



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Adsuwak. La plataforma web/mòvil de advertiment 3.0 con Symfony2

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Sergio Benavent Mascarell

Tutor: José Vicente Busquets Mataix

2014/2015



Resumen

El “*banner*” es una herramienta publicitaria muy utilizada en internet. Debido a su popularización el usuario ha acabado obviando la zona donde estos son situados y ya no logra captar su atención. El CTR, en inglés “*Click Through Rate*”, el cual es el número de veces que un usuario hace “*click*” sobre un banner respecto el número de veces que es mostrado por pantalla, es actualmente muy cercano a cero. Este es el fenómeno denominado ceguera banner. El usuario se centra únicamente en el contenido y ya no les presta atención.

AdSuwak, la nueva generación de anuncios interactivos no intrusivos, es la solución a este problema. Con adSuwak el usuario puede promocionar cualquier producto de forma integrada en el contenido de su web aportándole valor al usuario y captando su atención gracias a su interactividad.

Mediante la nueva plataforma, el anunciante podrá crear diferentes tipos de anuncios inteligentes totalmente personalizables y adaptarlos a su contenido. Con esta nueva idea se pretende renovar el concepto de banner y lograr aumentar las tasas de conversión.

Palabras clave: banner, anuncio, Adsuwak, CTR, Click Through Rate, Ceguera Banner.

Abstract

The banner is an advertising tool very used in Internet. Due to its popularity the user has finished obviating the area where they are located and no longer pays attention to them. CTR, in English "Click Through Rate", which is the number of times a user "click" on a banner about the number of times it is displayed on screen, it is now close to zero. This is the phenomenon called banner blindness. Users only focuses on the web content and no longer pays attention.

AdSuwak, the new generation of non-intrusive interactive ads, is the solution to this problem. With adSuwak the user can promote any product for integrated web content giving it its value to the user and capturing his attention thanks to its interactivity.

Through the new platform, advertisers can create different types of smart fully customizable and tailor ads to your content. With this new idea is intended to renew the concept of banner and achieve higher conversion rates.

Keywords: Banner, ad, Adsuwak, CTR, Click Through Rate, Banner Blindness.





Tabla de contenidos

1. Introducción.....	8
2. Especificación de requisitos.....	10
2.1. Introducción	10
2.1.1. Propósito	10
2.1.2. Ámbito del sistema	10
2.1.3. Definiciones, acrónimos y abreviaturas.....	10
2.1.4. Referencias.....	11
2.1.5. Visión general	11
2.2. Descripción general.....	11
2.2.1. Perspectiva del producto	11
2.2.2. Funciones del producto	12
2.2.3. Características de los usuarios	13
2.2.4. Restricciones	13
2.2.5. Suposiciones y dependencias	13
2.2.6. Requisitos futuros	14
2.3. Requisitos específicos	14
2.3.1. Interfaces Externas	14
2.3.2. Requisitos funcionales	14
2.3.3. Requisitos de rendimiento	20
2.3.4. Requisitos tecnológicos	21
2.3.5. Atributos del sistema.....	21
3. Análisis.....	22
3.1. Introducción	22
3.2. Diagramas de casos de uso.....	22
3.2.1. Actores	22
3.2.2. Acciones.....	24
3.3. Diagrama base de datos	32
3.4. Diagramas de actividad.....	33
3.4.1. Registro de usuario	33
3.4.2. Inicio de sesión.....	34
3.4.3. Cambiar datos de perfil	35
3.4.4. Cambiar datos de compañía.....	36
3.4.5. Nuevo Suwak.....	37



3.4.6. Modificar Suwak.....	38
3.4.7. Eliminar Suwak	39
4. Diseño	40
4.1. Arquitectura MVC	40
4.1.2. Frontend y backend	41
4.1.3. Estructura del proyecto	43
4.1.4. Diseño de los Suwaks.....	43
5. Modelo de negocio	47
5.1. Introducción	47
5.2. Segmento de clientes.....	48
5.3. Propuestas de valor	48
5.4. Canales de distribución y comunicación.....	49
5.5. Relaciones con el cliente	50
5.6. Fuentes de ingresos.....	51
5.7. Recursos clave	51
5.8. Actividades clave.....	52
5.9. Asociaciones clave.....	52
5.10. Estructura de costes	53
6. Implementación.....	55
6.1. Introducción	55
6.2. Principales tecnologías	55
6.2.1. HTML 5.....	55
6.2.2. JavaScript.....	55
6.2.3. CSS.....	56
6.2.4. PHP 5.....	56
6.2.5. Ajax.....	56
6.2.6. JQuery	56
6.2.7. Bootstrap	57
6.2.8. Controlador de versiones: GIT.....	57
6.2.9. Xampp	57
6.2.10. Symfony2.....	58
6.2.11. Composer	62
6.2.12. Creando bundles	63
6.2.13. Generando la base de datos con Doctrine	65
6.2.14. Generando las vistas con Twig	70
6.2.15. Sistema de rutas y el controlador	73



6.2.16. Sistema de formularios	75
6.3. Desarrollo de la plataforma	78
6.3.1. Habilitando el traductor	78
6.3.2. Formulario para nuevo Suwak	80
6.3.3. Sistema de generación de scripts	81
6.3.4. Gestión de los usuarios.....	82
6.3.5. Sistema de notificaciones	90
6.3.6. Sistema de trazabilidad	92
6.4. Instalación servidor	95
6.4.1. Creación de una cuenta.....	95
6.4.2. Creando un “ <i>Droplet</i> ”	96
6.4.3. Administrando el Droplet	100
6.4.4. Instalando Plesk	103
6.4.5. Instalando GIT	111
6.4.6. Generando claves ssh en el servidor.	111
6.4.7. Clonando el proyecto en el servidor.	112
6.4.8. Configurando el proyecto con el servidor	113
6.4.9. Creando la base de datos.....	114
6.4.10. Configurando el DNS.....	115
7. Pruebas.....	117
7.1. Validación de estándares	117
7.2. Pruebas funcionales	118
7.3. Pruebas distintas resoluciones.....	118
8. Conclusión	120
9. Bibliografía	122



1. Introducción

El banner es una herramienta publicitaria muy utilizada en internet. Su funcionamiento consiste en insertar una imagen en una página web con la finalidad de atraer a los usuarios al sitio web anunciante.

Este tipo de anuncios suelen estar formados por imágenes estáticas o animadas y suelen ser colocados en los márgenes del contenido principal o bien en las cabeceras de las páginas web. Con el tiempo y la aparición de nuevas tecnologías se crean banners más complejos y se convierten en anuncios animados y en algunos casos con sonido.

El funcionamiento de estos anuncios es muy sencillo. El anunciante crea un banner (lo suficientemente llamativo para captar la atención) y lo inserta en una zona muy visible de su página web. Cuando los visitantes accedan, este llamará su atención y si está interesado hará *click* sobre él.

Actualmente la gran mayoría de sitios web utilizan este recurso con el fin de dar a conocer y promocionar sus servicios, y en ocasiones obtener ingresos extra por colocar publicidad en su sitio web. En algunos casos, llegando a exceder su uso, los publicistas llenan sus páginas con molestos anuncios que provocan que los visitantes eludan su atención del banner y acaben abandonando el sitio.

El “*banner blindness*” o ceguera banner en español, es un término de usabilidad definido en un estudio realizado por Benway and Lane en 1998, en el que se determinó que los usuarios obviaban las zonas donde normalmente se sitúan los banners y únicamente se centraban en el contenido.

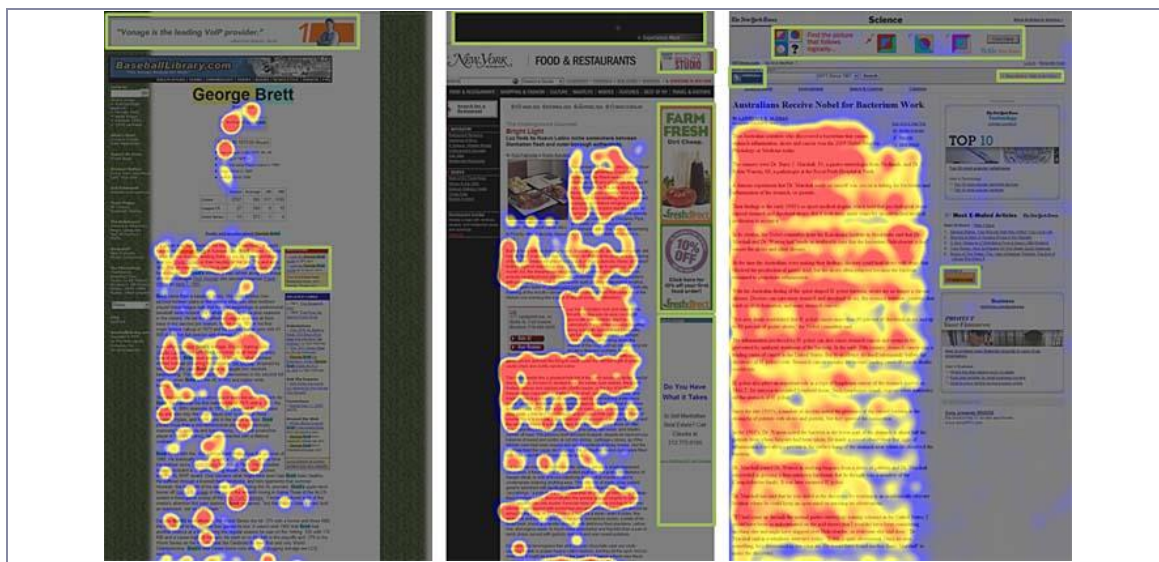


Figura 1.1. Estudio usabilidad Benway and Lane, 1998

Este comportamiento es cada vez más común. El CTR, en inglés “*Click Through Rate*”, es la relación entre el número de “*clicks*” que se realiza sobre un banner respecto

al número de veces que es mostrado por pantalla. Este valor, estudios demuestran que actualmente es muy cercano a cero, es decir, el banner ya no cumple con su función.

La solución a este problema es Adsuwak, la nueva generación de anuncios interactivos no intrusivos destinados a cambiar la forma con la que el usuario interactúa con la publicidad.

Con este tipo de anuncios, en adelante, Suwaks, el anunciante puede promocionar cualquier producto de manera integrada en el contenido de la web. El nuevo anuncio inteligente, estará totalmente relacionado con el contenido por lo que aportará valor al usuario, y además le permitirá realizar distintas acciones desde el mismo.

Mediante la nueva plataforma de advertisement 3.0 que ofrece Adsuwak, el usuario podrá crear diferentes tipos de Suwaks totalmente personalizados eligiendo entre tres tipos de acciones diferentes para que se adapte lo mejor posible al contenido.

La aplicación web será desarrollada mediante el framework Symfony2 en la versión “*Long Term Support*” LTS 2.3.37.

En los siguientes apartados, se desarrollarán los diferentes aspectos que se han tratado para llevar a cabo el proyecto. En primer lugar se realiza una especificación de requisitos para establecer que elementos debe contener la aplicación y cuál debe ser su comportamiento.

En segundo lugar se realiza un análisis más extenso y concreto sobre cómo se comporta la aplicación.

Seguidamente se desarrolla un modelo de negocio del proyecto en el que se realiza un análisis del mercado para conocer mejor a los clientes a los que va dirigida la aplicación y determinar cómo será el producto. En cuarto y quinto lugar se lleva a cabo el diseño e implementación de la aplicación y por último, con el proyecto ya finalizado se producirá la puesta en marcha de la aplicación web en un servidor real.



2. Especificación de requisitos

2.1. Introducción

En este apartado se tratan todos los puntos claves que definen el sistema en su totalidad. Será utilizado como guía a seguir durante el desarrollo del mismo.

2.1.1. Propósito

La especificación de requisitos de software “ERS” de un proyecto, es el proceso mediante el cual se describe toda la funcionalidad y comportamiento que debe presentar el sistema a desarrollar. Siguiendo el estándar IEEE Std. 830-1998, se procede en los siguientes apartados al análisis de los diferentes aspectos que debe cumplir la aplicación.

2.1.2. Ámbito del sistema

El objetivo del proyecto es el desarrollo de una plataforma en la que los usuarios puedan registrarse y gestionar sus anuncios inteligentes. La plataforma, en adelante “Adsuwak”, deberá realizar una completa gestión de los usuarios y además deberá facilitar al usuario la personalización de sus anuncios interactivos.

Los nuevos anuncios inteligentes, en adelante “Suwaks”, deberán permitir al usuario realizar diferentes acciones al usuario desde el mismo. Por ello, el usuario podrá escoger entre los Suwaks de tipo descarga, de tipo registro y de tipo enlace.

A su misma vez, la plataforma generará automáticamente el código necesario para que el usuario pueda insertar los Suwaks en su página web.

2.1.3. Definiciones, acrónimos y abreviaturas

- **Adsuwak:** Plataforma para crear y gestionar los Suwaks.
- **Suwak:** Rediseño del banner. Nuevo anuncio interactivo no intrusivo.
- **Frontend:** Parte frontal de la aplicación. Destinada a captar los usuarios visitantes.
- **Backend:** Parte interna de la aplicación.
- **Symfony:** Framework PHP orientado al desarrollo de aplicaciones web.
- **Lead:** Acción realizada desde el Suwak. Puede ser desde ser redireccionado hasta realizar un registro.
- **PHP:** Lenguaje de programación del lado del servidor.
- **CSS:** En inglés, “*cascading style sheets*”, son las hojas de estilo que personalizan una página web.



- **HTML:** Lenguaje de marcas de hipertexto, utilizado para la elaboración de páginas web.
- **Landing Page:** En español página de aterrizaje. Las “*landing page*” son las páginas que se encargan de recibir la primera visita de los usuarios.
- **Popup:** Ventana emergente.

2.1.4. Referencias

- IEEE Recommended Practices for Software Requirements specification ANSI/IEEE 830 1998.

2.1.5. Visión general

En los siguientes apartados, se describirá de una forma más detallada el funcionamiento de cada parte de la aplicación y como debe llevarse a cabo.

2.2. Descripción general

2.2.1. Perspectiva del producto

La finalidad del nuevo sistema de advertisement, consiste en redefinir o renovar el concepto de banner. Para ello se idean nuevos tipos de anuncios interactivos los cuales pretenden captar la atención del usuario mediante su interactividad. Debido a la falta de atención del usuario en las zonas donde normalmente se sitúa la publicidad, este nuevo tipo de anuncio, deberá ser insertado en el contenido de la web. Por ello, deberá estar totalmente relacionado con el contenido y será mínimamente intrusivo. Se logrará así captar la atención del usuario aportando nuevo contenido.

Para la creación de los Suwaks se ofrece al anunciante la plataforma Adsuwak mediante la cual podrá registrarse y crear sus propios anuncios. Es la parte backend de la aplicación.

La plataforma contiene la funcionalidad necesaria para la completa gestión de los usuarios, funciones tales como el registro, el *Log In*, la modificación del perfil o el cambio de contraseña. En lo respectivo a la creación de los nuevos Suwaks, se pone a disposición del usuario un formulario, segmentado en diferentes pasos, mediante el que podrá ir personalizando el nuevo banner e ir previsualizando los cambios que va introduciendo.

Una vez creado, si lo cree conveniente podrá editar sus propiedades o bien eliminarlo completamente.

Para la inserción del nuevo anuncio interactivo en la web, Adsuwak se encarga de generar el “script” correspondiente automáticamente y proporcionárselo al usuario.



Finalmente, en la parte Frontend de la aplicación se creará una sección en la que se mostrarán todos los anuncios existentes en la plataforma para facilitar así al usuario interesado la inserción de los nuevos anuncios.

Destacar que en esta plataforma el anunciante pagará por “Lead” o acción completada, no por interacción como pueden ser otras aplicaciones similares como “Google AdWords”. A su misma vez, el usuario recibirá compensación económica por acción completada desde su blog.

La aplicación Adsuwak es una aplicación autónoma en la que el funcionamiento de la misma no depende de otro sistema.

2.2.2. Funciones del producto

La aplicación debe ofrecer las siguientes funciones.

Gestión de usuarios

La plataforma contará con la funcionalidad necesaria para la completa gestión de los usuarios, funciones tales como son el registro, el inicio de sesión, la modificación del perfil o el cambio de contraseña.

Debe poner a disposición del usuario un formulario de registro en el que los usuarios puedan introducir sus datos y poder acceder al Backend.

A su misma vez, se debe posicionar en un lugar visible de la página principal un formulario mediante el cual el usuario pueda iniciar sesión.

Gestión de nuevos Suwaks

Debe existir un formulario lo más intuitivo posible para la creación de nuevos anuncios. De manera opcional, puede añadirse en el mismo una especie de ventana de previsualización, a modo de “*feedback*”, para que el anunciante vaya formándose una idea como va a resultar el anuncio.

Generación de script

Recogidos los datos introducidos por el anunciante, la plataforma debe ser capaz de tratarlos correspondientemente y generar un script. Una vez se haya generado, la plataforma deberá proporcionarle el enlace del script generado para que pueda ser insertado correspondientemente.

Traducción de los textos en dos idiomas

Debido a que la aplicación debe alcanzar el mayor número de usuarios posibles, se deberán traducir todos los textos disponibles en dos idiomas, inglés y español.

Alertas y notificaciones

Se deberán crear las correspondientes notificaciones asociadas a los diferentes eventos que se vayan produciendo. Estas notificaciones no podrán ser intrusivas y deberán poder eliminarse.



Trazabilidad de acciones del usuario

Por seguridad, la aplicación debe registrar todas las acciones básicas que realiza el usuario en la aplicación. Con ello, a la hora de solucionar las incidencias o en caso de haber algún problema se podrá revisar cual fue la última acción que realizó.

Concretamente se deben registrar las siguientes acciones:

- Creación de Suwak
- Eliminación de Suwak
- Modificación de Suwak
- Modificación datos de la compañía
- Modificación datos del perfil
- Modificación contraseña
- Inicio de sesión

2.2.3. Características de los usuarios

En la aplicación web se pueden distinguir dos tipos de usuarios, los anunciantes y los “bloggers”.

Los anunciantes son aquellos interesados en promocionar sus productos. Son los clientes de la aplicación. Idealmente serán empresas de marketing online que entre sus técnicas utilicen banners o marketing de contenidos.

Los usuarios de la aplicación son los “bloggers”. Este tipo de usuario es el que introducirá posteriormente los Suwaks creados en su blog. Recibirá una compensación económica por cada acción que se realice desde su web.

2.2.4. Restricciones

Formato y contenido de los Suwaks

El Suwak deberá estar formado por una imagen y un texto descriptivo. Ambos deben tener una estrecha relación con el contenido del blog en el que será insertado. El texto deberá captar la atención del usuario pero solo logrará su cometido si el texto le aporta valor al contenido. La imagen a su vez debe guardar relación con el contenido y el texto del anuncio. Será de vital importancia su elección puesto que será en lo primero que se fije el usuario.

2.2.5. Suposiciones y dependencias

Parte servidor

El servidor que contenga la aplicación web deberá de disponer de soporte para PHP5 y a su misma una vez un sistema gestor de base de datos como podría ser “*phpmyadmin*”.



Parte cliente

El usuario que acceda a la aplicación deberá disponer de un navegador compatible con el reciente estándar HTML5 y CSS3.

2.2.6. Requisitos futuros

La plataforma en un estado futuro deberá incorporar herramientas estadísticas para ofrecer mayor información al usuario sobre la eficacia de sus anuncios.

Cuando la aplicación alcance un número considerable de usuarios, para reducir recursos de procesamiento y tiempo de respuesta, se deberá almacenar el script en la base de datos a modo de caché.

A su vez deberá incorporar un sistema de “*matching*” que recoja todos los usuarios y el tipo de contenido de sus blogs para ofrecerle a los anunciantes automáticamente un listado de posibles blogs en los que insertar el anuncio.

En lo referente a los Suwaks habrá que hacerlos lo más personalizables posible. Por otra parte habrá que trabajar sobre el Suwak. Habrá que utilizar nuevas tecnologías que permitan su inserción en nuevos tipos de contenido.

2.3. Requisitos específicos

En esta parte del documento se tratan en alto detalle los requisitos que se deben satisfacer a la hora de desarrollar la aplicación. Esto permitirá posteriormente realizar una serie de pruebas para demostrar que el sistema satisface los requisitos establecidos.

2.3.1. Interfaces Externas

2.3.2. Requisitos funcionales

En esta sección se describirán detalladamente la funcionalidad que debe presentar la aplicación. Para ello, en primer lugar se mostrará la tabla donde se almacenará la información y seguidamente las acciones que se realizan sobre ella.

2.3.2.1. Clase “User”

Esta clase representa al usuario de la aplicación. Es la utilizada a la hora de crear nuevos usuarios.

Nombre	Tipo	Restricciones
Id	Int	Debe ser único
Username	Varchar	Valor no nulo
Email	Varchar	Debe ser un correo válido.
Enabled	Varchar	-



Salt	Varchar	Valor no nulo
Password	Varchar	Valor no nulo
Last Login	Datetime	Valor no nulo
Locked	Tinyint	-
Expired	Tinyint	-
Roles	Longtext	-
Phone	Varchar	-
Location	Int	-
Province	Int	-
Company	Int	-
Firstname	Varchar	-
Lastname	Varchar	-
Website	Varchar	-

Registro de nuevo usuario:

Esta función es la asociada a crear un nuevo usuario en la base de datos mediante el formulario de registro.

Este formulario únicamente pide al usuario que inserte un correo válido y una contraseña.

No se requiere ningún dato más a la hora del registro.

Se detalla el proceso a continuación:

Acción	Base de datos
1- Acceder al registro	
2- Rellenar campos	Se procede a la validación del formulario. Correcto: Se inserta en la base de datos el nuevo usuario. Incorrecto: Se muestran por pantalla los errores que ha cometido el usuario
3- Redirección a backend	

Modificar perfil

Esta función permite al usuario actualizar sus datos del perfil. Los campos que editaran su contenido son los siguientes:

- Nombre
- Apellidos
- Email
- Sitio Web
- Teléfono



Acción	Base de datos
1- Acceder al backend	
2- Abrir menú de usuario	
3- Acceder al formulario de actualización de datos	
4- Rellenar campos	Se procede a la validación del formulario. Correcto: Se modifican los datos del usuario relacionado. Incorrecto: Se muestran por pantalla los errores que ha cometido el usuario
5- Redirección a página de perfil	

Modificar contraseña

El usuario podrá renovar su contraseña cuando crea conveniente.

Acción	Base de datos
1- Acceder al backend	
2- Abrir menú de usuario	
3- Acceder al formulario de actualización de datos	
4- Rellenar campos	Se procede a la validación del formulario. Correcto: Se encripta y se inserta la nueva contraseña. Incorrecto: Se muestran por pantalla los errores que ha cometido el usuario
5- Redirección a página de perfil	

Cerrar Sesión

Mediante esta función el usuario podrá cerrar la sesión abierta.

Acción	Base de datos
Acceder al backend	
Abrir menú de usuario	
Pulsar en Cerrar Sesión	
Redirección a página de frontend	

2.3.2.2. Clase Companies

Esta entidad almacena toda la información referente a la compañía del usuario.

Nombre	Tipo	Restricciones
Id	Int	Debe ser único
Type	Int	-
Token	Varchar	-
Name	Varchar	-
Nif	Varchar(9)	Debe ser un Nif válido.

Address	Varchar	-
CP	Varchar (5)	No más de 5 números
Location	Varchar	-
Phone	Int (11)	No más de 11 números
Email	Varchar	Debe ser correo válido.
Url	Varchar	-
Date_reg	Datetime	-
Registry	Varchar	-

Modificar datos de compañía

Esta función permite al usuario insertar los datos de su compañía en la aplicación.

Acción	Base de datos
Acceder al backend	
Abrir menú de usuario	
Acceder al formulario de modificación de datos de compañía	
Insertar datos	Se procede a la validación del formulario. Correcto: Se almacenan los datos en la BD. Incorrecto: Se muestran por pantalla los errores que ha cometido el usuario
Redirección a página de frontend	

2.3.2.3. Clase Adsuwak

Esta es la clase que almacena las principales propiedades del Suwak. Contiene tres claves ajenas de las entidades “Advertisements”, “Companies” e “Images”.

Nombre	Tipo	Restricciones
Id	Int	Debe ser único
Token	Varchar	Valor no nulo
Advertisements_id	Int	Valor no nulo
Companies_id	Int	Valor no nulo
images_id	Int	Valor no nulo

2.3.2.4. Clase Advertisement

Contiene la información básica del anuncio.

Nombre	Tipo	Restricciones
Id	Int	Debe ser único
Title	Varchar	Valor no nulo
Description	LongText	Valor no nulo
Type	Varchar	Valor no nulo
Company_id	Int	Valor no nulo
Url_resource	Varchar	Valor no nulo
Advertisement_text	Varchar	Valor no nulo
Button_text	Varchar	Valor no nulo



Placeholder_download	Varchar	Valor no nulo
Gratitude_text	Varchar	Valor no nulo
Path	Varchar	Valor no nulo
Form_type	Varchar	Valor no nulo
Placeholder	Varchar	Valor no nulo
Var_name	Varchar	Valor no nulo
Form_type2	Varchar	Valor no nulo
Placeholder2	Varchar	Valor no nulo
Var_name2	Varchar	Valor no nulo
Form_type3	Varchar	Valor no nulo
Placeholder3	Varchar	Valor no nulo
Var_name3	Varchar	Valor no nulo
Overlay_description	LongText	Valor no nulo
Gratitude_button	Varchar	Valor no nulo

2.3.2.5. Clase Images

Contiene el “id” y “path” de la imagen que se mostrará en el fondo del anuncio

Nombre	Tipo	Restricciones
Id	Int	Debe ser único
Path	Varchar	Valor no nulo

Creación de nuevo Suwak

Esta es la función referente a la creación de un nuevo Suwak. Para ello, primeramente el anunciante deberá haber iniciado sesión en la plataforma. Seguidamente deberá acceder al formulario de creación y seguir paso por paso el formulario.

Acción	Base de datos
1- Iniciar sesión en la plataforma	
2- Abrir formulario de creación nuevo Suwak	
3- Rellenar campos	El formulario está dividido en cuatro pasos. En cada uno de ellos se configura una parte diferente del Suwak. Antes de pasar al siguiente paso se validan los campos. Correcto: Se pasa al siguiente paso Incorrecto: Se muestran por pantalla los errores que ha cometido el usuario Paso final: Se comprueba de nuevo los campos y si no ha habido ningún error se almacenan los datos.
4- Redirección a página de “Tus Suwaks”	

Visualización de Suwaks

Para la visualización se habita en el backend una galería con una previsualización de los diferentes anuncios creados. Cuando el usuario quiera ver el Suwak deberá hacer click sobre uno de ellos y se le desplegará un modal a modo de “*Popup*”.

Acción	Base de datos
1- Iniciar sesión en la plataforma	-
2- Seleccionar opción Tus Suwaks	-
3- Redirección a página de “Tus Suwaks”	-

Modificación de Suwak

Desde la opción “Tus Suwaks” del Backend cuando el usuario pulse en un Suwak se le desplegará el “*popup*”. En el mismo aparecerán una serie de propiedades que se le permite modificar. Son las siguientes:

- Título
- Descripción
- Texto del anuncio
- Url de destino
- Texto del botón

Acción	Base de datos
1- Iniciar sesión en la plataforma	-
2- Seleccionar opción Tus Suwaks	-
3- Abrir modal	-
4- Rellenar el formulario	Validar el formulario. Correcto: Se pasa al siguiente paso Incorrecto: Se muestran por pantalla los errores que ha cometido el usuario
5- Realizar llamada Ajax al servidor con los nuevos cambios	-
6- Redirección a página de “Tus Suwaks”	-



Eliminación de Suwak

Para la eliminación del Suwak el usuario debe acceder a la galería y desplegar el modal anteriormente mencionado. Una vez abierto, deberá pulsar en el botón eliminar.

Acción	Base de datos
1- Iniciar sesión en la plataforma	-
2- Seleccionar opción Tus Suwaks	-
3- Abrir modal	-
4- Pulsar en el botón eliminar	-
5- Realizar llamada Ajax al servidor con el id del Suwak	-
6- Redirección a página de “Tus Suwaks”	-

Inserción del Suwak en una web

Para la inserción del Suwak en una página web, el usuario deberá introducir el código script que le facilita la plataforma.

Acción	Base de datos
1- Iniciar sesión en la plataforma	-
2- Seleccionar opción Tus Suwaks	-
3- Abrir modal	-
4- Rellenar el formulario	Validar el formulario. Correcto: Se pasa al siguiente paso Incorrecto: Se muestran por pantalla los errores que ha cometido el usuario
5- Realizar llamada Ajax al servidor con los nuevos cambios	-
6- Redirección a página de “Tus Suwaks”	-

2.3.3. Requisitos de rendimiento

El sistema debe responder a las peticiones del usuario con un tiempo de respuesta adecuado. El tiempo de respuesta no debería ser mayor a cinco segundos para proporcionar una buena experiencia de usuario.

La información que se vaya generando deberá introducirse en una base de datos. El tiempo de respuesta asociado a consultas, altas y modificaciones no deberá ser mayor de 8 segundos.

2.3.4. Requisitos tecnológicos

El proyecto será llevado a cabo mediante el framework Symfony2. Se utilizará la versión LTS 2.3.37 debido a que es un proyecto a largo plazo y esta versión tiene asignada un periodo de revisión y mantenimiento hasta mayo del año 2016.

La versión de PHP será como mínimo la versión 5.3. Se utilizará el estándar HTML5 para explotar toda su nueva funcionalidad. Para mejorar la experiencia del usuario se utilizará la librería “*Jquery*” en su última versión y diferentes “plugins”.

2.3.5. Atributos del sistema

Requisitos de fiabilidad

La aplicación debe estar funcionando todo el tiempo. Debe ser capaz de atender las peticiones de los usuarios.

Debido a que no hay un sistema perfecto y puede haber caídas de servicio se deberá disponer una copia de seguridad de la aplicación configurada en un servidor de respaldo, por lo que en caso de que el servidor principal deje de estar disponible, poder suplirlo lo más rápido posible.

Mantenibilidad

El proyecto debe estar en constante proceso de mantenibilidad durante los primeros tres meses debido a su inmadurez. Con este periodo cumplido, habría que dedicarle un menor tiempo puesto que ya no debería haber errores graves de funcionamiento.

A su misma vez, para asegurar el correcto funcionamiento de la aplicación, se deberán revisar las copias de seguridad de la base de datos realizadas.

Portabilidad

El sistema a desarrollar será totalmente portable. Siguiendo los requisitos descritos en apartados anteriores, la aplicación al estar alojada en un servidor, debería poder ser ejecutada en cualquier navegador sin importar el sistema operativo.

A su vez, deberá tenerse en cuenta el diseño para diferentes tamaños de pantalla.

Seguridad

Es necesario restringir el acceso a la plataforma a usuarios registrados. Para ello mediante los formularios correspondientes se recogerán los datos de los usuarios y controlando la sesión se les permitirá el acceso al backend.

La aplicación debe ser capaz de detectar la inserción de datos anómalos y adoptar las medidas necesarias para evitar su uso.



La contraseña almacenada en la base de datos será encriptada bajo la utilización del algoritmo SHA-512.

3. Análisis

3.1. Introducción

En este capítulo se realiza un análisis de la aplicación. Se especifica la estructura que debe presentar y cuál debe ser su comportamiento.

El análisis se ha realizado utilizando el lenguaje UML, en inglés “*Unified Modeling Language*” mediante el cual se establecen tres tipos de diagrama: diagramas de comportamiento, de estructura y de interacción.

3.2. Diagramas de casos de uso

El diagrama de casos de uso es un diagrama de comportamiento UML. Identifica todas las acciones posibles que puede realizar el usuario en un escenario concreto de la aplicación.

En primer lugar se identificarán los actores del sistema y posteriormente se tratarán las diferentes acciones que pueden desempeñar.

3.2.1. Actores

Antes de comenzar con el análisis debemos en primer lugar identificar a los actores que van a interactuar con el sistema. En la figura 3.1 se muestran los usuarios que forman parte de la aplicación.

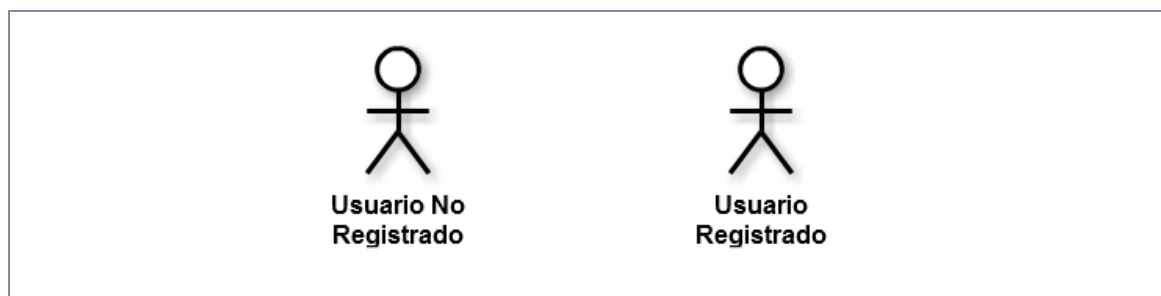


Figura 3.1. Actores del sistema

Existen dos actores tipo. Se distinguen entre usuario registrado y usuario no registrado.

3.2.1.1. Usuario no registrado

Los casos de uso que puede empeñar este usuario los puede desempeñar cualquier usuario. Se considera usuario no registrado a aquel que visita por primera vez la página web o bien todavía no se ha registrado *sesión*.

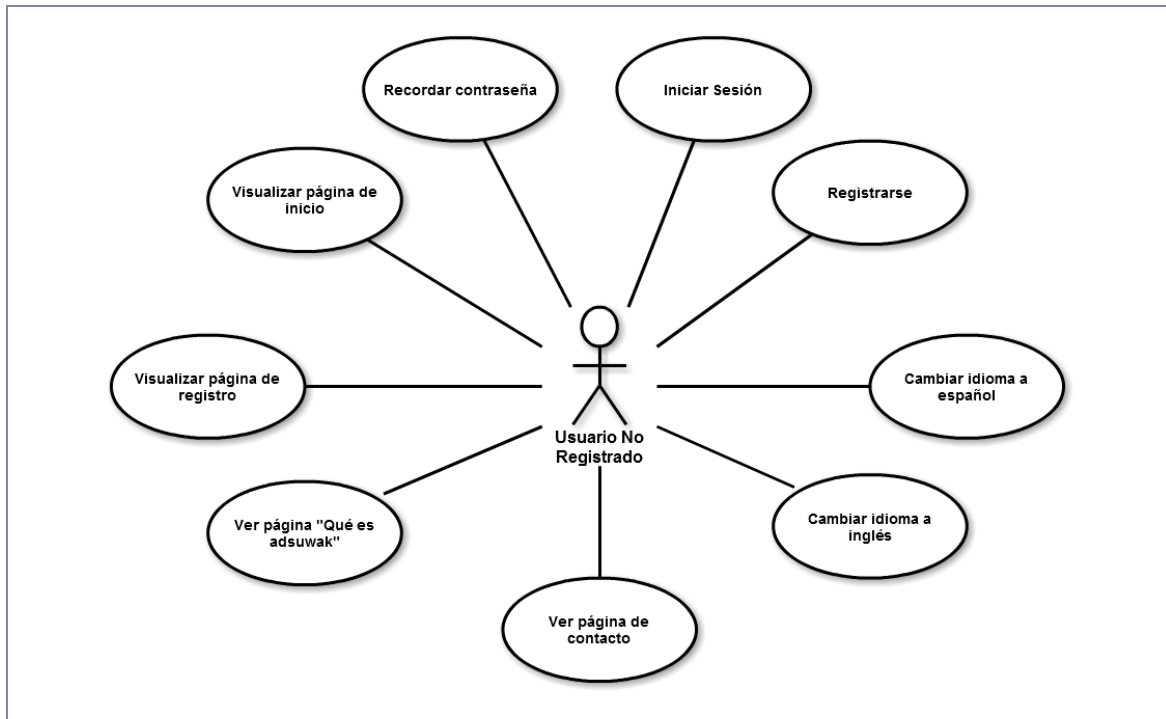


Figura 3.2 Casos de uso usuario no registrado

3.2.1.2. Usuario registrado

Un usuario registrado es aquel ha rellenado el formulario de registro y ha iniciado sesión en la aplicación. En la figura 3.3 se añaden los casos de uso que puede realizar este actor.

