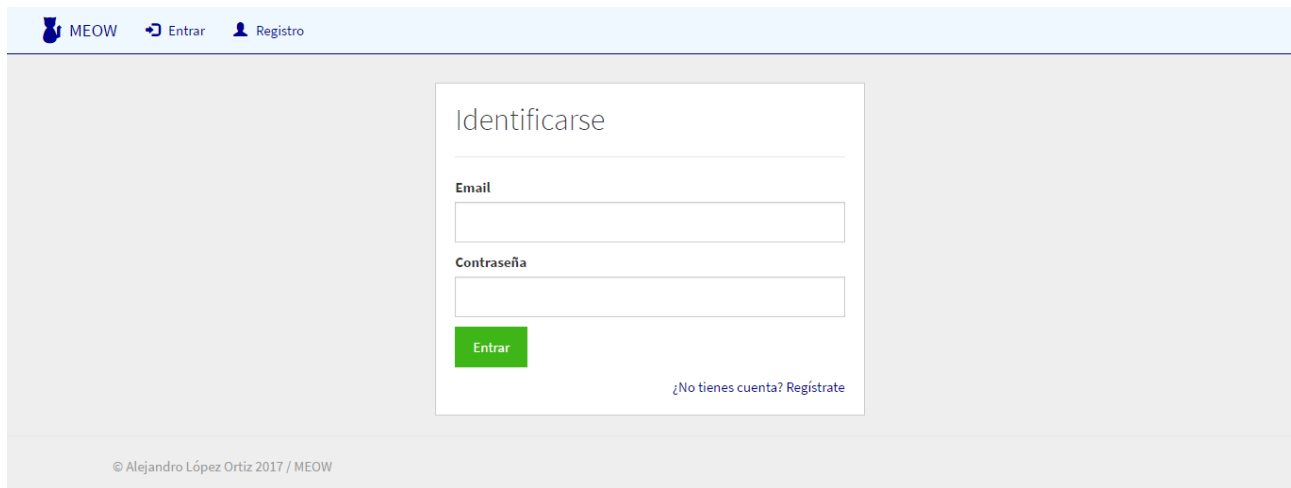


MEOW

Alejandro López Ortiz

1. MANUAL DE USUARIO

Vamos a poder registrarnos o loguearnos en un principio.



The desktop version of the MEOW login page features a light blue header with the MEOW logo, a login icon, and links for 'Entrar' and 'Registro'. The main content area is a light gray rectangle containing a white login form. The form is titled 'Identificarse' and includes an 'Email' field, a 'Contraseña' field, and a green 'Entrar' button. A link for '¿No tienes cuenta? Regístrate' is located at the bottom right of the form. The footer of the page contains the copyright notice '© Alejandro López Ortiz 2017 / MEOW'.

MEOW Entrar Registro

Identificarse

Email

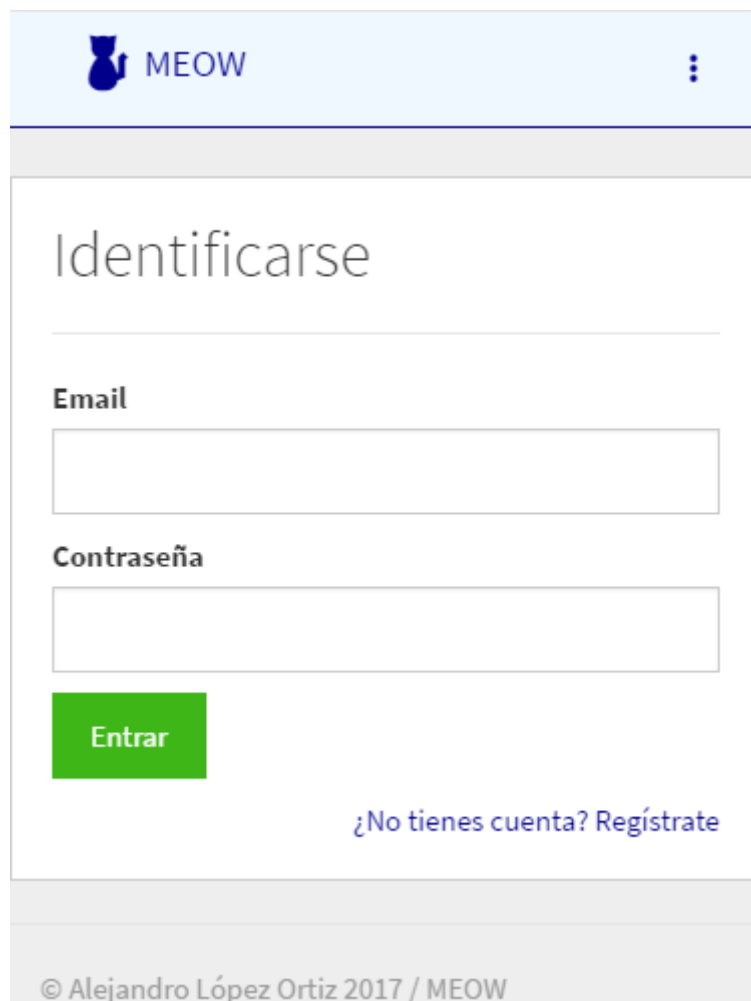
Contraseña

Entrar

[¿No tienes cuenta? Regístrate](#)

© Alejandro López Ortiz 2017 / MEOW

Versión móvil



The mobile version of the MEOW login page has a light blue header with the MEOW logo and a hamburger menu icon. The main content area is a white rectangle with a light gray border. It features the title 'Identificarse', an 'Email' field, a 'Contraseña' field, and a green 'Entrar' button. A link for '¿No tienes cuenta? Regístrate' is at the bottom right. The footer contains the copyright notice '© Alejandro López Ortiz 2017 / MEOW'.

MEOW

Identificarse

Email

Contraseña

Entrar

[¿No tienes cuenta? Regístrate](#)

© Alejandro López Ortiz 2017 / MEOW

En el proceso de registro podemos observar que tenemos el formulario a rellenar con todos los datos necesario para crear el usuario, en el campo de nick realizamos una petición ajax cada vez que salimos del campo para verificar si ese nick introducido esta disponible o no.

Al momento de registrarnos podemos iniciar sesión con nuestro usuario.

