

UNIVERSIDAD DE ALMERÍA

Escuela Politécnica Superior y Facultad de Ciencias Experimentales

Ingeniería Informática



Desarrollo de una aplicación de entrenamiento Mindfulness para plataformas móviles con PhoneGap

Autora: María del Carmen Castillo Soler

Directora: Consolación Gil Montoya

Almería, Abril 2014

Índice

1.	Introducción	8
1.1.	Contenido de la memoria	8
1.2.	Motivación del proyecto.....	9
1.3.	Objetivos	10
1.4.	Metodología	11
1.5.	Planificación	12
2.	Estado del arte	13
2.1.	Sistemas operativos para dispositivos móviles.....	13
2.2.	Taxonomía de aplicaciones móviles	15
a.	Aplicaciones nativas.....	15
b.	Aplicaciones web	15
c.	Aplicaciones híbridas	16
2.3.	Frameworks para aplicaciones híbridas	17
a.	Adobe Air	17
b.	Appcelerator Titanium	17
c.	Qt	18
3.	PhoneGap.....	19
3.1.	API.....	19
3.2.	Instalación	20
3.3.	Plugins.....	22
3.4.	PhoneGap Build	24

4.	Tecnologías subyacentes	26
4.1.	Html5	26
a.	Evolución de HTML	26
b.	Novedades	28
c.	Web semántica	46
4.2.	CSS3	52
a.	Animaciones	52
b.	Bordes redondeados	55
c.	Efectos de sombra	56
d.	Múltiples columnas	57
e.	Pseudoclasas	58
f.	Tipografías	60
4.3.	JavaScript	61
a.	JavaScript y la especificación ECMA	62
b.	Sintaxis y utilización de JavaScript	63
c.	Lenguajes basados en Clases vs. Lenguajes basados en Prototipos	64
d.	JQuery	68
5.	Técnicas de diseño y desarrollo	73
5.1.	Responsive Web Design	73
5.1.1.	Diseño fluido	74
5.1.2.	Imágenes flexibles	76
5.1.3.	Uso de Media Queries	77

5.1.4.	Limitaciones del Responsive Web Design	80
6.	Herramientas	81
6.1.	GitHub.....	81
6.2.	ScrumDo.....	81
6.3.	Eclipse IDE y Android SDK	82
6.4.	Microsoft OneNote.....	83
7.	Desarrollo de la aplicación.....	85
7.1.	Análisis	85
a.	Descomposición de tareas	87
a.	Realización de diagramas	89
7.2.	Diseño	96
a.	Diseño de la interfaz	96
b.	Adaptación a los distintos dispositivos con RWD.....	107
7.3.	Implementación	110
8.	Conclusiones y trabajo futuro.....	121
9.	Referencias y bibliografía	123
10.	Anexos	126
10.1.	Proceso de 21 días. Sergi Torres.....	126

Índice de figuras

Figura 1: Cuota de mercado de los SOs móviles	14
Figura 2: PhoneGap Build	24
Figura 3: ConfiGAP	25
Figura 4: Barra de progreso determinada	31
Figura 5: Barra de progreso indeterminada	31
Figura 6: Vista del elemento datalist en Chrome	35
Figura 7: Vista del atributo placeholder en Chrome	36
Figura 8: Vista del input de tipo search en Chrome	37
Figura 9: Vista de diferentes teclados según el tipo en iPhone	37
Figura 10: Vista del input de tipo range en Chrome	38
Figura 11: Vista del input de tipo number en Chrome	38
Figura 12: Vista de inputs para fecha y hora en Chrome	39
Figura 13: Vista del input de tipo color en Chrome	39
Figura 14: Layout previo a HTML5	47
Figura 15: Nuevas etiquetas semánticas de HTML5	47
Figura 16: Border-radius CSS3	55
Figura 17: Text-shadow CSS3	57
Figura 18: Creación de objetos con clases	65
Figura 19: Creación de objetos con prototipos	66
Figura 20: Regla para las medidas relativas con RWD	75
Figura 21: Diseño fluido con RWD	76

Figura 22: Media Queries convencionales.....	77
Figura 23: Gráfico de tipo stacked de ScrumDo.....	82
Figura 24: Diagrama de casos de uso	90
Figura 25: Diagrama de estados	92
Figura 26: Diagrama de estados del menú	93
Figura 27: Diagrama de estados de los ejercicios	94
Figura 28: Diagrama de estados de las alarmas.....	94
Figura 29: Diagrama de estados de la libreta	95
Figura 30: Diagrama de estados de las opciones	95
Figura 31: Interfaz de la ventana de inicio.....	97
Figura 32: Interfaz del menu	98
Figura 33: Interfaz de la ventana de introducción	99
Figura 34: Interfaz de la selección de ejercicios	100
Figura 35: Interfaz de un ejercicio.....	101
Figura 36: Interfaz de la gestión de alarmas.....	102
Figura 37: Crear nueva alarma	103
Figura 38: Interfaz de la gestión de la libreta	104
Figura 39: Interfaz de las opciones.....	105
Figura 40: Interfaz para cambiar el tono de alarma.....	106
Figura 41: Ventana de créditos	107
Figura 42: Menú en posición vertical	108
Figura 43: Menú en posición horizontal	108

Figura 44: Interfaz del ejercicio en vertical.....	109
Figura 45: Interfaz del ejercicio en horizontal	109
Figura 46: Añadir una nueva nota	114
Figura 47: Notificación de la alarma.....	116
Figura 48: Cambiar el tono de las alarmas	119

Índice de tablas

Tabla 1: Planificación del proyecto	12
Tabla 2: Formatos de audio soportados por los navegadores.....	29
Tabla 3: Formatos de vídeo soportados por los navegadores.....	32
Tabla 4: Propiedades de las media queries	79
Tabla 5: Historias del epic 1	87
Tabla 6: Historias del epic 2	87
Tabla 7: Historias del epic 3-1	88
Tabla 8: Historias del epic 3-2	89

1. Introducción

El presente documento refleja el desarrollo de una aplicación híbrida para dispositivos móviles realizada utilizando el framework PhoneGap, así como otras tecnologías y estrategias que han hecho posible su correcto funcionamiento cuidando tanto su implementación como su diseño.

También se muestra un resumen del estudio de los diferentes aspectos necesarios para el desarrollo de la aplicación (tecnologías, técnicas, herramientas, etc.), introduciendo de manera algo más teórica (aunque siempre aplicada a este proyecto) la información necesaria para su implementación.

1.1. Contenido de la memoria

El contenido de esta memoria se divide en diez puntos principales con el objetivo de hacer cómoda e intuitiva la lectura del documento. Dichos diez puntos son:

1. **Introducción:** En el actual punto del documento se desglosa el contenido de la memoria, justificando las motivaciones y comentando los objetivos principales que se desean abarcar con este proyecto, así como la planificación diseñada para alcanzar dichos objetivos.
2. **Estado del arte:** En este apartado se comentará el estado actual del mundo de los dispositivos móviles, repasando los principales sistemas operativos que utilizan. También se hablará de los diferentes tipos de aplicaciones en función de las plataformas para las que se desarrollan, comparando sus características principales. Por último, dado que el objetivo de este proyecto es desarrollar una aplicación híbrida, se estudiarán algunos frameworks, similares al utilizado, que permiten este fin.
3. **PhoneGap:** En este punto se explicarán las principales características del framework utilizado en el desarrollo del proyecto, comentando su instalación y el proceso de creación de una aplicación con PhoneGap.
4. **Tecnologías subyacentes:** En la implementación de una aplicación híbrida intervienen tecnologías web, tales como HTML5, CSS3 y JavaScript. En este apartado se resume el estudio de cada una de ellas realizado previamente a la implementación del proyecto.
5. **Técnicas de diseño y desarrollo:** Una de las técnicas utilizadas para el diseño de esta aplicación es el Responsive Web Design. En este punto se explicará qué es, en qué consiste y cómo aplicarlo.

6. **Herramientas:** En este punto se resumen las principales herramientas utilizadas durante el desarrollo del proyecto, tanto para su implementación como para su seguimiento.
7. **Desarrollo de la aplicación:** Este apartado resume el proceso de creación de la aplicación, mostrando el análisis, el diseño y la implementación de la misma, así como los aspectos más relevantes con los que se ha tenido que lidiar durante su desarrollo.
8. **Conclusiones y trabajo futuro:** Una vez descrito el proceso de desarrollo de este proyecto se comentarán las conclusiones obtenidas, y se propondrán futuras vías de continuación del trabajo realizado.
9. **Referencias y bibliografía:** Este apartado recoge las referencias que han ayudado a la elaboración del proyecto así como del presente documento.
10. **Anexos:** En el último punto de esta memoria se incluye el texto que da lugar al contenido de la aplicación: el Proceso de 21 días del autor Sergi Torres.

1.2. Motivación del proyecto

Los dispositivos móviles han experimentado en los últimos años un aumento exponencial en su demanda, convirtiéndose en elementos comunes e indispensables en el día a día de cada vez más personas. Según un informe realizado por Cisco[4], la demanda de nuevas conexiones móviles por parte de los consumidores va en aumento y no hay signos de que vaya a cambiar esta tendencia. Para el año 2017, Cisco prevé que existirán más de 10.000 millones de dispositivos móviles en el mundo.

Asociado a este hecho está también el del crecimiento de los mercados de aplicaciones móviles, que no hace más que aumentar las expectativas de los usuarios, que cada vez buscan una mayor cantidad y diversidad de contenidos en sus dispositivos. Pese a la cantidad de aplicaciones ya existentes, la facilidad de desarrollo por parte de cualquier usuario con los conocimientos necesarios hace posible seguir contribuyendo a estos nuevos mercados virtuales.

No obstante, uno de los problemas que se plantean a la hora de desarrollar una aplicación es el asociado a cómo implementarla. Si se toma la decisión de centrarse en una plataforma concreta (como Android o iOS), se reduce la difusión de la aplicación, que puede acceder a un menor público objetivo. Otra opción es desarrollar una aplicación multiplataforma, con lo que pueda llegar a cualquier usuario, aunque presente ciertas desventajas con respecto a las aplicaciones nativas.

La principal motivación de este proyecto consiste en desarrollar una aplicación que sea compatible con cualquier dispositivo móvil, independientemente de su sistema operativo o sus características físicas, con el fin de que pueda ser utilizada por el máximo número de usuarios posible.

1.3. Objetivos

El presente proyecto busca conocer y comprender las características y el funcionamiento del framework Apache Cordova, también conocido como PhoneGap, para la creación de aplicaciones multiplataforma; averiguando sus posibilidades y ventajas frente a otras alternativas. Además, abarca el desarrollo completo de una aplicación móvil de psicología basada en el “Proceso de 21 días” del autor Sergi Torres.

A través de su dispositivo móvil el usuario podrá disponer de todos los recursos necesarios para realizar los ejercicios de este proceso, sin preocuparse por el material, la planificación o el seguimiento. Será objetivo de la aplicación llevar un seguimiento del proceso, encargándose de gestionar estos aspectos automáticamente.

El uso de tecnologías multiplataforma para el desarrollo de aplicaciones móviles aporta grandes ventajas. Al tratarse de un mercado muy variable, no se puede garantizar que los fabricantes de sistemas operativos y dispositivos sigan siendo mayoritarios en distintos periodos de tiempo. El hecho de desarrollar aplicaciones que no dependan específicamente de una plataforma en concreto garantiza que los cambios de plataforma no sean problemáticos y acelera exponencialmente el desarrollo para varios dispositivos.

Para aprovechar las posibilidades del desarrollo de una aplicación híbrida o multiplataforma, se deberá hacer uso de PhoneGap como puente para utilizar los recursos propios del dispositivo.

Otro de los objetivos ligado al desarrollo de una aplicación de este tipo es el de estudiar y utilizar diferentes tecnologías de desarrollo web, como son HTML, CSS y JavaScript, ya que son además necesarias para el desarrollo con PhoneGap.

Por último, se estudiarán y se pondrán en práctica técnicas de diseño que permitan implementar una aplicación cómoda e intuitiva, cuidando la experiencia de usuario y haciéndola independiente del dispositivo en el que se ejecute, sin que éste suponga límites para su utilización.

1.4. Metodología

Para la realización de este proyecto se han combinado las técnicas de desarrollo tradicional del modelo en cascada con el modelo ágil Scrum, basado en un proceso iterativo e incremental.

El modelo en cascada, consistente en dividir el proyecto en varias etapas de manera que el inicio de cada una de ellas depende de la finalización de la etapa anterior, se ha utilizado para organizar el desarrollo del proyecto de manera global. No obstante, la implementación de la aplicación final se ha llevado a cabo siguiendo la metodología Scrum y sus técnicas de desarrollo ágil.

Las etapas relativas al proceso en cascada en las que se ha dividido el proyecto son las siguientes:

1. **Planificación del proyecto:** Esta tarea corresponde al plan de trabajo. En ella se determinan las tareas necesarias para alcanzar los objetivos del proyecto así como una estimación del esfuerzo y los recursos necesarios para realizar dichas tareas.
2. **Análisis y diseño del sistema:** Esta tarea comprende las siguientes actividades:
 - a) Una revisión de los requerimientos del proyecto para evaluar si son suficientes para comenzar el diseño e identificar los puntos abiertos que requieran una clarificación
 - b) La elaboración de un diseño de la aplicación por medio de diagramas UML de casos de uso y de secuencia, que se mostrarán más adelante.
 - c) Paralelamente a estas tareas, se lleva a cabo la instalación y configuración de las herramientas necesarias para el desarrollo del proyecto.
3. **Implementación de la aplicación:** Esta tarea comprende la parte central del proyecto, y se lleva a cabo manteniendo la metodología de trabajo ágil tipo Scrum, apoyada en la plataforma web ScrumDo.

La aplicación se divide en tareas sencillas que se reparten a lo largo del proceso de desarrollo según su prioridad formando iteraciones independientes. Estas tareas recogen la funcionalidad observada en las fases anteriores.

4. **Elaboración de la memoria del proyecto y presentación del mismo:** Esta tarea recoge la realización y entrega de todos los documentos asociados a este proyecto.

1.5. Planificación

Para la realización de este proyecto se han necesitado, aproximadamente, 900 horas, repartidas de la siguiente manera:

Tareas	Horas
1. Estudio e instalación de las herramientas necesarias	175
Documentación	100
Prueba de distintas herramientas	50
Elección y preparación del espacio de trabajo	25
2. Análisis y diseño inicial de la aplicación:	125
Descomposición en tareas	35
Realización de diagramas	50
Diseño de la interfaz	40
3. Implementación de la aplicación	430
Implementación de la interfaz	150
Implementación de la funcionalidad	150
Integración completa	50
Pruebas	80
4. Elaboración de la memoria	170
Total	900

Tabla 1: Planificación del proyecto

La etapa de documentación comenzó con la primera tarea, tal y como se muestra en la tabla 1, aunque ha seguido estando presente durante todo el proceso de desarrollo de este proyecto.

De la misma manera, dado que la implementación de la aplicación se ha llevado a cabo siguiendo una metodología ágil, con cada progreso obtenido en su desarrollo se realizaban las pruebas pertinentes para asegurar el correcto funcionamiento de cada avance. Las pruebas se han realizado de forma manual sobre varios dispositivos diferentes.

2. Estado del arte

El mercado de la telefonía móvil es uno de los más dinámicos dentro de la electrónica de consumo. Al contrario de lo que ocurre con televisores o minicadenas, los teléfonos móviles, así como las tecnologías asociadas a ellos, se renuevan mucho más frecuentemente. Y es que desde la aparición de los “*smartphones*”, impulsada por Apple en 2007, los móviles se convirtieron en un dispositivo cuyas funciones van mucho más allá de realizar llamadas telefónicas, y su uso diario creció exponencialmente.

Cada Smartphone dispone de un sistema operativo que actúa como un intermediario entre el usuario y las características físicas del dispositivo, permitiéndole personalizarlo lo máximo posible y sacar partido de sus funcionalidades. Para ello, y como respuesta a la flexibilidad de estos sistemas operativos surgen las aplicaciones móviles, que pueden ser desarrolladas por los propios usuarios.

Cada día existen más aplicaciones que tratan de hacerse un hueco en este nuevo mercado, con características y objetivos muy variados. Por lo general se encuentran disponibles a través de las plataformas de distribución operadas por las compañías propietarias de los sistemas operativos móviles. Esto generalmente implica encontrar diferentes aplicaciones para cada plataforma, ya que a menudo están íntimamente ligadas al sistema operativo para el que han sido desarrolladas.

Que una aplicación dependa directamente de un sistema operativo conlleva un problema, asociado a que la variedad de dispositivos del mercado es cada vez mayor. Y es que para que cualquier usuario pueda disponer de la aplicación, ésta deberá ser desarrollada independientemente para cada plataforma. Aunque tal y como se verá más adelante, el desarrollo de aplicaciones nativas también trae consigo ciertas ventajas.

A continuación se exponen los principales sistemas operativos móviles actuales, así como los diferentes tipos de aplicaciones disponibles para estos dispositivos. Por último, se comentarán algunas herramientas para el desarrollo de aplicaciones multiplataforma, similares al utilizado para el desarrollo de este proyecto.

2.1. Sistemas operativos para dispositivos móviles

Hablar de sistemas operativos móviles es hablar en primera instancia de los grandes dominadores del mercado en la actualidad: iOS y Android. Entre ambos arrasan la mayoría de estudios con entre el 85% y el 90% de cuota de mercado conjunta.

Les sigue Windows Phone, que ganó en 2013 el tercer puesto en sistemas operativos a nivel mundial con una cuota de mercado del 3.6%. Blackberry OS se sitúa como cuarta opción entre los usuarios de dispositivos móviles, habiendo reducido sus terminales en un 40% este año.

Existen otros sistemas operativos, como Firefox OS, Tizen o Amazon Fire OS, que tratan de abrirse un hueco en el mercado, aunque con muchos menos usuarios que los ya mencionados.

En el siguiente gráfico se muestran los datos de la cuota de mercado de 2013 a nivel mundial, según el estudio realizado por Kantar Worldpanel[18]:

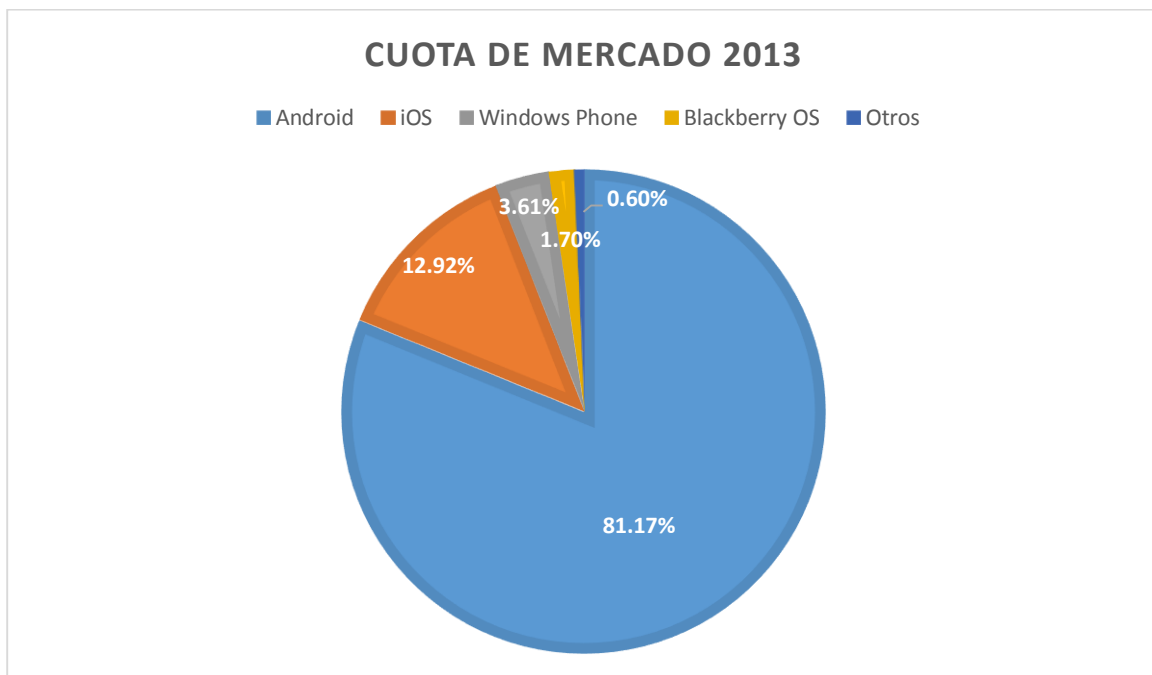


Figura 1: Cuota de mercado de los SOs móviles

Según el análisis realizado por la firma Telsyte sobre sistemas operativos móviles[30], la intención de compra de repetición aumenta a medida que los usuarios buscan la familiaridad con la plataforma o el modelo. También destaca que las generaciones más jóvenes valoran la calidad de la cámara del smartphone como un elemento “importante o crítico”, aunque un aspecto aún más importante a la hora de utilizar un dispositivo móvil son las aplicaciones.

Los desarrolladores son una pieza clave en el posicionamiento de cualquier sistema operativo. El hecho de que proporcionen más opciones de contenido y uso de un teléfono hacen que una plataforma sea más apetecible, puesto que permiten democratizar los gustos y las necesidades del público usuario.

2.2. Taxonomía de aplicaciones móviles

Existen tres tipos de aplicaciones móviles: nativas, web o híbridas. Es tarea del desarrollador analizar el tipo de proyecto que desea realizar para poder elegir, en función de sus pros y sus contras, el tipo de aplicación que mejor se adapte a sus requisitos.

A continuación se resumen las características de cada tipo de aplicación:

a. Aplicaciones nativas

Las aplicaciones nativas son aquellas desarrolladas para un sistema operativo móvil específico, utilizando el kit de desarrollo de software (SDK) de la plataforma, de manera que no pueden ser implementadas por otro sistema operativo diferente. Es decir, una aplicación nativa diseñada específicamente para Android no podrá instalarse en un iPhone.

Otra de las características de este tipo de aplicaciones es que ofrecen un rendimiento más rápido y un acceso directo a los servicios nativos del dispositivo, tales como la cámara, el gps o el acelerómetro. También se tiene acceso a todas las librerías gráficas del SO, que facilitan la implementación de la interfaz de usuario.

Las desventajas son fundamentalmente del tipo económico ya que para desarrollar aplicaciones nativas es necesario conocer los diferentes lenguajes de programación de cada sistema operativo, además de trabajar con los entornos de desarrollo convenientes. No será posible reutilizar el código de un SO en otro. Además hay que tener en cuenta que las aplicaciones necesitan actualizaciones a nuevas versiones del sistema, mantenimiento y/o aumento y mejoras de las funcionalidades, y todo ello debe desarrollarse directamente en el código nativo de cada plataforma.

b. Aplicaciones web

Las aplicaciones web móviles, a diferencia de las aplicaciones nativas, se ejecutan dentro del navegador del teléfono, siendo ejecutables desde cualquier dispositivo móvil.

La principal ventaja de este tipo de aplicaciones es que requieren una menor inversión inicial, debido a que la mayor parte del desarrollo no se debe repetir para cada sistema operativo. A consecuencia de ello, no es preciso conocer los distintos lenguajes de programación específicos de cada plataforma, con lo que se abaratan aún más los costes iniciales de presupuesto y/o el tiempo de aprendizaje.

Los defensores de HTML5 creen que la calidad de las aplicaciones es comparable con la de una aplicación nativa. Además, es más fácil obtener beneficios económicos y realizar cambios en la aplicación, ya que no se descargan directamente de una tienda de aplicaciones.

Algunos de los argumentos en contra de las aplicaciones web son el rendimiento, la incomodidad de ir a una URL en lugar de descargar una aplicación, y la falta de incompatibilidad con los navegadores, ya que no todos los navegadores para móviles son compatibles con el mismo subconjunto de características. Además hay que tener en cuenta que el acceso a los recursos del dispositivo es mucho más limitado que en el caso de las aplicaciones nativas.

c. Aplicaciones híbridas

Como su propio nombre indica, este tipo de aplicaciones combinan tecnologías de las aplicaciones nativas y web.

Se trata de aplicaciones escritas en HTML, JavaScript y CSS, compiladas con un "contenedor" nativo creado con herramientas de desarrollo multiplataforma. El aspecto es similar al de una aplicación nativa, al igual que su instalación (que se realiza a través de una tienda de aplicaciones), aunque gran parte de estas aplicaciones se diseñan utilizando páginas web.

La principal ventaja de las aplicaciones híbridas frente a las nativas es su facilidad de desarrollo. Al utilizar tecnologías web se evita la necesidad de conocer los distintos lenguajes de programación, tal y como ocurre en el caso del desarrollo de aplicaciones web. Por este motivo además, el código fuente de la aplicación puede ser utilizado para diferentes plataformas móviles.

Una ventaja con respecto a las aplicaciones web es que la herramienta de desarrollo multiplataforma con la que se implementa la aplicación permite acceder a buena parte de los servicios y sensores del dispositivo.

No obstante, las aplicaciones móviles nativas suelen ofrecer una mejor experiencia de usuario, debido fundamentalmente a la velocidad de respuesta de la aplicación. Además, hay que tener en cuenta la compatibilidad de los dispositivos con las últimas características de las tecnologías web usadas durante el desarrollo.

2.3. Frameworks para aplicaciones híbridas

Este proyecto se ha centrado en el desarrollo de una aplicación híbrida mediante el framework PhoneGap, que se explicará más adelante. No obstante, existen otras herramientas que permiten el desarrollo de este tipo de aplicaciones multiplataforma. Algunos de los más conocidos son:

a. Adobe Air

Adobe Integrated Runtime, también conocido como Adobe AIR, es un entorno multiplataforma desarrollado por Adobe Systems para la creación de aplicaciones RIA (Rich Internet Applications) que pueden ser ejecutadas como aplicaciones de escritorio o en dispositivos móviles. Estas aplicaciones se implementan usando Adobe Flash, Apache Flex (también conocido como Adobe Flex), HTML y Ajax.

Adobe AIR permite que los desarrolladores empaqueten el mismo código en aplicaciones nativas y juegos para ordenadores de sobremesa de Windows y Mac OS, así como para dispositivos iOS y Android.

Su principal ventaja radica en el amplio rango de dispositivos en el que se pueden ejecutar las aplicaciones desarrolladas mediante este framework, tales como ordenadores, tablets, smartphones o televisiones. Además, se recomienda su uso para aplicaciones que requieran una interfaz de usuario animada, con elementos en 2D y 3D, ya que el uso de Flash puede suponer una ventaja frente a HTML.

No obstante, el hecho de que Adobe comprara Nitobi (y los derechos sobre el nombre PhoneGap), combinado con la rápida disminución de flash erosiona la confianza de muchos desarrolladores a la hora de elegir AIR para el desarrollo de aplicaciones híbridas.

b. Appcelerator Titanium

Appcelerator Titanium es una plataforma para desarrollar aplicaciones móviles y de escritorio utilizando tecnologías web. Fue desarrollado por Appcelerator Inc. y lanzado en diciembre del 2008. En junio de 2009 se añadió soporte para el desarrollo de aplicaciones móviles para iOS y Android, y posteriormente también para Blackberry y Windows.

Este framework proporciona una herramienta multiplataforma para crear aplicaciones nativas, de manera que los desarrolladores escriben código JavaScript abstrayéndose de la interfaz gráfica, que se implementa mediante componentes nativos, mejorando la experiencia de usuario en comparación con otras alternativas de desarrollo híbrido.

Una de las desventajas del uso de Titanium es que los desarrolladores deben utilizar los SDKs de las distintas plataformas (y por tanto los entornos de desarrollo) de manera local. Además, el hecho de normalizar el proceso de construcción de la UI para que se comporte de manera nativa en cualquier plataforma supone un nuevo lenguaje que requiere un estudio previo de por parte del desarrollador; estudio que no será aplicable a otras tecnologías o herramientas más allá de este framework.

c. Qt

Qt es un framework de código abierto desarrollado por Digia, para la creación de aplicaciones multiplataforma e interfaces gráficas mediante C++ o QML, que es un lenguaje basado en CSS y JavaScript. Actualmente es compatible con los sistemas operativos móviles Android, iOS y Blackberry 10.

Este framework proporciona un amplio conjunto de las librerías que contienen APIs destinadas a facilitar el manejo de hilos, redes, o animaciones entre otras posibilidades.

Dispone de dos entornos de desarrollo propios: Qt Creator IDE y Qt Designer, además de la herramienta Qt Linguist que permite la traducción y la internacionalización de las aplicaciones – dando soporte para varios idiomas dentro de la aplicación (en un único archivo).

El servicio Qt Mobile está diseñado para desarrolladores individuales o pequeñas empresas y comprende todas las librerías de Qt, Qt Creator Enterprise IDE, y los servicios de Qt Cloud, así como licencia comercial y soporte de Digia. Permite el desarrollo de aplicaciones para Android y iOS, con un coste mensual de suscripción de 109€ al mes.

3. PhoneGap

PhoneGap es un framework que permite el desarrollo de aplicaciones híbridas para sistemas operativos móviles, haciendo uso de tecnologías web como JavaScript, HTML5 y CSS3, ofreciendo una única API basada en JavaScript para acceder a los servicios nativos del dispositivo. Con PhoneGap es posible desarrollar aplicaciones para los siguientes sistemas operativos:

- Amazon Fire OS
- Android
- Bada
- BlackBerry
- iOS
- Symbian
- Tizen
- Web OS
- Windows Phone

Inicialmente PhoneGap fue desarrollado por la empresa Nitobi pero tras la adquisición de ésta por parte de Adobe, todo el proyecto se trasladó a la Apache Software Foundation (ASF) con el nombre de Apache Cordova, convirtiéndose PhoneGap en una distribución de Apache Cordova totalmente libre para su uso comercial.

3.1. API

Para poder interactuar con las funcionalidades del hardware del sistema, PhoneGap dispone de una API escrita en JavaScript con la que se pueden utilizar fácilmente las siguientes características:

- **Acelerómetro:** Permite monitorear el movimiento, la orientación y las agitaciones del dispositivo.
- **Cámara:** Permite a los usuarios capturar fotografías para la aplicación o acceder a las imágenes almacenadas en el dispositivo.
- **Capturar:** Permite capturar audio y vídeo.
- **Compass:** Obtiene la dirección a la que apunta el dispositivo.

- **Conexión:** Permite determinar si existe conexión a internet, y en caso afirmativo, cómo de fuerte es la señal que se recibe.
- **Contactos:** Proporciona acceso a los contactos almacenados en la agenda del dispositivo.
- **Dispositivo:** Permite consultar información del hardware y del software del dispositivo, como por ejemplo su sistema operativo, el modelo o la versión de PhoneGap que se está utilizando.
- **Eventos:** Soporta varios eventos relacionados con el dispositivo, como por ejemplo los cambios en el nivel de la batería, los botones físicos que puedan existir en el dispositivo o el estado de la aplicación dentro de su ciclo de vida.
- **Ficheros:** Proporciona acceso al sistema de ficheros del dispositivo, incluyendo la lectura y escritura de archivos y directorios.
- **Geolocalización:** Permite acceder a la localización del dispositivo en base al sensor GPS del mismo o a la señal proporcionada por la conexión Wifi y las antenas de telefonía.
- **Globalización:** Obtiene información basada en el idioma, localización y zona horaria del usuario.
- **InAppBrowser:** Permite el acceso a páginas web a través de sus URL sin necesidad de salir de la aplicación.
- **Media:** Permite grabar y reproducir audio.
- **Notificación:** Proporciona varias maneras de notificar al usuario, incluyendo sonidos, y alertas basadas en vibración.
- **SplashScreen:** Muestra u oculta una pantalla inicial predefinida para la aplicación.
- **Storage:** Provee varias opciones para gestionar el almacenamiento de la aplicación: LocalStorage, WebSQL e IndexedDB.

3.2. Instalación

A partir de la versión 3.0 de PhoneGap, su instalación se realiza a través de la interfaz de línea de comandos (CLI). Esta herramienta permite crear proyectos y construirlos para diferentes

plataformas, tanto de manera local como en un servidor remoto, además de ejecutarlos tanto en un emulador como en un dispositivo real.

En primer lugar, es necesario instalar el SDK de la plataforma sobre la que se desee comenzar el proyecto. Es preciso tener en cuenta que tanto iOS como Windows Phone imponen el uso exclusivo de máquinas Mac o Windows respectivamente.

Para usar la herramienta de línea de comandos el primer paso es instalar Node.js, y posteriormente la utilidad “phonegap”. Para ello se ejecuta en la consola de node.js el siguiente comando:

```
sudo npm install -g phonegap
```

Una vez instalada, el siguiente paso es crear el proyecto mediante el comando:

```
phonegap create hello com.example.hello HelloWorld
```

El primer argumento identifica al directorio generado para el proyecto, en el que se creará el subdirectorio “www” con una estructura de carpetas que incluye algunas como “css”, “js” o “img”. También se crea el archivo “config.xml”, que contiene información necesaria para generar y distribuir la aplicación. Los otros dos parámetros son opcionales, y proporcionan un identificador basado en dominio para el proyecto y el título de la aplicación respectivamente.

Por defecto, la opción “create” de PhoneGap crea la estructura de una aplicación web, en donde la página principal es *www/index.html*.

Antes de poder compilar la aplicación se debe especificar al menos una plataforma para la que esté destinada. Para ello se debe ejecutar alguno de los siguientes comandos, teniendo en cuenta que tiene que haberse instalado previamente el SDK de la plataforma que se desee utilizar:

```
$ cordova platform add amazon-fireos  
$ cordova platform add android  
$ cordova platform add blackberry10  
$ cordova platform add firefoxos  
$ cordova platform add ios  
$ cordova platform add windows8  
$ cordova platform add wp7  
$ cordova platform add wp8
```

Estos comandos generan una estructura de carpetas diferente para cada plataforma. Pueden comprobarse las plataformas para las que se ha configurado el proyecto, o eliminar una plataforma utilizando los siguientes comandos respectivamente:

```
$ cordova platforms ls  
$ cordova platform rm android
```

El comando *“build”* compila la aplicación para la plataforma especificada, generando los archivos específicos en la subcarpeta *“platforms”* del proyecto. Para compilar la aplicación para Android se utilizará el comando:

```
$ phonegap build android
```

3.3. Plugins

Todas las características básicas de la API de PhoneGap están implementadas en forma de plugins, de manera que se comuniquen tanto con el dispositivo físico como con la aplicación. Para ello se componen de dos partes, una escrita en JavaScript y otra en el lenguaje nativo de la plataforma sobre la que se ejecuta la aplicación:

- La parte JavaScript del plugin actuará como interfaz para la aplicación web.
- La parte nativa del plugin será invocada desde el código JavaScript y contendrá la implementación de la funcionalidad.

Si se desean utilizar funcionalidades adicionales a las descritas en la API de PhoneGap, se pueden crear plugins personalizados o utilizar plugins de terceros.

