Trabajo de Fin de Grado

Herramienta de automatización de código para la gestión de aplicaciones web

Code automation tool for web applications management

Eduardo Jesús Díaz Dioniz

La Laguna, 4 de marzo de 2015

D. **Casiano Rodríguez León**, con N.I.F. 42.020.072S profesor Catedrático de Universidad adscrito al Departamento de Lenguajes y Sistemas Informáticos de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Herramienta de automatización de código para la gestión de aplicaciones web"

ha sido realizada bajo su dirección por D. **Eduardo Jesús Díaz Dioniz**, con N.I.F. 42.237.052-J.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 4 de marzo de 2015.

Agradecimientos

A Casiano Rodríguez León por dedicar su tiempo y esfuerzo en guiarme, asesorarme y motivarme durante mi trayectoria académica y especialmente, en el desarrollo de este Trabajo de Fin de Grado.

A mi madre por enseñarme el valor de la constancia y el esfuerzo, siendo todo un ejemplo a seguir.

A mi novia por apoyarme, alentarme y ser partícipe en todas mis aventuras.

A mis compañeros de la universidad y especialmente a mi primo. Las largas horas de estudio se hicieron mucho más amenas en vuestra compañía.

Y por último agradecer a los profesores de la Universidad de La Laguna su gran implicación y preocupación por conseguir una promoción cualificada para comenzar con impulso esta nueva etapa.

Licencia

Resumen

El objetivo de este Trabajo de Fin de Grado ha sido la integración de los conocimientos en los estudios de Ingeniería Informática acercando al alumno el desarrollo de soluciones web y la resolución de problemas computacionales fomentando el desarrollo de las destrezas adquiridas en los estudios de Grado.

En este Trabajo de Fin de Grado se propone el desarrollo de una aplicación que permita a los usuarios generar sus propias páginas web. Existe un gran número de desarrolladores y usuarios que necesitan obtener una solución rápida para visualizar sus datos o simplemente hacer operaciones básicas. Por ello, la aplicación tiene como objetivo abarcar todos esos problemas y darles una solución en la que cualquier usuario pueda acceder, crear su aplicación , subir o enlazar sus datos y tener una página web simple que le ofrezca una solución rápida.

En nuestra propuesta tenemos como objetivo la rapidez y la simpleza. Este objetivo nos ha condicionado en muchas de las decisiones que hemos tomado a lo largo del proyecto, entendiendo con esto que la creación de las páginas será lo más rápida posible, reduciendo las configuraciones complejas pero guardando siempre las medidas de seguridad necesarias.

Palabras clave: Generador de aplicaciones, Manipulación de datos, Gestión de usuarios, Aplicaciones web, PHP, HTML5, JavaScript.

Abstract

The aim of this end of degree work has been the integration of the knowledge acquired during the studies of Computer Engineering, in the context of web development.

We develop an application that allows users to generate their own web pages. There are a number of developers and users who need a quick solution to visualize their data and do simple operations on them. This application aims to give a solution that generates a simple website from the description provided by the user.

In our proposal we aim speed and simplicity. This objective has conditioned us in many of the decisions we made throughout the project, aiming the rapid creation of the pages, the reduction of complex configurations while keeping the necessary security measures.

Keywords: *Application Builder , Data manipulation , user management, web applications , PHP , HTML5 , JavaScript .*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1.

Introducción

Este capítulo recoge la descripción del problema definiendo los objetivos, el alcance y los antecedentes del mismo.

1.1 Antecedentes.

Actualmente el mundo de la informática está en auge. Los desarrollos tanto en dispositivos móviles, aplicaciones web o de escritorio, han crecido de forma descomunal y actualmente ya no concebimos el mundo sin esta tecnología. Muchos desarrolladores programan aplicaciones que suben a tiendas oficiales y se convierten en virales en poco tiempo, o las publicitan por redes sociales llegando a un target más amplio en un plazo de tiempo muy breve. Esto ha provocado que cualquier aplicación que necesite manipular datos de usuarios para su correcto funcionamiento, pueda volverse inabarcable en muy poco tiempo y que se necesite programar una aplicación para poder administrarlos. De ahí surge la idea principal de nuestro TFG, apoyar a los desarrolladores o a los usuarios cuando necesiten hacer aplicaciones simples para manipular datos.

Estudiando los proyectos que hay creados no hemos encontrado nada similar que cumpla con los criterios que hemos establecido. Existen soluciones parecidas como módulos de drupal y algunos para PHP pero estás más orientados al mantenimiento físico de los servidores y no del mantenimiento de los datos lógicos del usuario. Por otro lado estudiamos el framework koala para tener una idea de cómo visualizar los datos de los usuarios en la web.

1.2 Objetivo.

El objetivo del proyecto es crear una aplicación base que permita a los usuarios crear componentes o usarlos de otro desarrollador que permita solventar los problemas específicos que se tengan a la hora de manipular o gestionar datos.

Cualquier usuario podrá acceder a su cuenta autenticandose con su correo electrónico a través de los medios disponibles y tendrá la posibilidad de crear aplicaciones o usar las páginas que le hayan sido designadas por administradores o propietarios de la aplicación.

Cada usuario tendrá a su disposición todas sus fuentes de datos para crear las consultas que necesite, luego podrá asignarlas a las aplicaciones, permitiendo esto que cuando creemos las páginas sean seleccionables desde los componentes que pueden coger esos datos.

Los propietarios de las aplicaciones podrán designar a administradores que les ayuden en la tarea de crear páginas, por ello, los recursos, los usuarios y los datos serán asignados a la aplicación. Esto permitirá que los administradores puedan crear una página con usuarios o datos añadidos por otros administradores respetando la seguridad y la integridad de los datos.

1.3 Alcance.

La aplicación desarrollada pretende conseguir objetivos muy ambiciosos permitiendo realizar cualquier tipo de aplicaciones y ofrecer un abanico muy amplio de soluciones, sin embargo, al estudiar la viabilidad del trabajo de fin de grado nos damos cuenta de que no es posible abarcar la versión final del producto en los plazos y el volumen que exige la asignatura. Por ello planteamos crear la aplicación que gestione todo el flujo de trabajo y fundamentar las bases, los estándares y los protocolo a seguir en la continuación del trabajo.

Una vez estudiado el volumen de trabajo definimos varias fases del proyecto que detallaremos a continuación:

- Aplicación base: Se desarrollará la aplicación y se estipulara como se han de crear los componentes para que cualquier usuario con conocimiento pueda diseñarlos y crearlos siguiendo las herramientas planteadas. Esta fase es la que trabajaremos en este trabajo y dejaremos planteada las posibles líneas de investigación futuras.
- Componentes adicionales: En esta fase se desarrollarán componentes específicos que resuelvan problemas puntuales siguiendo los estándares definidos en la fase anterior.

1.4 Destinatarios.

Este proyecto va dirigido principalmente a desarrolladores que no posean tiempo o recursos suficientes como para crear aplicaciones que gestionen sus datos. También va dirigido a usuarios sin conocimientos en el desarrollo de aplicaciones que necesiten administrar y gestionar datos.

Además de los perfiles anteriormente citados incluimos cualquier usuario que necesite cubrir una necesidad de visualización o manipulación de datos y lo necesite lo más rápido posible y destinando los menores recursos posibles.

1.5 Tecnologías usadas.

Este trabajo de fin de grado ha sido realizado con tecnologías web siguiendo las estrategias y metodologías necesarias mejorar el acoplamiento de la aplicación. A continuación se enumeran las tecnologías usadas y sus propietarios reconociendo su autoría e indicando un enlace donde poder consultar su documentación y licencias:

HTML5 - World Wide Web Consortium.
https://www.w3.org/TR/html5/

CSS3 - World Wide Web Consortium.
https://www.w3.org/TR/css-2015/

JavaScript - Oracle Corporation.
https://www.oracle.com/es/

Bootstrap - Twitter Inc.
http://getbootstrap.com/

PHP - PHP Group.
http://www.php.net/

Apache HTTP Server Project - The Apache Software Foundation.
https://httpd.apache.org/

JQuery - “jQuery Foundation”.
https://jquery.com/

Medoo.in - Angel Lai.
http://medoo.in/

Google Sign-In - Alphabet Inc.
https://developers.google.com/identity/

Facebook Login - Facebook Inc.
https://developers.facebook.com/

Twitter OAuth - Twitter Inc.
https://dev.twitter.com/oauth

GitHub OAuth - Github Inc.
https://developer.github.com/v3/

DNI electrónico - Cuerpo Nacional de Policía.
http://www.dnielectronico.es/

Capítulo 2.

Diseño y configuración del sistema

1. Diseño del sistema.

Analizando el problema proponemos desarrollar la aplicación, distribuyendo el trabajo en subsistemas que funcionan de forma autónoma y están vinculados entre sí. A continuación mostraremos los diagramas de flujos de datos que se usaron para el diseño y explicaremos las decisiones tomadas.

