Icono framework Phaser

Este framework proporciona ciertas características ya implementadas como son las físicas, sistema de cámaras, sistema de entrada (ratón, táctil, teclado...), la administración del sonido, animaciones y muchas más. [24]

Una vez indicado por qué utilizo el framework Phaser, hablaré de sus clases más importantes:

- **Game.** Controlador principal del juego y su estructura puede ser similar a la mostrada en la ilustración 6.2:

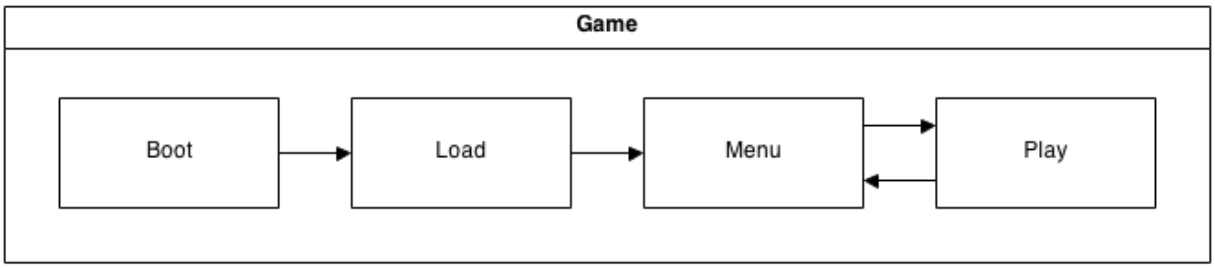


Ilustración 6.2 Estructura clase Game de Phaser

- **State.** Los states proporcionan acceso rápido a las funciones más comunes, como pueden ser los inputs, la cámara o la caché entre otros. Tiene una estructura similar a la ilustración 6.3. Se componen de unos métodos básicos para su funcionamiento y los más importantes son:
  - **Init.** Es el primer método en ser llamada y se utiliza para inicializar las variables u objetos necesarios.
  - **Preload.** Se suele utilizar para cargar los elementos necesarios del juego, como sonidos o imágenes.
  - **Create.** Se llama una vez que preload ha finalizado.

- **Render.** Sirve para renderizar el juego, pero no es necesario su uso ya que se renderiza de forma automática.
- **Update.** Será donde se agregue la lógica del state.

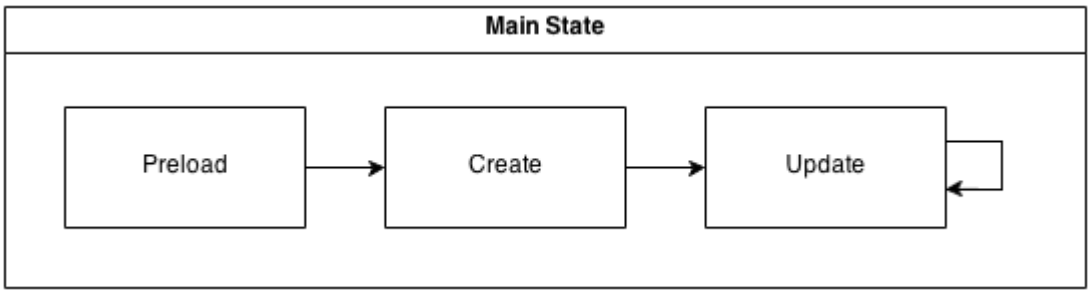


Ilustración 6.3 Estructura clase State de Phaser

- **Text.** Crea un objeto de tipo Text para visualizar texto.
- **Image.** Crea un objeto de tipo Imagen para visualizar imagen.
- **Sprite.** Crea un objeto de tipo Sprite que será utilizado para la parte visual del videojuego [25].

6.1.1. Crear un proyecto en Phaser desde cero

Todo el código fuente de la aplicación ha sido escrito con el editor de código Visual Studio Code.

Para iniciar el proyecto es necesario crear una carpeta vacía con el nombre del proyecto, en mi caso “Test” y a continuación hay que crear un fichero index.html con la estructura básica de un fichero html.

Una vez creado el fichero index.html creamos una carpeta llamada “src” que contendrá el código del framework Phaser y el código de nuestro proyecto. Por tanto, nos descargamos el archivo phaser.min.js de la web oficial del Phaser y lo añadimos a la carpeta “src” del proyecto.

A continuación creamos un nuevo fichero “main” con extensión .js, que será la clase principal del proyecto, y agregamos estos dos scripts al archivo index.html. Un código base para el fichero index.html podría ser el mostrado en la ilustración 6.4.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8"/>
```

```
        <meta name="viewport" content="width=device-width, initial-scale=1,
        maximum-scale=1, minimum-scale=1, user-scalable=no" />
        <title>Test - PHASER</title>
    </head>
    <body>
        </div>
        <div id="contenedor"></div>
        <script src="src/phaser.min.js"></script>
    </body>
</html>
```

Ilustración 6.4 Código base fichero index.html

Para continuar con la configuración base de Phaser es necesario acceder al archivo main.js y agregar los parámetros de configuración del juego. Estos parámetros serán diferentes para cada juego pero a continuación mostraré una configuración base de ejemplo. La ilustración 6.5 muestra una posible configuración.

```
const config = {
    title: 'Aplicacion de ejemplo - Framework Phaser',
    scale: {
        mode: Phaser.Scale.FIT,
        autoCenter: Phaser.Scale.CENTER_BOTH,
        parent: 'contenedor',
        width: 180,
        height: 320,
    },
    url: 'https://urldeejemplo.com',
    version: '0.0.1',
    type: Phaser.AUTO,
    backgroundColor: '#6F82ED'
};

const game = new Phaser.Game(config);
```

Ilustración 6.5 Código ejemplo configuración juego Phaser

Para seguir codificando la aplicación podemos descargar las autodefiniciones de Phaser que servirán de ayuda para el autocompletado. Para descargarlo hay que acceder al repositorio oficial de Phaser[25], descargar el archivo y agregarlo a una nueva carpeta del proyecto llamada “def”.

Por último, antes de finalizar la configuración de la aplicación Phaser es necesario instanciar el juego. Para ello, abrimos el archivo main.js que se creó anteriormente en la carpeta src y agregamos la siguiente línea “*const game = new*

[25] https://raw.githubusercontent.com/photonstorm/phaser/master/types/phaser.d.ts

Phaser.Game(config)". Ahora ya podemos lanzar la aplicación y comprobar que funciona correctamente.

Para ello, yo tengo instalado xampp en mi ordenador y funciona exactamente igual que si lanzáramos una web cualquiera. Arrancamos xampp y accedemos a la URL "localhost/Test/" donde podremos ver nuestra primera aplicación Phaser tal y como se muestra en la ilustración 6.6.

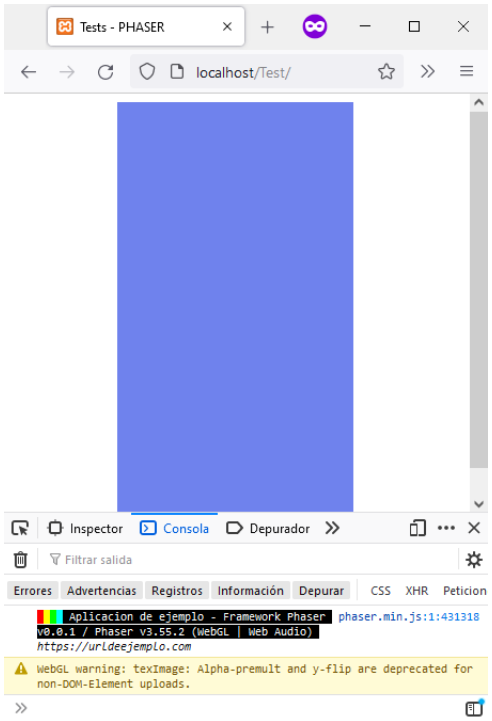


Ilustración 6.6 Aplicación de ejemplo Phaser

6.1.2. Convertir aplicación Phaser en PWA

En el capítulo anterior hemos visto cómo crear un proyecto con el framework Phaser desde cero pero aún no hemos visto cómo podemos convertir la aplicación desarrollada con Phaser en una PWA [26]. Para ello, es posible convertir el juego en una PWA sin utilizar software ni herramientas de terceros. Solo será necesario agregar algunos archivos al proyecto.

Comenzamos el proceso agregando un service worker al proyecto y para ello debemos crear un nuevo archivo llamado "sw.js" en la raíz del proyecto, que dejaremos vacío por ahora.

A continuación, creamos el fichero “load-sw.js” con el código indicado en la ilustración 6.7 y lo cargamos desde el index.html como si se tratara de cualquier otro fichero javascript.

```
if('serviceWorker' in navigator) {
    window.addEventListener('load', function() {
        navigator.serviceWorker.register('/Test/sw.js', {scope: '/Test/'}).the
n(function(registration) {
            console.log('ServiceWorker registration successful with scope: ',
registration.scope);
        }, function(err) {
            console.log('ServiceWorker registration failed: ', err);
        });
    });
}
```

Ilustración 6.7 Código carga service worker (fichero load-sw.js)

Una vez hemos comprobado que la carga del service worker funciona correctamente procedemos a configurar el service worker para que caché los archivos que deseemos de la aplicación. El cacheado de estos ficheros es necesario para poder utilizar la aplicación sin conexión a internet.

En este caso los archivos que queremos cachear son los archivos javascript y html del proyecto (index.html, main.js, phaser.min.js…). Para ello utilizaremos dos variables (myCache y files).

La variable myCache indicará la versión de la caché almacenada en el navegador. Nos servirá, entre otros, para identificar si la caché que se encuentra en el navegador actualmente necesita ser actualizada o no. Mientras que la variable files es un vector de los archivos que queremos almacenar en la caché del navegador.

Además de las variables indicadas será necesario un pequeño código que accederá a la caché del navegador, guarda la lista de ficheros en ella y confirma si se han guardado correctamente o no. El código necesario para este paso es el mostrado a continuación en la ilustración 6.8.

```
ar myCache = 'testApp-v1';
var files = [
    './',
    './index.html',
    './manifest.webmanifest',
```

```
    './def/phaser.d.ts',
    './src/load-sw.js',
    './src/main.js',
    './src/phaser.min.js',
    './assets/icon-192.png',
    './assets/icon-256.png',
    './assets/icon-512.png'
];

self.addEventListener('install', function(event) {
    console.log('sw instalado');
    event.waitUntil(
        caches.open(myCache).then(function(cache) {
            console.log('sw cacheando ficheros');
            return cache.addAll(files);
        }).catch(function(error) {
            console.log(error)
        })
    )
});
```

Ilustración 6.8 Código para guardar ficheros en caché (sw.js)

Una vez almacenados los ficheros necesarios en la caché, necesitamos hacer uso de los mismos. Para ello es necesario agregar el siguiente código, mostrado en la ilustración 6.9, al service worker el cuál se activará cada vez que un usuario visite la página web y se haya instalado el service worker en el navegador (no requiere ninguna acción por parte del usuario).

```
self.addEventListener('fetch', (event) => {
    console.log('sw fetch');
    console.log(event.request.url);
    event.respondWith(
        caches.match(event.request).then(function(response) {
            return response || fetch(event.request);
        }).catch(function(error) {
            console.log(error);
        })
    );
});
```

Ilustración 6.9 Código para utilizar ficheros en caché (sw.js)

Por último, es posible en algunas ocasiones sea necesario limpiar las cachés antiguas en el dispositivo del usuario y para ello debemos agregar el siguiente código mostrado en la ilustración 6.10.

```
self.addEventListener('activate', function(event) {
    console.log('sw activate');
    event.waitUntil(
```

```
        caches.keys().then(function(keyList) {
            return Promise.all(keyList.map(function(key) {
                if(key !== myCache) {
                    console.log('sw removing old cache', key);
                    return caches.delete(key);
                }
            }));
        })
    );
});
```

Ilustración 6.10 Código para limpiar caché (sw.js)

Una vez terminada la configuración del service worker hay que configurar el archivo manifest de la aplicación, utilizando un formato json, tal y como se muestra en el código de ejemplo de la ilustración 6.11.

Pese a tener un formato json el consorcio W3C recomienda que el manifest tenga la extensión .webmanifest.

```
{
    "short_name": "Phaser ejemplo",
    "name": "Aplicacion de ejemplo - Framework Phaser",
    "icons": [
      {
        "src": "assets/icon-192.png",
        "type": "image/png",
        "sizes": "192x192"
      },
      {
        "src": "assets/icon-256.png",
        "type": "image/png",
        "sizes": "256x256"
      },
      {
        "src": "assets/icon-512.png",
        "type": "image/png",
        "sizes": "512x512"
      }
    ],
    "start_url": ".",
    "display": "standalone",
    "description": "Aplicacion de ejemplo con el framework Phaser en HTML5."
}
```

Ilustración 6.11 Código ejemplo manifest (manifest.webmanifest)

Una vez tenemos el fichero manifest generado hay que enlazarlo desde la página index.html, por lo que es necesario agregar el cógido de la ilustración 6.12 y

ya quedaría convertida nuestra aplicación desarrollada con Phaser en una aplicación PWA.

```
<link rel="manifest" href="manifest.webmanifest" />
```

Ilustración 6.12 Enlaza el fichero manifest (index.html)

6.1.3. Mostrar botón para instalar aplicación a2hs

Ya hemos convertido nuestra aplicación en una PWA y ahora es momento de hacerle saber al usuario que lo que tiene delante es una PWA que puede ser instalada y para ello voy a crear una ventana emergente desde la que instalar la aplicación. Esto se conoce como a2hs (add to home screen).

Lo primero que debemos hacer es agregar el contenido html tal y como se ve en la siguiente ilustración 6.13 en el body de la vista html.

```
<div id="ventana-instalar">
      <div class="ventana-contenido">
            <span class="cerrar">&times;</span>
            <p>¿Quiere instalar la aplicación?</p>
            <button class="btn btn-
azul" onclick="offlinePrompt()">Instalar</button>
            <button class="btn btn-blanco cerrar">Cancelar</button>
      </div>
</div>
```

Ilustración 6.13 Código html para implementar botón instalar pwa (index.html)

Una vez tenemos el código html ya implementado le podemos aplicar una serie de estilos CSS para que quede visualmente más atractivo, tal y como se muestra en la ilustración 6.14.

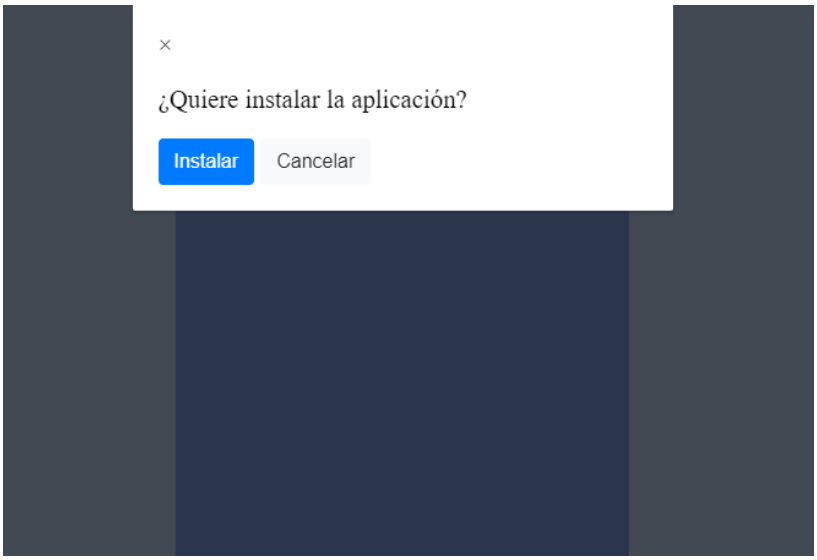


Ilustración 6.14 Ejemplo botón instalar pwa

Y por último implementamos el código javascript asociado a los botones de instalar y cancelar para que realicen las operaciones oportunas en cada caso. Este código queda indicado en la ilustración 6.15.