Para incluir un nuevo plugin en el proyecto se dispone del siguiente comando, en el que se especifica la dirección del repositorio en el que se encuentra su código fuente:

```
$ cordova plugin add https://git-wip-us.apache.org/repos/asf/cordova-plugin-device.git
```

En dicho repositorio debe haber un archivo de configuración llamado *“plugin.xml”* que especifique parámetros como su identificador, su nombre o la plataforma para la que ha sido desarrollado. Este archivo debe incluir también una etiqueta *“js-module”* en la que se indica la ruta en la que se encuentra la interfaz JavaScript del plugin.

Para que el código JavaScript actúe como una interfaz del plugin es necesario utilizar en esta parte la llamada al método **cordova.exec** para comunicarse con la parte nativa. Su sintaxis es la siguiente:

```
cordova.exec(function(winParam) {},  
             function(error) {},  
             "service",  
             "action",
```

```
["firstArgument", "secondArgument", 42, false]);
```

Los parámetros requeridos en esta función son:

- **function(winParam) {}**: Se trata de una función de callback. Se ejecutará en caso de que la llamada a *exec* se complete de manera satisfactoria.
- **function(error) {}**: Se trata de una función similar a la anterior que pasará a ejecutarse en caso de que exista algún error en la ejecución del método *exec*.
- **"service"**: Con este parámetro se especifica el nombre del que dispone el servicio en la parte nativa.
- **"action"**: Representa el nombre de la clase nativa con la que se vincula.
- **[/* arguments */]**: Se puede incluir un parámetro adicional en forma de array que recoge los argumentos que se deseen pasar a las clases de la parte nativa del plugin.

Una vez definida la parte JavaScript, se debe complementar con una implementación en lenguaje nativo. A continuación se listan algunas de las plataformas que aceptan el uso de plugins:

- Amazon Fire OS
- Android
- iOS
- BlackBerry 10
- Windows Phone

En el desarrollo de este proyecto se ha utilizado un plugin de terceros para personalizar la funcionalidad de la aplicación, conocido como **"LocalNotifications"**. Se trata de una extensión que permite producir notificaciones de manera local en el dispositivo, así como programarlas en el tiempo. Estas notificaciones pueden ser recogidas por la aplicación a través de su interfaz JavaScript para crear nuevas alarmas, modificarlas o simplemente consultar las notificaciones pendientes. A continuación se muestra un ejemplo de uso:


```
// Crea una notificación que se mostrará en 5 segundos
window.plugins.localNotification.add({
  fireDate      : Math.round(new Date().getTime()/1000 + 5),
  alertBody     : "Texto de la notificación",
  action        : "View",
  repeatInterval : "daily",
  soundName     : "beep.caf",
  badge         : 0,
  notificationId : 123,
  foreground    : function(notificationId){},
  background    : function(notificationId){}
});

// Cancela una notificación dado su identificador
window.plugins.localNotification.cancel(1234, callback);

// Cancela todas las notificaciones
window.plugins.localNotification.cancelAll(callback);
```

El plugin se encarga de manera nativa de crear las alarmas en el dispositivo móvil y gestionar su manipulación. Actualmente se encuentra disponible tanto para iOS como para Android.

3.4. PhoneGap Build

PhoneGap Build es un servicio de compilación y empaquetado en la nube para aplicaciones móviles basadas en PhoneGap. Su principal ventaja es que evita la necesidad de instalar el SDK de cada plataforma, así como los diferentes IDEs o herramientas que se deban utilizar. De esta forma maximizan la productividad reduciendo el tiempo de producción.

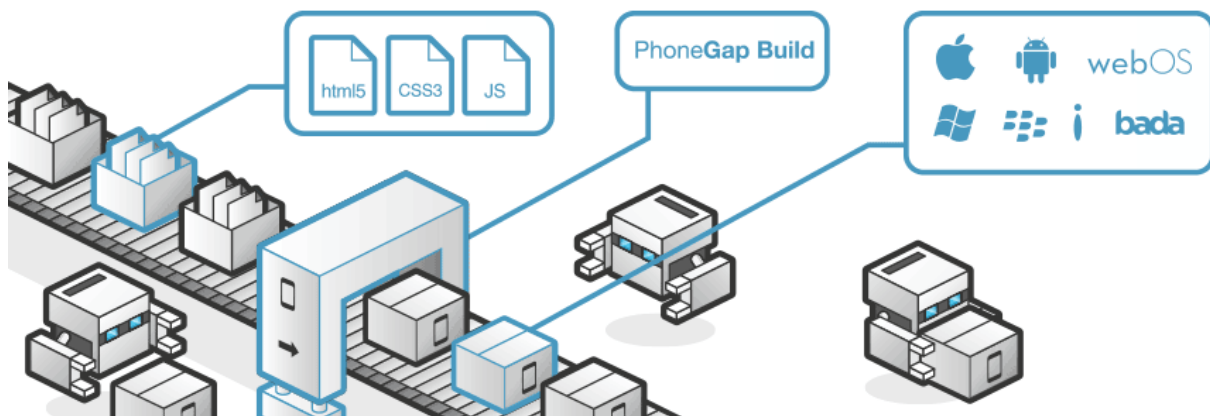


Figura 2: PhoneGap Build

Para utilizar esta herramienta basta con registrarse en su página web y vincular a la cuenta el repositorio en el que se aloja el código del proyecto. Existen tres opciones de registro: “Free Plan”, “Paid Plan” y “Adobe Creative Cloud Membership”. La primera de ellas es gratuita y permite la creación de una aplicación privada e ilimitadas aplicaciones públicas.

Una vez que se le proporciona a PhoneGap Build acceso al código del proyecto, se puede ejecutar su compilación para los diferentes sistemas operativos, sin necesidad de instalar ningún complemento adicional.

Para generar los archivos compilados para algunas plataformas es preciso indicar antes las claves de compilación y/o distribución. Una vez superado este paso, y cuando la compilación esté lista, se ofrece al usuario un código QR que facilita la instalación de la aplicación en los distintos dispositivos, así como los archivos compilados para cada plataforma.

Además, se puede trabajar de manera colaborativa en un proyecto de PhoneGap Build añadiendo miembros al equipo encargado del mismo y asignándoles diferentes roles.

Los proyectos desarrollados con PhoneGap disponen de un archivo de configuración, “config.xml”, en el que se especifica la meta-información relativa a la aplicación, como los permisos que requiere por parte del dispositivo, su icono o los plugins que utiliza. Para facilitar el proceso de creación de este archivo existe una herramienta, ConfiGAP, que ofrece una interfaz de usuario en la que rellenar los datos con los que se desee crear automáticamente el archivo de configuración:

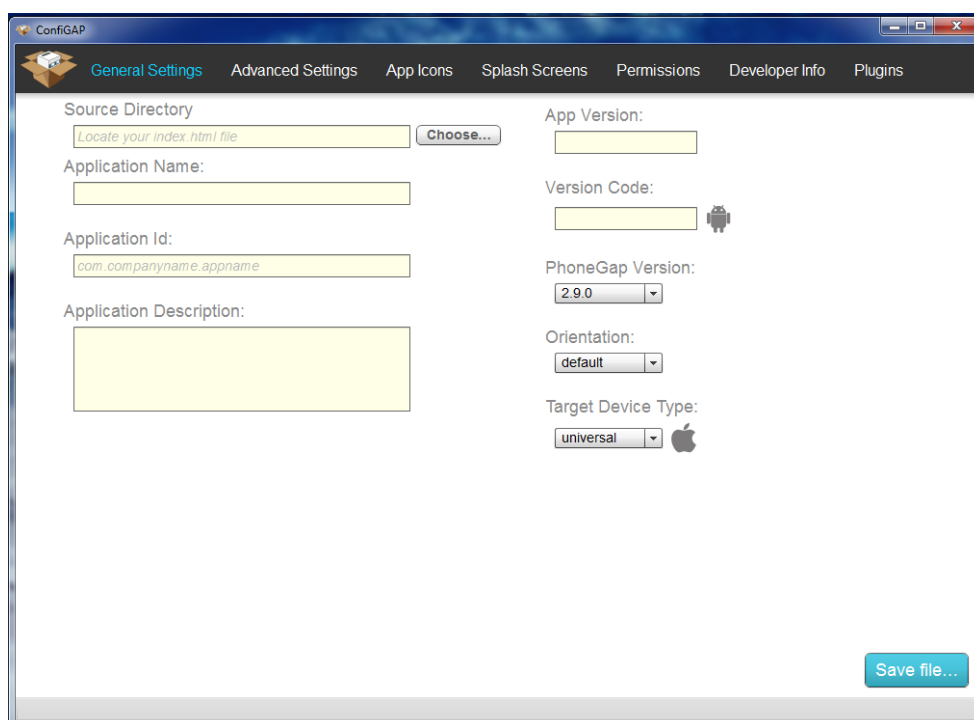


Figura 3: ConfiGAP

4. Tecnologías subyacentes

4.1. Html5

La quinta versión de HTML (de sus siglas en inglés HyperText Markup Language) es el último desarrollo del lenguaje web básico diseñado por el consorcio W3C (World Wide Web Consortium). A día de hoy todavía no es compatible con la mayoría de navegadores actuales, aunque cada vez se extiende más su uso debido a las posibilidades de las nuevas características que presenta.

Una de las principales características de HTML5 es su capacidad de reducir código a la vez que aumenta sus posibilidades de programación. A través de nuevas etiquetas, comandos, formularios web, etc. HTML5 es capaz de crear estructuras mucho más complejas de lo que tradicionalmente le resultaba posible. Esta nueva versión de HTML es mucho menos dependiente de otros lenguajes de programación a la hora de integrar elementos más dinámicos en el diseño web. El uso de dichos lenguajes “de apoyo” completaba aquellas facetas de programación en las que HTML se quedaba obsoleto, pero también añadían ciertos inconvenientes. El aumento del peso de las páginas web era un problema, además del hecho de que algunos de ellos, como por ejemplo Flash, no son legibles por parte de los buscadores de contenido. Esto llevaba a que la información contenida en aquellos elementos no pudiera ser rastreada por esos buscadores, y por tanto el diseñador de la aplicación web debía elegir entre diseño o posicionamiento SEO. HTML5 le da una importancia mucho mayor al diseño, consiguiendo un dinamismo superior con una estructura puramente HTML, es decir, perfectamente legible para los robots de los buscadores. Gracias a la quinta versión de HTML, lenguajes como JavaScript o Flash no interfieren en la estructura ni en la capacidad exportadora del contenido de las páginas, y se pueden crear formularios más complejos, gráficos más potentes y contenido más editable entre otras muchas posibilidades.

a. Evolución de HTML

En el artículo “The History of HTML5” de Matt Silverman para el blog de noticias Mashable[23] se muestra una infografía de la evolución de HTML5 desde su creación. A continuación se describen cronológicamente los acontecimientos que han tenido lugar desde la aparición del lenguaje de marcas HTML:

- **1989:** Aparece el HTML, que pronto sufrió varias revisiones: HTML 2, HTML 3.2, HTML 4 y HTML 4.01.
- **2000:** Se presenta el XHTML 1 como la evolución natural de HTML.

- **2002:** Se comienza a definir XHTML 2, una actualización de XHTML 1 con varias mejoras semánticas pero sin compatibilidad regresiva; era realmente un lenguaje nuevo.
- **2004:** Se constituye el Web Hypertext Application Technology Working Group (WHATWG), un grupo de trabajo con miembros de Apple, la Fundación Mozilla y Opera Software, y nacido al margen del W3C, con el propósito de crear una nueva versión del estándar desde un punto de vista destacadamente práctico, y no tan académico. Y es que siete años después de que el W3C publicase las recomendaciones oficiales de HTML 4.01 y XHTML 1, aún no había ningún consenso salido del W3C sobre el camino que se debía seguir.
- **2006:** el W3C reconoce el trabajo del WHATWG y lo toma como base para proponer una nueva actualización del estándar.
- **2008:** Se publica la primera la primera versión de HTML5 por Ian Hickson. Los expertos consideran que este lenguaje nunca estará acabado y que se encontrará en una evolución constante.

Se lanza el navegador de Mozilla, *Firefox 3*, que ya es compatible con HTML5.

- **2009:** El W3C abandona definitivamente XHTML2 (antes de terminar su desarrollo), y se concentra en definir HTML5.
- **2010:** YouTube ofrece un reproductor de vídeo basado en este lenguaje.

En el mes de abril, Steve Jobs escribe una larga carta rechazando el uso de Flash en los dispositivos de Apple, y proclamando así el uso de HTML5: "Flash ha sido diseñado para ordenadores con teclado y ratón, no para pantallas táctiles que utilizan los dedos". Añade que: "Apple apuesta claramente por HTML5 porque permite a los desarrolladores crear gráficos avanzados y animaciones, entre otros, sin necesidad de complementos de un tercero, como Flash".

En mayo, la plataforma web Scrib para el uso compartido de documentos se convierte a HTML5, mejorando así la experiencia de usuario para tablets y dispositivos móviles.

En agosto, el grupo musical Arcade Fire junto con el apoyo de Google lanza el vídeo interactivo "The Wilderness Downtown", aprovechando el poder de HTML5, que se convierte en todo un éxito.

- **2011:** Disney compra una ‘start-up’ que desarrolla juegos en HTML5 para intentar desarrollar juegos que funcionen directamente en la web, y con ello romper el monopolio de la App-store de Apple.

La aplicación web musical Pandora adapta su reproductor de audio a HTML5. Las críticas opinan que este nuevo sistema es más rápido y fácil de cargar que el anterior basado en Flash

Amazon crea el “Kindle Cloud Reader”, una nueva versión web de su aplicación de lectura para el Kindle eBook. Esta nueva versión HTML5 permite a los usuarios acceder a su contenido de forma offline directamente desde su navegador.

Twitter desarrolla una nueva versión HTML5 para iPad. Así mismo el periódico Boston Globe adapta su plataforma web a los distintos dispositivos móviles haciendo uso de RWD, CSS3 y HTML5.

En septiembre de 2011, 34 de las 100 páginas con más tráfico según el ranking de Alexa ya utilizan este lenguaje.

A finales de año Adobe deja de desarrollar Flash para dispositivos móviles para centrarse en el desarrollo de herramientas HTML5.

- **2012:** HTML5 llega a Flickr (en su nueva plataforma de subida de archivos) y a LinkedIn, que desarrolla una aplicación para iPad diseñada en un 95% con HTML5.

A través de la plataforma Wix.com se crean más de 1.000.000 de páginas web basadas en HTML5

- **2013:** Cada vez es más común el uso de HTML5 por parte de los desarrolladores web, y este cambio coincide con la necesidad de adaptar las aplicaciones a cada vez más navegadores y tamaños de pantalla. Según el artículo “*Development Landscape 2013*” publicado en el mes de agosto por la compañía *Forrester Research*[13], el 55% de los desarrolladores software ya utilizan HTML5.

b. Novedades

Nuevas etiquetas

A continuación se describen algunas de las etiquetas más significativas que han sido introducidas en esta nueva versión de HTML:

- **Audio:** Esta nueva etiqueta de HTML5 permite incluir archivos de audio sin necesidad de utilizar plugins adicionales. Actualmente hay tres formatos de archivo diferentes soportados por este elemento: MP3, Wav y Ogg. La siguiente tabla muestra la compatibilidad de estos formatos con los principales navegadores:

Navegador	MP3	Wav	Ogg
Internet Explorer	✓	✗	✗
Chrome	✓	✓	✓
Firefox	✓*	✓	✓
Safari	✓	✓	✗
Opera	✗	✓	✓

Tabla 2: Formatos de audio soportados por los navegadores

* El formato de audio MP3 es soportado en Firefox siempre y cuando el Sistema operativo proporciona un decodificador de MP3.

Para solucionar esta restricción de los navegadores, se puede definir más de un archivo en diferentes formatos utilizando únicamente una etiqueta audio a través de la etiqueta **<source>**.

Como ocurre con todas las etiquetas nuevas de HTML5, lo que se encuentre entre el inicio y el cierre de éstas solo será tenido en cuenta por navegadores que no soportan la nueva etiqueta.

El elemento de audio se agrega directamente al código HTML mediante el uso de la propiedad **src** para especificar el archivo de audio que se desea reproducir. Mediante esta propiedad se pueden agregar varios archivos en diferentes formatos

Además, esta etiqueta dispone de otros atributos:

- **“autoplay”:** atributo booleano que carga y ejecuta el sonido en cuanto se cargue la página.
- **“controls”:** atributo booleano que muestra los controles de reproducción integrados del navegador.
- **“loop”:** atributo booleano que hace que el sonido se reproduzca en bucle.

- **“preload”**: este atributo hace que el archivo de audio sea precargado en segundo plano por el navegador. Puede tomar tres posibles valores: *auto*, *metadata* y *none*.

El siguiente fragmento de código muestra un ejemplo de uso de la etiqueta <audio>:

```
<audio controls>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mpeg">
  Su navegador no soporta el elemento audio.
</audio>
```

- **Canvas**: La etiqueta <canvas> actúa como un marco para la creación de gráficos 2D y 3D. Este elemento permite realizar cuadrados, círculos, cajas, líneas y otras figuras geométricas con la ayuda de diferentes métodos y propiedades proporcionados por un lenguaje de script, generalmente JavaScript.

El propio elemento es relativamente simple, lo único que hay que especificar al usarlo son sus dimensiones:

```
<canvas id="entorno_canvas" width="360" height="240">;
</canvas>
```

Como comentábamos en el caso de la etiqueta <audio>, cualquier cosa escrita entre la apertura y cierre de la etiqueta <canvas> solamente será interpretada por navegadores que no soportan aún la nueva etiqueta. Es por tanto una buena práctica escribir un mensaje dentro de la etiqueta que se mostrará cuando ésta no sea soportada, alertando de ello al usuario.

Para comenzar a dibujar gráficos dentro de esta etiqueta, primero debemos referenciar el elemento canvas y adquirir su contexto (API).

```
var canvas = document.getElementById('entorno_canvas');
var context = canvas.getContext('2d');
```

Una vez adquirido el contexto, podemos empezar a dibujar en la superficie del canvas usando la API a tal efecto documentada en la web de WHATWG [44]. La API bidimensional ofrece muchas de las herramientas que podemos encontrar en cualquier aplicación de diseño gráfico como Adobe Illustrator o Inkscape: trazos, rellenos, gradientes, sombras, formas y curvas Bézier. Claro está deberemos especificar cada acción con ellas usando JavaScript.

El potencial de canvas reside en su habilidad para actualizar su contenido en tiempo real. Si usamos esa habilidad para responder a eventos de usuario, podemos crear herramientas y juegos que anteriormente a la nueva especificación hubiesen requerido de un plugin externo como Flash.

- **Embed:** Se utiliza como contenedor para cargar una aplicación externa o un contenido interactivo (plugin).

Los atributos de este elemento son los siguientes:

- **heightNew:** Especifica la altura del contenido integrado.
- **srcNew:** Especifica la dirección del archivo externo a incrustar.
- **typeNew:** Especifica el tipo MIME del contenido integrado.
- **widthNew:** Especifica el ancho del contenido incrustado.

La utilización de este elemento es realmente sencilla, basta con especificar el origen del archivo que queremos incrustar:

```
<embed src="helloworld.swf">
```

- **Progress:** Cuando se construye una página o una aplicación web, es importante informar al usuario del progreso de las tareas que está realizando, ya sean, por ejemplo, subir un archivo, reproducir un vídeo o importar datos. Gracias a la incorporación en HTML5 de la etiqueta **<progress>** esta labor es mucho más sencilla, ya que proporciona una barra de progreso que permite al usuario comprobar que la web, lejos de haberse bloqueado o haber ignorado su petición, está realizando la tarea.

Hay dos tipos de barras de progreso: indeterminadas (en caso de que no exista un valor, es decir, que no existan límites exactos para el proceso) o determinadas (cuando existe un valor que identifica el porcentaje de tarea que se ha completado). La siguiente imagen muestra el aspecto de ambos tipos de barras de progreso en el navegador Internet Explorer:



Figura 4: Barra de progreso determinada



Figura 5: Barra de progreso indeterminada

Existen tres atributos para el elemento **<progress>**: **max**, **value**, **position**:

- **max**: se trata de un número flotante que indica la cantidad de trabajo restante para que la tarea que se esté ejecutando pueda considerarse totalmente completada. El valor predeterminado de este atributo es el número 1.0.

Si no se incluye el atributo **max**, el rango de valores aceptado por defecto para la barra de progreso se encuentra entre 0.0 y 1.0.

- **value**: al igual que max, este atributo tiene como valor un número flotante, el cual representa el progreso actual de la tarea correspondiente. Su valor va desde 0.0 al valor máximo establecido en el atributo max.
- **position**: Este es un atributo de sólo lectura que se encarga de devolver la posición actual del elemento **progress**. En un elemento de progreso determinado, este valor equivale al resultado de la operación **value/max**. En el caso de tratarse de una barra de progreso indeterminada, este atributo toma el valor -1.

La utilización de esta etiqueta se muestra en el siguiente ejemplo:

```
<progress value="22" max="100"></progress>
```

- **Vídeo**: Esta etiqueta, permite incrustar archivos de vídeo en nuestro documento HTML sin necesidad de contar con reproductores de terceros. Los formatos aceptados por los principales navegadores son los siguientes:

Navegador	MP4	WebM	Ogg
Internet Explorer	✓	✗	✗
Chrome	✓	✓	✓
Firefox	✗	✓	✓
Safari	✓	✗	✗
Opera	✗	✓	✓

Tabla 3: Formatos de vídeo soportados por los navegadores

El elemento video es muy parecido al elemento audio, también dispone de los atributos **autoplay**, **loop** y **preload**. También se puede especificar la fuente de un archivo, bien usando el atributo **src** en la etiqueta de apertura, o bien usando el elemento **source** entre las etiquetas de apertura y cierre. Asimismo se pueden utilizar

los controles que ofrece el navegador de forma nativa utilizando el atributo **controls** (también se pueden configurar controles propios a través de JavaScript).

Otra de las propiedades del elemento video es **poster**, que permite definir una imagen representativa para el vídeo, que será mostrada por el navegador como portada del elemento antes de su reproducción.

Además, se debe definir un tamaño para el elemento de vídeo a través de los atributos **width** y **height**. A continuación se muestra un ejemplo de la utilización de esta etiqueta:

```
<video controls width="360" height="240" poster="poster.jpg">
  <source src="archivo.ogv" type="video/ogg" />
  <source src="archivo.mp4" type="video/mp4" />
</video>
```

Mejoras en los formularios

Los formularios de HTML5 incluyen nuevas funcionalidades pensadas para solucionar problemas cotidianos de forma declarativa con la nueva especificación del lenguaje. De esta forma ofrece una manera más inteligente de trabajar con formularios, integrando funcionalidad para la que antes era necesario el uso de JavaScript de forma declarativa e integrada directamente en los elementos de la especificación. Estas especificaciones formaban parte originalmente de las especificaciones de la WHATWG y fueron bautizadas como Web Forms 2.0, basadas en un trabajo previo de la W3C.

A continuación se describen algunas de estas novedades:

- **Autocomplete**

Con el tiempo, los navegadores web han ido implementando nuevas funcionalidades y mejoras, algunas de ellas centradas en mejorar la experiencia de usuario, la accesibilidad, la usabilidad o la seguridad. Una de ellas es la habilidad de la inmensa mayoría de los navegadores modernos de auto completar algunos campos de los formularios. Esta funcionalidad puede resultar muy útil o suponer un problema de seguridad y privacidad para el usuario.

La nueva especificación de HTML5 nos permite desactivar el auto completado en un formulario completo o solo en campos específicos. El atributo autocomplete nos permite definir dos valores para él: **“on”** y **“off”**:

```
<form action="/funcion" autocomplete="off">  
  ...  
</form>
```

El código anterior desactivaría el auto completado de todo el formulario. Para deshabilitar el auto completado en un único campo se debe establecer esta propiedad dentro de su declaración:

```
<input type="text" autocomplete="off" />
```

No existe contrapartida a este comportamiento en JavaScript para navegadores que no soportan el nuevo atributo y lo cierto es que este nuevo atributo de la nueva especificación de HTML5 es ligeramente diferente al resto de novedades que trae esta especificación dado que no añade o mejora una funcionalidad, sino que bloquea un comportamiento de los navegadores. Aunque su incorporación está justificada como solución al problema de seguridad que conlleva el auto completado de ciertos campos en ciertos formularios.

- **Autofocus**

Esta funcionalidad es usada por ejemplo en la página principal de Google. Cuando se accede a Google el foco siempre lo acapara el input de búsqueda. Esto se ha programado siempre en JavaScript de manera que cuando el documento carga, automáticamente se pone el foco sobre un elemento específico.

Con HTML5 se puede conseguir esta funcionalidad de manera muy sencilla a través de un atributo booleano llamado autofocus:

```
<input type="text" autofocus />
```

Este comportamiento arrastra consigo una problemática y es que algunos usuarios emplean la tecla del espacio para desplazar la página hacia abajo, con lo que esta nueva característica puede desfavorecer la usabilidad de la web. Esta es la causa principal por la que Twitter decidió eliminar ese comportamiento de su nuevo cliente y ahora para escribir en el campo de estado hay que poner el foco en él primero.

- **Datalist y List**

El nuevo elemento **datalist** permite realizar una tarea que durante muchos años se ha tenido que implementar a través de JavaScript. Se trata de un híbrido entre un elemento **input** y un elemento **select**.

Usando el atributo `list` con elementos `input` se puede especificar una lista de opciones en él, permitiendo al usuario seleccionar un valor de la lista o escribir uno que no esté en ella.

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

El aspecto gráfico de esta característica en Chrome es el siguiente:

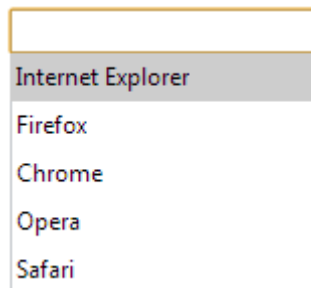


Figura 6: Vista del elemento datalist en Chrome

- **Patrones**

El atributo **pattern** permite especificar un patrón (representado en forma de expresión regular) que se usará en la validación del campo de texto. Se puede utilizar el atributo **title** para proporcionar al usuario un mensaje que le ayude a cumplir con el patrón especificado.

El siguiente ejemplo muestra cómo utilizar este atributo para validar un código postal:

```
<input id="cp" name="cp" pattern="[0-9][A-Z]{3}" title="Un código postal es un número de cinco cifras sin letras">
```

- **Placeholder**

Una receta típica usada mucho en formularios de búsqueda es la que sigue:

1. Cuando un campo no tiene valor, se inserta un placeholder (marca de posición) en él.
2. Si se focaliza dicho campo, se elimina el placeholder.
3. Si se abandona el campo sin añadir ningún valor, se vuelve a añadir el placeholder en él.

Normalmente suele mostrarse en un tono más claro que un valor válido del formulario. Esto siempre ha requerido del uso de JavaScript para ser llevado a cabo, pero con la nueva especificación de HTML5 este comportamiento puede definirse de una forma más sencilla:

```
<input type="text" placeholder="Búsqueda en Google" />
```

El resultado es el siguiente:

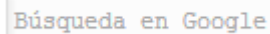
A screenshot of a text input field in a browser. The text 'Búsqueda en Google' is displayed in a light gray font, indicating it is a placeholder. The input field has a thin border and is centered on the page.

Figura 7: Vista del atributo placeholder en Chrome

- **Required**

Uno de los usos más extendidos de JavaScript en la parte cliente es sin duda la validación de formularios.

La validación en el lado cliente es importante ya que permite que un formulario no sea enviado si no es válido. Esto permite ahorrar un consumo innecesario de ancho de banda y sobre todo ofrece una salvaguarda del tiempo de espera de los usuarios, que son informados de inmediato de que algún campo del formulario no cumple los requisitos.

Una de las tareas de validación más extendidas es la de los campos requeridos. La nueva especificación de HTML5 incluye un atributo booleano llamado **required** que sirve para definir si un campo es requerido o no:

```
<input type="text" required/>
```

- **Nuevos tipos de inputs:**

HTML5 introduce 13 nuevos tipos de campos para los formularios. Su uso supone un doble beneficio, ya que reduce el tiempo de desarrollo de estos elementos a la vez que mejora la experiencia de usuario:

Search

El elemento representa una caja de búsqueda. Los saltos de línea son quitados del valor ingresado pero no se modifica ninguna otra sintaxis. También se puede añadir un histórico de búsquedas al elemento utilizando para ello el atributo **results**:

```
<input type="search" results="5"/>
```

El aspecto del elemento varía para adecuarlo a su nueva funcionalidad, añadiendo un icono de búsqueda.



Figura 8: Vista del input de tipo search en Chrome

Detalles de contacto

La nueva especificación añade tres nuevos tipos de valores para los detalles de contacto: correo electrónico, sitios web y números de teléfono.

```
<input id="correo" name="correo" type="email" />  
<input id="website" name="website" type="url" />  
<input id="telefono" name="telefono" type="tel" />
```

En esta ocasión el navegador los muestra como campos de texto normal. Sin embargo, el teclado que se despliega en un dispositivo móvil sí que difiere según el tipo de input:



Figura 9: Vista de diferentes teclados según el tipo en iPhone

Sliders

El elemento `range` permite fijar valores numéricos entre un máximo y un mínimo preestablecidos. Por defecto se asumen unos valores de cero a cien, aunque pueden especificarse usando los atributos `min` y `max`.

```
<input type="range" min="1" max="10" />
```



Figura 10: Vista del input de tipo range en Chrome

Spinner

Un spinner es un tipo de entrada numérica que puede incrementar o disminuir su valor a través de unos controles situados en su lado derecho. Se puede establecer un máximo y un mínimo para este elemento.

```
<input type="number" min="5" max="25" />
```



Figura 11: Vista del input de tipo number en Chrome

Calendarios: Fecha y hora

HTML5 introduce todo un rango de nuevos tipos relacionados con la selección de fechas y horas:

- **date**: define año, mes y día
- **datetime**: define año, mes, día, hora, minuto, segundos e información del **timezone**.
- **datetime-local**: define lo mismo que el anterior pero sin incluir la información del **timezone**.
- **time**: define la hora, minutos y segundos
- **month**: define el año y el mes pero sin el día
- **week**: se utiliza para fijar el número de semana del año (valor entre 1 y 53)

```
<input type="date" />  
<input type="datetime" />  
<input type="datetime-local" />  
<input type="time" />  
<input type="month" />  
<input type="week" />
```

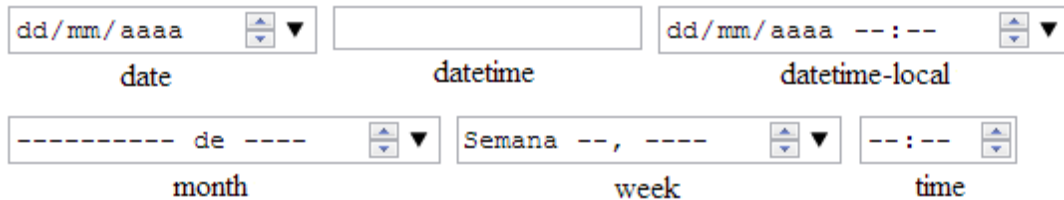


Figura 12: Vista de inputs para fecha y hora en Chrome

Selectores de color

Este nuevo tipo de campo permite al usuario seleccionar un color, y devuelve el valor hexadecimal del mismo. El selector de colores ofrece un número estándar de colores, además de la posibilidad de elegir cualquier otro de la paleta de colores del sistema operativo del usuario.

```
<input type="color" />
```

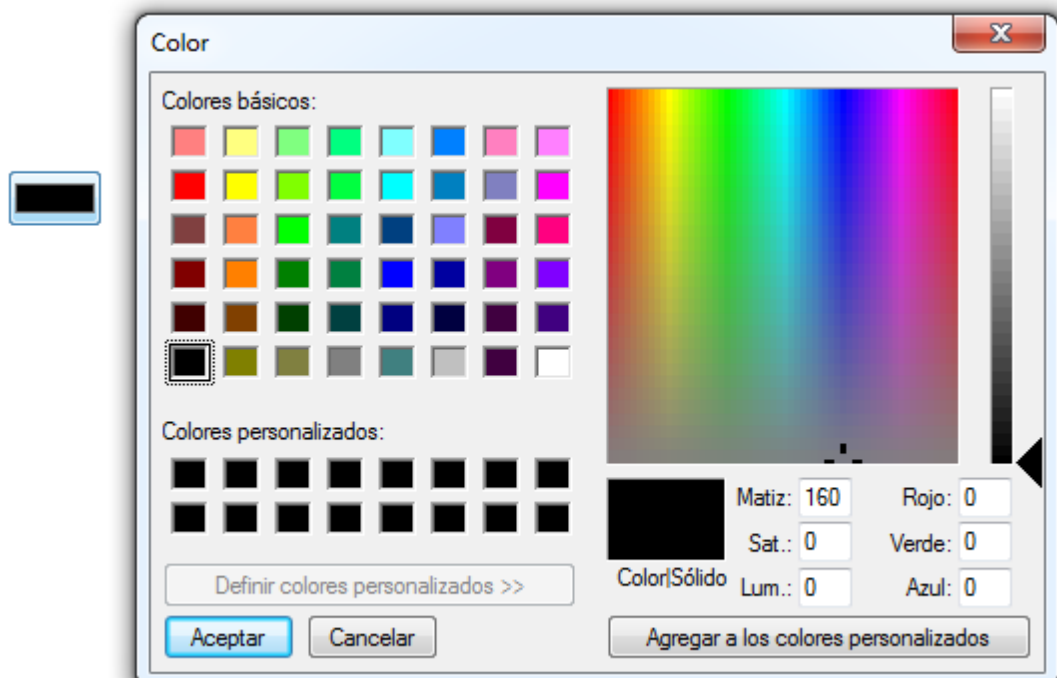


Figura 13: Vista del input de tipo color en Chrome

Nuevas funcionalidades

Las nuevas funcionalidades de la web en HTML5 proporcionan no sólo una mayor seguridad, sino también una mejor interacción. Algunas de las mejoras más destacables en este ámbito son:

- **Gestión del historial del navegador**

Todos los navegadores incluyen botones para retroceder y avanzar en la navegación, aunque en las anteriores versiones de HTML no era posible modificar el comportamiento de estos botones para que redirigieran a páginas concretas. Para controlar la navegación del usuario por la aplicación/página web era preciso el uso de JavaScript.

Sin embargo, HTML5 permite controlar la historia del navegador, es decir, dónde ha estado el usuario y dónde se encuentra ahora. Esto lo hace mediante el uso del objeto "**history**" que permite avanzar y retroceder, e incluso permite almacenar datos e información en el objeto "**state**" y que esa información sea posteriormente recuperada por el usuario cuando vuelva a esta página.