Una persona puede poseer varios roles. Un propietario es el usuario que tiene la capacidad de crear aplicaciones, por defecto todos los usuarios pueden ser propietarios de alguna aplicación. Cuando estos propietarios necesiten ayuda para crear o gestionar las páginas que creen, pueden añadir administradores que tendrán el mismo papel que el propietario pero con restricciones a la hora de eliminar la aplicación. Los usuarios son las personas que han sido designadas en las páginas para que las puedan usar.

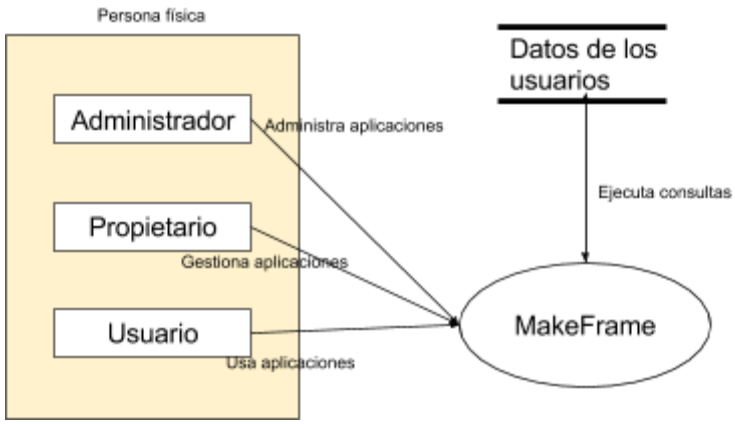

Figura 1. DFD nivel 0 o de contexto

Una vez detallado el diagrama de contexto pasaremos a ver el DFD de nivel 1 o diagrama de nivel superior. En este diagrama podemos observar el flujo de información con el que se interconectan los diferentes subsistemas y posteriormente los desglosaremos.

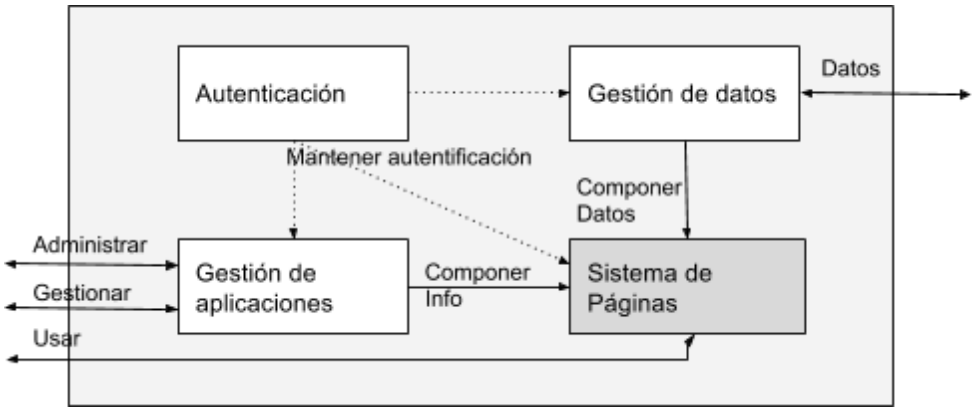

Figura 2. DFD nivel 1 o de nivel superior

La metodología que hemos usado en el desarrollo de la aplicación ha sido un modelo vista controlador. El principal motivo de este desarrollo es conseguir que el servidor funcione de forma autónoma sin importar la interfaz que se quiera aplicar, consiguiendo con esto que si en una vía de investigación futura se desea modificar la interfaz o programar otra, como por ejemplo una aplicación en android, se podrá hacer sin modificar el código común.

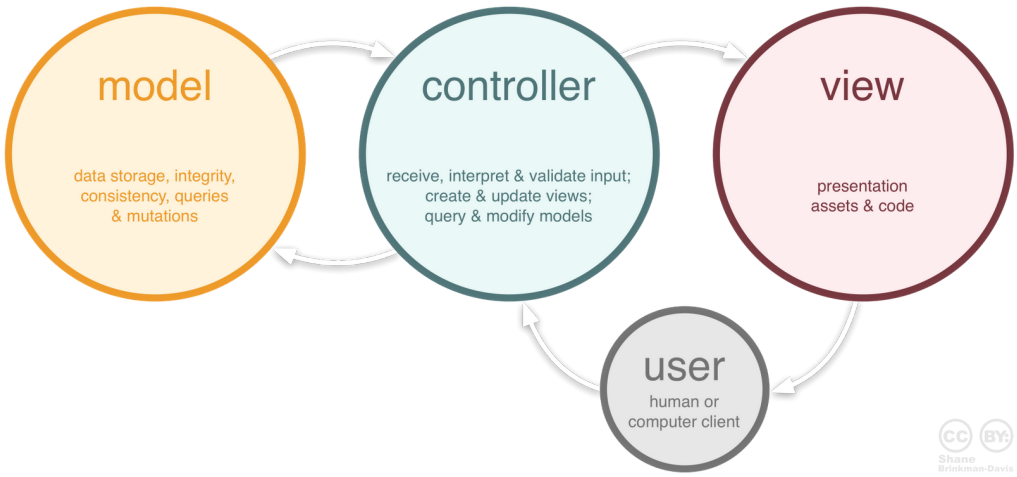

Figura 3. Ejemplo del MVC

Para comunicar la vista con el controlador hemos diseñado un lenguaje común basado en un array en formato json donde se incluyen dos campos:

- -El campo ‘data‘ que incluirá toda la información necesaria por la vista.
- El campo ccd o ‘código de comunicación directa’ el cual nos sirve para comunicar a la vista la acción necesaria. Para poner un ejemplo, a la hora de intentar acceder al sistema, el ccd 250 indica que el servidor permitió el paso al usuario y el ccd 251 indica a la vista que hubo un problema en el intento de acceso. Con estos datos, la vista ya tiene suficiente información para tomar una decisión u otra.

Para el diseño de la base de datos del sistema seguimos el siguiente esquema el cual fue realizado tras un análisis exhaustivo de los requisitos.

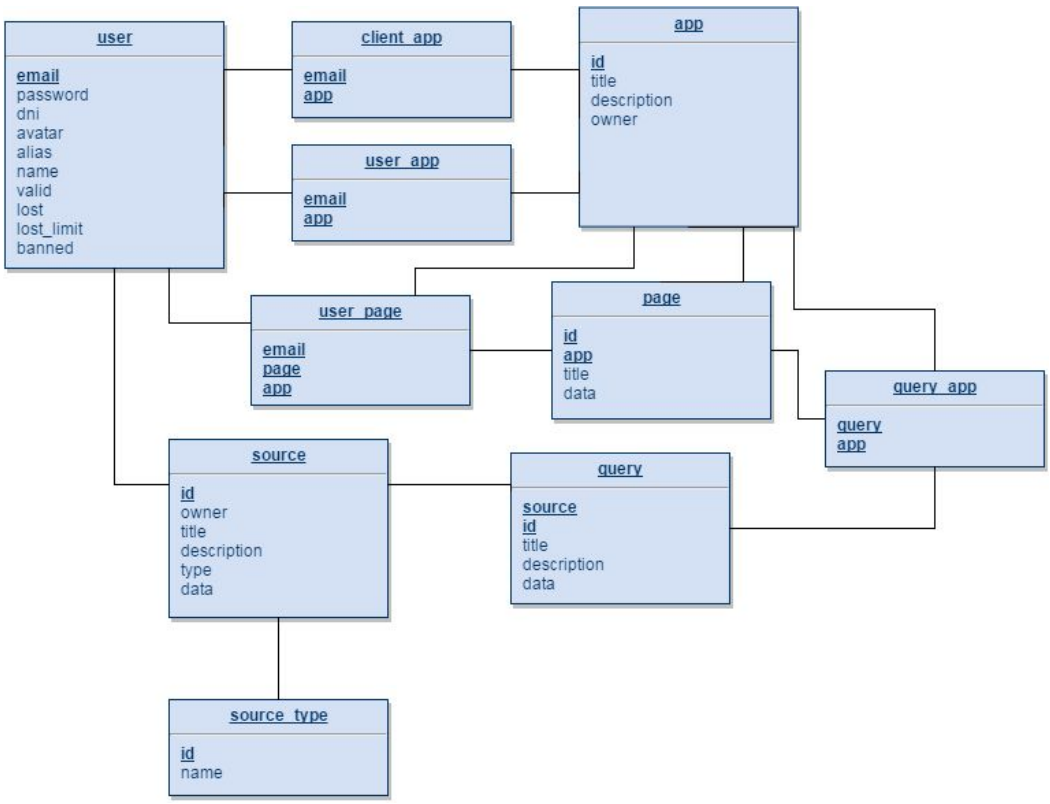

Figura 4. Diagrama de la base de datos

1.1. Autenticación del sistema.

Uno de los principios fundamentales del sistema es su seguridad en la autenticación. No podemos exigir que los usuarios usen métodos externos únicamente, ya que obligaríamos a que cedieran sus datos a terceros, por ello, es fundamental proveer de un servicio de autenticación por medio de contraseña en el que únicamente compartan esas datos con nosotros. Además de esto, para mejorar la rapidez en el acceso, hemos optado por añadir una autenticación por oauth. Este método es usado por múltiples empresas que poseen sistemas que identifican a usuarios y permiten que aplicaciones externas les deleguen esa parte. Para finalizar, añadiremos un método de identificación con certificados digitales, en concreto, el DNI electrónico.

