ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA


Trabajo Fin de Grado

# Diseño E Implementación De Una Aplicación Web Para La Gestión De CVs

AUTOR: Nuria Martínez Ortuño

DIRECTOR: José Fernando Cerdán Cartagena

CODIRECTOR: Andrés Cabrera Lozoya

Septiembre / 2014

# Agradecimientos

A mis padres, por todo el apoyo y el ánimo recibido durante toda mi vida, por animarme a hacer lo que me gusta, por recordarme siempre que no se consigue nada sin esfuerzo pero sobre todo por haber tenido tanta paciencia conmigo.

A mi hermano, por ser mi hermano y animarme a seguir de una manera indirecta.

A mi abuela Tana, por haberme cuidado desde pequeña y ser mi segunda madre.

A Emilio, por haber sido un pilar fundamental durante estos cuatro años, por haberme ayudado siempre y animarme a no abandonar nunca, como él dice “todo lo que se empieza, se termina”.

A mi director Fernando Cerdán, por haberme dado la oportunidad de realizar este proyecto y por todos los consejos dados durante las clases.

A mi codirector Andrés Cabrera, por haber estado siempre detrás mía preguntándome cómo iba, por las reuniones semanales, por ir poniéndome metas y ayudarme a conseguirlas, pero sobre todo por esos “Ánimo compañera”.

A la comunidad de Laravel “LaravelES” y a mi recién encontrado primo, por toda la ayuda recibida.

En general, a todos mis amigos, compañeros y profesores que durante mi paso por la universidad han contribuido a que sea un poco más feliz y un poco más sabia.

# Índice

# Índice de ilustraciones

# 1. Introducción

El proyecto nace como una idea innovadora que trata de acercar el mundo de las tecnologías Web al mundo académico a través de una aplicación para la gestión de CVs.

En la actualidad, aunque existen multitud de formatos de CV estandarizados, no existe ninguna herramienta que permita separar los datos de su presentación según un formato u otro, y que permita luego su generación dinámica. Así, se da la circunstancia de que los usuarios han de mantener actualizado su CV de forma manual en tantos documentos como formatos mantengan.

Por otra parte, el estado de la técnica, en general, y de las tecnologías móviles, en particular, permite estar permanentemente conectados a la red en cualquier lugar y a cada momento, permitiendo un servicio como el descrito de forma ubicua. De hecho, no existe en la actualidad ningún cuello de botella (debido a hardware, software o incluso ancho de banda de las comunicaciones) que limite a la hora de desarrollar nuevos servicios móviles basados en TIC. Así, esta "revolución móvil" deja un escenario extremadamente propicio para la creación de nuevas aplicaciones y servicios de valor añadido que hagan uso de esa conectividad.

## 1.1. Descripción del problema

El proyecto surge debido a la problemática que sufren algunas personas a la hora de hacer su curriculum, tanto en el ámbito académico como en el profesional. La dificultad radica en mantener actualizado el CV de forma manual, ya que cada cierto tiempo surgen nuevas tareas que añadir al curriculum, como experiencias profesionales, cursos, idiomas, etc.

A esta se añade el inconveniente de que no existe un formato estándar para la elaboración de CV, cada organismo oficial tiene un modelo específico, por lo que se tiene que diseñar un curriculum diferente en función del organismo. Para la realización del CV para un organismo en particular se debe seleccionar la plantilla adecuada y rellenar todos los campos manualmente, tarea que se tiene que repetir cada vez que se quiere hacer un CV para un organismo diferente, con el trabajo que ello conlleva.

El trabajo desarrollado en este TFG y presentado finalmente en este documento trata de arreglar este problema, intentando dar solución a los problemas a los que se enfrentan las personas a la hora de hacer y actualizar su CV.

## 1.2. Estado del arte

La computación en la nube o Cloud Computing es un modelo de acceso a los sistemas informáticos, en el que los datos y las aplicaciones están hospedados en Internet y en centros de cómputo remotos, de tal modo que pueden ser utilizados desde cualquier punto que tenga conexión a la red mundial. La computación en la nube permite que los consumidores y las empresas gestionen archivos y utilicen los programas, sin necesidad de instalarlos localmente en sus ordenadores. Esta tecnología ofrece un uso mucho más eficiente de los recursos, tales como almacenamiento, memoria, procesamiento y ancho de banda.

El término "nube" se utiliza como una metáfora de Internet, se utiliza una nube para representar Internet en los diagramas de red, como una abstracción de la infraestructura que representa.

Un ejemplo sencillo de computación en la nube es el sistema de documentos y aplicaciones electrónicas Google Docs / Google Apps. Para su uso no es necesario comprar ni instalar software o disponer de un servidor, basta con una conexión de banda ancha para poder utilizar cualquiera de sus servicios. El servidor y el software de gestión se encuentran en la nube (Internet) y son directamente administrados por el proveedor de servicios. De esta manera, es mucho más simple para el consumidor disfrutar de los beneficios. En otras palabras: la tecnología de la información se convierte en un servicio, que se consume de la misma manera que se consume la electricidad o el agua.

El uso tradicional del PC no ha cambiado, se instala un sistema operativo, se buscan aplicaciones y se instalan en el equipo para poder realizar diferentes tareas. La idea del almacenamiento en la nube es que, ya no se necesita que estas aplicaciones estén en el ordenador, pues se puede acceder a ellas desde Internet.

Lo mejor de todo es que la nube no fue creada para personas expertas en tecnología, sino para el usuario final que quiere solucionar las cosas de manera rápida y simple. Por ese motivo la mayoría de los servicios que hacen uso de esta tecnología son de lo más fáciles de usar.

Este paradigma también permite aprovechar mejor los recursos del PC, por ejemplo, Picasa, que es un servicio para alojar imágenes, permite editar las capturas a través de Internet (darle brillo, rotarlas, cortarlas, etc.), sin necesidad de tener ningún software alojado en el ordenador. De esta forma, el esfuerzo de procesamiento se aloja en los servidores de Google y no en el PC.

El uso de la nube se divide según usuarios comunes o empresas.

Para los **usuarios comunes**, el Cloud Computing probablemente sea cosa ya de todos los días. El uso de aplicaciones como Hotmail, Gmail o cualquier otro servicio de correo electrónico ya supone un uso de Cloud Computing, ya que se están almacenando datos en la nube.

Y es que poco a poco, el navegador se está convirtiendo en una especie de sistema operativo, dada la cantidad de usos que se le da. En el pasado, simplemente navegaba por la Web, y si se quería enviar un correo se tenía que abrir otra aplicación aparte. Hoy en día el navegador es capaz de hacer todo esto sin tener que abrir otras aplicaciones. Muchas de las aplicaciones que en el pasado eran aplicaciones nativas de Windows, Mac OS, etc, han dado el salto a la nube, donde son independientes del sistema que está accediendo a ellas, de manera que todo es portátil.

En el pasado, guardar un documento de Office, significaba que éste terminaba residiendo en el PC, porque se guardaba en la carpeta "Mis Documentos". Con el Cloud Computing, y cogiendo el servicio más común, Google Docs, cualquier documento que se cree reside en Internet. Esto no sólo significa que se puede acceder al archivo desde cualquier PC (siempre y cuando se tenga conexión a internet), sino también que se puede ver o editar desde el móvil, pc, tablet,.... Y no sólo eso: basta con darle acceso a otros usuarios, y éstos podrán ver y editar el documento al mismo tiempo. Además, si algún usuario de los que comparte el archivo tiene una sugerencia, puede hacerlo dejando una nota.

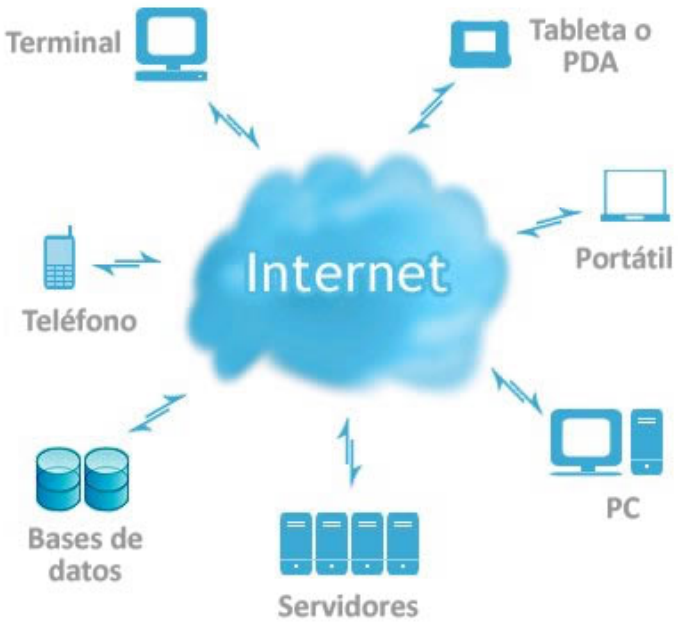



**Ilustración 1: Elementos que intervienen en la nube.**

Para las ***empresas***, el impacto de Cloud Computing es aún mayor, ya que una empresa no solo tiene que comprar hardware específico y que cumpla ciertas características, sino que además, tienen que comprar software y licencias para cada uno de los PCs que utilicen. Aparte de esto, se necesitará seguramente personal especializado que se encargue de mantener y actualizar todo el hardware y software.

Con la computación en nube, o Cloud Computing, todo está centralizado en la Web. Esto significa que se puede tener una sola aplicación, corriendo en un servidor, al cual todos los trabajadores tienen acceso, necesitando una sola licencia, o tal vez alquilándola solo por el tiempo que se necesite.

Una copia con licencia de un procesador de texto, por ejemplo, debe residir en el PC para crear el documento. El programa no tiene valor mientras el PC esté apagado por la noche. O peor aún, el mismo empleado puede que necesite otra licencia para escribir o editar un documento en su PC en casa, pues tiene que nuevamente instalarlo en dicho PC. Al implementar una solución de "Cloud Computing", estos mismos documentos, e incluso toda la aplicación, estaría disponible para este trabajador en su PC en casa, con tan sólo abrir el navegador, ingresar sus datos de usuario, y empezar a editar el documento.

La idea de la computación en la nube también reduce bastante los costos de implementación en hardware. Para software especializado, ya no es necesario tener el PC más rápido con grandes cantidades de RAM; bastaría con tener un PC relativamente económica que sirva de terminal, capaz de simplemente correr el software necesario para interactuar remotamente con la Web, que se encargaría de todo el trabajo de procesamiento.

*La computación en la nube se puede aplicar en casi cualquier entorno*: desde el pequeño comerciante que necesita un sitio Web de comercio electrónico, de forma rápida y barata, hasta las grandes empresas, que desean disminuir los gastos y evitar las dificultades de administrar un centro de cómputo complejo.

Los servicios de la computación en la nube cubren desde aplicaciones individuales de negocios, como el gestor de clientes y contactos, software contable y financiero o programas ofimáticos, hasta la externalización informática de alto rendimiento para complejos diseños en 3D, películas de cine o investigación científica.

El cliente puede en todo momento decidir qué aplicaciones usar y elegir entre aquellas que son gratuitas y las que no lo son. En el caso de las aplicaciones de pago, el costo irá en función de diversas variables, como el servicio contratado, el tiempo que se ha usado ese servicio, el volumen de tráfico de datos utilizado, el número de usuarios, etc.

El Cloud Computing tiene muchas **ventajas**, algunas de ellas son:

- **Acceso desde cualquier lugar y en cualquier momento**: El sistema en la nube está diseñado para ser utilizado a distancia, así que el usuario tendrá acceso a la mayoría de los sistemas en cualquier lugar donde se encuentre.
- **Bajo coste**: El usuario no tiene que invertir en comprar y mantener servidores y software, con el servicio en la nube se alquila un determinado servicio y se paga por lo que se consume, o incluso algunos servicios ofrecen un límite de almacenamiento, como en el caso de Dropbox que ofrece 2GB de almacenamiento gratuito, y si se quiere ampliar este espacio se tiene la posibilidad de pagar por espacio extra.
- **El prestador del servicio se encarga de todo**: El usuario no se tiene que encargar de nada, únicamente inicia sesión y accede a sus datos. El prestador del servicio se encarga de la asignación de recursos, mantenimiento de los datos, mejoras y actualizaciones, etc.
- **Rapidez:** la nube permite acceder a las aplicaciones y servicios sin tener que descargarlos, así las empresas ganan velocidad en la implantación de los proyectos.

El almacenamiento en la nube también tiene algunas **desventajas** como son:

- **Disponibilidad:** La disponibilidad de las aplicaciones están sujetas a la disponibilidad del acceso a internet.
- **Seguridad**: Los datos de un negocio no residen en las instalaciones de la empresa, lo que podría provocar una alta vulnerabilidad para la sustracción o robo de la información.
- **Escalabilidad a largo plazo:** A medida que más usuarios empiecen a compartir la infraestructura de la nube, la sobrecarga en los servidores de los proveedores aumentará, si la empresa no posee un esquema de crecimiento óptimo puede llevar a degradaciones en el servicio o altos niveles de jitter.
- **Privacidad:** La información queda expuesta a terceros que pueden copiarla o acceder a ella.

Así como, la aplicación Google Drive está disponible para el usuario en cualquier momento, se pretende que la aplicación de CVs también lo esté. Por lo que una vez evaluadas tanto las ventajas como las desventajas del Cloud Computing, se llega a la conclusión de que una aplicación alojada en la nube tendrá un mayor número de usuarios y será más fácil acceder a través de Internet, ya que lo que se pretende es poder crear o actualizar el CV desde cualquier lugar y en cualquier momento.

# 2. Solución propuesta

Desarrollar una aplicación Web que se adapte a todo tipo de dispositivos, que permita la creación y modificación del CV de los usuarios. Para esto se desarrollará una aplicación que dispondrá de diferentes formularios para la creación del CV. Los formularios se clasificarán por categorías, de manera que el usuario sepa utilizar la aplicación de una manera sencilla e intuitiva. Cada uno de los formularios constará de los campos básicos de esa categoría, como por ejemplo para la categoría identificación, se añadirán los campos nombre, primer apellido, segundo apellido, sexo, etc.

De esta manera el usuario irá añadiendo los campos que desee, y además se incorporará una funcionalidad extra, para que el usuario también pueda añadir los campos extra no presentes que pudiera necesitar. A esta funcionalidad se la denominará en lo sucesivo “campos dinámicos”. Así, el usuario podrá completar su CV de forma personalizada añadiendo conjuntos de datos como el siguiente:

- Nombre del campo.
- Valor del campo.
- Nombre del formulario en el que incluirlo.
- Posición del nuevo campo en relación a los existentes.

La aplicación es un software alojado en la nube que tendrá dos objetivos básicos:

1. Mantener una base de datos con todos los logros susceptibles de aparecer en el CV de los usuarios, fácilmente editable y ampliable de forma dinámica (pudiendo añadir nuevos campos, secciones, etc.).
2. Ofrecer la posibilidad de acomodar esos datos en una serie de plantillas (correspondientes, a su vez, con los formatos de CV estandarizados más comunes) para generar en tiempo real documentos descargables en Word adaptados a cada estándar.

Como se puede ver en la sección de descripción del problema los mayores inconvenientes son poder elegir una plantilla diferente sin tener que rehacer el curriculum y tener el curriculum actualizado, como se ha comentado en esta sección, todos los datos del cliente estarán guardados en una base de datos, esto soluciona el primer problema, ya que de esta manera el cliente solo tendrá que acceder a la aplicación, y si ha rellenado todos los datos que estime oportunos, únicamente tendrá que elegir una plantilla y descargar su curriculum. El segundo problema también se soluciona con la base de datos, ya que el cliente tendrá sus datos, y cuando acceda a la aplicación verá los datos que tenía guardados anteriormente, de manera que podrá modificarlos para tener actualizado su CV.

Además a todo esto se le añade la facilidad para el usuario de tener su CV en la nube, de manera que en cualquier momento o situación podrá modificarlo a través de cualquier dispositivo que tenga cercano. Esto hace que la creación de un CV sea una tarea sencilla, ya que ayuda al usuario a crearlo, esto conlleva a que ahora, la realización de un CV, se convierta en una tarea agradable para el usuario, incluso si es un usuario que nunca ha realizado un CV, ya que sabrá los campos que hacen falta rellenar para un CV estándar.

Con todo esto se consigue que el usuario pueda solucionar los problemas que tenía anteriormente, por los cuales, la creación de un CV se convertía en una tarea difícil y aburrida que hacía que el usuario no mantuviera su CV actualizado.

Finalmente y tras haber evaluado todas las características, ventajas y desventajas de las diferentes formas en las que se podría desarrollar este proyecto, se llega a la conclusión de que la mejor manera es la creación de una página Web de tipo responsive, que se adapta a todo tipo de dispositivos, estructurada en diferentes formularios para que el usuario rellene de una manera rápida el CV, añadiendo un formulario de tipo dinámico para la creación de campos extras y enlazada a una base de datos para poder guardar los datos de los clientes.

# 3. Descripción técnica de la solución

## 3.1. Arquitectura

La Web es el servicio más utilizado de la nube, debido a las ventajas enumeradas en el apartado 1.2. de este documento, principalmente debido a la apertura, ya que puede ser ampliada e implementada de diferentes formas sin modificar su funcionalidad. Como sistema, no impone ninguna restricción al tipo de recursos que se pueden albergar en ella, simplemente define como esos recursos pueden ser intercambiados entre ordenadores (y por tanto entre personas). Cualquiera puede añadir nuevos recursos y cualquiera puede enlazarlos.

La Web es un sistema de documentos/ recursos hiperenlazados accesibles vía internet. Como servicio, no necesita órganos de gobierno, a diferencia de Internet. Pero es necesario establecer estándares internacionales. Las tecnologías que se usan se especifican en estándares, en su mayoría publicados por el World Wide Web Consortium (W3C) y el Internet Engineering Task Force (IETF).

Normalmente, se usa la Web como sinónimo de Internet, este uso es incorrecto ya que Internet es un sistema global de redes interconectadas y que proporciona la infraestructura física y lógica para ofrecer múltiples servicios, es un servicio implementado sobre la infraestructura de Internet.

La web como sistema es un conjunto de tecnologías simples usadas para acceder a recursos vinculados entre sí. Las tecnologías que definen sirven para:

- Nombrar un recurso (URI/URL).
- Representar un recurso (HTML y otros).
- Acceder, transferir o interactuar con un recurso (HTTP).

Un recurso es cualquier información que pueda ser nombrada, como por ejemplo, un documento, una imagen, un video, etc. Los recursos se alojan en servidores. Un servidor es básicamente un computador ejecutando un programa servidor que atiende peticiones y devuelve documentos almacenados en su disco duro. El programa servidor se encarga de transformar la URI/URL en una petición adecuada y enviarla al servidor para obtener la respuesta y convertirla a un formato apropiado para el usuario.

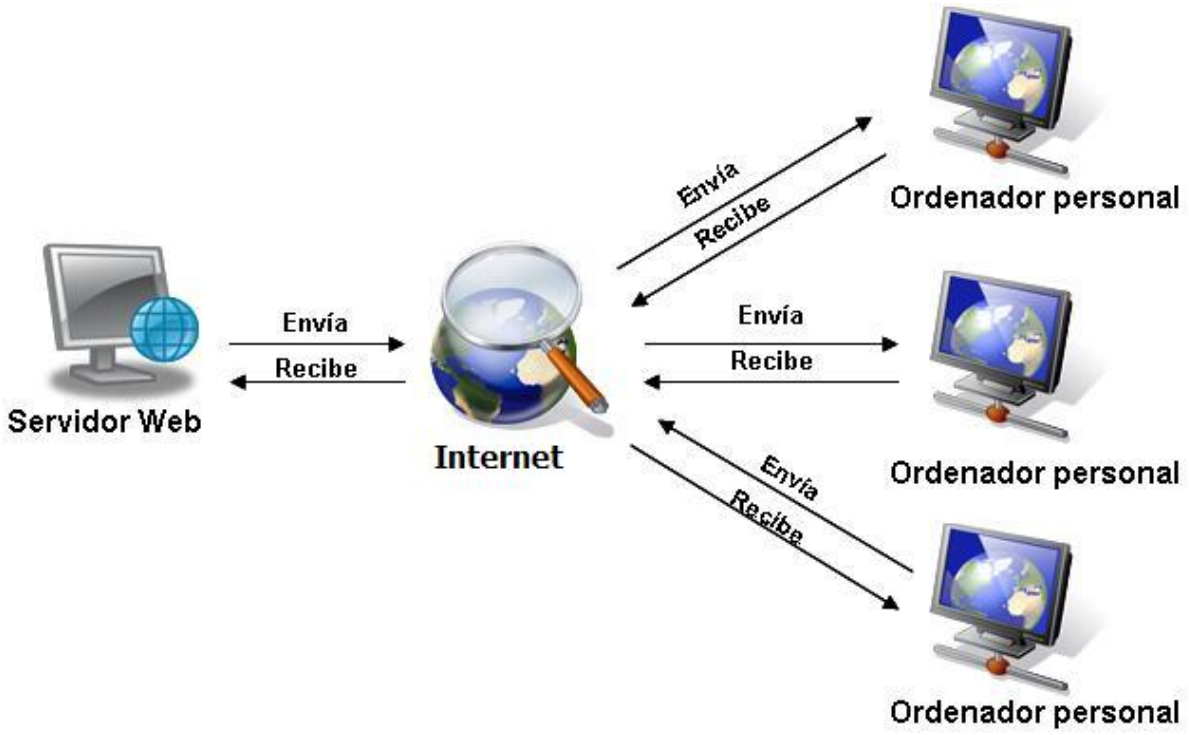



**Ilustración 2: Componentes de la Web.**

En la Web se trabaja de la siguiente manera:

- Los recursos se representan mediante el lenguaje HTML.
- Los recursos se identifican mediante URI/URL.
- Los documentos HTML se almacenan en servidores.
- El servidor ejecuta un programa servidor HTTP.
- El usuario introduce una URL en el navegador.
- El navegador genera una petición a partir de la URL.
- La petición se efectúa mediante el protocolo HTTP sobre TCP/IP.
- El servidor recibe la petición y devuelve el documento solicitado (código HTML más cabecera adicional).
- El navegador recibe los datos y los muestra por pantalla (procesa el HTML recibido).

