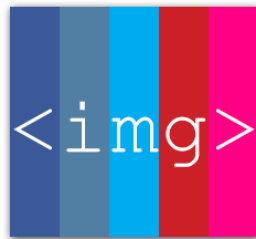


Trabajo de Final de Grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultat de Matemàtiques - Universitat de Barcelona



AGREGADOR SOCIAL DE IMÁGENES

Irama Aliaga Guallar

Director: Eloi Puertas Prats

Realizado en: Departamento de Matemática Aplicada y Análisis. UB

Barcelona, 20 de julio de 2014

Agradecimientos

A mi pareja, por su apoyo y ayuda incondicional frente a vientos y mareas.

A mi familia, que es la que siempre ha creído en mí.

A mis amigos, por sus ánimos constantes.

A mis compañeros de trabajo, por haber sufrido mis locuras.

A mis compañeras de piso, por haberme escuchado y aconsejado.

Y finalmente a mi tutor, por haberme brindado esta oportunidad.

Gracias a todos.

Resumen

En las últimas décadas la tecnología está a la orden del día y los smartphones o teléfonos inteligentes, se han convertido en la principal vía de acceso a las redes sociales. Hasta la fecha, el acceso a Internet ha estado unido a los ordenadores sin embargo, con el surgimiento de estos dispositivos esta forma de acceso ha cambiado drásticamente creando nuevas necesidades de adaptación.

Nace entonces la idea o concepto de la web 2.0. que apunta a una nueva generación de sitios donde los contenidos son dinámicos y producidos por los usuarios del portal dando cabida a la web como un punto de encuentro y forma de relación, y al smartphone como medio de transporte.

Este trabajo final de grado consiste en diseñar, implementar y desarrollar una aplicación web con la finalidad de poder agregar el contenido multimedia de diversas redes sociales. El propósito de esta aplicación es proporcionar al cliente una herramienta económica, fácil y sencilla la cual le permita organizar sus imágenes o fotografías que albergue en estas comunidades virtuales. Para ello, la aplicación web tendrá que ser accesible desde cualquier lugar y adaptable a cualquier dispositivo utilizado.

El proyecto está basado en la arquitectura cliente-servidor. En el lado del servidor se encuentra el motor de la aplicación, desarrollado en los lenguajes PHP y MySQL, además del control de las APIs de Facebook e Instagram que nos permiten la recopilación de imágenes. En el lado del cliente el diseño de la interfaz corre a cargo de HTML5, CSS3 y Javascript, el cual se basa en Initializr, un generador de plantillas personalizable y responsive que posibilita la adaptación del sitio web a todos los dispositivos existentes.

Resum

En les últimes dècades la tecnologia es troba a l'ordre del dia i els smartphones o telèfons intel·ligents, s'han convertit en la principal via d'accés a les xarxes socials. Fins ara, l'accés a Internet ha estat unit als ordinadors però, amb el sorgiment d'aquests dispositius aquesta forma d'accés ha canviat dràsticament creant noves necessitats d'adaptació.

Neix llavors la idea o concepte de la web 2.0. que apunta a una nova generació de llocs on els continguts són dinàmics i produïts pels usuaris del portal donant cabuda a la web com un punt de trobada i forma de relació, i al telèfon intel·ligent com a mitjà de transport.

Aquest treball final de grau consisteix a dissenyar, implementar i desenvolupar una aplicació web amb la finalitat de poder afegir el contingut multimèdia de diverses xarxes socials. El propòsit d'aquesta aplicació és proporcionar al client una eina econòmica, fàcil i senzilla la qual li permeti organitzar les seves imatges o fotografies que albergui en aquestes comunitats virtuals. Per a això, l'aplicació web haurà de ser accessible des de qualsevol lloc i adaptable a qualsevol dispositiu utilitzat.

El projecte està basat en l'arquitectura client-servidor. Al costat del servidor es troba el motor de l'aplicació, desenvolupat en els llenguatges PHP i MySQL, a més del control de les APIs de Facebook i Instagram que ens permeten la recopilació d'imatges. Al costat del client el disseny de la interfície és a càrrec d'HTML5, CSS3 i Javascript, el qual es basa en Initializr, un generador de plantilles personalitzable i responsive que possibilita l'adaptació del lloc web a tots els dispositius existents.

Abstract

In the last decades, the technology has been such a mainstream topic. Smartphones have become the main access to social networks. Until now, access to the Internet has been attached to computers. However, with the emergence of these devices, this way of access has changed dramatically, creating new adaptation needs.

It was then when web 2.0 concept was born, which is focused on a new generation of websites, where contents are dynamic and produced by users, leading to a meeting point and a form of relationship, where the smartphone would be the link

This project consist in developing a web application, in order to add multimedia content from different networks. The purpose of this application is to provide the customer a cheap, easy and simple tool which allows you to organize your images or photographs that customer has in his networks. To do this, the web application will must be accessible from anywhere and responsive to any device used.

The project is based on client-server architecture. On the server side there is the engine of application, developed in PHP and MySQL languages, besides the control of Facebook and Instagram APIs that allow us to collect images. On the client side, interface design is provided by HTML5, CSS3 and Javascript, which is based on Initializr, a template generator to build customizable and responsive websites.

DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

API	Application Programming Interface (Interfaz de programación de aplicaciones)
APP	Abreviatura de aplicación
CSS	Cascading Syle Sheets (Hojas de Estilo en Cascada)
CSS3	Cascading Syle Sheets, version 3 (Hojas de Estilo en Cascada, versión 3)
FQL	Facebook Query Languaje (Lenguaje de consultas de Facebook)
FRAMEWORK	Marco de trabajo
FTP	File Transfer Protocol (Protocolo de Transferencia de Archivos)
GUI	Graphical User Interface (Interfaz Gráfica de Usuario)
HBP5	HTML5 Boilerplate
HTML	HyperText Markup Language (Lenguaje de marcas de hipertexto)
HTML5	HyperText Markup Language, version 5 (Lenguaje de marcas de hipertexto, versión 5)
HTTP	Hypertext Transfer Protocol(protocolo de transferencia de hipertexto)
MARKET	Mercado de aplicaciones
MICROBLOGGING	Servicio de mensajes cortos
MySQL	Structured Query Language (Lenguaje Estructurado de Consultas)
OAuth	Open Authorization (Autorización Abierta)
PHP	Hypertext Pre-processor
SNIPPETS	Pequeñas partes reusables de código fuente, código binario o texto
SSL	Secure Sockets Layer (Capa de conexión segura)
URL	Uniform Resource Locator (Localizador de Recursos Uniformes)

Índice de contenido

1.Introducción.....	8
1.1.Motivación.....	8
1.2.Planteamiento del problema.....	8
1.3.Objetivo	9
1.4.Estructura del documento.....	10
2.Estado del arte.....	11
2.1.Redes sociales.....	11
2.2.Agregadores sociales.....	13
2.3.Aplicaciones web en HTML5.....	14
3.Análisis.....	16
3.1.API - Application Programming Interface	16
3.2.Aplicaciones web vs aplicaciones nativas.	17
3.3.Análisis de requisitos	19
3.4.Casos de uso.....	22
3.5.Lenguajes, librerías y herramientas.....	23
4.Diseño e implementación.....	28
4.1.Arquitectura de la aplicación.....	28
4.2.Mapa conceptual de Navegación	30
4.3.Base de datos.....	31
4.4.Back-end. Organización ficheros PHP:	34
4.5.Diseño e implementación del Front-End.....	43
4.6.Test de Usabilidad.....	46
5.Planificación y plan de empresa.....	49
5.1.Planificación.....	49
5.2.Plan de empresa.....	51
6.Conclusiones.....	57
6.1.Mejoras y Ampliaciones.....	57
7.Bibliografía.....	59
8.Anexos.....	60
8.1. Anexo I: Contenido ZIP	60
8.2. Anexo IV: Manual de usuario.....	65

Índice de ilustraciones

Ilustración 1: Esquema general de conexión de varias APIs.....	18
Ilustración 2: Acceso de una App web a un perfil de usuario mediante una API.....	20
Ilustración 3: Arquitectura Modelo-Vista-Controlador.....	24
Ilustración 4: Casos de uso de la aplicación web	26
Ilustración 5: Página inicial de Initializr Bootstrap (index.php).....	29
Ilustración 6: Arquitectura cliente-servidor.....	32
Ilustración 7: Aquitectura principal de la aplicación	33
Ilustración 8: Mapa conceptual de navegación de	34
Ilustración 9: Modelo base de datos entidad-relación de de	36
Ilustración 10: Modelo de la base de datos relacional de	38
Ilustración 11: Datos de la aplicación registrada en Facebook.....	42
Ilustración 12: Aplicación solicitando permisos al usuario.....	44
Ilustración 13: Login en Facebook mediante el uso del protocolo OAuth.....	47
Ilustración 14: Parametros de configuración de la aplicación registrada en Instagram.....	50
Ilustración 15: Concepto de diseño mobile-first.....	51
Ilustración 16: Vista reducida de la aplicación en dispositivos pequeños.....	52
Ilustración 17: Vista ampliada de la aplicación en dispositivos grandes.....	52
Ilustración 18: Respuestas 1 y 2 del test de usabilidad.....	54
Ilustración 19: Respuestas 3 y 4 del test de usabilidad.....	55
Ilustración 20: Respuestas 5, 6 y 7 del test de usabilidad.....	55
Ilustración 21: Respuestas 8 y 9 del test de usabilidad.....	56
Ilustración 22: Tareas principales del diagrama de Gantt.....	57
Ilustración 23: Dagrama de Gantt.....	58
Ilustración 24: De izquierda a derecha, características del usuario free y premium.....	59
Ilustración 25: Previsión a dos años vista de usuarios free y premium.....	60
Ilustración 26: Ingresos frente a costes de la aplicación	63
Ilustración 27: Curva del punto muerto.....	64
Ilustración 28: Contenido de la carpeta src.....	68
Ilustración 29: Carpetas básicas del proyecto.....	68
Ilustración 30: Contenido de la carpeta inter.....	69
Ilustración 31: Contenido de la carpeta css.....	70

Ilustración 32: Contenido de la carpeta extras.....	70
Ilustración 33: Contenido de la carpeta fonts.....	71
Ilustración 34: Contenido de la carpeta img.....	71
Ilustración 35: Contenido de la carpeta lib.....	72
Ilustración 36: Contenido y subcontenido de la carpeta js.....	72
Ilustración 37: Registro de un usuario.....	73
Ilustración 38: Logar un usuario.....	73
Ilustración 39: Menú que muestra la opción Logout.....	74
Ilustración 40: Activar red social.....	74
Ilustración 41: Vista de redes sociales activadas.....	74
Ilustración 42: Vista de las fotos de Facebook.....	75
Ilustración 43: Crear un nuevo álbum de fotos.....	75
Ilustración 44: Lista de álbums con el nuevo álbum insertado.....	76
Ilustración 45: Nuevo álbum creado.....	76
Ilustración 46: Seleccionar red social para añadir sus fotos a un álbum.....	77
Ilustración 47: Selección de fotos para añadir a un álbum.....	77
Ilustración 48: Vista principal de una foto.....	78
Ilustración 49: Borrar fotos de un álbum.....	78
Ilustración 50: Confirmación para borrar un álbum.....	79
Ilustración 51: Acceso a la configuración de la cuenta de usuario.....	79
Ilustración 52: Datos del usuario.....	80

Índice de tablas

Tabla 1: Redes sociales más populares en la actualidad.....	15
Tabla 2: Agregadores sociales más populares en la actualidad.....	16
Tabla 3: Ventajas e inconvenientes de una aplicación web.....	22
Tabla 4: Ventajas e inconvenientes de una aplicación nativa.....	23
Tabla 5: Costes de personal.....	61
Tabla 6: Costes materiales.....	62

1. Introducción

1.1. Motivación

Durante el siglo XX y XXI el desarrollo tecnológico ha ido creciendo de forma exponencial. Actualmente la tecnología ha cambiado indudablemente nuestra forma de vida, y juega un papel muy importante en nuestro día a día, llegando a veces al extremo de ser imprescindible. La evolución del teléfono móvil al dispositivo, actualmente conocido como smartphone o teléfono inteligente, a causado un gran impacto en la sociedad. Su principal uso, realizar llamadas telefónicas, se ha visto relegado por una gran cantidad de servicios adicionales.

Junto con los smartphones comenzaron a desarrollarse otros pequeños dispositivos como tablets y fablets que, debido a su potencia y capacidad llegan incluso a suplantar la funcionalidad de un ordenador. Estos aparatos nos permiten acceder a Internet cada vez con más rapidez estando en cualquier parte del mundo. Además nos ofrecen infinidad de posibilidades como ver la televisión en directo, controlar nuestras alarmas del hogar, suplantar a la cámara fotográfica o simplemente entretenernos haciendo las veces de una videoconsola.

Paralelamente al desarrollo de los smartphones, las redes sociales se han visto beneficiadas y han crecido de la mano de ésta tecnología. Estas redes establecen nuevos canales de comunicación dado que se permiten que el usuario comparta opiniones, curiosidades, fotos, vídeos o cualquier “momento” que desee de forma inmediata. Este fenómeno está fomentando una nueva necesidad, y esta necesidad da lugar a nuevos modelos de negocio.

Debido a estos dos grandes cambios en la sociedad que comportan el uso constante de distintos tipos de dispositivos electrónicos y las posibilidades que estos ofrecen junto a la gran aceptación y crecimiento de las comunidades virtuales en, surge la motivación de crear esta aplicación web que nos permite beneficiarnos de ambas situaciones.

1.2. Planteamiento del problema

Con la llegada de los nuevos dispositivos nombrados anteriormente el mundo tecnológico se ha visto obligado a evolucionar para así poder adaptarse, dar soporte y soluciones a sus clientes originando un nuevo modelo de negocio ya habíamos comentado.

Son cada día más las empresas que se dedican exclusivamente a desarrollar aplicaciones para dispositivos móviles y muchas otras las que deciden externalizar en ellas el desarrollo de sus aplicaciones. Pero no todo es tan fácil o sencillo, existen una serie de desventajas e inconvenientes a las cuales intentaremos buscar una solución.

Contratar a una empresa para que desarrolle una aplicación móvil o mejor dicho una aplicación nativa, requiere de un gran coste elevado que se verá multiplicado por el número de plataformas a las que se desee abarcar. Es decir, al existir varios sistemas operativos que

controlan los diferentes dispositivos existe la necesidad de crear una aplicación exclusiva para que sea compatible con cada sistema operativo que se presente.

Lo expuesto anteriormente nos lleva a la conclusión de que los costes de realización de estas aplicaciones, para que sean compatibles en cualquier dispositivo móvil e incluso con los ordenadores, necesitaría de una inversión de capital excesivamente elevada con la que actualmente la mayoría de las empresas existentes no se pueden permitir.

Entonces pensamos, ¿No sería mejor crear una aplicación versátil?, ¿es esto posible sin recurrir a las aplicaciones nativas? La respuesta a ambas preguntas es afirmativa y de aquí nace nuestro objetivo: crear una web 2.0. o lo que se entiende como aplicación web. Pero, ¿vamos a necesitar para ello grandes recursos? En este caso la respuesta es negativa. Como explicaremos más adelante en este mismo documento, para crear una aplicación web los recursos necesarios como desarrolladores son más bien pocos y los requisitos que debe cumplir el usuario para acceder a ella bastan con tener un dispositivo cualquiera que albergue un navegador web y permita la conexión a Internet.

Aproximadamente en el 2005 surge la idea o concepto de la web 2.0 que se entiende como el resultado del crecimiento de nuevas aplicaciones y sitios web. Los pilares de esta nueva idea vienen dados en parte por la evolución de la tecnología, de las redes sociales y del desarrollo de los servicios web enfocados al cliente. El nacimiento de este nuevo concepto junto a las nuevas versiones de los lenguajes HTML5, CSS3 y Javascript, que juntos proporcionan una potente herramienta de desarrollo web, nos permiten encontrar la solución al problema planteado anteriormente.

1.3. Objetivo

Analizando todo lo que hemos comentado en los apartados previos y teniendo claro que el objetivo final era crear una aplicación web en HTML5, CSS3 y JQuery adaptable y compatible a los diferentes dispositivos, necesitábamos determinar lo más importante, qué servicio iba a ofrecer a la sociedad nuestra aplicación.

En el planteamiento del problema hemos comentado la tendencia y necesidad de usar constantemente las redes sociales y compartir nuestro día a día. Cada vez es más común ver que la gente usa el móvil o tablet como cámara de fotos, puesto que la comodidad y resolución de éstas puede incluso superar las cámaras cotidianas.

Es muy común fotografiar cualquier acontecimiento y seguidamente publicarlo en el muro de Facebook o irse de vacaciones y subir una imagen del paisaje e incluso autofotografiarse a uno mismo por pensar que uno está guapo. Esta última idea ha llegado a crear una nueva tendencia. Quién no ha oído hablar del *selfie*? o ha visto una foto de un amigo publicada en Twitter con el hastag *#selfie*? Dada la gran aceptación de la población al uso de las redes sociales y la necesidad constante de compartir contenido multimedia, decidimos crear una aplicación que fuera capaz de manejar toda esta información.

Por norma general el usuario de Internet tiene uno o más perfiles en las diferentes redes sociales existentes, que algunas de ellas como Instagram o Pinterest, están en auge y se dedican única y exclusivamente a conformar una red social temática centrada en la compartición de imágenes o fotografías.

Cada usuario almacena fotos en las diferentes redes sociales en las que disponga de un perfil. Partiendo de esta base llegamos a la conclusión de que sería muy buena idea y sobretodo útil crear una aplicación, como antes hemos dicho, adaptable y responsive para poder agregar contenido multimedia, en concreto imágenes y fotografías, permitiendo a nuestro usuario recuperar, controlar y organizar de una manera más fácil, rápida y sencilla sus fotografías.