"**History**" es un objeto del documento que forma parte del objeto "**window**" en el navegador. Por este motivo, para hacer uso del objeto "**history**" es necesario seguir la notación propia del DOM: "**window.history.object**". Algunos de sus atributos y funciones son:

- **window.history.length**; Permite saber el tamaño del historial, es decir, el nº de entradas del navegador.
- **window.history.go(delta)**; Es una función general que acepta como argumento un parámetro, indicador del número de páginas que queremos movernos hacia delante o hacia atrás. Por ejemplo, con el valor -1 haremos que vuelva atrás una página. El parámetro es opcional, en caso de no existir recargará la página actual.
- **window.history.back()**; Retrocede a la página anterior. No acepta argumentos.
- **window.history.forward()**; Avanza a la página siguiente. No acepta argumentos.
- **window.history.pushState(data, title [,url])**; Permite incluir información en el historial. Para ello dispone de tres argumentos:

- **data**: datos que se desean almacenar.
- **title**: nombre con el que se van a referenciar esos datos.
- **url**: (parámetro opcional) indica la url en la que queremos que se referencien estos datos.
- **window.history.replaceState(data, title [,url]);** Permite reemplazar los datos guardados en el historial.
- **window.onpopstate;** Se trata de un evento que tiene lugar cuando los datos guardados en el historial se convierten en accesibles y se muestran.

Estas funciones permiten configurar la navegación a través de la web sin necesidad de JavaScript, acelerando con ello la navegación y permitiendo una mayor flexibilidad reduciendo el esfuerzo por parte del desarrollador.

- **Drag and Drop**

La función de arrastrar y soltar (*Drag and Drop* - o *DnD*) tiene una gran importancia en HTML5. En la especificación se define un mecanismo basado en eventos, el API de JavaScript y elementos de marcado adicionales para indicar que prácticamente cualquier tipo de elemento de una página se pueda arrastrar. La compatibilidad nativa del navegador con la función *DnD* permite ofrecer aplicaciones web más interactivas.

Hacer que un objeto se pueda arrastrar es muy sencillo. Solo hay que establecer el atributo **draggable=true** en el elemento que se quiere mover. La función de arrastre se puede habilitar prácticamente en cualquier elemento, incluidos archivos, imágenes, enlaces u otros nodos DOM. Si además se añade mediante CSS3 la propiedad “**cursor: move**” al elemento al que se aplica esta función, los usuarios recibirán una señal visual de que el elemento se puede mover.

Cabe destacar que en la mayoría de los navegadores los elementos de anclaje, las imágenes y las selecciones de texto que tienen un atributo **href** se pueden arrastrar de forma predeterminada.

Para organizar el flujo de *DnD*, es necesario un elemento de origen (en el que se origina el movimiento de arrastre), la carga de datos (lo que se desea mover) y un elemento de destino (el área en la que se soltarán los datos). El elemento de origen puede ser una imagen, una lista, un enlace, un objeto de archivo, un bloque de HTML o cualquier otra cosa. El elemento

de destino es la zona de arrastre (o un conjunto de zonas de arrastre) donde se aceptan los datos que el usuario intenta soltar. No todos los elementos pueden ser elementos de destino (por ejemplo, las imágenes).

Se deben tener en cuenta distintos eventos para controlar todo el proceso de arrastrar y soltar:

- **dragstart**: Se lanza cuando comienza el arrastrado del elemento.
- **drag**: Se lanza cada vez que se mueve el cursor y se está arrastrando un objeto.
- **dragenter**: Se lanza cuando el elemento arrastrado entra por primera vez en un elemento objetivo.
- **dragleave**: Se lanza cuando el objeto arrastrado deja de estar sobre un elemento objetivo.
- **dragover**: Se lanza cuando el objeto arrastrado se encuentra sobre un elemento objetivo.
- **drop**: Se lanza cuando el objeto arrastrado se suelta sobre un elemento objetivo.
- **dragend**: Se lanza cuando se deja de arrastrar el objeto.

La propiedad **dataTransfer** es el centro de desarrollo de toda la actividad de la función *DnD*, ya que contiene los datos que se envían en la acción de arrastre. Esta propiedad se establece en el evento **dragstart** y se lee y se procesa en el evento **drop**. Al activar **e.dataTransfer.setData(format, data)**, se establece el contenido del objeto en el tipo MIME y se transmite la carga de datos en forma de argumentos. **dataTransfer** expone una serie de propiedades para ofrecer señales visuales al usuario durante el proceso de arrastre. Estas propiedades también se pueden utilizar para controlar la respuesta de cada elemento de destino de la operación de arrastre a un determinado tipo de datos.

- **dataTransfer.effectAllowed**: Restringe el "tipo de arrastre" que puede realizar el usuario en el elemento. Se utiliza en el modelo de procesamiento de la operación de arrastrar y soltar para la inicialización de **dropEffect** durante los eventos **dragenter** y **dragover**. Esta propiedad admite los siguientes valores: **none**, **copy**, **copyLink**, **copyMove**, **link**, **linkMove**, **move**, **all** y **uninitialized**.
- **dataTransfer.dropEffect**: Controla la información que recibe el usuario durante los eventos **dragenter** y **dragover**. Cuando el usuario coloca el ratón sobre un elemento de destino, el cursor del navegador indica el tipo de operación que se va a realizar (por ejemplo, una operación de copia, un movimiento, etc.). La propiedad de efecto admite los siguientes valores: **none**, **copy**, **link** y **move**.

- **e.dataTransfer.setDragImage(img element, x, y)**: Establece un icono de arrastre en lugar de utilizar la información predeterminada del navegador.

Además de permitir mover elementos sobre la propia página web, la API de *DnD* permite arrastrar archivos del escritorio a una aplicación web en la ventana del navegador. Como ampliación de este concepto, Google Chrome permite arrastrar objetos de archivo del navegador al escritorio.

Para arrastrar un archivo desde el escritorio se deben utilizar los eventos de *DnD* del mismo modo que otros tipos de contenido. La diferencia principal se encuentra en el controlador **drop**. En lugar de utilizar **dataTransfer.getData()** para acceder a los archivos, sus datos se encuentran en la propiedad **dataTransfer.files**.

- **Almacenamiento**

Cuando los desarrolladores web quieren almacenar cualquier información del usuario, piensan inmediatamente en subir datos al servidor. Sin embargo, esto ha cambiado con HTML5, ya que actualmente existen varias tecnologías que permiten que las aplicaciones almacenen datos en los dispositivos cliente. Según lo que decida el desarrollador, la información puede sincronizarse también con el servidor o permanecer siempre en el cliente.

Hay varias razones por las que puede ser recomendable utilizar el almacenamiento en el cliente. En primer lugar, el almacenamiento en el cliente permite que una aplicación funcione cuando el usuario no está conectado, posiblemente sincronizando datos cuando vuelve a establecer conexión. En segundo lugar, aumenta el rendimiento, por lo que se puede mostrar una gran cantidad de datos en cuanto el usuario hace clic en el sitio en lugar de esperar a que vuelvan a descargarse. En tercer lugar, es un modelo de programación más sencillo que no requiere infraestructura de servidor. Por supuesto, los datos son más vulnerables y el usuario no puede acceder a ellos desde varios clientes, por lo que solo se debe utilizar para datos que no sean muy importantes, concretamente en versiones almacenadas en caché de datos específicos que también se encuentren "en la nube".

HTML5 ofrece cuatro tipos diferentes de almacenamiento de datos en una máquina de cliente local: almacenamiento local, almacenamiento de sesión, almacenamiento web SQL y base de datos indexada.

1. Web Storage:

El almacenamiento local y el de sesión son dos tipos diferentes de **Web Storage**. En el caso del almacenamiento local o “Local Storage” los datos almacenados no tienen fecha de caducidad, por lo que permanecen disponibles indefinidamente (se utiliza el objeto `localStorage`). Por el contrario, en el almacenamiento de sesión los datos almacenados sólo estarán disponibles durante la sesión de navegación, es decir, cuando ésta se cierre desaparecerán (se utiliza el objeto `sessionStorage`).

Aunque HTML 5 no fija nada acerca de la máxima capacidad, la mayor parte de los exploradores trabajan con hasta 5 MB de Web Storage.

Los datos se almacenan en pares de valores nombre/valor. La sintaxis de creación de una entrada de almacenamiento local o de sesión es similar:

```
localStorage[keyName] = data; | sessionStorage[keyName] = data;
```

Para la recuperación de los valores se sigue una sintaxis similar, pero inversa:

```
data = localStorage[keyName]; | data = sessionStorage[keyName];
```

También es soportada la sintaxis de punto (aunque no es la recomendada), de la forma:

```
localStorage.keyName = data; | sessionStorage.keyName = data;
```

Y para la recuperación de los valores:

```
data = localStorage.keyName; | data = sessionStorage.keyName;
```

La interface `Storage` define una serie de atributos y métodos con los que podremos realizar las diferentes acciones de almacenamiento y recuperación de datos (tanto para almacenamiento local como para el de sesión):

- **getItem(key)**: devuelve un string con el valor del elemento con clave `key`. Es equivalente a utilizar **`data = localStorage[keyName];`**
- **setItem(key, value)**: almacena un valor (`value`) referenciado por una clave (`key`). Es equivalente a utilizar **`localStorage[keyName] = data;`**
- **removeItem(key)**: elimina el par clave/valor con clave igual a `key`.
- **length**: atributo que contiene el número de elementos (pares clave/valor) almacenados.

- **key(index):** devuelve un string con la clave (no el valor) del elemento que ocupe la posición index dentro de la colección de datos.
- **clear():** elimina todos los elementos.

El valor almacenado para cada clave se guarda como una cadena de texto. Por ello, es recomendable utilizar JSON para convertir cualquier objeto a cadena antes de almacenarlo en el Web Storage, y así poder transformar de nuevo la cadena recuperada al objeto inicial.

En la aplicación del Proceso de 21 días, se ha utilizado el LocalStorage para almacenar la siguiente información:

- **Ejercicios realizados:** Se almacena el número del último ejercicio realizado para ofrecer al usuario el siguiente ejercicio pendiente y marcar aquellos que ya han sido completados.

```
var ejercicio = localStorage["hecho"];
```

- **Lista de notas:** Se almacena una cadena JSON que contiene los datos de cada una de las notas que el usuario ha apuntado en la libreta.

```
var notas = localStorage["notas"];
```

- **Lista de alarmas:** Se almacena una cadena JSON con todas las alarmas que haya programado el usuario y aún no se hayan lanzado.

```
var alarmas = localStorage["alarmas"];
```

- **Tono de las alarmas:** Se almacena un valor numérico que representa el tono personalizado por el usuario con el que se notificarán las alarmas.

```
var sonido = localStorage["sonido"];
```

2. Almacenamiento web SQL

HTML5 introduce la posibilidad de disponer de una base de datos local almacenada en el navegador del usuario. Mediante Web SQL Database, la W3C ofrece una API estándar destinada a manipular bases de datos en el lado del cliente mediante peticiones SQL de forma asíncrona.

Ejemplo de uso:

```
var db = window.openDatabase("DBName", "1.0", "description",
5*1024*1024); //5MB
db.transaction(function(tx) {
    tx.executeSql("SELECT * FROM test", [], successCallback,
errorCallback);
});
```

3. Base de datos indexada

Indexed Database se encuentra a medio camino entre Web Storage y Web SQL Database. Al igual que Web Storage, se trata de una sencilla asignación de valores y claves, pero admite índices similares a los de las bases de datos relacionales, por lo que la búsqueda de objetos que corresponden a un campo específico es rápida. Además, no es necesario repetir la acción de forma manual en todos los objetos almacenados.

c. Web semántica

La Web semántica comienza a ser realidad de la mano de HTML5. Con la introducción de nuevos elementos y conceptos que contribuyen a la expansión de la Web 3.0, HTML5 se ha transformado en un pilar indispensable de esta evolución.

Desde su concepción, HTML5 introduce etiquetas que permiten estructurar de una mejor manera las páginas Web, dotando a los elementos de un valor semántico. Estas etiquetas indican qué es el contenido que albergan, en lugar de cómo se debe formatear al mostrar el documento HTML en un navegador web.

Dentro del etiquetado semántico hay un conjunto de funciones que sirven para definir el esquema principal del documento, como HEADER, ARTICLE, FOOTER, etc. Todas esas etiquetas semánticas indican qué es el contenido que engloban y cuál es su relación con el conjunto de elementos del documento HTML:

Etiquetas semánticas estructurales

La aparición de estas nuevas etiquetas no sólo proporcionan una mayor coherencia al crear una organización del código, sino que además aportan una densidad mucho menor que las de su predecesor, HTML4, lo cual puede además favorecer al posicionamiento de la web en los buscadores.

El motivo es que muchas de estas etiquetas sustituyen a capas div que tenían un uso muy extendido entre los desarrolladores, por lo que sus creadores decidieron dotar a HTML5 de estas etiquetas para darle un mayor contenido semántico. De paso, como era obvio, se aprovechó y se simplificó la nomenclatura del código.

En las siguientes imágenes se puede comparar la estructura de una web antes y después de la aparición de estas nuevas etiquetas:

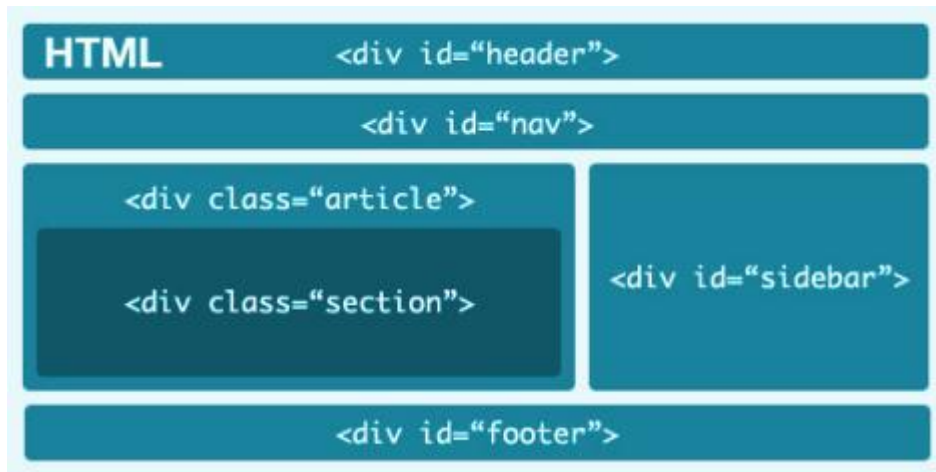


Figura 14: Layout previo a HTML5

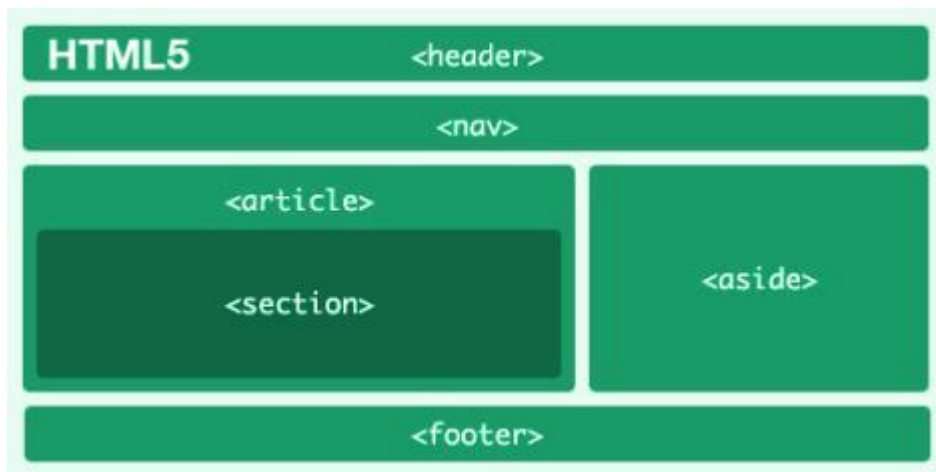


Figura 15: Nuevas etiquetas semánticas de HTML5

Así pues, las principales etiquetas semánticas son:

- **Section**

El nuevo elemento *section* tiene como cometido el agrupar elementos relacionados entre sí de forma temática. Es muy similar al uso que se le da al elemento div pero con la diferencia de que el elemento div no tiene ningún peso semántico y no nos informa

sobre el tipo de contenido que alberga. El elemento *section* se usa de forma explícita para agrupar contenido relacionado.

- **Header**

El elemento header (encabezamiento) representa la cabecera de una sección. Estas cabeceras pueden contener más que sólo el título, por ejemplo podría incluir subtítulos o información adicional.

- **Footer**

El elemento footer representa el pie de la sección a la que se la aplica. Un footer típicamente contiene información adicional sobre la sección en que se encuentra, como podría ser el autor, enlaces relacionados al documento, copyright, etc.

- **nav**

El elemento nav representa una sección para enlaces de navegación, útil para la navegación de un sitio o de una tabla de contenidos.

- **aside**

El elemento aside se puede usar para estructurar el contenido que está tangencialmente relacionado con el contenido de al lado; es decir, es útil para el marcado de las barras laterales (sidebars).

- **article**

El elemento article representa una sección independiente del documento web. Se recomienda utilizarlo en contenidos del tipo noticias o artículos de blogs, publicaciones en los foros o comentarios individuales.

Además de estas etiquetas, la web semántica cuenta con otros nuevos elementos para conseguir una mejora a dicho nivel de las aplicaciones web, como son los microdatos, los microformatos o el uso de RFDa:

Microdatos

Los microdatos son propiedades del etiquetado HTML5 que se utilizan para definir el significado del contenido de cada una de las etiquetas, es decir, su carga semántica. Para ello permiten especificar elementos personalizados en una página web mediante la sintaxis compuesta de pares nombre-valor con el contenido existente.

Para trabajar con microdatos se requieren al menos tres elementos:

- **<itemscope>**: Identifica al contenedor de los datos.
- **<itemprop>**: Representa al dato, es decir, lo utilizado para agregar la propiedad.
- **<itemtype>**: Define el tipo de metadato que se va a utilizar, según los establecidos por la organización `schema.org`.

A continuación se muestra un ejemplo de uso:

```
<div itemscope itemtype="<a href="http://www.data-vocabulary.org/Person/"
target="_blank">http://www.data-vocabulary.org/Person/</a>">
<p> Mi nombre es <span itemprop="name">Carmen Castillo</span></p>
</div>
```

Una de las principales ventajas del uso de los metadatos radica en el posicionamiento SEO de la página web. Al darle un significado semántico a la información introducida en el documento HTML, el motor de búsqueda puede llegar a integrar parte de esa información estructurada y mostrarla en los resultados, en lugar de mostrar simplemente el título de la página y un extracto del texto.

Microformatos

Un microformato es una etiqueta de código HTML con una serie de atributos para identificar el contenido, pero que le proporciona un valor semántico. Han sido diseñados pensando en primer lugar en la persona y el segundo lugar en la máquina, aunque pueden proporcionar grandes ventajas en el posicionamiento SEO de la web si se utilizan adecuadamente.

Un ejemplo básico del uso de microformatos es:

```
<a href="http://www.cualquierpagina.com" rel="tag">SEO</a>
```

Con la definición “tag” después de “rel” se le está dando relevancia sobre otros enlaces comunes que no contengan tag. A la hora que un robot de búsqueda pase por la página, interpretará con mayor importancia los enlaces que contengan tag y podrá formar una estructura de ellos según el contenido que muestren.

Algunos de los microformatos más extendidos son:

- **hCard**: para detalles de contactos.

```
<ul class="vcard">
  <li class="fn">Carmen Castillo</li>
  <li class="nickname">Car</li>
  <li class="org">Nombre de la compañía</li>
  <li class="tel">600-000-000</li>
  <li><a class="url" href="http://ejemplo.es/">url</a></li>
</ul>
```

- **hCalendar**: para eventos.

```
<p class="vevent">
  The <span class="summary">Evento organizado</span>
  el 15 de Enero de 2014, de
  <abbr class="dtstart" title="2001-01-15T14:00:00+06:00">2</abbr>a
  <abbr class="dtend" title="2001-01-15T16:00:00+06:00">4</abbr>pm
  en la
  <span class="location">Universidad de Almería</span>
</p>
```

- **hAtom**: para noticias y similares.
- **mark**: para marcar una determinada palabra o conjunto de palabras
<p>Esta será la palabra <mark>remarcada</mark></p>
- **time**: para fechas, horas o una combinación de ambas.
<time datetime="2011-09-12T13:30">12 de Septiembre del 2011 a las 1:30pm</time>
- **meter**: para marcar medidas definiendo que tales medidas son parte de una escala con unos valores máximos y mínimos.
<meter low="-10" high="100" min="12" max="50" optimum="26" value="25"></meter>

RDFa

La tecnología RDFa supone un conjunto de extensiones propuestas por la W3C para introducir mayor semántica en los documentos HTML utilizando atributos en etiquetas XHTML que hacen posible la introducción de infinidad de formatos. RDFa aprovecha atributos de los elementos meta y link de XHTML y los generaliza de forma que puedan ser utilizados en otros elementos.

Los atributos que se utilizan son

- **typeof**: indica de qué tipo es la instancia descrita.

- **about:** una URI que indica el recurso que describen los metadatos y que remite al documento actual por defecto.
- **rel, rev, href y resource:** atributos que establecen un relación o relación inversa con otro recurso.
- **property:** aporta una propiedad para el contenido de un elemento.
- **content:** atributo opcional que se sobrepone al contenido del elemento cuando se usa el atributo property.
- **datatype:** atributo opcional que indica el tipo de datos del contenido.

El uso de RDFa supone cinco ventajas principales:

1. **Independencia del editor:** cada sede web puede usar sus propios estándares.
2. **Reutilización de datos:** se debe tratar de no duplicar los datos; RDFa hace innecesario separar las secciones XML y HTML de los mismos contenidos.
3. **Autocontención:** las secciones de XML y HTML pueden mantenerse separadas.
4. **Modularidad** del esquema: Los atributos son reusables
5. **Escalabilidad:** se pueden añadir campos adicionales con la única condición de que se mantenga la capacidad de extraer semántica de los datos del archivo XHTML.

En el siguiente ejemplo se muestra el potencial de RDFa, que permite anotar el texto entrante con valores semánticos:

```
<p xmlns:dc="http://purl.org/dc/elements/1.1/"  
  about="http://www.example.com/books/wikinomics">  
  In his latest book  
  <em property="dc:title">Wikinomics</em>,  
  <span property="dc:creator">Don Tapscott</span>  
  explains deep changes in technology,  
  demographics and business.  
  The book is due to be published in  
  <span property="dc:date" content="2006-10-01">October 2006</span>.  
</p>
```

En definitiva, la web semántica propone un cambio en el concepto, pero también ofrece nuevas posibilidades para que los navegadores que trabajan en diversos dispositivos y también los motores de búsqueda, tengan más posibilidades para mejorar el resultado que ofrecen a los usuarios.

4.2. CSS3

CSS (Cascading Style Sheets) es un lenguaje de hojas de estilos que permite controlar la presentación semántica (el aspecto y el formato) de un documento escrito en HTML, XHTML o incluso XML. Proporciona la posibilidad de separar el contenido de su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

La primera especificación oficial de CSS, recomendada por la W3C fue CSS1, publicada en diciembre de 1996 y abandonada en abril de 2008. La segunda especificación, CSS2, fue desarrollada por la W3C y publicada como recomendación en mayo de 1998, aunque fue abandonada en la misma fecha que CSS1. La versión de CSS que utilizan todos los navegadores de hoy en día es CSS 2.1, una revisión de CSS 2 que fue publicada como recomendación oficial el 7 de junio de 2011.

Los primeros borradores de CSS3 fueron liberados en junio de 1999. Esta versión está dividida en varios documentos separados, llamados "módulos", que añaden nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad.

Las novedades de CSS3 permiten ahorrar tiempo y trabajo al poder seguir varias técnicas (bordes redondeados, sombra en el texto, sombra en las cajas, etc.) sin necesidad de usar un editor gráfico, aunque cuenta con la desventaja de no ser compatible 100% con ningún navegador.

Algunas de las novedades de CSS3 utilizadas en el desarrollo de este proyecto son las siguientes:

a. Animaciones

Las animaciones de CSS3 permiten animar la transición entre un estilo CSS y otro. Constan de dos componentes: un estilo que describe la animación y un conjunto de fotogramas que indican su estado inicial y final, así como posibles puntos intermedios en la misma.

Las animaciones CSS tienen tres ventajas principales sobre las técnicas tradicionales de animación basada en scripts:

- Es muy fácil crear animaciones sencillas, y no es necesario tener conocimientos de JavaScript.
- La animación se muestra correctamente, incluso en equipos poco potentes. En cambio animaciones simples desarrolladas con JavaScript pueden verse mal (a menos que estén muy bien programadas). El motor de renderizado puede usar técnicas de optimización como el "frame-skipping" u otras para conseguir que la animación se vea tan suave como sea posible.
- Dado que el navegador controla la secuencia de la animación, permitimos que optimice el rendimiento y la eficiencia de la misma, por ejemplo, reduciendo la frecuencia de actualización de animaciones que se ejecuten en pestañas que no estén visibles.

Para crear una secuencia de animación CSS usaremos la propiedad **animation** y sus subpropiedades. Con ellas podremos, no sólo configurar el ritmo y la duración de la animación, sino otros detalles sobre la propia secuencia.

Las subpropiedades de **animation** son:

- **animation-delay:** Tiempo de retardo entre el momento en que el elemento se carga y el comienzo de la secuencia de la animación.
- **animation-direction:** Indica si la animación debe retroceder hasta el fotograma de inicio al finalizar la secuencia o si debe comenzar desde el principio al llegar al final.
- **animation-duration:** Indica la cantidad de tiempo que la animación consume en completar su ciclo (duración).
- **animation-iteration-count:** El número de veces que se repite. Podemos indicar infinite para repetir la animación indefinidamente.
- **animation-name:** Especifica el nombre de la regla **@keyframes** que describe los fotogramas de la animación.
- **animation-play-state:** Permite pausar y reanudar la secuencia de la animación

- **animation-timing-function:** Indica el ritmo de la animación, es decir, como se muestran los fotogramas de la animación, estableciendo curvas de aceleración.
- **animation-fill-mode:** Especifica qué valores tendrán las propiedades después de finalizar la animación (los de antes de ejecutarla, los del último fotograma de la animación o ambos).

Una vez configurado el tiempo de la animación debemos definir su apariencia. Esto lo haremos estableciendo dos o más fotogramas usando la regla `@keyframes`. Cada fotograma describe cómo se muestra cada elemento animado en un momento dado durante la secuencia de la animación.

Desde que se define el tiempo y el ritmo de la animación, el fotograma usa `percentage` para indicar en qué momento tiene lugar la animación. 0% es el principio, 100% es el estado final de la animación. Debemos especificar estos dos momentos para que el navegador sepa dónde debe comenzar y finalizar. Debido a su importancia, estos dos momentos tienen alias especiales: `from` y `to`.

Además, opcionalmente se pueden incluir fotogramas que describan pasos intermedios entre el punto inicial y final de la animación.

En la aplicación desarrollada a lo largo de este proyecto, se utiliza esta característica de CSS3 para animar una serie de elementos HTML durante la reproducción de los posibles tonos seleccionables para las notificaciones. Para ello, en dichos contenedores se asignará la propiedad:

```
animation: loading 0.9s infinite alternate;
```

Que establece el nombre de la regla `@keyframes`, que se desarrollará durante 0.9 segundos de manera indefinida. Además asigna a la propiedad **animation-direction** el valor de **“alternate”**, indicando que la animación debe retroceder hasta el fotograma de inicio al finalizar la secuencia

```
@keyframes loading {
  0% {
    width: 10px;
    height: 10px;
    transform: translateZ(0);
  }
  100% {
    width: 24px;
    height: 24px;
    transform: translateZ(-21px);
  }
}
```

Esta regla establece un tamaño final para el elemento afectado mayor al tamaño inicial, trasladándolo 21px a lo largo del eje Z.

b. Bordes redondeados

Las características de CSS 3 incluyen bordes redondeados a través del atributo border-radius, que define la curvatura que debe tener el borde del elemento. La sintaxis de esta propiedad es la siguiente:

```
border-radius: 1-4 length|% / 1-4 length|%;
```

Se pueden establecer valores diferentes para el radio de cada una de las esquinas, aunque utilizando un único valor, el navegador establecerá el mismo radio para cada una de las esquinas del elemento.

Tanto para su uso en navegadores basados en Webkit (como Google Chrome o Safari) como para Mozilla es necesario usar el prefijo -webkit o -moz respectivamente a la hora de utilizar este atributo.

Los botones desarrollados para la aplicación móvil han hecho uso de esta propiedad de la siguiente manera:

```
#botonDialogo {
  border: 0.3em solid #132144;
  border-radius: 0px 40px;
  -moz-border-radius: 0px 40px;
  -webkit-border-radius: 0px 40px;
}
```



Figura 16: Border-radius CSS3

c. Efectos de sombra

Para añadir efectos de sombra sobre un texto CSS cuenta con el atributo `text-shadow` desde su versión CSS2. No obstante, hasta la llegada de CSS3 no había sido implementado dentro de los navegadores más comunes, por lo que para lograr tales resultados a menudo había que recurrir a programas de diseño gráfico.

La sintaxis de la propiedad es la siguiente:

`text-shadow: h-shadow v-shadow blur color;`

Consta por tanto de cuatro valores, que son:

- El desplazamiento horizontal de la sombra. Un desplazamiento negativo pondrá la sombra a la izquierda.
- El desplazamiento vertical. Un valor negativo dispone la sombra en la parte superior del texto, uno positivo la sombra estará por debajo del texto.
- El tercer parámetro es el radio de desenfoque. Con un valor igual a 0 la sombra será fuerte y con color liso. Cuanto mayor sea este valor más borrosa será la sombra.
- El último parámetro es el color de la sombra.

La propiedad `text-shadow` puede aplicar una o más sombras a un mismo texto.

Los motivos por los que es preferible utilizar CSS en lugar de imágenes para lograr sombras sobre el texto son principalmente dos:

En primer lugar, el uso de imágenes aumenta el consumo de ancho de banda y de conexiones HTTP. Al utilizar texto en su lugar se mejora la accesibilidad, tanto para las personas que utilizan lectores de pantalla como para los motores de búsqueda. Además, el zoom de la página funcionará mejor porque el texto se puede ampliar en lugar de utilizar la interpolación de píxeles para aumentar la escala de una imagen.

En segundo lugar esta propiedad es compatible en muchos navegadores:

- **Opera** implementa `text-shadow` desde la versión 9.5. Según el Centro de desarrolladores de Mozilla, Opera 9.x permite hasta 6 sombras sobre el mismo elemento.

- **Safari** ha tenido esta función desde la versión 1.1 (y otros navegadores basados en WebKit junto con él).
- **Internet Explorer** no es compatible con la propiedad `text-shadow`, pero se transforma en texto normal sin generar problemas. Además, si se desea emular algunas de las funciones de `text-shadow` en MSIE, se pueden utilizar los filtros propietarios "Shadow" y "DropShadow" de Microsoft.

Al igual que en MSIE, cuando otros navegadores más antiguos no soporten la característica (incluyendo Firefox 3 y más antiguos), se mostrará el texto normal, sin sombras.

Una advertencia que conviene mencionar es el "orden de dibujo": En el caso de Opera 9.x, la sombra que se especifica primero se dibuja abajo, siguiendo el orden de dibujo de CSS2. Sin embargo, en Firefox 3.5 la primera sombra que se especifica se dibuja en la parte superior, según el orden de dibujo de CSS3.

En la aplicación móvil se utiliza una sombra simple en algunos elementos, como por ejemplo el título inicial, de la siguiente manera:

```
.titulo {  
    text-shadow: 5px 5px 5px #D4D4D4;  
}
```

Proceso de 21 días

Figura 17: Text-shadow CSS3

d. Múltiples columnas

Antes de que CSS3 introdujese la posibilidad de crear diseños de varias columnas de manera rápida y sencilla solían utilizarse tablas para lograr este fin. No obstante, esta nueva característica presenta como principal ventaja la flexibilidad: el contenido fluye de una columna a otra, y el número de columnas puede variar en función del tamaño del medio en el que el contenido será visualizado.

Las propiedades de esta característica son las siguientes:

- **column-count**: Especifica el número de columnas en que se dividirá el elemento.
- **column-fill**: Determina cómo rellenar las columnas: de manera balanceada (las columnas tendrán aproximadamente el mismo tamaño) o de manera automática (se van rellenando de forma secuencial, pudiendo tener por tanto diferentes tamaños).
- **column-gap**: Establece el tamaño del espacio entre las columnas.

- **column-rule-color**: Especifica el color del borde que separa las columnas.
- **column-rule-style**: Establece el estilo de la línea entre las columnas. Puede adoptar los siguientes valores: *none*, *hidden*, *dotted*, *dashed*, *solid*, *double*, *groove*, *ridge*, *inset* o *outset*;
- **column-rule-width**: Representa el ancho de la línea que separa las columnas.
- **column-span**: Especifica cómo debe expandirse el elemento a través de las columnas.
- **column-width**: Determina el ancho de las columnas.

Esta propiedad se ha utilizado en la aplicación para reorganizar los elementos en función de la orientación del dispositivo, de manera que a través de las media queries, se establecen 4 o 2 columnas para albergar cierto contenido:

```
@media only screen and (orientation:landscape) {
  #bontonesEjercicio{
    -moz-column-count: 4;
    -webkit-column-count: 4;
    column-count: 4;
  }
}
```

e. Pseudoclasses

Una pseudo-clase es un estado o uso predefinido de un elemento al que se le puede aplicar un estilo independientemente de su estado por defecto. Existen cuatro tipos diferentes de pseudo-clases:

- **Enlace**: **:link** y **:visited** Se usan para dar estilo al enlace tanto en su estado por defecto como cuando ya ha sido visitado o cuando hacemos click en él.
- **Dinámicas**: **:hover**, **:active** y **:focus**. Estas pseudo-clases pueden ser aplicadas a cualquier elemento para definir cómo se muestran cuando el cursor está situado sobre él, haciendo click en él o bien cuando es seleccionado.
- **Estructurales**: Permiten dar estilo a elementos basándonos en una posición numérica exacta del elemento.
- **Otras**: Algunos elementos pueden ser estilizados de manera diferente basándonos, por ejemplo, en el lenguaje (**:lang()**)

CSS 3 añade tres nuevos selectores de atributos:

- **elemento[atributo^="valor"]**, selecciona todos los elementos que disponen de ese atributo y cuyo valor comienza exactamente por la cadena de texto indicada.
- **elemento[atributo\$="valor"]**, selecciona todos los elementos que disponen de ese atributo y cuyo valor termina exactamente por la cadena de texto indicada.
- **elemento[atributo*="valor"]**, selecciona todos los elementos que disponen de ese atributo y cuyo valor contiene la cadena de texto indicada.

Además de los selectores de atributos, incluye las siguientes novedades:

- **:checked**: Se aplica a cada elemento de tipo <input> que haya sido seleccionado.
- **:disabled**: Se aplica a cada elemento de tipo <input> que esté deshabilitado.
- **:empty**: Representa los elementos vacíos, es decir, aquellos sin hijos (incluyendo como tales a los nodos de texto).
- **:enabled**: Se aplica a cada elemento de tipo <input> que esté habilitado.
- **:first-of-type**: Establece las reglas de estilo para el primer elemento de la lista de elementos de un cierto tipo, de su padre.
- **:last-child**: Especifica el estilo del último elemento de su padre.
- **:last-of-type**: Establece las reglas de estilo para el último elemento de la lista de elementos secundarios de su padre.
- **:not**: Establece las reglas de estilo para los elementos que no contienen el selector especificado.
- **:nth-child**: Se utiliza sobre el elemento que ocupe una determinada posición según el orden de los elementos en el modelo DOM.
- **:nth-last-child**: Se utiliza sobre el elemento que ocupe una posición dada a partir del último hijo del selector según el orden de los elementos en el modelo DOM.
- **:nth-last-of-type**: Igual que **:nth-last-child** pero teniendo en cuenta el tipo del selector en lugar de sus hijos.