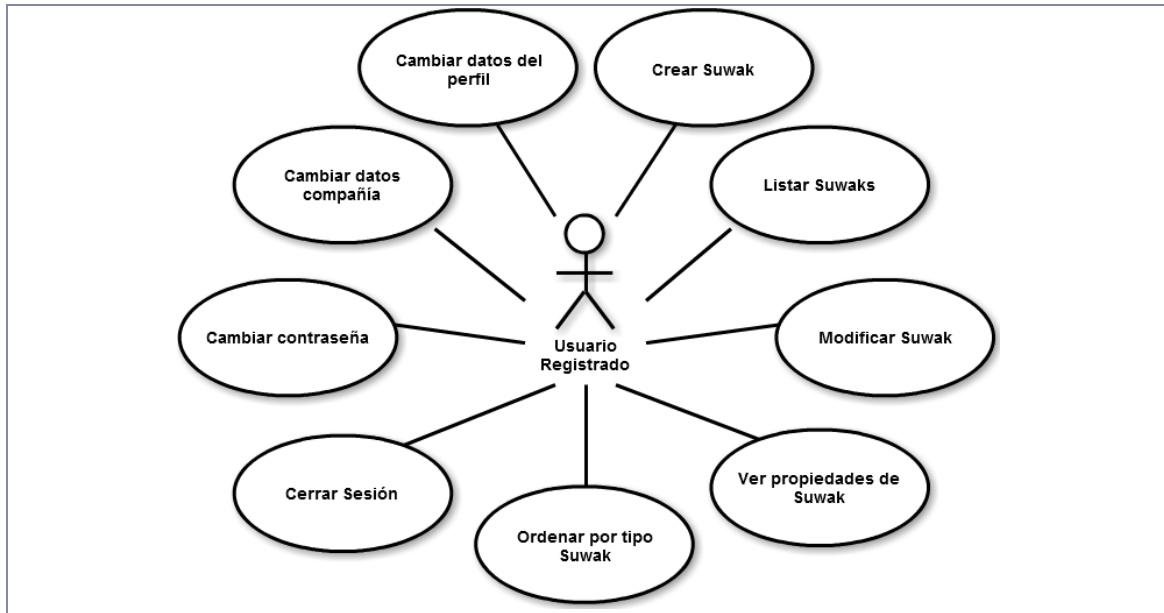


Figura 3.3 Casos de uso usuario registrado

3.2.2. Acciones

Para la acotaciones de las distintas funciones del sistema, el presente apartado se divide en los casos de uso que se pueden realizar en la parte frontend y en la parte backend.

Para este apartado se decide expresar los distintos casos de uso de la siguiente manera para entrar en mayor detalle.

3.2.2.1. Frontend

Caso de uso: Registrarse en la plataforma	
Actor: Usuario no registrado	
Precondición:	
Intenciones del usuario	Obligaciones del sistema
1. El usuario accede a la página principal	2.
3. Desde el menú accede a la página de registro	4.
5. Introduce un correo	6. Si el correo no es válido el sistema devuelve error
7. Introduce una contraseña	8.
9. Repite la contraseña introducida	10. Si las contraseñas no coinciden el sistema devuelve error.
11.	12. El sistema introduce los datos introducidos en la base de datos y valida el usuario
13. Recibe la notificación de activación de cuenta e inicia sesión	14.

Caso de uso: Iniciar Sesión	
Actor: Usuario no registrado	
Precondición: No haber iniciado sesión.	
Intenciones del usuario	Obligaciones del sistema
1. El usuario accede a la página principal	2.
3. Sitúa el ratón sobre la ventana de “Log In”	4.
5. Introduce el correo electrónico	6. Si no es un correo válido el sistema devuelve error
7. Introduce la contraseña	8.
9. Pulsa el botón “Iniciar Sesión”	10. El sistema valida los datos introducidos. Si no son válidos le devuelve un error.
11. Accede a la plataforma	12.

Caso de uso: Recordar contraseña	
Actor: Usuario no registrado	
Intenciones del usuario	Obligación del sistema
1. El usuario accede a la página principal	2.
3. Sitúa el ratón sobre la ventana de “Log In”	4.
5. No recuerda la contraseña y pulsa en “Recordar Contraseña”	6. Redireccionar a página de redirección de contraseña.
7. Introduce el correo de registro	8. Si no es un correo valido el sistema devuelve un error
9. Si el correo introducido en el formulario existe en la base de datos recibirá un correo con el reinicio de la contraseña.	10.

Caso de uso: Visualizar página de inicio	
Precondición: -	
Actor: Usuario no registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a la página principal	2.



Caso de uso: Visualizar página de registro	
Precondición: No haber iniciado sesión.	
Actor: Usuario no registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a la página principal	2.
3. Pulsar enlace de registro	4.

Caso de uso: Visualizar página de contacto	
Precondición: -	
Actor: Usuario no registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a página principal	2.
3. Pulsar enlace de página de contacto	4.

Caso de uso: Dejar comentario	
Precondición: -	
Actor: Usuario no registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a página principal	2.
3. Pulsar enlace de página de contacto	4.
5. Introducir correo	6. Si no es un correo válido el sistema devuelve un error
7. Introducir comentario	8. Inserta el código en la base de datos

Caso de uso: Seleccionar idioma en español	
Precondición: -	
Actor: Usuario no registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a página principal	2.
3. Pulsar opción de español	4. El sistema traduce todos los textos

Caso de uso: Seleccionar idioma en inglés	
Precondición: -	
Actor: Usuario no registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a página principal	2.
3. Del menú de selección de idioma escoger inglés	4. El sistema traduce todos los textos

Caso de uso: Seleccionar idioma en inglés	
Precondición: -	
Actor: Usuario no registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a página principal	2.
3. Del menú de selección de idioma escoger inglés	4. El sistema traduce todos los textos

Caso de uso: Ver lista de Suwaks	
Precondición: -	
Actor: Usuario no registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a página principal	2.
3. Pulsar enlace de lista de Suwaks	4. El sistema traduce todos los textos

3.2.2.2. Backend

Caso de uso: Cambiar contraseña	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Desplegar el menú de usuario	4.
5. Pulsar opción “perfil”	6.
7. En la nueva pantalla pulsar en la opción “cambiar contraseña”	8.
9. Introducir la contraseña actual	10.
11. Introducir la nueva contraseña	12.
13. Repetir la contraseña	14. Comprueba que la contraseña actual sea la correcta y si las dos nuevas contraseñas coinciden. 15. En caso de no coincidir devuelve un error. En caso contrario actualiza la contraseña.
16. Pulsar en “cambiar contraseña”	17. Guardar datos en la base de datos

Caso de uso: Actualizar perfil	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Desplegar el menú de usuario	4.
5. Pulsar opción “perfil”	6.
7. Introducir nombre	8. Si se introduce algún carácter no admitido devuelve error
9. Introducir apellidos	10. Si se introduce algún carácter no admitido devuelve error
11. Introducir email	12. Si se introduce un email no válido devuelve error.
13. Introducir sitio web	14. Si la URL introducida no es válida



	devolver error.
15. Introducir teléfono	16. Si se introducen más de nueve números devolver error.
17. Pulsar en actualizar	18. Guardar datos en la base de datos

Caso de uso: Actualizar datos de la compañía	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Desplegar el menú de usuario	4.
5. Pulsar opción “compañía”	6.
7. Introducir nombre de la compañía	8. Si se introduce algún carácter no admitido devuelve error
9. Introducir Nif de la compañía	10. Si se introduce un nif no válido devuelve error.
11. Introducir dirección	12. Si se introduce algún carácter no admitido devuelve error.
13. Introducir código postal	14. Si se introducen más de 5 números devuelve error.
15. Introducir teléfono	16. Si se introducen más de 11 números devolver error.
17. Introducir email	18. Si se introduce un email no válido devuelve error.
19. Introducir página web	20. Si la URL introducida no es válida devolver error.
21. Pulsar en actualizar	22. Guardar datos en la base de datos

Caso de uso: Cerrar Sesión	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Desplegar el menú de usuario	4.
5. Pulsar opción “Salir”	6. Redireccionar a página principal

Caso de uso: Listar Suwaks	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Seleccionar opción del menú “Tus Suwaks”	4.

Caso de uso: Ver propiedades de Suwak	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Seleccionar opción del menú “Tus Suwaks”	4.
5. Seleccionar un Suwak	6. Abrir modal con los datos del Suwak

Caso de uso: Clasificar Suwaks	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Seleccionar opción del menú “Tus Suwaks”	4.
5. Escoger una opción	6. Ocultar/mostrar los Suwaks relacionados con el tipo escogido.

Caso de uso: Modificar propiedades de Suwak	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Seleccionar opción del menú “Tus Suwaks”	4.
5. Seleccionar un Suwak	6. Abrir modal con los datos del Suwak
7. Insertar titulo	8. Si se introduce algún carácter no admitido devuelve error
9. Insertar descripción	10. Si se introduce algún carácter no admitido devuelve error
11. Insertar texto anuncio	12. Si se introduce algún carácter no admitido devuelve error
13. Insertar url de destino	14. Si se introduce algún carácter no admitido devuelve error
15. Insertar texto del botón	16. Si se introduce algún carácter no admitido devuelve error
17. Pulsar en “Guardar cambios”	18. Validar los datos introducidos y almacenar los cambios en la base de datos



Caso de uso: Eliminar Suwak	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Seleccionar opción del menú “Tus Suwaks”	4.
5. Seleccionar un Suwak	6. Abrir modal con los datos del Suwak
7. En el modal pulsar el botón “Eliminar Suwak”	8. Elimina el Suwak de la base de datos

Caso de uso: Conseguir código script de Suwak	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Seleccionar opción del menú “Tus Suwaks”	4.
5. Seleccionar un Suwak	6. Abrir modal con los datos del Suwak
7. En el modal ir a la sección “Código del script”	8. Elimina el Suwak de la base de datos

Caso de uso: Crear nuevo Suwak	
Precondición: Haber iniciado sesión	
Actor: Usuario registrado	
Intenciones del usuario	Obligación del sistema
1. Acceder a backend	2.
3. Desplegar el menú de usuario	4.
5. Seleccionar opción del menú “Nuevo Suwak”	6. Cargar paso uno del formulario
7. Rellenar el paso uno del formulario	8.
9. Introducir título	10. Si se introduce algún carácter no admitido devuelve error.
11. Introducir descripción	12. Si se introduce algún carácter no admitido devuelve error.
13. Pulsar en “Siguiente”	14. Validar que no hay ningún campo en blanco. Si es correcto cargar paso dos.
15. Rellenar paso dos del formulario	16.
17. Seleccionar tipo Suwak	18. Mostrar previsualización del tipo del Suwak y ocultar los campos que no correspondan con el tipo
19. Introducir url	20. Comprobar que sea una URL válida.
21. Introducir texto del anuncio	22. Si se introduce algún carácter no admitido devuelve error. Cargar el texto en la previsualización
23. Cargar imagen de anuncio	24. Cargar imagen en la



	previsualización.
25. Pulsar en “Siguiete”	26. Validar que no hay ningún campo en blanco. Si es correcto cargar paso tres.
27. Rellenar paso tres del formulario	28. Cargar previsualización
29. Introducir descripción	30. Si se introduce algún carácter no admitido devuelve error.
31. Introducir texto del botón	32. Si se introduce algún carácter no admitido devuelve error.
33. Si el tipo de Suwak escogido es de registro	34. Cargar formulario en la previsualización
35. Introducir campos de formulario	36. Cargar campos en previsualización.
37. Introducir logo de formulario	38. Cargar logo en previsualización
39. Pulsar en “Siguiete”	40. Validar que no hay ningún campo en blanco. Si es correcto cargar paso cuatro.
41. Rellenar paso cuatro del formulario	42.
43. Introducir mensaje agradecimiento	44. Si se introduce algún carácter no admitido devuelve error.
45. Introducir texto botón agradecimiento	46. Si se introduce algún carácter no admitido devuelve error.
47. Pulsar en “Finalizar”	48. Validar que no hay ningún campo en blanco. Si es correcto guardar Suwak en la base de datos.
49.	50. Redireccionar a “Tus Suwaks” y mostrar notificación “Suwak creado correctamente”



3.3. Diagrama base de datos

El diagrama de la base de datos, es una representación gráfica que muestra las entidades y sus propiedades y las relaciones entre diferentes clases.

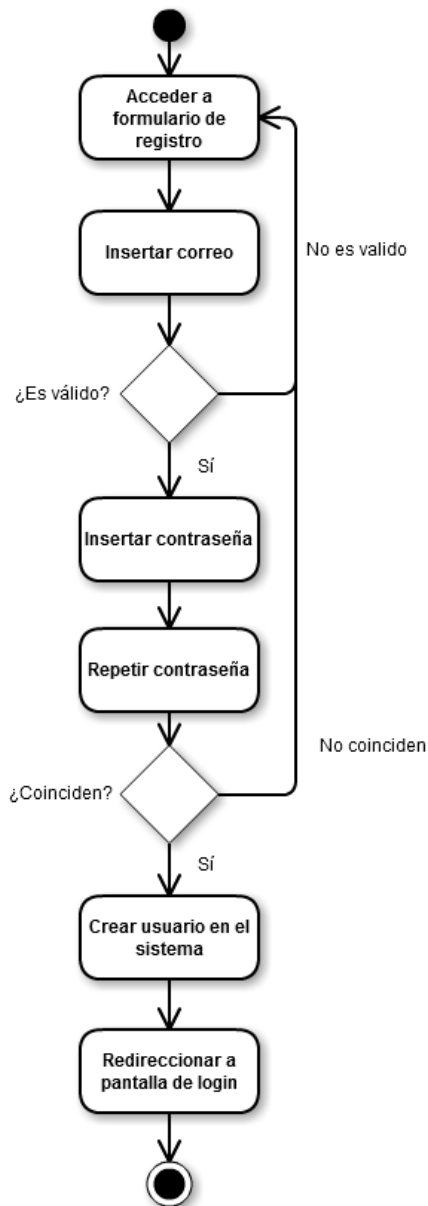


3.4. Diagramas de actividad

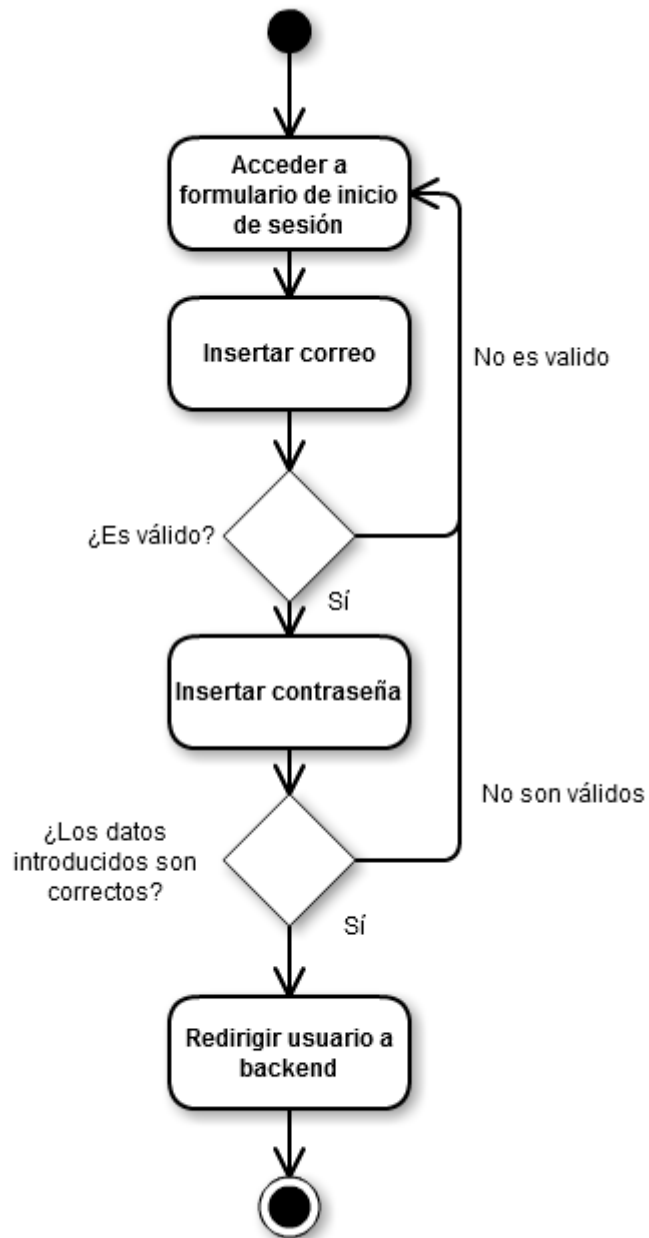
El diagrama de actividad es la representación gráfica del flujo o proceso que realiza el sistema para implementar una funcionalidad. Modelan el comportamiento que el usuario tendrá cuando interactúe con el sistema.

En este apartado se mostrarán los principales flujos de trabajo que se realizan en el sistema. Para no resultar repetitivos, algunas funciones tales como son la de modificación de los datos del usuario, se han englobado en un mismo diagrama de actividad.

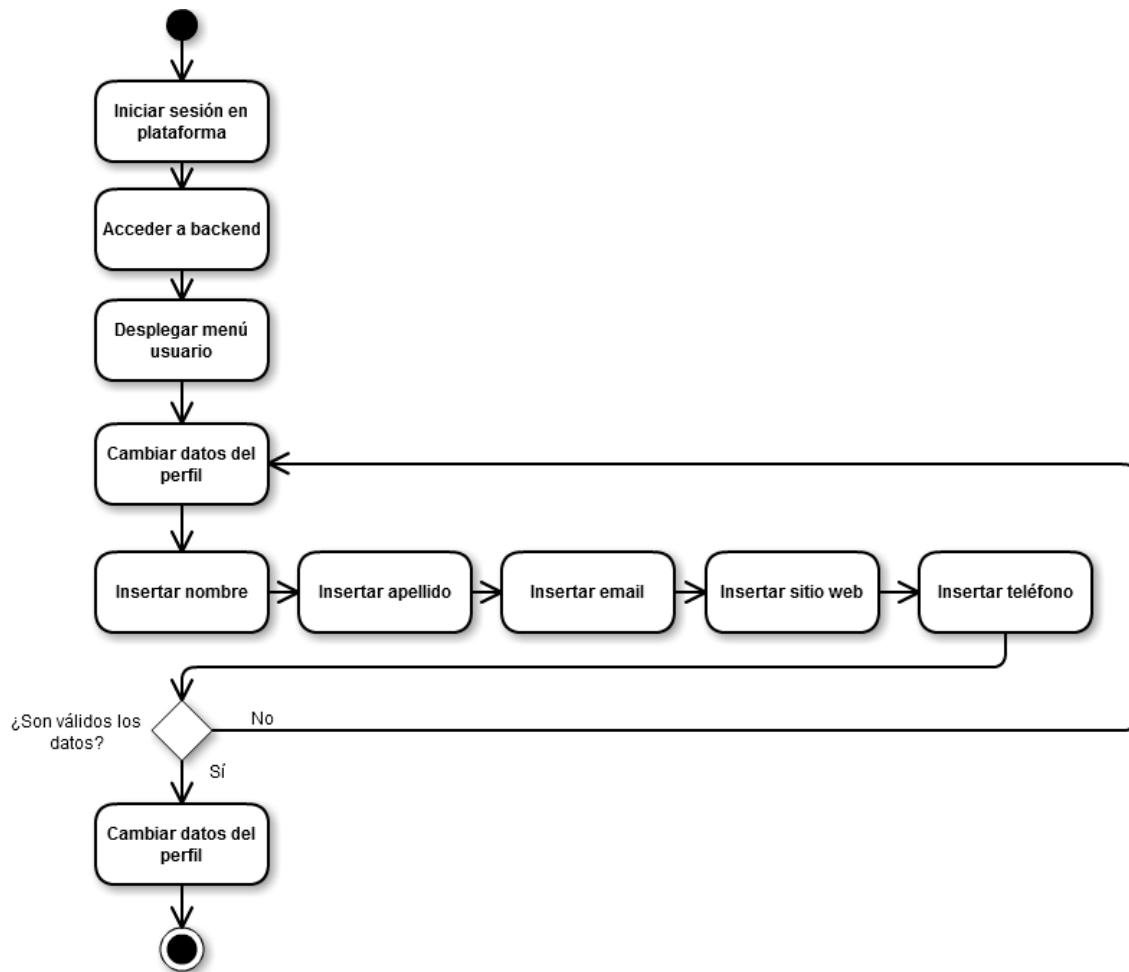
3.4.1. Registro de usuario



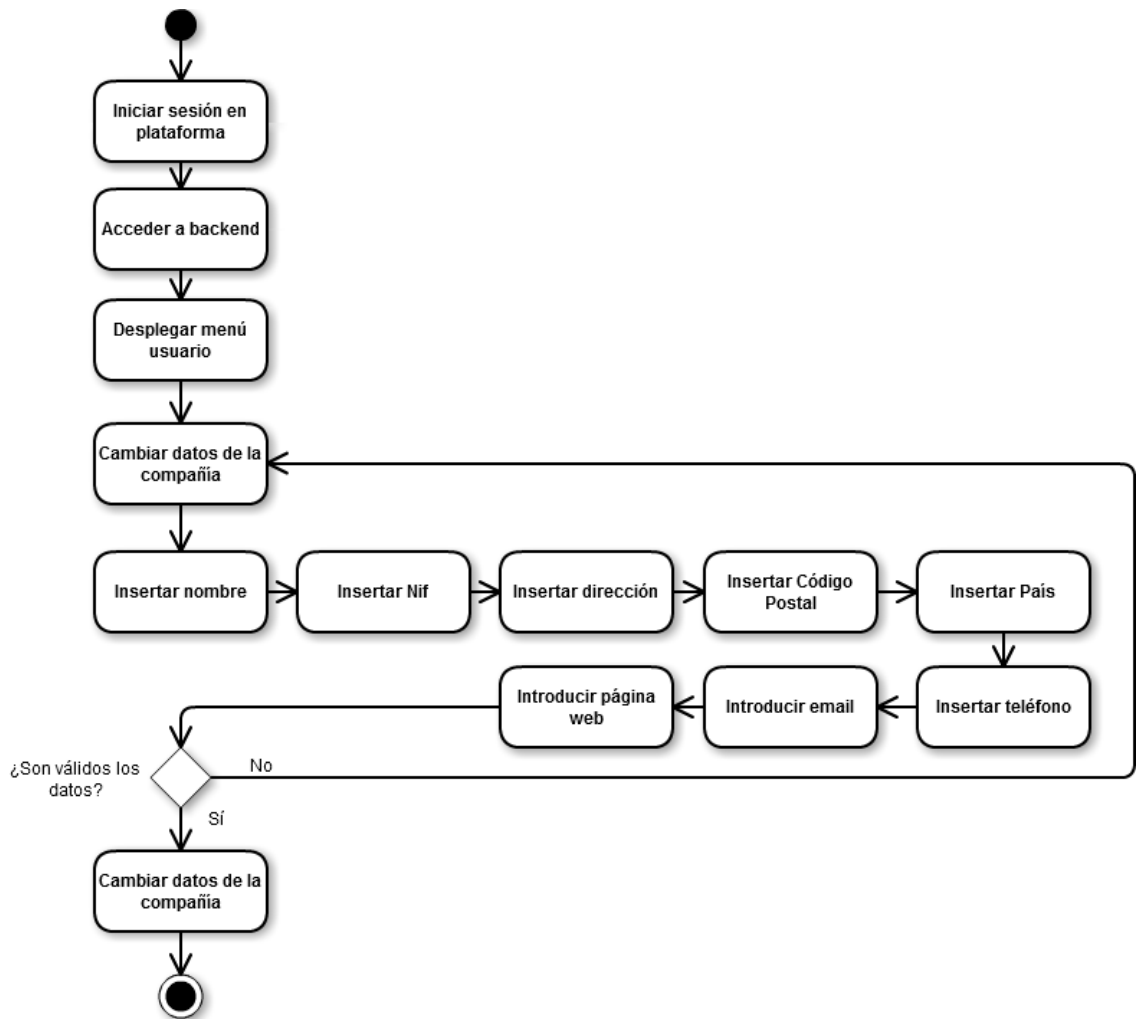
3.4.2. Inicio de sesión



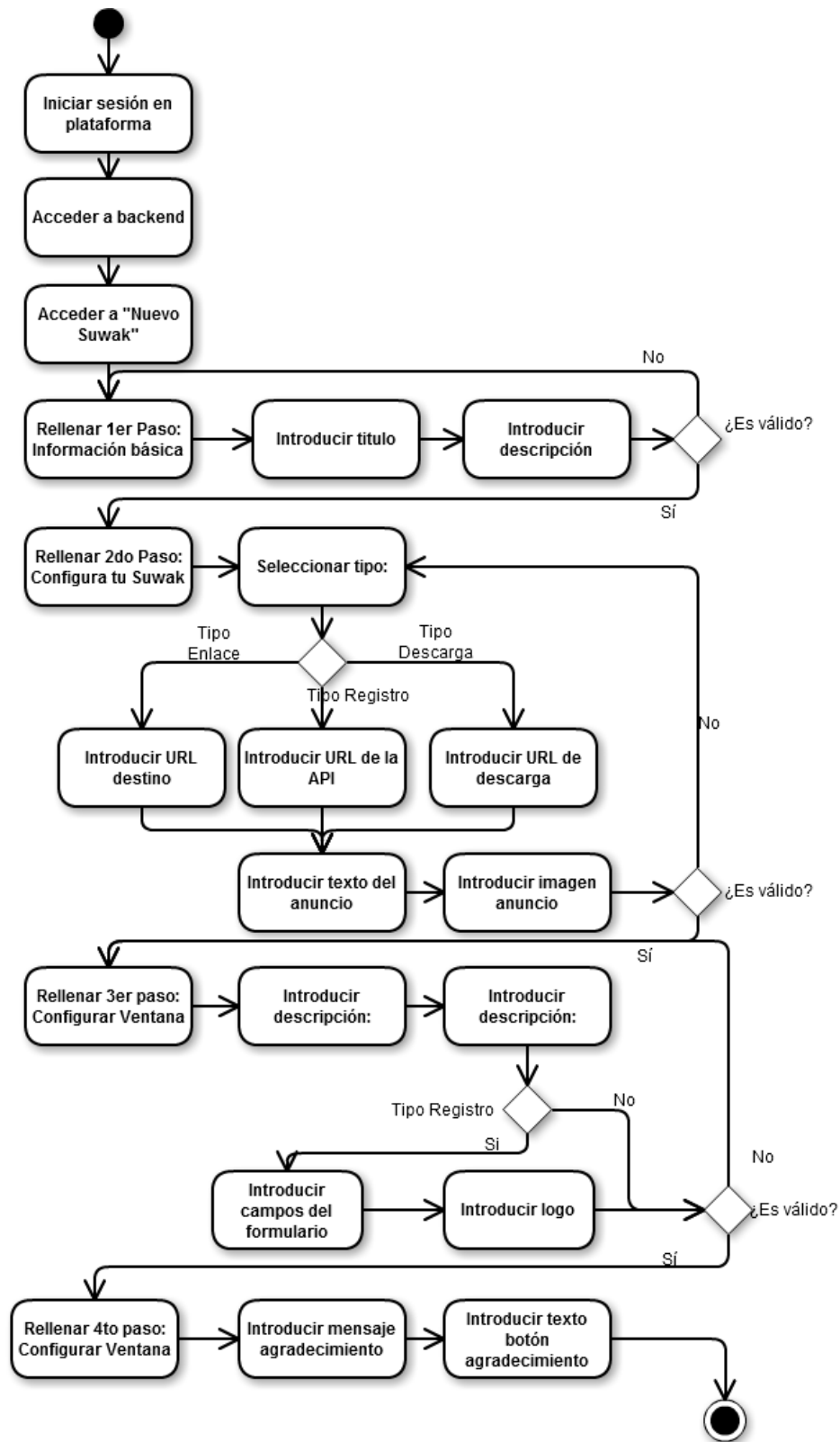
3.4.3. Cambiar datos de perfil



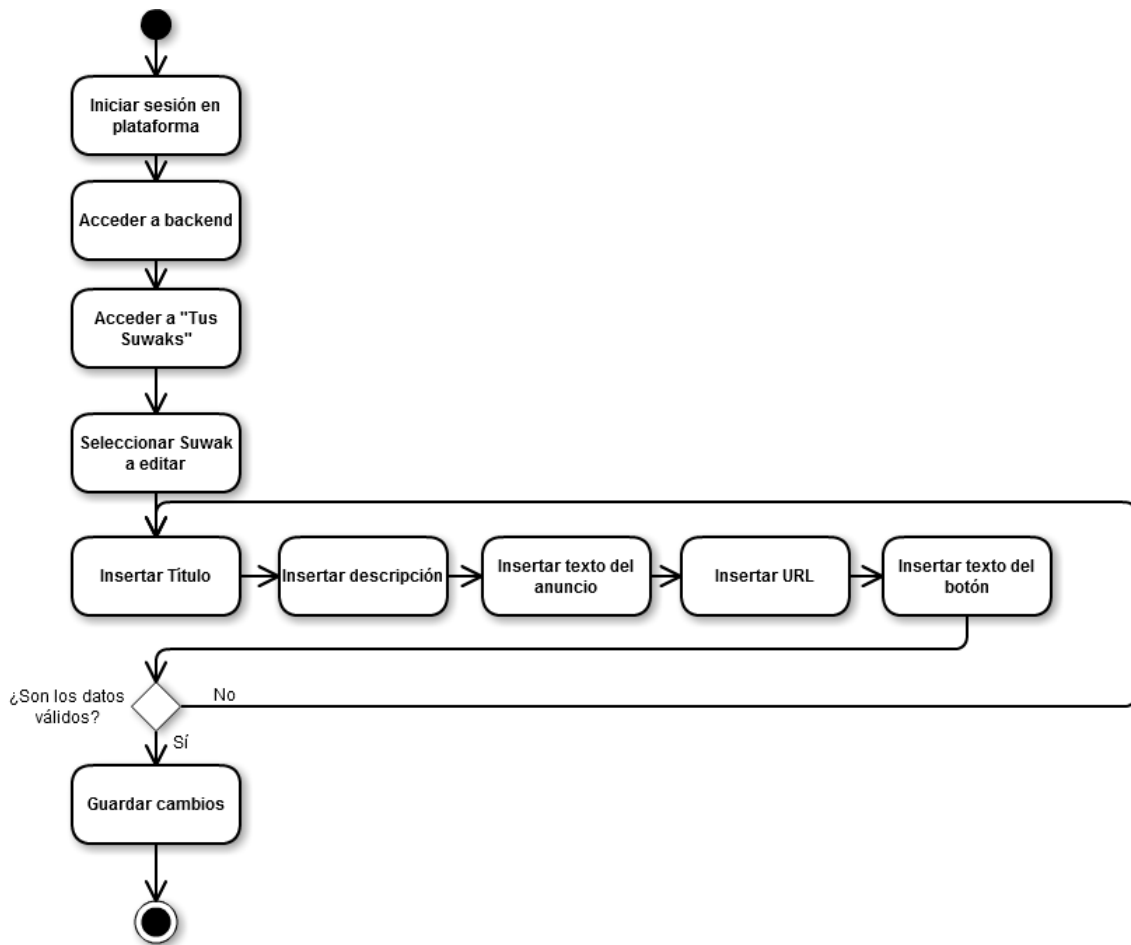
3.4.4. Cambiar datos de compañía



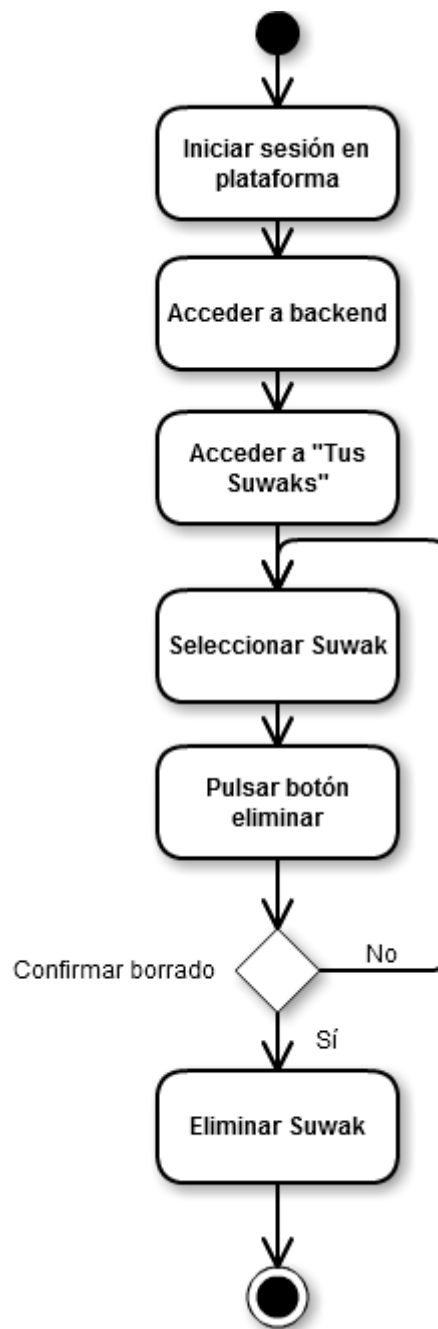
3.4.5. Nuevo Suwak



3.4.6. Modificar Suwak



3.4.7. Eliminar Suwak



4. Diseño

4.1. Arquitectura MVC

La aplicación Adsuwak, se ha llevado a cabo mediante el “*framework*” Symfony2. Es un “*framework*” PHP basado en la arquitectura MVC (Modelo-Vista-Controlador). Este patrón de diseño, introducido en los años 70 por Trygve Reenskaug, se caracteriza por dividir el código de la aplicación en tres capas diferentes. Estas divisiones lógicas son el modelo, la vista y el controlador. En la figura 4.1 se muestra un esquema que representa a esta arquitectura:

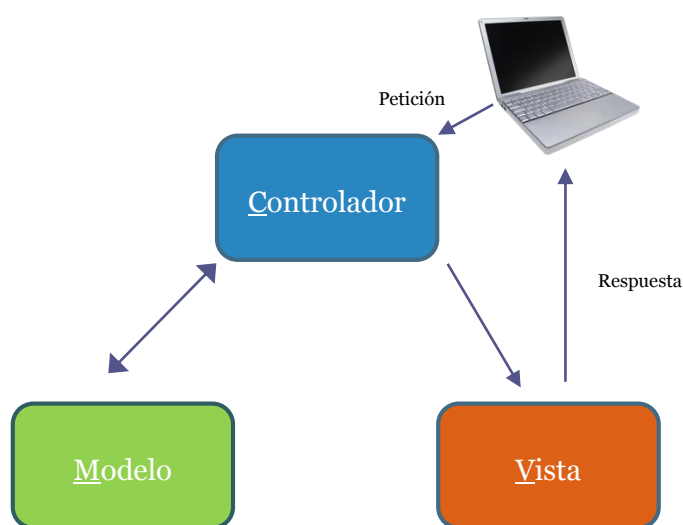


Figura 4.1 Esquema MVC

- **Capa Modelo:** La capa de modelado es la encargada de realizar toda la conexión con la base de datos y gestionar todas las operaciones que se realicen en ella.
- **Capa vista:** Es la responsable de representar toda la información por pantalla. Se encarga de la visualización de la información.
- **Capa controlador:** Se encarga del procesamiento de la información obtenida a través de la capa modelo y la prepara para devolverla a la capa de vista.