1.4. Estructura del documento

En el presente documento explicaremos el desarrollo completo del proyecto . Una vez expuesta en la introducción la motivación para su desarrollo, que problema se nos presenta y que objetivo hemos definido en el capítulo dos vamos a explicar el estado del arte, es decir, nos introduciremos en el mundo de las redes sociales, explicaremos como estos dan lugar a los agregadores de redes sociales y como mediante el uso de una aplicación web en HTML5 podemos desarrollar una web 2.0.

En el capítulo 3 daremos paso al análisis del proyecto. Planteamos como se realiza el acceso a la información que las redes sociales nos proporcionan, que ventajas obtenemos de crear una aplicación web analizado los requisitos que necesitaremos para desarrollarla y comentaremos que lenguajes, librerías y herramientas utilizamos para ello.

Una vez realizado el análisis, en el capítulo cuatro explicaremos el diseño y la implementación al completo de todas las partes de la aplicación . Detallaremos su arquitectura base, como está organizado y conformado el proyecto en sí, los diferentes elementos que lo forman así como el desarrollo del back-end y front-end del mismo, es decir, el motor de la aplicación y la interfaz gráfica de usuario y como conclusión que pruebas y resultados hemos obtenido.

En el capítulo cinco, exponemos la planificación del proyecto así como un pequeño plan de empresa que nos dará una idea global de los costes y viabilidad económica que conllevaría llevarlo a cabo así como los requisitos para el éxito de esta aplicación.

Finalmente en el último capítulo, el seis, exponemos las conclusiones del desarrollo de la aplicación web así como las posibles mejoras y ampliaciones que podrían llevarse a cabo.

2. Estado del arte

En este capítulo examinaré más minuciosamente los objetivos que en el apartado anterior hemos mencionado. Hablaré de las redes sociales más exitosas y el impacto que están teniendo en la sociedad. También explicaré como su gran crecimiento en los últimos años ha dado lugar a los agregadores sociales, aplicaciones capaces de reunir información de diversas fuentes sociales.

Finalmente haré hincapié de cómo el desarrollo de los nuevos dispositivos como smartphones, tablets, fablets o netbooks y portátiles, que tienen diferentes tamaños y sistemas operativos han provocado que los desarrolladores tengan que encontrar nuevas soluciones para hacer aplicaciones compatibles y adaptables a las diferentes plataformas existentes.

2.1. Redes sociales

La evolución de las redes sociales en los últimos años ha superado cualquier previsión. Podríamos decir que desde que estalla la burbuja de Internet en el 2000 hasta la actualidad es cuando más se han desarrollado.

Aproximadamente por el 2003 nace Myspace y Friendster, redes sociales pioneras. Estos sitios comienzan a crecer de modo exponencial y a popularizarse estas comunidades que permiten mantener el contacto y crear nuevas relaciones con otros usuarios. Poco a poco se han ido ampliando la variedad de servicios que ofrecen, desde publicar y compartir contenido multimedia hasta la creación de grupos especializados y difusión de eventos entre otras funcionalidades.

Progresivamente se fueron transformando en un nuevo medio de relación social y comenzaron a surgir nuevas redes comunidades virtuales como Facebook, que en 2005 se creó con el objetivo de relacionar a los estudiantes de la universidad de Harvard. Actualmente es la red social por preferencia contando en la actualidad con más de 1.230 millones de usuarios activos mensuales.

Frente a este boom de crecimiento y gran aceptación de los usuarios, muchas empresas como Google, Yahoo o Twitter comienzan a existir y a desarrollarse como red social. Frente a esta competitividad comienzan a crearse redes especializadas en una temática en concreto, como por ejemplo Instagram, actualmente comprado por Facebook, que centra sus servicios en compartir fotos con otros usuarios.

La gran evolución de la red social ha provocado un nuevo modelo de negocio que aprovecha la implementación de diversas aplicaciones de terceros para ser incluidas dentro de las propias plataformas sociales.

Existe un tipo de clasificación para estas comunidades sociales, podríamos clasificarlas como redes verticales u horizontales. Las horizontales son aquellas que buscan proveer herramientas para la interrelación en general, tales como Facebook o Google+, mientras que las verticales son mucho más específicas y orientadas a un público específico como puede ser LinkedIn, exclusiva para relaciones laborales, o en una actividad en concreto como Youtube, un sitio web donde los usuarios tan solo comparten y suben vídeos.

Algunas de las redes sociales actuales a destacar son:

<i>Red Social</i>	<i>Características</i>
Facebook	Creada en 2004. Es una de las redes sociales más completas y populares en la actualidad. Permite crear listas de amigos, compartir y etiquetar fotos y vídeos. Contiene chat integrado. Da la posibilidad de crear páginas, eventos, promociones y juegos. También está orientada también a empresas. Recientemente ha incorporado una etiqueta de metadatos denominada <i>hashtag</i> .
Flickr	Creada en 2004. Permite, almacenar, ordenar, buscar e incluso poner a la venta fotografías y vídeos. Cuenta con dos versiones, la gratuita, con un límite de doscientas imágenes por cuenta y una determinada resolución y la versión de pago, que dispone de almacenamiento ilimitado.
Pinterest	Creada en 2009. Permite a los usuarios compartir imágenes y ordenarlas en <i>pinboards</i> o tableros temáticos. Los usuarios pueden seguir a otros usuarios y compartir imágenes en Pinterest o en otras redes sociales. También permite poner precio a los productos mediante una casilla específica.
Instagram	Creada en 2010 y actualmente comprada por Facebook. Es una aplicación que permite compartir fotos con los usuarios. Cuenta con la funcionalidad de añadir efectos fotográficos como filtros, marcos, colores, etc También permite compartir las fotos en otras redes sociales como por ejemplo en Facebook o Twitter.

Tabla 1: Redes sociales más populares en la actualidad.

Desde hace poco tiempo atrás hemos observado que muchas redes sociales están realizando cambios significativos en sus interfaces gráficas de usuario. Cada día son más similares lo que a la larga provocará una mejor experiencia de usuario desde nuestro punto de vista puesto que un usuario de una red social le será mucho más fácil e intuitivo comenzar a usa otra diferente si ésta tiene el mismo aspecto y probablemente reiterará su uso así como difusión, con lo cual resultará beneficioso para la compañía.

2.2. Agregadores sociales

Más de dos tercios de la población con acceso a Internet son usuarios de redes sociales. La popularización de éstas ha provocado que una persona puedan llegar a tener incluso más de cinco perfiles en las diferentes comunidades virtuales existentes. Esto conlleva la necesidad, tanto para los usuarios como para los desarrolladores, de crear nuevas aplicaciones que faciliten y centralicen el acceso a las diferentes cuentas. Son los denominados agregadores sociales.

Un agregador social no es más que una aplicación web, móvil o de escritorio que permite al usuario manejar sus diferentes perfiles e incluso interaccionar directamente en la red social deseada. Existen varios tipos de agregadores y pueden ser tanto aplicaciones web como software instalable.

Algunos de los más importantes son:

<i>Agregador</i>	<i>Características</i>
Digg	Agregador de noticias, imágenes y videos más popular
Menéame	Agregador de enlace de noticias
DivoBlogger	Es un agregador social de enlaces blogs & webs
Hootsuite	Es un agregador de redes sociales
Colashme	Agregador social de imágenes para promocionarl el turismo rural
Draw.io	Agregador de servicios en la nube

Tabla 2: Agregadores sociales más populares en la actualidad.

En este proyecto crearemos un agregador de redes sociales con la finalidad de extraer las imágenes y fotografías que los usuarios compartan en sus perfiles. El funcionamiento básico como podemos observar en la ilustración 1 se realiza mediante conexiones HTTP a dos bandas. Por un lado está la conexión entre el usuario con el

agregador y por otro lado la del agregador con la red social. La relación entre el agregador y la red social viene determinada por su API(Interfaz de programación de aplicaciones) que es la encargada de proporcionarnos información solicitada.

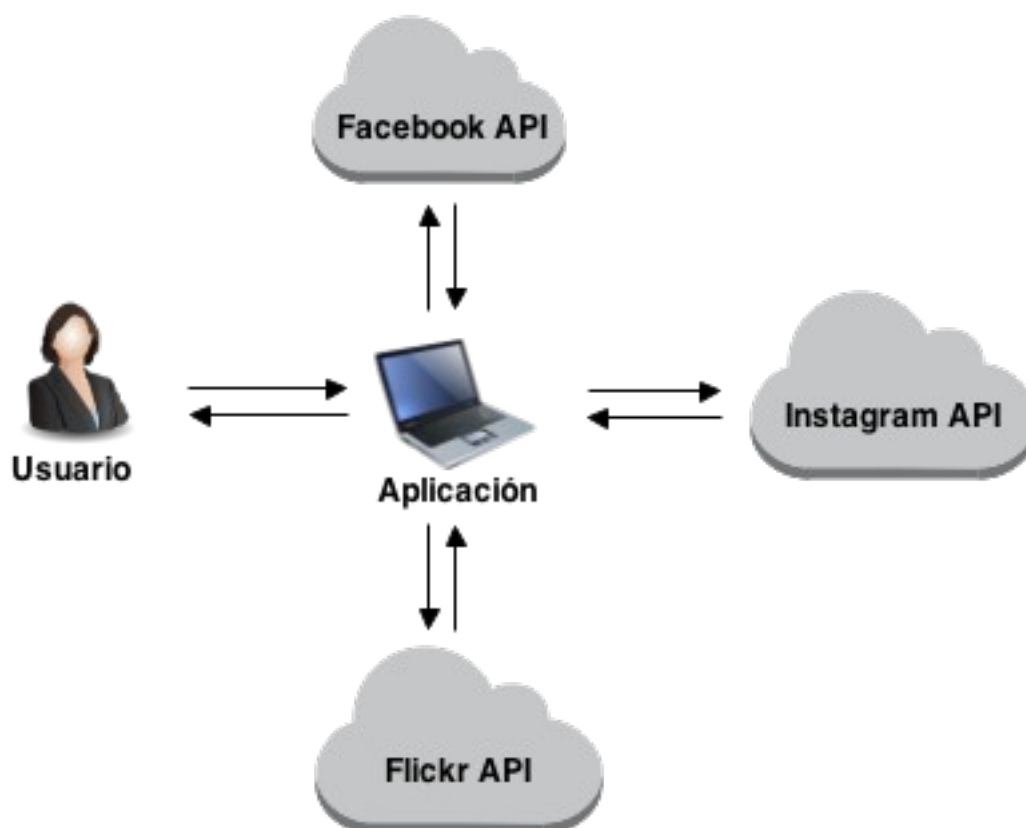


Ilustración 1: Esquema general de conexión de varias APIs.

2.3. Aplicaciones web en HTML5

Actualmente HTML5 junto a Javascript(JQuery) y CSS3 conforman una potente herramienta de programación de interfaces gráficas de usuario para aplicaciones web puesto que tiene el potencial de cambiar la web tal y como la conocemos.

El proyecto **HTML5 Boilerplate (HBP5)** ofrece una gran variedad de posibilidades para desarrollar una aplicación web en la parte del front-end. Es el template más famoso, robusto, rápido y adaptable para construir interfaces gráficas de usuario para sitios web.

Ofrece un estilo uniforme para todos los navegadores incorporando el uso de *media-queries*. Una *media-querie* es es un módulo CSS3 que permite la representación de contenido para adaptarse a condiciones como la resolución de pantalla.

Contiene los archivos *robots.txt* usado por el 80% de sitios web y el archivo *humans.txt*. Este último no es tan conocido, está enfocado a personas reales y contiene información sobre los autores del sitio web y las tecnologías usadas. Ambos están hechos para que los programas de búsqueda los encuentren.

Incluye la librería *Modernizr* que permite detectar que tipo de navegador está usando el usuario visitante de nuestra web y usa el archivo *.htaccess* que ofrece una configuración básica para que el servidor acepte nuevos tipos de archivo de vídeo y sonido así como archivos de fuente de texto.

También ofrece configuraciones de seguridad y compresión de archivos y cuenta con el fichero *404.html* que se incluye para que el desarrollador le de su propio diseño, aunque no se suele recomendar su uso.

3. Análisis

Como hemos comentado anteriormente en el capítulo introductorio, este proyecto tiene lo podemos dividir en dos grandes objetivos principales. El primero es crear un agregador social de fotografías e imágenes y el segundo es, que este agregador constituya una aplicación web responsive, es decir, adaptable, que permita a cualquier usuario utilizarla sin importar el lugar, dispositivo, navegador o sistema operativo desde el que acceda.

Cada objetivo equivaldrá a dos partes en el desarrollo del proyecto. Por un lado tendremos la implementación del funcionamiento en el lado del servidor de la aplicación web o denominado back-end y por otro lado la GUI (Interfaz Gráfica de Usuario) que será la parte visible que se muestre en el navegador o también llamada front-end.

Para implementar este agregador multimedia necesitaremos acceso a las redes sociales con tal de recuperar la información, en nuestro caso las imágenes. Puesto que directamente no podemos acceder al contenido que una red social contiene vamos a necesitar usar las herramientas que las propias comunidades virtuales proporcionan a terceros con el fin de que estos puedan desarrollar aplicaciones utilizando sus datos, cosa que obviamente beneficia a estas empresas. Estas herramientas se denominan APIs (Application Programming Interface).

3.1. API - Application Programming Interface

Una **API o Interfaz de Programación de Aplicaciones** en informática es un conjunto de funciones y procedimientos que ofrece una biblioteca para poder ser utilizado por otro software como una capa de abstracción. Esta capa enmascara la complejidad de acceso a una aplicación y a esta le permite proporcionar tan solo los datos que quiere hacer públicos.



Ilustración 2: Acceso de una App web a un perfil de usuario mediante una API.

La ilustración 2, define como mi aplicación web accederá por medio de la API, siempre y cuando la persona autorice su acceso, a las imágenes y otros posibles datos que el usuario haya almacenado en su perfil.

Cada día es más común que las nuevas aplicaciones webs o nativas usen las APIs para recuperar e interaccionar con los datos de un usuario. Estas a su vez están imponiendo el uso del protocolo OAuth 2.0 para conceder y garantizar un acceso más seguro. Este protocolo de conexión se está convirtiendo en un método estándar de seguridad y permite que las aplicaciones puedan usar las APIs de forma más fiable y facilitan el trabajo al desarrollador.

El protocolo de autorización que se utiliza es **protocolo OAuth** que proviene del término en inglés *Open Authorization*. Permite a un usuario propietario de ciertos recursos a autorizar a un tercero a que acceda a dichos recursos pero sin proporcionar en ningún momento a este tercero sus credenciales de autenticación. El protocolo tiene tres roles, el primero es el cliente que quiere acceder a un servicio en nombre de un usuario, en nuestro caso el cliente será nuestra aplicación web . El segundo es el proveedor de servicios, que como bien significa provee el servicio al cual quiere acceder el cliente, es decir, da acceso a los servicios de red social que utilicemos, y el tercer rol es el autenticador que se encarga de reconocer a los usuarios.

OAuth lo que define es como se comunican entre sí estos tres roles para conseguir que el cliente pueda acceder al servicio. Lo único que no define el protocolo es como se autentica el usuario, pero esto correrá por parte de la red social que implementemos, así que no nos debemos de preocupar de esta parte.

El objetivo final de este protocolo es que el cliente obtenga lo que se denomina un **Acces Token**. Enviando este token nuestra aplicación puede acceder a un recurso protegido en nombre del usuario que haya autenticado el autenticador. Alguno hablan de la “danza de los tokens” porque cada vez que el cliente hace una petición a la red social, se van intercambiando entre ellas alternadamente estos tokens o palabras de paso.

A continuación explicaré qué diferencias existe entre una web app, o aplicación web y una nativa y el porque de nuestra elección. Considero necesario este análisis previo puesto que es determinante para escoger el tipo de aplicación a desarrollar. Una vez tomada la decisión final, detallaré los casos de uso que darán una idea global de las funcionalidades de la aplicación y terminaré exponiendo que herramientas, lenguajes y librerías hemos utilizado para llevarla a cabo.

3.2. Aplicaciones web vs aplicaciones nativas.

Como en el párrafo anterior hemos dicho, existen las aplicaciones web y las aplicaciones nativas o frecuentemente llamadas aplicaciones móviles. Explicaré brevemente qué es cada una y cuales son las ventajas e inconvenientes que conlleva su implementación y desarrollo.

Aplicación web

Una aplicación web (o app web) no es más que una versión de la web optimizada para su perfecta visualización en dispositivos móviles gracias a HTML5, Javascript y CSS3 . Codificada en un lenguaje de programación para el desarrollo de aplicaciones web en el servidor.

<i>Ventajas</i>	<i>Inconvenientes</i>
✓ <i>Diseño responsive</i> ✓ <i>Hardware básico</i> ✓ <i>Pocos requisitos</i> ✓ <i>Libre de controles</i> ✓ <i>Sin actualizaciones</i> ✓ <i>Más económicas</i>	✗ <i>Menor promoción</i> ✗ <i>Conexión a Internet constante</i> ✗ <i>Menos eficiente</i> ✗ <i>Menos segura</i>

Tabla 3: Ventajas e inconvenientes de una aplicación web.

Analizando la tabla 3 podemos decir que las aplicaciones web tienen las ventajas de ser visualizadas en cualquier dispositivo, sin importar el sistema operativo que tengan. Tan solo será necesario un navegador e Internet. Por otro lado la necesidad de conexión constante a Internet se presenta como un inconveniente, lo que conlleva a que sean aplicaciones menos seguras ya que están expuestas a código malicioso que navegue por la red. Aunque HTML5 está desarrollando técnicas para crear apps offline, que no requieren conexión, todavía en la mayoría de los casos sigue siendo un inconveniente.

Dependiendo del objetivo de la aplicación, al ser desarrollada como web puede que conlleve una menor eficiencia para el usuario pero liberan a éste de tediosas e incómodas actualizaciones y aventajan al desarrollador en tiempo y en recursos económicos estando exentas de procesos de control al no alojarse en ningún market, lo cual provoca en su contra que tenga una menor promoción.