1.2. Gestión de los datos

El principal pilar de la aplicación es el enlace que ofrece con las fuentes de datos externas. La gestión de los datos se encarga de probar las conexiones, entablar comunicación con ficheros o bases de datos y extraer la información de una manera comprensible por el sistema. Para realizar esa tarea se apoya en el framework medoo.in permitiendole el acceso a cualquier base de datos que el usuario introduzca. Además de eso, permitimos enlazar a ficheros, aunque por motivos de seguridad e integridad de los datos solo permitimos consultas sobre ellos.

Uno de los principales problemas que tuvimos con los datos, fué como hacer que los usuarios pudieran compartir sus datos con los administradores de una aplicación para que éstos pudieran usarlos sin incumplir unas medidas de seguridad mínimas. Para ello añadimos en la configuración de la aplicación la opción de añadir todos los datos que se usarán dentro. Solo el propietario de los datos podrá modificarlos pero todos los administradores de la aplicación, podrán usarlos si están añadidos como datos de la aplicación. Cuando nos referimos al sistema de datos lo hemos dividido en dos subsistemas, el primero es cuando realizamos una conexión a una base de datos o fichero y el segundo, cuando hacemos una consulta ya sea inserción o actualización. Este sistema nos permite tener una capa de abstracción con las bases de datos.

1.3. Gestión de aplicaciones.

La principal actividad que se le brinda al usuario es el uso de las aplicaciones. El subsistema de la gestión de las aplicaciones se encarga de ofrecer al usuario la posibilidad de crearlas, asignarles los recursos que necesiten e incluso añadirle administradores para que ayuden al propietario en la labor de hacer esas aplicaciones. Otra de las principales funciones del gestor de aplicaciones es controlar los permisos en las aplicaciones y las páginas. Solo el propietario y excepcionalmente los administradores tienen permiso para modificar páginas que están en una aplicación.

1.4. Sistema de Páginas

El sistema de páginas viene compuesto por la gestión de las páginas (usuarios que la usan, componentes, ...) y la generación de las

páginas. Esta última se encarga de recibir el JSON estandarizado, que se encuentra en el anexo A, del navegador del cliente y convertirlo en datos html que puedan ser visitables por un usuario.

La gestión de las páginas era una parte fundamental para garantizar la seguridad en la plataforma. Este sistema debe controlar los usuarios que pueden ver o usar la aplicación ya que ahí estarán los datos sensibles del usuario.

La generación de la página también debe ser controlada para que los usuario que no tenga permisos no puedan manipular las páginas en las que no sean propietarios o administradores.

2. Configuración del sistema

Para configurar la aplicación será necesario contar con un sistema en el que debemos tener una configuración y servicios mínimos. A continuación detallamos un listado de los servicios:

- Servidor Apache
- Servidor PHP
- Base de datos MySQL.
- Servidor de correo electrónico.

1. Configuración del apache.

Para la configuración del servidor apache podemos seguir una de las múltiples guías que existen por internet. Nosotros hemos recopilado información de diversos sitios y hemos usado el comando apt. Una vez ejecutado el proceso de instalación, este nos guía para configurar el servidor.

En nuestra aplicación es necesario configurar varios virtualhost para el acceso por http, https y una última para el acceso con certificados electrónicos la cual detallaremos en el capítulo 3.

2. Configuración de la Base de datos.

Para la configuración de la base de datos será necesario instalar el paquete **mysql-server**. En las versiones nuevas se ejecuta un instalador el cual nos guía a la hora de instalarlo y configurarlo (contraseñas de acceso, puertos, etc).

Una vez instalado accedemos a la base de datos y ejecutamos el script que se adjunta con el código del sistema para crear la base de datos.

Modificaremos el fichero de configuración para el acceso a la base de datos con los datos correctos de nuestro sistema.

core/database/data/MakeFrameDB.php

```
$MakeFrameDB_connection = [
      'type'                    => Tipo de la base de datos,
      'name'                    => Nombre de la base de datos,
      'url'                     => Dirección de la base de datos,
      'user'                    => Usuario de la base de datos,
      'pass'                    => Contraseña de la base de datos,
      'charset'                 => Codificación de caracteres,
      'port'                    => Puerto por el que conectar,
      'prefix'                  => Si las tablas necesitan un
                                   prefijo
];
```

3. **Configuración del PHP.**

Para la configuración del servidor PHP ejecutaremos los siguientes comandos. El primer comando es el de instalación del servidor PHP.

```
sudo apt-get install libapache2-mod-php5
```

A continuación como necesitamos que el servidor PHP pueda mantener la conexión a MySQL instalaremos el complemento necesario.

```
sudo apt-get install php5-mysql
```

4. **Configuración de servidor de correo.**

Para que la aplicación funcione correctamente es necesario mantener un servidor de correo electrónico. Nosotros escogimos usar un servidor postfix. Cuando los usuarios pierden la contraseña es necesario que el servidor PHP use al de correos electrónicos para enviar las notificaciones pertinentes. Para instalar el servidor nos guiamos por el siguiente enlace que nos provee una página que ofrece servicios digitales.

https://www.digitalocean.com/community/tutorials/how-to-install-and-setup-postfix-on-ubuntu-14-04

El primer paso para instalar el servidor es el siguiente comando:

sudo apt-get install postfix.

A continuación en la guía se puede ver todas las configuraciones posibles y los comandos que nos ofrece el servidor. En nuestro caso no es

necesario realizar ningún tipo de configuración inicial ya que la configuración por defecto es más que suficiente.

NOTA: En el caso de que el servidor PHP no esté configurado para soportar el servidor de correo recomendamos realizar las siguientes configuraciones en el php.ini (Archivo de configuración del servidor).

php.ini.

```
...
SMTP = dirección del servidor de correo
...
```

Capítulo 3.

Desarrollo del proyecto.

Para el desarrollo del proyecto se han marcado múltiples hitos conformes con los subsistemas que crean la plataforma. A continuación detallaremos el desarrollo de esos subsistemas, los problemas encontrados y las decisiones tomadas.

1. Métodos de autentificación.

La autenticación en el sistema era uno de los principales pilares de la plataforma ya que contaremos con datos sensibles de usuarios a los que les debemos dar la mayor protección posible. Es por esto que hemos creado un abanico de métodos de autentificación para que el usuario use el que considere necesario.

El primer método que creamos es la autenticación por medio de contraseña. Para desarrollar este método solo fué necesario añadir unos campos en la tabla de usuario en la base de datos:

Campo contraseña: Por medidas de protección no almacenaremos la contraseña del usuario sino que usaremos un algoritmo de reducción criptográfico de 128 bits para hacer hash o resumen de la contraseña. En concreto usaremos el proporcionado por el servidor php (MD5). En vez de comparar las contraseñas de una forma clásica compararemos los resúmenes únicos.

Campo lost: En este campo almacenaremos un identificador único de la cuenta del usuario cuando el usuario notifique una pérdida de las credenciales de acceso. Este identificador se enviará por correo con un enlace debidamente construido. Cuando el usuario visite ese enlace el servidor entenderá que se ha autentificado en su correo electrónico y confiará en ese otro método para acreditar la identidad del usuario ofreciéndole la posibilidad del cambio de contraseña.

Campo lost_limit: Donde limitaremos en 20 minutos el tiempo en el que se permite el cambio de contraseña. Esta opción la

realizaremos por seguridad. Entendiendo más que suficiente 20 minutos para que el correo llegue a su destinatario y insuficiente para realizar cualquier técnica fraudulenta.

Al usuario introducir la contraseña existe una estructura la cual contabiliza los intentos de acceso, las direcciones ip de origen y toda la información proveniente del navegador. Esta información solo queda almacenada en esta estructura de información pudiendo ser consultada desde el interior de la aplicación pero no se almacena ni se suministra por motivos de confidencialidad y anonimato en el acceso. La idea fundamental es dar a los desarrolladores utilidades para mejorar la aplicación pero no almacenar información redundante.