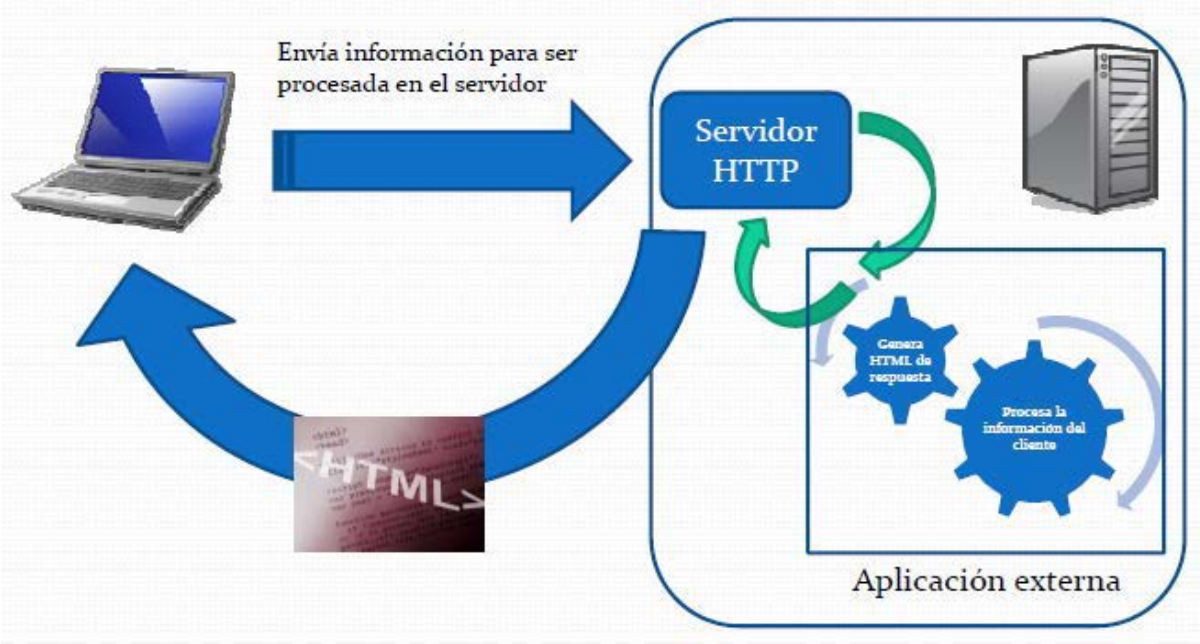



**Ilustración 3: Proceso que se produce al acceder a una página Web.**

A continuación se listan los componentes utilizados:

- **HTML** (HyperText Markup Language): Es un lenguaje de marcas mediante el que se representan los recursos. Es un estándar que sirve de referencia para la elaboración de páginas Web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página Web, como texto, imágenes, etc. Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la Web, sobre todo en lo referente a su escritura e interpretación.
- **Hiperenlace** (Hyperlink): En general, es una referencia (puntero) a datos externos. En HTML son marcas que incluyen un identificador URL que apunta a un documento externo (o a partes del propio documento) y que el usuario puede seguir.
- **Hipertexto:** Texto con hiperenlaces.
- **HTTP** (HyperText Transfer Protocol): Protocolo de transferencia de datos en la Web. Es un protocolo de nivel de aplicación, basado en texto, de petición/respuesta y sin estado.
- **URI** (Uniform Resource Identifier): Cadena de caracteres utilizada para identificar un recurso en la Web. Las URI se pueden clasificar en URL o URN. Las URL son localizadores e indican los medios para actuar sobre ellos. Las URN identifican

unívocamente un recurso pero no especifican como localizarlo o actuar sobre él.

- **Servidor HTTP:** Un programa que atiende peticiones realizadas mediante el protocolo HTTP.
- **Servidor:** Un equipo que ejecuta un servidor HTTP y almacena documentos/archivos/recursos que se pueden servir mediante HTTP.
- **Navegador:** Un programa que implementa un cliente HTTP y es capaz de procesar y representar documentos HTML y otros formatos usados como recursos en la Web.

## 3.2. Lenguajes y herramientas SW utilizados

### 3.2.1 Entorno de desarrollo

#### *Editor de texto*

El editor de texto utilizado para el desarrollo de la aplicación Web es Brackets. Brackets es un editor gratuito desarrollado por Adobe de código específico para desarrollo web front-end: HTML, CSS y JavaScript. Brackets está en desarrollo y las funcionalidades están en plena evolución. Se ha seleccionado Brackets debido a las múltiples ventajas que presenta con respecto a otros editores de texto.

Brackets tiene una **interfaz muy limpia y minimalista**. Su uso es muy intuitivo, como el de cualquier otro editor de texto y en vez de usar pestañas para mostrar los diferentes archivos, emplea un explorador de archivos en el lateral muy cómodo.

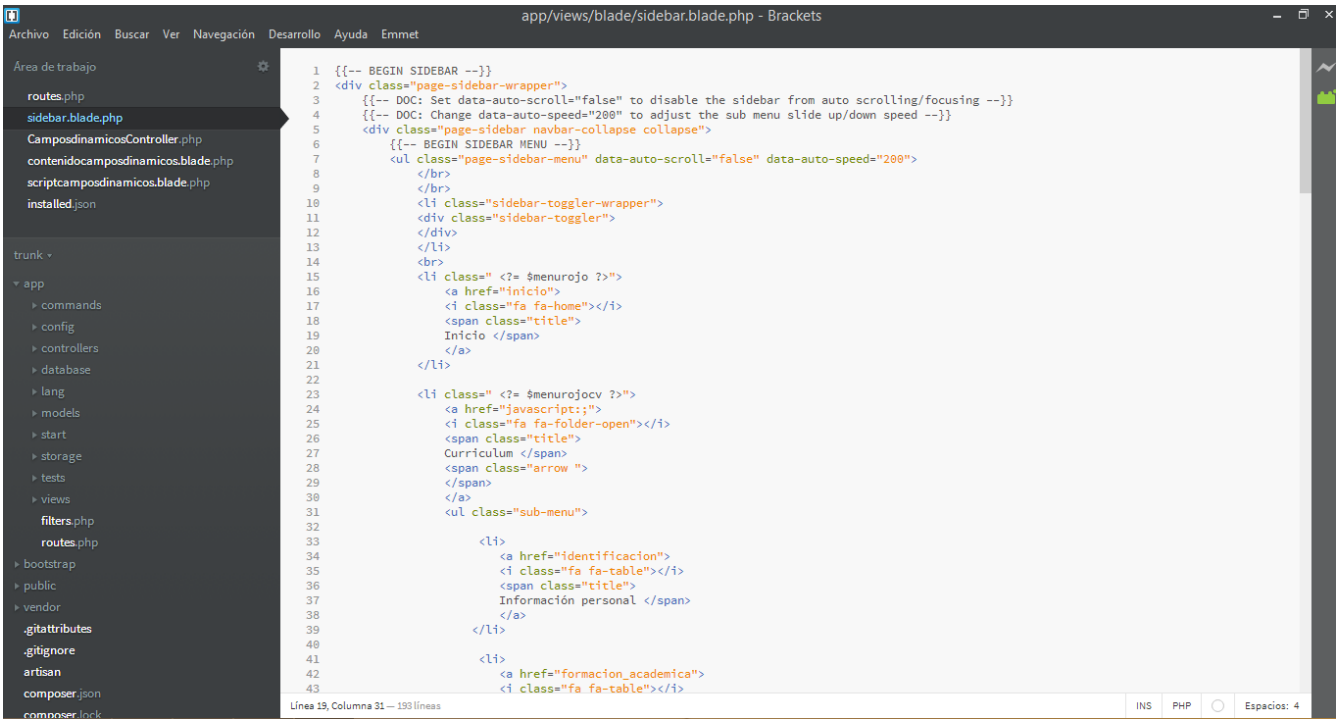



**Ilustración 4: Interfaz del editor de texto Brackets.**

## Características especiales de Brackets para el desarrollo web.

- **Vista previa en vivo.**

  Con esta característica se puede no solo visualizar el resultado del código HTML+CSS+Javascript, sino que se puede modificar en tiempo real cualquier parte del código y se verán reflejados instantáneamente los cambios en la ventana de previsualización. Por el momento la vista previa en vivo está disponible únicamente con el navegador Chrome, pero ya están trabajando para implementarla en otros navegadores.

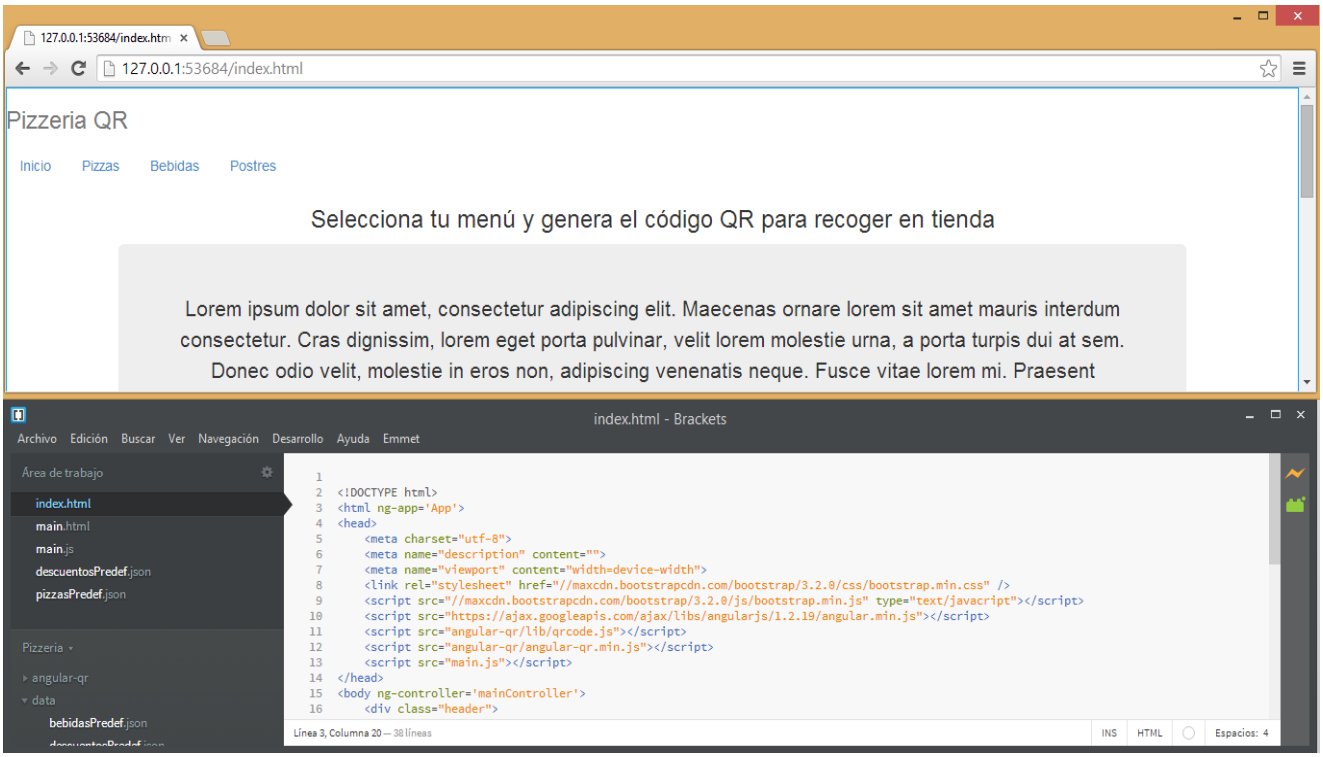



**Ilustración 5: Editor Brackets con la vista previa en vivo activada.**

- **Edición del código CSS directamente desde el código HTML.** Esta característica de Brackets consiste en permitir editar el archivo CSS que sirve de hoja de estilos a un documento HTML directamente desde el propio código HTML sin tener que abrir el archivo CSS, Brackets combina todo en la misma ventana. Para entenderlo mejor, en la siguiente imagen se muestra cómo se puede modificar el estilo de la etiqueta HTML H1 cuya hoja de estilo es main.css directamente desde la ventana con el código HTML.

  A esta opción se accede haciendo clic sobre la etiqueta HTML que se quiere modificar y pulsando la combinación Ctrl+E.

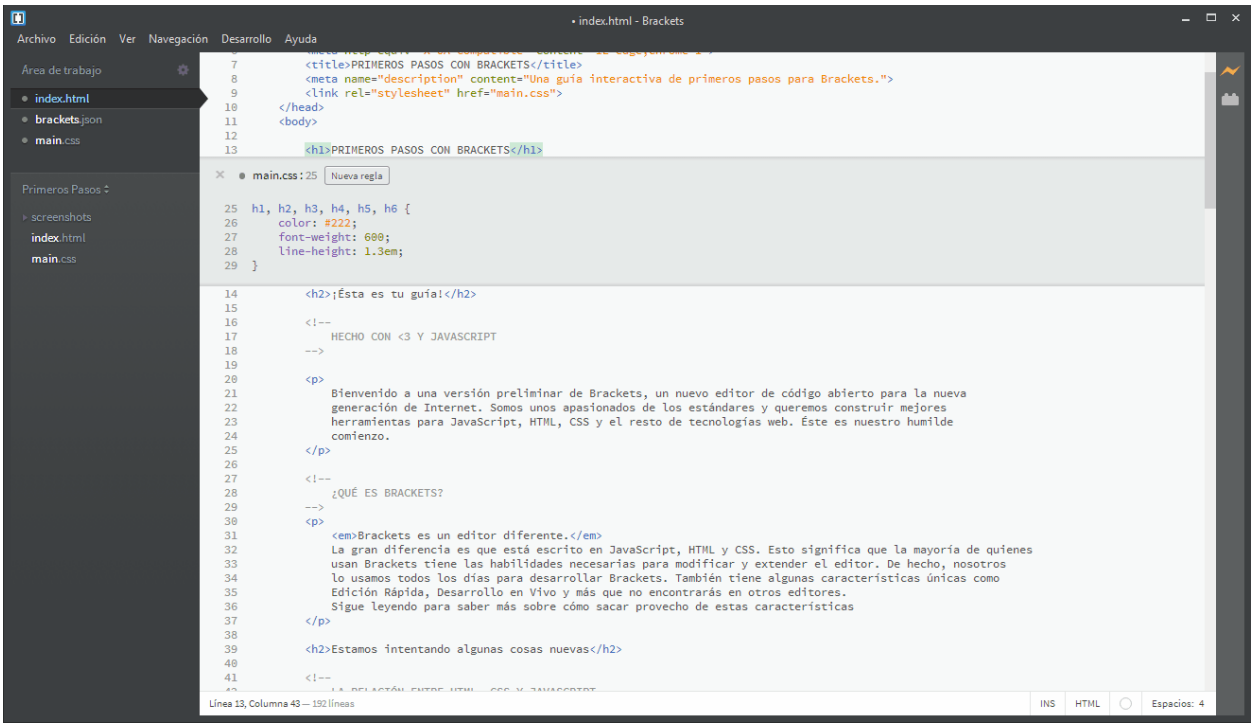



**Ilustración 6: Edición del código CSS directamente desde el código HTML.**

- **Brackets muestra las rutas de archivos.**
  Otra de las grandes ventajas de Brackets es que muestra los directorios del proyecto para localizar la ruta del recurso que se quiere emplear, ya sean imágenes, fuentes, etc. Esta opción se muestra cuando se emplea una propiedad HTML o CSS que haga uso de un archivo externo.

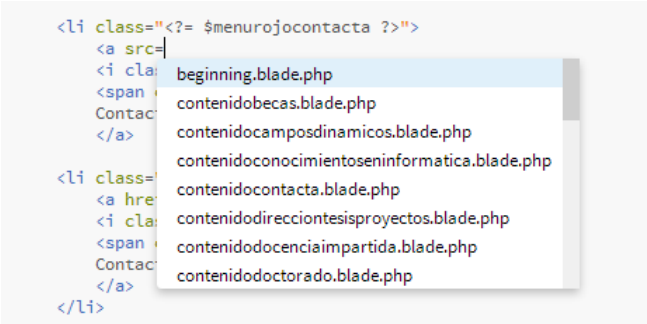



**Ilustración 7: Brackets muestra la ruta de archivos.**

- **Visualización de colores e imágenes directamente desde el código.**
  Cuando desde el código se hace referencia a una imagen o a un color, si se pasa el cursor por encima de la ruta de la imagen o por encima del valor del color, se muestra un cuadro en el que se puede visualizar la imagen o el color en cuestión.

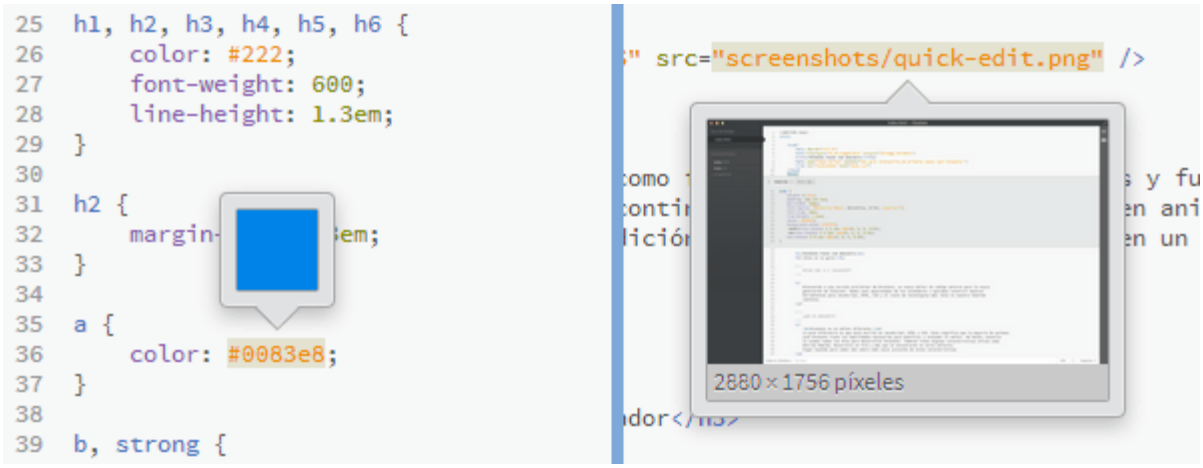



**Ilustración 8: Visualización de los colores e imágenes directamente desde el código.**

En el caso de los colores, si se hace clic en su valor y se pulsa la combinación de teclas Ctrl+E se muestra un selector de color visual muy útil.


**Ilustración 9: Selector de color al hacer clic y pulsar las combinación Ctrl+E sobre el valor del color.**

## *Servidor local*

XAMPP y WAMP son paquetes software que contiene un servidor web basados en software libre, que, de forma sencilla y requiriendo un mínimo conocimiento de las aplicaciones que utilizan, permiten publicar páginas-web desde el propio ordenador.

XAMPP es un acrónimo, sus siglas significan:

- **X:** para cualquier sistema operativo.
- **A:** Apache, es un servidor HTTP en software libre para cualquier plataforma. Tiene entre sus características bases de datos de autenticación y negociado de contenido o mensajes de error altamente configurables.
- **M:** MySQL, es un sistema de gestión de base de datos relacional, multihilo y multiusuario.
- **P:** PHP, es un lenguaje de programación interpretado, para crear webs dinámicas. Su gran versatilidad radica en que puede ser embebido dentro de código HTML.
- **P:** Perl, es un lenguaje de programación que toma características de C, de Lisp y, en menor grado, de muchos otros lenguajes.

Por su parte, las siglas de WAMP significan **W**indows (el Sistema Operativo sobre el que funciona), **A**pache, **M**ySQL y **P**HP.

Ambos incluyen phpMyAdmin, un cliente de MySQL muy popular que se ofrece en muchos sitios web de alojamiento, escrita en PHP, que permite la gestión de la base de datos MySQL (crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL en general) a través de páginas web, que proporcionan una interfaz de usuario muy orientativa.

La forma de trabajo con estos servidores consiste en crear el contenido que se quiera publicar en el ordenador (http://localhost/).

El servidor local elegido para la realización de la aplicación es XAMPP, debido a las siguientes ventajas:

- Es el entorno de desarrollo PHP más popular.
- Para Windows, Mac OS X y Linux
- Fácil instalación y configuración.
- Completamente gratuito.

## Instalación de XAMPP

XAMPP se puede descargar gratuitamente desde su página oficial.


**Ilustración 10: Página oficial de XAMPP.**

Una vez instalado XAMPP se pueden ver las posibles opciones de configuración y administración de la herramienta y sus módulos instalados.

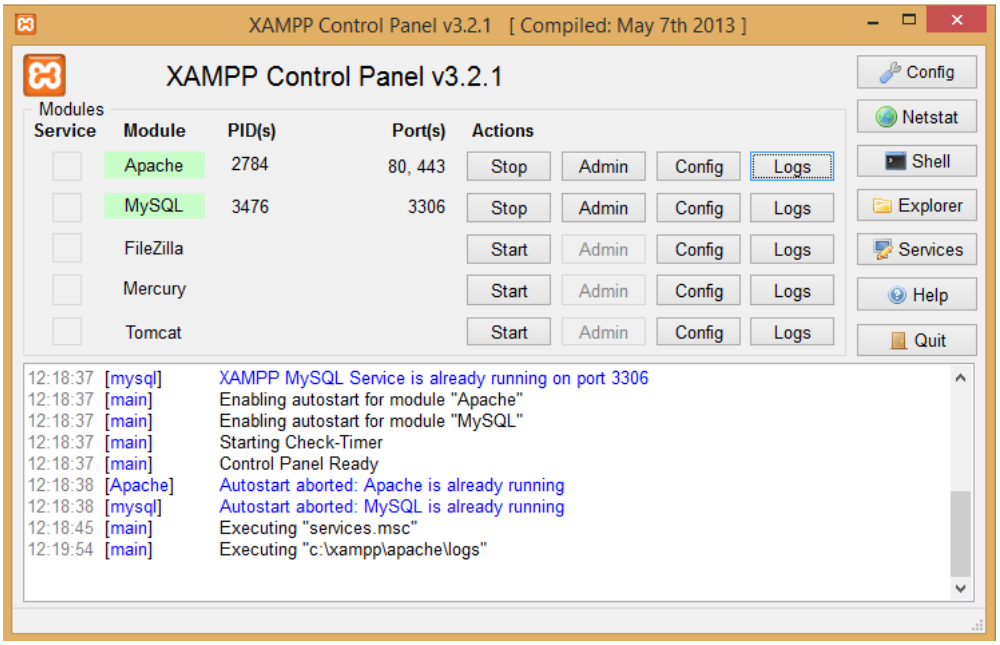



**Ilustración 11: Panel de control de XAMPP.**

En este panel de control se pueden ver todos los módulos instalados. Para cada módulo se puede parar su servicio (Stop), arrancarlo (Start), ver su estado, marcarlo como servicio y entrar en su panel de administración (Admin).