- **:nth-of-type**: Igual que **:nth-child** pero teniendo en cuenta el tipo del selector en lugar de sus hijo.
- **:only-child**: Se aplica cada elemento que resulte ser hijo único de su padre.
- **:only-of-type**: Igual que **:only-child** pero considerando que además sean del mismo tipo que el padre.
- **:root**: Especifica el elemento raíz del documento.
- **::selection**: Se aplica sobre la porción del element que es seleccionada por el usuario.
- **:target**: Se aplica sobre el elemento que se encuentre activo en ese momento.

En la práctica, se han utilizado pseudo-classes como por ejemplo **:nth-of-type** para establecer la animación de determinados elementos:

```
.loadingdiv:nth-of-type(2) {  
    background: #9D002C;  
    -webkit-animation-delay: 0.2s;  
    -moz-animation-delay: 0.2s;  
}
```

En este caso se selecciona el segundo hijo del div “.loadingdiv”, se cambia su color de fondo y se establece un tiempo de retardo para la animación de 0.2 segundos.

f. Tipografías

Hace unos años, la mejor solución para que el tipo de fuente definido por el diseñador web fuera respetado era utilizar alguna de las definidas como “safety fonts” o “fuentes comunes”, que están instaladas en la mayoría de los ordenadores de los usuarios que utilicen la aplicación. Para resolver esta limitación, CSS incluye la regla: **@font-face**, que pretende ser el medio para que el texto de una web pueda ser mostrado con una determinada tipografía con independencia de que el visitante la tenga instalada en su máquina o no.

Sin embargo ésta no es una característica propia de CSS3, ya que **@font-face** fue propuesta para CSS2 y ha sido implementada en Internet Explorer desde su versión 5. El problema es que su implementación recaía en el formato propietario Embedded Open Type (.eot), y ningún otro navegador decidió usarlo. Actualmente, y dado el cada vez más extendido uso de CSS3, esta regla permite la utilización de otros formatos como TrueType (.ttf) u OpenType (.otf).

Para utilizar esta regla el primer paso consiste en obtener los archivos relativos a las fuentes que se quieran utilizar. Una excelente herramienta para este fin es “**FontSquirrel**”, una aplicación web que incluye su propio generador de fuentes en línea. A partir de esta web se pueden obtener, no sólo los archivos de las fuentes, sino también la propia regla `@font-face`, y unas páginas html con la tipografía seleccionada para verificar el diseño del resultado final.

Una vez obtenidos los archivos necesarios, deben incluirse en el proyecto y utilizar la regla `@font-face` para enlazarlos siguiendo la siguiente sintaxis:

```
@font-face
{
    font-properties
}
```

Las propiedades de esta regla son las siguientes:

- **font-family:** Define el nombre con el que identificaremos a la fuente.
- **src:** Establece la URL donde se encuentra el archivo anteriormente descargado.
- **font-stretch:** Los posibles valores son: **normal**, **condensed**, **ultra-condensed**, **extra-condensed**, **semi-condensed**, **expanded**, **semi-expanded**, **extra-expanded**, **ultra-expanded**. El valor por defecto es "normal".
- **font-style:** Los posibles valores son: **normal**, **italic**, **oblique**. El valor por defecto es "normal".
- **font-weight:** Establece el tamaño de la letra. El valor por defecto es "normal".
- **unicode-range:** Define el rango de caracteres unicode que soporta la fuente. El valor por defecto es "U+0-10FFFF"

4.3. JavaScript

JavaScript es un lenguaje de script multiplataforma [cross-platform] orientado a objetos. Es un lenguaje pequeño y ligero, diseñado para una fácil incrustación en otros productos y aplicaciones, tales como los navegadores web. Además, se trata de un lenguaje interpretado, es decir, no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

El núcleo de JavaScript contiene un conjunto central de objetos, tales como **Array**, **Date** o **Math**, además de un conjunto central de elementos del lenguaje tales como los operadores, estructuras de control y sentencias. El núcleo de JavaScript puede ser extendido para una variedad de propósitos complementándolo con objetos adicionales; por ejemplo:

- **JavaScript del lado Cliente** extiende el núcleo del lenguaje proporcionando objetos para el control del navegador y su Document Object Model (DOM). Por ejemplo, las extensiones del lado del cliente permiten a una aplicación ubicar elementos en un formulario HTML y responder a eventos de usuario tales como las pulsaciones del ratón, entradas del formulario o navegación entre páginas.
- **JavaScript del lado Servidor** extiende el lenguaje proporcionando objetos relevantes para la ejecución de JavaScript en un servidor. Por ejemplo, las extensiones del lado del servidor permiten que una aplicación se comunique con una base de datos relacional, que la información se preserve de una llamada de la aplicación a otra, o que se puedan manipular archivos en un servidor.

JavaScript posee un **modelo basado en el prototipado** de objetos en lugar del modelo general basado en clases de objetos. El modelo basado en prototipos provee herencia dinámica, esto es, que la herencia puede variar para objetos individuales. JavaScript también soporta funciones sin ningún requerimiento declarativo especial. Las funciones pueden ser propiedades de los objetos al ejecutarse vagamente como tipos de métodos.

a. JavaScript y la especificación ECMA

Netscape desarrolló el lenguaje JavaScript, que se utilizó por primera vez en sus navegadores. Sin embargo, Netscape está trabajando con ECMA International - la asociación Europea para la estandarización de la información y de los sistemas de comunicación - para la entrega de un lenguaje de programación internacional estandarizado basado en el núcleo de JavaScript. Esta versión estandarizada de JavaScript, llamada ECMAScript, se comporta de la misma manera en todas las aplicaciones que soportan el estándar. Las compañías pueden usar el estándar abierto del lenguaje para desarrollar sus implementaciones en JavaScript. El estándar ECMAScript está documentado en la especificación ECMA-262.

A partir de 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, aunque los navegadores más antiguos soportan al menos la versión ECMAScript 3.

b. Sintaxis y utilización de JavaScript

JavaScript reconoce los siguientes tipos de valores:

- **Números.**
- **Valores Lógicos** (Booleanos).
- **Cadenas.**
- **null**: una palabra especial que denota un valor nulo; null es además un valor primitivo. Debido a que JavaScript discrimina mayúsculas y minúsculas [case-sensitive], no es lo mismo null que Null, NULL, o cualquier otra variación.
- **undefined**: una propiedad de alto nivel cuyo valor es indefinido; undefined es también un valor primitivo.

No existe una diferencia explícita entre números enteros y reales. Ni hay implícitamente un tipo de dato de fecha en JavaScript, aunque se puede usar el objeto Date y sus métodos para manejar fechas.

Al tratarse de un lenguaje dinámico no es necesario especificar el tipo de dato de una variable cuando ésta es declarada, ya que las variables son convertidas automáticamente cuando sea necesario durante la ejecución del script. Así, por ejemplo, se puede definir una variable como sigue a continuación:

```
var vble = 42
```

Y luego, asignar a la misma variable un valor de cadena, por ejemplo:

```
vble = "Ahora soy de tipo cadena"
```

Ya que JavaScript asigna tipos dinámicamente, esta asignación no provoca un mensaje de error.

En expresiones que incluyen valores numéricos y de cadena con el operador '+', JavaScript convierte los valores numéricos a cadena.

Cuando se declara una variable fuera de cualquier función, se le llama *variable global*, porque está disponible para cualquier sección del código en el documento. Cuando se declara una variable dentro de una función, se le llama *variable local*, porque está disponible sólo dentro de esa función.

Una variable puede ser declarada de dos maneras:

- **Con la palabra clave: var.** Por ejemplo, `var x = 42`. Esta sintaxis se puede usar para declarar variables locales y globales. Si no se especifica un valor inicial, la variable tendrá un valor de "undefined"
- **Simplemente asignándole un valor.** Por ejemplo, `x = 42`. Esto siempre declara una variable global y genera una advertencia estricta [strict warning] de JavaScript.

A parte de estos valores, los objetos y las funciones son también elementos fundamentales en el lenguaje, así como las expresiones y los operadores.

JavaScript tiene los siguientes tipos de operadores:

- Operadores de asignación
- Operadores de comparación
- Operadores aritméticos
- Operadores sobre bits
- Operadores lógicos
- Operadores de cadenas de caracteres (string)
- Operadores especiales

Además cuenta con las siguientes expresiones:

- **Aritméticas:** se evalúan como un número, por ejemplo 3.14159. (Generalmente usan Operadores aritméticos.)
- **De cadena:** se evalúan como una cadena de caracteres, por ejemplo, "Fred" o "234". (Generalmente usan Operadores de cadena de caracteres (string).)
- **Lógicos:** se evalúan como verdadero o falso. (A menudo emplean Operadores lógicos.)
- **De objeto:** se evalúan como un objeto.

c. [Lenguajes basados en Clases vs. Lenguajes basados en Prototipos](#)

La mayoría de los lenguajes de programación orientados a objetos están basados en la utilización de clases, sin embargo, los lenguajes basados en prototipos no utilizan ninguna noción formal de clase, los objetos se crean mediante un proceso de copia o clonación, a partir de otros objetos ya existentes o prototipos. Los lenguajes basados en prototipos prescinden de las clases, y sus funciones se llevan a cabo mediante mecanismos alternativos.

Uno de los principales papeles de una clase es definir la estructura y comportamiento de un conjunto de objetos. Mediante una clase se detallan los atributos y métodos que compartirán

todas sus instancias y que determinan su estado y comportamiento. Los objetos se crean siempre a partir de una clase, una plantilla a partir de la cual se construyen nuevos objetos

Los prototipos constituyen un nuevo concepto en el modelo de objetos, que es utilizado en sustitución de las clases para la construcción de objetos. La estructura y comportamiento de un conjunto de objetos no se describe mediante una clase, sino a través de un prototipo, que no es más que un objeto prefabricado, con una estructura, contenido y comportamiento predefinido, que se utiliza para crear nuevos objetos mediante un proceso de copia.

En los lenguajes basados en clases, los objetos se crean mediante un algoritmo (Figura 18). Cada clase determina la estructura, el comportamiento, y los valores iniciales de las variables de instancia del objeto recientemente creado. Sólo pueden crearse objetos de clases existentes y para crear un objeto con una estructura y/o comportamiento diferente es obligatorio definir previamente una nueva clase.

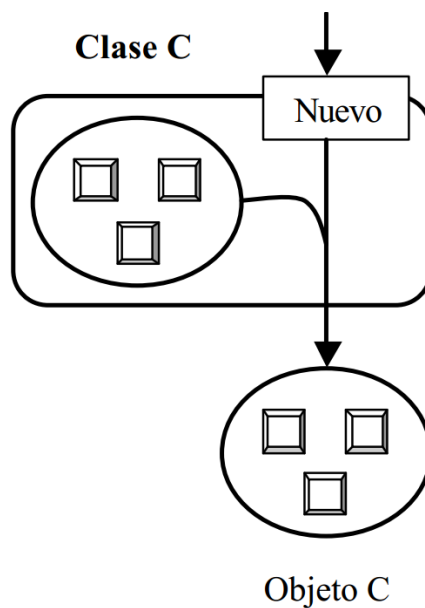


Figura 18: Creación de objetos con clases

En contraste, los prototipos son ya objetos inicializados, ocupan almacenamiento y tienen estado propio. Cada nuevo objeto se crea copiando un prototipo (Figura 19), proceso a través del cual adquiere la misma estructura, estado y comportamiento que el prototipo de partida. Posteriormente puede adaptarse mediante cambios individuales.

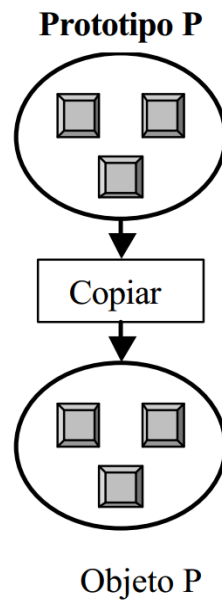


Figura 19: Creación de objetos con prototipos

Las clases son estáticas, cuando comienza la ejecución de un programa basado en clases no existe ningún objeto. Los prototipos son dinámicos, cuando una aplicación basada en prototipos arranca, necesita que ya estén creados por lo menos un prototipo de cada uno de los diferentes tipos de objetos básicos que se van a utilizar para poder realizar las pertinentes copias. Esto implica que se necesita un mecanismo para que cada vez que se invoca la aplicación se puedan usar los prototipos. Típicamente, esto se consigue mediante objetos persistentes que pueden activarse o almacenarse para su uso posterior.

Las principales ventajas del uso de lenguajes basados en prototipos frente a los lenguajes basados en clases son las siguientes:

- ✓ **Permiten cambios individuales.** En los sistemas basados en clases todos los objetos de una misma clase tienen las mismas propiedades, por lo que cuando se realiza cualquier cambio sobre dicha clase se modifican todas sus instancias. Sin embargo, en los lenguajes basados en prototipos es posible realizar cambios individuales a objetos sin afectar a otros objetos. Esta adaptación de cada objeto permite una especialización individual requerida en numerosas ocasiones en la programación orientada a objetos.
- ✓ **Facilitan la construcción de redes de objetos.** Un programa orientado a objetos se caracteriza porque está formado por una red de objetos que interactúan entre sí. Es importante, por tanto, no sólo la descripción de los objetos, sino cómo se conectan, cuándo se crean, qué relaciones se establecen entre ellos, etc. Como las clases son descripciones estáticas se necesita un esfuerzo extra para describir el comportamiento

dinámico de la red de objetos que conforman el sistema. En los sistemas basados en prototipos, se utiliza un acercamiento más directo, ya que los objetos se crean realmente y es más fácil enlazar unos con otros.

- ✓ **Permiten comportamiento dinámico.** Una característica que dota de mayor flexibilidad a los sistemas orientados a objetos es la posibilidad de cambiar la estructura y/o el comportamiento de los objetos en tiempo de ejecución. En lenguajes basados en clases esto sólo es posible si se dispone de algún mecanismo para modificar la propia clase. En lenguajes como C++, no hay nada parecido y para modificar la clase hay que modificar el código fuente. Hay lenguajes basados en prototipos donde es posible cambiar dinámicamente el comportamiento de un objeto simplemente cambiando su padre.
- ✓ **Adecuados para entornos distribuidos.** Los prototipos contienen tanto su estado como su comportamiento, y son independientes de las restricciones de las clases. Esto los hace especialmente útiles en situaciones en las que los objetos deben migrar de un nodo a otro. En un lenguaje basado en clases, un objeto consta de dos partes: su instancia y su clase, sin alguna de las dos el objeto está incompleto (sin la instancia no tiene estado y sin la clase no tendría comportamiento). En un sistema distribuido, esto plantea el problema de que siempre que se maneje un objeto, hay que manejar la clase a la que pertenece. Si por ejemplo, el objeto obj que pertenece a la clase clase_obj, se envía de un entorno local a un entorno remoto, o dicho entorno remoto dispone en su jerarquía de una copia de la clase clase_obj, o esta clase debe enviarse junto con el objeto.

No obstante, los lenguajes basados en prototipos presentan las siguientes limitaciones:

- × **Implementación complicada.** Los prototipos dan lugar a numerosos problemas técnicos que necesitan resolverse y que hacen que las herramientas de programación no sean muy eficientes. Es difícil conservar la identidad de los objetos y las relaciones entre ellos, ya que continuamente se están manipulando prototipos y una modificación a uno de ellos puede tener un efecto adverso en objetos creados con posterioridad.
- × **Falta de herramientas apropiadas.** La mayoría de los lenguajes basados en prototipos existentes hoy en día son proyectos de investigación, que no han sido distribuidos comercialmente y que por tanto no se han utilizado por un gran número de programadores.

- × **Falta de metodologías.** Para conseguir que los sistemas basados en prototipos estén preparados para el desarrollo de aplicaciones es necesario disponer de metodologías y herramientas automáticas que guíen el proceso de desarrollo de software. La principal cuestión que hay que tener en cuenta para resolver estos problemas es decidir si se pueden adaptar las metodologías actuales basadas en clases o si por el contrario interesa crear nuevas metodologías que partan de cero y se centren en los aspectos básicos de los sistemas basados en prototipos.

d. JQuery

jQuery es una librería JavaScript que simplifica la creación de páginas web interactivas. Funciona en todos los navegadores modernos y abstrae características específicas de cada uno de ellos, permitiendo al desarrollador enfocarse en el diseño y el resultado final, en lugar de tratar de desarrollar funciones complejas en navegadores individuales.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. De esta forma jQuery proporciona una mayor facilidad a la hora de:

- Buscar y manipular el contenido en una página HTML.
- Trabajar con el modelo de eventos de los navegadores modernos.
- Añadir efectos y transiciones sofisticadas, como animaciones disparadas por eventos.

Para incluir jQuery en un proyecto basta con descargarlo desde su página web oficial y referenciarlo en el documento que vaya a hacer uso de la librería de la siguiente manera:

```
<script src="jquery.js"></script>
```

A continuación se describen algunos de los aspectos claves en el uso de jQuery:

Funciones y selectores

El concepto más básico de jQuery es el de seleccionar algunos elementos y realizar acciones con ellos. La biblioteca soporta gran parte de los selectores CSS3 y algunos más no estandarizados.

Algunas de las técnicas más comunes para la selección de elementos son:

- **Selección de elementos en base a su ID**

```
$('#myId');
```

- **Selección de elementos en base al nombre de clase**

```
$('.div.myClass'); // si se especifica el tipo de elemento,  
// se mejora el rendimiento de la selección
```

- **Selección de elementos por su atributo**

```
$('.input[name=first_name]');
```

- **Selección de elementos en forma de selector CSS**

```
$('#contents ul.people li');
```

- **Pseudo-selectores**

```
$('.a.external:first'); // selecciona el primer elemento <a>  
// con la clase 'external'  
$('.tr:odd'); // selecciona todos los elementos <tr>  
// impares de una tabla  
$('#myForm :input'); // selecciona todos los elementos del tipo input  
// dentro del formulario #myForm  
$('.div:visible'); // selecciona todos los divs visibles  
$('.div:gt(2)'); // selecciona todos los divs excepto los  
// tres primeros  
$('.div:animated'); // selecciona todos los divs actualmente animados
```

Una vez realizada la selección de los elementos, es posible utilizarlos en conjunto con diferentes métodos. Éstos, generalmente, son de dos tipos: “getters” y “setters”. Los primeros devuelven una propiedad del elemento seleccionado mientras que los segundos fijan una propiedad para todos los elementos seleccionados.

Modificar elementos de HTML

jQuery permite redefinir los estilos y dimensiones de los objetos de una página web accediendo a sus propiedades CSS. Para ello basta con hacer uso del método `.css()` de la forma: `$('.h1').css('fontSize');`. El ejemplo anterior devuelve una cadena de caracteres, de la forma “19px”.

Para establecer varias propiedades CSS en un mismo elemento se puede usar la siguiente sintaxis:

```
$('#h1').css({
    'fontSize' : '100px',
    'color' : 'red'
});
```

A partir de la versión 1.6 de jQuery, utilizando `$.fn.css` también es posible establecer valores relativos en las propiedades CSS de un elemento determinado:

```
$('#h1').css({
    'fontSize' : '+=15px', // suma 15px al tamaño original del
                          // elemento
    'paddingTop' : '+=20px' // suma 20px al padding superior
                          // original del elemento
});
```

Para modificar las dimensiones de los objetos HTML se pueden usar también otros métodos de jQuery como son: `.height()`, `.innerHeight()`, `.innerWidth()`, `.outerHeight()`, `.outerWidth()` o `.width()`.

También es posible cambiar, mover, eliminar y duplicar elementos HTML, así como crear otros nuevos a través de una sintaxis sencilla. La documentación completa sobre los métodos de manipulación de jQuery se puede encontrar en la sección *Manipulation* de su API[40] <http://api.jquery.com/category/manipulation/>.

Eventos

jQuery provee métodos para asociar controladores de eventos a selectores. Cuando un evento ocurre se ejecuta la función provista. Dentro de la función, la palabra clave **this** hace referencia al elemento en que ocurre el evento.

Entre los métodos provistos se encuentran `$.fn.click`, `$.fn.focus`, `$.fn.blur`, `$.fn.change`, etc. Estos últimos son formas reducidas del método `$.fn.on` de jQuery (`$.fn.bind` en versiones anteriores a jQuery 1.7). El método `$.fn.on` es útil para vincular la misma función de controlador a múltiples eventos, para cuando se desea proveer información al controlador de evento, cuando se está trabajando con eventos personalizados o cuando se desea pasar un objeto a múltiples eventos y controladores.

Para vincular un evento utilizando un método reducido se utiliza una sintaxis como la siguiente:

```
$('#p').click(function() {
    console.log('click');
});
```

Para desvincular un controlador de evento se puede utilizar el método **\$.fn.off**, pasándole como argumento el tipo de evento a desconectar.

```
$('#p').off('click');
```

Para más detalles sobre los eventos en jQuery, se puede consultar el apartado *Events* de su API[40]. <http://api.jquery.com/category/events/>.

Efectos

Con jQuery, agregar efectos a una página es muy fácil. Estos efectos poseen una configuración predeterminada pero también es posible utilizar parámetros personalizados. Además es posible crear animaciones particulares estableciendo valores de propiedades CSS.

Los efectos más utilizados vienen incorporados dentro de la biblioteca en forma de métodos:

- **\$.fn.show**
Muestra el elemento seleccionado.
- **\$.fn.hide**
Oculta el elemento seleccionado.
- **\$.fn.fadeIn**
De forma animada, cambia la opacidad del elemento seleccionado al 100%.
- **\$.fn.fadeOut**
De forma animada, cambia la opacidad del elemento seleccionado al 0%.
- **\$.fn.slideDown**
Muestra el elemento seleccionado con un movimiento de deslizamiento vertical.
- **\$.fn.slideUp**
Oculta el elemento seleccionado con un movimiento de deslizamiento vertical.
- **\$.fn.slideToggle**

- Muestra u oculta el elemento seleccionado con un movimiento de deslizamiento vertical, dependiendo de si actualmente el elemento está visible o no.

Uso básico de un efecto incorporado

```
$('#h1').show();
```

Con la excepción de `$.fn.show` y `$.fn.hide`, todos los métodos tienen una duración predeterminada de la animación en 400ms. Este valor es posible cambiarlo a través de los argumentos de la función:

```
$('#h1').fadeIn(300); // desvanecimiento en 300ms  
$('#h1').fadeOut('slow'); // utiliza una definición de velocidad interna
```

Para una completa documentación sobre los diferentes tipos de efectos se puede visitar la sección *Effects* de la API de jQuery[40] <http://api.jquery.com/category/effects/>.

5. Técnicas de diseño y desarrollo

Para elaborar la aplicación de este proyecto se han tomado una serie de decisiones estratégicas que faciliten tanto la usabilidad como el desarrollo y el diseño de la misma. La más importante ha sido el uso de la técnica de Responsive Web Design para hacer la aplicación accesible desde cualquier dispositivo móvil sin que las diferentes resoluciones de estos interfieran con la estética del proyecto.

5.1. Responsive Web Design

Actualmente hay cada vez más diferencias entre los dispositivos capaces de acceder a una aplicación web. Se pueden utilizar ordenadores de sobremesa, portátiles, tablets, smartphones o incluso Smart TVs para navegar por internet o utilizar aplicaciones, y cada uno de estos aparatos cuenta con una pantalla de diferentes medidas. Pero no sólo se trata del tamaño, cada nueva generación de dispositivos modifica y mejora la resolución de la anterior, altera sus proporciones o implementa una nueva tecnología, de manera que conviven cuantiosos aparatos que aun siendo capaces de realizar la misma tarea, ofrecen representaciones muy diferentes de la misma.

Esta enorme variedad de dispositivos hace impracticable la realización de diferentes versiones de una misma aplicación para cada uno de ellos, motivo por el cual surgen técnicas como el RWD para ayudar a mejorar la experiencia de usuario.

El Responsive Web Design (RWD) o Diseño web adaptable, es una estrategia de diseño que persigue como único objetivo que una aplicación web sea capaz de adaptarse al medio a través del cual el usuario esté accediendo a la misma. Se centra en el uso de HTML y CSS para distribuir el contenido de la aplicación de forma armoniosa e independiente de las características propias de cada dispositivo, modificando la apariencia o la distribución de la web.

Los beneficios del uso de esta técnica son evidentes:

- El hecho de que una aplicación web se adapte tanto a pantallas grandes como pequeñas hace que el usuario se despreocupe de tener que utilizar zoom u otras medidas que faciliten su correcta visualización.
- Dado que permite presentar el contenido en función del ancho de la pantalla del dispositivo, no se requiere una versión móvil de la web que suprima o modifique los datos mostrados, originariamente pensados para ser representados en un monitor

convencional. Esto generalmente reduce la carga del servidor al albergar una única versión de la aplicación, a la vez que simplifica el mantenimiento de la misma.

- La estética de la aplicación (distribución de los contenidos, tamaño, forma, etc.) puede variar en función de la orientación del dispositivo, lo que proporciona al usuario una mayor comodidad a la hora de navegar por la web.

RWD hace uso de tres medidas principales para conseguir que la aplicación se adapte a cualquier dispositivo: utilizar un diseño fluido, imágenes flexibles y media queries. No obstante, el primer paso consiste en hacer un buen análisis del contenido que se desea que contenga la aplicación. Dado que no se aprecia igual, por ejemplo, la pantalla de un dispositivo móvil que una televisión, puede ser útil restringir la cantidad de información presentada en cada medio, eliminando por ejemplo publicidad o aquellos apartados menos relevantes para las pantallas más pequeñas. Para determinar el diseño del contenido de manera que sea fácilmente adaptable, pueden usarse técnicas como *Mobile first*, que consiste en diseñar la aplicación inicialmente para los dispositivos móviles e ir aumentando el contenido a medida que se dispone de más espacio para presentarlo o cambian las restricciones del medio.

5.1.1. Diseño fluido

Para conseguir que la aplicación se adapte de manera natural al dispositivo en el que se visualiza es necesario utilizar medidas relativas para definir sus elementos, dejando atrás el uso de píxeles como unidad de medida.

Las nuevas medidas basadas en porcentajes se pueden obtener a partir de diferentes pruebas (alterando las dimensiones de los elementos y observando su resultado con diferentes resoluciones hasta conseguir un efecto deseado) o si se tiene un diseño previo basado en píxeles, puede utilizarse la siguiente regla:

$$target \div context = result$$

De esta forma, para adaptar un elemento de, por ejemplo, 600px de ancho, que se encuentra dentro de un contenedor de 960x960 píxeles, se establecerá para dicho elemento un valor relativo de ancho del 62.5% ($600 \div 960 = 0.625$).

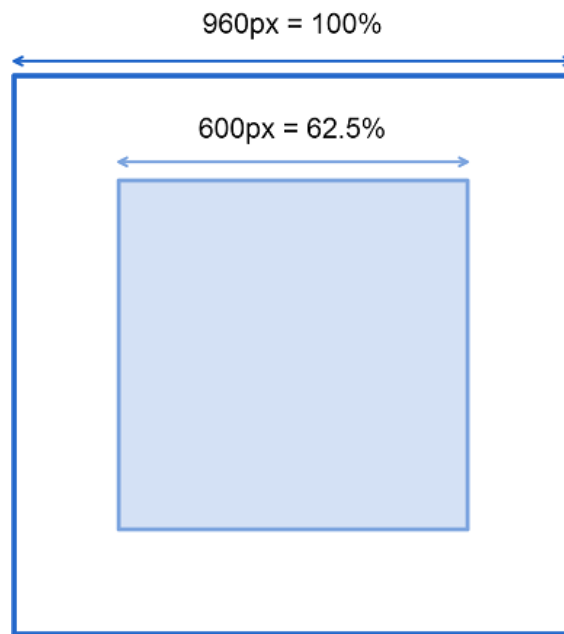


Figura 20: Regla para las medidas relativas con RWD

Al utilizar esta fórmula es necesario respetar los decimales obtenidos en el cálculo, ya que de lo contrario el resultado no será igual al conseguido de forma estática utilizando píxeles como unidad de medida.

Además de utilizar medidas porcentuales para definir las dimensiones de los objetos, también deben ser usadas para establecer márgenes, bordes o cualquier otra propiedad que no deba estar prefijada. Es recomendable no utilizar píxeles en la medida de lo posible a la hora de desarrollar la aplicación.

Para definir los tamaños de letra se puede utilizar también otra unidad de medida: el *em*. Un *em* equivale al ancho de la letra mayúscula “M” según el tamaño definido en el elemento que actúe como padre. Si no se ha fijado ningún tamaño previo, se tomará como medida el tamaño de fuente por defecto del navegador.

Para calcular el valor deseado en *ems*, se puede utilizar la misma fórmula que para los porcentajes. Por ejemplo, si se quiere utilizar un tamaño de letra de 20px en un elemento contenido en el cuerpo de la web, que se encuentra a 10px, su correspondiente medida en *ems* será: $20\text{px} \div 10\text{px} = 2\text{em}$

El *em*, al tratarse de una unidad relativa, también puede utilizarse para definir márgenes o bordes, pero hay que tener especial cuidado ya que el valor que se utilice depende directamente del padre. Por tanto utilizar una medida de, por ejemplo, 2em puede arrojar resultados muy diferentes según la estructura del documento HTML.

El diseño de la página debe ser lo más fluido posible para que la estructura pueda alterar su distribución si fuera necesario, en función del dispositivo que acceda a ella. En la siguiente imagen se muestra un ejemplo en el que las filas y columnas en que se divide el documento se modifican según el ancho de la pantalla:

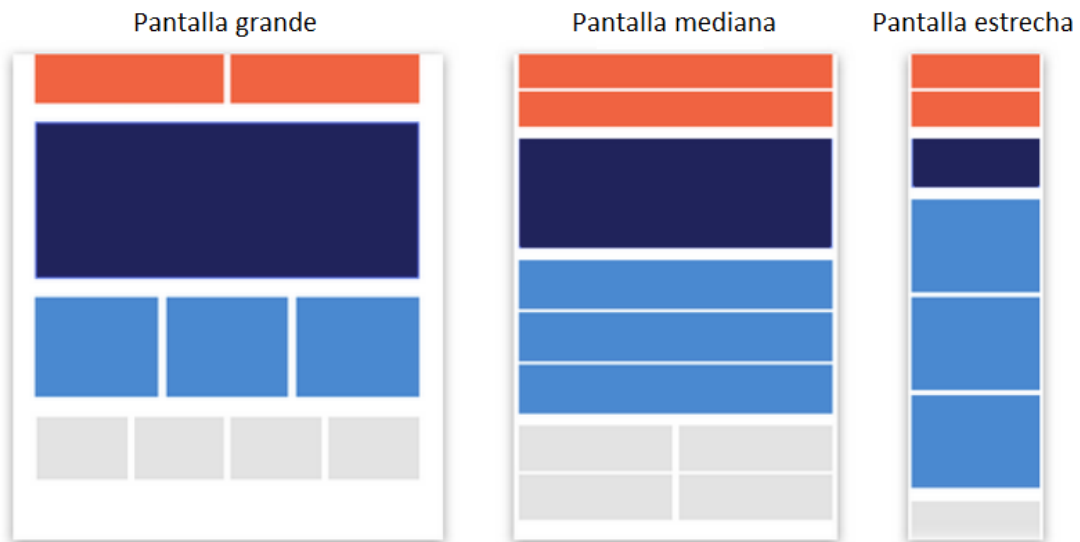


Figura 21: Diseño fluido con RWD

5.1.2. Imágenes flexibles

Las imágenes que se muestren en la web también deben ser flexibles para adaptarse al dispositivo. Si se utiliza una imagen demasiado grande, preparada para los monitores de ordenadores de sobremesa, no se verá correctamente en un dispositivo móvil. Del mismo modo que no se deben utilizar píxeles para establecer el tamaño de la estructura de la página, las imágenes deben tratarse de manera fluida, presentando un aspecto adecuado independientemente del medio.

Para que una imagen tome las proporciones deseadas se pueden usar diferentes técnicas, por ejemplo:

- Establecer los valores máximos o mínimos para sus dimensiones. Por ejemplo, si se define el atributo “max-width: 100%”, la imagen nunca sobrepasará el tamaño del contenedor, es decir “no se saldrá de la pantalla”, aunque, si por el contrario la pantalla es mayor que la imagen, ésta se mostrará a su tamaño original.
- Definir valores relativos para el ancho y/o el alto de la imagen, de forma que cuando sea demasiado pequeña para el contenedor, se escalará para rellenar el espacio asignado.

- Utilizar la imagen como fondo del elemento contenedor y asignar su tamaño mediante el atributo “background-size” de CSS3. Esta propiedad acepta un valor numérico (medido en píxeles o porcentaje) o los valores “cover” (que escala la imagen para que cubra todo el elemento) o contain (que mantiene la proporción de la imagen de forma que se muestre al máximo ancho y alto permitido sin deformarse).

5.1.3. Uso de Media Queries

Una *media query* es una característica CSS que supone un mecanismo para identificar no sólo el tipo de medio en el que se visualiza la aplicación, si no también algunas características físicas del dispositivo, como el tamaño o la resolución de pantalla.

Permiten construir una serie de reglas de estilo que sólo se ejecutarán cuando el dispositivo cumpla las características especificadas. En el siguiente ejemplo se define una media query en la que se especifica que cuando la pantalla tenga una anchura mayor a 1024 píxeles, la etiqueta h1 debe adoptar un tamaño de fuente de 2.5em y color blanco:

```
@media screen and (min-width: 1024px) {  
  h1 {  
    font-size:2.5em;  
    color:#fff;  
  }  
}
```

Es necesario establecer los puntos de ruptura en los que será necesario el uso de media queries para modificar el aspecto de la aplicación. Se pueden utilizar medidas fijas, que representen diferentes tipos de dispositivos según sus dimensiones. Las más populares actualmente son:

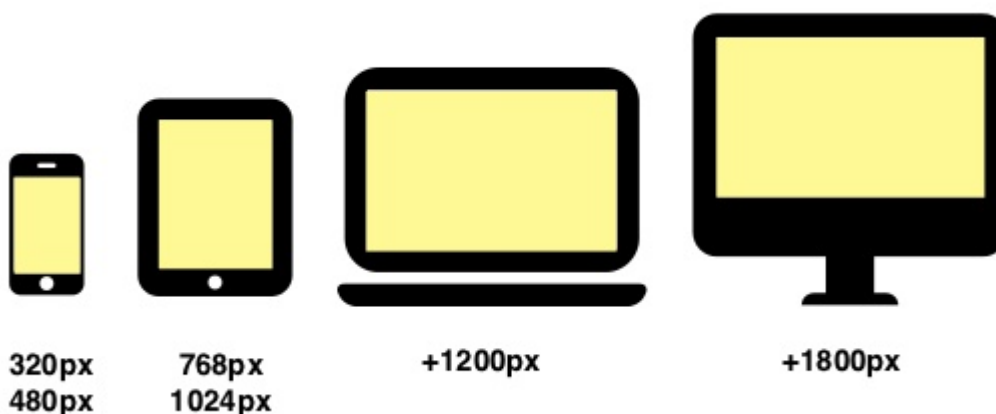


Figura 22: Media Queries convencionales

Aunque estas medidas no son demasiado fiables, ya que cada vez es más complicado hacer una clasificación de dispositivos por tipo (smartphone, tablets, ordenadores portátiles...) en los que los tamaños y resoluciones se mantengan dentro de los mismos rangos.