Registrarse

Nombre

Apellidos

Nick

Correo electrónico

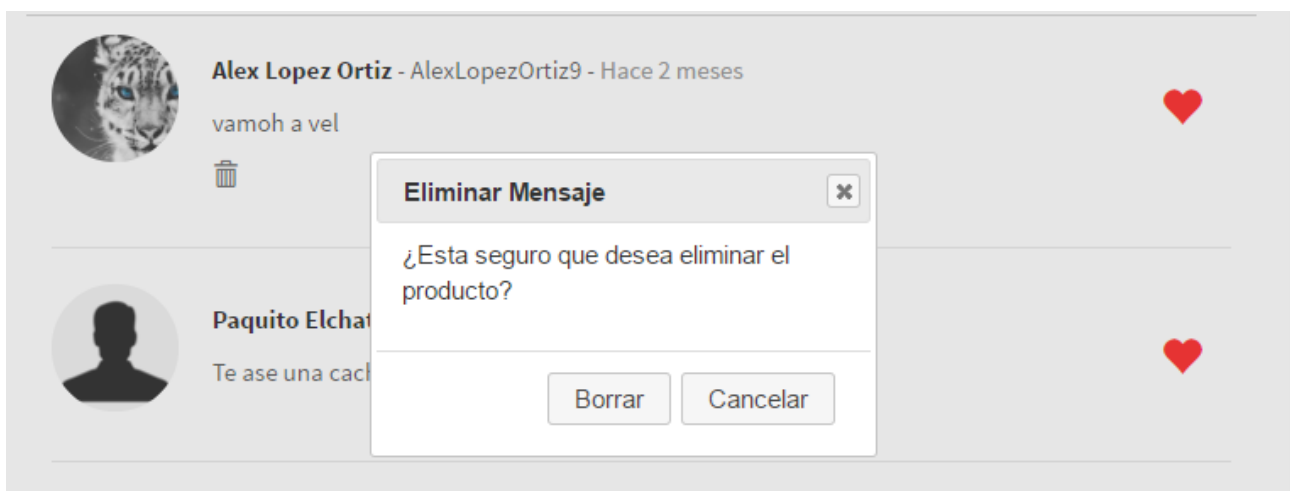
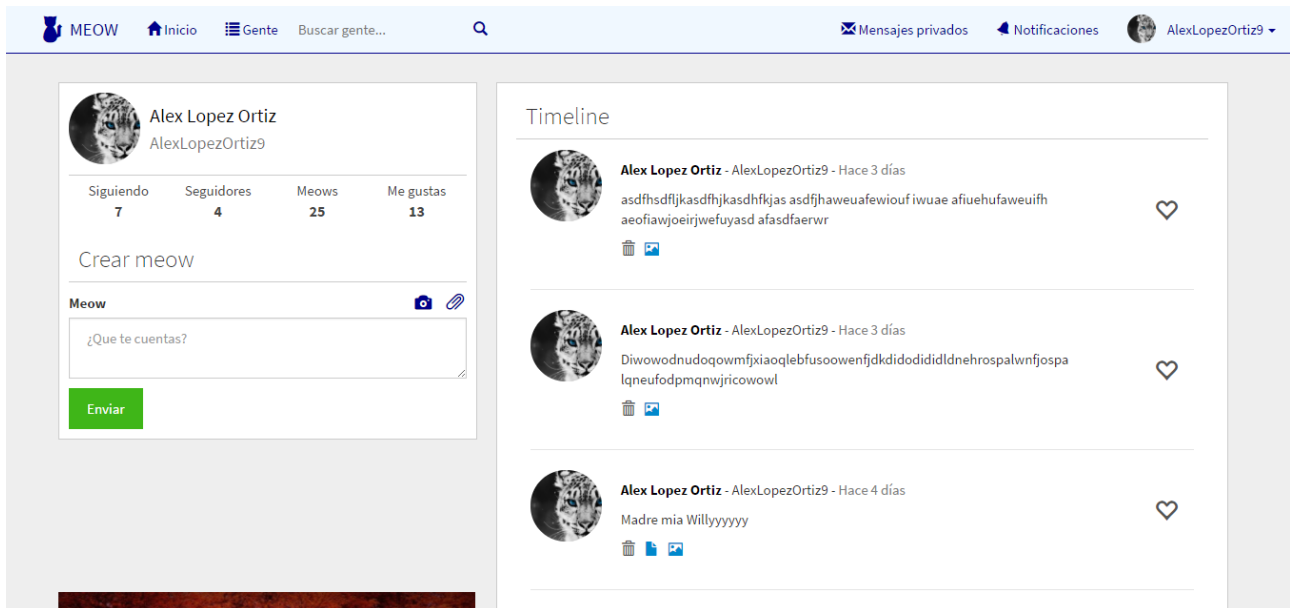
Contraseña

Repetir contraseña

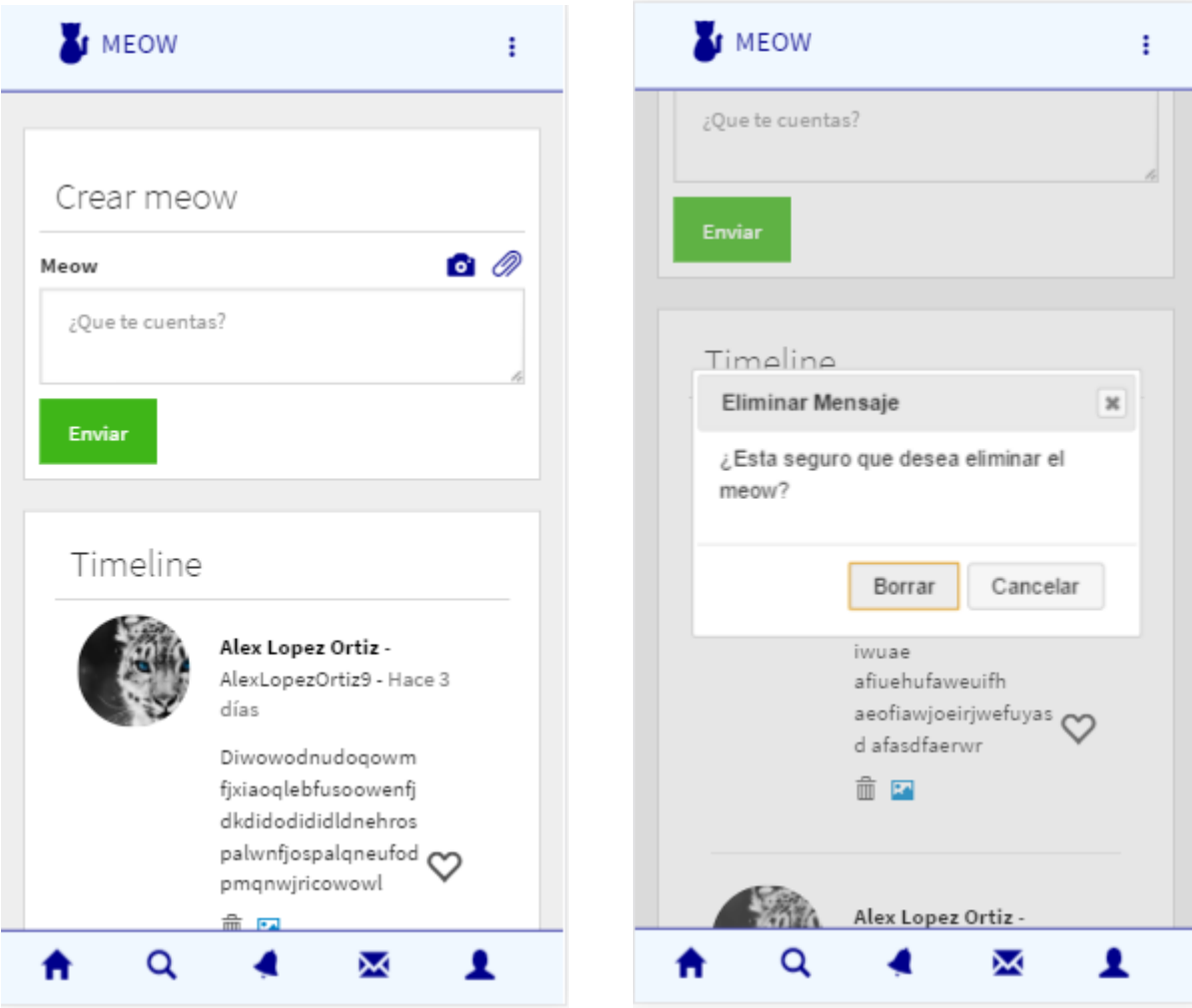
Registrarse

[¿Tienes cuenta? Inicia sesión](#)

Cuando iniciamos sesión con nuestro usuario nos carga la página principal en la que encontramos un timeline con las publicaciones de la gente a la que sigo y las nuestras propias, las cuales puedo borrar si quisiera realizando una petición ajax y se muestra la función de jQueryUI en la que nos sale el popup preguntándonos si queremos borrar dicha publicación o no.



Versión móvil:



También podemos buscar gente, podemos seguir y dejar de seguir a las personas que queramos, hay un buscador para buscar a personas registradas y ver su perfil sin necesidad de seguirlos aunque tenemos la opción de seguir o dejar de seguir, nos sale información de si nos sigue a nosotros o no y también podemos ver sus publicaciones y podemos marcarlas como me gusta o quitarlas de me gusta y también podemos ver la galería de imágenes que muestra todas las imágenes subidas del usuario pudiendo ver el meow y la fecha en que lo publicó, tenemos una tabla con las estadísticas del perfil que visitamos en la cual podemos ver el número de publicaciones, seguidores, personas que sigue y me gusta que tiene.

Gente

Total de personas: 8



Wisin Yandel - Wisin&Yandel

Esta es mi biografía

Dejar de seguir



Pepito Aasas - pepe2

Dejar de seguir



Paquito Elchatarra - PaKiTo

Seguir



Alex Lopez Ortiz

AlexLopezOrtiz9

Esta es mi biografía guapa....

