Universidad de Valladolid

Escuela Técnica Superior de Ingenieros de Telecomunicación

Grado en Ingeniería de Tecnologías de Telecomunicación

Desarrollo de una aplicación móvil para facilitar el aprendizaje de una lengua extranjera mediante ejercicios

Autor:
Borja Pérez Morán

Tutoras:
Dña. Míriam Antón Rodríguez
Dña. María Ángeles Pérez Juárez

Septiembre de 2017, Valladolid

TÍTULO	Desarrollo de una aplicación móvil para facilitar el aprendizaje de una lengua extranjera mediante ejercicios
AUTOR	Borja Pérez Morán
TUTORAS	Dña. Míriam Antón Rodríguez y Dña. Mª Ángeles Pérez Juárez
DEPARTAMENTO	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTA	Dña. Mª Ángeles Pérez Juárez
SECRETARIO	D. David González Ortega
VOCAL	D. Javier Aguiar Pérez
SUPLENTE	D. Mario Martínez Zarzuela
SUPLENTE	Dña. Míriam Antón Rodríguez

FECHA	6 de septiembre de 2017
CALIFICACIÓN	

Resumen

La enseñanza de una lengua extranjera requiere de materiales y recursos didácticos que motiven al alumno y le faciliten su aprendizaje. Para cualquier nativo digital, un recurso formativo que adopte un formato multimedia e interactivo y se le ofrezca a través de un canal integrado en su dinámica diaria como pueda ser su smartphone o tablet es susceptible de resultar atractivo, máxime si emplea un pensamiento y mecánica de jugabilidad (gamificación).

En este Trabajo Fin de Grado se pretende llevar a cabo el diseño y desarrollo de una aplicación híbrida que aproveche la versatilidad del desarrollo web, pero que tenga la capacidad de adaptación al dispositivo móvil usado para acceder a la misma como si de una app nativa se tratara. La aplicación propondrá de forma dinámica al usuario pequeños ejercicios, planteados de forma lúdica, basados por ejemplo en contenidos que el usuario esté visionando en un momento dado.

Keywords: aplicación, multiplataforma, idiomas, aprendizaje, multimedia.

Abstract

Learning a foreign language requires from tools who keep the student motivated and turn the learning easier. For millennials any multimedia resource in smartphones or tables, which are fully integrated in their daily routine, could be catchier. Even more if there is some gamification in it.

The main goal within this bachelor thesis is the design and development of a hybrid app. Taking advantage of the web technologies but working as if it is a native app. The app will challenge the user with some exercises while the user is watching some media content.

Keywords: app, multiplatform, languages, learning, multimedia.

A mi padre y madre, Carlos y Reden.
A mi hermana, María.

Todo lo que soy,
se lo debo a ellos.

Who let the words out? Who… Who…

Tabla de contenido

Índice de ilustraciones

Índice de tablas

1. Introducción

Hay 7.900 millones de smartphones en el mundo y ese número está creciendo exponencialmente. En España ya hay más dispositivos móviles que personas. Al desarrollar una aplicación móvil, esa cifra de dispositivos es el mercado al que te ofreces (Informe ditrendia, 2016). Pero no todos estos smartphones o tablets son homogéneos, pues existen diversos sistemas operativos. Incluso si solo nos centramos en iOS y Android, que en realidad prácticamente abarcarían todo el mercado, después nos encontramos con la posible escolla por las diferentes versiones, diferentes tamaños de pantalla, etc.

Las plataformas para el desarrollo de aplicaciones híbridas sortean las dificultades anteriores. La mayor parte de ellas se apoyan en las tecnologías web (HTML, CSS, JS), aunque también tienen desventajas frente a las soluciones específicas para cada plataforma y dispositivos.

Hay 7.500 millones de personas en el mundo y entre 3.000 y 5.000 idiomas de los cuales 600 cuentan con más de 100.000 hablantes (Elena Sanz, 2015). Ya en España, aparte del castellano, existen otras 3 lenguas cooficiales dependiendo de la región. El obstáculo comunicativo que existe cuando no se es capaz de hablar el mismo idioma entre personas es inmenso.

El aprendizaje de una o varias segundas lenguas que no sean la materna ya es un hábito arraigado en nuestra sociedad. Las ventajas que ofrece el conocimiento de idiomas son infinitas: mejora en la comunicación, posibilidad de viajar, abertura mayor del mercado laboral o percepción más amplia del mundo.

Actualmente se está viviendo una edad dorada en lo relativo a la producción de series de televisión y películas. La oferta generada entre ambas supera con creces la capacidad y tiempo que tiene una persona para poder visionar este contenido. Aun así, el auge de compañías de servicios VOD (video on demand) ha disparado el consumo de contenido multimedia en los últimos años, haciendo más accesible al público en general contenido con mayor calidad.

Contexto y Motivación

Observando estos datos con perspectiva, nos podemos hacer tres preguntas.

¿Por qué no desarrollar una app que llegue al mayor número de personas?

¿Por qué no fomentar el aprendizaje de idiomas con esa aplicación?

¿Por qué no intentar integrar el aprendizaje en el ocio diario?

Y otra más, ¿por qué he querido llevar a cabo este proyecto?

Siempre me he considerado un apasionado del cine y de las series. Es imposible contabilizar el número de películas que habré visto a lo largo de mi vida, de muchas no recuerdo ni el título ni la sinopsis. Y lo que tampoco recuerdo es el momento en el que empecé a consumir estos contenidos en versión original. Creo que hay una belleza extra –o más bien una esencia que se pierde en el doblaje- al ver una película o

serie en el idioma en que se rodó. Con seguridad, el mercado más fuerte en este negocio es el estadounidense y, por ende, cuando hablo de películas en versión original, posiblemente un 90% de las que veo estén en inglés. Dejando a un lado la cultura que pueden aportar ciertos contenidos multimedia, es un refuerzo increíblemente útil en el aprendizaje de un idioma.

Un idioma está en constante evolución y esto es enteramente reflejado en las series y películas. Aprendes vocabulario y aprendes a usarlo. Pero sobretodo acostumbras al oído a escuchar hablar en esa lengua.

Miro con admiración a los países que por el motivo que sea, no han doblado todo el contenido que emiten en la televisión. He tenido la oportunidad de comprobar el beneficio que supone exponer a una población infantil al visionado en versión original. Aprendes e interiorizas sonidos que no posee la lengua materna. Como anécdotas, es gracioso conocer gente de países balcánicos que aprendieron un poco de español gracias a "Los Serrano" o que compañeros en Portugal canten "Doraemon, el gato cósmico" en perfecto español, pues allí lo emitían doblado al castellano.

En definitiva, creo que a nivel personal me ha aportado mucho, sobre todo en cuestión de comprensión auditiva el haber visto series y películas en V.O.S (versión original subtitulada) y "lamento" que no me hayan forzado de pequeño a ver dibujos o jugar con las consolas en inglés, pues creo que los idiomas son un muro que con la edad es más difícil superar.

Que lo que importante es el contenido y no el idioma en que se presente.

Objetivos

El objetivo de este Trabajo Fin de Grado ha sido desarrollar un PMV (Producto Mínimo Viable) de una aplicación en la que llevaba pensando mucho tiempo. Una aplicación que dé respuesta a las preguntas del apartado anterior, teniendo en cuenta las limitaciones existentes, tanto de tiempo como de conocimientos. Que ofrezca algo novedoso y que el usuario lo crea útil y lo vuelva a usar.

Se ha estudiado el uso de un software que no suponga ninguna traba con respecto a cómo el usuario visualizaba contenido multimedia, que la experiencia de uso sea similar, pero con el valor añadido de estar sumergiéndote en otra lengua y forzando un poco más la atención al audio.

También se ha llevado a cabo un estudio de las tecnologías web y frameworks que se están usando actualmente en el mercado y con ellas se ha realizado este proyecto con el fin de que este trabajo no sirva únicamente como mero trabajo académico, sino que además aporte una visión sobre las soluciones que se emplean en el ámbito laboral.

He intentado adquirir nuevos conocimientos, tanto a nivel teórico como práctico. La experiencia que me ha aportado estudiar el Grado en Ingeniería de Tecnologías de Telecomunicación ha sido crucial, pues si bien no podemos aprender todo en la escuela, nos da la base necesaria para que los estudiantes tengamos la capacidad de aprender lo que nos propongamos relativo a los numerosos ámbitos y campos que abarca.

2. Estudio de las diferentes aproximaciones a las aplicaciones móviles

Introducción

En este capítulo abordaremos las ventajas y desventajas técnicas de las diferentes opciones entre las que podemos elegir para crear una aplicación móvil: nativas, web e híbridas.

Existen numerosos aspectos que hay que tener en cuenta para escoger la solución más óptima para nuestra aplicación móvil. Dependerá de cuál sea el propósito principal de la aplicación, si es necesario que se pueda ejecutar sin conexión a internet, el público al que va a ir dirigido, el presupuesto del proyecto, el tiempo de desarrollo, conocimientos necesarios, funcionalidades, diseño y rendimiento.

Ilustración 1: Diferentes aproximaciones al desarrollo móvil

Aplicaciones nativas

Las aplicaciones nativas están escritas usando las APIs (Application Program Interface) de bajo nivel que proveen las plataformas en las que se ejecutarán. Típicamente solo pueden ser ejecutadas en una única plataforma (iOS, Android, Windows Phone). El término nativo es usado para diferenciar este tipo de aplicaciones de las híbridas, como las que se desarrollan con Ionic.

Típicamente, el código nativo será la solución en la que más piensan los desarrolladores a la hora de crear una aplicación móvil. Para crear una app nativa, es necesario escribir en el lenguaje por defecto de la plataforma que se elija, Objetive-C o Swift para iOS, Java para Android, o C# o XAML para Windows.

Esta forma de afrontar el desarrollo posee fuertes ventajas sobre las otras. Las herramientas de desarrollo están integradas con la plataforma escogida. Tenemos a nuestra disposición IDEs (Integrated Development Environment) que están pensados de manera directa y casi exclusiva para el desarrollo en cada plataforma. Xcode para iOS y Android Studio para Android. Tampoco se necesita de software ni soluciones adicionales para acceder a las APIs nativas y a todas las funcionalidades disponibles.

Otra ventaja muy importante según que aplicación estemos llevando a cabo es que el rendimiento de esta será el mejor posible. No hay ninguna capa de código entre medias que afecte al rendimiento.

Pero no todo pueden ser ventajas. El primer inconveniente es que, al lanzar la app al mercado, posiblemente quieres que llegue al mayor número de personas y por ello se querrá lanzar tanto para iOS como para Android e incluso Windows. Llevar a cabo esto de manera nativa implica un nivel de destreza profesional en todos los diferentes lenguajes y APIs. No se podrá reusar el código en la parte del cliente y por tanto habrá que volver a escribirlo y posteriormente mantenerlo. Múltiples plataformas significan múltiples equipos de trabajo.

Aplicaciones web

Como ocurre con las aplicaciones híbridas, una aplicación web está escrita usando tecnologías web como HTML, CSS y JavaScript. Sin embargo, una aplicación web solo puede ser cargada en un navegador. También, aunque una aplicación web pueda tener un diseño responsivo y pueda ser usada en un navegador móvil, no podrá acceder a las funcionalidades nativas de los dispositivos como puedan ser la cámara, acelerómetro, lista de contactos, etc.…

La línea entre un sitio web móvil (diseño responsivo) y una aplicación web puede llegar a ser un poco difusa. Esta opción se reduce básicamente a crear una aplicación usando tecnologías web y entregársela al usuario mediante un navegador en su dispositivo.

Una ventaja clara de este formato es que puedes hacer llegar a todos los dispositivos y sistemas operativos la aplicación. Aligera los procesos de aceptación de la aplicación y actualizar, añadir nuevas funcionalidades o corregir bugs es muy sencillo, pues solo es necesario actualizar el contenido del servidor.

A la desventaja clara ya comentada de que al correr en un navegador existen unas limitaciones muy importantes a la hora del acceso a las capacidades el dispositivo, se le une que para los usuarios no es tan sencillo encontrar la aplicación, pues estos buscan en los markets de cada plataforma, mientras que poner una dirección URL en el navegador no es tan frecuente.

Aplicaciones híbridas

Una aplicación híbrida es una aplicación nativa que usa un navegador web sin marcos llamado WebView para ejecutar la aplicación web. Esta forma de desarrollo usa wrappers que se encargan de interactuar entre el dispositivo nativo y la WebView. La aproximación híbrida al desarrollo de apps aporta numerosas ventajas. Como ocurre con las aplicaciones web, la mayoría del código puede ser utilizado en varias plataformas. Solo se desarrolla en un lenguaje, por lo que mantener el código base es mucho más sencillo. La ventaja más importante sobre las aplicaciones web es que las híbridas sí que tienen acceso a las funcionalidades propias del dispositivo a través de un sistema de plugins u otro tipo de software.

Actualmente existen multitud de programadores web que no necesitarían aprender completamente un nuevo lenguaje para poder desarrollar apps híbridas.

Pero también existen serias desventajas. Como la aplicación no deja de ser una web app, está limitada a la capacidad que tenga el navegador en el dispositivo. Este rendimiento puede variar de forma muy amplia. Algunos dispositivos que sean más antiguos pueden tener rendimientos muy bajos, que no cumplan un mínimo ideal. La comunicación que se produce vía plugins introduce una nueva dependencia en el proyecto y no garantiza que la API vaya a estar disponible a través de este método. Tampoco se puede olvidar que los componentes de las interfaces nativas no estarán disponibles en la WebView. Toda la UI/UX (User Interface/User Experience) tendrá que ser escrita por el desarrollador.

En este proyecto se ha optado por el desarrollo híbrido, pues se ha visto una gran ventaja al uso de tecnologías web y a una posterior distribución multiplataforma de la aplicación. Las desventajas del rendimiento han sido determinantes, pues no es una aplicación que lo requiera en exceso. Se ha escogido el framework Ionic que usa esta aproximación y es que el equipo que hay detrás de Ionic ha tenido especial cuidado en recrear elementos UI para la WebView que se ven y sienten como sus respectivos elementos nativos. Por otra parte, Cordova nos ofrece su librería de plugins, que resuelve la falta de acceso a las funciones propias del dispositivo.

3. VLC media player

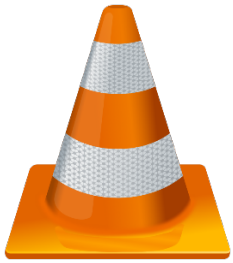


Ilustración 2: Logo de VLC media player

VLC es un reproductor multimedia libre y de código abierto multiplataforma y un framework que reproduce la mayoría de archivos multimedia, así como DVD, Audio CD, VCD y diversos protocolos de transmisión.

Los desarrolladores de VLC forman parte de la organización sin ánimo de lucro VideoLAN, con sede en Francia. Esta organización produce software multimedia gratuito que es lanzado con licencia GPL (General Public License). Nació como un proyecto de un estudiante en la École Centrale Paris, en 1996. Tras rescribirlo posteriormente a los dos años, se convirtió en open-source. A partir del 2009 se desligó completamente de esta escuela y este proyecto cuenta con desarrolladores de más de cuarenta países diferentes. Entre sus proyectos, aparte de VLC media player, podemos encontrar

- VLMC: VideoLAN Movie Creator es un software de edición de vídeo no lineal.
- DVBlast: es un simple pero poderoso demultiplexador MPEG-2/TS y aplicación de streaming.
- X264: es una aplicación gratuita para codificar streams de vídeo al formato H.264/MPEG-4 AVC.
- Multicat: es un conjunto de herramientas diseñadas para manipular con efectividad y facilidad *multicast streams*.

Una vez hemos descargado el programa, debemos configurar la interfaz web. Para ello, vamos a preferencias, luego a la pestaña de interfaces y en interfaces principales, activamos la interfaz web. Y dentro de la pestaña interfaces principales, abrimos en su submenú la pestaña Lua. Ahí escribiremos una contraseña. Esta contraseña la tendremos que escribir también en la vista de configuración de la app móvil.

Una vez que hayamos configurado la interfaz web de VLC, simplemente habrá que reiniciarlo y se habrán quedado guardadas ya nuestras preferencias.

Interfaz Web de VLC media player

El módulo http programado en Lua hace posible controlar VLC mediante una interfaz en el navegador o, por consiguiente, mediante conexiones http.

VLC montará un servicio web al que es posible acceder en la dirección IP local de nuestra red o en localhost/127.0.0.1 sí accedemos desde el mismo equipo desde el cual mandamos la petición.

http://localhost:8080/requests/status.json

Esa petición http nos devolverá el estado del VLC con datos relevantes sobre el fichero que se está reproduciendo en ese momento. A continuación se tenemos una muestra con los valores más relevantes para nuestro caso del fichero .json que obtenemos.


```
{
  "fullscreen":false,
  "audiodelay":0,
  "subtitledelay":0,
  "volume":230,
  "time":576,
  "length":6268,
  "state":"playing",
  "loop":false,
  "version":"2.2.4 Weatherwax",
  "information":{
    "chapter":0,
    "chapters":[0,1,2,3,4,5,6,7,8,9],
    "title":0,
    "category":{
      "meta":{
        "filename":"beginners.mkv"
      }
    }
  }
}
```


Por tanto, cuando realicemos esta petición podremos saber si el reproductor VLC está maximizado o no. Si existe retraso tanto en el audio como en los subtítulos. El volumen en que se encuentra el reproductor. El tiempo en segundos en el que se encuentra la reproducción y el tiempo total que dura. Si está pausado o en play y si se va a repetir una vez finalice. La versión del reproductor VLC. Por último, también ofrece información en caso de ser una serie, el capítulo que es y también el nombre del fichero multimedia que estamos reproduciendo.