Lo más recomendable es definir puntos de ruptura en función del contenido de la aplicación y no del dispositivo, es decir, utilizar una media query en cada punto en el que el diseño se rompa.

A continuación se muestra una lista de propiedades que se pueden utilizar para definir los puntos de ruptura de las media queries:

Propiedad	Valor	Descripción
width	Ancho (px)	Ancho del área visible
height	Alto (px)	Altura del área visible
device-width	Ancho (px)	Ancho total del dispositivo
device-height	Alto (px)	Altura total del dispositivo
orientation	“portrait” o “landscape”	Orientación del dispositivo
aspect-ratio	Relación (w/h)	Relación ancho/alto expresado con dos enteros separados por / (por ejemplo: 16/9)
device-aspect-ratio	Relación (w/h)	Relación ancho/alto del dispositivo, expresado con dos enteros separados por /
color	Número entero	Número de bits por color
color-index	Número entero	Número de entradas en la tabla de color del dispositivo
monochrome	Número entero	Número de bits por píxel en la memoria de vídeo monocromo (blanco y negro si no, el valor es 0)
resolution	Resolución	Densidad de píxeles del dispositivo, expresada con un entero seguido de “dpi” (puntos por pulgada) o “dpcm” (puntos por centímetro)
scan	Progressive o interlace	Proceso de digitalización utilizado por los dispositivos de TV

grid	0 o 1	Si se establece en 1, el dispositivo está basado en red, tal como un terminal de teletipo o pantalla del teléfono con una sola fuente fija (el 0 representa al resto de dispositivos)
-------------	-------	---

Tabla 4: Propiedades de las media queries

Por otra parte, se puede utilizar la etiqueta *media* en el propio documento HTML para elegir la hoja de estilos, que se cargará en función de las condiciones establecida, por ejemplo:

```
<link rel="stylesheet" href="smartphone.css" media="only screen and (min-width : 320px) and (max-width : 480px)">
```

También es importante tener en cuenta el “**viewport**”, definido como meta-información del documento HTML, que permite elegir cómo se mostrará la página en el navegador. A través de esta etiqueta se pueden especificar los siguientes parámetros para procurar la representación adecuada de la aplicación:

- **Width:** ancho de la página.
- **Height:** alto de la página, actúa igual que el width.
- **Initial-scale:** escala o zoom inicial de la página.
- **Minimum-scale:** zoom mínimo permitido para la página.
- **Maximum-scale:** zoom máximo permitido para la página.
- **User-scalable:** establece si está permitido o no hacer zoom.

Para que la aplicación se adapte lo máximo posible al dispositivo es recomendable establecer que el ancho de la página sea igual al de este. Además, se suele definir un zoom mínimo de 1.0, para que la página se muestre al tamaño al que fue preparada.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=yes">
```


5.1.4. Limitaciones del Responsive Web Design

El RWD no es infalible ni aporta todas las soluciones a los problemas de diseño. Se encuentra con unos cuantos retos ante los que surgen una serie de soluciones (temporales o incompletas en algunos casos). Los principales problemas son:

- La tipografía no se escala con el *viewport*. Para que su tamaño varíe en función de las dimensiones de la pantalla se puede utilizar, por ejemplo, FitText, un plugin basado en jQuery, que reescala el tamaño de las fuentes cuando sea necesario.
- Los vídeos externos incrustados en la aplicación tampoco se ajustan adecuadamente al dispositivo. No obstante, se puede usar FitVid, también un plugin basado en jQuery que permite que los vídeos se ajusten automáticamente.
- Los navegadores antiguos (como las versiones de Internet Explorer anteriores a IE8) no tienen soporte para Media Queries. Respond.js permite utilizar las reglas de estilo definidas en las media queries y aplicarlas en aquellos navegadores que no sean capaces de interpretarlas.

6. Herramientas

A continuación se describen algunas de las herramientas utilizadas durante el desarrollo de la aplicación:

6.1. GitHub

GitHub es un servicio de alojamiento de repositorios de software para el sistema de control de versiones Git. Actualmente ofrece un conjunto de características que facilitan la gestión de proyectos, como son:

- Un wiki que funciona con “*gollum*”, el cual opera con Git para el mantenimiento de las distintas versiones de las páginas.
- Un sistema de seguimiento de problemas que permite abrir un ticket detallando una incidencia, o cualquier sugerencia acerca del software.
- Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero.
- Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas del repositorio.
- Un sistema petición de integraciones (pull requests) para desarrolladores externos al repositorio.

GitHub es totalmente gratuito para alojar código open source, siendo estos repositorios de visibilidad pública. También ofrece varias modalidades de pago para repositorios privados, abarcando desde 5 repositorios privados por 7 dólares al mes, hasta 125 por 200 dólares al mes.

6.2. ScrumDo

Se trata de una aplicación web para la gestión de proyectos ágiles basados en la metodología Scrum. La herramienta permite la creación de historias de usuario, trazabilidad de las historias, gestión del backlog, definición y valoración de iteraciones, y gráficos de tipo *burndown* entre otros.

Sus principales funcionalidades son:

- Story list: Incluye gráficos de diferentes tipos que muestran la evolución de la iteración, y listas de tareas que reflejan tanto su estado como su peso asignado.
- Scrum board: tablero en el que se muestran las historias de usuario de una iteración en diferentes columnas según su estado. Pueden moverse de una columna a otra utilizando *drag&drop* para cambiar de estado.
- Herramienta para la definición de historias de usuario, categorías y *epics*.
- Planning Poker como método de estimación y valoración de las historias de usuario.
- Planificación de las iteraciones, por medio de la cual se pueden asignar historias a las iteraciones.
- Realización de estimaciones y predicciones basadas en la fecha de terminación de las tareas y sus puntos de historia.

Ofrece cinco posibilidades de registro, una de ellas gratuita, en la que se permite un único proyecto al que tendrán acceso un máximo de tres usuarios.

La siguiente imagen muestra un ejemplo de gráfico de tipo *Stacked*, obtenido durante el desarrollo de la aplicación:

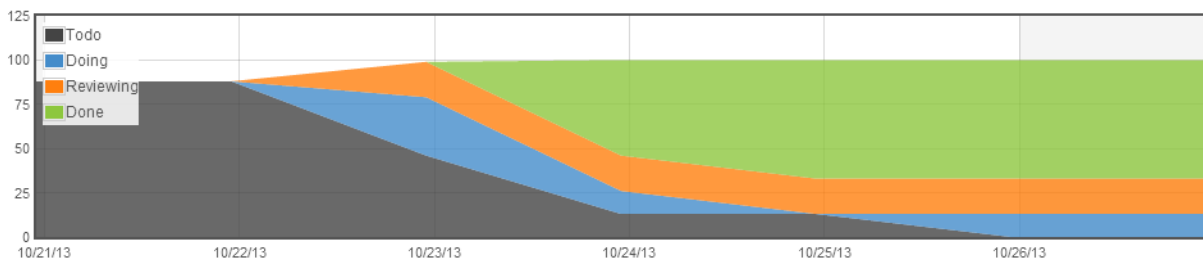


Figura 23: Gráfico de tipo stacked de ScrumDo

6.3. Eclipse IDE y Android SDK

Eclipse es un entorno de desarrollo genérico, que proporciona herramientas para la gestión de espacios de trabajo: desarrollo, despliegue, ejecución y depuración de aplicaciones. Sus principales características son:

- **Perspectivas, editores y vistas:** en Eclipse el concepto de trabajo está basado en las diferentes perspectivas, que no es otra cosa que una pre-configuración de ventanas y

editores relacionados entre sí, que facilitan la programación en un determinado entorno de trabajo.

- **Gestión de proyectos:** el desarrollo sobre Eclipse se basa en los proyectos, que son el conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros configuración, árbol de directorios, etc. Este IDE proporciona asistentes y ayudas para la creación de proyectos.
- **Depurador de código:** se incluye un depurador de uso fácil e intuitivo. Su uso está asociado a una perspectiva específica donde se muestra de forma ordenada toda la información necesaria para realizar dicha tarea.
- **Extensa colección de plugins:** existe una gran cantidad de complementos diseñados para este IDE, unos publicados por Eclipse, otros por terceros.

El SDK (Software Development Kit) de Android incluye un conjunto de herramientas de desarrollo necesarias para construir, probar y depurar aplicaciones para Android:

- Depurador de código
- Biblioteca
- Simulador de dispositivos móviles
- Documentación
- Ejemplos de código
- Tutoriales

Desde la página web oficial de desarrolladores de Android [37] se puede obtener el ADT Bundle, que incluye los componentes esenciales del SDK de Android y una versión de Eclipse con el complemento de Android (ADT – Android Developer Tools) preinstalado.

6.4. Microsoft OneNote

Microsoft OneNote es un bloc de notas digital que permite crear, capturar y almacenar todo tipo de información. Además, en su versión 2013, las notas se guardan automáticamente en la nube, con lo que es posible acceder a ellas desde cualquier dispositivo iniciando sesión en SkyDrive.

También es posible utilizar este programa en su versión web a través de las Office Web Apps de SkyDrive.

Algunas de las características que incorpora son:

- **Entrada a mano:** Se pueden crear notas a tomadas mediante dispositivos táctiles. OneNote puede almacenarlas como imágenes o convertir automáticamente la escritura a mano en texto.
- **Recortar:** Dispone de un atajo que permite seleccionar cualquier elemento presente en la pantalla y copiarlo directamente en la nota actual, o crear una nueva a partir de este recorte.
- **Incrustado de archivos:** Se pueden insertar imágenes, vídeos y documentos en las notas, así como hojas de cálculo de Excel o diagramas de Visio.
- **Autoguardado:** OneNote 2013 guarda automáticamente todo el contenido mientras se está modificando para evitar la pérdida de datos.

Durante el desarrollo del proyecto se ha utilizado esta herramienta con el fin de recoger información que fuera de utilidad para la realización de la presente memoria, así como documentar el avance progresivo de la aplicación. Los aspectos más relevantes anotados en OneNote durante el proceso han sido:

- **Documentación:** las distintas fuentes consultadas para la documentación del proyecto (libros, artículos, APIs, etc.) han sido almacenadas en una nota destinada a su correcta identificación y referencia posterior.
- **Diario:** cada día se han ido anotando los progresos obtenidos, así como las dificultades encontradas y sus soluciones.

7. Desarrollo de la aplicación

Partiendo del conocimiento relatado en el presente documento, se procede a realizar una aplicación móvil híbrida mediante PhoneGap.

Tal y como se comenta en el apartado [3. PhoneGap](#) de esta memoria, es necesario especificar al menos un sistema operativo sobre el que empezar a desarrollar la aplicación. La plataforma elegida ha sido Android, debido en primer lugar su tasa de aceptación entre los usuarios de dispositivos móviles a nivel nacional, ya que según las cifras de ventas de Kantar Worldpanel del último trimestre de 2013[18], el 86,2% de los smartphones vendidos en España utilizan Android como sistema operativo. Además, se ha escogido esta plataforma debido a la facilidad que ofrece a la hora de desarrollar aplicaciones y probarlas directamente en dispositivos reales, sin necesidad de instalarlas a través de ningún mercado de aplicaciones.

El contenido de la aplicación es obra del autor Sergi Torres, y se encuentra completo en el anexo [11.2. Proceso de 21 días. Sergi Torres](#).

7.1. Análisis

El proceso escrito por Sergi Torres consiste en un conjunto de 21 textos, organizados a lo largo de tres semanas, que deberán ser leídos día tras día de manera progresiva. Cada día se plantean varias preguntas que deben ser anotadas con el objetivo de poder repasarlas con posterioridad. Además, en algunos casos se pide que se dediquen varios momentos del día a meditar sobre la lectura asociada a dicho ejercicio, de forma que se interrumpan las actividades diarias (recomendablemente cada hora) para anotar de nuevo las impresiones que se vayan obteniendo sobre el tema tratado en el proceso.

La funcionalidad básica de la aplicación puede resumirse por tanto en tres aspectos clave:

- Visualizar los textos del proceso organizados por días.
- Anotar aspectos relacionados con cada ejercicio.
- Programar alarmas o recordatorios que ayuden a organizar el seguimiento del proceso.

Una vez reconocida la funcionalidad principal se analizan más detenidamente estos aspectos con el fin de encontrar otras posibilidades que completen o faciliten estas funciones:

Visualizar los ejercicios:

Dado que la aplicación se centra exclusivamente en el “Proceso de 21 días”, el contenido de la misma será estático. Se dispone por tanto de un número limitado y constante de ejercicios que el usuario de la aplicación irá completando a medida que avance el proceso.

Para mejorar la experiencia de usuario, es recomendable que la aplicación lleve la cuenta de los ejercicios que ya se han realizado. Además, como se deben realizar de manera progresiva, debería evitarse la posibilidad de adelantar ejercicios, ofreciendo al usuario únicamente la opción de leer el texto destinado para ese día o repasar los anteriores.

Anotar aspectos relacionados:

Según recomienda el autor del proceso, cada día se deben anotar en un cuaderno las impresiones recogidas por la lectura asignada. Ya que la aplicación pretende facilitar la realización de este proceso a través del dispositivo móvil, es preciso incorporar una manera cómoda de introducir notas conforme se realizan los ejercicios.

Dado el carácter reflexivo del proceso, es recomendable que las notas puedan consultarse e incluso modificarse conforme avance el mismo. Además deben poder asociarse al texto que dio lugar a su creación.

Programar alarmas o recordatorios

Algunos ejercicios requieren que el individuo dedique varios momentos del día a meditar sobre el texto leído. En ocasiones esto puede resultar complicado, ya que las obligaciones cotidianas hacen que se aplace el momento o incluso se olvide realizar esta tarea.

Para ayudar al usuario a reservar unos minutos a repasar las indicaciones del ejercicio se le pueden ofrecer una serie de alarmas programables, que suenen cada cierto intervalo de tiempo (el estipulado en el proceso es de una hora) y que abran directamente la libreta para minimizar la interrupción de la aplicación en las tareas que se estén realizando.

De nuevo con el fin de mejorar la experiencia de usuario, se puede ofrecer además la opción de modificar el tono de dichas alarmas, haciendo reconocible la tarea pendiente en cuanto suene el dispositivo.

Además, para que la aplicación pueda seguir resultando útil una vez terminado el proceso, debe existir la posibilidad de reiniciar el mismo, de manera que se pueda volver a comenzar desde cero sin necesidad de reinstalar la aplicación.

a. Descomposición de tareas

Una vez analizado el tipo de proyecto, así como la funcionalidad de la aplicación a desarrollar, se establecen una serie de tareas generales, conocidas en la metodología *Scrum* como *Epics*. Para facilitar este proceso se utiliza la herramienta *ScrumDo*.

Los principales *Epics* del proyecto son:

1. Estudio e instalación de las herramientas necesarias
2. Análisis y diseño inicial de la aplicación
3. Implementación de las distintas tareas
4. Integración completa
5. Pruebas

Los tres primeros puntos se descomponen en tareas más sencillas a las que se les asignan unos puntos de historia aproximados que indican su complejidad o duración:

Estudio e instalación de las herramientas necesarias	Puntos de historia
Documentación	∞
Prueba de distintas herramientas	2
Elección del espacio de trabajo	3

Tabla 5: Historias del epic 1

Análisis y diseño inicial de la aplicación	Puntos de historia
Realización de diagrama de casos de uso	8
Realización de diagrama de estados	13
Diseño de las ventanas	20

Tabla 6: Historias del epic 2

El tercer *epic*, “*Implementación de las distintas tareas*”, se divide a su vez en dos: la implementación del apartado gráfico, y la implementación de la funcionalidad.

Implementación de las distintas tareas: Implementación gráfica	Puntos de historia
Implementar la ventana principal	13
Implementar el menú	20
Implementar la introducción	8
Implementar la selección de ejercicios	40
Implementar la ventana del ejercicio	20
Implementar la interfaz de la consulta de las alarmas	13
Implementar la interfaz para crear y modificar las alarmas	20
Implementar la interfaz de la libreta	20
Implementar la ventana de las opciones	13
Implementar la ventana para cambiar el tono de las alarmas	20
Implementar la ventana de los créditos	8

Tabla 7: Historias del epic 3-1

Implementación de las distintas tareas: Implementación funcional	Puntos de historia
Enlazar alarmas desde el menú	0.5
Enlazar libreta dentro de los ejercicios	1
Guardar notas	13
Cargar notas	13
Programar crear alarma	20
Editar notas	13
Programar editar alarma	13
Borrar notas	13

Configurar opciones	5
Programar listado de alarmas	20
Enlazar alarmas dentro de los ejercicios	0.5
Programar eliminar alarma	13

Tabla 8: Historias del epic 3-2

Estas tareas se van agrupando para constituir las diferentes iteraciones del proceso de desarrollo de la aplicación. Tanto la duración como el número de tareas de cada iteración varían adaptándose a la velocidad con que se van consiguiendo los objetivos de las iteraciones anteriores, tal y como indican las metodologías de trabajo ágiles.

Por tanto, en la fase de análisis se diseña la primera iteración del proceso: “Documentación, análisis y diseño”, en la que se realizan las tareas correspondientes a los dos primeros epics descritos anteriormente:

- Realización de diagrama de casos de uso
- Realización de diagrama de estados
- Documentación
- Prueba de distintas herramientas
- Elección del espacio de trabajo
- Diseño de las ventanas

a. Realización de diagramas

Con el fin de recoger toda la funcionalidad de la aplicación y analizar los pasos necesarios para su desarrollo, se realizan dos tipos de diagramas: casos de uso y estados.

Dado que las aplicaciones híbridas con PhoneGap se desarrollan principalmente mediante HTML y Javascript (lenguaje basado en prototipos en lugar de clases), carece de sentido la realización de otro tipo de diagramas como los de clases o estados.

A continuación se analizan los resultados obtenidos en esta fase de diseño.

i. Diagrama de casos de uso

Este tipo de diagrama ofrece una descripción del comportamiento del sistema especificando los escenarios en los que la aplicación interactúa con personas, organizaciones o sistemas externos, los objetivos a conseguir y el ámbito del sistema.

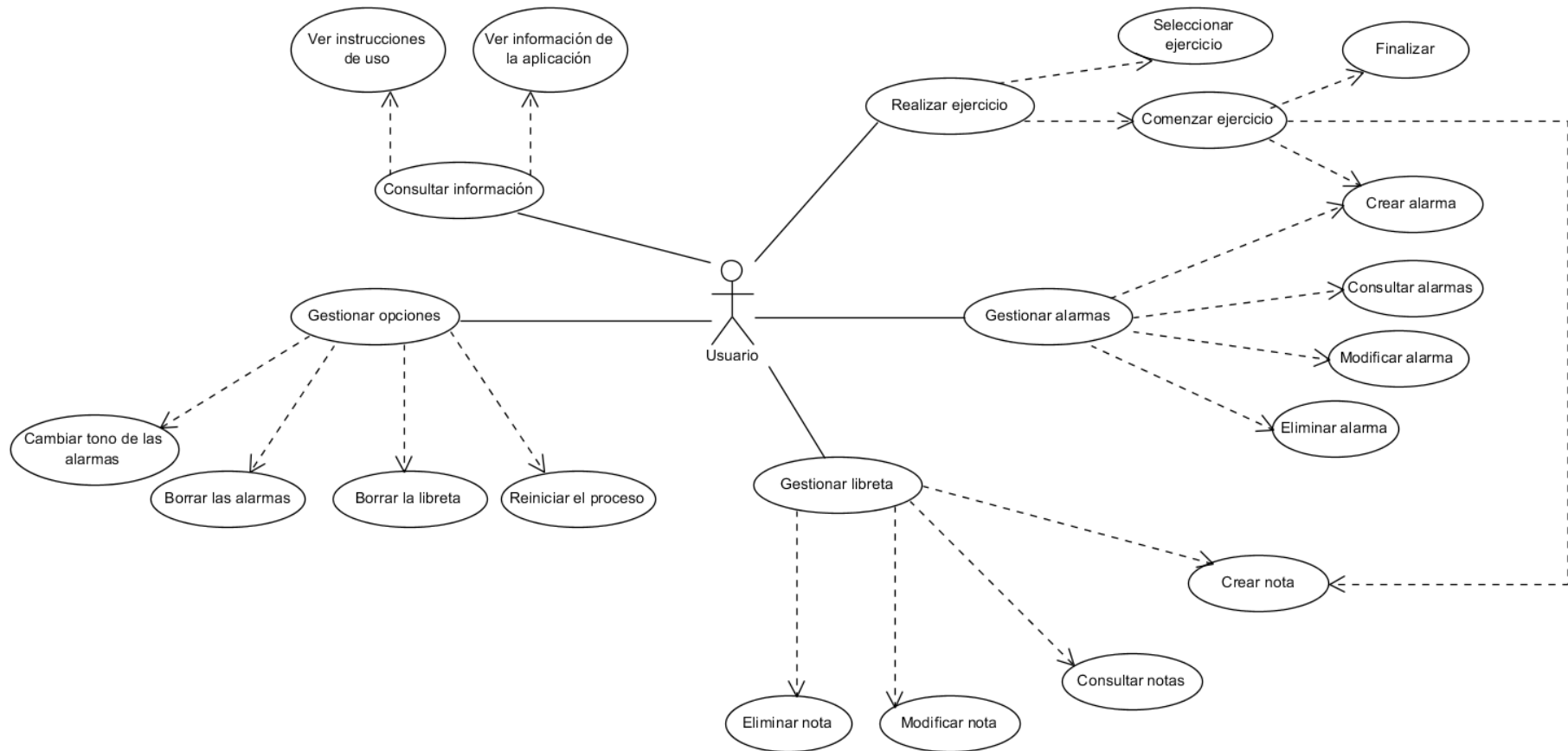


Figura 24: Diagrama de casos de uso

En esta aplicación sólo se dispone de un tipo de usuario, que ejecutará las tareas enunciadas en los apartados anteriores. En el diagrama anterior se desglosan estas tareas, relacionándolas entre sí.

ii. Diagrama de estados

Este tipo de diagramas muestra el conjunto de estados de la aplicación, que ocurren en respuesta a ciertos eventos, junto con sus consecuencias y acciones:

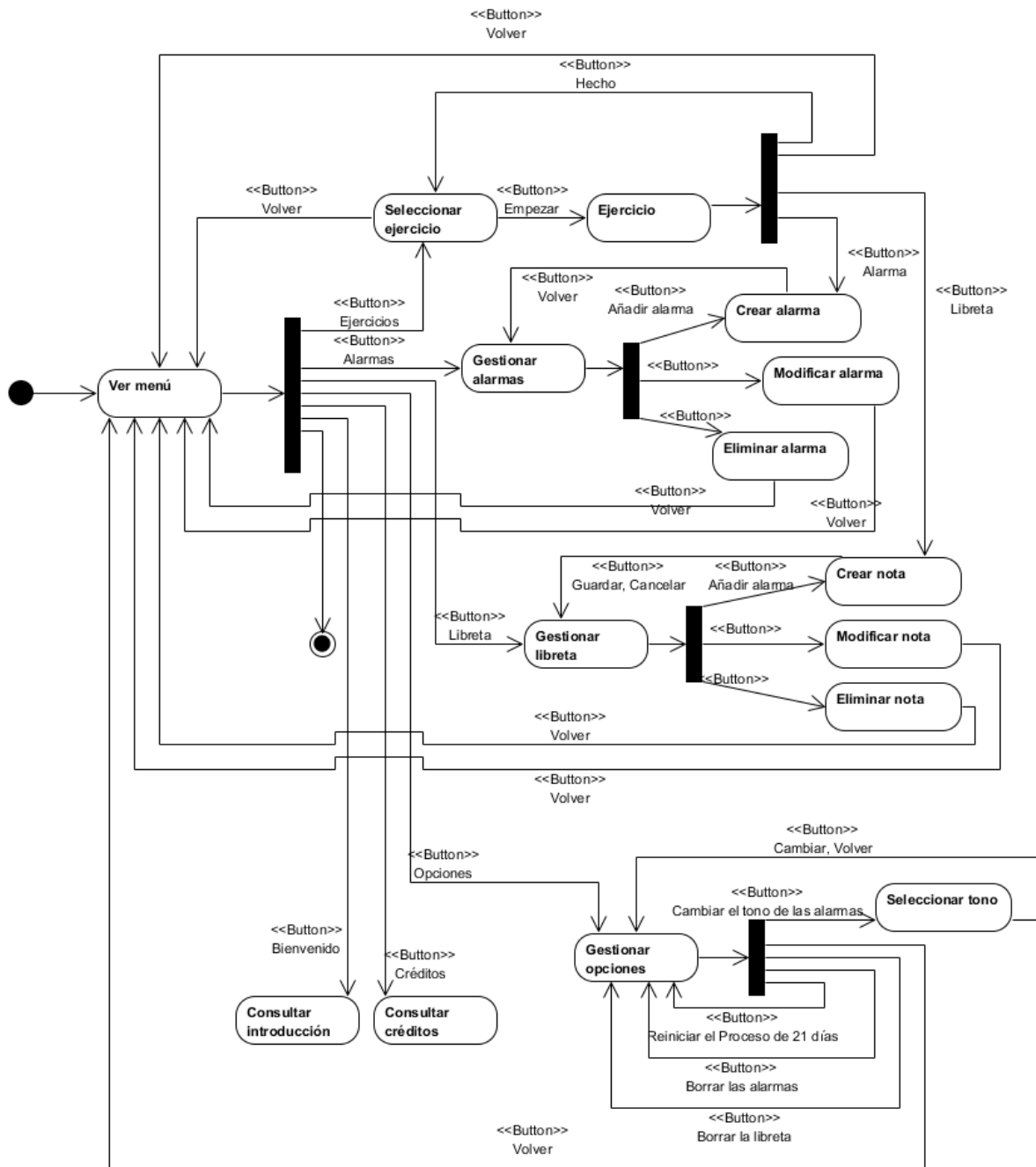


Figura 25: Diagrama de estados

Este diagrama ayuda además a reconocer los botones y ventanas que harán falta para implementar la interacción con el usuario. Para comprender mejor el diagrama global de la Figura 23 se desglosará en diferentes partes.

En primer lugar la aplicación deberá disponer de un menú, que de acceso a las distintas opciones que se ofrecen al usuario. Estas son: consultar información (tanto la introducción del

Proceso de 21 días como los títulos de crédito), dar paso a los ejercicios, y gestionar las alarmas, libreta y notas

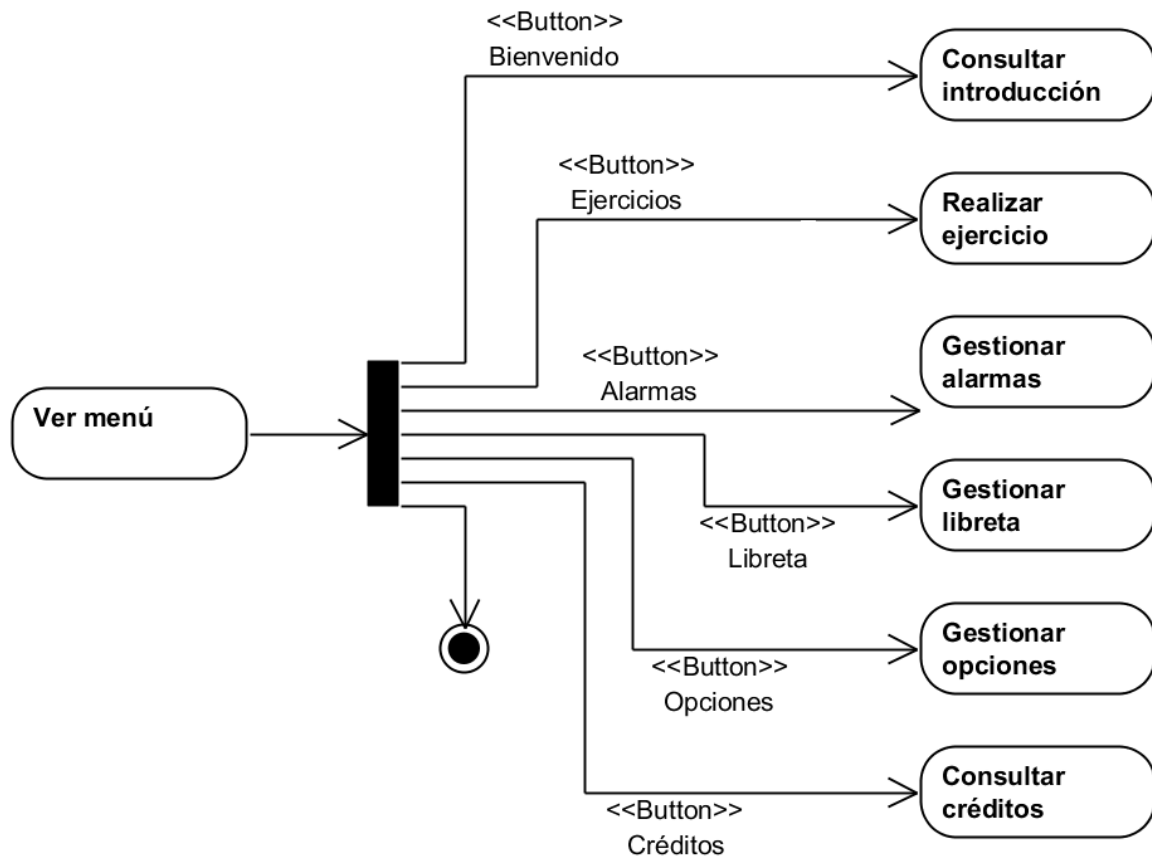


Figura 26: Diagrama de estados del menú

El aspecto central de la aplicación es el apartado de ejercicios. Se debe poder seleccionar uno de los 21 ejercicios del Proceso (el primer ejercicio de la lista de no realizados, o alguno de los que se han completado con anterioridad). Una vez dentro de la ventana en la que se muestra el texto del ejercicio se tiene que tener la posibilidad de crear una nueva nota en la libreta, un recordatorio en el listado de alarmas, o volver a la ventana anterior (habiendo completado o no el ejercicio asignado).

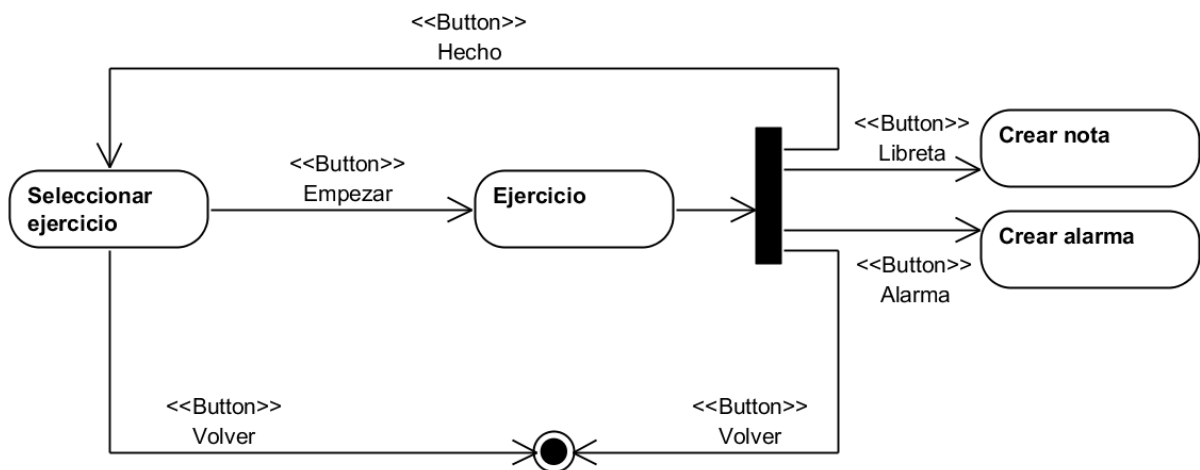


Figura 27: Diagrama de estados de los ejercicios

Tanto el apartado de las alarmas como el de las notas son muy parecidos entre sí. Ambos deben ofrecer la posibilidad de gestionar su contenido, creando nuevo material, y modificando o eliminando el existente.

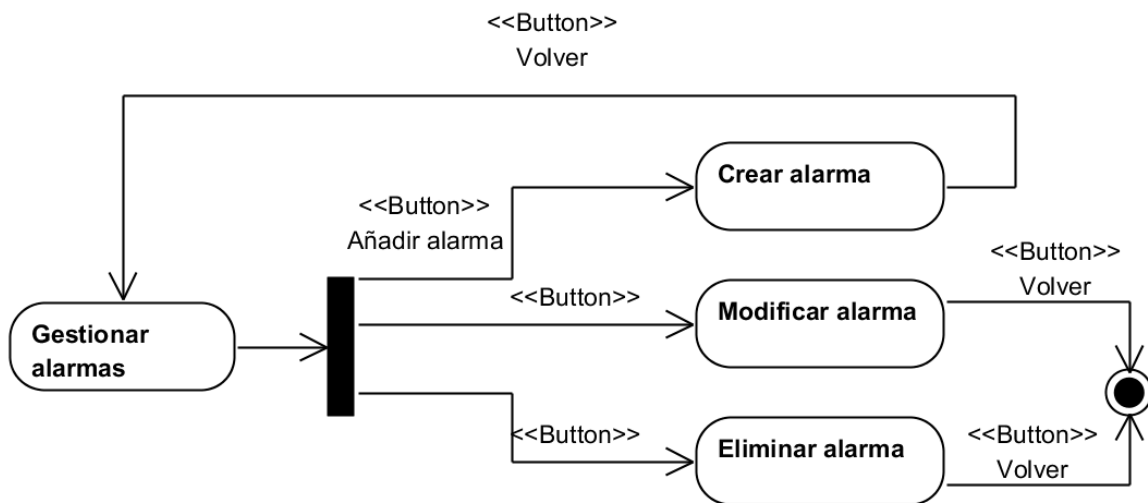


Figura 28: Diagrama de estados de las alarmas

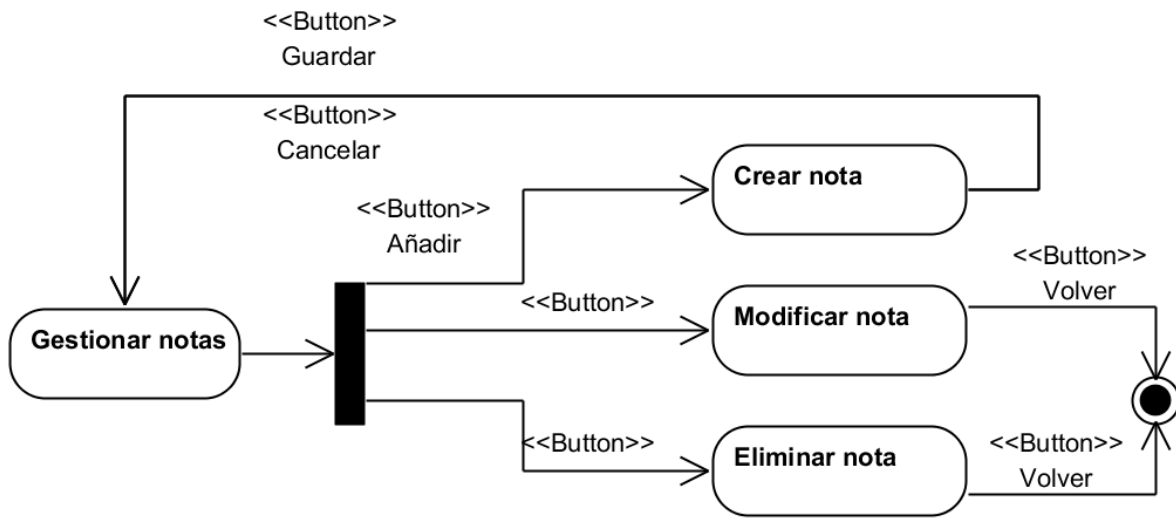


Figura 29: Diagrama de estados de la libreta

Por último, el apartado de **Opciones** incluye la opción de reiniciar el proceso eliminando las notas, las alarmas y la lista de ejercicios realizados. Además, se podrán borrar el contenido de la libreta y las alarmas por separado, así como modificar el sonido con el que se notifican dichos recordatorios.

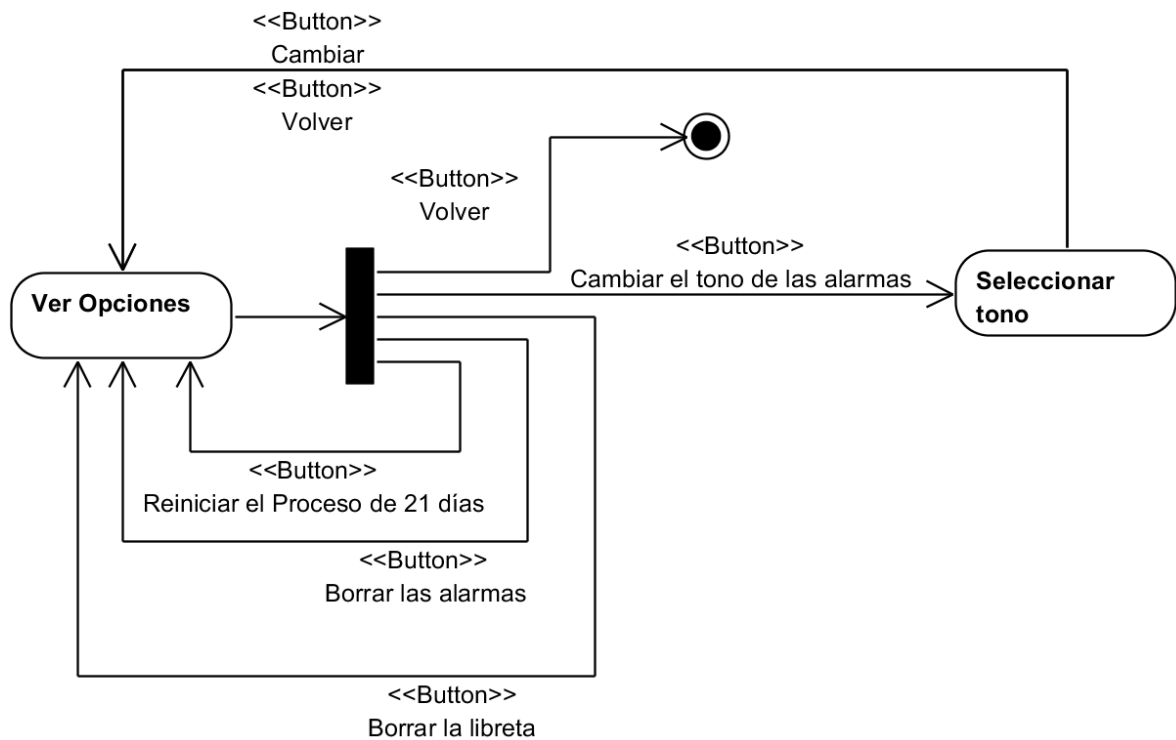


Figura 30: Diagrama de estados de las opciones

7.2. Diseño

Partiendo de la información obtenida en las fases previas, se crea la interfaz de la aplicación, ofreciendo al usuario la posibilidad de interactuar con la totalidad de las funcionalidades de la aplicación de la forma más cómoda e intuitiva posible.

Para elaborar este diseño se han elegido imágenes, sonidos y tipografías gratuitas y de libre acceso, que no comprometan la legalidad de la aplicación, obtenidas principalmente de fuentes como Fontsquirrel[38], Oringz[41] o Freepik[39]. Así mismo, se han creado estilos propios para mantener uniforme la estética del proyecto.

A continuación se muestran las diferentes ventanas de las que consta la aplicación:

a. Diseño de la interfaz

Al iniciar la aplicación se muestra una *Splash screen* o ventana principal que aúna la estética de la aplicación. En ella se observa tanto el título de la misma como el nombre del autor del contenido, Sergi Torres.

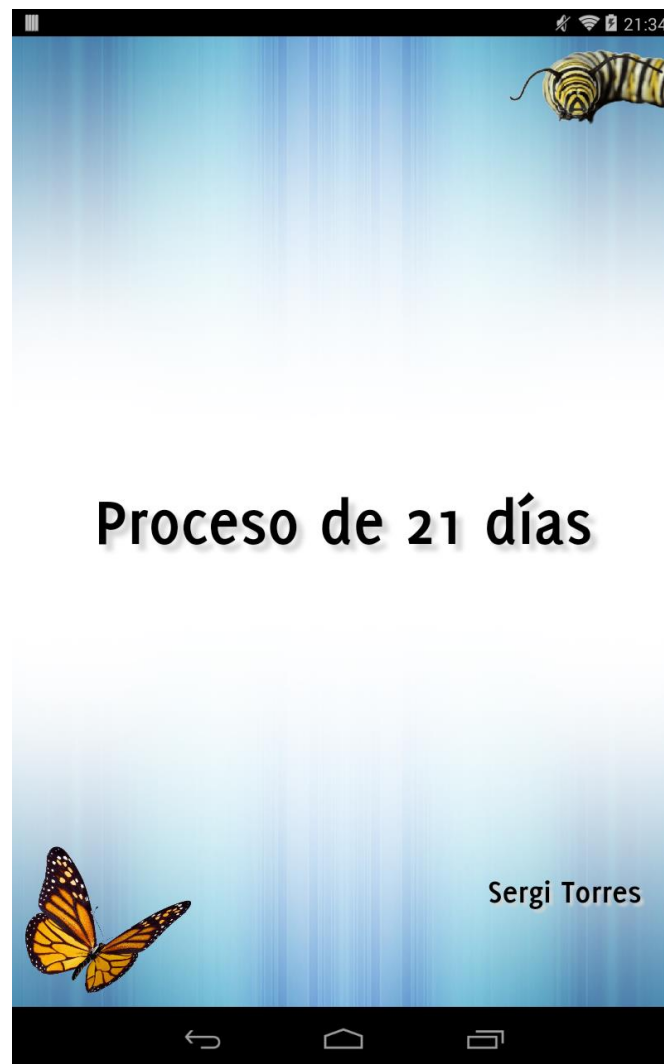


Figura 31: Interfaz de la ventana de inicio

Transcurridos 5 segundos, o bien una vez que el usuario haya pulsado sobre cualquier lugar de la pantalla, se da paso al menú, que contiene las funciones principales de la aplicación:



Figura 32: Interfaz del menú

La primera opción del menú muestra la *Introducción*. En este apartado se pueden leer las recomendaciones de Sergi Torres sobre cómo realizar el proceso y cómo se distribuyen los ejercicios:

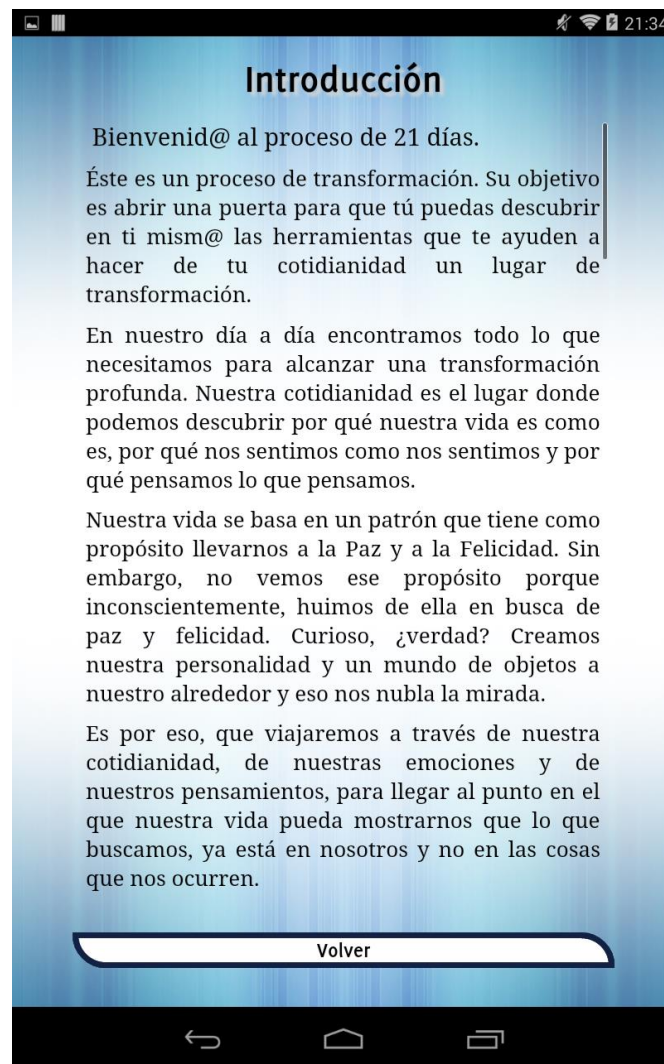


Figura 33: Interfaz de la ventana de introducción

Los 21 ejercicios del proceso suponen el eje fundamental de la aplicación. La primera ventana que se proporciona para este fin permite visualizar el título de cada uno de ellos así como el día de la semana con el que está relacionado. Sin embargo, sólo permitirá acceder al primer ejercicio no realizado de la lista. Los ejercicios que ya se hayan completado se marcarán en verde para facilitar su identificación.

Por defecto, al entrar en esta ventana se mostrará seleccionado el ejercicio que se deba realizar para continuar el proceso.

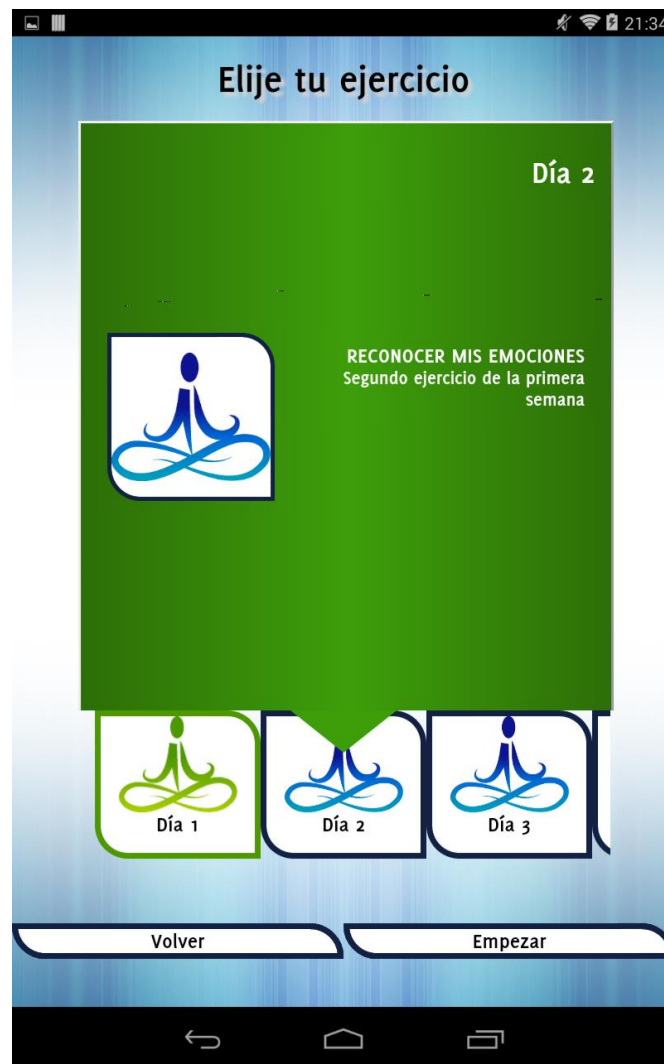


Figura 34: Interfaz de la selección de ejercicios

Una vez que se ha seleccionado un ejercicio se podrá leer el texto completo de dicho día acompañado de cuatro opciones: *Volver*, *Alarma*, *Libreta* y *Hecho*. Los botones de *Alarma* y *Libreta* suponen un acceso directo a dichas funcionalidades respectivamente. Al pulsar sobre el botón de *Hecho*, el ejercicio quedará marcado en verde en la pantalla anterior, de manera que se actualice el nuevo ejercicio pendiente.

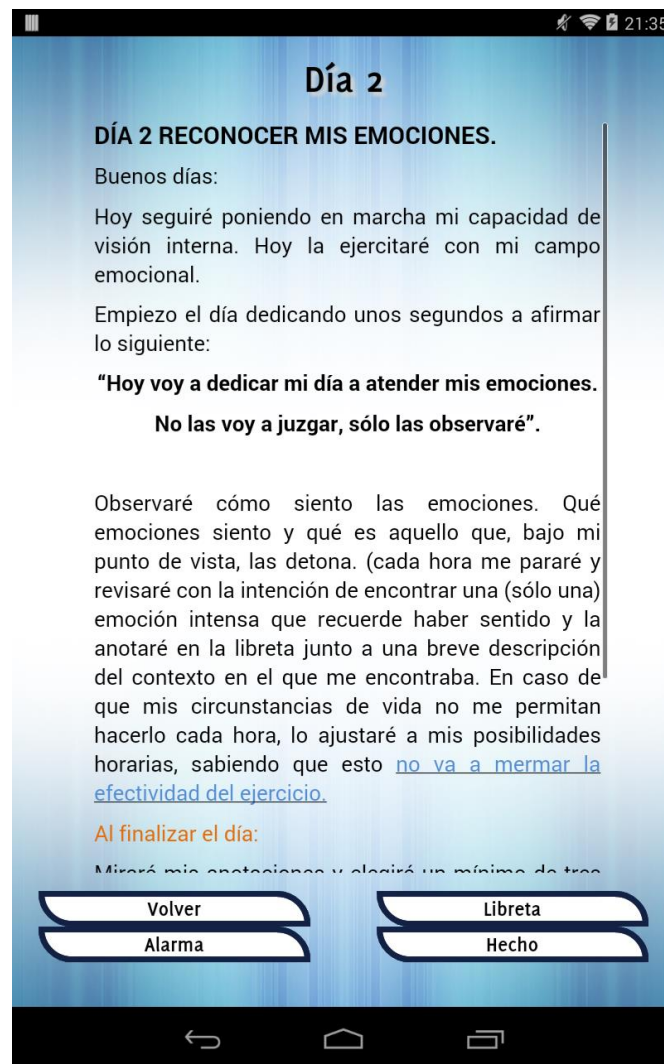


Figura 35: Interfaz de un ejercicio

En la ventana de gestión de alarmas se pueden consultar las alarmas que han sido creadas, además de modificarlas o eliminarlas. También se puede crear una nueva alarma desde esta misma pantalla.



Figura 36: Interfaz de la gestión de alarmas

Las alarmas sólo podrán programarse para el día presente, habiendo dos opciones: establecer una única alarma, o hacer que se repita automáticamente cada hora durante el resto del día, tal y como aconsejan los ejercicios que requieren esta práctica. De esta forma, en el ejercicio del segundo día, por ejemplo, el autor recomienda al lector detenerse cada hora para anotar en la libreta al menos una emoción intensa que haya experimentado en ese plazo de tiempo. Así pues, tras programar las alarmas, recibirá una notificación en su dispositivo que le llevará directamente a la libreta para facilitar la realización del ejercicio.

Nueva alarma

Título:
Día 2

Mensaje:

Hora:
12:27

Repetir cada hora

Volver Guardar

Figura 37: Crear nueva alarma

El apartado de la libreta es similar al de las alarmas. Desde esta ventana se pueden consultar todas las notas almacenadas durante el proceso, además de eliminar o modificar cualquiera de ellas. Al crear una nueva nota se sugerirá automáticamente como título de esta el día del ejercicio pendiente.

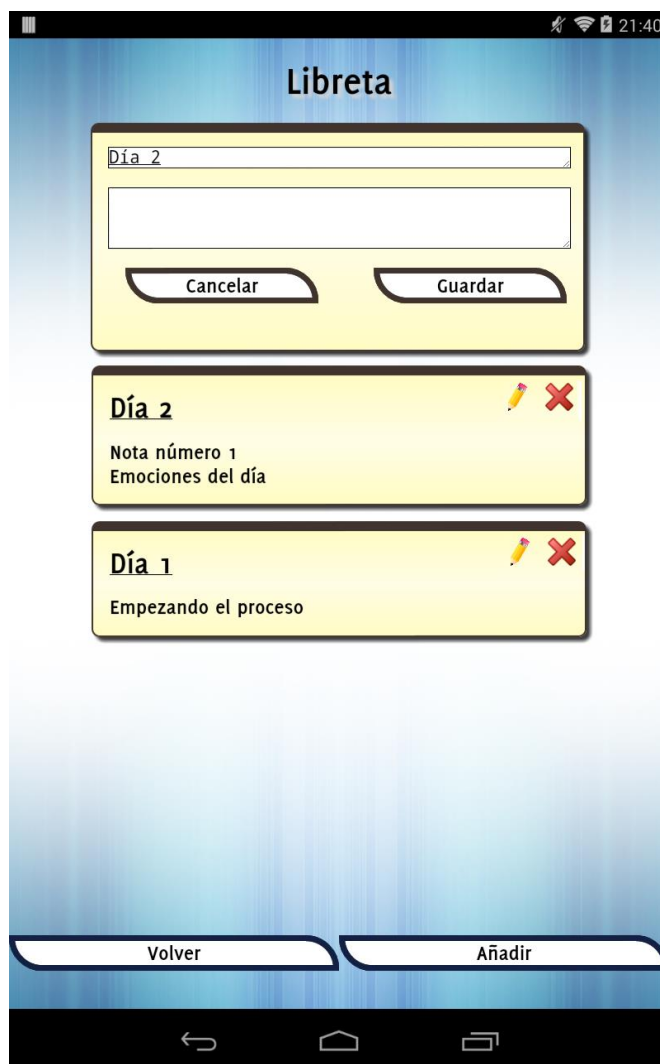


Figura 38: Interfaz de la gestión de la libreta

El apartado de opciones reúne la funcionalidad comentada en los apartados anteriores, ofreciendo la posibilidad de eliminar la información que se ha ido almacenando asociada al proceso en parte o en su totalidad. Además se ofrecen algunas melodías predefinidas con las que personalizar el tono de los recordatorios.

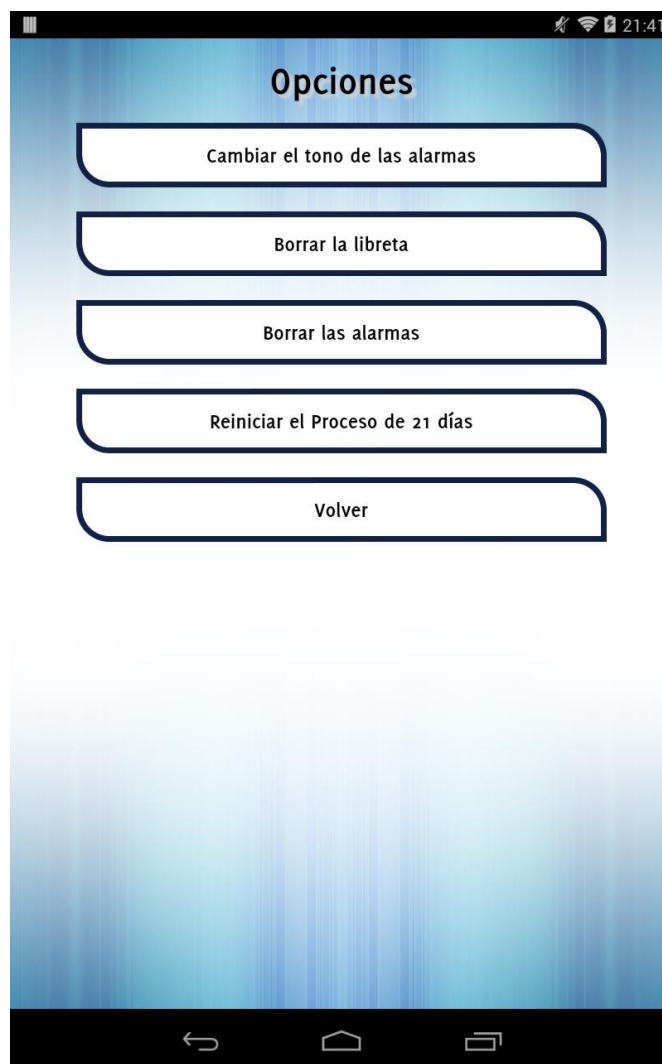


Figura 39: Interfaz de las opciones

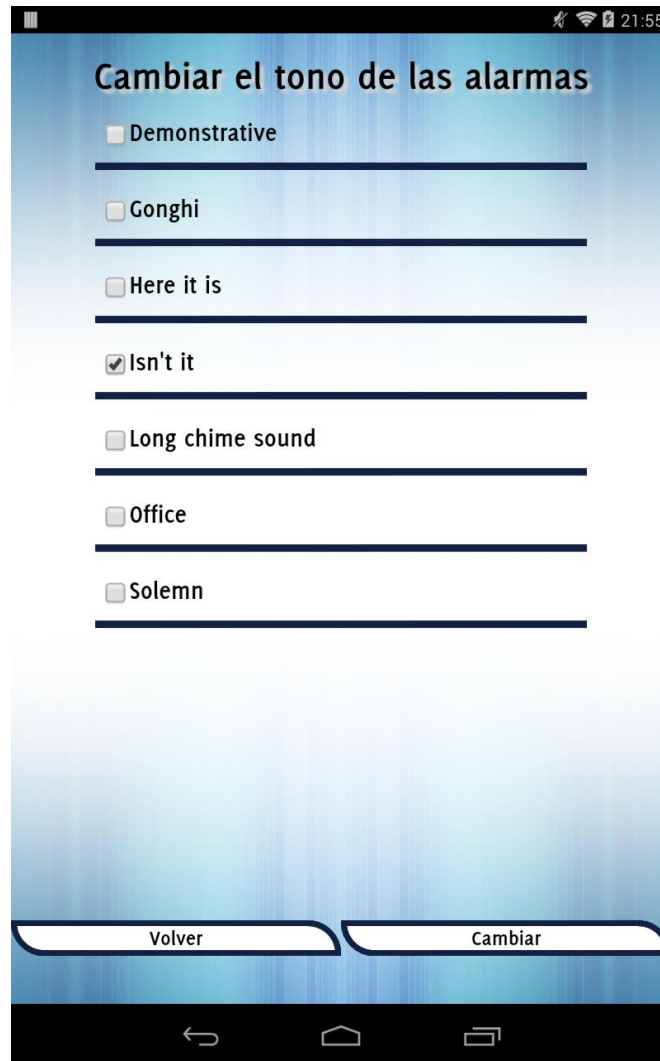


Figura 40: Interfaz para cambiar el tono de alarma

Por último, la aplicación cuenta con un apartado de *Créditos* que se puede observar en la siguiente imagen:

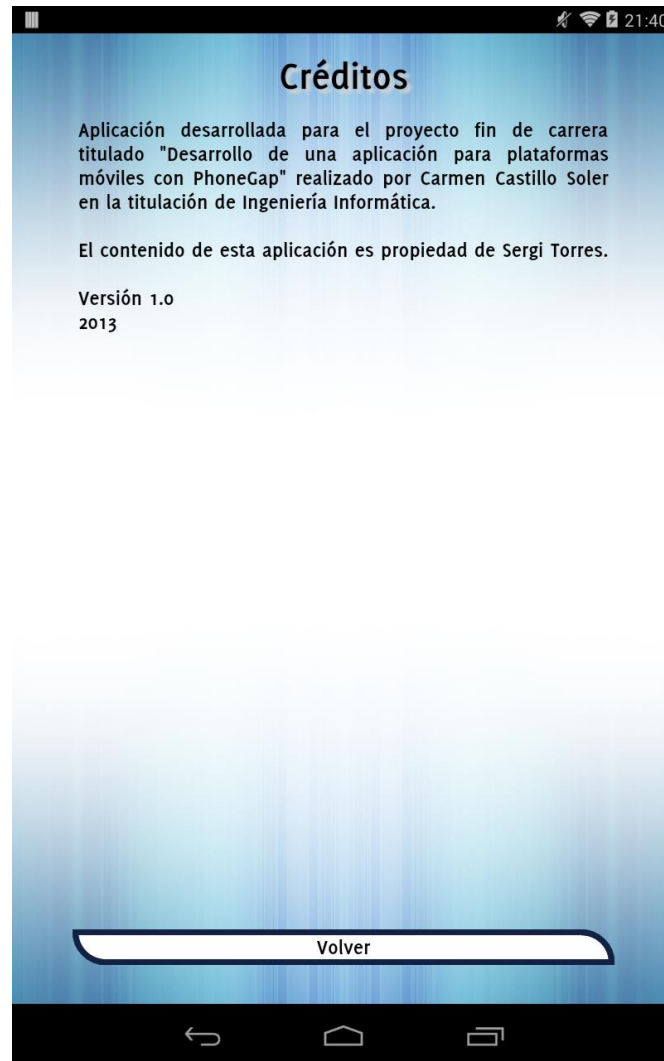


Figura 41: Ventana de créditos

b. Adaptación a los distintos dispositivos con RWD

La interfaz de esta aplicación ha sido diseñada siguiendo los principios del Responsive Web Design, de manera que se adapte correctamente al tamaño del dispositivo en el que se ejecute.

Uno de los principales aspectos tenidos en cuenta a la hora de realizar un diseño adaptativo es la orientación del dispositivo. Algunas de las ventanas de la aplicación cambian la distribución de sus componentes dependiendo de la posición en la que se sostenga el dispositivo móvil.

El primer ejemplo de ello es el de la ventana del menú principal, en la que se muestran los botones agrupados en tres filas y dos columnas en caso de que el dispositivo tenga una orientación vertical, o tres columnas y dos filas en caso contrario:



Figura 42: Menú en posición vertical

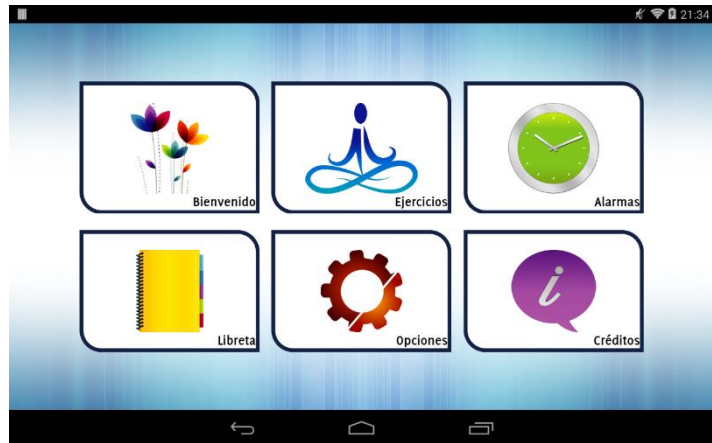


Figura 43: Menú en posición horizontal

Para conseguir este efecto se construyen dos estructuras HTML que contemplen ambas opciones respectivamente. Una de estas estructuras se ocultará en función de la orientación, gracias a la siguiente media query:

