

Proyecto de Desarrollo de Aplicaciones Web

JUEGO MULTIPLATAFORMA CON IONIC



I.E.S. Maria Enríquez
Gandia



ionic



Presentado por **Arturo Jerez Guerrero**

Ciclo Formativo de Grado Superior

Desarrollo de Aplicaciones Web

Curso **2018 - 2019**

Tabla de contenido

1. INTRODUCCIÓN	3
1.1. DESCRIPCIÓN DEL PROYECTO	3
1.2. INFORMACIÓN GENERAL	4
1.3. OBJETIVOS DEL PROYECTO	5
2. CONTEXTUALIZACIÓN DEL PROYECTO	6
2.1 OBJETIVOS CONCRETOS	6
2.2 TECNOLOGÍAS EVALUADAS	7
2.3 TECNOLOGÍAS ALTERNATIVAS	12
3. PLANIFICACIÓN DEL PROYECTO	12
3.1. TEMPORALIZACIÓN	13
4. DESARROLLO DEL PROYECTO	14
4.0. FASE 0: FASE PREVIA	14
4.0.1. INSTALACIÓN	14
4.0.1.1. IONIC FRAMEWORK	14
4.0.2. CONFIGURACIÓN	15
4.0.2.1. CREANDO NUESTRO PRIMER PROYECTO	15
4.1. FASE 1: FASE DE CONCEPCIÓN	17
4.1.1. IDEA PRINCIPAL	17
4.1.2. GÉNEROS	19
4.2. FASE 2: FASE DE DISEÑO	19
4.2.1. TEMPLATES DE IONIC FRAMEWORK	19
4.2.1.1. TEMPLATE TABS	20
4.2.1.2. TEMPLATE SIDE MENU	20
4.2.1.3. TEMPLATE BLANK	21
4.2.2. COLORES	21
4.2.3. CICLO DEL JUEGO	23
4.3. FASE 3: FASE DE PLANIFICACIÓN	23
4.4. FASE 4: FASE DE PRODUCCIÓN	26
4.4.1. CONTROLADORES	27
4.4.2. SERVICIOS	28
4.4.3. MODELOS	32

4.4.4. VISTAS	33
4.4.5. IMPLEMENTACIÓN	33
4.4.5.1. SONIDO	33
4.4.5.2. VIBRACIÓN	34
4.4.5.3. MENÚ	35
4.5. FASE 5: FASE DE PRUEBAS	36
4.6. FASE 6: FASE DE MARKETING	37
4.7. FASE 7: FASE DE MANTENIMIENTO	37
5. ESTRUCTURA VISUAL DE LA APLICACIÓN	38
5.1. ESQUELETO DE LA APLICACIÓN	38
5.2. ENTORNO VISUAL	38
6. TESTS DE PRUEBAS	41
8. BIBLIOGRAFÍA	41
8.1. LIBROS Y VÍDEOS	42
8.2. MANUALES WEB	42
8.3. FOROS/BLOGS	42
7. ANÁLISIS Y VALORACIÓN DE ALTERNATIVAS	43
7.1. ANÁLISIS Y CONCLUSIONES	43
7.1.1. OBJETIVOS	43
7.1.2. CONCLUSIONES	43
7.1.3. AUTOEVALUACIÓN	44
7.2. MEJORAS O ALTERNATIVAS PROPUESTAS	44
7.3. AGRADECIMIENTOS	45
9. ANEXOS	45
9.1. ESTRUCTURA DE CARPETAS	45
9.1.1. PÁGINAS	45
9.1.2. SERVICIOS	46
9.1.3. ASSETS	46
9.2. COMPILACIÓN PHONEGAP BUILD	46

1. INTRODUCCIÓN

En el proyecto se desarrolla el juego multiplataforma para dispositivos móviles de Simon Game. Este juego fue creado en 1978 por Ralph Baer y Howard J. Morrison siendo uno de los precursores de la industria de los videojuegos que por aquella época empezaba a nacer. Se trataba de una consola circular de juego que encendía luces y emitía sonidos que el jugador debía repetir en el mismo orden de la secuencia mostrada, pulsando las teclas de colores.



No fue hasta mediados de los noventa cuando este juego se desarrolló en la versión arcade. Fue desarrollado para Little Caesars Pizza y lo llamaron Brain Teaser.



A lo largo de los años la empresa que creó el juego fue lanzado diferentes modelos. Uno de los más populares fue Simon Pocket. Otras compañías también lanzaron juegos similares, siendo éstos un calco a Simon. Por ejemplo, la compañía Atari lanzó Touch Me, que era una maquinita que disponía de una pantalla led donde se veía la puntuación y una botonera de colores para que el jugador reprodujera la secuencia. La finalidad del juego era el mismo que el de Simon.

Actualmente existen varios juegos de Simon con estilos disponibles, alternando su aspecto y diferentes plataformas con las que disponer de él.

En el siguiente proyecto, se va a desarrollar el juego clásico mejorado como se verá de forma más detallada en los siguientes apartados.

1.1. DESCRIPCIÓN DEL PROYECTO

El proyecto consiste en una vez realizado un estudio de los elementos software existentes, conocer la mecánica del juego Simon para agregar una lógica que hace del juego un sistema funcional y jugable.

Para conseguir esto será necesario buscar las herramientas adecuados, seleccionar el entorno de desarrollo que más nos convenga. También es necesario saber las librerías existentes para el control de los elementos que interactúan con el usuario. Con toda la información software disponible se tiene que ser capaz de desarrollar el juego en el entorno que se ha elegido.

La mecánica del Simon consiste en la repetición por parte del jugador de una secuencia mostrada.

El juego ejecuta una secuencia de cuatro colores asociado a cuatro sonidos que el usuario deberá memorizar para posteriormente reproducirla.

La forma de reproducir la secuencia es presionando el botón del color correspondiente, ya sea verde, rojo, amarillo o azul.

La secuencia se debe reproducir tal cual el juego la ha mostrado, si no se reproduce exactamente igual, el jugador habrá perdido.

El juego dispone de varios niveles y cuatro dificultades, en cada nivel la secuencia a reproducir será mayor. En la dificultad fácil, el primer nivel la secuencia dispone de un valor. La dificultad irá aumentando y la secuencia pasará a tener una longitud mayor, incrementándose en cada nivel en uno. La dificultad media se incrementa en dos, la difícil en tres, y por último el hardcore.

El juego termina cuando el usuario falla una de las secuencias reproducidas.

1.2. INFORMACIÓN GENERAL

Nombre del Alumno: Arturo Jerez Guerrero

Ciclo: Desarrollo de Aplicaciones Web

Curso escolar: 2017 - 2019

Denominación del proyecto: Joc multiplataforma amb IONIC

Proyecto vinculado a FCT: No vinculado

Tutor Individual: Enrique Savall Seguí

Tutor Colectivo: José Ramón, Migueles Hernández



1.3. OBJETIVOS DEL PROYECTO

El objetivo principal del proyecto es el desarrollo y correcto funcionamiento del juego Simon corriendo en las diferentes plataformas móviles. Para obtener este objetivo, se han creado objetivos parciales divididos en principales y secundarios. Los objetivos principales serán aquellos que doten de la funcionalidad principal del juego y los objetivos secundarios serán los que añaden valor a la aplicación.

1. Objetivos principales

- a. Generación de secuencias
- b. Incluir distintos niveles de dificultad en el juego (Fácil, Medio, Difícil, Hardcore).
- c. Mostrar puntuación
- d. Incluir sonido
- e. Mostrar mensajes de ganado o perdido

2. Objetivos secundarios

- a. Mostrar menú desplegable
- b. Botón de Jugar.
- c. Botón de Cambiar nombre de usuario
- d. Almacenamiento del progreso en local
- e. Efectos de luz para cada botón
- f. Tutorial/Guía del juego
- g. Mostrar información breve sobre el juego y desarrollador

2. CONTEXTUALIZACIÓN DEL PROYECTO

He elegido este proyecto para un reto personal en el que intentó mejorar la versión clásica del juego Simon. Me pareció interesante desarrollar por mí mismo el clásico Simon, y además, mejorarlo a una nueva versión que amplía su jugabilidad.

También lo he elegido para publicarlo en el play store y poder monetizarlo, esto incluye que personas de todas las edades lleguen a jugarlo y opinar sobre él.

2.1 OBJETIVOS CONCRETOS

El juego intenta mejorar un diseño gráfico más atractivo para simón, además de su interactividad con el usuario que agrega un mejor entretenimiento.

El juego sigue una simple y única regla de generar una secuencia de colores que el usuario debe copiar en el orden correcto, cuantos más niveles haya superado, más complicado se volverá para el jugador memorizar toda la secuencia.

La máquina aumenta un color cada nivel superado.

Para conseguirlo, me marco unos items que mi juego incluirá:

Distintos niveles de dificultad: Contiene cuatro diferentes modos de juego.

- **Fácil:** Cuenta la secuencia de uno en uno, es el modo más sencillo del juego.
- **Medio:** Este modo es un poco más complicado que cuenta la secuencia de dos en dos, está pensado para jugadores más expertos del anterior modo.
- **Difícil:** Es mucho más complicado que el modo anterior, cuenta la secuencia de tres en tres, está pensado para los jugadores profesionales.
- **Hardcore:** Este modo es un poco diferente a los demás, no cuenta la secuencia progresivamente, sino que, cuenta la última secuencia añadida anteriormente más la nueva a añadir, haciendo del juego mucho más entretenido y complicado.

Menú intuitivo y fácil:

- **Dificultad:** Se muestra una lista con las cuatro diferentes modalidades.
 - Fácil
 - Medio
 - Difícil
 - Hardcore

- **Sonido:** Podemos activar o desactivar el equipo de sonido.
- **Ranking:** Muestra las puntuaciones del/los jugador/es.
- **Créditos:** Es una pequeña información del creador, la versión de la aplicación, y guía del juego.
- **Guía:** Muestra información sobre elementos de la aplicación y cómo jugar.

Otros añadidos:

- Sistema de vibración para los botones, esto es de gran ayuda por sí un jugador prefiriera jugar sin sonido.
- Ranking: Almacenamiento del nombre de usuario y su mejor puntuación en la plataforma local.

2.2 TECNOLOGÍAS EVALUADAS

Para el desarrollo de este proyecto se han utilizado elementos software.

Estos elementos son:

1. Herramientas:

- a. IONIC
- b. Angular
- c. TypeScript
- d. SASS
- e. HTML
- f. Node.js
- g. Git

2. Elementos software:

- a. Android
- b. IOS
- c. Windows 10
- d. Visual Studio Code

IONIC:

- El proyecto está realizado con un completo SDK de código abierto para el desarrollo de aplicaciones móviles híbridas.

- **¿Por qué lo utilizo?**

- Es una herramienta idónea para poner nuestra idea en desarrollo. Estructura el diseño de la aplicación gracias a unos componentes que hacen de la aplicación muy agradable a la vista, tiene una documentación limpia.

- **Ventajas:**

- Compatible en varias plataformas.
- Interfaz de usuario amigable, en el que incorpora JS y componentes CSS para la optimización.
- Framework libre y de código abierto que facilita la personalización del diseño gracias a componentes CSS, JS y HTML.
- Basado en Angular que amplía facilidad y funcionalidad para el desarrollo de las aplicaciones móviles más fácilmente.



ANGULARJS:

- Aumenta las aplicaciones en navegador basadas en modelo vista controlador (MVC) que con ello mejora la testeabilidad y rendimiento.

Mejora en la estructuración del contenido que sincroniza los modelos y vistas. Ionic utiliza AngularJS con el fin de crear un marco más

adecuado para desarrollar aplicaciones ricas y robustas. Ionic no sólo se ve bien, sino que su arquitectura central es robusta y seria para el desarrollo de aplicaciones.

- **¿Por qué lo utilizo?**

- Utiliza unas técnicas propias que hacen que, sin necesidad de buscar o perderse por internet, el usuario pueda desarrollar una app sin demasiado esfuerzo en pensar que hacer, o cómo hacerlo, ayuda bastante la documentación de este puesto que utiliza lenguaje typescript.

