



Proyecto Fin de Carrera

Plataforma de Alojamiento
(Título en inglés o castellano, tamaño 18)

Para acceder al Título de
INGENIERO EN INFORMÁTICA

Autor: Sergio Sainz Tudanca

Director: Carlos Blanco Bueno

Septiembre - 2015

Agradecimientos

Llega la hora de terminar, y con ello por qué no, echar la vista atrás a todos aquellos que estuvieron apoyándome todos estos años. Este proyecto es la finalización de varios años de trabajo y esfuerzo que sin ellos no hubiera sido posible.

Para empezar, quisiera agradecer a la Universidad de Cantabria, a la Facultad de Ciencias y a todo el equipo docente que me han enseñado a llegar hasta aquí y a entender lo que significa ser un ingeniero. En particular a mi tutor del proyecto Carlos Blanco Bueno ya que me animó y asesoró hasta finalizar el proyecto.

A mis compañeros de Emancipia, empresa en la que he desarrollado el proyecto. Elena, Beatriz, Guillermo, Paloma, Diego, Xutao, Pablo, seguro que me olvido de alguno, que me perdone. Gracias por acompañarme durante el proyecto dentro de la empresa haciendo de este viaje una experiencia inolvidable tanto en los buenos como en los malos momentos y por todo el apoyo recibido durante este tiempo. En especial a los socios fundadores de la empresa Alfonso y Martín por haber confiado en mí y no haberse dado por vencidos durante todo este tiempo.

No me quiero olvidar de mi familia por todo el apoyo recibido tanto económico como emocional durante la carrera y que espero que me sigan dando después de esto, sin ellos nunca hubiera podido estudiar y quien sabe algún día dedicarme a aquello que me apasiona.

A todos y cada uno de ellos, gracias de corazón por ayudarme a llegar hasta aquí.

Resumen del proyecto

Todos los años, miles de estudiantes, profesores e investigadores necesitan encontrar alojamiento en las inmediaciones de sus centros de estudio y trabajo. Por su parte, los propietarios deben esforzarse cada pocos meses en encontrar nuevos inquilinos, atenderles y administrar sus propiedades. En relación permanente con ambos colectivos se encuentran las instituciones educativas obligadas a minimizar los problemas de alojamiento a sus miembros.

Bajo esta situación nace Emancipia servicios de alojamiento una respuesta integral que, viene a solucionar los problemas de promoción y búsqueda de alojamiento de alquiler, así como la gestión diaria de la estancia: acomodación, atención, cobros, mantenimiento y reparaciones.

Hoy en día todas las empresas precisan de una importante presencia en la Red, más si sus clientes objetivo están comprendidos en una franja de edad de entre 18 y 30 años, que ya son todos nativos digitales, por lo que la Red y las nuevas herramientas de comunicación digital, son sus fuentes y medios para obtener información, bienes y servicios.

De esta forma el objetivo principal del proyecto es diseñar, construir y desplegar una herramienta digital capaz de mejorar la presencia de la empresa en la Red y reducir costes, automatizando procesos en la gestión y en la prestación de servicios derivados del día a día del alquiler en piso compartido, a la vez que hacer partícipe al usuario de dicha gestión.

Palabras clave:

Gestor de alojamiento, CakePhp, desarrollo web, pasarela de pago, automatización, PHP, HTML5, CSS3, diseño adaptado móviles.

Abstract

Fin

Contenido

Agradecimientos	2
Resumen del proyecto	3
Abstract.....	4
Lista de acrónimos.....	7
Índice de ilustraciones y tablas.....	8
CAPÍTULO1.....	10
Introducción	10
1.1 Motivación.....	10
1.2 Objetivos.....	12
CAPÍTULO2.....	13
Herramientas y tecnologías.....	13
2.1 Lenguajes.....	13
2.1.1 HTML5 / HTML4.1	13
2.1.2 CSS 2.1 / CSS3	14
2.2.3 PHP 5.3	15
2.2.4 Javascript	16
2.2 Tecnologías.....	16
2.2.1 Framework de desarrollo CakePHP	16
2.2.2 JQuery.....	19
2.2.2.1 GMap3	20
2.2.2.2 html2canvas	20
2.2.3 Bootstrap.....	21
2.2.4 MySql	21
2.2.5 Mandrill	22
2.2.6Métodos de pagos virtuales	23
2.3 Metodología de desarrollo	24
2.3.1 Metodologías Ágiles.....	24
2.3.2 Metodología de desarrollo iterativo e incremental	25
2.3.3 Iteraciones planificadas	27
CAPÍTULO3.....	28
Desarrollo e implementación	28
3.1 Captura y análisis de los requisitos.....	28

3.1.1 Requisitos funcionales.....	28
3.1.1.1 Requisitos funcionales del gestor interno.....	28
3.1.1.2 Requisitos funcionales del buscador.....	30
3.1.1.3 Requisitos funcionales del área de cliente	32
3.1.2 Requisitos no funcionales.....	34
3.2 Especificación de los casos de usos.....	35
3.2.1 Diagramas de casos de usos.....	35
3.3 Diseño del sistema	45
3.3.1. Diseño arquitectónico	45
3.3.2. Patrón de diseño MVC	47
3.4 Diseño del sistema	55
3.4.1.- Diseño detallado.....	55
3.5 Implementación	58
3.5.1 Implementación del buscador.....	59
3.5.1.1 Implementación de la capa de datos	60
3.5.1.2 Implementación de la capa de la capa de negocio.....	62
3.5.1.3 Implementación de la capa de presentación.	63
CAPÍTULO4.....	65
Validación y pruebas	65
4.1 Pruebas realizadas.....	65
4.1.1 Pruebas unitarias.....	65
4.1.2 Pruebas de integración.....	66
4.1.3 Pruebas de sistema.....	66
4.2.3.1 Pruebas de rendimiento	66
4.3.2 Pruebas de seguridad	67
4.4 Pruebas de aceptación.....	67
CAPÍTULO5.....	68
Conclusiones y.....	68
trabajos futuros.....	68
5.1 Conclusiones.....	68
5.2 Trabajos futuros.....	69

Lista de acrónimos

HTML	HyperTextMarkupLanguage
WHATWG	Web Hypertext Application Technology Working Group
CSS	Cascading Style Sheets
W3C	World Wide Web Consortium
OOP	ObjectOrientedProgramming
MIT	Massachusetts Institute of Technology
MVC	Model–View–Controller
IPs	Internet Protocol
BD	Base de datos
HASH	Hypertext Pre-processor
SQL	StructuredQueryLanguage
URL	UniformResourceLocator
CRUD	Create, Read, Update and Delete
Ajax	Asynchronous JavaScript
API	ApplicationProgramming Interface
DOM	DocumentObjectModel
JS	JavaScript
HTTP	Hypertext Transfer Protocol
TPV	Terminal Punto de Venta
XML	eXtensibleMarkupLanguage
SMTP	Simple Mail Transfer Protocol
CSV	Comma-SeparatedValues

Índice de ilustraciones y tablas

Ilustración 1	Evolución del número de estudiantes Erasmus en España.	5
Ilustración 2	Ejemplo de código CSS3 “media-query”	
Ilustración 3	Ejemplo de código CSS3 “animaciones”	
Ilustración 4	Modelo Ask /app/Model/ask.php	
Ilustración 5	Modelo Charge /app/Model/Charge.php	
Ilustración 6	Ejemplo de literal para traducir /app/Views/flats/contacto.ctp	
Ilustración 7	Ejemplo de búsqueda en la BD /app/Controllers/FlatsController.php	
Ilustración 8	Ejemplo de guardado recursivo /app/Controllers/FlatsController.php	
Ilustración 9	Mapa inteligente accesible desde el buscador	
Ilustración 10	Fragmento de código JS de la librería html2canvas	
Ilustración 11	Gráfico del funcionamiento de los servidores	
Ilustración 12	Extracto de la BD utilizando phpMyAdmin.	
Ilustración 13	Integración del protocolo SMTP con la plataforma	
Ilustración 14	Informe de seguimiento de Julio y Agosto 2015 en Mandrill	
Ilustración 15	Formulario de reserva con las modalidades del pago.	
Ilustración 16	Diagrama comparativo entre una metodología tradicional y el desarrollo ágil.	
Ilustración 17	Esquema de un desarrollo iterativo e incremental	
Ilustración 18	Captura de las tareas de un prototipo en la herramienta Trello.	
Ilustración 19	Diagrama de casos de uso general.	
Ilustración 20	Diagrama de casos de uso del Área de cliente.	
Ilustración 21	Diagramas de casos de uso del Portal buscador.	
Ilustración 22	Diagrama de secuencia del caso de uso P002 Realizar reserva	
Ilustración 23	Esquema de la arquitectura en tres capas.	
Ilustración 24	Ejemplo de View /app/View/Roomsusers/view.ctp	
Ilustración 25	layout PDF /app/View/Layouts/pdf/defaultp.ctp	
Ilustración 26	element del menú /app/View/elements/menu.ctp	
Ilustración 27	helper form /app/View/Users/edit.ctp	
Ilustración 28	AppController /app/Controllers/AppController.php	

Ilustración 30	PluginCakePdf /app/config/bootstrap.php	
Ilustración 31	Modelo Flat /app/Model/flat.php	
Ilustración 32	Diagrama de un petición en CakePhp	1
Ilustración 33	Diagrama del sistema de la plataforma Emancipia.	
Ilustración 34	Área de cliente http://app.emancipia.net	
Ilustración 35	Buscador http://pisos.emancipia.net/flats/portada	
Ilustración 36	Gestión interna http://emancipia.net	
Ilustración 37	Conexión a la base de datos /app/Config/database.php	
Ilustración 38	Modelos /httpdocs/app/Model/	
Ilustración 39	Modelo de la tabla Piso /app/Models/flat.php	
Ilustración 40	Controladores /pisos.emancipia.net/app/Controllers/	
Ilustración 41	Controlador de Flat /pisos.emancipia.net/app/Controllers/FlatsControllers.php	
Ilustración 42	función search_name del FlatsControllers.php	
Ilustración 43	función privada __sin_resultados del FlatsControllers.php	
Ilustración 44	Ficheros CSS y JS /pisos.emancipia.net/app/webroot/	
Ilustración 45	Mockup Diseño adaptado a dispositivos móviles.	
Tabla 1	Comparativa de aplicaciones de la plataforma.	
Tabla 2	Comparativa de rendimiento de las aplicaciones.	

CAPÍTULO 1

Introducción

Este primer capítulo describe brevemente los antecedentes que motivan el desarrollo de este Proyecto Fin de Carrera y expone los objetivos que se deben cumplir a la finalización del proyecto.

1.1 Motivación

En la última década se han modificado los hábitos de consumo y búsqueda de información, principalmente por el crecimiento de los distintos puntos de acceso a internet. Hasta hace unos pocos años, el acceso a la Red se realizaba únicamente a través de un ordenador personal, pero el progresivo aumento de la venta de los *smartphones* ha motivado un cambio del mercado y de cómo las empresas se relacionan con los usuarios, dando acceso a las empresas un número mayor de público objetivo.

Como se ha comentado anteriormente, nuestro público objetivo son ya nativos digitales, por lo que internet y las nuevas herramientas de comunicación son sus fuentes y medios para obtener información, bienes y servicios. Aunque en Junio del 2014 todavía existe una desconfianza a la hora de realizar pagos *on-line*, esa desconfianza está paulatinamente decreciendo gracias al empuje de las nuevas generaciones digitales y las nuevas y mejoradas implementaciones de medidas de seguridad.

En la última década el número de estudiantes Erasmus no ha parado de crecer, muchos de ellos eligen España como destino para completar sus estudios como se refleja en la Ilustración 1, llegando en el curso académico del 2011-12 a los 39.300 estudiantes [1].

Introducción - CAPÍTULO 1

Evolución de la movilidad de estudiantes en el programa Erasmus

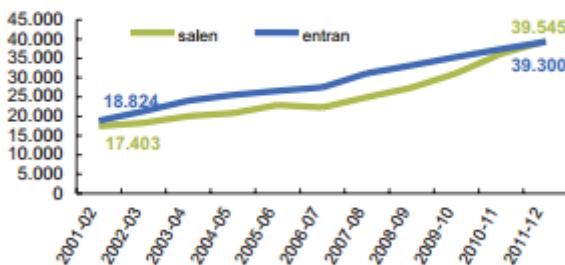


Ilustración 1 Evolución del número de estudiantes Erasmus en España.

En el curso académico 2013-2014, la Universidad de Cantabria recibió 435 estudiantes entre extranjeros y estudiantes de fuera de la capital de Cantabria, así como profesores e investigadores [2] que necesitan encontrar un alojamiento cercano a la universidad.

Por otra parte los propietarios de los inmuebles se ven abocados a multiplicar sus esfuerzos cada vez más en encontrar nuevos inquilinos, atenderles y administrar el día a día de sus propiedades. Debido al progresivo distanciamiento cultural de estos con los inquilinos y ante la pérdida de herramientas de promoción anticuadas (carteles, tablones, etc.) Ante esta nueva situación se sitúa Emancipia.

Desde Emancipia se quiere aportar un conjunto de servicios y herramientas orientadas a facilitar los primeros pasos de la emancipación de las personas y/o atender las dificultades derivadas de la necesidad de compatibilizar un buen rendimiento académico o investigador con determinadas tareas domésticas, ofreciendo así todas las comodidades de una residencia estudiantil pero con la libertad de un piso compartido.

Del mismo modo, a los propietarios se les ofrece una herramienta de gestión integral del alojamiento de su inmueble. Cubriendo así desde la promoción del inmueble en medios digitales, la gestión del día a día mediante la automatizando de procesos como los del cobro de la renta, el cálculo de los consumos, un sistema gestor de incidencias. Añadiendo así más transparencia de cara al inquilino.

De este modo, las empresas se han visto obligadas a modernizarse y a actualizar tanto sus herramientas de cara al público, como las incluidas en la gestión interna del propio negocio.

La motivación del proyecto se basa en este nuevo paradigma; en el cambio del modelo en que los usuarios interactúan con el producto, siendo la consecución del proyecto implementar una plataforma *on-line* adaptada a las nuevas necesidades del mercado que mejore y automatice los procesos derivados del alojamiento, permitiendo a

Emancipia poder soportar un mayor número de clientes sin que esto suponga un gran incremento en los gastos.

1.2 Objetivos

Con esta serie de premisas, se propone ofrecer una solución a estos problemas. Diseñar, implementar y desplegar una plataforma *on-line*, adaptaba a dispositivos móviles, escalable y modular que permita gestionar la actividad de Emancipia además de un portal transparente a los usuarios que contraten sus servicios.

De este modo se permite a Emancipia expandir su mercado y ampliar los servicios que presta.

Para poder alcanzar dicho objetivo se plantea la definición detallada de unos objetivos mínimos que han de cumplirse.

- Herramienta de **gestión de cobros** en la que se incluyan los gastos generados de un alojamiento compartido.
- Herramienta de **disponibilidad de los inmuebles y habitaciones** con vistas, listados y calendarios que agilicen la extracción y compresión de los datos.
- **Servidor de correoelectrónico:** que permita el envío de correos electrónicos a usuarios con diferentes plantillas.
- Integración de **plataformas de pagos virtuales** en la plataforma.
- **Herramienta de búsqueda** de habitaciones e inmuebles, con el que se pueda realizar reservas.
- Herramienta de **geolocalización de los inmuebles**.
- **Área del usuario** en el que se le muestre toda su información tanto personal como su historial de alojamiento.

CAPÍTULO 2

Herramientas y tecnologías

En este capítulo se explican brevemente los distintos lenguajes utilizados en el desarrollo del proyecto y las tecnologías en las que se ha apoyado la implementación y el despliegue de la plataforma. Por último presentar la metodología seguida en el transcurso del desarrollo del proyecto.

2.1 Lenguajes

2.1.1 HTML5 / HTML4.1

HTML es el lenguaje de marcado básico de casi todo el contenido web, con el que se escribe la estructura y la semántica del contenido de un documento web. El contenido dentro de una página web es etiquetado con elementos HTML como ``, `<title>`, `<p>`, `<div>`, entre otros.

HTML es un estándar internacional con especificaciones que son reguladas por el World Wide Web Consortium y el WHATWG. Es considerado un "estándar viviente" y está técnicamente siempre bajo construcción. La versión actual de la especificación HTML se conoce como HTML5 aunque todavía no está estandarizado oficialmente, siendo la quinta revisión del estándar que fue creado en 1990 [3].

HTML5 al no estar todavía estandarizado no se ha empleado completamente en el desarrollo del proyecto. No todos los navegadores actuales soportan las nuevas características del estándar, debido a ello en el desarrollo del proyecto se ha utilizado como base el estándar actual HTML4.1. altamente compatible con los principales navegadores actuales.

A continuación se detallaran algunas de las nuevas características de HTML5 [4]:

- Semántica: Nuevas etiquetas que permiten marcar el contenido con una mayor precisión según su contenido.
 - *header, footer:* estas etiquetas individuales ahorran tener que insertar IDs para cada uno, como se solía hacer anteriormente. Además, se

pueden insertar *headers* y *footers* para cada sección, en lugar de tener que hacerlo únicamente en general.

- *nav*: los elementos de navegación.
- Desconectado y almacenamiento: permite a páginas web almacenar datos, localmente, en el lado del cliente y operar fuera de línea de manera más eficiente. Usando IndexedDB
- Multimedia: nuevas etiquetas de marcado <audio><video> las cuales nos permitirán manipular archivos multimedia de manera nativa.
- Implementación de la etiqueta <canvas> la cual puede ser usada para el desarrollo de gráficos apoyándose en JavaScript.
- WebGL trae gráficos en 3D para la Web mediante la introducción de una API que cumple estrictamente la OpenGL ES 2.0
- Rendimiento e Integración: proporcionar una mayor optimización de la velocidad y un mejor uso del hardware del equipo.
- WebWorkers: permiten la ejecución de scripts en segundo plano evitando así la ralentización de la web.
- Geolocalización: permite compartir la geolocalización del navegador.

2.1.2 CSS 2.1 / CSS3

CSS es un lenguaje que es usado para separar la presentación de la estructura HTML en un documento web. Este lenguaje aplica reglas de estilo a los elementos HTML, quedando de esta manera separada de la estructura HTML.

El W3C es el encargado de formular la especificación y estandarización del CSS. El CSS puede ser definido en un archivo con extensión .css separado ó en el mismo archivo HTML mediante el atributo *style* de las etiquetas HTML. Los tres elementos principales en los que se basa el CSS son los selectores, atributos y valores [5].

Con CSS3 hubo un cambio en el desarrollo separándolo en módulos los cuales añaden nuevas funcionalidades a las definidas en la anterior estandarización CSS2, de manera que se preservan las anteriores para mantener la compatibilidad [6].

Algunas de las novedades destacadas incluidas en CSS3: [5]

- Bordes redondeados: “border-radius”
- Sombras: box-shadow
- Transiciones de los elementos: transition-property, transition-duration, transition-timing-function, transition-delay.
- Fuentes tipográficas: @font-fae
- CSS aplicable a distintas resoluciones: @media (min-width: 500px).

En la Ilustración 2 podemos ver un ejemplo aplicado de CSS3, en este caso se aplica un mínimo del ancho a una clase determinada, si el máximo de la resolución de la pantalla es de 700px

```
@media (max-width: 700px) {  
    .search-bar-select,  
    .search-bar-select:hover{  
        min-width: 180px;  
    }  
}
```

Ilustración 2 Ejemplo de código CSS3 “media-query”

```
.descubre-santander{  
    background-color:#555;  
    background-size: cover;  
    -webkit-animation: img-back 30s infinite ease-in-out;  
    -moz-animation: img-back 30s infinite ease-in-out;  
    animation: img-back 30s infinite ease-in-out;  
    background-size:100% 140%;  
}
```

Ilustración 3 Ejemplo de código CSS3 “animaciones”

En la Ilustración 3 vemos otro ejemplo de CSS3 aplicado en el proyecto, en el que se crea una animación con unas características definidas a una clase en particular. La repetición de la misma instrucción se debe a extender la regla a la compatibilidad de los distintos motores utilizados por los navegadores.

En el desarrollo del proyecto se ha utilizado principalmente el estándar CSS 2.1 debido a su alta compatibilidad en la mayoría de los navegadores actuales. Así mismo se han aprovechado algunas de las características del CSS3 siempre teniendo en cuenta su compatibilidad con los navegadores.

2.2.3 PHP 5.3

PHP es un acrónimo recursivo de "PHP: Hypertext Preprocessor" [7] que fue creado por Rasmus Lerdorf en 1994. Es un lenguaje de programación dinámico del lado del servidor el cual se puede incorporar directamente en el documento HTML, la versión actual es la 5, la versión 6 se encuentra en desarrollo. La versión 5.3 es la utilizada en el desarrollo del proyecto.

Las principales características de la versión 5 respecto a anteriores versiones.

- Un nuevo modelo OOP basado en el ZendEngine 2.0, Modelo de objetos completo.
- Una nueva extensión que mejora el soporte MySQL
- Soporte nativo integrado para SQLite

La elección de este lenguaje para el lado del servidor se debe a su alta compatibilidad con los servidores de alojamiento web actuales, su amplio uso para el desarrollo web y la gran comunidad que lo mantiene y lo da soporte.

2.2.4 Javascript

JavaScript es un lenguaje de programación interpretado. Se define como un lenguaje orientado a objetos, basado en prototipos, dinámico y soporta distintos estilos de programación: funcional, orientado a objetos e imperativo.

Su principal uso es del lado del cliente, aunque también tiene una vertiente del lado del servidor. Se implementa como parte del navegador web permitiendo mejoras en la interfaz de usuario y haciendo de esta forma páginas web dinámicas.

Todos los navegadores actuales interpretan el código JavaScript. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DocumentObjectModel.

