



RMF02: Tutorial (1)

Proyecto web de ejemplo:
análisis, diseño y preparación

Índice



- Antes de nada
- La aplicación
- Análisis
- Diseño
- Preparando

Antes de nada



- Esta presentación forma parte de los apuntes del certificado de profesionalidad **IFCD0210 – Desarrollo de aplicaciones con tecnologías web** que imparte Robert Sallent.



- También se incluye como documentación del *framework* RMF.

Antes de nada



- Este documento describe, a grandes rasgos, cómo realizar las fases de análisis y diseño de una pequeña aplicación que será implementada haciendo uso del *framework RMF*.
- La fase de implementación está documentada en la siguiente presentación.



Ejemplos aplicaciones creadas con RMF





RMF - RobS Micro Framework

Para el desarrollo de aplicaciones web

Hola [pepe](#) ([pepe@pepe.com](#)), eres administrador [Logout](#)

[Inicio](#) [Registro](#) [ADMIN](#)

Presentación

Este micro framework ha sido desarrollado con fines docentes para el CP de desarrollo de aplicaciones con tecnologías web (IFCD0210) que imparte Robert Sallent.

Es un ejemplo de arquitectura modelo-vista-controlador sencillo para entender los conceptos y poder trabajar con él.

A lo largo del curso se desarrollarán varios proyectos de ejemplo usando este pequeño framework, para ir entendiendo los conceptos básicos comunes a este tipo de herramientas de trabajo MVC existentes en PHP.

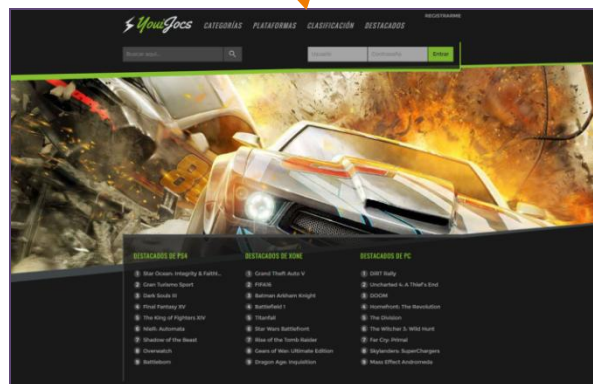
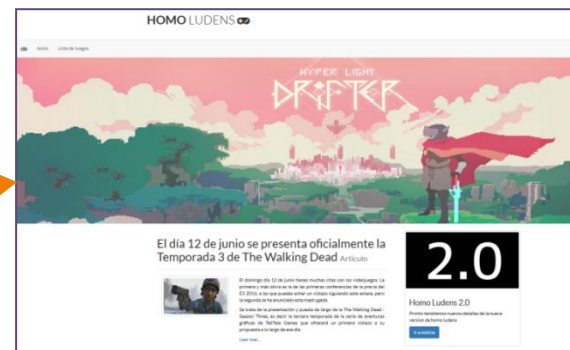
NO ES 100% SEGURO, así que no se debe usar para desarrollos en entornos de producción.

En el mismo curso, en el último módulo, utilizaremos otro CodeIgniter para desarrollos más complejos.

RobS micro Framework - solo para fines docentes

Robert Sallent [@](#) - CIFO del Vallès'16.