Cuando un usuario intenta acceder al servidor se crea un objeto de sesión en el que se registra toda la información respecto a los intentos de acceso. A continuación explicaremos los métodos que ofrece el objeto de sesión para controlar el acceso.

```
El objeto añade al registro un intento de acceso comprobando si el
usuario está baneado o que no está logueado.
function addTry(){
	$txt = "ERR";
	$this->last = date('l jS \of F Y h:i:s A');
	if($this->passed == false && $this->banned == null){
		$this->passed_access_attempts++;
		$txt = "TRY";
	Si el usuario está baneado se guarda otro tipo de registro que
	detalla los intentos de acceso por el usuario baneado. Esto se
	hace con el fin de analizar  el comportamiento de acceso para
	poder bloquear intentos de acceso extraños.
	}else if($this->passed == false && $this->banned != null){
		$this->banned_access_attempts++;
		$txt = "TBN";
	}
	array_push($this->data, $this->parse($txt));
}

Esta función añade en el objeto sesión que el usuario se ha
autentificado correctamente.
function addPas(){
	$this->last = date('l jS \of F Y h:i:s A');
	$this->passed = true;
	array_push($this->data, $this->parse('PAS'));
}

Función que añade un bloqueo sobre el usuario
function addBan(){
	$this->last = date('l jS \of F Y h:i:s A');
	$this->banned = date("Y-m-d H:i:s", strtotime("+".
```

```
        El bloqueo se añadirá hasta la hora establecida en el MANIFEST

        Manifest::AUTH_TIMEOUT_AFTER_OVERCOMMING_ATTEMPTS_ALLOWED));
        array_push($this->data, $this->parse('BAN'));
}
```

Cuando se hace uso de este objeto como por ejemplo en los intentos de acceso por medio de una contraseña podemos observar el registro que genera toda la información del usuario con el fin de mantener un registro de seguridad o para cualquier otro posible uso.

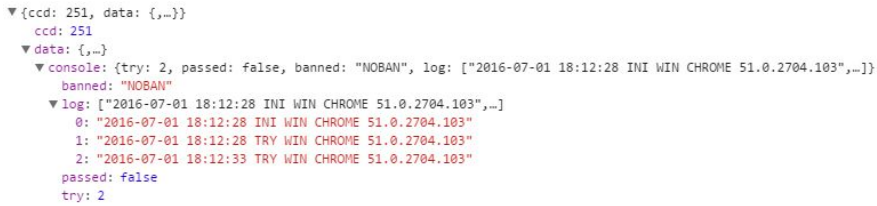

Figura 5. Ejemplo de respuesta al intentar loguearse

El segundo método que desarrollamos es la autenticación externa. Para desarrollar los métodos externos definimos un sistema en el cual se realiza una redirección hacia la plataforma deseada con toda la información necesaria, a partir de ahí esperamos a que el servicio intercambie la información necesaria y nos envíe nuevamente mediante redirección al usuario con su token. Una vez tenemos el token nosotros intercambiamos con los servicios externos toda la información que necesitemos. Todos los servicios nos ofrecen multitud de datos personales, fotografías y mucho más pero nosotros solo usaremos el email como identificador único siguiendo el mismo principio que nos hemos propuesto de usar solo la información necesaria dejando posibilidades para el desarrollo.

Empezamos por Google Sign-In la cual cuenta con una consola en la que un usuario puede registrar una aplicación y desde esa terminal puede monitorizarla y muchas funciones extra que no nos interesaron. Activamos la opción de autentificación y proveimos a google de toda la información que necesitaba de nuestra página.

Una vez el servicio estaba activo lo desarrollamos siguiendo las pautas marcadas por google para el lenguaje que estamos usando. En el siguiente enlace se muestra los pasos y el código para el uso de sus servicios

https://developers.google.com/+/web/samples/php .

El segundo servicio externo que añadimos es Twitter API y como el anterior accedimos a la página de twitter destinada para los desarrolladores y allí le dimos toda la información necesaria para crear una aplicación. Usamos la misma dinámica creada para google y seguimos los pasos indicados en la siguiente guia: https://dev.twitter.com/web/sign-in/implementing.

El código fue obtenido del siguiente enlace https://github.com/abraham/twitteroauth como base para poder ajustarlo a lo que necesitabamos. Una vez desarrollamos el servicio por twitter observamos que no era posible obtener el email de los usuarios que se autenticaban. Esto supuso un problema muy grave ya que nuestra única clave que identifica un usuario es el email. A partir de ahí nos pusimos en contacto con twitter y enviamos información adicional a través de un formulario en el apartado de desarrolladores. Trás unos días aceptaron la petición y nos concedieron permisos extra para obtener el email de los usuarios autentificados en twitter.

El tercer servicio que añadimos es Facebook. Como todos los anteriores tienen página de desarrolladores donde indicar todos los datos de tu proyecto, en este caso mucho más extensa ya que te permite crear páginas de facebook y hacer campañas de marketing con el producto creado. Como guía para el desarrollo y para poder entender la comunicación con facebook se ha seguido el siguiente enlace oficial https://developers.facebook.com/docs/reference/php del que se ha obtenido tanto fracciones de código como ejemplos.

El cuarto servicio que añadimos es GitHub. En la página de desarrolladores añadimos información de la página pero en este caso todos los ejemplos y soluciones que buscamos no conseguiamos que funcionaran con las versiones de los sistemas que teníamos ya configurados y tomamos la desición de hacer de forma manual las peticiones a github. Para esto usamos las librerías de curl incluidas en php y estudiamos las peticiones que debíamos realizar del siguiente enlace.

A continuación detallamos el estudio de las peticiones realizadas a los servidores de github:

```
"https://github.com/login/oauth/authorize?client_id=".$GITHUB_APP_KEY."&sco
pe=user:email";
```

Obtener el token del cliente con nuestra app key teniendo en cuenta el ámbito de los datos que queremos, con incluir user:email nos vale.

```
"https://github.com/login/oauth/access_token";
```

Cuando el usuario llega a nuestra web desde github debemos realizar una petición a la siguiente ruta con todos los datos de nuestra app por método post para recibir el token que nos permite recuperar la información del usuario.

```
"https://api.github.com/user?access_token=".token;
```

De esta petición obtenemos la información del usuario (Nombre, imágen, etc) a pesar de recuperarla no la usamos siguiendo la política que hemos establecido en este TFG.

```
"https://api.github.com/user/emails?access_token=".token;
```

De esta petición obtenemos los emails que hay asociados a la cuenta de github. Debido a que la cuenta github te permite tener asociados varios emails y nos los devuelven todos solo cogemos el que tiene el usuario marcado como primario ya que ellos lo toman como el email de la cuenta y solo puede haber un email primario asociado.

El tercer método de autentificación que desarrollamos fue el uso de certificados electrónicos, concretamente, escogimos usar DNIe ya que principalmente el ámbito de la aplicación será nacional. Para poder hacer uso de este sistema el usuario deberá poseer DNI 2.0 o superior con el certificado electrónico en vigor, un lector de tarjetas inteligentes que cumpla el estándar ISO-7816 y el navegador configurado correctamente. Para configurar el navegador se puede seguir el siguiente link de una página oficial: http://www.dnielectronico.es/PDFs/Explorer%20y%20Chrome.pdf

Para diseñar este sistema tuvimos que seguir los siguientes pasos que nos permitieran ofrecer al usuario este servicio:

- Obtener un par de claves SSL para el negociado con el navegador del cliente (Deben estar firmadas por una autoridad certificadora si se desea que no se le notifique al usuario el mensaje de error “No procede de un sitio de confianza o una entidad autorizada de confianza”).
- Definimos un puerto en el servidor apache para que las peticiones que reciba por ese puerto se les exija un negociado de claves (Protocolo seguro), no

modificamos el puerto 443 para no alterar el protocolo seguro https.

- Añadimos un virtual host en el servidor apache añadiendo la necesidad del negociado de claves y ponemos como entidad certificadora el certificado de la policía nacional puede ser descargado desde el siguiente enlace : http://www.dnielectronico.es/PortalDNIe/PRF1_Cons02.action?pag=REF_1100 .
- Dentro de la información que almacena el servidor proveniente del certificado se encuentra la información del usuario concretamente el DNI. Tal como hemos diseñado el sistema hemos incluido un campo dni en la base de datos para almacenar de forma opcional el DNI del usuario. Como indicamos en el campo contraseña no almacenaremos el dni sin tratar, lo modificaremos mediante un algoritmo y lo que guardaremos será un resumen o hash del dni y posteriormente cada intento de acceso comparará esos hash.

Ejemplo del virtual host:

```
NameVirtualHost *:PUERTO
<VirtualHost *:PUERTO>

        ...
         SSLEngine On
         SSLCertificateFile CERTIFICADO SSL PROPIO
         SSLCACertificateFile CERTIFICADO RAÍZ DE POLICÍA
         SSLVerifyClient require
         SSLVerifyDepth 2
         SSLOptions +StdEnvVars +ExportCertData

        ...

</VirtualHost>
```

2. Gestión de usuarios.

Para la gestión de los usuarios no hemos diseñado una interfaz clara para administrarlos, sin embargo, está integrado en el diseño de las páginas. Al administrar las aplicaciones podemos añadir los administradores que deseamos que tengan permisos para construir páginas en nuestra aplicación y los usuarios que van a poder ser usados para permitirles el acceso a las páginas.