4.1.1.1. Flujo de ejecución

El flujo de ejecución de este modelo es el siguiente:

- En primer lugar, el usuario realiza una petición (“*Request*”) HTTP a nuestro sitio web. Esta petición es captada por el controlador de nuestra aplicación. El controlador procesa la petición y determina que operaciones debe ejecutar para poder construir la respuesta a la petición del usuario. En caso de que necesite obtener o modificar los datos de la aplicación se comunicará con la capa “Modelo” y le transmitirá que información necesita obtener.
- La capa Modelo procesa las instrucciones recibidas por el controlador e interactúa con la base de datos para obtener la información. Seguidamente le devuelve los resultados al controlador.
- Una vez el Controlador tenga toda la información necesaria y la haya procesado, le transmite a la capa Vista los resultados para que construya la respuesta al usuario.
- La capa vista ahora generará una respuesta (Response) representando la información mediante el lenguaje de marcas de hipertexto HTML y tecnologías como “CSS” o “*JavaScript*”.

Este modelo ha adquirido en los últimos tiempos una gran presencia en los distintos “frameworks” de desarrollo web. Con esta separación de la parte visual de la programación y del acceso a datos, se pretende modularizar la aplicación y con ello ofrecer un mayor grado de abstracción al programador.

4.1.2. Frontend y backend

La aplicación se organiza en dos partes muy diferenciadas en cuanto al uso de la aplicación.

La parte frontal de la aplicación, en adelante “*Frontend*” es la encargada de acoger a los usuarios que visitan por primera vez la aplicación web. En ella se mostrará información sobre que es Adsuwak y los datos de contacto y permitirá al usuario visitante registrarse en la aplicación.



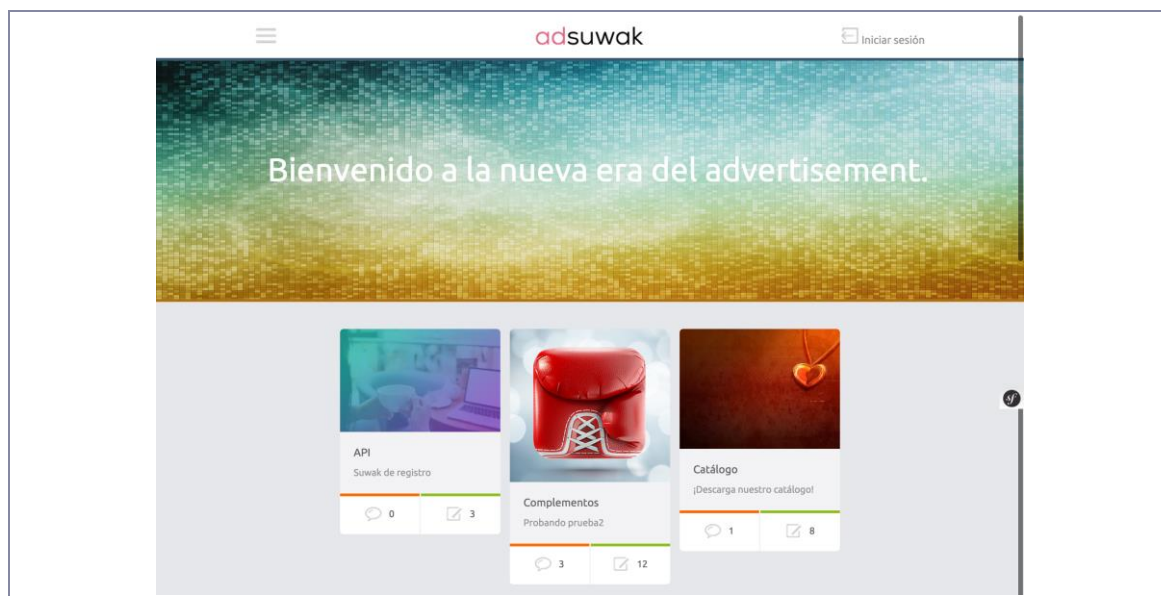


Figura 4.2 Imagen del Frontend de la aplicación

Esta parte de la aplicación debe presentar una estructura sencilla y atractiva para el usuario. No debe cargarse con demasiada información y poner exactamente aquello que el usuario debe saber. Será la primera impresión que se lleve el usuario acerca de Adsuwak, por lo que es de vital importancia que presente una estructura y organización adecuada.

Por otro lado, la parte interna de la aplicación o “Backend” será en la que los usuarios registrados puedan crear nuevos “Suwaks” y administrarlos.

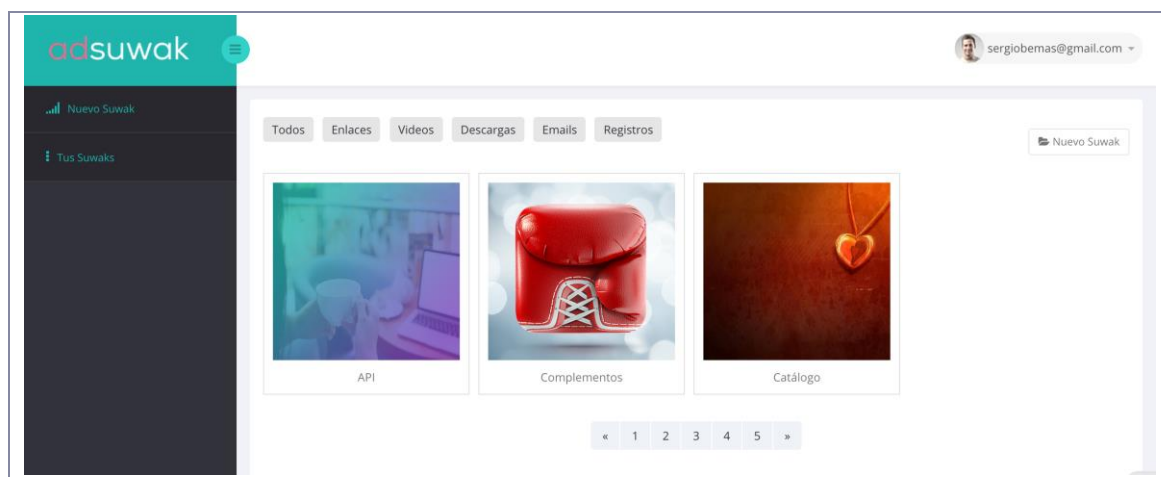


Figura 4.3 Imagen del Backend de la aplicación

En esta sección se deberá indicar adecuadamente u organizar debidamente las distintas opciones a realizar para que el usuario sepa en todo momento donde acudir y no se pierda en la aplicación.

Se ha elegido esta organización de la aplicación puesto que por un lado se necesita que la gente sepa que es Adsuwak y cómo funciona, y por otro lado aquellos que estén interesados en su utilización puedan personalizar su anuncio inteligente.

Para ello es necesario crear un área de restringido acceso a aquellos que se hayan registrado.

4.1.3. Estructura del proyecto

Siguiendo con la modularidad que propone el patrón MVC, el proyecto realizado se ha organizado en cuatro diferentes “*bundles*”.

- **“FrontendBundle”**: Será el encargado de mostrar la “*landing page*” de la web. Será la parte frontal de la aplicación y será el que atenderá todas las visitas de los usuarios a la web.
- **“BackendBundle”**: Es la plataforma dirigida a usuarios registrados. Desde aquí, los usuarios que hayan iniciado sesión, podrán crear nuevos banners y administrar sus existentes.
- **“APIBundle”**: Este módulo se encarga de la generación de los anuncios interactivos. Es donde reside el código propio de la aplicación. Será el encargado de servir los anuncios a los visitantes.
- **“UserBundle”**: es el utilizado para sobrescribir la información básica de “*FOSUserBundle*” tales como los formularios de registro y las vistas del “*login*” y del registro.

4.1.4. Diseño de los Suwaks

En este punto se describen los diferentes tipos de Suwaks disponibles en la aplicación.

Link Suwak

Es la renovación del banner. Su funcionamiento es el mismo salvo que el usuario no solo tiene que pulsar únicamente en la imagen. Se colocará el Suwak en el contenido de un blog y se le asignará una imagen relacionada con el contenido.





Figura 4.4 Suwak de tipo enlace

Para ello el usuario deberá configurar:

- Base del Suwak:
 - o Imagen de fondo: Imagen que aporte al contenido del texto.
 - o Texto relacionado con la imagen.
- Ventana desplegable:
 - o Texto
 - o La url del botón.

API Suwak

Este anuncio inteligente permite al usuario registrarse desde el mismo anuncio.



Figura 4.5 Suwak de tipo enlace

Este Suwak tiene que ser personalizable por el usuario, es decir, el creador del anuncio debe poder elegir que datos se van a incluir en el formulario. Por ejemplo puede añadir un campo para introducir el correo, o bien para que el usuario introduzca su página web.

Como máximo el anunciante podrá añadir tres campos al formulario. Esta elección se basa en que nadie rellenará un formulario que tenga muchos campos a rellenar.

De cada campo deberemos recoger la siguiente información:

- El tipo del campo de texto.
- El nombre de la variable.
- El “placeholder” o texto del campo de entrada del formulario.




Figura 4.6 Ejemplo formulario

La llamada Ajax deberá enviar dos cosas:

{nombre de la variable}:{contenido}

El nombre de la variable es el que se utilizará en el servidor para extraer el contenido de la variable.

Para crear este tipo de Suwak el usuario deberá configurar:

- Base del Suwak:
 - o Imagen de fondo: Imagen que aporte al contenido del texto.
 - o Texto relacionado con la imagen.
 - o La URL de la API (donde se enviarán los datos)
- Ventana desplegable:
 - o Logo de la empresa
 - o Descripción
 - o Los campos del formulario
 - o El texto del botón del formulario
- Texto agradecimiento:
 - o Texto
 - o Enlace

Download Suwak

Este tipo de Suwak puede ser utilizado por el anunciante para ofrecer su producto de una manera sencilla sin tener que usar intermediarios.



Figura 4.7 Suwak de tipo descarga

El usuario hará click en el botón del “Descargar” y mediante el enlace de descarga que habrá configurado el anunciante podrá descargar el archivo desde el mismo Suwak sin abandonar la web.

Para ello el anunciante deberá configurar desde la plataforma:

- Base del Suwak:
 - o Imagen de fondo: Imagen que aporte al contenido del texto.
 - o Texto relacionado con la imagen.
- Ventana desplegable:
 - o Descripción
 - o La URL del botón que contiene el enlace de descarga

5. Modelo de negocio

5.1. Introducción

Un modelo de negocio, según el libro “Business Model Generation” de Alexander Osterwalder y Yves Pigneur, describe las bases sobre las que una empresa crea, proporciona y capta valor. Mediante esta herramienta se establecen las pautas a seguir para atraer clientes y definir estrategias acerca del producto.

Existen varios tipos de modelos de negocio usados históricamente, no obstante, en el presente proyecto se utilizará el modelo conocido comúnmente como “Business Model Canvas”.

Este tipo de modelo de negocio permite realizar una evaluación dinámica del mercado y detectar sistemáticamente los elementos que generan valor al negocio. Para ello el autor propone trabajar en nueve bloques básicos mostrados en la figura 5.1 los cuales representan las diferentes áreas de la empresa.

En los siguientes apartados trabajaremos sobre cada una de estas áreas y detectaremos los elementos relevantes que conformarán la aplicación Adsuwak.

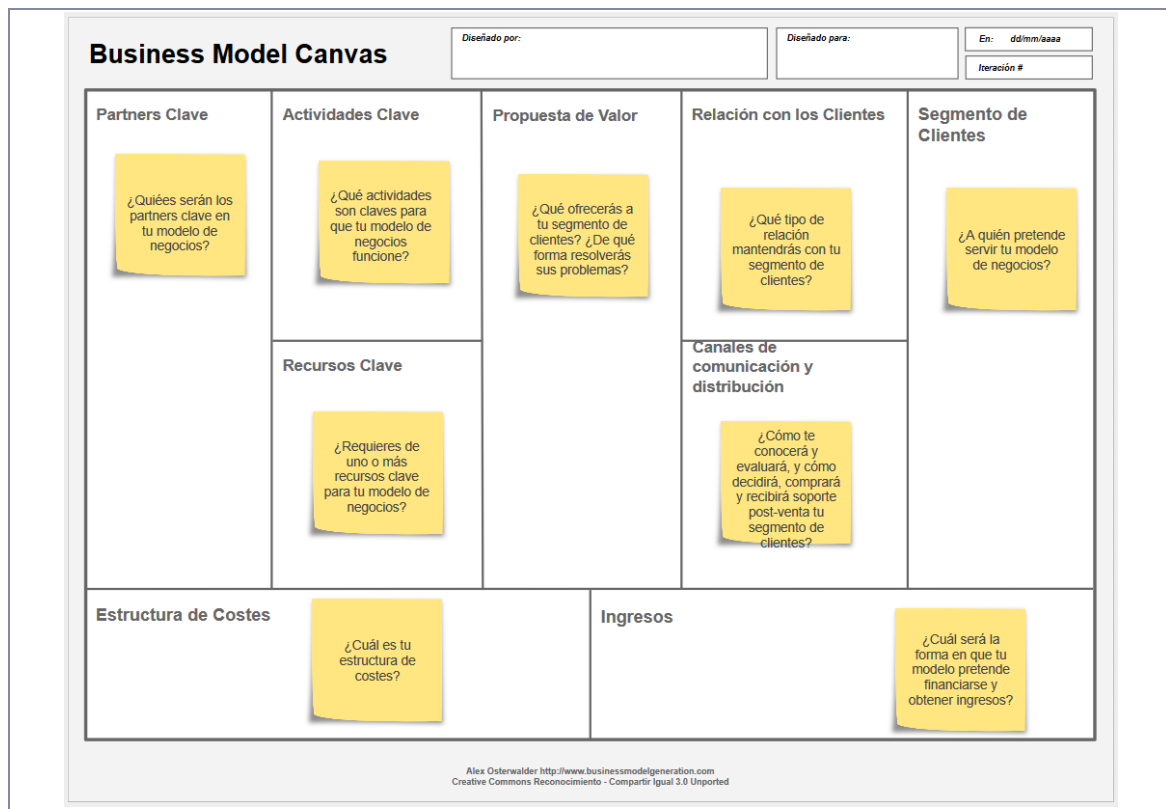


Figura 5.1 Business Model Canvas



5.2. Segmento de clientes

En este bloque se identifican y definen los clientes a los que está dirigida la nueva empresa. El proceso se divide en dos fases diferenciadas:

- **“Customer Discovery” (¿Quién es mi cliente?):** Es la fase de “descubrimiento de clientes”. Se averigua si existen clientes potenciales para el producto que estamos desarrollando. Para ello, es necesario ponerse en contacto con los posibles clientes que creamos que pueden estar interesados y recoger su opinión. No se trata de obtener opiniones para incorporar nuevas funcionalidades al producto, sino hablar con el cliente para detectar si tendría interés en obtener nuestro producto.

Los posibles usuarios que detectemos en esta etapa, serán los llamados “*early adopters*”, los cuales son el tipo de usuario que no le importa adquirir productos en fase de pruebas e inacabados para estar a la vanguardia tecnológica. Este tipo de usuario probablemente querrá adquirir nuestro producto aunque sea únicamente para probarlo.

- **“Customer Validation”:** En esta fase se debe trazar un plan de ventas con la finalidad de conseguir vender nuestro producto al mayor número posible de “*early adopters*”. Debe ser un plan repetitivo que podamos reproducir con todos los clientes diferentes mediante el cual podamos validar si el planteamiento que se realiza sobre el producto es el correcto. Cuando consigamos establecer un plan repetible de ventas y de descubrimiento de clientes será cuando validaremos que nuestro planteamiento es el correcto.

Tras aplicar las dos fases descritas anteriormente, se definen dos tipos usuarios a los que estará dirigida la aplicación:

-**Anunciantes:** Son los clientes de la aplicación. Serán aquellos que estén interesados en promocionar su producto. Para ello, accederán a la plataforma y crearán su *Suwak*. Idealmente serán las empresas de marketing que realicen marketing de contenidos las que se identificarán en este grupo.

-**Bloggers:** Son los usuarios de la aplicación. Este segmento corresponde a aquellos usuarios que deseen obtener alguna retribución económica por insertar anuncios en su página web.

5.3. Propuestas de valor

Este módulo describe cuales son las características de nuestro producto que lo diferencian del resto. Las propuestas de valor, son generalmente las que definen la elección del usuario a la hora de adquirir un producto u otro. Este conjunto de propuestas son las que satisfacen las necesidades de un determinado sector del mercado y constituye una serie de ventajas que la empresa ofrece a los clientes.

Para descubrir cuáles son las propuestas de valor de un producto se deben responder las siguientes preguntas:



- ¿Qué valor proporcionamos a nuestros clientes?
- ¿Qué problema soluciona nuestro producto?
- ¿Qué necesidades satisfacemos?

Una vez se haya sido capaz de encontrar respuesta a cada una de ellas tendremos los puntos claves que presenta el producto.

La aplicación en la que está basado todo el proyecto, tras aplicar lo descrito anteriormente, se ha obtenido la siguiente propuesta de valor:

Nuestra propuesta de valor, que nos distingue de otras empresas de anuncios, es la generación de nuevos anuncios inteligentes. Con estos nuevos anuncios se soluciona la baja conversión que presentan los banners actuales y la baja atención del usuario. Con ello, satisfacemos la necesidad de los anunciantes de dar a conocer su producto de una manera efectiva.

Para los “bloggers”, la plataforma añadirá un sistema de “matching” para potenciar la efectividad de los anuncios inteligentes.

5.4. Canales de distribución y comunicación

Una vez definido el cliente, el siguiente paso es definir los canales para acceder a él.

Los canales establecen la comunicación entre la empresa y el cliente. Es la manera por la que vamos a llegar al cliente.

Cumplen tres funciones básicas:

- **Función de comunicación:** Se utilizan para dar a conocer el servicio o producto.
- **Función de distribución:** Tienen el fin de hacer llegar el producto al destino objetivo, ya sea los distribuidores, los puntos de venta o a los mismos clientes.
- **Función de venta:** Facilitar la llegada del cliente a la empresa para bien adquirir nuevos productos o bien proporcionar atención postventa.

Para definir los canales de una empresa se deben responder a las siguientes preguntas:

- ¿Cómo establecemos actualmente el contacto con los clientes?
- ¿Qué canales prefieren nuestros segmentos de mercado?
- ¿Cómo se conjugan nuestros canales?
- ¿Cuáles tienen mejores resultados?
- ¿Cuáles son más rentables?

Los canales tienen 5 etapas de desarrollo, las cuales se utilizan para satisfacer diferentes necesidades y comunicarnos con los clientes:

- **Fase de información:** damos a conocer los productos y servicios de nuestra empresa.



- Fase de evaluación: Como ayudar a nuestros clientes para que nos den opinión sobre nuestra propuesta de valor.
- Fase de compra: Que los clientes puedan comprar nuestro producto.
- Fase de entrega: Como entregamos la propuesta de valor al cliente.
- Fase de postventa: Ofrecer un servicio de postventa del producto.

En Adsuwak, los diferentes canales que distinguimos son los siguientes:

- Plataforma Adsuwak.com
- Realizar llamadas telefónicas a empresas de marketing online
- Envío de correos electrónicos
- Publicación de artículos en webs tecnológicas.

5.5. Relaciones con el cliente

En este bloque se define el tipo de relación que establece la empresa con los diferentes tipos de cliente. Las relaciones se establecen para lograr diferentes objetivos:

- **Captación de clientes:** Lograr que el cliente utilice nuestro servicio en lugar del de la competencia.
- **Fidelización de clientes:** Mantener al cliente contento con el servicio.
- **Estimulación de la venta:** Lograr venderle al cliente cada vez más productos.

Los clientes de un producto, según el “*business model canvas*”, esperan recibir por parte de la empresa los siguientes tipos de canales:

- **Asistencia humana:** Se basa en la interacción humana. El cliente contacta con un representante de la empresa.
- **Asistencia personal exclusiva:** Un representante de la empresa, se dedica exclusivamente al caso de un cliente determinado.
- **Autoservicio:** La empresa pone a disposición del usuario todos los medios requeridos para que pueda cumplir con sus necesidades por ellos mismos.
- **Servicios automáticos:** Los servicios automáticos distinguen entre los usuarios y les ofrece la información personalizada según sus necesidades.
- **Comunidades:** Disponen los clientes de un sistema que les permita interactuar con otros usuarios y resolver cualquier duda que les surja.
- **Creación colectiva:** La empresa espera que el usuario colabore para desarrollar conjuntamente el producto.

En nuestro caso, la relación con los clientes se establecerá principalmente a través de la plataforma. No obstante, debido a que es una primera versión de la herramienta, aquellos que requieran de atención personalizada, sin distinción entre “*bloggers*” o anunciantes, podrán ponerse en contacto con nosotros directamente a través de llamadas telefónicas o correo electrónico.

5.6. Fuentes de ingresos

En este módulo se definen los flujos mediante los que vamos a obtener alguna retribución económica.

Según el “canvas” de Alexander Osterwalder, existen dos tipos de ingresos:

- Ingresos por transacciones derivadas de pagos puntuales de clientes
- Ingresos recurrentes derivados de pagos periódicos a cambio del suministro de una propuesta de valor o servicio de post venta de atención al cliente

Los ingresos obtenidos están fuertemente ligados al precio asignado a la propuesta de valor. Es muy complicado acertar el precio del producto de primeras, por lo que deberán realizarse constantemente validaciones para ajustar el precio correcto.

Los precios establecidos pueden ser de dos tipos:

- **Precios fijos:** Consiste en establecer un precio fijo para un producto basándose en la competencia o bien en la calidad de la propuesta de valor ofrecida.
- **Precios dinámicos:** Este tipo de precios cambian en función del mercado.

En Adsuwak, la principal fuente de ingresos será el precio que cobremos al anunciante por las acciones realizadas con éxito desde los anuncios interactivos.

5.7. Recursos clave

Los recursos clave de una empresa son aquellos de los que depende la empresa para su funcionamiento. Cualquier problema con un recurso clave supondría la paralización de la empresa.

Existen varias categorías en las que podemos clasificar los recursos clave:

- **Físicos:** Se incluyen los activos físicos tal como pueden ser las instalaciones donde se alojan los servidores.
- **Intelectuales:** Corresponden a los recursos intelectuales aquellos como licencias, patentes y la propiedad intelectual.
- **Humanos:** Se refieren a los recursos humanos de una empresa, es decir, los trabajadores.
- **Económicos:** Corresponden al dinero necesario para cubrir cualquier problema o bien utilizarlo para adquirir nuevos elementos.

En Adsuwak distinguimos varios recursos clave. El primero de ellos es la plataforma que permite al usuario la generación de nuevos anuncios. En segundo lugar, los servidores en los que se realiza todo el procesamiento de los datos y la generación del código de los nuevos anuncios. En tercer lugar, todo el departamento encargado de atención al cliente. Por último el departamento de marketing encargado de dar a conocer la nueva herramienta.



5.8. Actividades clave

En esta sección se describen aquellas actividades que debe llevar a cabo la empresa para que funcione su modelo de negocio. Estas actividades son las que deben ejecutarse para lograr el éxito y poder crear una propuesta de valor.

Las actividades clave se pueden dividir en las siguientes categorías:

- Producción: aquellas actividades que están orientadas a crear productos en grandes cantidades.
- Resolución de problemas: Son las encargas de proporcionar soluciones a los problemas de los clientes.
- Plataforma red: actividades que se preocupan del correcto funcionamiento de la plataforma.

Para nuestro modelo de negocio distinguimos de las siguientes actividades clave.

- Investigación de nuevas tecnologías para el desarrollo de nuevos tipos de anuncios
- Mantenimiento plataforma y servidor
- Contacto con los clientes y obtener su opinión.

5.9. Asociaciones clave

Las asociaciones clave son aquellas relaciones establecidas entre la empresa y la red de proveedores o socios.

En la mayoría de empresas se realizan este tipo de asociaciones para optimizar sus modelos de negocio, reducir riesgos y adquirir nuevos productos.

En el libro “Business Model Generation” de Alex Osterwalder se distinguen entre tres tipos de motivaciones para establecer relaciones clave:

- **Optimización y economía de escala:** Pertenecen a este tipo de relación, aquellas que busquen como objetivo optimizar la asignación de recursos y actividades para reducir costes.
- **Reducción de riesgos e incertidumbre:** Las empresas realizan este tipo de relaciones con la finalidad de afianzarse en el mercado.
- **Compra de determinados recursos y actividades:** Ocurren cuando las empresas recurren a otras organizaciones para obtener servicios y poder aumentar así su capacidad.

En nuestro caso un “*partner*” podría ser una agencia de marketing la cual nos proporcionase una cartera de clientes.

5.10. Estructura de costes

En este último apartado del “*canvas*” se definen los costes que conlleva poner en funcionamiento el modelo de negocio. Se consideran costes gastos tales como son la compra de materias primas, pago del personal, hospedaje de la plataforma en los servidores...

La finalidad es conocer todos los gastos que se deben asumir para poder estimar correctamente las ganancias.

El “*business model canvas*” distingue entre dos tipos de estructuras de costes, las que están realizadas en torno a los costes y por otra parte a las dirigidas en torno al valor.

Las del primer grupo se caracterizan por recortar el máximo número de gastos siempre y cuando sea posible proponiendo propuestas de valor de bajo precio. Por otra parte, las propuestas enfocadas al valor del producto se caracterizan por centrarse en la creación de valor.

En toda estructura de costes, se presentan las siguientes características:

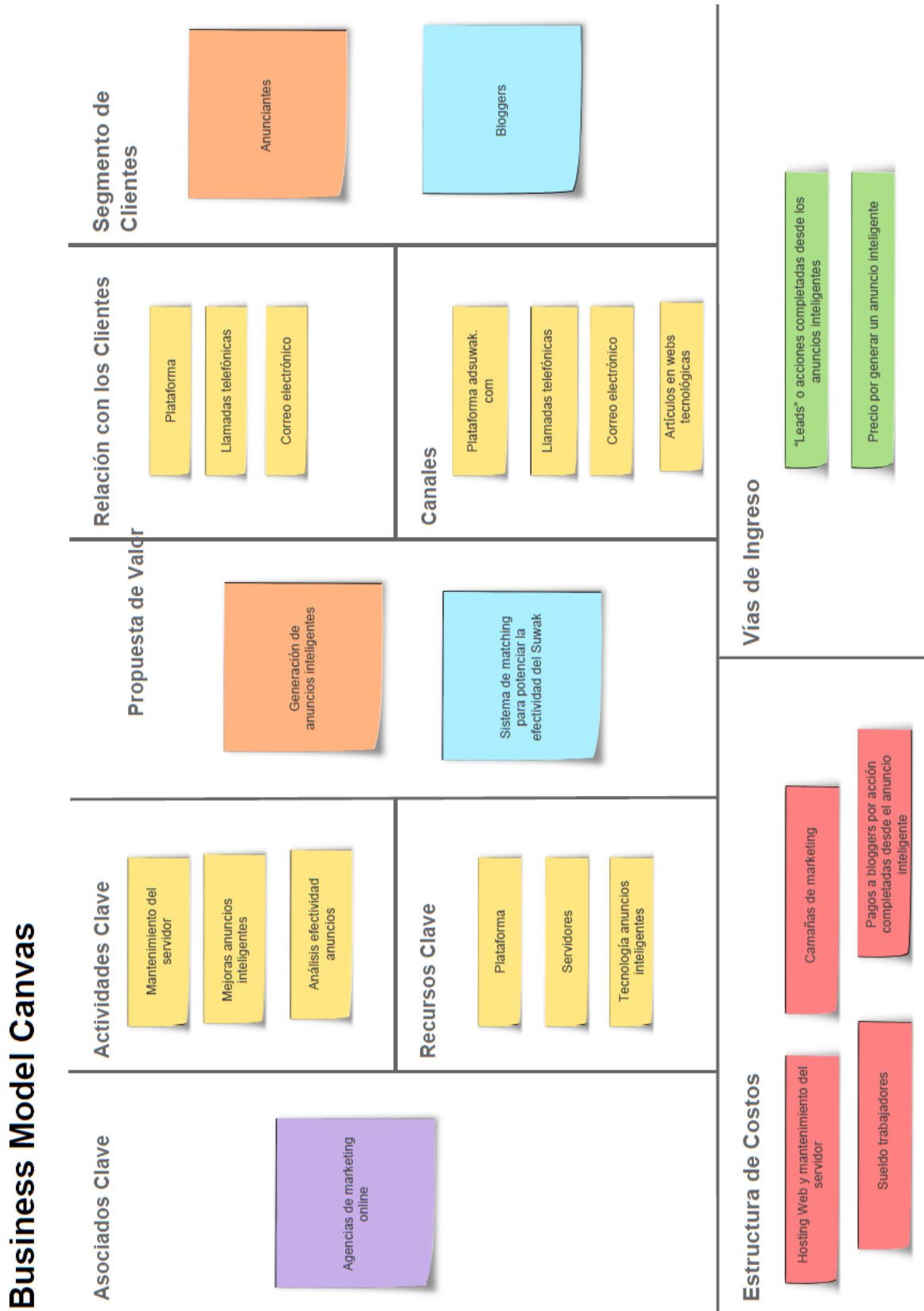
- Costes fijos: Son aquellos que no varían en función del volumen de las ventas o servicios producidos.
- Costes variables: Son aquellos que dependen directamente del volumen de la cantidad de productos producidos.
- Economías de escala: Los costes de adquisición del producto dependen del volumen de compra.
- Economías de campo: Los costes utilizados para hacer una acción, deben ser los mismos siempre que se produzcan en el mismo campo para otra acción.

En nuestro caso, los costes que presenta Adsuwak son los siguientes:

- Hosting web y mantenimiento del servidor
- Marketing
- Sueldo trabajadores.
- Pagos a bloggers por acción completada desde el anuncio interactivo.



Una vez finalizado el estudio de los diferentes bloques, se adjunta en la figura 5.2 el “business model canvas” resultante del análisis de mercado de Adsuwak.



Creado por Marcelo Pizarro Miranda en base al trabajo de Alex Osterwalder <http://www.businessmodelgeneration.com> y a la inspiración de David Bland <http://agile-dzone.com/news/how-to-create-business-model>
 Alex Osterwalder <http://www.businessmodelgeneration.com>
 Creative Commons Reconocimiento - Compartir Igual 3.0 Unported

Figura 5.2. Adsuwak Business Model Canvas



6. Implementación

6.1. Introducción

En este capítulo se explica de una manera muy detallada las diferentes decisiones que se han tomado en el desarrollo de la aplicación así como los diferentes aspectos técnicos que conlleva. Para ello, en los casos que lo precisen, se añade a la explicación capturas de pantalla con el fin de complementar la información y clarificar todo el proceso de desarrollo.

Esta sección se divide en tres apartados. En primer lugar, se realizará una descripción básica de las diferentes tecnologías utilizadas en el proyecto. Seguidamente, en el apartado 6.3 se tratan funciones concretas que están presentes en la aplicación como la gestión de usuarios o la traducción de textos. Finalmente, en la sección “Instalación del servidor” se lleva a cabo la puesta en producción de la aplicación en un hosting web. Se realiza la configuración básica del servidor habiéndolo adquirido desde cero hasta la completa configuración de la aplicación.

6.2. Principales tecnologías

6.2.1. HTML 5

HTML, “*HyperText Markup Language*” (lenguaje de marcas de hipertexto) es el lenguaje utilizado para la creación de páginas web. Define la estructura básica y contenido que contendrá una página web (texto, imágenes, videos, etc.).

En la aplicación se utiliza el estándar HTML 5, el cual es la quinta revisión del lenguaje HTML publicada en octubre del año 2014. Se utiliza esta versión para aprovechar todo el potencial que ofrece el lenguaje HTML.

6.2.2. JavaScript

JavaScript es un lenguaje de programación interpretado, el cual es utilizado para crea páginas web dinámicas.

Es utilizado en el lado del cliente, implementado en el navegador del usuario y está orientado al trabajo con la interfaz de usuario. Permite realizar animaciones y alterar de una forma dinámica el contenido del documento generado.

Se ha utilizado en el proyecto para la generación de los anuncios inteligentes y para realizar alguna animación en los archivos HTML.



6.2.3. CSS

CSS, “*cascading style sheets*” (hojas de estilo en cascada), son un conjunto de reglas que permiten modificar la apariencia visual de los documentos HTML. La idea básica de CSS es separar la estructura del documento de la parte visual.

Actualmente los navegadores tienen formas distintas de interpretar los estilos CSS por lo que en algunos casos un estilo definido en una página web podría no verse correctamente en un navegador específico. Es necesario adaptar los estilos creados a todos los navegadores.

6.2.4. PHP 5

PHP, “*Hypertext Preprocessor*”, es un lenguaje de programación del lado del servidor, el cual es utilizado para generar contenido dinámico. El código es interpretado por un servidor web y permite incorporar nuevo código HTML directamente en el documento.

Symfony2 está totalmente basado en PHP5 por lo que su utilización está presente en todo el proyecto.

6.2.5. Ajax

Ajax, “*Asynchronous JavaScript And XML*” permite realizar una comunicación asíncrona entre en el navegador y el servidor. De esta forma es posible crear páginas web interactivas modificando su contenido sin necesidad de recargarla mejorando la usabilidad con el usuario.

Está ligado al uso de “*JavaScript*”. En el proyecto su utilización se utiliza en el Suwak de tipo registro, en el que se envían los datos a un servidor externo desde el mismo anuncio.

6.2.6. JQuery

JQuery es una biblioteca externa de *JavaScript* la cual simplifica la interacción con el lenguaje HTML, realizar cambios en el contenido, añadir animaciones y realizar llamadas Ajax de una manera mucho más sencilla.

Fue creada por inicialmente por John Resig y actualmente es raro encontrar una página web que no utilice *JQuery*.

6.2.7. Bootstrap

Bootstrap es un framework desarrollado por Mark Otto y Jacob Thornton, el cual permite desarrollar páginas web utilizando HTML, CSS, JavaScript y en algunos casos JQuery.

Incorpora una serie de elementos por defecto definidos con lo que permite al usuario mediante un catálogo incorporarlos fácilmente en su proyecto.

En la aplicación se utiliza bootstrapp en el “template” utilizado en el backend.

6.2.8. Controlador de versiones: GIT

Para el correcto desarrollo del proyecto utilizaremos un controlador de versiones con el fin de tener registrados los cambios que se van realizando.