Este agrupa todas las funcionalidades para hacer routing, testing, sencillamente sin agregarlo todo desde cero.



- **Ventajas:**

- Nos facilita no tener que pensar demasiado para empezar a desarrollar nuestra app.
- Puede programar en ECMAScript puro con lenguaje typescript.
- Angular utiliza los componentes web para adoptar el contenido de manera mucho más atractiva.
- Seguirá siendo estable en cuanto a futuras versiones del framework.
- Utiliza un gran soporte de herramientas.
- Ionic trabaja perfectamente con AngularJS.

- **Características:**

- AngularJS es el framework más utilizado para el desarrollo de aplicaciones web.
- El uso del lenguaje typescript potenció sus características
- Puede Integrarse con tecnologías, para generar proyectos CLI en node.js, y trabajar con las vistas con React.

TYPESCRIPT :

- Es un lenguaje de código abierto basado en JavaScript, y uno de los más utilizados con el framework de angular. Cuenta con herramienta orientada a objetos, además de utilizar un compilador propio que este es ejecutado en JavaScript.



- **¿Por qué lo utilizo?**

- En gran parte, este lenguaje es generado y utilizado en conjunto con el framework de angular de manera que todo el proyecto ha sido realizado con TypeScript, exceptuando algunas funciones adicionales para añadir efectos que utilicé JavaScript puro.

- **Ventajas:**

- Escalabilidad del código en una interfaz orientada al desarrollo.
- Posee una gran comunidad y herramientas.
- Demuestra promover la cualidad del código y entendibilidad.
- Testeabilidad. Con la inyección de dependencias, testeo fácil.
- Promueve la innovación y el cambio.
- Utiliza un código limpio.
- Se puede escribir fácilmente código orientado a objetos sin mucho conocimiento.
- El Refactoring con las herramientas de Typescript es mucho más fácil y rápido.

SASS:

- Basta con pequeños toques de pinceladas para dar estilo a un párrafo o un título, esta herramienta proporciona una gran utilidad para unificar varias cajas de estilo en un solo cajón.
- **¿Por qué lo utilizo?**
 - Proporciona unos estilos a la aplicación muy agradables a la vista, guarda bloques de código mixins para hacer más flexible los estilos.
- **Ventajas:**
 - Función de anidamiento que permite escribir jerarquías o niveles.
 - Función de mixins.
 - Compatibilidad con CSS3.
 - Ficheros css bien formados.
 - Contenidos elegantes, divertidos y versátiles al incluir funciones para manipular en forma rápida y efectiva los colores, tamaños, fuentes, y otros valores.



HTML:

- Un lenguaje de marcas que moldea la estructura de una página a gusto de los usuarios, es una buena forma de realizar una página web desde cero.
- **¿Por qué lo utilizo?**
 - Ayuda mucho a reestructurar partes pequeñas de la aplicación, así se le puede dar forma a gusto de cualquiera. La experiencia trabajando con este lenguaje me ha llevado a aprender mucho sobre sus atajos y propiedades que hacen de él, un manejo muy efectivo y productivo para nuestras aplicaciones.
- **Ventajas:**
 - Sencillo que permite describir hipertexto.
 - Texto presentado de forma estructurada y agradable.
 - No necesita de grandes conocimientos cuando se cuenta con un editor de páginas web o WYSIWYG.
 - Archivos pequeños.
 - Despliegue rápido.
 - Lenguaje de fácil aprendizaje.



- admiten todos los exploradores.

Node.js:



- Entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos.
- ¿Por qué lo utilizo?
 - Me ha servido para instalar plugins, dependencias de ionic, dentro del IDE de visual studio code. Tiene compatibilidad absoluta con esta herramienta y muchas más.
- **Ventajas:**
 - Es fácil de aprender dado su parecido a JavaScript.
 - Rendimiento de gran calidad y disminuye el margen de experimentar errores técnicos.
 - Permite crear aplicaciones altamente escalables e innovadoras.
 - Desarrollo de aplicaciones mucho más rápido.
 - Reduce los tiempos de trabajo.

Git:



- Un sistema de control de versiones para el flujo del trabajo que proporciona una eficiencia y confiabilidad del mantenimiento de las versiones de aplicaciones.
- ¿Por qué lo utilizo?
 - Me proporciona unas herramientas para desarrollar un trabajo de manera inteligente y rápida para un gran número de archivos.
- **Ventajas:**
 - Rapidez en la gestión de ramas.
 - Gestión distribuida, los cambios se importan como ramas adicionales.
 - Gestión eficiente de proyectos grandes.
 - Realmacenamiento periódico en paquetes.

2.3 TECNOLOGÍAS ALTERNATIVAS

Para este proyecto también se tuvieron en cuenta otros elementos que fueron descartadas.

FIREBASE:



- Esta aplicación compila apps rápido, sin administrar la infraestructura construida Google. Tiene alta Compatibilidad con los dispositivos móviles.
- **Ventajas:**
 - Almacena y sincroniza los datos de la app a escala global
 - Ejecuta código back-end
 - Autentica usuarios de forma simple y segura
 - Entrega recursos de aplicaciones web con velocidad y seguridad
 - Almacena y envía archivos a google
 - Almacena y sincroniza datos de app en milisegundos
 - Prioriza y soluciona problemas con informes de fallas potentes y en tiempo real
 - Posee testeabilidad propia alojados en google
- **Desventajas:**
 - Límite de conexiones simultáneas
 - Base de datos no tan complejas
 - Algunas funcionalidades no disponibles en las analíticas
 - Pruebas en la nube limitadas
 - Precios

3. PLANIFICACIÓN DEL PROYECTO

El proyecto sigue unas fases que mantiene la aplicación en su mejor estado.

1. **Fase de Concepción:** Añade el género o géneros, aspecto e ideas que se quiere que tenga la aplicación.
2. **Fase de Diseño:** Agrega estilo, un ciclo del juego, y efectos.
3. **Fase de Planificación:** Se calcula el tiempo de los objetivos.
4. **Fase de Producción:** La programación, el arte e implementación.
5. **Fase de Pruebas:** Errores y bugs a corregir.
6. **Fase de Distribución:** Asegura el funcionamiento, versiones.

- 7. Fase de Mantenimiento:** Parches, actualizaciones y nuevas implementaciones que se llevarán a cabo para la aplicación.

3.1. TEMPORALIZACIÓN

Para poder llevar a cabo el seguimiento de las fases, desde el comienzo se elaboró un diagrama de Gantt que ha ido evolucionando a medida que se ha desarrollado el proyecto.

	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9	Semana 10	Semana 11	Semana 12
FASE DE CONCEPCIÓN												
FASE DE DISEÑO												
FASE DE PLANIFICACIÓN												
FASE DE PRODUCCIÓN												
FASE DE PRUEBAS												
FASE DE DISTRIBUCIÓN												
FASE DE MANTENIMIENTO												
TIEMPO	4 Días	3 Días + 2 Días	5 Días + 1 Día	6 Días + 7 Días	7 Días	7 Días	5 Días	2 Días + 7 Días	7 Días	6 Días	1 Día + 7 Días	7 Días
TIEMPO CALCULADO	4	5	6		32			9		13		15
TIEMPO TOTAL	84	Días										

4. DESARROLLO DEL PROYECTO

4.0. FASE 0: FASE PREVIA

Esta fase cuenta con una guía de instalación de todas las herramientas que he necesitado para instalar ionic.

En la guía combinó la mayoría de herramientas que he utilizado con la instalación del propio framework de ionic, adjunto además para cada herramienta un enlace del sitio web oficial dónde se redacta más información sobre estas herramientas y el instalador que esta web trae consigo.

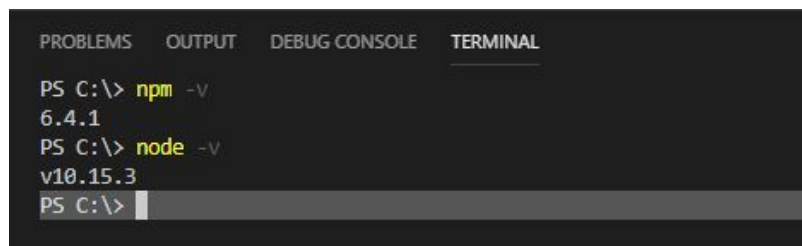
4.0.1. INSTALACIÓN

4.0.1.1. IONIC FRAMEWORK

El proceso de instalación es sencillo, debemos tener el entorno de visual studio code instalado antes, teniendo esto preparado vamos con la instalación.

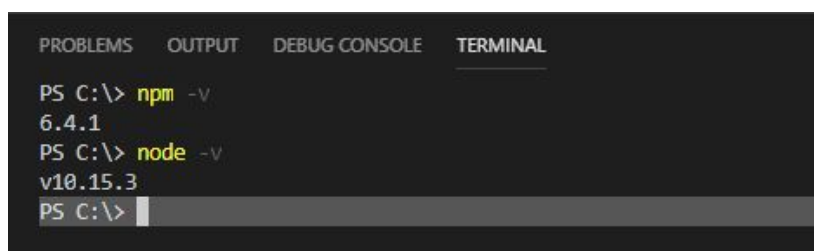
Primero que nada, deberemos instalar NodeJS ya que este cuenta con un manejador de paquetes llamado npm el cual nos será necesario para instalar muchas de nuestras dependencias. Para instalar NodeJS debemos dirigirnos al [sitio oficial](#) y descargar su instalador.

Para poder saber que la instalación fue satisfactoria y además para saber qué versión instalamos de NodeJS y de npm se pueden correr los comandos **node -v**, **npm -v**. Esto es posible hacerlo mediante una terminal que tiene el entorno vscode integrado en la pestaña de terminal.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\> npm -v
6.4.1
PS C:\> node -v
v10.15.3
PS C:\>
```

A continuación, deberemos instalar Git el cual es necesario para tener un control de versionado de nuestro proyecto y además es utilizado para descargar librerías que necesita ionic para ser ejecutado. Esto lo pueden descargar e instalar desde su [sitio oficial](#) y al igual que para NodeJS se puede saber si la instalación fue realizada correctamente ejecutando desde una terminal el comando **git --version**, el cual nos dará qué versión hemos instalado en nuestro ordenador.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\> npm -v
6.4.1
PS C:\> node -v
v10.15.3
PS C:\>
```

Por último, para completar el setup de ionic instalaremos el ionic CLI que es un complemento del Cordova CLI al que añade una serie de características adicionales. Para instalar Apache Cordova debemos hacerlo a través del manejador de paquetes npm, lo que debemos hacer es abrir una terminal y ejecutar el comando **npm install -g cordova**.

Se instalar con la variable -g para así hacerlo de forma global y poder acceder a él desde cualquier directorio. Una vez lo instalemos podemos verificar que todo haya salido correctamente ejecutando el comando **cordova -v**.



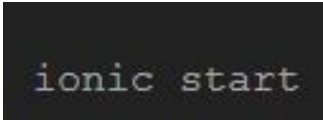
```
PS C:\> cordova -v
9.0.0
```

Luego de tener todo esto instalado procederemos a incorporar Ionic Framework, para realizar esto debemos abrir una terminal e ingresar como administrador para luego ejecutar el comando **npm install -g ionic**. Una vez esté instalado comprobamos como en los casos anteriores que versión tenemos, esto lo podemos ver ejecutando el comando **ionic -v**.