Para probar que la instalación de XAMPP fue exitosa basta con poner en el navegador "http://localhost"o "http://127.0.0.1"y aparecerá la aplicación de administración web. En ella hay una sección de administración web de XAMPP, una sección de interesantes demos y otra con herramientas incluidas en el paquete como phpMyAdmin, FileZilla FTP, Webalizer, etc.

### 3.2.2. Lado del cliente

A continuación se explican los lenguajes y tecnologías web utilizadas para el desarrollo de la aplicación de gestión de CVs del TFG.

#### *HTML*

HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Las siglas corresponden a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto, que podría ser traducido como Lenguaje de Formato de Documentos para Hipertexto.

Se trata de un formato abierto que surgió a partir de las etiquetas SGML (Standard Generalized Markup Language). Concepto traducido generalmente como “Estándar de Lenguaje de Marcado Generalizado” y que se entiende como un sistema que permite ordenar y etiquetar diversos documentos dentro de una lista.

HTML está formado por etiquetas que permiten interconectar diversos conceptos y formatos, es un lenguaje de marcas. Un lenguaje de marcas es un vocabulario (conjunto de marcas) que permite anotar los datos de un documento.

Los lenguajes de marcas permiten estructurar los contenidos de un documento, dotándolos de contexto o significado.

Un lenguaje de marcas indica al navegador como tiene que mostrar el contenido, es por esto que HTML separa el "contenido" (palabras, imágenes, audio, video, etc.) de la "presentación" (la definición del tipo de contenido y las instrucciones de cómo esos contenidos tienen que mostrarse).

HTML emplea un conjunto de elementos predefinidos que permiten identificar los distintos tipos de elementos. Estos elementos contienen una o más etiquetas que contienen o expresan el contenido. Estas etiquetas suelen ir encapsuladas entre los símbolos <>, y las etiquetas de cierre (que indican el final de un determinado contenido) están precedidas por una barra /.

Por ejemplo, el elemento  consiste en una etiqueta de inicio " " y una de cierre " ".

```
{{-- BEGIN FORM--}}
{{ Form::open(array('url' => 'becas')) }}

    <div class="form-body" class="horizontal-form">
        <div class="row">
            <div class="col-md-12">
                <div class="form-group">
                    <label class="control-label">Beca</label>
                    <input type="text" id="beca" class="form-control" name="beca">
                    <span class="help-block">
                    Introduzca el nombre de la beca. </span>
                </div>
            </div>
        </div>
```

**Ilustración 12: Ejemplo de uso de etiquetas HTML.**

Cuando este contenido se muestra en una página web, mediante un navegador, aparece así:

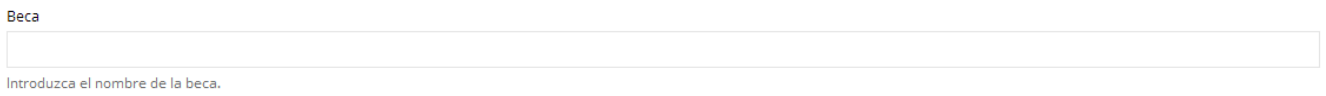



**Ilustración 13: Ejemplo de como se muestra en el navegador.**

El navegador emplea las etiquetas como guías para saber cómo debe ser mostrado el contenido que hay dentro de dichas etiquetas.

Los elementos que contienen contenidos, normalmente suelen contener también otros elementos. Por ejemplo, el elemento <input> puede estar dentro del elemento , como se muestra en la Ilustración 12.

Algunos elementos no contienen otros elementos. Como es el caso de la etiqueta imagen ("<img>") que simplemente especifica el nombre del archivo que contiene la imagen como atributo:

```
<img alt="" class="img-thumbnail" src="{{ url('imgs/'.Auth::user()->Fotografia) }}" height="200px" width="200px" />
```

**Ilustración 14: Ejemplo de uso de la etiqueta img.**

En ocasiones, suele ponerse una barra "/" al final de la etiqueta, justo antes del cierre de la misma ">" para indicar el final de la misma "/>". Aunque se trata de algo opcional en HTML.

Los documentos HTML están escritos en texto plano. Pueden ser escritos mediante editores de texto capaces de guardar contenido de texto plano (aunque la mayor parte de los creadores de código HTML prefieren editores especializados que resaltan las partes de código propias de la sintaxis del HTML y muestran el DOM). El DOM es la estructura de objetos que genera el navegador cuando se carga un documento, se puede alterar mediante Javascript para cambiar dinámicamente los contenidos y aspecto de la página. Los nombres de las etiquetas, pueden escribirse en mayúsculas o en minúsculas. Aunque, el W3C (Asociación Global que vela por mantener los estándares HTML) recomiendan usar minúsculas.

Las etiquetas de comienzo y final de un elemento deben estar adecuadamente anidadas, esto significa que las etiquetas de cierre deben escribirse en el orden inverso al de las etiquetas de inicio. La regla del anidamiento de etiquetas tiene que cumplirse de forma escrupulosa para poder escribir código válido.

La etiqueta de comienzo puede contener información adicional, tal y como puede verse en el siguiente ejemplo. Dicha información es lo que se conoce como atributos. Los atributos suelen consistir en dos partes:

- Un atributo nombre (name).
- Un atributo valor (value).

Algunos atributos sólo pueden tener un único valor. Son atributos Booleanos y pueden ser incluidos para especificar el nombre del atributo, o dejar su valor vacío.

```
<option value="DNI" selected>DNI</option>
```

**Ilustración 15: Ejemplo de uso de un atributo.**

Además de las etiquetas y el contenido, un documento de HTML debe contener una declaración doctype en la primera línea. En el HTML actual esto se escribe del siguiente modo:

```
1  <!DOCTYPE html>
```

**Ilustración 16: Uso de doctype.**

El doctype le dice al navegador que interprete el código HTML y CSS de acuerdo a los estándares web del W3C (Asociación Global que vela por mantener los estándares HTML) y que no trate de emular que se trata de un Internet Explorer de los 90's.

HTML tiene un mecanismo para poder introducir comentarios al código que no serán mostrados en la página cuando esta sea interpretada o leída por un navegador web. Esto suele emplearse para añadir explicaciones al código, o dejar notas para explicar a otras personas cómo trabaja el código de la página, o simplemente para dejar recordatorios para uno mismo. Los comentarios en HTML están contenidos entre los siguientes símbolos:

```
<!-- BEGIN CONTENT -->
```

**Ilustración 17: Comentarios en HTML.**

## Limitaciones de HTML

### *Falta de interactividad*

Uno de los principales problemas de HTML desde su inicio era la falta de mecanismos para interactuar con el usuario: un documento HTML es estático, no cambia ante las acciones del usuario (como mover el ratón).

Esto impide desarrollar contenidos interactivos y aplicaciones complejas para la web.

**Solución**: se añade código a los documentos que será ejecutado por el cliente (es decir, el navegador).

- Javascript, es el lenguaje "ligero" estándar y se ha convertido en parte fundamental de HTML 5.
- Otra opción es ejecutar código mediante plugins (como applets de Java).

## HTML5

La versión actual de la especificación HTML se conoce como **HTML5**.

HTML5 es la nueva versión del lenguaje de marcado que se usa para estructurar páginas web, que actualmente todavía sigue en su evolución, gracias a él con características nuevas y modificaciones que mejorará significativamente este nuevo estándar.

HTML5 es mejor, simplemente, porque es una tecnología que supera a la actual HTML, porque es lo nuevo que estandariza la W3C, porque es una nueva tecnología y como toda nueva tecnología siempre viene con cosas que van a impresionar, porque llega de la mano de CSS3, una evolución notable de las hojas de estilo que se conocían y porque revaloriza el papel de JavaScript en la Web, como el lenguaje que "sabe hablar" con las nuevas APIs que llegan con HTML5.

En resumen, HTML5 conduce a una fusión entre JavaScript como lenguaje de programación, HTML como modelo semántico y css3 que es la evolución del css como el lenguaje de los estilos, que se dedica a dar un mejor aspecto a los proyectos.

A continuación alguna de las reglas establecidas para HTML5:

- Las nuevas características debe basarse en HTML, CSS, DOM y JavaScript.
- Reducir la necesidad de plugins externos (como Flash).
- Mejor manejo de errores.
- Más marcado para reemplazar secuencias de comandos.
- HTML5 debe ser independiente del dispositivo.
- El proceso de desarrollo debe ser visible para el público.

## *CSS*

CSS son las siglas de Cascade Style Sheet que traducido significa hojas de estilo en cascada. Las hojas de estilo es una tecnología que permite controlar la apariencia de una página web. CSS describe como los elementos dispuestos en la página son presentados al usuario.

Con CSS se puede especificar estilos como el tamaño, fuentes, color, espaciado entre textos y recuadros así como el lugar donde disponer texto e imágenes en la página. El lenguaje de las Hojas de Estilo está definido en la Especificaciones CSS1, CSS2 y CSS3 del World Wide Web Consortium (W3C), es un estándar aceptado por toda la industria relacionada con la Web.

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.

```
body {
  background-color: #3d3d3d;
}
```

**Ilustración 18: Ejemplo de uso de un selector con su declaración.**

Body es el selector y lo demás es la declaración.

Los selectores son patrones de elección de reglas que se aplicarán a un elemento, funcionan como enlace entre el documento y el estilo, especificando los elementos que se van a ver afectados por esa declaración. La declaración es la parte de la regla que establece cuál será el efecto. En el ejemplo anterior, el selector body indica que todos los elementos dentro de body se verán afectados por la declaración donde se establece que la propiedad color de fondo va a tener el valor #3d3d3d (negro) para todos los elementos del documento o documentos que estén vinculados a esa hoja de estilo.

| Selector | Significado | Ejemplo |
|---|---|---|
| * | Cualquier elemento | * {font-family: Arial} |
| E | Cualquier elemento E | LI {font-family: Arial}<br>P {color: black} |
| E > F | Cualquier elemento F *hijo* de E | UL>LI {font-size: 70%}<br>OL>LI {font-size: 115%} |
| E F | Cualquier elemento F *descendiente* de E | H1 EM {color: blue} |
| E + F | F inmediatamente después de E (en el mismo nivel del árbol) | H1+ EM {color: blue} |
| E[foo="a"] | Cualquier elemento E con el atributo foo igual a "a" | IMG[src="logo.gif"] {width:100px} |

**Ilustración 19: Tabla de uso de selectores.**

Se denominan hojas en "cascada" ya que los elementos hijos heredan por defecto los valores de estilo de los padres. Las propiedades de un elemento se asignan por valores específicos, heredados o por defecto (de mayor a menor preferencia, respectivamente). Selectores más específicos sobreescriben a los más generales. Las reglas se aplican por especificidad, no por el orden en que aparezcan en el documento, aunque si se repiten, se elige la última aparición.

CSS define cómo se visualizan los elementos

- **Modelo de cajas**: cada elemento tiene asociado una caja (el elemento incluye las marca y lo que contienen) rectangular.
- Se pueden definir propiedades para cada una de las áreas.

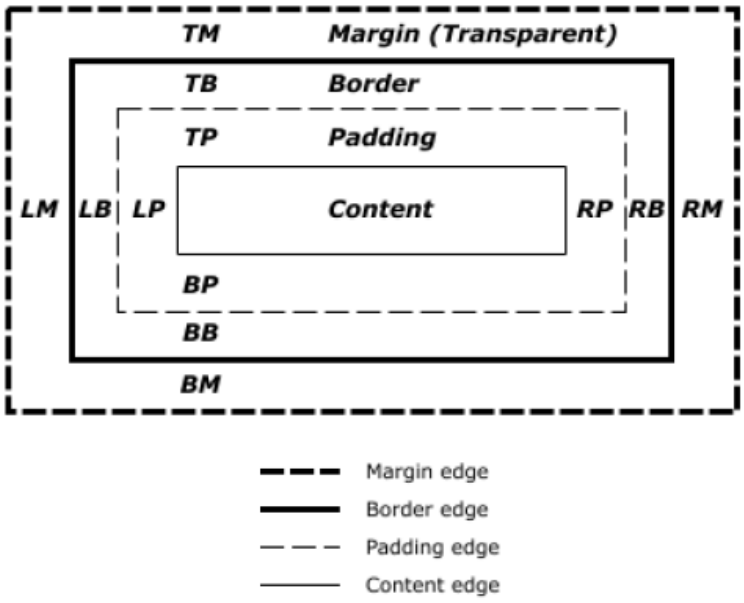



Ilustración 20: Modelo de cajas de CSS.

CSS especifica cómo se posicionan los elementos en pantalla:

- Cada elemento del árbol genera cero o más cajas de acuerdo al modelo de cajas. El posicionamiento depende de las dimensiones de la caja, relaciones entre elementos y el esquema de posicionamiento.
- El esquema de posicionamiento incluye tres modos:
  - **Flujo normal**, es el usado por defecto, posiciona las cajas consecutivamente vertical (block) u horizontalmente (inline).
  - **Flotante** (float): una caja se pone en su posición normal y luego se desplaza al máximo a la derecha o izquierda. El resto del contenido fluye alrededor de ella.
  - **Absoluto**: puede ser desplazado o fijo. La caja se saca del flujo y se sitúa en una posición, pero el resto del contenido no fluye, es decir, la caja absoluta puede superponerse a otras cajas.

Las tres formas más conocidas de dar estilo a un documento son las siguientes:

- Utilizando una hoja de estilo **externa** que estará vinculada a un documento a través del elemento <link>, el cual debe ir situado en la sección <head>.

```
<link href="{{asset('assets/admin/layout/css/themes/default.css')}}" rel="stylesheet" type="text/css" id="style_color"/>
```

Ilustración 21: Link, dentro de <head>, que enlaza al fichero css.

- Utilizando el elemento <style>, en el **interior** del documento al que se le quiere dar estilo, y que generalmente se situaría en la sección <head>. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.

```
<head>
<style type="text/css">

  body {
    padding-left: 11em;
    font-family: Georgia, "Times New Roman", serif;
    color: red;
    background-color: #d8da3d;
  }

  h1 {
    font-family: Helvetica, Geneva, Arial, sans-serif;
  }

</style>
</head>
```

**Ilustración 22: Ejemplo de uso de <style> en el interior de un documento.**

- Utilizando estilos **directamente sobre aquellos elementos que lo permiten** a través del atributo <style> dentro de <body>. Pero este tipo de definición del estilo pierde las ventajas que ofrecen las hojas de estilo al mezclarse el contenido con la presentación.

Para obtener las mayores ventajas de las CSS es preferible asociar un documento externo.

## Limitaciones

- Los selectores no pueden usarse en orden ascendente (hacia padres u otros ancestros). La razón que se ha usado para justificar esta carencia por parte de la W3C, es para proteger el rendimiento del navegador, que de otra manera, podría verse comprometido.
- Dificultad para el alineamiento vertical.
- Ausencia de expresiones de cálculo numérico para especificar valores, por ejemplo margin-left:

```
margin-left: 10% - 3em + 4px;
```

**Ilustración 23: Ejemplo de margin-left.**

- Las pseudo-clases dinámicas (como :hover) no se pueden controlar o deshabilitar desde el navegador, lo que las hace susceptibles de abuso por parte de los diseñadores en banners, o ventana emergentes.

### Ventajas

Algunas ventajas de utilizar CSS son:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Separación del contenido de la presentación.
- Optimización del ancho de banda de la conexión, pues pueden definirse los mismos estilos para muchos elementos con un sólo selector; o porque un mismo archivo CSS puede servir para una multitud de documentos.
- Mejora en la accesibilidad del documento.

## *Javascript*

A Javascript se le denomina "del lado del cliente" porque donde se ejecuta es en el navegador (cliente web), en contraposición a lenguajes como PHP que se ejecutan del "lado del servidor". En el lado que se ocupa con Javascript, el cliente, es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con todos los navegadores modernos se ha convertido en un estándar como lenguaje de programación del lado del cliente.

Con Javascript se puede crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, con que cuenta este lenguaje es el propio navegador y todos los elementos que hay dentro de una página.

Javascript nace por la falta de interactividad con el usuario en el cliente. El contenido (HTML) no se modifica dinámicamente como resultado de las acciones del usuario, a no ser que se invoque una nueva petición al servidor y se reciba nuevo contenido.

Javascript, es un lenguaje de programación interpretado, cuyo intérprete es habitualmente implementado por un navegador web. El intérprete suele ser parte del navegador: motor de Javascript (Javascript engine). Permite que los scripts enviados al cliente como parte de un documento puedan:

- Interactuar con el usuario: mediante la ejecución de instrucciones como resultado de la ocurrencia de eventos generados por el usuario.
- Alterar el contenido mostrado.
- Comunicarse asíncronamente con el servidor.
- Controlar la funcionalidad del navegador.

- Acceso a los recursos de la plataforma de ejecución del navegador: sistema de archivos, geolocalización, vídeo, audio, etc.

El código a ejecutar por el cliente (script) se incluye junto con el documento, mediante el elemento de HTML <script>. Se puede incluir de dos formas:

- **Interno**: Este elemento incluye el propio código:
  <script> document.write("Hola Mundo"); </script>
- **Externo**: incluye una URL (absoluta o relativa) con el código a ejecutar:
  <script src="miscript.js“ type= "text/javascript" ></script>

La ejecución del código se produce de tres formas posibles, en función de la presencia de los siguientes atributos:

- **Ejecución diferida**: se ejecuta al terminar de cargar el documento. Ejemplo:
  <script src="demo_defer.js" defer></script>
- **Ejecución asíncrona** (nuevo en HTML 5): se ejecuta inmediatamente, en paralelo con la carga del resto del documento. Ejemplo:
  <script src="demo_defer.js" async></script>
- **Ejecución normal**: si no aparece ninguno de los atributos anteriores, se ejecuta el código inmediatamente, antes de continuar con la carga y análisis del resto del documento (parse).

En muchos casos el código consiste en funciones que se ejecutarán cuando sean invocadas, normalmente al ocurrir ciertos eventos.

**Ejecución basada en eventos**: el código javascript se ejecuta al ocurrir un evento.

- Se asocia una función javascript (previamente declarada) a un evento, que es invocada cuando ocurre.

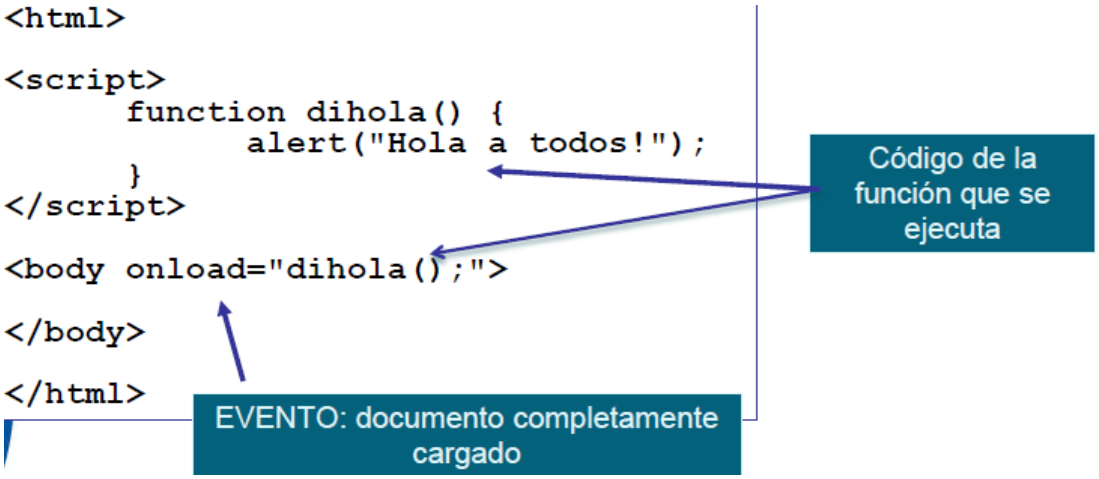



Ilustración 24: Ejemplo de uso del evento onload de Javascript.

Un **evento** es una acción o suceso detectado por el sistema que puede ser procesado.

La asignación de código a eventos se puede hacer de dos formas:

- **Atributos de evento HTML**: son atributos de ciertos elementos HTML que permiten asignar la ejecución de una función cuando ocurren.
  <h1 onclick="changetext(this)">Clic en el texto!</h1>
- **Asignación mediante código:** usando la interfaz DOM mediante javascript.
  <script>
  document.getElementById("miboton").onclick=validarFormulario;
  </script>

  En este caso se tiene que tener en cuenta que el elemento debe existir previamente, o la llamada a getElementById() devolverá null. Es decir, el documento HTML debe haberse procesado (parse) previamente y ese elemento existir.

Javascript es un lenguaje basado en prototipos (objetos que se clonan y extienden dinámicamente), con tipos dinámicos.

- Variables, declaración e inicialización: var y=1; var c; c=0;
- Tipos dinámicos: no es necesario declararlos y una variable puede albergar diferentes tipos sucesivamente.
- Orientado a objetos: todo es un objeto. var s=new String('hola');
  - Se accede a los métodos y propiedades con "." var l=s.length;
  - Un objeto también se puede declarar dinámicamente.

```
var myCar = new Object();
myCar.make = "Ford";
myCar.model = "Mustang";
myCar.year = 1969;
```

```
function Car(make, model, year) {
 this.make = make;
this.model = model;
this.year = year; }
var c = new Car("Ford", "Fiesta",1996);
```

Ilustración 25: Ejemplo del lenguaje Javascript.

```
for (var i = 0; i < 9; i++) {
        n += i;
        myfunc(n);
}
```

Enumera las propiedades de un objeto

```
for (var prop in obj) {
alert(objName + "." + prop + " = " + obj.prop + "\n";)
 }
```

```
var n = 0; var x = 0;
while (n < 3) {
        n++; x += n;
}
```

```
if (cipher_char == from_char) {
        result = result + to_char;
        x++;
} else {
        result = result + clear_char;
 }
```

```
function calc_sales(units_a, units_b, units_c) {
        return units_a*79 + units_b * 129 + units_c * 699;
 }
```

Ilustración 26: Estructuras de control, bucles y funciones de Javascript.