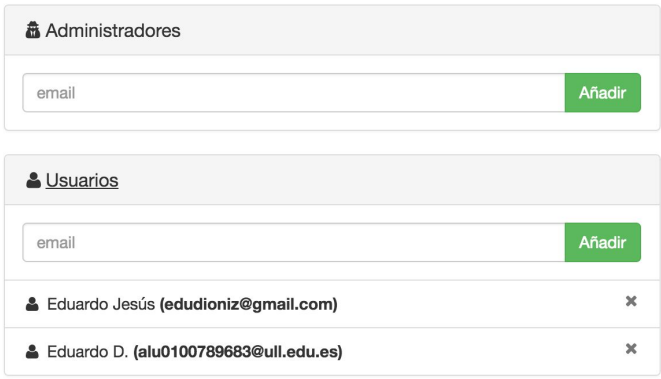

Figura 6. Panel para gestionar los usuarios en las aplicaciones

3. Gestión de los datos.

En la gestión de los datos hemos trabajado en tres vertientes, el desarrollo de la interfaz como añadir datos , actualizarlos, eliminarlos y hacer las pruebas correspondientes, también trabajamos en el enlace de ficheros como datos para su uso, y por último, trabajamos en la conexión con bases de datos de los usuarios.

La fase de diseño ha sido simple con el uso de bootstrap y una pequeña configuración que permita visualizar las consultas. En la segunda vertiente, la lectura de ficheros, hemos estudiado como aplicaciones transforman los ficheros csv y excel en json y hemos aplicado un formato compatible en el que los datos están representados en un array bidimensional. El proceso de la lectura se inicia desde el navegador del cliente, el cual se encarga de leer los datos y enviarlos vía ajax al servidor. En la parte de gestion de las bases de datos nos ha ayudado apoyarnos en el framework medoo.in el cual nos permitía acceder a multiples bases de datos ahorrandonos esa complejidad. En esta parte la interfaz traduce los datos que introduce el usuario a los datos que recibe medoo.in.

Al eliminar o editar las fuentes de los datos se tienen que tener muy en cuenta que afecta de forma inminente a toda la estructura generada bajo él. Cuando hablamos de la estructura generada bajo la fuente de datos nos referimos a las consultas que se han hecho de esa fuente de datos y todas las páginas que estén usando esas fuentes. Para remediar estos problemas hemos editado de forma recursiva todos los datos mencionados para que genere inestabilidad en el sistema y se mantenga la integridad de los datos.

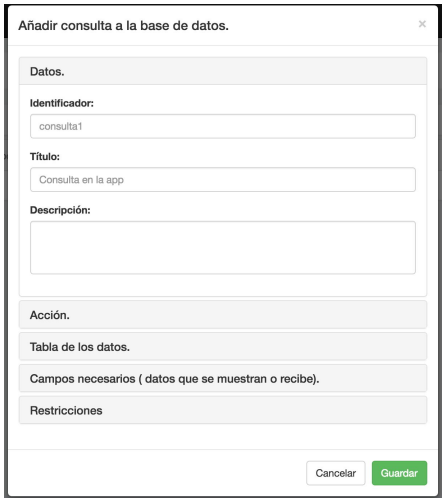

Figura 7. Interfaz para añadir las consultas

Para mejorar la seguridad con la aplicación y la reutilización del código no solo por nosotros sino por la persona que modifique o implemente código para la aplicación, diseñamos una interfaz para gestionar las consultas. En esta interfaz se almacenan como constantes todas las posibles consultas que se harán a la base de datos del sistema y además de esto también se definirá los parámetros que recibe y cómo se ejecuta y trata esa consulta. Las peculiaridades de la interfaz las detallaremos a continuación:

/Contenido acortado/

Las consultas que se realicen estarán definidas previamente en la clase MakeFrameDB. Serán constantes accesibles públicamente para poder indicar a cualquier objeto MakeFrameDB la consulta que se desea realizar.

```
const FUNCTION_TYPE_LOGIN_BY_PASS                    = 0;
const FUNCTION_TYPE_LOGIN_BY_DNIE                    = 1;
const FUNCTION_TYPE_LOGIN_BY_OAUT                    = 2;
const FUNCTION_TYPE_GET_APP_BY_USER                  = 3;
const FUNCTION_TYPE_GET_PAGE_BY_USER                 = 4;
```

.....

/Contenido acortado/

.....

A continuación podemos visualizar un ejemplo de sentencia en el que se reciben los parámetros por $arg y son tratados en una sentencia SQL. Se puede apreciar el uso del Framework Medoo.in en la interfaz de la base de datos del sistema.

```
case self::FUNCTION_TYPE_GET_QUERY_FROM_APP :
      $return_value = $db->select(
            "query_app",
            [
            "[>]app" => ["app" => "id"],
            "[>]user_app" => ["app" => "app"],
            "[>]query" => ["query" => "id"],
            "[>]source" => ["query.source" => "id"],
            ],
```

```
                ['query_app.query(id)', 'query.title', 'source.id(id_0)'],
                [
                        "AND" => [
                "query_app.app" => $arg['app'],
                "OR" => [
                                "app.owner"         => $arg['email'],
                                "user_app.email"    => $arg['email']
                        ]
                ]
            ]);
break;
```

4. Gestión de las aplicaciones.

Para la gestión de las aplicaciones usamos una interfaz básica que permita listarlas, editarlas y borrarlas, además de esto, la parte de edición de la aplicación controlará los recursos que vamos a tener en las creaciones y ediciones de las páginas.

La aplicación permite traspasarse a otro propietario con la finalidad de que un administrador u otro usuario pueda dirigir la aplicación sin perder los datos.

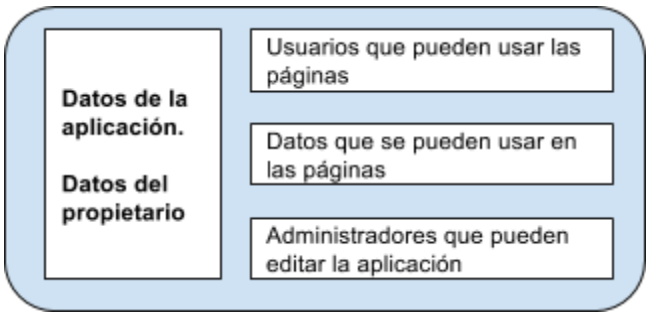

Figura 8. Ejemplo de datos que se almacenan en una aplicación.

5. Gestión de las páginas.

La gestión de las páginas es lo que nos ha llevado más trabajo debido a su complejidad. Para comenzar diseñamos la generación de las páginas, para ello, creamos una interfaz que se genera a partir del estándar creado para definir una página (Anexo A.1).

Figura 9. Generación de una página aún vacía.

Como se puede visualizar en la imagen anterior, se ofrece una serie de opciones para generar las páginas, las cuales estarán compuestas por componentes. El componente más básico es el contenedor, el cual es el único que se puede añadir en la página principal, dentro de estos componentes podemos añadir otros, en nuestro caso, definimos como componentes básicos párrafos e imágenes. Otras de las herramientas que ofrece es importar y exportar una página en el formato que hemos especificado, de este modo, los usuarios pueden descargar una configuración de una página e importarla en otro sistema.

Cuando ésta página se carga, se analiza recursivamente todo el árbol de componentes para poder ir generándolo, de este modo, no almacenamos la forma visual de la configuración sino que la generamos cada intento de edición. La forma en que se visualiza y se configura está definido por la configuración de cada componente.

Figura 10. Generación de una página con una tabla.

La parte que se encarga de ir construyendo el documento para generar la página funciona de la siguiente manera, cuando un usuario pincha en un componente se añade en una variable junto con los valores por defecto que nos devuelve el seleccionado. Además de añadirse como componente se añade como hijo del componente padre en el que se encuentra. Al intentar eliminar el componente, se tiene todos los hijos

que posee para poder hacerlo de forma recursiva hasta que no hayan ni hijos ni el componente que se quería eliminar.

Al clicar para editar un componente se abrirá un submenú donde configurar los parámetros que permita configurarlo. Actualmente los componentes indican cuales son los parámetros a editar y cómo quieren que se muestre al usuario (para profundizar en la edición de componentes mirarse Anexo B. Manual para crear componentes). Al editar estos componentes usan funciones del sistema que generan sobre el menú de configuración las entradas de los datos y las asocian a los valores a configurar. Una vez se guarda el componente, todos estos datos no son enviados al servidor php sino que el navegador web, modifica las variables que se enviarán al guardar la página. Los componentes pueden asociarse a las fuentes de datos que permita la aplicación y posteriormente cuando se guarde la página, esta se genera y se obtienen los datos de esas fuentes.

Cuando un usuario guarda la página que está creando, la petición llega al servidor PHP, el cual añade en la base de datos toda la información necesaria, a continuación, crea una estructura de carpetas en el caso de que sea una página de nueva creación. En la carpeta “app” en la raíz de la aplicación, se va a crear esta estructura de la que hablamos, en la que el primer nivel de serán carpetas que tienen el mismo nombre que el identificador de cada aplicación, dentro de estas carpetas está el segundo nivel con el nombre de las páginas. En el segundo nivel se genera un index.php en el cual el servidor PHP crea de un json al contenido html extrayendo la información de las fuentes seleccionadas.