GALLERY

Siguiendo

7

Seguidores

4

Publicaciones

24

Me gustas

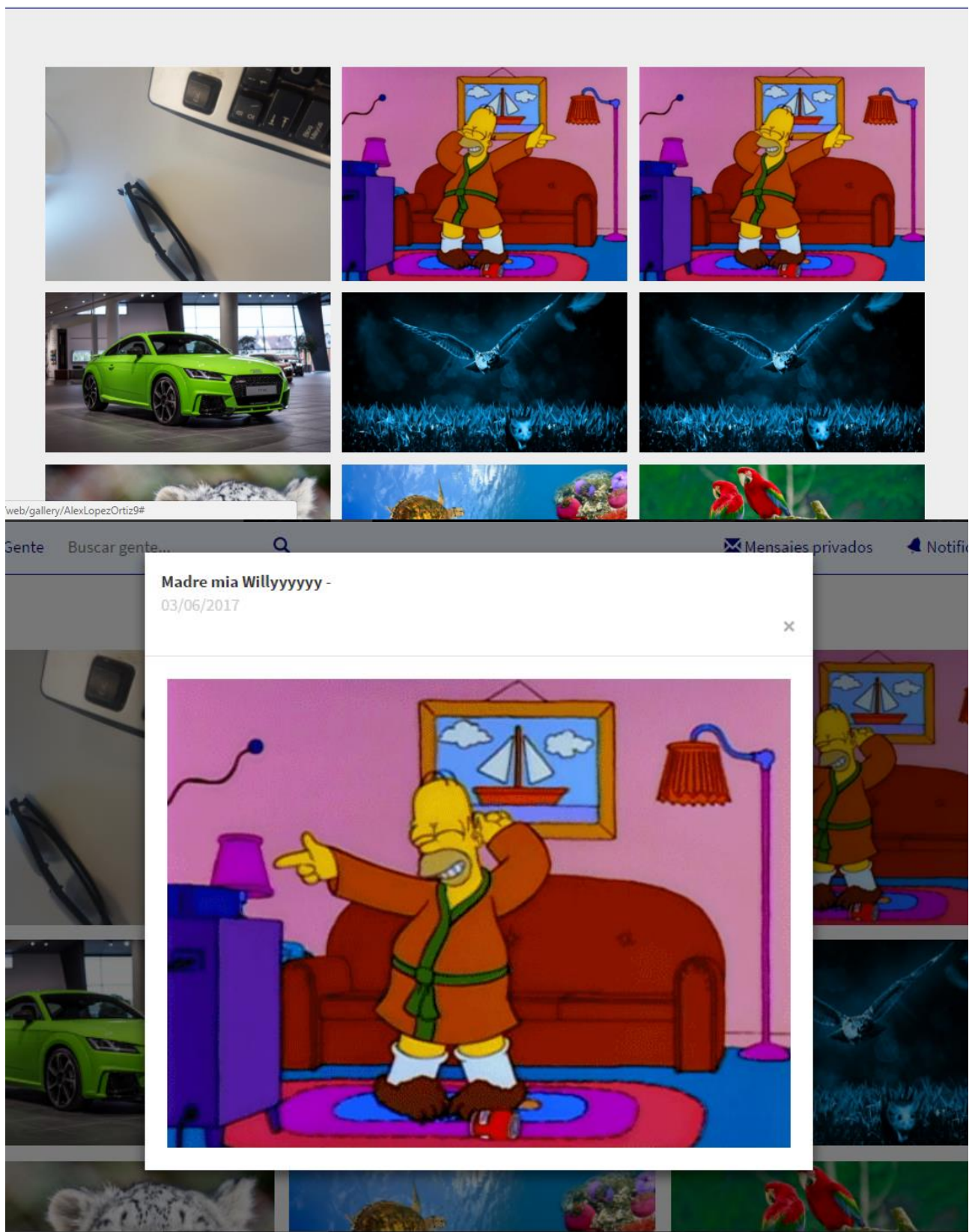
13




Alex Lopez Ortiz - AlexLopezOrtiz9 - Hace 3 días


Diwowodnudoqowmfjxiaoqlbfusoowenfjdkdidididldnehrospalwnfjospalqneufodpmqnrwircowowl





Versión móvil:

 MEOW








Alex Lopez Ortiz


AlexLopezOrtiz9


Esta es mi biografía guapa....


GALLERY






Siguiendo	Seguidores
7	4
Publicaciones	Me gustas
24	13



 MEOW







Madre mia Willyyyyyy -

03/06/2017

×







Desde el timeline de la pagina principal también podemos darle a me gusta a las publicaciones y automáticamente nos sale en nuestra lista de me gusta, podemos mostrar la imagen que se adjunta y también podemos visualizar y descargar el archivo que se adjunta en la publicación.



Alex Lopez Ortiz - AlexLopezOrtiz9 - Hace 2 meses

Hola nueva prueba con todo



Alex Lopez Ortiz - AlexLopezC


Hola nueva prueba con todo



Con este botón podremos visualizar y

descargar el archivo.

Podemos crear nuestras publicaciones en las que podemos subir también fotos y archivos y automáticamente sale en el timeline de todos los usuarios que me siguen y también en el mío. En la página principal tenemos nuestra tarjeta personal con acceso directo a nuestro perfil y se muestran todos nuestros datos (nombre, nick, biografía, foto perfil, publicaciones, estadísticas). También podemos acceder a la sección de mi perfil que sería un link igual al perfil de usuario.



Alex Lopez Ortiz
AlexLopezOrtiz9

Siguiendo7



Seguidores4

Meows24

Me gustas13



Crear meow



Meow





Enviar

Meow

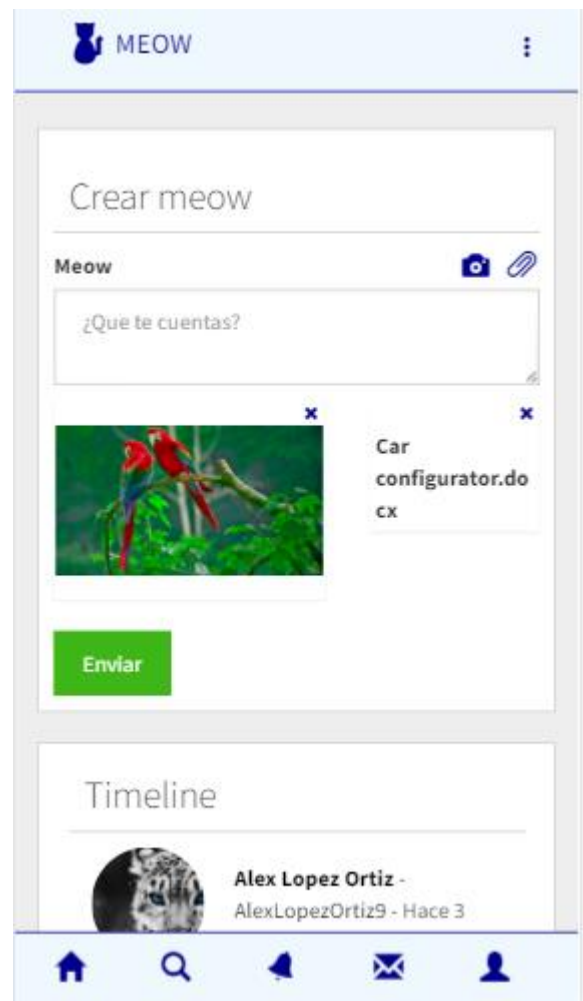
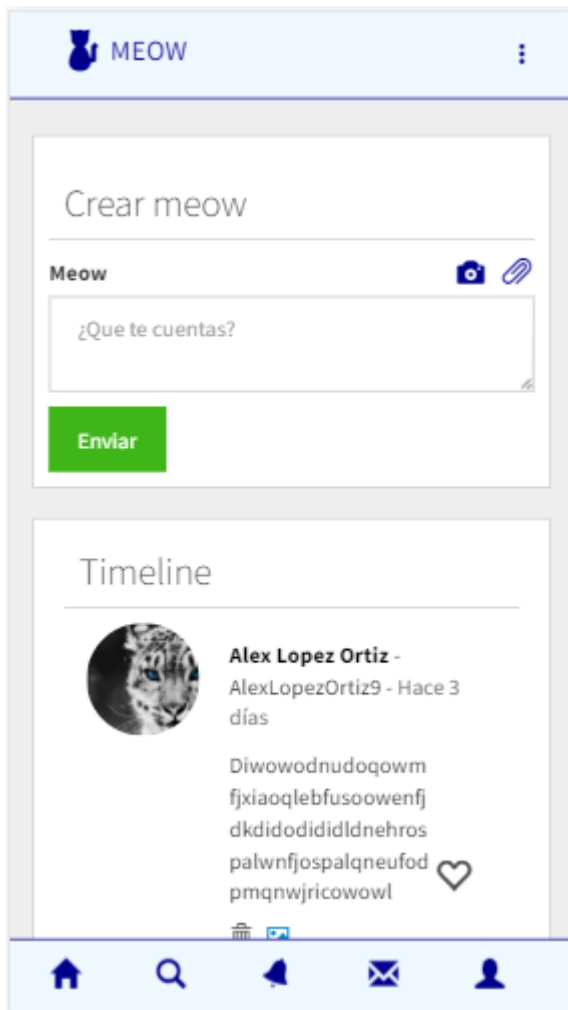