2.2 Tecnologías

2.2.1 Framework de desarrollo CakePHP

CakePHP [9] es un *framework* de desarrollo para PHP, de distribución libre y de código abierto. Creado en 2005 bajo licencia MIT por Michal Tatarynowicz basándose en Ruby OnRails.

Según su arquitectura, se basa en el **patrón de diseño MVC** (Modelo, Vista y Controlador), incluye las clases *Controller*, *Model* y *View*, pero también incluye otras clases y objetos que hacen que el desarrollo en MVC sea un poco más rápido y agradable al desarrollador.

Los *Components*, *Behaviors* y *Helpers* son clases que proporcionan extensibilidad y reusabilidad, agregando rápidamente nuevas funcionalidades a las clases base. Esto reforzado con el paradigma de orientación a objetos, permite el desarrollo de aplicaciones altamente modularizables.

A continuación se detallaran brevemente las características más destacables del *framework* implementadas en el desarrollo del proyecto [10].

- **Validaciones integradas:** Los modelos de CakePHP reflejan las entidades y relaciones de la base de datos. En la Ilustración 4 se reflejan las relaciones que tiene la tabla *asks* con las tablas *flats* y *rooms*. Además, para cada campo de las tablas se pueden definir reglas de validación propias o utilizar las reglas predefinidas del core del CakePHP.

```
class Ask extends AppModel {  
    var $belongsTo = array('User',  
        'Flat' => array(  
            'className' => 'Flat',  
            'foreignKey' => 'key_id',  
            'conditions' => array('Ask.type_key' => 1)),  
        'Room' => array(  
            'className' => 'Room',  
            'foreignKey' => 'key_id',  
            'conditions' => array('Ask.type_key' => 2))  
    );  
}
```

Ilustración 4 Modelo Ask /app/Model/ask.php

En cada modificación o inserción de un registro, CakePHP validará todos los campos automáticamente, a menos que se le indique lo contrario desde el controlador. Entre las reglas de validación integradas de CakePhppodemos destacar: validar fechas, validar tarjetas de crédito, validar código postales, comprobar ips y urls, asegurar registros únicos entre otras.

En la Ilustración 5 se muestra la regla de validación predefinida del CakePhpisUnique, que nos asegura que el campo en este caso, el campo “referencia” va a ser único en la BD.

```
public $validate = array(  
    'referencia' => array(  
        'rule' => array('isUnique'),  
        'message' => 'La referencia debe ser única'  
    )  
);
```

Ilustración 5 Modelo Charge /app/Model/Charge.php

- **Url amigables:** Las urls antiguas de los desarrollos web, solían componerse de variables pasadas por parámetro en la propia dirección web, comprensibles para una maquina pero incomprendibles para un humano, esto hacía que la url careciera de carácter semántico para el usuario. Además de que los buscadores web estaban empezando a dar relevancia a las urls para su posicionamiento. Por todo ello nacieron lo que hoy se conoce como prettyurl, urls con carácter semántico. CakePHP provee de una herramienta para la creación de las urls amigables.
- **Internacionalización:** Debido a que la web es susceptible de ser visitada desde todas las partes del mundo, debería de ser posible una traducción del contenido. Para ello CakePHP posee una herramienta la cual dota de internacionalización a los literales que necesitemos dentro de nuestras vistas, permitiendo múltiples traducciones. Esto se consigue rodeando el literal del código PHP de la siguiente forma ___('texto a traducir'). Permitiendo además

que se detecte el país desde el cual se produce la visita y así poder cambiar el idioma de entrada automáticamente.

```
<div class="row">
    <div class="col-md-10 col-md-offset-1">
        <h2 class="white xlg helvetica"><?php echo __("Contacta con nosotros"); ?></h2>
        <h3 class="white md helvetica"><?php echo __("Si no has encontrado la respuesta que buscabas o no te ha resultado útil<br>puedes visitarnos en nuestras oficinas"); ?></h3>
    </div>
</div>
```

Ilustración 6 Ejemplo de literal para traducir /app/Views/flats/contacto.ctp

- **Consultas anidadas:** Hacer una consulta a la base de datos en CakePHP no implica necesariamente escribir ni una sola instrucción de SQL, aunque te permite hacerlo. Existe el método *find*, que se hereda en los modelos, el cual permite parametrizar las búsquedas. En los modelos de CakePHP se definen las relaciones entre las distintas entidades. De modo que al hacer una consulta, CakePHP nos retorna los campos del registro asociado, y toda la información relacionada a dicho registro teniendo en cuenta sus relaciones con las demás tablas.

En la Ilustración 7 se muestra un ejemplo de la instrucción *find*. En este caso utilizada sobre el Controlador Flat. El resultado de la instrucción *find*nos retornaría los alquileres de unos determinados inmuebles,entre unas fechas dadas y ordenados descendente por su identificador y la fecha de creación del resgistro. La búsqueda retorna un *array* asociativo con los resultados, en caso de no encontrarlos el retorno sería NULL.

```
$conditions = array('OR' => array(
    array('Room.flat_id' => $flats_id, 'Roomsuser.fecha_alta <=' => $inicio,
          'Roomsuser.fecha_baja >=' => $inicio),
    array('Room.flat_id' => $flats_id, 'Roomsuser.fecha_baja >=' => $fin,
          'Roomsuser.fecha_alta <=' => $fin)));
$fields = array('Roomsuser.*');
$order = array('Roomsuser.room_id', 'Roomsuser.created');
$recursive = 2;
$habitaciones_ocupadas = $this->Flat->Room->Roomsuser->find('all',
    compact('fields', 'conditions', 'order', 'recursive'));
```

Ilustración 7 Ejemplo de búsqueda en la BD /app/Controllers/FlatsController.php

- **Guardados recursivos:** al permitir consultas anidadas, CakePHP también permite guardar información de forma recursiva y múltiple. Con el método *saveAll* que recibe como parámetro un *hash*, en el que las llaves son los nombres de los modelos y el valor de cada entrada es un *array*asociativocon los campos que se quieren guardar.

CakePHP automáticamente guardará los registros y actualizará las claves ajenas según la especificación de cada modelo. El método *saveAll* funciona en modo transaccional, es decir que valida todos los campos, y solo guarda en la BD, si todos los campos pasan la validación especificada en los modelos respectivos. El modo transaccional ayuda a mantener la integridad de la BD y a evitar inconsistencias en los datos.

En la Ilustración 8 podemos ver un ejemplo de guardado múltiple y recursivo de los inmuebles, junto con sus datos relacionados. Complementado el guardado con el parámetro *deep* el cual añade anidación y el parámetro *validate* que comprueba primero la validación de los datos antes de proceder al guardado.

```
if ($this->Flat->saveAll($save, array('deep' => true, 'validate' => true))){
    $this->Session->setFlash(__('Inmueble guardado'));
} else{
    $this->Session->setFlash(__('Error al guardar el Inmueble'));
}
```

Ilustración 8 Ejemplo de guardado recursivo /app/Controllers/FlatsController.php

Otras características del *framework* que en mayor o menor medida se han aprovechado en el desarrollo del proyecto.

- Compatible con PHP4 y PHP5.
- CRUD integrado para la interacción con la base de datos.
- Despachador de peticiones (*dispatcher*), con URLs, rutas personalizadas y limpias.
- Plantillas rápidas y flexibles (sintaxis de PHP y *helpers*).
- Ayudantes para AJAX, JavaScript, formularios HTML.
- Componentes predefinidos Email, *Cookie*, Seguridad, Sesión y Manejo de solicitudes.

Hace tiempo que el desarrollo de un proyecto de software complejo no se desarrolla desde cero, debido a que la elección de un *framework* de desarrollo para el proyecto nos proporciona una gran ayuda, en por ejemplo la generación del código, la estructura, la modularidad, en la implementación de medidas de seguridad y del registro de eventos. Todas estas ayudas llevarían mucho tiempo implementarlas desde cero, incrementando así el coste del desarrollo.

2.2.2 JQuery

JQuery [11] es una librería JavaScript, rápida, pequeña y con gran diversidad de funciones predefinidas. Esta hace que el HTML sea transversal y su manipulación, gestión de eventos, animaciones y eventos Ajax sean más sencillos. La librería proporciona una API para su manipulación. La librería extiende una gran compatibilidad en los navegadores más actuales.

Desarrollo e implementación – CAPÍTULO 3

La elección de la librería JQuery fue la de proveer al usuario de una mejor experiencia en la interfaz de la plataforma.

2.2.2.1 GMap3

Es un potente *plugin* para crear y administrar los mapas de Google, que se apoya en las librerías de JavaScript: JQuery y Google Maps Library. GMap3 permite manipular tantos marcadores como los objetos del mapa, permitiendo así asociarlos a datos personalizados y eventos [12].

La utilización del *plugin* se debe a las limitaciones que se encontraron con la librería oficial de Google Maps, que no da soporte a funcionalidades avanzadas, como el filtrado de marcadores, rutas dinámicas entre puntos personalizados, búsqueda y posicionamiento de direcciones dinámicas. La principal utilidad ha sido la creación del “Mapa inteligente de los inmuebles”, en la Ilustración 9 podemos ver un ejemplo del mapa.

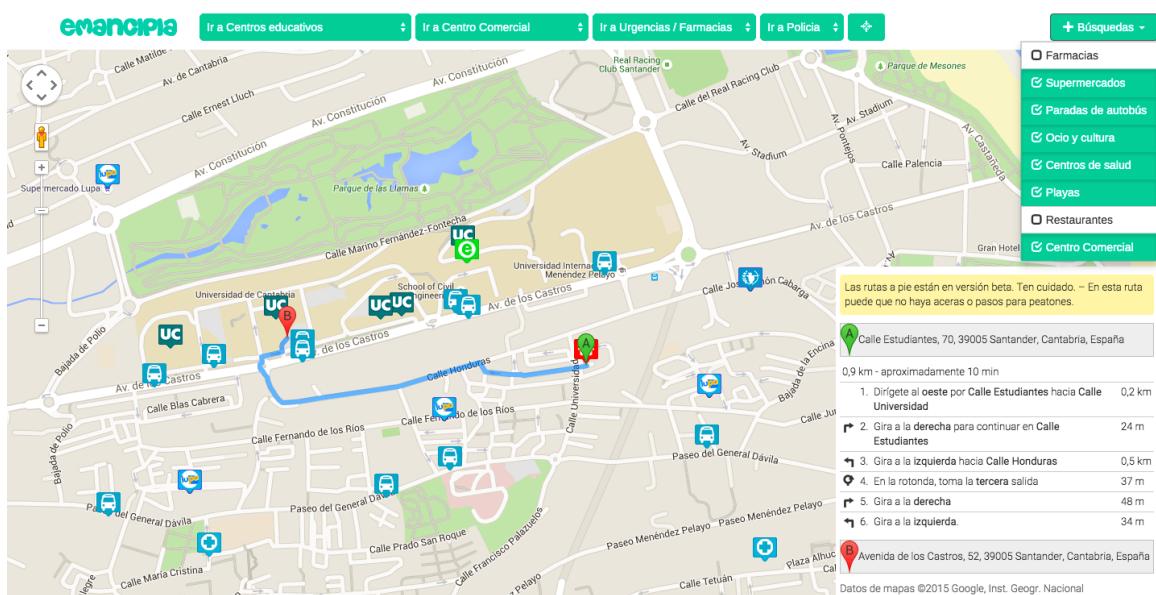


Ilustración 9 Mapa inteligente accesible desde el buscador.

2.2.2.2 html2canvas

html2canvas es un *script* que nos permite generar capturas de pantalla en distintos formatos de imagen, tomando los elementos HTML y CSS de una web. El *script* recorre a través del DOM que se carga en la página, recopila toda la información de la página y construye una representación de ella [13].

Es importante recalcar que el *script* no realiza una captura de pantalla si no que interpreta el HTML y el CSS de la página, y crea una imagen en un formato que se define en la salida. Ya que construye una interpretación del código no es 100% fiable y tiene las restricciones de las distintas reglas CSS que es capaz de procesar.

```
$(document).ready(function() {
    var delay=2000;
    setTimeout(function(){
        html2canvas($("#image"), {
            onrendered: function(canvas) {
                $('#area').attr('src', canvas.toDataURL('image/png', 1.0));
                $('#image').addClass('hidden');
            }
        });
    },delay);
});
```

Ilustración 10 Fragmento de código JS de la librería html2canvas.

2.2.3 Bootstrap

Bootstrap es un *framework front-end* para el desarrollo adaptativo, tanto en plataformas móviles, como en el desarrollo web, basado en HTML, CSS y JS. Comenzó en 2011 como una solución interna para las inconsistencias en el desarrollo dentro del equipo de ingeniería de Twitter Inc. Posteriormente fue lanzada al público como proyecto *Open Source* en Github bajo licencia del MIT [14].

La elección de este *framework* para el desarrollo *front-end* del proyecto se debe a la facilidad de completar un estilo homogéneo en toda la plataforma y de desarrollar un diseño adaptado a dispositivos móviles.

2.2.4 MySql

MySql es el sistema de gestión de bases de datos relacionales, multihilo y multiusuario más popular para el desarrollo de proyectos web, programado en los lenguajes C y C++ con licencia GPL y desarrollado por MySQL AB, ahora en la propiedad de la empresa Sun Microsystems [15].

La elección de este gestor se debe a su alta compatibilidad con los servicios de alojamiento web que podemos encontrar en el mercado, la amplia comunidad y documentación que podemos encontrar, nota de esto es la existencia de múltiples *forks* del sistema adaptados a necesidades más específicas, un ejemplo, es el fork de MySql desarrollado y utilizado por Twitter Inc.

Para la manipulación de la base de datos, tanto de su estructura como de los datos, se utiliza phpMyAdmin como podemos ver en la Ilustración 12, herramienta escrita en PHP que permite la fácil manipulación de sistemas MySql. Como motor de almacenamiento MySql se utiliza InnoDB, por su soporte de transacciones, el bloqueo de registros y permitir las características ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), garantizando así la integridad de nuestras tablas.

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño
labels	★ Estructura Insertar Eliminar	5	InnoDB	utf8_general_ci	16 KB
bodies	★ Estructura Insertar Eliminar	2,203	InnoDB	utf8_general_ci	14.6 MB
e2lastchanges	★ Estructura Insertar Eliminar	9	InnoDB	utf8_general_ci	16 KB
billings	★ Estructura Insertar Eliminar	4	InnoDB	utf8_general_ci	16 KB
e2chargesstates	★ Estructura Insertar Eliminar	1,726	InnoDB	utf8_general_ci	160 KB
occupancies	★ Estructura Insertar Eliminar	2,960	InnoDB	utf8_general_ci	448 KB
reportsdepartures	★ Estructura Insertar Eliminar	0	InnoDB	utf8_general_ci	16 KB
consumptionbalances	★ Estructura Insertar Eliminar	1,263	InnoDB	utf8_general_ci	144 KB
templates	★ Estructura Insertar Eliminar	11	InnoDB	utf8_general_ci	48 KB
departuresreports	★ Estructura Insertar Eliminar	40	InnoDB	utf8_general_ci	16 KB
compensates	★ Estructura Insertar Eliminar	194	InnoDB	utf8_general_ci	16 KB
tableconsumptions	★ Estructura Insertar Eliminar	36	InnoDB	utf8_general_ci	16 KB
loginrecords	★ Estructura Insertar Eliminar	804	InnoDB	utf8_general_ci	160 KB
departurescharges	★ Estructura Insertar Eliminar	609	InnoDB	utf8_general_ci	64 KB
emailusers	★ Estructura Insertar Eliminar	8,306	InnoDB	utf8_general_ci	688 KB
paypals	★ Estructura Insertar Eliminar	4	InnoDB	utf8_general_ci	16 KB

Ilustración 11 Extracto de la BD utilizando phpMyAdmin.

2.2.5 Mandrill

Mandrill es una herramienta desarrollada por MailChimp, plataforma mundial de marketing *on-line*. La función de dicha herramienta es la de sustituir al servidor web de correo tradicional aportando fiabilidad y rapidez en el envío, así como otras nuevas funcionalidades [16].

Entre en las principales funcionalidades que nos aporta podemos destacar:

- Una infraestructura global distribuida, teniendo servidores alrededor del mundo, reduciendo así la latencia e incrementando el periodo del envío de los emails.
- Autentificación automática de los emails.
- Seguimiento de *clicks* y apertura de los emails por parte de los destinatarios.
- Registro de emails rechazados o rebotados.
- Registro geográfico de apertura del email.
- Identificación del dispositivo utilizado en su apertura.

Mandrill nos aporta un envío más fiable y rápido de los correos electrónicos a través de una integración SMTP, como vemos en la Ilustración 13. Además ofrece la posibilidad de descarga de un archivo con el registro de estado de los emails enviados, el cual se exporta en formato csv, lo cual nos permite integrarlo con la plataforma y elaborar seguimientos y estrategias de mercado futuras.

```
public $mandril = array(
    'transport' => 'Smtp',
    'host' => 'smtp.mandrillapp.com',
    'port' => 587,
    'username' => 'sergio@emancipia.net',
    'password' => '*****',
    'timeout' => 60,
    'client' => null,
    'log' => false,
    'charset' => 'utf-8',
    'headerCharset' => 'utf-8',
);
```

Ilustración 12 Integración del protocolo SMTP con la plataforma

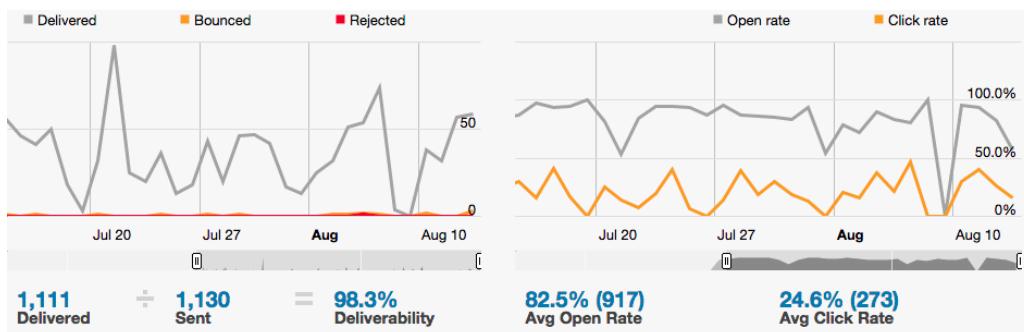


Ilustración 13 Informe de seguimiento de Julio y Agosto 2015 en Mandrill.

2.2.6Métodos de pagos virtuales

Uno de los más importantes objetivos de la plataforma es implementar métodos de pagos virtuales e integrarlos. De entre los muchos métodos de pagos virtuales se seleccionaron dos opciones, PayPal por ser el medio de pago virtual más utilizado y el TPV del Banco Sabadell al ya trabajar Emancipia con esta entidad Bancaria.

Formulario de Reserva

Ilustración 14 Formulario de reserva con las modalidades del pago.

2.2.6.1 PayPal

PayPal es una empresa de comercio electrónico estadounidense creada en 1998, fundada por Peter Thiel, que nació de la fusión de Confinity y X.com. Después de un rápido crecimiento fue comprada por eBay en octubre del 2002 pasando a ser la principal forma de pago del portal de subastas.

Aunque PayPal no es una entidad bancaria está sujeta a legislación por parte de la Unión Europea. La sede central de la empresa está en San José, California (EEUU). Aunque en los últimos años le han salido fuertes competidores, AliPay, Google Checkout o AlertPAY a día de hoy sigue siendo líder de transacciones virtuales con más de 169 millones de clientes en todo el mundo y operando en más de 20 divisas distintas [17].

PayPal mediante una API nos ofrece, una plataforma de pago segura, con ella se pueden realizar transacciones a la mayor parte del mundo, en distintas divisas y por una pequeña comisión.

2.2.6.2 Pasarela de pago segura del Banco Sabadell

Se optó por la implantación del pago por tarjeta a través del Banco Sabadell ya que Emancipia ya trabaja con la entidad. La integración se realizó mediante formularios HTML, recibiendo la respuesta de la transacción del banco en un documento XML mediante el método POST. Cabe destacar que se implementó la compra electrónica segura que, bajo los protocolos internacionales Verifiedby VISA y MasterCard SecureCode asegura una amplia seguridad y protección en los pagos. La plataforma no almacena los datos de las tarjetas de los usuarios, ni asocia dichas tarjetas a los usuarios, sino que se tratan como datos de carácter personal amparados por la Ley Orgánica 15/1999 de Protección de Datos, más conocida como la LOPD [18].

2.3 Metodología de desarrollo

En cualquier proyecto de software un factor importante y necesario es el uso y la correcta elección de una metodología de trabajo, donde se definirán cada una de las etapas del proceso de producción. Con la elección de una metodología conseguiremos mejorar nuestra productividad durante el desarrollo de la plataforma y la calidad del producto final presentado.

2.3.1 Metodologías Ágiles

A mediados de la década de 1990 surgió un movimiento contrario a las metodologías de desarrollo de software clásicas. El movimiento carecía de las características principales de las metodologías clásicas, muy estructuradas y estrictas. Las empresas cada vez más dinámicas, demandaban productos más flexibles con constantes

evoluciones de los requisitos durante su desarrollo y una temprana entrega de prototipos con funcionales parciales. En la Ilustración 16 podemos ver una comparativa gráfica de una metodología tradicional frente a un desarrollo ágil.

Pero no fue hasta el año 2001 cuando se reunió la comunidad del software y se adoptó el nombre de métodos ágiles para estas nuevas metodologías. Las metodologías que se puedan consideran ágiles deben cumplir el “Manifiesto ágil” que no es más que una serie de principios que se pueden agrupar en: [19]

- Los individuos y sus interacciones, por encima de los procesos y las herramientas.
- El desarrollo de software que funciona, antes que la documentación exhaustiva.
- La colaboración con el cliente por encima de la negociación contractual.
- La respuesta al cambio por encima del planteamiento inicial.