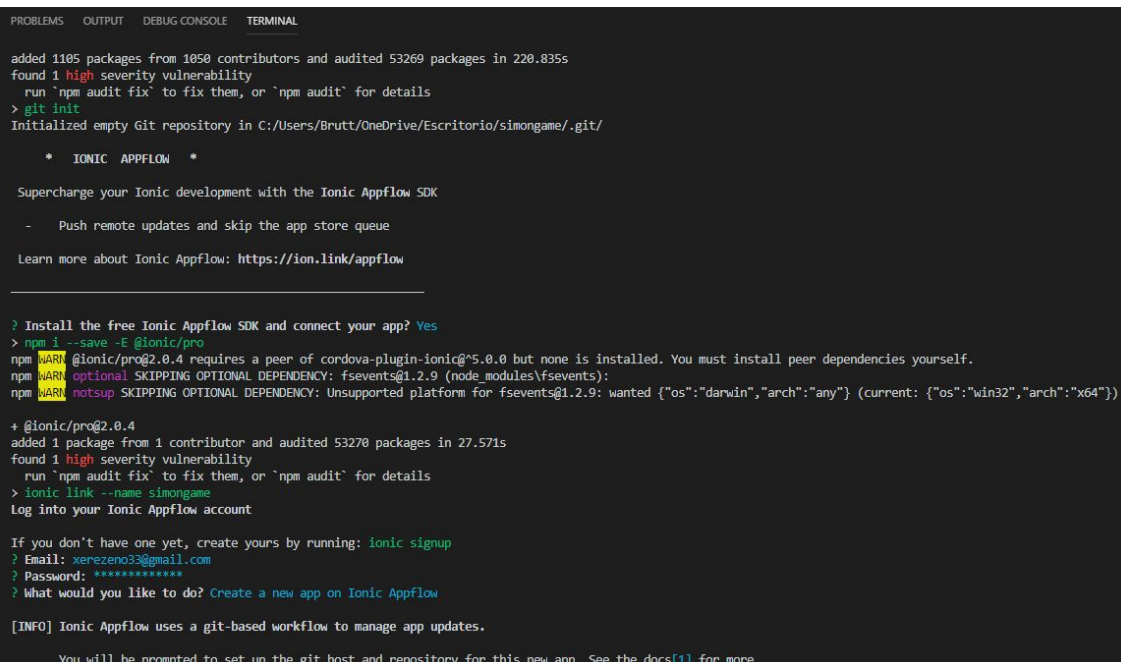
4.0.2. CONFIGURACIÓN

4.0.2.1. CREANDO NUESTRO PRIMER PROYECTO

Vamos a comenzar con la creación de nuestro proyecto en donde utilizaremos el template tabs que es el que viene por defecto. Primero que nada, deberemos de crear una carpeta en donde almacenaremos nuestros distintos proyectos, en mi caso la cree en el escritorio con el nombre de “simongame”. Una vez realizado esto debemos acceder a la terminal de vscode y situarnos en dicha carpeta recién creada, una vez hecho esto debemos ejecutar el siguiente comando.



```
ionic start
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
added 1105 packages from 1050 contributors and audited 53269 packages in 220.835s
found 1 high severity vulnerability
  run `npm audit fix` to fix them, or `npm audit` for details
> git init
Initialized empty Git repository in C:/Users/Brutt/OneDrive/Escritorio/simongame/.git/

* IONIC APPFLOW *

Supercharge your Ionic development with the Ionic Appflow SDK

- Push remote updates and skip the app store queue

Learn more about Ionic Appflow: https://ion.link/appflow

? Install the free Ionic Appflow SDK and connect your app? Yes
> npm i --save -E @ionic/pro
npm WARN @ionic/pro@2.0.4 requires a peer of cordova-plugin-ionic@*5.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ @ionic/pro@2.0.4
added 1 package from 1 contributor and audited 53270 packages in 27.571s
found 1 high severity vulnerability
  run `npm audit fix` to fix them, or `npm audit` for details
> ionic link --name simongame
Log into your Ionic Appflow account

If you don't have one yet, create yours by running: ionic signup
? Email: xerezeno3@gmail.com
? Password: *****
? What would you like to do? Create a new app on Ionic Appflow

[INFO] Ionic Appflow uses a git-based workflow to manage app updates.

You will be prompted to set up the git host and repository for this new app. See the docs[1] for more
```


Una vez creado nuestro proyecto el cual le colocamos como nombre **simongame** debemos situarnos en la carpeta que nos creó con dicho nombre ejecutando el comando **cd simongame**. Luego de hacer esto debemos ejecutar nuestro servidor para poder ver nuestra aplicación ejecutado en nuestro navegador que tengamos por defecto, es recomendable que se utilice Google Chrome ya que tiene un apartado para desarrolladores que nos será de utilidad. Para ejecutar esto debemos ejecutar desde la terminal y situados en nuestra carpeta “simongame” el siguiente comando:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Brutt\OneDrive\Escritorio\musical> ionic serve
> ng run app:serve --host=0.0.0.0 --port=8100
[ng] WARNING: This is a simple server for use in testing or debugging Angular applications
[ng] locally. It hasn't been reviewed for security issues.
[ng] Binding this server to an open connection can result in compromising your application or
[ng] computer. Using a different host than the one passed to the "--host" flag might result in
[ng] websocket connection issues. You might need to use "--disableHostCheck" if that's the
[ng] case.
[INFO] Waiting for connectivity with ng...
[INFO] Waiting for connectivity with ng...

[INFO] Development server running!

Local: http://localhost:8100
External: http://192.168.1.103:8100

Use Ctrl+C to quit this process

[INFO] Browser window opened to http://localhost:8100!