Car
configurator.docx

Enviar



Cuando hacemos scroll se ve que se hace una paginación infinita, al cargar una serie de paginas de esta paginación infinita nos sale el botón de ver más publicaciones lo que nos permite que sea mas amigable la navegación con nuestro timeline.

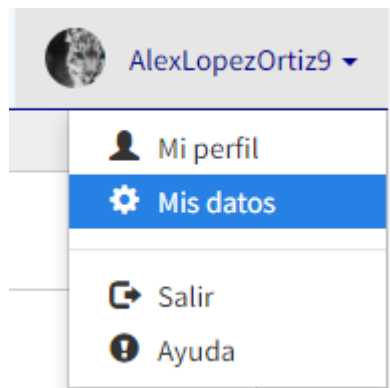


Wisin Yandel - Wisin&Yandel - Hace 2 meses

Audi la mejor marca de coches



En la sección de mis datos podemos editar nuestros datos e imagen de perfil.



Nombre

Alex

Apellidos

Lopez Ortiz

Nick

AlexLopezOrtiz9

Correo electrónico

alex@alex.com

Biografía

Esta es mi biografía guapa....

Foto

Seleccionar archivo Ningún archivo seleccionado

Guardar

[¿Quieres cambiar tu contraseña?](#)

Versión móvil:

 MEOW 

Nombre

Apellidos

Nick

Correo electrónico

Biografía

Foto

Ningún ar...eccionado

[¿Quieres cambiar tu contraseña?](#)

Podemos cerrar sesión y entrar sin problemas.

Tenemos un sistema de notificaciones que se activan cuando nos dan me gusta en nuestras publicaciones o nos siguen, esto se realiza con peticiones ajax.



Versión móvil:



Hay una mensajería privada que podemos enviar mensajes privados y recibirlos de la gente. Dentro de la mensajería tenemos una lista con los mensajes recibidos y otra con los mensajes enviados.

Mensajería privada

Enviar nuevo mensaje privado

Para:

Silvino Matamoros - SilvinoMatamoros

Mensaje

Imagen

Seleccionar archivo

Ningún archivo seleccionado

Archivo


Seleccionar archivo

Ningún archivo seleccionado

Enviar

Ver mensajes enviados


Mensajes recibidos



Wisin Yandel

Wisin&Yandel - Hace 2 meses


OLaaaaaaa



Wisin Yandel

Wisin&Yandel - Hace 2 meses


holaaaa de nuevo



Wisin Yandel

Wisin&Yandel - Hace 2 meses

QUE BIEN SE VE








Wisin Yandel

Wisin&Yandel - Hace 2 meses

QUE GUAY TU FOTO

Notificaciones

-  [Alex Lopez Ortiz](#) le ha gustado una de tus publicaciones
-  [Wisin Yandel](#) le ha gustado una de tus publicaciones
-  [Pepito Aasas](#) te esta siguiendo
-  [Pepito Aasas](#) le ha gustado una de tus publicaciones
-  [Pepito Aasas](#) le ha gustado una de tus publicaciones

[« Previous](#)

1	2	3	4
---	---	---	---

[Next »](#)

2. MANUAL TÉCNICO

2.1. Modelo de Datos

La base de datos consta de 6 tablas:

- Users
- Publications
- Likes
- Following
- Private_messages
- Notifications

Tabla	Acción
<input type="checkbox"/> following	★ Examinar Estructura Buscar Insertar Vaciar Eliminar
<input type="checkbox"/> likes	★ Examinar Estructura Buscar Insertar Vaciar Eliminar
<input type="checkbox"/> notifications	★ Examinar Estructura Buscar Insertar Vaciar Eliminar
<input type="checkbox"/> private_messages	★ Examinar Estructura Buscar Insertar Vaciar Eliminar
<input type="checkbox"/> publications	★ Examinar Estructura Buscar Insertar Vaciar Eliminar
<input type="checkbox"/> users	★ Examinar Estructura Buscar Insertar Vaciar Eliminar
6 tablas	Número de filas



En la tabla Users nos encontramos con las siguientes columnas:

- Id: Identificación del usuario
- Role: Tipo de usuario
- Email: Email para el registro e inicio de sesión del usuario
- Name: Nombre elegido por el usuario
- Surname: Apellido del usuario
- Password: Contraseña elegida por el usuario
- Nick: Nick elegido por el usuario para su cuenta
- Bio: Biografía en la que el usuario pondrá los datos que quiera
- Active: Para ver si está activo o no
- Image: Imagen de perfil del usuario

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo
<input type="checkbox"/>	1	id 	int(255)			No
<input type="checkbox"/>	2	role	varchar(20)	utf8_bin		Sí
<input type="checkbox"/>	3	email 	varchar(255)	utf8_bin		Sí
<input type="checkbox"/>	4	name	varchar(255)	utf8_bin		Sí
<input type="checkbox"/>	5	surname	varchar(255)	utf8_bin		Sí
<input type="checkbox"/>	6	password	varchar(255)	utf8_bin		Sí
<input type="checkbox"/>	7	nick 	varchar(50)	utf8_bin		Sí
<input type="checkbox"/>	8	bio	varchar(255)	utf8_bin		Sí
<input type="checkbox"/>	9	active	varchar(2)	utf8_bin		Sí
<input type="checkbox"/>	10	image	varchar(255)	utf8_bin		Sí

En la tabla Publications nos encontramos con las siguientes columnas:

- Id: Identificación de la publicación
- User_id: Identificación del usuario que ha creado la publicación
- Text: Texto introducido en la publicación
- Document: Documento subido en la publicación
- Image: Imagen subida en la publicación
- Status:
- Created_at: Para saber la hora de publicación

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo
<input type="checkbox"/>	1	id 	int(255)			No
<input type="checkbox"/>	2	user_id 	int(255)			Sí
<input type="checkbox"/>	3	text	mediumtext	utf8_bin		Sí
<input type="checkbox"/>	4	document	varchar(100)	utf8_bin		Sí
<input type="checkbox"/>	5	image	varchar(255)	utf8_bin		Sí
<input type="checkbox"/>	6	status	varchar(30)	utf8_bin		Sí
<input type="checkbox"/>	7	created_at	datetime			Sí

En la tabla Likes nos encontramos las siguientes columnas:

- Id: Identificador del like
- User: Id del usuario que le da like a la publicación
- Publication: Id de la publicación a la que le han dado like

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo
<input type="checkbox"/> 1	id 🔑	int(255)			No
<input type="checkbox"/> 2	user 🔑	int(255)			Sí
<input type="checkbox"/> 3	publication 🔑	int(255)			Sí