```
let deferredPrompt;
var ventana = document.getElementById('ventana-instalar');
window.addEventListener('beforeinstallprompt', function (e) {
    console.log('beforeinstallprompt triggered');
    e.preventDefault();
    deferredPrompt = e;
    ventana.style.display = 'block';
});
// Cerrar
var cerrar = document.getElementsByClassName('cerrar')[0];
cerrar.onclick = function() {
    ventana.style.display = 'none';
}
var cancelar = document.getElementsByClassName('cerrar')[1];
cancelar.onclick = function() {
    ventana.style.display = 'none';
}
window.onclick = function(event) {
    if(event.target == ventana) {
        ventana.style.display = 'none';
    }
}
// Instalar
function offlinePrompt() {
    ventana.style.display = 'none';
    deferredPrompt.prompt();
}
```

Ilustración 6.15 Código javascript implementación instalar pwa (installable.js)

Una vez completados estos pasos ya tengo una aplicación PWA fácilmente instalable en cualquier dispositivo con navegador web compatible.

Es importante tener en cuenta que tanto en Mozilla Firefox como en otros navegadores no es posible realizar la instalación de las PWA como tal, ya que no son compatibles con esta opción. En caniuse[26] podemos consultar los navegadores compatibles con esta tecnología.

[26] https://caniuse.com/web-app-manifest

El código de ejemplo utilizado en los apartados 6.1.1, 6.1.2 y 6.1.3 estará disponible en mi repositorio[27] de github y puede ser utilizado como código base para cualquier aplicación PWA desarrollada con Phaser.

6.2. Jira para la administración de tareas

Jira [28] es una herramienta online desarrollada por la empresa australiana Atlassian que nos facilita la administración de tareas de los proyectos. Es capaz de gestionar todo tipo de casos de uso, requisitos software y casos de prueba.

El software Jira [27] viene especialmente bien para mi proyecto ya que proporciona tableros kanban listos para usar. A continuación, tal y como muestra la ilustración 6.16, podemos ver un ejemplo de un tablero kanban generado por Jira.

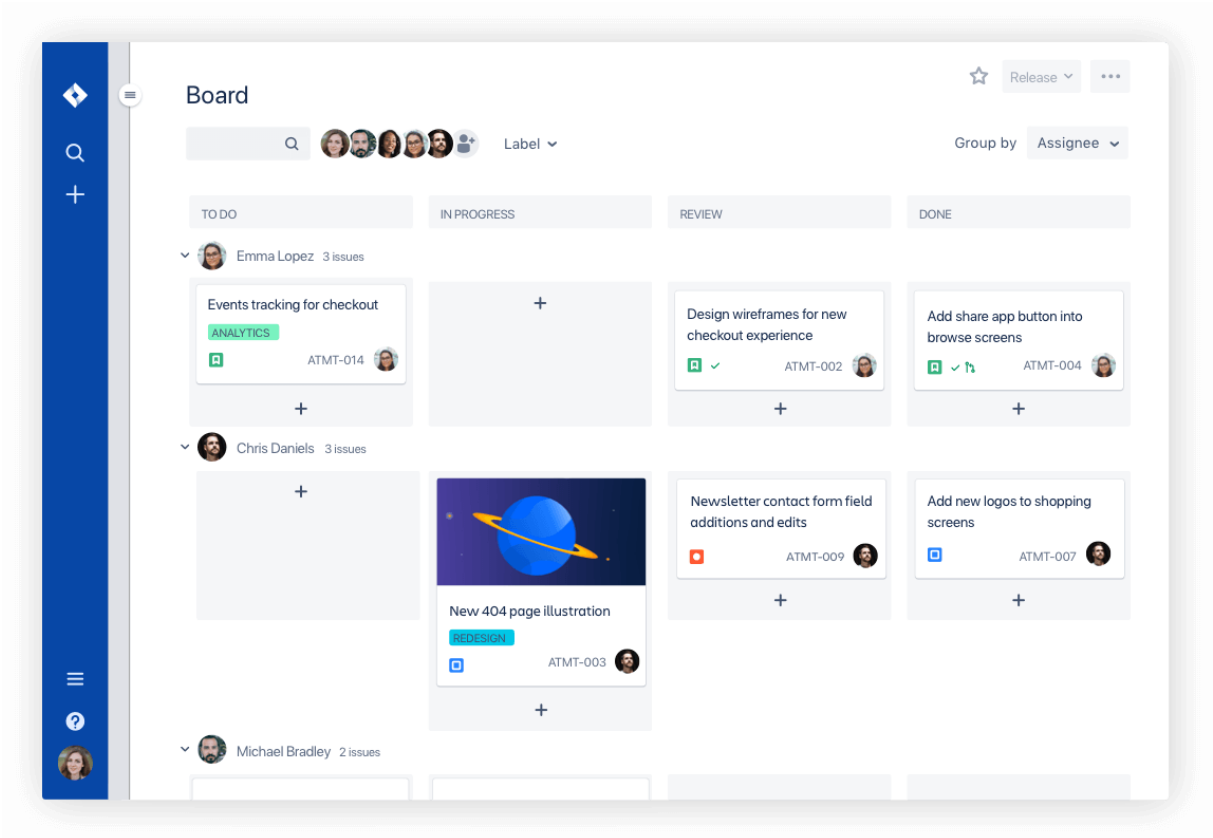


Ilustración 6.16 Tablero Jira

[27] https://github.com/jcme0003/CodigoBasePhaserPWA.git
[28] https://www.atlassian.com/es/software/jira

6.3. GIT para el control de versiones

Los sistemas de control de versiones buscan gestionar los proyectos de una forma ágil, que nos permita volver a un estado anterior o ser testigo de todas y cada una de las versiones lanzadas hasta obtener el producto final [28].

En mi caso, GIT, más concretamente la platadorma Github[29], me permitirá tener el código siempre actualizado a la última versión e ir haciendo entregables cada vez que se complete un hito importante.

6.4. Otras tecnologías

Para la parte visual del videojuego, así como la edición o personalización de botones, se ha utilizado el programa Adobe Photoshop[30].

La herramienta Visual Paradigm[31] la he utilizado para la creación de los Storyboards de la aplicación.

7. Desarrollo del software

A lo largo de este apartado hablaré sobre el proyecto que he llevado a cabo y todo el proceso de desarrollo del mismo haciendo uso de las tecnologías PWA y Phaser. Si quieres saber cómo crear un proyecto en Phaser desde cero y convertirlo en una PWA visita el apartado 6.1.1 y 6.1.2 donde se describe todo el proceso paso a paso.

En mi caso, he desarrollado una aplicación para demostrar la potencia y ventajas que tiene el uso de las PWA sobre las aplicaciones nativas o híbridas. Este ejemplo será un videojuego al que podremos jugar tanto en dispositivos móviles como en ordenadores y la tecnología usada para el desarrollo del mismo es el framework Phaser[32].

La aplicación desarrollada es el famoso juego de “busca las parejas” basado en la versión del tradicional juego con cartas [29]. El juego comienza con todas las

[29] https://github.com/
[30] https://www.adobe.com/es/products/photoshop.html
[31] https://www.visual-paradigm.com/
[32] https://phaser.io/

cartas boca abajo y se van volteando de dos en dos, si las dos cartas volteadas son las mismas se quedan boca arriba y en caso contrario se vuelven a poner boca abajo.

El objetivo del juego es conseguir encontrar todas las parejas en el mínimo tiempo posible, ya que el juego incluirá la opción de consultar la clasificación de los mejores 10 resultados.

7.1. Tareas a realizar

Las tareas que se realizarán para el proyecto de “Busca las parejas” serán las que se indican en la tabla 7.1.

Para organizar y priorizar el desarrollo de unas tareas sobre otras utilizaré la técnica MoSCoW [30], que es un sistema de priorización de tareas. Establece 4 estados que corresponden a cada una de las letras mayúsculas de su nombre, es decir:

- M: “tiene que estar”. Aquellas tareas que se encuentres en este punto es una tarea que debe realizarse si o si, ya que es una tarea necesaria para que el producto, en este caso una aplicación, sea un éxito.
- S: “debería estar”. No son tareas imprescindibles para el correcto funcionamiento pero el producto, en su versión final, debe incluir estas tareas. En casos muy extremos podrían ser prescindible
- C: “podría estar”. Son aquellas tareas que nos gustaría que estuviesen en la versión final del producto pero que no son obligatorias para el correcto funcionamiento del mismo. Pueden ser eliminadas fácilmente si no hay recursos para realizarlas.
- W: “no estará”. Requisitos rechazados desde un primer momento, que podrían implementarse en un futuro si se reclasificaran en alguna de las categorías anteriores.

La principal ventaja de esta técnica MoSCoW es que permite organizar los recursos de forma más eficiente.

Tareas	Subtareas	Valor

Definir paleta de colores del juego	Definir paleta de colores	M
Definir páginas del juego	Crear la página de inicio del juego	M
	Crear página menú principal	M
	Crear página tablero	M
	Crear página ganador con puntuación	S
	Crear página clasificación	S
Agregar iconos y efectos visuales	Añadir icono aplicación	M
	Añadir imágenes e iconos	M
	Agregar fuentes	M
Agregar efectos de sonido	Añadir sonidos al juego	C
Realizar lógica del juego	Comprobar si dos cartas son iguales	M
	Obtener puntuación final	M
	Voltear las cartas	M
Lógica del proceso de clasificación	Mostrar clasificación de jugadores	S
	Ordenar clasificación por puntos	S
Apartado de opciones	Crear ventana de opciones	C
	Codificar funcionalidad de botones y opciones	C
	Crear ventana información sobre el juego	C

Tabla 7.1 Tareas Proyecto Busca las parejas

7.2. Entregable 1

El proceso de implementación se divide en diferentes entregables que se irán completando cuando se codifiquen las tareas seleccionadas.

7.2.1. Tareas entregable

Para este primer entregable se definirá la paleta de colores y se desarrollarán las páginas de la aplicación, ofreciéndole al jugador la posibilidad de navegar entre ellas mostrándole un contenido de ejemplo.

Tareas	Subtareas	Valor
Definir paleta de colores del juego	Definir paleta de colores	M
Definir páginas del juego	Crear la página de inicio del juego	M
	Crear página menú principal	M
	Crear página tablero	M
	Crear página ganador con puntuación	S
	Crear página clasificación	S

Tabla 7.2 Tareas Proyecto Busca las parejas

7.2.2. Storyboards entregable

Para este primer entregable del proyecto se han creado las storyboards mostradas en la ilustración 7.1, que representarán el aspecto visual que tendrá el proyecto al finalizar el primer entregable.

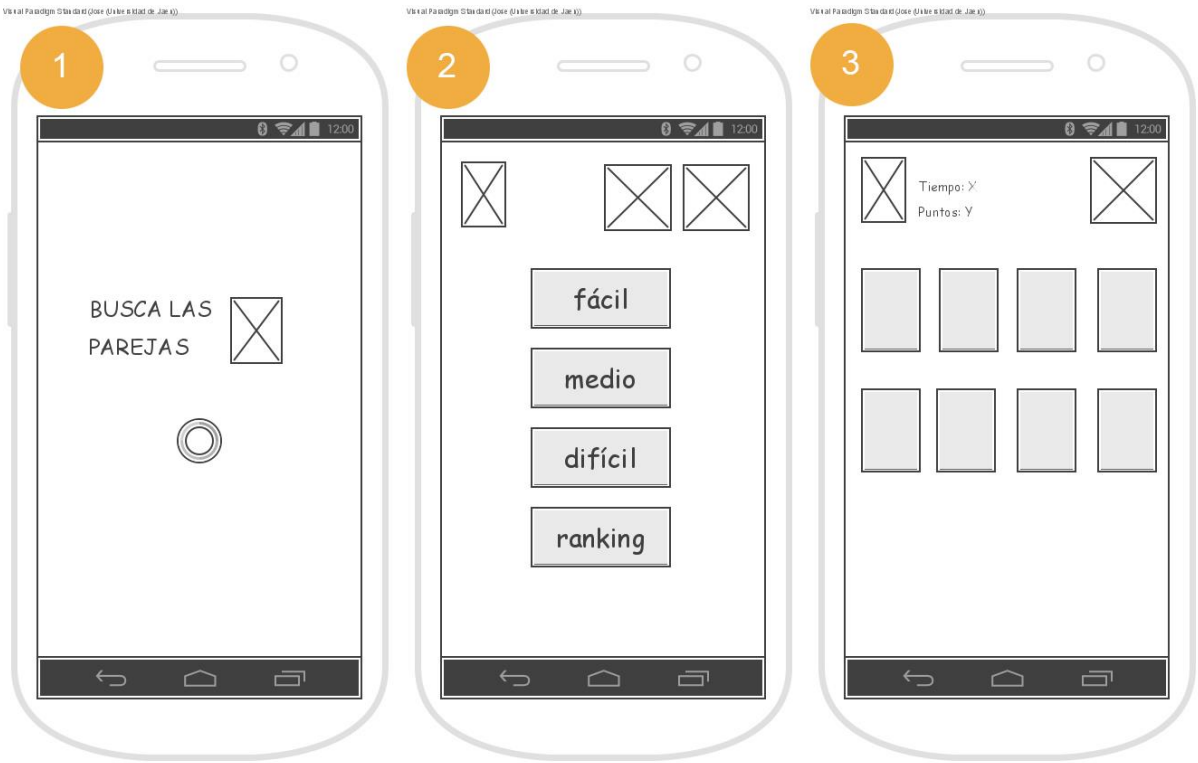


Ilustración 7.1 Storyboards Busca las parejas

Podemos observar 3 “vistas” claramente diferenciadas; 1- la vista inicial que da la bienvenida a la aplicación y se realiza la precarga de los elementos de la aplicación. Visualmente, la página principal consistirá en una página con el color principal de la aplicación como color de fondo, el nombre de la aplicación en color blanco junto al logo y un icono de carga. 2.- se muestra la escena de bienvenida donde se podrá seleccionar el nivel de dificultad, además de otras opciones que se desarrollarán en siguientes entregables. 3- el tablero donde se jugará la partida, se muestra el tiempo que está durando la partida y el número de puntos acumulados hasta ahora.

7.2.3. Paleta de colores

Antes de escribir ninguna línea de código hay que decidir la paleta de colores que se va a aplicar en la aplicación. En mi caso los colores que se han seleccionado son los mostrados en la tabla 7.3.

Componente	Color
Fondo	#5ed5fb
Botones del menú	#ffc078
Texto botones del menú	#f7db0e

Texto	#fff

Tabla 7.3 Paleta de colores

7.2.4. Implementación entregable

Para comenzar se creará una aplicación PWA basada en el framework Phaser tal y como se explicó anteriormente en el apartado 6.1.1 llamada “BuscaParejas”.

Como ya he definido los colores de la aplicación voy a desarrollar las diferentes páginas de la misma para lo que necesitamos crear diferentes clases javascript (Bootloader, Preload, Menu…) asignadas como escenas de Phaser.

La página de inicio es la clase Bootloader de mi aplicación, es la escena principal que precargará todas las imágenes, fuentes, audios y cualquier tipo de recurso necesario para el correcto funcionamiento de la aplicación. La clase Bootloader no tiene una interfaz visual.

Una vez precargado un recurso en el Bootloader no será necesario volver a cargarlo. Esta precarga se ha realizado como se muestra en la ilustración 7.2.