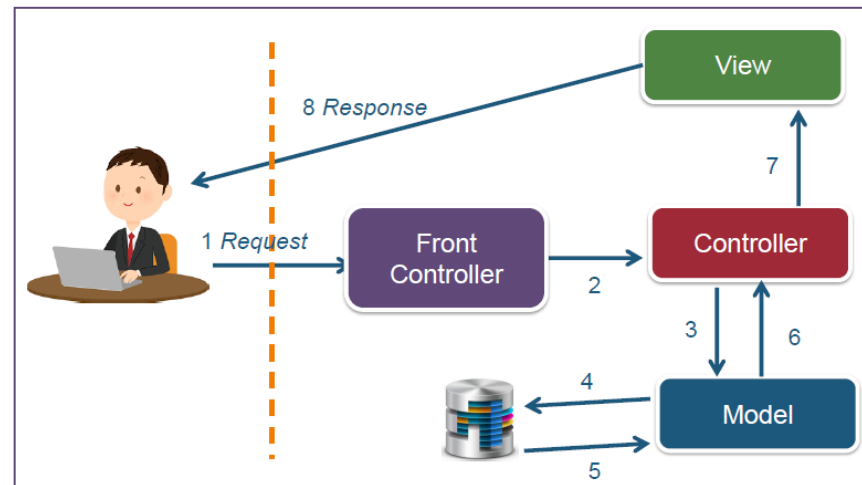




Antes de nada



- Recordemos que *RMF* trabaja usando una arquitectura **Modelo – Vista - Controlador** con **controlador frontal** (*dispatcher*).



- En la fase de diseño lo tendremos presente.

Primeros pasos

Análisis del proyecto



La aplicación



- Vamos a crear una aplicación sencilla: una web de venta de segunda mano, donde los usuarios registrados podrán poner sus anuncios y el resto de usuarios podrá verlos.
- El análisis inicial ha determinado que habrá **tres perfiles de usuario** (roles): **usuario no registrado**, **usuario registrado** y **administrador**.