La sintaxis de JavaScript es muy similar a la de otros lenguajes de programación como Java y C. Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

- No se tienen en cuenta los espacios en blanco y las nuevas líneas.
- Se distinguen las mayúsculas y minúsculas.
- No se define el tipo de las variables.
- No es necesario terminar cada sentencia con el carácter de punto y coma (;).
- Se pueden incluir comentarios.

## *JQuery*

JQuery es uno de los complementos más esenciales para el desarrollo web, usado en millones de sitios en toda la web, ya que facilita mucho el desarrollo de aplicaciones enriquecidas del lado del cliente, en Javascript, compatibles con todos los navegadores.

Conviene aclarar que jQuery no es un lenguaje, sino una serie de funciones y métodos de Javascript. Por tanto, Javascript es el lenguaje y jQuery es una librería que se puede usar opcionalmente si se quiere facilitar la vida cuando se programa en Javascript. A veces se puede referir a jQuery como framework o incluso como un API de funciones, útiles en la mayoría de proyectos web.

Antes de llegar jQuery los desarrolladores estaban obligados a discriminar entre los diversos navegadores, para ejecutar aquel código Javascript que funcionaba en cada browser. Con la llegada de jQuery la principal ventaja es que ya no se necesita preocuparse sobre si el navegador del usuario es Explorer, Chrome, Firefox, etc. sino que la propia librería hará el trabajo "sucio" y ejecutará el código que sea compatible con el software del cliente que está accediendo a la web. Para ello se usan las funciones que jQuery proporciona, dentro de un grandísimo abanico de funcionalidades que además se extiende por medio de miles de plugins que ofrece la comunidad para implementar cualquier tipo de comportamiento.

JQuery es un framework Javascript, pero qué es un framework. Pues es un producto que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Framework son unas librerías de código que contienen procesos o rutinas ya listos para usar. Los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar.

Cuando un desarrollador tiene que utilizar Javascript, generalmente tiene que preocuparse por hacer scripts compatibles con varios navegadores y para ello tiene que incorporar mucho código que lo único que hace es detectar el browser del usuario, para hacer una u otra cosa dependiendo de si es Internet Explorer, Firefox, Opera, etc. JQuery es donde más puede ayudar, puesto que implementa una serie de clases (de programación orientada a objetos) que permiten programar sin preocuparse del navegador con el que el usuario está visitando la página, ya que funcionan de exacta forma en todas las plataformas más habituales.

Así pues, este framework Javascript, ofrece una infraestructura con la que se tendrá mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Por ejemplo, con jQuery se obtiene ayuda en la creación de interfaces de usuario, efectos dinámicos,

aplicaciones que hacen uso de Ajax, etc. Cuando se programa Javascript con jQuery se tiene a disposición una interfaz de programación que permitirá hacer cosas con el navegador que funcionará para todos los visitantes. Simplemente se debe conocer las librerías del framework y programar utilizando las clases, sus propiedades y métodos para la consecución de los objetivos.

Además, todas estas ventajas que sin duda son muy de agradecer, con jQuery se pueden obtener de manera gratuita, ya que el framework tiene licencia para uso en cualquier tipo de plataforma, personal o comercial. Para ello simplemente se tiene que incluir en las páginas un script Javascript que contiene el código de jQuery, que se puede descargar de la propia página web del producto, y comenzar a utilizar el framework.

El archivo del framework ocupa unos 56 KB, lo que es bastante razonable y no retrasará mucho la carga de la página. Además, el servidor lo enviará al cliente la primera vez que visite una página del sitio. En siguientes páginas el cliente ya tendrá el archivo del framework, por lo que no necesitará transferirlo y lo tomará de la caché. Con lo que la carga de la página sólo se verá afectada por el peso de este framework una vez por usuario. Las ventajas a la hora de desarrollo de las aplicaciones, así como las puertas que abre jQuery compensan extraordinariamente el peso del paquete.

### Ventajas

Es importante comentar que jQuery no es el único framework que existe en el mercado. Existen varias soluciones similares que también funcionan muy bien, que básicamente sirven para hacer lo mismo. Como es normal, cada uno de los frameworks tiene sus ventajas e inconvenientes, pero jQuery es un producto con una aceptación por parte de los programadores muy buena y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones. Además, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. Otra cosa muy interesante es la dilatada comunidad de creadores de plugins o componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar asuntos como interfaces de usuario, galerías, votaciones, efectos diversos, etc.

En general, JQuery aporta las siguientes ventajas:

- Ahorra muchas líneas de código.
- Hace transparente el soporte de la aplicación para los navegadores principales.
- Provee de un mecanismo para la captura de eventos.
- Provee un conjunto de funciones para animar el contenido de la página en forma muy sencilla.
- Integra funcionalidades para trabajar con AJAX.

El motivo del uso de JQuery en este proyecto es debido a estas ventajas, ya que JQuery provee simplicidad y potencia a la hora de desarrollar la aplicación.

Se ha usado JQuery principalmente, en el formulario de Campos dinámicos, ya que para la creación del mismo se necesitaba de una herramienta que permitiese seleccionar un campo en función de otro, es decir, en el formulario de campos dinámicos hay cuatro campos, dos de ellos son select, el primero era para seleccionar el nombre de la tabla en la cual se quería añadir el nuevo campo, y el segundo tenia los valores de los campo pertenecientes a esa tabla, en estos se necesitaba que al seleccionar uno en el primero, el segundo select se actualizará con los campos que dependían del primero, además esto debía de ocurrir sin tener que recargar la página, ya que si no se podría provocar un error al guardar un formulario con un campo que no correspondía a ese formulario.

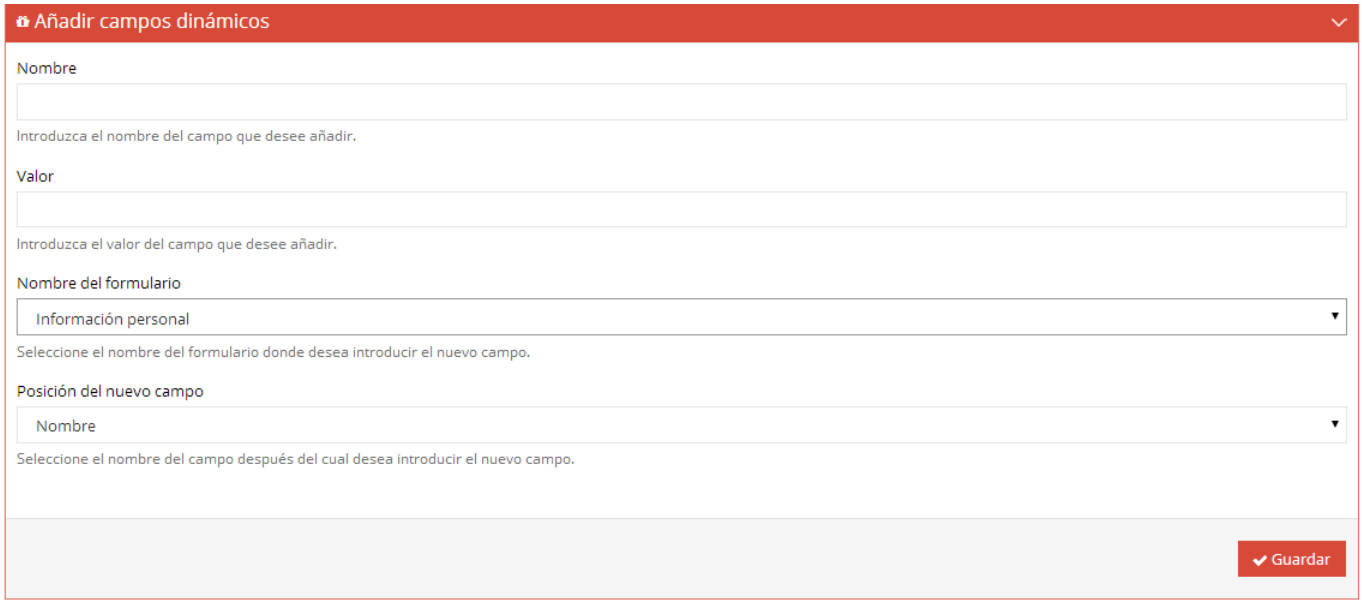



**Ilustración 27: Formulario de campos dinámicos de la aplicación en la que se muestran los cuatros campos.**

Con JQuery se ha solucionado esto de una manera sencilla. En la ilustración 27 se puede observar como al cargar la página se cargan los select con los valores por defectos, estos son, para “Nombre del formulario” se carga con “Información personal” y para “Posición del nuevo campo” se carga con “Nombre”.

Además, al pinchar sobre el select de “Nombre del formulario” se abre el desplegable con todas las opciones disponibles, como se puede ver en la ilustración 28.

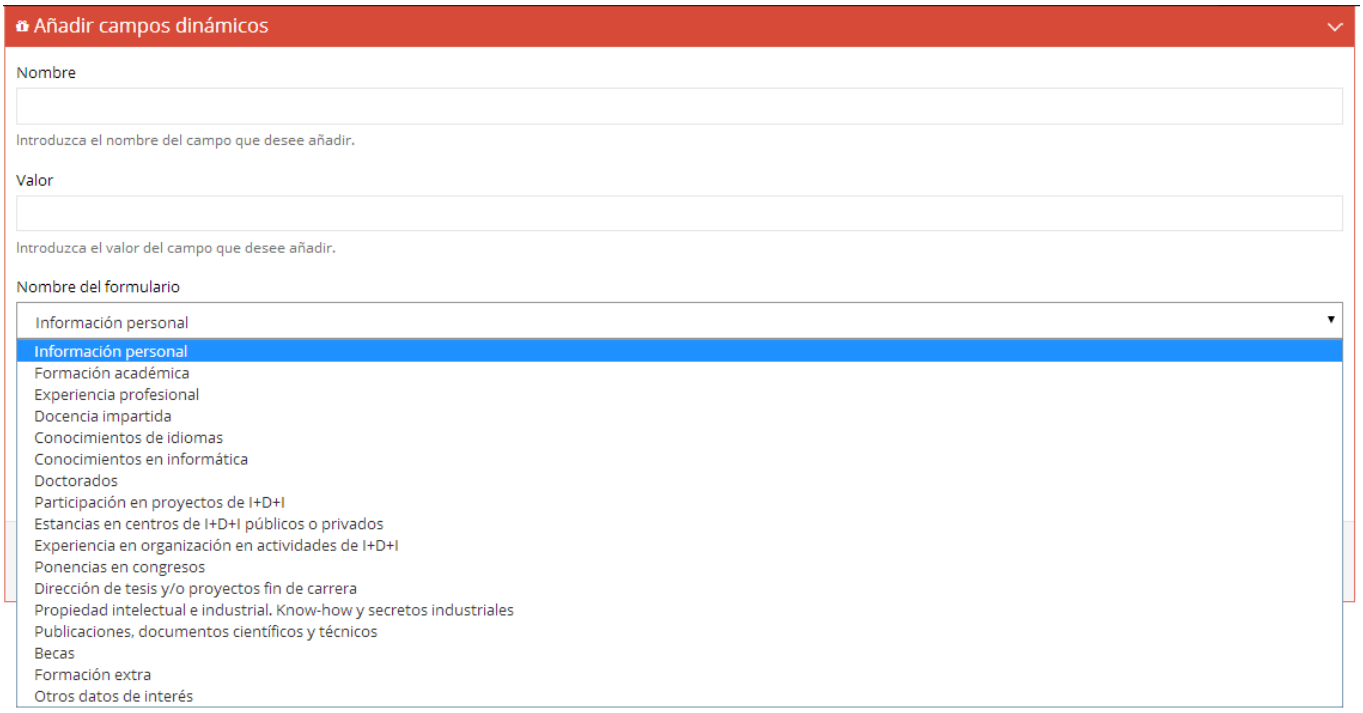



**Ilustración 28: Formulario de campos dinámicos con el select "Nombre del formulario" desplegado.**

Al seleccionar uno, por ejemplo "Formación académica", el segundo select se actualiza con los campos disponibles en esa tabla, esto se puede comprobar en la ilustración 29.

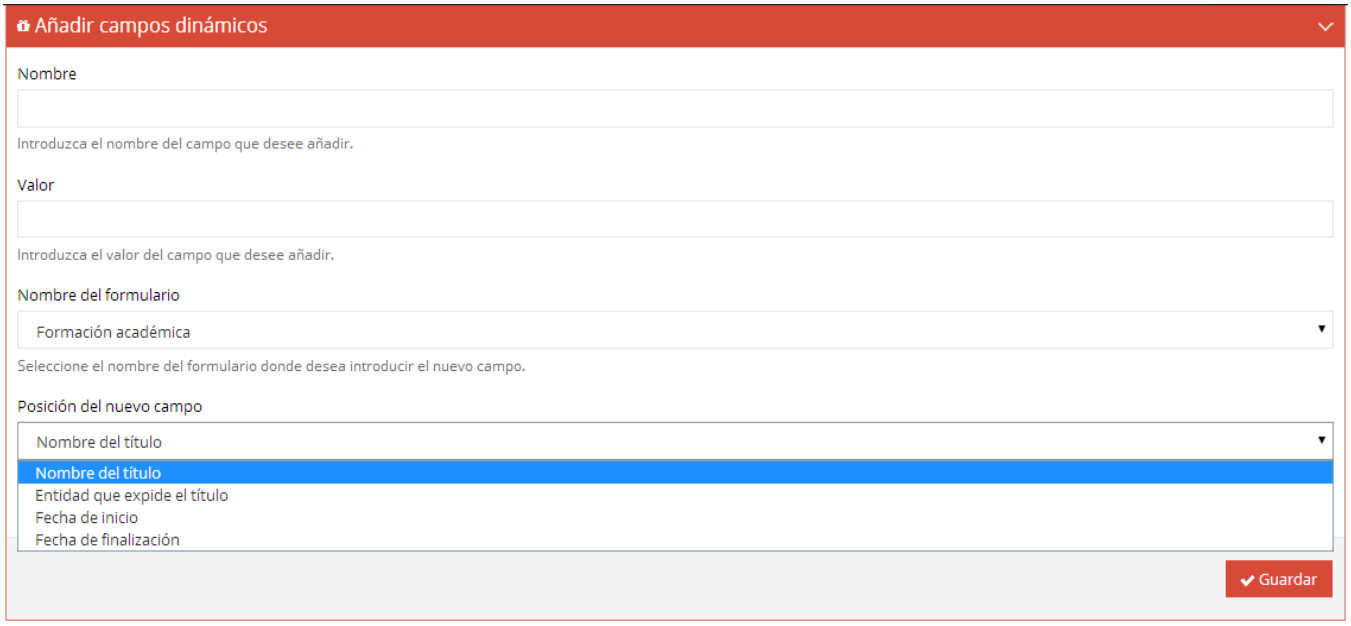


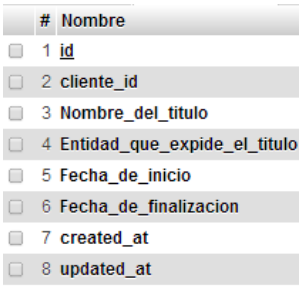

**Ilustración 29: Formulario de campos dinámicos con el select de Posición del nuevo campo desplegado.**

Esta es la solución que se ha considerado más adecuada, ya que hace que la tarea de añadir un nuevo campo al CV sea para el cliente la más sencilla. Con esta solución el cliente entiende como añadir un nuevo campo, ya que sin el uso de JQuery, la tarea de añadir un nuevo campo sería una tarea complicada. El cliente tendría que escribir correctamente el nombre del formulario, con el mismo nombre que aparece en la base de datos. Como se puede ver en la ilustración 30, los nombres de las tablas de la base de datos, son nombres "complicados", ya que por convecciones del framework utilizado (Laravel), así se tiene que hacer.

| Tabla | Acción |
|---|---|
| becas | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| campos_dinamicos | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| clientes | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| conocimientos_de_idiomas | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| direccion_tesis_proyectos | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| docencias_impartidas | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| doctorados | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| estancias_idi_centros | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| experiencias_profesionales | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| experiencia_idi_actividades | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| formaciones_academicas | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| formaciones_extras | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| informatica_conocimientos | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| know_how_secretos_industriales | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| migrations | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| otros_datos | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| participacion_idi_proyectos | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| ponencias_congresos | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| publicaciones | Examinar Estructura Buscar Insertar Vaciar Eliminar |
| subscripciones | Examinar Estructura Buscar Insertar Vaciar Eliminar |

**Ilustración 30: Tablas existentes en la base de datos.**

Además, no son solo los nombres de las tablas sino que también, los nombres de los campos de las tablas están nombrados con el mismo formato, por lo que esta tarea complicaría aún más todo el proceso de añadir un nuevo campo y haría que el cliente se pudiera equivocar al escribir el nombre de un campo.



| # | Nombre |
|---|---|
| 1 | id |
| 2 | cliente_id |
| 3 | Nombre_del_titulo |
| 4 | Entidad_que_expide_el_titulo |
| 5 | Fecha_de_inicio |
| 6 | Fecha_de_finalizacion |
| 7 | created_at |
| 8 | updated_at |

**Ilustración 31: Nombre de los campos de la tabla de Formación académica.**

Con JQuery se ha conseguido solucionar una de las tareas más importante de esta aplicación, ya que una de las cosas innovadoras de crear una aplicación para la creación de CVs era el poder añadir nuevos campos dinámicamente que se ajustaran a las necesidades del usuario de la aplicación.

## *Twitter Bootstrap*

Bootstrap es el primer framework front y back-end, desde agosto de 2011 se trata de un framework open source. Consiste en un conjunto de herramientas creadas para sitios y aplicaciones web. Contiene plantillas de diseño basadas en HTML y CSS, así como extensiones JavaScript.

Bootstrap fue desarrollado por Mark Otto y Jacob Thornton en Twitter, como framework para mejorar la consistencia al interno de las herramientas internas. Antes de Bootstrap, algunas librerías se usaban para el desarrollo de interfaces, lo que daba lugar a inconsistencias y un alto nivel de mantenimiento.

### Características

- Bootstrap es compatible con las últimas versiones de la mayoría de los navegadores.
- A partir de la versión 2.0 también cuenta con **responsive design**, es decir se ajusta dinámicamente a las características del dispositivo.
- Como proyecto open source, se encuentra disponible en GitHub y los desarrolladores pueden participar en el proyecto incluyendo sus contribuciones en la plataforma.

Bootstrap es un conjunto de diferentes proyectos:

- **960 Grid System**:
  Es una librería de CSS enfocada a la maquetación de páginas web. Se utiliza un contenedor principal de 960 pixeles. Hay dos variantes: 12 y 16 columnas, que se pueden usar de manera individual o conjunta. Con Grid System se consiguen layouts paginados de manera perfecta.
- **Less:**
  Es una técnica de CSS dinámico. LESS extiende CSS para dotarlo de un comportamiento dinámico a través de variables, mixins, operaciones y funciones.
- **Bootstrap Tutorial:**
  Contiene la lógica y los componentes gráficos que permiten la realización de un trabajo perfecto y adaptado a nuestras exigencias.

Además bootstrap cuenta con las siguientes ventajas:

- **Twitter y la sección aurea**:
  La sección aurea es una relación de proporcionalidad entre medidas. Aquello que tiene una relación de sección aurea es agradable a la vista y el diseño de Twitter cumple esta relación. Por lo tanto, usando Bootstrap la aplicación web será más agradable a la vista.

Ilustración 32: Sección aurea de Bootstrap.

- **Velocidad de desarrollo:**
  Con Bootstrap no es necesario desarrollar ni código CSS ni código JavaScript, todo lo necesario lo proporciona el framework. De esta forma solo hay que preocuparse de desarrollar código HTML.
- **Layout responsive:**
  Con una clase se puede gestionar el responsive del layout, de forma que sea directamente compatible con los formatos de los diferentes dispositivos.
- **Componentes integrados:**
  Los componentes JavaScript más utilizados ya están incluidos en el framework, esto hace que la posibilidad de conflictos entre librerías sea nula lo que aumenta la fiabilidad del proyecto.

## Diferencias entre la versión 2.0 y la versión 3.0

- **Mobile-first:**
  Con Bootstrap 2 existían directivas que permitían hacer la interfaz mobile-friendly, con la nueva versión todo el proyecto se ha modificado de forma que sea mobile-friendly desde el inicio.
- **Modificación Grid System:**
  Se ha reescrito el sistema grid system. El mayor cambio se observa en la declaración de las columnas. El sistema se vuelve más versátil de forma que se puede tener control total sobre cómo se visualizará la página en los tres tipos de dispositivos principales (desktop, Tablet, smartphones).
- **Flat Design:**
  Actualmente el diseño web se ha convertido en un instrumento para presentar los contenidos en un determinado dispositivo. Bootstrap sigue la tendencia actual de eliminar los elementos de diseño innecesarios, los botones con relieve se sustituyen por botones planos, las barras de navegación pasan a ser monocromáticas en lugar de con gradiente…

- **Mejoras:**
  Se han añadido elementos Panels, que son útiles para crear widgets en zonas secundarias. También se han introducido List Group, lo que lo acerca más a un framework para diseñar GUI para móviles. Otros cambios son que se han reescrito las cuadrículas de thumbnail para hacerlas más compresibles, así como el Carousel, que se encuentra más ordenado y con una apariencia más elegante.

La elección de Bootstrap a la hora de desarrollar la aplicación es debida a la necesidad de crear una página **responsive**, es decir, una página que se adapte a todo tipo de dispositivos, esto es necesario debido a la gran heterogeneidad de dispositivos existentes. Con Bootstrap se ha conseguido que la página se adapte al navegador, de tal manera que si el usuario minimiza la página, esta se adaptará al navegador, así como también se adaptará a los dispositivos móviles si el usuario accede a ella a través de otro dispositivo que no sea un PC.