Los variados comandos que tenemos a nuestra disposición se muestran en la siguiente tabla.

n_play&input=<uri>&option=<option>	pl_sort&id=<id>&val=<val>
in_enqueue&input=<uri>	pl_random
addsubtitle&val=<uri>	pl_loop
pl_play&id=<id>	pl_repeat
pl_pause&id=<id>	pl_sd&val=<val>
pl_forceresume	fullscreen
pl_forcepause	volume&val=<val>
pl_stop	seek&val=<val>
pl_next	preamp&val=<val in dB>
pl_previous	enableeq&val=<0 or 1>
pl_delete&id=<id>	setpreset&val=<presetid>
pl_empty	title&val=<val>
audiodelay&val=<delayinseconds>	audio_track&val=<val>
subdelay&val=<delayinseconds>	video_track&val=<val>
aspectratio&val=<newratio>	subtitle_track&val=<val>

Tabla 1: Comandos para la interfaz web VLC

4. OpenSubtitles.org

Ilustración 3: Logo de OpenSubtitles.org

Introducción

OpenSubtitles.org es una base de datos con subtítulos en diferentes idiomas. Para la realización de este trabajo he escogido esta página web debido a la experiencia anterior como usuario frecuente de este sitio. Cuándo se realizan búsquedas web de ficheros .srt (SubRip Text) –cuándo se busca un subtítulo en Google- suele figurar entre los primeros resultados que devuelve el navegador y es que tiene almacenados más de cuatro millones de subtítulos y está respaldado por una comunidad muy activa.

El fichero más básico de subtítulos es el formato .srt. Está formado por cuatro partes, todas ellas texto:

1. Un número que indica que subtítulo es dentro de la secuencia.
2. El instante de tiempo en el que el subtítulo aparecerá en la pantalla y cuándo desaparecerá.
3. El subtítulo propiamente dicho.
4. Una línea en blanco indicando el inicio de un nuevo subtítulo.

La tercera parte del subtítulo, cuando se añade a un fichero .mkv (Matroska, contenedor multimedia open-source) es convertida a UTF-8. La segunda parte se usa como código de tiempo del bloque y su duración, pero nada más es usado. Aclarar que Matroska no es un formato de compresión de vídeo o audio (video codec), es más bien como un sobre o envoltorio para el que puede haber diferentes flujos de audio, vídeo y subtítulos, permitiendo al usuario almacenar una película entera en un único fichero. Un ejemplo de un fichero .srt sería:

```
1
00:02:17,440 --> 00:02:20,375
Senator, we're making
our final approach into Coruscant.


2
00:02:20,476 --> 00:02:22,501
Very good, Lieutenant.
```

No es necesario un registro previo para descargar subtítulos de manera puntual y posee una búsqueda avanzada con filtros como: género, puntuación, año, etc. Al realizar la búsqueda, devuelve subtítulos en los diferentes idiomas que esté disponible y permite buscar también varios subtítulos de manera simultánea.

Tiene la opción de que te lleguen los subtítulos al correo electrónico o por rss (rich site summary), el sitio web es intencionadamente fácil de parsear (puedes obtener todas las páginas parseando XML) y actúa a su vez de servidor para otros sitios web que proveen subtítulos. Están abiertos a propuestas para implementar nuevas funcionalidades.

Pero por encima de este resumen sobre lo que es OpenSubtitles y sus características más destacables está que ofrecen una API para interactuar con su contenido.

Desarrollar usando OpenSubtitles.org

Para poder desarrollar un software que haga uso de la API de OpenSubtitles debemos pedir una clave "UserAgent". Si no se dispone de un UserAgent válido, nos devolverá error.

Una vez registrados en el sitio web, procedemos a solicitarla enviando un correo al administrador del sitio:

```
Required info
=============
Your name: Borja Pérez Morán
Your registered OS username: borpermor
Contact mail: borpermor@gmail.com
Title of useragent: borpermor
Version of useragent: v1
Programming language: JavaScript
Approximate users of program: I don't know yet... (Bachelor thesis)
Opensource: yes
Upload feature: no
```

El uso de una API no crea visitas reales a una página web, por lo que tampoco genera beneficios que pueda provenir de la publicidad con la que muchas veces se sufraga el costo de los servidores. Por ello, desde la sección para desarrolladores, solicitan que se sigan ciertas prácticas para promover el sitio web y propagar su marca. La mayor parte de estas peticiones se limitan a simples links, logos e imágenes y diferentes menciones dentro del software que se desarrolle.

La API de OpenSubtitles.org soporta cualquier extensión de vídeo y cualquier tamaño de fichero (siempre que supere los 128 kb). Los más comunes son los siguientes:

```
*.3g2, , *.3gp, *.3gp2, *.3gpp, *.60d, *.ajp, *.asf, *.asx, *.avchd, *.avi,
*.bik, *.bix, *.box, *.cam, *.dat, *.divx, *.dmf, *.dv, *.dvr-ms, *.evo, *.flc,
*.fli, *.flic, *.flv, *.flx, *.gvi, *.gvp, *.h264, *.m1v, *.m2p, *.m2ts, *.m2v,
*.m4e, *.m4v, *.mjp, *.mjpeg, *.mjpg, *.mkv, *.moov, *.mov, *.movhd, *.movie,
*.movx, *.mp4, *.mpe, *.mpeg, *.mpg, *.mpv, *.mpv2, *.mxf, *.nsv, *.nut, *.ogg,
*.ogm, *.omf, *.ps, *.qt, *.ram, *.rm, *.rmvb, *.swf, *.ts, *.vfw, *.vid,
*.video, *.viv, *.vivo, *.vob, *.vro, *.wm, *.wmv, *.wmx, *.wrap, *.wvx, *.wx,
*.x264, *.xvid
```

En cuanto al formato de subtítulo soportado, son los siguientes:

```
*.srt, *.sub, *.smi, *.txt, *.ssa, *.ass, *.mpl
```

Hay que tener en cuenta el límite de tráfico que soporta, siendo el máximo 40 peticiones http cada 10 segundos por dirección IP. Se puede consultar las restantes peticiones para cumplir el límite en la cabecera http. También en la cabecera se puede establecer el idioma preferido para la respuesta en el campo "HTTP HEADER Accept-Language"

El código de la lengua en la que se encuentra escrito el subtítulo sigue el ISO639. Es un conjunto de estándares internacionales que lista una abreviación para el nombre del idioma. Se puede consultar la tabla entera en las notas bibliográficas.

ISO language name	639.1	ISO language name	639.1
Arabic	ar	Hindi	hi
Chinese	zh	Portuguese	pt
English	en	Russian	ru
French	fr	Spanish	es
German	de	Zulu	zu

Tabla 2: Algunas lenguas con su código ISO639.1

Una funcionalidad muy interesante que proporciona también esta API es el IMDB (Internet Movie Data Base) ID, del cual, se podrá mediante otras peticiones http obtener una vasta información sobre la película que se vaya a visualizar. Se ha estudiado su uso a la hora de buscar una imagen del vídeo para añadir interactividad y visibilidad a la aplicación móvil, pero debido a que la inmensa mayoría de los ficheros no aporta ese dato, no ha merecido la pena usarlo.

Conviene hacer un buen manejo de los códigos de estado que devuelve el protocolo http, pues no es lo mismo que nos avise de un fallo en el servidor (5xx), a que nos alerte de que hemos alcanzado el límite de descargas (407).

Protocolo OSDb

OSDb (OpenSubtitles Database)es un protocolo introducido por primera vez en la página web OpenSubtitles.org. Su propósito principal es eliminar en lo posible el "tedioso" proceso de buscar subtítulos en la web y conseguir que encontrar subtítulos para vídeos sea lo más rápido y simple posible.

El protocolo hace uso de una función hash, obtenida específicamente del fichero de vídeo en lugar de tener varios subtítulos que coinciden para un único título. Entonces al buscar el subtítulo, se obtendrán aquellos que encajan al máximo (según el criterio de otros usuarios) con el vídeo. En lugar de buscar el subtítulo en la web, este protocolo puede ser implementado en aplicaciones con una interfaz de usuario más "amigable", convirtiendo esta búsqueda en un proceso rápido para la descarga de subtítulos.

¿Cómo funciona la descarga de subtítulos?

Una aplicación que use el protocolo OSDb seguirá típicamente estos pasos:

1. Seleccionar el fichero de vídeo del que se quiere obtener el subtítulo.

2. La aplicación calcula el hash único para este fichero.

3. La aplicación contacta remotamente con el servidor OSDb y hace la petición de los subtítulos que coincidan con el hash específico.

4. La aplicación muestra la lista de los subtítulos que han coincidido. Esto incluye su fecha de creación, su valoración, etc.

5. El usuario decide el subtítulo que desea usar y lo descarga.

Se puede seleccionar los idiomas en los que se tiene interés para evitar coincidencias con subtítulos en lenguas que no interesen al usuario.

¿Cómo funciona la subida de un subtítulo?

Este protocolo puede ser utilizado a su vez para añadir un subtítulo a la base de datos de OpenSubtitles.org.

Los pasos a seguir serían los siguientes:

1. Seleccionar los ficheros de vídeo y de subtítulo.

2. La aplicación calcula el md5 para el subtítulo y el hash para el vídeo y envía esta información al servidor OSDb.

3. Si ya existiese el mismo subtítulo, se añade el videohash a la base de datos, para que la próxima vez sea posible encontrar este subtítulo usando la aplicación.

4. El usuario rellena cierta información como la descripción, imdb, etc.

5. Se ha completado la subida del subtítulo y estará ya disponible para la descarga.

Node.js OpenSubtitles-API

Dentro del gestor NPM (Node Package Manager) se encuentra el paquete **opensubtitles-api**. Gracias a él, se tendrá la capacidad de descargar y subir subtítulos a OpenSubtitles.org mediante Node.js. Este módulo requiere un Agente de Usuario válido.

Aparte de poder usar todas las llamadas a los métodos de forma asíncrona, también permite directamente usar los siguientes métodos que han sido modificados como son:

- Search: función encadenada que retorna el subtítulo que más encaje al fichero multimedia basándose en la información proporcionada.

- Upload: función que únicamente requiere la ruta al vídeo y al fichero de subtítulo para subir un subtítulo nuevo a la base de datos de OpenSubtitles. El flujo será hacer Login, luego TryUploadSubtitles y por último UploadSubtitles.

- Hash: función que retorna el Hash y el tamaño de vídeo para un vídeo proporcionado.

- Md5: función que devuelve un hash para un subtítulo dado.

- Identify: función que devuelve metadatos basados en el hash del fichero de vídeo.

Para instalar el módulo tenemos que ir al directorio donde se encuentra el proyecto y desde la línea de comandos escribir:

```
npm install opensubtitles-api
```

Si se omite el usuario y la contraseña no todos los métodos estarán disponibles. Por defecto no usa SSL (Secure Sockets Layer) y si se utiliza otro **endpoint** para la comunicación, se sobrescribirá el que viene por defecto http://api.opensubtitles.org:80/xml-rpc.

Un ejemplo básico de uso de esta API sería:

```
const OS = require('opensubtitles-api');
const OpenSubtitles = new OS('UserAgent');

OpenSubtitles.api.LogIn('username', 'password', 'en', 'UserAgent').then...
```

Los métodos disponibles son:

LogIn	SearchMovieOnIMDB	GetUserInfo	AddRequest
LogOut	GetIMDBMovieDetails	TryUploadSubtitles	GetComments
SearchSubtitles	GuessMovieFromString	UploadSubtitles	AddComment
SearchToMail	ReportWrongMovieHash	DetectLanguage	AutoUpdate
CheckSubHash	ReportWrongImdbMovie	InsterMovie	SuggestMovie
CheckMovieHash	GetSubLanguages	SubtitlesVote	QuickSuggest
CheckMovieHash2	GetAvailableTranslations	SetSubscribeUrl	NoOperation
InsertMovieHash	GetTranslation	SubscribeToHash	ServerInfo

Tabla 3: Métodos disponibles en la API de OpenSubtitles

5. Servicios de traducción

Para ofrecer un servicio de diccionario en la aplicación, se ha realizado una búsqueda de los diferentes servicios disponibles.

WordReference API

Ilustración 4: Logo de Word Reference

La opción que en un primer momento pensé que se iba a adaptar mejor a las necesidades de la app era WordReference.com, que desde el año 2011 ofrece una API con la siguiente visión:

"Estamos felices de ofrecer una API para ser usada por desarrolladores y estamos expectantes por ver qué herramientas innovadoras y útiles pueden resultar. Para aquellas personas acostumbradas a Google Translate API, entended que esto es muy diferente. WordReference ofrece traducciones de diccionario para palabras y frases, no traducción de textos hechos por una máquina."

Obviando la subjetividad de su visión general, en mi opinión personal, habría preferido usar la API de WordReference, pero desafortunadamente han dejado de dar claves para nuevos desarrolladores debido al abusivo uso de la API.

Otra opción que barajé fue realizar peticiones http directamente a su página web y **parsear** posteriormente la página entera para obtener la traducción.

Google Translator API

Ilustración 5: Logo de Google Translator

Para hacer uso de esta API que proporciona Google a su servicio de traducción, debemos registrarnos y cumplimentar unos determinados pasos. Este servicio no es gratuito, pero Google ofrece una prueba de un año con ciertas limitaciones de tráfico en el proyecto.

En "Cloud Platform Console", tenemos que ir a la pestaña de proyectos y crear o seleccionar el proyecto en el que vayamos a usar los servicios de traducción. Posteriormente debemos de ir a habilitar la facturación en ese proyecto. Nos pedirán que vinculemos un método de pago. Una vez configurado el pago, el siguiente paso es habilitar "Cloud Translation API". Por último, tendremos que ir, dentro de nuestra consola con los proyectos de Google, a la zona de Credenciales, que nos indicarán la **key** que tenemos que usar para que las peticiones que hagamos con la API vayan con su correcta autentificación y nos sea posible realizarlas.

Una vez tengamos la clave para la autentificación, la manera en que se han hecho la petición a la API es la siguiente:

```
https://www.googleapis.com/language/translate/v2?key=xxXXxxXXxxXXxxXXxxXXxx
&target=es&source=en&q=movies
```

Los parámetros usados en la petición son los siguientes:

- Versión: la versión actualmente en uso de esta API es la v2.
- Key: es la clave de autentificación necesaria para poder realizar la petición. Es un identificador único y con el cuál miden el tráfico que se ha generado desde cada cuenta.
- Target: idioma al cual se traducirá el texto solicitado.
- Source: idioma en el que se encuentra el texto que se desea traducir. Google Translator API también ofrece un servicio con el cuál se detecta automáticamente el idioma de entrada. En este caso como se utiliza para traducir palabras sueltas y se conoce a priori el idioma en el que están escritas, nos ahorramos generar tráfico extra indicándolo directamente en la petición.
- Q: es el texto del que se desea obtener la traducción. Puede ser más de una palabra.

Y la respuesta que obtenemos de la API es la siguiente:

```
{
  "data": {
    "translations": [
      {
        "translatedText": "películas"
      }
    ]
  }
}
```

La API nos devuelve un fichero en formato JSON en el cual resulta sencillo acceder a la traducción proporcionada ya que será el valor que contiene la **key** "translatedText".

Otros servicios

Durante el estudio y búsqueda se han encontrado diferentes diccionarios que proveían su propia API para su uso. Pero, o bien estos eran de pago (Google también lo es, pero ofrece una versión de prueba de un año) o bien se centraba en dos idiomas únicamente, cuando lo que se pretende es construir una app para aprender diferentes y numerosos idiomas y no solamente uno.

6. Estudio y justificación de las tecnologías a usar

En este apartado se llevará a cabo un análisis de las tecnologías y lenguajes que mayoritariamente se han empleado en la realización de este proyecto.

Node.js

Ilustración 6: Logo de Node.JS

Node.js es un entorno en tiempo de ejecución que permite al desarrollador usar JavaScript en el lado del servidor. Fue construido bajo el motor V8 de Google Chrome. Node.js hace uso de un modelo dirigido a eventos que no bloquea los I/O. No solo se utiliza para servicios web, también es habitualmente usado para construir herramientas para desarrolladores (Ionic CLI, que veremos posteriormente es una de ellas). Npm es el gestor de paquetes más extendido dentro de Node.js.

La historia nos remonta a cuando Google distribuyó V8 como código abierto bajo licencia BSD (Berkeley Software Distribution). Esto animó a los programadores a testarlo y usarlo. Fue Ryan Dahl, a finales del 2009, quien trabajando con el motor V8 para crear un entorno JavaScript para el servidor, presentó Node.js a la comunidad.

Los principales motivos por los que trabajó con V8 de Google Chrome fueron:

- El motor V8 es muy rápido.
- V8 está especializado en la Web, por lo que es la herramienta perfecta para trabajar con http, dns y tcp.
- JavaScript es un lenguaje de programación muy conocido en la Web y fácilmente disponible para la mayoría de los desarrolladores.

Básicamente, Node.js es un entorno JavaScript especializado en la gestión de eventos desde el servidor. Es decir, podemos utilizar JavaScript para crear aplicaciones para el servidor del mismo modo que lo hacemos con lenguajes como PHP, Ruby o Python. Principalmente está centrado en las redes y se utiliza para crear software que interactúa con ellas.

Node.js ha supuesto un cambio de mentalidad muy importante, pues la mayoría de los desarrolladores estaban acostumbrados al uso de JavaScript para manipular e interactuar con el contenido de una página Web desde el navegador. A esta forma de trabajar se la conoce como JavaScript del lado del cliente, porque tiene lugar en el navegador del cliente. Sin embargo, con esta nueva forma de JavaScript en el lado del servidor, se ejecuta en el servidor antes de enviar la página al navegador.

Npm, el gestor de paquetes de Node.js

Ilustración 7: Logo de npm

Npm es un gestor de paquetes para Node.js. Gracias a él, podemos crear, compartir y reutilizar módulos en nuestras aplicaciones Node.js. También se puede utilizar para compartir aplicaciones Node.js completas. Los módulos son librerías de código que se pueden volver a utilizar en otros proyectos.

La mayoría de los módulos se publican bajo licencias de código abierto. Por regla general, se pueden instalar, modificar y distribuir libremente. Estos módulos presentan una oportunidad sensacional para los desarrolladores con poca experiencia porque les permiten empezar a trabajar basándonos en los conocimientos de otros programadores. El hecho de que podamos utilizar todos los módulos desde npm no implica que se hayan probado todos ni que sean estables. Una buena práctica es fijarse en el número de veces que se ha descargado un proyecto y en la cantidad de problemas que se han descubierto. Así tendremos una idea aproximada de su fiabilidad y madurez.