Aplicación nativa

Una aplicación nativa es aquella que está íntegramente programada en el entorno de desarrollo específico de cada sistema operativo. Es una aplicación software desarrollada para smartphones o tabletas y diseñada para explotar al máximo las características del dispositivo móvil. Se implementan en lenguaje nativo del propio terminal como por ejemplo Java para dispositivos Android, C# para Windows Phone u Objective-C para iOS.

<i>Ventajas</i>	<i>Inconvenientes</i>
✓ <i>Facil promoción</i> ✓ <i>Offline</i> ✓ <i>Uso de Widgets</i> ✓ <i>Uso de recursos</i>	✗ <i>Actualizaciones</i> ✗ <i>Controles</i> ✗ <i>Más gasto económico</i> ✗ <i>Implementación del lenguaje</i> ✗ <i>Más segura</i>

Tabla 4: Ventajas e inconvenientes de una aplicación nativa.

Una de las principales ventajas de este tipo de aplicaciones es que pueden ser offline, no requieren conexión a Internet constante aunque la mayoría de las ellas la utilice. Al trabajar sobre el lenguaje que soporta el dispositivo permite el uso de todos sus recursos y la eficiencia aumenta. Actualmente con HTML5 el uso de la geolocalización por ejemplo ya está disponible pero otras funcionalidades que pueden ser muy interesantes todavía no.

Las apps nativas se benefician de los markets (o mercados de aplicaciones) como canales de distribución y publicidad pero esto tiene un precio. Si la aplicación es para iOS, el desarrollador necesitará pagar una licencia anual de unos 75€ aproximadamente y un ordenador Mac (actualizado a la última versión de MacOS). Si la aplicación es para Android se deberá pagar una única cuota de 25€ además de tener que pasar en ambos casos controles de calidad siendo los de iOS mucho más rigurosos.

Claramente una aplicación nativa es mucho más cara además que necesita un equipo, por pequeño que sea, de desarrolladores formados y experimentados en los diferentes lenguajes requeridos.

Un ejemplo de aplicación web y nativa es Pixlr-o-matic, que sirve para aplicar filtros, marcos y luces a nuestras fotografías. Existe la versión online desarrollada en HTML5(<https://pixlr.com/o-matic/>) y la versión Android que podemos descargar e instalar desde el market (<https://play.google.com/store/apps/details?id=pixlr.OMatic&hl=es>).

Una vez entendidas las diferencias entra ambas necesitamos pensar que objetivos y requisitos tendrá nuestra aplicación para poder escoger el tipo que implementaremos. Pues bien, escogimos la aplicación web por las ventajas que anteriormente hemos comentado como pueden ser el menor tiempo de desarrollo, porque resulta más barata y porque para nuestro objetivo explicado en el capítulo de introducción es más que suficiente para desarrollar este proyecto.

3.3. Análisis de requisitos

Anteriormente habíamos separado la aplicación en dos partes, en el back-end o lado del servidor y en el front-end o lado del cliente. En este apartado explicaremos que requisitos vamos a necesitar para el correcto desarrollo de la aplicación.

Requisitos del back-end

En la parte del back-end necesitaremos un servidor para alojar la aplicación el cual podemos contratar en cualquier hosting de Internet y una base de datos para guardar información que consideremos oportuna.

Generalmente los hostings vienen con una aplicativo web que permite la subida de archivos, pero es preferible usar FTP mediante un programa adicional para ello, como pueda ser Filezilla, siempre que el servicio contratado del hosting permita acceso vía FTP.

El patrón básico de arquitectura sobre la que está construida esta aplicación es el conocido MVC (Modelo-Vista-Controlador).

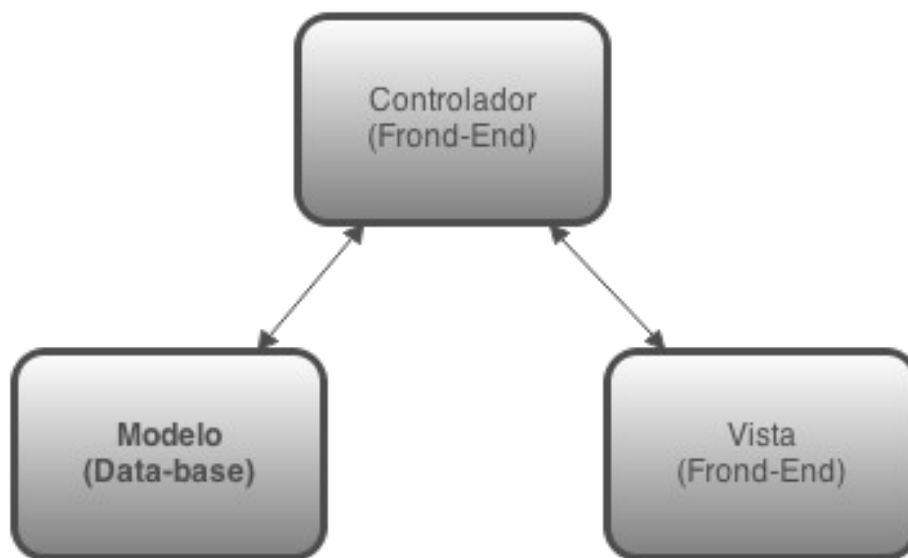


Ilustración 3: Arquitectura Modelo-Vista-Controlador.

El Modelo se encarga de mantener la persistencia de datos mediante la base de datos. La Vista representa a la interfaz gráfica de usuario, el front-end de la aplicación, mientras que el Controlador corresponderá al back-end, el encargado de manejar la interacción del usuario haciendo de intermediario entre el Modelo y la Vista. Además será quien gestione las llamadas a las APIs de las distintas redes sociales que utiliza esta aplicación.

Requisitos del front-end

La interfaz debe ser sencilla y poco recargada, con colores agradables a la vista. La tipología usada debe ser legible y de un tamaño adecuado, es decir, ni muy grande ni muy pequeña. La regla general que se debe aplicar a una GUI es la intuitividad. Ésta es la característica hará que el usuario que interaccione con la interfaz de la aplicación perciba rápidamente la idea del funcionamiento de uso.

Todo ello se une a la necesidad de crear un **diseño web responsive** o adaptable, lo que significa que la aplicación se tendrá que modelar y adaptar al dispositivo en el que sea visualizada sin importar el tamaño que tenga o las condiciones impuestas por el navegador

utilizado. Esto se consigue mediante el uso de estructuras e imágenes fluidas así como *media-queries* en la hoja de estilos CSS que consiguen adaptar el sitio web al entorno del usuario.

El usuario tan solo necesitará contar con una conexión a Internet y un dispositivo que contenga un navegador capaz de soportar HTML5.

Requisitos de estructura

La estructura que reciben casi todas las aplicaciones web es la basada en tres capas. La primera capa sería la formada por el cliente o navegador. La segunda capa representa el motor que ha de ser capaz de manejar PHP, y la tercera y última capa la constituye la base de datos.

3.4. Casos de uso

La ilustración 4 muestra las posibles acciones que un usuario podrá ejecutar dentro de nuestra aplicación.

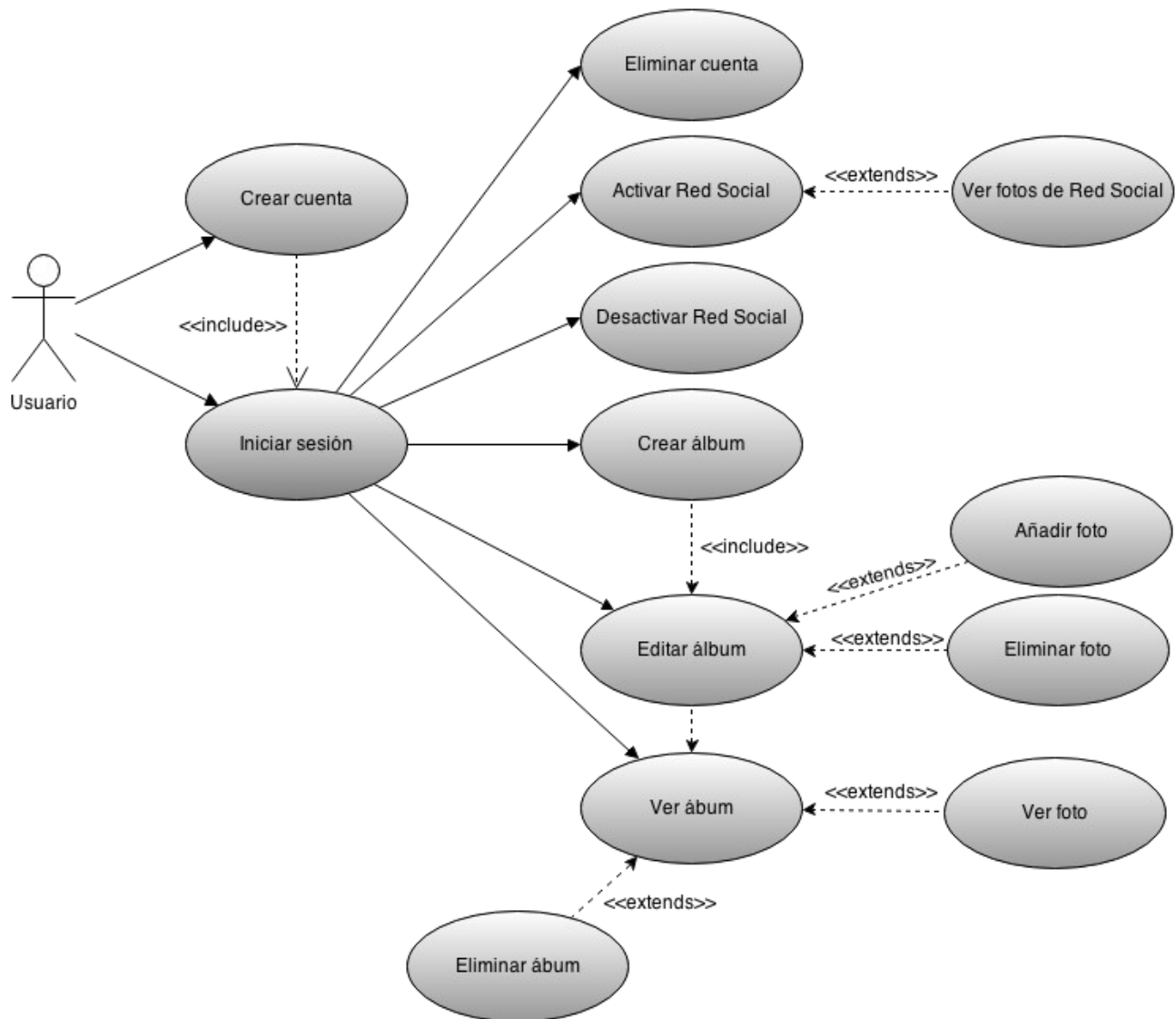


Ilustración 4: Casos de uso de la aplicación web

Será requisito imprescindible para el usuario crearse una cuenta dentro de nuestro aplicativo para posteriormente poder iniciar sesión en él. Una vez registrado sus datos y con la sesión de usuario activa este podrá activar o desactivar tantas redes sociales como la aplicación proporcione. La acción de activar una red social le permitirá ver las fotos que contiene su perfil dentro de esta red.

Independientemente el usuario podrá crear, ver, editar y eliminar álbums de fotos que a su vez contendrán las fotos que el usuario desee incluir desde sus redes sociales. Estas imágenes podrán ser añadidas, eliminadas o simplemente vistas.

3.5. Lenguajes, librerías y herramientas

Una vez analizado los requerimientos en el apartado anterior, vamos a hablar del lenguaje y librerías utilizadas así como las herramientas que son necesarias para desarrollar el proyecto.

Empezaremos hablando de las utilizadas en el back-end (o lado del servidor), como el PHP y MYSQL, luego pasaremos a las empleadas para desarrollar el front-end (o lado del cliente) y finalmente comentaré que software hemos necesitado y que características de hosting hemos requerido para alojar la aplicación.

PHP - Hypertext Pre-processor

PHP es un lenguaje de programación de uso general de código embebido en páginas HTML y ejecutado en el lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico el cual está caracterizado por su soporte para gran cantidad de bases de datos, como por ejemplo en nuestro caso MySQL.

La razón de haber usado este lenguaje es porque es un producto de código abierto, el cual tiene detrás un gran equipo de programación que constantemente está creando mejoras y extensiones del lenguaje. Es fácil de aprender y muy semejante a Java, un lenguaje que ya conocíamos anteriormente.

MySQL – Structured Query Language

MySQL es un gestor de base de datos operacional, de programación y gestión de datos del tipo relacional. Una base de datos es una colección estructurada de tablas que contienen información. Para acceder a ellas y procesar los datos así como manipularlos se usa este gestor. Permite el uso de bases de datos multiusuario a través de la web y en diferentes lenguajes de programación que se adaptan a diversas necesidades y requerimientos y se caracteriza por su rapidez en la búsqueda de datos.

Este es el gestor que utilizaremos para realizar consultas a nuestra base de datos que además es soportado por el lenguaje PHP el que utilizaremos en el lado del servidor. La combinación de ambos nos permitirá utilizar la información almacenada en la base de datos.

HTML5, Javascript y CSS3

Desde la parte cliente utilizaremos HTML5 , que es el la nueva versión 5 de HTML. Es un lenguaje marcas que se usa para estructurar y presentar el contenido para la web. Esta versión incluye nuevas mejoras y funcionalidades propias y otras a través de APIs de Javascript o CSS3.

Podríamos decir que en conjunto forman una familia de tecnologías que combinadas proporcionan una serie de posibilidades para construir aplicaciones y sitios web.

HTML5 ofrece muchas nuevas posibilidades, algunas de ellas son:

- Reproducción de audio sin necesidad de utilizar plugins.
- Nuevas etiquetas para contenido específico como `<section>` o `<nav>`.
- Bases de datos locales
- Geolocalización.
- Nuevas APIs para interfaz de usuario como “drag and drop” (arrastrar y soltar)

JavaScript

Es un lenguaje de programación interpretado del lado del cliente, es en decir que se ejecuta en el navegador web. Permite crear efectos especiales y definir interacciones con el usuario. EL navegador es el encargado de interpretar y ejecutar los mandatos de JavaScript.

Actualmente gracias a las APIs Javascript de HTML5 podemos acceder a todo tipo de recursos adicionales, como la cámara o el espacio para almacenamiento de datos.

JQuery

JQuery es la librería más conocida de Javascript, es software libre y código abierto. Esta librería se ha convertido en un complemento en la mayoría de sitios webs debido a su facilidad de uso y su potencial.

Usa plugins para programar nuevas funcionalidades como la validación de formularios, galerías de imágenes o diversos efectos y es capaz de ejecutarse libre de errores sin importar el navegador.

CSS3

CSS3 es la versión 3 de CSS. Es un conjunto de hojas de estilos que se utilizan para determinar el aspecto y el formato visual de un documento escrito en HTML.

Esta nueva versión incorpora nuevas e interesantes novedades que permiten dinamizar más nuestras webs, separar mejor los contenidos, optimizar el ancho de banda y mejorar la accesibilidad del documento eliminando las tediosas tablas para el control del diseño que anteriormente se utilizaba en el desarrollo front-end de interfaces de usuario.

Cuando hablamos de HTML5 realmente hablamos de estas 3 tecnologías conjuntamente. Su potencia, evolución y mejora hacen que sean las candidatas perfectas para desarrollar esta aplicación web. Necesitaremos varias cosas que cada una de ellas nos proporciona como los nuevos *tags* o etiquetas de HTML5, los efectos visuales proporcionados por JQuery o las hojas de estilo modernas de CSS3.

Initializr

Es un generador de plantillas responsive basado en HTML5 Boilerplate, explicado en el apartado 2.3,. La idea de Initializr es simplificar Boilerplate que permite que el desarrollador pueda escoger de forma precisa los archivos que desea facilitando el inicio del desarrollo de la aplicación.

Una de las diferencias es que Boilerplate genera un plantilla de inicio vacía mientras que Initializr genera una plantilla más completa con los bloques básicos como *header*, *footer*, una barra de navegación y otra lateral. El aspecto inicial es el que se muestra en la ilustración 5 y es la versión que decidimos usar porque se adaptaba más a las necesidades del proyecto además de que era la opción más completa para empezar desde el proyecto desde cero.

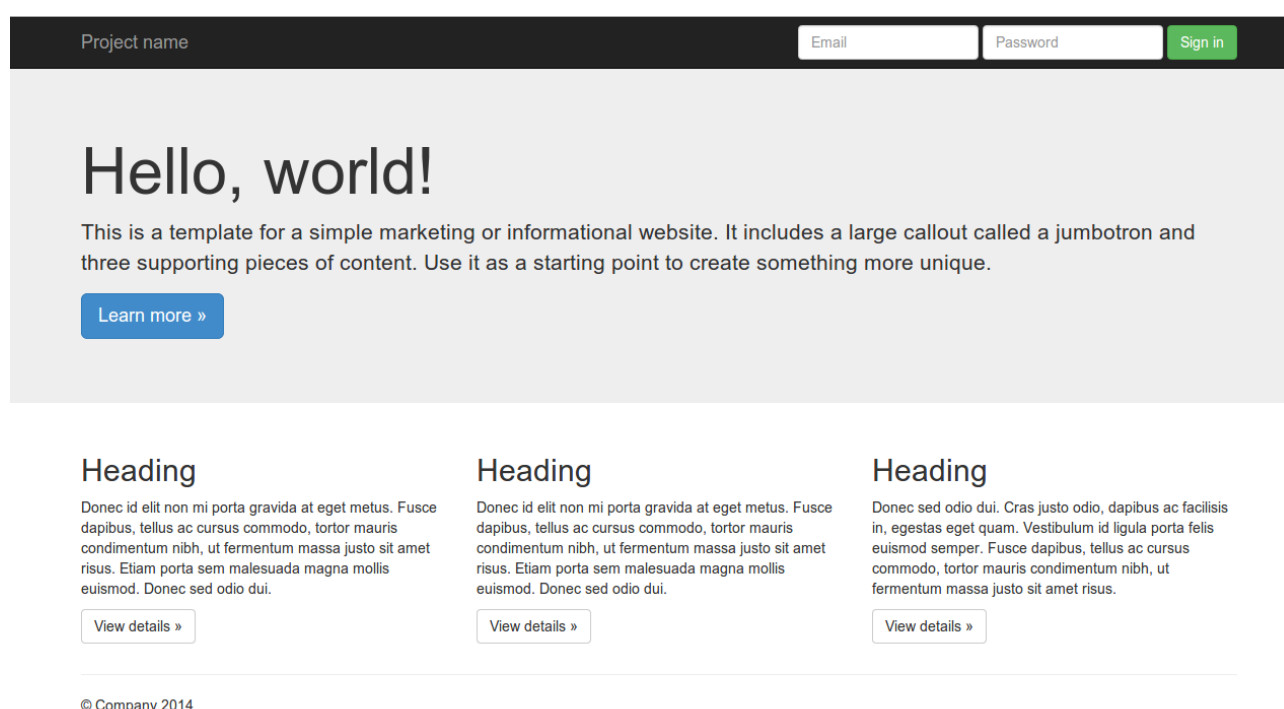


Ilustración 5: Página inicial de Initializr Bootstrap (index.php).