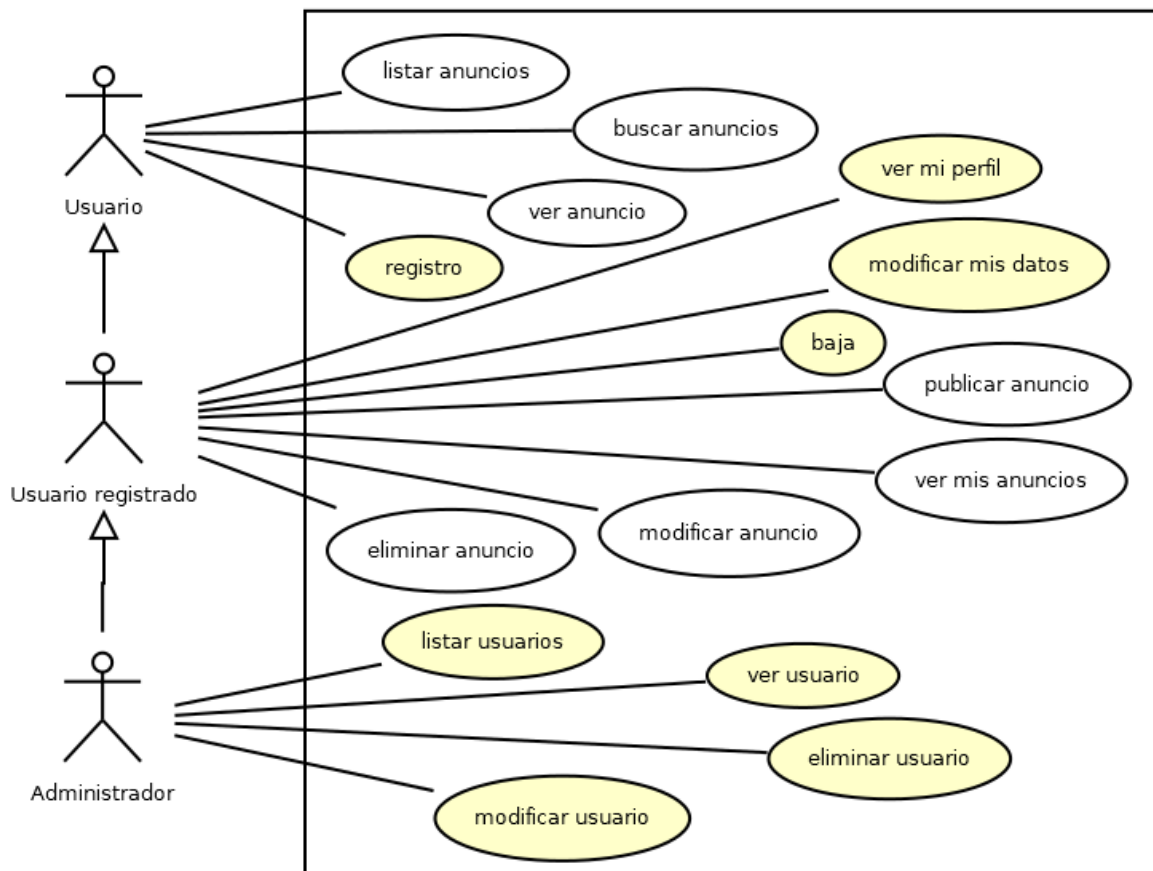


La aplicación



- Los **usuarios no registrados** pueden: registrarse, ver el listado de anuncios, buscar anuncios y ver los detalles de un anuncio concreto.
- Los **usuarios registrados** además deben poder: identificarse, modificar sus datos, darse de baja, publicar anuncios, ver sus anuncios, modificar sus anuncios y eliminar sus anuncios.
- El **administrador** podrá realizar una gestión completa de usuarios y además eliminar cualquier anuncio.

Ejemplo diagrama de casos de uso



Casos de uso



- De los casos de uso anteriores, *RMF* ya dispone de una implementación para todos los que se han coloreado (es decir, que ya vienen implementados de serie los relacionados con usuarios).

Hola **Administrador** (*admin@rmf*) , eres administrador [Logout](#)

Inicio ADMIN

Lista de usuarios registrados

Lista de usuarios registrados en la aplicación RMF - RobS Micro Framework

Imagen	Usuario	Nombre	Email	Operaciones
	admin	Administrador	admin@rmf	
	Robert	robert	robert@juegayestudia.com	

Crear un nuevo usuario

Casos de uso



- Los casos de uso se recogen y documentan en el documento de especificación.
- En este documento tan solo se muestra un resumen de los casos de uso a implementar (los relativos a anuncios).

CASO DE USO: REGISTRO DE USUARIO			
Versión	1.0	Fecha	02/09/2016
Autores	Robert		
Descripción	A continuación se describe cómo un usuario se puede registrar en la aplicación "anuncios".		
Actores	Usuario		
Precondiciones	El usuario no debe estar registrado previamente.		
Flujo principal	<ol style="list-style-type: none">1. El usuario accede a la aplicación "anuncios"2. El usuario selecciona la opción "registro" que se encuentra disponible en la zona de menú.3. El sistema muestra un formulario de registro solicitando la siguiente información: nombre, apellidos...4. Tras rellenar los datos solicitados, el usuario pulsa el botón "aceptar".5. El sistema toma los datos, los valida y los guarda.6. El sistema informa al usuario que se ha completado exitosamente el proceso de registro.		
Flujos alternativos	En caso de error en los datos, se informará al usuario para los modifique.		
Postcondiciones	Los datos del usuario quedan guardados.		
Requerimientos no funcionales	El proceso de registro debe funcionar desde dispositivos móviles.		
Prioridad	Alta		
Comentarios	Si el usuario no selecciona una foto para el perfil, se usará una imagen de usuario por defecto.		

Casos de uso (cualquier usuario)



- **Listar anuncios:** si el usuario selecciona la opción “listar anuncios” del menú, se le mostrará un listado de todos los anuncios.
- **Buscar anuncios:** se muestra un formulario que permite buscar anuncios a partir del título o descripción, precio... cuando el usuario indique los criterios y seleccione “buscar”, se le mostrarán los anuncios coincidentes.
- **Ver anuncios:** cuando el usuario haga clic sobre un anuncio desde el listado de anuncios, verá los detalles del mismo.

Casos de uso (usuarios registrados)



- **Publicar anuncio:** un usuario registrado puede publicar un anuncio seleccionando la opción en el menú e introduciendo los datos en el formulario que aparece a continuación.
- **Ver mis anuncios:** la opción “ver mis anuncios” permite a un usuario registrado obtener un listado de sus anuncios.
- **Modificar anuncio:** desde un listado de anuncios, el propietario dispondrá de un enlace para editar los datos mediante formulario.
- **Eliminar anuncio:** desde un listado de anuncios, el propietario y el administrador podrán eliminar (se pide confirmación).