El seguimiento de estos principios básicos para un desarrollo ágil tiene como objetivo minimizar las tareas secundarias sobre las principales, no perdiendo de vista el producto final; atender las peticiones del cliente; continuas entregas de funcionalidades parciales; adaptar el desarrollo a cambios en los requisitos; y favorecer la comunicación entre el equipo de desarrollo y el cliente durante el proceso de desarrollo.



Ilustración 15 Diagrama comparativo entre una metodología tradicional y el desarrollo ágil.

2.3.2 Metodología de desarrollo iterativo e incremental

La metodología elegida para el proyecto es la de un modelo de desarrollo iterativo e incremental [21] en la cual podemos entender a las iteraciones como miniproyectos, en cada una de las iteraciones se repite un proceso de trabajo similar, para proporcionar un resultado completo sobre el producto final, de manera que el cliente pueda obtener los beneficios del proyecto de forma incremental.

Para ello, cada requisito planteado en la captura se debe completar en una única iteración, se deberá realizar todas las tareas necesarias para completarlo incluyendo en dichas tareas las pruebas y documentación necesarias. Una vez acabada la iteración, parte del producto deberá estar preparado para ser entregado al cliente con un mínimo esfuerzo de preparación. De esta manera no se deja para el final del proyecto ninguna actividad arriesgada relacionada con la entrega de requisitos.

Como podemos ver en la Ilustración 17 en cada nueva iteración el equipo evolucionará el producto, incluyendo un nuevo requisito o ampliando un requisito ya planteado haciendo así una entrega incremental, a partir de los resultados completados en las iteraciones anteriores.

Un aspecto fundamental para el correcto desarrollo iterativo e incremental es la priorización de los requisitos en función de un valor cuantitativo aportado por el cliente [20].

Ventajas de la metodología

- Ofrece una retroalimentación temprana al cliente.
- Posibilidad de presentar un prototipo en las primeras iteraciones.
- El producto final no tiene que estar definido, si no que va definiendo a medida que avanzan las iteraciones.

Desventajas de la metodología

- La entrega de un prototipo inicial, aunque funcional puede carecer de robustez.
- Requiere un compromiso del cliente en la fase de desarrollo.

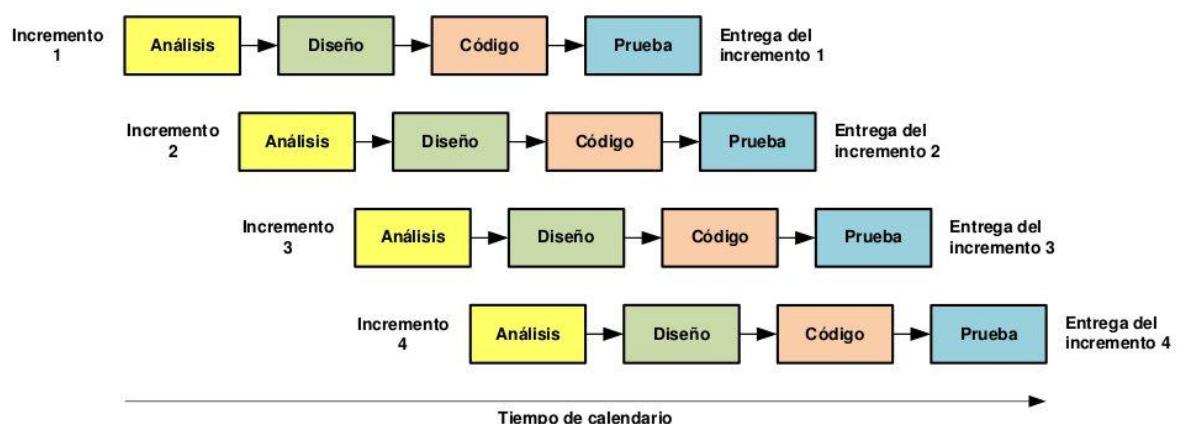


Ilustración 16 Esquema de un desarrollo iterativo e incremental

La elección del modelo de desarrollo iterativo e incremental se debe a las características del proyecto en cuanto al desconocimiento por parte del cliente del resultado final del mismo, así como el requerimiento de construir prototipos con funcionalidades primarias, antes de tener finalizado el proyecto completo. De la misma forma me permitió ir incluyendo o descartando miembros al equipo de trabajo a

medida que se avanzaba con el desarrollo proyecto sin tener que modificar la metodología de trabajo o las tareas ya asignadas.

2.3.3 Iteraciones planificadas

Para el seguimiento del día a día de las distintas iteraciones del proyecto, gestionar los cambiantes plazos de entrega de los prototipos, añadir nuevas funcionalidades, modificar las ya existentes y gestionar el equipo de trabajo se ha utilizado la herramienta de gestor de tareas Trello Ilustración 18.

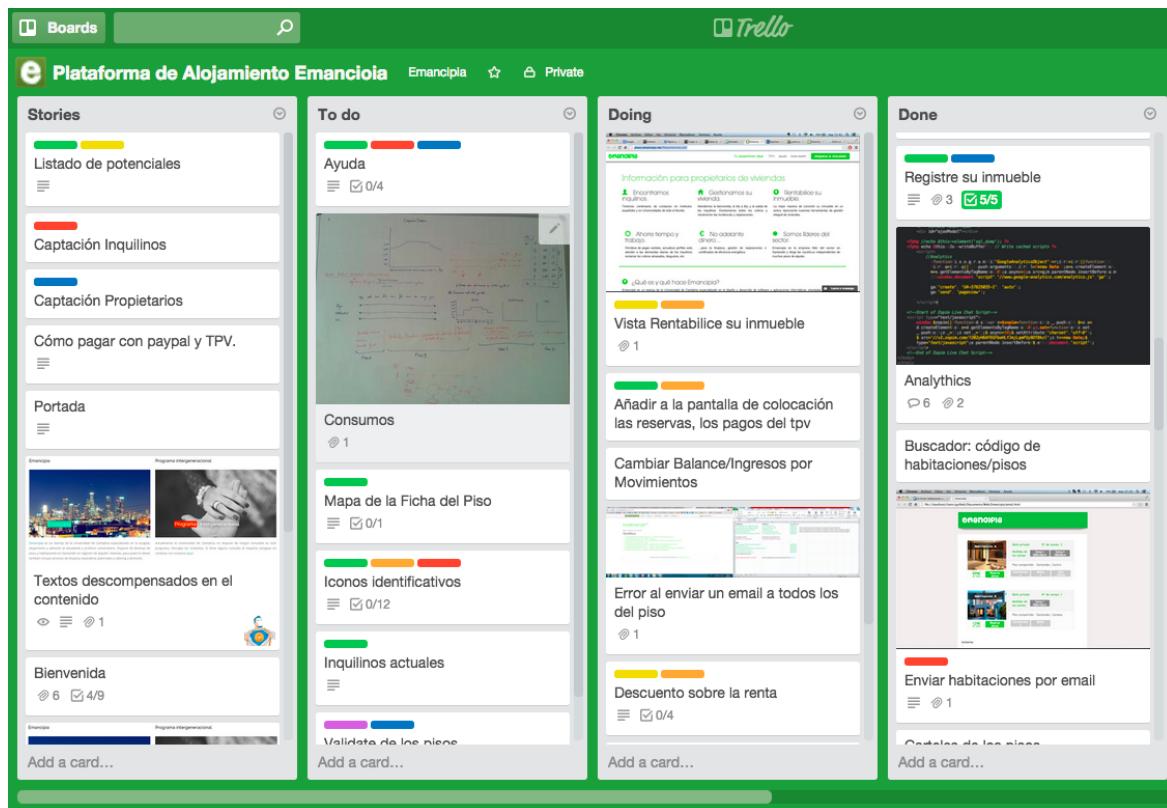


Ilustración 17 Captura de las tareas de un prototipo en la herramienta Trello.

Trello [22] es un gestor de tareas on-line que se basa en el método Kanban para la gestión de proyectos. Es un gestor muy potente que destaca por su versatilidad. Se compone de listas que se le agregan otras listas compuestas de tarjetas. A cada tarjeta se le pueden agregar otras listas o tarjetas además de imágenes, vídeos, documentos, etc. Permite el uso colaborativo entre diferentes miembros del equipo. Implementa un calendario en el que se puede marcar hitos en distintas tareas y ver su planificación. Además de permitir su integración con servicios externos como Google Calendar, Google Drive, Dropbox, OneDrive, etc.

CAPÍTULO 3

Desarrollo e implementación

En el presente capítulo y tras haber expuesto los objetivos (Capítulo 1), se procede a mostrar los requisitos software. Más adelante se muestra una visión global de la plataforma y una explicación más detallada de los componentes que la forman.

3.1 Captura y análisis de los requisitos

Una vez definidos los objetivos y entendiendo el contexto de la plataforma, se procede a la captura de requisitos, los cuales han sido obtenidos en varias reuniones con la empresa. Los requisitos son clave a la hora de entender y poder identificar correctamente las funcionalidades se esperan del producto final.

Los requisitos se han ido modificando durante el tiempo de desarrollo adecuándose este hecho a la metodología empleada y a continuación se muestran únicamente los requisitos finales e implementados en la plataforma.

3.1.1 Requisitos funcionales

Los requisitos funcionales (RF) definen la naturaleza del funcionamiento de la plataforma, es decir, cómo esta interacciona con su entorno y la extensión de su funcionamiento completo. Después de la obtención de los objetivos se decidió separar la plataforma en tres aplicaciones separándolos por el usuarioprincipal al que atienden. A continuación se expone una lista con los requisitos funcionales de este proyecto.

1. Gestor interno para los administradores de Emancipia.
2. Portal de búsqueda de inmuebles público.
3. Área de cliente para los usuarios registrados de Emancipia.

3.1.1.1 Requisitos funcionales del gestor interno.

ID	Descripción del requisito
RF001	El sistema será de acceso privado y estará protegido por un inicio de sesión.

Desarrollo e implementación – CAPÍTULO 3

RF002	El sistema sólo será accesible a usuarios con rol de administrador.
RF003	Se utilizará el correo electrónico del usuario para iniciar sesión en la plataforma siendo identificador del usuario.
RF004	El sistema caducará la sesión del usuario por un periodo de inactividad.
RF005	Solo los usuarios administradores darán de alta a otros administradores.
RF006	El sistema permitirá distintos roles de usuario.
RF007	En la vista de perfil de un usuario se mostrará su perfil básico y un resumen de sus datos relacionados.
RF008	Los datos relacionados de un usuario serán datos bancarios, personas de contacto, documentación, estudios e información personal.
RF009	El sistema actuará como servidor de correo electrónico saliente.
RF010	El sistema implementará plantillas predefinidas HTML para el correo electrónico.
RF011	El sistema implementará el envío de habitaciones e inmuebles en plantillas HTML predefinidas.
RF012	El sistema implementará el envío de archivos adjuntos en los correos electrónicos.
RF013	Las plantillas HTML deberán de visualizarse correctamente en dispositivos móviles.
RF014	El sistema implementará firmas de los usuarios administradores para las plantillas de los correos electrónicos.
RF015	El sistema implementará un sistema de comprobación de recepción de correos electrónicos.
RF016	El sistema implementará un gestor de cobros.
RF017	El sistema calculará mes a mes los cobros del alquiler en función de los días del alojamiento.
RF018	El sistema calculará la comisión del servicio prestado sobre la renta cobrada del alojamiento.
RF019	El sistema gestiona los cobros a los propietarios en base a lo ingresado por el concepto de renta y la comisión del servicio prestado.
RF020	El sistema implementará la gestión de los consumos derivados del alquiler.
RF021	El sistema calculará la ocupación diaria de los inquilinos en relación a la ocupación del inmueble.
RF022	El sistema calculará el importe de los consumos individualmente en relación a la ocupación de los inquilinos en el inmueble.

Desarrollo e implementación – CAPÍTULO 3

RF023	El sistema calculará una estimación de consumos en base al consumo del inmueble y la ocupación de los inquilinos.
RF024	El sistema compensará la estimación de los consumos, una vez obtenidos los consumos reales del inmueble.
RF025	El sistema implementará una liquidación del alojamiento, estimando los gastos futuros del alquiler.
RF026	El sistema implementará un gestor de recibos.
RF027	El sistema implementará el envío de recibos por correo electrónico con plantillas HTML predefinidas.
RF028	El sistema implementará herramientas de verificación de los cobros realizados a los usuarios.
RF029	El sistema implementará un balance del estado de los pagos del usuario.
RF030	El sistema implementará un gestor de llegadas programadas.
RF031	El sistema implementará avisos sobre las llegadas programadas no atendidas.
RF032	El sistema implementará un gestor de salidas programadas.
RF033	El sistema implementará avisos sobre las salidas programadas no atendidas.
RF034	El sistema implementará un gestor de las preferencias de alojamiento.
RF035	El sistema implementará avisos sobre las preferencias de alojamiento no atendidas.
RF036	El sistema implementará una vista gráfica de la disponibilidad de los inmuebles y habitaciones.
RF037	El sistema implementará un gestor de marcadores geolocalizados clasificándolos en diferentes categorías.
RF038	El sistema implementará la búsqueda masiva de marcadores filtrados por categorías.

3.1.1.1.2 Requisitos funcionales del buscador

ID	Descripción del requisito
RF039	El sistema será de acceso público, no requerirá de registro.
RF040	El sistema implementará la internacionalización del contenido.
RF041	El usuario podrá realizar búsquedas tanto de pisos completos como de habitaciones privadas pudiendo filtrar por sus características.
RF042	El buscador deberá de ser accesible desde la portada con un mínimo de parámetros de búsqueda.

Desarrollo e implementación – CAPÍTULO 3

RF043	El sistema geolocalizará los resultados de la búsqueda en un mapa interactivo habilitado en la vista del buscador.
RF044	El sistema posicionará el mapa en un punto equidistante respecto a los resultados de la búsqueda.
RF045	El sistema permitirá realizar búsquedas por todas las características siendo sólo obligatoria la elección entre piso o habitación en la búsqueda.
RF046	El sistema ordenará la búsqueda de resultados según un índice interno.
RF047	El sistema calculará la renta con un máximo y un mínimo en la búsqueda de resultados.
RF048	El sistema paginará los resultados de las búsquedas.
RF049	El sistema mostrará una búsqueda simple y otra avanzada para las búsquedas.
RF050	El sistema dispondrá de una sección para guardar pisos o habitaciones llamada favoritos.
RF051	El sistema ofrecerá una vista individual con las características del piso o de la habitación.
RF052	El usuario en la vista individual podrá realizar consultas mediante un formulario.
RF053	El sistema generará una respuesta estándar al correo electrónico del usuario con el estado de la consulta.
RF054	El sistema permitirá visualizar contenido multimedia en la vista individual.
RF055	El sistema geolocalizará el piso en la vista individual, así como los marcadores más cercanos a su posición.
RF056	El sistema mostrará un calendario anual con las fechas disponibles para su reserva.
RF057	El usuario podrá elegir las fechas en las que tiene previsto reservar y proceder a la reserva una vez elegidas las fechas.
RF058	El sistema filtrara las fechas de la reserva con otras reservas y ocupaciones evitando solapamientos entre las fechas.
RF059	El sistema controlará que la reserva de una habitación en unas fechas desactive la reserva del inmueble completo en esas mismas fechas.
RF060	El sistema implementará métodos de pagos seguros para el usuario.
RF061	El usuario podrá realizar el pago de la reserva mediante diferentes formas de pago: Tarjeta de Crédito, Paypal y Transferencia Bancaria.
RF062	El sistema obligará a completar todos los campos especificados en la reserva para continuar.
RF063	El sistema enviará un correo electrónico al usuario una vez completado el formulario de reserva.

Desarrollo e implementación – CAPÍTULO 3

RF064	El sistema no aceptará reservas realizadas fuera de unas fechas previamente definidas.
RF065	El sistema podrá verificar el pago de las pasarelas de pago.
RF066	El sistema bloqueará las fechas de las reservas, en las cuales se haya confirmado su pago para evitar futuros solapamientos.
RF067	El usuario en la vista individual podrá añadir a favoritos el inmueble o habitación.
RF068	El usuario podrá compartir la vista individual en las principales redes sociales.
RF069	El usuario tendrá acceso a un mapa inteligente centrado en la ubicación del inmueble.
RF070	En el mapa inteligente se destacarán distintos puntos de interés, así como un sistema de filtrado de dichos puntos accesible al usuario.
RF071	El sistema implementará en el mapa inteligente el poder crear rutas entre puntos del mapa, detallando las indicaciones de la ruta creada.
RF072	El usuario podrá acceder al mapa inteligente desde los resultados de la búsqueda.
RF073	El sistema implementará un formulario con las preferencias de alojamiento del usuario.
RF074	El sistema registrará al usuario comprobando antes si el usuario ya está dado de alta en la plataforma.
RF075	El sistema responderá al correo electrónico del usuario con una respuesta estándar.
RF076	El sistema implementará campos obligatorios en el formulario de alojamiento.
RF077	El sistema no permitirá el envío del formulario hasta aceptar el aviso legal.
RF078	El sistema implementará anti-robots (captcha) para evitar el spam generado en específicos formularios.

3.1.1.3 Requisitos funcionales del área de cliente

ID	Descripción del requisito
RF079	El sistema será de acceso privado y estará protegido por un inicio de sesión.
RF080	El sistema sólo será accesible a usuarios ya registrados en el sistema.
RF081	El sistema no permitirá el registro de nuevos usuarios.
RF082	El sistema implementará un sistema de recordatorio de contraseña.
RF083	El sistema implementará la internacionalización del contenido.
RF084	El sistema implementará un acceso de administrador a la aplicación.
RF085	El sistema implementará un log de acceso de los usuarios no administradores.

Desarrollo e implementación – CAPÍTULO 3

RF086	El sistema visualizará el contenido en el idioma de preferencia del usuario.
RF087	El sistema permitirá realizar modificaciones en los datos personales del usuario.
RF088	El sistema no permitirá la modificación del email del usuario.
RF089	El sistema registrará las modificaciones realizadas por los usuarios.
RF090	El sistema proveerá al usuario de una vista que estará compuesta por un resumen de su información: datos personales, alquileres, facturación y correos electrónicos.
RF091	El usuario podrá subir ficheros a la aplicación.
RF092	El sistema mostrará un registro de todos los correos electrónicos que se le han enviado al usuario desde la plataforma.
RF093	El usuario podrá ver un desglose de todos sus alquileres.
RF094	El usuario en la vista de su alquiler se podrá descargar su contrato.
RF095	El usuario en la vista de su alquiler se le mostrará información relativa a su recogida y llegada.
RF096	El usuario en la vista de su alquiler podrá ver los recibos correspondientes a las fechas de su alquiler.
RF097	El usuario en la vista de su alquiler podrá ver un desglose de la renta y los gastos. La visualización será también en modo gráfico.
RF098	El usuario en la vista de su alquiler podrá ver y descargarse las facturas reales del inmueble correspondientes a los consumos en las fechas de su alquiler.
RF099	El usuario en la vista de su alquiler podrá visualizar un desglose de la liquidación generada de su contrato.
RF100	El usuario en la vista de facturación visualizará un desglose de los recibos, ingresos y cargos generados de todos sus alquileres.
RF101	El usuario tendrá acceso a una pasarela de pago segura dentro de la aplicación.
RF102	El sistema recibirá notificaciones del pago de la pasarela comprobando si ha sido correcto.
RF103	El sistema generará respuestas automáticas vía correo electrónico con el resultado del pago.
RF104	El usuario podrá cambiar la visualización del idioma en todo momento.

RF105	El usuario podrá enlazar su cuenta de facebook con su cuenta de usuario.
-------	--

3.1.2 Requisitos no funcionales

Los requisitos no funcionales desempeñan un rol fundamental en el desarrollo, estos se fundamentan en características o cualidades del sistema que establecen restricciones o juzgan el rendimiento del mismo, en lugar de funcionalidades específicas.

Estos requisitos los podemos clasificar en diferentes categorías, algunas de las cuales son: seguridad, disponibilidad, accesibilidad, escalabilidad, extensibilidad, tolerancia a fallos, rendimiento, portabilidad, usabilidad y compatibilidad. Los requisitos no funcionales son los que nos ayudan a elegir una y otra implementación de la arquitectura de software.

A continuación se procede a listar los distintos requisitos no funcionales agrupándolos por la categoría a la que pertenecen.

RNF Seguridad		
ID	Descripción del Requisito	Importancia
RNF001	El pago deberá de realizarse mediante una plataforma de pago segura.	Alta
RNF002	El sistema deberá comunicarse con el banco para comprobar los pagos realizados desde la plataforma.	Alta
RNF003	Los archivos subidos a la plataforma deberán de cumplir requerimientos de peso y tipo de archivo.	Alta
RNF004	La identificación del usuario debe de ser segura y fiable.	Alta

RNF Accesibilidad		
ID	Descripción del Requisito	Importancia
RNF005	El sitio web deberá de ser compatible con la mayoría de las versiones de los navegadores más utilizados del mercado.	Media
RNF006	El sitio web deberá tener una accesibilidad aceptable en los principales navegadores de dispositivos móviles.	Media

RNF Mantenibilidad y Extensibilidad		
ID	Descripción del Requisito	Importancia

RNF007	El sistema deberá de seguir un diseño modular para facilitar tanto su mantenimiento como su ampliación.	Alta
RNF008	El servidor permitirá su ampliación añadiendo nuevos planes de recursos.	Alta

RNF Normativo		
ID	Descripción del Requisito	Importancia
RNF009	La plataforma deberá cumplir con la Ley Orgánica 15/1999 de Protección de datos de carácter personal (LOPD)	Alta

RNF Usabilidad		
ID	Descripción del Requisito	Importancia
RNF010	El sistema deberá de cumplir la regla de los 3 <i>clicks</i> .	Media

3.2 Especificación de los casos de usos

Una vez realizada la captura y definición de los requisitos funcionales y no funcionales del sistema, para continuar con el desarrollo se procede a detallar más en profundidad las funcionalidades mediante la especificación de los casos, los cuales se definirán mediante diagramas y plantillas.