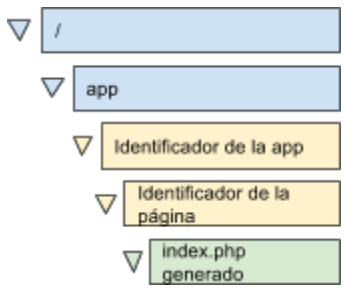

Figura 11. Estructura de carpetas al crear las páginas

Al eliminar un componente es necesario realizar una serie de comprobaciones para borrarlo de forma satisfactoria preservando la integridad de los datos. Es importante que el identificador que tenga el componente o los componentes hijos quedará sin uso, ya que comprendimos que era más costoso

refactorizar todo el esquema para reutilizar los índices. Es por ello que decidimos que fuera un índice numérico ascendente que queda almacenado en el esquema para que si se exporta o se guarda se pueda continuar modificando sin problemas en el futuro.

```
function deleteComponent(type, o){
	Para el componente contenedor hay que realizar otras compobaciones
	if(type == "grid"){
		father = $(o).parent().parent().parent().parent().parent().parent();
		item=$(o).parent().parent().parent().parent().find('div.separator');

		Eliminamos los hijos del componente que queremos borrar.
		deleteChildren(item.data('item'));

		Buscamos en donde está como hijo el componente que deceamos eliminar
		y lo borramos.
		if(jsonHasChildren[father.data('item')].indexOf(item.data('item'))
		!== -1) {
			jsonHasChildren[father.data('item')]
				.splice(jsonHasChildren[father.data('item')]
					.indexOf(item.data('item')), 1);
		}


		Si estoy borrando un contenedor múltiple y es el último contenedor
		ese espacio se quedará vacío. Tenemos que tener en cuenta que hay que
		eliminar a el padre que contiene el contenedor a eliminar.   if(
		jsonHasChildren[father.data('item')].length == 0){
			delete json_data['body'][item.data('item')];
			delete jsonHasChildren[item.data('item')];

			Hay que comprobar si el elemento que deseamos eliminar está en
			la página principal o está dentro de otro componente para
			tratar el componente vacío o la página vacía.
			if(jsonHasChildren[0].indexOf(father.data('item')) !== -1) {
				jsonHasChildren[0].splice(jsonHasChildren[0].indexOf(fat
				her.data('item')), 1);

				Si la página principal se queda sin hijos asignar el
				contenedor que indica que no hay nada en el interior
				(Visualmente).
				if(jsonHasChildren[0].length == 0){
					$('#main_container').html(container_empty);
				}
			}else{
				Debemos eliminar el componente de dentro de otro
				componente
				if(jsonHasChildren[father.parent().data('item')]
					.indexOf(father.data('item')) !== -1) {

					jsonHasChildren[father.parent().data('item')]
					.splice(jsonHasChildren[father.parent().data('ite
					m')].indexOf(father.data('item')), 1);
				}

				***** Se añadió la corrección del error descrito en el
				siguiente párrafo ******
```

```
                }


                delete json_data['body'][father.data('item')];
                delete jsonHasChildren[father.data('item')];


                father.remove();

        Si el contenedor no es múltiple y solo está ese componente
        lo borramos sin más comprobaciones.
        }else{
                delete json_data['body'][item.data('item')];
                delete jsonHasChildren[item.data('item')];

                item.parent().parent().parent().parent().remove();
        }
}else if(type == "item"){
        /Contenido acortado/
        En el caso de que sea un ítem o componente únicamente hacemos
        comprobaciones rutinarias que el contenedor siga teniendo ítems
        cuando lo eliminamos y en el caso de que no añadimos el contenedor
        que indica que está vacío.
}
```

Cada vez que un usuario edita la página y la guarda, esta se envía al servidor PHP y se realiza una serie de comprobaciones con los datos y la información de la página recibidos. Al recibir los parámetros se obtiene el identificador que tiene la página y el identificador editado en el formulario. En el caso en el que difieran es necesario no solo editar la información en la base de datos y de forma recursiva en todas las tablas donde aparezca el identificador (Permisos de los usuarios sobre las páginas, etc) sino también en la estructura interna de las páginas de modo que el servidor tiene que ejecutar comandos del sistema para trasladar esa información generada a otra ruta. A continuación detallamos el proceso que realiza.

```
/Contenido acortado/
Se trata toda la información que se reciben y se prepara para tratar el cambio de
directorio debido a que se está editando información sensible. Además se comprueba
que no se puede almacenar de forma simple y que o hay que crearla o editar los
identificadores.
if(($args['last']=='' || !isset($args['last'])) && !is_dir($route.$alias."/")){

        /Contenido acortado/
        Se obtienen los datos y se tratan para el envío a la base de datos.
        $data_db=MakeFrameDB::exec(MakeFrameDB::FUNCTION_TYPE_NEW_PAGE_BY_APP,$v);


        En el caso de que la base de datos nos indique que no hubo problemas y que
        ese identificador no existía, creamos el directorio con los comandos del
        sistema.
        if($data_db[1] == null && $data_db[2] == null){
                mkdir($route.$alias."/", 0777, true);
```

```
            $error = false;
        }

En el caso que haya cambiado el identificador y no tenga valores incorrectos enviamos otra petición distinta a la base de datos en la que indicamos que hay que modificar los valores existentes. En este caso
}else if($args['last']!='' && isset($args['last'])){

        /Contenido acortado/
        Se obtienen los datos y se tratan para el envío a la base de datos.
        $data_db=
            MakeFrameDB::exec(MakeFrameDB::FUNCTION_TYPE_UPDATE_PAGE_BY_APP,$v);

        En el caso de que la base de datos nos indique que no hubo problemas y que ese identificador no existía, renombramos el directorio con los comandos del sistema.
        if($data_db[1] == null && $data_db[2] == null){
            rename($route.$args['last']."/", $route.$alias."/");
            $error = false;
        }
}


/Contenido acortado/
En la ruta donde debe estár el fichero generado, abrimos un documento index.php y escribimos el contenido HTML que nos devuelve la función compose que detallaremos más abajo.
$route = "../app/".$args['app']."/".$alias."/";
$file_open_link = fopen($route."index.php", "w");
fwrite($file_open_link, self::compose( json_decode($args['data'], true) ));
fclose($file_open_link);
$response['data'] = $data_db;
```

Al finalizar el desarrollo nos dimos cuenta de que había un error en la visualización de los componentes cuando los eliminabamos. La dificultad para resolver este error residía en la complejidad de tener, por ejemplo, varios contenedores en una misma línea y al eliminar un componente hay que hacer múltiples comprobaciones con su padre, con los hijos que tiene y tener en cuenta cuando está sin componentes para añadir una etiqueta que lo indique. Al final únicamente era un error de concepto donde no se comprobaba bien los hijos que tenía el padre. Se soluciona sustituyendo la cuenta de los componentes añadidos por el número de hijos con respecto a su padre.

```
Cuando decidimos que vamos a eliminar el padre debido a que se ha quedado sin hijos debemos añadir una última comprobación mirando que el padre de nuestro padre tampoco tiene hijos. Así si podemos modificar el html para borrar todo el contenido de nuestro padre y añadir que ese ítem está vacío.
if(jsonHasChildren[father.parent().data('item')].length==0){
        father.parent().html(item_empty);
}
```

Cuando se le pide el servidor PHP que genere la página se inicia un proceso recursivo que va generando todos y cada unos de los componentes y le

añade las cabeceras y la cola al documento usando el patrón de diseño decorator para devolverlo al script que lo escribe en el servidor. A continuación detallaremos las peculiaridades del algoritmo.

```
public static function compose($args){
	return
		Concatenamos la cabecera del documento HTML.
		ViewDecorator::getTop().
			Ejecutamos de forma recursiva iniciando el componente inicial
			la función que irá por todos los componentes generandolos y
			devolviendolos.
			ViewFactory::generate(0,$args).
				Concatenamos la cola al documento HTML.
				ViewDecorator::getBottom();
}
```

A continuación detallaremos la función generate que se encuentra en el código anterior.

```
public static function generate($i, $args){
	/Contenido acortado/
	En el listado de hijos no se almacenan los nodos que no tienen hijos.
	Entendemos que son nodos terminales así que llegamos en la recursividad
	hasta ahí.
	if(!array_key_exists($i, $args['header']['children_list'])){
		Comprobar si el elemento es un elemento complejo.
		if(in_array($args['body'][$i]['type'], $complex_type)){
			Generar un elemento complejo.
			$r= self::compose_complex_terminal($args['body'][$i]);

		}else{
			Generar un elemento simple.
			$r = self::compose_terminal($args['body'][$i]);
		}
	Si el elemento no es terminal y tiene hijos.
	}else{
		Lanzamos la misma sentencia por todos sus hijos con la finalidad de
		llegar a todos los elementos.
		foreach ($args['header']['children_list'][$i] as $value) {
			$r .= self::generate($value, $args);

		}
		Como no es un elemento sino un contenedor usamos esta funcion.
		if($i != 0){
			$r=self::compose_container($return_value, $args['body'][$i]);
		}
	}
	return $r;
}
```

6. Adecuaciones a la normativa vigente.

Con el fin de adaptar la aplicación a las normativas vigentes aplicables hemos redactado unos términos legales visibles desde la web donde se indica la

información personal, las condiciones del servicio y los derechos de los usuarios con el fin de cumplir con la Ley de Servicios de la Sociedad de la Información (34/2002 del 11 de julio). En esos términos añadimos como hacer uso de los derechos ARCO a fin de cumplir con la normativa vigente de la Ley Orgánica de Protección de Datos (15/1999 del 13 diciembre). Además de todo esto se detalla en una página las cookies usadas y se incluye un script externo que muestra una ventana que informa sobre esas cookies al usuario con el fin de cumplir con la Ley de Servicios de la Sociedad de la Información Art. 22.2 (o "Ley de cookies").