Editor de código SublimeText

Es un editor de texto y de código fuente gratis y desarrollado en C++ y Python. Se creó originalmente como una extensión de Vim y se caracteriza porque es muy ligero y a la vez completo. Es intuitivo y soporta el uso de *snippets*, *plugins* y sistemas de construcción de código. Algunas de sus ventajas son:

- Incluye un minimapa en el lateral derecho muy útil para desplazarse por el archivo.
- Posibilidad de varios *layouts* a la vez.
- Previsualización de archivos.
- Cuenta con multicursor, multipleselección, multicopy y multipaste.
- Cuenta con soporte nativo para muchos lenguajes entre ellos PHP y HTML.
- Su búsqueda dinámica es muy útil para su uso en proyectos o directorios.
- Tiene marcado de llaves, autocompletado y acceso directo a línea.
- *Snippets*, código predefinido con una combinación de teclas asignadas.

Es el editor que hemos decidido utilizar para desarrollar el código fuente de la aplicación por ser el más completo de todos por todas las características nombradas anteriormente.

Web Hosting

El alojamiento web o frecuentemente llamado web hosting es un servicio donde un proveedor alquila un servidor conectado a Internet en el cual puedes alojar todo tipo de ficheros que serán accesibles desde el propio Internet.

El uso mas típico de este servicio es alojar un sitio web, es decir, copiar tus archivos HTML junto con las librerías, *scripts* y hojas de estilo utilizados en el servidor contratado en Internet. Esto permitirá que la página web desarrollada sea accesible desde cualquier otro lado del mundo.

Hay hostings gratuitos que suelen incluir las prestaciones más importantes como el acceso vía FTP, correo electrónico y soporte para aplicaciones PHP pero todos ellos cuentan con una serie de limitaciones que hacen recurrir a cualquier web o aplicación web un poco desarrollada a los de pago.

En los hostings de pago podemos encontrar 1&1, hosting Linux, uno de los más conocidos en la actualidad siendo líder en el mercado español y es el que usaremos para este proyecto.

El pack que necesitamos para poder alojar la aplicación web y que funcione correctamente es el “1&1 pack inicial” con un importe aproximado de 26€ al año. Este pack contiene todas las funcionalidades que necesitamos. De entre todos los servicios que ofrece nombraremos solo los que vamos a utilizar.

- Principalmente el dominio está incluido, en nuestro caso es www.casairama.com.
- Cuenta con un espacio web de 1.000 MB suficiente para alojar esta primera versión de la aplicación.
- Soporte para PHP y MySQL.
- PhpMyAdmin para gestionar la base de datos

- 1 base de datos.
- Subida de ficheros por FTP.

Para la subida de archivos por FTP uso el cliente Filezilla. Es un cliente multiplataforma de código abierto y software libre que se usamos para transferir los archivos desde el ordenador personal al hosting y nos permite gestionar con rapidez y cómodamente los archivos en el lado del servidor.

4. Diseño e implementación

Como hemos explicado en reiteradas ocasiones, una aplicación web consta de dos partes fundamentales, el back-end y el front-end, siendo la primera el lado del servidor desarrollada por los programadores usando, en nuestro caso, el lenguaje PHP 5.5. La segunda en el lado del cliente suele ser llevada a cabo por los diseñadores gráficos que se encargan de maquetar la estructura semántica del contenido con HTML5, codificar el diseño en los CSS y agregar la interacción con el usuario por medio de Javascript y JQuery.

En este capítulo primero exponemos la arquitectura de la aplicación, la organización del sitio web, que representa como están almacenados los ficheros, y el diseño de la base de datos para su posterior uso. Una vez planteada la aplicación detallamos las funcionalidades más relevantes de la misma que pertenecen al lado del back-end. Para finalizar daremos detalles de las partes más relevantes que incumben al diseño del front-end.

4.1. Arquitectura de la aplicación

La aplicación está basada en la arquitectura cliente-servidor que es un modelo de aplicación distribuida, en la que el cliente hace una petición al servidor y éste envía uno o varios mensajes de respuesta.

El protocolo que usamos para realizar esta comunicación es el HTTP o protocolo de transferencia de hipertexto, tal como se muestra en la ilustración número 6.

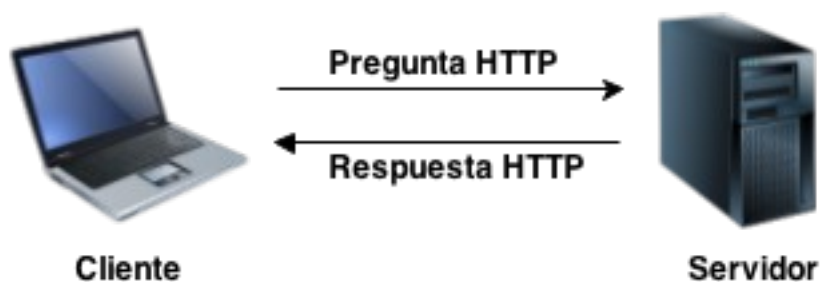


Ilustración 6: Arquitectura cliente-servidor.

El servidor que aloja nuestra aplicación es el hosting 1&1, como explicamos al final del capítulo 3. Este servidor es el encargado de manejar a los múltiples clientes que hacen peticiones constantemente y de realizar las consultas a la base de datos para mandarle de vuelta la información que solicite.

Arquitectura general

La arquitectura general de esta aplicación web se representada en la ilustración número 7 en la cual podemos ver que es multicliente y cumple con el análisis de requisitos expuesto en el apartado 3.2.

Observando la ilustración advertimos tres partes diferentes. Por un lado la tenemos la parte del cliente, por otro lado la del servidor con su respectiva base de datos y la parte restante es la correspondiente a las APIs de las redes sociales.

Cada usuario o cliente, por medio de la interfaz de usuario, hará peticiones del tipo HTTP al servidor. Éste las interpretará y accederá si es necesario a la base de datos o hará llamadas a las APIs para obtener la información que se precise. Una vez reciba los datos y realice los procesos necesarios enviará la respuesta HTTP de vuelta al cliente.

Las peticiones que se hacen a la base de datos son consultas realizadas en el lenguaje SQL mientras que las peticiones que se hacen a las APIs también pueden ser o bien peticiones HTTP o bien consultas, por ejemplo, en FQL (Facebook Query Language) un lenguaje proporcionado por la red social de Facebook prácticamente idéntico a SQL.

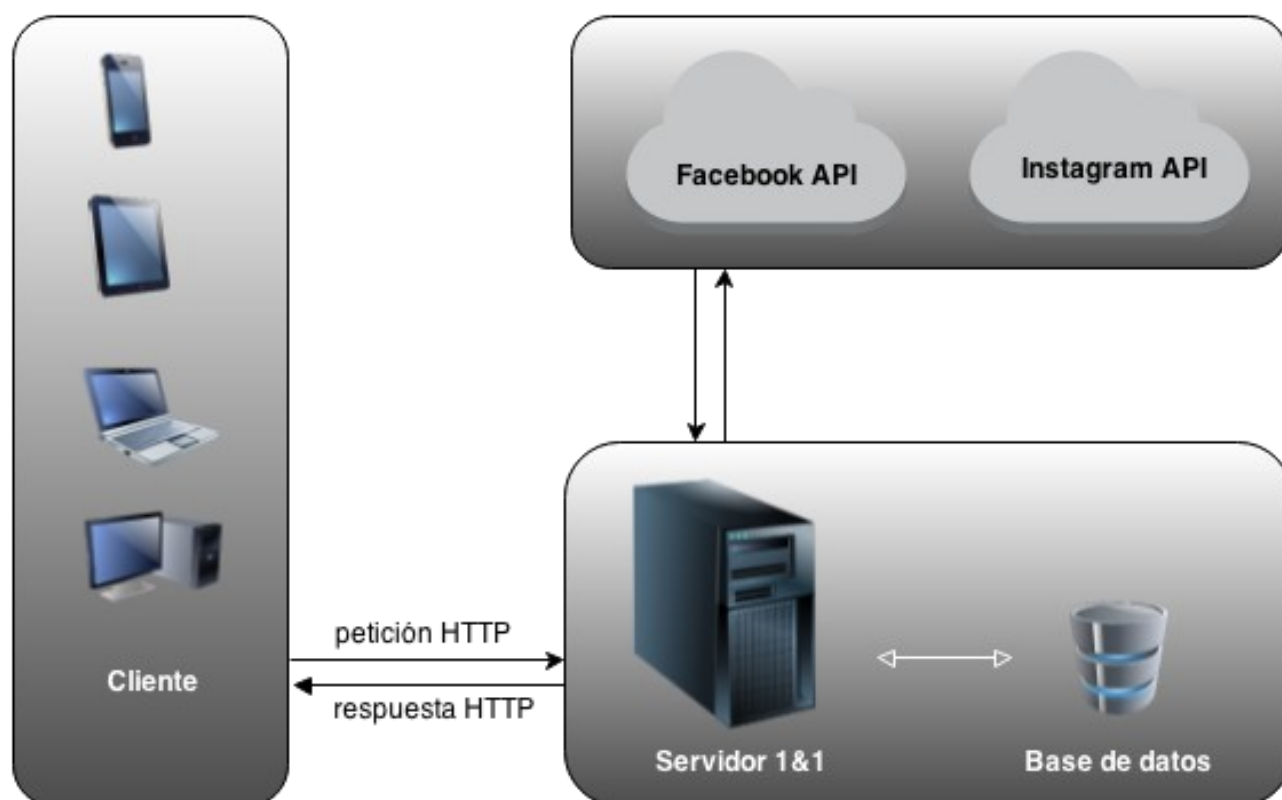


Ilustración 7: Arquitectura principal de la aplicación .

Pongamos un ejemplo de como sería una petición cliente servidor para ver las fotos de un perfil de Facebook.

Suponiendo que Alicia está logeada en Facebook y en nuestra aplicación, y que ha consentido el acceso a sus datos, se le procederá a mostrar un enlace o link para ver en nuestra web las fotos que tiene en su perfil de Facebook. La acción de darle a ese link

creará una petición HTTP al servidor que este interpretará y que reaccionará realizando otra petición HTTP a Facebook para comprobar que Alicia está conectada. Facebook devolverá su estado y en caso afirmativo se le enviará una consulta FQL que proporcionará a nuestro servidor los datos de todas las imágenes que Alicia tiene en su perfil de Facebook.

El servidor procesará entre todos los datos devueltos las URL de cada imagen recuperada de Facebook. Este conjunto o array de imágenes en forma de URL se enviará al lado del cliente que será éste el encargado de mostrar estas URL en forma de imagen para que Alicia, como usuario final pueda ver cómodamente sus fotos de Facebook en nuestra aplicación web tan solo con un click de ratón.

4.2. Mapa conceptual de Navegación

En este punto vamos a explicar el mapa de navegación para hacernos una idea global y poder orientarnos fácilmente dentro de la aplicación. Este mapa de navegación representa de forma esquemática la estructura de hipertexto mediante el uso de nodos y subnodos.

Es un tanto difícil de crear este tipo de mapa cuando no es una página web estática ya que, al tratarse de una aplicación web o web 2.0, contiene contenido dinámico y variable que no siempre se mostrará al usuario.

En el diagrama que se incluye a continuación hemos determinado aquellas páginas variables con un asterisco (*), es decir, aquellas páginas que solo se mostrarán si el usuario interacciona con la aplicación y decide crearlas o mostrarlas. Hemos valorado incluirlas todas en el diagrama puesto que al no ser muchas ayudan a la compresión del sitio web.

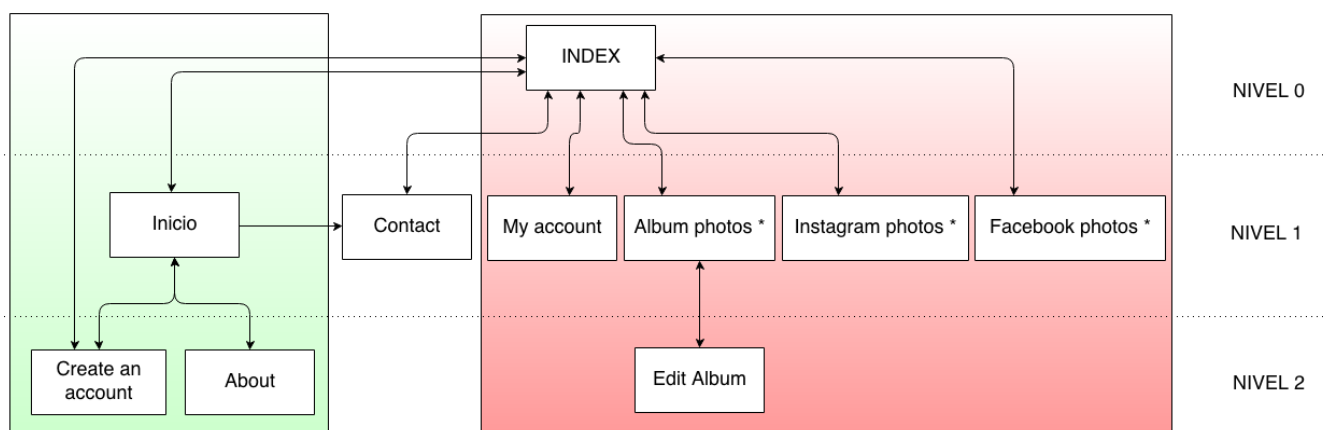


Ilustración 8: Mapa conceptual de navegación de .

En la imagen anterior, ilustración 8, se muestra el mapa de forma conceptual con una estructura jerárquica representando la principal información y navegación dentro de la aplicación.

Cada nodo representa por su título, en forma de palabra activa, una estancia de la página web e informa de la rama jerárquica o nivel en la que se encuentra la página. Aquellos nodos que van acompañados del símbolo asterisco (*), como antes hemos

explicado, constituyen los sitios dinámicos que serán creados o destruidos por el usuario. La relación entre los nodos y subnodos se representa mediante flechas direccionales o bidireccionales. También cabe resaltar que el usuario podrá desplazarse horizontalmente entre los diferentes niveles.

La aplicación tendrá dos tipos de usuario. Los logeados, que previamente se habrán registrado en el sistema y los no registrados o anónimos. Dependiendo del tipo de usuario que acceda a la página dividimos en tres zonas el mapa de navegación. La marcada en verde es a la única que permite acceso al usuario anónimo. La de color rojo solo será accesible para los usuarios logeados y finalmente la zona que no está señalizada, la zona del medio, será accesible pos ambos usuarios.

Comenzamos en el nivel 0, que es el nivel donde se encuentra la página madre o principal de la aplicación, el *INDEX*. Partiendo desde aquí y usando el control de sesiones se envía al usuario anónimo a la página de *Inicio* si este no está logeado o se quedará en la misma en caso contrario. Si la aplicación nos redirige a la página de *Inicio* tan tan solo podremos acceder horizontalmente a las páginas *Create an account* o *About*.

Una vez logeado y posicionado en *Index*, este usuario podrá acceder a las páginas de del nivel 1. Estas representan las dos barras de navegación que contiene la aplicación web. La primera de ellas formada por las páginas *Contact* y *My Account* y segunda por *Album photos**, *Instagam Photos**, *Facebook Photos**. Estas últimas que llevan asterisco son páginas dinámicas, es decir, éstas serán mostradas como links dinámicos en la barra de navegación a medida que el usuario active o desactive redes sociales o cree o elimine nuevos álbums de fotos. Entre ellas se permite navegar indistintamente ya que ambas barras constituyen los menús principales de la aplicación.

En el nivel 3 solo tenemos la página *Edit Album* que es la única que tiene un paso dirigido, es decir, requiere pasar a través de *Album Photos** y no permite el acceso al nivel uno.

Para más información de como está estructurado el proyecto podemos consultar el Anexo I, el cual muestra la organización de carpetas y ficheros, y detalla brevemente los más relevantes para la comprensión del desarrollo de este proyecto.

4.3. Base de datos

La base de datos es la encargada de almacenar en el servidor la información para cada usuario de nuestra aplicación.

La información principal y requerida que siempre se tendrá que almacenar será la introducida por un usuario cuando este se registre en nuestra aplicación. Los datos que guardaremos son el nombre y apellidos de usuario, el email y un *hash*, en caso de que no almacenemos el password.

En la ilustración 8 que viene a continuación representamos el diagrama entidad-relación de la base de datos que muestra de qué clases y atributos consta nuestra aplicación así como la relación que mantiene cada entidad.