3.2.1 Diagramas de casos de usos

Los diagramas de casos de uso [23] definen un comportamiento deseado del sistema, representan gráficamente los requisitos funcionales. En este apartado no se explicarán las funciones sino como deberán de comportarse. Los diagramas se componen de secuencias de acciones, actores del sistema y variantes que extienden de los casos.

Primero se muestra el diagrama general de la plataforma, para más adelante ir detallando más en profundidad los distintos escenarios que la componen.

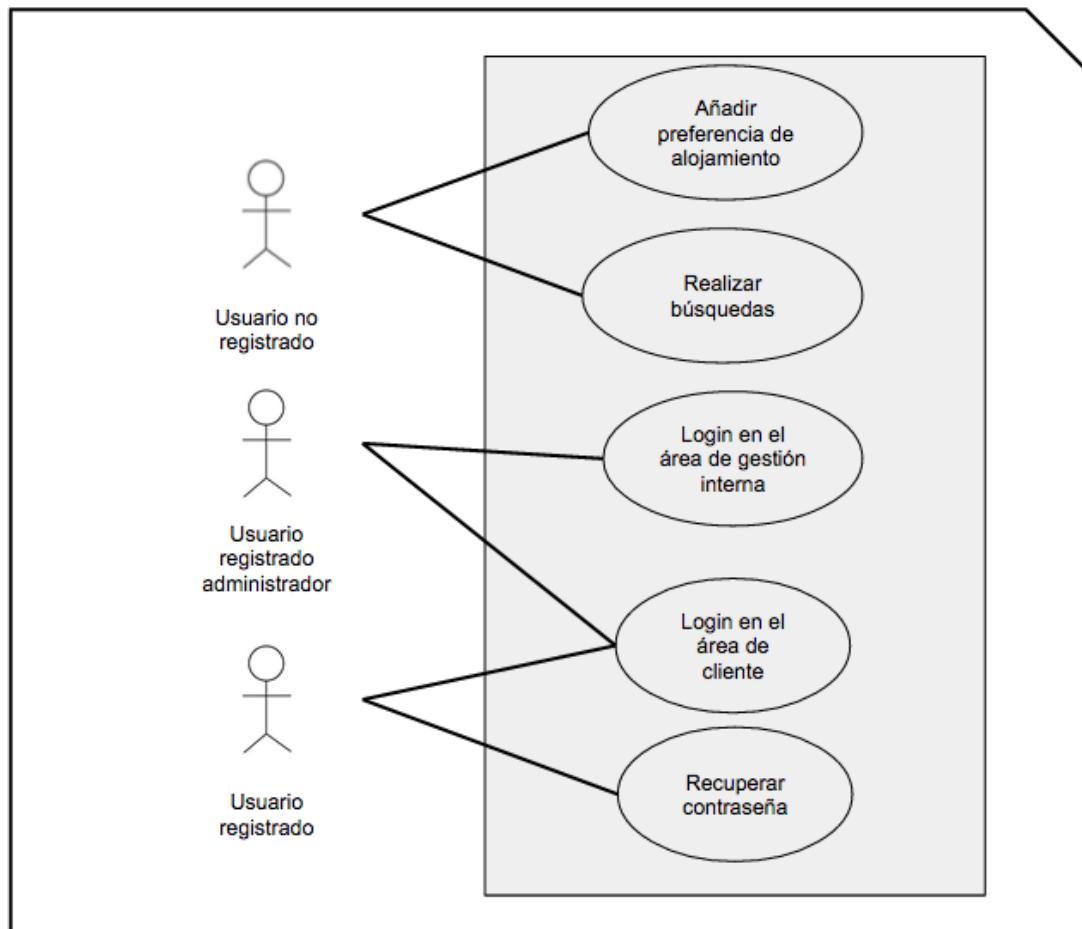


Ilustración 18 Diagrama de casos de uso general.

En la Ilustración 19 se muestra el diagrama de casos de uso general se muestra los distintos actores que interactúan con el sistema y los accesos que tienen a las tres aplicaciones que forman la plataforma de Emancipia.

La plataforma se divide en tres aplicaciones: Buscador, Gestor y Área cliente. Respecto a los actores que interactúan con el sistema los podemos clasificar en:

- **Usuario no registrado:** cualquier usuario que acceda a la plataforma
- **Usuario registrado administrador:** usuarios ya registrados con rol de administrador del sistema.
- **Usuario registrado:** usuarios que ya previamente han sido dados de alta en la plataforma.

A continuación detallarán las aplicaciones del Área de cliente y del Buscador público. Con el fin de resumir la redacción del proyecto no se ha detallado la especificación del Área de gestión interna.

3.2.1.1 Especificación del Área de cliente

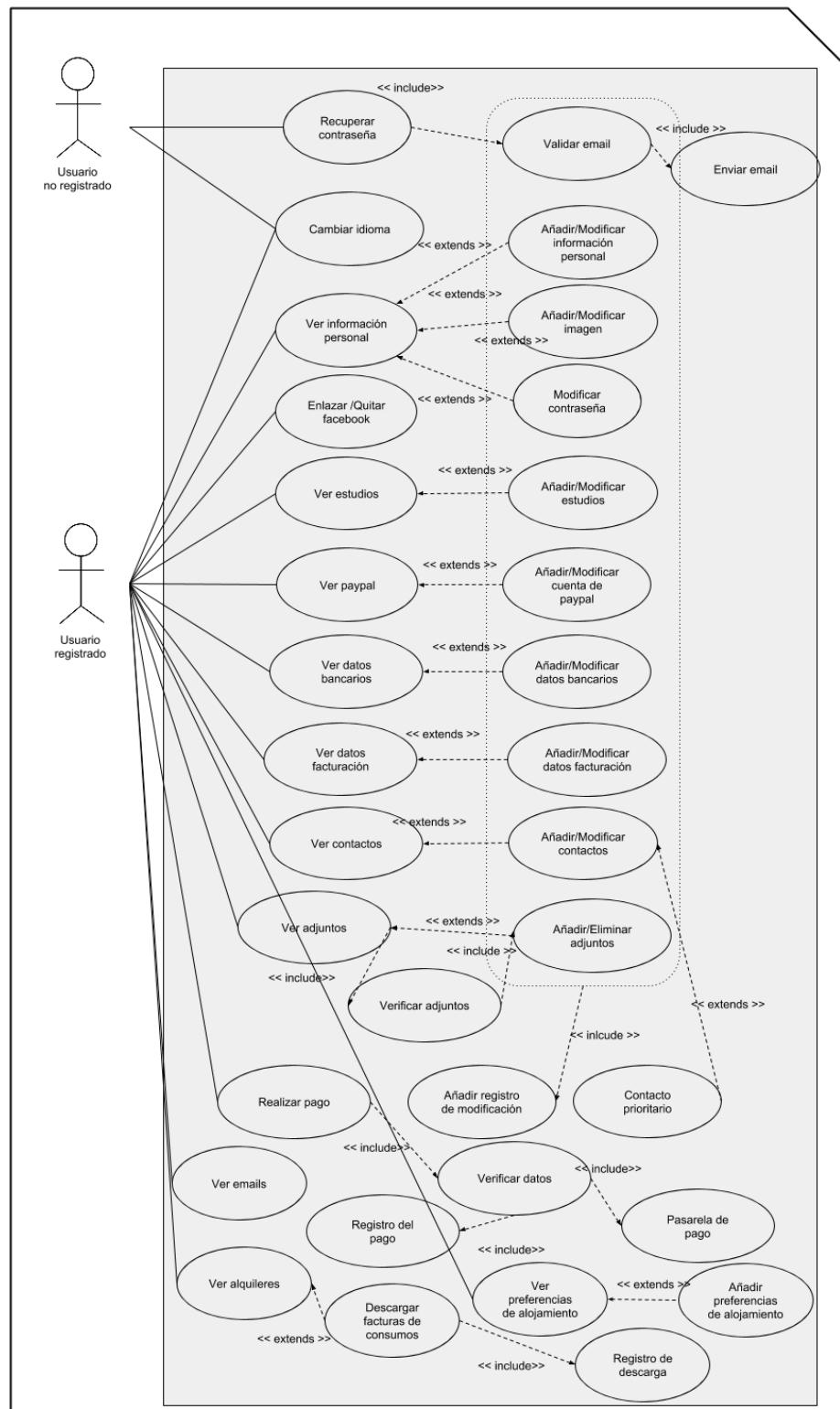


Ilustración 19 Diagrama de casos de uso del Área de cliente.

En la Ilustración 20 se muestra el diagrama de casos de usos que corresponde al área de cliente, podemos observar los distintos actores que interactúan con el sistema y los

Desarrollo e implementación – CAPÍTULO 3

diferentes casos de usos que se presentan. Podemos destacar el caso de uso Añadir registro de modificación que registra los cambios que realiza el usuario logeado manteniendo los registros antiguos para poder compararlos con los nuevos.

A continuación se detallan más en profundidad los casos de uso más relevantes mediante plantillas de casos de uso.

ID	P001
Nombre	Añadir/Modificar imagen
Descripción	El usuario cambia la imagen de su perfil personal en la aplicación.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario debe de estar logeado en la aplicación.
Flujo principal	<ol style="list-style-type: none">1. El usuario elige su imagen de perfil2. El sistema valida que la imagen es correcta y cumple unos parámetros definidos en el sistema.3. El sistema registra que el usuario ha realizado un cambio de su imagen de perfil.4. El sistema actualiza la imagen del perfil del usuario.
Flujo alternativo	<ul style="list-style-type: none">• Flujo alternativo 1: Imagen incorrecta, tamaño de imagen excedido. 2.- El sistema muestra un mensaje de error, indicando que el tamaño de imagen ha sido excedido, solo permitiendo imágenes de menos de 2MB.• Flujo alternativo 2: Imagen incorrecta, tipo de archivo incorrecto. 2.- El sistema muestra un mensaje de error, indicando que el tipo de archivo es incorrecto solo permitiendo archivos de tipo PNG y JPG
Postcondiciones	Borrado del servidor de la antigua imagen de perfil. Se actualiza la imagen.

Desarrollo e implementación – CAPÍTULO 3

ID	P002
Nombre	Modificar contraseña.
Descripción	El usuario cambia la contraseña de acceso a su perfil.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario debe de estar logeado en la aplicación.
Flujo principal	<ol style="list-style-type: none"> 1. El usuario introduce dos veces la nueva contraseña de acceso a la aplicación. 2. El sistema comprueba que las contraseñas son iguales. 3. El sistema verifica que la contraseña es correcta, según los parámetros definidos en el sistema. 4. El sistema modifica la contraseña de acceso al usuario. 5. El sistema registra que el usuario ha realizado un cambio de contraseña. 6. El sistema envía un email al usuario con los nuevos datos de acceso.
Flujo alternativo	<ul style="list-style-type: none"> • Flujo alternativo 1: Contraseñas distintas. 2.- El sistema muestra un mensaje de error, indicando que las contraseñas no coinciden. • Flujo alternativo 2: Contraseña incorrecta. 2.- El sistema muestra un mensaje de error, indicando que las contraseñas no cumplen con el nivel de seguridad básico.
Postcondiciones	Se modifica la contraseña de acceso a la aplicación.

ID	P003
Nombre	Recuperar contraseña
Descripción	El usuario ha olvidado su contraseña y solicita una nueva.
Actores primarios y secundarios	Usuario

Desarrollo e implementación – CAPÍTULO 3

Precondiciones	
Flujo principal	<ol style="list-style-type: none"> 1. El usuario introduce su email de acceso a la aplicación. 2. El sistema registra que el usuario ha realizado una petición de nueva contraseña. 3. El sistema modifica la contraseña del usuario. 4. El sistema envía un email al usuario con los nuevos datos de acceso a la aplicación.
Flujo alternativo	<ul style="list-style-type: none"> • Flujo alternativo 1: Email incorrecto <ol style="list-style-type: none"> 1. El sistema muestra un mensaje de error, indicando que el email no pertenece a ningún usuario de la aplicación y no deja continuar el caso de uso.
Postcondiciones	Se modifica la contraseña de acceso a la aplicación.

ID	P004
Nombre	Realizar pago
Descripción	El realiza un pago mediante la pasarela de pago.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario debe de estar logeado en la aplicación.
Flujo principal	<ol style="list-style-type: none"> 1. El usuario introduce el concepto del pago y la cantidad. 2. El sistema verifica los datos introducidos. 3. El usuario verifica que los datos del pago son correctos. 4. El sistema registra el intento del pago. 5. El usuario es redirigido a la pasarela de pago. 6. El sistema comprueba el pago mediante la respuesta de la pasarela de pago. 7. El sistema registra el pago.
Flujo alternativo	<ul style="list-style-type: none"> • Flujo alternativo 1: Datos inválidos <p>El sistema muestra un mensaje de error, indicando que los datos son inválidos y no deja continuar el caso.</p>

	<ul style="list-style-type: none">• Flujo alternativo 2: Datos incorrectos 3.- El usuario ha introducido mal los datos del pago. Vuelve al paso 1.
Postcondiciones	Se modifica el balance del usuario.

3.2.1.2 Especificación del buscador público

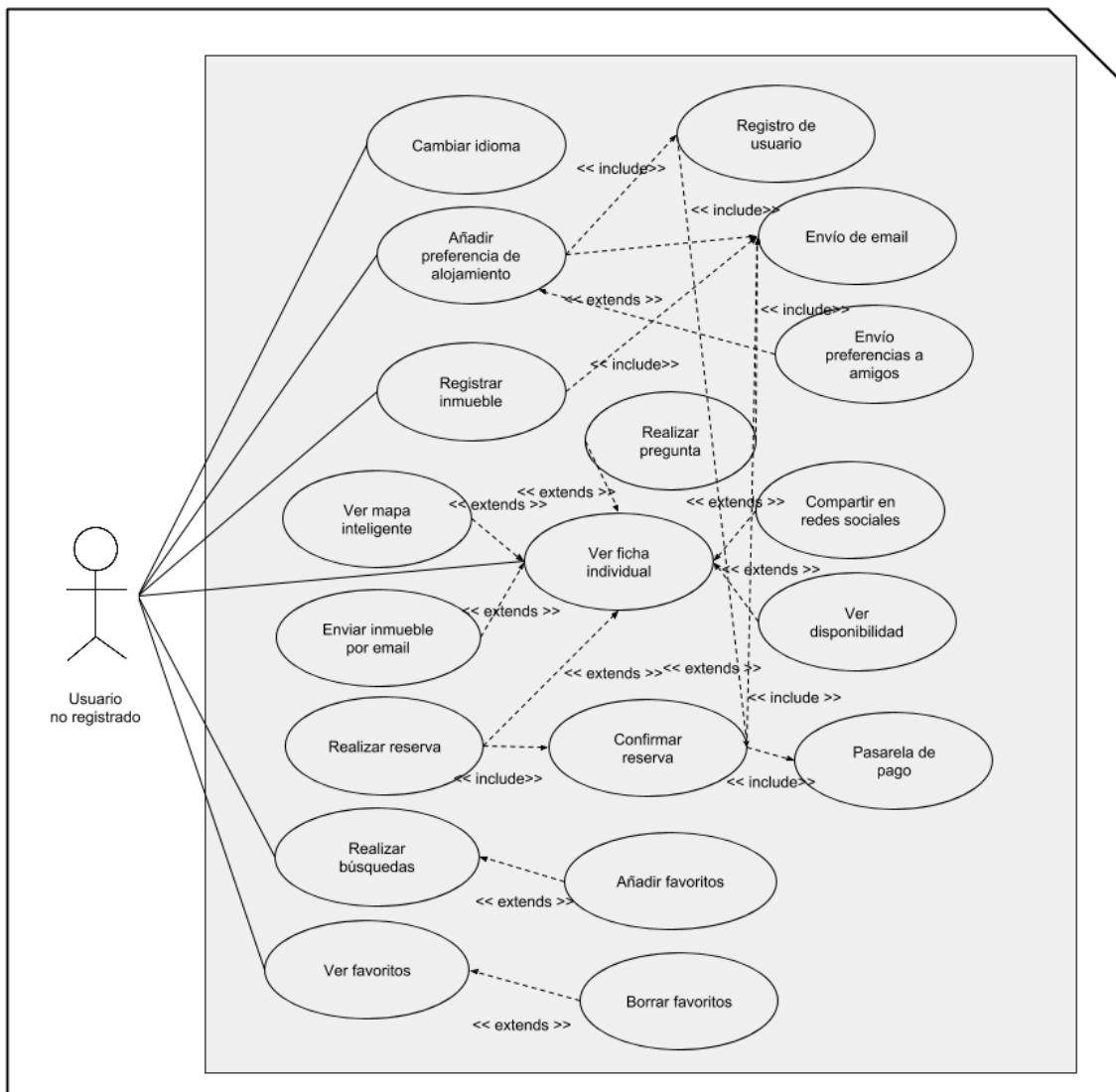


Ilustración 20 Diagramas de casos de uso del Portal buscador.

En la Ilustración 21 se muestra el diagrama de casos de uso del buscador, vemos que en todos los casos de uso el actor que interactúa no necesita de iniciar sesión. Podemos destacar el caso de uso confirmar reserva que atiende al RF9999 que especifica que solo se permitirán reservas si se completan todos los campos y las

Desarrollo e implementación – CAPÍTULO 3

fechas de la reserva están comprendidas entre unas dadas por el administrador de la plataforma.

A continuación se detallan más en profundidad los casos de uso más relevantes mediante plantillas de casos de uso.

ID	P001
Nombre	Realizar búsquedas
Descripción	El usuario busca mediante unos filtros resultados de inmuebles o habitaciones.
Actores primarios y secundarios	Usuario
Precondiciones	Se han insertado inmuebles ó habitaciones activos en la base de datos.
Flujo principal	<ol style="list-style-type: none">1. El usuario accede a la sección del buscador.2. El usuario modifica los filtros de búsqueda.3. Al usuario se le presentan los resultados de la búsqueda.4. Se geolocalizan los resultados de la búsqueda en el mapa.
Flujo alternativo	<ul style="list-style-type: none">• Flujo alternativo 1: No se producen resultados.<ol style="list-style-type: none">3. Al usuario se le muestra un mensaje que notifica que no se han encontrado resultados, pero el sistema mostrará unos resultados ya prefijados.4. Se geolocalizan los resultados prefijados en el mapa.
Postcondiciones	

ID	P002
Nombre	Realizar reserva
Descripción	El usuario realiza una reserva de su estancia en un inmueble ó habitación.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario ha seleccionado un inmueble ó habitación activa.
Flujo principal	<ol style="list-style-type: none">1. El usuario selecciona las fechas de su estancia.2. El sistema valida las fechas.3. El usuario introduce los datos requeridos para la reserva (nombre, email y modo de pago).4. El sistema valida los períodos de reserva.5. El usuario verifica la información de la reserva y el pago.6. El usuario procede al pago de la reserva.

Desarrollo e implementación – CAPÍTULO 3

	<p>7. El sistema verifica el pago de la reserva.</p> <p>8. El sistema registra la reserva.</p>
Flujo alternativo	<ul style="list-style-type: none"> • Flujo alternativo 1: Las fechas introducidas no son válidas <ol style="list-style-type: none"> 2. El sistema muestra un mensaje de error y retorna al usuario a que vuelva a introducir las fechas. • Flujo alternativo 2: Datos incompletos de la reserva. <ol style="list-style-type: none"> 3. El sistema muestra un mensaje de error e indica los datos incorrectos de la reserva. • Flujo alternativo 3: Periodos de reserva no predefinidos. <ol style="list-style-type: none"> 4. El sistema muestra un mensaje de periodos no predefinidos y termina la ejecución del caso de uso. • Flujo alternativo 4: Información incorrecta de la reserva. <ol style="list-style-type: none"> 5. El sistema permite al usuario modificar su información de pago.
Postcondiciones	<ul style="list-style-type: none"> • El sistema crea al usuario si este no existe. • El sistema da de baja el inmuebles o habitación en los períodos de reserva seleccionados. • El sistema envía un email al usuario con los datos de la reserva.

ID	P003
Nombre	Añadir preferencias de alojamiento
Descripción	El usuario rellena un formulario con sus preferencias de alojamiento.
Actores primarios y secundarios	Usuario
Precondiciones	
Flujo principal	<ol style="list-style-type: none"> 1. El usuario introduce las preferencias de su alojamiento. 2. El usuario envía el formulario. 3. El sistema envía un email confirmando la preferencia. 4. El usuario rellena datos extra de la preferencia de alojamiento. 5. El sistema permite al usuario el acceso a su área personal.
Flujo alternativo	<ul style="list-style-type: none"> • Flujo alternativo 1: Datos incompletos de la preferencia. <ol style="list-style-type: none"> 1. El sistema muestra un mensaje de error e indica los datos obligatorios de la preferencia de alojamiento.
Postcondiciones	<ul style="list-style-type: none"> • El sistema crea al usuario si este no existe. • El sistema envía un email a cada uno de los amigos con los que ha solicitado compartir piso el usuario. • El sistema avisará de una nueva preferencia a un responsable

Desarrollo e implementación – CAPÍTULO 3

	designado por el administrador.
--	---------------------------------

ID	P004
Nombre	Realizar pregunta
Descripción	El usuario realiza una pregunta sobre un inmueble o habitación.
Actores primarios y secundarios	Usuario
Precondiciones	El usuario ha seleccionado un inmueble ó habitación activa.
Flujo principal	<ol style="list-style-type: none">1. El usuario rellena los datos relativos a la pregunta.2. El usuario resuelve el captcha.3. El usuario envía la pregunta.
Flujo alternativo	<ul style="list-style-type: none">• Flujo alternativo 1: Captcha erróneo.<ol style="list-style-type: none">1. El sistema muestra un mensaje de error indicando el captcha erróneo.
Postcondiciones	<ul style="list-style-type: none">• El sistema crea al usuario si este no existe.• El sistema envía al usuario un extracto de su pregunta.• El sistema avisa de una nueva pregunta a un responsable designado por el administrador.