Con el fin de cumplir con la normativa vigente con respecto a la inscripción de ficheros de titularidad privada en la Agencia Española de Protección de Datos se iniciará el proceso de inscripción. Como nota informativa se expondrá a los usuarios las obligaciones que estos deben tener con respecto a los datos de carácter personal ya que tanto los términos y condiciones como los datos que se inscribirán serán solo para el funcionamiento de la aplicación y no para los datos que posean los usuarios. En ningún caso se sobreentenderá que los usuarios se encuentran exentos de esta responsabilidad.

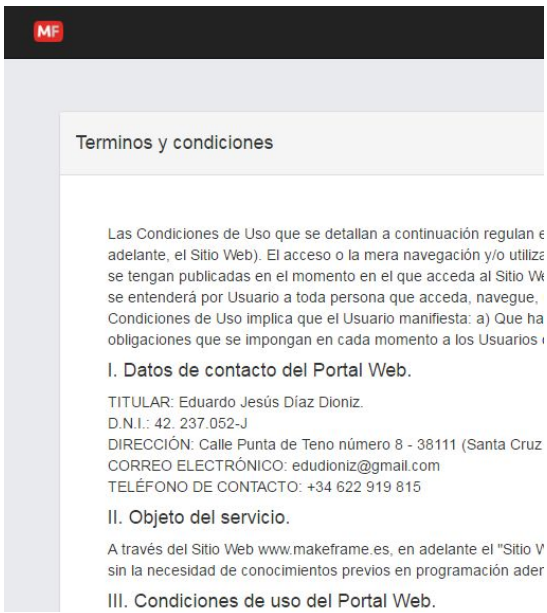

Figura 12. Página que indica los términos y condiciones.

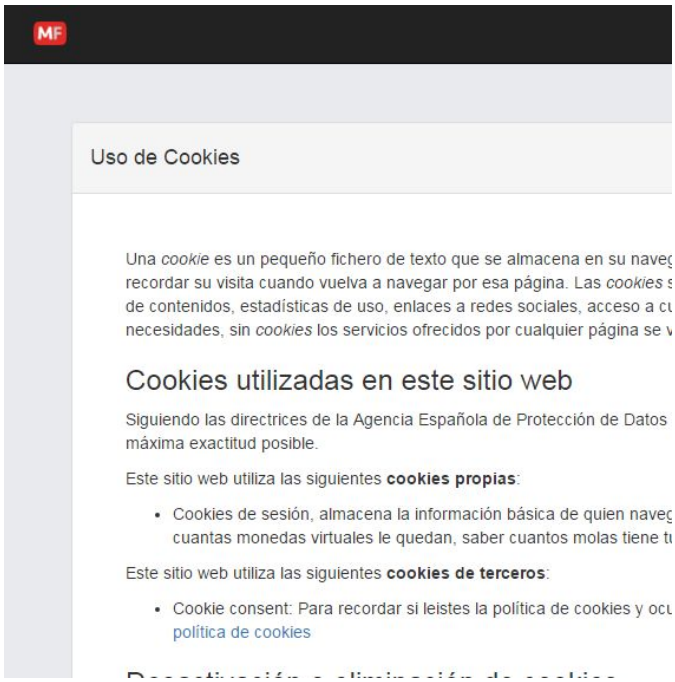

Figura 13. Página que indica la política de cookies.

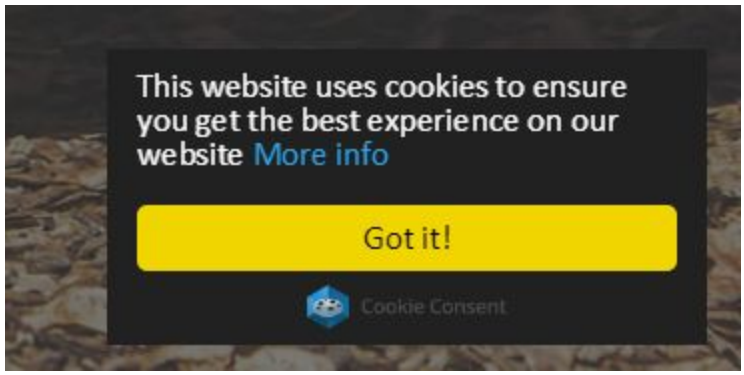

Figura 14. Captura del script que indica el uso de cookies.

Capítulo 4.

Objetivos, Conclusiones y líneas futuras.

Objetivos conseguidos.

Durante el desarrollo nos preocupaba que el producto final que quedará tras este TFG pudiera no ser útil debido a que harían falta demasiados componentes para cumplir con todas las necesidades de los usuarios. A pesar de esto durante el desarrollo contactamos con varias asociaciones y una ONG se vio interesada. Las últimas semanas de testeo fueron realizadas por la ONG y quedaron muy satisfechos con la resolución de su problema. En concreto, tenían la necesidad de mantener en una web información pública sobre las actividades realizadas y su rendimiento. Actualmente se encuentran configurando una base de datos con la intención de realizar una web para que los usuarios que accedan vean sus horas de voluntariado. Además de esto, cuentan con personal que tiene conocimientos en informática y ya poseen el manual para la creación de componentes que figura en el anexo B, con la intención de que comiencen a añadirlos a final de año.

Conclusiones

En el desarrollo de este Trabajo de Fin de Grado hemos conseguidos una propuesta estable y funcional de un framework permitiendo que muchos usuarios aceleren la creación de aplicaciones. No sólo aceleramos la creación de webs sino las consultas que hacemos a las bases de datos y cómo enlazamos esa salida con los componentes de la web.

Líneas futuras.

Nada más comenzar el proyecto acotamos lo que íbamos a desarrollar y puntualizamos cuales eran las líneas futuras para ampliar la aplicación. Como parte fundamental definimos la realización de otro proyecto únicamente dedicado a crear componentes para mejorar la usabilidad de la aplicación. Otra vertiente de desarrollo sería añadir

facilidades a la generación de las páginas ya que a pesar de tener todo lo necesario para crear la web de una forma rápida creemos que podría poseer herramientas adicionales como por ejemplo *"clonado de los componentes"*. Con estas herramientas se podría acelerar aún más la creación de las páginas y mejorar su utilidad.

Capítulo 5.

Conclusions

In the development of this work end of degree, we have achieved a stable and functional proposal for a framework that will allow many users to accelerate the creation of applications. But not only accelerate the creation of a web also we could queries databases and know how we link this output with the components of the web.

Future lines.

Just start the project delimit what we were going to develop and what were the future we point lines to extend the application.

As a fundamental part, we define the completion of another project solely dedicated to creating components to improve the usability of the application. Another aspect of development would be add facilities to generate pages, because despite having everything you need to create a website quickly, we could have additional tools such as cloned components. With these tools it could further accelerate the creation of the pages.

Capítulo 6.

Presupuesto

A continuación se detallan todos los costos que implicó el desarrollo del proyecto y se describe una previsión para el futuro mantenimiento y estimando los costos a partir de la continuación de los servicios contratados. Además de eso añadiremos otra estimación a partir de los precios actuales en caso de verse en la necesidad de ampliar los servicios.

Detalle de los costos producidos durante el desarrollo:

Descripción	Periodicidad	Coste	Coste anual
Dominio .es: makeframe.es	Anual	6,99€	6,99€
Hosting: VPS SSD 1 OVH	Mensual	2,99€	35,88€
Personal para el desarrollo: 1 Informático (10€/h * 6h * 60 días)	Única	3600€	3600€

Tabla 1. Costos durante la fase de desarrollo.

Detalle de los costos estimados para el mantenimiento a partir de los contratos ya estipulados:

Descripción	Periodicidad	Coste	Coste anual
Dominio .es: makeframe.es	Anual	6,99€	6,99€
Hosting: VPS SSD 1 OVH	Mensual	2,99€	35,88€
Personal de mantenimiento: 1 Informático (10€/h * 2h * 10 días)	Mensual	200€	2400€

Tabla 2. Costos estimados para el mantenimiento.