Utilizaremos la herramienta “*Git*”. Esta herramienta es un software controlador de versiones creado por Linus Torvalds en el año 2005. Fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Permite para mantener un historial de cambios de los archivos fuente de una aplicación y una base de datos local la cual va registrando cada cambio que se van realizando sobre los archivos pudiendo así llevar un control sobre las modificaciones realizadas.

Además facilita el trabajo en equipo. Cada desarrollador tiene su propio repositorio en el que se registran todos los cambios que se realizan en el código común.

Repositorio online: Bitbucket

Como complemento opcional utilizaremos el repositorio remoto Bitbucket. Los dos repositorios más utilizados son Bitbucket y GitHub, no obstante debido a que Bitbucket nos resulta más familiar y es gratis nos decantamos por su utilización.

6.2.9. Xampp

Es una conocida herramienta de desarrollo utilizada por los desarrolladores de aplicaciones web. Esta herramienta da lugar a un entorno de desarrollo formado por las siguientes tecnologías:

- Servidor web Apache.
- Base de datos MySQL
- PHP
- Perl



Cuenta actualmente con una gran comunidad de usuarios que recomiendan su utilización. Con este entorno de desarrollo podremos realizar diferentes pruebas de la aplicación a lo largo de su desarrollo sin necesidad de tener que subir la aplicación al servidor final.

Nos decantamos por su utilización debido a que es software libre y además con la instalación de un solo paquete podemos disponer de todas las tecnologías que posteriormente necesitaremos en un entorno de producción.

6.2.10. Symfony2

Symfony es un framework PHP basado en la arquitectura Modelo-Vista-Controlador el cual ha sido diseñado para optimizar el desarrollo de aplicaciones web.

Está formado por un conjunto de diferentes tecnologías que en su totalidad logran formar un framework muy estable. Integra tecnologías tales como “Doctrine”, un potente ORM (“*Object-Relational mapping*”) el cual se encarga de la comunicación con la base de datos, “Twig” el cual es un potente generador de plantillas basado en PHP y el lenguaje YAML el cual permite crear archivos de configuración muy legibles.

A su misma vez, incorpora un mecanismo de generación de formularios el cual nos permite mantener todos los formularios situados en un mismo lugar pudiendo así reutilizarlos en lugares diferentes.

Symfony incorpora una potente consola de comandos que permite generar automáticamente algunos elementos tales como nuevos “*bundles*” o nuevas entidades.

Otro aspecto a resaltar, dispone de un gran número de librerías externas disponibles que extienden su funcionamiento básico lo cual permite solucionar algunos problemas básicos muy fácilmente y centrarse en el desarrollo real de la aplicación.

Este framework cuenta actualmente con diferentes versiones. Actualmente, la última versión disponible es la 2.7. Además dispone de las versiones LTS (*Long Term Support*) las cuales tienen asignado un periodo mayor de mantenimiento. En el proyecto se utilizará la versión LTS 2.3.37 debido a que tiene asignada un periodo de revisión y mantenimiento hasta mayo del año 2016.

Symfony es actualmente muy conocido por su gran grado de abstracción que ofrece al usuario y cuenta además con el soporte de una gran comunidad de usuarios que recomiendan su utilización y con una extensa documentación.

En los siguientes apartados se realizará una instalación desde cero y trataremos los principales temas de configuración a realizar en el proyecto.

6.2.10.1. Instalación

Para la descarga del framework accedemos a la página web oficial de descarga de Symfony, <http://symfony.com/download> y seleccionamos la versión 2.3.



Use **Symfony 2.6**
for the latest features

end of support
July 2015

Use **Symfony 2.3**
for long-term support

end of support
May 2016

🕒 First, [install the Symfony Installer](#). Then, execute this command:

```
$ symfony new my_project 2.3
```

📄 COPY

Figura 6.1 Descargar Symfony 2.3

Symfony proporciona un instalador para facilitar su descarga. En primer lugar nos descargamos el instalador siguiendo las siguientes instrucciones:

🕒 First, [install the Symfony Installer](#). Then, execute this command:

The **Symfony Installer** is a small PHP application that must be installed once in your computer. It greatly simplifies the creation of new projects based on the Symfony framework.

Installation on Linux and Mac OS X

```
$ sudo curl -Ls http://symfony.com/installer -o /usr/local/bin/symfony
$ sudo chmod a+x /usr/local/bin/symfony
```

Installation on Windows

```
c:\> php -r "readfile('http://symfony.com/installer');" > symfony
```

Move the downloaded file to your projects directory and execute it as `php symfony`.

Figura 6.2 Instrucciones descarga instalador

Con el instalador descargado, debemos ejecutar ahora el comando:

```
$ php symfony new nombre_del_proyecto 2.3
```

Una vez se haya terminado la ejecución del comando, situaremos la carpeta descargada en la carpeta “*htdocs*” del servidor Apache2 y probar su funcionamiento.

Para acceder, deberemos escribir en el navegador la dirección:

http://localhost/Symfony/web/app_dev.php



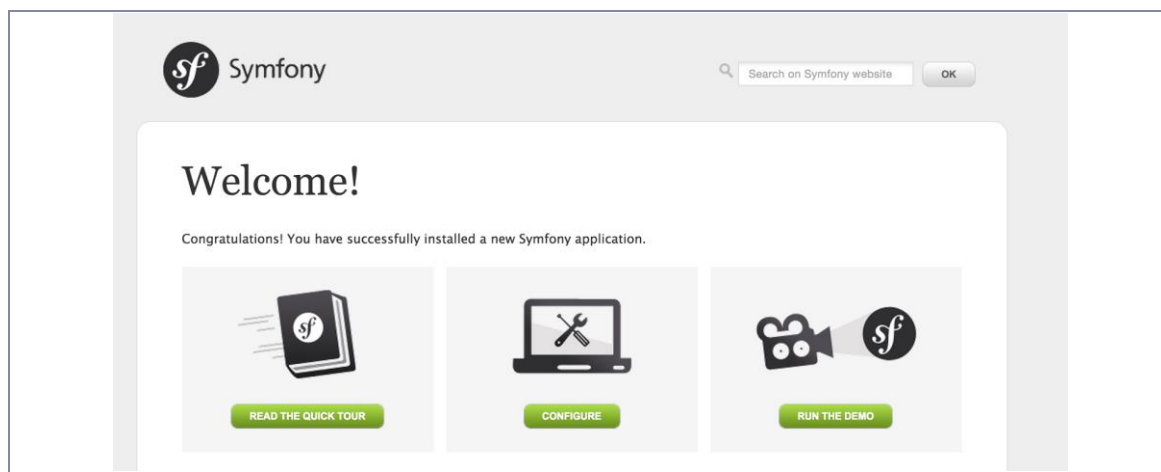


Figura 6.3 Instalación satisfactoria

6.2.10.2. Configuración

El siguiente paso a realizar una vez hayamos instalado Symfony será ejecutar el configurador básico. Para ello en la figura que se muestra al final del apartado anterior debemos pulsar en “Configure”.

El asistente nos permite configurar la conexión con la base de datos de una forma muy sencilla. Para ello deberemos rellenar el formulario que se muestra en la 6.4.

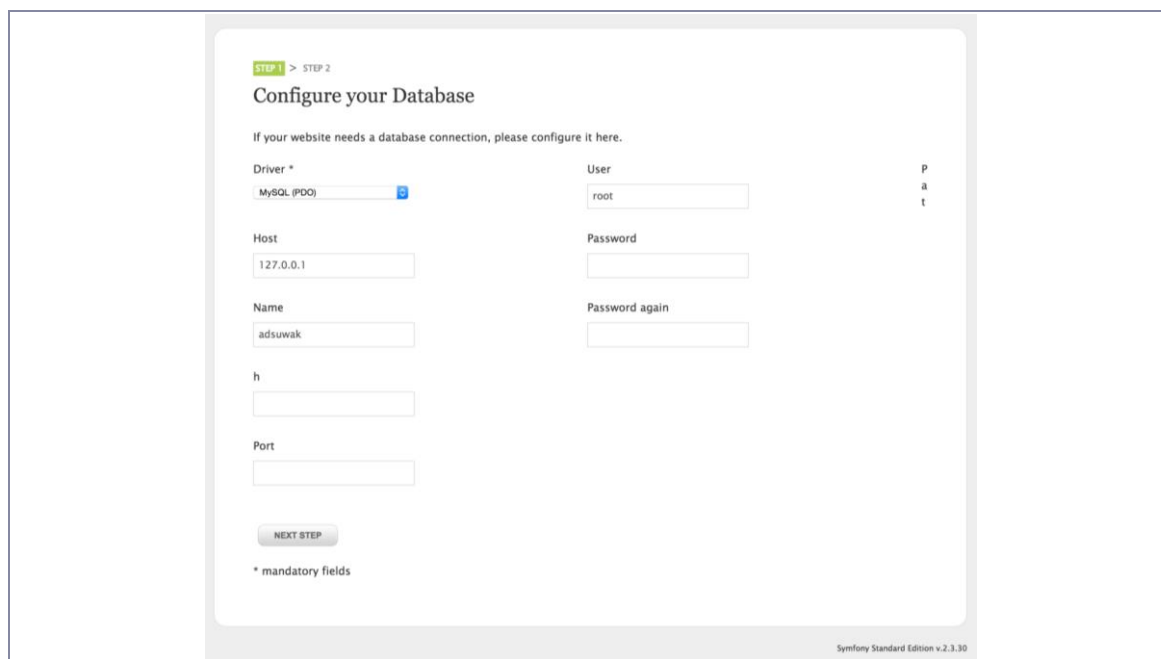


Figura 6.4 Asistente configuración Symfony

Los datos insertados corresponden a la base de datos local que se instala con el entorno XAMPP.

Al pulsar en siguiente nos indicará que insertemos un “Secreto Global” el cual se utilizará posteriormente para la validación de formularios frente ataques CSRF.

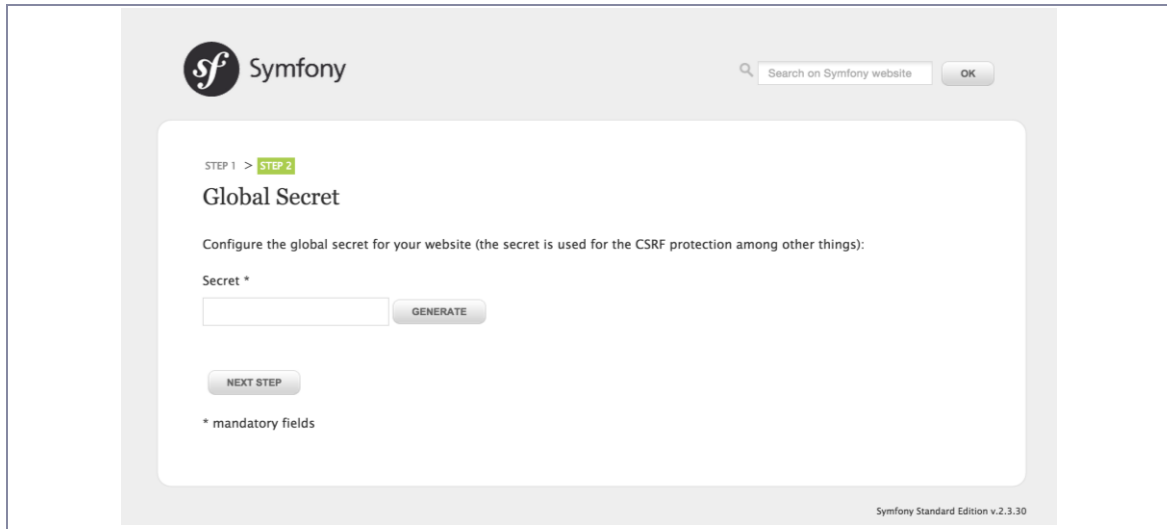


Figura 6.5 Token CSRF

Al pulsar en siguiente, se nos mostrará la imagen de los parámetros que ha sido insertado en el archivo “*parameters.yml*”.

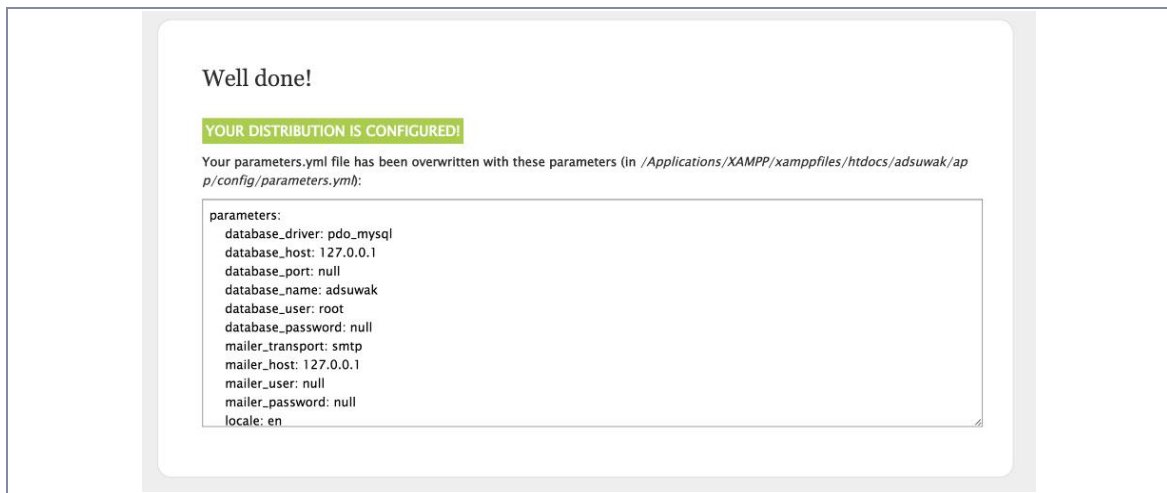


Figura 6.6 Configuración satisfactoria

6.2.10.3. Cambiando los permisos

Cuando se realiza una primera instalación del framework, es muy común que el servidor devuelva el siguiente error:

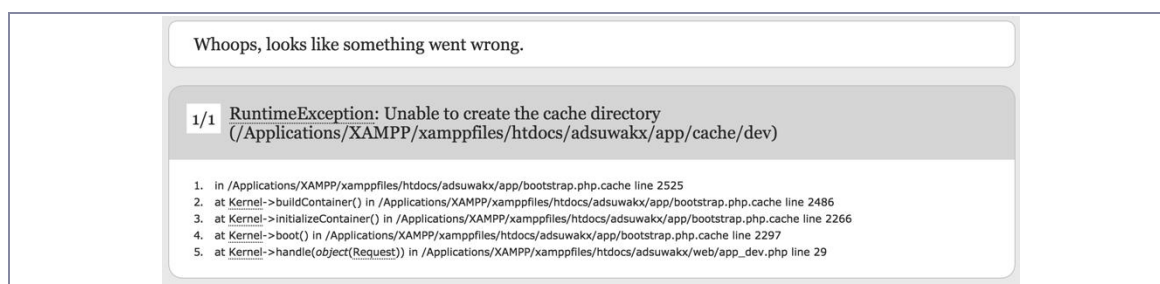


Figura 6.7 Error permisos escritura

Esto es debido a que Symfony intenta modificar las carpetas “app/cache” y “app/logs” pero no tiene permisos para hacerlo. Symfony lo pueden ejecutar dos “usuarios” distintos, es decir, el servidor y la consola. Para solucionar este error debemos modificar los permisos de estas carpetas en cuestión para que puedan acceder todos los usuarios.

6.2.11. Composer

Es un gestor de dependencias en PHP. Es utilizado en Symfony2 para facilitar su instalación y su administración.

Symfony es un framework formado por un conjunto de librerías externas las cuales deben ser actualizadas constantemente y en caso de querer actualizarlo sería necesario actualizar cada una de las dependencias y comprobar si hay incompatibilidades.

Composer, mediante una lista de dependencias, realiza esta tarea. Comprueba cada librería y decide cuales y que versión hay que instalar y el orden de instalación. La lista de dependencias del proyecto se puede encontrar en el archivo “composer.json”. En él se incluyen el nombre de los paquetes y su versión.

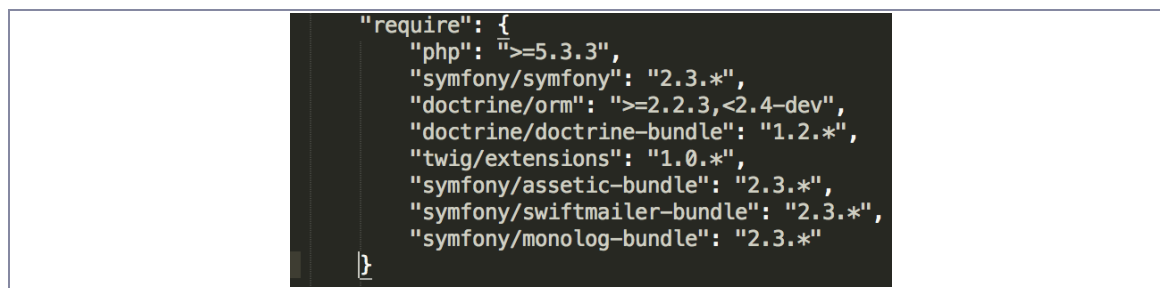


Figura 6.8 Ejemplo composer.json

Instalación

Para instalar “Composer” en el nuevo proyecto debemos descargar el archivo “composer.phar” del sitio oficial de Composer. <https://getcomposer.org/download/>

Con el archivo descargado debemos copiarlo en la carpeta raíz del proyecto.

Ahora para actualizar todas las dependencias del proyecto debemos ejecutar el siguiente comando el cual actualizará todos los componentes de nuestra aplicación:

```
$ php composer.phar update
```

6.2.12. Creando bundles

Symfony organiza toda la funcionalidad en diferentes “bundles” o módulos. Con ello se pretende modularizar la aplicación y poder utilizarlo en diferentes proyectos sin necesidad de repetir código. Un “bundle” es un directorio el cual tiene una estructura de carpetas y archivos definidos. Tiene un identificador el cual será utilizado para referenciarlo en las diferentes partes del proyecto.

Los “bundles” propios son situados en la carpeta “/src” del proyecto y los que son proporcionados por terceros se encuentran en el directorio “vendor/”.

Veamos los diferentes módulos que se han creado en la aplicación:

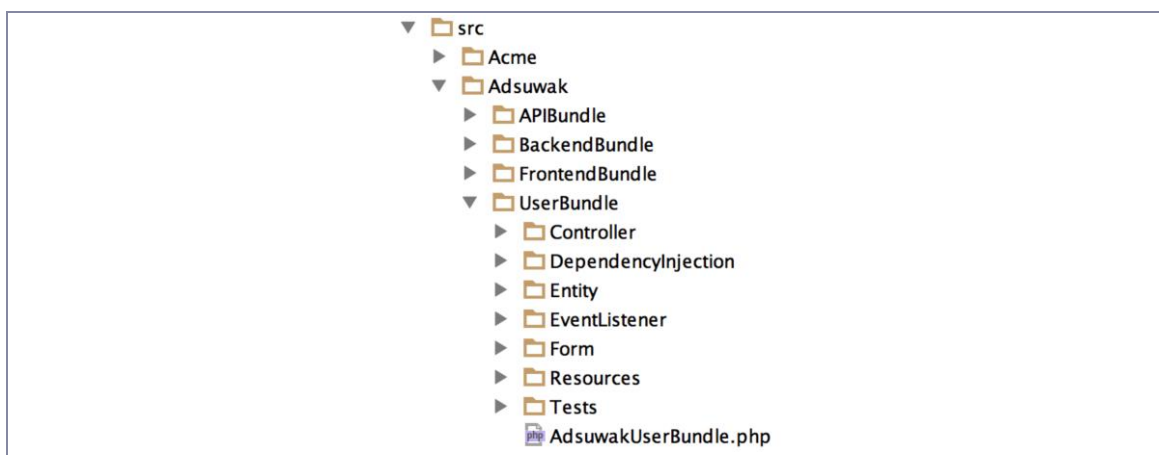


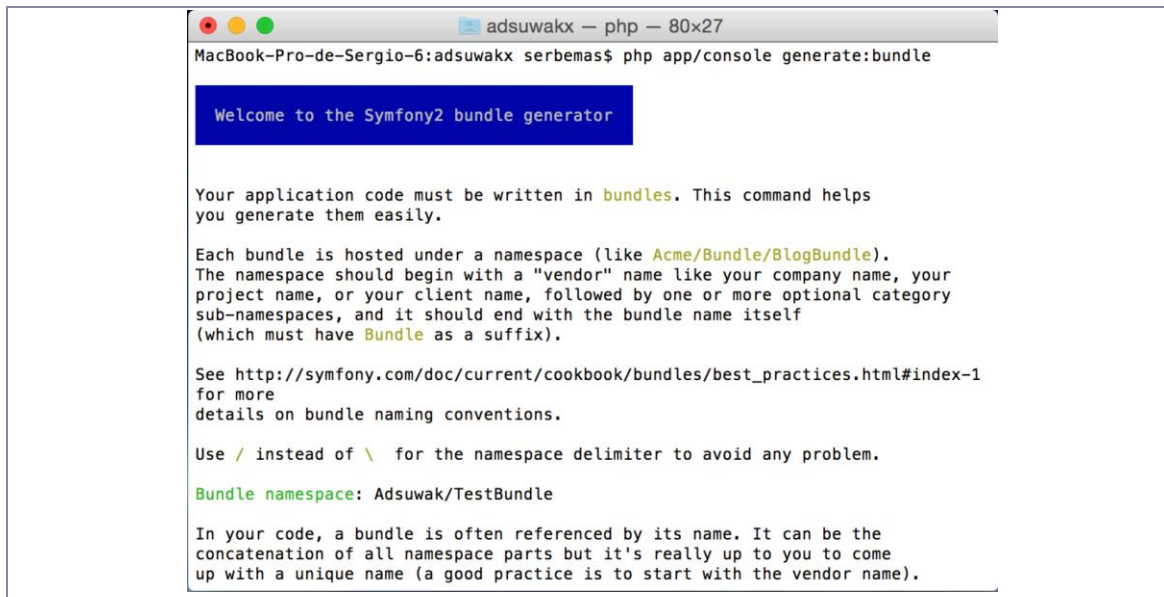
Figura 6.9 Bundles del proyecto

Tal y como se puede observar en la 6.9, el módulo “UserBundle” presenta en su interior una estructura de directorios. Esta estructura está presente en todos los “bundles”.

Podemos crear tantos módulos como necesitemos en la aplicación. Para ello debemos ejecutar el siguiente comando

```
$ php app/console generate:bundle
```

Este comando ejecutará un asistente el cual nos permitirá crear un nuevo módulo respondiendo a una serie de preguntas.



```
MacBook-Pro-de-Sergio-6:adsuwakx serbemas$ php app/console generate:bundle

Welcome to the Symfony2 bundle generator

Your application code must be written in bundles. This command helps
you generate them easily.

Each bundle is hosted under a namespace (like Acme/Bundle/BlogBundle).
The namespace should begin with a "vendor" name like your company name, your
project name, or your client name, followed by one or more optional category
sub-namespaces, and it should end with the bundle name itself
(which must have Bundle as a suffix).

See http://symfony.com/doc/current/cookbook/bundles/best_practices.html#index-1
for more
details on bundle naming conventions.

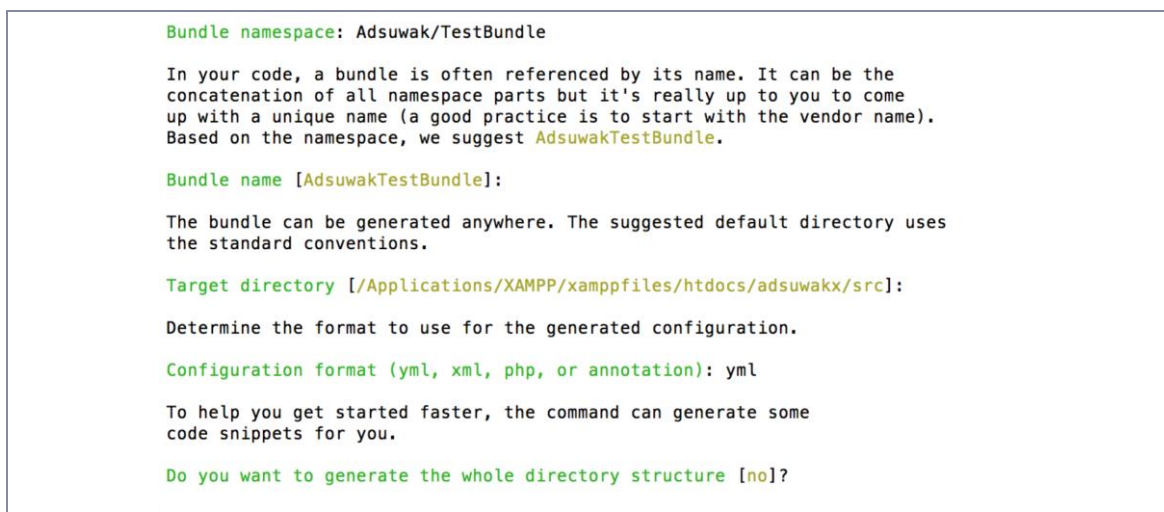
Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace: Adsuwak/TestBundle

In your code, a bundle is often referenced by its name. It can be the
concatenation of all namespace parts but it's really up to you to come
up with a unique name (a good practice is to start with the vendor name).
```

Figura 6.10 Asistente creación bundle. Parte 1.

“Bundle namespace” es el directorio en el que se creará el nuevo “bundle”. Seguidamente nos preguntará por el nombre del mismo, el directorio donde será almacenado y el formato de archivo que se utilizará en los archivos de configuración.



```
Bundle namespace: Adsuwak/TestBundle

In your code, a bundle is often referenced by its name. It can be the
concatenation of all namespace parts but it's really up to you to come
up with a unique name (a good practice is to start with the vendor name).
Based on the namespace, we suggest AdsuwakTestBundle.

Bundle name [AdsuwakTestBundle]:

The bundle can be generated anywhere. The suggested default directory uses
the standard conventions.

Target directory [/Applications/XAMPP/xamppfiles/htdocs/adsuwakx/src]:

Determine the format to use for the generated configuration.

Configuration format (yml, xml, php, or annotation): yml

To help you get started faster, the command can generate some
code snippets for you.

Do you want to generate the whole directory structure [no]?
```

Figura 6.11 Asistente creación “bundle”. Parte 2.

Finalmente deberemos confirmar si todos los datos son introducidos correctamente y confirmar que se actualice el archivo “AppKernel.php” y se active el “bundle” en la aplicación.


```
Summary before generation

You are going to generate a "Adsuwak\TestBundle\AdsuwakTestBundle" bundle
in "/Applications/XAMPP/xamppfiles/htdocs/adsuwakx/src/" using the "yaml" format.

Do you confirm generation [yes]? yes

Bundle generation

Generating the bundle code: OK
Checking that the bundle is autoloaded: OK
Confirm automatic update of your Kernel [yes]? yes
Enabling the bundle inside the Kernel: OK
Confirm automatic update of the Routing [yes]? yes
Importing the bundle routing resource: OK

You can now start using the generated code!
```

Figura 6.12 Asistente creación “bundle”. Parte 3.

6.2.13. Generando la base de datos con Doctrine

Como hemos mencionado en la introducción, Symfony2 utiliza el ORM, en inglés “*Object Relational Mapping*”, Doctrine.

Mediante este ORM, Symfony no trabaja directamente con las tablas de la base de datos sino que cada tabla se mapea en un objeto en la aplicación cuyas propiedades serán los campos de la tabla.

Con esto se trabaja con un modelo orientado a objetos lo cual permite abstraer al programador de las diferentes sentencias que debe ejecutar para almacenar la información en la base de datos. Además, permite abstraer al desarrollador del motor de la base de datos que se esté utilizando.

En los siguientes puntos de esta sección crearemos las tablas de la base de datos y realizaremos varios tipos de relaciones.

6.2.13.1. Creando la base de datos

En el punto 6.2.2 hemos configurado la conexión del framework con la base de datos. Antes de proceder a la creación de las entidades de la base de datos debemos crear la base de datos. Para ello ejecutaremos el siguiente comando:

```
php app/console doctrine:database:create
```

6.2.13.2. Entidades

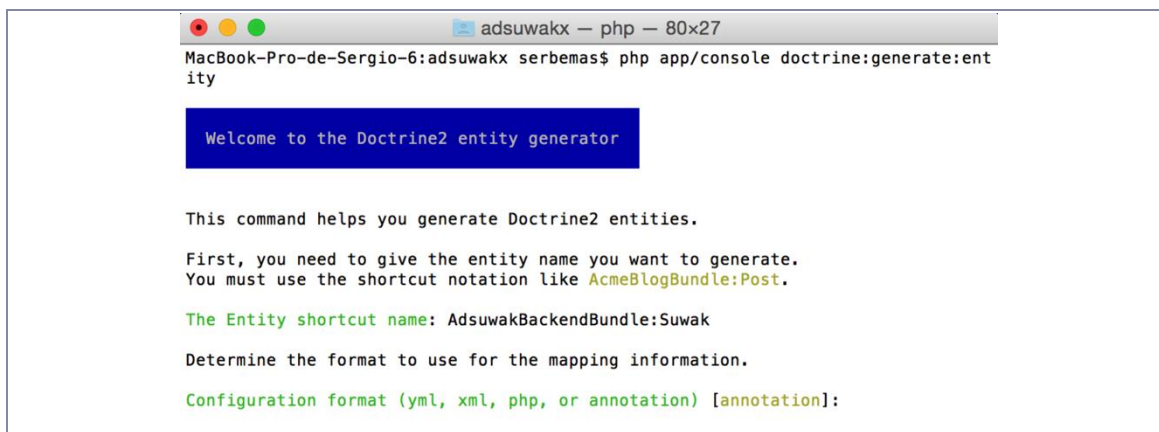
Para generar una tabla en la base de datos necesitamos crear una entidad en la aplicación.

Una entidad es la representación de una tabla de la base de datos. Por ello necesitaremos crear tantas entidades como tablas necesitemos en la BD.

Para crear una nueva entidad deberemos ejecutar el siguiente comando:

```
$ php app/console doctrine:generate:entity
```

El siguiente comando ejecuta un asistente que respondiendo una serie de preguntas nos permite crear automáticamente la entidad.



```
MacBook-Pro-de-Sergio-6:adsuwakx serbemas$ php app/console doctrine:generate:entity

Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate.
You must use the shortcut notation like AcmeBlogBundle:Post.

The Entity shortcut name: AdsuwakBackendBundle:Suwak

Determine the format to use for the mapping information.

Configuration format (yml, xml, php, or annotation) [annotation]:
```

Figura 6.13 Asistente creación entidades. Parte 1

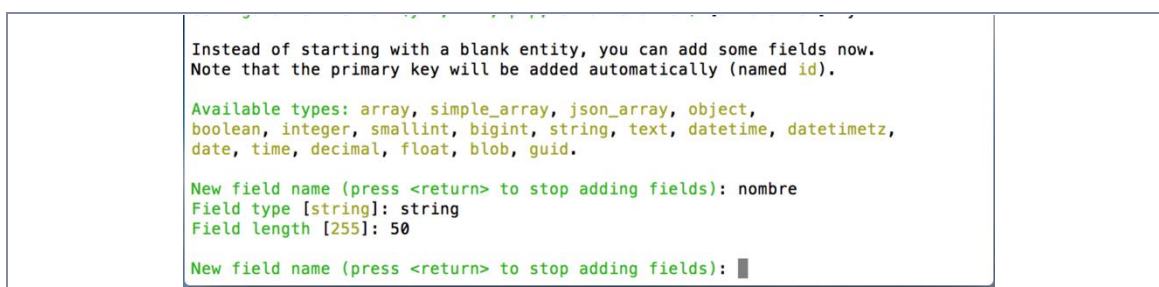
El asistente en primer lugar nos pedirá que insertemos el nombre de la entidad. Deberemos seguir el siguiente patrón:

```
Identificador_del_bundle: Nombre_entidad
```

En este caso añadimos lo siguiente: “AdsuwakBackendBundle:Suwak” y pulsamos “Enter”.

Seguidamente deberemos insertar el formato en el que se creará entidad. Doctrine nos permite elegir entre distintos formatos. Como la mayoría de información utiliza anotaciones nosotros optamos también por este formato. Para seleccionar basta con pulsar “Enter”.

El siguiente paso, nos empezará a preguntar cuáles serán las columnas de la nueva tabla. Por cada columna deberemos introducir su nombre, seleccionar el tipo y la longitud.



```
Instead of starting with a blank entity, you can add some fields now.
Note that the primary key will be added automatically (named id).

Available types: array, simple_array, json_array, object,
boolean, integer, smallint, bigint, string, text, datetime, datetimetz,
date, time, decimal, float, blob, guid.

New field name (press <return> to stop adding fields): nombre
Field type [string]: string
Field length [255]: 50

New field name (press <return> to stop adding fields):
```

Figura 6.14 Asistente creación entidades. Parte 2

Insertaremos tantas columnas como necesitemos en la aplicación. No es necesario crear el atributo Id que actúa de clave primaria puesto que el comando la genera automáticamente.

Para terminar de crear la entidad demos pulsar “Enter” en lugar de introducir un nuevo nombre de la propiedad. Contestamos que sí y contestamos afirmativamente el resto de preguntas

En este punto el asistente nos preguntará si deseamos crear un repositorio de la entidad.

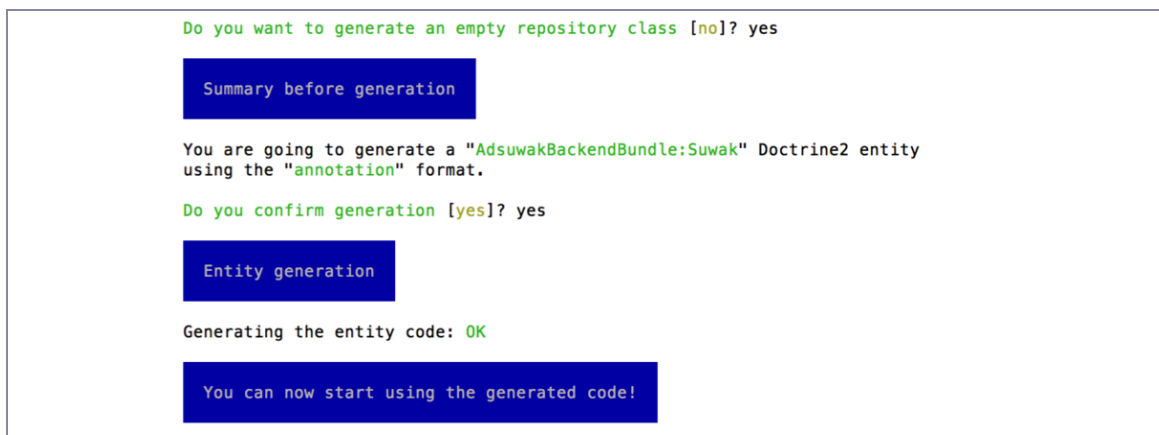


Figura 6.15 Asistente creación entidades. Parte 3

Una vez finalizado el asistente, se habrán creado en la carpeta “Entity” del “bundle” que hemos seleccionado dos archivos, “Suwak.php” y “SuwakRepository.php”.

Suwak.php representa al objeto que luego Doctrine insertara en la base de datos y “SuwakRepository.php” se utilizará para generar código DQL y poder realizar nuevas consultas.