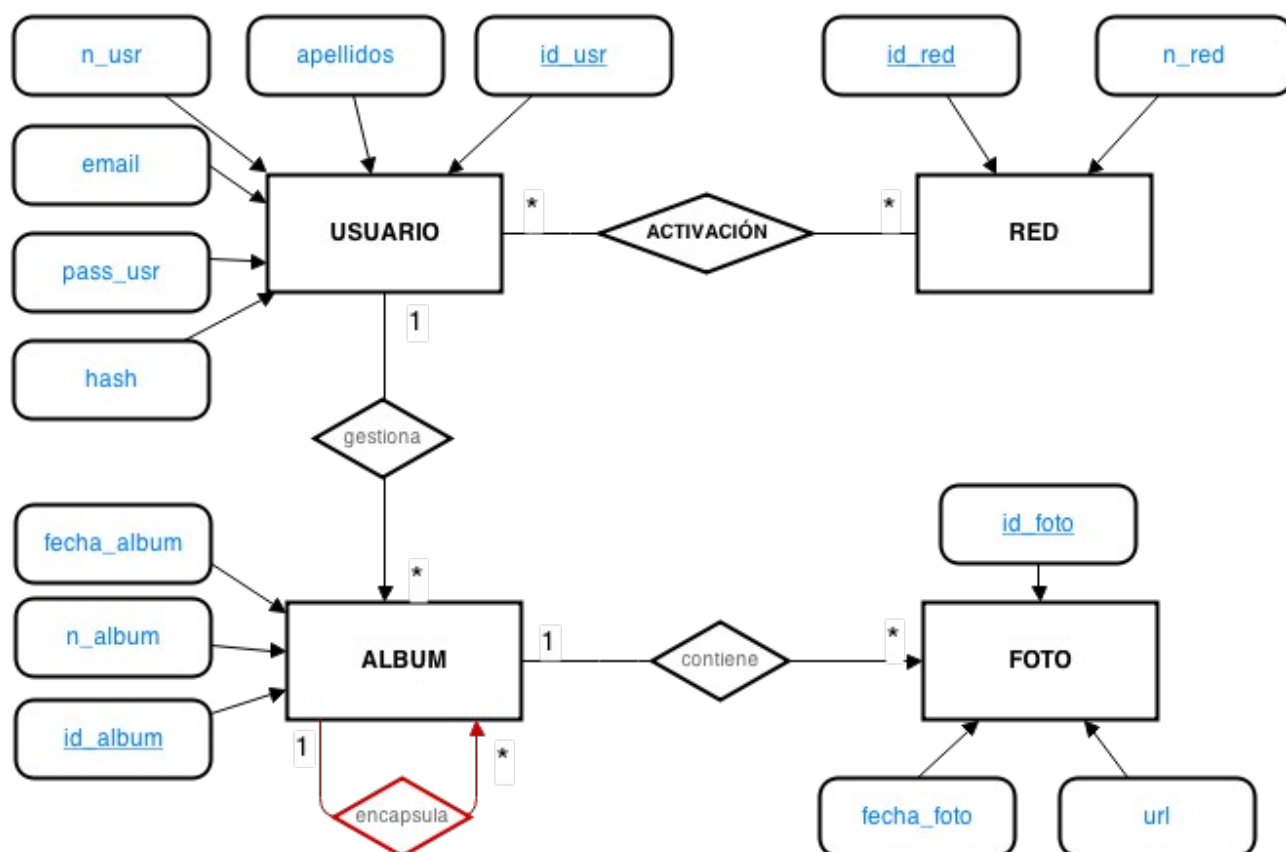


Ilustración 9: Modelo base de datos entidad-relación de de .

Este diagrama entidad-relación se compone de un usuario, que podrá activar tantas redes sociales como se le ofrezcan y estas a su vez contendrán muchos usuarios que las activen. Además este usuario podrá gestionar, es decir, crear, editar, ver y eliminar de 1 a N álbums los cuales almacenarán un conjunto de fotos. La autorrelación que mostramos en rojo identifica que un álbum podrá contener varios sub-álbums en su interior. Lo hemos remarcado en color rojo puesto que finalmente no se ha implementado esta funcionalidad.

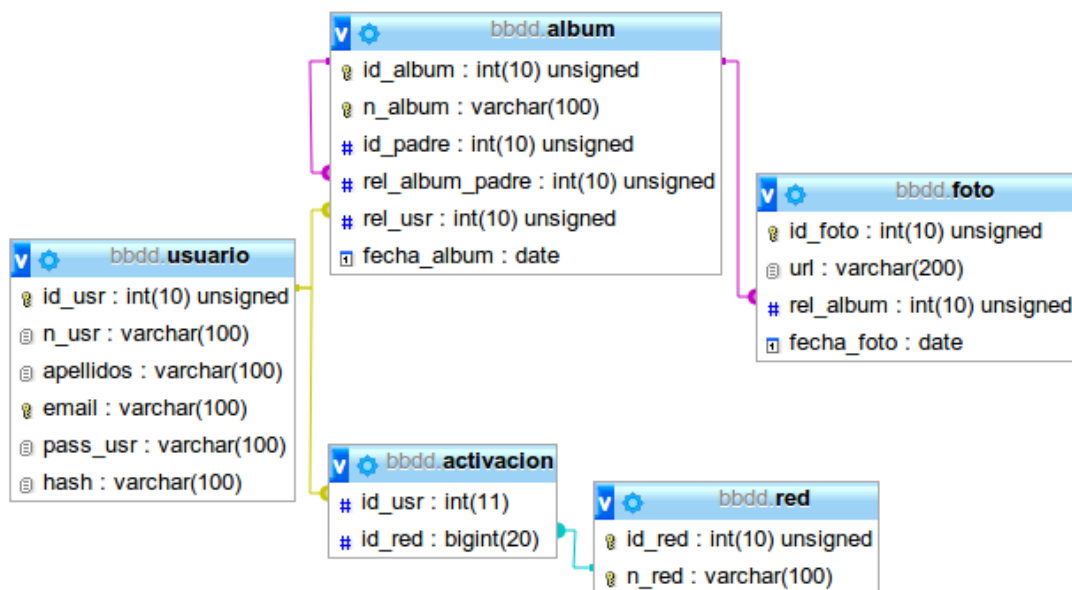


Ilustración 10: Modelo de la base de datos relacional de

La ilustración 10 muestra el modelo de base de datos relacional como resultado del diseño entidad-relación anteriormente explicado de la base de datos de la aplicación.

Conexión a la base de datos

Para la conexión a la base de datos se necesitaremos una serie de parámetros de información que nos proporciona el host 1&1 y que configuraremos en el fichero *config.php*, el cual podemos determinar su posición en el Anexo I del presente documento. Los datos que incluiremos en este fichero son:

- *host*: El nombre del host que aloja la base de datos.
- *bbdd*: El identificador de la base de datos.
- *user*: El nombre de usuario de la base de datos.
- *password*: El password de la base de datos.

Ahora ya podemos crear la conexión por medio de una consulta usando la función *mysql_connect(host, user, password)*. Una vez establecida la conexión elegimos la base de datos que vamos a usar mediante la función *mysql_select_db(bbdd)*.

Consultas a la base de datos

Para realizar consultas a la base de datos necesitaremos la conexión que anteriormente hemos configurado. Estas consultas las haremos usando la función `mysql_query(query_sql)`, que nos devolverá un *array* con los datos que hayamos solicitado en nuestra *query* o consulta.

Las consultas que usamos son del tipo :

- *SELECT* : sirve para recuperar información.
- *INSERT* : se usa para insertar una nueva entrada en una tabla.
- *UPDATE* : actualiza una serie de valores.
- *DELETE*: se usa para borrar entradas de tablas.

Al inicio del proyecto nuestro servidor estaba configurado con PHP 4.3 y actualmente lo hemos subido de versión, a PHP 5.5. Esto implica que ciertas funciones de PHP tales como `mysql_connect()` y `mysql_query()` queden obsoletas o desaprobadas y deban ser reemplazadas por otras nuevas. Sería necesario migrar la aplicación al completo usando las nuevas funciones puesto que es cuestión de tiempo que PHP 5.5 no de soporte a las viejas.

4.4. Back-end. Organización ficheros PHP:

Ahora que ya hemos visto en los apartados anteriores el mapa de navegación de la aplicación web y la base de datos que utiliza, podemos proceder a explicar el desarrollo interno o motor de la misma. En este subapartado exponemos aquellas funcionalidades más importantes empezando por el control de sesiones de usuario, siguiendo por el registro y encriptación de sus datos y finalizando por definir las APIS implementadas y su funcionamiento.

Registro/login de usuario usando algoritmo BYCRIP

El registro y el login se realizan por medio de formularios o también conocidos como *forms* que utilizan el protocolo HTTP como vía de comunicación. Estos *forms* implementan el **método POST**, proporcionado este por el protocolo para enviar la información del formulario y que la URL queda limpia.

En la etiqueta del formulario, `<form>`, indicamos el método a utilizar en el campo *method*, que como antes hemos dicho es el *POST*, el encargado de enviar las variables que declaramos a la página que se indica mediante el campo *action*.

Para el registro del usuario las variables a enviar son el nombre de usuario, los apellidos, una dirección de email y el password o el *hash*. Para el *login* tan solo nos hace falta el email y el password. Estos datos como antes hemos explicado, usando POST se envían cuando el usuario pulsa el botón del formulario del tipo *submit*. Además hemos incluido en estos *forms* un campo oculto que también se manda y, determina en el destino especificado en *action*, qué acción debemos manejar en el lado del servidor.

Como hemos dicho, la información pasa al lado del servidor y es el *script controlUsuario.php* el encargado de controlar la proveniencia de los datos y de ejecutar la opción que corresponda. Recordamos que se puede consultar su posicionamiento en el Anexo I. Para el caso de registrar a un usuario primero recuperamos los datos (nombre, apellidos, email y password o *hash*) y mediante una consulta del tipo *INSERT* incluimos la información en la base de datos. Previamente encriptamos el password.

Para el login del usuario utilizaremos el mismo *script* que maneja esta petición realizando una consulta del tipo *SELECT* con el nombre del usuario. Si éste existe, procede a recuperar el password encriptado o el *hash* que posteriormente tiene que ser validado y si todo es correcto el usuario se loga e inicia sesión en la aplicación.

Algoritmo de encriptación BCRIPT.

Al principio de desarrollar la aplicación se implementó la encriptación del password con el algoritmo SHA256 que es una función hash criptográfica. Una función hash es un algoritmo que transforma un conjunto arbitrario de datos en un único valor de longitud fija, el *hash*.

Uno de los problemas de los algoritmos criptográficos es que no dejan de ser una función matemática que puede ser rota por fuerza bruta y ha de renovarse cada cierto tiempo, así que decidimos investigar algo más seguro para encriptar el password de los usuarios y nos decantamos por el **algoritmo BCRIPT** puesto que PHP 5.5 lo soporta y es más robusto, seguro y lento y aunque para el humano sea imperceptible para un ataque que comprueba millones de combinaciones se convierte en un impedimento.

Para encriptar el password lo hacemos mediante la siguiente función:

```
$hash = password_hash($pass_usr, PASSWORD_BCRYPT, $options);
```

que genera una variable hash de la contraseña insertada por el usuario. A la función le enviamos la contraseña, la constante *PASSWORD_BCRYPT* que define el algoritmo a usar y una serie de opciones en las cuales podemos determinar un coste de computación y un *salt* o cadena extra que se añade al hash para hacerlo más complejo si cabe. Finalmente el resultado lo almacenamos en la base de datos.

Como podemos ver en la el modelo entidad-relación de la base de datos tenemos el atributo *password* y el atributo *hash*. En el *password* anteriormente lo usábamos para guardar la contraseña encriptada con SHA256 pero con este nuevo algoritmo es suficiente con el campo *hash*.

Para logear al usuario primero recuperamos la información de la base de datos y procedemos a verificar el si el password introducido corresponde al password encriptado. Para ello usamos la función *password_verify()* que nos devolverá verdadero si las contraseñas coinciden o falso en caso contrario.

Control de sesiones.

Anteriormente hemos nombrado conceptos como comprobar sesión o iniciar sesión. Esto significa comprobar si el usuario está logeado o no, para poder tener un control de a qué páginas concederle acceso. Para ello hacemos el uso de **variables de sesión** que PHP nos proporciona y que permiten hacer persistente la información de estado entre peticiones de páginas.

El flujo de trabajo es el siguiente, desde la página en la que el usuario se encuentre, se incluye el *script funciones.php* que contiene un método al cual se llama para comprobar si la sesión está iniciada o no usando las variables de sesión. Si la variable de sesión que contiene el identificador del usuario, en nuestro caso es el email, está vacía significa que no hay sesión, con lo cual el usuario deberá logarse para poder entrar en la aplicación.

Una sesión se puede crear o iniciar o eliminar. Para crear la sesión o reanudarla si ya existía anteriormente, usamos el método *session_start()*. En el caso de que nuestro usuario acceda al enlace proporcionado para deslogarse la aplicación llamará al script *logout.php* el cual borra la sesión de *cookies* y después borra y destruye la información de sesión mediante el uso de *session_unset()* y *session_destroy()*.

Recordamos que todos los *scripts* que nombramos a lo largo del documento los podemos consultar en el Anexo I.

Uso de APIs

Como bien hemos descrito antes las APIs son herramientas proporcionadas por las redes sociales para que nuestros sitios web accedan a la información que brindan. Para ello se usamos el protocolo OAuth 2.0 el cual hemos explicado en el apartado 3 del presente documento. En nuestro caso hemos escogido Facebook e Instagram por su popularidad y similitud entre sus APIs.

Para el uso de una API necesitamos primero registrarnos como *developers* o desarrolladores y crear una aplicación en las redes sociales a utilizar. Primeramente explicaremos los pasos y requisitos que se nos han presentado para obtener los datos de conexión con la API de Facebook y después cómo usamos estos datos para acceder a los perfiles de usuarios de dicha red, así como el tratamiento que nuestra aplicación hará con estos datos. Finalmente comentaremos las diferencias encontradas en la API de Instagram.

Registro de una aplicación en Facebook

Para usar la API de Facebook hemos tenido que acreditarnos como desarrolladores en el portal Facebook Developers (<https://developers.facebook.com/>) puesto que nos exigen que nos identifiquemos inequívocamente para controlar posibles acciones fraudulentas. Una vez acreditados por Facebook procedemos a registrar nuestra aplicación . Para ello necesitamos proporcionar el nombre de la aplicación, un correo

electrónico y la dirección o URL donde se aloja la aplicación web que será quien use los datos de Facebook. Con estos datos bastará para crear una conexión básica.

The image shows the Facebook App registration interface. It has three tabs: 'Basic', 'Advanced', and 'Migrations'. The 'Basic' tab is active. It contains the following fields:

- ID de la aplicación:** 205506669632195
- App Secret:** A masked field with a 'Mostrar' button.
- Display Name:**
- Namespace:** img_aplicacion
- App Domains:** www.casairama.com
- Correo Electrónico de Contacto:** irama.ag@gmail.com

Below these fields is a section titled 'Página web' with a close button (x). It contains:

- URL del sitio:** http://www.casairama.com/
- Mobile Site URL:** URL of your mobile site

Below the 'Página web' section is a button '+ Add Platform'. At the bottom of the form are three buttons: 'Eliminar aplicación' (red), 'Eliminar' (light blue), and 'Guardar cambios' (dark blue).

Ilustración 11: Datos de la aplicación registrada en Facebook.

La ilustración 11 nos proporciona el resultado final de registrar la aplicación. Como podemos ver nos crea el identificador de la aplicación (*App ID*) y número secreto (*App Secret*) el cual podemos restablecerlo tantas veces como queramos.

Debemos recalcar que Facebook es muy minucioso frente a la URL de nuestro sitio web. Inicialmente en este proyecto usó un hosting gratuito, <http://www.000webhost.com/> pero tuvimos que cambiar de host a un privado puesto que el gratuito Facebook lo detectaba como un sitio peligroso y no me concedía permiso para registrar la aplicación en Facebook.

Conexión con la Graph API de Facebook

Una vez registrada la aplicación y obtenido los parámetros *APP ID* y *APP Secret* ya podemos proseguir a conectar al usuario con la API mediante el protocolo **OAuth 2.0**.

Para ello tenemos que incluir la librería de Facebook en nuestro proyecto, tal como mostramos en el diagrama del Anexo I, para poder así hacer uso de las funciones que ésta nos proporciona. Una vez añadida la librería, creamos un fichero de configuración *fbconfig.php* el cual contendrá la configuración inicial de la conexión y requerirá el uso de la librería. Es en este fichero donde definimos los valores *APP ID* y *APP Secret* para enviarlos por parámetro al constructor del objeto de Facebook. El código quedaría de la siguiente forma:

```
//Definición de APP ID y APP Secret
$fb_apikey = '205506669632195';
$fb_secret = '764e39a2e9403a385c6761bc40b990f9';

//Creación del objeto facebook
$facebook = new Facebook(array(
    'appId' => $fb_apikey,
    'secret' => $fb_secret,
));
```

Una vez instanciado nuestro objeto de Facebook cuando queramos acceder a Facebook llamaremos a la función proporcionada por la librería de Facebook *getLoginUrl()*. A esta función le pasamos los parámetros una URL de redirección a la cual se devolverá el usuario, y el *scope*, o permisos que se le solicitarán. A continuación un ejemplo de llamada a esta función:

```
$loginURLFacebook = $facebook->getLoginURL(array(
    'scope'          => 'user_photos',
    'display'        => 'popup',
    'redirect_uri'   => 'http://www.casairama.com/inter/fbfotos.php'
));
```

Si no pidiéramos ningún permiso al usuario solo podríamos acceder a su perfil público. Facebook recomienda como una buena práctica pedir solo aquellos permisos que realmente nuestra aplicación web vaya a necesitar, puesto que el consumidor de la aplicación será consciente de esta petición y si es excesiva puede provocar que los deniegue y por consiguiente que deje de utilizar nuestra aplicación web.