Además, Bootstrap en su página oficial ofrece una serie de clases para crear el estilo de botones, tablas, imágenes, menús, etc. Todo esto ha sido de mucha utilidad en el desarrollo de la aplicación, ya que ha permitido crear elementos con un estilo muy agradable a la vista del usuario.

### 3.2.3 Lado del servidor

A continuación se introducen las tecnologías del lado del servidor utilizadas: PHP, el motor de Base de datos MySQL y el framework PHP Laravel.

#### *PHP*

PHP (Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de otros lenguajes del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo.

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales.

PHP está enfocado principalmente a la programación de scripts del lado del servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa CGI (Common Gateway Interface), como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Aunque PHP puede hacer mucho más.

Existen principalmente tres campos principales donde se usan scripts de PHP.

- **Scripts del lado del servidor.** Este es el campo más tradicional y el foco principal. Son necesarias tres cosas para que esto funcione. El analizador de PHP (módulo CGI o servidor), un servidor web y un navegador web. Es necesario ejecutar el servidor con una instalación de PHP conectada. Se puede acceder al resultado del programa de PHP con un navegador, viendo la página de PHP a través del servidor.
- **Scripts desde la línea de comandos.** Se puede crear un script de PHP y ejecutarlo sin necesidad de un servidor o navegador. Solamente es necesario el analizador de PHP para utilizarlo de esta manera. Este tipo de uso es ideal para scripts que se ejecuten con regularidad empleando cron (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden usarse para tareas simples de procesamiento de texto.
- **Escribir aplicaciones de escritorio.** Probablemente PHP no sea el lenguaje más apropiado para crear aplicaciones de escritorio con una interfaz gráfica de usuario, pero si se conoce bien PHP, y se quisiera utilizar algunas características avanzadas de PHP en aplicaciones del lado del cliente, se puede utilizar para escribir dichos programas.

PHP puede emplearse en todos los sistemas operativos principales, incluyendo Linux, muchas variantes de Unix, Microsoft Windows, Mac OS X y probablemente otros más. PHP admite la mayoría de servidores web de hoy en día, incluyendo Apache, IIS, y muchos otros. PHP funciona tanto como módulo como procesador de CGI.

De modo que con PHP, se tiene la libertad de elegir el sistema operativo y el servidor web. Además, se tiene la posibilidad de utilizar programación por procedimientos o programación orientada a objetos (POO), o una mezcla de ambas.

Con PHP no se está limitado a generar HTML. Entre las capacidades de PHP se incluyen la creación de imágenes, ficheros PDF e incluso películas Flash generadas sobre la marcha. También se puede generar fácilmente cualquier tipo de texto, como XHTML y cualquier otro tipo de fichero XML. PHP puede autogenerar estos ficheros y guardarlos en el sistema de ficheros en vez de

imprimirlos en pantalla, creando una caché en el lado del servidor para contenido dinámico.

Una de las características más potentes y destacables de PHP es su soporte para un amplio abanico de bases de datos. Escribir una página web con acceso a una base de datos es increíblemente simple utilizando una de las extensiones específicas de bases de datos (p.ej., para mysql).

### *MySQL*

El software MySQL proporciona un servidor de base de datos SQL (Structured Query Language) muy rápido, multi-threaded, multi usuario y robusto. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido. MySQL es una marca registrada de MySQL AB.

El software MySQL tiene una doble licencia. Los usuarios pueden elegir entre usar el software MySQL como un producto Open Source bajo los términos de la licencia GNU General Public License o pueden adquirir una licencia comercial estándar de MySQL AB.

### Modelo entidad-relación

El modelo es solo y exclusivamente un método del que se dispone para diseñar esquemas que posteriormente se deben de implementar en un gestor de BBDD (bases de datos). Este modelo se representa a través de diagramas y está formado por varios elementos.

Este modelo habitualmente, además de disponer de un diagrama que ayuda a entender los datos y como se relacionan entre ellos, debe de ser completado con un pequeño resumen con la lista de los atributos y las relaciones de cada elemento.

Las entidades representan cosas u objetos (ya sean reales o abstractos), que se diferencian claramente entre sí. Los atributos definen o identifican las características de entidad (es el contenido de esta entidad). Cada entidad contiene distintos atributos, que dan información sobre esta entidad. Estos atributos pueden ser de distintos tipos (numéricos, texto, fecha…).

Las relaciones son un vínculo que permite definir una dependencia entre varias entidades, es decir, permite exigir que varias entidades compartan ciertos atributos de forma indispensable.

Existen tres tipos de relaciones:

- **Relaciones de uno a uno.** Estas relaciones existen por ejemplo en el caso de una persona y su dni, una persona sólo puede tener un dni, y un dni sólo puede pertenecer a una persona. Para llevar a cabo esta relación en la base de datos simplemente se debe crear una tabla usuarios y una tabla dnis, para hacer referencia al dni de cada usuario basta con crear un campo en la tabla dnis el cuál actuará como clave foránea haciendo referencia al usuario a través de su id. Una id, clave primara, es un identificador único y autoincremental que deben tener todas las tablas de una base de datos.

  Una clave foránea es una clave que se usa en una tabla secundaria, en este ejemplo en la tabla dnis, y que coincide con la clave primaria (id) de la tabla con la que se quiere relacionar, en este caso, con la tabla usuarios.
- **Relaciones de uno a muchos.** El ejemplo perfecto para estas relaciones es entre usuarios y posts de un blog, un usuario puede tener muchos posts, pero un post sólo puede pertenecer a un usuario, sirve lo mismo que en la relación de uno a uno. La única diferencia entre estas dos relaciones, es que la clave foránea entre usuarios y dnis puede estar tanto en la tabla usuarios con un campo id_dni como en la tabla dnis con un campo id_usuario. En cambio, en una relación de uno a muchos la clave foránea siempre debe estar en la tabla que hace la relación de muchos, en este caso sería la tabla posts.
- **Relaciones de muchos a muchos.** Este tipo de relaciones son las más complicadas. Por ejemplo se puede decir que la relación entre usuarios y películas (alquileres de un videoclub), un usuario puede alquilar muchas películas, y una película puede ser alquilada por muchos usuarios.
  Estas relaciones no pueden ser llevadas a cabo con simples claves foráneas ya que se necesitaría una por cada registro, cosa completamente inviable. Para este tipo de relaciones se debe crear una tercera tabla, conocida como tabla pivote, que por convención su nombre suele ser usuarios_peliculas para el caso, es decir, los nombres de las dos tablas separados por guiones.
  Estas tablas deben contener como mínimo dos campos, usuario_id y pelicula_id que harán referencia a las claves primarias de sus respectivas tablas.
  La función de esta tabla es la de poder enlazar a los usuarios y las películas a través de sus claves primarias, es decir, si se tiene un usuario con id 1 y una película con id 120 en sus respectivas tablas, para poder unirlos, se debería crear un nuevo registro en la tabla usuarios_peliculas con esos datos.

## *Laravel*

Laravel es un marco de desarrollo web MVC escrito en PHP [4], por lo que se hace necesario explicar que es un “modelo-vista-controlador” o “MVC”. [1]

MVC es un patrón de arquitectura de software que separa la lógica de la capa de presentación (Vistas/HTML).

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores. Lo primero que se tiene que saber es que ayuda a crear aplicaciones con mayor calidad, ya que asegura calidad en los programas que se realizan y esa calidad atiende a diversos parámetros que son deseables para todo desarrollo, como la estructuración de los programas o reutilización del código, lo que debe influir positivamente en la facilidad de desarrollo y el mantenimiento.

### *Ventajas*

- Al principio, en el HTML se mezclaba tanto el contenido como la presentación. Es decir, en el propio HTML habían etiquetas como "font" que sirven para definir las características de una fuente, o atributos como "bgcolor" que definen el color de un fondo. El resultado es que tanto el contenido como la presentación estaban juntos y si algún día se pretendía cambiar la forma con la que se mostraba una página, se obligaba a cambiar cada uno de los archivos HTML que componen una web, tocando todas y cada una de las etiquetas que hay en el documento. Con el tiempo se observó que eso no era práctico y se creó el lenguaje CSS, en el que se separó la responsabilidad de aplicar el formato de una web.
- Al escribir programas en lenguajes como PHP, cualquiera comienza mezclando tanto el código PHP como el código HTML (e incluso el Javascript) en el mismo archivo. Esto produce lo que se denomina el "Código Espagueti". Si algún día se pretende cambiar el modo en cómo se quiere que se muestre el contenido, se está obligado a repasar todas y cada una de las páginas que tiene el proyecto. Sería mucho más útil que el HTML estuviera separado del PHP.
- Si se quiere que en un equipo intervengan perfiles distintos de profesionales y trabajen de manera autónoma, como diseñadores o programadores, ambos tienen que tocar los mismos archivos y el diseñador se tiene necesariamente que relacionar con mucho código en un lenguaje de programación que puede no serle familiar, siendo que a éste quizás solo le interesan los bloques donde hay HTML. De nuevo, sería mucho más fácil la separación del código.
- Durante la manipulación de datos en una aplicación es posible que se esté accediendo a los mismos datos en lugares distintos. Por ejemplo, se puede acceder a los datos de

un artículo desde la página donde se muestra éste, la página donde se listan los artículos de un manual o la página de backend donde se administran los artículos de un sitio web. Si un día se cambian los datos de los artículos, se está obligado a cambiar, página a página, todos los lugares donde se consumían datos de los artículos. Además, si se tiene el código de acceso a datos disperso por decenas de lugares, es posible que se esté repitiendo las mismas sentencias de acceso a esos datos y por tanto no se esté reutilizando código.

Hay decenas de casos similares en los que resultaría útil aplicar una arquitectura como el MVC, con la que se tiene que separar el código atendiendo a sus responsabilidades.

Ahora que ya se tiene una idea de las ventajas que puede aportar el MVC, se analizan las diversas partes o conceptos en los que se debe separar el código de las aplicaciones.

Modelo: Es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos están habitualmente en una base de datos, por lo que en los modelos se tienen todas las funciones que accederán a las tablas y harán los correspondientes selects, updates, inserts, etc.

Vista**:** Las vistas, como su nombre hace entender, contienen el código de la aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que permitirá renderizar los estados de la aplicación en HTML. En las vistas nada más están los códigos HTML y PHP que permiten mostrar la salida.

Controlador: Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc. En realidad es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de la aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo.

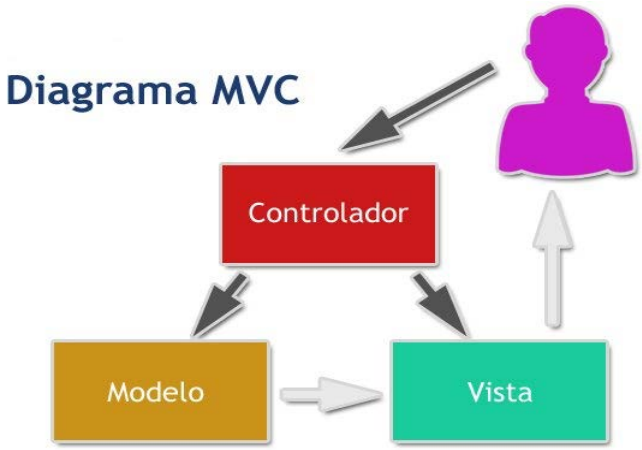



**Ilustración 33: Diagrama del Modelo-Vista-Controlador (MVC).**

En esta imagen se representa con flechas los modos de colaboración entre los distintos elementos que formarían una aplicación MVC, junto con el usuario. Como se puede ver, los controladores, con su lógica de negocio, hacen de puente entre los modelos y las vistas. Pero además en algunos casos los modelos pueden enviar datos a las vistas. A continuación se muestra paso a paso cómo sería el flujo de trabajo característico en un esquema MVC.

1. El usuario realiza una solicitud al sitio web. Generalmente estará desencadenada por acceder a una página del sitio. Esa solicitud le llega al controlador.
2. El controlador comunica tanto con modelos como con vistas. A los modelos les solicita datos o les manda realizar actualizaciones de los datos. A las vistas les solicita la salida correspondiente, una vez se hayan realizado las operaciones pertinentes según la lógica del negocio.
3. Para producir la salida, en ocasiones las vistas pueden solicitar más información a los modelos. En ocasiones, el controlador será el responsable de solicitar todos los datos a los modelos y de enviarlos a las vistas, haciendo de puente entre unos y otros.
4. Las vistas envían al usuario la salida. Aunque en ocasiones esa salida puede ir de vuelta al controlador y sería éste el que hace el envío al cliente.

El que Laravel haga uso de este patrón hace que el desarrollo de esta aplicación sea mucho más rápida e intuitiva, ya que clasifica los ficheros de código en distintas carpetas y hace más cómodo programar la aplicación porque no se mezcla el código, en definitiva, ayuda a mantener, extender y cambiar la aplicación fácilmente.

Laravel fue diseñado con la filosofía de utilizar convención sobre configuración. Esto significa que hace suposiciones inteligentes sobre lo que se está tratando de lograr, por lo que en la mayoría de las situaciones, será capaz de lograr las metas con mucho menos código. No

todas las aplicaciones y las bases de datos con las que se trabaja serán diseñadas usando esta convección. Afortunadamente, Laravel es lo suficientemente flexible para funcionar con el sistema, no importa cuán único es.

Laravel ha sido diseñado para atacar el punto dulce entre el minimalismo y la funcionalidad. Es más fácil de entender las bases de código de menor tamaño y Laravel trata sobre la implementación de soluciones, de manera que es limpio, sencillo y elegante. Los desarrolladores de PHP desde hace mucho tiempo se encuentran muchos aspectos de Laravel familiarizados, ya que es una evolución de los framework de desarrollo PHP que han venido antes de él.

Laravel es uno de los pocos frameworks de PHP que ofrece código modular. Esto se logra a través de una combinación de los controladores y el sistema de paquetes. Los controladores permiten cambiar fácilmente y extender el almacenamiento en caché, la sesión, base de datos, y la funcionalidad de autenticación. Usando paquetes, es capaz de empaquetar cualquier tipo de código, ya sea para la propia reutilización o para proveer al resto de la comunidad Laravel del código desarrollado. Esto es muy emocionante, porque cualquier cosa que se puede escribir en Laravel puede ser empaquetado como un paquete, de bibliotecas enteras simples para las aplicaciones web. La página web de Laravel permite navegar por los paquetes que han sido construidos por la comunidad, así como mostrar su propia cuenta. Es un recurso valioso de librerías de terceros y subsistemas que pueden facilitar enormemente el desarrollo de la aplicación web.

Laravel también proporciona un conjunto de herramientas de última generación para interactuar con las bases de datos. Las migraciones son una de las herramientas para interactuar con la base de datos, que permiten diseñar y modificar fácilmente la base de datos de una manera independiente de la plataforma.

Las migraciones pueden ser ejecutadas en cualquiera de los tipos de base de datos que soporta Laravel (MySQL, PostgreSQL, MSSQL, y SQLite) y no se tendrá ningún problema de compatibilidad.

El constructor fluido de consultas de Laravel abstrae las diferencias entre los distintos tipos de bases de datos.

La Implementación ActiveRecord de Laravel se llama **Eloquent**. Interactuar con una base de datos de una manera orientada a objetos es el estándar moderno. Con Eloquent, se puede crear, recuperar, actualizar y eliminar los registros de la base de datos sin necesidad de escribir una sola línea de SQL. Además de esto, Eloquent ofrece una potente gestión de relaciones y que incluso puede manejar la paginación automáticamente.

Laravel también viene con una herramienta de interfaz de línea de comandos denominada **Artisan**. Con Artisan, un desarrollador puede interactuar con su aplicación para desencadenar acciones como la ejecución de las migraciones, la ejecución de pruebas unitarias, y la ejecución de tareas programadas. Artisan también es completamente extensible para que se pueda escribir cualquier tipo de funcionalidad que se desee.

Laravel incluye un sistema de rutas que permite manejar fácilmente las URL del sitio web. También incorpora un helper HTML, con el que se puede crear enlaces dentro del sitio web que se actualizará automáticamente a sí mismos si se cambian las direcciones URL, esto hace mucho más fácil el trabajo de mantenimiento de la aplicación.

El motor de plantillas de Laravel, **Blade**, permite usar una bonita sintaxis para incluir código PHP en las vistas. También incluye un número de atajos que permiten el uso de características existentes de Laravel. Las plantillas Blade hacen uso de la cache para darle mayor velocidad.

## Instalación

Para poder instalar Laravel, se necesita tener instalado un servidor local para poder probar la aplicación. En este caso se utiliza XAMPP, como se ha explicado anteriormente.

Desde la versión 4 de Laravel, la creación de un proyecto nuevo se maneja con Composer. Composer es un manejador de dependencias para PHP. Esto quiere decir que Composer va a descargar de sus repositorios todas las librerías y las dependencias, con las versiones requeridas por el proyecto, y va a manejarlas en un solo lugar de manera ordenada. En otras palabras, Composer es como un recetario que se encarga de descargar todo lo que se necesite para ejecutar un proyecto de manera que libera de la tediosa tarea de descargar cada librería de manera separada. [1][5]

### *Instalación de Composer en Windows*

En Windows la instalación se puede hacer mediante un instalador ejecutable que se descarga en la página de Composer. El instalador solicitará la ubicación del **php.exe**, la cual dependerá de donde se haya instalado XAMPP. [6]

El instalador se encargará de modificar la variable **PATH** para que se pueda hacer uso de Composer desde cualquier lugar en la consola. Para probar que la instalación se llevó acabo correctamente se tiene que ejecutar el siguiente comando, aparecerá una lista con todos los posibles comandos que acepta Composer.

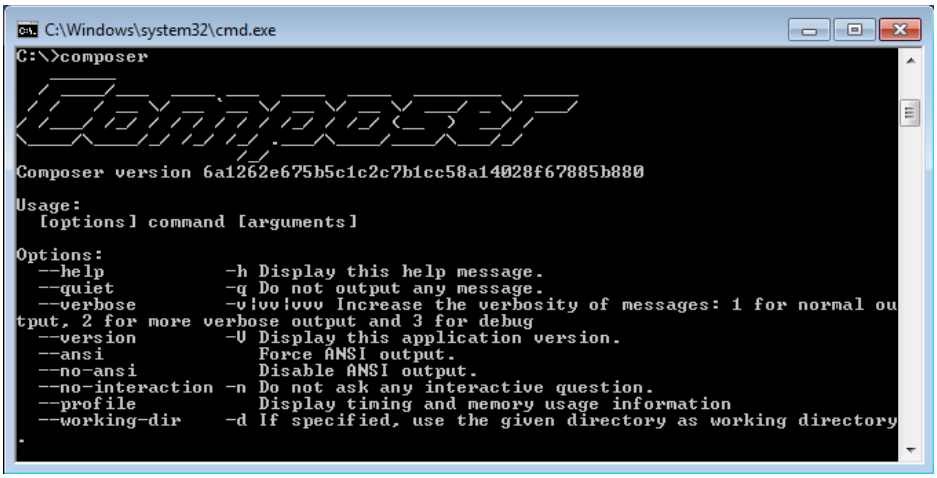



**Ilustración 34: Lista de comandos.**

### *Instalación de Laravel*

Una vez que se tenga Composer funcionando en el ordenador se puede descargar una copia de Laravel 4 para crear el proyecto.

Ejecutando un comando de Laravel, se descargará una copia completa de la versión más reciente de Laravel desde los repositorios de Composer con todas las dependencias y librerías que el framework necesita.

Para probar que funciona correctamente, se debe poner en marcha el servidor Apache y entrar desde el navegador a localhost/laravel/public.

### Estructura

El directorio raíz del proyecto tiene la siguiente estructura:

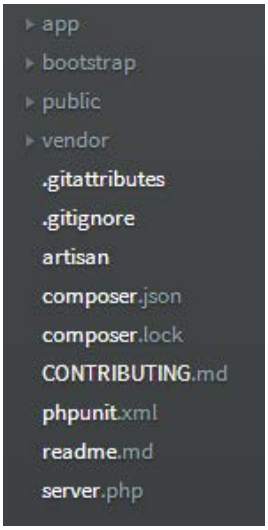



**Ilustración 35: Directorio raíz del proyecto.**

- **app.** Lo primero es el directorio app. app es la carpeta principal de la aplicación, en ella estará gran parte del código del proyecto. Eso incluye clases que puedan ofrecer funcionalidad a la aplicación, archivos de configuración y más. La carpeta app es bastante importante por lo que se cubrirá en detalle más adelante.
- **bootsrap.** Esta carpeta contiene los siguientes archivos:
    - autoload.php
    - paths.php
    - start.php

  El directorio bootstrap contiene unos pocos archivos que están relacionados con los procedimientos de inicialización del framework. El archivo autoload.php contiene la mayoría de esos procedimientos y solo debería ser editado por usuarios experimentados de Laravel. El archivo paths.php contiene una matriz de las rutas comunes del sistema que son usadas por el framework. Los contenidos del directorio bootstrap solo deberían ser editados por usuarios experimentados de Laravel que necesiten alterar la forma del sistema de archivos del framework.
- **public.** Esta carpeta contiene los siguientes archivos:
    - packages/
    - .htaccess
    - favicon.ico
    - index.php
    - robots.txt

  El directorio public debería ser la única entrada web de una aplicación Laravel. Normalmente es donde se encuentran los archivos CSS, JavaScript e imágenes. [3]

  El archivo packages será usado para contener cualquier fichero que necesite ser instalado por paquetes de terceras partes. Se mantienen en un directorio separado para que no creen conflictos con los ficheros de la aplicación.

  Laravel viene con un archivo .htaccess para servidores Apache. Contiene algunas directivas de configuración estándar que tendrán sentido para la mayoría de los usuarios del framework.

  Por defecto, los navegadores web intentarán buscar en la raíz de un sitio para encontrar un archivo favicon.ico. Este es el archivo que controla la pequeña imagen de 16 x 16px que se muestra en las pestañas del navegador. El problema es que cuando el archivo no existe, al servidor web le gusta quejarse sobre ello. Esto causa entradas en el log innecesarias. Para contrarrestar este problema, Laravel da un archivo favicon.icon en blanco, que puede ser reemplazado más tarde si se desea.

  El archivo index.php es el controlador frontal del framework de Laravel. Es el primer

archivo que el servidor web ejecuta cuando llega una petición del navegador. Es el archivo que lanzará la inicialización del framework.
Laravel ha incluido el archivo robots.txt que permite todos los hosts por defecto.