La tecnología



- La tecnología que usaremos para el desarrollo es la que hemos estudiado en el curso-.:
 - *HTML*, *CSS*, *JavaScript* (si hace falta) para las vistas.
 - *PHP* para el lado del servidor.
 - *MySQL* o *MariaDB* para la base de datos.
 - *RMF* como *framework* base para el desarrollo de la aplicación (arquitectura Modelo-Vista-Controlador).

Diseño

Diseñando la solución



Diseño



- En este punto, realizaremos el diseño de la base de datos, y plantearemos todo lo que necesitamos para el *MVC*.
- Respecto a los elementos del *MVC*, determinaremos:
 - Los elementos que necesitamos (para modelo, controladores y vistas).
 - Los nombres de estos elementos (clases, carpetas y ficheros).
 - Las operaciones que deben realizar.
 - Los nombres para las operaciones (métodos).
 - Las rutas de la aplicación (para los enlaces en las vistas).
- No comenzaremos la implementación hasta que no esté todo.

El modelo de datos

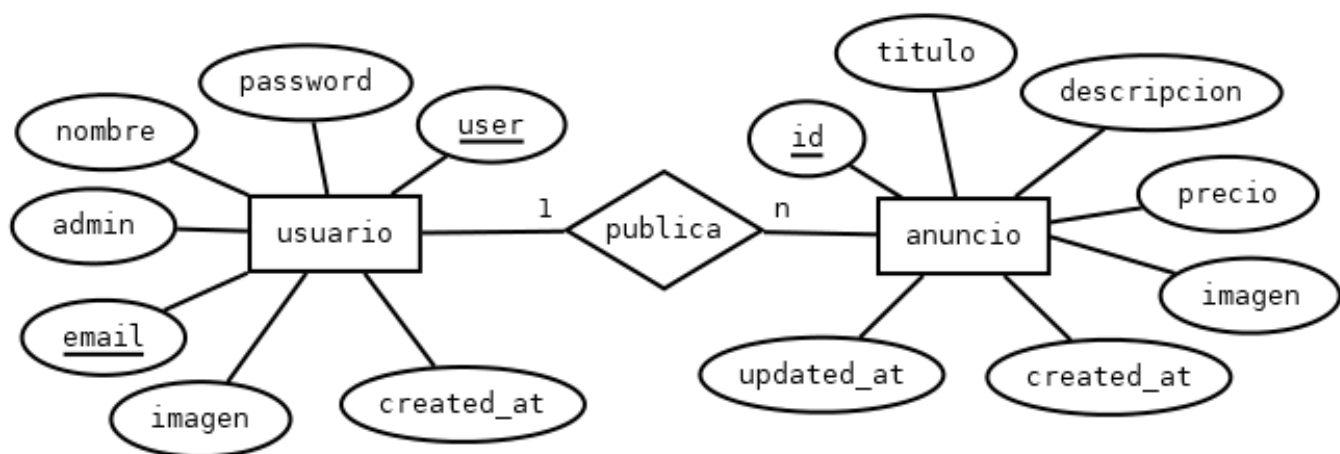


- A partir de los casos de uso, queda bastante claro que la aplicación necesita manejar información de **usuarios** y **anuncios**.
 - No haremos cambios en la gestión de usuarios que hace el *framework*.
 - Debemos determinar la información sobre los anuncios que será relevante para el sistema.
- Para diseñar la estructura de la base de datos, haremos uso de los modelos entidad-relación (ER) y relacional.

El modelo de datos



- Este será el diagrama entidad-relación:

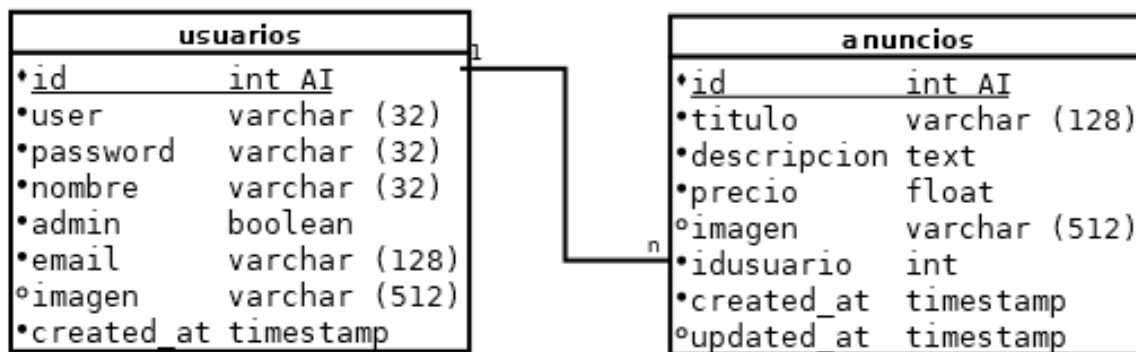


- En principio, de cada anuncio necesitamos: identificador (único), título, descripción, precio, imagen, fecha de publicación y fecha de modificación.



El modelo de datos

- El modelo relacional para la *BDD* queda de la siguiente forma:





El modelo del MVC

- Por lo que respecta modelo del MVC, tendremos dos clases: *UsuarioModel* (ya implementada) y *AnuncioModel*.
- Si miramos **los casos de uso**, descubrimos que para los anuncios necesitamos:
 1. Una forma de **recuperar** todos los anuncios (para el listado): método `get()`.
 2. Una forma de recuperar los anuncios mediante un filtro (para la búsqueda): método `getFiltered()`.

El modelo del MVC



3. Recuperar anuncios a partir de un id de usuario (“ver mis anuncios”): método `getByUser()`.
 4. Recuperar un anuncio por id (“ver detalles”): método `getById ()`.
 5. Recuperar el número total de anuncios: método `total()`.
 6. **Guardar** un anuncio: método `guardar ()`.
 7. **Modificar** los datos de un anuncio: método `actualizar()`.
 8. **Borrar** un anuncio: método `borrar()`.
- Estos ocho métodos deberán estar implementados dentro de la clase *AnuncioModel*.

La lógica (controlador)



- De la misma forma que con el modelo, podemos determinar que necesitaremos dos controladores a partir de la observación de los casos de uso:
 - Uno para gestionar las **operaciones con los usuarios**.
 - Otro para gestionar las **operaciones con los anuncios**.
- Como el controlador **Usuario** ya se encuentra implementado, nos centraremos en la implementación del controlador de anuncios.

La lógica (controlador)



- Los métodos del controlador de anuncios serán:
 - `index()` (por defecto): invocará a `listar`.
 - `listar()`: listar anuncios.
 - `buscar()`: buscar anuncios.
 - `ver(id)`: ver anuncio.
 - `crear()` y `guardar()`: para crear un anuncio (2 pasos).
 - `editar(id)` y `modificar()`: para modificar un anuncio (2 pasos).
 - `borrar(id)` y `destruir()`: para eliminar un anuncio (2 pasos).

Las rutas



- *RMF* utiliza **URLs amigables**, donde el controlador, método y parámetro se codifican en la URL de la siguiente forma:

`urlproyecto/controlador/método/parámetro`

- Por ejemplo, la ruta para ver el anuncio con identificador 3, tendrá la siguiente forma: `urlproyecto/anuncio/ver/3`
- Los enlaces tendrán un aspecto similar a éste:

`Lista de anuncios`

Las vistas (GUI)



- A partir de la descripción de los casos de uso también detectamos qué vistas necesitaremos para llevar a cabo las operaciones y cómo navega el usuario a través de ellas.
- Por ejemplo:
 - Para la operación “listar anuncios”, necesitamos una vista que muestre el listado de anuncios en forma de tabla o similar.
 - Para la operación “ver anuncio”, el usuario debe pasar por el listado de anuncios o el buscador y, sobre la vista con los resultados, hacer clic en un anuncio.
 - Etc.

Las vistas (GUI)



- Para diseñar las vistas podemos usar multitud de herramientas diferentes, pero muchas de ellas no son gratuitas o requieren registro.
- Siguiendo mi filosofía de usar herramientas libres y/o gratuitas en este curso, usaré la herramienta Pencil para hacer un diseño sencillo de la GUI.