Diagrama de secuencia del caso de uso P002. Realizar reserva, Ilustración 22, en que se establece la conexión con el TPV del Banco Sabadell para realizar el pago on-line y la comprobación del éxito del pago.

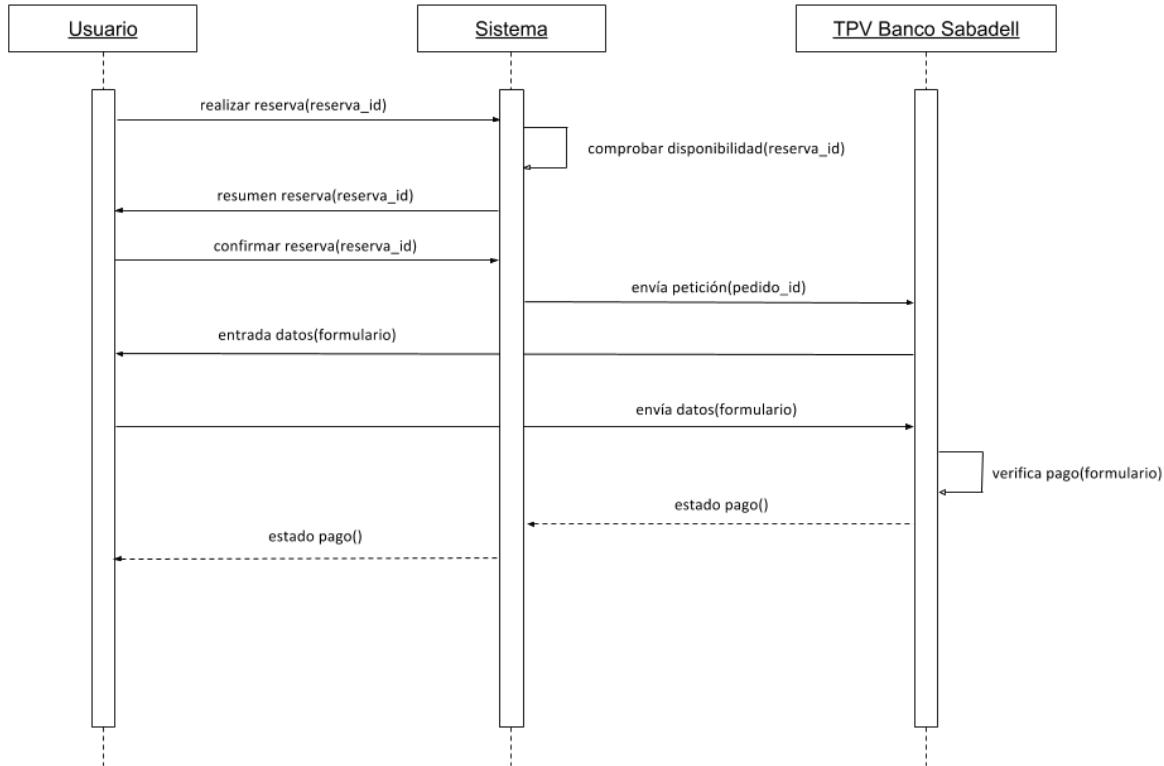


Ilustración 21 Diagrama de secuencia del caso de uso P002 Realizar reserva

3.3 Diseño del sistema

Para la descripción del diseño del sistema se ha seguido la certificación estándar ISO 12207, en la que se identifican dos tipos de diseño de software. El diseño arquitectónico y el diseño detallado.

3.3.1. Diseño arquitectónico

El diseño arquitectónico representa el más alto nivel de abstracción en la estructura del sistema, es decir los componentes y sus relaciones.

La arquitectura escogida para el proyecto es una arquitectura de tres capas, esta arquitectura separa la lógica del proyecto en capas y pone hincapié en cómo se interconectan entre sí. Al estar separado en capas lleva el desarrollo a varios niveles, de tal forma que al realizar cambios en una capa, las demás capas del sistema no se ven afectadas.

- Capa de presentación: será la encargada de presentar los datos del sistema para que el usuario final pueda visualizarlos así como permitir la comunicación entre el usuario final y el sistema. La comunicación entre capas sólo podrá realizarse con la capa de negocio.
- Capa de negocio: Esta capa será la encargada de controlar el cumplimiento de las reglas de negocio, y por lo tanto de realizar la negociación con las demás capas. En ella residen las funciones que ejecutará el sistema, que tras recibir la petición del usuario procedente de la capa de presentación, procesa la información y envía las respuestas, tras todo el proceso de comunicación a la capa de datos. La comunicación entre capas será con ambas capas, la de presentación para recibir las peticiones del usuario y con la de datos para realizar las solicitudes referentes a los datos necesarios requeridos por el usuario, solicitando la consulta o almacenamiento de estos.
- Capa de datos: Esta capa es la encargada de almacenar los datos del sistema, y de los usuarios. Estará formada por uno o varios sistemas gestores de bases de datos. La comunicación entre capas se realizará solo con la capa de negocio, para responder a las solicitudes provenientes de ella.

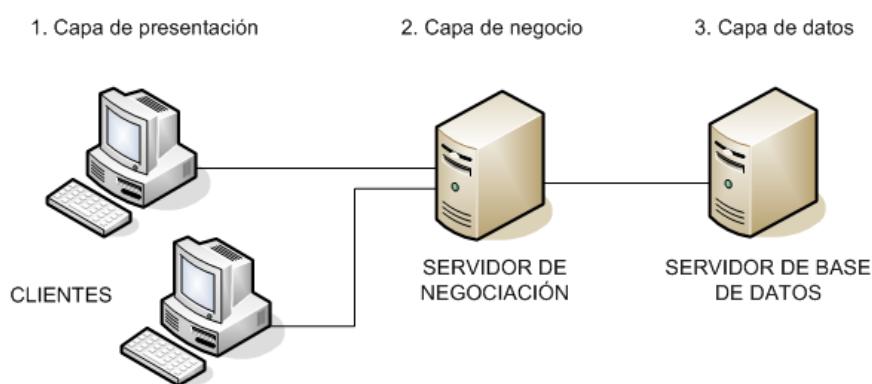


Ilustración 22 Esquema de la arquitectura en tres capas.

La elección principal de este tipo de modelo arquitectural, es la posibilidad de dividir el desarrollo en varios niveles, y que los cambios o actualizaciones en un nivel no afecten necesariamente al conjunto de la aplicación o al resto de capas o niveles, lo cual es un requisito de alta importancia en el desarrollo de este tipo de proyecto debido a su más que probable crecimiento y con ello a la modificación de las capas para adecuarnos a los nuevos paradigmas que se nos presenten.

3.3.2. Patrón de diseño MVC

Debido a la utilización del *framework* de desarrollo CakePhp para el proyecto, cabe resaltar la separación de las capas así como el uso del patrón de diseño MVC (modelo, vista controlador) utilizado por el *framework* [24].

Modelo: El modelo representa la parte del sistema de la estructura de datos del negocio. Esto significa que es responsable de la recuperación de datos convirtiéndolos en conceptos significativos para el sistema, así como su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos.

Vista: Una presentación de los datos del modelo, estando separada de los objetos del modelo. Es responsable del uso de la información de la cual se dispone para producir cualquier interfaz de presentación. La capa de la vista no se limita únicamente a HTML o texto que represente los datos, sino que puede ser utilizada para ofrecer una amplia variedad de formatos en función de sus necesidades tales como videos, música, documentos, etc.

Controlador: La capa del controlador gestiona las peticiones de los usuarios. Es responsable de responder la información solicitada con la ayuda tanto del modelo como de la vista. Los controladores pueden ser vistos como administradores diciendo que todos los recursos necesarios para completar una tarea se deleguen a los trabajadores más adecuados. Los controladores esperan peticiones de los clientes, comprueban su validez de acuerdo a las normas de autenticación o autorización, delega la búsqueda de datos al modelo y selecciona el tipo de respuesta más adecuado según las preferencias del cliente. Finalmente delega este proceso de presentación a la capa de la vista.

¿Por qué utilizar MVC? [25] Debido a que es un patrón de diseño de software verdaderamente probado que convierte una aplicación en un paquete modular fácil de mantener, y aumentar la rapidez del desarrollo. La separación de las tareas de tu aplicación en modelos, vistas y controladores hace que su aplicación sea además muy fácil de entender.

Las nuevas características se añaden fácilmente y agregar cosas nuevas a código viejo se hace muy sencillo. El diseño modular también permite a los desarrolladores y los diseñadores trabajar simultáneamente, incluyendo la capacidad de hacer prototipos rápidos. La separación también permite a los desarrolladores hacer cambios en una parte de la aplicación sin afectar a los demás.

Patrón de diseño MVC en el *framework* de desarrollo CakePhp

View (vistas): Las vistas en CakePHP son las interfaces hacia los usuarios, contienen toda la lógica de representación necesaria para los datos recibidos del controlador. Los ficheros se codifican principalmente con código HTML y JavaScript, así como de código PHP para los datos recibidos del controlador, la extensión de los ficheros por defecto es .ctp (*CakePHPTemplate*).

Los ficheros de las vistas se crearán en la ruta /app/views/, en una carpeta que llevará el nombre del controlador y el nombre del fichero de la acción que realizan. Así la vista de añadir un piso estará en la ruta /app/views/flats/add.ctp siendo flats el controlador y add la acción.

```
<div class="container">
<div class="row">
    <?php echo $this->element('menu', array('site' => 2)); ?>

    <div class="col-md-8" id="alquiler">
        <div class="col-sm-12 col-md-12 margin-bottom-25">
            <h3 class="gray"><span class="glyphicon glyphicon-home"></span> <?php echo __("Alquiler") . ' - ' . $al['Roomsuser']['rent']; ?></h3>
        </div>

        <!--Mi alojamiento-->
        <div class="col-sm-6 col-md-7">
            <div class="color-container simple-white panel-no-border">
                <div class="panel-heading no-border-radius back6">
                    <div class="col-md-10">
                        <h5 class="white"><span class="white glyphicon glyphicon-home"></span> <?php echo __('Mi alojamiento'); ?></h5>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-12">
                        <div class="row c-content">
                            <div class="col-md-12">
                                <div class="row">
                                    <div class="col-md-5 col-img">
                                        <div class="row">
                                            <div class="col-md-12">
                                                
                                            </div>
                                        </div>
                                    </div>
                                    <div class="col-md-7">
                                        <ul class="info-list">
                                            <li class="info-list-elem"><?php echo __('Dirección:'); ?> <span class="data">
                                                <?php echo $flat['Flat']['calle']; ?></span></li>
                                            <li class="info-list-elem"><?php echo __('Habitación:'); ?> <span class="data">
                                                <?php echo $al['Room']['codigo']; ?></span></li>
                                            </ul>
                                            <a class="btn btn-gray" target="_blank" href="http://pisos.emancipia.net/flats/
                                                view/<?php echo $flat['Flat']['id']; ?><?php echo $al['Room']['id']; ?>"><?php
                                                echo __("Ficha del Inmueble"); ?></a>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

Ilustración 23 Ejemplo de View /app/View/Roomsusers/view.ctp

Las vistas en CakePHP se complementa con:

- **layouts:** ficheros de tipo vista que contiene el código de presentación sobre el que se carga las vistas. En ellos suele contenerse el *header* y el *footer* de la aplicación. Ejemplo en la Ilustración 25.
- **elements:** trozo de código de vista más pequeño y reutilizable. Los elementos generalmente son renderizados dentro de las vistas, los elementos pueden recibir parámetros. Ejemplo en la Ilustración 26.

- **helpers:** clases que encapsulan la lógica de la vista que es necesaria. Entre los más destacados están los necesarios para construir formularios, proveer de funcionalidad AJAX, paginar los datos del modelo o servir de feeds RSS. Ejemplo en la Ilustración 27

```
<?php
    header("Content-type: application/pdf");
    echo $this->Html->charset();
?>
<link rel="stylesheet" type="text/css" href="css/emancipia_pdf.css" />
<div id="container">
    <div id="pdf">
        <?php echo $content_for_layout; ?>
    </div>
</div>
```

Ilustración 24 layout PDF /app/View/Layouts/pdf/defaultp.ctp

```
<div class="row">
    <?php echo $this->element('menu', array('site' => 0)); ?>
    <div class="col-md-8">
        <div class="col-sm-12 col-md-12 margin-bottom-25">
            <?php echo $this->Form->input('User.name', array('label' => array('class' => 'col-md-3 control-label',
                'text' => ___('Nombre')))); ?>
            <?php echo $this->Form->input('User.surname', array('label' => array('class' => 'col-md-3 control-label',
                'text' => ___('Apellidos')))); ?>
        </div>
    </div>
```

Ilustración 25element del menú /app/View/elements/menu.ctp

```
<div class="col-md-12">
    <?php echo $this->Form->input('User.name', array('label' => array('class' => 'col-md-3 control-label',
        'text' => ___('Nombre')))); ?>
    <?php echo $this->Form->input('User.surname', array('label' => array('class' => 'col-md-3 control-label',
        'text' => ___('Apellidos')))); ?>
</div>
```

Ilustración 26 helper form /app/View/Users/edit.ctp

Controllers (controladores): El controlador se encarga de manejar la lógica de negocio de la aplicación. Suelen ir emparejados a un solo modelo, aunque desde un mismo controlador se pueden instanciar otros modelos. En el convenio para los nombres de ficheros del CakePHP, los controladores se nombran según el modelo que controlan, y se ponen siempre en plural, por ejemplo FlatsControllers.php será el controlador que maneja la lógica de los pisos *flats*.

Los controladores son sub-clases de la clase AppController de CakePHP Ilustración 28, que a su vez extiende la clase principal Controller. La clase AppController está definida en /app/Controller/AppController.php y contiene los métodos que son compartidos entre todos los controladores de la aplicación.

Desarrollo e implementación – CAPÍTULO 3

```
class AppController extends Controller {
    public $components = array('Session', 'Auth', 'RequestHandler', 'Aux', 'Paginator', 'Emailaux', 'UploadFtp', 'Tpvaux', 'Cur
    public $helpers = array('Html', 'Form', 'Js');

    public function beforeFilter() {
        //IDIOMA
        if ($this->Session->check('Config.language')) {
            Configure::write('Config.language', $this->Session->read('Config.language'));
        }
        $this->Auth->loginAction = array('controller' => 'users', 'action' => 'login');
        $this->Auth->loginRedirect = array('controller' => 'users', 'action' => 'v');
        $this->Auth->authError = ' ';
    }

    public function afterFilter() {
        if ($this->response->statusCode() === 404) {
            return $this->redirect(
                array(
                    'controller' => 'errors',
                    'action'     => 'error404'
                )
            );
        }
        if ($this->response->statusCode() === 1500) {
            return $this->redirect(
                array(
                    'controller' => 'errors',
                    'action'     => 'errorE500'
                )
            );
        }
    }
}
```

Ilustración 27AppController /app/Controllers/AppController.php.

En los controladores se definen los métodos a los que normalmente se les llamará acciones. Una acción es un único método público o privado del controlador. Los controladores únicamente están escritos en código PHP, los métodos pueden ser públicos asociados a vistas o privados que sólo pueden ser llamados desde el propio controlador.

En la Ilustración 29 podemos ver una acción, en concreto la función view del controlador de usuarios UsersController en la que se define la vista de perfil de un usuario y las distintas búsquedas necesarias para mostrar toda la información requerida, como las comprobaciones necesarias y las llamadas a los componentes necesarios.

Desarrollo e implementación – CAPÍTULO 3

```
public function view($user_id = null){  
    $id = $this->Session->read('Auth.User.id');  
    $this->User->unbindModel(array('hasMany' => array('Email')));  
    $emancipia = $this->Aux->emancipia();  
    if(in_array($this->Session->read('Auth.User.group_id'),$emancipia)){  
        $id = $user_id;  
    }  
  
    $user = $this->User->findById($id);  
  
    $this->LoadModel('Contact');  
    $conditions = array('Contact.user_id' => $user['User']['id'], 'Contact.contacto' => 1);  
    $contacto_pre = $this->Contact->find('first',compact('conditions'));  
  
    $educationType = $this->Aux->educationType();  
    $idioma = $this->Aux->idioma();  
    $genero = $this->Aux->genero();  
    $doc_type = $this->Aux->documentType();  
    $nie_type = $this->Aux->nieType();  
  
    $this->LoadModel('Roomsuser');  
    $fields = array('Roomsuser.id');  
    $conditions = array('Roomsuser.user_id' => $id);  
    $list_al = $this->User->Roomsuser->find('list', compact('fields', 'conditions'));  
  
    $this->loadModel('Document');  
    $conditions = array('OR' => array(array('Document.key_id' => $id,'Document.type_key' => 1),array('Document.key_id'  
    $documentos = $this->Document->find('all',compact('conditions'));  
  
    foreach ($documentos as $key => $d) {  
    }  
    if(!empty($doc)){  
        ksort($doc);  
    }  
  
    //Facebook  
    $this->LoadModel('Socialnetwork');  
    $conditions = array('Socialnetwork.user_id' => $id,'Socialnetwork.type' => 1,'Socialnetwork.estado' => 1);  
    $fb = $this->Socialnetwork->find('first',compact('conditions'));  
  
    $title_for_layout = ___('Emancipia: ').$user['User']['name_compl'];  
    $this->set(compact('title_for_layout', 'user', 'idioma', 'contacto_pre', 'genero', 'educationType', 'doc_type', 'doc', '  
}
```

Ilustración 28 Función View /app/Controllers/UsersControllers.php

Los controladores se pueden completar con *components* y *plugins*.

- **Components:** son ficheros de lógica que son compartidos entre los distintos controladores. CakePHP dispone de componentes ya compilados en el núcleo, para ayudar con un desarrollo rápido y estructurado. Algunos de los principales componentes en el núcleo: seguridad, sesiones, emails. CakePHP también ofrece la posibilidad de crear componentes propios para extender funcionalidades a diferentes controladores.
- **Plugins:** Un *plugin* es una pequeña aplicación empaquetada con sus respectivos controladores, modelos y vistas que nos permite extender de nuevas funcionalidades a nuestra propia aplicación. Un *plugin* es una instancia más compleja de un componente.

Un ejemplo de *plugin* utilizado en la plataforma es el CakePdf el cual nos permite la creación de PDF dinámicos. Como vemos en la Ilustración 30 la carga del *plugin*.