- **vendor.** El directorio vendor contiene todos los paquetes de Composer que son utilizados por la aplicación. Por supuesto, esto incluye el paquete del framework de Laravel.
- **.gitattributes.** Laravel ofrece algunos ajustes por defecto para control de versiones con Git. Git es el sistema de control de versiones más popular.
- **.gitignore.** El archivo .gitignore contiene algunos ajustes para informar a Git de qué carpetas no debería controlar.
- **artisan.** El archivo artisan es un ejecutable que es usado para ejecutar la interfaz de línea de comandos Artisan para Laravel. Artisan contiene un buen número de comandos útiles para ofrecer atajos o funcionalidad adicional al framework.[3]
- **composer.json y composer.lock.** Tanto composer.json como composer.lock, contienen información sobre los paquetes de Composer usados por este proyecto.
- **phpunit.xml.** El archivo phpunit.xml ofrece configuración por defecto para las pruebas unitarias de PHPUnit. Gestionará la carga de dependencias de Composer y ejecutará cualquier prueba ubicada en la carpeta app/tests.

El directorio de la aplicación tiene la siguiente forma:

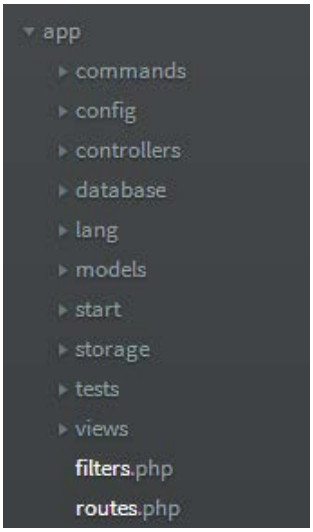



**Ilustración 36: Directorio app de la aplicación.**

- **commands.** Este directorio contiene cualquier comando, que pueda necesitar la aplicación, de Artisan. Artisan no solo ofrece funcionalidad por defecto para ayudar a construir un proyecto, sino que también ofrece la oportunidad de crear comandos.

- **config.** La configuración tanto para el framework como para la aplicación se mantiene en este directorio. La configuración de Laravel existe como un conjunto de archivos PHP que contienen matrices clave-valor. Este directorio también contiene sub-directorios que permiten distintas configuraciones cargadas en diferentes entornos.
- **controllers.** Este directorio contendrá los controladores. Laravel facilita dos maneras de enrutado, el uso de controllers (controladores) y el uso de routes (rutas). Los controladores pueden ser usados para facilitar lógica a la aplicación, y unir las partes separadas de la aplicación. Este directorio ha sido añadido al archivo composer.json por defecto para la auto carga de clases. [3]
- **database.** Si se escoge una base de datos como método para guardar cosas a largo plazo, este directorio será usado para contener los archivos que crearán el esquema de la base de datos, y los métodos para completarla con datos de ejemplo.
- **lang.** El directorio lang contiene archivos PHP con matrices de cadenas que pueden ser usados para dar soporte de traducción a la aplicación. Se pueden crear sub carpetas para que tener distintos ficheros para múltiples idiomas.
- **models.** Este directorio contendrá los modelos. Los modelos son usados para representar el modelo de negocio o facilitar interacción con el almacenamiento. Al igual que el directorio controllers, este ha sido añadido a la sección de carga automática del archivo composer.json por defecto.
- **start.** Mientras que el directorio bootstrap contiene los procedimientos de arranque que pertenecen al framework, el directorio start contiene los procedimientos de arranque que pertenecen a la aplicación.
- **storage.** Cuando Laravel necesita escribir algo en el disco, lo hace en el directorio storage. Por este motivo el servidor web debe poder escribir en esta ubicación.
- **tests.** El directorio tests contiene todas las pruebas unitarias y de aceptación para la aplicación. La configuración por defecto de PHPUnit que ha sido incluida por Laravel buscará pruebas en este directorio por defecto.
- **views.** El directorio views es usado para contener las plantillas visuales de la aplicación. Se facilita una vista hello por defecto.
- **filters.php.** El archivo filters.php es usado para contener cualquier filtro de rutas de la aplicación.
- **routes.php.** El archivo routes contiene todas las rutas de la aplicación.

## Base de datos

El siguiente paso en la creación de la aplicación, es la configuración de la base de datos. Para esto se ha creado una base de datos con el nombre CVs en MySQL. Una vez creada la base de datos se edita el archivo de configuración en Laravel para que se pueda realizar la conexión. Para ello se abre el archivo /app/config/database.php y se editan los campos de la conexión MySQL. [4]

```
'mysql' => array(
    'driver'    => 'mysql',
    'host'      => 'localhost',
    'database'  => 'CVs',
    'username'  => '   ',
    'password'  => '          ',
    'charset'   => 'utf8',
    'collation' => 'utf8_spanish_ci',
    'prefix'    => '',
),
```

**Ilustración 37: Conexión MySQL.**

Para la creación de las tablas de la base de datos se usan migraciones. Las migraciones son un número de scripts PHP que son usados para cambiar la estructura y/o contenido de la base de datos. Las migraciones tienen una fecha y hora marcadas por lo que se ejecutan siempre en el orden correcto.

Laravel mantiene un registro de qué migraciones ha ejecutado en la tabla migrations. De esta forma, solo ejecutará las migraciones adicionales que hayan podido ser añadidas. [4]

Usando migraciones siempre se tendrá la misma estructura de la base de datos, de manera consistente y estable.

Para crear una migración se tiene que usar la interfaz de línea de comandos Artisan. [1] Para ello se tiene que ejecutar el siguiente comando:

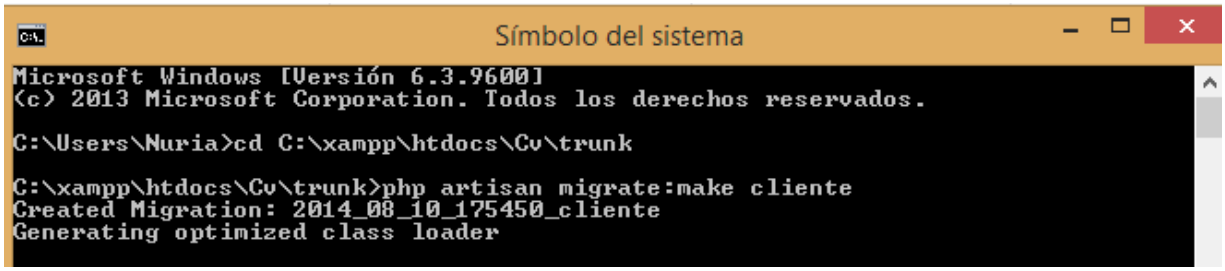



**Ilustración 38: Comando que debe ejecutarse para la creación de una migración.**

Se llama al método de Artisan, migrate:make y se le da un nombre a la migración. Laravel generara una plantilla de migración en el directorio app/database/migrations. La plantilla estará ubicada en un fichero nombrado con el parámetro que se le ha puesto al método migrate:make, con una fecha y hora añadidas. En este caso estará ubicada en app/database/migrations/2014_08_10_175450_cliente.php

Al abrir este fichero se ve la estructura del mismo:

```php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class Cliente extends Migration {

    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        //
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }

}
```

**Ilustración 39: Estructura de una migración.**

En la clase migración, hay dos métodos públicos up() y down(). El método up sirve para ejecutar la migración, mientras que el método down hace lo contrario, la deshace. [2]

Finalmente, un fichero de una migración debe tener la siguiente estructura:

```php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class Cliente extends Migration {

    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('cliente',function($table)
        {
            $table->create();
            $table->increments('id');
            $table->integer('email');
            $table->integer('nombre')

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('cliente');
    }

}
```

**Ilustración 40: Estructura de una migración.**

De esta forma se puede crear una tabla llamada cliente en la base de datos, como se puede ver, el método up crea la tabla y le añade un campo id, un campo email y otro campo nombre, también se le añade los métodos created_at y update_at, estos se añaden automáticamente usando el método timestamps. Por el contrario, el método down elimina la tabla cliente.

Antes de ejecutar una migración se debe ejecutar el comando php artisan migrate:install, este comando crea la tabla migrations en la base de datos. Para migrar la tabla cliente a la base de datos, se debe ejecutar el siguiente comando:

```
C:\xampp\htdocs\Cv\trunk>php artisan migrate
<?phMigrated: 2014_08_10_175450_cliente
```

**Ilustración 41: Comando para realizar una migración.**

Con este comando se crea la tabla en la base de datos con los campos definidos. Si por algún motivo se tiene que alterar alguno de los archivos de migración, se puede usar el comando migrate:refresh de Artisan para deshacer todas las migraciones y luego ejecutarlas una vez más.

El comando php artisan migrate:rollback deshace la última migración hecha con migrate, mientras que el comando php artisan migrate:reset deshace todas las migraciones. [4]

Una vez que se ha creado la base de datos, se tienen que crear las vistas, los controladores, los modelos y las rutas para unir las vistas con los controladores.

## Vistas

Las vistas están formadas por código HTML, aunque pueden incluir pequeños trozos de código php como variables, condicionales o bucles. Por su parte Laravel utiliza un motor de plantillas llamado **Blade**, que facilitará el desarrollo del código y resumirá el mismo. [2]

Las vistas deben crearse dentro de la carpeta views, dentro de app -> views. Para poder generar un layout con blade, basta con crear un archivo dentro de la carpeta app/views con un nombre que tenga extensión .blade.php, usando esta expresión, Laravel sabe que es una vista realizada con el gestor de plantillas.

Blade utiliza unas funciones que son las que ayudan a crear las plantillas mediante herencia y creación de secciones para no tener que repetir código HTML en cada vista. En la aplicación se han creado las carpetas layout y blade. Layout contiene las plantillas con llamadas a funciones include(). La función include() llama a la vista indicada y la muestra en el lugar correspondiente del HTML. Esto es muy útil ya que evita que se repita el código HTML para cada vista, por ejemplo, para el código sidebar, que es el mismo en todas las vistas, se usa un include('blade.sidebar') que llamará al fichero situado en blade/sidebar.blade.php. [7]

```
1  |
2      @include('blade.beginning')
3      @include('blade.head')
4  <body>
5      @include('blade.header')
6      @include('blade.sidebar')
7      @include('blade.contenidobecas')
8      @include('blade.footer')
9      @include('blade.end')
10
11 </body>
12 </html>
```

**Ilustración 42: Plantilla usada en la aplicación.**

Como se puede observar en la ilustración 42, se incluye en un HTML los códigos de beginning, head, header, sidebar, contenido, footer y end. De esta manera, lo único que se tiene que cambiar para cada vista es el fichero de contenido, ya que cada formulario contendrá un contenido diferente.

Para mostrar una variable se utilizan los dobles corchetes: {{$nombre}}, como se puede ver, es mucho más sencillo que con php. Además, los comentarios no son visibles en el código fuente de la aplicación, al contrario que en con los comentarios tradicionales, donde sí se muestran. Los comentarios se ponen de la siguiente manera: {{-- Esto es un comentario --}}. [7]

Blade utiliza las siguientes estructuras de control: [7]

- For:

```
@for ($i = 0; $i <= count($comments); $i++)
    The comment body is {{ $comments[$i] }}
@endfor
```

**Ilustración 43: Bucle for.**

- Foreach:

```
@foreach ($comments as $comment)
    The comment body is {{ $comment->body }}.
@endforeach
```

**Ilustración 44: Bucle foreach.**

- While:

```
@while ($something)
    I am still looping!
@endwhile
```

**Ilustración 45: Bucle while.**

- If:

```
@if ( $message == true )
    I'm displaying the message!
@endif
```

**Ilustración 46: Estructura if.**

- If else:

```
@if ( $message == 'success' )
    It was a success!
@elseif ( $message == 'error' )
    An error occurred.
@else
    Did it work?
@endif
```

**Ilustración 47: Estructura if else.**

- For else:

```
@forelse ($posts as $post)
    {{ $post->body }}
@empty
    There are not posts in the array!
@endforelse
```

Ilustración 48: Estructura for else.

- Unless:

```
@unless(Auth::check())
    Login
@endunless

// Equivalent to…

<?php if ( ! Auth::check()): ?>
    Login
<?php endif; ?>
```

Ilustración 49: Estructura unless.

Una de las grandes ventajas de Blade es que no usa el mismo código tradicional php, es decir utiliza uno más sencillo que sustituye a este, por ejemplo, para crear un bucle foreach que recorra los usuarios y para cada usuario muestre su nombre, con php se haría así:

```
<ul>
  <?php foreach $users as $user : ?>
  <li><?php echo $user->name; ?></li>
  <?php endforeach; ?>
</ul>
```

Ilustración 50: Ejemplo de bucle con php.

Y con Blade se haría así:

```
<ul>
  @foreach ($users as $user)
  <li>{{ $user->name }}</li>
  @endforeach
</ul>
```

Ilustración 51: Ejemplo de bucle con Blade.

Como se puede observar, la sintaxis es mucho más clara, sencilla y limpia que la utilizada con php.

## Modelos

Una de las cosas más densas, pero necesarias al desarrollar una aplicación web es tener que crear las consultas para la base de datos, además que estas consultas suelen repetirse entre tabla y tabla. Insertar registros, modificarlos, eliminarlos, buscar un registro por su id, listar registros, etc. Eloquent es una clase ORM de Laravel que permitirá solucionar este problema.

En español las siglas ORM significan “Mapeo Objeto-Relacional” y es una técnica de programación. Esto significa que cuando se aplica esta técnica se puede acceder a los registros de la base datos como si fueran objetos de PHP y no tener que ejecutar código SQL. Esto es posible porque cada tabla de la base datos es manejada por una clase del proyecto (modelos). Esta técnica también permite crear las relaciones como propiedades de los objetos y así poder relacionarlos de una manera más sencilla cuando se quieran realizar operaciones sobre ellos. Esto permite independizarse de una base de datos en específico, porque al tratar con objetos y no con la base de datos directamente, se puede cambiar el motor de base de datos en cualquier momento, y el código continuará funcionando sin ningún problema.

Un modelo Eloquent puede ser utilizado para consultar una tabla asociada de la base de datos, así como también representar una fila dentro de la tabla. Los modelos son almacenados comúnmente en el directorio app/models.

Un modelo Cliente.php se define de la siguiente manera:

```
class Cliente extends Eloquent {}
```

Con Eloquent no hay que decirle que tabla utilizar. Eloquent tiene una variedad de convenciones, una de las cuales es utilizar el nombre plural del modelo como la tabla de la base de datos, por ejemplo, si el modelo se llama “Cliente”, la tabla de la base de datos se debe llamar “clientes”. [2]

Eloquent asume que en cada tabla hay una clave primaria llamada id.

Las relaciones entre las tablas se definen en los modelos, de esta forma al crear un objeto de algún modelo ya estará relacionado con los modelos (clases) a los que pertenezca.

Las relaciones se crean de la siguiente manera: [5]

- **Uno a uno.** Esta es la relación más sencilla de todas.

```
public function subscripciones()
{
    return $this->hasOne('Subscripcione','cliente_id','id');
}
```

Ilustración 52: Relación de uno a uno.

De esta manera se define la relación creada en la aplicación, en la que un cliente tiene una y solo una subscripción.
El primer argumento pasado a la función hasOne es el nombre del modelo, el segundo argumento es la clave foránea utilizada y el tercer argumento la clave local.
El inverso de esta relación es el método belongsTo.

```
class Subscripcione extends Eloquent
{
    public function clientes()
    {
        return $this->belongsTo('Cliente');
    }
}
```

Ilustración 53: Relación belongsTo.

En este caso, una subscripción pertenece a un solo cliente.

- **Uno a muchos.** Se puede modelar esta relación de la siguiente manera:

```
public function conocimientos_de_idiomas()
{
    return $this->hasMany('ConocimientosDeIdioma','cliente_id','id');
}
```

Ilustración 54: Relación de uno a muchos.

En la ilustración 54 se ha definido una relación de uno a muchos con el método hasMany, en el que un cliente puede saber muchos idiomas. Como en la relación de uno a uno, el primer argumento pasado es el nombre del modelo, el segundo la clave foránea y el tercero la clave local.
En una relación de uno a muchos, el método contrario de hasMany es belongsTo, en la siguiente ilustración se puede ver como un idioma pertenece a un cliente, al contrario de antes en el que un cliente podía tener muchos idiomas.

```php
class ConocimientosDeIdioma extends Eloquent
{
    public function clientes()
    {
        return $this->belongsTo('Cliente');
    }
}
```

Ilustración 55: Relación belongsTo de conocimientos de idiomas.

- **Muchos a muchos.** Esta es la relación más complicada ya que se requiere de una tabla pivote. Esta relación no ha sido necesario implementarla en la aplicación.
  Se define la relación de muchos a muchos usando el método belongsToMany.

```php
Class Profesor extends Eloquent{

    public function asignaturas(){
        return $this->belongsToMany('Asignatura', 'profesor_asignatura', 'profesor_id', 'asignatura_id');
    }
}
```

Ilustración 56: Relación de muchos a muchos.

Esta función recibe cuatro parámetros, el primero es el modelo con el cual se quiere hacer relación, el segundo es la tabla que contiene los ids de los dos modelos y hace la unión. El tercer y cuarto parámetro son los nombres de los id que Laravel debe buscar en la tabla intermedia para hacer la relación.

## Controladores

Para unir el modelo y la vista se crea el tercer componente de MVC, el controlador. Los controladores representan una parte de la aplicación, representados con clases, que a su vez estarán divididos en acciones, representados como métodos, se puede pensar en las acciones como la alternativa directa a las closures (definidas en el apartado de enrutamiento), son muy similares tanto en apariencia como en funcionalidad. [7]

Los controladores son generalmente almacenados en el directorio app/controllers, y este directorio se encuentra registrado por defecto en la opción classmap del documento composer.json. Sin embargo, los controladores pueden técnicamente encontrarse en cualquier directorio o subdirectorio. Las declaraciones de ruta no son dependientes del archivo de la clase controlador en el disco. Así, mientras que Composer sepa cómo auto-cargar la clase controlador, esta puede ser ubicada en el lugar que se desee.

Todos los controladores deben extender de la clase BaseController. El BaseController también se almacena en el directorio app/controllers y puede utilizarse como un lugar para poner la lógica del controlador compartida. El BaseController extiende de la clase Controller.

Un controlador viene a ser una clase que se encarga de gestionar todas las peticiones realizadas por un cliente de una aplicación, en este caso cumplirá la función de interceptar la petición de una vista en la aplicación, procesar cualquier operación impuesta y por ultimo pasará a renderizar la vista.

En teoría es buena práctica tener un controlador por cada tabla en la base de datos y dicho controlador atenderá a todas las vistas que requiera la interacción de datos con dicha tabla.

Los nombres de los controladores tienen que cumplir:

- La primera letra de cada palabra en mayúsculas.
- Ha de terminar en Controller.
- Sin espacios ni guiones.

Para agregar un controlador al proyecto de Laravel, se debe ir a la carpeta "controllers" dentro de "app", una vez ahí sólo hace falta crear un archivo. Por ejemplo "TestController.php" y dentro de ese archivo se pondrá el código.

```php
<?php
class TestController extends BaseController {
   public function getIndex(){
      return 'Bienvenido a nuestro primer controlador';
   }
   public function getUsuarios(){
      $users = User::all();
      $real_names = '';
      foreach($users as $item){
         $real_names .= "{$item->real_name} <br />";
      }
      return $real_names;
   }
}
?>

```

**Ilustración 57: Código de ejemplo de un controlador.**

Como se ha comentado anteriormente, el controlador extiende de la clase BaseController. Dentro de este controlador, hay dos funciones, getIndex y getUsuarios.

La función getIndex no recibe ningún parámetro, esta es la función principal, es la que se llama cuando un usuario accede por primera vez a la página. Esta función devuelve una cadena de caracteres, que se mostrará al usuario cuando acceda.

La segunda función de este ejemplo, getUsuarios, tampoco recibe parámetros. Esta función, obtiene de la base de datos todos los usuarios existentes mediante el método User::all().

A continuación, crea una variable a la que asigna el nombre de cada usuario mediante la sentencia $item-> real_name, por último devuelve la variable en la que se habían guardados los nombres de los usuarios.

Los controladores pueden llamar a las vistas mediante el método View::make, la vista se pueden situar en diferentes carpetas, por lo que para separar las carpetas, se utilizan puntos, por ejemplo, para llamar a una vista situada en vistas dentro de la carpeta view, se tiene que poner: vistas.vistadeejemplo. Por defecto, Laravel busca una vista que se llame de esa manera y que tenga extensión .blade.php, en el ejemplo anterior, buscaría la vista "vistadeejemplo.blade.php".

En la vista se pueden mostrar valores de la base de datos, esto se consigue mediante el uso de una variable y llamando al nombre del campo de la base de datos que se quiere mostrar, por ejemplo {{$users->real_name}}, esto se puede utilizar siempre y cuando en el controlador se cree una variable "users" y se llame al método User::all().

```
3
4       public function getUsuario($id){
5         $user = User::find($id);
6         return "{$user->real_name} / {$user->email}";
7       }
8
```

**Ilustración 58: Ejemplo de una función que recibe un parámetro.**

En los controladores también se pueden crear funciones que reciban parámetros, en esta imagen se puede ver como la función getUsuario recibe un parámetro. Esta función busca un usuario en la base de datos con el id pasado como parámetro, y devuelve el nombre y el email de ese usuario.

Los controladores también se pueden crear por línea de comandos, como se puede observar en la siguiente ilustración.

```
php artisan controller:make PhotoController
```

**Ilustración 59: Comando para crear un controlador**

Este comando creará un controlador llamado PhotoController.

Laravel proporciona un tipo de controladores, llamados controladores RESTful. Estos controladores se llaman con la siguiente declaración de ruta. [5]

```
Route::resource('photo', 'PhotoController');
```

**Ilustración 60: Enrutamiento de un controlador RESTful.**

Esta crea múltiples rutas para manejar una variedad de acciones RESTful.