En la tabla Following nos encontramos las siguientes columnas:

- Id: Identificador de following
- User: Identificador del usuario que sigue a alguien
- Followed: Id del usuario que han seguido

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo
1	id 🔑	int(255)			No
2	user 🔑	int(255)			Sí
3	followed 🔑	int(255)			Sí

En la tabla Private_messages nos encontramos con las siguientes columnas:

- Id: Identificador del mensaje
- Message: El texto que se envía en el mensaje
- Emitter: Id del usuario que envía el mensaje
- Receiver: Id del usuario que recibe el mensaje
- File: Archivo que enviamos en el mensaje
- Image: Imagen que enviamos en el mensaje
- Readed: Para marcar el mensaje como leído
- Created_at: Para el momento del envío del mensaje

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo
1	id 🔑	int(255)			No
2	message	longtext	utf8_bin		Sí
3	emitter 🔑	int(255)			Sí
4	receiver 🔑	int(255)			Sí
5	file	varchar(255)	utf8_bin		Sí
6	image	varchar(255)	utf8_bin		Sí
7	readed	varchar(3)	utf8_bin		Sí
8	created_at	datetime			Sí

En la tabla Notifications nos encontramos con las siguientes columnas:

- Id: Id de la notificación
- User_id: Id del usuario al que va dirigida la notificación
- Type: Para saber si es tipo follow o like
- Type_id: Id del usuario que realiza la notificación
- Readed: Para marcar como leída la notificación
- Created_at: Para saber el momento de la notificación
- Extra: Id de la publicación que gusta en caso de que sea notificación por publicación sino el valor por defecto es null

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo
1	id 🔑	int(255)			No
2	user_id 🔑	int(255)			Sí
3	type	varchar(255)	utf8_bin		Sí
4	type_id	int(255)			Sí
5	readed	varchar(3)	utf8_bin		Sí
6	created_at	datetime			Sí
7	extra	varchar(100)	utf8_bin		Sí

Creación del Proyecto Symfony 3

En primer lugar instalé el servidor de aplicaciones web Wamp64. Después instalé Composer que es un gestor de dependencias de php con el que podremos descargar paquetes y actualizarlos.

```
C:\Users\alex1>composer -v

Composer version 1.3.2 2017-01-27 18:23:41

Usage:
  command [options] [arguments]
```

A continuación instalé el emulador de la consola de Linux Cygwin.



El IDE que utilicé fue el Netbeans.

Después de haber preparado el entorno de desarrollo, comencé con la instalación de Symfony3. Utilizando la consola Cywing para crear el proyecto dentro de wamp64.

```
alex1@LAPTOP-3J848S49 /cygdrive/c/wamp64/www
$ composer create-project symfony-standard-edition social-network/ "3.1.*"
```

Utilizo un bundle de Symfony para tener las paginaciones en los listados de cosas que va a tener la web. En el archivo de composer.json ponemos:

```
{
  "require": {
    "php": ">=5.5.9",
    "symfony/symfony": "3.1.*",
    "doctrine/orm": "^2.5",
    "doctrine/doctrine-bundle": "^1.6",
    "doctrine/doctrine-cache-bundle": "^1.2",
    "symfony/swiftmailer-bundle": "^2.3.10",
    "symfony/monolog-bundle": "^3.0.2",
    "symfony/polyfill-apcu": "^1.0",
    "sensio/distribution-bundle": "^5.0",
    "sensio/framework-extra-bundle": "^3.0.2",
    "incenteev/composer-parameter-handler": "^2.0",
    "knplabs/knp-paginator-bundle": "2.5.*"
  },
}
```

Y en la consola se pone dentro del directorio de nuestro proyecto:

```
alex1@LAPTOP-3J848S49 /cygdrive/c/wamp64/www/social-network
$ composer update
```

Así instalo los nuevos paquetes, las actualizaciones y el nuevo paquete de la paginación.

La aplicación está dividida en dos Bundles. Uno será para guardar las entidades y modelos y trabajos con la base de datos y el otro para el resto de la aplicación.

El que se encarga del resto de la aplicación esta por defecto creado en el proyecto Symfony, pero el otro Bundle lo vamos a crear nosotros con la consola dentro de la carpeta de nuestro proyecto:

```
$ php bin/console generate:bundle

Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [no]: no
Your application code must be written in bundles. This command helps
you generate them easily.

Give your bundle a descriptive name, like BlogBundle.
Bundle name: BackendBundle

Bundles are usually generated into the src/ directory. Unless you're
doing something custom, hit enter to keep this default!

Target Directory [src/]:

What format do you want to use for your generated configuration?
Configuration format (annotation, yml, xml, php) [annotation]: yml

Bundle generation

> Generating a sample bundle skeleton into C:\wamp\www\curso-social-network\app\../src/BackendBundle OK!
```

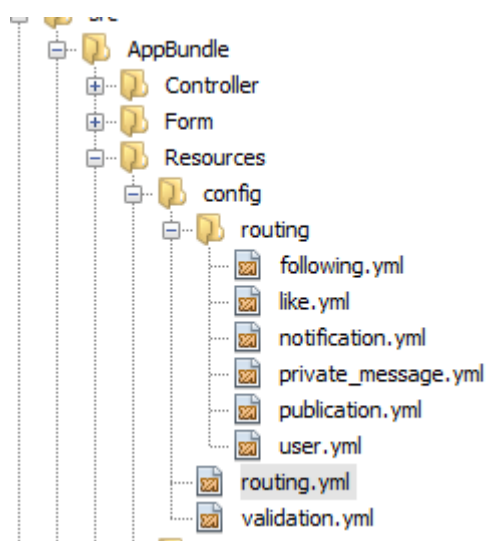
Ahora al crear el BackendBundle voy a pasar nuestras tablas de la base de datos a código doctrine. Importo la base de datos a yml:

```
$ php bin/console doctrine:mapping:import BackendBundle yml
```

Ahora genero las entidades para poder utilizarlas en el código:

```
$ php bin/console doctrine:generate:entities BackendBundle
```

Creo los ficheros de rutas que voy a necesitar para cargar las diferentes rutas de los controladores.



Creo una ruta por defecto:

```
app_homepage:
  path: /
  defaults: { _controller: AppBundle:User:login }
```

Con esto cargo directamente el fichero de rutas.

```
app:
  resource: "@AppBundle/Resources/config/routing.yml"
  prefix: /
```

Empiezo con la maquetación de la página principal utilizando el motor de plantillas TWIG.

Antes de empezar con el login y el registro, indico en que entidad se va a utilizar este protocolo de cifrado y después indico el algoritmo de cifrado que es bcrypt y después, cuántas veces se vuelve a cifrar la contraseña ya cifrada

```
security:
#Indicamos en que entidad se va a utilizar
#Y despues inidicamos el algoritmo de cifrado
encoders:
  BackendBundle\Entity\User:
    algorithm: bcrypt
    cost: 4
```

Esto nos va generar una clase con un formulario basada en la entidad especificada.

```
$ php bin/console doctrine:generate:form BackendBundle:User
```

Cuando se crea el formulario realizo modificaciones para hacerlo útil:

```
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('name', TextType::class, array(
            'label' => 'Nombre',
            'required' => 'required',
            'attr' => array(
                'class' => 'form-name form-control'
            )
        ))
        ->add('surname', TextType::class, array(
            'label' => 'Apellidos',
            'required' => 'required',
            'attr' => array(
                'class' => 'form-surname form-control'
            )
        ))
        ->add('nick', TextType::class, array(
            'label' => 'Nick',
            'required' => 'required',
            'attr' => array(
                'class' => 'form-nick form-control nick-input'
            )
        ))
        ->add('email', EmailType::class, array(
            'label' => 'Correo electrónico',
            'required' => 'required',
            'attr' => array(
                'class' => 'form-email form-control'
            )
        ))
    }
```



```

->add('password', RepeatedType::class, array(
    'type' => PasswordType::class,
    'first_options' => array('label' => 'Contraseña',
        'required' => 'required',
        'attr' => array(
            'class' => 'form-password form-control'
        )),
    'second_options' => array('label' => 'Repetir contraseña',
        'required' => 'required',
        'attr' => array(
            'class' => 'form-password form-control'
        ))
))
->add('Registrarse', SubmitType::class, array(
    'attr' => array(
        'class' => "form-submit btn btn-success"
    )
))

```

Lo muestro en la vista de ésta manera:

```

<div class="col-lg-6 box-form">

    <h2>Registrarse</h2>
    <hr>

    {{form_start(form, {'action':' ', 'method':'POST'})}}
    {{form_errors(form)}}
    {{form_end(form)}}

</div>

```

Utilizo las sesiones para hacer la función de los mensajes flush que es una pequeña alerta de bootstrap y que cuando volvemos a actualizar la página ese mensaje flush ya no esté. Así que en UserController creo la sesión.