[ng] i [wdm]: wait until bundle finished: /
[ng] Date: 2019-05-11T14:14:10.470Z
[ng] Hash: 434ba32b3d72c7eb0a94
[ng] Time: 36167ms
[ng] chunk {common} common.js, common.js.map (common) 18.5 kB [rendered]
[ng] chunk {es2015-polyfills} es2015-polyfills.js, es2015-polyfills.js.map (es2015-polyfills) 284 kB [initial] [rendered]
[ng] chunk {home-home-module} home-home-module.js, home-home-module.js.map (home-home-module) 6.28 kB [rendered]
[ng] chunk {main} main.js, main.js.map (main) 31.8 kB [initial] [rendered]
[ng] chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 237 kB [initial] [rendered]
[ng] chunk {runtime} runtime.js, runtime.js.map (runtime) 8.79 kB [entry] [rendered]
[ng] chunk {styles} styles.js, styles.js.map (styles) 84.6 kB [initial] [rendered]
[ng] chunk {vendor} vendor.js, vendor.js.map (vendor) 4.36 MB [initial] [rendered]
[INFO] ... and 173 additional chunks
[ng] i [wdm]: Compiled successfully.
```

Luego de ejecutar esta acción se nos abrirá una pestaña nueva en el navegador con nuestra aplicación ejecutándose, para poder utilizar las funciones de desarrollador de Google Chrome y ejecutar nuestra app en base a un dispositivo móvil debemos de ir al menú de Chrome luego dirigirnos a la pestaña de “Más herramientas” y luego hacer click en la opción “Herramientas del desarrollador”, de esta forma se nos abrirá una solapa en la parte inferior del navegador en donde tendremos la opción de ver nuestra aplicación sobre un móvil. Para hacer esto basta solamente con pinchar en el botón que tiene como dibujo un celular, de manera más rápida se puede hacer ejecutando la combinación de teclas **Ctrl + Shift + m** o directamente podemos presionar **f12** y directamente entra el navegador en modo celular.

Una vez estemos ahí se podrá ver en la parte inferior, si generamos el proyecto cómo **tabs** se cuenta con 3 tabs con los nombres “Home”, “About” y “Contact” en los cuales podremos pinchar y

nos dirigirá a otra pantalla. Para poder modificar estos nombres o si es necesaria para agregar o eliminar alguno debemos editar el archivo **tabs.html** que se encuentra dentro de nuestro directorio **simongame**, la ruta específica de este archivo es la siguiente:

Simongame/src/pages/tabs/tabs.html

En este archivo veremos una serie de etiquetas que nos provee ionic en donde funcionan como tabs de nuestro proyecto, lo que nosotros vemos actualmente es lo siguiente:

<ion-tabs>

<ion-tab [root]="tab1Root" tabTitle="Home" tabIcon="home"></ion-tab>

<ion-tab [root]="tab2Root" tabTitle="About" tabIcon="information-circle"></ion-tab>

<ion-tab [root]="tab3Root" tabTitle="Contact" tabIcon="contacts"></ion-tab>

</ion-tabs>

4.1. FASE 1: FASE DE CONCEPCIÓN

En esta fase cuento los géneros, ideas que necesita y que se han agregado a la aplicación.

4.1.1. IDEA PRINCIPAL

Para empezar, mi idea tenía lugar en hacer una versión mejorada del clásico juego Simon para los distintos dispositivos móviles. Eso me dio que pensar para ver que podía incluir nuevo y que mejoras aplicarle. Estos son algunas de las ideas que se le han aplicado:

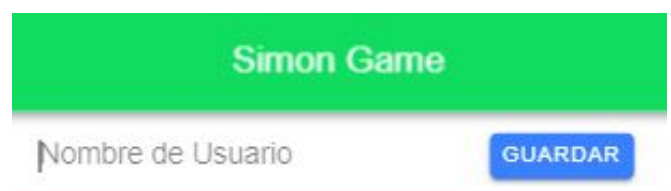
1. Realizar un panel de control(menú), que muestra las diferentes dificultades del juego.



Además, habilita y deshabilita el control de sonido, muestra las puntuaciones de los usuarios, una página donde muestra una pequeña información del juego y el creador, otra página que muestra una guía de cómo jugar.



2. Botón de cambiar el nombre del usuario que nos permite añadir nuevos nombres de jugadores, esto nos permite diferenciar a los usuarios que más puntuación tienen en el juego.



3. Almacenamiento local en el dispositivo para tener siempre el ranking y nombre de usuario guardados, sin necesidad de tener que cargarlas cada vez que entremos a la aplicación.



4.1.2. GÉNEROS

La aplicación contiene dos géneros que han sido aplicados por su estilo de juego, los géneros son:

1. Indie

Representa proyectos de bajo presupuesto que han sido desarrollados por empresas pequeñas o un personal de menos de 15 personas.

2. Puzzle

Este género ha sido definido para las aplicaciones que contienen rompecabezas al estilo clásico del Tetris.

4.2. FASE 2: FASE DE DISEÑO

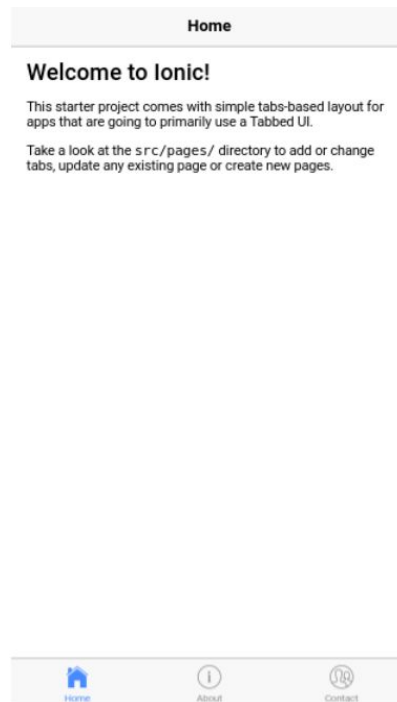
Aquí cuento y desarrollo el proceso que seguí para agregar los estilos, colores y darle una vista agradable a la aplicación.

4.2.1. TEMPLATES DE IONIC FRAMEWORK

Primero que nada, voy a hablar sobre los distintos templates con los que contamos ya que es fundamental para saber con cuál nos conviene empezar.

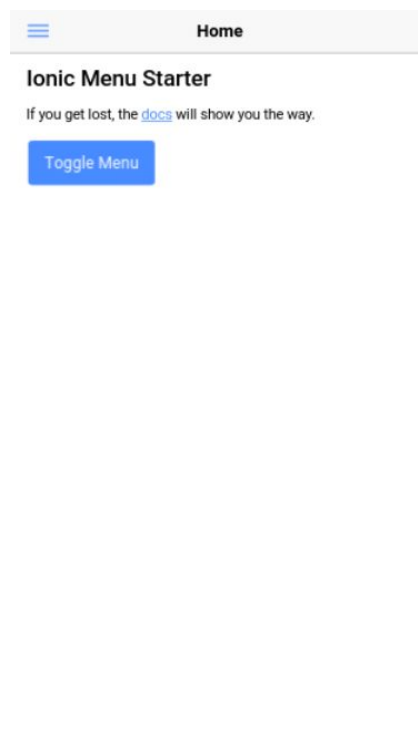
4.2.1.1. TEMPLATE TABS

Este nos brinda una serie de tres tabs como mencione en la fase previa, estos nos muestran información la cual puede ser modificado según nosotros necesitemos. Este template es el que viene por defecto si es que no se especifica ningún otro, más adelante veremos de cómo hacerlo.



4.2.1.2. TEMPLATE SIDE MENU

Dicho template nos crea un menú lateral para que nosotros luego lo modifiquemos a nuestra necesidad.



4.2.1.3. TEMPLATE BLANK

El template nos crea un proyecto en blanco, para que nosotros comencemos a crear nuestra aplicación desde cero.



4.2.2. COLORES

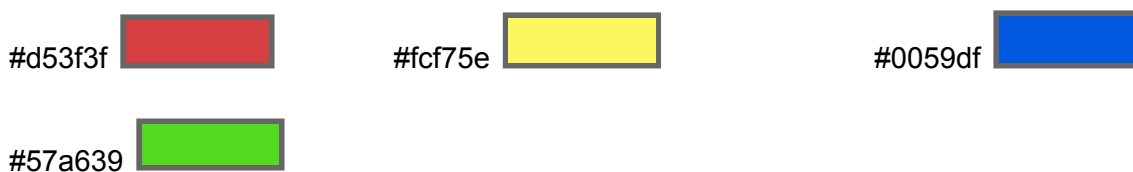
He utilizado una combinación de tonos frescos y agradables a la vista.

- **Botones**

Se basan en colores simples adornados con efectos la mayoría, los botones originales del juego forman los colores verde, rojo, amarillo y azul. Mientras que para los botones de la página principal le agregué el mismo tono que de la cabecera.

Concretamente el tono de colores ha sido estos:

Para los botones del Simon:



Para los botones de la página principal:

#00df00



- **Fondo**

Le agregue un color blanco puro, es sencillo y tiene una tonalidad exacta para visualizar el juego a la perfección.

#ffffff



- **Cabecera**

Tiene un color verde más vivo y fuerte que cambia el aspecto visual de la aplicación, su color original fue azul marino que no hacía buen contraste con el fondo blanco de la aplicación.

#00df00



- **Menú**

En el panel se le han agregado varios colores que ha sido para mí, una buena elección por su combinación de colores, se han agregado para el tono de fondo blanco nieve y para las cabeceras de cada lista, un rojo natural.

El tono blanco ha sido el mismo que del fondo.

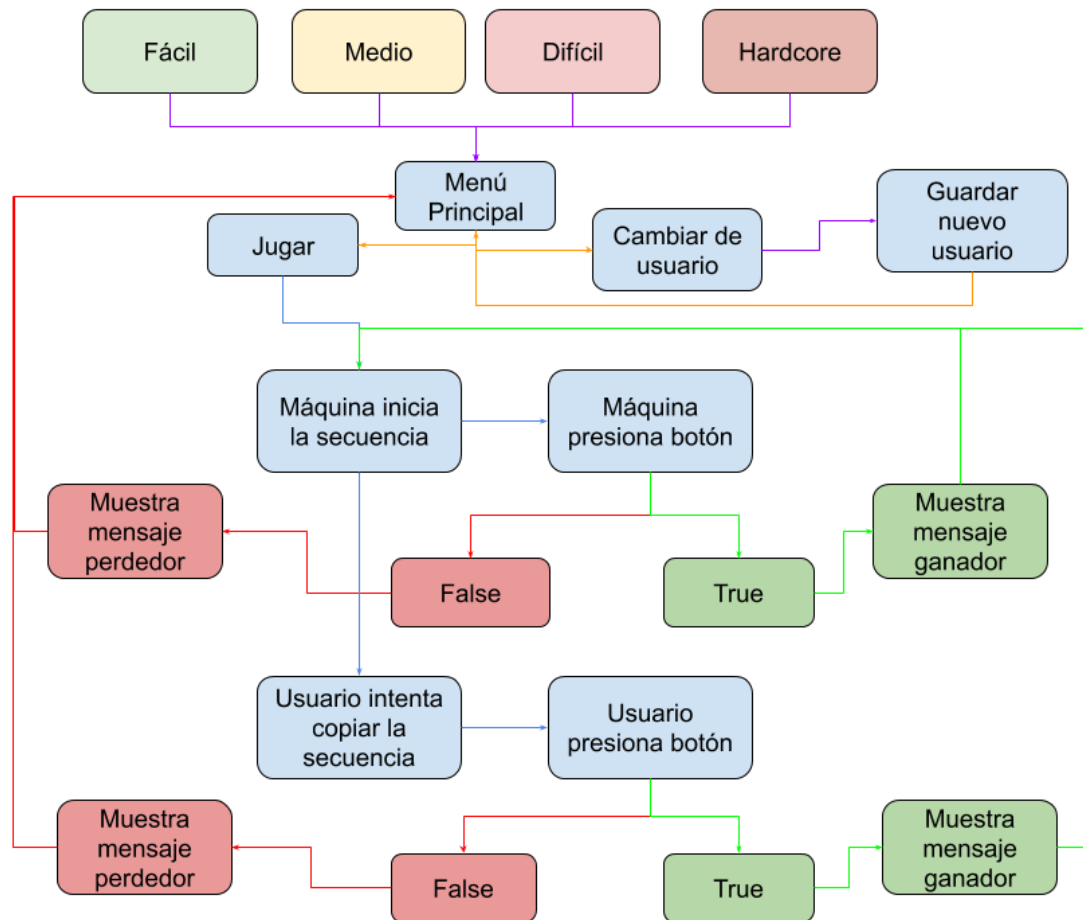
#ffffff



#ff0000



4.2.3. CICLO DEL JUEGO



4.3. FASE 3: FASE DE PLANIFICACIÓN

El tiempo que se invierte en una aplicación, se debe a la dificultad de los propios objetivos que se han propuesto. La mayoría de estas ya puede ser porque debemos agregar nuevas librerías, y no tenemos conocimientos de esta para seguir nuestro camino y tenemos que partir el tiempo de nuestro objetivo para estudiar esa nueva librería que implementamos en nuestra aplicación.

El tiempo que le he dedicado a los objetivos son:

Objetivos principales

1. Generación de secuencias

Ha sido el objetivo que más tiempo me ha llevado concluir, su tiempo total realizado fue alrededor de unas 17 horas

2. Incluir distintos niveles de dificultad en el juego (Fácil, Medio, Difícil, Hardcore).

Entre dificultad y dificultad no me ha llevado demasiado, ha sido sencillo realizar un cambio, puedo decir que ha sido uno de los objetivos más rápidos en concluir. Su tiempo total realizado fue de 2 horas y 13 minutos.

3. Mostrar nombre de usuario y puntuación

Este punto ha costado más que el anterior, pero menos que el primero, ya que tuve que almacenar los datos en local. Tuve problemillas del tercer mundo con esto puesto que almacenaba los datos, y al momento de mostrarlos, solo mostraba el contenido que se guardó anteriormente, pero se pudo resolver. Su tiempo total realizado fue alrededor de 7 u 8 horas.

4. Incluir sonido

Me ha tomado más o menos el mismo tiempo que el punto anterior, he encontrado por el camino algunos errores personales, y a causa de ello he perdido bastante tiempo he concluir este objetivo. Su tiempo total realizado fue alrededor de 7 horas.

5. Mostrar varios mensajes de ganado o perdido

Este punto lo he concluido rápido, ha sido bastante sencillo realizar contenido de varios mensajes y hacer uso de ellos cuando ganaba o perdía el jugador. Su tiempo total realizado fue alrededor de 2 horas y 38 minutos.

Objetivos secundarios

6. Mostrar menú desplegable

El menú no me ha llevado tiempo a penas, el propio ionic me lo genera e implementa en la aplicación de manera que no me ha llevado tiempo en absoluto, ha sido casi instantáneo. Su tiempo total realizado fue de 10 minutos como máximo.

7. Botón de Jugar

Entre estilos que se le han aplicado y hacerlo funcional, ha sido no más de 2 horas. Me ha llevado más tiempo el hecho de agregarle unos efectos que le dan un mejor apartado de diseño al botón, y a la aplicación sobretodo. Su tiempo total realizado fue de 1 hora y 56 minutos.

8. Botón de Cambiar nombre de usuario

Me ha llevado el mismo tiempo que el botón de jugar, su tiempo se vio alargado por la dificultad que fue guardar diferentes usuarios en el almacenamiento local. Su tiempo total realizado fue 2 horas y 34 minutos.

9. Almacenamiento del progreso en local

No hace falta hablar mucho sobre este punto, ya sabéis lo que pudo haberme llevado concluirlo, su forma de establecer unos valores y devolverlos tal cual los generamos suena fácil de hacer, pero a la hora de la verdad, puede llevar tiempo. A mí me llevó mucho tiempo en corregir el problema que me devolvía la información que se guardaba anteriormente, y no en su momento en el que se generó, pero, como todo código siempre hay una solución para resolver los problemas. Su tiempo total realizado fue de 5 horas y 27 minutos.

10. Efectos de luz/brillo para cada botón

Respecto a este punto, fue simplemente un detalle que le ha venido muy bien a la aplicación, su resultado como podéis ver en la imagen, está bien conseguido en mi opinión. Le otorga un diseño visual más agradable al simon. Su tiempo total realizado fue de 1 hora.

-- FOTO --

11. Tutorial/Guía del juego

Qué sería de una aplicación sin una guía, pues que los usuarios sin una orientación pierden el interés por una página, tienda, etc. Y estas acaban por abandonar la página. Pues este punto se ha tratado sobre eso, para orientar a nuevos usuarios a entender cómo funciona la aplicación o cuál es su finalidad. Su tiempo total realizado fue de 1 hora y 25 minutos.

12. Mostrar información breve sobre el juego y desarrollador

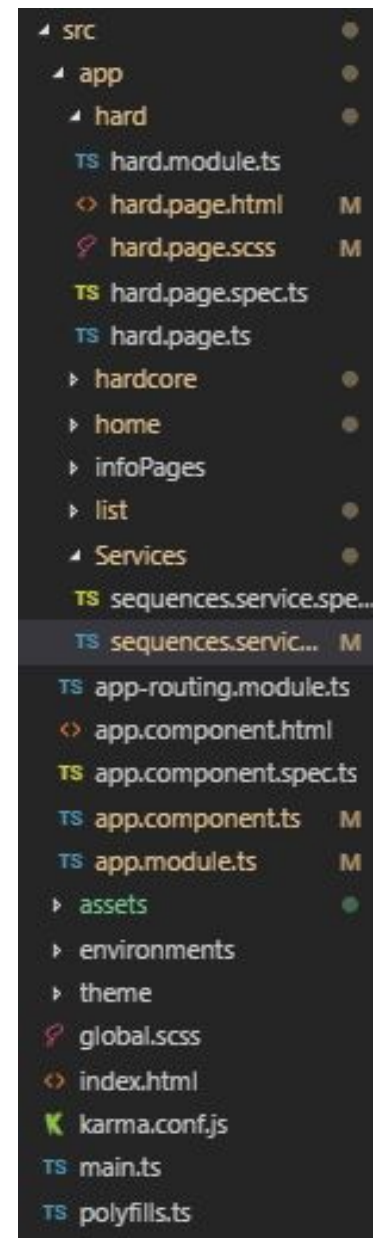
Este punto trata lo mismo que el punto anterior, pero aquí muestra quién ha sido sus desarrolladores y qué elementos o funciones se le aplicaron al juego, es una información muy breve. Su tiempo total realizado fue de 45 minutos.

4.4. FASE 4: FASE DE PRODUCCIÓN

La parte de programación es un punto muy a tener en cuenta para todos los proyectos, cada línea de código que se añade, cada nueva variable, constante o función que se aplican, todo tiene la mayor prioridad que se le puede dar a nuestro proyecto, porque el tiempo que se gasta en aprender nuevos elementos, funciones para añadir a nuestro proyecto es o puede ser infinito, pero, no siempre tenemos la capacidad o tiempo de hacer todo esto posible.

El lenguaje que he utilizado para el desarrollo de la aplicación ha sido TypeScript, un lenguaje basado en el JavaScript, su extensión de ficheros es “.ts”.

En esta fase muestro mi parte de programación detallada, cómo lo he estructurado todo y he realizado su lógica, el directorio “**app**” guarda subcarpetas de las dificultades, servicios, vistas, controladores y modelos que esto lo estructura junto a los ficheros de estilos para las vistas, como ficheros de test para mejorar el funcionamiento de la aplicación, todo esto lo he realizado de la siguiente manera:



4.4.1. CONTROLADORES

Maneja la comunicación con las vistas de la aplicación, está mantienen un flujo para hacer funcionar el juego. Este responde a los eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos).