```
/** CAKE PDF **/
CakePlugin::load('CakePdf', array('bootstrap' => true, 'routes' => true));
Configure::write('CakePdf', array(
    'engine' => 'CakePdf.DomPdf',
    'options' => array(
        'print-media-type' => false,
        'outline' => true,
        'encoding' => 'UTF8',
        'dpi' => 96
    ),
    'orientation' => 'portrait',
    'margin' => 0,
    'download' => false,
    'filename' => 'Emancipia'
));

```

Ilustración 29PluginCakePdf /app/config/bootstrap.php

Models (modelos): Los modelos en CakePHP representan la abstracción de la estructura y definición del modelo de datos.

Las principales características que componen los modelos son:

- La elección de la base de datos que contiene la tabla del modelo.
- Definición de la tabla y su renombramiento si hiciera falta.
- Definir la relación entre las distintas tablas y la cardinalidad.
- Crear funciones predefinidas o propias de validación de los campos.
- Crear campos virtuales asociados al mismo modelo o a otros.
- Definición de métodos *callback* de control como afterFind o beforeSave.

Los modelos están escritos en PHP. La convención en CakePHP para nombrar a los modelos es el nombre de la tabla asociada en singular dentro de la jerarquía de directorios /app/Model/, el fichero /app/Model/flat.php se asociaría a la tabla *flats*.

Desarrollo e implementación – CAPÍTULO 3

```
<?php
class Flat extends AppModel {
    var $belongsTo = array('User');
    var $hasMany = array(
        'Bathroom' => array( 'className' => 'Bathroom', 'dependent' => true),
        'Record' => array(),
        'Receipt' => array(),
        'Room' => array(),
        'Image' => array(),
        'Keyconsumption' => array(),
        'Contract' => array(),
        'Consumo' => array(),
        'Consumption' => array(),
        'Compensacion' => array(),
        'Pending' => array(),
        'Booking' => array(),
        'Rent' => array(),
        'Available' => array(),
    );
    var $hasOne = array('Clothesline', 'Kitchen', 'Living', 'Terrace', 'Hall', 'Otherflat', 'Paid');

    var $virtualFields = array(
        'codigo' => 'CONCAT(Flat.name, " - ", Flat.direccion)',
        'calle' => 'CONCAT(Flat.direccion, " ", Flat.numero, " ", Flat.bloque, " ", Flat.portal, " ", Flat.escalera, " ", Flat.piso)',
        'price' => ' SELECT price FROM rents where (flat_id = `Flat`.`id` AND room_id = 0) ORDER BY created DESC LIMIT 1',
        'camas' => 'SELECT SUM(camas)',
        'huesp' => 'SELECT SUM(ocupantes_max)',
        'aseo' => 'SELECT SUM(aseo)',
        'internet' => 'SELECT int_internet',
        'lavadora' => 'SELECT eg_lavadora',
        'calefaccion' => 'SELECT tipo_calefaccion'
    );
    var $validate = array(
        'name' => array(
            'required' => array(
                'rule' => array('notEmpty'),
                'message' => 'El código no puede estar vacío'
            )
        ),
        'n_habitaciones' => array(),
        'n_banos' => array()
    );
}
?>
```

Ilustración 30 Modelo Flat /app/Model/flat.php

El ciclo de una petición en CakePHP

Para entender mejor la conexión entre los distintos componentes que forman el CakePHP, en la Ilustración 32 se muestra los caminos que sigue una petición *url*, desde que la inicia un cliente hasta que recibe la respuesta [25].

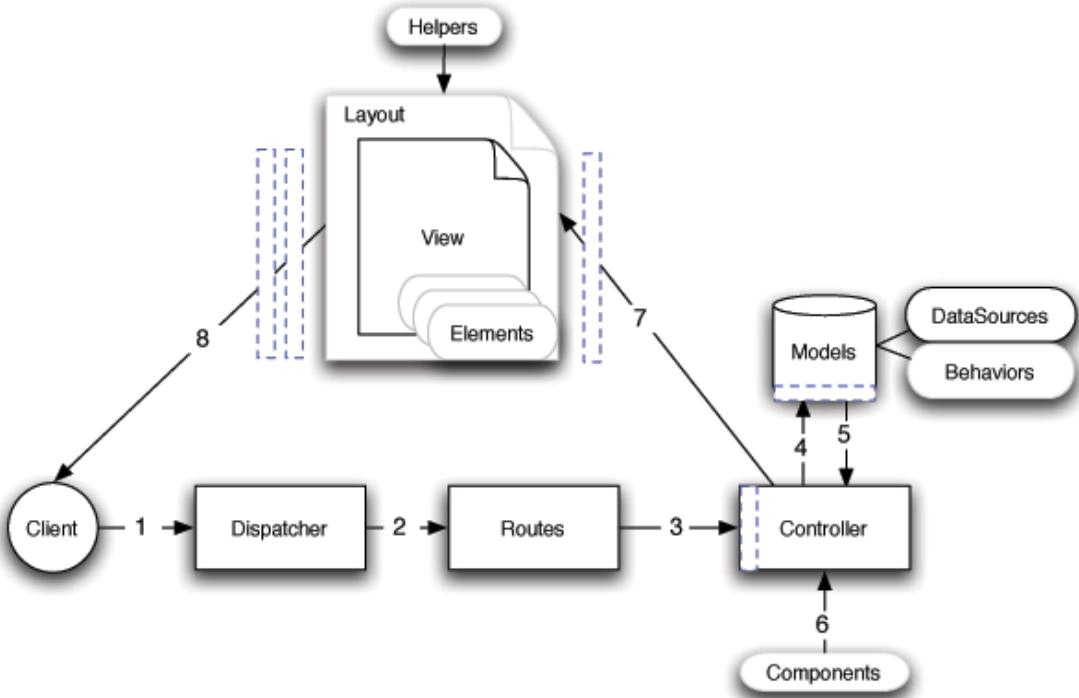


Ilustración 31 Diagrama de un petición en CakePhp

1. El ciclo de una petición típica en CakePHP comienza cuando un usuario solicita una página o un recurso de tu aplicación.
2. El navegador realiza una petición (*request*) al servidor web.
3. El Routes interpreta la URL para extraer los parámetros para esta petición: el controlador, la acción y cualquier otro argumento que afecte a la lógica de negocio durante el *request*.
4. Usando las rutas, se construye una URL objetivo relacionada con una acción de un controlador (un método específico en una clase controlador). El *callbackbeforeFilter()* de este controlador es invocado antes de ejecutar ningún otro método.
5. El controlador puede utilizar uno o varios modelos para acceder a los datos así como cualquier *callback* del modelo, comportamiento (*behavior*), o *DataSource* que sea aplicable puede ser ejecutado en este momento. Aunque utilizar un modelo no es obligatorio, todos los controladores de CakePHP requieren inicialmente un modelo.
6. Una vez el modelo ha recuperado los datos, es devuelto al controlador. Se aplican aquí los *callbacks* del modelo.
7. El controlador puede utilizar componentes para refinar los datos o realizar otras operaciones (manipular la sesión, autenticación o enviar emails, por ejemplo).
8. Una vez el controlador ha empleado los modelos y componentes para preparar los datos, se envían a la vista. Los *callback* del controlador pueden ser ejecutados antes de que los datos sean enviados. La lógica de la vista se realiza

en este punto. Esto puede incluir el uso de elementos (*elements*) y/o helpers. Por defecto, las vistas son generadas dentro de una plantilla (*layout*).

9. *Callback* adicionales pueden ejecutarse ahora (como *afterFilter*) en el controlador. La vista, ya generada por completo, se envía al navegador para su visualización.

3.4 Diseño del sistema

Una vez definido el sistema arquitectónico de la aplicación debemos adaptar los requerimientos a la arquitectura dada. Para ello, el desarrollo del sistema comienza con el diseño en detalle del servidor de la aplicación y un diagrama de las conexiones entre los distintos componentes.

3.4.1.- Diseño detallado

En este apartado se muestra el diseño detallado de la plataforma, el cual ha sido realizado con los componentes que lo forman y la conexión que existe entre ellos. En el diagrama se han tenido en cuenta tanto los componentes propios implementados para el proyecto como los externos en los que se han apoyado diferentes funcionalidades.

En la Ilustración 33 podemos ver todos los componentes que forman la plataforma. El diseño se divide en las tres capas del diseño arquitectónico elegido: la capa de presentación, capa de negocio y la capa de persistencia de datos.

Capa de presentación: Tenemos las interfaces de la plataforma, con las que van a interactuar los usuarios. Cada aplicación tiene su capa de presentación propia. Los componentes que forman la capa son: *views*, *elements*, *helpers* y *layouts*.

Capa de negocio: Donde se encuentra la lógica de negocio. Cada aplicación tiene su propia capa de negocio. Los componentes que forman la capa los podemos clasificar en internos y externos.

- Componentes internos: son los que se han desarrollado para el proyecto, los podemos clasificar en *views*, *components* y *plugins*.
- Componentes externos: hay ciertas funcionalidades del sistema que no podemos controlar directamente, ya que precisamos de empresas externas. Para ello las empresas ofrecen herramientas para ayudar a los desarrolladores e interactuar con ellas, a estas herramientas se las conoce como APIs. En el desarrollo del proyecto hemos utilizado.
 - Mandrill: como gestor de correo saliente.

Desarrollo e implementación – CAPÍTULO 3

- PayPal: como pasarela de pago segura.
- TPV Sabadell: como pasarela de pago segura.
- 1and1: como servidor ftps para almacenar ficheros.
- Facebook: enlazar usuarios con la red social.

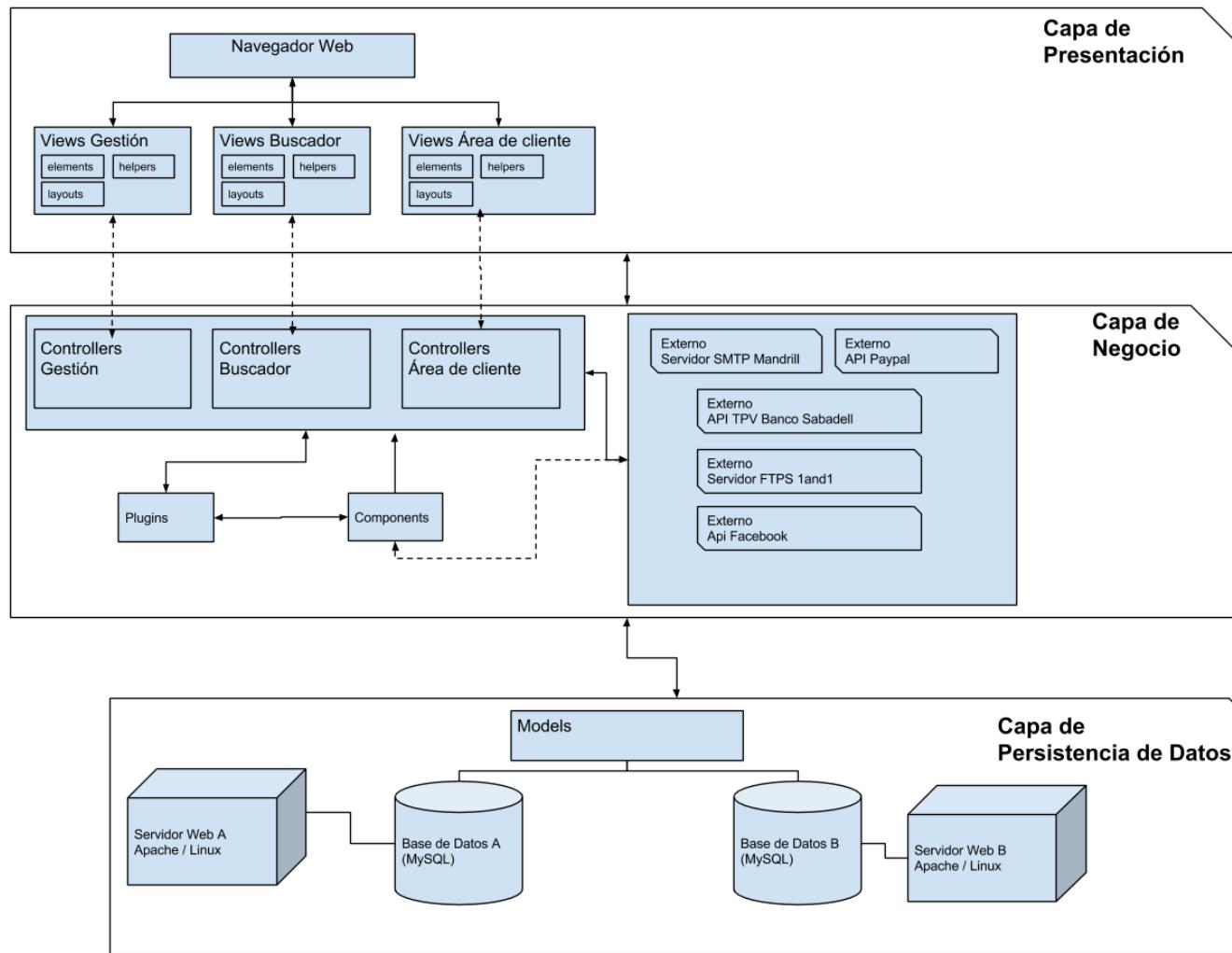


Ilustración 32 Diagrama del sistema de la plataforma Emancipia.

3.5 Implementación

En este apartado se explicará la implementación del proyecto, esta es la fase del desarrollo en la que las iteraciones conllevan más horas de trabajo. En la explicación se detallará brevemente como se implemento cada capa, así como alguna característica destacable de la implementación. El desarrollo iterativo e incremental promueve el *feedback* del cliente por medio de los prototipos, esto favorece la satisfacción final del cliente, pero puede llegar a aumentar el tiempo de desarrollo, sobre todo en la fase de implementación, ya que se sufren cambios en los requisitos o la aparición de algunos nuevos.

La implementación de la plataforma ha sido dividida en tres aplicaciones separadas a nivel lógico, físicamente están ubicadas en el mismo servidor. La separación surge de la principal funcionalidad que lleva a cabo cada una de ellas.

Plataforma Emancipia:

- **Aplicación de gestión interna** (<http://emancipia.net>) dedicada principalmente a la gestión del día a día de los alojamientos.
- **Buscador público** (<http://pisos.emancipia.net>) aplicación abierta a los usuarios en la cual los usuarios pueden realizar búsquedas filtradas de los distintos alojamientos, solicitar información sobre ellos y realizar las reservas.
- **Área de cliente** (<http://app.emancipia.net>) aplicación cerrada solo accesible a usuarios registrados en la cual podrán ver su información personal y la relativa a sus alojamientos.

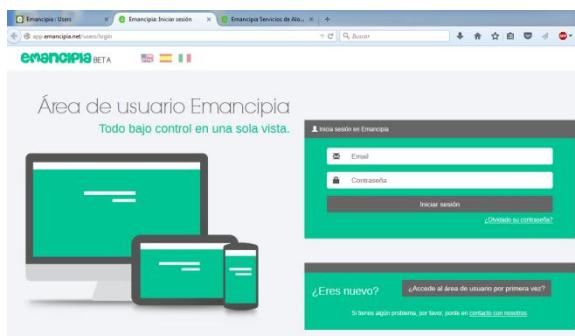


Ilustración 33 Área de cliente <http://app.emancipia.net>



Ilustración 34 Buscador <http://pisos.emancipia.net/flats/portada>

Anexo 1



Ilustración 35 Gestión interna <http://emancipia.net>

Las tres aplicaciones que forman la plataforma han seguido un parecido proceso de implementación y se han desarrollado con las mismas herramientas y tecnologías. Aunque debido a las características particulares de cada una y sobre todo al inicio del desarrollo de cada una de ellas, cabe destacar algunas variaciones en las tecnologías utilizadas representadas en la tabla 1

	Gestión interna www.emancipia.net	Buscador público pisos.emancipia.net	Área de cliente app.emancipia.net
Acceso a la aplicación	administradores	público	usuario registrados
Programación del servidor	CakePhp v2.3.6	CakePhp v2.3.6	CakePhp v2.5.3
Programación del cliente	HTML5/CSS3.1	HTML5/CSS3.1	HTML5/CSS3.1
Bootstrap	v2.3.2	v3.1.1	v3.2.0
jQuery	v1.10.1	v1.10.1	v1.11.1
Internacionalización del contenido	Español	Español, Ingles y Francés	Español, Ingles, Francés e Italiano.
Diseño adaptado a móviles	no	si	si

Tabla 1 Comparativa de aplicaciones de la plataforma.

A modo de ejemplo se explicará solo la implementación de la aplicación correspondiente al buscador público, ya que como se comentó anteriormente toda la plataforma ha seguido el mismo proceso.

3.5.1 Implementación del buscador

El orden en el que se detalla el proceso de implementación del sistema no se ajusta al realizado realmente, ya que se han ido cambiando los requisitos durante la desarrollo y las iteraciones han ido aumentándose.

3.5.1.1 Implementación de la capa de datos

La aplicación se nutre de datos almacenados en una base de datos, utilizando para su manipulación el sistema gestor de bases de datos MySQL. La administración de estas se lleva a cabo con el software phpMyAdmin.

```
class DATABASE_CONFIG {
    public $default = array(
        'datasource' => 'Database/Mysql',
        'persistent' => false,
        'host' => 'localhost',
        'login' => '*****',
        'password' => '*****',
        'database' => 'emancipia_gestion',
        'prefix' => '',
        'encoding' => 'utf8',
    );
    public $emancipia_2 = array(
        'datasource' => 'Database/Mysql',
        'persistent' => false,
        'host' => '146.255.96.15',
        'login' => '*****',
        'password' => '*****',
        'database' => 'housingon_emancipia_2',
        'prefix' => '',
        'encoding' => 'utf8',
    );
}
```

Ilustración 36 Conexión a la base de datos /app/Config/database.php

En la Ilustración 37 se muestra la configuración de la conexión a la base de datos, como se puede apreciar se realizan dos conexiones, ya que la plataforma está dividida en dos bases de datos. Esta división se debe a problemas de espacio de almacenamiento y una separación física de los datos por sus características. Debido a que tenemos datos que son muy pesados y que tienen poca carga en el día a día de la plataforma. Estos datos principalmente son registros de acceso, cálculos de estimaciones e información obsoleta.

La capa de acceso datos esta implementada en código PHP. En ella se detallan las tablas y la relación que existe entre ellas. La cada de datos es compartida por todas las aplicaciones de la plataforma. El sistema de ficheros se encuentra en la ruta /htdocs/app/Model/ dicha ruta es accesible desde toda la plataforma.

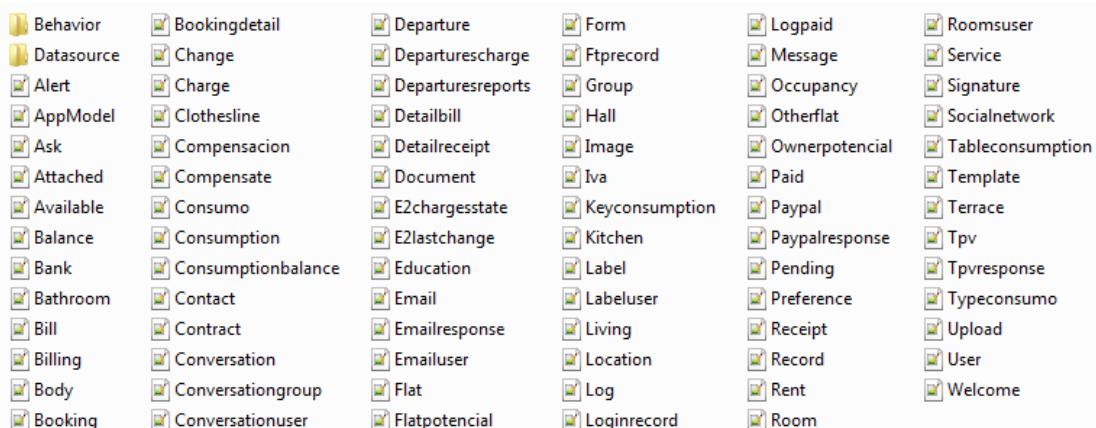


Ilustración 37 Modelos httpdocs/app/Model/

Anexo 1

En la ilustración 38 podemos ver el sistema de ficheros de los modelos de la plataforma, cada tabla de la base de datos se le asocia a un modelo, aunque CakePhp no obliga a la creación de un modelo por tabla, es recomendable crearlo, ya que si no se encuentra, el *framework* creará un modelo virtual con los parámetros por defecto.

```
<?php
class Flat extends AppModel {
    var $belongsTo = array('User');
    var $hasMany = array(
        'Bathroom' => array('className' => 'Bathroom', 'dependent' => true),
        'Record' => array('className' => 'Record', 'dependent' => true),
        'Receipt' => array('className' => 'Receipt', 'dependent' => true),
        'Room' => array('className' => 'Room', 'dependent' => true),
        'Image' => array('className' => 'Image', 'dependent' => true, 'order' => 'Image.orden DESC'),
        'Keyconsumption' => array('className' => 'Keyconsumption', 'dependent' => true, 'order' => 'Keyconsumption.created DESC'),
        'Contract' => array('className' => 'Contract', 'dependent' => true, 'order' => 'Contract.created DESC'),
        'Consumo' => array('className' => 'Consumo', 'dependent' => true, 'order' => 'Consumo.created DESC'),
        'Consumption' => array('className' => 'Consumption', 'dependent' => true, 'order' => 'Consumption.created DESC'),
        'Compensacion' => array('className' => 'Compensacion', 'dependent' => true, 'order' => 'Compensacion.created DESC'),
        'Pending' => array('className' => 'Pending', 'conditions' => array('Pending.estado' => '1')),
        'Booking' => array('className' => 'Booking', 'dependent' => true, 'order' => 'Booking.created DESC'),
        'Rent' => array('className' => 'Rent', 'dependent' => true, 'order' => 'Rent.created DESC'),
        'Available' => array('className' => 'Available', 'dependent' => true, 'order' => 'Available.created DESC'),
    );
    var $hasOne = array('Clothesline', 'Kitchen', 'Living', 'Terrace', 'Hall', 'Otherflat', 'Paid');

    var $virtualFields = array(
        'codigo' => 'CONCAT(Flat.name, " - ", Flat.direccion)',
        'calle' => 'CONCAT(Flat.direccion, " ", Flat.numero, " ", Flat.bloque, " ", Flat.portal, " ", Flat.escalera, " ", Flat.piso, " ", Flat.letra)',
        'price' => 'SELECT price FROM rents where (flat_id = `Flat`.`id` AND room_id = 0) ORDER BY created DESC LIMIT 1',
        'camas' => 'SELECT SUM(camas) FROM rooms where (flat_id = `Flat`.`id`) ORDER BY created DESC',
        'cama_supletoria' => 'SELECT SUM(cama_supletoria) FROM rooms where (flat_id = `Flat`.`id`) ORDER BY created DESC',
        'huesp' => 'SELECT SUM(ocupantes_max) FROM rooms where (flat_id = `Flat`.`id`) ORDER BY created DESC',
        'aseo' => 'SELECT SUM(aseo) FROM bathrooms where (flat_id = `Flat`.`id` AND aseo = 1) ORDER BY created DESC',
        'internet' => 'SELECT int.internet FROM otherflats where (flat_id = `Flat`.`id`) ORDER BY created DESC',
        'lavadora' => 'SELECT eg.lavadora FROM otherflats where (flat_id = `Flat`.`id`) ORDER BY created DESC',
        'calefaccion' => 'SELECT tipo_calefaccion FROM otherflats where (flat_id = `Flat`.`id`) ORDER BY created DESC',
        'cocina' => 'SELECT tipo_cocina FROM kitchens where (flat_id = `Flat`.`id`) ORDER BY created DESC',
    );
    var $validate = array(
        'name' => array('required' => array('rule' => array('notEmpty'), 'message' => 'El código no puede estar vacío' )),
        'n_habitaciones' => array('required' => array('rule' => array('notEmpty'), 'message' => 'El campo habitaciones no puede estar vacío' )),
        'n_banos' => array('required' => array('rule' => array('notEmpty'), 'message' => 'El campo baños no puede estar vacío'))
    );
}
?>
```

Ilustración 38 Modelo de la tabla Piso /app/Models/flat.php

En la ilustración 39 tenemos un modelo, en concreto el que corresponde a la tabla flats, ubicado en la ruta /app/Models/flat.php. En el podemos ver:

Relaciones con las demás tablas de la plataforma.

- \$belongsTo => Un piso pertenece a un único usuario.
- \$hasMany => Un piso puede tener 0 o muchos baños.
- \$hasOne => Un piso puede tener 0 o 1 tendedero.

Las relaciones las podemos definir más específicamente con parámetros como:

- className => nombre de la clase para interactuar con ella.
- dependent => Borrado en cascada
- order => El orden del array asociativo retornado.
- conditions => Añadir condiciones para que la relación se cumpla.

\$virtualFields => Lo que en bases de datos llamaríamos campos calculados. En los cuales se pueden definir desde concatenaciones de campos a búsquedas más complejas en otras tablas.

\$validate => validación de campos, previa a guardar los datos en la base de datos.

3.5.1.2 Implementación de la capa de la capa de negocio.

En la capa de negocio es donde se concentra toda la lógica de la plataforma. Está formada principalmente por los controladores. Esta únicamente escrita en PHP y se conecta con la capa de datos y presentación.

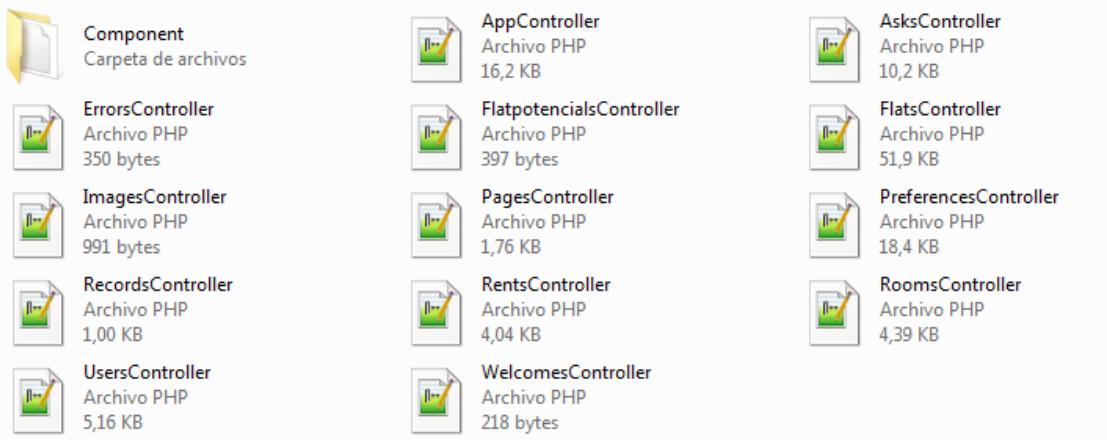


Ilustración 39 Controladores /pisos.emancipia.net/app/Controllers/

En la ilustración 40 podemos ver el sistema de ficheros de los controladores de la aplicación, cada modelo está asociado a un controlador, aunque desde un controlador se puede instanciar otros modelos.