Ejemplo aplicación pencil





PENCIL PROJECT

- [Home](#)
- [Features](#)
- [Downloads](#)
- [Stencils & Templates](#)
- [Wiki](#)

The first public beta build of the new Pencil 3.0 is now available for testing. [Learn more...](#)

An open-source GUI prototyping tool that's available for ALL platforms.

Pencil is built for the purpose of providing a free and open-source GUI prototyping tool that people can easily install and use to create mockups in popular desktop platforms.

The latest stable version of Pencil is **2.0.5** with **minor enhancements and bug-fixes**.

[Download for Ubuntu
Version 2.0.5, .deb, ~4.7 MB](#)

[For other platforms?
See all downloads »](#)



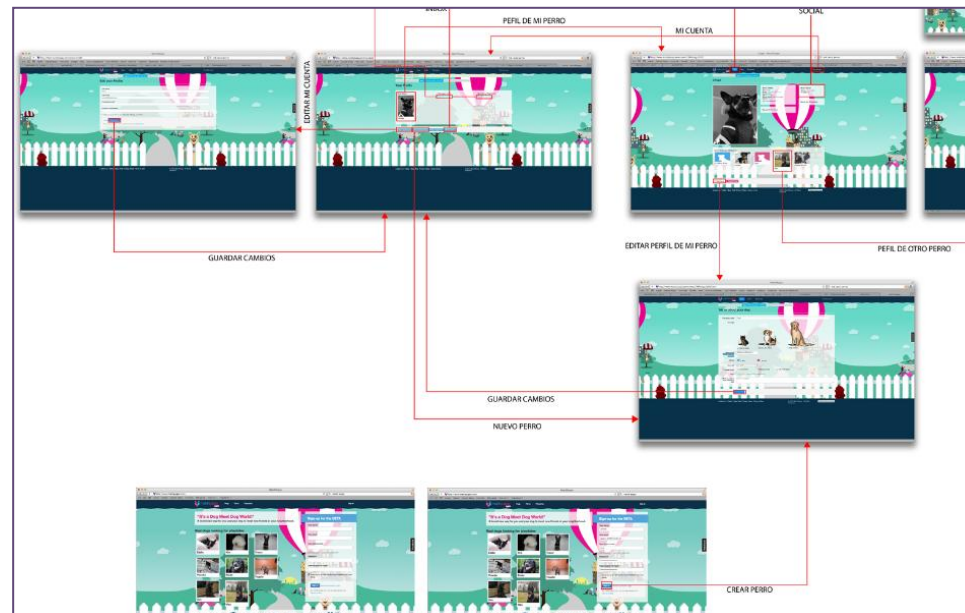
Project News

June 04, 2015 Pencil 3.0.0 beta 1 is **available for testing**.

Las vistas (GUI)



- También se deben hacer diagramas o **mapas de navegación** del sitio.



Proceso de creación vistas



- En la tarea de creación de las vistas:
 - Se hace un diseño inicial que se muestra y valida el cliente (**diseñadores**).
 - Después se maqueta (**maquetadores**).
 - Finalmente los **programadores** añadirán el código *PHP* que genera contenido de forma dinámica.



Preparando



Preparando la base de datos y el *framework*



La base de datos



- Para comenzar, debemos crear la base de datos y la tabla para los anuncios.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 	int(11)			No	Ninguna
2	titulo	varchar(128)	utf8_spanish_ci		No	Ninguna
3	descripcion	text	utf8_spanish_ci		No	Ninguna
4	precio	float			No	0
5	imagen	varchar(512)	utf8_spanish_ci		Sí	NULL
6	idusuario 	int(11)			No	Ninguna
7	created_at	timestamp			No	CURRENT_TIMESTAMP
8	updated_at	timestamp		on update CURRENT_TIMESTAMP	Sí	NULL



Preparando el framework



- Ahora vamos a descargar y configurar el *framework*, tal y como se describió en la anterior presentación.
- En este caso he optado por:
 - Descargar de <https://github.com/robertsallent/rmf> el fichero comprimido.
 - Descomprimir la carpeta *project* en mi *workspace* y renombrarla a *rmf_anuncios*.
 - Crear un nuevo proyecto de *Eclipse* con el nombre “*rmf_anuncios*”.