Una vez instalado npm podremos instalar los módulos desde el símbolo del sistema. Para que npm instale los módulos en el sitio adecuado, debemos encontrarnos en la carpeta del proyecto cuando ejecutemos el comando:

```
npm install [nombre_módulo]
```

Hay dos maneras para instalar módulos con este gestor:

1. La más recomendable es la instalación de manera local. También es la más común. Supone que la librería se instalará localmente en nuestro proyecto, concretamente en una carpeta llamada node_modules. Si hacemos uso del comando anterior, será el comportamiento por defecto.

2. La otra opción que existe es la denominada global. Existe la posibilidad de que algunos módulos cuenten con un ejecutable del que podamos hacer uso cuando trabajemos directamente desde el sistema de archivos. Uno de los módulos que utiliza este tipo de instalación es Express, que se trata de un framework Node.js de desarrollo web que incorpora un generador de estructuras para sitios web. Para instalar módulos de forma global bastará con añadir la etiqueta –g cuando los instalemos.

Para utilizarlos módulos desde las aplicaciones Node.js, tendremos que solicitarlos. Las solicitudes de los módulos tendrán la siguiente sintaxis:

```
var module = require('módulo');
```

HTML5 CSS3 JS

Estos tres lenguajes son los pilares que soportan el front-end (la capa de presentación o interfaz) del desarrollo web. El front-end es la parte del software con la que los usuarios van a interactuar. Esta se ocupará de la recolección de los datos que van a ser introducidos por el usuario, pero procesados por el back-end (capa que actúa de motor) y una vez reciba de vuelta la respuesta, se encargará de exponerlos de nuevo al usuario.

HTML5

Ilustración 8: Logo de HTML

HTML (HyperText Markup Language) es el elemento de construcción más básico de una página web. Describe y define el contenido de una página web. Su uso se complementa con otras tecnologías para definir la apariencia y presentación, como hace CSS o la funcionalidad de la misma, de la que se encarga JavaScript.

Si analizamos las palabras que forman HTML encontramos:

- HyperText: que se refiere a los enlaces que conectan unas páginas con otras, ya sea dentro de una misma página web o entre sitios diferentes. Estos enlaces son un aspecto fundamental de la web.

- Markup: que es utilizado para marcar textos, imágenes y otros contenidos que se muestran en el navegador. Algunos marcadores incluyen elementos como **<head>, <title>, <body>, <header>, <footer>, <p>, , <img>,** y muchos más.

Los elementos de HTML son añadidos al contenido de la página para describir la estructura de esta. Típicamente un elemento consta de las etiquetas de apertura y de cierre y entre medias se añade su contenido. Estas etiquetas normalmente vienen en parejas, pero hay algunos elementos vacíos que no tienen contenido. Se dice que constan de una etiqueta que se auto cierra. Las etiquetas de apertura llevan consigo atributos, que nos dicen muchas cosas más sobre el elemento. Los atributos tienen un nombre y un valor. El valor suele estar entrecomillado.

Actualmente nos encontramos con la versión HTML 5, cuyo primer borrador público fue presentado en el 2008. Esta versión incluyó nuevos elementos semánticos como son el **<header>, <footer>, <article>** y **<section>.** Cuando manejamos elementos form, podemos especificar nuevos atributos como **number, date, time, calendar y range.** Posee también nuevos elementos gráficos y multimedia, siendo respectivamente **<sgv>, <canvas>** y **<audio>, <video>.** En cuanto a las nuevas APIs, nos encontramos con las de Geolocalización, Drag & Drop, Local Storage, Application Cache, Web Workers y SSE.

Ilustración 9: Logo de CSS3

CSS (Cascading Style Sheets) es usado para definir los estilos de las páginas web, incluyendo el diseño, plantillas y variaciones en el display para diferentes dispositivos y tamaños de pantalla.

CSS resolvió un gran problema, pues HTML nunca estuvo destinado a incluir etiquetas para dar formato a una página web. HTML fue creado solo para describir el contenido de la web. Cuando en la especificación HTML 3.2 se incluyeron etiquetas con atributos de color y fuente, comenzó una pesadilla para los desarrolladores. El desarrollo de páginas web extensas, donde la información de fuentes y color se añadía en cada página, se convirtió en un proceso largo, tedioso y caro.

Para resolver este problema el W3W (World Wide Web Consortium) creó CSS. CSS eliminó el formato de estilo de las páginas HTML.

El uso de CSS ahorra una carga muy importante de trabajo. Las definiciones de estilo normalmente se guardan en un fichero externo con extensión .css. Con la hoja de estilos externa, puedes cambiar el aspecto de un sitio web entero modificando únicamente un fichero.

CSS ha ido evolucionando a la par que el resto de tecnologías web y en la actualidad los desarrolladores CSS se han adaptado a los preprocesadores. Un preprocesador CSS es un lenguaje de scripts que extiende CSS y que posteriormente lo compila a un CSS regular.

Esto ha surgido del hecho de que CSS puede llegar a ser bastante repetitivo y pequeñas tareas como buscar el código hexadecimal de colores, cerrar las etiquetas, etc. pueden ser procesos en los que se pierda tiempo extra.

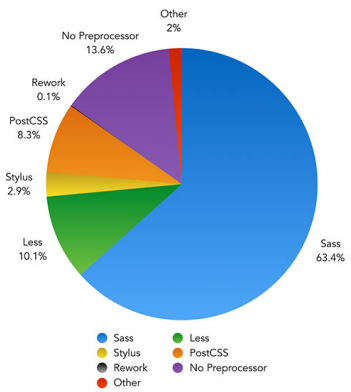


Ilustración 10: Encuesta sobre uso de preprocesadores CSS

La ventaja más grande de usar un preprocesador es que evitas repetirte a ti mismo, pero también va más allá y se obtiene un código más limpio con partes y variables reusables, que ahorrarán tiempo. Es más sencillo de mantener, con snippets y librerías. Implementa cálculos y lógica. En definitiva, todo queda más organizado y preparado para usar. Todo esto intenta seguir una misma filosofía y buenas prácticas que empieza a ser conocido como CSS Dry:

- Agrupar propiedades reusables juntas.
- Nombrar a estos grupos de una manera lógica.
- Añadir tus selectores de CSS a estos grupos.

```
#LIGHT-WHITE-BACKGROUND,
.translation,
.entry .wp-caption,
#full-article .entry img,
.recent-comment .comment-text,
.roundup h3,
.post-header-sharing,
#post-categories td.label,
#post-archive roundup h3,
.subscription-manager ol,
.light-white-background
{
  background-color: #fff;
  border-color: #ccc;
}

#MEDIUM-WHITE-BACKGROUND,
textarea:focus,
input:focus,
#author-email-form .input-focus,
#respond p input:focus,
.wpfc7 input:focus,
.medium-white-background
{
  background-color: #fff;
  border-color: #bbb;
}
```

Ilustración 11: Ejemplo de buenas prácticas DRY

SASS

SASS es un lenguaje de hoja de estilo que compila a CSS y que es usado por Ionic. Sass es como CSS, pero con funcionalidades extra como las variables, mixins, los bucles for y reglas anidadas. Proporciona muchas funcionalidades para la manipulación de colores y otros valores. Posee características avanzadas como el control de directivas para librerías.

Para poder ejecutar SASS, se ha de tener Ruby instalado. En las últimas versiones de Linux y OSX ya viene preinstalado. Pero también existen otras librerías que llevan SASS a Node.JS, sin necesidad de instalar Ruby.

En la elaboración de este proyecto se ha elegido utilizar SASS en ciertas ocasiones porque era lo recomendado por los proveedores del framework. El preprocesador LESS también ha sido una opción que he tenido en cuenta, pero analizando los datos de diferentes estudios, se concluye que en este momento SASS es más utilizado y conocido.

Ilustración 12: Logo de JavaScript

JavaScript (comúnmente abreviado como JS) es un lenguaje ligero e interpretado orientado a objetos con funciones de primera clase (acepta funciones como parámetros de entrada de otras funciones). Aunque típicamente se conoce como un lenguaje de script para páginas web, se utiliza también en numerosos entornos fuera del navegador, como pueden ser Node.js, Apache CouchDB y Adobe Acrobat.Javascript es un lenguaje basado en prototipos, multi-paradigma, dinámico y que soporta la orientación a objetos, la programación funcional e imperativa.

Fue desarrollado por Netscape. ECMAScript es el lenguaje de scripts que forman las bases de JavaScript. Está estandarizado por la Organización de Estándares Internacionales ECMA.

Contrariamente a la creencia popular al escuchar por primera vez hablar de JavaScript, no es Java interpretado. Se puede resumir con que es un lenguaje de scripts dinámico que soporta el prototipado basado en la construcción de objetos. La sintaxis básica es intencionadamente similar a Java y a C++, para reducir el número de nuevos conceptos requeridos para aprender el lenguaje.

Entre sus características dinámicas se incluyen la construcción de objetos en tiempo de ejecución, las listas de parámetros variables, la creación de scripts dinámicos (mediante `eval`), la introspección de objetos (mediante `for … in`) y la recuperación de código fuente (los programas en JavaScript pueden descompilar cuerpos de funciones en el texto original).

Posiblemente sea uno de los lenguajes de programación más incomprendido que existe, empezando por el nombre (originalmente se llamaba LiveScript). Aunque la sintaxis sea más parecida a Java o a C, realmente tiene más en común lenguajes como **Lisp** o **Scheme**. Posee objetos que pueden contener datos y métodos. Estos objetos pueden contener otros objetos. No tiene clases, pero tiene constructores, que hace lo que hacen las clases. No tiene herencia orientada a clase, pero sí que la tiene orientada a prototipado.

Angular 2

Ilustración 13: Logo de Angular

Angular 2 es una plataforma que facilita la creación de aplicaciones en la web. Fue la segunda versión del popular MV* (*Model-View-**) framework de Google para crear aplicaciones más complejas en el navegador, o incluso en otros ambientes.

Angular es usado para desarrollar aplicaciones en HTML y en JS o en otro lenguaje del estilo, como TypeScript, que compila a JavaScript. Está formado por multitud de librerías, algunas de ellas se integran en el core pero otras son opcionales.

Las aplicaciones Angular se crearán a través de las plantillas HTML con marcado propio de Angular. Se escribirán clases components para gestionar los modelos. Cuando se inserten servicios, aumentará la lógica de la aplicación. Será posible agrupar los servicios y los components en módulos.

Nos podemos hacer una idea superficial de cómo funciona Angular si observamos el siguiente mapa.

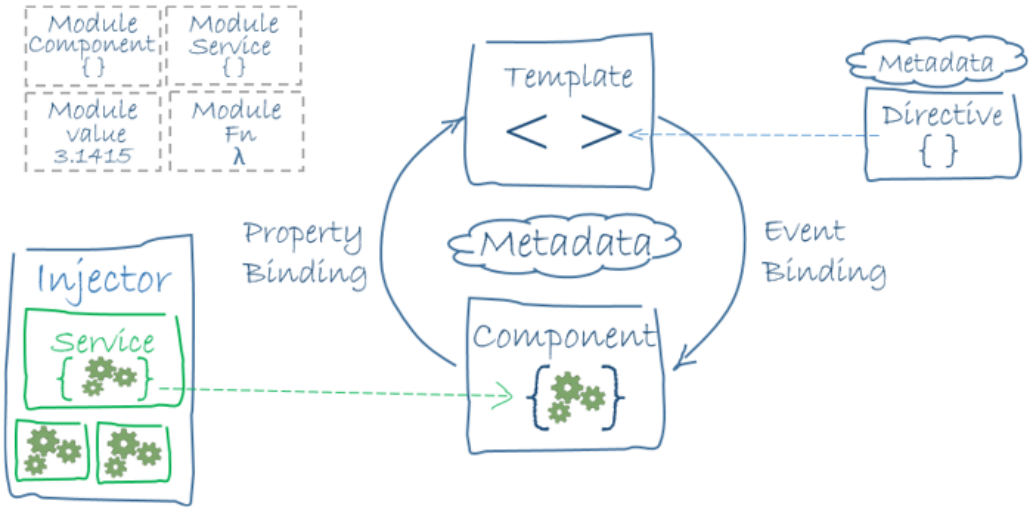


Ilustración 14: Mapa conceptual de Angular

Módulos

Las aplicaciones Angular son modulares y Angular tiene su sistema propio de modularidad llamado NgModules. Al menos todas las aplicaciones tendrán una clase NgModule, el módulo raíz (*root module*), que se suele nombrar como AppModule.

Salvo que la aplicación sea pequeña, la mayoría de las aplicaciones tienen más módulos (*feature modules*), cada uno de ellos con una función en concreto.

Un NgModule, ya sea root o feature, es una clase con el decorator @NgModule. Los decorators son funciones que modifican clases de JavaScript. Angular usa los decorators para añadir metadatos a las clases, para que sepa qué clases se han modificado y cómo debe utilizarlas.

Las propiedades más importantes de los módulos son:

- Declarations: las clases tipo vista que pertenecen a ese módulo. Angular tiene tres tipos de vistas: componentes, directivas y tuberías.

- Exports: el subconjunto de declaraciones que deben ser visibles y usables en la plantilla de componente de otros módulos.

- Imports: otros módulos que exportan clases y son necesitados por las plantillas de componente declaradas en el módulo "this".

- Providers: creadores de servicios con los que el módulo contribuye. Son accesibles en todas las partes de la aplicación.

- Bootstrap: la vista principal de la aplicación (*root component*) que alberga las otras vistas. Solo el *root module* debe tener esta propiedad.

Aunque JavaScript tenga también su propio sistema de módulos para gestionar colecciones de objetos, es completamente diferente y no guarda relación con el sistema NgModule. En JavaScript cada fichero es un módulo y todos los objetos definidos en el fichero, pertenecen a ese módulo. El módulo declara algunos objetos como públicos marcándolos con la palabra reservada **export**. Otros ficheros JS usan las declaraciones **import** para acceder a objetos públicos de otros módulos.

Las librerías de Angular exportan una colección de módulos JavaScript. Los nombres de las librerías comienzan con el prefijo **@angular**. Se pueden instalar con el gestor de paquetes NPM e importar posteriormente partes de la librería con la declaración **import**. Algunos ejemplos:

```
import { Component } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
```

Componentes y plantillas

Un componente controla una parte de la pantalla denominada vista. Se define la lógica de la aplicación dentro de la clase, a la vista que tendrá que respaldar dicha componente y su función. La clase interactúa con la vista a través de la API de propiedades y métodos.
Angular crea, actualiza y destruye componentes mientras el usuario se mueve por la aplicación.

Cuando se define un componente, se define junto con una plantilla. La plantilla es una forma de HTML que le dice a Angular como renderizar el componente.

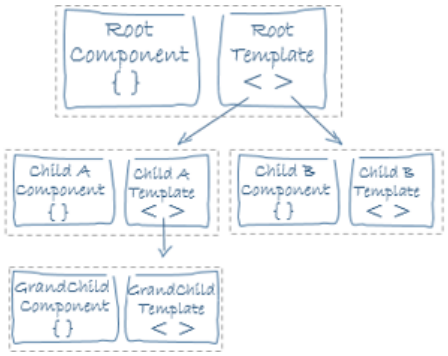


Ilustración 15: Relación entre componentes y plantillas

Metadata

Al observar el mapa general, vemos que entre la plantilla y el componente existe algo más, los metadatos. Los metadatos dicen a Angular cómo procesar la clase. En un principio, cuando estamos creando un componente, lo creamos declarando una clase, y no existiría ninguna evidencia o uso de Angular si no fuese por los metadatos que se incluyen posteriormente. Un componente no deja de ser una clase hasta que no se le dice al framework de Angular que lo trate como tal.

Las opciones de configuración más típicas para un componente son:

- Selector: es un selector de CSS que dice a Angular cuando crear y añadir la instancia de la componente cuando encuentre la etiqueta correspondiente en el fichero HTML.

- TemplateURL: define la dirección del fichero HTML.

- Providers: en este array se definen los servicios que el componente necesita.

Data Binding

Si no tuviésemos un framework, el traspaso de datos al HTML o convertir las respuestas del usuario en acciones serían tareas de las que nos tendríamos que ocupar y son tareas bastante tediosas.

Para evitar esto, Angular proporciona "*data binding*", que es el mecanismo con el cual se coordinan partes de la plantilla HTML con partes del componente.

Hay cuatro formas de llevar a cabo esa unión, cada una de ellas tiene una dirección: al DOM, desde el DOM o ambas direcciones.

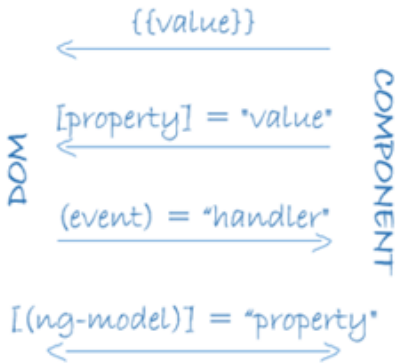


Ilustración 16: Interacciones entre el DOM y las componentes

Directivas

Las plantillas de Angular son dinámicas, eso significa que cuando se renderizan, el DOM se construye y modifica de acuerdo con las instrucciones dadas por las directivas. Una directiva es una clase con el decorator @Directive. Existen dos tipos de directivas:

- Estructurales: estas directivas alteran el layout del DOM añadiendo, borrando y remplazando elementos.
- Atributos: estas directivas alteran bien la apariencia o bien el comportamiento de un elemento existente. En la plantilla HTML aparecerán como elementos típicos.

Servicios

Esta categoría es muy amplia y abarca cualquier valor, función o característica que la aplicación necesite. Casi todo puede entrar en servicios, pero se suele acotar como una clase con un propósito bien definido. Básicamente tiene que llevar a cabo alguna tarea en concreto, pero hacerla bien.

Angular no tiene ninguna definición sobre lo que tiene que ser un servicio. Tampoco hay una clase base, ni tampoco un sitio para registrarlos.