Al llamar a esta función, Facebook nos devuelve una dirección que es a la debemos redirigir a nuestro usuario para proceder al proceso de autenticación y del cual obtendremos el *Acces Token*, el identificador que nos permitirá demostrar en cada petición que tenemos acceso a esos datos.

Una vez enviado al usuario a la URL se pueden dar dos casos. El primero que el usuario no haya autorizado a la aplicación al acceso de sus datos, con lo cual nos saltará la siguiente ventana:



Ilustración 12: Aplicación solicitando permisos al usuario.

La ilustración 12 pide los permisos por defecto que son perfil público y lista de amigos y los adicionales que es el acceso a las fotos. Este último lo hemos solicitado mediante el parámetro *scope* que antes hemos definido.

El segundo caso se da cuando el usuario no está logeado en Facebook. Al enviarlo a la dirección proporcionada por esta red social que hemos obtenido mediante el método *getLoginURL()* saltará la pantalla mostrada en la ilustración 13. Será entonces Facebook el encargado de manejar los datos que introduzca el usuario y de autenticarlo eximiendo de responsabilidad a nuestra aplicación.



Ilustración 13: Login en Facebook mediante el uso del protocolo OAuth.

En ambos casos, se reenviará al usuario a la página que habremos especificado mediante el parámetro *redirect_uri*.

Desconexión con la Graph API de Facebook

Para la desconexión de Facebook existe la función *getLogoutUrl()* pero nosotros en este caso no la usamos puesto que siempre hablamos de activar o desactivar la red social. Para ver las fotos es requisito imprescindible el estar logeado y que nuestra aplicación tenga los permisos del usuario para poder recuperar su datos. Por el contrario para no mostrar estas fotos no es necesario que el usuario se deslogue por completo usando esta función, puesto que esto produce que se cierre su sesión de Facebook en la sesión actual del navegador.

De todos modos tenemos un control y cada vez que el usuario decida desactivar Facebook, destruiremos la sesión desde nuestra aplicación, pero no deslogamos al usuario. Esto lo hacemos mediante el uso de la siguiente función:

```
$facebook->destroySession();
```

Para posibles accesos, volveremos a solicitar los datos a Facebook y si todo está en orden el usuario verá sus fotos con normalidad, en caso contrario se le pedirá por parte de Facebook que se autentique.

Uso de Graph API de Facebook

En los pasos anteriores hemos registrado la aplicación y hemos realizado la conexión inicial con la API. Ahora es el momento de realizar las llamadas al Graph API de Facebook.

La petición que hacemos desde nuestra aplicación es acceder y recuperar las URL de todas las imágenes que el usuario disponga en su perfil de Facebook. Para ello le pedimos a Facebook el objeto */me* de la siguiente forma:

```
$fbme = $facebook->api('/me', 'GET');
```

El objeto */me* es un lo que se suele llamar un *endpoint* o punto final. Esto significa que le pedimos acceso a la raíz del usuario, a todos sus datos y así poder recuperar el identificador de usuario para posteriormente hacer una multiconsulta de los datos requeridos que se muestra a continuación.

```
// Multiconsulta que recupera las URLs del tamaño grande y pequeño de las fotos del usuario

$query1 = 'SELECT aid FROM album WHERE owner IN (SELECT uid FROM user WHERE uid=me())';

$query2 = 'SELECT src_big_webp, src_small FROM photo WHERE aid IN (SELECT aid FROM #query1)';

$queries = '{
    "query1": "' . $query1 . '",
    "query2": "' . $query2 . '"
}';

//Envío la consulta a la API
$albums = $facebook->api(array(
    'method' => 'fql.multiquery',
    'queries' => $queries
));
```

Esta multiconsulta se realiza en FQL, el lenguaje de consultas proporcionado por Facebook y será enviada a la API para que la procese y nos devuelva la información.

API de Instagram

Como ya había comentado al inicio del documento, Instagram ha sido comprado por Facebook lo cual nos permite conectar con su API de manera muy similar a Facebook. Previamente crearemos la cuenta de desarrollador, añadimos la librería a nuestro proyecto y con los datos APP ID y APP Secret crearemos el fichero de configuración *instaconfig.php* El funcionamiento es muy similar, así que no lo detallaremos como anteriormente he hecho con Facebook, tan solo vamos a recalcar las diferencias.

En el caso de Facebook, los redirects o direcciones de vuelta donde envía Facebook al usuario son libres, es decir, pueden ser varias páginas diferentes siempre que estén dentro del dominio indicado en la creación de la aplicación.

Esto último es configurable y podríamos determinar una URL en concreto si quisiéramos un mayor control, que es lo que Instagram nos permite, tan solo una URL válida a la cual redirigir al usuario. En nuestro caso será la página *instafotos.php*, que es la que hemos configuramos la aplicación registrada en Instagram mostrado en la ilustración 14.

En la desactivación de la red social Instragram no haremos nada puesto que solo nos ofrece redirigir al usuario a esta página de logout:

LOGOUT URL: <https://instagram.com/accounts/login/?next=/accounts/logout/>

y como antes hemos explicado en el caso de Facebook, a nuestra aplicación solo le interesa no mostrar las fotos y eliminar sus datos internos de esta red, no deslogar al cliente por completo en sus sesión del navegador.

Gestionar clientes

Informar de un problemaRegistrar un nuevo cliente

Iramapfc

ELIMINAREDIT

CLIENT INFO

CLIENT ID	0456a0e64d614367a43705a96c4f2b37
CLIENT SECRET	cccccccccccccccccccccccccccccccccccccc
WEBSITE URL	http://www.casairama.com
REDIRECT URI	http://www.casairama.com/inter/instafotos.php

pfc

Ilustración 14: Parametros de configuración de la aplicación registrada en Instagram.

Otra de las diferencias es el acceso a los datos. Aquí no usaremos consultas en un lenguaje propio, aquí accedemos mediante el uso de la función `getUserMedia()` después de haber creado previamente el objeto `Instagram`. A esta función le pasamos el identificador del usuario que previamente hemos solicitado al recibir el objeto `token`. La llamada a esta función devuelve el *endpoint media*, con el cual ya podemos recuperar toda la información de las imágenes en forma de URL y almacenarla en un vector de la siguiente forma:

```

$i=0;
foreach ($media->data as $foto) {
    $foto[$i]=$foto->images->standard_resolution->URL;
    $i+=1;
}

```

Como podemos ver el funcionamiento es el mismo, lo único que varía un poco son las peticiones a la información. Cada red social tiene su API y con esta su pertinente documentación en la cual podemos encontrar explicaciones y ejemplos de su uso y funcionamiento.

4.5. Diseño e implementación del Front-End

Una vez analizado todo el lado del back-end explicaremos toda la parte front-end, la parte que refleja todo el trabajo que el back-end realiza. Primeramente vamos a explicar que es un diseño responsive y que técnica hemos utilizado para desarrollar la interficie gráfica de usuario de la aplicación y finalmente explicaremos el uso de Initializr como plantilla base para el desarrollo del front-end así como las técnicas para conseguir grids fluidos.

Diseño responsive con Initializr

El diseño *responsive* o diseño adaptable es una filosofía en el desarrollo de aplicaciones y páginas web que permite que sean visualizadas perfectamente en todo tipo de dispositivos sin necesidad de crear diferentes versiones de la aplicación web. Haciendo referencia a lo explicado en el apartado 3, la plantilla que nos proporciona Initializr para el desarrollo del front-end, determinaremos como la hemos usado para construir esta parte de nuestro sitio web.

Como bien dijimos, ésta no es una plantilla completamente vacía como la que proporciona Boilerplate. Partiendo de su base por defecto diseñamos el formato general de la aplicación siguiendo el concepto **mobile-first**. Este concepto nace del actual cambio en la manera de navegar de las personas y consiste en diseñar primero la aplicación para terminales móviles y luego adaptarlo progresivamente a pantallas de ordenador, como muestra la siguiente figura.

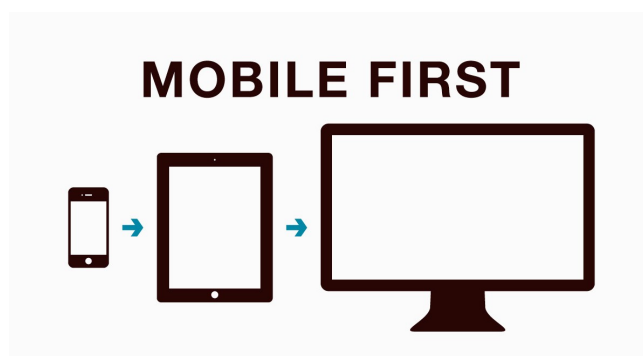


Ilustración 15: Concepto de diseño mobile-first.

Mobile-first es una innovación que obliga al diseñador y maquetador a priorizar la información disponiéndola en un espacio reducido como primeramente será la pantalla del dispositivo del smarthpone, obligándole a categorizar y ordenar la información según la importancia que ésta tenga. Por consecuente esta técnica, además de ayudar a conseguir un diseño eficaz y responsive, ayuda a gestionar los datos y a eliminar aquellos redundantes o sobrantes lo que hará que el sitio web sea más ligero, organizado y sencillo. A continuación mostro dos imágenes que representan dos tamaños diferentes que adopta nuestra aplicación, el primero para un smartphone y el segundo para un ordenador.

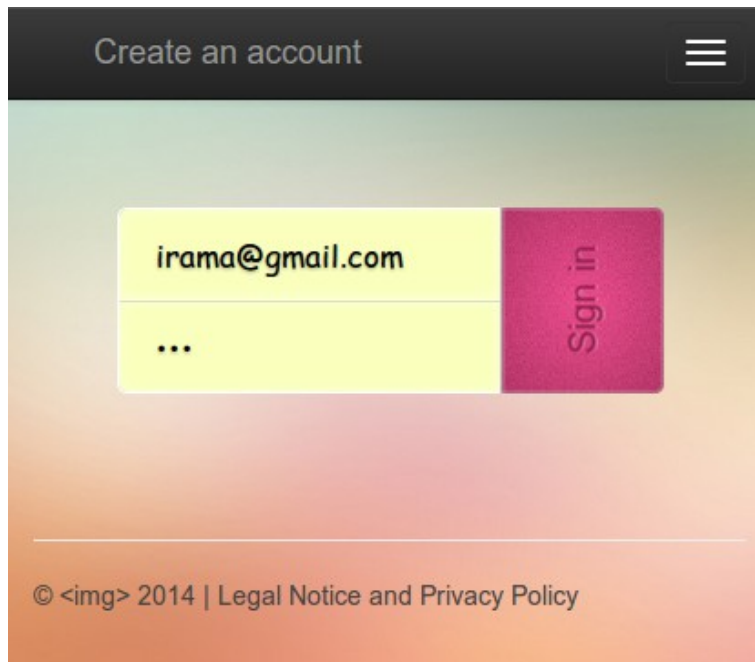


Ilustración 16: Vista reducida de la aplicación en dispositivos pequeños.

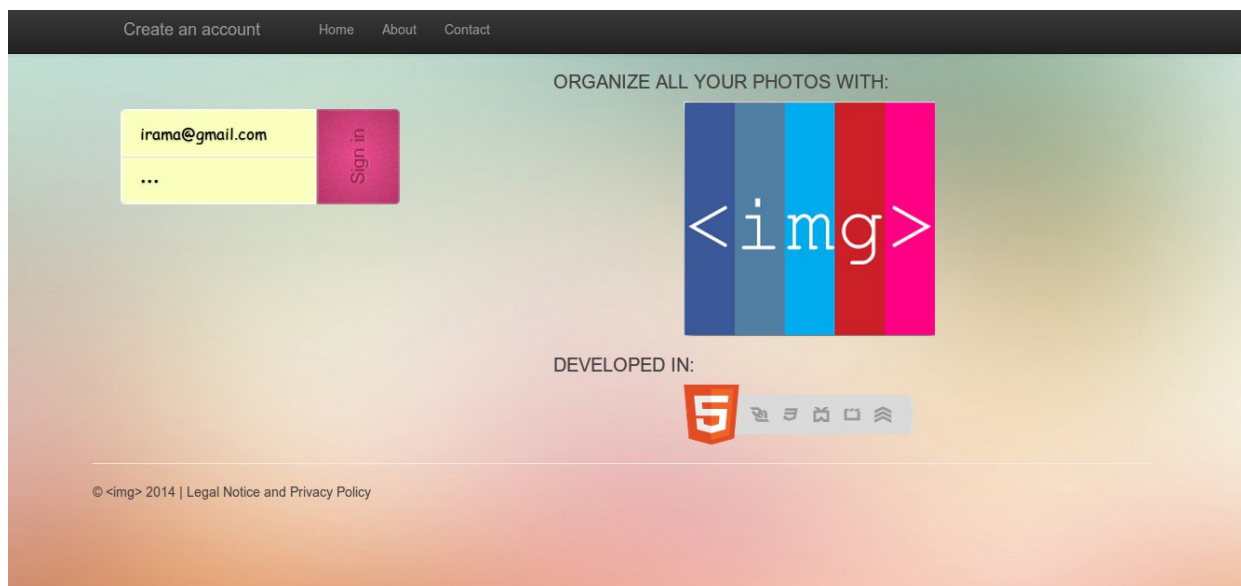


Ilustración 17: Vista ampliada de la aplicación en dispositivos grandes.

Media-queries

La diferencia que podemos observar entre la ilustración 16 y la ilustración 17 es que, en la primera ilustración el menú de navegación se ha contraído y se muestra mediante el icono  y la imagen de la derecha ha desaparecido.

Estos dos efectos se consiguen usando las *media-queries* en la hoja de estilos CSS3. En nuestro caso hemos usado las que el proyecto Initializr nos proporciona definidas en el fichero *bootstrap.css* aunque es muy importante entender su funcionamiento para poder desarrollar la aplicación con éxito. La estructura básica de una media-query es la siguiente:

```
@media (max-width: 980px) {  
    //Propiedades CSS a aplicar a las ventanas de tamaño 980px o menor  
}  
@media (min-width: 980px) {  
    //Propiedades CSS a aplicar a las ventanas de tamaño 980px o mayor  
}
```

Para el ejemplo anterior hemos mostrado con imágenes los resultados usando una *media-querie* para mostrar la imagen, es decir, el div contenedor de la imagen la derecha inicialmente se declara en las hojas de estilo como oculto y es la *media-querie* la encargada de comprobar si se encuentra en una pantalla más grande determinando esta dimensión en píxeles de anchura y en caso afirmativo cambiar el modo de display del contenedor. A continuación mostramos un trozo de código que define lo anteriormente explicado.

```
@media (min-width: 1200px) {  
    .visible-sm.visible-lg {  
        display: block !important;  
    }  
}
```

De esta forma vamos maquetando y definiendo los estilos de cada contenedor, y elemento de la página para determinar su posición dependiendo de la resolución en píxeles que muestre nuestra aplicación.

Personalizar Initializr

Como hemos comentado en varias ocasiones Initializr es una plantilla front-end sobre la que desarrollamos nuestro proyecto. Inicialmente viene con una serie de estilos **Bootstrap** definidos y unas determinadas funciones Javascript. A parte de esto es muy interesante la propuesta que ofrece. Entre otras posibles opciones para los estilos y scripts propios incluye dos archivos, *main.css* y *main.js* que son los que se usan para personalizar y definir la apariencia visual e interacción del usuario.

Estos ficheros vacíos permiten que el usuario incluya su personalización sin tener que modificar la base. Por ejemplo, nosotros para cambiar el background de las páginas hemos redefinido esta característica en *main.css* mediante el siguiente código:

```
body {
  background: #eedfcc URL(../img/bg3.jpg) no-repeat center top;
  -webkit-background-size: cover;
  -moz-background-size: cover;
  background-size: cover;
  font-family: 'Montserrat', Arial, sans-serif;
}
```

De esta forma si bien no se encontrara nuestra imagen de fondo *bg3.jpg* o el navegador fallase al cargarla, Initializr automáticamente recurriría a sus ficheros de configuración base, en este caso sería *bootstrap.css*, para mostrar el fondo por defecto y no dejar así, la aplicación incompleta. Para que se haga efectivo este cambio de background, debemos incluir en la página deseada el fichero *main.css*.

En el fichero *main.js* tan solo hemos definido una pequeña función. Que nos permite enviarle el identificador de un div y permitir cargar otro en su lugar.

Esta es una de las características más importantes de esta plantilla, que usa *Modenizr* para determinar qué características tiene el navegador y tratar de adaptar el contenido web de forma que le permita al navegador mostrarlo en su totalidad.

4.6. Test de Usabilidad

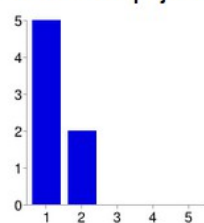
En este apartado hemos realizado un pequeño test de usabilidad a personas cercanas de nuestro entorno. Los datos que hemos obtenido son los siguientes:

7 respuestas

[Publicar datos de análisis](#)

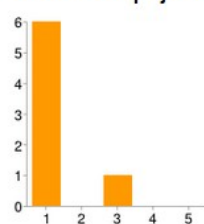
Resumen

Valore la complejidad de registro de la aplicación



1	5	71%
2	2	29%
3	0	0%
4	0	0%
5	0	0%

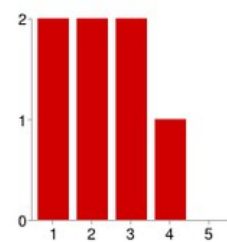
Valore la complejidad de login de la aplicación



1	6	86%
2	0	0%
3	1	14%
4	0	0%
5	0	0%

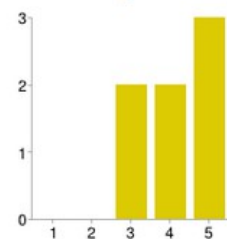
Ilustración 18: Respuestas 1 y 2 del test de usabilidad.