```
preload() {
      this.load.setPath('./assets/');
      this.load.image(['titulo-
preload', 'logo', 'menu', 'facil', 'medio', 'dificil', 'clasificacion', 'bck-
card-1']);
      this.load.atlas('sprite-loading-bar', 'sprite-loading-bar.png', 'sprite-
loading-bar_atlas.json');
      this.load.animation('loadingBarAnim', 'sprite-loading-bar_anim.json');
      this.load.image('font', 'fonts/font.png');
      this.load.json('fontConfig', 'fonts/font.json');

      this.load.on('complete', () => {
            const fuente = this.cache.json.get('fontConfig');
            this.cache.bitmapFont.add('futilePro', Phaser.GameObjects.RetroFon
t.Parse(this, fuente));
            this.scene.start('Preload');
      });
}
```

Ilustración 7.2 Precarga de recursos (Bootloader.js)

Como se puede observar en la ilustración anterior 7.2 una vez que se ha completado la carga de todos los recursos se lanza la escena Preload. Esta escena será la página inicial que se mostrará en la aplicación simulando el inicio de la misma mostrando el logo y un reloj de carga.

Una vez cargados los recursos, y lanzada la escena Preload, es hora de agregar la imagen principal y el sprite animado que simula la carga de la aplicación. El código necesario para esto es el indicado en la ilustración 7.3.

```
create() {
    this.titulo = this.add.image(220, 300, 'titulo-preload');
    this.loadingBar = this.add.sprite(220, 400, 'sprite-loading-bar');
    this.loadingBar.anims.play('loading-bar-start');
    setTimeout(() => {
        this.scene.start('Menu');
    }, 4000);
}
```

Ilustración 7.3 Agregar título y sprite de carga animado (Preload.js)

En mi caso, simulo la carga de la aplicación durante 4 segundos antes de lanzar la escena de menú principal donde se mostrarán las opciones del juego. Una vez cargada la escena del menú principal creo sus elementos (logo, texto y botones) pero con una diferencia a los creados anteriormente y es que en esta escena voy a agrupar los elementos en un contenedor para poder aplicar efectos y acciones a todos los elementos contenidos en el contenedor.

Para ver el código necesario para agrupar el texto del menú principal y los botones en un contenedor vaya a la ilustración 7.4.

```
this.texto = this.add.bitmapText(0, 0, 'futilePro', 'NUEVA PARTIDA', 16, 1);
this.facil = this.add.image(0, 0, 'facil').setInteractive();
this.medio = this.add.image(0, 0, 'medio').setInteractive();
this.dificil = this.add.image(0, 0, 'dificil').setInteractive();
this.clasificacion = this.add.image(0, 0, 'clasificacion').setInteractive();

const opciones = this.add.container(this.centroCanvas.width, 0);
opciones.add([
    this.texto,
    this.facil,
    this.medio,
    this.dificil,
    this.clasificacion
]);
```

Ilustración 7.4 Agrupar elementos a un contenedor (Menu.js)

Es muy importante agregar la propiedad de interactividad, con el método setInteractive(), a las imágenes que puedan ser pulsadas por el usuario y realizar alguna acción tras el click, ya que sin esta propiedad no se produciría ninguna acción.

Uno de los beneficios que tiene el agregar diferentes elementos a un contenedor es la posibilidad de añadir animaciones a todos los elementos de forma más limpia y elegante en el código. En la ilustración 7.5 se puede observar cómo se agregar una animación a las opciones del menú en muy pocas líneas de código.

```
this.add.tween({
    targets: [opciones],
    ease: 'Bounce',
    y: 150,
    duration: 1250
});
```

Ilustración 7.5 Agregar animaciones a un contenedor (Menu.js)

Para alinear, como es en este caso, los botones es necesario indicarlo en el código. Esto lo podemos ver en la ilustración 7.6.

```
Phaser.Display.Align.In.Center(this.texto, cabecera, 0, 0);
Phaser.Display.Align.To.BottomCenter(this.facil, this.texto, 0, 20);
Phaser.Display.Align.To.BottomCenter(this.medio, this.facil, 0, 20);
Phaser.Display.Align.To.BottomCenter(this.dificil, this.medio, 0, 20);
Phaser.Display.Align.To.BottomCenter(this.clasificacion, this.dificil, 0, 20);
```

Ilustración 7.6 Alinear elementos (Menu.js)

Y por último, en la escena del menú principal, faltaría agregar los eventos a los botones del menú principal. Cuando se pulse uno de los botones disponibles correspondientes a la dificultad de la partida “fácil”, “medio” o “difícil” se lanzará la escena Play con un número de cartas determinado, según la dificultad seleccionada. Cuanto mayor sea la dificultad más cartas se mostrarán. También se podría seleccionar la opción “ranking” que mostraría un listado de los mejores jugadores y sus resultados obtenidos.

Para esta primera entrega los eventos del menú de opciones funcionarían de la forma indicada en la ilustración 7.7. Más adelante se modificará este código para que dependiendo de la dificultad se muestre un número de cartas u otro.

```
this.facil.on(Phaser.Input.Events.POINTER_UP, () => {
    this.scene.start('Play');
});

this.medio.on(Phaser.Input.Events.POINTER_UP, () => {
    this.scene.start('Play');
});

this.dificil.on(Phaser.Input.Events.POINTER_UP, () => {
    this.scene.start('Play');
```

```
});

this.clasificacion.on(Phaser.Input.Events.POINTER_UP, () => {
    this.scene.start('Ranking');
});
```

Ilustración 7.7 Eventos botones menú principal (Menu.js)

Como se puede observar en la ilustración 7.7 los eventos de las opciones de dificultad nos llevarán a la escena Play que será donde se muestren las cartas y el jugador comience su partida. El código para crear las cartas en la escena Play es el indicado en la ilustración 7.8.

```
init() {
    this.reparteCartas();
}

reparteCartas() {
    this.add.image(70, 200, 'bck-card-1');
    this.add.image(170, 200, 'bck-card-1');
    this.add.image(270, 200, 'bck-card-1');
    this.add.image(370, 200, 'bck-card-1');
    this.add.image(70, 350, 'bck-card-1');
    this.add.image(170, 350, 'bck-card-1');
    this.add.image(270, 350, 'bck-card-1');
    this.add.image(370, 350, 'bck-card-1');
}
```

Ilustración 7.8 Crear cartas en la escena Play (Play.js)

La escena del ranking únicamente mostrará el contenedor de la cabecera con la opción de volver al menú principal, ya que su contenido final se implementará en las siguientes entregas.

Aunque no se haya comentado nada al respecto, el icono de la aplicación (el que aparece cuando se instala en un dispositivo móvil) se ha agregado en la configuración de manifest y el código correspondiente puede observarse en la ilustración 7.9.

```
"icons": [{
    "src": "assets/icon-192.png",
    "type": "image/png",
    "sizes": "192x192"
},{
    "src": "assets/icon-256.png",
    "type": "image/png",
    "sizes": "256x256"
},{
```

```
    "src": "assets/icon-512.png",
    "type": "image/png",
    "sizes": "512x512"
}]
```

Ilustración 7.9 Cargar iconos aplicación (manifest.webmanifest)

7.2.5. Resultados

Una vez finalizadas las tareas propuestas para este primer entregable se ha subido el código[33] al github a su rama correspondiente ya que se dispone de una aplicación básica funcional con una interfaz propuesta y en la que se pueden empezar a ver detalles o características a mejorar. En la ilustración 7.10 puede observarse cómo se encuentra la aplicación en este punto.

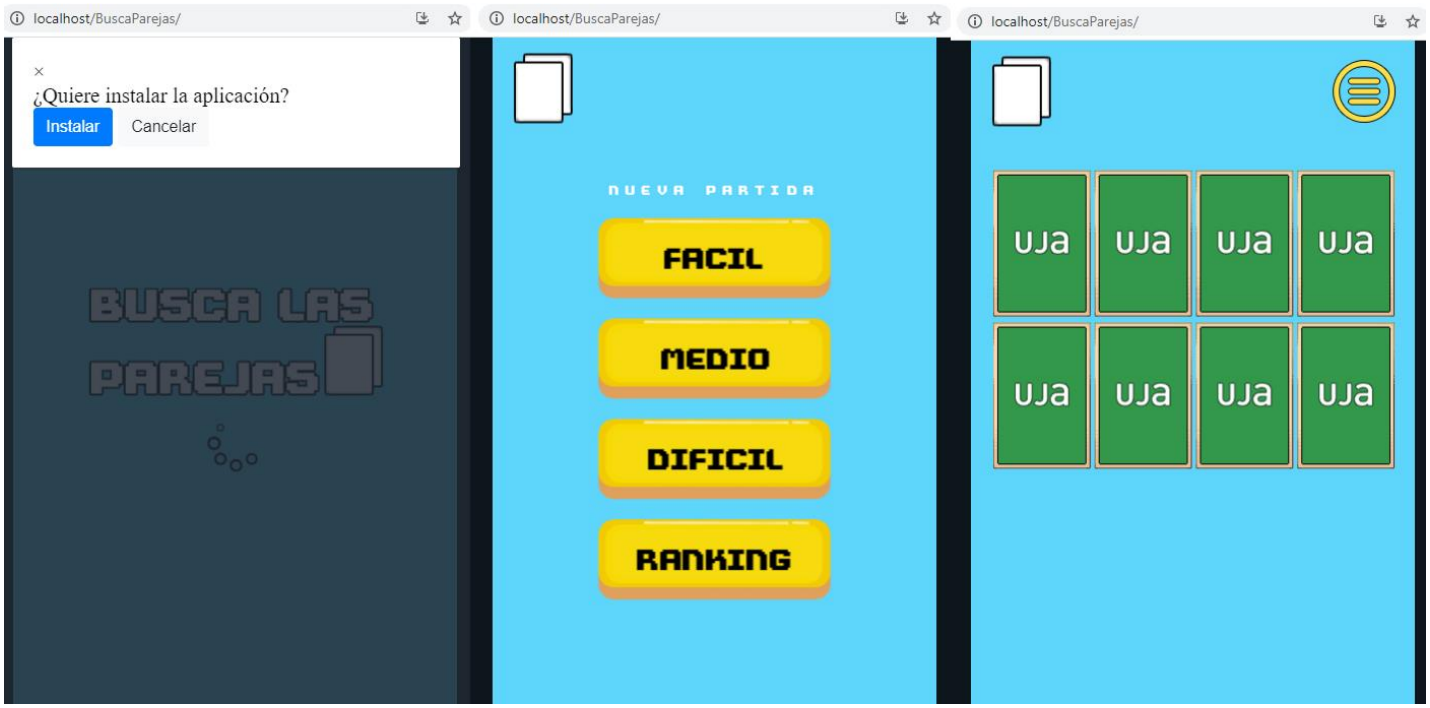


Ilustración 7.10 Interfaz aplicación a la finalización del primer entregable

Puede verse el estado del repositorio de github en el primer entregable en la ilustración 7.11.

[33] https://github.com/jcme0003/BuscaParejas/tree/Primer_entregable/

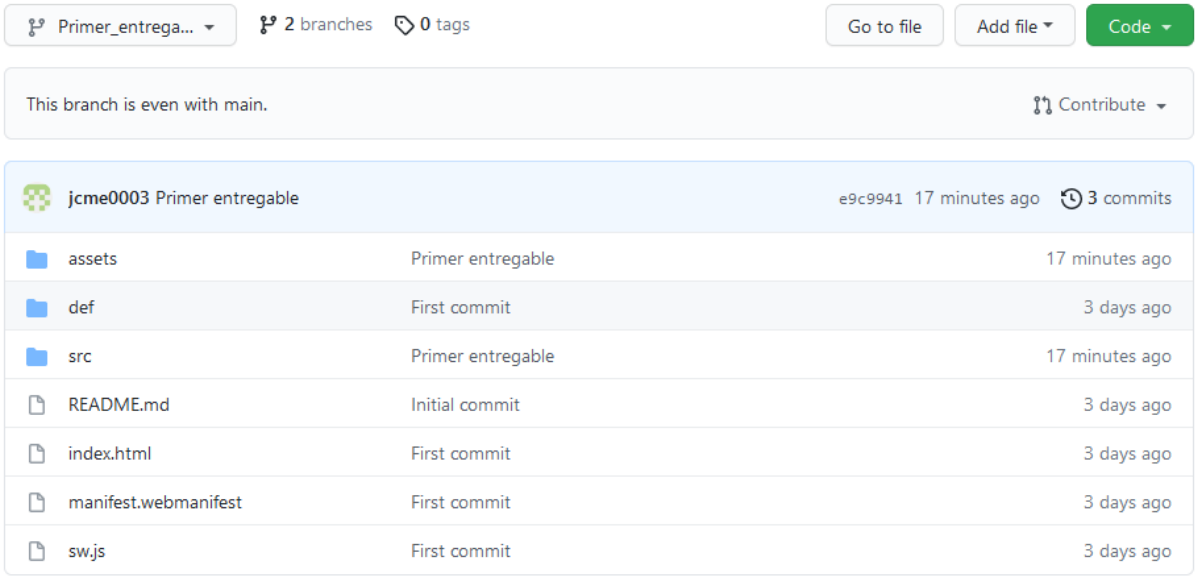


Ilustración 7.11 Estado del repositorio github a la finalización del primer entregable

7.3. Entregable 2

Tras definir las páginas de la aplicación en el primer entregable voy a codificar la lógica del juego para que sea posible disfrutar de la experiencia del juego.

7.3.1. Tareas entregable

Para este segundo entregable se realizarán las tareas necesarias para que el jugador pueda jugar y obtener una puntuación final, tras finalizar la partida.

Tareas	Subtareas	Valor
Realizar lógica del juego	Comprobar si dos cartas son iguales	M
	Obtener puntuación final	M
	Voltear las cartas	M

Tabla 7.4 Tareas Proyecto Busca las parejas

7.3.2. Storyboards entregable

Una vez se finaliza la partida y se obtiene la puntuación final, se lanza una ventana emergente donde se muestra la puntuación final, unas opciones en forma de botones y un campo de texto donde el usuario puede incluir su nombre para el ranking de puntuaciones. Para esta ventana emergente se ha diseñado el storyboard de la ilustración 7.12.

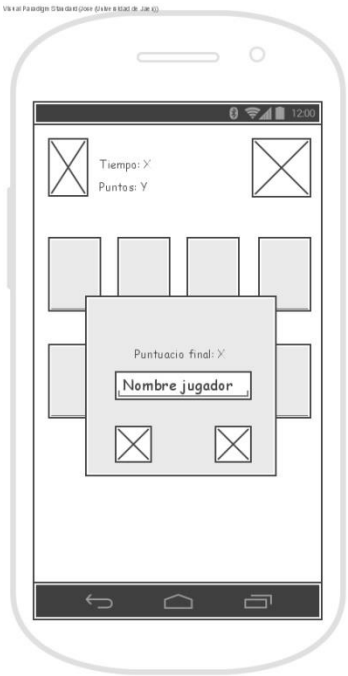


Ilustración 7.12 Storyboards Busca las parejas

7.3.3. Implementación entregable

Para realizar la lógica del juego será necesario compartir una serie de opciones y parámetros de la aplicación y para ello debemos crear una nueva carpeta dentro de la carpeta “src” llamada “share” y un fichero con extensión “.js” con el nombre “options”.

En su interior generaré un código en formato javascript con las opciones compartidas de la aplicación por defecto, con la dificultad de la partida que por defecto será fácil, su tablero correspondiente (con las cartas boca abajo), el tiempo inicializado a cero y el número de parejas encontradas que en un principio es cero. Se puede ver el código correspondiente en la ilustración 7.13.

```
const options = {
    dificultad: 'facil',
    tablero: [
        [-1, -1, -1, -1],
        [-1, -1, -1, -1],
        [-1, -1, -1, -1],
        [-1, -1, -1, -1]
    ],
    maxIntentos: 2
};

export default options;
```

Ilustración 7.13 Configuración compartida (options.js)

La dificultad de la partida como ya se explicó anteriormente, determinará el número de cartas que habrá sobre la mesa y se modificará cuando el usuario o jugador pulse en una de las opciones del menú principal. La ilustración 7.14 muestra cómo se modifica la dificultad según la opción seleccionada.