```
ngOnInit() {  
  this.name = window.localStorage.getItem('username');  
  if (this.name != null && this.name != undefined) {  
    this.getNickName();  
  } else {  
    window.localStorage.clear();  
  }  
}  
  
getNickName() {  
  let name = window.localStorage.getItem('username');  
  if (name != null && name != undefined) {  
    document.getElementById('user').innerHTML=`Jugador: ${name}`;  
  }  
  document.getElementById('start').style.display = "block";  
}  
  
pageUser(): void {  
  navigator.vibrate(200);  
  this.nav.navigateRoot("/username");  
}  
  
startGame(id):void {  
  navigator.vibrate(200);  
  this.contador = 1;  
  this.puntuacion = 0;  
  document.getElementById('start').style.display = "none";  
  document.getElementById('content').style.display = "block";  
  document.getElementById('contador').innerHTML=`Nivel ${this.contador} `;  
  document.getElementById('score').innerHTML=`Puntuación: ${this.puntuacion} `;  
  this.utils.startGame(id);  
}  
  
jugadaPlayer(id) {  
  this.utils.jugadaPlayerHardcore(id);  
}
```

También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta de 'modelo', por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo' que esto permite la comunicación entre ambas de manera que haga funcionar la aplicación.

4.4.2. SERVICIOS

Aquí se almacena toda la lógica, funcionalidad de la aplicación que sigue desde las secuencias que forma la máquina, hasta los mensajes que muestran al ganar o perder intentando copiar la secuencia.

Par empezar con un servicio es importante declarar todas las variables que vayamos a utilizar en el juego.

```
computerSeq: Array<number>; // Secuencia jugada por la máquina
playerSeq: Array<number>; // Secuencia jugada por el jugador
computerCurrentButton: number; // Secuencia de botones/colores jugada por la máquina
playerCurrentButton: number; // Secuencia de botones/colores jugada por el jugador
medium: number; // Modo normal de 2 en 2
hard: number; // Modo difícil de 3 en 3
time: number; // Tiempo entre secuencia
messageLose: Array<string>;
messageWin: Array<string>;
contador: number;
ranking: Array<string>;
puntuacion: number;
name: string;

startGame(id): void { // Inicializa y establece los datos del juego
    this.playerSeq = []; // Secuencia del Jugador
    this.computerSeq = []; // Secuencia de la máquina
    this.computerCurrentButton = 0; // Inicializador de la secuencia
    this.playerCurrentButton = 0; // Inicializador de la secuencia
    this.medium = 2;
    this.hard = 3;
    this.time = 2000;
    this.contador = 2;
    this.puntuacion = 100;
    this.ranking = [];
    this.messageLose = ['Has Perdido', 'Prueba otra vez', 'Ups vaya...', 'Casi no fallas'];
    this.messageWin = ['Has Ganado', 'Buen Trabajo', 'Sigue así', 'Estas On Fire'];

    switch(id) {
        case 1:
            this.anadirJugadaFacil(this.time);
            break;
        case 2:
            this.anadirJugadaMedio(this.time);
            break;
    }
}
```

Nuevas variables se declaran indicando el tipo primero que seguidamente las inicializamos al empezar a jugar, por eso llamo a la función “**startGame**” y le paso un identificador que se refiere a la dificultad elegida para su posterior uso en las funciones que deban estar presente. He declarado array, string, number, etc.

Una vez elegimos el modo, este ejecuta una jugada que se identificará a la dificultad que el usuario haya elegido, desde fácil hasta hardcore.

```
anadirJugadaFacil(time):void {
  this.computerSeq.push(this.getRandom());
  this.ejecutarJugada(time);
}

ejecutarJugada(time):void {
  setTimeout(() => {
    if (this.computerSeq[this.computerCurrentButton] == 0) {
      this.playSonido(0);
      document.getElementById('button-red').classList.add('active-red')
      setTimeout(this.clickOut, 1000)
    } else if (this.computerSeq[this.computerCurrentButton] == 1) {
      this.playSonido(1);
      document.getElementById('button-green').classList.add('active-green')
      setTimeout(this.clickOut, 1000)
    } else if (this.computerSeq[this.computerCurrentButton] == 2) {
      this.playSonido(2);
      document.getElementById('button-blue').classList.add('active-blue')
      setTimeout(this.clickOut, 1000)
    } else {
      this.playSonido(3);
      document.getElementById('button-yellow').classList.add('active-yellow')
      setTimeout(this.clickOut, 1000)
    }
    if (this.computerSeq.length > this.computerCurrentButton + 1) {
      ++this.computerCurrentButton;
      this.ejecutarJugada(time)
    }
  }, 2000);
}

playSonido(id) { // Maneja el sonido que suena cuándo se pulsa el botón indicado
  let sonido = (<HTMLAudioElement>document.getElementById('sonido' + id));
  if (sonido) {
```

Ejecutar una jugada es fácil si lo pensamos, pero en el desarrollo es más complicado, todo el seguimiento que realiza la máquina para realizar una y otra vez la misma secuencia que el jugador debe memorizar y acertar en el mismo orden que se realizó la primera vez.

Toda aplicación guarda en local datos del juego para que no podamos reiniciar el estado una y otra vez al iniciar la aplicación, de esta manera se mejora la interactividad y el entretenimiento. Lo que hago es establecer nuevas variables que guardan esta información una vez acabemos la partida o guardemos un nuevo nombre de usuario en nuestra aplicación.

```
nuevoRanking(puntuacion, name): number {
  let puntos: number;
  if (this.ranking.length > 5) {
    this.ranking.pop();
    puntos = this.ranking.push(puntuacion, name);
    return puntos;
  } else {
    puntos = this.ranking.push(puntuacion, name);
    return puntos;
  }
}

jugadaPlayer(id): void {
  this.playSonido(id);
  if (id === this.computerSeq[this.playerCurrentButton]) {
    if (this.playerCurrentButton === this.computerSeq.length-1) {
      this.playerCurrentButton = 0;
      this.computerCurrentButton = 0; // Borrar inicialización para modo Hardcore xD
      this.messageWinner(this.messageWin, this.contador, this.puntuacion); // Muestra el mensaje de winner
      this.contador = (this.contador + 1);
      this.puntuacion = (this.puntuacion + 100);
      this.time = (this.time - 100);
      setInterval(this.removeMessage, 2000); // Borra el mensaje pasado unos segundos
      this.anadirJugadaFacil(this.time); // Llama a la función que añade una jugada más a la secuencia de la máquina
    } else {
      ++this.playerCurrentButton;
    }
  } else { // Muestra el mensaje loser y recarga la página
    this.messageLoser(this.messageLose);
    this.name = window.localStorage.getItem('username');
    window.localStorage.setItem('ranking', JSON.stringify(this.nuevoRanking(this.puntuacion, this.name)));
    setInterval(this.reload, 1500);
  }
}
```

Para recibir y mostrar estos datos, solo debo recogerlo y mostrarlo cuando el usuario quiere verlo.

Es verdad que la máquina ejecuta una secuencia para que el usuario la memorice y este la copie exactamente en el mismo orden que fue ejecutada, pero cuando se trata de aplicar este desarrollo al funcionamiento de la aplicación, es mucho más complicado que el primero. La máquina solo ejecuta una secuencia aleatoria de colores, pero, el jugador tiene que copiar esto y luego comprobar su correcta ejecución que de esa manera, si decimos que fue correcto el orden, un mensaje lo notifica en verde que la partida sigue con un nuevo nivel y un mensaje establecido de forma aleatoria, por el contrario si el mensaje lo notifica de color rojo, la aplicación se reinicia y almacena la puntuación en local.

El tema de los mensajes como anteriormente he citado textualmente, hay que saber cuándo un mensaje debe notificarse al usuario de que ha ganado o perdido la partida y diferenciar qué mensaje es bueno y malo.

```
messageLoser(message):string { // Muestra el mensaje de "Has Perdido"
  let randomMessage = this.getRandom();
  switch(randomMessage) {
    case 0:
      document.getElementById('mensajeG').innerHTML="";
      return document.getElementById('mensajeP').innerHTML=message[randomMessage];
    case 1:
      document.getElementById('mensajeG').innerHTML="";
      return document.getElementById('mensajeP').innerHTML=message[randomMessage];
    case 2:
      document.getElementById('mensajeG').innerHTML="";
      return document.getElementById('mensajeP').innerHTML=message[randomMessage];
    case 3:
      document.getElementById('mensajeG').innerHTML="|";
      return document.getElementById('mensajeP').innerHTML=message[randomMessage];
  }
}

messageWinner(message, contador, puntuacion):string { // Muestra el mensaje de "Has Ganado"
  let randomMessage = this.getRandom();
  document.getElementById('contador').innerHTML=`Nivel ${contador}`;
  document.getElementById('score').innerHTML=`Puntuación: ${puntuacion}`;
  switch(randomMessage) {
    case 0:
      document.getElementById('mensajeP').innerHTML="";
      return document.getElementById('mensajeG').innerHTML=message[randomMessage];
    case 1:
      document.getElementById('mensajeP').innerHTML="";
      return document.getElementById('mensajeG').innerHTML=message[randomMessage];
    case 2:
      document.getElementById('mensajeP').innerHTML="";
      return document.getElementById('mensajeG').innerHTML=message[randomMessage];
    case 3:
      document.getElementById('mensajeP').innerHTML="";
      return document.getElementById('mensajeG').innerHTML=message[randomMessage];
  }
}
```

Los mensajes se muestran en color verde que notifican que se ha superado la secuencia con éxito, mientras que los mensajes en color rojo notifican que la secuencia ha sido un fracaso y sin esperas el juego es reiniciado.

4.4.3. MODELOS

Representa la información de manera que el sistema opere, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación. Envía a la 'vista' o 'vistas' aquella parte de la información que en cada momento se le solicita para que sea mostrada al usuario. Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: 'home',
    loadChildren: './home/home.module#HomePageModule'
  },
  {
    path: 'list',
    loadChildren: './list/list.module#ListPageModule'
  },
  { path: 'hard', loadChildren: './hard/hard.module#HardPageModule' },
  { path: 'hardcore', loadChildren: './hardcore/hardcore.module#HardcorePageModule' },
  { path: 'username', loadChildren: './infoPages/username/username.module#UsernamePageModule' },
  { path: 'credit', loadChildren: './infoPages/credit/credit.module#CreditPageModule' },
  { path: 'tutorial', loadChildren: './infoPages/tutorial/tutorial.module#TutorialPageModule' },
  { path: 'ranking', loadChildren: './infoPages/ranking/ranking.module#RankingPageModule' }
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ]
})
```

Está llena de funciones que permite la navegación a través de la aplicación, este tiene asignado una página de bienvenida que es inicializada como página principal (root) una vez abrimos la aplicación.