Con la entidad nueva creada debemos ahora crear la estructura de tablas en la base de datos. Para ello ejecutamos el siguiente comando:

```
$ php app/console doctrine:schema:update
```

Este proceso lo deberemos repetir para todas las entidades que necesitemos crear en la aplicación.

6.2.13.3. Creando asociaciones entre objetos

Una vez hayamos creado todas las entidades u objetos de la aplicación necesitaremos establecer las relaciones entre ellas. Los objetos creados pueden relacionarse de distintas formas:

- Un objeto A puede estar asociado a muchos objetos B, pero uno B solo puede estar asociado A. Esta relación es llamada uno-a-muchos (“one to many”).



- Muchos objetos A pueden estar asociados a un sólo objeto B, y B puede estar asociados a A. Esta relación es llamada muchos-a-uno (“*many to one*”).
- Uno o varios objetos A pueden estar asociados a uno o varios objetos B y viceversa. Esta relación es llamada muchos a muchos (*many to many*),
- Un sólo objeto A puede estar asociado con un sólo objeto B. Esta relación es llamada uno-a-uno (*one to one*).

Para crear una asociación One-To-Many y Many-To-One utilizaremos las entidades User y Adsuwak las cuales contienen una relacion de este tipo.

Una compañía (“Companies.php”) puede tener asignados varios Suwaks (“Adsuwak.php”) pero un Suwak solo puede pertenecer a una compañía.

En la clase “Companies.php” definiremos el siguiente atributo con las siguientes anotaciones:

```
/**
 * @ORM\OneToMany(targetEntity="Adsuwak" , mappedBy="companies" , cascade={"all"})
 */
protected $adsuwak;
```

Figura 6.16 Ejemplo relación OneToMany

Las anotaciones que van entre /* y */ indican que el atributo Adsuwak establece una relación del tipo “OneToMany”.

“TargetEntity” hace referencia a la clase con la que se establece la relación, y “mappedBy” hace referencia al atributo que se utiliza en dicha clase para hacer referencia a “Companies”.

Por contrario, en la clase “Adsuwak.php” definiremos el siguiente atributo:

```
/**
 * @ORM\ManyToOne(targetEntity="Companies" , inversedBy="adsuwak" , cascade={"persist"})
 */
protected $companies;
```

Figura 6.17 Ejemplo relación ManyToOne

De nuevo, las anotaciones establecen una relación del tipo “ManyToOne” con “Companies”. El campo que hace función de clave ajena en la entidad “Companies” se llama “adsuwak”.

Debido a que la entidad Companies puede tener asignado varios objetos de tipo Adsuwak, en el constructor deberemos crear una estructura para almacenar los diferentes objetos.

```

public function __construct()
{
    $this->images = new \Doctrine\Common\Collections\ArrayCollection();
    $this->advertisements = new \Doctrine\Common\Collections\ArrayCollection();
    $this->adsuwak = new \Doctrine\Common\Collections\ArrayCollection();
    $this->token = substr(md5(uniqid(rand(), true)), 0, 32);
    $this->dateReg = new \DateTime('now');
}

```

Figura 6.18 Ejemplo estructura ArrayCollection

Esta estructura es la denominada “ArrayCollection” utilizada comúnmente en este tipo de relaciones.

Para el resto del tipo de relaciones se debe seguir el mismo proceso, únicamente que en la relación “OneToOne”, al únicamente utilizar un objeto de cada tipo no es necesario utilizar “ArrayCollection”

6.2.13.4. Consultando la información.

Para obtener la información almacenada en las tablas podemos utilizar los siguientes métodos:

- findAll(): Obtiene todos los registros de la tabla. Devuelve un “array”.
- find(): Obtiene un registro a partir de la clave primaria de una tabla.
- findBy(): Obtiene los registros a partir de un atributo en concreto. Retorna un “array”.
- findOneBy(): Es similar al anterior, pero en lugar de devolver un “array” devuelve un único elemento.

Un ejemplo de funcionamiento sería en el método “editAction()” cuando recuperamos el objeto “Adswuwak” a partir de un id para modificar su información.

```

$adsuwak = $em->getRepository('AdswuwakBackendBundle:Adswuwak')->find($id);

```

Figura 6.19 Ejemplo obteniendo un objeto Adswuwak

6.2.13.5. Manipulación de datos

Para trabajar con la base de datos, en primer lugar debemos utilizar el servicio de Doctrine.

```

$em = $this->getDoctrine()->getManager();

```

Figura 6.20 Cargando doctrine

Para insertar un nuevo objeto en la base de datos deberemos en primer lugar crear dicho objeto, y seguidamente utilizar los “setters” de la entidad para actualizar la información. Veamos un ejemplo:



```
$step = new Steps();  
$step->setAction("Eliminado Suwak");  
$step->setIp($this->container->get('request')->getClientIp());  
$step->setDate(new \DateTime());  
$step->setUser($user);
```

Figura 6.21 Ejemplo insertando objeto en la base de datos

Seguidamente, una vez tengamos el objeto creado deberemos persistirlo en la base de datos. Esto se realiza añadiendo las dos siguientes instrucciones:

```
$em->persist($step);  
$em->flush();
```

Figura 6.22 Persistiendo el objeto

6.2.14. Generando las vistas con Twig

Symfony utiliza por defecto el motor de plantillas Twig. Es un motor de plantillas desarrollado para trabajar con el lenguaje PHP con el objetivo de facilitar el trabajo a los desarrolladores que trabajen con el patrón MVC.

Las principales características de Twig son las siguientes:

- **Rápido:** compila los “*templates*” a código PHP optimizado.
- **Seguro:** Dispone de un módulo para evaluar el código no verificado mejorando la seguridad.
- **Flexible:** Ofrece al usuario una gran cantidad de etiquetas que puede utilizar el usuario y además permite realizar nuevas.

Una de sus principales características, es que permite añadir en una plantilla estructuras de flujo de control. Con estas estructuras podremos recorrer “*arrays*” como lo haríamos en PHP y trabajar individualmente con cada elemento.

Por otra parte implementa un sistema de herencia de plantillas el cual permite trabajar la reutilización de código en las diferentes partes de la aplicación.

Ofrece además una sintaxis sencilla e intuitiva que permite al usuario aprender este nuevo lenguaje de una manera muy cómoda.

6.2.14.1. Usos básicos

Estas son las principales estructuras utilizadas en Twig:

- **{# comentario #}**: Esta estructura es la utilizada para realizar comentarios en el código.
- **{{ mostrar_algo }}**: Se utiliza para mostrar el contenido de una variable.
- **{% hacer_algo %}**: Permite realizar acciones como puede ser recorrer un “*array*”.

Todas estas funciones se pueden hacer directamente en PHP. No obstante, es bastante más legible las instrucciones utilizadas por Twig que con PHP.



- `{{ variable.nombre | upper }}`: Twig además incorpora una series de filtros que permite modificar la información fácilmente.

Otra funcionalidad es el uso de la estructura de control if-else.

```

{% if variable.Nombre %}

{% endif %}
```

Mediante esta sentencia podremos generar el código deseado según si se cumple una condición.

6.2.14.2. Herencia de plantillas

Con Twig podemos utilizar una estructura de plantillas multinivel. Para ello proporciona los bloques y el método `“extends”` para indicar que plantilla hereda.

Con este mecanismo en el proyecto definimos un `“layout”` en el cual por ejemplo situamos todos los elementos que deberán estar presentes en todas las páginas.

Para las vistas del backend hemos utilizado la siguiente estructura:

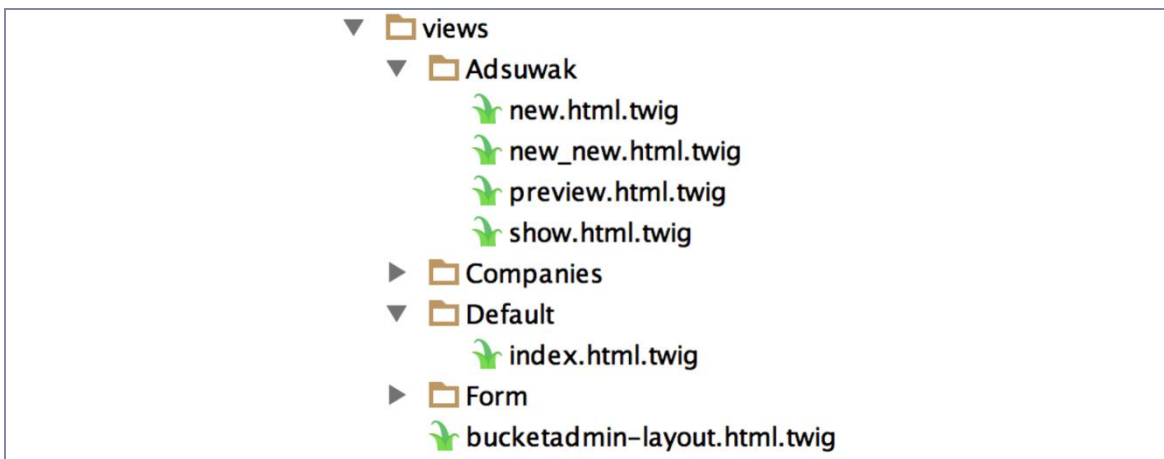


Figura 6.23 Estructura vistas backend

En el archivo `“bucketadmin-layout.html.twig”` es el archivo del que extenderán las demás plantillas. En el hemos situado el menú de usuario y el menú lateral izquierdo.

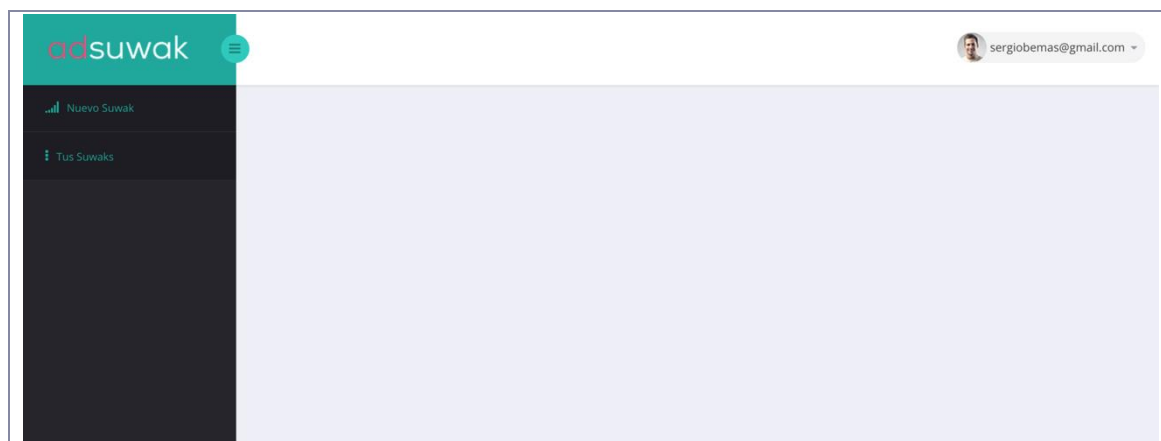


Figura 6.24 Representación Layout

La plantilla está dividida en diferentes bloques:

- **{% block stylesheets %}**: En él se sitúan todas las hojas de estilo
- **{% block head %}**: Corresponde al “head” del lenguaje “html”
- **{% block body %}**: Corresponde al “body” del lenguaje “html”
- **{% block javascripts %}**: Se sitúan en él todas las librerías JavaScript.

El contenido se organiza en bloques debido a que así cualquier plantilla que herede este “layout” podrá añadir nuevo contenido a los bloques fácilmente.

Seguidamente, en el directorio Adsuwak hemos situado todas las vistas correspondientes a la gestión de los Suwaks. Por ejemplo, en su interior la plantilla “new.html.twig” extiende del “layout” y le añade el formulario para crear los nuevos Suwaks.

```
{% extends 'AdsuwakBackendBundle::bucketadmin-layout.html.twig' %}
{% form_theme form 'AdsuwakBackendBundle:Form:fields.html.twig' %}

{% block stylesheets ...%}
{% block head ...%}

{% block content ...%}

{% block javascripts ...%}
```

Figura 6.25 Organización en bloques del “layout”

Como se puede observar en la figura 6.25 hay varios elementos “{% block... %}”. Estos elementos se utilizan para añadir contenido a los bloques definidos en el “layout”.

6.2.14.3. Aplicando estilos a los formularios.

Symfony utiliza plantillas por defecto para renderizar los elementos del formulario. No obstante permite al usuario personalizar cómo se muestra cada parte del formulario.

Esto tiene sentido por ejemplo en nuestro caso al utilizar una plantilla, esta incorpora sus propios elementos del formulario. Para poder utilizarlos debemos crear una nueva plantilla que defina los elementos de los formularios.

En Twig cada fragmento del formulario está representado por un bloque. Para editar un bloque en concreto deberemos modificarlo en la nueva plantilla.

Veamos un ejemplo. Para que cuando Symfony represente un elemento “input” de tipo “email” le ponga por defecto el atributo `class="form-control"` crearemos una nueva plantilla con el nombre “profile_fields.html.twig”.

```
{% block email_row %}
  {% spaceless %}
    <div class="form-group">
      <label for="{{ id }}">{{ label |trans({}, 'FOSUserBundle') }}</label>
      <input type="email" class="form-control" {{ block('email_widget') }}>
    </div>
  {% endspaceless %}
{% endblock email_row %}
```

Figura 6.26 Definiendo atributos por defecto a elemento de formulario

En ella definiremos todos los atributos que queramos que se utilicen por defecto. Seguidamente, en la plantilla que queramos aplicarle el nuevo estilo definido deberemos añadirle la línea:

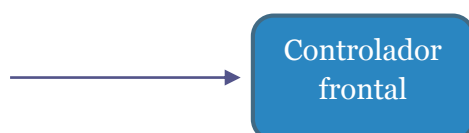
```
{% form_theme form 'FOSUserBundle:Form:profile_fields.html.twig' %}
```

Figura 6.27 Añadiendo el estilo a la vista

6.2.15. Sistema de rutas y el controlador

6.2.15.1. Controlador frontal y entornos de ejecución

En el patrón MVC (Modelo-Vista-Controlador) el encargado atender las petición del web del usuario es el Controlador frontal.



En Symfony existen dos entornos diferentes de ejecución. El primero de ellos, es el llamado “entorno de producción” el cual es el que utilizaran los usuarios cuando la

aplicación este subida en el servidor. En este entorno se desactivan algunas opciones de configuración para trabajar al máximo rendimiento.

Por otro lado, el entorno de desarrollo es el que será utilizado cuando se está desarrollando la aplicación. Este entorno ofrece información extra al desarrollador permitiéndole así depurar los errores con los que se va encontrando.

En Symfony existen dos tipos de controladores frontales los cuales se utilizan respectivamente en los entornos de producción y desarrollo.

- **App.php:** Es el controlador utilizado en el modo de producción. Cuando se produce un error interno en el servidor devuelve el código de error.
- **App_dev.php:** Es el controlador utilizado para el entorno de desarrollo o depuración de la aplicación. Su principal diferencia respecto al controlador de producción es que cuando se produce un error devuelve información relacionada la cual ayuda al desarrollador a solucionarla.

Para utilizar un entorno u otro bastará con añadir al final de la URL:

- Entorno de producción:

<http://localhost/adsuwak/app.php/>

- Entorno de desarrollo:

http://localhost/adsuwak/app_dev.php/

6.2.15.2. Sistema de “routing”

Independientemente del controlador frontal que estemos utilizando, cuando recibe una petición del usuario debe encaminar esta petición por la aplicación al controlador interno correcto. Para ello Symfony hace uso de un sistema de rutas, las cuales se utilizan para poder encaminar las peticiones a través de la aplicación.



Una ruta es una asociación de una URL hacia un controlador. Estas se definen en el archivo “routing.yml” de la aplicación. Veamos un ejemplo:

```
backend_ad_new:  
  path: /ad/new  
  defaults: { _controller: AdsuwakBackendBundle:Adsuwak:new }
```

Figura 6.28 Ejemplo de ruta

Backend_ad_new: es el nombre de la ruta. Este nombre se utiliza como identificador y deberá ser único en toda la aplicación.

Path: es el patrón de la ruta. Es lo que permite al sistema de “routing” discriminar entre una ruta u otra.

Defaults a su vez indica cual es controlador asociado a esa ruta y que método corresponde.

Para definir las rutas de la aplicación correctamente se deben definir en el archivo “routing.yml” correspondiente a cada “bundle”. Con esto a la hora de borrar el “bundle” o querer usarlo en un nuevo proyecto únicamente debemos importar su archivo de rutas en el routing.yml de la aplicación.

En la figura 6.29 se muestra parte del contenido del sistema de rutas del “BackendBundle”

```
backend_ad_preview:
  path: /ad/preview/{id}
  defaults: { _controller: AdsuwakBackendBundle:Adsuwak:preview }

backend_ad_edit_companies:
  path: /ad/companies/edit
  defaults: { _controller: AdsuwakBackendBundle:Default:editCompanies }

backend_test:
  path: /ad/test
  defaults: { _controller: AdsuwakBackendBundle:Default:test }
```

Figura 6.29 Parte del archivo routing.yml

Finalmente en la figura 6.30 se muestra como se importa dicho sistema de rutas en el “routing” general.

```
adsuwak_backend:
  resource: "@AdsuwakBackendBundle/Resources/config/routing.yml"
  prefix: /{_locale}/app
  requirements:
    _locale: es|en
  defaults: { _locale: es}
```

Figura 6.30 Importando rutas

6.2.16. Sistema de formularios

En la aplicación se hace uso de diferentes formularios para la gestión de los usuarios y para la modificación de algunos datos. Para su gestión, Symfony2 incorpora un componente llamado “Form” el cual está dedicado a la gestión de los formularios HTML.

Una de las principales ventajas que aporta este servicio, es que al generar los formularios independientemente del código, permite su reutilización en distintas partes del código.

Este servicio permite generar los formularios de dos maneras distintas:

- Desde el controlador
- Creando una clase “Type” y creando el formulario a través de dicha clase.

En la aplicación se utilizará la segunda opción puesto que al generar el formulario en una clase permite la reutilización de dicho formulario en distintas partes del código de la aplicación.

6.2.16.1. Creando un nuevo formulario

En primer lugar, para crear un formulario deberemos definir una nueva clase “Type” en el directorio “Form/Type” del “bundle”. En la nueva clase “Type” se definirán los campos que formaran el formulario y a partir de los cuales el servicio lo generará.

A modo de ejemplo vamos a crear la clase “CompaniesFormType.php” la cual permitirá al usuario modificar los datos de la compañía. En primer lugar, la nueva clase debe extender de la clase “AbstractType” y debe contener el método “buildForm()” donde se especifican los campos que debe incluir el formulario.

```
class CompaniesFormType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options){...}

    public function getName(){...}
}
```

Figura 6.31 Estructura clase Type

A su vez también debe contener el método “getName()” el cual debe devolver un nombre que identifique unívocamente el “Type” que estamos creando.

En el método “buildForm()” se definen los elementos que contiene el formulario.

```
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('name', 'text', array(
            'attr' => array(
                'placeholder' => $this->translator->trans("companies.form.name.placeholder", array(),
                    'maxlength' => '255'
                ),
            ),
            'required' => true,
            'label' => $this->translator->trans("companies.form.name.label", array(), "companies"),
        ))
        ->add('nif', 'text', array(
            'attr' => array(
                'placeholder' => $this->translator->trans("companies.form.nif.placeholder", array(),
                    'maxlength' => '9'
                ),
            ),
            'required' => true,
            'label' => $this->translator->trans("companies.form.nif.label", array(), "companies"),
        ))
}
```

Figura 6.32 Añadiendo campos al formulario

En la variable “\$builder”, en la que se le ha pasado como parámetro el servicio que genera los formularios, se añaden los campos necesarios junto con sus propiedades.

En la figura 6.32, se muestra un ejemplo en el que se le añade al formulario un elemento “input” de tipo “text” (<input type=”text”...) en el que se le añaden los atributos “placeholder” que muestra el texto por defecto y “label” que le asocia una etiqueta al elemento.



6.2.16.2. Cargando y validando el formulario

Con la nueva clase definida, podemos ya utilizarla para crear un formulario en cualquier controlador de la aplicación. Para realizar este paso, siguiendo con el ejemplo anterior, la figura 6.33 se demuestra cómo utilizar la clase y generar un nuevo formulario.

```
$form = $this->createForm(new CompaniesFormType($this->container->get('translator')), $company);
```

Figura 6.33 Creando formulario en controlador

6.2.16.3. Representando el formulario

Con el formulario creado y añadido en el controlador, debemos indicarle a la plantilla que represente el formulario.

La forma más sencilla de hacerlo es utilizando la instrucción Twig.

```
<form action="#" method="post" {{ form_ctype(form) }}>
    {{ form_widget(form) }}

    <input type="submit" />
</form>
```

Figura 6.34 Ejemplo mostrando formulario

Con la función “form_widget (form)” se representan todos los elementos del formulario automáticamente. El problema de utilizar esta instrucción, es que el código se genera automáticamente y puede que a la hora de generar el archivo HTML no se aplique el estilo que deseemos. Existe otra para mostrar un formulario:

```
<div class="form-group">
  <label class="col-lg-2 control-label">{{ form_label(form.advertisements.type) }}</label>
  <div class="col-lg-8">
    {{ form_widget(form.advertisements.type) }}
    {{ form_errors(form.advertisements.type) }}
  </div>
</div>
<div class="form-group">
  <label class="col-lg-2 control-label">{{ form_label(form.advertisements.urlResource) }}</l
  <div class="col-lg-8">
    {{ form_widget(form.advertisements.urlResource) }}
    {{ form_errors(form.advertisements.urlResource) }}
  </div>
```

Figura 6.35 Ejemplo formulario por partes

En la imagen 6.35 se muestra como se representa un formulario parte por parte. A partir del objeto “form” que se le pasa a la vista como parámetro, podemos representar el formulario por partes.

En la aplicación se ha utilizado esta última opción para tener mayor flexibilidad a la hora de representar los formularios.

6.3. Desarrollo de la plataforma

6.3.1. Habilitando el traductor

“*Translator*” es un servicio incorporado en el framework Symfony2. Mediante esta utilidad se pueden traducir cadenas de texto y ofrecer los textos de la web en un idioma u otro dependiendo de la configuración del idioma del navegador del usuario visitante.

Según la configuración del idioma del navegador, cuando este realiza una petición HTTP a una página web, añade en la petición un parámetro denominado “*locale*” el cual establece que preferencia de idioma tiene el usuario.

Symfony al recibir una petición debe determinar el “*locale*” del usuario definido en la petición y devolver el texto en el idioma determinado.

Para ello se debe habilitar su utilización modificando el archivo de configuración “/app/config/config.yml”

```
framework:
  translator: { fallbacks: ["%locale%"] }
```

Figura 6.36 Habilitando translator en config.yml

6.3.1.1. Definiendo el “locale”

Es necesario definir el “*locale*” en la sesión del usuario. No obstante, para poder ofrecer una página en dos idiomas deberemos definir la configuración regional en la URL.

Para ello deberemos modificar el sistema de enrutamiento y utilizar el parámetro “_locale: es|en” para establecer el “*locale*” del usuario. Veamos un ejemplo:

```
adsuwak_frontend:
  resource: "@AdsuwakFrontendBundle/Resources/config/routing.yml"
  prefix: /{_locale}/
  requirements:
    _locale: es|en
  defaults: { _locale: es}

adsuwak_backend:
  resource: "@AdsuwakBackendBundle/Resources/config/routing.yml"
  prefix: /{_locale}/app
  requirements:
    _locale: es|en
  defaults: { _locale: es}
```

Figura 6.37 Añadiendo atributo "locale" a las rutas

Con los parámetros “*requeriments*” y “*defaults*” añadidos en la figura 6.32, cuando el usuario por ejemplo acceda a la ruta “/es/app”, Symfony establecerá el “*locale*” del usuario automáticamente a “es”.

Como se observa en la imagen, deberemos establecer los dos parámetros anteriormente comentados en las rutas principales que importan las rutas del “bundle”.

6.3.1.2. Diccionarios de traducción

Este servicio permite el uso de diccionarios de traducción creados por el usuario. Con ellos definiremos las traducciones de la plataforma y posteriormente cuando el sistema tenga que realizar alguna traducción consultará el diccionario personalizado y obtendrá la traducción.

El diccionario de traducciones puede ser creado en diferentes formatos (php, yml, xml). En el proyecto se utilizará el formato YAML debido a su fácil edición.

Los archivos de traducción se situarán en la carpeta “Resources/Translations” de cada “bundle”.

Los diccionarios creados deben tener un nombre estructurado:

“domain.locale.loader”

- **Domain:** Es el nombre del diccionario de traducción.
- **Locale:** El “locale” al que pertenece la traducción (inglés en, español es)
- **Loader:** Formato del diccionario. Si se utiliza YAML será yml.

Veamos un ejemplo de un archivo de traducción:

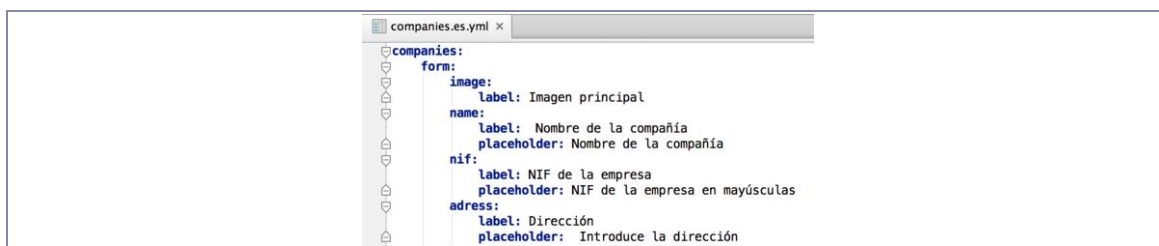


Figura 6.38 Ejemplo archivo de traducción

Con los diccionarios creados ahora deberemos utilizarlos en las vistas. Para realizar una traducción directamente desde Twig deberemos hacerlo tal y como se muestra en la figura 6.34:

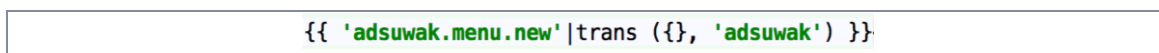


Figura 6.39 Utilizando traducciones en las plantillas

- “adsuwak.menu.new” es la variable de traducción definida en el diccionario.
- “adsuwak” es el nombre del diccionario de traducción

Deberemos utilizar la expresión de Twig mostrada arriba siempre que queramos mostrar un texto en varios idiomas.



6.3.2. Formulario para nuevo Suwak

El formulario de creación de nuevos Suwaks, será el que permita al usuario personalizar su anuncio inteligente.

Debido a que el formulario contiene una gran cantidad de diferentes opciones se ha decidido dividirlo en cuatro pasos para que le resulte más cómodo al usuario y no cargarle con excesiva información.

A su misma vez se ha añadido una previsualización del anuncio el cual se modifica conforme lo va modificando el usuario.

Figura 6.40 Formulario creación nuevo Suwak

Para dividir el formulario en distintos pasos se ha utilizado el “plugin” de JQuery “Jquery Steps”. Este “plugin” es un asistente de formularios, que haciendo uso de unas pocas líneas de código JavaScript permite dividir un formulario en varios pasos.

Existen otros similares pero debido a su facilidad de uso se ha decido usarlo en el proyecto.

Para poder aplicar el plugin en el formulario, en primer lugar deberemos descargarlo de la página oficial <http://www.jquery-steps.com/> y posteriormente situarlo en la carpeta que contiene todos los archivos JavaScript del “template”.

```

{{ form_start(form) }}
<h2>{{ 'adsuwak.form.first_step'|trans({}, 'adsuwak') }}</h2>
<section...>

<h2>{{ 'adsuwak.form.second_step'|trans({}, 'adsuwak') }}</h2>
<section...>

<h2>{{ 'adsuwak.form.third_step'|trans({}, 'adsuwak') }}</h2>
<section...>

<h2>{{ 'adsuwak.form.fourth_step'|trans({}, 'adsuwak') }}</h2>
<section...>
{{ form_end(form) }}
    
```

Figura 6.41 Código del formulario nuevo Suwak

En la 6.36 se muestra parte del formulario utilizado. El “plugin” para dividirlo se utiliza unas etiquetas de referencia a partir de las cuales genera las diferentes páginas.


```

<script>
var cont = 0;
$(function () {
  $("form").steps({
    headerTag: "h2",
    bodyTag: "section",
    transitionEffect: "slideLeft",
    onStepChanging: function (event, currentIndex, newIndex)
    {
      $("form").validate().settings.ignore = ":disabled,:hidden";
      return $("form").valid();
    },
    onStepChanged: function (event, currentIndex, newIndex)
    {
      if(currentIndex==1){
        $('#Overlay').hide('fast');
        $('#Suwak').show('fast');
      }else if(currentIndex==2){
        $('#Suwak').hide();
        $('#Overlay').show('fast');
      }else{
        $('#Suwak').hide();
        $('#Overlay').hide();
      }
    },
    onFinished: function (event, currentIndex) {
      $("form").submit();
    }
  });
});
</script>

```

Figura 6.42 Código plugin JQuery Steps para división del formulario

Estas referencias son las etiquetas “<section>”. En la 6.37 se muestra la configuración del plugin. El parámetro:

- **“bodyTag”** es el utilizado para “trocear” el formulario.
- **“headerTag”** determina el título del paso.
- **“transitionEffect”**: Asigna el efecto a realizar cuando se cambia de paso.
- **“onStepChanging”**: Es el evento que se produce cuando se cambia de paso. Aquí realizamos una validación de los campos, es decir, cuando el usuario cambie de paso el plugin validará los campos y si son correctos le permitirá cambiar de paso.
- **“onStepChanged”**: Es el evento que se produce cuando se ha cambiado satisfactoriamente de paso. Se utiliza para mostrar y ocultar las previsualizaciones de los diferentes pasos.
- **“onFinished”**: Representa al evento que se produce cuando se han cumplido todos los pasos. En este caso, cuando el usuario haya completado todos los pasos deberemos hacer el “submit” del formulario.

6.3.3. Sistema de generación de scripts

La plataforma debe proporcionar al anunciante un código script para que este lo inserte en una página web.

En primer lugar un Suwak tiene definido en la entidad “Adsuwak” un campo denominado “token” el cual es un identificador único e irrepetible. Este campo ha sido generado aleatoriamente utilizando md5

```

$adsuwak->setToken(substr(md5(uniqid(rand(), true)), 0, 32));

```

Figura 6.43 Generando token

En el sistema de rutas, se ha definido la siguiente ruta:



```
adsuwak_api_homepage:
  path:      services/{token}
  defaults: { _controller: AdsuwakAPIBundle:Default:index }
```

Figura 6.44 Ruta asociada

Para la realización de esta tarea, el controlador en lugar de representar una vista debe devolver un código JavaScript. Para ello, en el controlador “DefaultController.php” del “bundle” “APIBundle” debemos configurar el tipo de respuesta que devolvemos al usuario.

```
$rendered = $this->renderView('AdsuwakAPIBundle:Default:index.js.twig', array('adsuwak' => $adsuwak));
$response = new \Symfony\Component\HttpFoundation\Response( $rendered );
//Envío de respuesta
$response->headers->set( 'Content-Type', 'text/javascript' );
return $response;
```

Figura 6.45 Configurando el tipo de respuesta en el controlador

En la variable “\$rendered” almacenamos el código script del anuncio generado y en la variable creamos un nuevo objeto “Response” a partir del código script.

Seguidamente configuramos las cabeceras de la cabecera de la respuesta HTTP y establecemos el contenido como “text/javascript”.

6.3.4. Gestión de los usuarios

Introducción

El componente de seguridad que forma parte de Symfony2 permite gestionar la autenticación de los usuarios en la plataforma. No obstante, en Symfony2 no hay ningún sistema encargado de gestionar los usuarios. Por ello deberíamos crear y mantener las sesiones manualmente.

Actualmente existen varias soluciones a este problema. Una de ellas es el “bundle” “FOSUserBundle” creado por el grupo Friends of Symfony, el cual proporciona un sistema de gestión de usuarios.

Se opta por su utilización debido a que automatiza todos los pasos a realizar en la gestión de los usuarios y no resulta tan tedioso su manipulación.

6.3.4.1. Instalación

En primer lugar se instalará el “bundle” externo. Desde Symfony 2.1 las dependencias empezaron a ser manejadas con Composer. Por ello para instalar este “bundle” de terceros debemos modificar el archivo “composer.json” y añadir:

```
"require": {  
    ...  
    "friendsofsymfony/user-bundle": "2.0.*@dev",  
    ...  
},
```

Figura 6.46 Añadiendo dependencia a composer

Ahora debemos actualizar todas las dependencias mediante el siguiente comando:

```
$ php composer.phar update
```

En este punto se ha instalado ya el “bundle”. Ahora se debe habilitar en el proyecto registrando el “bundle” en el archivo AppKernel.php

```
public function registerBundles()  
{  
    $bundles = array(  
        new FOS\UserBundle\FOSUserBundle(),  
    );  
}
```

Figura 6.47 Habilitando el nuevo bundle

Tras realizar estos pasos deberemos configurarlo y adaptarlo a la aplicación.

6.3.4.2. Configuración

“FOSUserBundle” utiliza unos formularios y unas vistas por defecto. En caso de querer modificarlas para adaptarlas a nuestro proyecto debemos sobrescribir el elemento que queramos personalizar.

Para gestionar toda la configuración de una manera correcta y ordenada crearemos un nuevo “bundle” en el proyecto llamado “UserBundle” en el cual insertaremos todas las modificaciones necesarias para la adaptación del plugin a nuestra aplicación.

Para crear el nuevo “bundle” ejecutamos el comando:

```
$ php app/console generate:bundle
```

```

adsuwak — php — 80x24
app      composer.json  composer.phar  vendor
MacBook-Pro-de-Sergio-6:adsuwak serbemas$ php app/console generate:bundle

Welcome to the Symfony2 bundle generator

Your application code must be written in bundles. This command helps
you generate them easily.

Each bundle is hosted under a namespace (like Acme/Bundle/BlogBundle).
The namespace should begin with a "vendor" name like your company name, your
project name, or your client name, followed by one or more optional category
sub-namespaces, and it should end with the bundle name itself
(which must have Bundle as a suffix).

See http://symfony.com/doc/current/cookbook/bundles/best_practices.html#index-1
for more
details on bundle naming conventions.

Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace: AdsuwakUserBundle
    
```

Tras completar todos los pasos mencionados en el 6.2.12 se habrá creado el nuevo “bundle”. El siguiente paso será crear una entidad User que sobrescriba a la que usa por defecto el plugin.

6.3.4.3. Creando entidad User

Una vez creado el nuevo “bundle”, el siguiente paso a realizar es crear la entidad “User” la cual contendrá todos los datos de los usuarios de la aplicación. Para ello, creamos el directorio “Entity” en el nuevo “bundle” y en su interior creamos la nueva la entidad User.php

```

class User extends BaseUser
{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;
}
    
```

Figura 6.48 Fragmento de la entidad User

A su vez deseamos añadir nuevos campos a la clase para almacenar la información que deseamos necesario. Para ello se añaden en clase los campos los campos “name”, “surname” y “phone” y se generan sus propiedades como hacemos en el apartado 6.2.12 con el comando:

```

$ php app/console doctrine:generate:entities AdsuwakUserBundle
    
```

6.3.4.4. Modificar config.yml

A su misma vez, debemos modificar el archivo config.yml y añadir unas directivas de configuración.



```
fos_user:
  db_driver: orm # other valid values are 'mongodb', 'couchdb' and 'propel'
  firewall_name: main
  user_class: Adsuwak\UserBundle\Entity\User
```

Figura 6.49 Nueva configuración *config.yml*

Con esta configuración, le indicamos a Symfony que utilice el nuevo “*bundle*” para gestionar los usuarios y utilice la entidad *User* como la clase-usuario.