Las clases componentes deben ser ligeras. No tienen que buscar datos en el servidor, ni validar la entrada del usuario o hacer log en la consola. Estas tareas se delegan a los servicios. Angular no impedirá que se construyan componentes que no sigan la buena práctica anterior y realice todo por sí misma, pero sí que dará facilidad para que se implementen los servicios en las componentes gracias a la inyección de dependencias (*dependency injection*).

TypeScript

Ilustración 17: Logo de TypeScript

TypeScript es un **superset** de JavaScript, lo que significa que disponemos de JavaScript junto con unas características extra, como las *interfaces* y *type declarations*. Ionic está creado con TypeScript, pero su uso es opcional a la hora de desarrollar la aplicación.

Es un lenguaje gratuito y open-source (bajo licencia Apache 2.0) que ha sido desarrollado y está siendo mantenido por Microsoft. Apareció por primera vez en octubre del 2012. Aunque su mayor fuente de influencia proviene evidentemente del lenguaje JavaScript, también se basa en lenguajes claramente orientados a objetos como son Java y C#.

TypeScript se desarrolló para satisfacer las necesidades de equipos de programadores que crean y mantienen grandes programas escritos en JS. Ayuda a definir interfaces entre componentes del software y a tener una mejor visión sobre el comportamiento de las bibliotecas existentes en JavaScript. También al organizar el código en módulos que se llaman de manera dinámica, se evitan los conflictos al nombrar el código.

TypeScript parte pues de la misma sintaxis y semántica que millones de programadores de JavaScript ya conocen. Se puede por ello usar código ya existente en JS, incorporar librerías y usar llamadas en TypeScript desde código en JS. TypeScript se compilará en código JavaScript debidamente estructurado y limpio, que se podrá ejecutar en cualquier navegador, en Node.JS o en cualquier motor JS que soporte ECMAScript 3 o superior.

El lenguaje tipado ofrece entre otras ventajas la refactorización (reestructurar el código fuente alterando su estructura interna sin cambiar su comportamiento interno) al desarrollar aplicaciones en JS.

TypeScript soporta todas las últimas características de JS incluyendo incluso sus futuras propuestas como las funciones asíncronas y los decorators. También incluye las clases y los módulos. Las clases permiten al programador expresar patrones comunes de código orientado a objetos de una manera estándar, haciendo características como la herencia más legible. Los módulos permiten organizar de una mejor manera el código en componentes.

Aunque exista software como Visual Studio 2017 (o VS 2015 Update 3) que ya incluye TS por defecto, el compilador en línea de comandos de TypeScript se puede instalar como un paquete de Node.JS.

```
npm install -g typescript
```

Para poder compilar posteriormente también desde la línea de comandos. Aunque se use la extensión .ts, este código no deja de ser JS, cómo se verá en el fichero compilado de salida, que sí que tendrá la extensión .js.

7. Apache Cordova

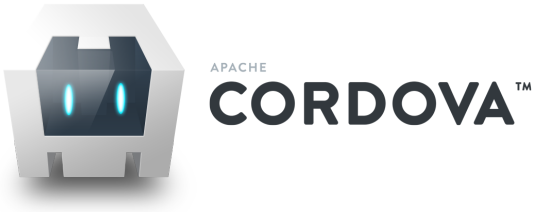


Ilustración 18: Logo de Apache Cordova

Apache Cordova es un framework para desarrollo móvil open-source. Posibilita usar tecnologías web como son HTML5, CSS3 y JS para crear aplicaciones multiplataforma. Las aplicaciones se adaptarán a cada plataforma con sus wrappers correspondientes. Proporcionará las API para poder acceder a las diversas funcionalidades del dispositivo como sensores, datos, estado de la red, etc.

Sus orígenes se remontan al proyecto open-source llamado PhoneGap creado por Nitobi con el objetivo de empaquetar código HTML5 con wrappers nativos para la creación de aplicaciones móviles. Adobe compró Nitobi y la tecnología PhoneGap, pero contribuyó con el código de PhoneGap a la Fundación de Software Apache para asegurarse que siguiese siendo open-source. En ese momento el nombre fue cambiado a Cordova.

Adobe mantiene por su cuenta aún el nombre PhoneGap y le añade valor con herramientas de ayuda y servicios tan famosos como el PhoneGap Build. Esto bajo la perspectiva de un desarrollador es bastante bueno, pues mientras el framework en sí se mantiene como open-source, sí deseas ese valor añadido que proporciona Adobe, compras la correspondiente licencia.

Apache Cordova permite extender la aplicación a otras plataformas sin tener que reescribir el código en cada uno de los lenguajes. Esto ahorra grandes costes y tiempos en el proceso de desarrollo de la app.

Para comprender mejor los componentes que forman una aplicación en Cordova, podemos observar el diagrama de la arquitectura a alto nivel.

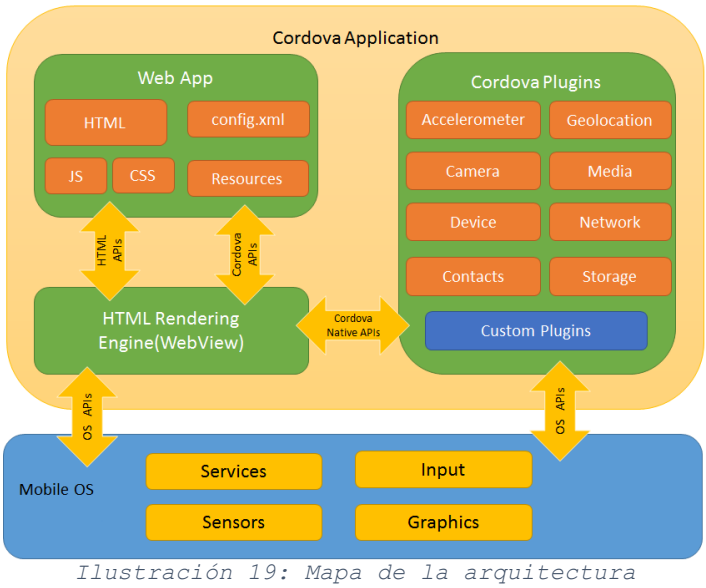


Ilustración 19: Mapa de la arquitectura

En Apache Cordova cabe destacar:

- WebView: Cordova habilita una WebView que proporcionará a la aplicación la interfaz de usuario. Puede ser en ciertos casos una componente dentro de otra más grande, como cuando se mezcla una aplicación híbrida con componentes nativos.

- WebApp: es la parte donde el código de la aplicación se encuentra. La app como tal se implementa como una página web, de hecho por defecto existirá un fichero llamado index.html que referenciará otros ficheros CSS, JS, imágenes, archivos multimedia o cualquier recurso que sea necesario. Debe incluir un fichero muy importante, que es el config.xml que ofrecerá información sobre parámetros que afecten al funcionamiento de la aplicación, como por ejemplo si se puede rotar.

- Plugins: los plugins son fundamentales en el ecosistema Cordova. Ofrecen una interfaz para Cordova y componentes nativos que se comunican entre ellos y las APIs del dispositivo móvil. Gracias a esto seremos capaces de invocar código nativo desde JavaScript. El proyecto Apache Cordova mantiene una serie de plugins en su llamado "Core Plugins", entre los que se encuentra el acceso a la batería, cámara, contactos, etc. También existen plugins desarrollados por terceros. Muchos de esos se pueden encontrar en NPM.

A la hora de desarrollar una aplicación en Cordova tenemos que elegir la forma en la que lo vamos a hacer, pues existen dos flujos de trabajo que, si bien pueden realizar la misma tarea, cada uno ofrece diferentes ventajas sobre el otro.

- Multiplataforma: este flujo de trabajo permite correr la aplicación en multitud de diferentes sistemas operativos móviles con muy poco desarrollo específico para cada uno. Hay un nivel de abstracción mucho mayor. Este es el flujo de trabajo recomendado.

- Centrado en una plataforma: cuando únicamente nos queremos centrar en una plataforma y queremos ser capaces de modificarla posteriormente a un nivel más bajo, usaremos esta opción. Esto nos permitirá usar el SDK correspondiente de cada plataforma. Se podría usar este método para crear apps multiplataforma, pero al irnos a un nivel más bajo de arquitectura, generalmente es mucho más difícil y hay que hacer las cosas por separado para cada una de ellas.

Una vez se cambie del flujo de trabajo multiplataforma hacia el de una única plataforma, no se puede volver hacia atrás. Así que conviene hacer un estudio previo de las necesidades antes de ponerse a desarrollar la aplicación y ver que flujo se adapta mejor.

Ya hemos visto la importancia que tienen los plugins en Cordova, pero aún falta por matizar que el desarrollador puede incluso crearse sus plugins personalizados. Un plugin es un paquete de código que permite a la WebView de Cordova que el dispositivo renderiza, comunicarse con la plataforma nativa en la que se está ejecutando. Los plugins ofrecen un acceso a funcionalidades que no suelen estar disponible en las aplicaciones web. Los plugins aúnan una interfaz JavaScript con las correspondientes librerías de código nativo.

8. Ionic 2

"Code once. Run everywhere"

Ilustración 20: Logo de Ionic

El mundo del desarrollo móvil actualmente está muy fragmentado por multitud de plataformas, frameworks y tecnologías. El equipo de Ionic se propuso cubrir el hueco que hay con su framework open-source que permite a los desarrolladores construir apps que el usuario pueda sentir como nativas con tecnologías web como HTML, CSS, AngularJS… Ionic da la oportunidad a los desarrolladores de front end de convertirse en desarrolladores de apps.

Ionic ha sobresalido sobre otros frameworks por los siguientes motivos:

- Herramientas y servicios: Ionic ofrece poderosas herramientas desde la línea de comandos para crear una nueva aplicación, generar los iconos (más de 900 ionicons) adecuados para cada plataforma, visualizar en el navegador los cambios que se van realizando en la aplicación, etc.

- Documentación: cuando se entra en la documentación de Ionic se obtienen resultados concisos con buenos ejemplos para poder comprender cómo funciona. También guían en la instalación, en los tests y en la publicación en los diversos markets.

- Comunidad: este framework está respaldado por una comunidad muy activa en foros técnicos como StaclOverflow en los que se expondrán las dudas que acontezcan a los desarrolladores. Tiene también la ventaja de apoyarse en la comunidad de Angular.

- Mercado: Ionic ofrece un rico y variado Marketplace donde se pueden obtener temas, plugins y apps básicas. Algunas de manera gratuita y otras de pago.

- Gratuito y open-source: está licenciado bajo la licencia open-source MIT. Esto se supone que se puede utilizar Ionic para proyectos personales y empresariales sin tener que pagar por ello.

Las aplicaciones Ionic son creadas y desarrolladas en su mayor parte con la línea de comandos y usan Cordova para construir la app como si se tratase de una app nativa. Esto significa que haremos uso de diferentes utilidades de Cordova para poder seguir el proceso de desarrollo.

Ilustración 21: Filosofía Ionic

Arquitectura

A la hora de construir nuestra aplicación hemos de tener claro el nivel en el que nos encontramos y con qué arquitectura estamos trabajando.

Ilustración 22: Arquitectura en forma de pila

Los SDKs (Software Development Kit) nativos de cada sistema operativo forman la base del framework. Las funcionalidades nativas del dispositivo como el acelerómetro, cámara, contactos, son alcanzadas por los plugins de Cordova. Ionic posteriormente ofrece un estilo nativo en los controles, como en las etiquetas y campos de texto. También se encarga de las interacciones del usuario, como presionar y escribir. Angular tiene un papel muy importante en las funcionalidades del núcleo de Ionic. No es obligatorio que los desarrolladores utilicen Angular en sus proyectos, pero sí que es recomendable.

Comenzando con Ionic2

Para crear proyectos Ionic es necesario instalar la última versión del CLI (*command-line interface*) y de Cordova. Antes de hacer esto, es necesaria también la versión más reciente de Node.js.

```
npm install -g ionic cordova
```

Una vez hecho, crearemos la aplicación desde la línea de comandos. Según la plataforma en la que queramos desarrollar la aplicación, será preciso seguir una serie de pasos. Revisamos por encima los necesarios para iOS y Android.

Guía para la plataforma iOS

Las herramientas necesarias para desarrollar aplicaciones iOS solo pueden ser utilizadas en el sistema operativo OS X. Se pueden probar muchas funcionalidades de Cordova usando el simulador iOS instalado en el iOS SDK y Xcode, pero se necesita un dispositivo real para testear todas las funcionalidades de la aplicación por completo antes de subirla a la App Store. Ese dispositivo tendrá que tener al menos iOS 8.

Requisitos que han de ser instalados:

- Xcode: El IDE es el centro para todo el desarrollo relacionado con Apple. Estrechamente integrado con Cocoa y el framework Cocoa Touch, Xcode es el medio de producción recomendado por Apple para crear apps para Mac, iPhone, iPad, Apple Watch, y Apple TV.

- iOS-deploy: Es un paquete que se encuentra en NPM y será usado para instalar y hacer debug en las apps de iOS sin usar Xcode.

La instalación de Xcode configurará la mayor parte de las cosas necesarias para poder crear y desarrollar un proyecto con Cordova.

Guía para la plataforma Android

Hemos de configurar el SDK para poder desarrollar aplicaciones Cordova para los dispositivos Android. Instalaremos el SDK Android, que a diferencia de Xcode, puede ser instalado tanto en OS X, Linux o Windows.

Versión de Cordova-Android	Niveles de API Android soportados
6.X.X	16-25
5.X.X	14-23
4.1.X	14-22
4.0.X	10-22
3.7.X	10-21

Tabla 4: Equivalencia entre versión Cordova y Android

Requisitos que han de ser instalados:

- Android Studio: el IDE oficial para Android. Proporciona las herramientas más rápidas para crear apps en todas las clases de dispositivos Android. Ofrece edición de códigos de primer nivel, depuración, herramientas de rendimiento y un sistema de compilación e implementación.

- Paquetes para los niveles de API: el nivel de API es un valor entero que identifica de manera única la revisión de la API Framework que ofrece una versión de la plataforma Android. La API Framework contiene un conjunto básico de paquetes y clases, así como de atributos y elementos XML para declarar archivos de manifiesto o recursos y acceder a estos, también incluye un conjunto de itents y de permisos que las aplicaciones pueden solicitar, al igual que cumplimientos de permisos incluidos en el sistema.

- Configurar las variables de entorno: que son necesarias por el CLI de Cordova y que han de ser configuradas manualmente. Estas son JAVA_HOME y ANDROID_HOME, que respectivamente indican la ruta a la instalación de JDK (*Java SE Development Kit*) y la del SDK de Android respectivamente.

Organización de un proyecto en Ionic 2

Ionic organiza el directorio del proyecto por características. Toda la lógica, las plantillas y los estilos de las funcionalidades se encuentran en la carpeta src.

El fichero desde donde es cargada toda la aplicación es index.html. Service-worker.js se encarga del comportamiento offline. En la carpeta "theme" se recogen todas las variables de estilo definidas.

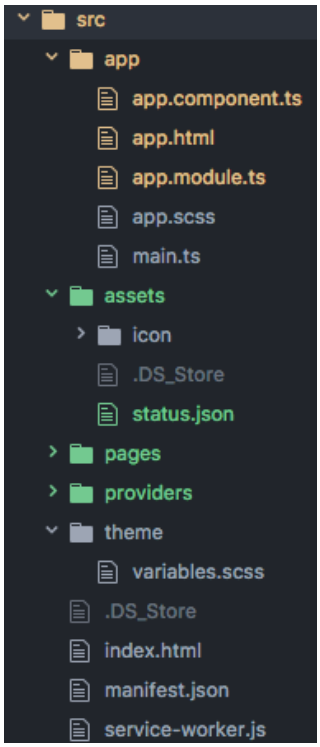


Ilustración 23: src

Dentro de la carpeta app se encuentran los niveles de la app y se define cómo la aplicación va a ser compilada y empaquetada. Encontramos:

- app.module.ts: representa la app como un módulo.
- app.component.ts: componente raíz de la app.
- app.html: vista principal de la app.
- app.scss: define los estilos de la app.
- main.ts: usado para hacer bootstrapping de la app.

En la carpeta "pages" se encuentran tres ficheros por cada vista/página creada en para la app. Un fichero html que será la plantilla. Otro ts en el que se encuentra el TypeScript de la página. Aquí se definirá el decorator @Component. También cada página contendrá un fichero scss, que contendrá el SASS personalizado para cada página.

En la carpeta "providers" encontraremos un fichero TypeScript donde estará el código de cada servicio que creemos.

Cuando generemos nuestra aplicación, el código compilado irá a otra carpeta llamada "www". En general no es necesario modificar el código que se encuentra en esta carpeta.

Versiones de Ionic

Ionic se llega actualmente por su tercera versión.

El cambio de la versión 1 a la 2 supuso una gran evolución en el framework. No solo por él, sino que también adoptó el cambio de versión de Angular, que la segunda versión poco tiene que ver con la primera. Por eso, aunque algunas cosas parezcan iguales entre ambas versiones, Ionic 2 es prácticamente un nuevo framework.

Esto no ocurre con la tercera versión que fue anunciada en Marzo del 2017. Ionic 3 es más una actualización que el cambio radical que sucedió entre Ionic 1 e Ionic 2. También anunciaron que a partir de ahora, para referirse al framework Ionic omitirían el número de la versión.

9. Base de datos

Muchas aplicaciones requieren guardar datos de alguna manera para que el usuario pueda realizar acciones en la aplicación, después cierre la app y al volver a abrirla la aplicación recuerde los diferentes datos guardados de la última vez que se abrió.

Por defecto, todo lo que se haga en la aplicación se perderá tan pronto como el usuario la cierre. Para algunas aplicaciones puede ser válido, pero una gran mayoría necesitan de la capacidad de almacenamiento del dispositivo.

La creación de bases de datos locales permite no depender de la conexión a internet cuando los datos son requeridos. No siempre los datos tienen por qué almacenarse en el servidor, además es más rápido un almacenamiento local y posteriormente una sincronización al servidor con un proceso que se encuentre en background.

Hay varias maneras de conseguir que un dato sea persistente a lo largo de diferentes sesiones.