4.4.4. VISTAS

Parte visual de la aplicación dónde se presentan diseños que más nos parece agradables, frescos, etc.

```
<ion-header>
  <ion-toolbar color="success" class="ion-text-center">
    <ion-buttons slot="start">
      <ion-menu-button></ion-menu-button>
    </ion-buttons>
    <ion-title>
      SIMON GAME
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div id="start">
    <div id="image">
      
    </div>
    <p id="user"></p>
    <ion-buttons>
      <ion-button class="bubbly-button" (click)="startGame(1)">JUGAR</ion-button>
    </ion-buttons>
    <ion-buttons>
      <ion-button id="butusChange" routerLink="/username" class="bubbly-button" (click)="pageUser()">CAMBIAR USUARIO</ion-button>
    </ion-buttons>
  </div>

  <div id="score"></div>

  <ion-grid id="content">
    <ion-row>
      <ion-col size="6">
        <ion-card id="button-green" color="success" (click)="jugadaPlayer(1)">
        </ion-card>
      </ion-col>
      <ion-col size="6">
        <ion-card id="button-red" color="danger" (click)="jugadaPlayer(0)">
        </ion-card>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

Presenta el 'modelo' (información y *lógica*) en un formato adecuado para la interacción del usuario, de manera que la información se represente como salida.

4.4.5. IMPLEMENTACIÓN

La aplicación puede tener muchas o inmensas implementaciones a lo largo del desarrollo de este que hacen para bien, mucho más funcional con más contenido jugable e interactivo.

Para Simon, he aplicado pequeñas implementaciones que han hecho más interesante y agradable al juego.

4.4.5.1. SONIDO

El sistema de sonido es primordial para un juego, y si hablamos de un juego en el que su parte de jugabilidad se debe la mayor parte al sonido, pues, con más razón para incluirlo. He optado por darle un toque retro al juego, unos sonidos de calidad 8 bits que en aquellos años cautivaron a más de una persona.

```
muteSonido() {  
  (<HTMLAudioElement>document.getElementById('sonido0')).volume = 0;  
  (<HTMLAudioElement>document.getElementById('sonido1')).volume = 0;  
  (<HTMLAudioElement>document.getElementById('sonido2')).volume = 0;  
  (<HTMLAudioElement>document.getElementById('sonido3')).volume = 0;  
}  
  
unmuteSonido() {  
  (<HTMLAudioElement>document.getElementById('sonido0')).volume = 1;  
  (<HTMLAudioElement>document.getElementById('sonido1')).volume = 1;  
  (<HTMLAudioElement>document.getElementById('sonido2')).volume = 1;  
  (<HTMLAudioElement>document.getElementById('sonido3')).volume = 1;  
}
```

Son pistas de al menos un segundo que reciben bien en el juego cuando el jugador o máquina pulsa uno de los botones, y por supuesto, cada sonido es distinto. Los tonos varían entre agudos y graves para notar la diferencia entre estos cuando la máquina pulse un botón.

4.4.5.2. VIBRACIÓN

```
playSonido(id) { // Maneja el sonido que suena cuándo se pulsa el botón indicado  
  let sonido = (<HTMLAudioElement>document.getElementById('sonido' + id));  
  if (sonido) {  
    sonido.pause();  
  }  
  navigator.vibrate(300);  
  sonido.play();  
}
```

Por otra parte, he incluido un sistema de vibración que este se activa cuando pulsas un botón, solo se activa una pequeña cantidad lo más rápido para notar que se ha pulsado un botón.

```
pageUser(): void {  
  navigator.vibrate(200);  
  this.nav.navigateRoot("/username");  
}  
  
startGame(id):void {  
  navigator.vibrate(200);  
  this.contador = 1;  
  this.puntuacion = 0;  
  document.getElementById('start').style.display = "none";  
  document.getElementById('content').style.display = "block";  
  document.getElementById('contador').innerHTML = `Nivel ${this.contador}`;  
  document.getElementById('score').innerHTML = `Puntuación: ${this.puntuacion}`;  
  this.utils.startGame(id);  
}
```

4.4.5.3. MENÚ

```
<ion-header>
  <ion-toolbar color="success">
    <ion-title>PANEL DE CONTROL</ion-title>
  </ion-toolbar>
</ion-header>
<ion-content>
  <ion-list>
    <ion-list-header color="danger">
      <ion-label>CONFIGURACIÓN</ion-label>
    </ion-list-header>
    <ion-menu-toggle *ngFor="let s of sound">
      <ion-item>
        <ion-icon slot="start" [name]="s.icon"></ion-icon>
        <ion-label>
          Sonido
        </ion-label>
        <ion-toggle id="audio" (click)="changeSound()"></ion-toggle>
      </ion-item>
    </ion-menu-toggle>
    <ion-list-header color="danger">
      <ion-label>DIFICULTAD</ion-label>
    </ion-list-header>
    <ion-menu-toggle auto-hide="false" *ngFor="let p of difficulties">
      <ion-item [routerDirection]="root" [routerLink]="[p.url]">
        <ion-icon slot="start" [name]="p.icon"></ion-icon>
        <ion-label>
          {{p.title}}
        </ion-label>
      </ion-item>
    </ion-menu-toggle>
  </ion-list>
</ion-content>
```

Implemente un menú deslizante en la parte izquierda de la aplicación para poder configurar de la mejor manera posible la navegación a través de ella.