```
/*Pantallas en vertical*/  
@media only screen and (orientation:portrait) {  
    #menuHorizontal{  
        display: none;  
    }  
}
```

Los botones de la interfaz del ejercicio también alteran su disposición, de manera que cuando el dispositivo cuenta con una orientación vertical se muestran en dos columnas, mientras que en el caso contrario se establecen en una única fila.

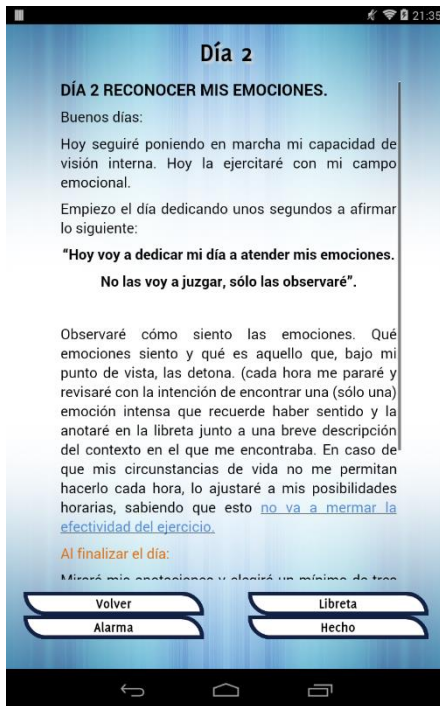


Figura 44: Interfaz del ejercicio en vertical

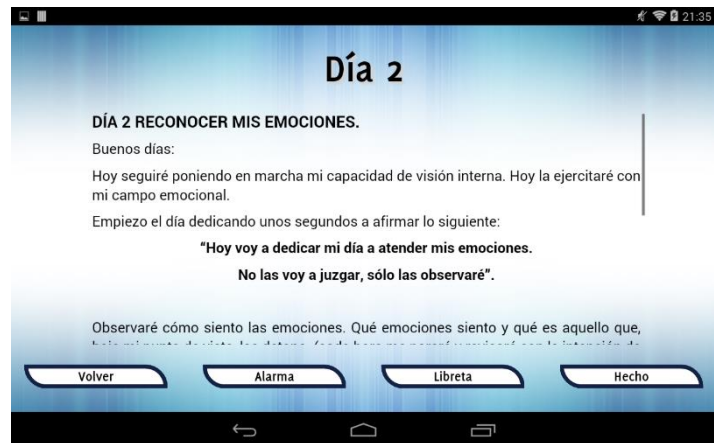


Figura 45: Interfaz del ejercicio en horizontal

Para ello se utiliza la propiedad “column-count” de CSS3, con la que se establecen por defecto 2 columnas para el contenedor de los botones, y se amplía a 4 en caso de que el dispositivo esté en horizontal:

```
/*Pantallas en horizontal*/
@media only screen and (orientation:landscape) {
    #bontonesEjercicio{
        -moz-column-count: 4;
        -webkit-column-count: 4;
        column-count: 4;
    }
}
```

Además de la orientación del dispositivo, también se tiene en cuenta su tamaño y su resolución, utilizando propiedades como min-width, max-width, o -webkit-min-device-pixel-ratio.

Los puntos de ruptura se han determinado en función del contenido, a través de múltiples pruebas en diferentes dispositivos.

7.3. Implementación

La implementación del proyecto se ha llevado a cabo de manera conjunta con el diseño de la interfaz, ya que la funcionalidad de esta aplicación es tan importante como su usabilidad.

Como se comentaba en apartados anteriores del presente documento, se ha utilizado fundamentalmente HTML y JavaScript para crear la estructura y la funcionalidad de la aplicación, y CSS para la parte estética.

A continuación se mencionan algunas peculiaridades de la implementación de esta aplicación:

Carga del contenido

Tanto la introducción como los 21 ejercicios del proceso han sido elaborados por el autor Sergi Torres. Para incorporar estos textos al proyecto se ha utilizado una llamada Ajax de tipo GET que realiza una petición síncrona de los datos y los muestra en un contenedor del DOM de la página en cuestión:

```
function cargarIntro(){
    $.ajax({
        type:"GET",
        dataType: "html",
        async: false,
        cache: true,
        url: "Intro.html",
        success: function(data) {
            $('#contenido').append(data)
        }
    });
}
```

Desplazamiento horizontal

La ventana de selección de ejercicios permite el desplazamiento horizontal para visualizar rápidamente el nombre y descripción de cada uno de los ejercicios del proceso. Este movimiento se ha llevado a cabo gracias a la librería de JavaScript: iScroll, en su cuarta versión[26].

Para comenzar a utilizar la librería basta con enlazarla en la página HTML:

```
<script type="text/javascript" src="js/iscroll4.js"></script>
```

La API de la librería recomienda utilizar una estructura sencilla en el DOM del archivo HTML, eliminando etiquetas innecesarias y elementos anidados. El scroll se aplicará únicamente al primer hijo del elemento contenedor. La estructura usada en la aplicación es la siguiente:

```
<div id="wrapper-horizontal">
  <div id="scroller_horizontal">
    <ul>
      <li>
        <div id="Dia1" class="boton">...</div>
      </li>
      ...
    </ul>
  </div>
</div>
```

Para inicializar el desplazamiento hay que esperar a que el DOM se haya cargado por completo, con lo que se ha utilizado el evento `onDOMContentLoaded`, asignándole la siguiente función:

```
function loaded() {
  myScroll = new iScroll('wrapper-horizontal', {
    hScrollbar: false,
    snap: '.boton',
    onScrollEnd: function(){
      cargarEjercicio();
    }
  });
}
```

En esta función se especifican los siguientes parámetros:

- **hScrollbar:** `false` evita que se muestre una barra de desplazamiento horizontal sobre el elemento
- **snap:** `'.boton'` establece que el desplazamiento se haga entre los botones que representan a cada ejercicio considerándolos como unidades indivisibles (evitando que el scroll quede entre dos ejercicios consecutivos).
- La función asignada al evento **onScrollEnd** muestra en pantalla la información del ejercicio que está siendo seleccionado cuando termina el desplazamiento

Al cambiar la orientación del dispositivo, la disposición y los elementos de la ventana se modifican para adaptarse a la pantalla. Para evitar que JavaScript utilice referencias desactualizadas de los elementos del DOM a la hora de realizar el scroll se utiliza el método **refresh** cuando se producen estos cambios.

Al abrir la ventana de selección de ejercicios se realiza un scroll automático para centrar en pantalla el ejercicio que debe realizar el usuario. Esto se hace utilizando la función **scrollToElement** de la siguiente manera:

```
myScroll.scrollToElement('li:nth-child('+ejercicio+')',200);
```

El primer parámetro es un selector de CSS3 que indica el ejercicio al que debe desplazarse. El segundo corresponde al tiempo destinado para dicha operación.

Para ofrecer información de cada ejercicio en la ventana de selección se utiliza un archivo xml que contiene tres campos ("Día", "Descripción" y "Semana") para cada uno de ellos. A este archivo se accede de la siguiente manera:

```
xmlDocIndice=abrirFichero("indiceEjercicios.xml");
x=xmlDocIndice.getElementsByTagName("descripcion")[posicion-1];
descripcion=x.childNodes[0].nodeValue;
x=xmlDocIndice.getElementsByTagName("semana")[posicion-1];
semana=x.childNodes[0].nodeValue;
```

Donde **posicion** es un número entero entre 0 y 20 que representa al ejercicio.

Al salir de la ventana de selección, se destruye el elemento myScroll para ahorrar espacio en memoria con el método **destroy**.

Ejercicios realizados

Cada vez que se completa un ejercicio, éste es anotado en el LocalStorage de la aplicación, con el fin de conocer cuál es el ejercicio que debe ser sugerido al usuario. Al abrir la ventana de selección se calcula la posición del último ejercicio realizado (mediante el método **scrollToElement** anteriormente descrito), y se cambian las propiedades de estilo de los anteriores a él para identificarlos mejor.