```
this.facil.on(Phaser.Input.Events.POINTER_UP, () => {
      options.dificultad = 'facil';
      this.scene.start('Play');
});

this.medio.on(Phaser.Input.Events.POINTER_UP, () => {
      options.dificultad = 'medio';
      this.scene.start('Play');
});

this.dificil.on(Phaser.Input.Events.POINTER_UP, () => {
      options.dificultad = 'dificil';
      this.scene.start('Play');
});
```

Ilustración 7.14 Actualizar dificultad (Menu.js)

Es importante que no se nos olvide importar la clase options al principio del fichero con el código de la ilustración 7.15, ya que de no hacerlo se lanzará un error 404 y la aplicación no funcionaría.

```
import options from "../share/options.js";
```

Ilustración 7.15 Importar clase options (Menu.js)

Visualmente en la aplicación no se ha realizado ningún cambio ya que se siguen mostrando igual todas las escenas. Para que el número de cartas sobre la mesa cambie, hay que actualizar el método reparteCartas() de la escena Play.js. El nuevo método quedaría actualizado con el código mostrado en la ilustración 7.16.

```
    /**
     * Pinta cartas en el tablero de juego
     */
    reparteCartas() {
        // facil
        if(options.dificultad === 'facil') {
            this.parejas = 4;
            this.barajaCartas();
            this.tablero = [
                // ARRIBA
                this.add.image(this.centroCanvas.width - 150,
                    this.centroCanvas.height - 150,
                    'bck-card-1'),
                this.add.image(this.centroCanvas.width - 50,
```

```
                this.centroCanvas.height - 150,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 50,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 150,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            // ABAJO
            this.add.image(this.centroCanvas.width - 150,
                this.centroCanvas.height - 25,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width - 50,
                this.centroCanvas.height - 25,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 50,
                this.centroCanvas.height - 25,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 150,
                this.centroCanvas.height - 25,
                'bck-card-1')
        ];
    }

    // medio
    if(options.dificultad === 'medio') {
        this.parejas = 6;
        this.barajaCartas();
        this.tablero = [
            // ARRIBA
            this.add.image(this.centroCanvas.width - 150,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width - 50,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 50,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 150,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            // MEDIO
            this.add.image(this.centroCanvas.width - 150,
                this.centroCanvas.height - 25,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width - 50,
                this.centroCanvas.height - 25,
                'bck-card-1'),
```

```
            this.add.image(this.centroCanvas.width + 50,
                this.centroCanvas.height - 25,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 150,
                this.centroCanvas.height - 25,
                'bck-card-1'),
            // ABAJO
            this.add.image(this.centroCanvas.width - 150,
                this.centroCanvas.height + 100,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width - 50,
                this.centroCanvas.height + 100,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 50,
                this.centroCanvas.height + 100,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 150,
                this.centroCanvas.height + 100,
                'bck-card-1')
        ];
    }

    // dificil
    if(options.dificultad === 'dificil') {
        this.parejas = 8;
        this.barajaCartas();
        this.tablero = [
            // PRIMERA
            this.add.image(this.centroCanvas.width - 150,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width - 50,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 50,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 150,
                this.centroCanvas.height - 150,
                'bck-card-1'),
            // SEGUNDA
            this.add.image(this.centroCanvas.width - 150,
                this.centroCanvas.height - 25,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width - 50,
                this.centroCanvas.height - 25,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 50,
                this.centroCanvas.height - 25,
```

```
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 150,
                this.centroCanvas.height - 25,
                'bck-card-1'),
            // TERCERA
            this.add.image(this.centroCanvas.width - 150,
                this.centroCanvas.height + 100,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width - 50,
                this.centroCanvas.height + 100,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 50,
                this.centroCanvas.height + 100,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 150,
                this.centroCanvas.height + 100,
                'bck-card-1'),
            // CUARTA
            this.add.image(this.centroCanvas.width - 150,
                this.centroCanvas.height + 225,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width - 50,
                this.centroCanvas.height + 225,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 50,
                this.centroCanvas.height + 225,
                'bck-card-1'),
            this.add.image(this.centroCanvas.width + 150,
                this.centroCanvas.height + 225,
                'bck-card-1')
        ];
    }
}
```

Ilustración 7.16 Método reparteCartas (Play.js)

Como se puede observar, el método reparteCartas() de la ilustración 7.16 coloca en pantalla las imágenes de las cartas boca abajo pero hace falta una función que baraje las cartas y las coloque sobre la mesa, es decir, le dé un valor a cada carta colocada de forma aleatoria entre las disponibles. Para ello se han implementado los métodos initBaraja() barajaCartas() y sacaCartaRnd() cuyo código se indica en la ilustración 7.17.

```
/**
 * Inicializa valores de baraja a cero
 */
initBaraja() {
    for(var i = 0; i < this.parejas * 2; i++) {
```

```
            this.baraja[i] = 0;
        }
    }

     /**
     * Simula el proceso de barajar cartas.
     * En primer lugar se inicializa la baraja para despues ir sacando cartas
aleatorias,
     * por ultimo se van agregando las cartas sacadas al tablero de juego.
     */
    barajaCartas() {
        this.initBaraja();
        for(var i = 0; i < this.parejas * 2; i++) {
            this.sacaCartaRnd();
            options.tablero[this.obtenerPosTablero(i).x][this.obtenerPosTabler
o(i).y] = this.carta;
        }
    }

     /**
     * Saca carta de la baraja de forma aleatoria
     */
    sacaCartaRnd() {
        do {
            this.carta = Phaser.Math.Between(this.minCarta, this.parejas);
        } while(this.baraja[this.carta] >= 2);

        this.baraja[this.carta] ++;
    }
```

Ilustración 7.17 Métodos necesarios para barajar y colocar cartas sobre la mesa (Play.js)

Una vez que tenemos colocadas las cartas sobre el tablero y se ha asignado un valor para cada una de ellas es el momento de escuchar los eventos que el usuarios puede realizar sobre ellas, en este caso concreto el evento click. Para ello he creado un método que va asignando el escuchador de eventos carta por carta y realizando las operaciones pertinentes cuando una de ellas sea pulsada.

```
     /**
     * Escucha los eventos que se produzcan en el tablero de juego
     */
    escuchaEventosTablero() {
        this.tablero.map((carta, i) => {
            carta.setInteractive();
            carta.on(Phaser.Input.Events.POINTER_UP, () => {
                carta.disableInteractive();
                this.valor = options.tablero[this.obtenerPosTablero(i).x][this
.obtenerPosTablero(i).y];
                carta.frame = this.textures.getFrame('card-' + this.valor);
```

```
                this.cartasIntentos[this.intento] = carta;
                this.checkCartas(this.valor);
            });
        });
    }
```

Ilustración 7.18 Escucha los eventos realizados sobre las cartas (Play.js)

Como puede observarse en la ilustración 7.18 se recorren todas las cartas una a una y se le activa el atributo interactive para que puedan ser pulsadas por el jugador. Una vez una carta sea pulsada se desactivará este atributo (hasta que se pulse otra carta) ya que no se podrá voltear la misma carta dos veces, sería absurdo que una carta pudiera ser su propia pareja.

Se almacena la carta seleccionada en un vector de cartas para que en caso de no ser las mismas se pueda reactivar la propiedad interactive a las cartas pulsadas (puede observarse la ilustración 7.19 mostrada a continuación para ver cómo funciona esto). Por último, se comprobará si se ha encontrado una pareja con el método checkCartas().

```
    /**
     * Comprueba si las cartas seleccionadas son iguales o no
     * @param {*} valor indica el valor de la carta seleccionada
     */
    checkCartas(valor) {
        this.iguales = true;
        this.valorIntentos[this.intento] = valor;

        if(this.intento == options.maxIntentos - 1) {
            this.intento = 0;
            for(var i = options.maxIntentos - 1; i > 0; i--) {
                if(this.valorIntentos[0] != this.valorIntentos[i]) {
                    this.iguales = false;
                }
            }

            if(this.iguales === false) {
                this.disableInteractiveCartas();
                setTimeout(() => {
                    for(var i = 0; i < options.maxIntentos; i++) {
                        this.cartasIntentos[i].setInteractive();
                        this.cartasIntentos[i].frame = this.textures.getFrame(
'bck-card-1');
                    }
                    this.setInteractiveCartas();
                }, 1000);
```

```
                this.actualizaPuntos(2, false);
            } else {
                this.parejasEncontradas++;
                this.actualizaPuntos(10, true)
                this.checkVictoria();
            }
        } else {
            this.intento++;
        }
    }
```

Ilustración 7.19 Comprueba si se han seleccionado las mismas imagenes (Play.js)

Cuando se ha alcanzado el número máximo de cartas volteadas posibles para encontrar una pareja (número de intentos) lo primero que se debe comprobar es si se han encontrado las parejas o no y en caso de haberlas encontrado se aumenta el número de parejas encontradas en la partida, se actualizan los puntos del jugador y se comprueba si se han encontrado todas las parejas posibles o no.

Si por el contrario las cartas volteadas son distintas se vuelven a voltear y se penaliza al jugador restándole puntos pero antes de esto debe de producirse un tiempo de espera antes de volver a poner las cartas boca abajo para que el jugador pueda ver el valor de las cartas volteadas anteriormente. Por supuesto, es necesario desactivar la propiedad interactive de las cartas hasta que no se vuelvan a poner boca abajo.

Para este juego el método que comprueba si se ha ganado o no es bastante simple, tal y como se puede observar en la ilustración 7.20, ya que es suficiente con comprobar si el número de parejas encontradas durante la partida es igual al número de parejas que se han repartido. En caso de haber encontrado todas las parejas se mostrará una pequeña ventana con la puntuación del jugador, un campo donde poder introducir el Nick del jugador (es el nombre que se mostrará en la página de la clasificación de jugadores) y la opción de volver a jugar otra partida igual o volver al menú principal.

```
    /**
     * Comprueba si el jugador a encontrado todas las parejas de cartas
     */
    checkVictoria() {
        if(this.parejasEncontradas === this.parejas) {
            clearInterval(this.segundero);
```

```
            this.tablero_text.setText('PUNTUACION\nFINAL: ' + (this.puntuacion
+ this.segundos));
            this.add.tween({
                targets: this.tableroContainer,
                scaleX: 1,
                scaleY: 1,
                ease: 'Bounce',
                duration: 500
            });
        }
    }
```

Ilustración 7.20 Comprueba si se han encontrado todas las parejas (Play.js)

La puntuación final de la partida se calcula restándole los puntos conseguidos al tiempo que se ha tardado en encontrar todas las parejas. Por cada pareja que se encuentra se suman 10 puntos a la puntuación actual mientras que si se han seleccionado unas cartas que no coinciden se restan 2 puntos si la partida tiene una dificultad fácil, 4 puntos si la dificultad es de medio y 6 puntos si la dificultad es difícil, a la puntuación del jugador.

Esta variación agregada a la forma de sumar puntos conseguirá que sea más difícil alcanzar la mejor puntuación del juego.

7.3.4. Resultados

Tras completar las tareas propuestas para esta segunda entrega y subirlas al correspondiente repositorio y rama de github se puede ver, en la ilustración 7.21, la nueva ventana de opciones que aparece cuando se completa una partida y el marcador de puntos actuales y segundos transcurridos.

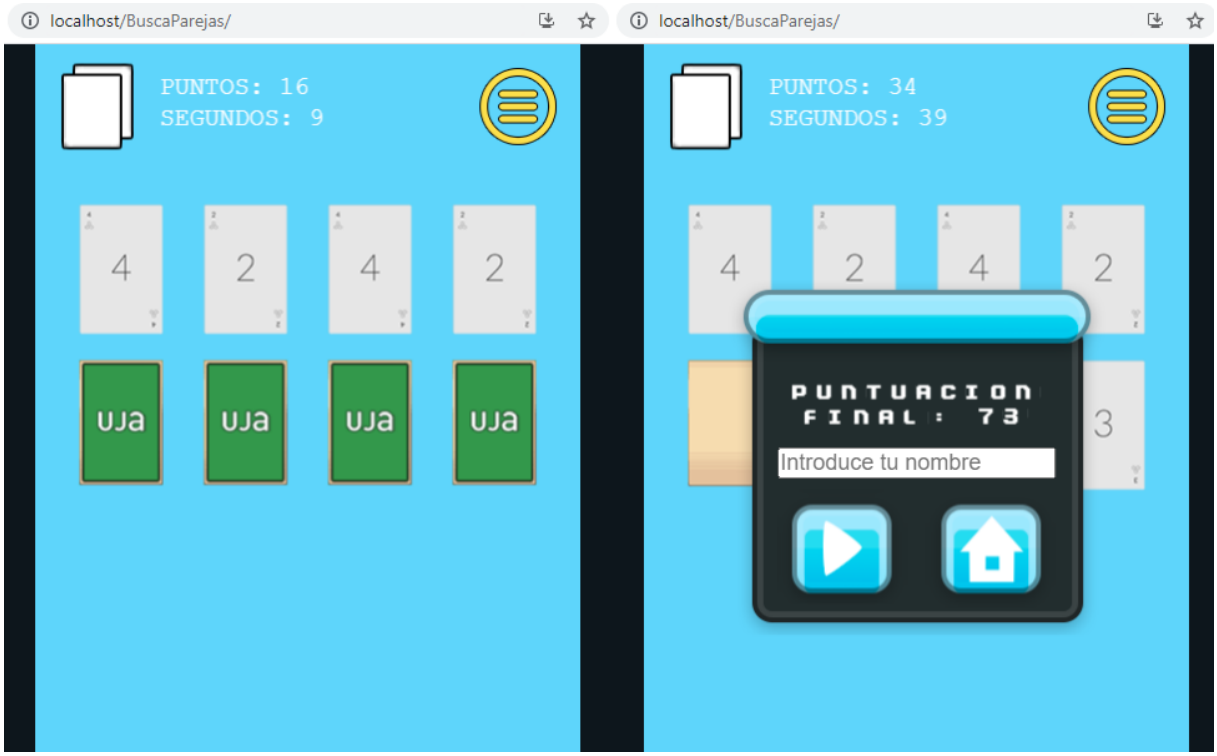


Ilustración 7.21 Marcadores y venta resultado partida implementados en el segundo entregable

En la ilustración 7.22 se muestra el estado del repositorio tras finalizar el segundo entregable[34].

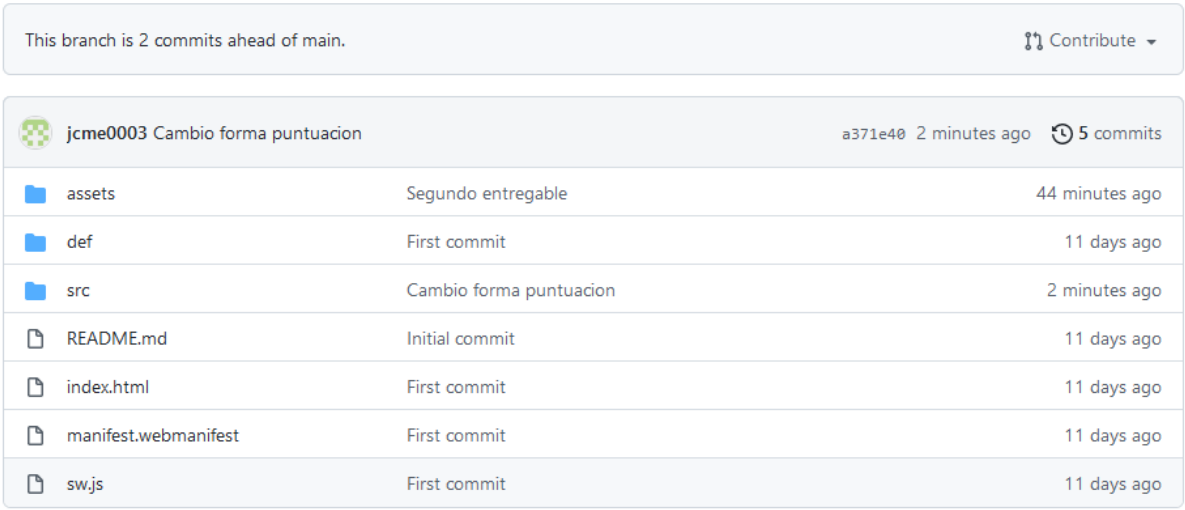


Ilustración 7.22 Estado del repositorio github a la finalización del segundo entregable

7.4. Entregable 3

Tras definir las partes de la aplicación correspondientes al segundo entregable es hora de realizar la parte correspondiente al proceso de configuración y

[34] https://github.com/jcme0003/BuscaParejas/tree/Segundo_entregable

visualización de la clasificación global del juego. Todo el proceso se indicará en el apartado “implementación del entregable”.

7.4.1. Tareas entregable

Para este tercer entregable se realizarán las tareas necesarias para que el usuario pueda jugar y obtener una puntuación final, tras finalizar la partida.

Tareas	Subtareas	Valor
Lógica del proceso de clasificación	Mostrar clasificación de jugadores	S
	Ordenar clasificación por puntos	S

Tabla 7.5 Tareas Proyecto Busca las parejas

7.4.2. Implementación entregable

Es el turno ahora de realizar la funcionalidad relacionada con la clasificación de los jugadores. Para ello voy a crear un archivo llamado “ranking.json” que almacenará las puntuaciones junto al nombre de cada uno de los jugadores. El archivo, como puede verse en la ilustración 7.23., contendrá unos datos de ejemplo para realizar en primer lugar la funcionalidad de mostrar el ranking actual al jugador.