6.3.4.5. Sobrescribiendo el “*bundle*”

Para poder sobrescribir la configuración por defecto del plugin “*FOSUserBundle*” debemos modificar el archivo “*AdsuwakUserBundle.php*” y añadir:

```
class AdsuwakUserBundle extends Bundle
{
    public function getParent()
    {
        return 'FOSUserBundle';
    }
}
```

Figura 6.50 Contenido archivo *AdsuwakUserBundle.php*

Con la función “*getParent()*” se indica que es un “*bundle*” hijo de “*FOSUserBundle*” y así podremos modificar cómodamente la configuración.

6.3.4.6. Sobrescribiendo formularios

En el apartado 6.1.6.2 hemos creado una nueva entidad *User* y le hemos añadido tres campos nuevos que no están definidos en la clase *BaseUser*.

El formulario que utiliza por defecto *FOSUserBundle* no incluye los campos nuevos que se ha creado por lo que es necesario modificarlo y añadir las nuevas propiedades.

En primer lugar, en *AdsuwakUserBundle* crearemos un nuevo directorio llamado “*Forms*” y en su interior otro llamado “*Type*”. Ahora creamos un nuevo archivo PHP con el nombre “*RegistrationFormType.php*”

```
<?php
namespace Adsuwak\UserBundle\Form\Type;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use FOS\UserBundle\Form\Type\RegistrationFormType as BaseType;

class RegistrationFormType extends BaseType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
```

Figura 6.51 Contenido de *RegistrationFormType.php*. Parte 1

Y añadimos al formulario por defecto los nuevos campos:



```

$builder
  ->add('email', 'email', array(
    'label' => 'form.email',
    'attr'=>array(
      'placeholder'=> 'form.email'),
    'translation_domain' => 'FOSUserBundle',
    'error_bubbling' => true
  ))
  ->add('plainPassword', 'repeated', array(
    'type' => 'password',
    'options' => array('translation_domain' => 'FOSUserBundle'),
    'first_options' => array(
      'label' => 'form.password',
      'attr'=>array(
        'placeholder'=> 'form.password'),
      ),
    'second_options' => array(
      'label' => 'form.password_confirmation',
      'attr'=>array(
        'placeholder'=> 'form.password_confirmation'),
      ),
    'invalid_message' => 'fos_user.password.mismatch',
    'error_bubbling' => true
  ))
  ;

```

Figura 6.52 Contenido de RegistrationFormType.php. Parte 2

Por otra parte, en la aplicación se desea almacenar información relacionada con el perfil del usuario. El proceso a seguir es el mismo que el que se acaba de realizar con el registro, es decir, añadir los campos deseados a la entidad “User” y seguidamente crear un formulario que sobrescriba al existente en el “bundle” “FOSUserBundle”, se decide no incluirlo en la presente memoria con tal de no resultar repetitivos.

6.3.4.7. Sobrescribiendo las vistas

Una vez hayamos redefinido los formularios deberemos modificar también las vistas que lo representan.

Como sucede con los formularios, “FOSUserBundle” también incluye vistas por defecto. Para sobrescribir una vista en concreto, en el directorio “Resources/views/” debemos crear una carpeta con la misma estructura que en “FOSUserBundle” para poder sobrescribirla.

Siguiendo el caso descrito en el apartado anterior, vamos crear una nueva vista para modificar el “template” por defecto del formulario de registro.

Para ello en primer lugar, deberemos crear una nueva carpeta con el nombre “Registration”.

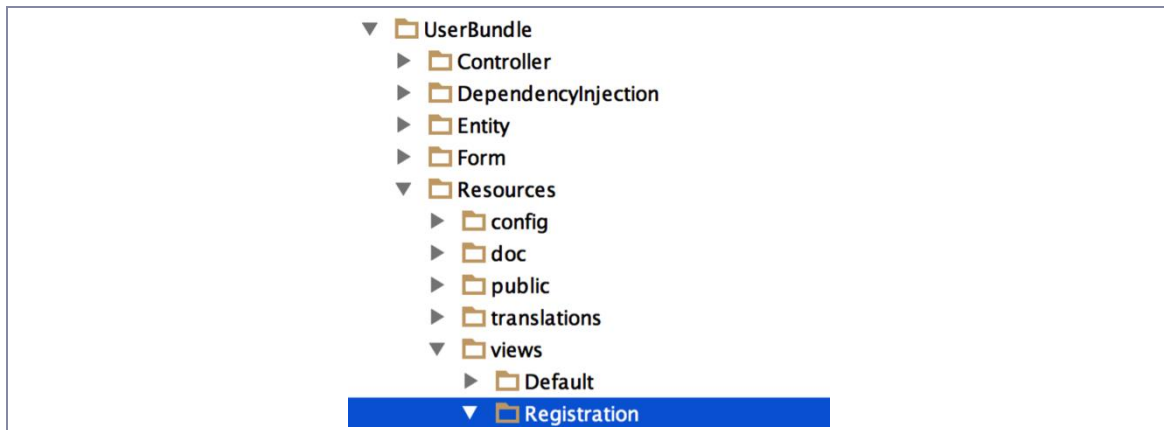


Figura 6.53 Nuevo directorio Registration

Ahora, siguiendo la estructura de carpetas presente en “FOSUserBundle” debemos crear los mismos archivos. En la figura 6.61 se muestra a la izquierda la estructura del “bundle” “UserBundle” y en la parte derecha la estructura presente en el “plugin”

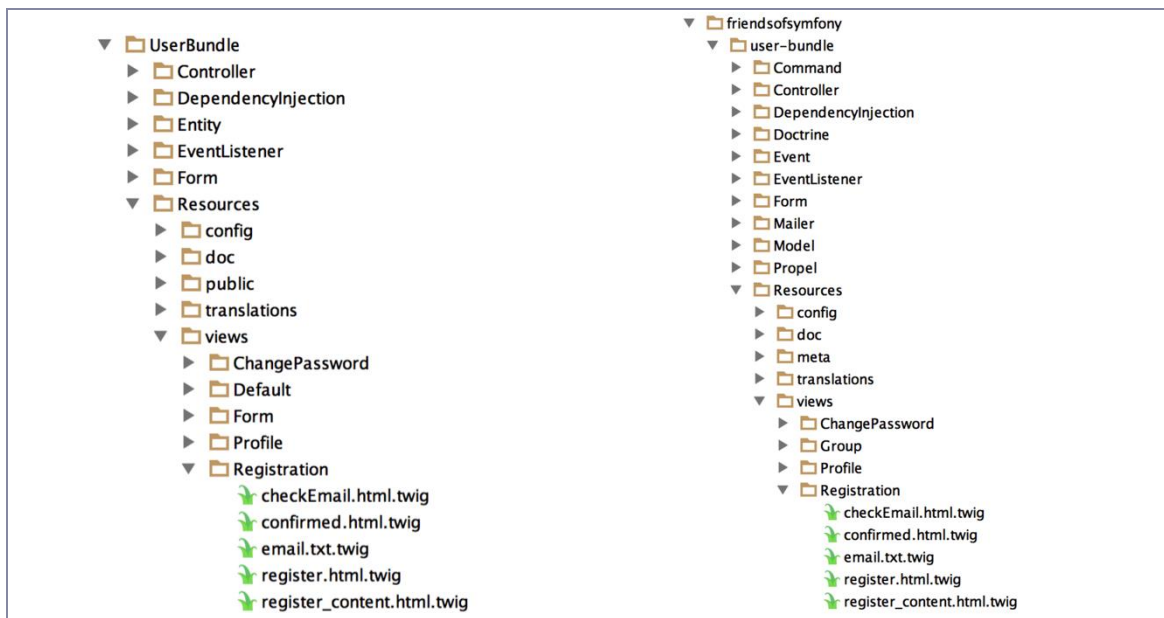


Figura 6.54 Estructura UserBundle y FOSUserBundle

Los archivos en cuestión que debemos sobrescribir son “register.html.twig” y “register_content.html.twig”.

En el archivo “register.html.twig” indicamos que extienda del archivo base del “template” del frontend el cual hemos definido en el 6.2.14.2 para cargar la estructura básica de la página (el menú, los estilos, etc...).

```

{% extends "AdsuwakFrontendBundle::layout.html.twig" %}

{% block content %}
    {% block fos_user_content %}
        {% include "FOSUserBundle:Registration:register_content.html.twig" %}
    {% endblock fos_user_content %}
{% endblock %}

```

Figura 6.55 Contenido archivo Register.html.twig



Como se puede observar en la figura 6.57, en el archivo “register.html.twig” se carga el contenido de otra vista llamada “register_content.html.twig”. En este segundo archivo es donde se representa el formulario de registro.

A modo de ejemplo se añade una captura con el contenido del archivo.

```

{% trans_default_domain 'FOSUserBundle' %}
<div id="register_form">
  <h2>Registros</h2>
  <form action="{{ path('fos_user_registration_register') }}" {{ form_ctype(form) }} method="POST" class="fos_user_r
  <div>
    <input type="submit" value="{{ 'registration.submit'|trans }}" />#}
  <section>
    {{ form_widget(form) }}
    {{ form_rest(form) }}
  </section>
  <input type="submit" class = "button-color-1" value="{{ 'registration.submit'|trans }}" />
</div>
</form>
<a href="" title="Lost password">{{ 'landing.menu.lost_password'|trans ({} , 'landing') }}</a>
</div>

```

Figura 6.56 Contenido archivo Register_content.html.twig

Como ocurre en punto anterior, este proceso se debe realizar para todos aquellos “templates” que se deseen sobrescribir.

6.3.4.8. Creando un nuevo servicio

Una vez se haya creado el nuevo formulario, para que el “bundle” “FOSUserBundle” lo reconozca debemos declarar un nuevo servicio. Para ello debemos ir al archivo “services.yml” del módulo “UserBundle” y añadir lo siguiente:

```

services:
  mi_propio.registration.form.type:
    class: Adsuwak\UserBundle\Form\Type\RegistrationFormType
    arguments: [%fos_user.model.user.class%]
    tags:
      - { name: form.type, alias: mi_user_registration}

```

Figura 6.57 Nuevo servicio formulario de registro

Con el servicio creado, debemos configurar el plugin para que utilice el nuevo formulario. Para ello debemos editar el archivo “app/config.yml” y añadir lo siguiente:

```

fos_user:
  db_driver: orm # other valid values are 'mongodb', 'couchdb' and 'propel'
  firewall_name: main
  user_class: Adsuwak\UserBundle\Entity\User
  registration:
    form:
      type: mi_user_registration

```

Figura 6.58 Nueva configuración config.yml

En este punto ya estará listo el formulario de registro de nuestra aplicación.

Registro

Email:

Contraseña:

Repita la contraseña:

Registrar

[¿No recuerdas tu contraseña?](#)

Figura 6.59 Nuevo formulario de registro de la aplicación

6.3.4.9. Redirecciones

La extensión FOSUserBundle proporciona un sistema de gestión de usuarios y no un sistema de autenticación. Por lo tanto la redirección no depende del plugin si no del componente “Security” del framework Symfony2.

Para configurar la redirección bastará con definir la página por defecto. Para ello deberemos modificar el archivo “*app/config/security.yml*”

```

firewalls:
  main:
    pattern: ^/
    form_login:
      provider: fos_userbundle
      login_path: fos_user_security_login
      use_forward: false
      check_path: fos_user_security_check
      failure_path: null
      csrf_provider: form.csrf_provider
      default_target_path: backend_homepage
    logout:
      path: fos_user_security_logout
      target: adsuwak_frontend_homepage

```

Figura 6.60 Estableciendo página por defecto

En la figura 6.62 se define tanto la redirección cuando el usuario inicia sesión en la plataforma como cuando cierra sesión. Estas redirecciones se definen en los parámetros “form->default_target_path” y “logout->target”.

6.3.5. Sistema de notificaciones

Para dar un “*feedback*” al usuario, y que sepa que todo ha ido correctamente debemos generar una notificación diciéndole como ha ido el proceso.

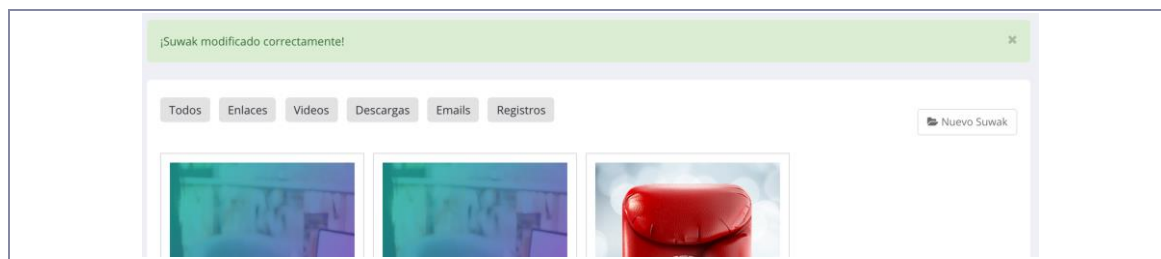


Figura 6.61 Ejemplo notificaciones del sistema

En Symfony las notificaciones de usuario se almacenan en la sesión del usuario. El controlador crea un mensaje flash y lo añade a la sesión del usuario. Seguidamente redirige el usuario a una nueva página.

Veamos cómo se realiza el proceso en primer lugar desde el controlador:

```
$this->get('session')->getFlashBag()->set('notice', '¡Suwak creado correctamente!');
return $this->redirect($this->generateURL('backend_homepage'));
```

Figura 6.62 Añadiendo notificaciones en el controlador

En el controlador se añade el mensaje flash con identificador “*notice*” y con el mensaje “¡Suwak creado correctamente!”. Seguidamente redirecciona al usuario a la nueva página.

Esta nueva página deberá contener un fragmento de código que muestra este mensaje.

```
{% if app.session.started %}
{% for flashMessage in app.session.flashbag.get('notice') %}
<div class="ui-bar ui-bar-sr" id="alert_info">
<p>{{ flashMessage }}</p>
</div>
{% endfor %}
```

Figura 6.63 Cargando las notificaciones en la vista

Mediante este fragmento de código se muestran las notificaciones definidas en el controlador. Estas notificaciones se muestran una vez únicamente por lo que al recargar la página ya no se mostrarán.

6.3.5.1. En FOSUserBundle

Para añadir las notificaciones de los eventos que se producen en la gestión de los usuarios debemos modificar los controladores y añadir las notificaciones. Una solución, sería redefinir todo el controlador ya implementado en el “*bundle*”, pero ello requeriría repetir toda la parte lógica y la funcionalidad.

En “*FOSUserBundle*” existen una serie de eventos que se ejecutan cuando el usuario va realizando las acciones. En la mayoría de las ocasiones resulta más efectivo y conveniente usar los eventos para implementar la nueva funcionalidad.

Los controladores que proporciona el “bundle” por defecto, a lo largo de su ejecución ejecutan una serie de eventos.

En general, todos lanzan un evento “SUCCESS” cuando la validación del formulario que ha utilizado el usuario es satisfactoria y “COMPLETED” cuando el evento ha sido completado.

Los eventos “SUCCESS” permiten configurar la respuesta y los “COMPLETED” permiten acceder a la respuesta antes de ser enviada al usuario.

En primer lugar, creamos dentro de UserBundle el directorio “EventListeners”. Seguidamente, en su interior creamos la clase “PHP” que se encargara de gestionar el evento.

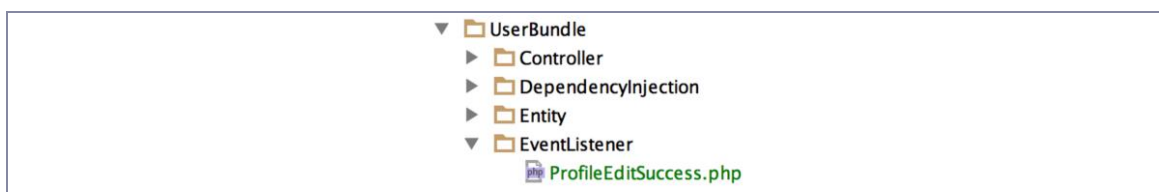


Figura 6.64 Creando listener de fosUserBundle

En este ejemplo que vamos a realizar vamos a crear un “Listener” para el evento que se ejecuta cuando el usuario cambia sus datos del perfil satisfactoriamente.

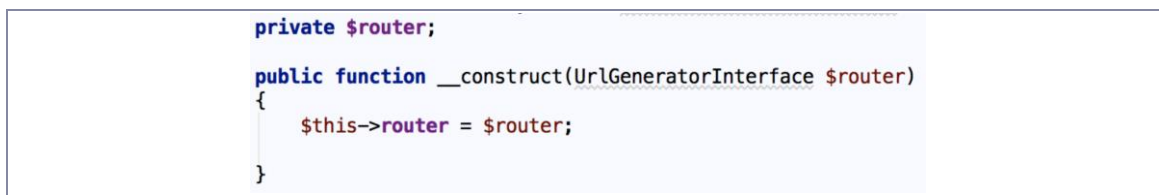


Figura 6.65 Contenido listener. Parte 1

En primer lugar, en la variable “router” se crea el servicio el cual se encargará de crear la ruta a la que se redirigido el usuario.

Cada evento tiene asignado por defecto un método el cual contiene la lógica. El siguiente método captura el evento y le añade la siguiente funcionalidad.



Figura 6.66 Contenido listener. Parte 2

Como se puede apreciar en la imagen, recuperamos del evento la sesión y le añadimos un “FlashBag”. Seguidamente le decimos que nos redirija a la ruta que previamente hemos definido y le hemos añadido el “FlashBag”.



El último paso a realizar es definir un nuevo servicio en el “*bundle*”.

```

adsuwak_user.profile_edit:
  class: Adsuwak\UserBundle\EventListener\ProfileEditSuccess
  arguments: [@router ]
  tags:
    - { name: kernel.event_subscriber }
    
```

Figura 6.67 Creando servicio asociado al listener

6.3.6. Sistema de trazabilidad

Un sistema de trazabilidad, según Wikipedia, es un conjunto de disciplinas que permiten obtener el seguimiento de los productos a lo largo de cualquier cadena. En nuestro caso, un sistema de trazabilidad sería aquel que registrase todas las acciones que el usuario realice en la plataforma.

Un sistema de estas características resulta muy beneficioso en aquellas aplicaciones que estén dirigidas al uso de diferentes usuarios para poder tener un control de todas las acciones que se hayan realizado. Esto bien puede utilizarse desde para depurar errores en la aplicación hasta detectar actividad sospechosa.

Estos son los diferentes puntos que registra la aplicación:

- Creación de Suwak
- Eliminación de Suwak
- Modificación de Suwak
- Modificación datos de la compañía
- Modificación datos del perfil
- Modificación contraseña
- Inicio de sesión

Para llevarlo a cabo hemos creado una entidad llamada “Steps”. En la figura 6.63 se muestran la estructura que presenta la tabla.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo
1	id	int(11)			No
2	action	varchar(255)	utf8_unicode_ci		No
3	ip	varchar(255)	utf8_unicode_ci		No
4	date	datetime			No
5	user_id	int(11)			Sí

Figura 6.68 Nueva entidad Steps

La nueva entidad está relacionada con la tabla “User”. Se almacenarán la acción que realiza el usuario, la dirección ip desde la que se realiza el paso y una fecha que indica el día y la hora exacta de su realización.



Para poder añadir un “step” cuando un usuario inicia sesión en la plataforma necesitamos crear un nuevo “Listener”.

En primer lugar creamos las relaciones en las entidades. El tipo de relación será One-To-Many Many-To-One

```
/**
 * @ORM\ManyToOne(targetEntity="Adsuwak\UserBundle\Entity\User", mappedBy="steps")
 */
protected $user;
```

Figura 6.69 Relación ManyToOne entidad Steps.php

```
/**
 * @ORM\OneToMany(targetEntity="Adsuwak\BackendBundle\Entity\Steps", mappedBy="user")
 */
protected $steps;
```

Figura 6.70 Relación OneToMany entidad User.php

Una vez creadas las relaciones creamos los métodos consultores y modificadores mediante los siguientes comandos.

-Clase Steps

```
php app/console doctrine:generate:entities AdsuwakBackendBundle
```

-Clase User

```
php app/console doctrine:generate:entities AdsuwakUserBundle
```

Ahora actualizamos la base de datos

```
php app/console doctrine:schema:update --force
```

En este punto, debemos situar en cada controlador en el que queramos registrar la acción del usuario el siguiente código:

```
$step = new Steps();
$step->setAction("Mostrar Suwaks");
$step->setIp($this->container->get('request')->getClientIp());
$step->setDate(new \DateTime());
$step->setUser($user);
$em->persist($step);
$em->flush();
$user->addStep($step);
$em->flush();
```

Figura 6.71 Añadiendo un nuevo step”

Como se puede observar en la figura 6.66 se crea un nuevo “Step” y le añadimos la acción que está realizando, el momento en la que se ejecuta y la dirección ip extraída de la petición.



Por otra parte, para poder controlar ciertos eventos en la aplicación, debemos utilizar los “Listeners” que hemos utilizado anteriormente. Estos implementan la interfaz “EventSubscriberInterface” por lo que no podremos utilizar directamente los siguientes métodos en el código.

```
$this->getDoctrine()->getManager();

$user = $this->get('security.context')->getToken()->getUser();
```

Para poder utilizarlos, tal y como se muestra en la figura 6.67, debemos usar el inyector de dependencias y en el servicio que implementamos el “Listener” hemos de pasarle como argumento los servicios “EntityManager” y “SecurityContext”.

```
adsuwak_user.profile_edit:
class: Adsuwak\UserBundle\EventListener\ProfileEditSuccess
arguments: [@router, @doctrine.orm.entity_manager, @security.context ]
tags:
- { name: kernel.event_subscriber }
```

Figura 6.72 Añadiendo nuevos argumentos al servicio profile_edit

Siguiendo con el proceso debemos modificar el archivo “ProfileEditSuccess”.

A su misma vez añadimos a la clase:

```
use Doctrine\ORM\EntityManager;
use Adsuwak\BackendBundle\Entity\Steps;
use Symfony\Component\Security\Core\SecurityContext;
```

Figura 6.73 Añadiendo nuevas clases

Posteriormente, en el “Listener” debemos crear las siguientes variables globales:

```
protected $em;
protected $securityContext;
```

Figura 6.74 Creando variables globales

En el constructor ahora debemos proporcionarle los parámetros pasados en los argumentos del servicio.

```
public function __construct(UrlGeneratorInterface $router, EntityManager $em, SecurityContext $securityContext)
{
    $this->router = $router;
    $this->em = $em;
    $this->securityContext = $securityContext;
}
```

Figura 6.75 Actualizando el constructor

Ya en punto podemos utilizar el código anterior.

```

//Get the request from the event triggered
$request = $event->getRequest();
$user = $this->securityContext->getToken()->getUser();
$url = $this->router->generate('fos_user_profile_show');

$step = new Steps();
$step->setAction("Perfil modificado");
$step->setIp($request->getClientIp());
$step->setDate(new \DateTime());
$step->setUser($user);
$this->em->persist($step);
$this->em->flush();
$user->addStep($step);
$this->em->flush();

```

Figura 6.76 Registrando las acciones realizadas.

6.4. Instalación servidor

Con el proyecto ya finalizado el siguiente paso es la puesta en marcha de un servidor web donde se alojará la aplicación. Para ello necesitamos de un “*hosting web*”. Existen actualmente una gran cantidad de hostings que ofrecen gran variedad de productos y servicios a diferentes precios. Para la puesta en marcha de la aplicación hemos decidido contratar un hosting web en <https://www.digitalocean.com/> debido a que ofrecen una buena relación precio/servicios prestados.

6.4.1. Creación de una cuenta

El primer paso que deberemos realizar es la creación de una cuenta para la administración de la maquina remota. Para ello nos dirigiremos a la página principal del hosting y nos registraremos en la plataforma.

Figura 6.77 Registro en DigitalOcean. Parte 1

Una vez hayamos introducido el correo y la contraseña deberemos confirmar la cuenta con un correo que nos envía la plataforma.

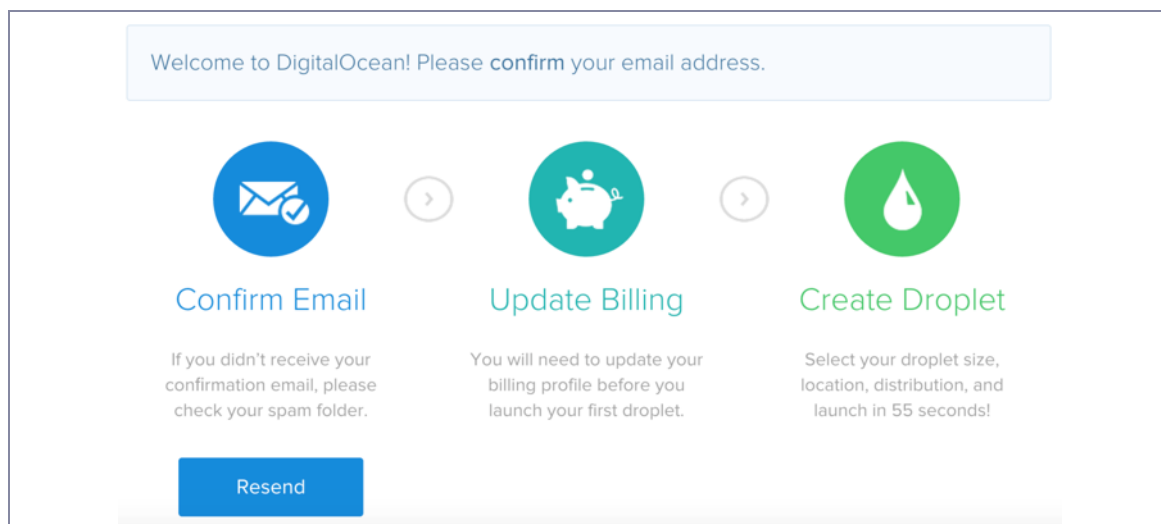


Figura 6.78 Registro en DigitalOcean. Parte 2

Con la cuenta confirmada deberemos proceder al pago de la cuenta. Para ello se proporcionan diferentes métodos de pago.

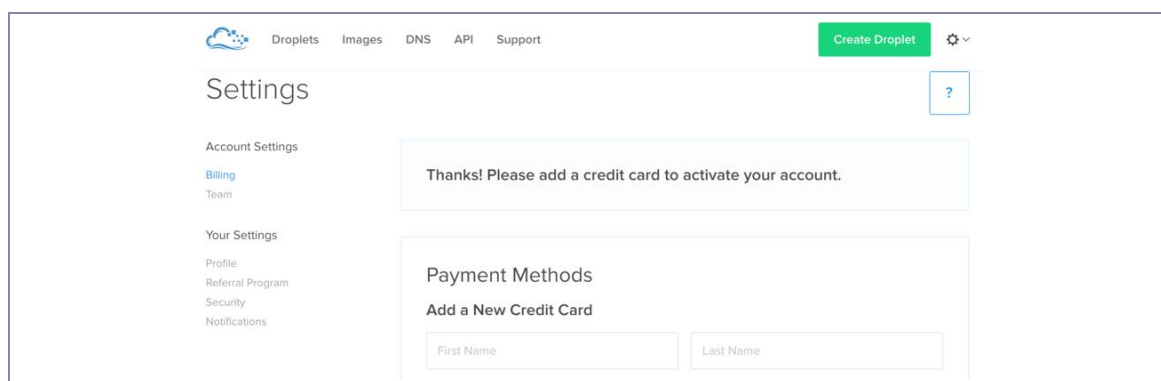


Figura 6.79 Registro en DigitalOcean. Parte 3

6.4.2. Creando un “Droplet”

Con la el pago ya completado el siguiente paso a seguir es crear un “Droplet”. Un “droplet” es el servidor virtual que contendrá la aplicación web.

Para pulsamos en “Create Droplet”.

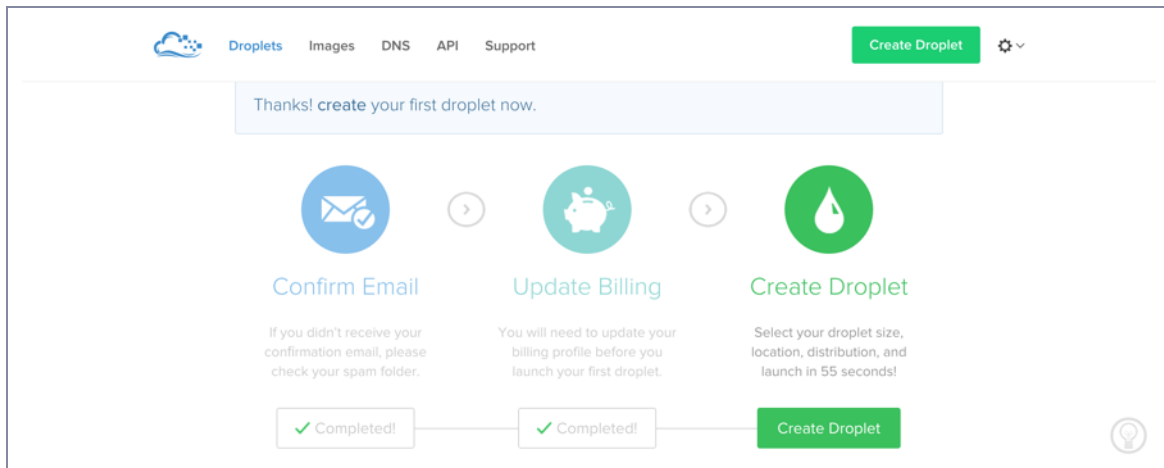


Figura 6.80 Creando un droplet

6.4.2.1. Seleccionando el tipo de “Droplet”

Existen varios tipos de Droplets con diferentes configuraciones *hardware*. Tal y como hemos comentado anteriormente, en el proyecto se ha desarrollado una primera versión de AdSuwak por lo que por el momento es suficiente con el plan más básico.

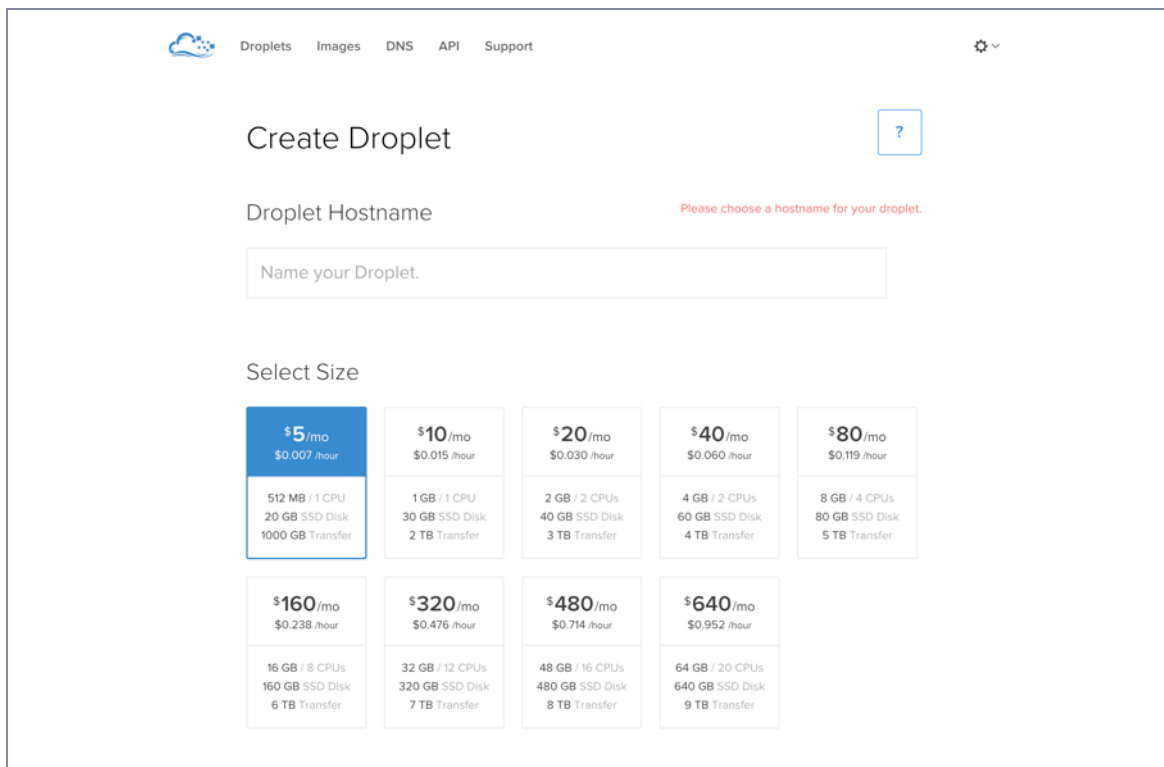


Figura 6.81 Asistente creación droplet

En primer lugar insertamos en el campo “*Droplet Hostname*” el nombre que asignaremos a la máquina virtual y seguidamente seleccionamos el plan al que queremos ceñirnos.

6.4.2.2. Eligiendo la ubicación del servidor

Tal y como se puede apreciar en la figura 6.77 el siguiente paso a completar es la elección de la ubicación del servidor. Existen varias localizaciones disponibles, no obstante, debido a que queremos un servidor lo más próximo posible a España hemos seleccionado como destino Frankfurt.

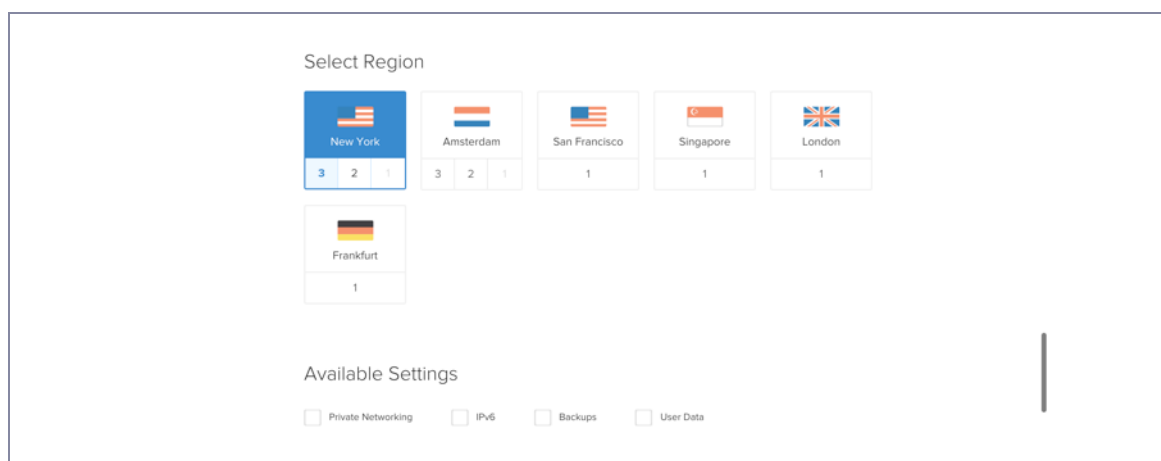


Figura 6.82 Eligiendo ubicación máquina virtual

6.4.2.3. Agregando opciones

Tal y como se muestra en la figura 6.83 se nos da a elegir algunas opciones extra para la configuración del servidor. Como lo que queremos en un primer momento es validar el funcionamiento de la aplicación se decide no escoger ninguna y en caso de necesitarlas activarlas en su debido momento.