```
const routes: Routes = [
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: 'home',
    loadChildren: './home/home.module#HomePageModule'
  },
  {
    path: 'list',
    loadChildren: './list/list.module#ListPageModule'
  },
  { path: 'hard', loadChildren: './hard/hard.module#HardPageModule' },
  { path: 'hardcore', loadChildren: './hardcore/hardcore.module#HardcorePageModule' },
  { path: 'username', loadChildren: './infoPages/username/username.module#UsernamePageModule' },
  { path: 'credit', loadChildren: './infoPages/credit/credit.module#CreditPageModule' },
  { path: 'tutorial', loadChildren: './infoPages/tutorial/tutorial.module#TutorialPageModule' },
  { path: 'ranking', loadChildren: './infoPages/ranking/ranking.module#RankingPageModule' }
];
```


Una vez abrimos el juego, podemos optar por mostrar/deslizar el menú a partir del cual escoges las diferentes dificultades del juego, páginas que nos informa un poco del juego y su creador, el ranking de puntuación, activación o desactivación del sonido y una guía de cómo se juega a simon o directamente optamos por jugar. Si nos fijamos en el menú puede darnos la sensación de tener un aspecto divertido.

4.5. FASE 5: FASE DE PRUEBAS

En esta fase se corrigen los errores del proceso de programación y se mejora la jugabilidad a medida que se prueba el juego. Todo desarrollo de una aplicación deben tener tests que comprueben los errores que deben ser rápidamente solucionados para que esto no vaya a más y sea casi imposible luego de solucionar.



Generalmente se realizan dos tipos de pruebas que están son: las pruebas alpha, y las pruebas beta. Las primeras tienen el objetivo de corregir defectos graves y mejorar características fundamentales no contempladas en la fase de diseño, mientras que las segundas se enfocan en detectar fallos menores y perfilar la experiencia de usuario.

4.6. FASE 6: FASE DE MARKETING

Después de habernos asegurado de que todo funciona correctamente y, tras los cambios realizados por el feedback ofrecido en las versiones Alpha y Beta, llega el turno de publicar los juegos. La distribución consiste, en mi caso al ser mobile, en publicar las aplicaciones en distintas tiendas virtuales. De forma paralela, y en especial durante los primeros meses de vida del juego, se lanzan las campañas de marketing ideadas y preparadas previamente.

La idea del márketing es para dar a conocer el videojuego y conseguir el mayor número de jugadores posibles. No tiene un orden concreto dentro del desarrollo, pues algunas empresas empiezan a hacer campaña de sus videojuegos meses e incluso años antes de publicarlos. La verdad es que depende de los recursos que los desarrolladores quieran destinar a promocionar la obra y no tiene porqué ser un departamento dentro de la propia empresa, sino que tanto la distribución como el márketing se pueden delegar a empresas externas especialistas en estas áreas.

4.7. FASE 7: FASE DE MANTENIMIENTO

Pese a que el juego esté finalizado y en las manos de los jugadores, su ciclo de vida aún está lejos de terminar. La fase de mantenimiento es el momento de arreglar nuevos errores, mejorarlo, etc. Ésto se hace sacando parches o actualizaciones al mercado.

Sin embargo es también una oportunidad para seguir sacándole partido. Ya sea en forma de microtransacciones, suscripciones de pago o incluso con expansiones completas que añaden nuevas características al videojuego sin modificar en profundidad el motor del mismo, digamos que sería más o menos como aprovechar al máximo la base inicial.

Pero no acaba todo con el lanzamiento, durante toda la vida del videojuego será necesario un mantenimiento y una serie de actualizaciones para mantener el producto al día de las últimas novedades de los lenguajes de programación y software.

5. ESTRUCTURA VISUAL DE LA APLICACIÓN

Si se quiere tener una perspectiva de cómo podría llegar a ser nuestra aplicación, siempre se debería tener en mente hacer unos bocetos de ella antes, proporciona un diseño que se puede asemejar al principal, simplemente utilizó este método que me proporciona una idea previa para la realización del diseño.

5.1. ESQUELETO DE LA APLICACIÓN

El esqueleto muestra cómo debería quedar la aplicación una vez terminado el desarrollo de la misma.

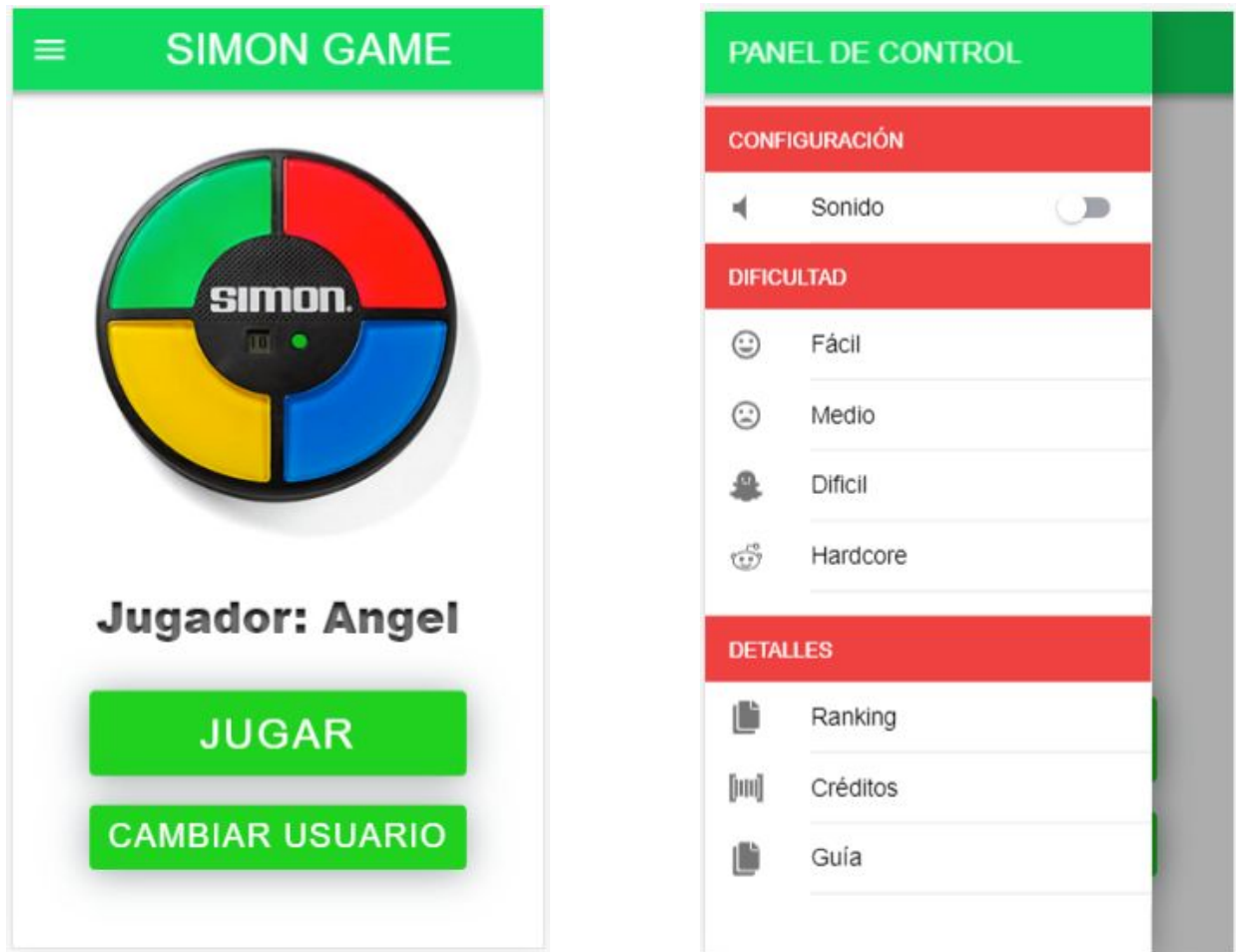


El primer diseño realizado, fue caótico para una aplicación que solo debía tener lo esencial, por eso tomé libertad de corregir varios problemas en contenido añadido, e ir probando a quitar cada elemento, imagen, que sobraba o que no hacía falta en la aplicación.

Finalmente el diseño estático que he utilizado ha sido el que veis en las imágenes, un diseño limpio.

5.2. DISEÑO FINAL

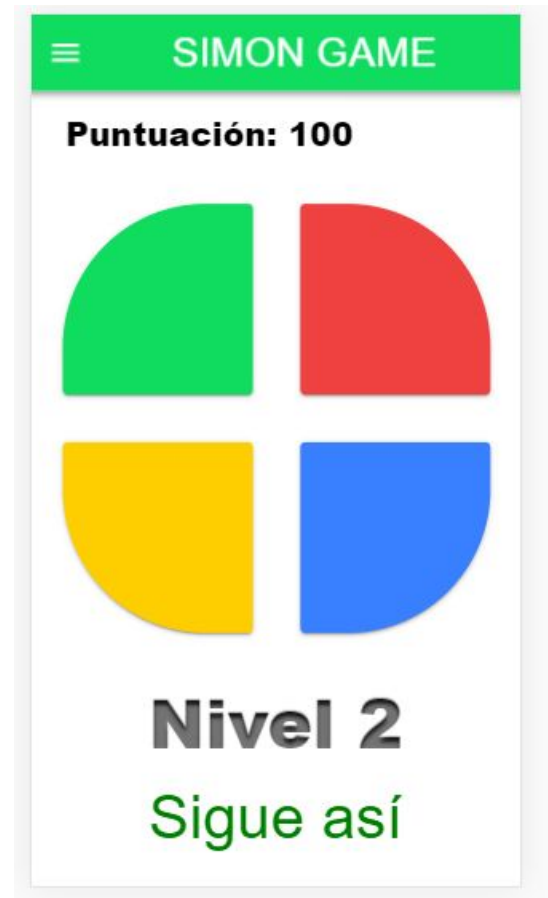
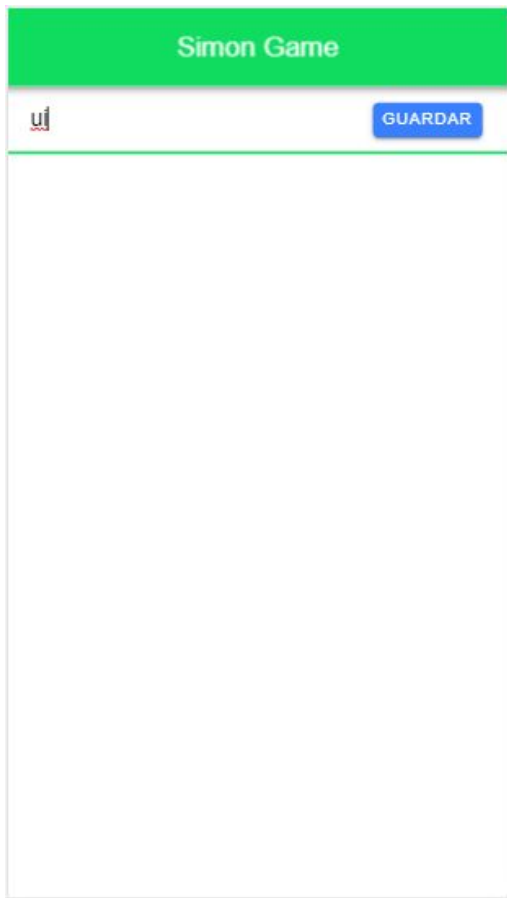
Si damos por hecho que una aplicación está finalizada, siempre hay contenido que mostrar para revelar a los usuarios como es su diseño. Para eso voy a mostrar en este punto varias fotos de la aplicación, llamados más concretamente “Mockups”.



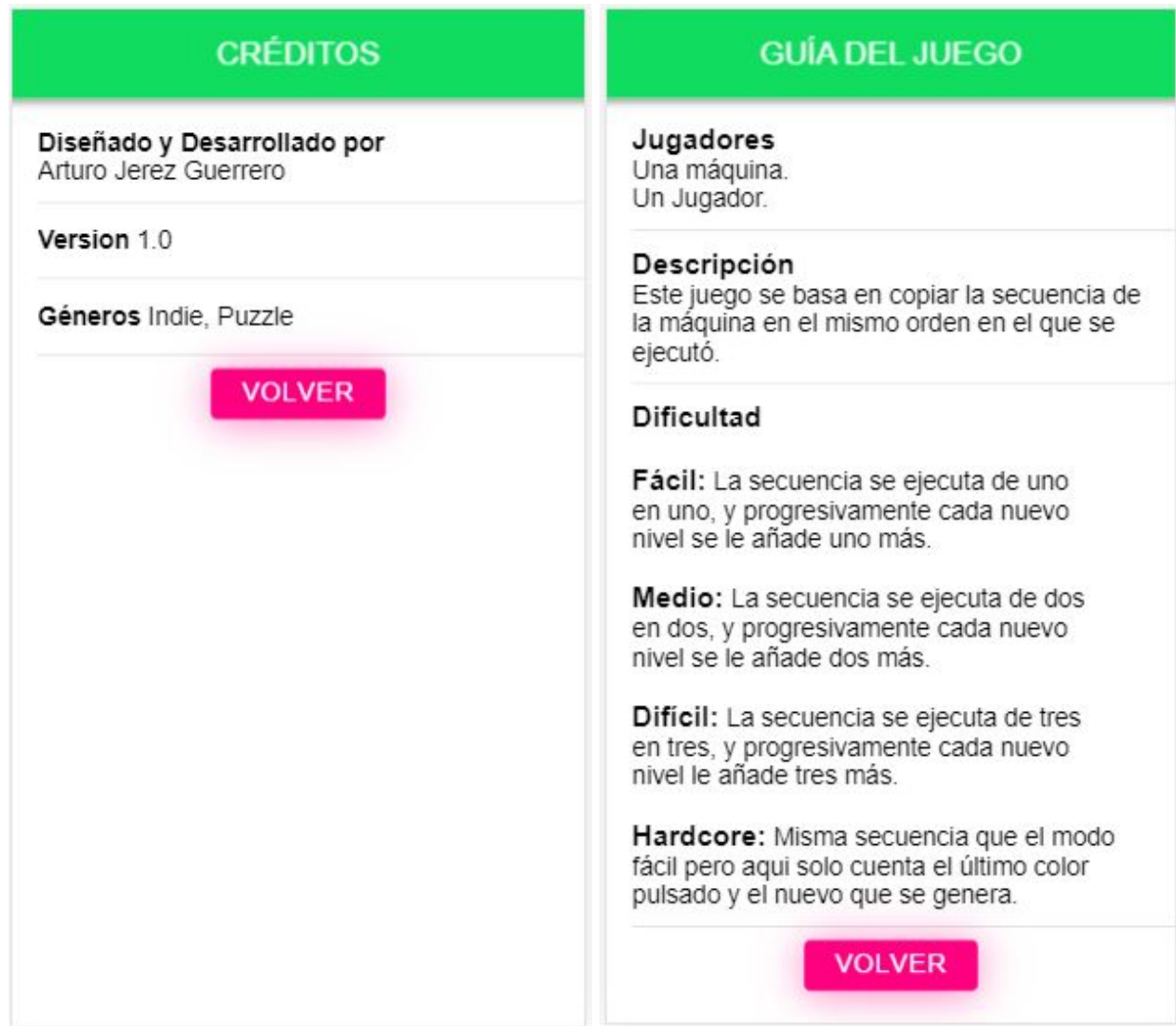
1. Menú principal que contiene tres botones: el menú deslizante en la cabecera al lado del título, jugar y cambiar usuario. También contiene una imagen del clásico juego de simon para hacer referencia a este y un texto que muestra el nombre de usuario que está actualmente jugando.
2. El menú deslizante o cómo yo lo llamo “panel de control”, está formado por casi el 95% de la funcionalidad de todo el juego. Para empezar tiene en la primera opción el control de sonido, se puede activar o desactivar.

Lo siguiente sería las diferentes dificultades de la aplicación que va desde fácil, medio, difícil hasta hardcore, por defecto al abrir la aplicación está siempre establecido la dificultad fácil.

Por último, todo el sector de detalles dónde se explica un poco la información del juego, su versión y creador. Una guía que nos ayuda a conocer y entender la jugabilidad de simon para todos los nuevos usuarios que nunca hayan jugado a él.



1. La primera imagen tiene lugar cambiar nombre del usuario a uno nuevo, está formado por un simple input y un botón para guardar los datos en local.
2. La segunda imagen visualiza cómo es el juego una vez iniciado, esta simplemente formado con cuatro botones de distintos colores y sonidos cada uno. Tiene una puntuación en la parte superior y un contador en la parte inferior, además de mensajes que notifican partida ganada o perdida.



1. Página de créditos donde nos muestra los datos del creador, del juego y la versión que puede modificarse mediante actualizaciones o parches que pueda tener a lo largo del ciclo de vida del juego.
2. Guía del juego que muestra un breve tutorial dónde nos facilita su entendimiento y funcionamiento para aquellas personas que nunca han tocado o jugado al simon, esta guía es recomendable si no queremos sufrir con cada partida o dificultad a la que nos enfrentemos.

6. TESTS DE PRUEBAS

Una aplicación debe estar bien provista de test unitarios para comprobar el buen uso del código de nuestro juego. Estos son realizados gracias a ficheros generados por ionic que integran fácilmente el uso de estos tests a la aplicación.

```
import { CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { RankingPage } from '../ranking.page';

describe('RankingPage', () => {
  let component: RankingPage;
  let fixture: ComponentFixture<RankingPage>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ RankingPage ],
      schemas: [CUSTOM_ELEMENTS_SCHEMA],
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(RankingPage);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

8. BIBLIOGRAFÍA

Toda aplicación contiene una cantidad de librerías, api's externas que se importan a lo largo del desarrollo de estas, incluyen nuevos aspectos, funcionalidad a la aplicación. Gracias a estas nos permiten añadir mucho más contenido a nuestras aplicaciones.

8.1. LIBROS Y VÍDEOS

He tenido diferentes medios por los que encontrar información y simplificar todo lo posible gracias a videos sobre la estructuración que debe seguir una aplicación desarrollada con ionic o cómo empezar a configurar mi aplicación. Estos son los enlaces de algunas rutas que me han ayudado en gran parte.

[CURSO-DESDE-CERO](#)

[IONIC-ANGULAR](#)

[ARQUITECTURA-IONIC](#)

8.2. MANUALES WEB

La mayor cantidad de manuales utilizados para la aplicación, ha sido la documentación del framework de ionic, está muy bien documentado y se encuentra fácil lo que se quiere buscar.

[INSTALACIÓN-DE-IONIC4](#)

[USO-DE-COMPONENTES](#)

8.3. FOROS/BLOGS

La forma más sencilla que me ha sacado de dudas durante el desarrollo de la aplicación, han sido los foros a los que he consultado y preguntado.

Búsquedas en foro de Stackoverflow:

<https://es.stackoverflow.com/questions/252222/c%C3%B3mo-puedo-generar-4-ion-card-con-un-color-diferente-cada-uno>

<https://es.stackoverflow.com/questions/254411/problema-al-cargar-mi-aplicacion-en-ionic-se-ralentiza-el-navegador>

7. ANÁLISIS Y VALORACIÓN DE ALTERNATIVAS

7.1. ANÁLISIS Y CONCLUSIONES

Al comienzo del proyecto se marcaron unos objetivos principales y secundarios que se deberían alcanzar si se quería dar el proyecto como finalizado. A continuación se analizarán estos objetivos con el fin de sacar conclusiones sobre la elaboración del proyecto.