Almacenamiento del navegador

Los navegadores a día de hoy cuentan con un soporte para almacenamiento local. La forma más simple de usar este servicio a través de Ionic es:

```
import {Storage} from '@ionic/storage';
```

Y posteriormente añadir al providers. Todo eso en ***app.module.ts***. Es un almacenamiento con "key:value". El problema de este almacenamiento es que no es completamente estable y está limitado a 5mb. Si el dispositivo trata de liberar espacio, es posible que este tipo de datos sean serios candidatos a ser borrados.

Si queremos que el sistema operativo no nos borre los datos, tendremos que recurrir a un sistema de almacenamiento nativo. Ionic 2 va a hacer uso automáticamente del mejor método de almacenamiento que tenga al alcance.

SQLite

Ilustración 24: Logo de SQLite

SQLite es una gran alternativa para el almacenamiento offline de datos. Esta librería no necesita de otros servidores, ni de mucha configuración y está basado en las bases de datos SQL. A diferencia de otras BBDD SQL, SQLite no tiene procesos separados de servidor. Lee y escribe directamente en los ficheros de disco.

Otra ventaja que posee es su capacidad de adaptación a diferentes arquitecturas. Se puede copiar una base de datos entre sistemas de 32 y 64 bits o entre arquitecturas Big-endian y Little-endian.

Para poder usar el almacenamiento basado en SQLite necesitamos instalar el siguiente plugin de Cordova, pues SQLite no está pensado para trabajar en JavaScript.

```
ionic plugin add cordova-sqlite-storage
```

En **app.js** inicializamos y la configuramos.

```
Import {Storage, SqlStorage}
Dentro de initializaApp(){
      This.platform.ready().then() => {
            This.storage = new Storage(SqlStorage);
            This.storage.querey("CREATE TABLE IF NOT EXISTS")
}
```

CouchDB

Ilustración 25: Logo de CouchDB

CouchDB es una base de datos NoSQL basada en documentos que puede ser fácilmente sincronizada con otras bases de datos CouchDB. Los motivos por los que se ha pensado en esta BD para un futuro es por la simplicidad con la que puede ser escalada, la velocidad en las operaciones de leer y escribir datos, pero sobre todos los motivos destaca su habilidad para sincronizarse. Esto es especialmente útil cuando trabajamos en aplicaciones móviles, donde no es extraño que un usuario utilice la misma aplicación en una Tablet o un Smartphone.

CouchDB posee incluso un servidor web con una API http que permite interactuar directamente mediante su url si no se quiere añadir ningún tipo de middleware de servidor.

PouchDB

Ilustración 26: Logo de PouchDB

PouchDB es el homólogo a CouchDB en los navegadores, por lo tanto nos encontramos ante una base de datos NoSQL. Se pueden replicar datos de manera fácil desde una base de datos remota. Es una BD open-source en

JavaScript sencilla de implementar. Fue creada para ayudar a los desarrolladores web a construir aplicaciones que trabajasen offline de igual manera que conectadas a Internet. Se almacenan los datos localmente cuando la app está offline y posteriormente cuando es posible realizar la conexión se sincroniza con CouchDB y servidores compatibles, manteniendo en todo momento los datos del usuario sincronizados sin importar desde qué dispositivo se encuentre.

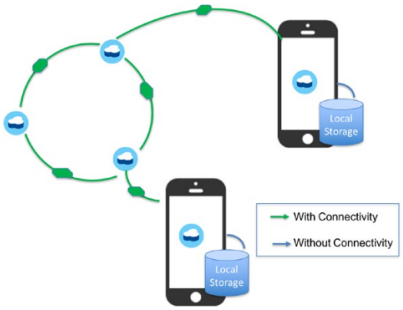


Ilustración 27: Sincronización offline y online

PouchDB puede ejecutarse en varios navegadores a la vez, con lo que la velocidad de ejecutar queries se incrementará. Es una base de datos bastante ligera que puede ser instalada mediante scripts en dispositivos móviles.

10. Electron

Ilustración 28: Logo de Electron

La historia comienza con Atom, que es el editor de textos de free source de la compañía GitHub. El sueño de Atom era construir una aplicación de escritorio mediante HTML, CSS y JS. Para conseguir eso, el equipo a cargo de desarrollarlo probó primero con el framework embebido de Chrome. También probaron con WJS, llamado entonces node WebKit. Pero no terminó de ser del todo lo que buscaban, así que llegaron a la conclusión de que iban a tener que desarrollar algo específico para la visión que tenían sobre Atom. Todo eso fue en enero 2013. Fue open-source en mayo del 2014. Al principio se llamaba Atom Shell, porque fue construido para Atom, pero quedó claro a medida que crecía el número de personas que usaban Electron, que iba a necesitar su propio nombre y no estar atado al editor de texto.

¿Qué es y cómo funciona?

Electron es un framework para construir aplicaciones de escritorio multiplataforma con tecnología web.

Este pequeño resumen de poco más de una línea implica muchas más cosas en verdad. Implica que estás construyendo una aplicación multiplataforma con HTML, CSS, JS y no estás usando Objective-C o C#. Tienes un único lenguaje base y no tres. Pensando en compañías significa que tienes un equipo y no tres. Se podrían incluir módulos nativos en la aplicación de Electron y usarlos, pero no es necesario. Puedes hacerlo todo con tecnología web.

Electron utiliza la librería de contenido de Chromium, que es la versión open-source del navegador Chrome. Esta se encarga de renderizar las páginas web y luego se combinado con Node.JS en una rutina en tiempo de ejecución.

Google desarrolló un motor de JavaScript para Chrome llamado v8 y sobre ese motor nació Node.JS, así que tiene sentido combinar estas dos cosas que contarán con las últimas características que salgan en Chrome, por lo que todo lo que Google desarrolle y utilice en Chrome, va a estar disponible en Electron también.

Al desarrollar la aplicación, solo es preciso diseñarla en un navegador, por lo que las personas que estén acostumbradas al desarrollo y diseño web y usen las diferentes etiquetas y prefijos y continuamente estén comprobando cómo resulta en los diferentes navegadores ahora ya no tendrán ese problema.

Debido a que Node.JS es back-end nos permite acceder al sistema de ficheros. También se tiene disponible el más de medio millón de módulos que almacena NPM y que pueden ser usados en cualquier lugar de la aplicación de Electron.

Las aplicaciones desarrolladas con Electron para Windows, Mac y Linux, disponen de los respectivos elementos UI nativas, como los diálogos para abrir ficheros, los estilos de los menús, mensajes de alerta.

Usando Electron

La principal cuestión que hay que entender es que a la hora de usar Electron, hay dos procesos:

- El proceso principal (main process) controla el ciclo de vida de la aplicación. Controla cuando se inicializa y cuando se termina la aplicación y proporciona diversos módulos que han sido desarrollados específicamente para este proceso, por ejemplo, el módulo de diálogos que permite interactuar con las ventanas al abrir un fichero y los elementos UI nativos de los menús. Hay otro módulo llamado IPC (inter process communication). Otro módulo llamado Browser Window permite crear el otro proceso, el renderer process.

- El proceso de renderizado (render process) es la página web. También tiene módulos desarrollados para él y tiene acceso a la API de Node.JS y también al DOM, ya que como hemos dicho es una página web. Tiene otro módulo IPC que se comunica con el del proceso principal. Un ejemplo práctico sería que tuvieses un botón en la página web cuya finalidad sea abrir un nuevo fichero, entonces el IPC del renderizador manda un mensaje al IPC del proceso principal, ya que es ese proceso el que se encarga del sistema de ficheros. El proceso principal es el que inicia el diálogo para abrir el fichero y luego retornará la respuesta del usuario de vuelta al proceso de renderizado.
  Se pueden tener varios procesos renderizados, tantos como se necesiten. No todos tienen porqué ser páginas web visibles, simplemente pueden ejecutar código JavaScript en background y así agilizar la carga de la página web visible.
  Una similitud sería lo que ocurre con Chrome y las pestañas. Cada pestaña sería un proceso renderizado, pero si cierras todas las pestañas, aún tienes las opciones de Chrome disponibles (archivo, edición, vista, etc…) eso sería el proceso principal.

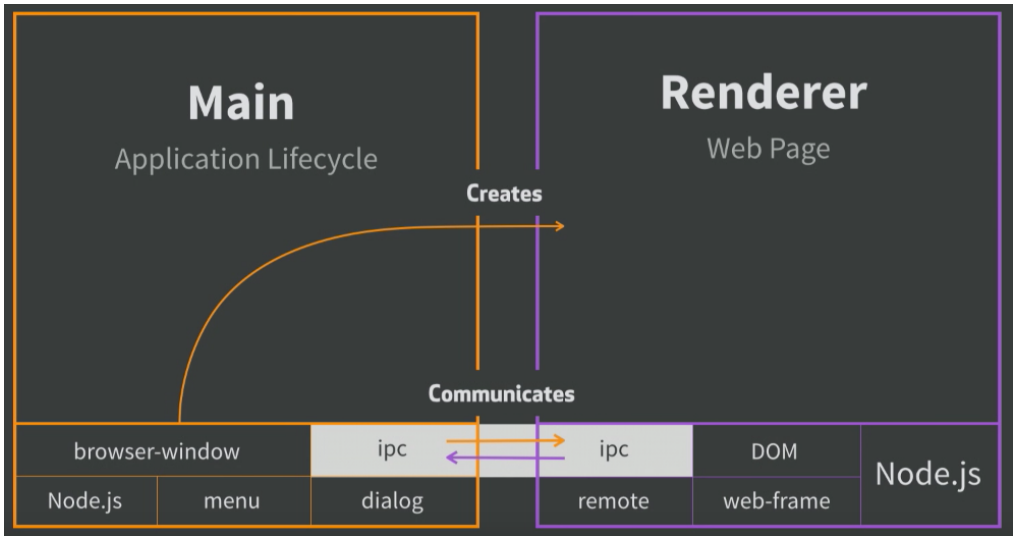


Ilustración 29: Interacción entre procesos

Crear una nueva aplicación en Electron

Nos situamos en el directorio deseado y desde el terminal escribimos

```
npm init
```

Esto nos ayudará a crear un fichero package.json que servirá para configurar los parámetros iniciales de nuestra aplicación.

```
npm install -g electron
npm install electron –save
```

Para que quede como una dependencia.

Dentro de package.json añadiremos la siguiente línea para que podamos iniciar nuestra aplicación desde la línea de comandos de una manera estándar para las aplicaciones que usan NPM.

```
“start”: “electron .”
```

Ahora, una vez que estemos en el directorio correcto, podremos teclear:

```
npm start
```

En el main.js creamos un objeto de Electron que contenga app y BrowserWindow. También traemos el módulo path y el URL, estos dos son para llamar al index.html.

Ficheros más relevantes

Package.json

Este fichero debe ser un JSON y no un simple objeto JavaScript. Dentro de él, encontraremos el nombre y la versión, que serán campos obligatorios. También podemos escribir una descripción, palabras claves de la aplicación, una dirección URL del proyecto, la licencia que está siendo usada, los autores y contribuidores… También encontramos el campo main, que es la ID del módulo que sirve de punto de entrada al programa.

```
“main”: “main.js”; // main process
```

Main.js

En el main.js vamos a usar en principio dos módulos para el proceso principal.

Uno es app, que nos dará la aplicación y el otro es BrowserWindow. Cuando la aplicación esté lista:

```
app.on(‘ready’,function(){ … })
```

crearemos la ventana del navegador:

```
mainWindow = new BrowserWindow
```

con las propiedades que queramos. Diremos entonces que cargue el fichero index.html, que explicaremos a continuación.

Index.html

Es un fichero HTML que cuando sea renderizado por el proceso BrowserWindow, tendrá acceso a Node.JS. Dentro del fichero HTML podemos tener sin problemas una etiqueta **<script>** con un **var** que diga

```
var fs = require("fs")
```

tal como si estuviésemos escribiendo en Node.JS, pero recordemos que estamos en index.html.

Electron globalmente

Tenemos un módulo llamado electron-prebuilt

```
npm install -g electron-prebuilt
```

```
electron .
```

Que nos permite ejecutar aplicaciones Electron desde la línea de comandos.

Otros módulos importantes son:

- Electron-packager: este módulo resultará útil a la hora de exportar nuestra aplicación a las diferentes plataformas, haciendo el proceso transparente.

- Electron-compile: evitará tener que cerrar y abrir nuevamente la aplicación cada vez que modifiquemos algo en el código.

Estos dos módulos han sido creados directamente por la comunidad que hay detrás de Electron.

Al estar basado en Chrome, otras de las primeras ventajas que se pueden apreciar es que en cualquier momento podemos abrir el "dev tools", que es muy útil a la hora de hacer debugging y de diseñar la aplicación de la misma manera que diseñaríamos un sitio web

Dentro de las compañías que han elegido Electron para desarrollar sus aplicaciones de escritorio nos encontramos con: Slack, Microsoft, miles de startups y una cantidad enorme también de proyectos open-source.

11. Análisis

A continuación, se analizarán los requisitos funcionales más importantes de la aplicación. Para llevar a cabo dicho análisis se definen una serie de casos de uso que pasarán a ser descritos.

Comenzar reproducción

ID	Comenzar reproducción
Creado por	Borja Pérez Morán
Actores	Usuario
Descripción	Una vez seleccionado el fichero multimedia y escogidos los idiomas, se abrirá el reproductor VLC.
Precondición	El usuario ha configurado el programa VLC como reproductor multimedia por defecto. El usuario elige si desea que sean mostrados todos los subtítulos o solo los codificados en UTF-8.
Flujo normal	**1.** El usuario selecciona el fichero multimedia. **2.** El sistema muestra los botones para seleccionar: • idioma del subtítulo. • idioma al que traducir el subtítulo. • tiempo entre desafíos. **3.** El usuario una vez haya elegido los parámetros anteriores, clica en el botón "Start Watching". **4.** El sistema abrirá el fichero con el programa por defecto.
Flujo alternativo	**3.** El usuario cancela la acción y cierra el modal. **4.** El sistema no encuentra subtítulos disponibles y se alerta al usuario.
Postcondición	Se ha logrado el objetivo del caso de uso

Tabla 5: Análisis "Comenzar reproducción"

Traducir palabra

ID	Traducir palabra
Creado por	Borja Pérez Morán
Actores	Usuario
Descripción	El usuario selecciona la palabra que no entiende.
Precondición	Ninguna
Flujo normal	**1.** El usuario pausa la reproducción. **2.** El sistema muestra los subtítulos que estaban en la pantalla del reproductor. **3.** El usuario presiona sobre la palabra que no entiende. **4.** El sistema utiliza la API de Google Translator para encontrar una traducción. **5.** El usuario obtiene la traducción en el idioma que seleccionó desde la app de escritorio.
Flujo alternativo	**4.** El sistema muestra la misma palabra porque Google Translator no devuelve ninguna traducción.
Postcondición	Se ha logrado el objetivo del caso de uso

Tabla 6: Análisis "Traducir palabra"

Guardar palabra

ID	Guardar palabra
Creado por	Borja Pérez Morán
Actores	Usuario
Descripción	Almacena una palabra que haya sido seleccionada por el usuario y traducida por el software.
Precondición	El usuario detiene la reproducción y selecciona una palabra desde la vista "Go Learning"
Flujo normal	**1.** El usuario presiona una palabra que no entienda. **2.** El sistema muestra su traducción y ofrece almacenar la palabra. **3.** El usuario decide guardar la palabra para su posterior revisión. **4.** El sistema continúa la reproducción del contenido multimedia.
Flujo alternativo	**3.** El usuario elige no guardar la palabra
Postcondición	Se ha logrado el objetivo del caso de uso

Tabla 7: Análisis "Guardar palabra"

Completar espacio

ID	Completar espacio
Creado por	Borja Pérez Morán
Actores	Usuario
Descripción	El usuario tiene que completar el subtítulo que acaba de escuchar con una palabra que ha sido eliminada. Dispone de tres intentos.
Precondición	El usuario ha visto un hueco donde falta una palabra durante la reproducción del contenido multimedia.
Flujo normal	**1.** El usuario pausa la reproducción. **2.** El sistema muestra el subtítulo que acaba de visualizar en el ordenador. **3.** El usuario escribe correctamente la palabra que ha escuchado. **4.** El sistema continúa la reproducción del contenido multimedia.
Flujo alternativo	**3.** El usuario no escribe correctamente la palabra que ha escuchado tras haber intentado tres veces. **4.** El sistema mostrará la palabra y continúa la reproducción del contenido multimedia.
Postcondición	Se ha logrado el objetivo del caso de uso

Tabla 8: Análisis "Completar espacio"

Configurar aplicación

ID	Configurar aplicación
Creado por	Borja Pérez Morán
Actores	Usuario
Descripción	El usuario tiene que configurar la aplicación para que el móvil se conecte con el servidor web del ordenador. El móvil se encuentra conectado a la misma red que el ordenador.
Precondición	El usuario ha visto un hueco donde falta una palabra durante la reproducción del contenido multimedia.
Flujo normal	**1.** El usuario escribe: • la dirección IP del ordenador en el que se está ejecutando el VLC. • la contraseña de la interfaz http del reproductor VLC. • El sistema operativo del ordenador. **2.** El sistema recoge esas variables que posteriormente serán utilizadas para establecer la conexión al servidor web.
Flujo alternativo	
Postcondición	Se ha logrado el objetivo del caso de uso
Notas	Esta configuración se ha de llevar a cabo la primera vez que se habrá la aplicación o si se desea usar la aplicación en otro ordenador.