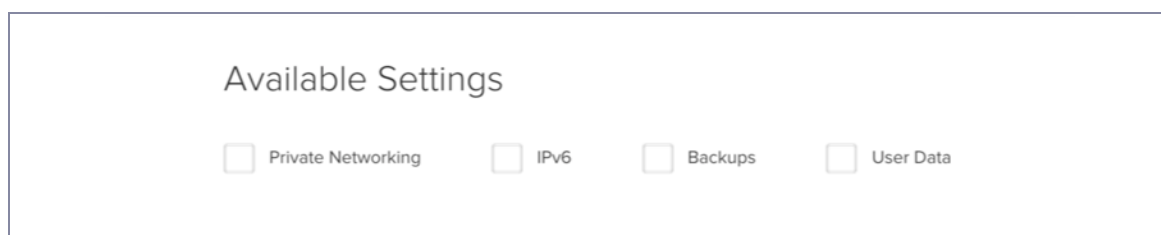


Figura 6.83 Agregando opciones

6.4.2.4. Seleccionando el sistema operativo

La máquina virtual puede tener asignado diferentes sistemas operativos. Nuestra elección ha sido Ubuntu debido a la gran cantidad de información disponible en la red.

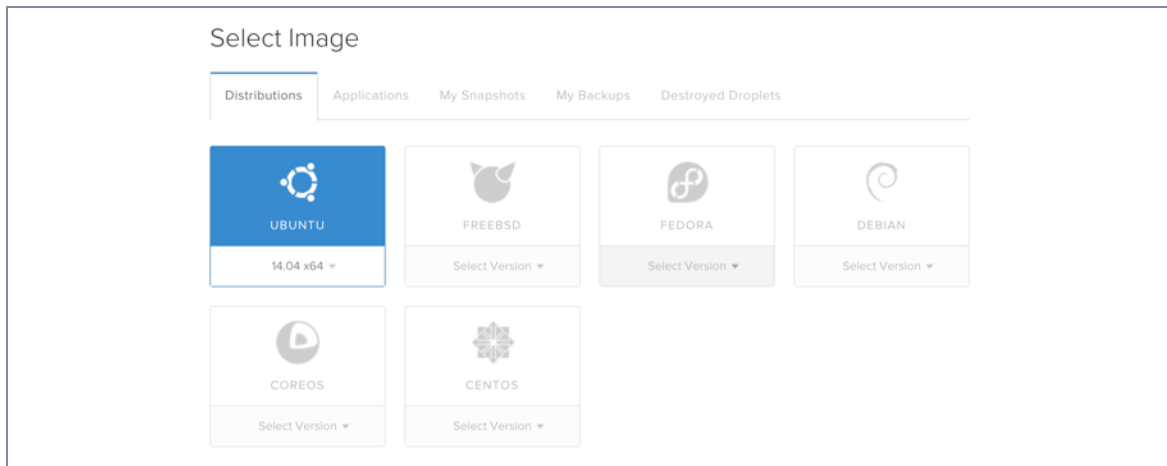


Figura 6.84 Asignando un sistema operativo

Por otro lado, en este mismo paso se nos ofrece la posibilidad de instalar automáticamente diferentes aplicaciones que contendrá el servidor. Se decide no utilizar esta opción e instalar manualmente los diferentes paquetes.

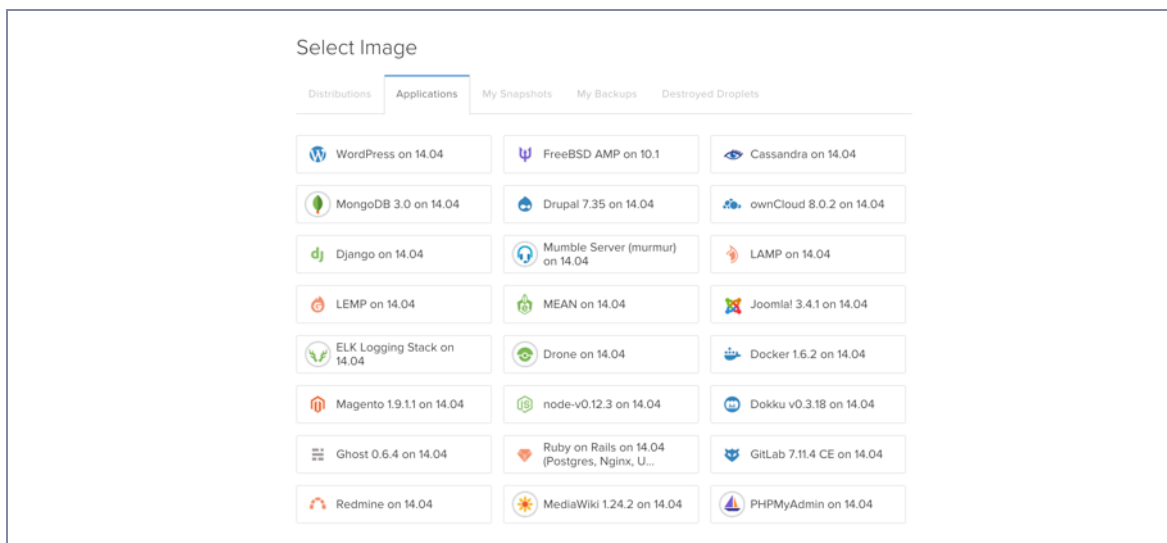


Figura 6.85 Autoinstalador de aplicaciones

6.4.2.5. Añadiendo claves ssh (opcional)

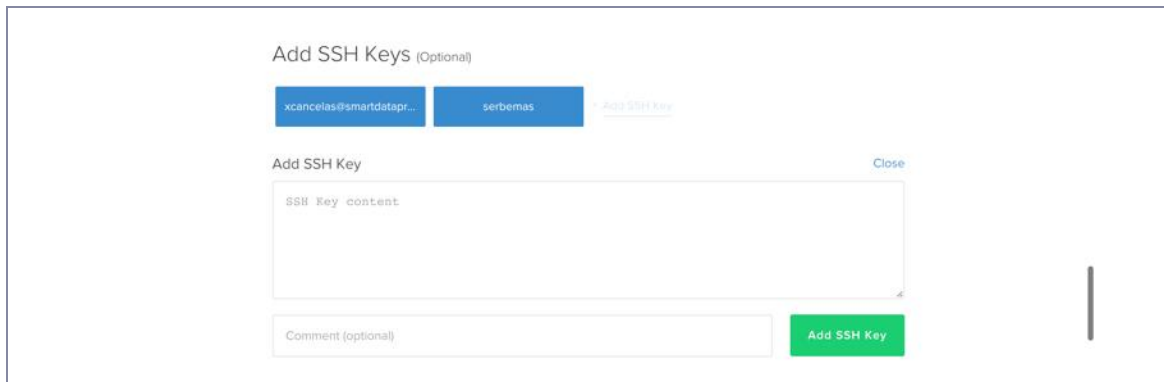


Figura 6.86 Añadiendo claves ssh a la máquina virtual

Como opción extra, se nos ofrece la posibilidad de añadir claves públicas a la máquina virtual para poder realizar la conexión SSH sin necesidad de introducir contraseña cada vez que nos conectemos. En caso de que tengamos ya generadas el par de claves ssh deberemos introducir la clave pública en el campo *Add SSH Key*.

```
$ cat ~/.ssh/id_rsa.pub | pbcopy
```

Nota: Este comando copia al portapeles la clave pública

6.4.2.6. Finalizando la instalación

Con el resto de los pasos anteriores completados podemos proceder a la creación del *Droplet* pulsando en “Create Droplet”.

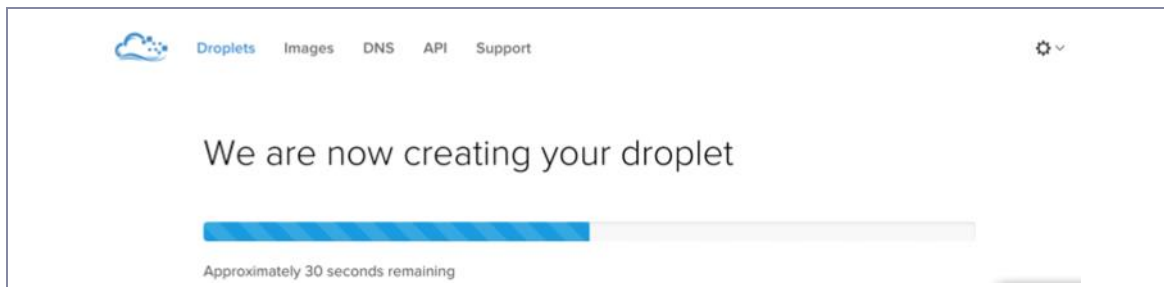


Figura 6.87 Finalizando creación de droplet

6.4.3. Administrando el Droplet

Finalizado el proceso de creación, pasados unos segundos se nos redirigirá a la ventana de administración de los hostings. Como se puede observar en la figura 6.83 se ha creado la nueva máquina con el nombre AdSuwak.


Image	Name	IP Address	Status	Memory	Disk	Region
	adsuwak	46.101.182.187	Active	512 MB	20 GB	fra1

Figura 6.88 Administración de droplet

Para acceder al panel de administración de la máquina virtual debemos pulsar sobre el nombre que anteriormente le hemos asignado.

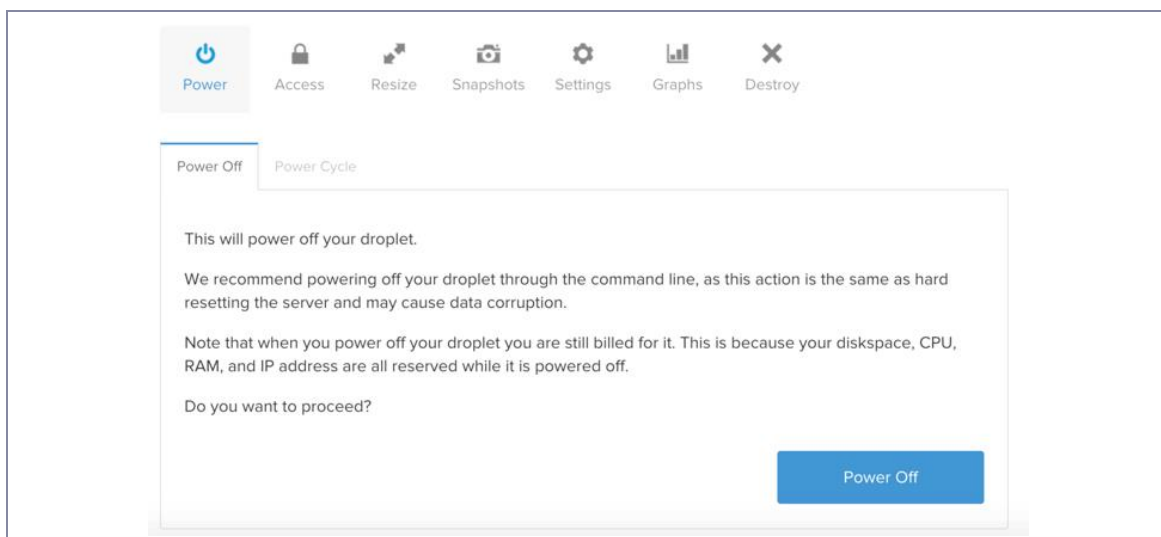


Figura 6.89 Panel administración droplet

6.4.3.1. Accediendo al Droplet

Para acceder al servidor podemos realizarlo de dos diferentes maneras. La primera de ellas y menos recomendable, es usando una consola VNC. VNC, en inglés Virtual Network Computing, es un software el cual permite acceder remotamente a un servidor a través de un ordenador cliente.

La segunda opción, y más recomendable es utilizar el protocolo SSH. SSH, en inglés Secure Shell, es un protocolo el cual permite establecer conexiones seguras utilizando técnicas de cifrado las cuales hacen que la información viaje a través de la red de manera encriptada ocultando así toda la información enviada.

Para este tipo de conexión necesitaremos haber instalado en nuestro sistema operativo un cliente ssh. Con el cliente instalado deberemos utilizar los siguientes comandos:



-Establecer la conexión:

Abrimos el terminal e introducimos:

```
$ ssh nombre_usuario@direccion_ip
```

La primera vez que accedamos por ssh se nos mostrará por el terminal un aviso como:

```
Last login: Wed Jan  4 11:51:18 on console
John-Smiths-Mac:~ macuser$ ssh u12345678@1and1help.com
The authenticity of host '1and1help.com (74.208.111.111)' can't be established.
DSA key fingerprint is 34:47:0f:e9:1a:c2:eb:56:eb:cc:58:59:3a:02:80:b6.
Are you sure you want to continue connecting (yes/no)? yes
```

Figura 6.90 Primera conexión ssh con el servidor

Este aviso se muestra puesto a que es la primera vez que accedemos al servidor y la clave pública no está guardada en el archivo “know_hosts”. Respondemos “yes” para seguir con el proceso de conexión.

Si hemos completado el paso 6.4.2.5, el servidor comprobará que nuestra clave pública es la que previamente hemos configurado y no necesitaremos una contraseña para acceder.

```
serbemas — root@adsuwak: ~ — ssh — 80x21
Last login: Sat Jun 13 22:56:13 on ttys000
MacBook-Pro-de-Sergio-3:~ serbemas$ ssh root@46.101.182.187
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Sat Jun 13 19:16:56 EDT 2015

System load:  0.0          Processes:    67
Usage of /:   7.9% of 19.56GB  Users logged in:  0
Memory usage: 11%          IP address for eth0: 46.101.182.187
Swap usage:   0%

Graph this data and manage this system at:
https://landscape.canonical.com/

38 packages can be updated.
16 updates are security updates.

Last login: Sat Jun 13 19:17:00 2015 from 84.126.134.78.dyn.user.ono.com
root@adsuwak:~#
```

Figura 6.91 Conexión ssh realizada con el servidor

6.4.4. Instalando Plesk

Plesk es un software de administración utilizado en los servidores para facilitar la gestión un hosting web. Permite crear y modificar cuentas de correo electrónico, administrar las bases de datos, gestionar cuentas ftp, configuración de dominios y subdominios, etc.

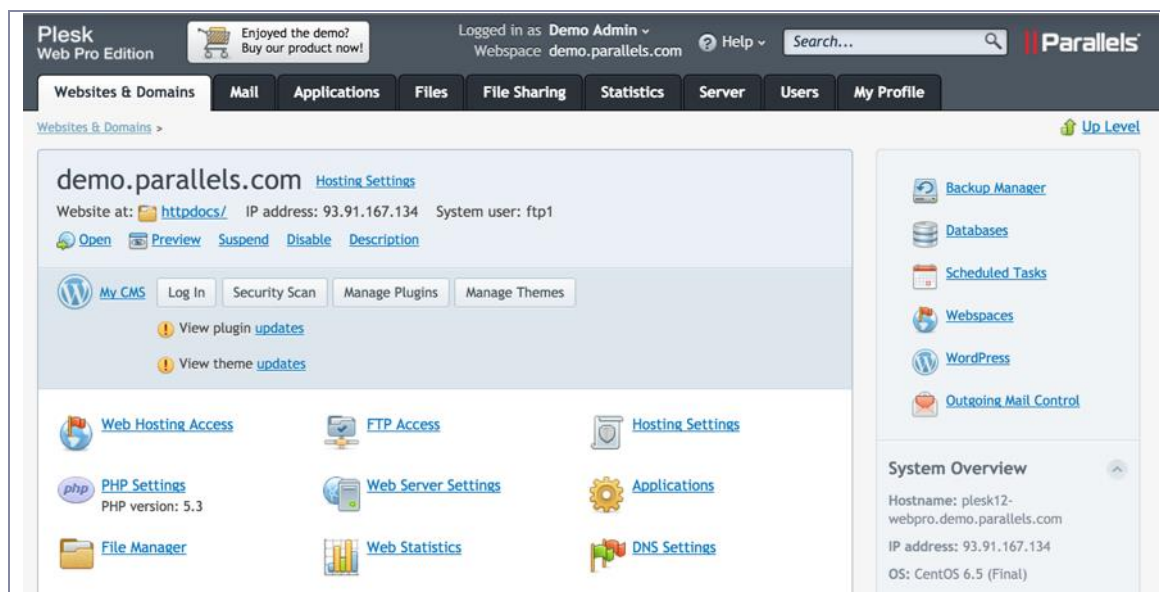


Figura 6.92 Plesk

Es un software opcional, es decir, no es de obligatorio uso para configurar un servidor. No obstante, su instalación facilita muchísimo la administración del servidor automatizando una gran cantidad de tareas que en caso contrario deberíamos ejecutar manualmente sobre el servidor.

En caso de no utilizar una herramienta de este tipo, por ejemplo, deberíamos instalar manualmente un entorno LAMP o similar.

Existen otros paneles de administración de este tipo. Una alternativa sería el panel de administración Cpanel. Actualmente, Cpanel es el panel de control más utilizado en la mayoría de hostings web, no obstante, debido a que Plesk ofrece actualmente un mes de prueba, y que no existen grandes diferencias entre ellos, optamos por esta opción.

6.4.4.1. Instalación

Para la instalación de plesk, en primer lugar deberemos instalar “wget” en nuestro servidor. “Wget” es una herramienta que permite descargar archivos fácilmente desde el terminal. Para ello ejecutamos el siguiente comando:

```
$ apt-get install wget
```

Existen diferentes tipos de instalación de Plesk. El tipo más cómodo y sencillo es la instalación “One Click Installer”.



Podemos utilizar el siguiente comando el cual descargará y ejecutará el instalador de Plesk.

```
$ wget -O - http://autoinstall.plesk.com/one-click-installer | sh
```

Durante la instalación es posible que obtengamos el siguiente error:

“Warning! Unable to detect the fully qualified domain name of the host.”

Para solucionarlo debemos especificar el “host” y el “hostname”. Editaremos los archivos:

- /etc/hosts

-/etc/hostname

En ellos escribiremos respectivamente:

- Dir_ip_servidor adsuwak.adsuwak.com

- adsuwak.adsuwak.com

Con estos cambios iniciamos de nuevo la instalación ejecutando de nuevo el anterior comando.

```
$ wget -O - http://autoinstall.plesk.com/one-click-installer | sh
```

Tras esperar un tiempo habrá finalizado la instalación.

6.4.4.2. Configurando Plesk

Con la instalación finalizada deberemos iniciar sesión en el panel de administración.

Para acceder al nuevo panel deberemos introducir la siguiente dirección en el navegador:

https://direccion_ip_servidor:8443/

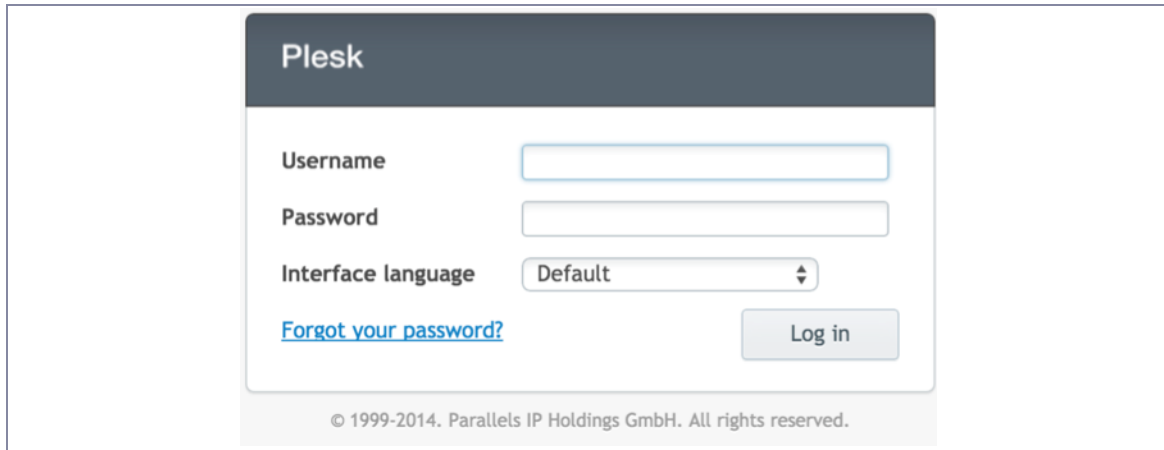


Figura 6.93 Accediendo a Plesk

En esta pantalla deberemos introducir el usuario y la contraseña utilizados para acceder al servidor.

Habiendo iniciado sesión, el servidor nos informará de los términos y condiciones del servicio que deberemos aceptar si queremos utilizar el panel.

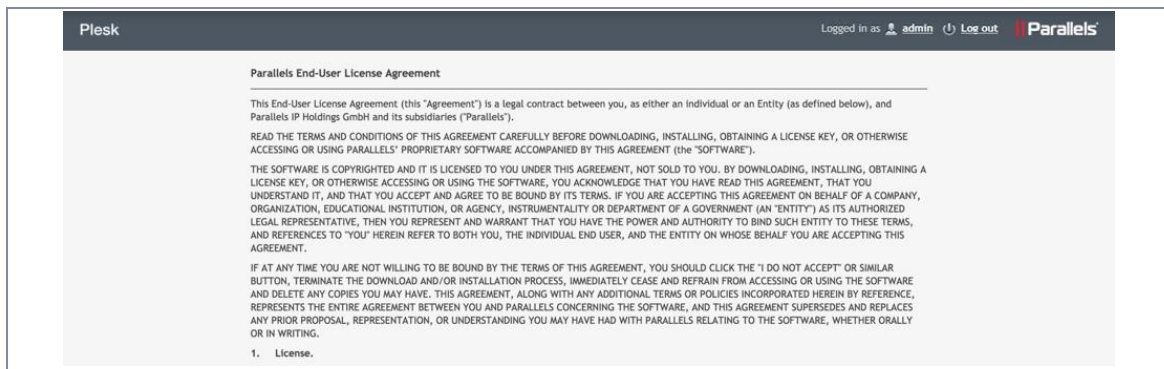


Figura 6.94 Aceptando términos y condiciones

Tras aceptar los términos y condiciones, el siguiente paso a seguir es completar los datos de administrador.



Administrator information

⚠ Warning: Please enter administrator's information. This information is required to operate Parallels Plesk.

Company name *

Contact name *

Phone *

Fax

Email *

Address *

City *

State/Province *

Postal/ZIP code *

Country

I would like to receive security-related information and other technical notifications from Parallels.

Figura 6.95 Formulario de administrador

Cuando hayamos rellenado el formulario, finalmente la aplicación nos redirigirá al panel de administración.

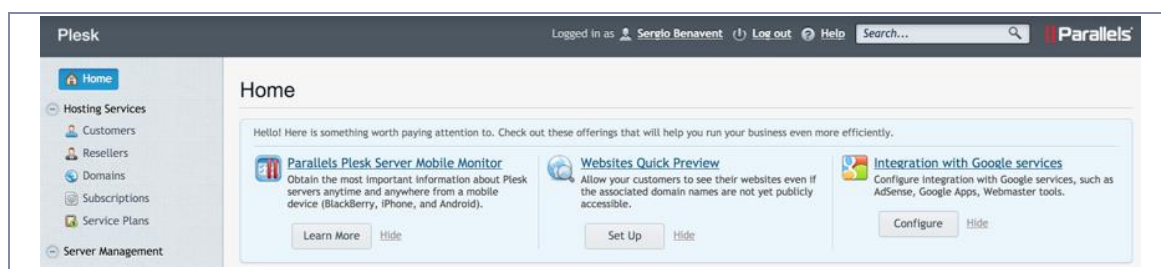


Figura 6.96 Panel de administración de Plesk

6.4.4.3. Seleccionando la licencia

Antes de poder empezar a configurar todo el servidor, necesitamos obtener una licencia.

La primera vez que accedamos al panel nos mostrará el aviso de la figura 6.92. En él nos informa de lo que acabamos de comentar y además proporciona un enlace para obtener la licencia gratuita.

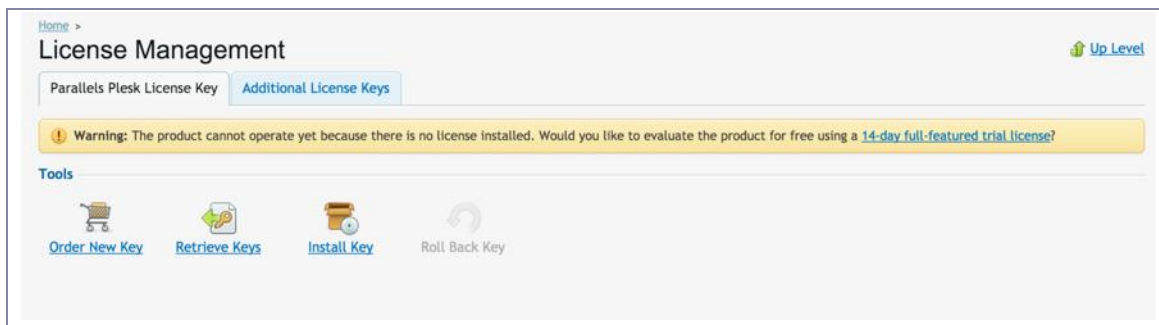


Figura 6.97 Aviso de licencia

Pulsando sobre el enlace, nos redirigirá a la página de compra. En ella existen diferentes tipos de compra, seleccionaremos la opción “15-day trial”.

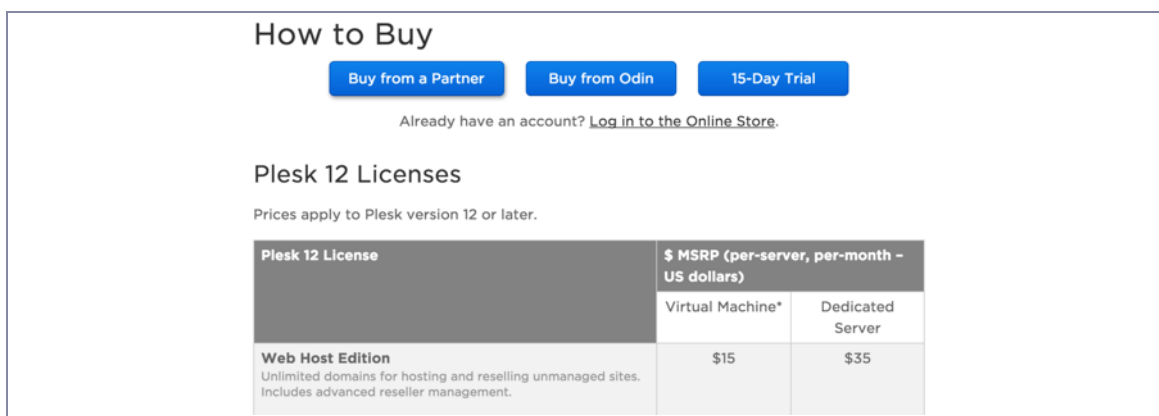


Figura 6.98 Adquiriendo licencia gratuita

Tras proporcionar los datos que nos solicitan, obtendremos un archivo xml el cual contendrá la licencia. Para instalarla simplemente iremos al gestor de licencias “License Manager” y pulsaremos sobre “Install key”.

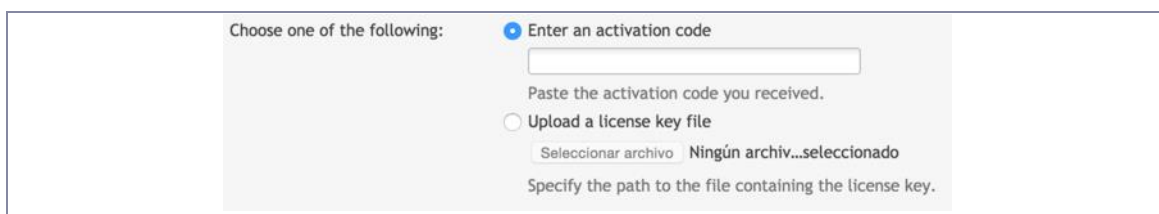


Figura 6.99 Instalando la licencia

Finalmente nos dará a elegir entre introducir un código o subir al servidor la nueva licencia. Escogeremos la segunda opción.

6.4.4.4. Creando suscripciones

Para la gestión de nuestro servidor deberemos crear una nueva suscripción o dominio.

Para ello, tal y como se muestra en la figura 6.95, en el menú de la izquierda pulsamos en “Subscriptions”. Seguidamente en la nueva ventana pulsamos en “Add New Subscription”.



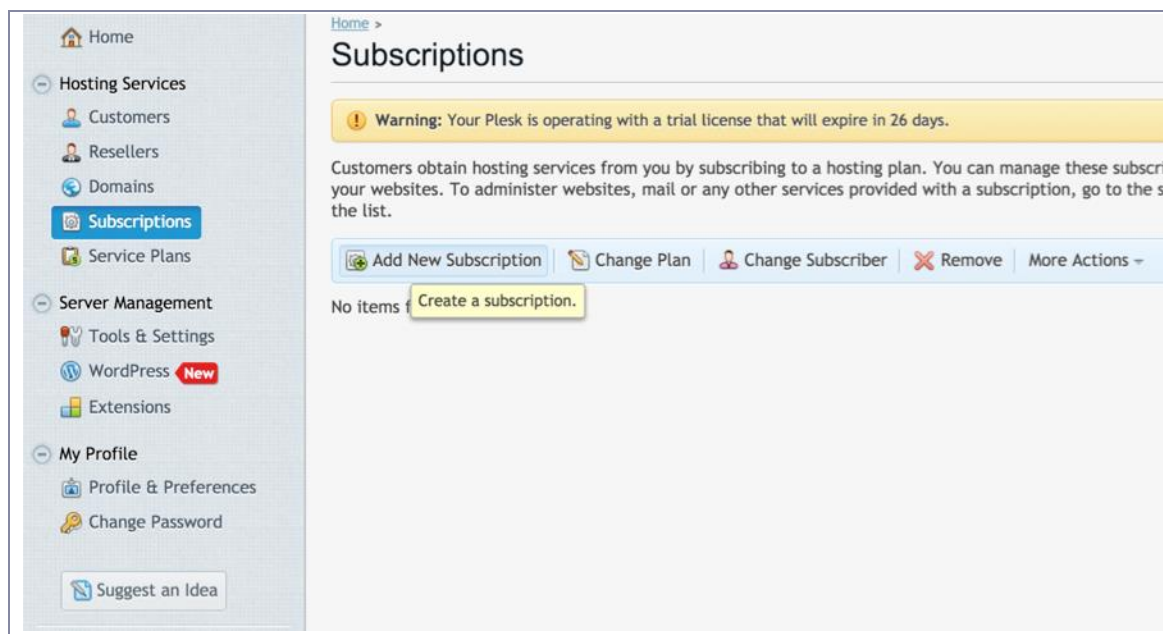


Figura 6.100 Creando nueva suscripción. Parte 1

Ahora deberemos configurar el nombre el domino que utilizará el servidor y crear un usuario asociado.

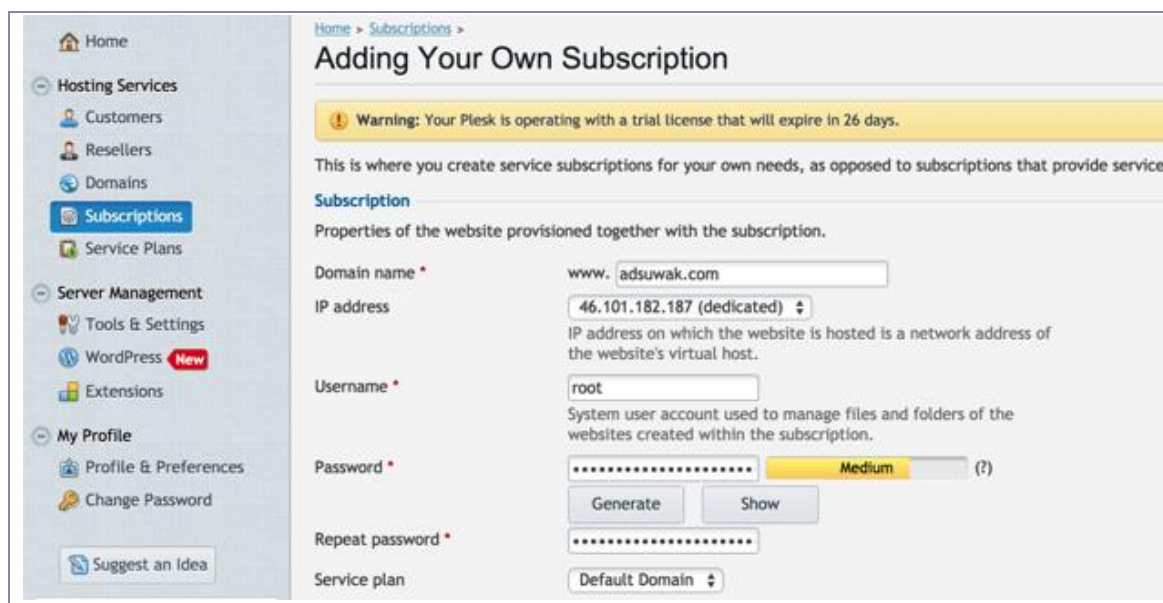


Figura 6.101 Creando nueva suscripción. Parte 2

Tras completar todos los campos correspondientes, si no se produce ningún error deberíamos ver por pantalla algo como la siguiente figura.

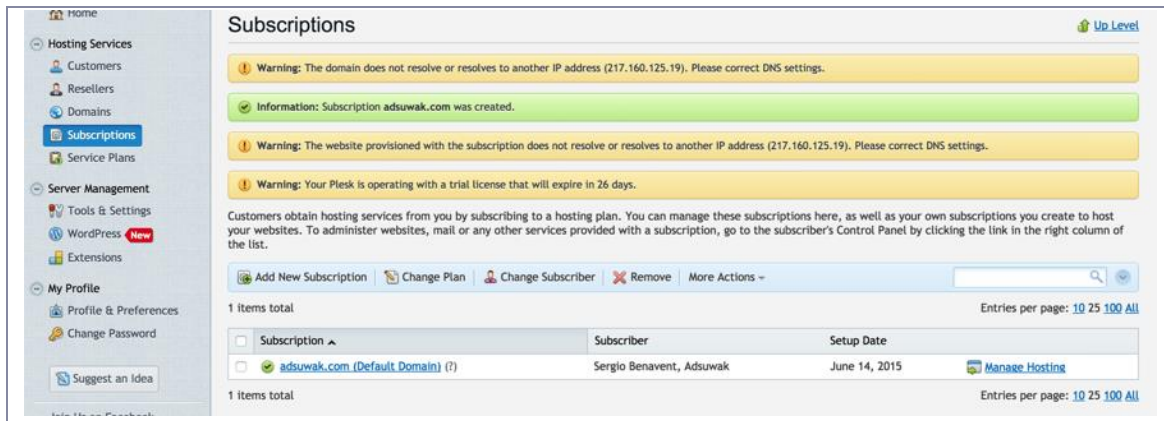


Figura 6.102 Creación de subscripción finalizada

6.4.4.5. Creando cuentas de correo

En esta sección crearemos las cuentas de correo que posteriormente utilizaremos para el envío de correos desde el servidor.

En primer lugar debemos acceder a la subscripción que creamos en el paso anterior y en el menú superior seleccionar la opción "Mail".