```
{
    "ranking": [
        {"nombre": "Ana Mari", "puntos": 27},
        {"nombre": "Pepe", "puntos": 26},
        {"nombre": "Oscar", "puntos": 20},
        {"nombre": "Mario", "puntos": 19},
        {"nombre": "Anonimo", "puntos": 17},
        {"nombre": "Juan", "puntos": 15},
        {"nombre": "Felipe", "puntos": 12},
        {"nombre": "Hugo", "puntos": 8},
        {"nombre": "Victor", "puntos": 4},
        {"nombre": "Fernando", "puntos": 3}
    ]
}
```

Ilustración 7.23 Ranking de jugadores (ranking.json)

Para cargar los datos del fichero json debemos cargar el mismo en el método preload, tal y como se muestra en la ilustración 7.24. Una vez cargado el archivo json se accede a los datos agregando el código de la ilustración 7.25 en el método create de la escena actual.

```
preload() {
```

```
      this.load.text('ranking', './src/share/ranking.json');
}
```

Ilustración 7.24 Carga json con ranking de jugadores (ranking.js)

```
this.ranking = JSON.parse(this.cache.text.get('ranking'));
```

Ilustración 7.25 Carga datos json ranking de jugadores (ranking.js)

Por último, se ha creado un método llamado pintaRanking() (ver implementación en la ilustración 7.26.) que recorre las mejores puntuaciones almacenadas en el fichero json y crea el contenido html necesario para mostrar por pantalla al usuario los datos con sus estilos correspondientes.

```
pintaRanking() {
        this.container = document.createElement('div');

        this.ranking.ranking.forEach(element => {
            this.content = document.createElement('p');
            this.text = document.createTextNode(element.nombre + ': ' + elemen
t.puntos + ' pts');
            this.content.appendChild(this.text);
            this.container.appendChild(this.content);
        });

        this.styles = 'width: 400px; max-width: 100%; max-
height: 550px; margin: 80px 20px 0 0; color: #fff; font-size: 36px; font-
weight: bold; text-shadow: 1px 0 0 #000, -1px 0 0 #000, 0 1px 0 #000, 0 -
1px 0 #000, 1px 1px #000, -1px -1px 0 #000, 1px -1px 0 #000, -1px 1px 0 #000;'
        this.content = this.add.dom(this.centroCanvas.width, this.centroCanvas
.height, this.container, this.styles);
}
```

Ilustración 7.26 Método pintaRanking (Ranking.js)

La funcionalidad de actualizar el fichero json necesita de un archivo php que será el que reciba el nombre y puntuación del jugador y la agregue al fichero json en caso de estar entre las 10 mejores.

Cuando un jugador encuentra todas las parejas y completa la partida puede decidir entre repetir la partida o volver al menú principal, en cualquiera de los dos casos se llamará al método checkRanking(), cuya implementación puede verse en la ilustración 7.27 y realizará la petición POST a ranking.php.

```
/**
     * Hace una petición POST a ranking.php y comprueba si la puntuación del j
ugador actual
     * es una de las 10 mejores o no. En caos de estar entre las 10 mejores la
 agrega
     * al fichero ranking.json.
```

```
     */
    checkRanking() {
        this.nombre = this.inputText.getChildByName('nombre').value;

        if(this.nombre === '') {
            this.nombre = 'Anónimo';
        }

        var url = 'ranking.php';
        var data = new FormData();
        data.append('nombre', this.nombre);
        data.append('puntos', this.puntuacion);

        fetch(url, {
            method: 'POST',
            body: data
        }).then(res => res.json());
    }
```

Ilustración 7.27 Método checkRanking (Play.js)

La funcionalidad de ranking.php se muestra en la ilustración 7.28 y consiste en lo siguiente, recibe la solicitud post realizada por el método checkRanking(), pasándole el nombre (en caso de ser introducido por el usuario, en caso contrario será anónimo) y los puntos conseguidos por el usuario en la partida. Una vez ha recibido los datos los añade al array y este es ordenado por la puntuación de cada jugador en orden descendente, es decir, de mayor a menor. Por último, se elimina el jugador en última posición dejan únicamente las 10 mejores puntuaciones conseguidas por los jugadores.

```
<?php
$dir = __DIR__ . '/src/share/ranking.json';
$datos_json = json_decode(file_get_contents($dir));

$user = array(
    "nombre" => $_POST["nombre"],
    "puntos" => intval($_POST["puntos"])
);
array_push($datos_json -> ranking, $user);

file_put_contents($dir, json_encode($datos_json));
$datos_json = json_decode(file_get_contents($dir));
$newJson = [];

usort($datos_json -> ranking, function($a, $b) {
    return $a->puntos > $b->puntos ? -1 : 1;
});
```

```php
array_pop($datos_json -> ranking);
$newJson = json_encode($datos_json);
echo $newJson;

file_put_contents($dir, $newJson);
?>
```

Ilustración 7.28 Fichero ranking.php

7.4.3. Resultados

Tras completar las tareas propuestas para este tercer entregable y subirlas al correspondiente repositorio y rama de github, cuyo código[35] puede verse en la ilustración 7.29., ya se ha desarrollado la funcionalidad que muestra y actualiza el ranking de las 10 mejores puntuaciones, como el de la ilustración 7.30.

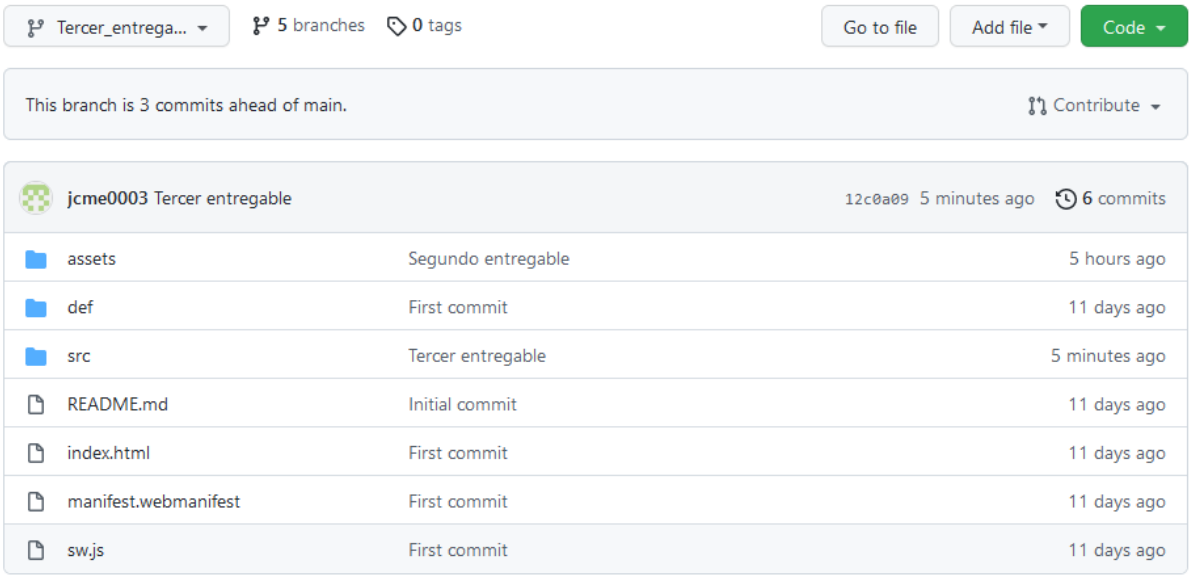


Ilustración 7.29 Estado del repositorio github a la finalización del tercer entregable

[35] https://github.com/jcme0003/BuscaParejas/tree/Tercer_entregable

Ilustración 7.30 Ranking implementado en el tercer entregable

7.5. Entregable 4

Una vez finalizada la entrega número 3 se van a agregar una serie de sonidos al videojuego que le aportarán un extra de realismo. Todo el código y modificaciones necesarias se indicarán en el apartado “implementación del entregable”.

7.5.1. Tareas entregable

Para este cuarto entregable se realizarán las tareas necesarias para que se escuchen una serie de sonidos cuando el jugador realice unas determinadas acciones.

Tareas	Subtareas	Valor
Agregar efectos de sonido	Añadir sonidos al juego	C

Tabla 7.6 Tareas entregable 4 Proyecto Busca las parejas

7.5.2. Implementación entregable

Llega el momento de agregar sonidos a ciertas acciones del juego. Una de las más importantes es girar la carta seleccionada sobre el tablero de juego. Para ello, hay que acceder a la escena Play.js y en el método escuchaEventosTablero() agregar el código que se muestra subrayado en color rojo en la ilustración 7.31.

```
escuchaEventosTablero() {
        this.tablero.map((carta, i) => {
            carta.setInteractive();
            carta.on(Phaser.Input.Events.POINTER_UP, () => {
                carta.disableInteractive();
                this.audioFlipCard.play();
                this.valor = options.tablero[this.obtenerPosTablero(i).x][this
.obtenerPosTablero(i).y];
                carta.frame = this.textures.getFrame('card-' + this.valor);
                this.cartasIntentos[this.intento] = carta;
                this.checkCartas(this.valor);
            });
        });
}
```

Ilustración 7.31 Lanzar audio flipcard (Play.js)

Antes de realizar esta modificación se debe inicializar y cargar los audios en la escena Bootloader y en el método init() de la escena Play.js. En la ilustración 7.32 podemos ver cómo se carga el audio en el Bootloader y cómo se inicializa el audio "flipcard" en la ilustración 7.33.

```
preload() {
        this.load.audio('flipcard', 'flipcard.wav');
}
```

Ilustración 7.32 Precarga de audio flipcard (Bootloader.js)

```
init() {
        this.audioFlipCard = this.sound.add('flipcard');
}
```

Ilustración 7.33 Inicializa audio flipcard (Play.js)

Además de lanzar el audio "flipcard" cuando las cartas están boca abajo y se selecciona una de ellas debe lanzarse también cuando las cartas seleccionadas no son pareja y se vuelven a voltear. El código que hace que este sonido suene en el caso indicado es el código mostrado en la ilustración 7.34 mostrada a continuación.

```
checkCartas(valor) {
        this.iguales = true;
        this.valorIntentos[this.intento] = valor;

        if(this.intento == options.maxIntentos - 1) {
            this.intento = 0;
            for(var i = options.maxIntentos - 1; i > 0; i--) {
                if(this.valorIntentos[0] != this.valorIntentos[i]) {
                    this.iguales = false;
                }
            }
```

```
            if(this.iguales === false) {
                this.disableInteractiveCartas();
                setTimeout(() => {
                    this.audioFlipCard.play();
                    for(var i = 0; i < options.maxIntentos; i++) {
                        this.cartasIntentos[i].setInteractive();
                        this.cartasIntentos[i].frame = this.textures.getFrame(
'bck-card-1');
                    }
                    this.setInteractiveCartas();
                }, 1000);
                this.actualizaPuntos(2, false);
            } else {
                this.parejasEncontradas++;
                this.actualizaPuntos(10, true)
                this.checkVictoria();
            }
        } else {
            this.intento++;
        }
}
```

Ilustración 7.34 Lanzar audio flipcard (Play.js)

Otro sonido que se ha agregado a la aplicación es el sonido que se emitirá cuando un jugador gana la partida. Para este caso el código necesario se agrega en el método checkVictoria() de la forma que se indica en la ilustración 7.35., subrayado en color rojo.

```
checkVictoria() {
        if(this.parejasEncontradas === this.parejas)
            clearInterval(this.segundero);
            this.audioWin.play();
            this.tablero_text.setText('PUNTUACION\nFINAL: ' + (this.puntuacion
 - (this.segundos/2)));
            this.add.tween({
                targets: this.tableroContainer,
                scaleX: 1,
                scaleY: 1,
                ease: 'Bounce',
                duration: 500
            });
        }
    }
```

Ilustración 7.35 Lanzar audio de victoria (Play.js)

Por último, se ha agregado un sonido cuando el usuario gana la partida y decide jugar otra en el mismo nivel de dificultad. Este audio se precarga y se inicializa al igual que los anteriores en la escena Bootloader y en el método init() de

la escena Play.js respectivamente. Para que suene cuando el jugador decide repetir partida se debe actualizar el método con la línea subrayada en color rojo como se muestra en la siguiente ilustración 7.36.