Le resulta difícil entender el funcionamiento global de la aplicación



1	2	29%
2	2	29%
3	2	29%
4	1	14%
5	0	0%

Como de agradable le parece la aplicación?



1	0	0%
2	0	0%
3	2	29%
4	2	29%
5	3	43%

Ilustración 19: Respuestas 3 y 4 del test de usabilidad.

Le parece útil esta aplicación web?



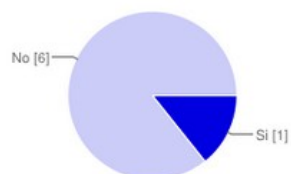
Si	7	100%
No	0	0%

Usaría en un futuro esta aplicación?



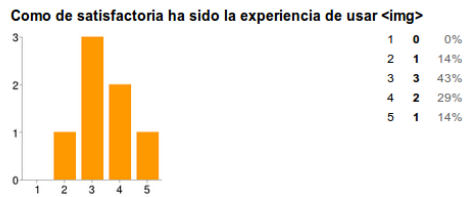
Si	6	86%
No	1	14%

Si la usara, pagaría por ella?



Si	1	14%
No	6	86%

Ilustración 20: Respuestas 5, 6 y 7 del test de usabilidad.



Cambiarías alguna funcionalidad? Cuál?

No hay mensajes de ayuda Pagaría por ella si tuviera muchas mas redes sociales y ofreciera más servicios. No salen mensajes de ayuda y hay cosas que no se entienden Cambiaría los menús de navegación No
 Me gustaría que hubieran más opciones para las fotos Nada

Ilustración 21: Respuestas 8 y 9 del test de usabilidad.

En conclusión, a nuestros usuarios les ha parecido fácil el acceso y el registro. Les parece más o menos agradable a la vista e intuitiva. Todos creen que es una buena idea esta aplicación y solo uno de ellos no la utilizaría, por otro lado de los siete usuarios, solo uno pagaría por ella. Finalmente la experiencia de los usuarios ha sido positiva y los comentarios ayudarán a su mejora y evolución.

5. Planificación y plan de empresa

5.1. Planificación

En este apartado vamos a mostrar la planificación que se ha llevado a cabo en el desarrollo del proyecto mediante el uso de un diagrama de Gantt, el cual nos permite exponer el tiempo de dedicación para realizar una serie de tareas y actividades que comprenden un tiempo total de desarrollo del trabajo final de grado.

Las reuniones con el tutor han sido de media 1 reunión cada 15 días, y las horas de trabajo que comprenden desde septiembre del 2013 hasta mayo del 2014 se han sido 3 horas de media al día sin tener en cuenta fines de semana. El período comprendido entre el 1 de junio al 20 de junio incluye la ampliación a 8 horas diarias contando fines de semana.

La planificación como podemos observar se divide en cuatro tareas fundamentales:

Id	Nombre de tarea	Duración	Comienzo
1	T1. Planificación	5 días	jue 05/09/13
4	T2. Análisis previo	49 días	lun 16/09/13
12	T3. Diseño e Implementación	137 días	jue 21/11/13
23	T4. Preparación final	17 días	vie 30/05/14

Ilustración 22: Tareas principales del diagrama de Gantt.

La tarea 1 abarca desde la información del funcionamiento de la asignatura Trabajo Final de Grado, pasando por el análisis y propuestas de diferentes proyectos hasta la elección final y asignación tanto de tutor como de proyecto.

En la tarea 2 se contempla la fase de investigación y elección de las herramientas, la preparación del entorno de trabajo y los primeros diseños de los casos de uso, la base de datos y de la interfaz gráfica.

La tarea 3 y más extensa comprende toda la fase de desarrollo e implementación. También incluye la resolución de errores y la primera parte de la redacción de la memoria del proyecto.

Finalmente en la tarea número 4 se han realizado los últimos pasos que han sido corrección de errores y últimos cambios, redacción final de la memoria y revisión general y entrega del proyecto.

Diagrama de gantt

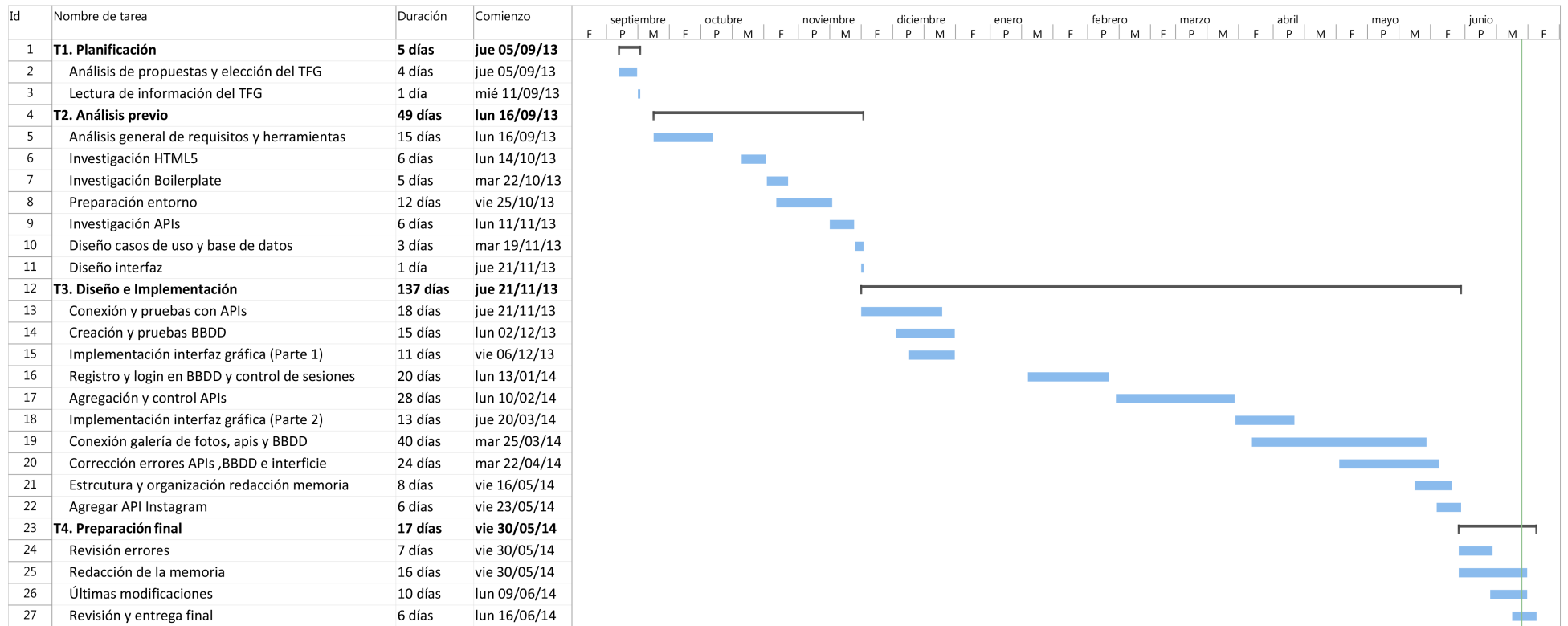


Ilustración 23: Dagrama de Gantt.

5.2. Plan de empresa

A continuación analizaremos el plan de empresa que nos serviría para llevar a cabo una aplicación como la nuestra y conseguir rentabilizar el proyecto. Exponemos la oportunidad de negocio de la que hablábamos en el capítulo de introducción y estimaremos la viabilidad económica.

Este plan de empresa lo dividiremos en dos apartados principales, el primero el cual atiende al modelo de negocio y las acciones estratégicas y el segundo, que incluye toda la parte financiera del proyecto así como un pequeño balance genérico.

1) Modelo de negocio

La previsión que estimamos para el desarrollo de la aplicación web requiere de un período máximo de 6 meses. A partir del sexto mes la aplicación saldrá al mercado. El modelo de negocio lo establecemos diferenciando los posibles clientes potenciales, por un lado existirá el cliente que utilizará la aplicación gratis o lo que se suele decir *Free user* y por el otro el *premium user* o lo que equivale a usuario de pago.

La ilustración 24 detalla que el *free user* podrá acceder a nuestra aplicación por un coste de 0€ pero con la restricción de un máximo de 2 activaciones de las redes sociales que se ofrezcan. Además esta modalidad incluye publicidad de terceros en la aplicación.

Sin embargo, el *premium user* pagará 0,99€ mensuales y podrá disfrutar de la activación de tantas redes sociales como la aplicación le ofrezca y no contendrá ningún tipo de publicidad externa.

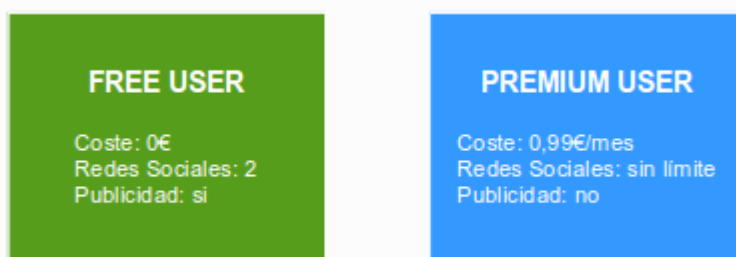


Ilustración 24: De izquierda a derecha, características del usuario free y premium.

Uno de los objetivos de esta aplicación siempre ha sido ofrecer la mayor versatilidad posible para llegar a los máximos usuarios posibles. La ventaja de esta finalidad es que, al ser compatible con el 100% de los dispositivos móviles existentes, solo necesitaríamos traducirla a los diferentes idiomas de los posibles clientes más potenciales para así poder abarcar un mercado internacional.

A continuación podemos comprobar una previsión del número de usuarios necesarios para la rentabilidad del proyecto. Esta previsión a dos años vista se ha estimado estableciendo el mes 14 como mes de umbral de rentabilidad, o lo que se suele llamar punto muerto que en el siguiente punto explicaremos.

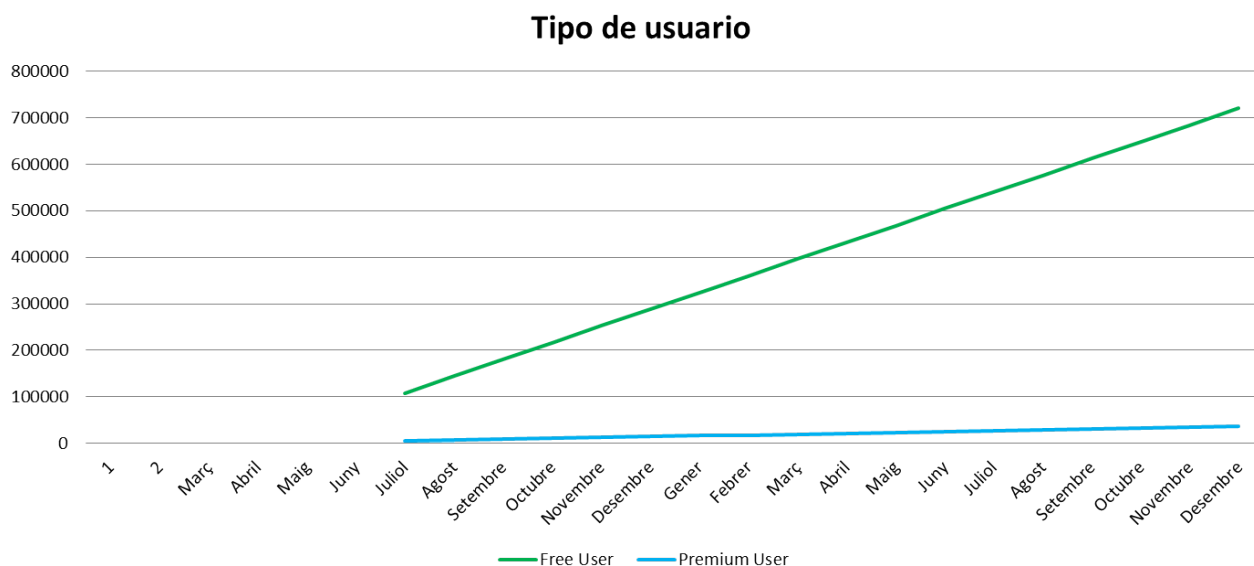


Ilustración 25: Previsión a dos años vista de usuarios free y premium.

La gráfica anterior muestra la previsión de usuarios necesarios para rentabilizar el proyecto a partir del mes 12, evidentemente el número de usuarios gratis será superior al de usuarios premium, los cuales hemos contabilizado 5% del total y que producirán un ingreso de 0,99€ mensuales cada uno.

Suponiendo unos 135.000€ de capital inicial se necesitarían aproximadamente 15.000 usuarios premium para en el mes 12 comenzar a ganar dinero.

2) Viabilidad económica

En este apartado evaluamos los costes de personal y los de equipamiento. Prevemos que solo habrán costes fijos y no costes variables. Finalmente haremos un balance global para determinar cuando la empresa dejará de perder dinero y comenzará a ganar beneficios.

Costes de personal

Como vemos en la tabla número 5 el equipo estará compuesto de un jefe de proyecto que se encargará de la dirección y toma de decisiones. Un diseñador web para la creación de la interfaz de usuario y dos programadores para el desarrollo de la aplicación. Finalmente necesitaremos un community manager que será el encargado de gestionar la publicidad de la aplicación, darla a conocer y captar nuevos clientes.

Uno de los dos programadores tendrá una duración determinada, el período de desarrollo de la aplicación, seis meses, puesto que es durante este tiempo cuando más carga de trabajo habrá. Una vez pasados los seis meses la aplicación será lanzada al mercado, y es desde entonces cuando sólo contaremos con un programador. Las funciones de éste serán el mantenimiento constante de la aplicación así como del desarrollo progresivo de mejoras o pequeñas ampliaciones.

Puesto	Plazas	Salario bruto
Jefe de proyecto	1	30.000€
Diseñador web	1	24.000€
Programador aplicación	2	25.000€
Comunity Manager	1	24.000€

Tabla 5: Costes de personal

En cuanto a la percepción al cumplimiento legislativo vigente en acuerdo a la creación de un contrato laboral, a cada trabajador se le realizará una retención del 30% de seguridad social y un 10% de IRPF sobre su salario bruto que se muestra acorde al puesto de trabajo en la tabla numero 5.

Costes de equipamiento

En la tabla 6 mostramos los costes totales fijos que se prevén.

En los costes de equipamiento tendremos cuatro grandes grupos a diferenciar, los costes de servicio que incluyen la compra de 5 ordenadores portátiles, un presupuesto para equipamiento adicional, calculado en unidades y el alquiler mensual de un servidor web.

En los costes de categorizados como suministros encontramos gastos mensuales del alquiler de una oficina, la luz, el agua, servicios de limpieza y el seguro de la empresa.

Finalmente como otros gastos englobamos el coste de una gestoría y 500€ de publicidad para la aplicación mensuales.

Oficina	Concepto	Cantidad	Coste
Servicio	Ordenadores portátiles	5	500€/u
	Equipamiento adicional	1	600€/u
	Alquiler Servidor web	1	500€/mes
Suministros	Alquiler oficina	1	500€/mes
	Luz	1	100€/mes
	Agua	1	200€/mes
	Limpieza	1	100€/mes
	Seguro	1	50€/mes
Otros	Gestoría	1	200€/mes
	Publicidad	1	500€/mes

Tabla 6: Costes materiales.

Balance de ingresos y costes

Los ingresos provendrán como hemos explicado en el modelo de negocio principalmente de los usuarios premium que pagarán 0,99€ mensuales. Aunque hemos incluido publicidad de terceros en la modalidad *free user* despreciamos esta ganancia puesto que es casi imperceptible por el momento.

Para poder hacer balance entre estos ingresos y el gasto económico que supone la creación de la empresa así como el desarrollo y lanzamiento de la aplicación, tal como hemos mostrado con las dos tablas anteriores, hemos creado la siguiente una gráfica que nos permite ver la trayectoria económica.

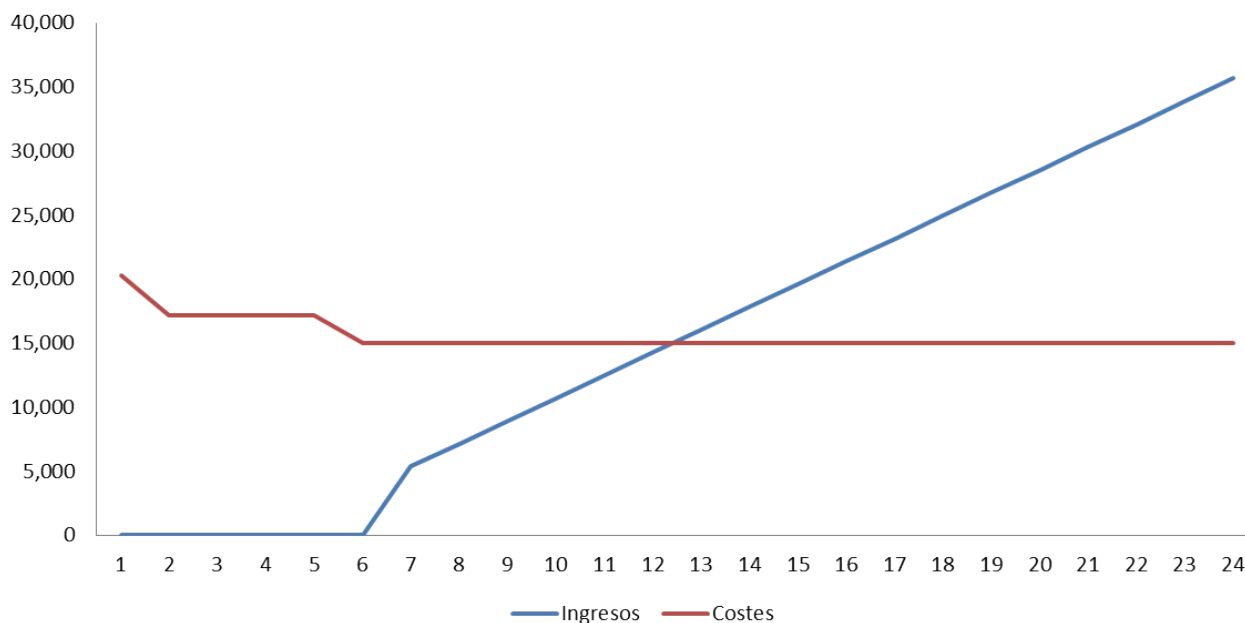


Ilustración 26: Ingresos frente a costes de la aplicación .