Los controladores RESTful están formados por los métodos index, créate, store, show, edit, update y destroy. [2]

| Verb | Path | Action | Route Name |
|---|---|---|---|
| GET | /resource | index | resource.index |
| GET | /resource/create | create | resource.create |
| POST | /resource | store | resource.store |
| GET | /resource/{resource} | show | resource.show |
| GET | /resource/{resource}/edit | edit | resource.edit |
| PUT/PATCH | /resource/{resource} | update | resource.update |
| DELETE | /resource/{resource} | destroy | resource.destroy |

**Ilustración 61: Acciones manejadas por un controlador RESTful.**

Estas acciones son usadas para el caso de un formulario en el que se tiene que rellenar datos, almacenar esos datos en la base de datos, mostrarlos, editarlos, actualizarlos y eliminarlos.

La ruta RESTful es la encargada de llamar a la acción correspondiente. Las acciones son las siguientes:

- "index", mostrará una vista.
- "create", devolverá la vista del formulario con el método View::make().
- "store", almacenará, en la base de datos, los datos obtenidos con el método POST, es decir, los datos que el usuario rellene en el formulario, estos datos se pueden obtener con la función Input::get().
- "show", mostrará los datos existentes en la base de datos.
- "edit", editará los datos. Esta acción recibirá un parámetro id para identificar al usuario y así poder modificar los datos que tenía guardados en la base de datos.
- "update", esta acción es parecida a "edit", solo que en esta ocasión actualizará los datos en lugar de editarlos.
- "destroy", eliminará datos de la base de datos. Esta acción recibirá un parámetro id y eliminará los datos correspondientes al usuario con ese id, esto se puede realizar con el método delete(), por ejemplo $usuario->delete().

Laravel proporciona muchos métodos que facilitan el manejo de los datos, como el método delete(), mencionado anteriormente, o el método save() que guarda datos en la base de datos, por ejemplo, si un usuario ha introducido su nombre en un formulario y se quiere guardar en la base de datos, se tendría que llamar a un método “store”, como el mostrado en la siguiente ilustración.

```
4       public function store(){
5
6       $usuario->id = Auth::user();
7       $usuario->Nombre = Input::get('Nombre'):
8       $usuario->save();
9       return Redirect::to('ejemplo')->with('mensaje', 'datos guardados');
10
11      }
12
```

**Ilustración 62: Ejemplo de función store.**

Este método obtiene el id del usuario actual mediante el método Auth::user(), a continuación obtiene el nombre mediante Input::get(), después guarda ese dato en la base de datos, y por último redirecciona a la ruta “ejemplo” pasando un mensaje, esto se puede hacer con el método with().

## Enrutamiento

Por último se tienen que crear las rutas, esta es la manera que el framework brinda para poder llegar hasta la acción y el controlador que se quiera.

Enrutamiento es la acción de enlazar una URL con una función en la aplicación. En Laravel, es posible enrutar de dos maneras. Se puede enrutar a una closure o a una acción (función) en un controlador. [7]

Las rutas se declaran en el archivo routes.php. Este archivo representa la conexión entre la URL del sitio y la función que contiene la lógica de la aplicación.

```
Route::get('inicio', function()
{
    $view = View::make('layout.inicio');
    $view->titulo = 'Inicio';
    $view->menurojocv = ' ';
    $view->menurojo = 'active';
    $view->menurojocontacta ='';
    return $view;
});
```

**Ilustración 63: Ejemplo de ruta closure.**

Las rutas están siempre declaradas usando la clase Route. Eso es lo que se tiene al principio, antes de ::. La parte get es el método que se usa para ‘capturar’ las peticiones que son

realizadas usando el verbo 'GET' de HTTP hacia una URL concreta. El primer parámetro es la URL que se usa para la ruta y el segundo parámetro es una función anónima, para el caso de las rutas closure, para el caso de enrutar a un controlador, el segundo parámetro es el nombre del controlador seguido del símbolo '@' más el método del controlador al que se quiere llamar.

```
Route::get('conocimientos_en_informatica','ConocimientoseninformaticaController@showIndex');
Route::post('conocimientos_en_informatica','ConocimientoseninformaticaController@postCreate');
Route::post('conocimientos_en_informatica/edit/{id}','ConocimientoseninformaticaController@edit');
Route::get('conocimientos_en_informatica/delete/{id}','ConocimientoseninformaticaController@delete');
```

**Ilustración 64: Rutas usando enrutamiento a un controlador.**

Como se puede ver en la ilustración 64, se usa el método get para llamar a los métodos en lo que no se va a pasar ningún dato, ya que no se mostrará en la URL del navegador, y el método post para los métodos en los que el usuario va a introducir algún dato.

Todas las peticiones realizadas por un navegador web contienen un verbo. La mayoría de las veces, el verbo será GET, que es usado para solicitar una página web. Se envía una petición GET cada vez que se escribe una nueva dirección web en el navegador.

Aunque no es la única petición. También está POST, que es usada para hacer una petición y ofrecer algunos datos. Normalmente se usa para enviar un formulario en el que se necesita enviar los datos sin mostrarlo en la URL.

Hay otros verbos HTTP disponibles. He aquí algunos de los métodos que la clase de enrutado tiene disponibles:

- Route::get();
- Route::post();
- Route::put();
- Route::delete();
- Route::any();

Todos esos métodos aceptan los mismos parámetros, por lo que se puede usar cualquier método HTTP que sea apropiado para la situación. Esto es conocido como enrutado RESTful, esto significa que responden a diferentes verbos HTTP. Se usa GET para hacer peticiones, y POST cuando se tiene que mandar datos adicionales con la petición.

El método Route::any() es usado para hacerlo coincidir con cualquier verbo HTTP.

Una característica potente de Laravel son los filtros que se pueden ejecutar antes y después de una petición hecha a la aplicación. Los filtros se añaden en el fichero routes.php.

```
Route::group(array('before' => 'auth'), function()
{
```

**Ilustración 65: Uso del filtro before en una ruta.**

En la ilustración 65 se hace uso del filtro before, el cual comprueba antes de realizar la petición si el usuario esta autentificado.

Otra cosa muy útil es la utilización del método Route::group, se utiliza cuando se encuentran varias rutas a las que se les necesita aplicar el mismo filtro.

Laravel también proporciona una clase Validator, la cual ayuda a validar formularios, modelos de la base de datos, etc. Esta clase permite declarar reglas para los datos de entrada pasados, y según si pasa las reglas mostrar un mensaje. [6]

En esta aplicación se ha usado validaciones para el proceso de registro.

```
Route::post('registro', function(){

    $rules = array(
        'Correo_electronico' => 'required|email|unique:clientes',
        'Password' => 'required|same:confirmarcontraseña|min:5',
        'confirmarcontraseña' => 'required|same:Password'
    );

    $messages = array(
            'required' => 'El campo :attribute es obligatorio.',
            'min' => 'El campo :attribute no puede tener menos de :min carácteres.',
            'email' => 'El campo :attribute debe ser un email válido.',
            'unique' => 'El email ingresado ya existe en la base de datos'
        );

    $validation = Validator::make(Input::all(), $rules, $messages);
    if ($validation->fails())
    {
        return Redirect::to('login')->withErrors( $validation)->withInput();
    }
    else{
    $input = Input::all();
    // al momento de crear el usuario la clave debe ser encriptada
    // para utilizamos la función estática make de la clase Hash
    // esta función encripta el texto para que sea almacenado de manera segura
    $input['Password'] = Hash::make($input['Password']);

    cliente::create($input);

    return Redirect::to('login')->with('mensaje_registro', 'Usuario Registrado');
    }
});
```

**Ilustración 66: Uso de una validación.**

Como se puede ver, se establecen unas reglas de validación, a continuación se crean unos mensajes y por último se llama al método Validator::make que comprueba si los datos de entrada cumplen las reglas establecidas.

Por último, decir que Laravel proporciona un método llamado Hash::make, el cual es utilizado para calcular el código hash de una contraseña, de manera que las contraseñas nunca serán guardadas en plano, si no que se guarda el código hash de esta, proporcionando así una mayor seguridad para el usuario. [6]

A continuación se explican los conceptos de “Ingeniería del software” y “patrones de diseño”, estos son usados por Laravel, ya que Laravel ha sido diseñado para mejorar la calidad del software al reducir tanto el coste de desarrollo inicial y los costos de mantenimiento, y para mejorar la experiencia de trabajar con aplicaciones, proporcionando sintaxis clara y un conjunto básico de funcionalidad que ahorrará horas de tiempo de ejecución.

## *Ingeniería del software*

La ingeniería de software es una disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de los programas informáticos (software).

Esta disciplina trasciende la actividad de programación, que es el pilar fundamental a la hora de crear una aplicación. El ingeniero de software se encarga de toda la gestión del proyecto para que éste se pueda desarrollar en un plazo determinado y con el presupuesto previsto.

La ingeniería de software, por lo tanto, incluye el análisis previo de la situación, el diseño del proyecto, el desarrollo del software, las pruebas necesarias para confirmar su correcto funcionamiento y la implementación del sistema.

Cabe destacar que el proceso de desarrollo de software implica lo que se conoce como ciclo de vida del software, que está formado por cuatro etapas: concepción, elaboración, construcción y transición.

La concepción fija el alcance del proyecto y desarrolla el modelo de negocio; la elaboración define el plan del proyecto, detalla las características y fundamenta la arquitectura; la construcción es el desarrollo del producto; y la transición es la transferencia del producto terminado a los usuarios.

Una vez que se completa este ciclo, entra en juego el mantenimiento del software. Se trata de una fase de esta ingeniería donde se solucionan los errores descubiertos (muchas veces advertidos por los propios usuarios) y se incorporan actualizaciones para hacer frente a los nuevos requisitos. El proceso de mantenimiento incorpora además nuevos desarrollos, para permitir que el software pueda cumplir con una mayor cantidad de tareas.

Un campo directamente relacionado con la ingeniería de software es la arquitectura de sistemas, que consiste en determinar y esquematizar la estructura general del proyecto, diagramando su esqueleto con un grado relativamente alto de especificidad y señalando los distintos componentes que serán necesarios para llevar a cabo el desarrollo, tales como aplicaciones complementarias y bases de datos. Se trata de un punto fundamental del proceso, y es muchas veces la clave del éxito de un producto informático.

En este proyecto se han seguido las siguientes fases:

- Familiarización con las herramientas (SVN, MVC, etc), metodología de trabajo, entorno de trabajo y frameworks (JQuery y Bootstrap, pero sobre todo Laravel).
- Definición de las características deseables de la aplicación. Toma de requisitos y especificación de la idea del proyecto. Acotación de la funcionalidad.
- Definición del modelo (Esquema de tablas y relaciones de la BD), que se utiliza en el proyecto. MySQL Workbench.
- Implementación de la aplicación. Codificación, utilizando Laravel + jQuery + HTML + Bootstrap.
- Pruebas de la aplicación.

## *Patrones de diseño*

El software cambia, para anticiparse a los cambios en los requisitos se tiene que diseñar pensando en qué aspectos pueden cambiar. Los patrones de diseño están orientados al cambio.

Los patrones pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento existente.

Los patrones NO pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

No es obligatorio utilizar los patrones. Es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón. Abusar o forzar el uso de los patrones puede ser un error.

Un antipatrón de diseño es un patrón de diseño que conduce a una mala solución para un problema. Evitar los antipatrones siempre que sea posible, requiere su reconocimiento e identificación dentro del ciclo de vida del software.

Los patrones de diseño son soluciones bien pensadas a problemas conocidos de

programación. Muchos programadores han padecido de estos problemas antes y han utilizado estas “soluciones” para ponerles remedio.

Un ejemplo de un patrón de diseño es el modelo-vista-controlador, utilizado para el desarrollo de la aplicación, ya que divide el diseño en distintas responsabilidades.

## 3.3. Funcionalidad de la aplicación

Cualquier cliente que se dé de alta en la Web, dispondrá de una serie de asistentes (formularios) con una serie de categorías y campos rellenables para definir todos los aspectos de su CV. El servicio estará disponible desde cualquier dispositivo (fijo o móvil) con una conexión a Internet: el diseño de la interfaz es responsive, por tanto.

Una vez rellenado y guardados los datos, podrá exportar en Word su CV adaptándolo a la plantilla seleccionada en cada caso. El cliente podrá descargar esta plantilla tantas veces como desee, ya que sus datos quedarán guardados en la base de datos, por lo que cada vez que acceda a la aplicación tendrá sus datos disponibles.

Como primera versión de los campos y categorías rellenables en la plataforma, se tiene:

- Información personal.
- Formación académica.
- Experiencia profesional.
- Docencia impartida.
- Conocimiento de idiomas.
- Conocimientos en informática.
- Doctorados.
- Participación en proyectos de I+D+I.
- Estancias en centros de I+D+I.
- Experiencia en organización en actividades de I+D+I.
- Ponencias en congresos.
- Dirección de tesis y/o proyectos fin de carrera.
- Propiedad intelectual e industrial. Know-how y secretos industriales.
- Publicaciones, documentos científicos y técnicos.
- Becas.
- Formación extra.
- Otros datos de interés.
- Campos dinámicos.

La aplicación estará alojada de forma íntegra en la nube, con la siguiente estructura:

Dominio principal:

- Alojará la "Web corporativa" de la plataforma, con un diseño atractivo. En resumen, una landing page tipo, con:
  - Vídeo/s explicativos sencillos (ej. landing pages similares: dropbox, no-ip.org, etc.)
  - En líneas generales, en principio, similares a la Web de Dropbox, en apariencia y funcionalidad: sencilla y directa. Minimalista.
  - En un futuro, se orientará el diseño hacia un portal de contenidos que incluya muchos más aspectos de la plataforma

Se enumeran algunas características, que se irán ampliando según necesidades:

- Gestión de datos de clientes (CRUD)
- Gestión de suscripciones de clientes (cuotas, fechas, etc.)
- Gestión del espacio Web del cliente y de la BD (subdominio, credenciales BD, etc.)
- Gestión de módulos del cliente (según cuotas, plantillas de CV pagadas, etc.)
- … y cualquier otra información que sea procedente en la relación entre la empresa y los clientes.

### 3.3.1. Características del sistema (por la parte del cliente)

- Cada cliente dispondrá de una interfaz donde podrá gestionar totalmente cada uno de los ítems de su CV, así como definir nuevos.
- El cliente podrá crear, modificar, obtener y borrar cada ítem de su CV.
- Podrá adquirir plantillas para personalizar la interfaz y obtener el Word de su CV en distintos formatos, respectivamente.

### 3.3.2. Consideraciones técnicas adicionales

Para la exportación de documentos y la creación de plantillas Word se propone la librería opensource de PHP WORD.

De esta manera, se pondrán a disposición del cliente dichas plantillas bajo pedido.

Otras de las ventajas del sistema (o bazas comerciales) que se ofrece es la posibilidad de mantener su CV actualizado en un único portal Web y luego poder exportarlo a múltiples formatos normalizados o personalizados.

### 3.3.3. Manual de usuario

En adelante se entenderá “Los usuarios” como cada uno de los clientes que se den de alta en la Web y, por tanto, disfruten del servicio.

Cuando un usuario acceda a la página de la aplicación lo primero que verá será una landing page, en ella el usuario se podrá registrar en la aplicación o si ya lo está, acceder a la aplicación.

Para iniciar sesión, el usuario deberá introducir su correo electrónico y contraseña.

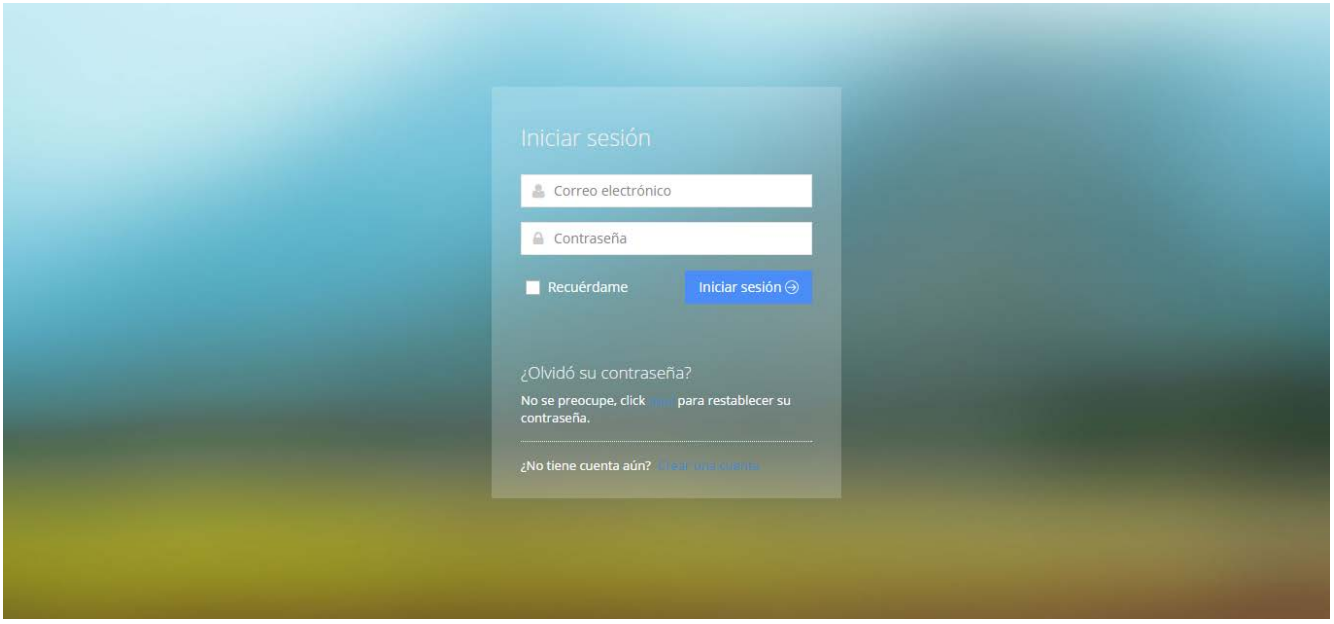



**Ilustración 67: Página de login de la aplicación.**

En esta página el usuario podrá iniciar sesión, registrarse si no lo ha hecho antes o restablecer su contraseña.

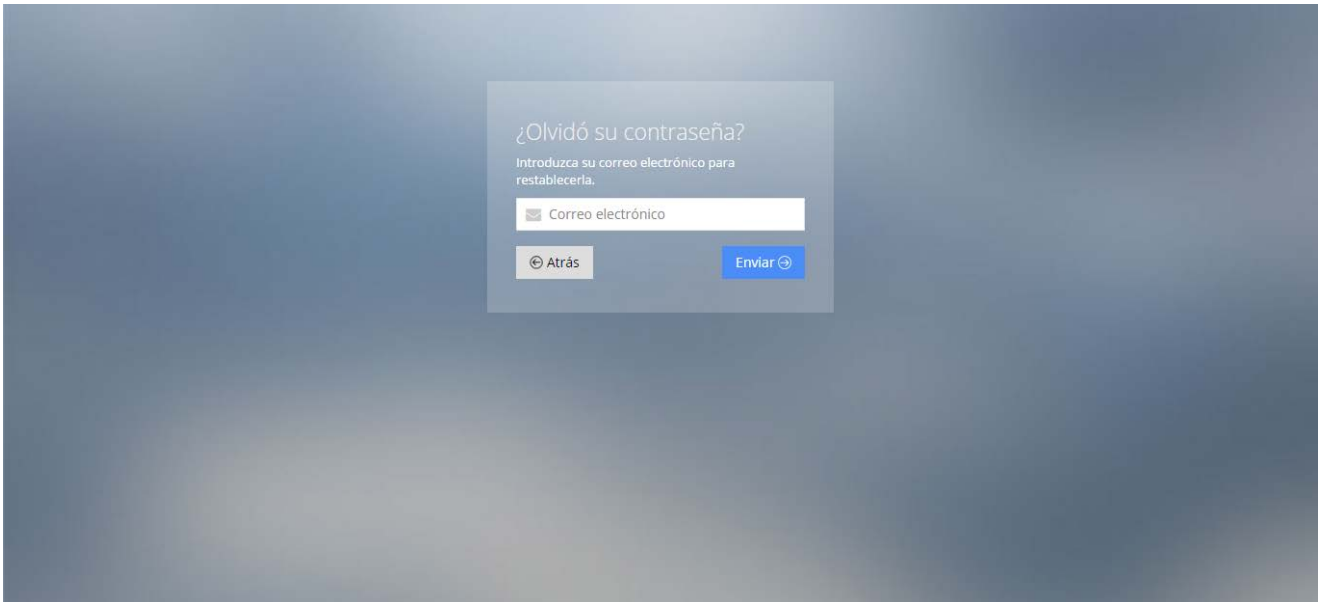



**Ilustración 68: Página de "¿Olvidó su contraseña?" de la aplicación.**

Para restablecer su contraseña, deberá introducir su correo electrónico y se le enviará un mensaje con un enlace para cambiar la contraseña. El enlace llevará al usuario a la página mostrada en la ilustración 69, en esta podrá cambiar la contraseña.

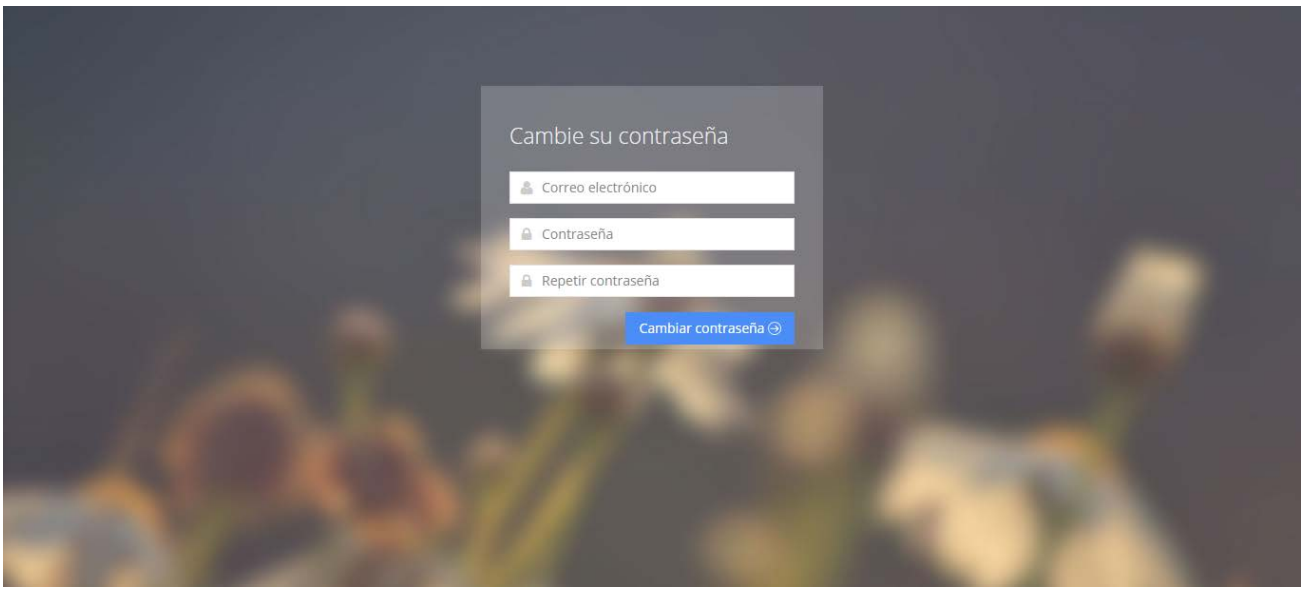



**Ilustración 69: Página para cambiar la contraseña de la aplicación.**

Una vez que el usuario haya accedido a la aplicación llegará a una página de inicio, donde habrá una serie de instrucciones para guiar al usuario en la elaboración de su CV.