```
this.tablero_button_replay.on(Phaser.Input.Events.POINTER_UP, () => {
      this.audioDraw.play();
      this.add.tween({
            targets: this.tableroContainer,
            scaleX: 0,
            scaleY: 0,
            duration: 1000,
            ease: 'Bounce',
            onComplete: () => {
                  this.checkRanking();
                  this.scene.start('Reload');
            }
      });
});
```

Ilustración 7.36 Lanzar audio de repetir partida (Play.js)

7.5.3. Resultados

En este entregable número cuatro, la aplicación no ha variado nada visualmente ya que lo que se han agregado han sido los sonidos en distintas acciones. En la ilustración 7.37 mostrada a continuación puede verse el estado de la aplicación una vez que se ha subido el código[36] al repositorio y rama de github correspondiente, con la tarea para la entrega completada.

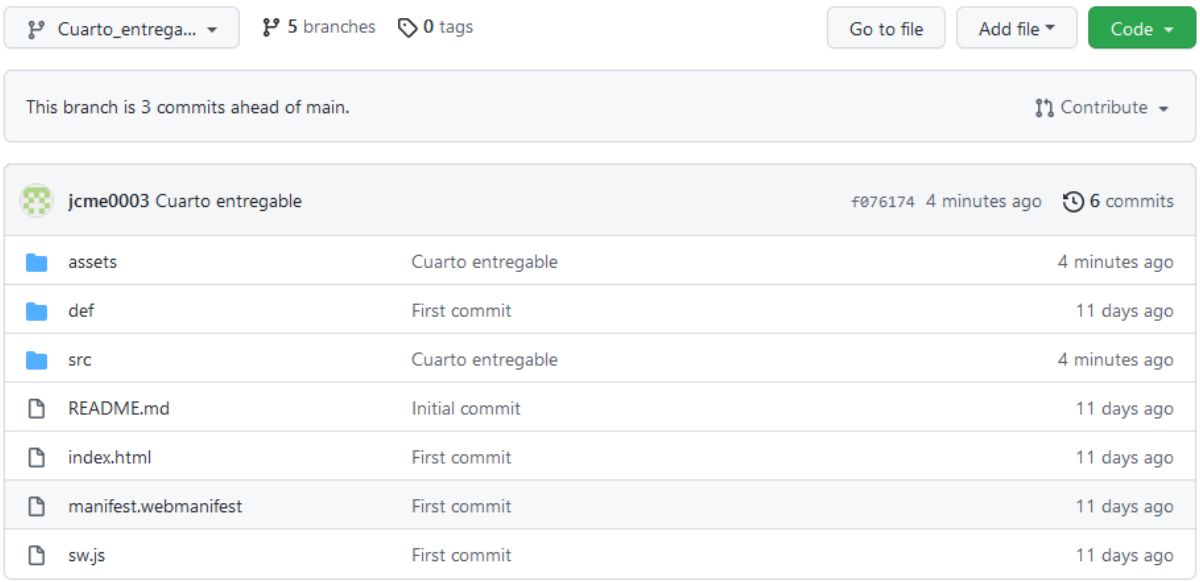


Ilustración 7.37 Estado del repositorio github a la finalización del cuarto entregable

[36] https://github.com/jcme0003/BuscaParejas/tree/Cuarto_entregable

7.6. Entregable 5

Para este último entregable se agregará un apartado de opciones donde el usuario podrá configurar ciertas características del juego, como pueden ser el volumen y el tipo de cartas con las que quiere jugar. En el apartado “implementación del entregable” se agregarán ilustraciones con el código y modificaciones necesarias para realizar esta tarea.

7.6.1. Tareas entregable

Las tareas que se van a desarrollar en este quinto entregable se muestran a continuación en la tabla 7.7.

Tareas	Subtareas	Valor
Apartado de opciones	Crear ventana de opciones	C
	Codificar funcionalidad de botones y opciones	C
	Crear ventana información sobre el juego	C

Tabla 7.7 Tareas entregable 4 Proyecto Busca las parejas

7.6.2. Storyboards entregable

En este quinto y último entregable, se agregarán dos ventanas emergentes, una con información sobre las reglas del juego y otra con las opciones configurables del mismo. En la ilustración 7.38, pueden verse los storyboards diseñados para estas ventanas emergentes.

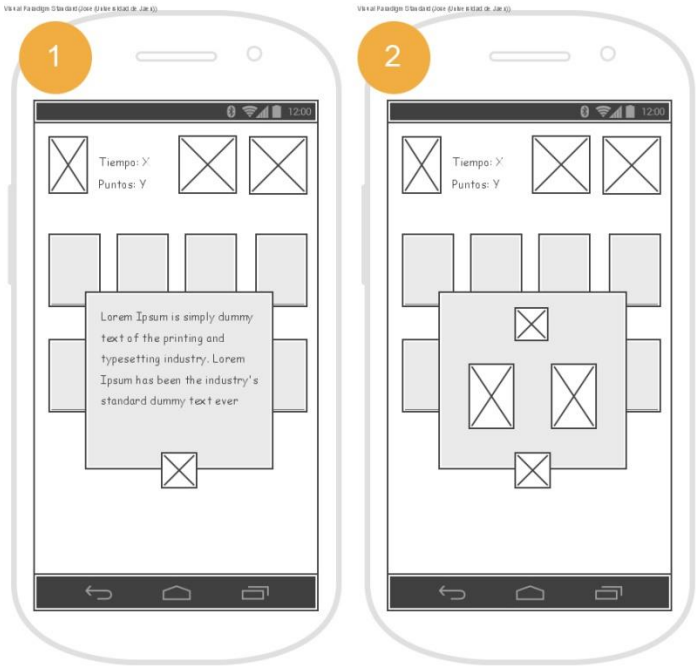


Ilustración 7.38 Storyboards ventanas emergentes

7.6.3. Implementación entregable

El menú de opciones solo estará accesible desde la vista o escena del menú principal y para ello será necesario agregar un icono que en este caso se ha decidido que sea un icono con forma de “engranaje” como el que se puede ver en la ilustración 7.39.

Ilustración 7.39 Icono de opciones del menú principal

La imagen se precargará en el Bootloader y se debe inicializar en la escena del menú principal (Menu.js) como se ha indicado para casos similares en los anteriores entregables. A continuación se debe decidir la imagen que se va a lanzar como ventana de opciones que en mi caso me he decidido por la imagen indicada en la ilustración 7.40.

Ilustración 7.40 Ventana de opciones

Ahora ha llegado el momento de lanzar la ventana de opciones una vez el usuario pulse sobre el icono correspondiente y para ello debe escucharse el evento on click sobre el icono, tal y como se muestra en la ilustración 7.41.

```
this.opciones.on(Phaser.Input.Events.POINTER_UP, () => {
      this.add.tween({
            targets: this.opcionesContainer,
            scaleX: 1,
            scaleY: 1,
            ease: 'Bounce',
            duration: 500
      });
});
```

Ilustración 7.41 Evento para lanzar ventana de opciones (Menu.js)

En esta ventana de opciones se ofrecerá al usuario la posibilidad de personalizar algunos aspectos relativos al juego como son:

- Activar/desactivar sonidos.
- Elegir la baraja de cartas con la que se quiere jugar.

Para ello se cargarán las correspondientes imágenes del audio y las barajas, además de un botón para aceptar las opciones seleccionadas, tal y como se muestra en la ilustración 7.42. También deben alinearse dentro del contenedor para

que se muestren todas las opciones de forma correcta, puede verse el código en la ilustración 7.43.

```
this.opciones_text = this.add.bitmapText(0, 0, 'futilePro', 'OPCIONES:', 20, 1
);
this.opciones_sound = this.add.image(0, 0, 'sound_' + options.audio).setIntera
ctive();
this.opciones_card_1 = this.add.image(0, 0, 'card-1-0').setInteractive();
this.opciones_card_2 = this.add.image(0, 0, 'card-2-1').setInteractive();
this.opciones_ok_button = this.add.image(0, 0, 'ok_button').setInteractive();
```

Ilustración 7.42 Cargar imágenes de audio, barajas de cartas y botón ok (Menu.js)

```
Phaser.Display.Align.In.Center(this.opciones_text, this.opciones_tablero, 0, -
130);
Phaser.Display.Align.To.BottomCenter(this.opciones_sound, this.opciones_text,
0, 10);
Phaser.Display.Align.To.BottomLeft(this.opciones_card_1, this.opciones_sound,
20, 0);
Phaser.Display.Align.To.BottomRight(this.opciones_card_2, this.opciones_sound,
 20, 0);
Phaser.Display.Align.To.BottomCenter(this.opciones_ok_button, this.opciones_so
und, 0, 100);
```

Ilustración 7.43 Código para alinear elementos en el menú de opciones (Menu.js)

Dicho contenedor contendrá todos estos elementos y facilitará en gran medida el código a desarrollar. Un ejemplo de esta facilidad que proporciona el contenedor lo podemos ver en la ilustración 7.41 anterior, que únicamente lanza el contenedor cuando se pulsa en icono de opciones. En la ilustración 7.44 se agregan todos los elementos al contenedor y éste se oculta y se centra en el dispositivo del usuario.

```
this.opcionesContainer = this.add.container(this.centroCanvas.width,
this.centroCanvas.height);

this.opcionesContainer.add([
      this.opciones_tablero,
      this.opciones_text,
      this.opciones_sound,
      this.opciones_card_1,
      this.opciones_card_2,
      this.opciones_ok_button
]);
this.opcionesContainer.setDepth(2);
this.opcionesContainer.setScale(0);
```

Ilustración 7.44 Código para agregar elementos al contenedor de opciones y centrarlo (Menu.js)

Por último, faltaría el código encargado de activar/desactivar el audio y de mostrar un tipo de baraja según las opciones del usuario. Para el audio lo que se ha hecho es, además de captar el evento click, modificar el valor de la opción audio por

true si estaba a false o a false si estaba a true y por cambiar la imagen que se muestra en la ventana de opciones según este valor y se ha hecho de la forma que se muestra en la ilustración 7.45.

```
this.opciones_sound.on(Phaser.Input.Events.POINTER_UP, () => {
      options.audio = !options.audio;
      this.opciones_sound.frame = this.textures.getFrame('sound_' + options.au
dio);
});
```

Ilustración 7.45 Código para activar/desactivar audio del juego (Menu.js)

Para modificar la baraja de cartas con la que se va a jugar se ha asignado también un evento para cada tipo de baraja mostrada en el menú de opciones y cuando se pulse un tipo de baraja se modificará el valor de la baraja seleccionada en las opciones generales del juego o fichero options.js, el código necesario se puede consultar en la ilustración 7.46., cambiando también las imágenes de las barajas.

```
this.opciones_card_1.on(Phaser.Input.Events.POINTER_UP, () => {
      options.baraja = 1;
      this.opciones_card_1.frame = this.textures.getFrame('card-1-0');
      this.opciones_card_2.frame = this.textures.getFrame('card-2-1');
});
this.opciones_card_2.on(Phaser.Input.Events.POINTER_UP, () => {
      options.baraja = 2;
      this.opciones_card_1.frame = this.textures.getFrame('card-1-1');
      this.opciones_card_2.frame = this.textures.getFrame('card-2-0');
});
```

Ilustración 7.46 Código para seleccionar tipo de baraja (Menu.js)

Cuando una baraja es seleccionada por el jugador se mostrará como en la ilustración 7.47 por el contrario, las barajas no seleccionadas por el usuario se mostrarán de la forma que se indica en la ilustración 7.48.

Ilustración 7.47 Baraja seleccionada

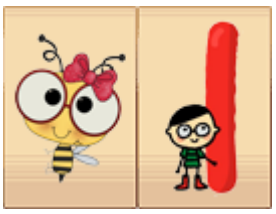


Ilustración 7.48 Baraja no seleccionada

Por último, cuando el botón OK sea pulsado por el usuario los datos serán guardados y se cerrará la ventana de opciones, cuyo código se muestra en la ilustración 7.49.

```
this.opciones_ok_button.on(Phaser.Input.Events.POINTER_UP, () => {
      this.add.tween({
```

```
            targets: this.opcionesContainer,
            scaleX: 0,
            scaleY: 0,
            ease: 'Bounce',
            duration: 500
      });
});
```

Ilustración 7.49 Código para cerrar ventana cuando se pulsa el botón ok (Menu.js)

Ya se ha completado el apartado de opciones del juego pero falta una última tarea por realizar, que es la de mostrar una ventana con información sobre el juego, cómo jugar y cómo se puntúa. En este caso, el icono será el de la ilustración 7.50 que se mostrará en la parte superior de la pantalla, y la ventana emergente será la misma que se ha utilizado para el menú de opciones, es decir, la mostrada anteriormente en la ilustración 7.40.

Ilustración 7.50 Icono info del menú principal

El texto a mostrar se agregará junto a la imagen de la venta emergente, al igual que se ha hecho anteriormente con el menú opciones, al contenedor llamado infoContainer y dicho contenedor se mostrará cuando el icono de info sea clicado y se ocultará cuando se pulse el icono de la ilustración 7.51 que representa la opción de cerrar ventana.

Ilustración 7.51 Icono cerrar ventana info del menú principal

Las funcionalidades de abrir y cerrar la ventana de información sobre cómo funciona el juego se controlan tal y como se muestran en la ilustración 7.52.

```
this.info.on(Phaser.Input.Events.POINTER_UP, () => {
      this.add.tween({
            targets: this.infoContainer,
            scaleX: 1,
            scaleY: 1,
```