Tabla 9: Análisis "Configurar aplicación"

Ayuda

ID	Ayuda
Creado por	Borja Pérez Morán
Actores	Usuario
Descripción	El usuario puede tener dudas en el proceso de configuración de las aplicaciones.
Precondición	Ninguna
Flujo normal	**1.** El usuario entra en la vista "Help?" de la aplicación móvil. **2.** El sistema muestra mediante slides problemas o dudas considerados frecuentes. **3.** El usuario obtiene la información necesaria para resolver sus dudas.
Flujo alternativo	
Postcondición	Se ha logrado el objetivo del caso de uso

Tabla 10: Análisis "Ayuda"

12. La aplicación: Watch Your Words!

Manual de usuário para la app móvil

Home & Menu

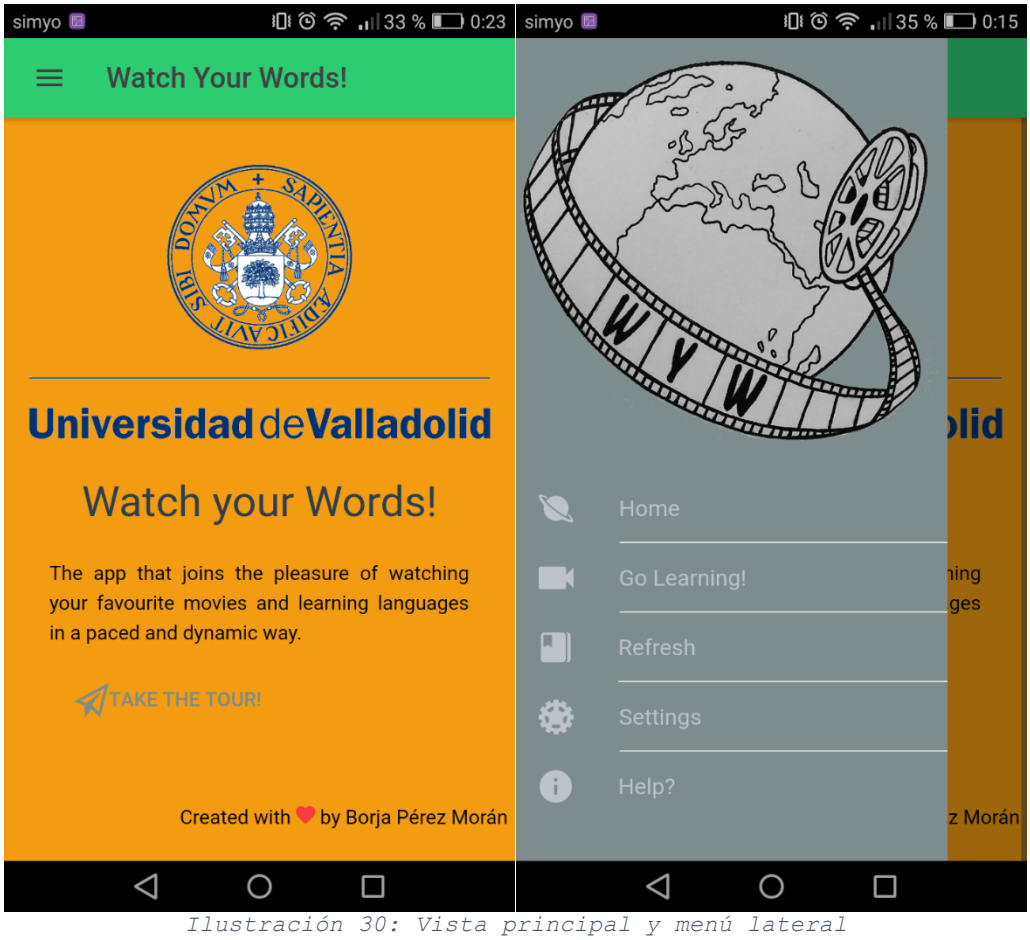


Ilustración 30: Vista principal y menú lateral

Es la primera vista que verá el usuario cuando abra la app. En ella se puede ver el logotipo de la Universidad de Valladolid, el nombre de la app con una pequeña descripción y un botón que enlaza con la vista "Help?".

En esta primera vista, el usuario deberá abrir el menú que se encuentra escondido a la izquierda. Ya sea mediante la acción de deslizamiento, o pulsando el botón del menú. También podrá ir directamente a la ayuda prestada si clica en "Take the tour".

El menú estará disponible en todo momento en las diferentes vistas de la app y permitirá una navegación fluida por las diferentes secciones de la misma. Consta del logotipo de **Watch Your Words!** y de una lista con las diferentes vistas con un pequeño icono orientativo.

En el pie de página aparece la referencia al autor de la aplicación.

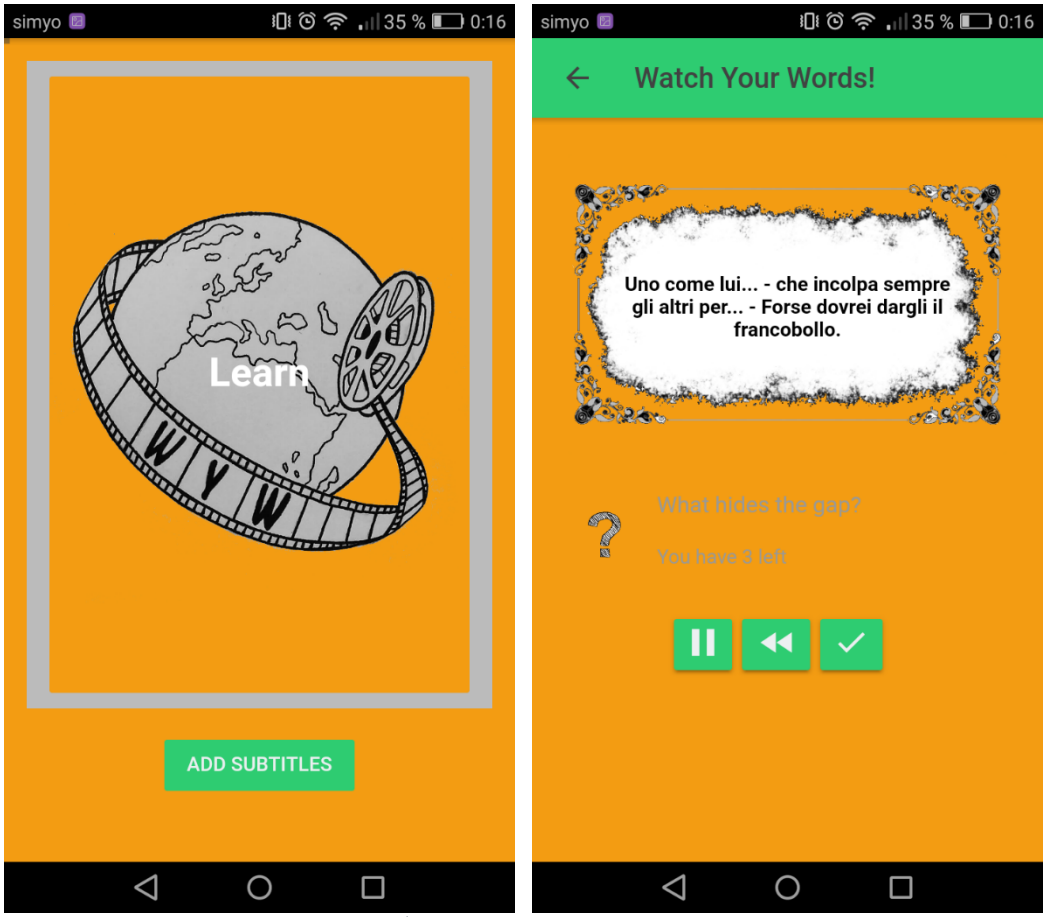


Ilustración 31: Vista de Go Learning

En el escenario típico de que el usuario ya haya introducido la dirección IP de su ordenador, y ambos dispositivos (móvil e ordenador) se encuentren conectados a la misma red local, el logotipo de la aplicación Watch Your Words, se visualizará en la pantalla del móvil, con la palabra "Learn" en el medio. Aparecerá un botón para que el usuario añada el subtítulo al reproductor VLC.

Esta es la pantalla que deberá permanecer mientras dure el visionado de la película. Tiene dos funcionalidades:

- En el momento que el usuario no entienda una palabra, con que pulse en la imagen, la película se parará e instantáneamente se mostrarán los subtítulos que se acaban de reproducir. Con seleccionar mediante una pulsación la palabra que no se ha entendido, la app mostrará en un pop-up esa palabra traducida al idioma que se haya elegido previamente en la aplicación de escritorio. Así mismo, se ofrecerá al usuario la opción de guardar o no esa palabra, para que en cualquier momento pueda refrescar sus conocimientos. En un futuro, se utilizarán estas palabras para proponer nuevos desafíos al usuario que consigan mejorar su experiencia al usar la aplicación.

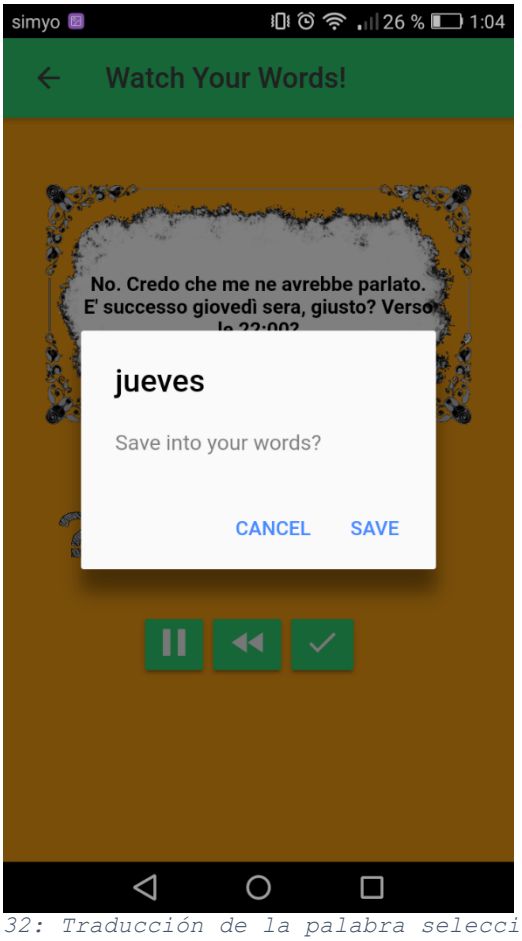


Ilustración 32: Traducción de la palabra seleccionada

- En el momento que el usuario vea en la pantalla de su ordenador que falta una palabra en el subtítulo, deberá pulsar en la imagen y la película también se pausará. Aparecerá en el dispositivo móvil el mismo subtítulo con el hueco y una frase que le pregunte la palabra que se ha eliminado. Cuando el usuario escriba la palabra, en caso de estar bien, se continuará automáticamente la reproducción. Si necesita escuchar de nuevo la película, podrá pulsar el botón de rebobinar, y volverá a escuchar los tres segundos previos. En caso de no acertar tras varios intentos, se podrá continuar el visionado de la película y se mostrará la palabra que era. También en este momento, si diese la casualidad de que el usuario no sepa una palabra, puede pedir la traducción y decidir si desea guardarla o no.

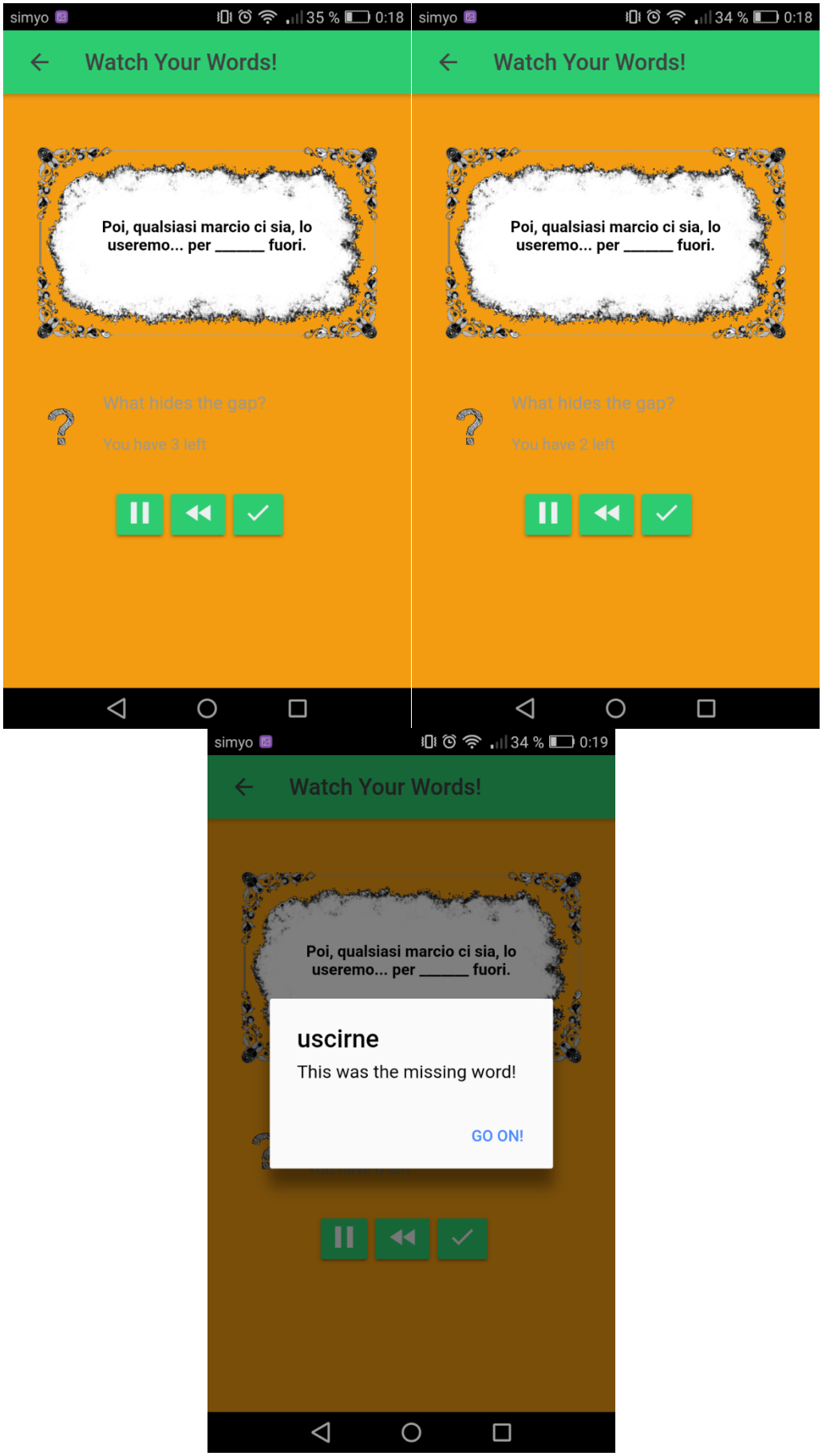


Ilustración 33: El usuario dispone de 3 intentos

Refresh

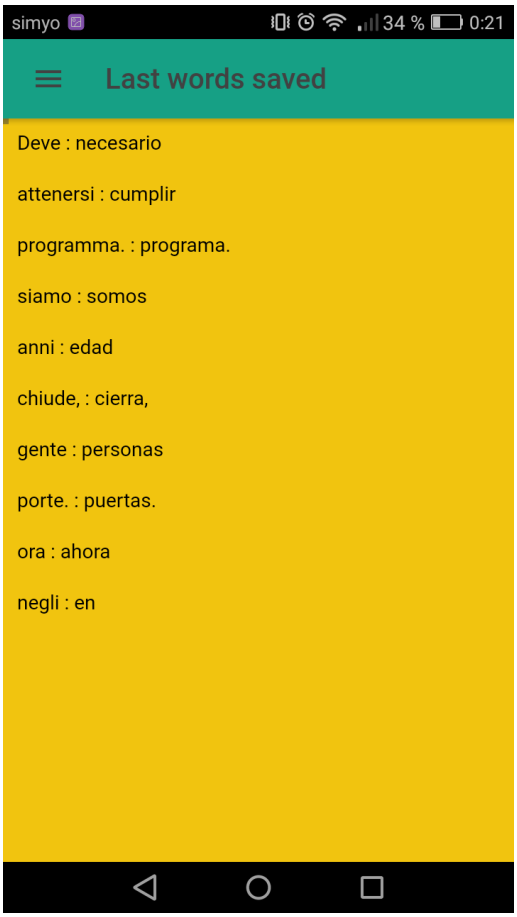


Ilustración 34: Palabras guardadas con su traducción

Las palabras que se hayan guardado en la otra pantalla quedarán reflejadas en esta vista junto con su traducción. Están almacenadas en la memoria del teléfono y en una futura versión de la app, se almacenarán en una base de datos. En esta versión de la aplicación, solo se almacena el último grupo de palabras que guardó el usuario.

Settings

Esta vista se limita únicamente a la configuración de la dirección IP del ordenador, la contraseña para la interfaz web del reproductor VLC y el sistema operativo. Se almacena en la memoria del teléfono, por lo que solo es necesario la configuración en su primer uso o cuando se ejecute en un ordenador diferente. Una vez escrita, el usuario deberá clicar en el botón de guardar para que surta efecto.

La vista recuerda al usuario de dónde puede obtener la dirección IP y que la contraseña ha de ser configurada en las preferencias del reproductor VLC.