```
function colorearCompletados() {
    posicion = parseInt(localStorage["hecho"]);
    for ( var i = 0; i < posicion-1; i++) {
        id = "Dia"+(i+1);
        elem = document.getElementById(id);
        e=elem.children[0];
        e.style.backgroundImage="url(img/ejercicio4.png)";
        elem.style.borderColor="#4F9900";
    }
}
```

Gestión de notas

Las notas se almacenan en un array que es convertido mediante el método **JSON.stringify()** para guardarlo en el almacenamiento interno de la aplicación haciendo uso del LocalStorage.

Para cargar en pantalla las notas existentes se utiliza el siguiente método:

```
function cargarNotas() {
    var storedNotes = localStorage.getItem("notes");
    if (typeof storedNotes !== "undefined") {
        if (storedNotes !== null && storedNotes.length > 0) {
            var notesArray = JSON.parse(storedNotes);
            count = notesArray.length;
            for (var i = 0; i < count; i++) {
                var nota = notesArray[i];
                anadirALibreta(i, nota.Title, nota.Content);
            }
        }
    }
}
```

Se obtienen las notas guardadas en el almacenamiento local, y por cada una de ellas se muestra en la ventana de la libreta su título y su contenido a través de la función **anadirALibreta**.

Las notas se almacenan según la fecha de creación, mostrando como última la primera nota creada. De esta forma, aunque una nota sea editada no se modifica su posición dentro de la lista de notas.

Para añadir una nueva nota se crea dinámicamente un elemento en el DOM de la página HTML. Antes de guardarla se verifica que tanto el título como la descripción no estén vacíos.



Figura 46: Añadir una nueva nota

Para modificar o eliminar las notas, cada una de ellas dispone de dos botones en la esquina superior derecha. Al presionar sobre el de editar se crea un cuadro similar al de introducir una nueva nota, relleno con los datos de aquella que se desea modificar (su anterior título y descripción).

El método utilizado para editar una nota se describe a continuación:

```
function editarNota(event) {  
  /** Evita que se editen varias notas a la vez **/  
  if($("#nNota").length==0){  
  
  /** Rellena el cuadro de edición con los datos anteriores de la nota **/  
    elem = event.currentTarget.id;  
    nota = document.getElementsByClassName("nota")[elem];  
    titulo = $(nota).children()[1];  
    contenido = $(nota).children()[2];  
    anadirNota();  
    nuevaTitulo = document.getElementsByClassName("nota-titulo")[0];  
    nuevaTitulo.innerHTML = titulo.innerHTML;  
    nuevaContenido = document.getElementsByClassName("nota-contenido")[0];  
    nuevaContenido.innerHTML = contenido.innerHTML;  
  }  
}
```

```
/** Oculta de la lista de notas aquella que se está editando */
    elem2 = (this).id;
    elementoLista = document.getElementById("obj"+elem2);
    $(nota).hide();

/** Si se guarda la edición, se busca la nota original y se modifica su
contenido */
    btnGuardar = document.getElementsByClassName("botonDialogo")[1];
    btnGuardar.onclick = function() {
        t=$("#textarea.nota-titulo").val();
        c=$("#textarea.nota-contenido").val();
        $(nota).find( ".nota-titulo" ).text(t);
        $(nota).find( ".nota-contenido" ).text(c);
        cerrarNuevaNota();
        $(nota).show();
        guardarNotas();
    }
    btnCancelar = document.getElementsByClassName("botonDialogo")[0];
    btnCancelar.onclick = function() {
        cerrarNuevaNota();
        $(nota).show();
    }
}
}
```

Gestión de alarmas

Las alarmas, al igual que las notas, son almacenadas en el LocalStorage del dispositivo. Cada alarma programada se representa como un objeto json con la siguiente estructura:

```
{
  "date": "fecha",
  "ticker": "tituloNotificacion",
  "message": "mensajeNotificacion",
  "repeatDaily": false,
  "id": "identificador",
  "sonido": "sonido"
}
```

El atributo "ticker" constituye el título que muestra el sistema operativo en la notificación:

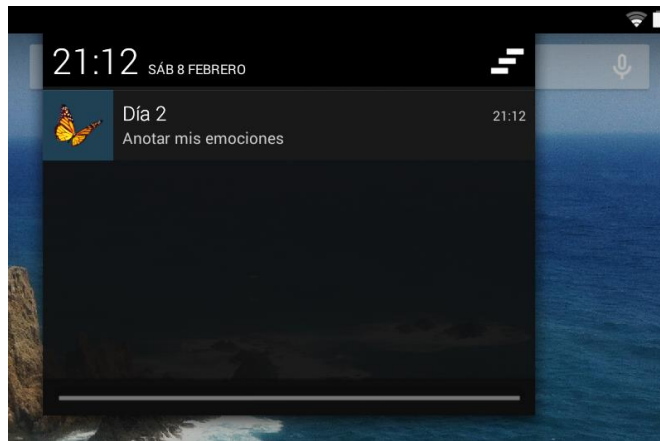


Figura 47: Notificación de la alarma

“repeatDaily” es un booleano que indica si la alarma se repetirá diariamente. Dado que cada ejercicio tiene unas características diferentes y no siempre es necesario establecer alarmas, este parámetro estará siempre asignado a “false”.

Estos objetos se ordenan por la hora en la que deben realizarse las notificaciones y se convierten mediante el método JSON.parse para guardarlos en el almacenamiento interno. De esta manera, aunque se edite una alarma y se modifique su hora, se mantendrá un orden temporal, viéndose primero las alarmas más próximas. Para mostrar las alarmas se recogen estos datos y se crea dinámicamente la estructura donde volcar la información en la interfaz.

Para que se ejecuten estas notificaciones en el dispositivo aún sin estar utilizando la aplicación, se ha utilizado el plugin para PhoneGap “LocalNotification”, añadiendo la librería JavaScript en la ventana de las alarmas:

```
<script type="text/javascript" src="js/LocalNotification.js"></script>
```

Para añadir una nueva alarma se hace una llamada al método add del plugin:

```
plugins.localNotification.add({  
    date : fecha,  
    ticker : tituloAlarma,  
    message : mensajeAlarma,  
    repeatDaily : false,  
    id : identificador,  
    sonido : sonido  
});
```

Con esto se envía la información a las clases Java con las que conecta la librería LocalNotification.js, que se encargan de crear una notificación en lenguaje nativo.

El plugin ha sido modificado para añadir la opción de definir el sonido de la alarma a través de la aplicación, sin tener que acceder al código nativo. Para ello se ha redefinido la llamada en la interfaz JavaScript del plugin, así como las clases Java que crean la notificación en el sistema operativo. De esta forma, cuando se le pasa el identificador del sonido al método “add” comentado, la clase final Android encargada de añadir la notificación al sistema calcula la ruta en la que se encuentra el archivo de audio y sustituye el tono por defecto de las notificaciones del sistema para esta aplicación.

Además se ha modificado de manera nativa en el plugin el icono de la notificación, para que coincida con el logo de la aplicación, y se ha añadido un aviso (conocido en Android como *toast*) con el mensaje de la alarma que se muestra en pantalla cuando suena la notificación:

```
final NotificationManager notificationMgr = (NotificationManager)
systemService;
final Notification notification = new Notification(R.drawable.icono,
tickerText, System.currentTimeMillis());
Intent notificationIntent = new Intent(context, Libreta.class);
final PendingIntent contentIntent = PendingIntent.getActivity(context, 0,
notificationIntent, 0);

switch (notificacionSonido) {
    case 0:
        notification.sound = Uri.parse("android.resource://" +
context.getPackageName() + "/" + R.raw.demonstrative);
        break;
        ...
        ...
        ...
    case 7:
        notification.sound = Uri.parse("android.resource://" +
context.getPackageName() + "/" + R.raw.solemn);
        break;
    default:
        notification.defaults |= Notification.DEFAULT_SOUND;
        break;
}

notification.vibrate = new long[] { 0, 100, 200, 300 };
notification.flags |= Notification.FLAG_AUTO_CANCEL;
notification.setLatestEventInfo(context, notificationTitle,
notificationSubText, contentIntent);

notificationMgr.notify(notificationId, notification);

Toast toast = Toast.makeText(context, notificationSubText,
        Toast.LENGTH_LONG);
toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
toast.show();
```

Para cancelar una alarma, se hace una llamada al método `cancel`, al que se le pasa como parámetro el identificador de la misma.

```
plugins.localNotification.cancel(id);
```

También es posible eliminar todas las alarmas conjuntamente mediante el método `cancelAll`:

```
plugins.localNotification.cancelAll();
```

Reiniciar el proceso

Dado que la información del progreso se almacena principalmente en el `LocalStorage` de la aplicación, reiniciar el proceso requiere como primer paso limpiar este almacenamiento. Tal y como se comentaba en apartados previos del presente documento, esta acción se realiza a través del método `removeItem`. Para eliminar las alarmas pendientes además es necesario cancelarlas de cara al sistema.

A continuación se muestra la función completa encargada de reiniciar el proceso de 21 días:

```
function reiniciarProceso() {
    var borrar = confirm(String.fromCharCode(191)+"Est\u00E9 seguro?\ Se
    borrar\u00E9 la informaci\u00F3n relativa al proceso, incluyendo la libreta
    y las alarmas")
    if ( borrar ){
        localStorage.removeItem("dia");
        localStorage.removeItem("hecho");
        localStorage.removeItem("notas");
        localStorage.removeItem("sonido");

        if (typeof plugins !== "undefined")
            plugins.localNotification.cancelAll();
        localStorage.removeItem("alarmas");

        self.close();
        goBack();
    }
}
```

De la misma manera, existen métodos específicos para eliminar tan solo parte de la información asociada al proceso, ofreciendo en la ventana de Opciones la posibilidad de borrar las alarmas o las notas individualmente

Cambiar el tono de las notificaciones

Una de las opciones de la aplicación radica en cambiar el sonido de las notificaciones, para lo que se proponen siete tonos diferentes. Antes de cambiarlo, se permite escuchar el tono de alarma haciendo uso del objeto Media de PhoneGap.

Para ello se establece la ruta del archivo según el elemento seleccionado en la interfaz, y se reproduce mediante el método `play()`. Mientras se está reproduciendo un sonido, se muestra una animación que identifica el proceso de reproducción a la vez que evita que el usuario seleccione varios sonidos al mismo tiempo.



Figura 48: Cambiar el tono de las alarmas

El método utilizado para este fin se resume a continuación:

```
function cargarTonos() {  
  /** Selecciona el elemento pulsado y desmarca los demás */  
  clearGroup();  
  
  /** Muestra la animacion de carga */  
  $(".loadingdiv").css({"display":"block"});  
}
```



```
/** Establece la ruta del archivo en funcion del elemento seleccionado*/
    elem = parseInt(event.currentTarget.id);
    switch (elem) {
    case 1:
        url = "/android_asset/www/res/audio/demonstrative.mp3";
        break;
    ...
    ...
    ...
    case 7:
        url = "/android_asset/www/res/audio/solemn.mp3";
        break;
    }