```
            ease: 'Bounce',
            duration: 500
      });
});
this.info_close.on(Phaser.Input.Events.POINTER_UP, () => {
      this.add.tween({
            targets: this.infoContainer,
            scaleX: 0,
            scaleY: 0,
            ease: 'Bounce',
            duration: 500
      });
});
```

Ilustración 7.52 Código para abrir y cerrar ventana información sobre el juego (Menu.js)

7.6.4. Resultados

Una vez finalizado el entregable número cinco se ha subido el [código](#)[37] a la rama de github “Quinto_entregable”, en la ilustración 7.53 puede verse el estado final del repositorio.

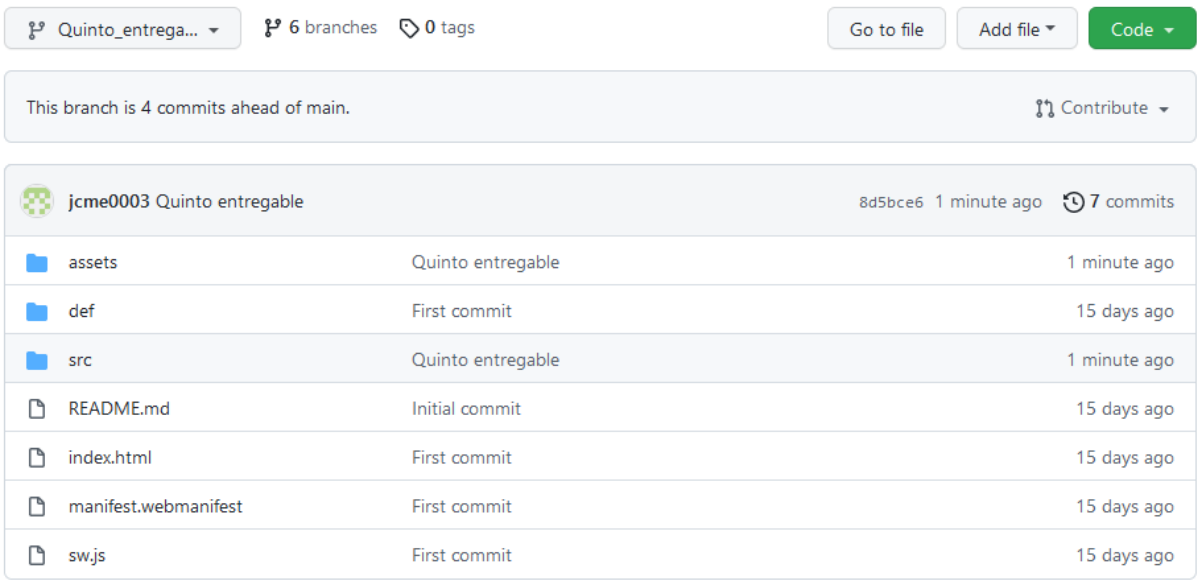


Ilustración 7.53 Estado del repositorio github a la finalización del quinto entregable

El resultado de las nuevas ventanas implementadas puede verse en la ilustración 7.54 donde se muestra la ventana correspondiente con la ventana de información sobre el juego junto a la ventana de opciones.

[37] https://github.com/jcme0003/BuscaParejas/tree/Quinto_entregable

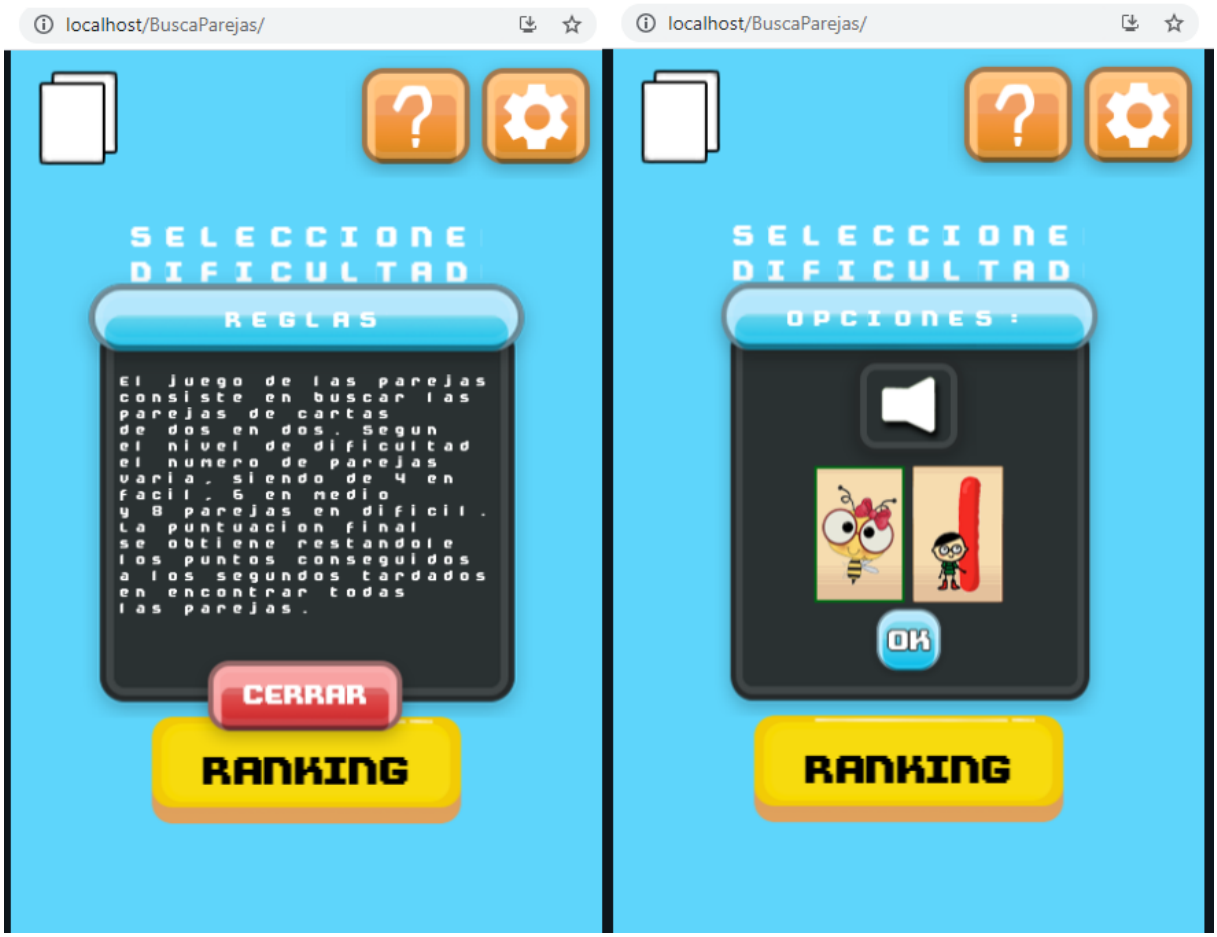


Ilustración 7.54 Ventanas de información sobre el juego (reglas) y ventana opciones

7.6.5. Pruebas usabilidad del sistema

Con el objetivo de obtener los resultados más fiables posibles y evaluar así el producto final, se va a realizar un cuestionario incluido en el apéndice, Cuestionario, por un grupo formado por 10 usuarios de diferentes edades, mostradas en la tabla 7.8, y con diferentes conocimientos sobre las nuevas tecnologías.

Usuario	Edad	Media
1	22	4,90
2	54	4,60
3	29	4,80
4	36	4,60
5	18	4,60
6	65	4,30
7	48	4,60
8	10	5,00
9	80	4,40

10	72	4,40

Tabla 7.8 Resultado por edades de los usuarios pruebas de usabilidad

Los resultados obtenidos, mostrados en la tabla 7.9 se tendrán en cuenta para futuras mejoras o actualizaciones del sistema ya que se facilitará un apartado donde el usuario podrá agregar cualquier tipo de comentario que considere oportuno.

Pregunta	22	54	29	36	18	65	48	10	80	72
1.- Me gustaría utilizar esta aplicación frecuentemente	4	2	3	1	1	3	2	5	5	5
2.- No he encontrado bugs o errores	5	5	5	5	5	5	5	5	5	5
3.- Visualmente, la aplicación es elegante (colores, textos, legibilidad…)	5	5	5	5	5	4	5	5	3	3
4.- No he necesitado conocimientos avanzados en nuevas tecnologías para usar la aplicación	5	4	5	5	5	3	4	5	3	3
5.- Las opciones de la aplicación estaban bien integradas	5	5	5	5	5	5	5	5	5	5
6.- En mi opinión, la mayoría de personas podrían aprender a utilizar la aplicación rápidamente	5	5	5	5	5	4	5	5	4	4
7.- He podido instalar la aplicación fácilmente	5	5	5	5	5	5	5	5	5	5
8.- La aplicación cumple con su cometido inicial	5	5	5	5	5	5	5	5	5	5
9.- Los botones son lo suficientemente grande como para poder pulsarlos correctamente	5	5	5	5	5	4	5	5	4	4
10.- En términos generales, la aplicación fue fácil de usar	5	5	5	5	5	5	5	5	5	5

Tabla 7.9 Respuestas de los usuarios a las pruebas de usabilidad

Una vez se ha realizado el análisis de los resultados obtenidos de las encuestas y analizando la tabla 7.10 con la media y desviación típica, se ha llegado a la conclusión de que la aplicación no presenta errores o bugs, las opciones están bien integradas en la aplicación, es fácil de instalar, la aplicación cumple con su cometido y es fácil de usar. Sin embargo, algunos tipos de usuarios no tienen intención de utilizarla de forma frecuente y otros necesitaron ayuda de alguien con más conocimientos.

Esto quiere decir que los usuarios con una edad más avanzada necesitan algo de ayuda ya que no están tan acostumbrados a las nuevas tecnologías como los usuarios de menor edad.

Pregunta	Media ± desv. típica
1.- Me gustaría utilizar esta aplicación frecuentemente	3,10 ± 1,51
2.- No he encontrado bugs o errores	5,00 ± 0,00
3.- Visualmente, la aplicación es elegante (colores, textos, legibilidad…)	4,50 ± 0,81
4.- No he necesitado conocimientos avanzados en nuevas tecnologías para usar la aplicación	4,20 ± 0,87
5.- Las opciones de la aplicación estaban bien integradas	5,00 ± 0,00
6.- En mi opinión, la mayoría de personas podrían aprender a utilizar la aplicación rápidamente	4,70 ± 0,46
7.- He podido instalar la aplicación fácilmente	5,00 ± 0,00
8.- La aplicación cumple con su cometido inicial	5,00 ± 0,00
9.- Los botones son lo suficientemente grande como para poder pulsarlos correctamente	4,70 ± 0,46
10.- En términos generales, la aplicación fue fácil de usar	5,00 ± 0,00

Tabla 7.10 Desviación típica respuestas a las pruebas de usabilidad

8. Conclusión

El Trabajo Fin de Grado muestra con qué facilidad puede transformarse una aplicación web en una Progressive Web App y aparentar, ante los ojos de un usuario normal, ser una aplicación nativa sin necesidad de instalarse desde la Google Play en caso de un dispositivo Android o desde la App Store en dispositivos Apple. Las PWA pueden instalarse como una aplicación normal desde el navegador web y

pueden funcionar sin conexión a internet, lo que las hace especialmente interesantes.

Sin embargo, y pese a todas las ventajas que ofrecen las PWA, también tienen limitaciones y las más importantes son que no todos los navegadores soportan la opción de instalar (a2hs) ni la opción de enviar notificaciones push. Esto no significa que no puedan utilizarse en todos los dispositivos pero sí que el usuario no podrá disfrutar al máximo de la experiencia que ofrecen las PWA, aun así existen soluciones como por ejemplo enviar un mensaje SMS a los usuarios que no les podamos enviar notificaciones push.

9. Trabajos futuros

A la finalización de la aplicación se ha utilizado el software Lighthouse y se han obtenido los datos mostrados en la ilustración 9.1. Si nos fijamos en la puntuación obtenida en el apartado de rendimiento (performance) vemos que se ha obtenido muy mala nota, solo 46 puntos.

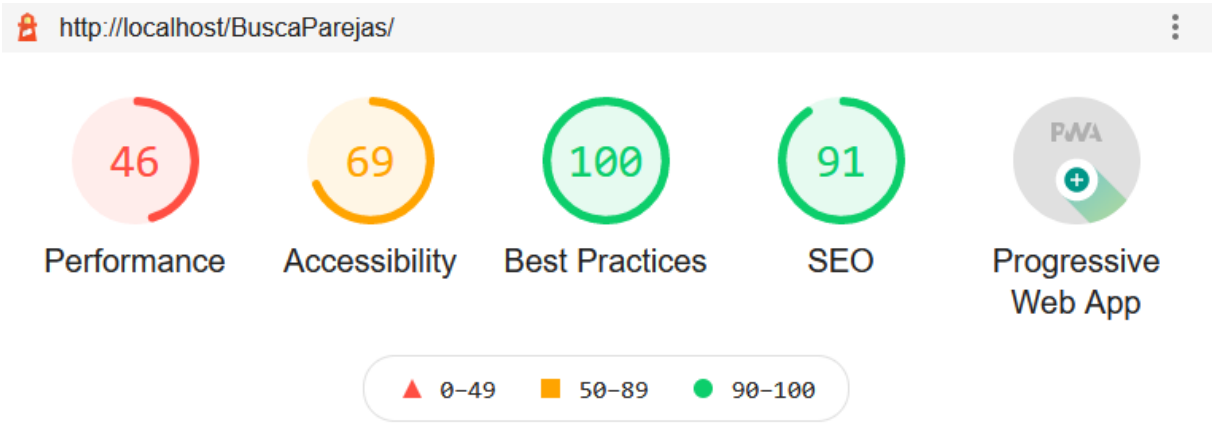


Ilustración 9.1 Resultados Lighthouse

En siguientes ampliaciones o mejoras de la aplicación podría optarse por mejorar este apartado e intentar mejorar el rendimiento de la misma, mejorando el tiempo de espera, tiempo de carga o eliminando la carga innecesaria de ciertos elementos. Esto podría ayudar a mejorar la experiencia de usuario del jugador y a atraer a un mayor número de usuarios.

Por supuesto, no me puedo olvidar de la parte de accesibilidad que también ha salido un poco baja con una puntuación de 69 y teniendo en cuenta que los principales usuarios de esta aplicación son personas mayores o personas con cierta discapacidad, es un apartado de la aplicación que habría que mejorar en futuras actualizaciones.

Para facilitarnos el trabajo Lighthouse nos indica en qué puntos está fallando la aplicación y qué acciones podríamos tomar para mejorar el resultado y la experiencia de usuario. Por ejemplo, en mi caso, algunas de las recomendaciones que me aporta en el apartado de performance son las mostradas en la ilustración 9.2.

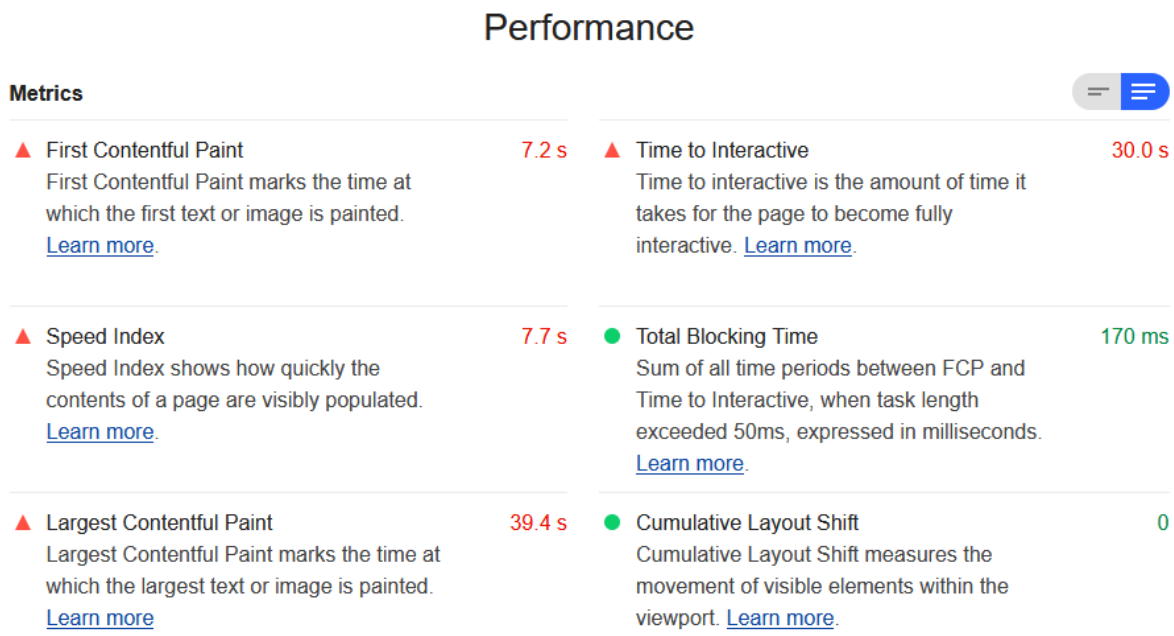


Ilustración 9.2 Consejos apartado performance Lighthouse

10. Apéndice

10.1. Instalación y configuración del sistema

Para proceder a instalar la aplicación hace falta un dispositivo con un navegador web y conexión a internet para acceder a la web donde se encuentre alojada la aplicación y pulsar en instalar en caso de que el navegador nos muestre la ventana emergente o buscando la opción de instalar en el menú del navegador.

Para probarlo he lanzado el software ngrok [38] junto a la herramienta de desarrollo xampp[39] donde se encuentra instalada la aplicación. Al lanzar ngrok nos ofrece una URL desde la que podemos acceder a la aplicación desde cualquier dispositivo con conexión a internet.