Ahora en el registro realizo la comprobación por Ajax del nickname.

```

$(document).ready(function(){

    $(".nick-input").blur(function() {
        var nick = this.value;

        $.ajax({
            url: URL+"/nick-test", //Esta va a ser la ruta en sí de esa acción
            data: {nick: nick}, //Le pasamos como data nick que es el parámetro
            type: "POST",
            success: function(response){
                if(response == "used"){
                    $(".nick-input").css("border", "1px solid red");
                    $(".form-submit").attr("disabled", true);
                } else {
                    $(".nick-input").css("border", "1px solid green");
                    $(".form-submit").attr("disabled", false);
                }
            }
        });
    });

});

```

Para realizar el sistema de paginación, instancio el knp_paginator en AppKernel que es donde están todos los bundles que tiene la app.

```
$bundles = [
    new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
    new Symfony\Bundle\SecurityBundle\SecurityBundle(),
    new Symfony\Bundle\TwigBundle\TwigBundle(),
    new Symfony\Bundle\MonologBundle\MonologBundle(),
    new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
    new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),
    new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
    new \Knp\Bundle\PaginatorBundle\KnpPaginatorBundle(),
    new AppBundle\AppBundle(),
    new BackendBundle\BackendBundle(),
];
```

Configuro la paginación en el config.yml

```
#KnpPagination
knp_paginator:
    #Rango de pagina
    page_range: 5
    #Opciones por defecto
    default_options:
        page_name: page
        #sort_field_miname -> nos va a permitir tener el nombre del parametro en la url
        sort_field_name: sort
        sort_direction_name: direction
        distinct: true
    template:
        pagination: AppBundle:Layouts:custom_pagination.html.twig
        sortable: KnpPaginatorBundle:Pagination:sortable_link.html.twig
```

Después de la sección gente hago la parte del buscador en la que creo el formulario con Symfony y también creo el Action del buscador.

Para terminar con la sección de gente realizo el js de users en el que utilizamos el plugin de Ajax (ias).

```

var ias = JQuery.ias({
    container: ".box-users", //El contenedor que tiene todos los usuarios que vamos a paginar
    item: '.user-item', //El item a paginar es el que tiene la clase .user-item
    pagination: '.pagination', //Los controles de paginacion van a estar dentro de la clase .pagination
    next: '.pagination .next_link', //Va a hacer una peticion a ajax para cargar la siguiente pag utilizando
    triggerPageThreshold: 5 //Cada cuantos elementos va a tener que lanzar la peticion ajax
});

//Lo que va a mostrar tras pasar 3 bloques de carga(paginaciones) y nos saldra el boton de ver mas pe
ias.extension(new IASTriggerExtension({
    text: 'Ver más personas',
    offset: 3
}));
//Con esto va a cargar la imagen de carga
ias.extension(new IASSpinnerExtension({
    src: URL+'../assets/images/ajax-loader.gif'
}));
//El boton que saldra cuando ya no existan mas elementos a mostrar
ias.extension(new IASNoneLeftExtension({
    text: 'No hay mas personas'
}));

ias.on('ready', function(event){
    followButtons();
});

ias.on('rendered', function(event){
    followButtons();
});

```

El sistema de Following en el cual podemos seguir y dejar de seguir a otros usuarios, realizando su Action como en las demás ocasiones y creamos una extensión de Twig.

```

public function getFilters() {
    return array(
        //Esto nos va a permitir definir un filtro de twig, el filtro se va a llamar following
        //y le pasamos un array diciendo que esta en esta clase y que metodo va a ser que lo llamamos
        // followingFilter y ese es el metodo que se va a llamar cuando llamamos al filtro dentro de la vista
        new \Twig_SimpleFilter('following', array($this, 'followingFilter'))
    );
}

public function followingFilter($user, $followed){ //Esta funcion recibe como parametros el $user osea
//el usuario que esta identificado y el $followed que es el usuario que supuestamente estoy siguiendo
    $following_repo = $this->doctrine->getRepository('BackendBundle:Following');
    ///Aqui comprobamos si seguimos o no al $followed
    //Ahora pedimos que nos saque el registro cuyo user sea con el que estoy logueado y
    //cuyo followed (si es que existe) sea el que nos esta llegando por parametro
    $user_following = $following_repo->findOneBy(array(
        "user" => $user,
        "followed" => $followed
    ));

    if(!empty($user_following) && is_object($user_following)){
        $result = true;
    }else{
        $result = false;
    }

    return $result;
    //Con todo esto hemos hecho la comprobacion de si seguimos a un usuario o no
}

```

Luego su funcionalidad en Ajax y el cambio de botones con jquery.

```
function followButtons(){
    $(".btn-follow").unbind("click").click(function(){
        $(this).addClass("hidden");
        $(this).parent().find(".btn-unfollow").removeClass("hidden");

        $.ajax({
            url: URL+"/follow",
            type: "POST",
            data: {followed: $(this).attr("data-followed")}, //Esto saca
            success: function(response){
                console.log(response);
            }
        });
    });

    $(".btn-unfollow").unbind("click").click(function(){
        $(this).addClass("hidden");
        $(this).parent().find(".btn-follow").removeClass("hidden");

        $.ajax({
            url: URL+"/unfollow",
            type: "POST",
            data: {followed: $(this).attr("data-followed")}, //Esto saca
            success: function(response){
                console.log(response);
            }
        });
    });
}
```

En las publicaciones podemos subir archivos ya sean imágenes o documentos y tienen restricciones de que tipo se acepta y cual no.

```
if (!empty($file) && $file != null) {
    $ext = $file->guessExtension(); //Capturamos la extension del fichero con esta funcion
    if ($ext == 'jpg' || $ext == 'jpeg' || $ext == 'png' || $ext == 'gif') {
        $file_name = $user->getId() . time() . "." . $ext;
        $file->move("uploads/publications/images", $file_name);

        $publication->setImage($file_name);
    } else {
        $publication->setImage(null);
    }
} else {
    $publication->setImage(null);
}

//upload document
$doc = $form['document']->getData();

if (!empty($doc) && $doc != null) {
    $ext = $doc->guessExtension(); //Capturamos la extension del fichero con esta funcion
    if ($ext == 'pdf' || $ext == 'doc' || $ext == 'docx') {
        $file_name = $user->getId() . time() . "." . $ext;
        $doc->move("uploads/publications/documents", $file_name);

        $publication->setDocument($file_name);
    } else {
        $publication->setDocument(null);
    }
} else {
    $publication->setDocument(null);
}
```

Cuando seleccionamos la imagen a subir o el documento mostramos una previsualización de los elegidos y lo realizo mediante jquery con una función que realiza la lectura del documento y luego la muestro.

```
if(document.getElementById('backendbundle_user_image') != null){
    document.getElementById('backendbundle_user_image').addEventListener('change', archivo, false);

    $('.boton-eliminar-archivo').on('click', function(e){
        var $el = $('#backendbundle_user_image');
        $el.wrap('<form>').closest('form').get(0).reset();
        $el.unwrap();
        $(".nombre-image").css("display", "none");
    });

    var insertar_en_doc = document.querySelector(".nombre-document");
    var archivo_doc = document.createElement("div");
    $('#backendbundle_user_document').on('change', function(e){
        file_in = document.querySelector("#backendbundle_user_document");
        var files = e.target.files;
        for(var i=0;f=files[i];i++){
            $(".nombre-document").css("display", "inline-block");
            archivo_doc.style.display= "inline-block";
            archivo_doc.style.float = "left";
            archivo_doc.style.margin = "3px";
            archivo_doc.innerHTML = f.name;
            insertar_en_doc.appendChild(archivo_doc);
        }
        $('.boton-eliminar-archivo-doc').on('click', function(e){
            var $el = $('#backendbundle_user_document');
            $el.wrap('<form>').closest('form').get(0).reset();
            $el.unwrap();
            $(".nombre-document").css("display", "none");
        });
    });
}
```