/** Reproduce el archivo de audio y oculta la animación al terminar */
    var my_media = new Media(url,
        function () {
            $(".loadingdiv").css({"display":"none"});
        },
        function (err) {
            console.log("playAudio():Audio Error: " + err);
        }
    );
    my_media.play();
}
```

8. Conclusiones y trabajo futuro

Este proyecto se ha basado en el uso del framework PhoneGap para el desarrollo de aplicaciones móviles híbridas. Para ello se han estudiado varias tecnologías punteras tanto para el desarrollo web como móvil, como son HTML5, CSS3 o JavaScript. Con ello se ha implementado una aplicación híbrida que hace uso de los recursos del dispositivo, y cuyo código fuente permite su compilación en diferentes plataformas.

Además, el desarrollo de esta aplicación se ha llevado a cabo utilizando una metodología ágil, apoyada tanto en herramientas web, como ScrumDo, como en aplicaciones de escritorio tales como Microsoft OneNote, facilitando el seguimiento del progreso.

La inversión inicial de tiempo dedicada al aprendizaje de todas estas tecnologías ha permitido analizar y plantear la implementación de este proyecto, aunque dicho aprendizaje haya seguido existiendo (en menor medida) durante todo el desarrollo del mismo.

Debido a la creciente cantidad de aplicaciones móviles que se pueden encontrar actualmente en el mercado, ofrecer una buena experiencia de usuario es crucial para que una aplicación destaque entre aquellas que dispongan de una temática similar. Con esta idea en mente, y teniendo en cuenta la gran variedad de dispositivos móviles existentes, se han utilizado técnicas como el Responsive Web Design para el diseño de la aplicación, cuidando la estética y la usabilidad en todo punto.

Pese a tratarse de una aplicación híbrida, se ha enfocado su desarrollo para la compilación con Android, debido tanto a su cuota de mercado como a la facilidad que ofrece este sistema operativo para la depuración de aplicaciones en dispositivos físicos. Por ello, el siguiente paso lógico consistiría en exportarla a otras plataformas, comenzando por iOS y Windows Phone por ser las siguientes en venta de dispositivos móviles a nivel mundial.

Para adaptar la aplicación a otros sistemas operativos es necesario tener en cuenta que uno de los plugins utilizados en su implementación (aquel que permite enviar notificaciones al usuario según las alarmas programadas) utiliza código nativo Android, por lo que deberían crearse plugins equivalentes para cada plataforma que permitan el correcto funcionamiento de esta utilidad. Una vez hecha esta modificación, bastaría con utilizar la herramienta PhoneGap Build para generar los archivos compilados correspondientes.

Dado que la aplicación se centra en contenido de terceros no se ha procedido a su publicación en ningún mercado de aplicaciones. No obstante, en caso de contar con el consentimiento del autor de los textos utilizados, podría publicarse en distintas plataformas (inicialmente Google

Play ya que se encuentra actualmente disponible en Android) o compartirse de forma directa (a través, por ejemplo, de PhoneGap Build).

Además, en caso de su publicación podría tenerse en cuenta la opción de monetizar la aplicación para obtener beneficios. Para ello habría que estudiar diversas opciones, tales como cobrar por las descargas de los usuarios o introducir publicidad, propia o externa.

9. Referencias y bibliografía

1. **Diseño y maquetación web para smartphones y tablets.**
Antonio Fernandez Clement. Octubre de 2012.
2. **Responsive Web Design with HTML5 and CSS3.**
Ben Frain. Packt Publishing Ltd. Abril de 2012.
3. **HTML5 and CSS3 Responsive Web Design Cookbook.**
Benjamin LaGrone. Packt Publishing Ltd. Mayo de 2013.
4. **Global Mobile Data Traffic Forecast Update, 2013–2018.**
Cisco Visual Networking Index. Febrero de 2014.
5. **Beginning HTML5 and CSS3. The Web Evolved.**
C. Murphy, R. Clark, O. Studholme, D. Manian. Apress. Diciembre de 2012.
6. **Aplicaciones móviles de la mano de PhoneGap.**
Dairo Galeano. Octubre de 2012.
7. **Faux Columns.**
Dan Cederholm. A List Apart. Enero de 2004.
8. **CSS3: the missing manual, 3rd edition.**
David Sawyer McFarland. O'Reilly Media; Diciembre de 2012.
9. **Responsive Web Design.**
Ethan Marcotte. A Book Apart. 2011.
10. **Introducción a jQuery.**
Gabriel Kaplan. Septiembre de 2013.
11. **Introducción a CSS.**
Javier Eguíluz Pérez. Libros Web. Diciembre de 2008.
12. **Introducción a JavaScript.**
Javier Eguíluz Pérez. Libros Web. Junio de 2008.
13. **Development Landscape: 2013. Developer Forrsights North America and Europe.**
Jeffrey S. Hammond, Vivian Brown, Phil Murphy, Rowan Curran. Agosto de 2013

14. A Dao of Web Design.

John Allsopp. A List Apart. Abril de 2000.

15. Primeros pasos con PhoneGap para Android.

Jorge Iván Meza Martínez. Mayo de 2011.

16. El panorama de los sistemas operativos móviles en 2013.

José M. Velo. Febrero de 2013.

17. Developing for Android: PhoneGap Versus Native.

Juha-Matti Laaksone. Junio de 2013.

18. Informe de ventas trimestral.

Kantar Worldpanel. Enero de 2014.

19. PhoneGap 2.x Mobile Application Development Hotshot.

Kerri Shotts. Packt Publishing Ltd. Febrero de 2013.

20. Mobile first.

Luke Wroblewski. A Book Apart. 2011.

21. Dive into HTML5.

Mark Pilgrim. 2013.

22. HTML5: Up and Running.

Mark Pilgrim. O'Reily. Agosto de 2010.

23. The History of HTML5.

Matt Silverman. Julio de 2012.

24. Manual de CSS3.

Miguel Ángel Álvarez. Junio de 2008.

25. jQuery desde Cero: Introducción.

Oscar González. Octubre de 2013.

26. iScroll: Implementando para PhoneGap.

PhoneGapSpain. Agosto de 2013.

27. Fundamentos de jQuery.

R. Murphey, L. D'Onofrio. Agosto de 2013.

28. **HTML5 forms input types.**
Richard Clark. Febrero de 2013.
29. **The State of Responsive Web Design.**
Stéphanie Walter. Smashing Magazine. Mayo de 2013.
30. **Australian Mobile Services Market Study 2011-2015.**
Telsyte Consulting. Junio de 2011.
31. **Building PhoneGap apps with jQuery Mobile.**
The jQuery Foundation. 2011-12.
32. **HTML Design Principles.**
W3C. Noviembre de 2007.
33. **Taxonomy of Mobile Application Platforms.**
W3C. Septiembre de 2013.
34. **CSS3 Introduction.**
W3School. 2014.
35. **HTML Tutorial – (HTML5 Compliant).**
W3School. 2014.
36. **HTML Living Standard.**
WHATWG. Febrero de 2014.
37. **Android Developers.** <http://developer.android.com/>
38. **Font Squirrel.** <http://www.fontsquirrel.com/>
39. **Freepik.** <http://www.freepik.com/>
40. **jQuery API.** <http://api.jquery.com/>
41. **Or!ngz.** <http://oringz.com/>
42. **PhoneGap Documentation.** <http://docs.phonegap.com/en/edge/index.html>
43. API del plugin **iScroll.** <http://cubiq.org/iscroll-4>
44. **HTML Living Standard** <http://www.whatwg.org/specs/web-apps/current-work/>

10. Anexos

10.1. Proceso de 21 días. Sergi Torres



El proceso de 21 días

Bienvenid@ al proceso de 21 días.

Éste es un proceso de transformación. Su objetivo es abrir una puerta para que tú puedas descubrir en ti mism@ las herramientas que te ayuden a hacer de tu cotidianidad un lugar de transformación.

En nuestro día a día encontramos todo lo que necesitamos para alcanzar una transformación profunda. Nuestra cotidianidad es el lugar donde podemos descubrir por qué nuestra vida es como es, por qué nos sentimos como nos sentimos y por qué pensamos lo que pensamos.

Nuestra vida se basa en un patrón que tiene como propósito llevarnos a la Paz y a la Felicidad. Sin embargo, no vemos ese propósito porque inconscientemente, huimos de ella en busca de paz y felicidad. Curioso, ¿verdad? Creamos nuestra personalidad y un mundo de objetos a nuestro alrededor y eso nos nubla la mirada.

Es por eso, que viajaremos a través de nuestra cotidianidad, de nuestras emociones y de nuestros pensamientos, para llegar al punto en el que nuestra vida pueda mostrarnos que lo que buscamos, ya está en nosotros y no en las cosas que nos ocurren.

Este proceso tiene algunas reglas para poder llevarlo a cabo de la forma más profunda posible. Son éstas:

1.- **Debes decidir tú, libremente, realizar este proceso.** No sirven de nada las recomendaciones de otros, pues están basadas en sus experiencias y no en las tuyas y lo que buscamos es trabajar con las tuyas.

2.- Es básico que **no cambies la dinámica de tu vida cotidiana para realizar el proceso.** Este es un proceso que debe ser aplicado a tu vida más corriente, por lo tanto, es importante que estés en contacto con tu entorno cotidiano: tu familia, tus amigos, en tu trabajo y dentro de tus costumbres habituales. En caso de que tu contexto haya cambiado de forma natural como por ejemplo por vacaciones, ese nuevo contexto también es “propicio”.



El proceso de 21 días



3.- Este es un programa abierto y flexible. Aun así, está construido sobre una estructura que no debe ser modificada a no ser que aquél que te acompaña en el proceso (tutor) así lo crea necesario. Por lo tanto, **es importante seguir las instrucciones del programa sin modificarlas**. Verás que la base estructural del programa es flexible.

4.- Este proceso es totalmente individual. Por ello se requiere compromiso con un@ mism@, respeto por un@ mism@ y dedicación a un@ mism@.

Debido a que nuestra herramienta de trabajo va a ser nuestro contexto cotidiano, **es recomendable** que no se comunique a las personas con las que convives habitualmente que estás utilizándolas para tu proceso. Esto podría alterar su comportamiento en relación a ti.

5.- No importa si eres practicante de alguna religión o si realizas alguna práctica espiritual. Tampoco el tipo de alimentación o si fumas. **En caso de tomar drogas o alcohol de forma habitual es importante primero atender la adicción** con algún otro programa específico para adicciones guiado por un profesional.

Si has tenido en algún momento de tu vida algún trastorno psicológico que haya sido tratado por profesionales, háznoslo saber.

¿Por qué realizamos el programa?

Este programa tiene la función de ayudarte a ver qué hay detrás de lo que hemos construido en nuestra vida y facilitar un proceso que ya está empezando a darse de forma natural en los seres humanos. Todos los seres estamos en evolución. Quizá nos toca dar un paso consciente en nuestra evolución particular, un paso que implica dejar lo viejo y entrar en un nuevo patrón de pensamiento inspirado por lo más auténtico y esencial de cada uno de nosotros. Algunos lo han realizado ya o están realizándolo. El paso más importante es aprender a soltar aquellos patrones de sufrimiento, tan arraigados en nuestra cultura-sociedad y en nuestras vidas cotidianas.

¿Qué esperamos del programa?



El proceso de 21 días



El objetivo de este programa, es mostrarte algunas herramientas que puedas usar para dejar de ser la víctima de tu vida y empezar a vivirla según los dictados de tu propio corazón. Para ello, debemos aprender a escucharlo atentamente.

No podemos esperar que este proceso nos convierta la vida en un jardín de flores, pero sí te mostrará cómo limpiar la tierra, cómo abonarla y cómo plantar la semillas que tú tienes en ese bolsillo trasero en el que nunca has buscado.

¿Cómo cambiará el programa mi vida?

Nada de lo que puedas imaginar que este proceso puede aportarle a tu vida te sirve en este proceso, pues son ideas previas al proceso. Es muy recomendable no atender a tus propias expectativas de aquellos cambios que puedes obtener en tu vida al terminarlo. **Los que realizan este programa lo hacen por un impulso interior y no por una expectativa exterior.** Como las orugas cuando crean la crisálida guiadas por un impulso interno y no por las ganas de ser una mariposa amarilla o roja.

Algunas ideas más.

- Este proceso no es un final sino el principio de un cambio a llevar a cabo por ti.
- Es interesante estar atento a nuevas ideas o sueños, sensaciones o intuiciones que puedan surgir durante el proceso. Éstas pueden sugerirte algunos cambios en tus hábitos durante el proceso.
- Es importante que cada día leas las instrucciones para el día presente. No te anticipes en la lectura del programa leyendo los días posteriores. Lee sólo aquella parte del día que vas a trabajar. Verás que por regla general el programa diario está dividido en tres partes:
 1. Pensamiento semilla o propósito del día
 2. Práctica del día
 3. Reflexión al terminar el día.

No te avances, así tu práctica será fresca y genuina sin estar marcada por lo que ya sabes que vas a realizar luego o al día siguiente.

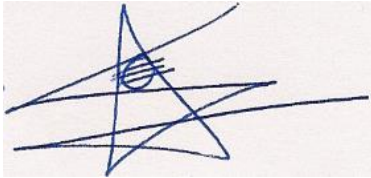


El proceso de 21 días

- Necesitarás una libreta para tus anotaciones.
- ¡Disfrútalo!

Muchas gracias y ¡adelante!

Sergi Torres



El proceso de 21 días



1ª SEMANA: VISIÓN

Lunes	Reconocer mis pensamientos	1
Martes	Reconocer mis emociones	2
Miércoles	La calidad de mis pensamientos	3
Jueves	Mapa mental pensamiento-emoción	4
Viernes	Entrenando mi mente	5
Sábado	INTEGRACIÓN Y PRÁCTICA	6
Domingo	INTEGRACIÓN Y PRÁCTICA	7

Nota: Es importante no leer las instrucciones de los días posteriores, así como también es importante leer sólo hasta la parte que voy a poner en práctica. Por ejemplo: Por la mañana leo la primera parte “Propósito del día”, luego leo las instrucciones para la práctica del día y por la tarde-noche (al finalizar mi día) leo la parte de “Al finalizar el día”. Esta parte está en color naranja para avisarnos que ahí es donde debemos dejar de leer y retomarlo al final del día.

DÍA 1 RECONOCER MIS PENSAMIENTOS.

Buenos días:

¡Bienvenido al primer día! Hoy entraré en contacto con mi forma de pensar. Es importante que pueda darme cuenta de que detrás de cada situación estoy yo interpretándola a través de mis pensamientos. Esto lo hacemos de una forma tan habituada que no nos damos cuenta.

Empiezo el día dedicando unos segundos a afirmar lo siguiente:





El proceso de 21 días

“Hoy voy a dedicar mi día a atender mis pensamientos.

No los voy a juzgar, sólo los observaré”.

Dedicaré el día a poner en marcha mi capacidad de atención. Estaré atento a mis pensamientos. Observaré qué pensamientos pienso dentro de cada área de mi cotidianidad: en el trabajo, en casa, con mi familia-amigos, (pareja, si la tengo).

No necesito apuntarlos. Los pensamientos ahora no son importantes. Lo importante es mi decisión de darme cuenta de mi capacidad de ser consciente de ellos. [No se trata de estar consciente de todo lo que pienso.](#)

La atención completa de un instante puede hacerme avanzar muchísimo más que mi autocrítica por no haber estado atento todo el día. De hecho, mi autocrítica es también un pensamiento al que puedo observar. Debo recordar que hago esto para aprender y no para hacerlo bien. [En este proceso no existe correcto o incorrecto, sino el descubrimiento de mi mismo.](#)

Al finalizar el día:

Me siento cómodo justo antes de dormirme y respondo muy brevemente a la pregunta:

¿Qué he descubierto hoy de mí? (anoto en la libreta)

DÍA 2 RECONOCER MIS EMOCIONES .

Buenos días:

Hoy seguiré poniendo en marcha mi capacidad de visión interna. Hoy la ejercitaré con mi campo emocional.

Empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar mi día a atender mis emociones.

No las voy a juzgar, sólo las observaré”.





El proceso de 21 días

Observaré cómo siento las emociones. Qué emociones siento y qué es aquello que, bajo mi punto de vista, las detona. (cada hora me pararé y revisaré con la intención de encontrar una (sólo una) emoción intensa que recuerde haber sentido y la anotaré en la libreta junto a una breve descripción del contexto en el que me encontraba. En caso de que mis circunstancias de vida no me permitan hacerlo cada hora, lo ajustaré a mis posibilidades horarias, sabiendo que esto [no va a mermar la efectividad del ejercicio](#).

Al finalizar el día:

Miraré mis anotaciones y elegiré un mínimo de tres de esas emociones anotadas, con la intención de descubrir si contienen miedo. Puede que éste se perciba muy sutilmente debajo de las emociones anotadas en la libreta. Una vez terminado esto me pregunto:

¿Qué he descubierto hoy de mí? (anoto en la libreta)

DÍA 3 LA CALIDAD DE MIS PENSAMIENTOS.

Buenos días:

Hoy voy a afinar mi capacidad de estar atento.

Empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar mi día a atender la calidad de mis pensamientos. No la voy a juzgar, sólo la observaré”.

Hoy pondré mi atención en la calidad de mis pensamientos. Para poder ver la calidad debo empezar el ejercicio observando los pensamientos tal y como lo hice el primer día. La calidad la veré en el ejercicio que haré al terminar el día. Es importante estar atento también a la posibilidad de juzgar mis pensamientos. [Esos juicios son también pensamientos](#). (voy anotando en una libreta igual que ayer cada hora un pensamiento y las circunstancias en las que lo pensé).

Al finalizar el día: Miraré mis anotaciones y elegiré un mínimo de tres de esos pensamientos. Entonces miro en qué situación pensé ese pensamiento y por cada pensamiento me pregunto :





El proceso de 21 días

¿Si en lugar de elegir pensar este pensamiento, hubiera elegido pensar otro pensamiento distinto, hubiera visto la situación de forma distinta?

Una vez respondida me pregunto:

¿Qué emoción producía mi pensamiento?

Terminada esta autoindagación con los pensamientos de la libreta puedo ver qué calidad emocional tienen mis pensamientos.

¿Es una calidad de pensamiento que me beneficia?

Al finalizar me pregunto:

¿Qué he descubierto hoy de mí? (anoto en la libreta)

DÍA 4 EL VÍNCULO PENSAMIENTO-EMOCIÓN.

Buenos días:

Seguimos con el trabajo anterior.

Empiezo el día dedicando unos segundos a afirmar lo siguiente:

**“Hoy voy a dedicar mi día a descubrir qué hay detrás
de lo que me ocurre en mi vida”.**

Uniré lo aprendido hasta hoy. (No usaremos libreta). Me haré consciente de mis emociones e iré a mirar qué pensamiento hay detrás de esa emoción. Cuando sienta una emoción intensa me pararé un instante y me preguntaré:

**¿Qué es lo que estoy pensando
que me ha llevado a sentir esto que siento?**

Observo mis emociones y busco qué pensamiento o idea es la causa de esa emoción. De esta manera, descubro cómo las circunstancias en las que yo siento esa emoción, pasan a un segundo plano y mis ideas-pensamientos y mis sensaciones-emociones a primer plano.





El proceso de 21 días

Al finalizar el día me pregunto:

**¿Qué implicaría para mí que mi vida
sólo tuviera que ver con lo que yo pienso
y no con las situaciones externas?**

Una vez respondida la pregunta anterior:

¿Qué he descubierto hoy de mí? (anoto en la libreta)

DÍA 5 Entrenando MI mente

Buenos días:

Empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar mi día a descubrir las profundidades de mi mente. Sin miedo a ver”.

Hoy aprendo a usar una herramienta que me ayudará a mirar con más honestidad y claridad mi manera de pensar, mi sistema de pensamiento. Trabajaré con los mapas mentales.

Nuestra mente (que incluye a nuestro campo emocional, pues también es parte de ella) está diseñada para que la consciencia fluya a través de ella. Los mayoría de seres humanos, sin embargo la usamos para almacenar ideas y emociones, creando así estructuras de pensamiento (recuerdos, creencias, resentimientos). Esto nos lleva a usar una herramienta como la mente, con un fin para el que no está realmente diseñada: crear nuestra personalidad, de la cual se desprende nuestra forma de ser y de ver al mundo, a los demás y a nosotros mismos.

Mapas Mentales:

En una hoja de papel dibujo un círculo concéntrico. En él escribo mi nombre. Luego cierro los ojos y miro qué pensamiento está almacenado en mi mente y lo escribo vinculándolo al círculo con mi nombre tal como muestra la imagen. Una vez he localizado el pensamiento, miro su calidad al preguntarme ¿Qué siento cuando yo pienso este pensamiento? (sensación o emoción) y lo escribo vinculado al círculo del pensamiento.



El proceso de 21 días



Sigo mirando el mismo pensamiento. Esta vez miro que siento detrás de esa emoción o sensación y si encuentro alguna, la escribo vinculada al mismo pensamiento de nuevo y sigo mirando hasta que no sienta nada más. Es importante que me quede un momento sintiendo antes de pensar que ya no queda nada más almacenado. Una vez seguro de que no queda nada más, paso a mirar otro pensamiento almacenado y sigo con el mismo proceso de descubrir la calidad-emociones de éste. Con cinco pensamientos bien observados, será suficiente, aunque si lo deseo puedo seguir anotando algunos más. Es importante que tanto los pensamientos como las emociones lleguen a tu mente. No te esfuerces en encontrarlos, más bien **permítete** ver aquello que no ves.



Al finalizar el día me pregunto:

¿Qué implicaría que todo el mundo que veo a mi alrededor fuera la manifestación de mi manera de pensar?

Una vez respondida la pregunta anterior:

¿Qué he descubierto hoy de mí? (anoto en la libreta)



El proceso de 21 días



DÍA 6 INTEGRACIÓN Y PRÁCTICA.

Buenos días:

Empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a permitirme ver todo lo que me rodea como un efecto de mi forma de pensar”.

Hoy es un día de integración y práctica. Sin una directriz definida, voy a girar alrededor de la afirmación con la que he empezado el día.

Al finalizar el día me pregunto:

¿Qué he descubierto hoy de mí? (anoto en la libreta)

DÍA 7 INTEGRACIÓN Y PRÁCTICA.

Buenos días:

Empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a permitirme ver mi vida como una extensión de mí mismo”.

Hoy es el segundo día de integración y práctica. Al igual que ayer, sin una directriz definida, voy a girar alrededor de la afirmación con la que he empezado el día.

Al finalizar el día me pregunto:

¿Qué he descubierto hoy de mí? (anoto en la libreta)



El proceso de 21 días



2^a Semana

CONEXIÓN

Lunes	Contactar con mi vida	8
Martes	Contactar con el Principio Esencial	9
Miércoles	Contactar con mi alma	10
Jueves	Contactar con mi función de vida	11
Viernes	Integrar vida, alma, función, principio Esencial	12
Sábado	INTEGRACIÓN Y PRÁCTICA	13
Domingo	INTEGRACIÓN Y PRÁCTICA	14

Nota: Es importante no leer las instrucciones de los días posteriores, así como también es importante leer sólo hasta la parte que voy a poner en práctica. Por ejemplo: Por la mañana leo la primera parte “Propósito del día”, luego leo las instrucciones para la práctica del día y por la tarde-noche (al finalizar mi día) leo la parte de “Al finalizar el día”. Esta parte está en color naranja para avisarnos que ahí es donde debemos dejar de leer y retomarlo al final del día.

8 Contactar con mi vida.

Buenos días:

¡Bienvenido al octavo día! Hoy entraré en contacto con mi vida.

Empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar el día a sentir mi vida





El proceso de 21 días

de una forma que jamás he sentido antes”.

Luego a modo de meditación dedicaré un momento a realizar una imagen mental de mi vida (arquetipo) . Para hacerlo me sentaré cómodamente, cerraré los ojos y le pediré a mi mente lo siguiente:

“Ofréceme una imagen que represente a mi vida entera”.

Esta imagen vendrá a mi mente, así que dedicaré pacientemente un momento para que esto ocurra.

La imagen va a representar mi vida al completo. No voy a preocuparme por la imagen que me ha llegado: puede ser un recuerdo de algo ocurrido o la imagen de un anhelo en la vida o un símbolo abstracto o geométrico, un color. No importa que imagen llega a la mente, la acojo como arquetipo que va a representar mi vida durante el día de hoy.

Durante el día, cada hora me pararé un minuto. Cerraré los ojos e iré a encontrar el arquetipo de mi vida. Luego lo miraré y me abriré a amarla. No me preocuparé en caso de no sentir nada especial o si lo que siento difiere del amor. **Mi ejercicio consiste en abrirme a amar esa imagen** y no en conseguir amarla. Es posible que durante la práctica de cada hora, la imagen de mi vida cambie. En caso de que cambie: LO PERMITO y a la siguiente hora iré a encontrar la nueva imagen y me abriré a amarla.

Dedicaré todo el día a esta práctica.

Al finalizar el día:

Me siento cómodamente y me abro a amar mi vida. Observo que lo que he estado haciendo durante el día es abrirme a amar la imagen que mi mente tiene de mi vida.

Me abro a sentir mi vida más allá de esa imagen. Observo a medida que respiro pausadamente, que mi vida sólo está teniendo lugar en este mismo instante: no antes, no después. Observo que siempre vivo en el momento en el que mi vida está naciendo. La siento y la respiro. Observo que este principio de vida que siento es lo que me da origen a mí, es lo que me permite respirar, pensar, sentir, ser.

**“Existo debido a este Principio de Vida y no
debido a lo que hago, pienso o siento”**



El proceso de 21 días



Luego:

¿Qué he descubierto hoy en mí? (anoto en la libreta)

DÍA 9 Contactar con mi Principio de Vida

Hoy entraré en contacto con mi Principio de Vida.

Empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar el día a sentir mi Principio de Vida”.

Hoy voy a organizar mi tiempo de forma que, en la medida que no fuerce ningún acontecimiento, dedicaré 7 minutos de cada hora a conectar y sentir el Principio de Vida, mi origen. Hoy me abro a descubrir que mi cotidianidad sí puede convertirse en mi propio “monasterio”.

Durante la práctica de hoy voy a centrarme en [mirar la sensación o consciencia de ser producto de ese Principio de Vida](#). Voy a abrirme a sentirme el efecto de esa Causa interna y a unirme a Él.

Al finalizar el día:

Me siento cómodamente y me abro a sentir mi Principio de Vida y me digo:

“Este Principio de Vida está en todo.

Nada de lo que yo hago, pienso o siento puede alterar eso”

Luego:

¿Qué he descubierto hoy en mí? (anoto en la libreta)

DÍA 10 Contactar con mi alma.

Hoy pondré mi atención en una parte de mí que es esencial, simple y profunda.

Empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar el día a sentir mi Alma”.





El proceso de 21 días

Hoy dirigiré mi mirada a aquella parte de mí, que está detrás de mis pensamientos y de mis emociones. Hoy voy a centrar mi mirada a aquella parte de mí, esencial, que es producto del Principio de Vida. Voy a dirigir mi mirada a aquella parte de mí que comparto con toda forma de vida.

Buscaré, lo más temprano posible, un momento del día en el que pueda dedicar al menos 15 minutos para poder parar y centrarme en mirar esa parte esencial de mí. Cerraré los ojos y me preguntaré a mí mismo:

“¿Qué existe en mí que comparto con toda forma de vida?”

Pondré mi atención en encontrar aquello de mí que existe debido al Principio de Vida. Sabré que lo he encontrado porque sentiré paz, expansión, apertura, libertad,... Es posible que en lugar de conectar con mi alma, conecte con los obstáculos que yo mismo escondí y los puse entre mi conciencia y mi alma, a medida que a lo largo de mi vida fui almacenando resentimientos, ira, sufrimiento, etc. En ese caso puede que sienta miedo, tristeza, frustración, rabia, indiferencia, nada,... De ser así, no tengo porqué cuestionarlo. [Simplemente sé que para mí está siendo así y sigo adelante con el plan del día.](#)

Una vez dedicado este tiempo a contactar con el alma, indiferentemente de si conecté con ella o con mis obstáculos, seguimos adelante con el día. NOTA: no conectar con nada es haber conectado con la “sensación de no haber conectado con nada”. Esa sensación también es un obstáculo que dirige a la mente a desistir de su voluntad de conectar con el alma.

A lo largo del día, cada hora dedicaré al menos 1 minuto a respirar esa parte esencial de mí. En caso de que conectase con un obstáculo, lo que haré será pararme a sentirlo de nuevo y respirarlo. Ese gesto, también me dirige hacia el origen esencial del obstáculo (mi alma). Si no retorna la misma sensación anterior, me centraré en la actual. Debo recordar que la consciencia es transformadora de por sí. Así que es posible que, cada vez que conecte con algo que no sea propiamente de naturaleza esencia, puede que se transforme.

Hoy me centraré en conectar con mi alma sin pararme a valorar aquello con lo que conecto. Donde pongo mi enfoque es en [“conectar con mi alma”](#) y no en “aquello con lo que conecto”. De esta forma, el enfoque de mi mente y mi voluntad se alinean apuntando hacia mi alma.



El proceso de 21 días



Durante el día estaré especialmente receptivo a todo lo que vaya ocurriendo. Voy a vivir todas las situaciones del día en las que recuerde este ejercicio de forma abierta y receptiva. Voy a permitir a las situaciones que toquen lo más profundo de mi mismo.

Si estoy atento, podré ver como todo lo que ocurre a mi alrededor está intentando tocar mi alma. En ese intento es posible que impacte en un obstáculo. Eso me favorece también, porque gracias a esos acontecimientos externos, podré ver los obstáculos más profundos que por mí mismo no pude o no quise ver.

Al finalizar el día:

Me siento cómodamente y conecto durante al menos 5 minutos con esa parte profunda de nuevo y la respiro, la toco, la siento. Al terminar me pregunto:

“Qué he descubierto hoy en mí?” (anoto en mi libreta).

DÍA 11 Contactar con mi Función de vida.

Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a conectar con mi función esencial en mi vida”

Hoy voy a dedicar parte del día a integrar los tres días anteriores. Lo haré regresando cada hora a ese espacio interno, esencial. Pararé cada hora y durante al menos un minuto, respiraré mi presencia. En ella se encuentra la vida, su esencia y aquello en nosotros que es producto de estas dos.

En caso de que aún siga encontrándome con obstáculos, respiraré agradecidamente esas sensaciones o emociones.

Para conectar con mi función de vida, encontraré un momento del día en el que pueda dedicar unos 30 minutos. No importa el “cuando”. Sí buscaré que sea un momento que pueda dedicarme a mí, completamente (a ser posible antes de **“Al finalizar el día”**).

Luego:

1. Buscaré un espacio donde poder sentarme cómodamente y entrar en conexión con mi presencia. Iniciaré un instante de respiración de mi presencia.



El proceso de 21 días



2. Una vez pueda sentirla con claridad dirigiré la siguiente pregunta hacia mi presencia:

¿Cuál es mi función de vida?

Y esperaré en silencio a que la respuesta surja de dentro de mí. Este es un silencio basado en la honestidad, basado en el hecho de que estoy viviendo una vida sin saber por qué estoy viviéndola. Es un silencio en el que no busco la respuesta sino que me abro a recibirla.

2 a. En caso de que ésta no llegue, suelto el ejercicio y lo repito en la tarde-noche en **“Al finalizar el día”**.

3. En caso de recibir una respuesta, que puede llegar de muchas formas (voz interior, imagen, sensación, ...), Me abro a descubrir como mi función es una expresión o extensión de mí mismo, de mi vida.
4. Una vez aquí, me paro un momento y afirmo lo siguiente:

“Permito que mi vida tome la forma que deba

alcanzar para poder expresar completamente

mi función de vida”

5. Regreso a mi presencia respirándola y sigo el día.

Nota: Ten en cuenta que la función puede ser genérica y puede irse concretando o variando de forma a lo largo de la vida.

Al finalizar el día:

Me siento cómodamente:

a) En el caso de que anteriormente **no consiguiera conectar** con mi función de vida, realizo de nuevo el paso anterior. Una vez realizado, tanto si conecto con ella o no, paso al paso siguiente: Me abro a descubrir que este instante presente es la forma actual de mi función.

b) En caso de que sí consiguiera conectar con mi función, conecto con mi presencia y luego con mi función. Luego, me abro a descubrir que este instante presente es la forma actual de mi función.

Al terminar:

¿Qué he descubierto hoy en mí? (anoto en la libreta)



El proceso de 21 días



DÍA 12 Integrar mi vida, mi alma, mi función y Principio de Vida.

Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Mi vida, mi alma y mi función, son una herramienta a través de la cual la consciencia universo se expresa.”

Todo, absolutamente todo, si lo miro desde la perspectiva de la consciencia de la totalidad, es una forma de expresión universal. Todo contiene la voluntad de la consciencia universal de expresarse a través suyo. Eso me incluye a mí y todo lo que represento como ser humano. Detrás de cada cosa, persona o situación hay consciencia expresándose plenamente.

Nada en realidad, puede ser un obstáculo a la consciencia universal que fluye a través de mí, sin embargo si puedo elegir no darme cuenta de ello y pasar a identificarme con mis pensamientos y emociones en lugar de permitirlos fluir a través de mí.

Así que me encuentro dentro de sólo dos posibilidades:

1. Abrirme a ser una expresión universal, abriéndome a mi vida a mi alma y a mi función.
2. Cerrarme a ser consciente de ser una expresión universal, evitando sentir emociones, mendigando amor y respeto a los demás, creyéndome mis pensamientos y evitando vivir mi vida tal y como es ahora.

Ninguna de estas dos opciones es correcta o incorrecta. Simplemente me dirigen a un mundo o a otro, en función de mi elección.

Hoy dedicaré el día a estar atento a esa elección. Esta elección se realiza a cada instante y sus efectos físicos, mentales y emocionales son inmediatos. Observaré, en los momentos del día en los que recuerde el ejercicio, qué estoy eligiendo hacer en esta situación o con esta persona o conmigo mismo preguntándome: ¿Estoy abriéndome a vivirlo o estoy decidiendo evitarlo? Luego me pregunto:

¿Qué siento cuando me abro a vivirlo?

¿Qué siento cuando me cierro a vivirlo?



El proceso de 21 días



Al finalizar el día:

Me siento cómodamente y dedico unos instantes a ver qué he descubrierto hoy de mí. Luego me pregunto:

¿Estoy realmente abierto a vivir?

Yo, como forma de vida, parte de un universo,

¿Estoy realmente dispuesto ser una expresión viva de éste?

DÍA 13 Integración y Práctica.

Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a abrirme a ser una expresión de Vida”

Hoy voy a dedicar el día a respirar mi presencia viéndola como una forma de expresión de la Vida. Sin ninguna estructura temporal concreta, cada vez que recuerde el ejercicio, me abriré a respirar mi presencia y a sentirme el efecto de la Vida. La situación en la que me encuentre no va a representar un obstáculo, más bien lo contrario. Voy a permitir, que las situaciones en las que voy a ir encontrando hoy, señalen aquellos espacios de mí en los que no quiero abrirme a mi presencia para así poder decidir lo contrario.

Al finalizar el día:

Me siento cómodamente y dedico unos instantes a conectar con mi presencia. Luego me pregunto:

¿Qué he descubrierto hoy en mí? (anoto en la libreta)



El proceso de 21 días



DÍA 14 Integración y Práctica.

Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar mi día a sentir mi presencia”

Hoy dedicaré mi día a respirar mi presencia. Cada vez que lo recuerde, me pararé un instante y respiraré mi presencia en ese mismo instante. No importa si estoy sintiendo emociones intensas o estoy en medio de una reacción por mi parte. Simplemente me abriré a sentir mi presencia y a permitirme a mí mismo estar viviendo eso que acontece, sea lo que sea. Debo recordar que la motivación de sentir mi presencia no es alcanzar nada ni evitar sentir emociones de ningún tipo. La motivación de sentir la presencia es simplemente sentirme, sin más. Sin pretensiones añadidas, sin expectativas de resultados.

Al finalizar el día:

Me siento cómodamente y dedico unos instantes a conectar con mi presencia. Luego me pregunto:

¿Qué he descubierto hoy en mí? (anoto en la libreta).

Luego leo las instrucciones para el día siguiente (15º) debido a que la práctica de mañana empieza justo al despertar.



El proceso de 21 días



3^a Semana

LIBERACIÓN

Lunes	Suelto mi pasado	15
Martes	Abriendo paso a la Felicidad	16
Miércoles	Abriendo centros de conciencia (1)	17
Jueves	Abriendo centros de conciencia (2)	18
Viernes	Abriendo centros de conciencia (3)	19
Sábado	Establezco mi propósito de vida	20
Domingo	Confianza, Aceptación y Agradecimiento	21

Nota: Es importante no leer las instrucciones de los días posteriores, así como también es importante leer sólo hasta la parte que voy a poner en práctica. Por ejemplo: Por la mañana leo la primera parte "Propósito del día", luego leo las instrucciones para la práctica del día y por la tarde-noche (al finalizar mi día) leo la parte de "Al finalizar el día". Esta parte está en color naranja para avisarnos que ahí es donde debemos dejar de leer y retomarlo al final del día.

DÍA 15 Suelto mi pasado

Al despertar por la mañana:

Justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí. (esto lo haremos cada mañana al despertar hasta el final del proceso).





El proceso de 21 días

Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

**“Hoy voy a dedicar mi día soltar
todo lo que no pertenece a mi presente”.**

Hoy vamos a dedicar el día a encarar esta última semana de liberación. A lo largo del día, cada vez que recuerde el pensamiento-propósito de hoy, voy a parar y a respirar mi presencia. Se trata de hacerme consciente de que estoy en mi presente, que estoy justo en el único momento en el que mi vida está teniendo lugar. De esta forma deposito mi atención en el presente y la alejo de la “forma” en la que ocurre mi presente. Hoy no me interesan los acontecimientos, ni cómo son, ni porqué son así, ni si me gustan o no. Voy estar atento a que me interese el hecho de que está ocurriendo lo que ocurre, sin interferir, sólo estando abierto a vivirlo, a sentirme, a sentirlo.

A lo largo del día busco un momento en el que pueda sentarme tranquilamente y dedicarme durante 45 minutos a explorar mi “mente almacén”. En estos 45 minutos, realizaré un **mapa mental** con la intención de ver qué es aquello que aún arrastro de mi pasado y que no sólo no me sirve, sino que además altera la calidad de mis pensamientos felices.

Tengo la tendencia de almacenar recuerdos y emociones, convirtiéndome así en un almacén. Sin embargo soy un río de pensamientos y emociones que podría fluir abiertamente permitiendo así un orden natural interno. Hoy voy a abrirme a reestablecer ese orden. ¡Voy a abrir la puerta de mi almacén!

Delante de un papel en blanco dibujo un círculo en el centro y escribo mi nombre. (igual que en el día 5º del proceso). Entonces paro, cierro los ojos y miro que almaceno dentro de mí . No importa si es una idea, un recuerdo, un deseo de hace mucho tiempo o una emoción. Una vez visto lo que sea que veo, lo anoto partiendo del círculo central y miro todas sus asociaciones. (puede ser útil mirar la imagen del 5º día).

En caso de ser un pensamiento, ya sea en forma de recuerdo o de deseo, miro todas las emociones y sensaciones asociadas a ese pensamiento. Miro qué siento cuando pienso o recuerdo “eso”.



El proceso de 21 días



En caso de ser una sensación o emoción, me abro a ver si hay más sensaciones o emociones vinculadas a ella y finalmente miro qué idea o pensamiento está apoyando esa sensación.

En caso de que no llegue a ver lo que hay detrás de lo que pienso o siento, no me preocuparé. Lo importante no es verlo, sino abrirme a verlo.

Una vez he plasmado en el papel todo lo que he visto en mi almacén, miro el papel y valoro si vale la pena seguir sosteniendo esas ideas, recuerdos, emociones. En el caso de querer liberarme, cierro los ojos y pienso.

**“Yo no soy un almacén de recuerdos
que no me permiten ser feliz ahora”.**

¡Hoy me libero de mí mismo!

Al finalizar el día:

Cierro los ojos y miro si encuentro algún recuerdo doloroso en mí. En caso de encontrarlo, lo miro y me despido de él. **Es muy importante no liberarme a través de rechazar los recuerdos.** La aceptación es la acción más liberadora que existe, teniendo en cuenta que aceptar no es resignarme, sino abrirme a que eso sea bienvenido y despedido. Rechazar emociones o pensamientos los refuerza porque **les estoy prestando atención (consciencia)**. En cambio, una vez bienvenido, puedo soltarlo. Igual que cuando abro la palma de la mano y la suave brisa se lleva esa mota de polvo que guardaba.

De esta manera, empiezo una forma de pensar que le permite a lo que pasa por la mente elevarse y no quedarse retenido en ella, creando así, estructuras rígidas de pensamiento, formando las grandes vías neuronales, basadas en creencias y opiniones y que al mismo tiempo producen la sensación de no poder salir de ellas. **¡Hoy es un día de liberación!**

Luego:

¿Qué he descubierto hoy en mí? (anoto en la libreta).

Nota: Al despertar mañana, justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí.



El proceso de 21 días



DÍA 16 Abro paso a la felicidad

Al despertar por la mañana:

Justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí. (esto lo haremos cada mañana al despertar hasta el final del proceso).

Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar mi día a sentir la felicidad”.

Hoy decidiré alinear mi vida con la felicidad. Cada vez que recuerde el pensamiento semilla de hoy, voy a tomar conciencia de lo que siento justo en ese momento. Una vez esté en contacto con las sensaciones y/o emociones que siento, sin importar lo que sea que siento, me abro a descubrir que eso que siento es la forma que adopta mi felicidad en este momento presente y pensaré.

“Esto que siento ahora es la forma presente

en la que se manifiesta mi felicidad”.

Todo lo que siento surge de mí. Todo lo que experimento surge de ese lugar profundo en mí y a medida que se aleja de mí va tomando forma. Esa forma puede ser cualquier emoción o sensación.

Cuando mi esencia contacta con mi campo emocional, se convierte en emoción y su calidad emocional, surgirá en función de mi enfoque o pensamiento, a través del cual interpreto lo que ocurre en mi presente.

Al finalizar el día:

Me siento cómodamente y observo que siento. Me abro a descubrir que eso que siento, sea lo que sea, es una forma de felicidad. Me mantengo sintiéndolo sin ninguna intención de hacer nada con ello. Sin la intención de mantenerlo, en caso de que me resulte



El proceso de 21 días



agradable lo que siento, y sin la intención de cambiarlo, en caso de que me resulte desagradable lo que siento. Y observo qué ocurre.

Luego:

¿Qué he descubierto hoy acerca de mi felicidad?

(anoto en la libreta).

Nota: Al despertar mañana, justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí.

DÍA 17 Abriendo centros de consciencia (1)

Al despertar por la mañana:

Justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí. (esto lo haremos cada mañana al despertar hasta el final del proceso).

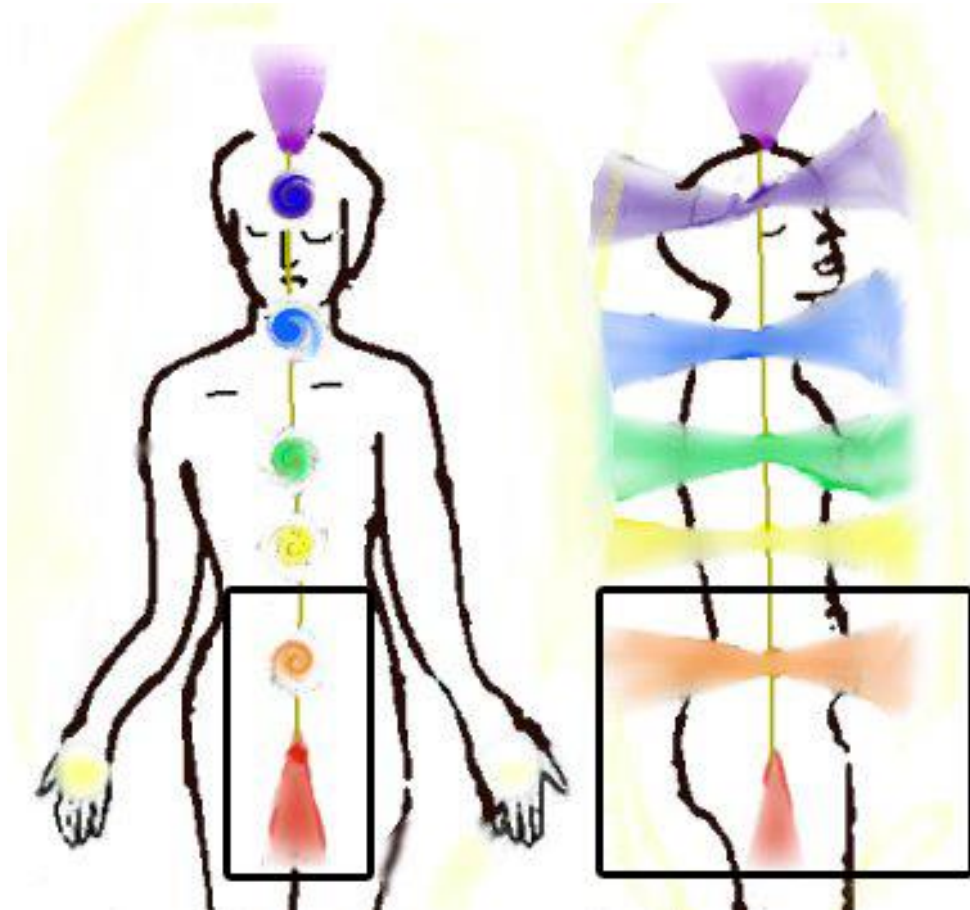
Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar mi día sentir mis dos centros de consciencia elementales”



El proceso de 21 días



Hoy es un día de pocas directrices y especificaciones. A lo largo del día voy a sentir mi presencia a través de los centros primero y segundo llamados también raíz y sexual.

Más allá de lo que yo pueda conocer o desconocer acerca de estos centros, mi propósito es poner mi atención en ellos. De nuevo sin ninguna pretensión más que sentirme desde esos espacios y observar. En caso de no saber dónde están exactamente ubicados, no me preocupo y paso a descubrirlos por mí mismo. Recuerdo que no se trata de saber, sino de descubrir.

Estos centros fueron descritos y llamados chakras (ruedas) en la cultura la Hindú. También existen referencias en escritos Vedas datados del siglo VII a.C. En nuestro caso lo que queremos de nuevo es descubrir qué ocurre en mí cuando pongo mi atención en esos centros y no lo que saben otros o yo ya sé de antemano.





El proceso de 21 días

A lo largo del día, cada vez que recuerde el pensamiento semilla me pararé y sentiré mi presencia a través de estos dos centros y observaré.

Al finalizar el día:

Me siento cómodamente y me siento a través de esos dos centros. Observo y me pregunto:

**¿Qué sensaciones o emociones he descubierto
en cada uno de ellos?**

Una vez visto lo que siento, sea lo que sea, dirijo toda mi atención a ello, lo respiro y lo suelto. Primero en uno de los centros y luego en el otro.

Luego:

¿Qué he descubierto hoy acerca de mí?

(anoto en la libreta).

Nota: Al despertar mañana, justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí.

DÍA 18 Abriendo centros de consciencia (2)

Al despertar por la mañana:

Justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí. (esto lo haremos cada mañana al despertar hasta el final del proceso).

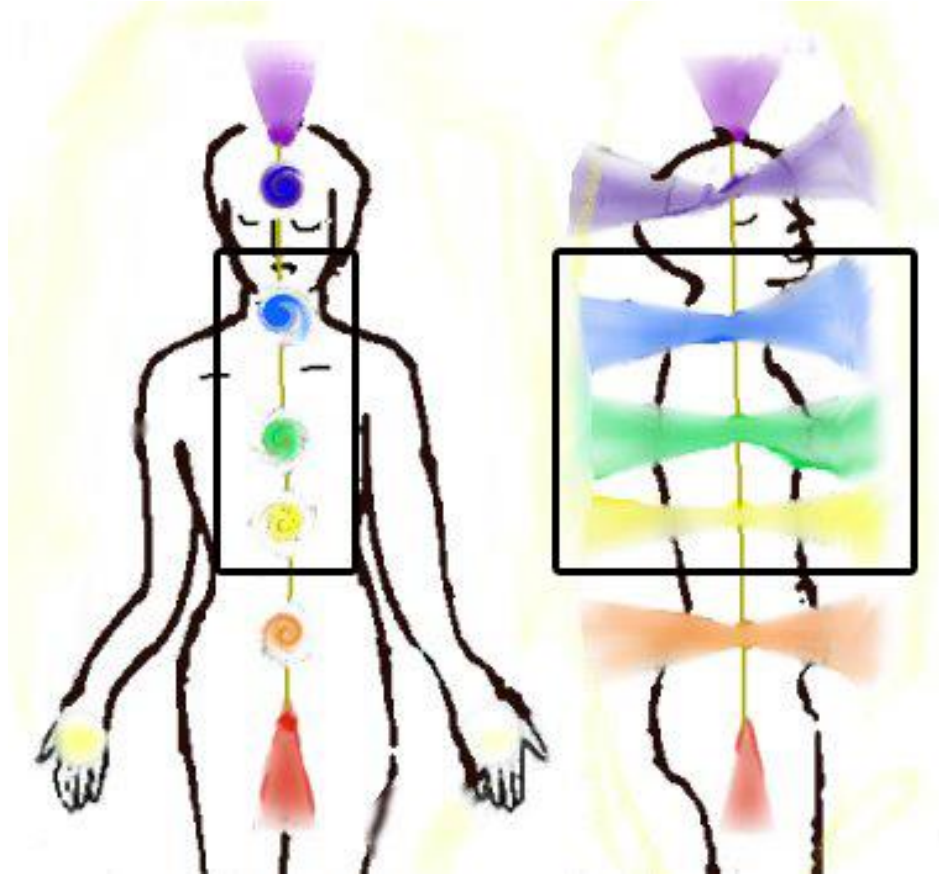
Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar mi día sentir mis dos centros de consciencia emocionales”



El proceso de 21 días



A lo largo del día voy a sentir mi presencia a través de los centros tercero, cuarto y quinto llamados también plexo solar, cardíaco y laríngeo.

Al igual que ayer, cada vez que recuerde el pensamiento semilla me pararé y sentiré mi presencia a través de estos tres centros y observaré.

Al finalizar el día:

Me siento cómodamente y me siento a través de esos tres centros. Me pregunto:

¿Qué sensaciones o emociones he descubierto

en cada uno de ellos?

Una vez visto lo que siento, sea lo que sea, dirijo toda mi atención a ello, lo respiro y lo suelto. Primero en uno de los centros y luego en el otro y termino con el siguiente.



El proceso de 21 días



Luego:

¿Qué he descubierto hoy acerca de mí?

(anoto en la libreta).

Nota: Al despertar mañana, justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí.

DÍA 19 Abriendo centros de consciencia (3)

Al despertar por la mañana:

Justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí. (esto lo haremos cada mañana al despertar hasta el final del proceso).

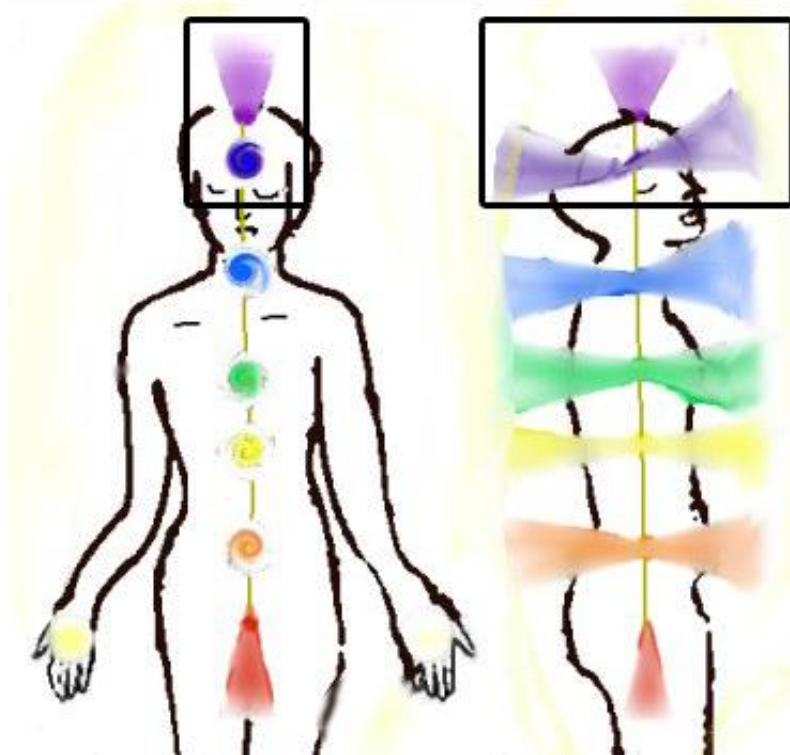
Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar mi día sentir mis dos centros de consciencia mentales”



El proceso de 21 días



A lo largo del día voy a sentir mi presencia a través de los centros sexto y séptimo llamados también tercer ojo, corona.

Al igual que los dos días anteriores, cada vez que recuerde el pensamiento semilla me pararé y sentiré mi presencia a través de estos dos centros y observaré.

Al finalizar el día:

Me siento cómodamente y me siento a través de esos dos centros. Me pregunto:

¿Qué sensaciones he descubierto

en cada uno de ellos?

Una vez visto lo que siento, sea lo que sea, dirijo toda mi atención a ello, lo respiro y lo suelto. Primero en uno de los centros y luego en el otro.



El proceso de 21 días



Luego:

¿Qué he descubierto hoy acerca de mí?

(anoto en la libreta).

Nota: Al despertar mañana, justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí.

DÍA 20 Establezco el propósito de vida.

Al despertar por la mañana:

Justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí. (esto lo haremos cada mañana al despertar hasta el final del proceso).

Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

**“Hoy voy a dedicar mi día a establecer
mi propósito de vida”.**

Muy buenos días:

Hoy estableceré mi propósito en la vida. Una vez ya he visto que todo lo que yo experimento en mi vida se origina en mí, hoy voy a establecer un propósito que pueda unir conscientemente el flujo de vida que nace a través de mí y mi enfoque mental.

Para poder hacerlo voy a programar un momento del día en el que sepa que puedo dedicar unos 30 minutos a estar en contacto profundo conmigo mismo.



El proceso de 21 días



Durante esos minutos, me sentaré cómodamente y me sentiré. Iré en busca de mi presencia. Una vez la sienta me abriré a descubrir cuáles son las cualidades más profundas y esenciales de mi corazón (centro esencial de mi consciencia).

Mi corazón emana sólo amor. Amor no es una sensación o una emoción. Amor es pura consciencia de plenitud pudiéndose manifestar en todas sus facetas humanas: Compasión, Paz, Felicidad, Unidad, Presencia, Atemporalidad, Libertad, Servicio... Hoy miraré en mi corazón con la intención de descubrir de qué "forma" mi corazón se expresa en esencia en este mundo, en mi vida.

Una vez reconocida paso a sentirla con toda mi atención y mentalmente afirmo:

"La (el) _____ es mi propósito en la Vida"

Ejemplo: "El Servicio es mi propósito en la Vida"

Y permanezco unos instantes sintiendo mi afirmación. De esta forma mi enfoque mental se une al enfoque imperturbable de mi corazón. Observo que mi propósito no es el establecimiento de una meta futura, sino más bien el acceso a mi función presente que emana en mi corazón y que toma forma en este instante a través de mi mente.

Una vez hecha esta práctica, dedico el resto del día a pararme cada vez que recuerde mi propósito y simplemente lo respiro.

Al finalizar el día:

Me siento cómodamente y paso a sentir mi propósito establecido. Observo cómo me siento. Luego afirmo:

Mi mente y mi corazón son uno.

No soy mis pensamientos.

No soy mis sensaciones ni mis emociones.

No soy mi cuerpo"

Soy Consciencia.



El proceso de 21 días



Nota: Al despertar mañana, justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí.

DÍA 21 Confianza, Aceptación y Agradecimiento.

Al despertar por la mañana:

Justo al ser consciente de que estoy despierto, inspiraré y me uniré a esa parte interna mía. Me uniré a la Vida que hoy se va a expresar a través de mí. (esto lo haremos cada mañana al despertar hasta el final del proceso).

Buenos días:

Hoy empiezo el día dedicando unos segundos a afirmar lo siguiente:

“Hoy voy a dedicar mi día a sentir mi confianza, mi aceptación y mi agradecimiento”

Confianza, aceptación y agradecimiento, los tres pilares de la consciencia en Paz.

Una vez he descubierto que mi vida es consciencia en la que aprendo a descubrirme, sé que puedo confiar en ella plenamente. No hay ningún acontecimiento en mi vida que esté ocurriendo para fastidiarme o para ponerme a prueba. No hay motivos por los que temer a la vida. La vida sólo ocurre y yo, como persona, soy una de sus “formas de vida”.

Es por esto que puedo abrirme a aceptarla en todas sus expresiones sin resistirme o desconfiar.

Es por ello que puedo abrirme a sentirme agradecido por vivir, por ser.

Cada vez que recuerde el pensamiento semilla, me paro un instante para sentir todo aquello que me rodea en este momento. Esta es la forma que toma mi vida ahora. Me abro a sentir mi confianza por ello. Una vez la sienta, me abro a sentir mi aceptación. Una vez la sienta paso a sentir mi agradecimiento por ella.





El proceso de 21 días

No me preocupo si me cuesta conectar o sentir los tres pilares. Atiendo, sólo, a mi voluntad de querer conectar con esos tres pilares.

Al finalizar el día:

Me siento en un lugar cómodo y siento mi confianza, mi aceptación y mi agradecimiento por mi vida. Una vez sentido, me abro a descubrir la Paz que emana de mi confianza, de mi aceptación y de mi agradecimiento.

Sin más, me abro a sentir mi propósito (establecido desde ayer) y afirmo.

**“Decido vivir mi (propósito) guiado por mi confianza,
mi aceptación y mi agradecimiento por mi vida
y la de todos los seres vivos”.**