Preparando el framework



- También creo un *virtualhost* con el nombre de “*anuncios.local*”.
- Indicaremos en el fichero `Config.php` los datos de la base de datos y ejecutaremos el script `setup.php`.



- No olvidemos relacionar la tabla usuarios y anuncios en la *BDD*.

```
ALTER TABLE `anuncios` ADD FOREIGN KEY (`idusuario`) REFERENCES
`usuarios`(`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Ejemplo estructura de la BDD



Ejemplo probando que funciona





RMF - RobS Micro Framework

Para el desarrollo de aplicaciones

Hola **Administrador** (*admin@rmf*), eres administrador

[Inicio](#)

Panel de control

Operaciones para el administrador

i A continuación se muestran las operaciones que puede hacer el administrador.



- Listar
- Nuevo

Usuarios



- Operación 1
- Operación 2

Entidad 1



- Operación 1
- Operación 2

Entidad 2



- 1 usuarios registrados

Estadísticas

Perfil de usuario

i Estos son tus datos de usuario.

Mis datos	
Usuario:	admin
Nombre:	Administrador
Email:	admin@rmf
Fecha de alta:	08/04/2019
Hora de alta:	10:25:18



[Editar mis datos](#)



Añadiendo algunos datos



- Introduciremos algunos registros en las tablas de la base de datos.
 - Podemos crear unos cuantos usuarios directamente mediante la interfaz gráfica del *framework*.
 - Crearemos algunos anuncios directamente con algún cliente de *MySQL*, como el *MySQL Workbench*, el *phpmyadmin* o similares.
- Estos datos los usaremos en las diferentes pruebas que haremos durante la etapa de implementación.

Ejemplo datos de la BDD



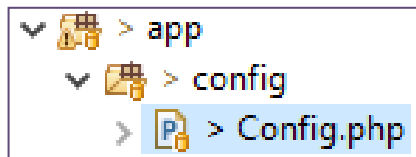
id	titulo	descripcion	precio	imagen	idusuario
1	Mesa escritorio	Escritorio de madera	100	NULL	2
2	Plato de ducha	Plato de ducha seminuevo, no muy limpio.	50	platoducha.jpg	4
3	Mesa de billar	Mesa de billar americano	200	NULL	3
4	Peluche	Peluche de gato	10	NULL	4
5	Reloj de pared	Reloj de pared antiguo.	200	reloj.png	3
6	Coche Ford	Ford Focus 2012 60000 kms	5000	NULL	2
7	Mueble comedor	Mueble para el comedor	100	mueble.jpg	4
8	Mesa nueva	Una mesa como cualquier otra.	50	mesa.png	3
9	Coche Seat	Coche Seat Toledo	2000	NULL	4
10	Juego de mesa	Juego parchís por un lado y oca por el otro	5	juego.jpg	2



Modificando la configuración

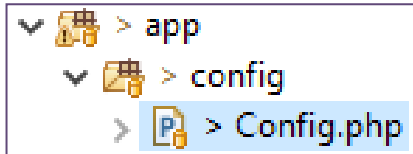


- Una vez que está todo funcionando, es un buen momento para modificar las opciones de configuración del proyecto en el fichero `app/config/Config.php`.
- Deberíamos indicar el nombre del proyecto, las carpetas para el MVC y para las imágenes...



```
class Config{  
    // CONFIGURACIÓN GENERAL  
    private $app_name='CIFO Anuncios';  
    private $company_name='Curso apps web';  
}
```

Ejemplo editando las carpetas en la configuración



```
// DIRECTORIOS POR DEFECTO PARA EL MVC
// son los directorios en los que el autoload buscará las clases
private $model_directory='app/model';
private $view_directory='app/views/';
private $controller_directory='app/controllers/';
private $template_directory='app/templates/';

private $library_directory='core/libraries/';

// DIRECTORIOS AUTOLOAD
// rutas adicionales donde el autoload debe buscar las clases que necesite
private $autoload_directories=[];
```


Todo listo



- Llegados a este punto, ya tenemos todo planificado y pensado para comenzar con la implementación.
- En el siguiente documento se recoge cómo implementar la aplicación haciendo uso del *framework RMF*.