En el timeline cuando la publicación que hay tiene una imagen realizamos una función con JQuery en la que habilitamos dicha foto. Y si la publicación es nuestra saldrá el icono de la basura con el cual borramos dicha modificación con Ajax abriendo primero una ventana de JQuery UI.

```
$(".btn-img").unbind("click").click(function(){
    $(this).parent().find('.pub-image').fadeToggle();
});

$(".btn-delete-pub").unbind("click").click(function(){
    /*jQueryUI*/
    $( "#dialogoborrar" ).dialog({
        autoOpen: false,
        resizable: false,
        modal: true,
        buttons: {
            //BOTON DE BORRAR
            "Borrar": function() {
                //Ajax con post
                $.get(URL + '/publication/remove/' + id, function(data, status){
                    $("#" + id).parent().parent().fadeOut('slow');
                }); //post
                //Cerrar la ventana de dialogo
                $(this).dialog("close");
            },
            "Cancelar": function() {
                //Cerrar la ventana de dialogo
                $(this).dialog("close");
            }
        }
    }); //buttons
});
$(document).on("click", ".btn-delete-pub", function(){
    id = $(this).attr("data-id");
    $( "#dialogoborrar" ).dialog("open");
});
```

El sistema de me gustas también lo hago con Ajax y realizo el cambio de botón.

```
 $('[data-toggle="tooltip"]').tooltip();//Asi hacemos el bocadillito de me gusta

$(".btn-like").unbind('click').click(function(){
    $(this).addClass('hidden');
    $(this).parent().find('.btn-unlike').removeClass('hidden');

    $.ajax({
        url: URL+'/like/'+$(this).attr('data-id'),
        type: 'GET',
        success: function(response){
            console.log(response);
        }
    });
});

$(".btn-unlike").unbind('click').click(function(){
    $(this).addClass('hidden');
    $(this).parent().find('.btn-like').removeClass('hidden');

    $.ajax({
        url: URL+'/unlike/'+$(this).attr('data-id'),
        type: 'GET',
        success: function(response){
            console.log(response);
        }
    });
});
```

En el home presentamos unas estadísticas con nuestros datos de seguimientos publicaciones y me gustas. Para esto se necesita otra extensión de twig que se llama userStatsFilter.

```
public function userStatsFilter($user){
    $following_repo = $this->doctrine->getRepository('BackendBundle:Following');
    $publication_repo = $this->doctrine->getRepository('BackendBundle:Publication');
    $like_repo = $this->doctrine->getRepository('BackendBundle:Like');

    //Sacamos todos los registros de la tabla following cuyo usuario es igual al usuario que
    //le pasamos por parametro. Osea cuantos usuarios seguimos
    $user_following = $following_repo->findBy(array('user' => $user));
    //Sacamos nuestros seguidores
    $user_followers = $following_repo->findBy(array('followed' => $user));
    //Sacamos todas las publicaciones del usuario que indicamos
    $user_publications = $publication_repo->findBy(array('user' => $user));
    //Todas las publicaciones que nos gustan
    $user_likes = $like_repo->findBy(array('user' => $user));

    $result = array(
        'following' => count($user_following),
        'followers' => count($user_followers),
        'publications' => count($user_publications),
        'likes' => count($user_likes)
    );//Con esto nos devuelve un array con toda la informacion que necesitamos de
    //marcadores y estadisticas

    return $result;
}
```

Para la sección de notificaciones realizo un servicio que puedo utilizar en todos los controladores de la app.

```
public function __construct($manager){
    $this->manager = $manager;
} //Con esto hemos conseguido tener el entity manager de doctrine aqui para poder
//trabajar con entidades y repositorios.

public function set($user, $type, $typeId, $extra = null){
    //Recibe $user que es el user para el que va la notificacion
    //$type es para el tipo de notificacion que es
    //$typeId se recibe en el caso de que tengamos que guardar un id de un
    //registro en concreto para luego sacar el usuario que le a dado like

    $em = $this->manager;

    $notification = new Notification();
    $notification->setUser($user); //guardamos el usuario que nos llega por parametro
    $notification->setType($type); //guardamos el tipo de notificacion que nos va a llegar
    $notification->setTypeId($typeId); //guardamos lo que nos llega por parametro
    $notification->setReaded(0); //ponemos que no esta leído por defecto
    $notification->setCreatedAt(new \DateTime("now")); //Le indicamos la fecha
    //en formato DateTime
    $notification->setExtra($extra);

    $em->persist($notification);
    $flush = $em->flush();

    if($flush == null){
        $status = true;
    }else{
        $status = false;
    }
}
```


Hago lo mismo con el sistema de notificaciones por Ajax que realizo la comprobación de que si dichas notificaciones están leídas o no y muestro el label con las notificaciones.

```
notifications();

setInterval(function(){
    notifications();
}, 5000);

});

function notifications(){
    $.ajax({
        url: URL+'/notifications/get',
        type: 'GET',
        success: function(response){
            $(".label-notifications").html(response);
            $(".label-notificaciones").html(response);

            if(response == 0){
                $(".label-notifications").addClass("hidden");
                $(".label-notificaciones").addClass("hidden");
            }else{
                $(".label-notifications").removeClass("hidden");
                $(".label-notificaciones").removeClass("hidden");
                if(($window).width() <= 400){
                    $(".label-notificaciones").parent().parent().css("margin-right", "1%");
                } else if(($window).width() > 400 && (($window).width() <= 489)){
                    $(".label-notificaciones").parent().parent().css("margin-right", "3%");
                } else if(($window).width() > 489 && (($window).width() <= 600)){
                    $(".label-notificaciones").parent().parent().css("margin-right", "5%");
                } else if(($window).width() > 600){
                    $(".label-notificaciones").parent().parent().css("margin-right", "9%");
                }
            }
        }
    });
}
```

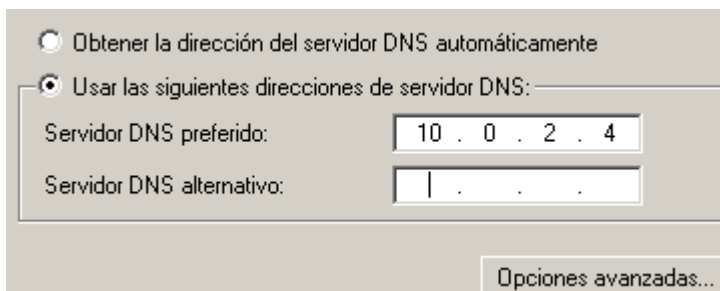
SERVIDOR WEB MÁQUINA DEBIAN

Editar el fichero de interfaces para colocar una ip fija.

```
root@debian8:~# nano /etc/network/interfaces_
```

```
# The primary network interface
allow-hotplug eth0
#iface eth0 inet dhcp
iface eth0 inet static
    address 10.0.2.4
    netmask 255.255.255.0
    network 10.0.2.0
    broadcast 10.0.2.255
    gateway 10.0.2.1
```


Después hay que ir a la máquina cliente en la que habrá que configurar como dirección de DNS la IP de la máquina servidor Debian.



Obtener la dirección del servidor DNS automáticamente
☒ Usar las siguientes direcciones de servidor DNS:

Servidor DNS preferido: 10 . 0 . 2 . 4
Servidor DNS alternativo: | . . .

Opciones avanzadas...