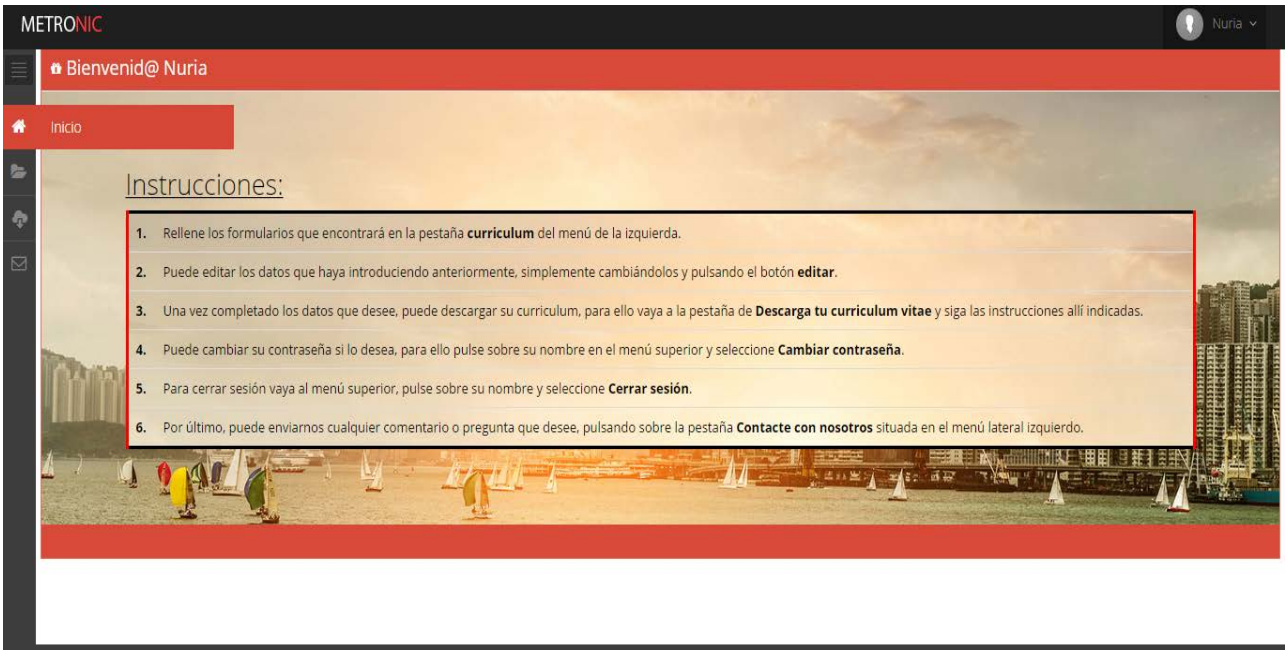



**Ilustración 70: Página de inicio de la aplicación.**

En esta página de inicio, se encontrará con un menú lateral donde podrá seleccionar los distintos formularios disponibles en la aplicación. Cuando el usuario seleccione alguno de los formularios, la aplicación se redireccionará a ese formulario para mostrárselo al usuario, si el formulario seleccionado es el de información personal, el usuario podrá rellenar los campos y guardarlos, y una vez guardados, el usuario verá en el mismo formulario los datos que había introducido, de esta manera el usuario podrá modificar sus datos. Por el contrario, si el usuario selecciona cualquiera de los otros formularios, el usuario podrá rellenarlos como el anterior pero esta vez, cuando el usuario guarde los datos rellenados, la aplicación mostrara los datos en otro formulario debajo del principal, de esta manera el usuario podrá modificar o eliminar sus datos, además de añadir otros nuevos, esto se puede ver en la ilustración 71.

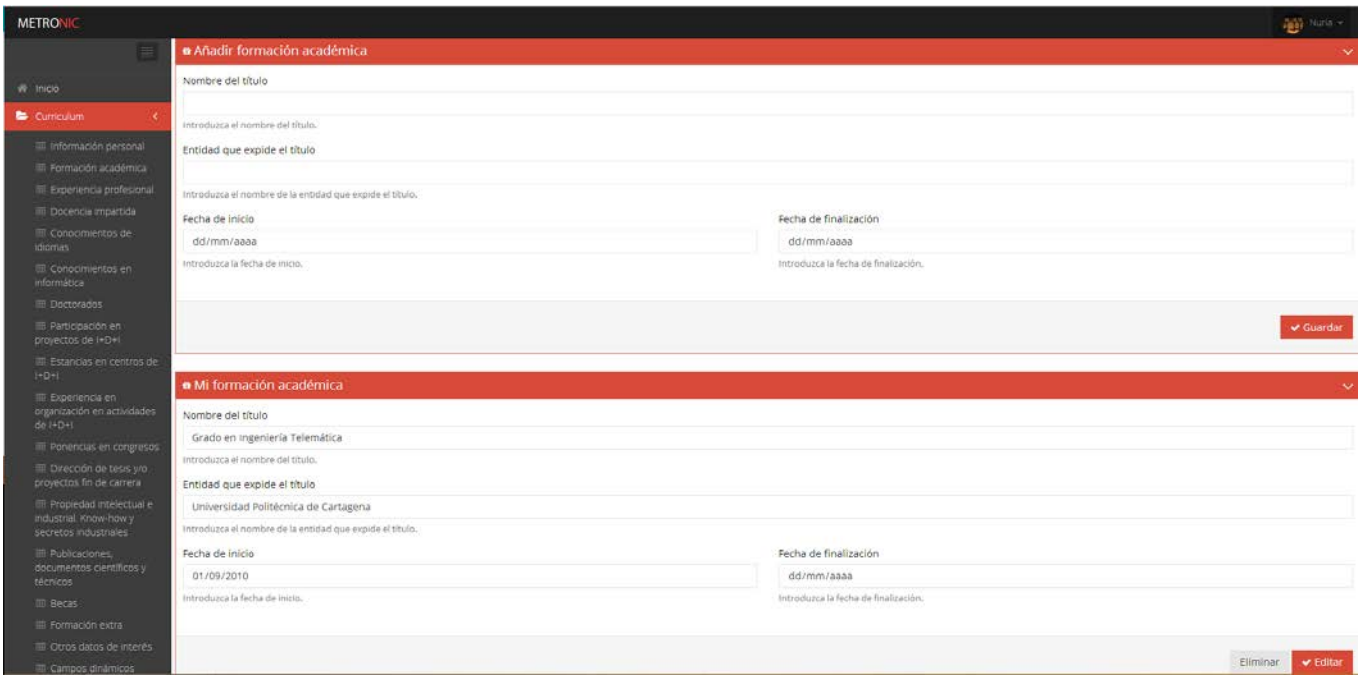



**Ilustración 71: Formulario de formación académica de la aplicación.**

El usuario dispondrá de un formulario para cambiar su contraseña, para ello tendrá que pinchar sobre su nombre en la barra superior de la aplicación, al hacerlo se desplegará un menú en el que tendrá dos opciones, cambiar la contraseña o cerrar la sesión.

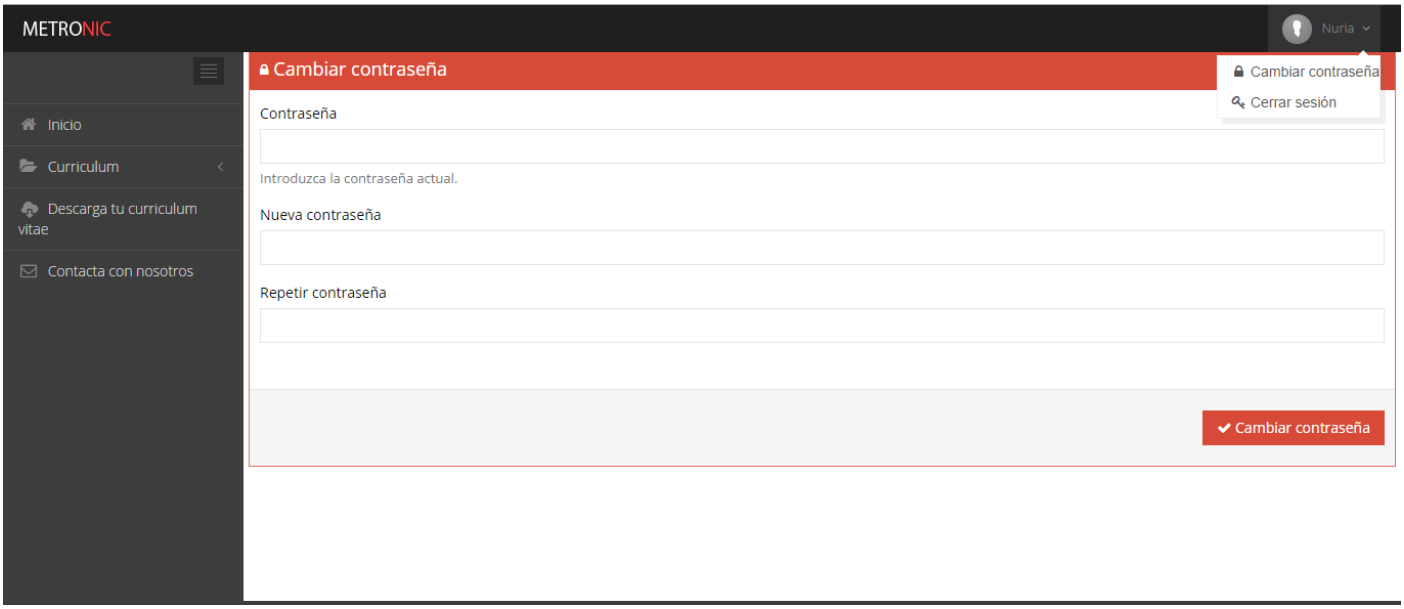



**Ilustración 72: Formulario para cambiar la contraseña de la aplicación.**

Una vez rellenados todos los datos, el cliente puede descargar su curriculum en la pestaña del menú “Descarga tu curriculum vitae”, ahí solo tendrá que pulsar sobre “Plantillas” y elegir la plantilla que desee.

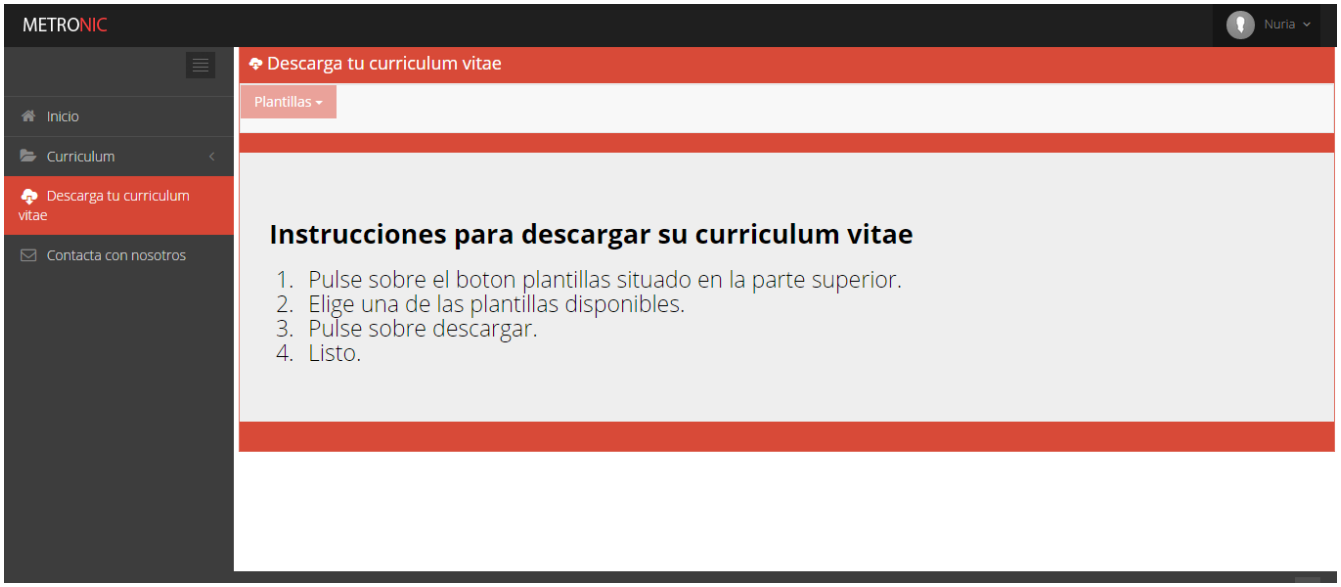



**Ilustración 73: Descarga tu curriculum.**

Además estará disponible un formulario de contacto, en el que el usuario podrá enviar cualquier comentario, sugerencia o duda sobre la aplicación. Para ello el usuario tendrá que seleccionar del menú la opción de “Contacta con nosotros” y rellenar los campos.

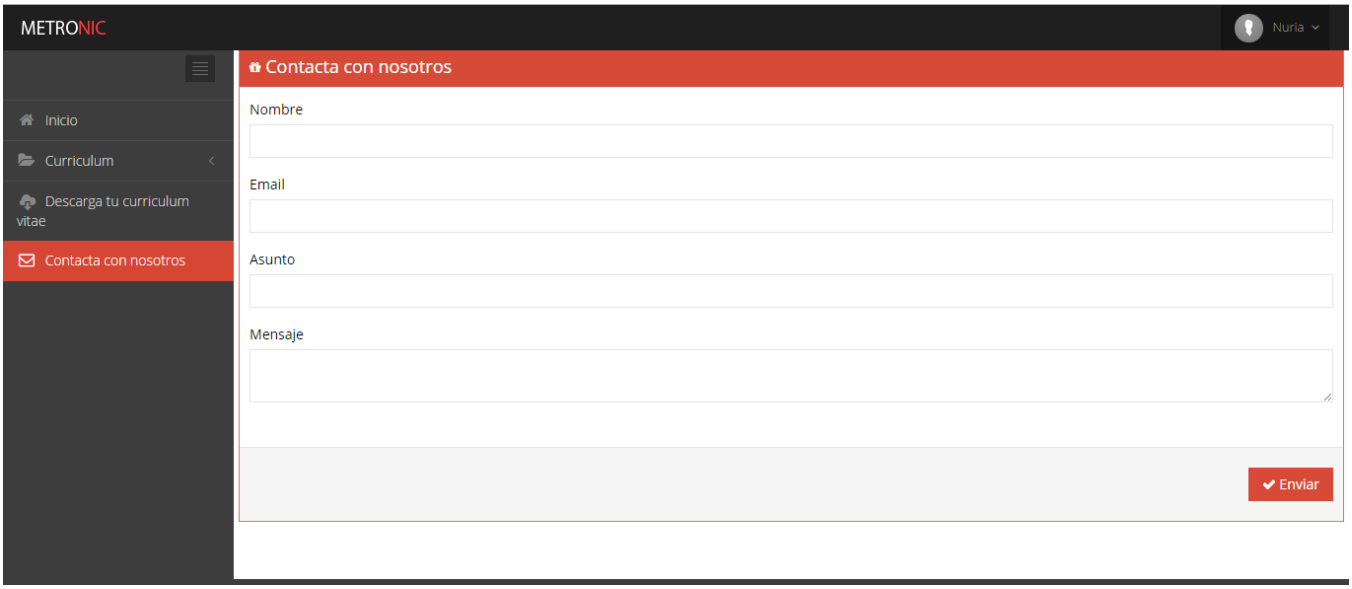



**Ilustración 74: Formulario de contacto de la aplicación.**

Por último, destacar que el menú de la aplicación tiene un botón para poder comprimirlo, de manera que sea más cómodo para el usuario poder rellenar los formularios, teniendo así una visión mayor de la aplicación.

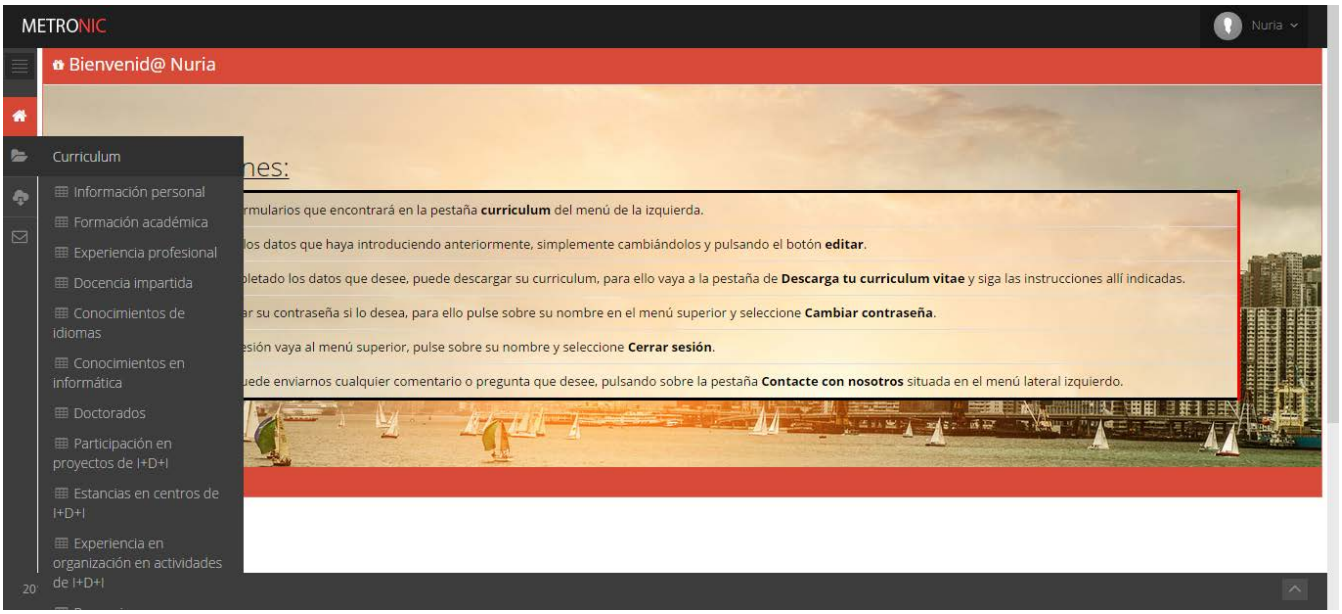



**Ilustración 75: Menú comprimido de la aplicación.**

# 4. Conclusiones y líneas futuras

## 4.1. Conclusión

En este documento se ha identificado los problemas comunes de los usuarios en el área de las TIC de desarrollo web, mediante el estudio del arte y la gran documentación existente se ha podido lograr la finalidad de este proyecto, conseguir diseñar y desarrollar una aplicación web alojada en la nube y de tipo responsive, es decir, todo lo desarrollado en el apartado de solución propuesta.

Se ha prestado mucha atención en conseguir que el usuario este "cómodo" en todo momento en el mismo instante de acceder a la aplicación, por lo que se ha diseñado la aplicación teniendo en cuenta la apariencia y la comodidad, se han elegido colores agradables a la vista y se han creado textos de ayuda con instrucciones para el usuario, además se ha creado un formulario de contacto, para que además de que el usuario cuente su experiencia en la aplicación, pregunte dudas o inconvenientes que pudiera tener en el desarrollo de su CV. El resultado de todo esto es una aplicación que facilita al usuario la creación de su curriculum vitae.

Cabe destacar el framework Laravel, ya que ha sido con esta herramienta con la que se ha conseguido el desarrollo de la mayor parte de la aplicación. Este potente framework, al principio un poco complicado de entender, pero después fácil de utilizar, es uno de los frameworks más completos que existen, ya que tiene funciones para hacer todo lo que se desee y de una manera sencilla, además cuenta con una extensa documentación y foros en los que te pueden ayudar, en poco tiempo, a resolver cualquier problema.

Finalmente, se ha llegado a la conclusión de que una aplicación desarrollada con Laravel y alojada en la nube es la solución más eficiente, ya que la web es el entorno más potente para la creación de una aplicación de esta índole, ya que esta siempre disponible y es de fácil acceso.

## 4.2. Líneas futuras

Aunque las características esbozadas hasta el momento dejan abierta la posibilidad de expandir el sistema en infinitas direcciones, a continuación se listan una serie de posibles extras a nivel operativo para expandir el modelo de negocio aún más:

- Creación de una APP para Android o iOS que se conecte de forma remota a la base de datos y ofrezca las características de exportación de documentos y edición que brinda la Web.
- Implantación de una fuerte imagen de compañía especializada en el tema. Publicidad en adwords para acelerar la implantación del servicio.

# 5. Bibliografía y referencias

[1] Jack Vo. (2014). Learning Laravel: The Easiest Way.

[2] Hardik Dangar. (2013). Learning Laravel 4 Application Development.

[3] Dayle Rees & Antonio Laguna. (2013). Laravel: Code Happy (ES) Desarrollo de aplicaciones con el Framework de PHP Laravel para principiantes.

[4] Shawn McCool. (2012). Laravel Starter.

[5] Terry Matula. (2013). Laravel Application Development Cookbook.

[6] Raphaël Saunier. (2014). Getting Started with Laravel 4.

[7] Dayle Rees y Antonio Laguna. (2013). Laravel: Code Bright (ES) Desarrollo de aplicaciones web con la versión 4 del framework Laravel para principiantes.