```

1 <?php
2 class FlatsController extends AppController {
3     var $name = 'Flats';
4
5     public function policy(){}
6     public function cookies(){}
7     public function infoowner(){}
8     public function contacto(){}
9     public function trabaja(){}
10    public function nosotros(){}
11    public function help(){}
12    public function emancipia(){}
13    public function condiciones(){}
14    public function faq(){}
15    public function ad(){}
16    public function thx(){}
17    public function encuentra(){}
18    public function portada(){}
19    public function invierte(){}
20    public function paypalreceive(){}
21    public function tpvreceive(){}
22    public function booking_offer($id){}
23    public function booking($flat_id = null, $room_id = null){}
24    public function reserva($flat_id = null, $room_id = null){}
25    public function disponibilidad($flat_id = null, $room_id = null){}
26    public function tpv($t = null, $r = null){}
27    public function fav(){}
28    public function deletefav(){}
29    public function viewfav($id = null, $list = null){}
30    public function favoritos($id = null){}
31    public function cookie($flat = null, $room = null){}
32    public function index(){}
33    public function buscar(){}
34    private function __disponibles($inicio = null, $fin = null){}
35    public function search_name($tipo = null){}
36    private function __sin_resultados($hab = 0){}
37    public function search($favTipo = null){}
38    public function map($id = null){}
39    public function mapa($flat_id = null, $room_id = null){}
40    public function view($id = null, $room_id = null){}
41    public function ver($id = null, $room_id = null){}
42    public function h($id = null){}
43    public function asignar(){}
44}
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
626
627
627
628
629
629
630
631
632
633
634
635
636
637
637
638
639
639
640
641
642
643
644
645
645
646
647
647
648
649
649
650
651
652
653
654
655
656
656
657
658
658
659
659
660
661
662
663
664
665
666
666
667
668
668
669
669
670
671
672
673
674
675
676
676
677
678
678
679
679
680
681
682
683
684
685
685
686
686
687
687
688
688
689
689
690
691
692
693
694
695
695
696
696
697
697
698
698
699
699
700
701
702
703
704
705
705
706
706
707
707
708
708
709
709
710
711
712
713
714
715
715
716
716
717
717
718
718
719
719
720
721
722
723
724
725
725
726
726
727
727
728
728
729
729
730
731
732
733
734
735
735
736
736
737
737
738
738
739
739
740
741
742
743
744
744
745
745
746
746
747
747
748
748
749
749
750
751
752
753
754
754
755
755
756
756
757
757
758
758
759
759
760
761
762
763
764
764
765
765
766
766
767
767
768
768
769
769
770
771
772
773
774
774
775
775
776
776
777
777
778
778
779
779
780
781
782
783
784
784
785
785
786
786
787
787
788
788
789
789
790
791
792
793
794
794
795
795
796
796
797
797
798
798
799
799
800
801
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
811
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
821
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
831
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
841
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
851
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
861
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
871
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
881
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
891
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
901
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
921
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
931
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
941
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
951
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
961
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
971
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
981
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1571
157
```

Anexo 1

```
709
710     public function search_name() {
711         if (!empty($this->data)){
712             if(empty($this->data['Flat']['buscar'])){
713                 $this->redirect(array('action' => 'buscar'));
714             }
715             $this->loadModel('Room');
716             $this->Paginator->settings = array(
717                 'conditions' => array('OR' => array(
718                     array('Room.codigo_room LIKE' => '%'.$this->data['Flat']['buscar'].'%'),
719                     , 'Room.buscador =' => 1),
720                     array('Flat.direccion LIKE' => '%'.$this->data['Flat']['buscar'].'%'),
721                     , 'Room.buscador =' => 1)
722                 )),
723                 'order' => array('Room.prioridad' => 'desc'),
724                 'recursive' => 1,
725                 'limit' => 5,
726                 'max_record' => 5
727             );
728             $rooms = $this->Paginator->paginate('Room');
729             $hab = 1;
730             $this->data = $rooms;
731             $favoritos = 0;
732             $this->set(compact('favoritos', 'hab'));
733         }
734         if (empty($this->data)){
735             //prd($this->data);
736             $sin_resultados = true;
737             $this->data = $this->__sin_resultados();
738             $this->set(compact('sin_resultados'));
739         }
740         $this->render('buscar');
741     }
742 }
743
```

Ilustración 41 función search_name del FlatsControllers.php

En la ilustración 42 podemos ver un ejemplo de función en concreto la función para buscar habitaciones por su código o dirección. Para ello cargamos el modelo Room y realizamos una búsqueda en la base de datos mediante la instrucción paginate. Si el retorno de la llamada resulta nulo, llamamos a la función privada __sin_resultados, Ilustración 43 para que nos retorne las habitaciones destacadas, con esto evitamos que la página se quede en blanco.