En mi caso he probado con un teléfono móvil Android y un iPhone y en ambos casos he podido utilizar la aplicación sin problemas desde el navegador web. Para instalar la aplicación hay que diferenciar entre los dos sistemas operativos:

- **Android**

En el caso de Android, el navegador que mejor soporta la tecnología PWA es Google Chrome, por ello se recomienda acceder a la web desde este navegador y bien pulsar en el botón instalar que se muestra en la parte superior de la ventana o dentro de las opciones de menú buscar la opción “instalar aplicación”. Puede ver un ejemplo visual en la ilustración 10.1.

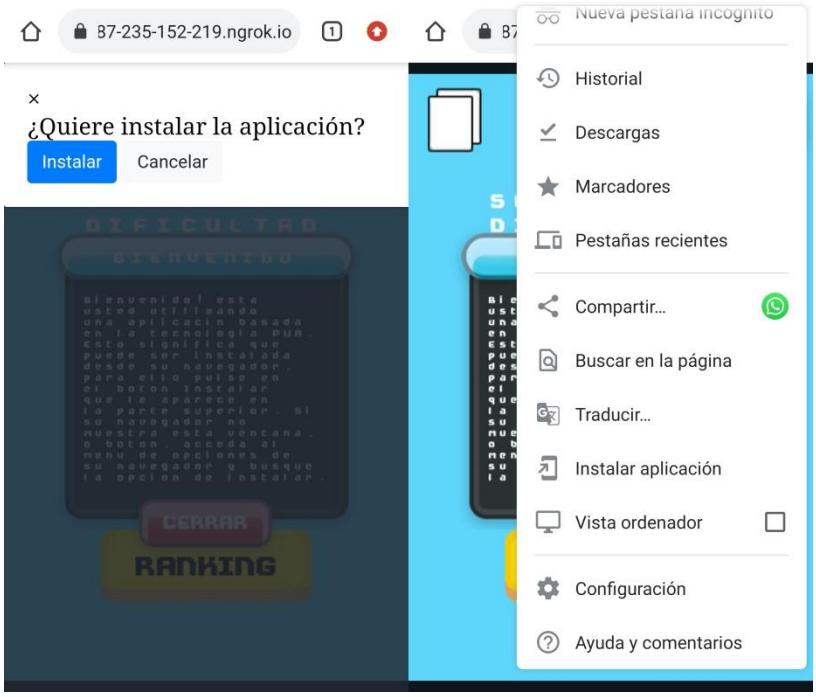


Ilustración 10.1 Instalar PWA desde el navegador Google Chrome

Una vez instalada la aplicación funciona sin conexión a internet tal y como se muestra en la ilustración 10.2.

[38] https://ngrok.com/
[39] https://www.apachefriends.org/es/index.html

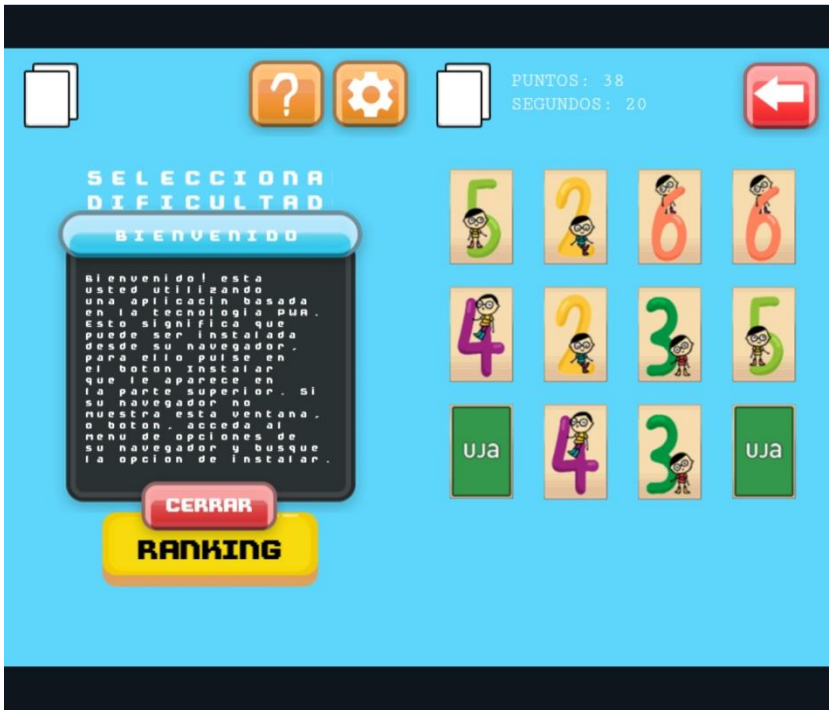


Ilustración 10.2 PWA instalada sin conexión a internet

- **IPhone**

En los dispositivos de Apple, como puede ser un IPhone, se necesita de conexión a internet para que la PWA funcione correctamente al 100%. Es por eso que la instalación y las pruebas de funcionamiento se llevarán a cabo con conexión a internet. El proceso de instalación se muestra a continuación en la ilustración 10.3.

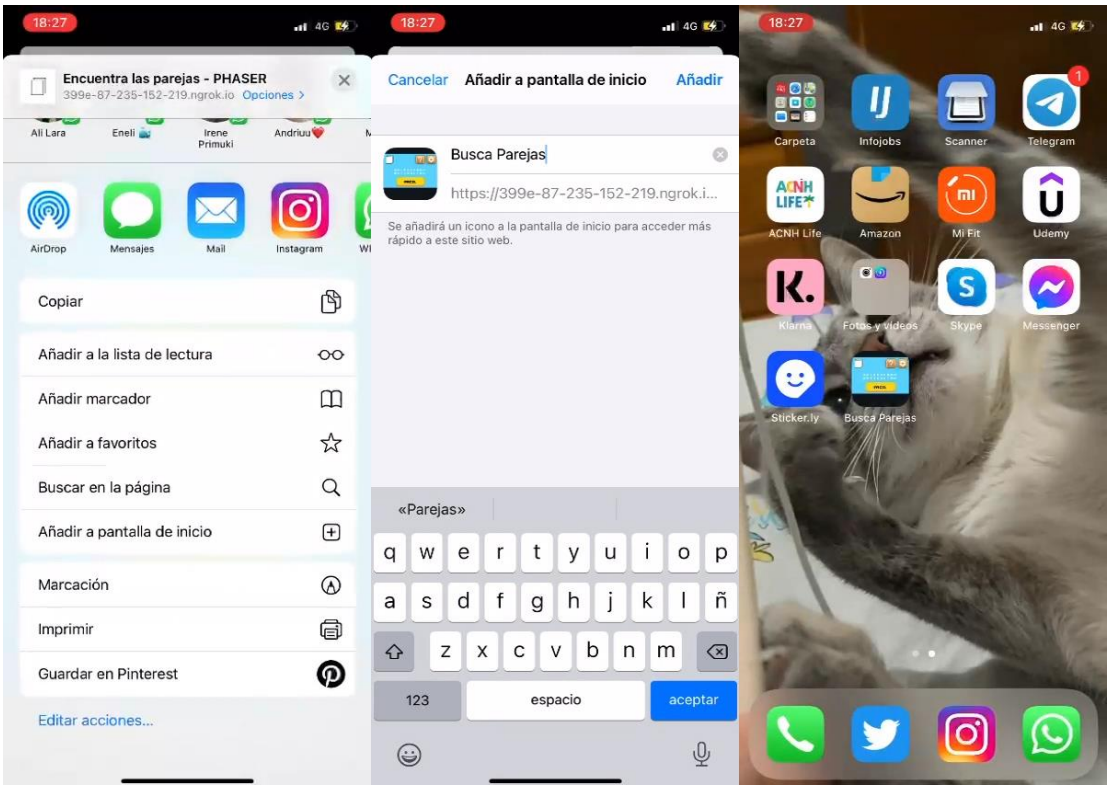


Ilustración 10.3 Instalar PWA en IPhone

Una vez instalada solo hay que lanzarla y jugar, en la ilustración 10.4 puede observarse como funciona correctamente en un IPhone.

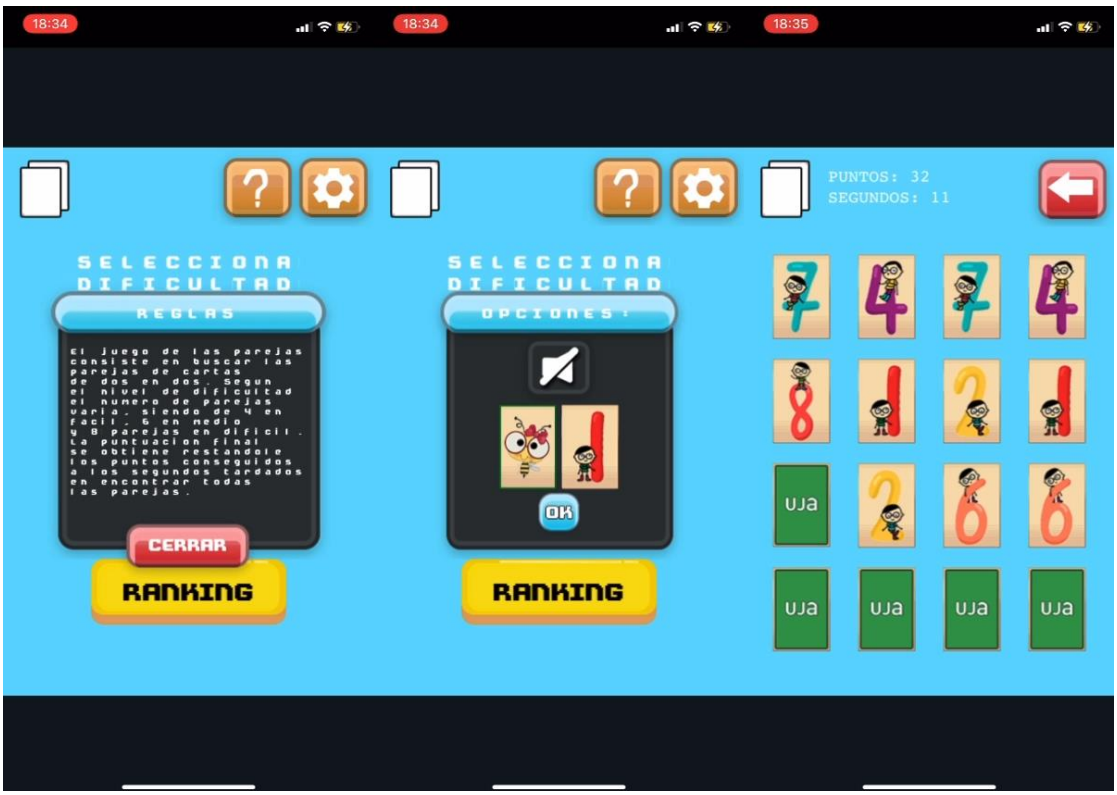


Ilustración 10.4 Prueba de PWA en IPhone

10.2. Cuestionario

Nombre: _____

Edad: ___

"Estudio de usabilidad"	
Pregunta	**Puntuación entre 0 y 5 (siendo 0 muy desacuerdo y 5 muy de acuerdo)**
1.- Me gustaría utilizar esta aplicación frecuentemente	[ ]
2.- No he encontrado bugs o errores	[ ]
3.- Visualmente, la aplicación es elegante (colores, textos, legibilidad…)	[ ]
4.- No he necesitado conocimientos avanzados en nuevas tecnologías para usar la aplicación	[ ]
5.- Las opciones de la aplicación estaban bien integradas	[ ]
6.- En mi opinión, la mayoría de personas podrían aprender a utilizar la aplicación rápidamente	[ ]
7.- He podido instalar la aplicación fácilmente	[ ]
8.- La aplicación cumple con su cometido inicial	[ ]
9.- Los botones son lo suficientemente grande como para poder pulsarlos correctamente	[ ]
10.- En términos generales, la aplicación fue fácil de usar	[ ]

Observaciones:

11. Bibliografía

[1] Ley de Moore. [Online]. Disponible en: https://es.wikipedia.org/wiki/Ley_de_Moore [Último acceso 1 Septiembre 2021]

[2] Evolución del protocolo HTTP. [Online]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP [Último acceso 1 Septiembre 2021]

[3] Protocolo de transferencia de hipertexto. [Online]. Disponible en: https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto [Último acceso 1 Septiembre 2021]

[4] Tiempo de ida y vuelta (RTT). [Online]. Disponible en: https://es.wikipedia.org/wiki/Tiempo_de_ida_y_vuelta [Último acceso 1 Septiembre 2021]

[5] Apuntes asignatura “Desarrollo de aplicaciones empresariales” curso 2020/2021. Impartida por el profesor Antonio Jesús Rueda Ruíz.

[6] The evolution of the web. [Online]. Disponible en: http://www.evolutionoftheweb.com/?hl=es [Último acceso el 31 de Enero de 2021]

[7] What are Progressive Web Apps?. [Online]. Disponible en: https://web.dev/what-are-pwas/ [Último acceso 1 Septiembre 2021]

[8] Jason Grigsby (2018). “Progressive Web Apps”. A Book Apart

[9] 10 Best PWA Examples That Would Urge You to Start Your Own. [Online]. Disponible en: https://bsscommerce.com/blog/pwa-examples/ [Último acceso 1 Septiembre 2021]

[10] The next billion users: trivago embrace progressive web apps as the future of mobile. [Online]. Disponible en: https://www.thinkwithgoogle.com/intl/en-gb/marketing-strategies/app-and-mobile/trivago-embrace-progressive-web-apps-as-the-future-of-mobile/ [Último acceso 1 Septiembre 2021]

[11] Por qué una Aplicación Web Progresiva podría ser adecuada para ti. [Online]. Disponible en: https://www.thinkwithgoogle.com/intl/es-es/estrategias-de-marketing/aplicaciones-y-moviles/por-que-una-aplicacion-web-progresiva-podri-ser-adecuada-para-ti/ [Último acceso 1 Septiembre 2021]

[12] PWA Marketplaces: Top examples from benchmarking study. [Online]. Disponible en: https://divante.com/blog/top-30-pwa-marketplaces/ [Último acceso 1 Septiembre 2021]

[13] Appscope: las aplicaciones web progresivas que te liberan espacio en el móvil. [Online]. Disponible en: https://www.aplicacionesparamoviles.com/appscope-aplicaciones-web-progresivas-pwa/ [Último acceso 1 Septiembre 2021]

[14] Progressive Web Apps vs Native Apps: Detailed Comparison. [Online]. Disponible en: https://www.codica.com/blog/progressive-web-apps-vs-native/ [Último acceso 1 Septiembre 2021]

[15] Apuntes asignatura “Desarrollo Ágil” curso 2019/2020. Impartida por el profesor Víctor M. Rivas Santos.

[16] ¿Qué es el SEO y por qué lo necesito?. [Online]. Disponible en: https://www.40defiebre.com/guia-seo/que-es-seo-por-que-necesito [Último acceso 1 Septiembre 2021]

[17] ¿Cómo podemos optimizar las PWA para mejorar el SEO?. [Online]. Disponible en: https://www.ttandem.com/blog/que-son-las-pwa-o-progressive-web-applications/las-pwa-y-el-seo/ [Último acceso 1 Septiembre 2021]

[18] Qué es Lighthouse. [Online]. Disponible en: https://www.am-design.es/que-es-lighthouse [Último acceso 1 Septiembre 2021]

[19] Brad Williams, David Damstra, Hal Stern (2015). “Professional WordPress: Design and Development, 3rd Edition”. Wrox.

[20] ¿Qué es un CMS?. [Online]. Disponible en: https://www.hostinger.mx/tutoriales/que-es-un-cms [Último acceso 1 Septiembre 2021]

[21] Qué CMS usar para e-commerce y cuál es el más recomendable para tu tienda. [Online]. Disponible en: https://www.doofinder.com/es/blog/cms-tienda-online [Último acceso 1 Septiembre 2021]

[22] Find out what websites are built with. [Online]. Disponible en: https://www.wappalyzer.com/ [Último acceso 1 Septiembre 2021]

[23] PWA, Qué Es, Plugins y Cómo Convertir Tu Web en App. [Online]. Disponible en: https://ignaciosantiago.com/pwa-wordpress/ [Último acceso 1 Septiembre 2021]

[24] Phaser 3 API Documentation. [Online]. Disponible en: https://newdocs.phaser.io/docs/3.55.2 [Último acceso 1 Septiembre 2021]

[25] Ventajas y Desventajas. PhaserDoc. [Online]. Disponible en: https://overnaru.gitbooks.io/phaserdoc/content/methods.html [Último acceso 1 Septiembre 2021]

[26] Phaser Progressive Web Apps Tutorial – How to Create Offline-First Games. [Online]. Disponible en: https://gamedevacademy.org/phaser-progressive-web-apps-tutorial/ [Último acceso 1 Septiembre 2021]

[27] Cómo usar Jira Software | Guía oficial del comprador y el usuario. [Online]. Disponible en: https://www.atlassian.com/es/software/jira/guides [Último acceso 1 Septiembre 2021]

[28] La importancia de usar sistemas de control de versiones. [Online]. Disponible en: https://platzi.com/blog/git-control-versiones/ [Último acceso 1 Septiembre 2021]

[29] Memoria – Busca las parejas. [Online]. Disponible en: https://adgaming.ibv.org/es/contenido-formativo/buscar-juegos-serios/listado-de-resultados/memory-busca-las-parejas/ [Último acceso 1 Septiembre 2021]

[30] Método MoSCoW para priorizar tareas. [Online]. Disponible en: https://marialeal.com/priorizar-metodo-moscow/ [Último acceso 1 Septiembre 2021]