Figura 6.103 Creando cuenta de correo. Parte 1

Pulsamos sobre la opción "Create Email Address" y se nos mostrará un formulario como el de la figura 6.104



Figura 6.104 Creando cuenta de correo. Parte 2

Existen diferentes opciones de configuración de la cuenta de email. En nuestro caso crearemos una cuenta sencilla asignándole un nombre y una contraseña.

Email address	User	Usage
abuse@adsuwak.com	abuse@adsuwak.com	4.00 KB used of Unlimited

Figura 6.105 Creando cuenta de correo. Parte 3

Tras completar estos sencillos dispondremos de una cuenta de email ya lista para enviar y recibir correos.

6.4.4.6. Creando una base de datos

En este apartado crearemos una base de datos en la que guardaremos toda la información de nuestra aplicación. Para ello, accederemos a la suscripción que hemos creado en el apartado anterior.

En la suscripción, en el apartado “Websites and Domains” del menú superior, entramos en el apartado “Databases”.

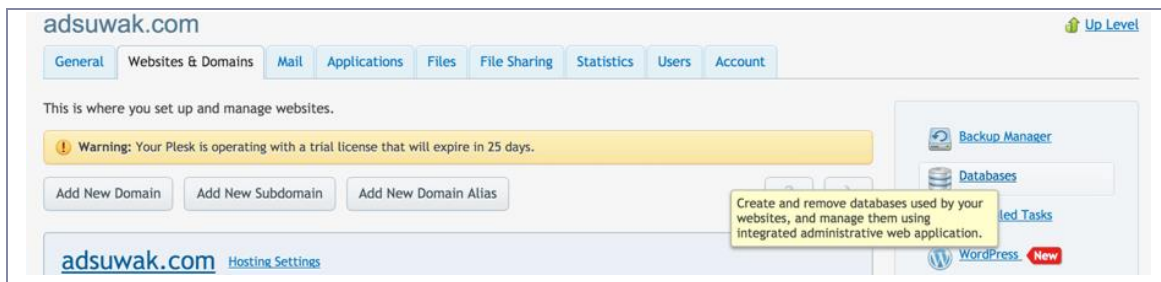


Figura 6.106 Creando base de datos. Parte 1

En esta sección, deberemos ahora crear una nueva base de datos. Pulsamos en “Add New Database”.

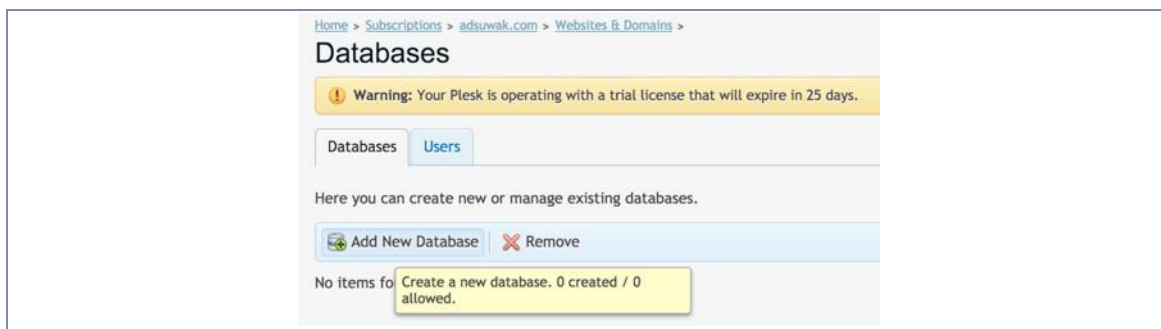


Figura 6.107 Creando base de datos. Parte 2

En este nuevo apartado debemos definir los atributos de la base de datos. En primer lugar deberemos asignar un nombre que la identifique únicamente. Seguidamente deberemos crear un usuario para acceder a ella.

Con estos dos simples pasos, habremos creado nuestra base de datos para la aplicación.

6.4.5. Instalando GIT

A lo largo del desarrollo de la aplicación se ha usado el controlador de versiones GIT. Por ello, y tal y como hemos explicado en apartados anteriores debemos instalar a su vez este software en el servidor.

```
$ sudo apt-get install git
```

6.4.6. Generando claves ssh en el servidor.

Para poder acceder al repositorio online necesitaremos previamente crear en la máquina virtual el par de claves pública y privada que identifique unívocamente nuestra máquina.

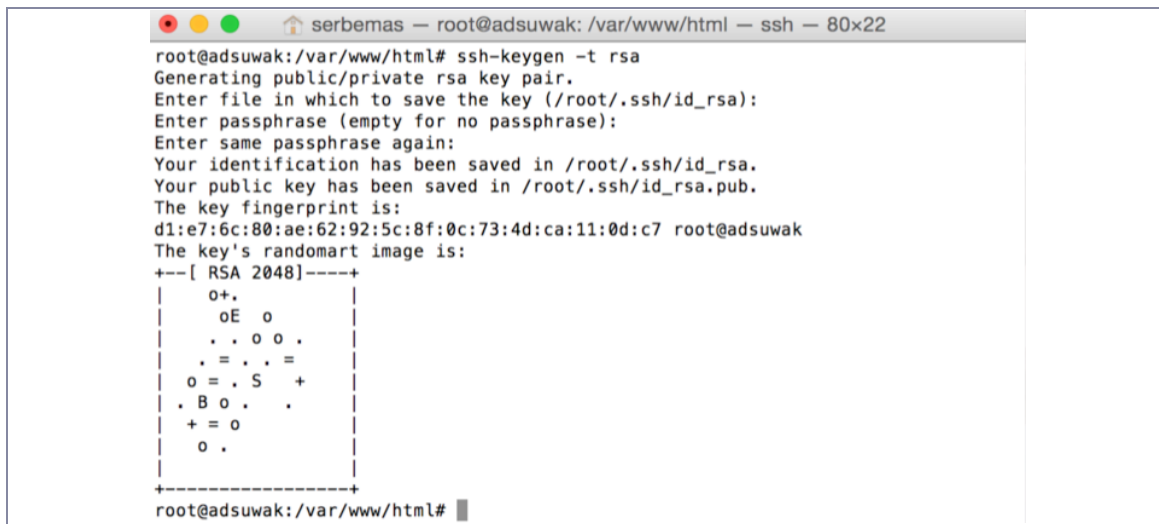


Para ello deberemos ejecutar el siguiente comando:

```
$ ssh-keygen -t rsa
```

Una vez introducido se nos mostrará en el terminal las siguientes opciones:

- Directorio en el que guardaremos las claves.
- Contraseña de las claves



```
serbemas — root@adsuwak: /var/www/html — ssh — 80x22
root@adsuwak:/var/www/html# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
d1:e7:6c:80:ae:62:92:5c:8f:0c:73:4d:ca:11:0d:c7 root@adsuwak
The key's randomart image is:
+--[ RSA 2048]-----+
  o+.
   oE o
  . . o o .
 . = . . =
 o = . S +
 . B o . .
 + = o
  o .
+-----+
root@adsuwak:/var/www/html#
```

Figura 6.108 Creación claves SSH

Tal y como se muestra en la figura 6.103 para el directorio hemos elegido el por defecto y hemos introducido una contraseña.

Tras estos sencillos pasos tenemos ya creadas en el servidor el par de claves.

Ahora debemos asignar estas claves al servidor virtual. Para ello ejecutamos en el terminal:

```
$ ssh-copy-id root@dir_ip_servidor
```

Con este comando, el servidor ahora usará las últimas claves creadas.

6.4.7. Clonando el proyecto en el servidor.

Una vez instalado git podremos clonar nuestro proyecto en el servidor. Para ello nos dirigiremos a la carpeta en la que el servidor apache2 muestra las páginas web.

```
$ cd /var/www/html
```

Una vez situados en este punto ejecutaremos el siguiente comando, el cual descargará la rama “development” del repositorio

```
$ git clone -b development -single-branch git@bitbucket.org:serbemas/adsuwak.git
```



```
git clone git@bitbucket.org:serbemas/adswak.git
```

Para que el servidor pueda acceder al repositorio online, deberemos haber añadido previamente su clave pública al repositorio.

Para ello ejecutamos el comando:

```
cat ~/.ssh/id_rsa.pub | pbcopy
```

Este comando nos copiará el contenido de la clave pública en el portapapeles.

Debemos ahora añadir la clave al repositorio online.

En este punto podremos, habiéndole permitido el acceso del server al repositorio, podemos clonar el proyecto. Para ello introducimos de nuevo:

6.4.8. Configurando el proyecto con el servidor

En este punto, habiendo seguido los anteriores apartados deberemos tener en nuestra carpeta del servidor una carpeta con nuestro proyecto, en nuestro caso AdSuwak.

Ahora el primer paso que debemos hacer, es resolver las dependencias que pueda tener el proyecto dentro del servidor. Para ello utilizaremos Composer y actualizaremos sus dependencias.

6.4.8.1. Actualizando composer

Nos situamos en el interior del proyecto y ejecutamos el siguiente comando:

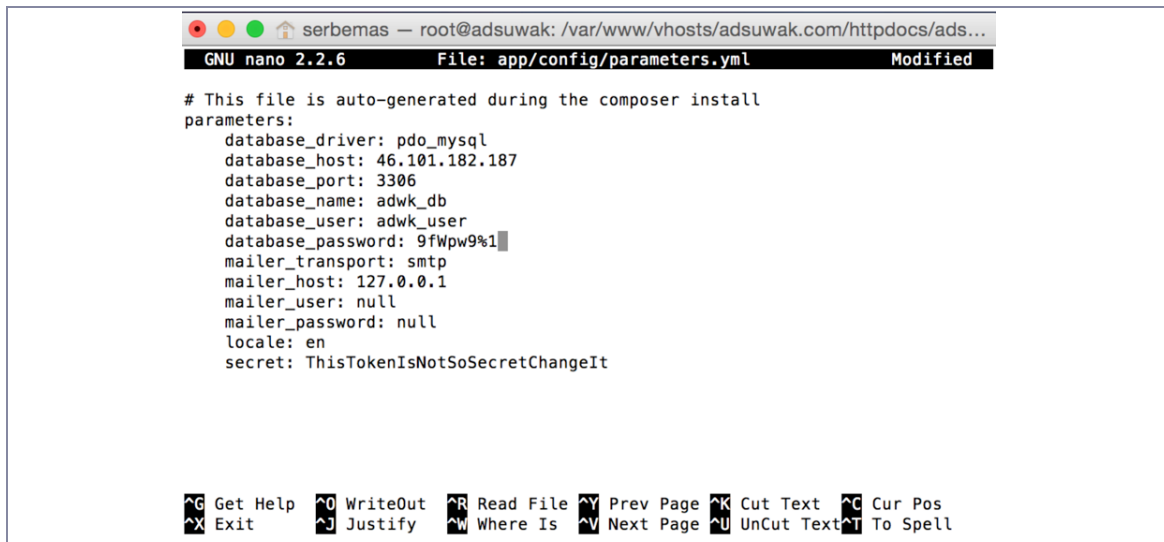
```
$ php composer.phar self-update
```

Seguidamente, una vez se haya descargado la última versión de Composer, actualizaremos las dependencias del proyecto:

```
$ php composer.phar update
```

Al final del proceso deberemos configurar los parámetros correspondientes al servidor.





```

serbemas — root@adsuwak: /var/www/vhosts/adsuwak.com/httpdocs/ads...
GNU nano 2.2.6 File: app/config/parameters.yml Modified

# This file is auto-generated during the composer install
parameters:
  database_driver: pdo_mysql
  database_host: 46.101.182.187
  database_port: 3306
  database_name: adwk_db
  database_user: adwk_user
  database_password: 9fWpw9%1
  mailer_transport: smtp
  mailer_host: 127.0.0.1
  mailer_user: null
  mailer_password: null
  locale: en
  secret: ThisTokenIsNotSoSecretChangeIt

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
    
```

Figura 6.109 Modificando parameters.yml

6.4.9. Creando la base de datos

En la sección 6.4.4.6 creamos una base de datos desde el panel. Por ello, el siguiente paso es crear el esquema de la base de datos sobre la base de datos.

Creamos el esquema:

```
$ php app/console doctrine:schema:create
```

Creamos las entidades:

```
$ php app/console doctrine:generate:entities nombre_del_Bundle
```

Actualizamos base de datos:

```
$ php app/console doctrine:schema:update --dump-sql
```

```
$ php app/console doctrine:schema:update --force
```

6.4.9.1. Cambiando permisos

Debemos ahora cambiar los permisos de las carpetas “app/cache” y “app/logs” para que el servidor pueda modificarlas.

-Modificar permisos app/cache:

```
chmod -R 777 app/cache/
```

-Modificar permisos app/log:

```
$ sudo chmod -R 777 app/logs
```

-Para finalizar borramos la cache:

```
$ php symfony cache:clear
```

6.4.10. Configurando el DNS

El sistema DNS, en inglés “*Domain Name System*” es un servicio vital utilizado para traducir las direcciones ip en nombres de dominios. Este sistema de traducción de direcciones ip facilita enormemente la navegación debido a que evita que tengamos que recordar la dirección IP del servidor y las direcciones sean más fáciles de recordar.

Para acceder a la máquina virtual que hemos estado configurando a lo largo de este apartado, hasta ahora lo hemos hecho mediante su dirección ip. En este apartado asociaremos un dominio que previamente hemos obtenido y lo configuraremos para que apunte al servidor. Con esto los usuarios podrán acceder fácilmente escribiendo en el navegador:

www.adsuwak.com

6.4.10.1. Registros tipo A/AAAA

En nuestro caso disponemos el dominio “adsuwak.com” previamente comprado. Por lo tanto entraremos en su panel de administración y lo configuraremos:



The screenshot shows a web interface titled "Registros A/AAAA (Direcciones IP)". It features two radio buttons for "Dirección IP (Registro A)": "Dirección IP 1&1" (unselected) and "Otra dirección IP" (selected). Below this, there is a note: "Los registros CNAME solo son soportados para subdominios. Para redireccionar su dominio, vaya a > Configuración de dominio → Editar destino". There are two input fields: "Dirección IPv4:" with the value "46.101.182.187" and "Dirección IPv6:" which is empty. Both fields have a help icon (question mark) to their right.

Figura 6.110 DNS tipo A/AAAA

En la figura se muestran los registros de tipo A/AAAA asociados al dominio. Este tipo de registros son los utilizados para apuntar un “hostname” a una dirección ip. Los registros de tipo A corresponden a las direcciones IPv4 y los de tipo AAAA a direcciones IPv6.

Como el servidor no está configurado con direcciones IPv6 únicamente añadiremos al dominio la dirección IPv4 asociada a la máquina virtual.



6.4.10.2. Registros de tipo MX

Figura 6.111 DNS tipo MX

Al crear una subscripción se crean automáticamente los subdominios correspondientes a los servicios ftp o mail tales como ftp.adsuwak.com o mail.adsuwak.com.

Los subdominios debemos asociarlos también a un dominio en el DNS.

En la figura 6.107, se procede a configurar un nuevo subdominio. En este caso, como el subdominio es utilizado para la gestión del correo debemos configurar además de los registros de tipo A/AAAA los registros de tipo MX

Figura 6.112 Configuración DNS de subdominio

Los registros “Mail Exchange Record” MX, en español “registro de intercambio de correo”, especifica cuáles son los servidores de correo del servidor.

7. Pruebas

Una vez finalizado el desarrollo de la aplicación, esta debe cumplir con los requisitos establecidos y cumplir con una serie de estándares que aseguren el correcto funcionamiento de la misma.

7.1. Validación de estándares

Para evaluar la aplicación, en primer lugar someteremos el código fuente los validadores que ofrece W3C (“World Wide Web Consortium”) mediante los que comprobaremos que no existan errores en el código HTML y CSS.

Existen diversas herramientas que permiten realizar esta tarea. En el proyecto utilizaremos la herramienta que proporciona W3C “Unicorn”. Esta herramienta nos permite validar para un mismo documento las tecnologías:

- HTML
- CSS
- MovilOk (adaptable a dispositivos móviles)
- RSS/Atom

Existen validadores que realizan estas tareas por separado, no obstante, se decide la utilización de Unicorn debido a que nos permite comprobarlo todo al mismo tiempo.

Para realizar el test debemos acceder a la página oficial de la herramienta:

<https://validator.w3.org/unicorn/>

Tras la primera ejecución el validador nos devuelve una serie de errores. En la figura 7.1 mostramos un ejemplo de cuáles son los errores a corregir



Figura 7.1 Ejemplo errores en la validación

Tras realizar una revisión de todos los errores cometidos y haber procedido a su solución, finalmente se muestra la siguiente imagen cuando todo está correctamente:



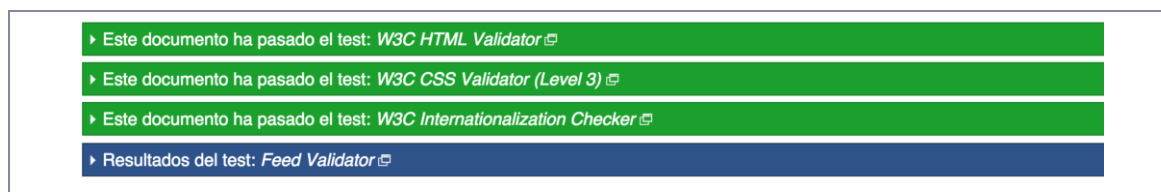


Figura 7.2 Ejemplo página correcta

7.2. Pruebas funcionales

La aplicación además de estar desarrollada con un código correcto y que se atenga a los estándares, debe cumplir con toda la funcionalidad descrita en la especificación de requisitos.

Mediante las pruebas funcionales se valida el cumplimiento de las especificaciones funcionales para detectar posibles errores producidos en la fase de desarrollo.

Este proceso es importante debido a que los defectos encontrados y solventados durante la fase de pruebas supondrán un ahorro en tiempo y dinero.

Para su realización, la aplicación ha sido probada manualmente en su versión final, es decir, desde el servidor. Para ello se han reproducido todos los casos descritos que debe presentar la aplicación.

7.3. Pruebas distintas resoluciones

Otro aspecto que se debe comprobar en la aplicación, es la correcta visualización de los elementos en pantallas con diferentes resoluciones.

Para poder llevar a cabo este tipo de prueba, existen complementos que redimensionan la ventana del navegador a una resolución concreta y poder así comprobar fácilmente su visualización.

Para llevarlo a cabo, se opta por utilizar el plugin “Window Resizer” para el navegador “Google Chrome”, el cual podemos encontrar en su página oficial.

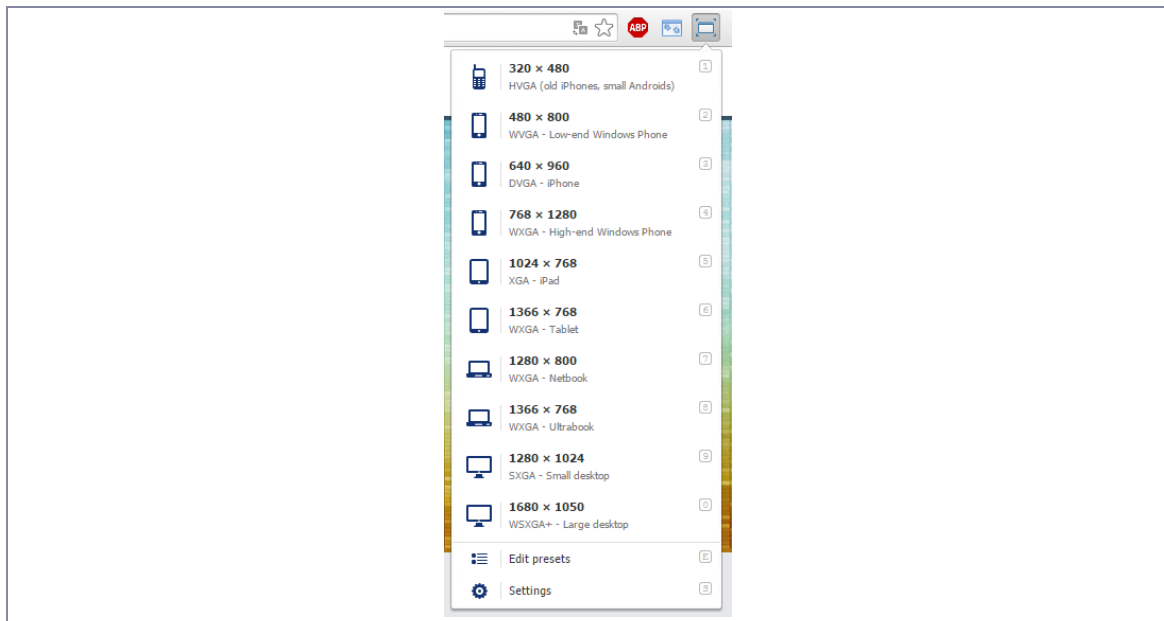


Figura 7.3 Ejemplo resoluciones plugin Windows Resizer.

Esta herramienta redimensiona la pantalla del navegador al tamaño elegido.

Tras probar con todos los tamaños disponibles, se concluye que para tamaños igual o superiores a la resolución 640 x 960 la aplicación se adapta correctamente. A su vez, para los tamaños inferiores a esa resolución los elementos están muy juntos y no resulta muy usable para el usuario.

8. Conclusión

La idea de Adsuwak surge debido a que actualmente el banner ya no está cumpliendo con su función. El principal problema es que resultan prácticamente invisibles al usuario y ya no logran captar su atención.

A pesar de la baja efectividad que tienen hoy en día, hay estudios que calculan que solamente en España, el pasado año, se invirtieron más de mil millones de euros en este tipo de publicidad.

Por ello, surge la necesidad de crear un nuevo tipo de anuncio que ya no logre únicamente captar su atención, sino que además le permita realizar acciones desde el mismo para evitar que sea publicidad invasiva. Para lograrlo, el nuevo anuncio inteligente se deberá colocar en el contenido de la web y estar totalmente relacionado con el contenido.

Con esta idea básica, por una parte se decide realizar una plataforma en la que los anunciantes puedan crear sus propios anuncios interactivos y adaptarlos para que se adapten lo mejor posible al contenido y por otra parte, se decide realizar tres tipos diferentes de anuncios inteligentes que permitan al usuario realizar distintas acciones.

Para el desarrollo de la plataforma se ha elegido el framework Symfony2 debido a que es un framework que cada vez está más demandado en el mundo laboral y que es muy utilizado por las “startups”. Este framework tiene una curva de aprendizaje muy alta en la que en poco tiempo el usuario logra aprender una gran cantidad de información.

Para la creación de los anuncios interactivos se decide utilizar únicamente JavaScript y no una librería tal como JQuery para evitar sobrecargar en exceso la página y permitir que funcione en el mayor número posible de navegadores.

La aplicación desarrollada es una primera aproximación a la versión final. Será utilizada para terminar de validar la idea y valorar qué cambios hay que realizar para que realmente logre cumplir con su cometido.

A nivel personal, la realización del proyecto ha supuesto un gran paso en mi formación profesional. He estudiado tecnologías en una profundidad que en la carrera no había podido trabajar.

En el trabajo se han tratado temas de diferentes áreas. En primer lugar, más orientado al sector empresarial, se trabaja con un modelo de negocio que actualmente utilizan las empresas más populares del momento tales como Twitter o Facebook. Seguidamente, mediante el desarrollo e implementación de la aplicación se tratan temas relacionados la ingeniería del software y por último con la configuración de un hosting web desde cero se trabaja como administrador de sistemas.

Todo ello ha supuesto tener que dedicarle una gran cantidad de tiempo a investigar sobre los diferentes temas a desarrollar y buscar y comparar mucha información para luego ponerlo en práctica.



La memoria se ha pretendido que sea lo mayor clara posible respecto a temas de implementación y desarrollo, puesto que entiendo que puede servir de ayuda a gente que se encuentre en un futuro en mi misma situación.

El proyecto de final de grado ha sido una oportunidad que no he querido desaprovechar y la cual me ha permitido afianzar los conocimientos adquiridos durante los cuatro años de la carrera y además ponerlos en práctica en un caso real. Todo ello me hace sentirme preparado para incorporarme al mundo laboral.



9. Bibliografía

- [1] Ceguera Banner o Banner Blindness. Wikipedia [online]. [Fecha de consulta: 08-05-2015]. Disponible en: https://es.wikipedia.org/wiki/Banner_blindness
- [2] Click Through Rate, CTR. Wikipedia [online]. [Fecha de consulta: 08-05-2015]. Disponible en: https://es.wikipedia.org/wiki/Proporci%C3%B3n_de_clics
- [3] IEE. IEE [online]. [Fecha de consulta: 13-05-2015]. Disponible en: <https://www.ieee.org/index.html>
- [4] Especificación de requisitos según el estándar de IEEE 830. Universidad complutense de informática [online]. [Fecha de consulta: 14-05-2015]. Disponible en: <https://www.fdi.ucm.es/profesor/gmendez/docs/iso809/ieee830.pdf>
- [5] Lenguaje UML. UML [online]. [Fecha de consulta: 22-06-2015]. Disponible en: <http://www.uml.org/>
- [6] Casos de uso. Wikipedia [online]. [Fecha de consulta: 28-06-2015]. Disponible en: https://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso
- [7] Diagrama entidad relación. Wikipedia [online]. [Fecha de consulta: 29-06-2015]. Disponible en: https://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n
- [8] Diagramas de actividad. Computadores y Tiempo Real [online]. [Fecha de consulta: 30-06-2015].
http://www.ctr.unican.es/asignaturas/procodis_3_ii/doc/statediagram.pdf
- [9] Modelos de Negocio, Apuntes asignatura Modelos de Negocio y Areas Funcionales. ETSIInf
- [10] Modelo de negocio. Definición [online]. [Fecha de consulta: 02-05-2015]. Disponible en: <http://definicion.de/modelo-de-negocio/>
- [11] Modelo de negocio. Wikipedia [online]. [Fecha de consulta: 02-05-2015]. Disponible en: https://es.wikipedia.org/wiki/Modelo_de_negocio
- [12] Libro Business Model Generation. Autores: Alexander Osterwalder, Yves Pigneur.
- [13] Customer development. Emprenderalia [online] [Fecha de consulta: 03-05-2015]. Disponible en: <http://www.emprenderalia.com/aprende-a-crear-tu-startup-con-customer-development/>
- [14] Canales. Creascion [online] [Fecha de consulta: 04-05-2015]. Disponible en: <http://creascion.com/conociendo-el-modelo-canvas-parte-4-canales/>
- [15] Relaciones con clientes. Creascion [online] [Fecha de consulta: 04-05-2015]. Disponible en <http://creascion.com/conociendo-el-modelo-canvas-parte-5-relaciones-con-clientes/>

- [16] Recursos clave. Creacion [online] [Fecha de consulta: 04-05-2015]. Disponible en: <http://creacion.com/conociendo-el-modelo-canvas-parte-7-recursos-clave/>
- [17] Actividades Clave. Creacion [online] [Fecha de consulta: 04-05-2015]. Disponible en: <http://creacion.com/conociendo-el-modelo-canvas-parte-8-actividades-clave/>
- [18] Arquitectura Modelo-Vista-Controlador. Wikipedia [online] [Fecha de consulta: 22-05-2015]. Disponible en: <https://es.wikipedia.org/wiki/Modelo%2%80%93vista%2%80%93controlador>
- [19] HTML. W3C [online] [Fecha de consulta: 03-05-2015]. Disponible en: <http://www.w3schools.com/html/>
- [20] HTML. Wikipedia [online] [Fecha de consulta: 03-05-2015]. Disponible en: <https://es.wikipedia.org/wiki/HTML>
- [21] HTML. W3C [online] [Fecha de consulta: 03-05-2015]. Disponible en: https://es.wikipedia.org/wiki/World_Wide_Web_Consortium
- [22] Javascript. Wikipedia [online] [Fecha de consulta: 03-05-2015]. Disponible en: <https://es.wikipedia.org/wiki/JavaScript>
- [23] Javascript. Librosweb [online] [Fecha de consulta: 03-05-2015]. Disponible en https://librosweb.es/libro/javascript/capitulo_1.html
- [24] CSS. Wikipedia [online] [Fecha de consulta: 03-05-2015]. Disponible en: https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
- [25] PHP. PHP [online] [Fecha de consulta: 03-05-2015]. Disponible en: <http://php.net/manual/es/intro-what-is.php>
- [26] PHP. Wikipedia [online] [Fecha de consulta: 03-05-2015]. Disponible en: <https://es.wikipedia.org/?title=PHP>
- [27] Ajax. Wikipedia [online] [Fecha de consulta: 03-05-2015]. Disponible en: <https://es.wikipedia.org/wiki/AJAX>
- [28] Twitter Bootstrap. Wikipedia [online] [Fecha de consulta: 03-05-2015]. Disponible en: https://es.wikipedia.org/wiki/Twitter_Bootstrap
- [29] JQuery. Wikipedia [online] [Fecha de consulta: 03-05-2015]. Disponible en: <https://es.wikipedia.org/wiki/JQuery>
- [30] JQuery API. JQuery [online] [Fecha de consulta: 05-05-2015]. Disponible en: <https://api.jquery.com/>
- [31] Git fast-version-control. GIT [online] [Fecha de consulta: 06-05-2015]. Disponible en: <https://git-scm.com/>



- [32] GIT. Wikipedia [online] [Fecha de consulta: 06-05-2015]. Disponible en: <https://es.wikipedia.org/wiki/Git>
- [33] Bitbucket. Wikipedia [online] [Fecha de consulta: 06-05-2015]. Disponible en: <https://bitbucket.org/>
- [34] XAMPP. Slideshare [online] [Fecha de consulta: 12-05-2015]. Disponible en: <http://es.slideshare.net/kissees/xampp-25917007>
- [35] WAMPP. Slideshare [online] [Fecha de consulta: 12-05-2015]. Disponible en: <http://es.slideshare.net/aimerodriguezrodriguez/que-es-wamp-server>
- [36] Apache. Apachefriends [online] [Fecha de consulta: 12-05-2015]. Disponible en: <https://www.apachefriends.org/es/index.html>
- [37] Apache. Apachefriends [online] [Fecha de consulta: 11-04-2015]. Disponible en: <https://www.apachefriends.org/es/index.html>
- [38] Libro Desarrollo web ágil con Symfony2. Autor: Javier Eguiluz
- [39] Symfony 2.3. Librosweb [online] [Fecha de consulta: 13-04-2015]. Disponible en: http://librosweb.es/libro/symfony_2_3/
- [40] Symfony sitio web oficial. Symfony [online] [Fecha de consulta: 13-04-2015]. Disponible en: <http://symfony.com/>
- [41] The Symfony “cookbook”. Symfony [online] [Fecha de consulta: 15-04-2015]. Disponible en: <https://symfony.com/doc/2.3/cookbook/index.html>
- [42] El modelo y la persistencia de datos. Juandarodriguez [online] [Fecha de consulta: 19-05-2015]. Disponible en: <http://juandarodriguez.es/cursosf20/unidad7.html>
- [43] Doctrine. Doctrine [online] [Fecha de consulta: 19-05-2015]. Disponible en: <http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/working-with-associations.html>
- [44] Api Twig. Twig [online] [Fecha de consulta: 19-05-2015]. Disponible en: <http://gitnacho.github.io/Twig/api.html>
- [45] Plantillas Twig. Twig [online] [Fecha de consulta: 20-05-2015]. Disponible en: <http://www.acens.com/wp-content/images/2014/06/twig-plantillas-wp-acens.pdf>
- [46] Formularios. Librosweb [online] [Fecha de consulta: 25-05-2015]. Disponible en: http://librosweb.es/libro/symfony_2_3/capitulo_12.html
- [47] Temas para formularios. Librosweb [online] [Fecha de consulta: 25-05-2015]. Disponible en: http://librosweb.es/libro/symfony_2_x/capitulo_12/temas_para_formularios.html

- [48] Validación y Formularios. Juandarodriguez [online] [Fecha de consulta: 25-05-2015].
Disponible en: <http://juandarodriguez.es/cursosf20/unidad7.html>
- [49] Composer. Sitio oficial. Composer [online] [Fecha de consulta: 18-04-2015].
Disponible en: <https://getcomposer.org/doc/>
- [50] Archivo composer.json. Sitio oficial. Composer [online] [Fecha de consulta: 18-04-2015].
Disponible en: http://librosweb.es/libro/composer/capitulo_2/preparando_el_archivo_composerjson_del_proyecto.html
- [51] Instalación de composer. Symfony [online] [Fecha de consulta: 18-04-2015].
Disponible en: <http://symfony.es/documentacion/guia-de-instalacion-de-composer/>
- [52] Instalar bundles de terceros. Mycyberacademy [online] [Fecha de consulta: 18-04-2015].
Disponible en: <http://mycyberacademy.com/como-instalar-bundles-de-terceros-en-symfony-2-3-6/>
- [53] Documentación oficial FOSUserBundle. GitHub [online] [Fecha de consulta: 25-05-2015].
Disponible en: <https://github.com/FriendsOfSymfony/FOSUserBundle/blob/master/Resources/doc/index.md>
- [54] Sobrescribiendo controladores FOSUserBundle. GitHub [online] [Fecha de consulta: 25-05-2015].
Disponible en: https://github.com/FriendsOfSymfony/FOSUserBundle/blob/master/Resources/doc/overriding_controllers.md
- [55] Eventos FOSUserBundle. GitHub [online] [Fecha de consulta: 25-05-2015].
Disponible en: https://github.com/FriendsOfSymfony/FOSUserBundle/blob/master/Resources/doc/controller_events.md
- [56]. Gestionando la sesión. Librosweb [online] [Fecha de consulta: 25-05-2015].
Disponible en: http://librosweb.es/libro/symfony_2_x/capitulo_5/gestionando_la_sesion.html
- [57] Servicio Translator. Symfony [online] [Fecha de consulta: 18-04-2015].
Disponible en: <http://symfony.com/doc/current/book/translation.html>
- [58]. Gestionando configuración regional. Librosweb [online] [Fecha de consulta: 17-06-2015].
Disponible en: http://librosweb.es/libro/symfony_2_x/capitulo_15/gestionando_la_configuracion_regional_del_usuario.html
- [59]. Traducción básica. Librosweb [online] [Fecha de consulta: 17-06-2015].
Disponible en: http://librosweb.es/libro/symfony_2_x/capitulo_15/traduccion_basica.html



- [60]. Sistema de trazabilidad. Wikipedia [online] [Fecha de consulta: 22-06-2015]. Disponible en: https://es.wikipedia.org/wiki/Sistema_de_trazabilidad
- [61]. Crear nuevo droplet. DigitalOcean [online] [Fecha de consulta: 23-06-2015]. Disponible en: <https://www.digitalocean.com/community/tutorials/how-to-create-your-first-digitalocean-droplet-virtual-server>
- [62]. Crear claves SSH. DigitalOcean [online] [Fecha de consulta: 23-06-2015]. Disponible en: <https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2>
- [63]. Instalando phpmyadmin. DigitalOcean[online] [Fecha de consulta: 23-06-2015]. Disponible en: <https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-14-04>
- [64]. Tecnología VNC. ITE [online] [Fecha de consulta: 23-06-2015]. Disponible en: http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m5/servidor_vnc.html
- [65]. Panel Plesk. Odin [online] [Fecha de consulta: 23-06-2015]. Disponible en: <http://www.odin.com/products/plesk/>
- [66]. Validación de estándares. Vitaminaweb [online] [Fecha de consulta: 02-07-2015]. Disponible en: http://vitaminaweb.com/unicorn-validador-unificado-del-w3c_540