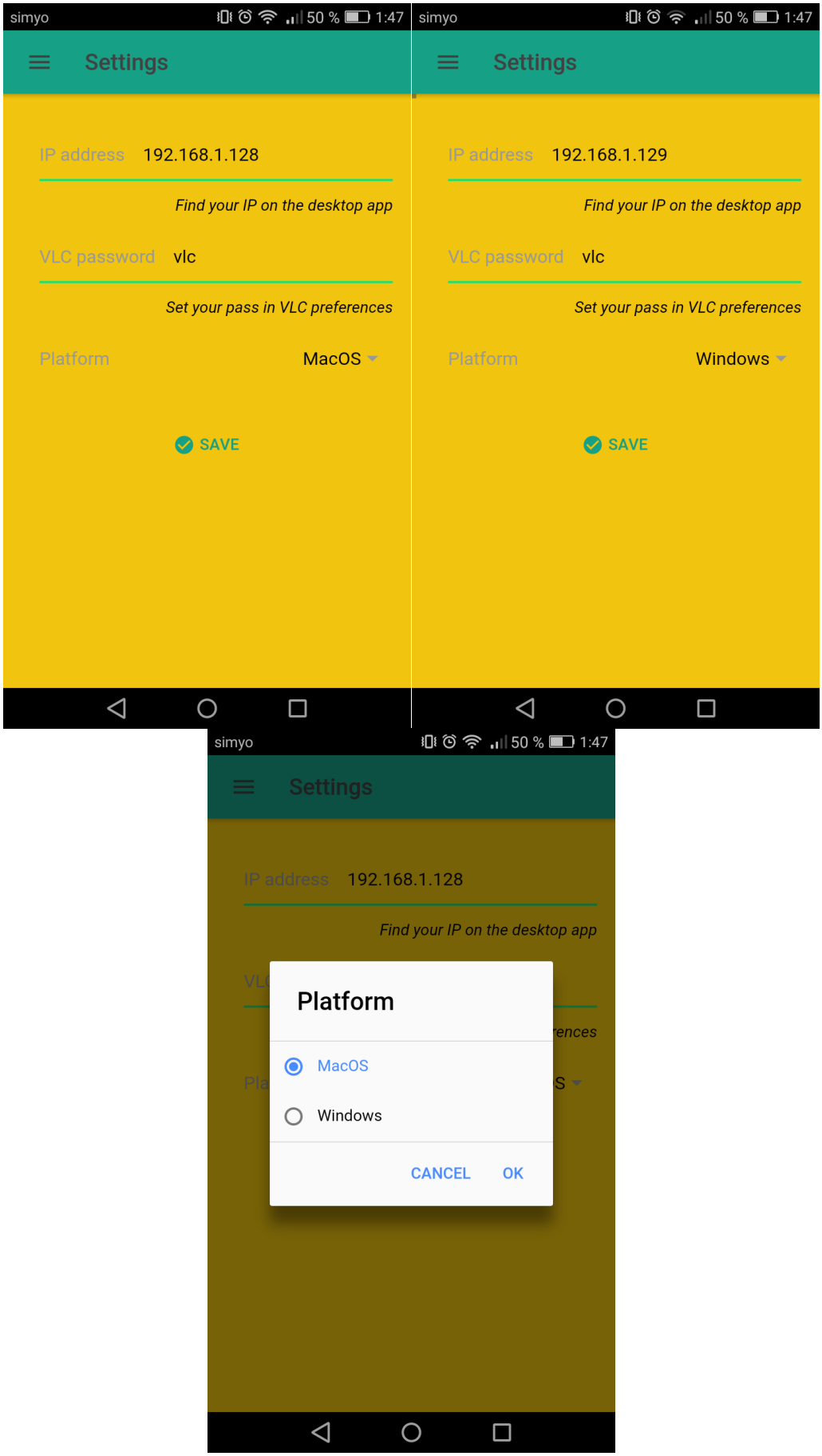
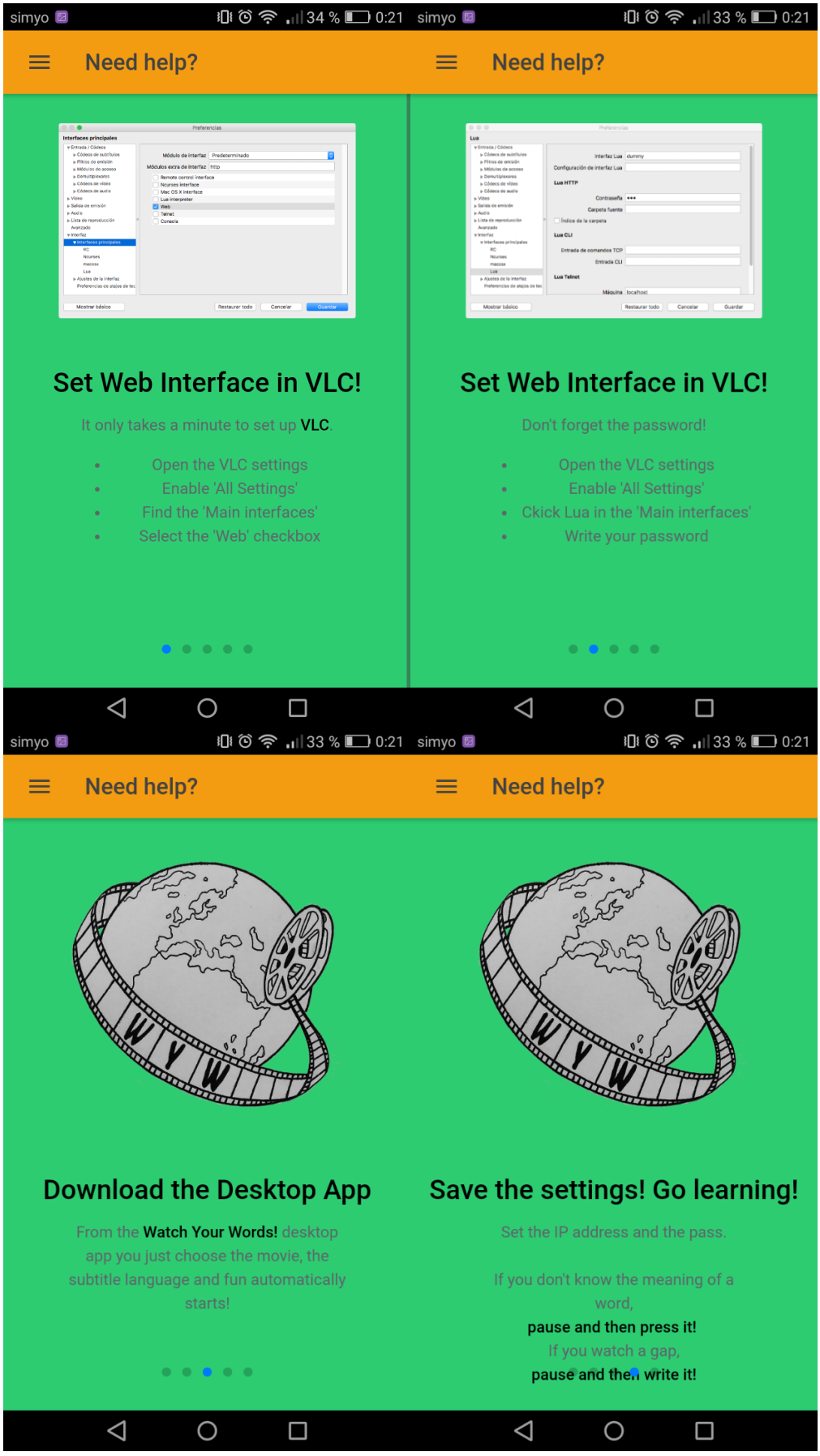


Ilustración 35: Vista de la configuración

Help?


simyo
34 % 0:21
Need help?
Set Web Interface in VLC!
It only takes a minute to set up VLC
Open the VLC settings
Enable 'All Settings'
Find the 'Main interfaces'
Select the 'Web' checkbox
simyo
33 % 0:21
Need help?
Set Web Interface in VLC!
Don't forget the password!
Open the VLC settings
Enable 'All Settings'
Ckick Lua in the 'Main interfaces'
Write your password
simyo
33 % 0:21
Need help?
Download the Desktop App
From the Watch Your Words! desktop app you just choose the movie, the subtitle language and fun automatically starts!
simyo
33 % 0:21
Need help?
Save the settings! Go learning!
Set the IP address and the pass.
If you don't know the meaning of a word,
pause and then press it!
If you watch a gap,
pause and then write it!

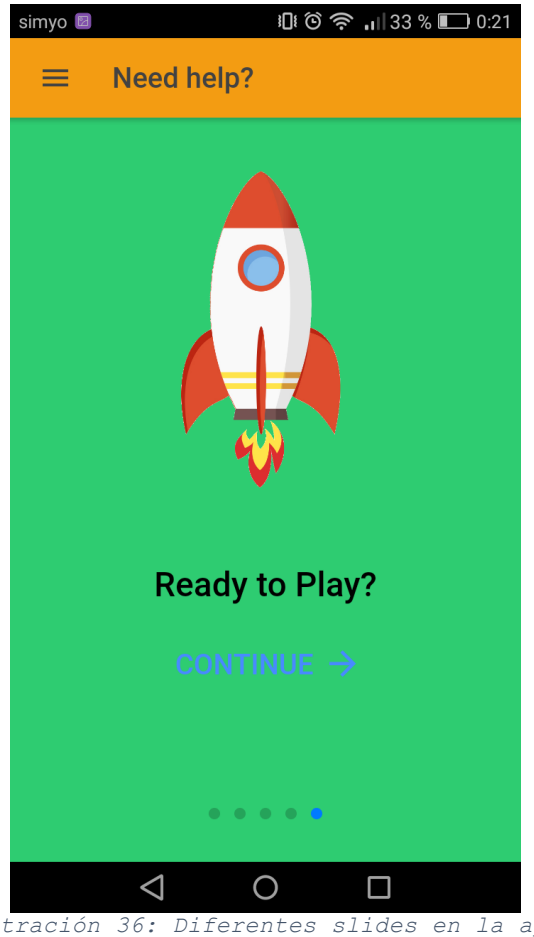


Ilustración 36: Diferentes slides en la ayuda

Al entrar aquí el usuario verá un slide en el que se muestra cómo y dónde ha de ser configurada la interfaz web del reproductor VLC y también el establecimiento de la contraseña. Después se muestra otro en el que se indica que hay que descargar la aplicación de escritorio y posteriormente elegir y tras configurar varios parámetros (idiomas y tiempo) comenzar a ver la película.

También se explican los momentos en los que el usuario debe pausar la reproducción del contenido y las opciones de las que dispondrá. Por último, una vez que haya revisado los pasos anteriores, podrá ir directamente a la vista "Go Learning" desde el último slide.

Splash screen e Icons

Cuando se comercializa la aplicación en los diferentes *markets* hay que crear un gran número de iconos y *splash screens* tanto para Android como iOS debido a los diferentes tamaños de los dispositivos.

Android tiene seis tamaños. En iOS diferencian hasta si la pantalla es retina para el mismo icono. Para esta tarea, Ionic proporciona un comando para la línea de comandos que puede generar todos estos iconos y splash screens simplemente proveyéndolo con una imagen en formato .png, .psd o .ai en el directorio /resources. Las imágenes deberán tener un tamaño mínimo para que se puedan crear todas las necesarias a partir de estas.

Manual de usuario para la app de escritorio

Ilustración 37: Diferentes fondos de la aplicación

La aplicación de escritorio del software "Watch Your Words!" se encontrará disponible para su descarga en la página web borja.pm. Es multiplataforma y se puede ejecutar en Windows, Linux y MacOS.

En un futuro se estudiará la viabilidad para introducirla a los diferentes markets de los sistemas operativos, para poder llegar a un mayor número de clientes de una forma más rápida y eficaz.

Tener que descargar una app para el escritorio a mayores de la app móvil, puede generar aversión por parte del usuario final. Para aminorar esto, en el diseño de dicha aplicación se ha seguido una filosofía de diseño que persigue el minimalismo y que una vez instalada este software, al usuario le resulte casi indiferente abrir el contenido multimedia desde el reproductor predefinido que usando esta app.

Desde el momento en que se abre, hasta que el usuario visualiza la película, el proceso es sumamente intuitivo. El programa va guiando al usuario a través de la configuración mínima que se necesita para que disfrute posteriormente de la película conjunta a la aplicación móvil. La app móvil será capaz de sincronizarse con el reproductor y usará la configuración introducida anteriormente de manera transparente al usuario.

Nada más abrir la aplicación, el usuario verá el nombre del software con un único botón debajo y como fondo tendrá una vista aleatoria de diferentes ciudades del mundo. Detrás de esto, mi idea es transmitir un concepto de universalidad a un usuario que desea mejorar sus conocimientos en idiomas de manera voluntaria.

El checkbox hará un filtrado por el tipo de codificación de los subtítulos que devuelva la API de OpenSubtitles. Con este checkbox activado, solo se mostrarán los subtítulos que estén codificados en UTF-8. Se avisa al usuario de que la experiencia será mejor y es que el software será capaz de decodificar bien los caracteres y no aparecerán símbolos de interrogación en lugar del carácter especial que sea. Esto ocurre con letras acentuadas o caracteres propios de los idiomas que difieran del alfabeto tradicional. Aun así, al limitar bastante los subtítulos, se da la opción de que se puedan escoger otros subtítulos con otra codificación.

Al pulsar sobre el botón "Learn", irán apareciendo las opciones de configuración necesarias una vez se haya rellenado la anterior. Las opciones necesarias son:

- Escoger el fichero de vídeo que se va a visualizar.

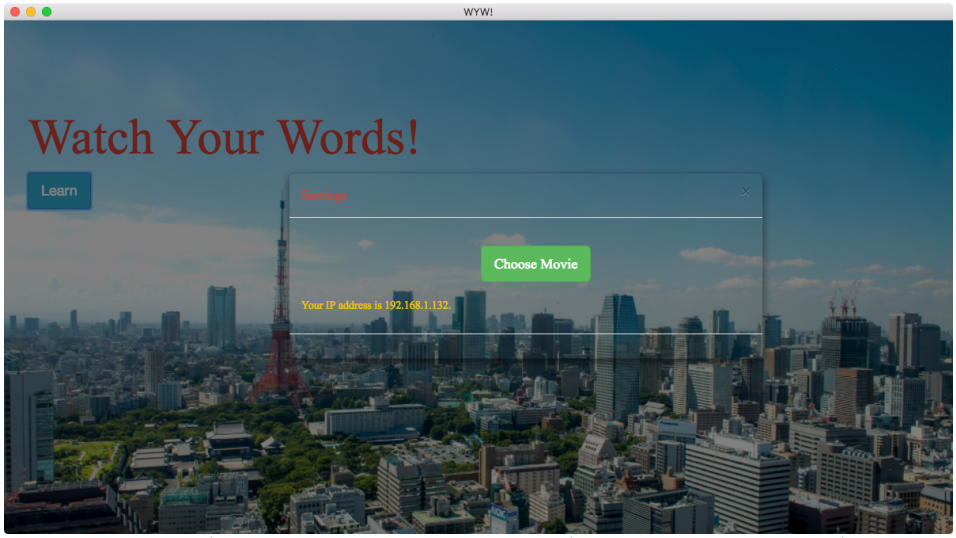


Ilustración 38: Ventana con el botón para elegir el vídeo

- Escoger el idioma del subtítulo que está disponible para ese fichero en concreto. Si no se encuentra ninguno de manera automática, el usuario verá una alerta indicándole que cambie el fichero.

- Escoger el idioma en el que se quiere obtener la traducción de las palabras que el usuario no entienda durante el visionado de la película.

- Escoger el intervalo de tiempo en el que el software propondrá los desafíos de manera cíclica.

Una vez estén cumplimentados estos campos, se mostrará en esa misma ventana "modal" la dirección IP local del ordenador del usuario. Este es el único dato que necesitará posteriormente la app móvil y que en principio solo será necesario que rellene una vez.

Ilustración 39: Ventana con el botón para comenzar la reproducción

Finalizado este proceso, cuando se pulse el botón "start watching" se abrirá de manera automática el reproductor multimedia VLC con el contenido que el usuario seleccionó previamente y los subtítulos (ya modificados).

Requisitos del sistema

Para la aplicación de escritorio será necesario:

- Sistema operativo: MacOS o Windows.
- Desactivar el firewall
- Conexión a internet.
- Al menos 300mb de espacio disponible en el disco duro para su uso.

Para la aplicación móvil será necesario:

- Sistema operativo: Android versión de nivel API 16 o superior.
- Conexión a internet.
- Esta conexión tendrá que ser a la misma red a la que esté conectado el ordenador que ejecute la aplicación de escritorio.
- Al menos 10mb disponibles en la memoria para su instalación.
- La aplicación no necesitará de ningún permiso adicional (acceso a cámara, contactos, acelerómetro, etc.).

13. Modelo de negocio Canvas

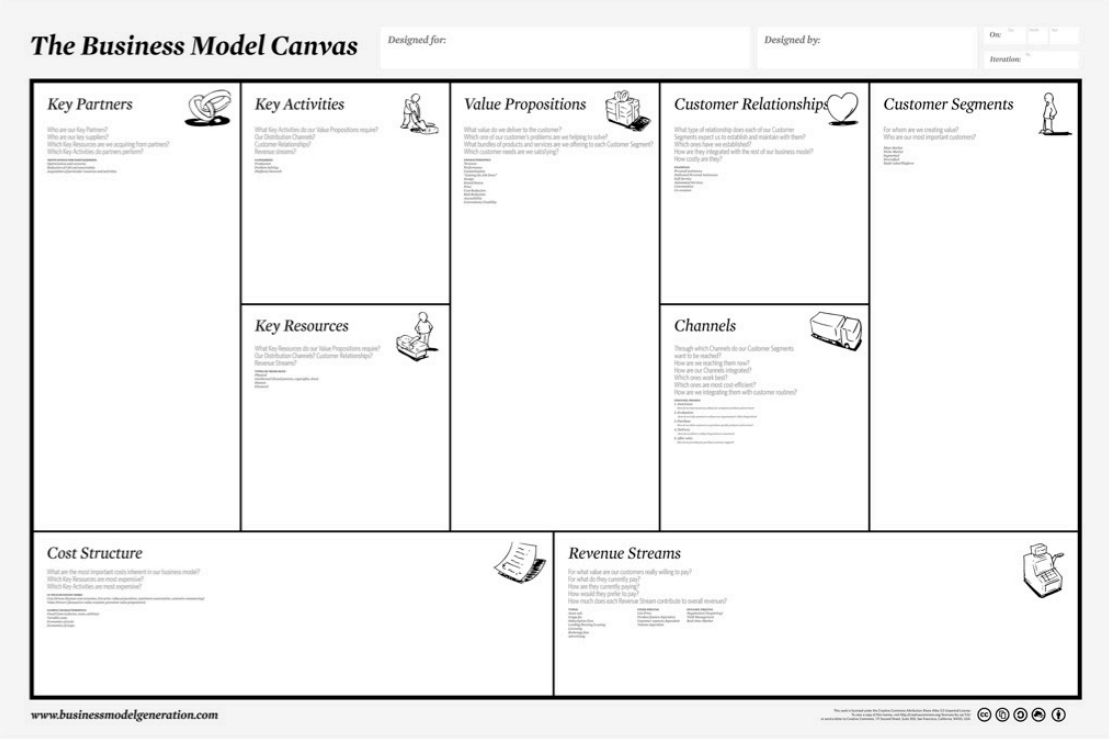


Ilustración 40: Plantilla para el modelo de negocio Canvas

Canvas define el modelo de negocio de una organización con nueve bloques. Ayuda a discutir, mapear, diseñar e inventar nuevos modelos de negocio. Un error muy frecuente a la hora de construir un negocio es que el proyecto nace enfocado en el producto y no en el cliente, haciendo que una idea que en la mente del creador sea maravillosa no tenga ningún interés comercial.