```
744
745     private function __sin_resultados() {
746         $favoritos = 0;
747         $title_for_layout = 'Habitaciones';
748
749         $this->Flat->unbindModel(
750             array('hasMany' => array(
751                 'Record', 'Receipt', 'Contract', 'Keyconsumption',
752                 'Consumo', 'Consumption', 'Rent', 'Available',
753                 'Booking', 'Pending', 'Compensacion')
754             )
755         );
756
757         $this->Paginator->settings = array(
758             'conditions' => array('Room.buscador =' => 1),
759             'order' => array('Room.prioridad' => 'desc'),
760             'recursive' => 1,
761             'max_record' => 5,
762             'limit' => 5
763         );
764
765         $resultados = $this->Paginator->paginate('Room');
766         $piso = 0;
767         $hab = 1;
768
769         return $resultados;
770     }
771 }
```

Ilustración 42 función privada __sin_resultados del FlatsControllers.php

3.5.1.3 Implementación de la capa de presentación.

En la capa de presentación es donde se encuentran las interfaces con las que interactúan los usuarios del sistema. Los archivos están escritos en: código HTML para la estructura, código CSS para el diseño, código JavaScript para dotar de dinamismo y código PHP para mostrar los datos.

Anexo 1

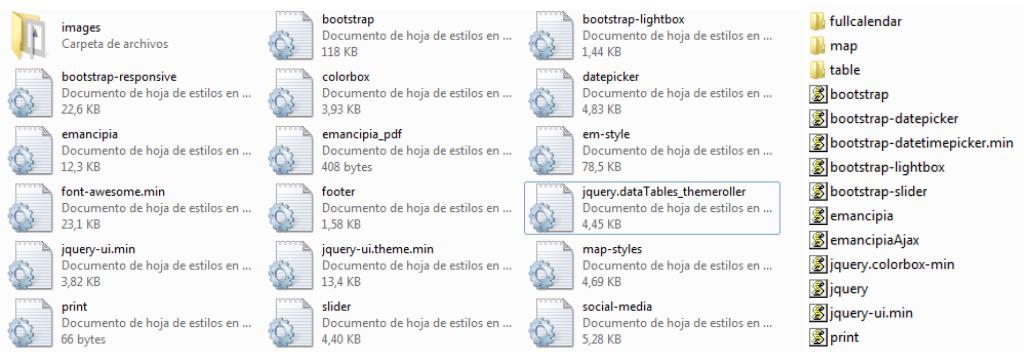


Ilustración 43 Ficheros CSS y JS pisos.emancipia.net/app/webroot /

En la ilustración 44 podemos ver el sistema de ficheros de los estilos y el javascript de la aplicación. Un requisito importante es el diseño adaptado a dispositivos móviles, esto se consigue con hojas de estilo y javascript en la Ilustración 45 se ve el resultado.

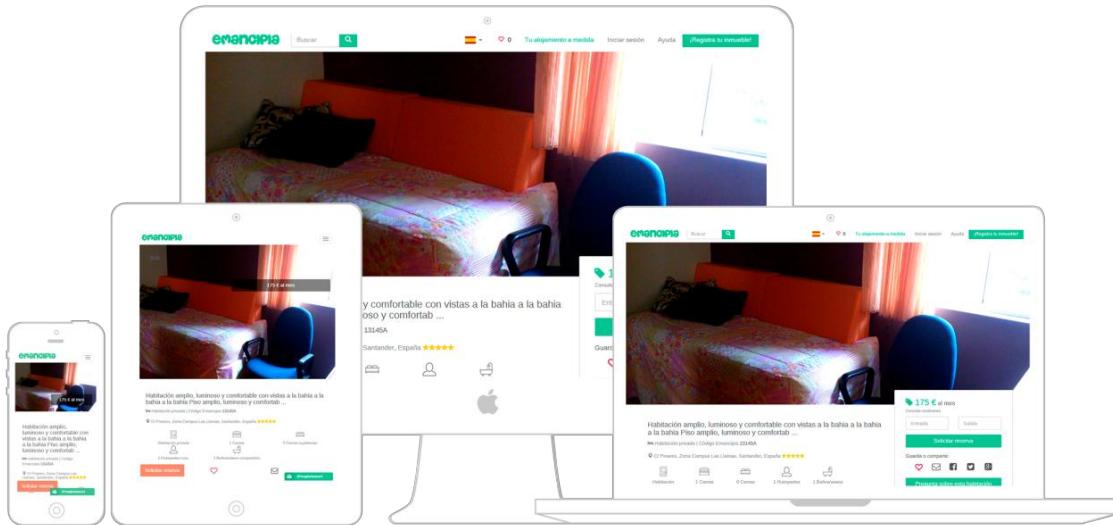


Ilustración 44 Mockup Diseño adaptado a dispositivos móviles.

CAPÍTULO 4

Validación y pruebas

En este capítulo se describe brevemente las validaciones y pruebas de la plataforma desarrollada.

4.1 Pruebas realizadas

En todas las metodologías es necesario aplicar una etapa de pruebas y validaciones para garantizar un correcto funcionamiento del producto. En la metodología utilizada en el desarrollo, la metodología iterativa e incremental las pruebas se llevan a cabo en cada iteración realizada individualmente, consiguiendo minimizar así los posibles errores y que no se hereden en posteriores iteraciones del producto. Llamamos errores no solo a un mal funcionamiento del software sino también a que el prototipo presente un comportamiento no deseado por el cliente ya sea una funcionalidad mal desarrollada o una funcionalidad que el cliente decida modificar del prototipo al término de la iteración planificada.

Para una correcta evaluación en la metodología se pueden clasificar los tipos de pruebas en:

1. Pruebas unitarias
2. Pruebas de integración
3. Pruebas del sistema
4. Pruebas de rendimiento
5. Pruebas de aceptación por parte del cliente

4.1.1 Pruebas unitarias

Las pruebas unitarias se centran en la unidad más pequeña de un diseño de software, lo que podríamos conocer como los módulos, en nuestro caso acciones/funciones de los distintos controladores. Las pruebas se realizan por cada módulo individualmente.

Anexo 1

Para la realización de las pruebas se han diseñado una serie de test para cada módulo con unos valores críticos que garanticen una cobertura razonable de sentencias y condiciones. Asegurando así recorrer todas las líneas del código.

Debido al gran número de controladores y funciones no ha sido posible recoger en la memoria todas las pruebas realizadas. A continuación se muestra un pequeño ejemplo de test unitario de una función propia.

Ejemplo de test unitario.

4.1.2 Pruebas de integración

Una vez realizadas las pruebas unitarias en cada módulo por separado se ha ido ampliando el rango con la realización de las pruebas de integración. Las pruebas de integración verifican la correcta respuesta entre distintos módulos. Estas pruebas se han ido realizando a medida que se daban por finalizados módulos y se iban ensamblando unos con otros, en distintas iteraciones hasta completar la funcionalidad completa, para ello las pruebas de integración se han realizado de forma incremental.

4.1.3 Pruebas de sistema

El siguiente paso son las pruebas del sistema, las pruebas del sistema miden el comportamiento del sistema en su conjunto. Hasta ahora las pruebas han medido los fallos funcionales del sistema. Las pruebas del sistema se realizan para verificar el correcto cumplimiento de los requisitos no funcionales, estas pruebas se clasifican en: pruebas de rendimiento del sistema y pruebas de seguridad

4.2.3.1 Pruebas de rendimiento

Las pruebas de rendimiento verifican la correcta visualización y carga en distintos navegadores, como se especifica en el RNF99 y en dispositivos móviles como se especifica en el RNF99.

Las pruebas en ordenador se han realizado sobre Internet Explorer v11, Google Chrome v44, Mozilla Firefox v40 y Safari v8. Las pruebas en la versión móvil se han realizado sobre el navegador Chrome de un Andorid/Honor 6 y sobre Safari en un Apple/Iphone 6 plus. Los resultados de las pruebas realizadas se pueden ver en la Tabla 2

	Buscador público		Área de cliente	
	Visualización	Tiempo de carga	Visualización	Tiempo de carga
Internet Explorer v11	80%	2.1 seg	85%	1.9 seg
Google Chrome v44	100%	1.8 seg	100%	1.7 seg
Mozilla Firefox v40	95%	1.85 seg	96%	1.8 seg
Safari v8	95%	2.2seg	95%	2 seg

Honor 6/ Chrome	90%	2.5 seg	90%	2.1 seg
Iphone 6 plus/ Safari	85%	2.5 seg	86%	2 seg

Tabla 2 Comparativa de rendimiento de las aplicaciones.

Los porcentajes y tiempos de carga se han obtenido por una media ponderada teniendo en cuenta el número de páginas individuales de cada aplicación y una estimación media del número de visualizaciones de esa página respecto a las demás. Los tiempos de carga se han realizado con la herramienta on-line pingdom.

4.3.2 Pruebas de seguridad

Las pruebas de seguridad tienen como principal objetivo que ningún usuario malicioso, este o no este registrado, pueda aprovecharse de alguna funcionalidad de forma incorrecta comprometiendo así el funcionamiento del sistema.

En la plataforma se ha ido dando acceso paulatinamente a usuarios los cuales nos han servido para realizar las pruebas de seguridad, así mismo miembros de la empresa han probado las distintas funcionalidades intentando hacer un uso fraudulento de ellas. Los informes recibidos se han utilizado para las pruebas.

4.4 Pruebas de aceptación

Las pruebas de aceptación miden el nivel de satisfacción del cliente, el cumplimiento de los objetivos y los requisitos sobre el producto final. Al tratarse de un proyecto interno de la empresa, las pruebas de aceptación se han realizado a medida que se avanzaba en el desarrollo y se iban entregando los prototipos con funcionalidades terminadas.

Aunque la plataforma todavía sigue en desarrollo la Empresa en la última iteración dio por cumplidos de manera satisfactoria todos los objetivos y requisitos planteados al inicio.

CAPÍTULO 5

Conclusiones y trabajos futuros

Una vez terminado el proyecto, y expuesto todo el proceso de su desarrollo, en este capítulo se exponen las conclusiones alcanzadas, así como unas líneas de trabajo futuro a seguir en un posterior desarrollo de la plataforma.

5.1 Conclusiones

El desarrollo del proyecto tenía como objetivo diseñar, implementar y desplegar un sistema de gestión de alojamientos para la empresa Emancipia. Para lograrlo se ha seguido las fases del ciclo de vida de un desarrollo de software. Estas fases incluyen desde las capturas de requisitos, el análisis del problema, el diseño e implementación y por último una fase de evaluación y pruebas del producto. Para ello se ha seguido una metodología de desarrollo iterativo e incremental.

En el capítulo 1 se enumeraban una serie de objetivos a cumplir en el desarrollo y se puede concluir que se han cumplido todos de manera satisfactoria. Las principales metas alcanzadas en el proyecto las podemos separar en las tres aplicaciones implementadas:

Gestor interno de Emancipia.

- Se ha implementado un sistema de **gestión de cobros** capaz de generar automáticamente todos los cobros mensuales de un alojamiento asociándolos a los inquilinos y a los propietarios respectivamente, permitiendo desglosar los cobros en renta, consumos asociados, adelantos, comisiones, reparaciones, etc. Para su posterior envío por correo electrónico.
- Se ha implementado una herramienta visual de **disponibilidad de los inmuebles** y habitaciones permitiendo filtrar los resultados por las características de estos y las fechas de consulta. Separando los resultados por distintos tipos de ocupaciones como reservas, bajas, contratos, reformas y alquileres.

- Se ha integrado mediante el servicio externo Mandrill un servidor de correo electrónico con el que por medio de **plantillas HTML** se pueden realizar envíos personalizados dentro de la plataforma.

Portal de búsqueda de alojamiento.

- Se ha desarrollado un portal de búsqueda de inmuebles y habitaciones dinámico con un **diseño adaptado a dispositivos móviles** y permitiendo la visualización en varios idiomas.
- Se ha desarrollado un **mapa inteligente** con el que los usuarios pueden tener una mejor idea de la ubicación real del inmueble, de lo que le rodea y las distancias entre distintos puntos de interés.
- Se ha integrado diferentes **métodos de pagos virtuales** permitiendo a los usuarios realizar los pagos dentro de la plataforma.

Área personal de cliente.

- Se ha desarrollado un área de cliente en la cual se le muestra al usuario **clara y ordenada toda su información** tanto personal como del alojamiento así como las distintas interacciones que tiene con Emancipia: correos electrónicos, recibos, pagos, etc.
- Se ha desarrollado un **diseño adaptado a dispositivos móviles** y permitiendo la visualización en varios idiomas.

A través de las tres aplicaciones que forman la plataforma de alojamiento, Emancipia ha incrementado su número de clientes sin que esto haya supuesto un sobrecoste considerable y consiguiendo haber **mejorado la satisfacción final de los usuarios**.

Finalmente, decir que aunque se ha conseguido una plataforma que abarca los aspectos más importantes en la gestión del alojamiento, todavía quedan muchos aspectos con posibilidad de mejora y muchas nuevas funcionalidades que añadir a la plataforma. Algunas de estas posibles nuevas funcionalidades se comentan en el siguiente apartado trabajos futuros.

5.2 Trabajos futuros

Como sucede en todo tipo de proyectos durante el tiempo de su desarrollo se plantean nuevas funcionalidades o mejoras de las ya previstas fruto de la retroalimentación a medida que se va desarrollando o una vez ya desplegado, por los propios usuarios del sistema. Algunos de los posibles trabajos futuros que se han planteado son:

- **Seguridad:** Mejorar aún más la seguridad de la plataforma añadiendo sistemas *captcha*s más avanzados y una capa extra de seguridad propia a nivel del servidor.
- **App móvil:** La implementación de un *webservices* aprovechando la arquitectura de separación en capas para el desarrollo de aplicaciones móviles.
- **Servidor:** El despliegue a servidor con más recursos dedicados a la plataforma. Servidor Virtual Privado.

Anexo 1

- **Roles:** Permitir que un mismo usuario tenga más de un rol dentro de la plataforma.
- **Propietarios:** Posibilidad de dividir la propiedad de un inmueble en porcentajes y que estos porcentajes se respeten en el pago del inmueble.
- **Documentación detallada del sistema:** Debido a la falta de tiempo durante el desarrollo e implementación de la plataforma no se ha realizado una documentación detallada.
- **Pasarela de pago:** dar la posibilidad al usuario de domiciliar los pagos de la renta mensual.
- **Reconocimiento de caracteres:** implementación de un sistema de reconocimiento de caracteres para que el sistema pueda interpretar las distintas facturas de consumos.

Bibliografía

- [1] Ministerio de Educación, Cultura y Deporte. *Datos Básicos del Sistema Universitario Español. Curso 2013/2014* [En línea]. Disponible en:
http://www.mecd.gob.es/dms/mecd/educacion-mecd/areas-educacion/universidades/estadisticas-informes/datos-cifras/DATOS_CIFRAS_13_14.pdf [Última visita: 28/08/2015]
- [2] *La UC en cifras* [En línea]. Disponible en:
https://www.unican.es/WebUC/Internet/Estudiantes_internacionales/La+UC+en+cifras.htm [Última visita: 28/08/2015]
- [3] Jorge Franganillo. *HTML5: el nuevo estándar básico de la Web* [En línea] Disponible en: <http://franganillo.es/html5.pdf> [Última visita: 15/6/2015]
- [4] HTML5 [En línea] Disponible en:
<https://developer.mozilla.org/es/docs/HTML/HTML5> [Última visita: 15/06/2015]
- [5] Elika J. Etemad Cascading Style Sheets (CSS) [En línea] Disponible en:
<http://www.w3.org/TR/CSS/> [Última visita: 10/06/2015]
- [6] CSS3 [En línea] Disponible en:
<https://developer.mozilla.org/es/docs/Web/CSS/CSS3> [Última visita: 10/06/2015]
- [7] History of PHP [En línea] Disponible en: <http://php.net/manual/en/history.php.php> [Última visita: 09/06/2015]
- [8] JavaScript [En línea] Disponible en:
<https://developer.mozilla.org/es/docs/Web/JavaScript> [Última visita: 15/06/2015]
- [9] Web oficial CakePHP [En línea] Disponible en: <http://cakephp.org/> [Última visita: 01/09/2015]
- [10] Book Oficial del CakePHP [En línea] Disponible en:
<http://book.cakephp.org/2.0/en/index.html> [Última visita: 01/09/2015]
- [11] Web oficial jQuery [En línea] Disponible en: <http://jquery.com/> [Última visita: 25/08/2015]
- [12] Web oficial gMap3 [En línea] Disponible en: <http://gmap3.net/> [Última visita: 25/08/2015]
- [13] Web oficial html2canvas [En línea] Disponible en:
<http://html2canvas.hertzen.com/> [Última visita: 25/08/2015]
- [14] Web oficial bootstrap [En línea] Disponible en: <http://getbootstrap.com/> [Última visita: 25/08/2015]

Anexo 1

[15] Web oficial MySql [En línea] Disponible en <https://www.mysql.com/> [Última visita: 25/08/2015]

[16] Web oficial Mandrill [En línea] Disponible en <https://www.mandrill.com/> [Última visita: 25/08/2015]

[17] Web oficial Paypal [En línea] Disponible en <https://www.paypal.com/es/home> [Última visita: 25/08/2015]

[18] Banco Sabadell. *TPV Virtual Manual operativo y de instalación* [En línea]
Disponible en: <https://www.bancsabadell.com/cs/Satellite/SabAtl/TPV-Virtual/1191332200922/es/> [Última visita: 15/08/2015]

[19] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, y D. Thomas, «Manifiesto por el Desarrollo Ágil de Software», feb-2001. [En línea]. Disponible en:

<http://www.agilemanifesto.org/iso/es/manifesto.html> [Última visita: 15/08/2015]

[20] Desarrollo iterativo e incremental [En línea] Disponible en:
<http://www.proyectosagiles.org/desarrollo-iterativo-incremental> [Última visita: 19/08/2015]

[21] Sommerville, 2012. Ingeniería del Software. 9a Edición, Addison-Wesley. 2012.

[22] Web oficial trello [En línea] Disponible en: <https://trello.com/home> [Última visita: 19/08/2015]

[23] Alistair Cockburn, “Writting Effective Use Cases”. Addison-Wesley, 2000.

[24] E.Gamma, Patrones de Diseño. Addison-Wesley. 2002.

[25] Web oficial CakePhp Entendiendo el Modelo - Vista - Controlador [En línea]
Disponible en: <http://book.cakephp.org/2.0/es/cakephp-overview/understanding-model-view-controller.html> [Última visita: 14/08/2015]

Anexo 1

Ejemplo de práctico

Para una mejor comprensión del funcionamiento del sistema, realizaremos varios ejemplos ilustrativos, explicando mediante capturas de pantalla los procesos más relevantes de la plataforma.

1 Reserva de una habitación

Paso 1.1: El usuario accede a la página principal del buscador de alojamiento <http://emancipia.es>

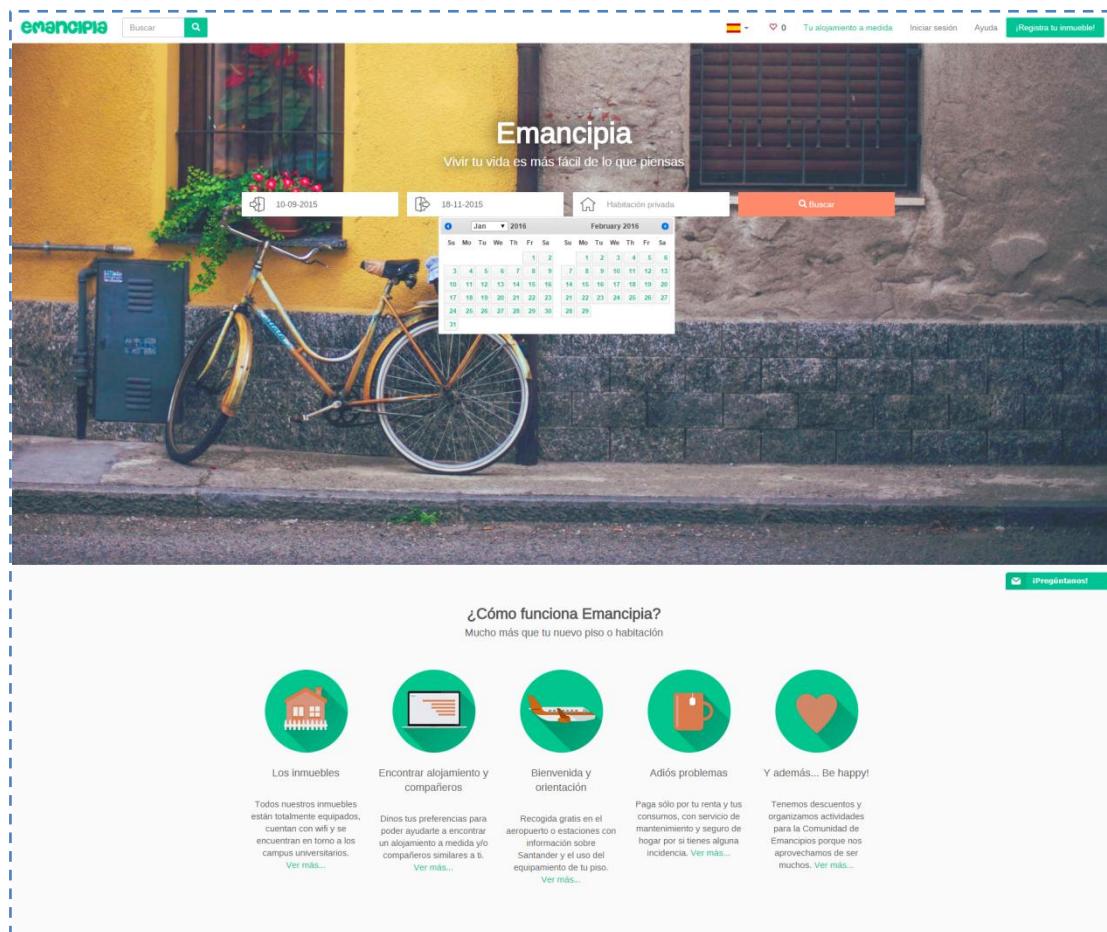


Ilustración 45 Ejemplo 1.1

Anexo 1

Paso 1.2: El sistema muestra al usuario una serie de habitaciones en función de la búsqueda. El usuario en este punto puede: seguir buscando mediante el buscador, añadir a favoritos o enviar por correo electrónico las habitaciones que más le gusten o seleccionar una habitación.

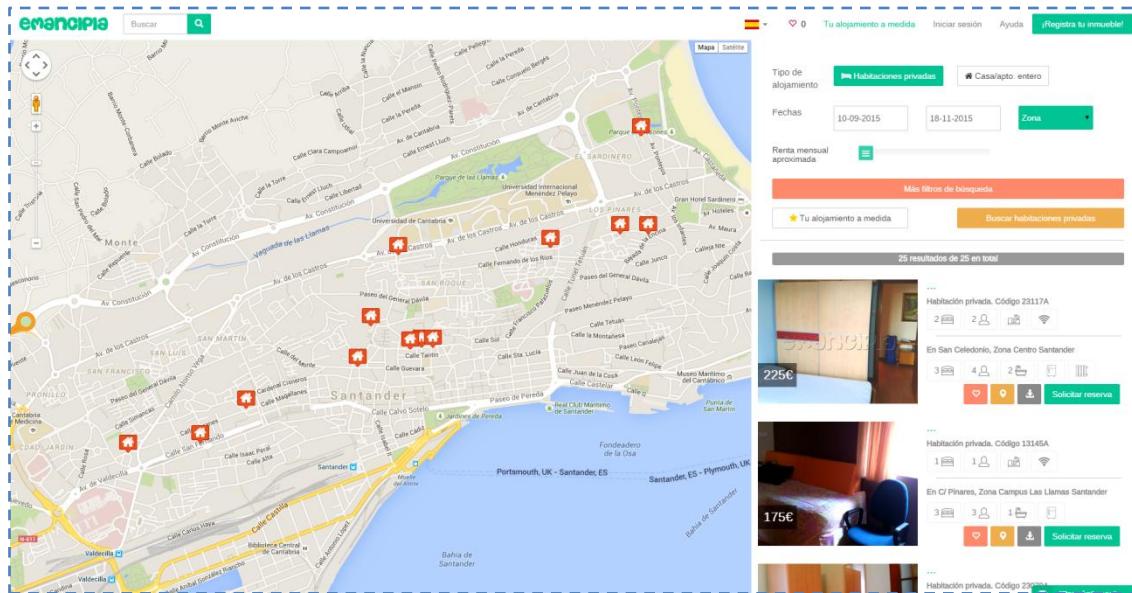


Ilustración 46 Ejemplo 1.2

Paso 1.3: El usuario ve las características de la habitación, las fotos de las estancias, la disponibilidad de fechas para la reserva y la ubicación del inmueble.

Anexo 1

The screenshot shows a listing for a room on the Emancipia website. At the top, there's a header with the Emancipia logo, a search bar, and various navigation links including 'Iniciar sesión' and 'Ayuda'. A green button on the right says '¡Registra tu inmueble!'. Below the header is a large photograph of a room featuring a white desk with a lamp and a green chair, a wooden floor, and a bed with a blue patterned bedsheet. To the right of the photo, there's a price box showing '185 € al mes' with a green tag icon, a 'Consultar condiciones' link, and date fields for '16-02-2016' and '18-02-2016'. A large green 'Solicitar reserva' button is below these. Further down, there's a section for sharing with social media icons for heart, email, Facebook, Twitter, and Google+. A green 'Pregunta sobre esta habitación' button is also present. Below this main section, there are tabs for 'Galería', 'Disponibilidad', 'Características', 'Habitaciones', and 'Mapa'. A 'Galería fotográfica del Inmueble' section shows a grid of small thumbnail images. At the very bottom of the page, there's a footer with a progress bar and a 'Vídeo' button.

Ilustración 47 Ejemplo 1.3

Paso 4.1: Después de que el usuario haya seleccionado unas fechas de reserva correctas, pasa a formalizar la reserva mediante el pago. El pago le puede realizar mediante los tres pagos aceptados: PayPal, Pasarela de Pago Virtual (tarjeta de crédito) o transferencia bancaria.

Anexo 1

The screenshot shows the 'Formulario de Reserva' (Reservation Form) on the Emancipia website. At the top, there's a navigation bar with 'emancipia', a search bar, language selection (Spanish), user stats (0 favorites), a link to 'Tu alojamiento a medida', a 'Iniciar sesión' button, 'Ayuda', and a green '¡Registra tu inmueble!' button.

The main content area has a progress bar at the top:

- Buscar piso (Completed)
- Inmueble encontrado (Completed)
- Rellenar datos (In Progress)
- Confirmar datos (Not Started)
- Realizar pago (Not Started)

Datos personales

Nombre *: inquilino

Email *: inquilino@emancipia.net

Métodos de pago *

Pago con tarjeta (Sabadell, Servired, MasterCard, Verified by VISA)

Transferencia Bancaria (Sabadell)

PayPal (PayPal logo)

Tu elección:

Habitación privada. Código 23031B
1 dormitorio, 1 baño, 185€/mes, WiFi

En San Celedonio, Zona Centro Santander

Fecha de Entrada: 07-09-2015 | Fecha de Salida: 18-02-2016

Continuar con el pago | **Volver**

Servicios

- ¿Cómo funciona Emancipia?
- Encuentra tu nuevo hogar
- Información para propietarios
- Centro de ayuda

Emancipia

- Sobre nosotros
- Contacto
- Trabaja con nosotros
- Invierte en Emancipia

Interactúa

- Facebook
- Twitter
- Google +
- Youtube
- Linkedin
- Pinterest

Pago seguro

- Sabadell
- Servired
- MasterCard
- Verified by VISA
- PayPal

© Emancipia Servicios de Alojamiento S.L. 2013-2015. Todos los derechos reservados. El uso de este sitio web implica la aceptación de los [Términos y Condiciones de Uso](#) así como la [Política de Privacidad](#) y la de [Uso de Cookies](#).

Ilustración 48 Ejemplo 1.4

Paso 1.5: Una vez llenados todos los datos de la reserva, el usuario pasa a confirmar los datos. En esta pantalla al usuario se le da la posibilidad de cambiar sus datos de reserva.

Anexo 1

The screenshot shows the 'Pasarela de pago Segura de Emancipia' (Secure Payment Gateway) interface. At the top, there's a navigation bar with 'emancipia', a search bar, language selection (Spanish), user count (0), a link to 'Tu alojamiento a medida', a 'Iniciar sesión' button, 'Ayuda', and a green button to '¡Registra tu inmueble!'. Below the navigation, there's a progress bar with five steps: 'Buscar piso' (checked), 'Inmueble encontrado' (checked), 'Rellenar datos' (checked), 'Confirmar datos' (unchecked), and 'Realizar pago' (unchecked). A list of instructions below the steps says: 1. Compruebe si los datos son correctos 2. Click en Continuar con el pago para proceder ha realizar el pago. A section titled 'Tus Datos' lists: Titular: inquilino, Concepto: 23031B -, Cantidad: 120 €. At the bottom are two buttons: 'Continuar con el pago' (green) and 'Cancelar' (orange).

Ilustración 49 Ejemplo 1.5

Paso 1.6: El usuario verifica sus datos y procede a conectarse con la pasarela de pago para proceder al pago. Este es el último paso, una vez cumplimentado se le redigire a la página web del banco para proceder al pago.

The screenshot shows the 'Sabadell' payment gateway interface. At the top, it says 'Selección su idioma | Castellano'. Below that, a progress bar shows four steps: 1. Selección método de pago, 2. Comprobación autenticación, 3. Solicitud Autorización, and 4. Resultado Transacción. Step 1 shows 'Datos de la operación' with details: Importe: 120,00 €, Comercio: EMANCIPIA SERVICIOS DE AL, Terminal: 327532685-1, Pedido: 1441660859, Fecha: 07/09/2015 23:21, Descripción producto: 23031B -. Step 2 shows 'Pagar con Tarjeta' with fields for N° Tarjeta, Caducidad (mm/aa), and Cód. Seguridad. Buttons for 'Cancelar' and 'Pagar' are at the bottom. The 'Servired' logo is visible at the bottom left. The footer includes 'Powered by Redsys' and a copyright notice: '(c) 2014 Redsys Servicios de Procesamiento, SL - Todos los derechos reservados. - Aviso legal - Privacidad'.

Ilustración 50 Ejemplo 1.6

Paso 1.7: El usuario procede a realizar el pago. Una vez rellenado el pago la API del banco enviará una respuesta a la plataforma para que esta puede comprobar el estado del pago del usuario.

2 Preferencia de Alojamiento

Paso 2.1: El usuario accede al formulario de alojamiento <http://pisos.emancipia.net> y rellena los datos del alojamiento. Todos los datos son obligatorios.

emancipia Buscar   0 Tu alojamiento a medida Iniciar sesión Ayuda ¡Registra tu inmueble!

¡Hola! Bienvenido al Programa Emancipia

Rellena este formulario para comenzar el proceso de búsqueda. Un asistente de Emancipia te atenderá personalmente para encontrar un alojamiento a tu medida. Es totalmente gratuito y no compromete a nada.

Preferencias de alojamiento

Habitación privada en piso compartido

Con amigos

Introduce el email de las personas con las que quieres compartir piso. A tu amigo le llegarán un correo informándole sobre las preferencias que has elegido.

amigo1@emancipia.net
amigo2@emancipia.net

Añadir otro amigo

10-09-2015 26-11-2015

250 € Campus Las Llamas

Inglés Masculino 18 - 20 años

¿Quieres añadir más información para facilitar nuestro trabajo?

Datos de contacto

inquilino

inquilino@emancipia.net

Si nos proporcionas alguno de los métodos siguientes podremos contactar contigo más fácilmente:

WhatsApp Telegram Line Viber Teléfono

+34 666 123 456

Al hacer click aceptas las condiciones y términos de uso Aviso legal y condiciones

Enviar Formulario



 ¡Pregúntanos!

Ilustración 51 Ejemplo 2.1

Anexo 1

Paso 2.2: Una vez completados y enviados los datos, la aplicación comprueba el estado del usuario y procede a su alta en la plataforma. Al usuario se le pide más datos para llenar, estos segundos datos son opcionales.

The screenshot shows a user profile creation form on the emancipia platform. At the top, there is a navigation bar with the logo 'emancipia', a search bar, and links for 'Iniciar sesión' (Log in) and 'Ayuda' (Help). A green button on the right says 'Registra tu inmueble!' (Register your property!).

The main area starts with a greeting '¡Hola Emm-inq!' and a message: 'Para poder ayudarte mejor es importante que nos digas dónde vas a estudiar o trabajar y de dónde vienes.'

Two tabs are visible: 'Estudiantes, profesores e investigadores' (selected) and 'Trabajadores y otros'. Below these tabs is a dropdown menu set to 'Estudiante'. Underneath it are three more dropdown menus: 'Uni. de Cantabria', 'Facultad de ciencias', and 'Ing. Informática'.

The next section is titled '¿De dónde eres?'. It includes dropdown menus for 'Estados Unidos' (set to 'Chicago') and 'Chicago' (set to 'University Chicago'). A dropdown menu for '¿De qué universidad o centro educativo/investigación vienes?' is open, showing 'University Chicago'.

A question '¿Quieres añadir más información para facilitar nuestro trabajo?' is followed by a text input field containing '¿Quieres añadir más información para facilitar nuestro trabajo?' and a 'Completar las preferencias' button.

At the bottom right is a green button with an envelope icon labeled 'iPregúntanos!'

Ilustración 52 Ejemplo 2.2

Paso 2.3: El sistema verifica los datos y procede a informar al usuario que puede modificarlos y que un asistente atenderá su petición.

Anexo 1



Ilustración 53 Ejemplo 2.3

Paso 2.4: El sistema envía un correo electrónico con una plantilla predefinida con información.

emancipia

Hello inquilino,

Thank you for requesting accommodation information on Emancipia.

In a few hours or days you will receive information about accommodation in response to the preferences you have indicated to us.

Remember that you can edit or modify your accommodation preferences entering your User Area on the Emancipia application web.

[Access to your User Area](#)

[Are you new? Request your password](#)

If you have any questions, please do not hesitate to [contact us](#).

Emancipia is a startup from the University of Cantabria specialized in accommodation. To learn more you can access [emancipia.es](#) or look for us on [Facebook](#).

--

 Un saludo del Equipo de Emancipia.
info@emancipia.es
[\(+34\) 648 100 288](tel:+34648100288) – [\(+34\) 942 035 704](tel:+34942035704)

Information and Administration
info@emancipia.es [\(+34\) 942 035 704](tel:+34942035704)
Emancipia Servicios de Alojamiento S.L. C.I.F. B-39774468
(CDTUC) Fase A Oficina 207 Avd. de los Castros s/n. 39005 Santander. Spain.

Emergencies and Fixes
fix@emancipia.es [\(+34\) 648 100 288](tel:+34648100288)



SI NO ES PRECISO, NO IMPRIMA ESTE MENSAJE Este mensaje de correo electrónico y sus archivos adjuntos son confidenciales y están legalmente protegidos. Se dirige exclusivamente al destinatario o destinatarios. No está autorizado el acceso a este mensaje por otras personas. Sólo se permite su uso a quienes estén expresamente autorizados. Si usted no es la persona a la que va dirigido, cualquier uso, tratamiento, información, copia o distribución y cualquier acción u omisión basada en la información contenida en este mensaje queda prohibida y es ilegal. Así mismo, de acuerdo a la Ley 15/1999 sobre Protección de Datos personales, le informamos que los datos personales utilizados para la presente comunicación serán incluidos en un fichero debidamente inscrito en el Registro de la Agencia Española de Protección de Datos, con la finalidad de posibilitar las comunicaciones a través de correo electrónico de EMANCIPIA, S.L. con los distintos contactos que ésta mantiene dentro del ejercicio de su actividad y el envío de información comercial de nuestra compañía. Sin perjuicio de ello se le informa de que usted podrá ejercitar los derechos de Acceso, Rectificación, Cancelación y Oposición dirigiéndose por escrito a EMANCIPIA, S.L. en Centro de Desarrollo Tecnológico de la Universidad de Cantabria. (CDTUC) Fase A Oficina 207 Avd. de los Castros s/n. 39005 Santander. Spain.

Ilustración 54 Ejemplo 2.4