Además de los costes detallados anteriormente hemos estudiado la posibilidad de que los servicios contratados así como el servicio técnico no sea capaz de atender un volumen elevado de clientes. Por ello hemos redactado una última

estimación con una potencia de cálculo y capacidad superior la cual podría proveer el servicio a un mayor número de usuarios.

Descripción	Periodicidad	Coste	Coste anual
Dominio .es: makeframe.es	Anual	6,99€	6,99€
Hosting: VPS SSD 3 OVH	Mensual	11,99€	143,88€
Personal de mantenimiento: 2 Informáticos (10€/h * 2h * 10 días)	Mensual	400€	4800€

Tabla 3. Costos estimados para el mantenimiento aumentando la capacidad del servidor.

Capítulo 7.

Apéndice A: Estándares de la aplicación.

1. Estándar de las páginas.

Durante el desarrollo ha sido necesaria la creación de un estándar para que el navegador del usuario y la aplicación se entendieran. Las páginas guardadas se pueden exportar y usaremos esa utilidad para poder visualizar con más detalle el estándar. A continuación lo explicaremos a través de un ejemplo.

```
{
	"header": {
		"meta": {
		"title": /Título de la página/,
		"alias": /Identificador de la página/,
		"user": [
			/Usuarios que pueden hacer uso de la página. (Por
			seguridad no se exportan)/
		]
	},
	"last": /Últimos cambios/,
	"unique_id":
		/Identificador único de componentes/,
	"download": /fecha de última descarga/,
	    "children_list": {
			/Listado de los hijos que tienen sus componentes.
			Ejemplo: componente 1 es contenedor del 2 y del 3/
	      "1": [
	        2,
	        3,
	        4
	      ],
	      "status": "init"
	    }
	  },
	  "body": {
	    /Datos de los componentes/
	  }
	}
```

2.Estándar de los componentes.

Los componentes tienen una estructura básica en la que cada uno tiene toda su información. A continuación mostraremos la estructura y un ejemplo de componente.

```
/Identificador/: {
   "type": /Tipo de componente/,
   "data":
             /Datos específicos del componente/,
   "father":
             /Identificador del componente que lo contiene/
}
```

Por ejemplo, usaremos el componente ya creado tabla que contiene muchas variables, todas ellas necesarias para su funcionamiento:

```
/ID/: {
       /Tipo de componente: Tabla/
      "type": "table",
      "data": {
        "type_source":
                          /De donde extrae los datos/,
        "title": /título de la tabla/,
        "data": /datos de la tabla/,
        "source": /Fuente de los datos/,
        "query": /Donde obtener los datos/
      },
      "father":
             /El identificador del componente que lo contiene/
    },
       "status": "init"
```

Apéndice B: Guía de desarrollo de componentes.

Esta guía describe con todo detalle cómo los usuarios pueden crear sus propios componentes. Para definir un nuevo componente en la aplicación lo único necesario es seguir los siguientes pasos.

Paso 1: Detallar como se verá para clicar en el menú de componentes y que valores por defecto tendrá.

gview.php línea 344.

```
*Añadir aquí el componente. En esta parte va la forma visual y los valores
*predefinidos.
*Lo que el usuario ve cuando selecciona el componente en la ventana.
*Aquí hay un ejemplo de componente.

<div class="col-sm-2 content_tip">
    <div class="tip" data-dismiss="modal" onclick="
      addElement('modal', {
             'type':'button', 'data': {
                    'html':'Aceptar', 'class':'btn btn-default btn-block'
             }
      })">
      <span class="fa-stack fa-lg text-info">
             <i class="fa fa-square fa-stack-2x"></i>
             <i class="fa fa-square fa-stack-1x fa-inverse"></i>
      </span>Botón
    </div>
</div>
```

Paso 2: Detallar cómo se verá el componente en la página de construcción (Mientras se edite la página).

gview.js linea 587.

```
*Añadir aquí el componente. En esta parte debes indicar cómo se va a
*visualizar el componente ya en la página de construcción. Simplemente el
*texto que lo identifique, color y imagen.
*
*Para el color usará la clase: text-(color).
*Para la imagen usará la librería de iconos Font-Awesome.


var config = {
 'image'  : { 'color': 'green', 'type': 'fa-picture-o', 'text': 'Imagen'},
 ...
};
```

Paso 3: Detallar cómo editar el componente. (Qué datos son necesarios para configurar todos los valores del componente).

gview.js linea 550.

```
*Añadir aquí el componente. En esta parte va como proceder a configurar el
*componente. Hay funciones por defecto que te permiten añadir cuadros de
*texto, checkboxs, etc. Debes darle a todos los componentes el atributo
*name ya que se recibirá mediante forms.
*Es necesario añadir una condición al if - else con el tipo de dato que
estamos configurando.

}else if(obj['type'] == /TIPO QUE ESTAMOS CREANDO/){
    /FUNCIONES PARA LA EDICIÓN/
}

*Las funciones para el formulario de edición son las siguientes:

NOTA: Para que los elementos tenga los valores almacenados es necesario que
se añada esta sentencia en el value:
      json_data['body'][o]['data'][/Clave del valor buscado/]
```

```
append_input_setting(
	/Elemento jquery de destino/,
	/Título o texto informativo/,
	o,
	/Placeholder/,
	/NAME/,
	/VALUE/
);

append_textarea_setting(
	/Elemento jquery de destino/,
	/Título o texto informativo/,
	o,
	/Placeholder/,
	/NAME/,
	/VALUE/
	/Atributo data-parse (Función de parseo de la entrada)/
);

append_checkbox_setting(
	/Elemento jquery de destino/,
	/Título o texto informativo/,
	o,
	/NAM/,
	/VALUE/
	/Atributo data-parse (Función de parseo de la entrada)/
);

append_multiple_setting(
	/Elemento jquery de destino/,
	/atributo Name que se envía/,
	/ID del objeto que se envía/,
	/atributo Name de las pestañas/,
	[ {'title': /título/, 'id':/Identificador/, 'value': /atributo value/} ],
	/ID de la pestaña activa/
);

append_combo_setting(
	/Elemento jquery de destino/,
	{'id':/identificador/, 'name': /atributo name/},
	/Datos para seleccionar en el combobox/
);
```

Paso 4: Detallar cómo tiene que tratar la información recibida por el formulario, el servidor PHP.

Existen dos formas de tratar las etiquetas. Una básica en que las etiquetas se les añaden como atributos las variables que se reciben. Por ejemplo si editamos un componente ‘img’ y tenemos un ‘input’ en la edición con name ‘src’ al usar esta opción se integrará automáticamente como atributo. La otra opción es para los componentes complejos y necesitan un mayor tratamiento que añadir simplemente atributos.

ViewFactory línea 6.

*Debemos añadir a la variable `$html_data` el siguiente array:

```
/identificador/ => [
      /llave de apertura/,
      /llave de cierre/,
      /(opcional) cierre si la etiqueta no lleva autocierre/
];


*Ejemplo:
‘p’ => [
      ‘<p’,
      ‘>’,
      ‘</p>’
];
```

ViewFactory línea 38 (Si el elemento necesita trato especial).

```
*En el caso de que no sea suficiente añadir los datos como atributos del
*objeto que estamos editando podemos añadirlo a la lista de objetos
*complejos para que nos permita editarlo según nuestras necesidades.

$complex_type = ['table', /AÑADIR EL IDENTIFICADOR DEL ITEM COMPLEJO/];
```

ViewFactory línea 80 (Si el elemento necesita trato especial).

```
*Es necesario añadir una condición al if - else con el objeto complejo que
queremos tratar.

}else if($args['type'] == /TIPO QUE ESTAMOS CREANDO/){
    /PARSEAR MANUALMENTE LOS DATOS/
}
```

Capítulo 8.

Bibliografía

[1] PHP Manual. http://php.net/manual/en/

[2] Crockford, Douglas. JavaScript: The Good Parts: The Good Parts. O'Reilly. 2008.

[3] Flanagan, David. JavaScript: The Definitive Guide. O'Reilly. 2011

[4] LeBlanc, Jonathan. Identity and Data Security for Web Development: Best Practices. O'Reilly. 2016.

[5] Pitt, Chris. Pro PHP MVC. Apress. 2012.

[6] Lockhart, Josh. Modern PHP: New Features and Good Practices. O'Reilly. 2015.

[7] DNI electrónico. http://www.dnielectronico.es/PortalDNIe/

[8] S. Pressman, Roger. Ingeniería del Software, Un enfoque práctico. McGraw-Hill Interamericana de España S.L., 2010.

[9] Gamma, Erich. Patrones de diseño. Pearson Educación, 2002.

[10] Noticias jurídicas, Ley de servicios de la sociedad de la información (34/2002 del 11 de julio) http://noticias.juridicas.com/base_datos/Admin/l34-2002.html

[11] BOE, Ley orgánica de protección de datos de carácter personal (15/1999 del 13 de diciembre) https://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750