7.1.1. OBJETIVOS

Los objetivos que se han cumplido son:

- A. Generación de secuencias
- B. Incluir distintos niveles de dificultad en el juego (Fácil, Medio, Difícil, Hardcore).
- C. Mostrar puntuación
- D. Incluir sonido
- E. Mostrar mensajes de ganado o perdido
- F. Mostrar menú desplegable
- G. Botón de Jugar.
- H. Botón de Cambiar nombre de usuario
- I. Almacenamiento del progreso en local
- J. Efectos de luz para cada botón
- K. Tutorial/Guía del juego
- L. Mostrar información breve sobre el juego y desarrollador

En mi caso, de los objetivos propuestos no ha habido ninguno que no haya conseguido.

7.1.2. CONCLUSIONES

Como se puede ver en el apartado anterior todos los objetivos que me propuse han sido cumplidos. Los objetivos principales se han cumplido por lo que hacen al juego funcional. Las posibles mejoras no se han realizado por lo que quedaría como línea futura.

Se ha sido capaz de desarrollar un juego que se ejecute en los dispositivos móviles, en android, en ios y en windows phone.

7.1.3. AUTOEVALUACIÓN

Este proyecto ha supuesto un reto personal y una gran satisfacción el haberlo conseguido. El principio fue duro ya que era algo totalmente desconocido para mi. Tuve que documentarme bastante sobre angularJS y ionic 4 ya que no tenía ideas relacionada con estos frameworks, además ionic se actualizó a una versión nueva poco después de iniciar el desarrollo con lo que todavía tenía problemas al realizar ejecuciones, tests o llamar a elementos del DOM.

Una vez que esta primera fase se ha superado, el hecho de programar la secuencias que la máquina debía generar y el usuario debía copiar ha resultado un poco complicado ya que tuve que cambiar la forma de pensar para programar algo que está constantemente ejecutándose en un bucle cada nueva secuencia. Gracias a los consejos de mi tutor de cómo podría implementarlo pude empezar a ver y entender cómo se programa de esa forma.

Cuando ya se conoce el funcionamiento de la aplicación, no es complicado de programar el juego, el lenguaje TypeScript es basado en JavaScript que es uno de los lenguajes que domino aunque le veo algunas deficiencias.

Desarrollando este proyecto me he dado cuenta, que he conocido un lenguaje de programación fácil de adaptar a mis conocimientos y a cualquier otro y que es más interesante el desarrollo para la realización de una aplicación web.

Debido a la dedicación parcial del proyecto, la línea temporal de realización se ha tenido que ir modificando para poder cumplir los objetivos en los plazos establecidos.

7.2. MEJORAS O ALTERNATIVAS PROPUESTAS

1. Para un futuro quiero añadir 2 jugadores a la partida para mejorar el entretenimiento y poder pasar un rato muy divertido con los amigos o familiares, esto me pareció buena idea implementarlo en un futuro.
2. Implementar cuenta de usuario mucho más versátil y funcional que permita almacenar nuestros datos, sin perder información o progreso de la aplicación, ya que actualmente utilizo un sistema poco seguro que solo guarda un único nombre de usuario que este se almacena en local, pero una vez sea modificado, se pierde todo el progreso.
3. Almacenamiento del progreso en la nube, para que todo usuario que tenga una cuenta pueda iniciar sesión en cualquier otro dispositivo, ya puede ser el de un amigo o familiar.

7.3. AGRADECIMIENTOS

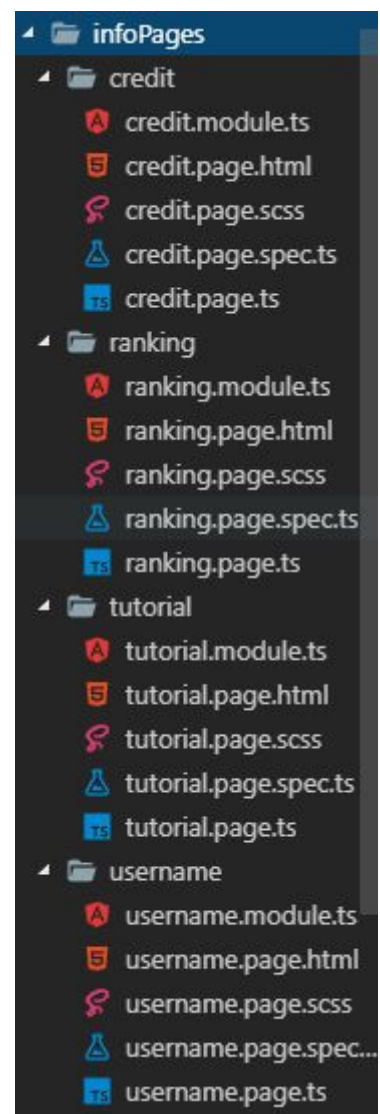
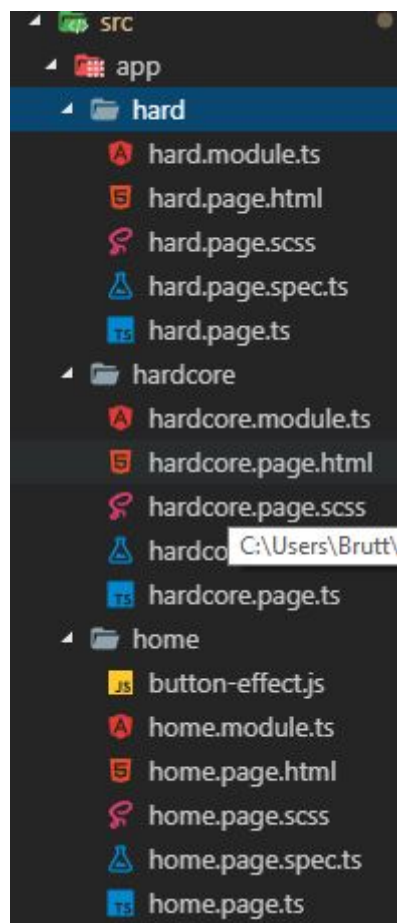
Llegados a este último punto del proyecto, me gustaría agradecer el esfuerzo y la dedicación de todos y cada uno de los profesores tanto de este curso como del anterior, por la motivación y los retos que me han planteado, ya que sin ellos no hubiera podido llegar hasta dónde he llegado. Tampoco quiero olvidarme de mis compañeros de clase que me han apoyado en todo momento y me han ayudado en determinados momentos del ciclo.

9. ANEXOS

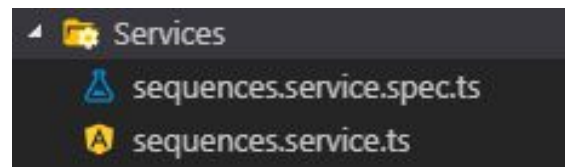
9.1. ESTRUCTURA DE CARPETAS

Siempre hay que tener una buena estructura de carpetas que define lo que hay en el contenido de nuestro código. Mi siguiente estructura de carpetas ha sido la siguiente:

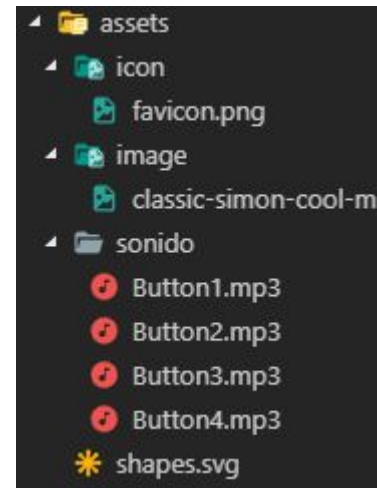
9.1.1. PÁGINAS



9.1.2. SERVICIOS

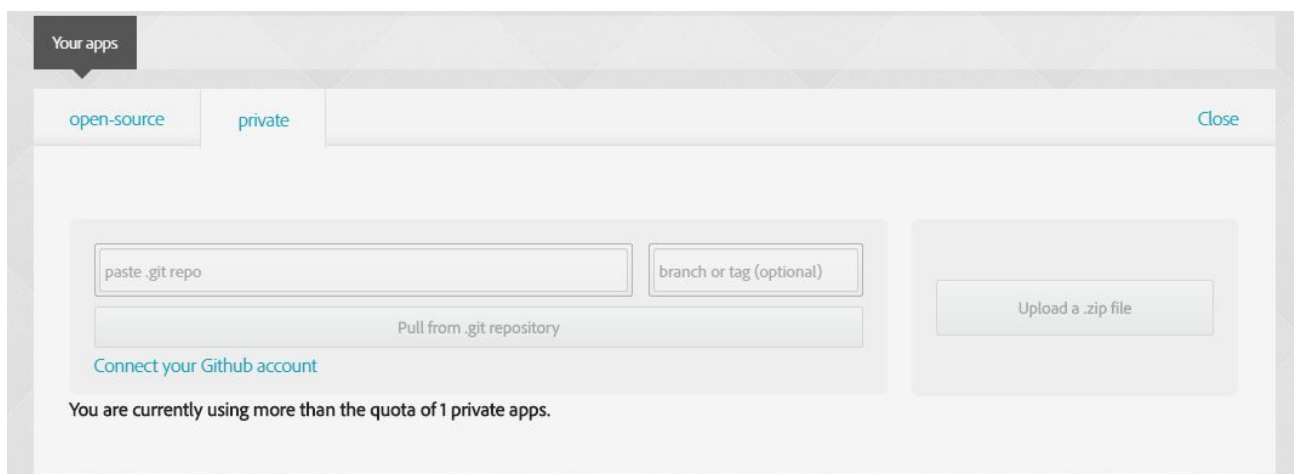


9.1.3. ASSETS

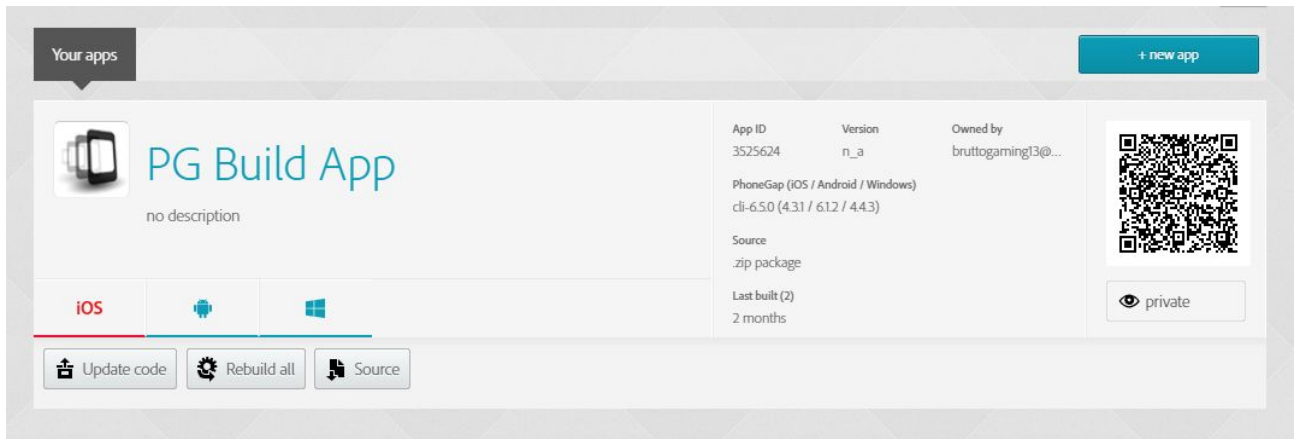


9.2. COMPILACIÓN PHONEGAP BUILD

Sin duda, las aplicaciones desarrolladas para dispositivos móviles es un gran privilegio tener a mano una herramienta que compila para estas plataformas. Gracias al framework de ionic es posible compilar la aplicación tanto para beta, como para producción.



Para beta quiere decir que la aplicación está en fase de prueba para encontrar algunos errores y corregirlos, mientras que para producción, la aplicación está lista para su uso colectivo en la plataforma global de red (Play Store) y/o plataforma mac (App Store).



Nuestra aplicación compilada lista para subir a phonegap dónde nos compila la aplicación para poder instalarla en nuestros dispositivos móviles (android, ios y web).