Para que una idea o proyecto sea válida, es importante que solucione un problema. Pero debe ser un problema real que tenga aquel cliente que va a pagar por el producto.

Los bloques del modelo Canvas son los siguientes:

- Segmento de clientes: Son el centro de cualquier modelo. Sin clientes no hay negocio. Las personas u organizaciones para las que crea valor, incluyendo usuarios y clientes que pagan.

- Propuesta de valor para cada segmento: Lo que te diferencia de la competencia y que será capaz de generar ingresos.

- Canales para llegar a los clientes: Incluye venta y postventa. Tanto como cómo vamos a explicar a los clientes el valor del proyecto como el cómo vamos a ofrecérsela posteriormente.

- Relaciones con los clientes: Las relaciones que se establecen con cada uno de los segmentos de clientes. Pueden ser personales, automatizadas, individuales o colectivas…

- Flujos de ingreso que genera: Consecuencia de todo lo demás, pero que debes saber desde un primer momento. Aunque podrán ir evolucionando y cambiando, como el modelo de negocio.

- Recursos clave que requiere para crear valor: De qué manera se hará la propuesta de valor y con qué medios (humanos, tecnológicos, físicos) se cuenta. Hay que tener cuidado con esto, pues un negocio será viable o no, dependiendo de los recursos empleados.

- Actividades clave que requiere para crear valor: Define lo que es el producto o lo que quiere ser. Posiblemente sea lo más complicado de definir dentro del modelo de negocio.

- Aliados clave: Las asociaciones entre terceros para colaboración, compartir experiencias, costes o recursos…

- Estructura de costos del modelo de negocio: Costes que tiene la empresa que pueden ser calculados después de analizar las actividades, recursos y asociaciones clave.

14. Presupuesto y material usado

Existen numerosos factores a la hora de calcular el precio de desarrollo de una aplicación móvil. La complejidad de la app, las funcionalidades que tiene que cumplir, si hace uso de bases de datos, si permite login al usuario… Todo esto hace que el proceso de desarrollo sea más largo y por ello resulte en una aplicación más cara.

Las diversas plataformas en las que estará disponible pueden duplicar equipos -y por tanto presupuesto- si no se aborda con una visión de desarrollo híbrida, en la que solo es necesario un equipo para el crearla y posteriormente mantenerla.

No voy a entrar en datos económicos totales, pues serían mayormente suposiciones nada seguras. Hay calculadoras que te dan una aproximación rápida, como Yeeply, pero creo que sería necesario un estudio más exhaustivo que un cuestionario de diez preguntas.

Se ha llevado a cabo un desarrollo del software individual. Esto, hoy en día no suele ser lo común. En general, el trabajo en equipo permite alcanzar un trabajo de mayor calidad en menos tiempo, en el cual, cada parte del equipo se especializa y centra en ciertos apartados, aunque siempre puedan ayudarse entre ellos.

Hay algunas cosas que son necesarias. Hay una diferencia grande entre lanzar aplicaciones para Android o para iOS. Mientras Android pide una licencia que cuesta $25 y no es perecedera, iOS requiere que cada año se renueve la licencia de desarrollador, y cuesta $99. También el hardware es importante, pues se puede desarrollar para Android sin importar el sistema operativo en el que estés (Windows, Linux, MacOS) pero, por el contrario, para desarrollar en iOS es necesario disponer de MacOS y por ello de un ordenador de la marca Apple.

El desarrollo mayoritario de esta aplicación se llevó a cabo en un Mac Mini. Para tareas más pequeñas, mera documentación o porque la situación lo requería, se ha utilizado un ordenador portátil con Windows. Sobre este portátil también se han ido probando las diferentes modificaciones que habría que hacer para que la aplicación de escritorio desarrollada en Electron se pueda distribuir en Windows.

Durante la evolución de este proyecto, se ha adquirido la licencia de desarrollador Android por un valor de $25.

Se ha intentado en todo momento el uso de software open-source y gratuito. Los IDEs empleados han sido Atom y Sublime Text 2. Ambos open-source. Pese a que Sublime Text 2 no es gratuito, tiene una licencia de prueba ilimitada que recuerda al desarrollador que adquiera la licencia de pago de vez en cuando. Todas las tecnologías empleadas, Electron, Apache Cordova, el framework Ionic, Node.JS, Angular, las diferentes bases de datos probadas, NPM, son también software libre bajo diferentes licencias. Para la edición del logotipo de la aplicación y otras imágenes se ha usado GIMP, programa con funcionalidades y características similares a Adobe Photoshop, pero GIMP con una filosofía completamente open-source.

15. Conclusiones y líneas futuras

En este apartado me gustaría dar una visión más personal de lo que me ha supuesto llevar a cabo este proyecto. Desde la misma fase de documentación para estudiar las diferentes formas de implementar la idea, hasta este mismo momento, en el que último el desarrollo tanto de la aplicación como de este documento.

Cuando propuse realizar este TFG lo hacía con mucha ilusión, pero con una idea no muy cerrada. A medida que me he ido encontrando diferentes dificultades he tenido que ir variando y adecuando a algo que pudiese realizar. El objetivo de afrontar estas dificultades con soluciones y tecnologías actuales ha sido primordial. He aprendido los principios básicos de varias tecnologías con muy poco tiempo de vida, que están en auge con una progresión asombrosa en número de usuarios y que presiento que voy a continuar usando en mi próxima etapa ya enfocada al mercado laboral.

Recuerdo que de las asignaturas que más satisfacción me han aportado a lo largo de la carrera se encuentran las relacionadas con la programación y es por el mero hecho de sentir y ver que en lo que has invertido tiempo y esfuerzo finalmente se convierte en algo tangible y que -con suerte- funciona. En el plan de estudios no se ha ofrecido ninguna asignatura enfocada al desarrollo de aplicaciones móviles, pero siendo este un tema que me interesa me pareció una buena opción que el TFG me aportase la base de unos conocimientos que se me hacían a priori interesantes y he podido comprobar que efectivamente sí lo son.

Durante el proceso de creación de la app, he visto como una idea simple se ha convertido en innumerables quebraderos de cabeza. Y como esos quebraderos de cabeza que pensaba que eran insalvables se han terminado convirtiendo en un producto final, a disposición de todo aquel que crea útil la idea inicial.

Integrar en el software servicios de terceros tiene evidentes ventajas ya que se consiguen funcionalidades que no se tendrían por si solas o el acceso a una gran cantidad datos que sería imposible recopilarlos uno solo. En el caso de este proyecto, el software se apoya en una base de datos de subtítulos que han sido aportados y creados manualmente por una comunidad de usuarios de manera altruista. Solo cabe por tanto agradecer la involucración de esta comunidad. Pero hay que tener en cuenta que es una fuente de posibles errores. También se crea una dependencia sobre estos servicios de terceros que pueden afectar gravemente al software ya que en caso de que se deje de ofrecer o incluso mantener estos servicios, posiblemente el software que depende de ellos también se tenga que dejar de ofrecer.

Se ha explicado anteriormente el modelo Canvas y aplicarlo es lo que tengo en mente. Últimamente he tenido la oportunidad de asistir a varios relacionados con el mundo de las StartUps, de emprendedores con pequeñas ideas –con fuerte carácter tecnológico- y creo haber aprendido lecciones a la hora de lanzar una aplicación. La capacidad de pivotaje es posiblemente una de las lecciones más importantes. Lo que ideé como una app para mí, me gustaría intentar validarla en un ambiente más educativo. Intentar saber si puede tener cierta salida en aulas de idiomas.

No todo es el desarrollo de software al crear una app. No se pueden dejar a un lado factores claves como son el diseño o el marketing. En las primeras etapas de desarrollo quizás se sienta más necesario el trabajo del desarrollador, pero cuando el esqueleto de la aplicación empieza a tener las funcionalidades deseadas hay que dotar a esa app de

un diseño que haga al usuario volver a usarlo, unos logotipos que llamen la atención y creen marca e identifiquen el producto. Una vez que el código se presenta de forma adecuada en una aplicación, el siguiente paso es decir al mundo (o al menos al segmento del mercado que potencialmente la usarán) que existe. Que existe y que les ayudará a resolver problemas que actualmente tienen. Ahí es donde entra el marketing y la estrategia que empleará para dar a conocer la app.

Gracias al párrafo previo se ha llegado a la conclusión de que una idea no vale nada. Seguramente sea la conclusión que menos esperaba encontrarme al comienzo del desarrollo de este trabajo. Aunque suene pesimista, bajo mi punto de vista es cierto. No servirá de mucho un software que resuelva los problemas del cliente si el usuario no es capaz de usarlo porque no se ha dedicado tiempo y dinero necesario a crear una interfaz sencilla y atractiva. Tampoco será suficiente haber resuelto lo anterior, tener un software que cumpla todos los requisitos y que el usuario sea capaz de manejarlo sin dificultad si el cliente no sabe que existe ese software. Es por esto por lo que creo haber aprendido que lo más importante no es la idea, sino la capacidad de llevarla a cabo. Encontrar un equipo en el que los miembros se sientan cómodos y su trabajo sea complementario con el fin conjunto del desarrollo y distribución del producto.

Líneas futuras

Es mi intención que el trabajo desarrollado durante estos meses no se detenga aquí.

A lo largo de este documento se ha estado hablando de aplicaciones multiplataforma, pero para la realización del MVP se ha optado por crear estas aplicaciones para el hardware en el que han sido desarrolladas: aplicación de escritorio para MacOS y aplicación móvil para Android. Por ello el siguiente paso natural es aprovechar la ventaja por el cual se han escogido estas tecnologías y crear la aplicación de escritorio para Windows y Linux y la aplicación móvil para iOS y posiblemente Windows Phone.

Antes de incluir muchas más funcionalidades, centrarse en un mejor manejo de los errores. Se ha hablado anteriormente de que la aplicación depende por completo de los subtítulos que altruistamente se suben a la base de datos de OpenSubtitles.org. Los ficheros de subtítulo .str deben seguir un cierto formato. De no seguir el formato establecido, el software desarrollado no será capaz de parsearlos y por ello no logrará crear los ficheros que usará la aplicación móvil.

Mejorar el servicio de traducción. Pues actualmente se utiliza los servicios que provee la API de Google Translator. Estos son muy amplios y el abanico de idiomas que ofrecen es inmejorable, pero a nivel personal, hubiese preferido los servicios que ofrece la API de WordReference. Como ya se explicó, no ofrecen más keys para su API, pero una opción sería dedicar tiempo y programar alguna función que parsee su página web u otra quizás más viable, intentar buscar a alguien que venda su key. También sería interesante ir ofreciendo mejores progresivas en los idiomas más utilizados, y que, en lugar de una traducción directa, la aplicación ofreciese a mayores sinónimos o una definición en esa misma lengua.

Incorporar nuevos juegos y desafíos. Una vez el programa sea capaz de ofrecer sinónimos para las palabras que desee el usuario, se podrían también emplear para algunas pequeñas pruebas. Por ejemplo: periódicamente se le ofrece al usuario de una palabra que ha aparecido

en el subtítulo, tres palabras, de las cuales solo una será sinónima. Será esta la palabra que tenga que elegir para poder continuar con el visionado del contenido multimedia.

Mejorar la gestión de la base de datos en la cual se almacenan las palabras que busca el usuario será también una prioridad, pues una idea futura es dotar a la aplicación de escritorio de una cierta inteligencia y que las palabras que elimine no sean meramente palabras al azar con más de un determinado número de caracteres, sino que realice una búsqueda de estas palabras que el usuario ha buscado y las elimine del subtítulo con la intención de que sean recordadas. Para esto sería necesario que la base de datos del teléfono se sincronice con la del servidor (en este caso app de escritorio), por lo que se haría uso de PouchDB y CouchDB, que ya han sido analizadas.

En las primeras etapas de desarrollo se estudió la posibilidad de integrarlo con plataformas VOD como por ejemplo Netflix. Esta opción que no fue viable por exceder en dificultad lo adecuado a un Trabajo Fin de Grado, se me hace la única forma posible de dotar a Watch Your Words de una verdadera utilidad, al disponer fácilmente de un amplio catálogo de contenido multimedia con varias opciones de idiomas, tanto en el audio como en los subtítulos. Esto supondría una carga importante de trabajo, pero también la culminación completa de este proyecto.

15. Bibliografía

- ORNBO, GEORGE. Node.JS. 1ª ed. Madrid: Grupo Anaya, S.A. 2013

- GRIFFITH, CRHIS. Mobile App Development with Ionic 2 – Cross-platform apps with Ionic, Angular & Cordova. 1ª ed. Sebastopol: O'Reilly Media, Inc. 2017

- JUDE, JOSEPH & JUSTIN, JOYCE. Learn Ionic 2 – Develop Multi-platform Mobile Apps. 1ª ed. New York: Springer Science+Business Media Finance Inc. 2017

- Nolan, Ashley (2016) The State of Front-End Tooling 2016 – Results. Recuperado en Agosto de 2017, de https://ashleynolan.co.uk/blog/frontend-tooling-survey-2016-results

- Jackson, Brian (2017) CSS Preprocessors – Sass vs LESS. Recuperado en Agosto de 2017, de https://www.keycdn.com/blog/sass-vs-less/

- Bradley, Steven (2012) DRY CSS: Don't Repeat Your CSS. Recuperado en Agosto de 2017, de http://vanseodesign.com/css/dry-principles/

- Mozilla Foundation (2017) About JavaScript. Recuperado en Agosto de 2017, de https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

- Crockford, Douglas (2001) JavaScript: The World's Most Misunderstood Programming Language. Recuperado en Agosto de 2017, de http://javascript.crockford.com/javascript.html

- StackOverflow (2017) Prototype based vs. class based inheritance. Recuperado en Agosto de 2017, de https://stackoverflow.com/questions/816071/prototype-based-vs-class-based-inheritance

- NPM (2017) opensubtitles-api. Recuperado en Agosto de 2017, de https://www.npmjs.com/package/opensubtitles-api

- OpenSubtitles.org (2017) OSDb: Open Subtitles DataBase. Recuperado en Agosto de 2017, de http://trac.opensubtitles.org/projects/opensubtitles/wiki/OSDb

- OpenSubtitles.org (2017) Read First for Developers. Recuperado en Agosto de 2017, de http://trac.opensubtitles.org/projects/opensubtitles/wiki/DevReadFirst

- WordReference (2017) WordReference API. Recuperado en Agosto de 2017, de http://www.wordreference.com/docs/api.aspx

- Ionic Team (2016) Learn Angular 2. Recuperado en Agosto de 2017, de http://learnangular2.com/

- Angular (2017) Architecture Overview. Recuperado en Agosto de 2017, de https://angular.io/guide/architecture

- Wikipedia (2017) List of ISSO 639-2 codes. Recuperado en Agosto de 2017, de https://en.wikipedia.org/wiki/List_of_ISO_639-2_codes

- TypeScript (2017) Documentation. Recuperado en Agosto de 2017, de http://www.typescriptlang.org/docs/home.html

- GitHub (2016) TypeScript Language Specification. Recuperado en Agosto de 2017, de https://github.com/Microsoft/TypeScript/blob/master/doc/spec.md

- Apache Cordova (2017) Documentation: Overview. Recuperado en Agosto de 2017, de https://cordova.apache.org/docs/en/latest/guide/overview/index.html

- StackOverflow (2012) How do Adobe Phonegap and Apache Cordova differ?. Recuperado en Agosto de 2017, de https://stackoverflow.com/questions/12318424/how-do-adobe-phonegap-and-apache-cordova-differ

- Morony, Josh (2017) CouchDB, PouchDB, and Ionic 2: Why PouchDB?. Recuperado en Agosto de 2017, de https://www.joshmorony.com/couchdb-pouchdb-and-ionic-2-why-pouchdb/

- PouchDB (2017) Introduction to PouchDB. Recuperado en Agosto de 2017, de https://pouchdb.com/

- Apache CouchDB (2017) Apache CouchDB 2.1 Documentation. Recuperado en Agosto de 2017, de http://couchdb.apache.org/

- Morony, Josh (2017) CouchDB, PouchDB, and Ionic 2: Starting a New Project. Recuperado en Agosto de 2017, de https://www.joshmorony.com/couchdb-pouchdb-and-ionic-2-starting-a-new-project/

- Apache Cordova (2017) Customize Icons. Recuperado en Agosto de 2017, de https://cordova.apache.org/docs/en/latest/config_ref/images.html

- Apache Cordova (2017) Reference Plugin APIs Splashscreen. Recuperado en Agosto de 2017, de https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-splashscreen/

- Escudero, Javier (2016) Emprendedores. Cómo se elabora um modelo Canvas. Recuperado en Agosto de 2017, de http://www.emprendedores.es/gestion/modelo-3

- Innokabi (2014) Modelo Canvas explicado paso a paso y com ejemplos. Recuperado en Agosto de 2017, de http://innokabi.com/canvas-de-modelo-de-negocio/

- Ditrendia (2016) Informe ditrendia 2016: Mobile en España y en el Mundo. Recuperado en Agosto de 2017, de http://www.amic.media/media/files/file_352_1050.pdf

- Sanz, Elena (2015) Muy Interesante: ¿Cuántos idiomas se hablan en el mundo? Recuperado en Agosto de 2017, de

https://www.muyinteresante.es/cultura/arte-cultura/articulo/icuantos-idiomas-se-hablan-en-el-mundo