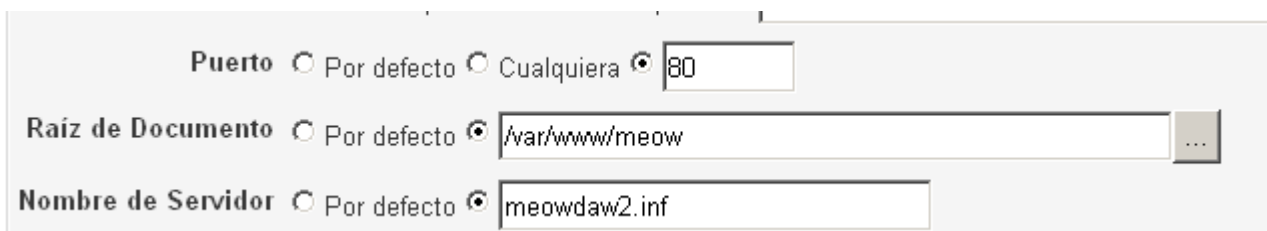
Entrar en el Webmin y crear una nueva zona maestra en la sección de Servidor DNS. Y crear la dirección a la que yo he llamado meowdaw2.inf

Nombre
<input type="checkbox"/> server.inf.
<input type="checkbox"/> meowdaw2.inf.

Comprobamos en la máquina cliente que nos hace ping a la dirección creada.

```
Haciendo ping a meowdaw2.inf [10.0.2.254] con 32 bytes de datos:
Respuesta desde 10.0.2.254: bytes=32 tiempo<1m TTL=64
Respuesta desde 10.0.2.254: bytes=32 tiempo<1m TTL=64
Respuesta desde 10.0.2.254: bytes=32 tiempo<1m TTL=64
Respuesta desde 10.0.2.254: bytes=32 tiempo<1m TTL=64
```

Crear el nuevo servidor web apache

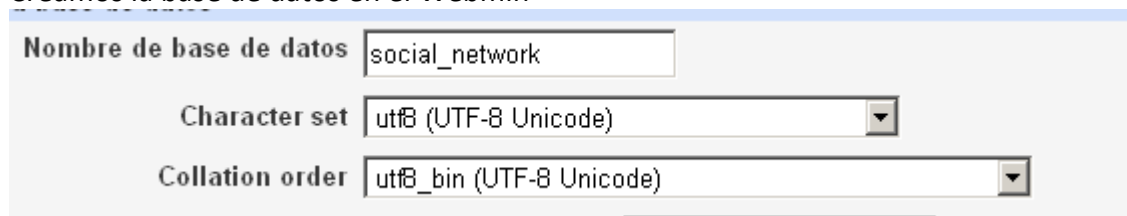


Puerto ☐ Por defecto ☐ Cualquiera ☒ 80

Raíz de Documento ☐ Por defecto ☒ /var/www/meow ...

Nombre de Servidor ☐ Por defecto ☒ meowdaw2.inf

Creamos la base de datos en el Webmin



Nombre de base de datos social_network

Character set utf8 (UTF-8 Unicode)

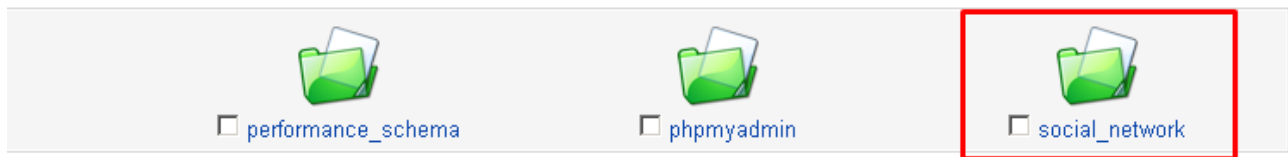
Collation order utf8_bin (UTF-8 Unicode)

Servidor de Base de Datos MySQL


Versión 5.5.54 de MySQL

Buscar Documentos..

e datos



e datos

Entramos al phpMyAdmin  /10.0.2.254/phpmyadmin/ e importamos la base de datos a social_network.

Extraemos la aplicación en el directorio del servidor virtual.



/var/www/meow/

Pages: 1

Total: 9 files and 9 directories

	Name	Actions	Size
<input type="checkbox"/>	app	a	4 kB
<input type="checkbox"/>	bin	a	4 kB
<input type="checkbox"/>	cgi-bin	a	4 kB
<input type="checkbox"/>	nbproject	a	4 kB
<input type="checkbox"/>	src	a	4 kB
<input type="checkbox"/>	tests	a	4 kB
<input type="checkbox"/>	var	a	4 kB
<input type="checkbox"/>	vendor	a	4 kB
<input type="checkbox"/>	web	a	4 kB
<input type="checkbox"/>	.gitignore	a 	248 bytes
<input type="checkbox"/>	.htaccess	a 	113 bytes
<input type="checkbox"/>	LICENSE	a 	1.04 kB
<input type="checkbox"/>	README.md	a 	2.25 kB

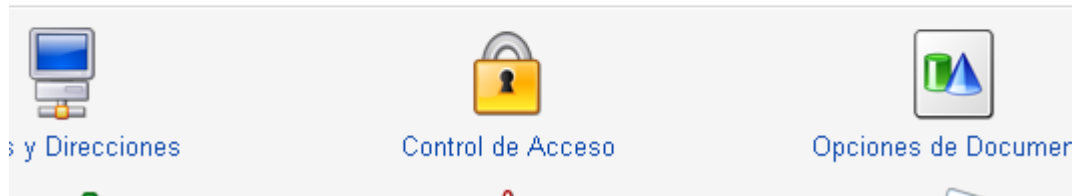
Para realizar la asignación de contraseña al directorio de documentación primero creamos la contraseña en nuestra maquina debían

```
root@debian8:/var/www/meow/documentacion# htpasswd -c /etc/apache2/contraseñas  
usuario_
```

Ponemos la contraseña que queramos cuando nos lo pidan y en este caso he utilizado como contraseña usuario. Luego al realizar esto vamos al Webmin y realizamos modificaciones en el directorio de documentación en el control de acceso

Opciones de Por-Directorio

Para Directory /var/www/meow/documentacion en meowdaw2.inf:80



Control de Acceso para Directory /var/www/web1/a

Nombre de ámbito de autenticación ☐ Por defecto ☒ Filtro de acceso

Tipo de Autenticación ☒ Basic

Restringir acceso por login ☐ Defecto ☐ Sólo estos usuarios:
☐ Sólo estos grupos:
☒ Todos los usuarios válidos

Los clientes deben satisfacer ☒ Por defecto ☐ Todos los controles de acceso ☐ Cualquier control de acceso

Pass basic login failures to next module? ☐ Si ☐ No ☒ Defecto

Basic login user file types ☒ Text file ☐ DBM database

Archivo de texto de usuario ☐ Defecto ☒ /etc/apache2/contrasenias [Editar usuarios](#)

Restringir acceso Orden de chequeo de acceso: ☐ Denegar y luego permitir ☐ Permitir y luego denegar ☐ Fallo mutuo ☒ Por defecto

Acción	Condición
<input type="button" value="▼"/>	<input type="button" value="Todos los requerimientos"/>

Al realizar este último paso, al entrar en la dirección que creamos nos encontraremos con la aplicación en marcha.

MEOW [Entrar](#) [Registro](#)

Identificarse

Email

Contraseña

[¿No tienes cuenta? Regístrate](#)

© Alejandro López Ortiz 2017 / MEOW

Google Chrome dejará de recibir actualizaciones de Google Chrome porque ya no

Se requiere autenticación

http://meowdaw2.inf necesita un nombre de usuario y una contraseña.

Tu conexión con este sitio no es privada.

Nombre de usuario:

Contraseña:

Iniciar sesión

Cancelar

INGLÉS

In this app you will be able to see the timeline in which you will see the meows of the people you follow and your own, you can give to like and delete yours. You can write meows with photos and documents and you will see in the timeline.

There is the section of people, where you can search who you want and see all registered users. You can follow and unfollow the people.

You can send private messages to the users you follow and you will receive notifications in case of private messages or if someone likes meow you.

In the section of profiles you will see the stats of the person you are viewing the profile, their meows and a gallery with the photos that have uploaded.