Como podemos observar en el gráfico anterior los ingresos son 0€ hasta el sexto mes, el del lanzamiento de la aplicación. Una vez puesta en el mercado los ingresos generados por la captación de usuarios premium comenzarán a aumentar.

Hasta entonces solo habrán gastos, en los que se contempla además de los explicados anteriormente la campaña publicitaria en la que se estudiará el perfil del usuario que se necesitará para determinar el qué clientes son más potenciales.

Además a partir del sexto mes se prevé que los gastos sean constantes puesto que no será necesario hacer ninguna inversión adicional. Para seguir manteniendo la aplicación tendremos un programador y un diseñador que irán poco a poco mejorando y ampliando sus funcionalidades, así como el jefe de proyecto y community manager seguirán con sus respectivas funciones. El pequeño desnivel que sufre la curva de los costes se produce debido a que habrá un sueldo menos que pagar, el del programador de apoyo que antes hemos explicado.

Como consecuencia del análisis y balance anteriormente expuesto podemos dar una visión económica global teniendo en cuenta el contraste entre gastos acumulados y la necesidad de financiación.

En la siguiente ilustración podemos observar la gráfica que muestra que en el mes 12 llegamos al umbral de rentabilidad o punto muerto como antes hemos nombrado, suponiendo una inversión inicial de 135.000€.

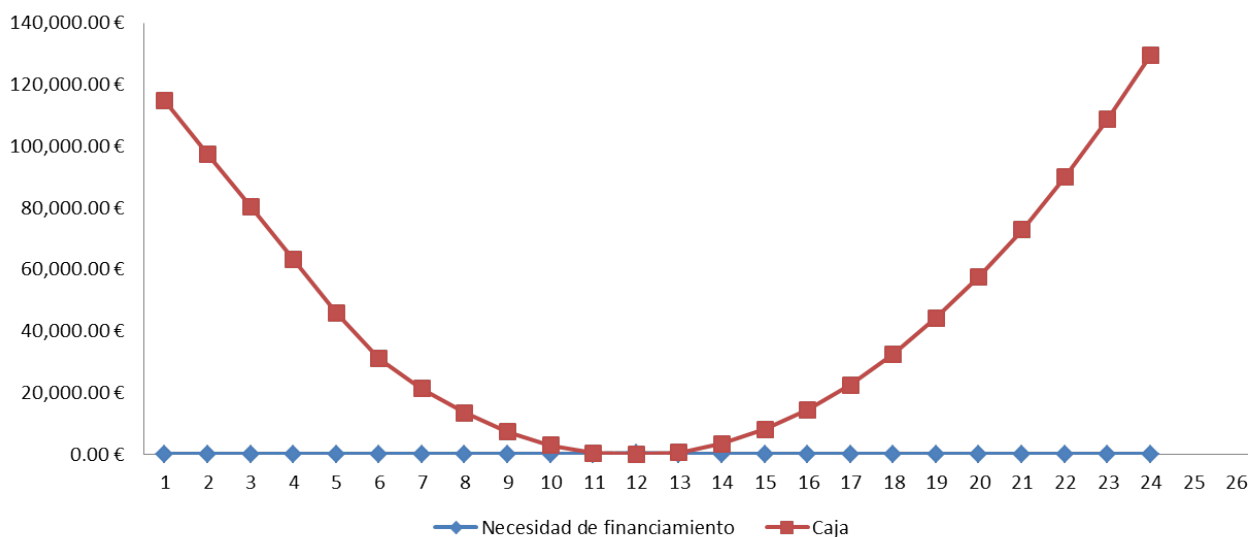


Ilustración 27: Curva del punto muerto.

Esta gráfica que presenta en el mes 12 su punto muerto, demuestra que, con el capital de 135.000€ iniciales y contando con un mínimo aproximado de 15.000 usuarios, ese mes la empresa no obtendría ni beneficios ni pérdidas, es decir, del mes 12 en adelante comenzaría a generar beneficios.

Finalmente hemos consultado algunas fuentes y el último informe de Fundación Telefónica, actualmente existen alrededor de 6.800 millones de usuarios con dispositivo móvil. De éstos, el 64,1% se encuentra dado de alta en alguna red social. La gran cuota de mercado a la que nos dirigimos nos facilitará lograr el objetivo del punto muerto.

Fuente: <http://www.puromarketing.com/12/19020/mundo-casi-tantos-telefonos-moviles-como-personas.html>

6. Conclusiones

Dado que hemos llegado hacia el final del documento haré un balance de todo lo que aquí hemos expuesto. Puedo decir que el proyecto desarrollado ha cumplido con el objetivo principal propuesto, crear una aplicación web con las características de un agregador social de imágenes siendo responsive y basado en la tecnología HTML5, CSS3 Y Javascript.

La web cumple con los requisitos expuestos en el apartado 2, cubre los casos de uso que, en el apartado 3 hemos explicado, e implementa la base de datos que hemos mostrado en el apartado 4. Pero bien hemos de ser conscientes que es una versión beta o inicial. Soy consciente de que quedan cosas por mejorar y optimizar pero en definitiva proporciona una primera base sólida sobre la que seguir trabajando y a la que le podemos añadir tantas extensiones como se nos ocurran.

Comenzaré explicando qué mejoras creo que deberían ser necesarias y ampliaciones me gustaría seguir desarrollando la aplicación, así como la conclusión final.

6.1. Mejoras y Ampliaciones

Como programadora y diseñadora de la aplicación web, detallaré brevemente por puntos que mejoras incluiría así como que posibles ampliaciones me parecen interesantes.

- *Menor número de páginas:* Me gustaría que la aplicación redujese su número de páginas, esto implicaría que muchos *containers* o *divs* fueran dinámicos para cargar solo la parte de página necesaria y no toda. Esto implicaría cambios en el front-end y pero sobre todo en el back-end teniendo que optimizar mucho más el código devuelto por el servidor.
- *Validación de formularios:* Los formularios deberían estar mejor validados y comprobados para evitar que el uso de robots informáticos accedan a ellos. Algunos de ellos deberían usar códigos de reconocimiento.
- *Optimizar del uso de APIs:* Es un mundo muy grande y complejo el que ofrecen las APIs de redes sociales como Facebook y creo que se podría optimizar y mejorar la obtención de datos así como su conexión.
- *Diseño:* El diseño de la aplicación considero que requiere más vida, imágenes más dinámicas y menús más modernos.

Como ampliaciones diré lo siguiente:

- *Hosting con SSL:* Actualmente el pack que tenemos contratado con 1&1 no dispone del certificado de SSL. Este certificado es muy importante puesto que fuerza el protocolo de transferencia segura de hipertexto (HTTPS) y proporcionaría mayor seguridad a nuestros usuarios creando lo que se llama una “conexión segura”.
- *Más redes sociales:* Cuantas más redes sociales contenga implementadas mayor será la oferta para nuestros usuarios.

- *Agregador de vídeos:* Dado que la nueva versión de HTML esta enfocada sobre todo al contenido multimedia, me gustaría poder incluir vídeos y reproducirlos desde nuestra web.
- *Aplicación móvil usando Phonegap:* Phonegap es un framework que nos permite adaptar nuestra aplicación HTML5, Javascript y CSS3 a diferentes sistemas operativos simulando ser una aplicación nativa más.
- *Poder compartir y descargar:* Sería casi fundamental poder compartir las fotos de una red social en otra red diferente usando , así como al crear un álbum de fotos tener la opción de descargarlo a nuestro dispositivo.
- *Reconocimiento de dispositivo:* Otra de las funcionalidades a añadir que me parecen muy interesantes es que la propia aplicación detectara el dispositivo desde el que accedemos y pudiera recuperar también las fotos que almacenamos en local.
- *Agregar filtros:* Éste es otro de los puntos fuertes que podría tener la aplicación. El uso de filtros marcos y demás elementos para personalizar las imágenes.

Como vemos se me ocurren mil y una mejoras y ampliaciones. Para concluir el documento quiero daré mi opinión personal. Empecé esta aventura algo perdida o como se suele decir, “como pollo sin cabeza”.

Los grandes problemas y dificultades que he tenido han sido situarme y entender que posibilidades me ofrecía el proyecto Boilerplate y diferenciarlo de las antiguas plantillas para la creación de páginas web. La conexión con las APIs y su gestión, sobre todo la de Facebook, fue uno de mis mayores quebraderos de cabeza. Quizás han sido los inconvenientes más grandes puesto que fueron las dos tareas importantes con las que comencé a desarrollar el proyecto.

Lo más importante que me llevo de haber realizado este proyecto ha sido el conocimiento y la experiencia que he adquirido durante este período de tiempo. Quizás si ahora tuviera que volver a hacerlo a sabiendas de lo vivido calcularía los tiempos de otra forma y organizaría las tareas en diferente orden. Considero que desarrollar un proyecto final de carrera no es solo crear una aplicación o un programa informático, sino que permite aprender a gestionar el tiempo, los problemas y adversidades que se presentan, hacernos crecer como profesionales y aprender cada día cosas nuevas.

Al principio no entendía el funcionamiento de muchas herramientas, ni sabía utilizar una API, ni lo que era un agregador social, pero en definitiva me siento orgullosa de haber sido capaz de aplicar conceptos e ideas que se me han enseñado a lo largo de la carrera. Además este proyecto me ha permitido ampliar, mis escasos conocimientos sobre la web y haber mejorado mis técnicas así como haber mejorado mi competencia en los lenguajes de programación que he utilizado.

Realmente ha sido muy gratificante partir de la nada y poder ir creciendo y creando este sitio web hasta cumplir el objetivo marcado. También me ha servido para darme cuenta, ahora que termino el período de enseñanza y debo introducirme en el mundo laboral, que éste es un campo de la informática en el me gustaría trabajar además de tener muchas salidas y un futuro prometedor.

7. Bibliografía

- Documentación HTML5 Boilerplate. URL: <https://github.com/h5bp/html5-boilerplate/blob/v4.3.0/doc/TOC.md>
- Initializr Bootstrap. URL: <http://getbootstrap.com/getting-started/>
- Guía básica html5. URL: <http://www.thuer.com.ar/blog/2009/html-5-para-disenadores-web>
- Tutoriales HTML5. URL: <http://www.html5rocks.com/es/tutorials/?page=1>
- Diseño responsive. URL: <http://www.esandra.com/disenio-web-responsive-por-donde-empezar>
- Galería de fotos. URL: <http://blueimp.github.io/Bootstrap-Image-Gallery/>
- Image Picker. URL: <http://rvera.github.io/image-picker/>
- Bootstrap. URL: <http://getbootstrap.com/css/>
- Documentación HTML5 Boilerplate. URL: <https://github.com/h5bp/html5-boilerplate/blob/v4.3.0/doc/TOC.md>
- Documentación API Facebook. URL: <https://developers.facebook.com/>
- Documentación API Instagram. URL: <http://instagram.com/developer/>
- Diposit digital de la UB. URL: <http://diposit.ub.edu/dspace/handle/2445/34465/simple-search?query=web>
- Wikipedia: URL: <http://es.wikipedia.org/>
- Markup validate service: URL: <http://validator.w3.org/>
- Encriptación passwords en PHP5.5. URL: <https://donnierock.com/tag/encriptacion/>
- Manual de PHP. URL: <http://www.php.net//manual/es/index.php>
- Manual MySQL: URL: <http://www.php.net//manual/es/set.mysqlinfo.php>

8. Anexos

8.1. Anexo I: Contenido ZIP

La carpeta src comprimida en .zip contiene dos subcarpetas principales:



Ilustración 28: Contenido de la carpeta src.

- 1) Una contiene la base de datos en un fichero comprimido:
`db328708091img.sql.gz`
- 2) La carpeta inter contiene toda la aplicación web que continuación mostramos de forma estructurada y transparente. El contenido esta organizado en diversas carpetas y a su vez en subcarpetas que contienen diversos archivos del proyecto. Solo detallamos brevemente el contenido y los scripts más relevantes.

Carpetas principales de la aplicación



Ilustración 29: Carpetas básicas del proyecto

La carpeta principal es *inter* y es la que contiene la división de los ficheros en carpetas de segundo nivel que son las que mostramos en esta imagen. Las páginas web que se mostrarán al usuario estarán en esta carpeta de primer nivel.

Contenido de la carpeta *inter*



Ilustración 30: Contenido de la carpeta inter.

Aquí se encuentran todos los ficheros que hemos definido en el apartado 4.2. Son los en mayoría los ficheros que se muestran al usuario.

- ***index.php***: el fichero de inicial de la aplicación.
- ***recargasrsesion.php***: contiene los divs o containers que iremos recargando dentro de la página *index.php*.
- ***inicio.php***: pagina de login.
- ***recargar.php***: contiene los divs o containers que iremos recargando dentro de la página *inicio.php*.

- **instafotos.php**: muestra las fotos de Instagram.
- **fbfotos.php**: muestra las fotos de Facebook.
- **interconfig.php**: configuración de conexión con Instagram.
- **fbconfig.php**: configuración de conexión con Facebook.
- **editalbum.php**: permite editar un album de fotos.

Contenido de la carpeta *css*

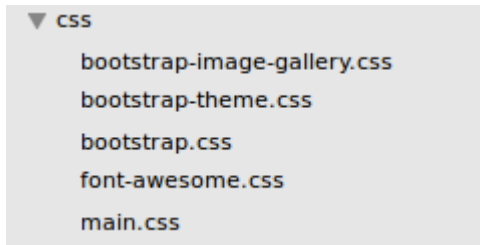


Ilustración 31: Contenido de la carpeta *css*.

Esta carpeta almacena las hojas de estilo *css*:

- **bootstrap.css**: hoja de estilo principal predeterminada.
- **bootstrap-image-gallery.css**: hoja de estilo de la galería de fotos.
- **font-awesome.css**: hoja de estilo con la fuente determinada del login.
- **main.css**: fichero para la personalización de estilos.

Contenido de la carpeta *extras*

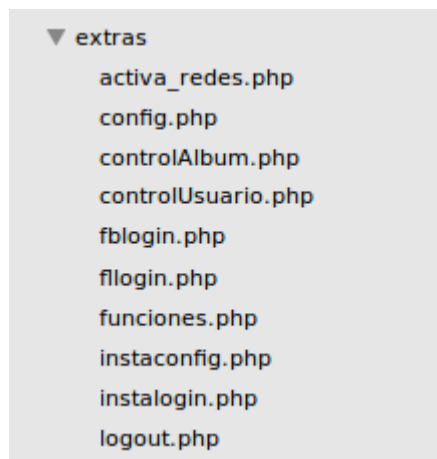


Ilustración 32: Contenido de la carpeta *extras*.

Esta carpeta contiene los ficheros que conforman el motor principal de la aplicación:

- **activa_redes.php**: Activa las redes sociales y controla el acceso a ellas.
- **config.php**: configura la base de datos.

- ***controlAlbum.php***: controla las posibles acciones a realizar sobre un álbum de fotos. Crearlo, renombrarlo, añadir fotos, eliminar fotos y borrar álbum.
- ***controlUsuario.php***: controla si un usuario se registra, se logea y cambia el email y/o el password.
- ***fblogin.php***: controla los accesos a Facebook.
- ***instalogin.php***: controla los accesos a Instagram.
- ***Logout.php***: desloga a un usuario de la aplicación.

Contenido de la carpeta *fonts*

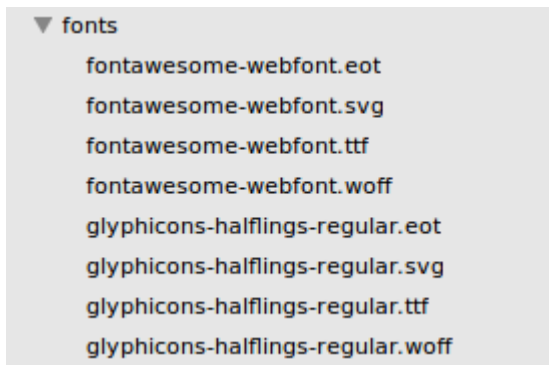


Ilustración 33: Contenido de la carpeta fonts

De esta carpeta solo cabe nombrar que incluye ficheros de configuración de fuentes de texto.

Contenido de la carpeta *img*

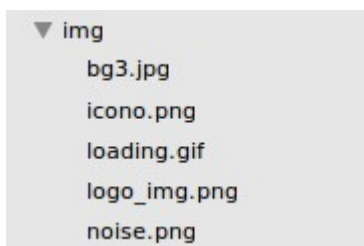


Ilustración 34: Contenido de la carpeta img

Las imágenes que conforman los estilos de la aplicación se almacenarán en esta carpeta.

Contenido y subcontenido de la carpeta *lib*

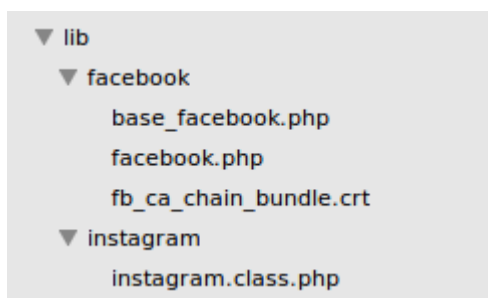


Ilustración 35: Contenido de la carpeta lib.

Esta es la carpeta que almacena todas las librerías. Contiene dos subcarpetas:

- **facebook:** contiene la librería de Facebook.
- **instagram:** contiene la librería de Instagram.

Contenido y subcontenido de la carpeta *js*

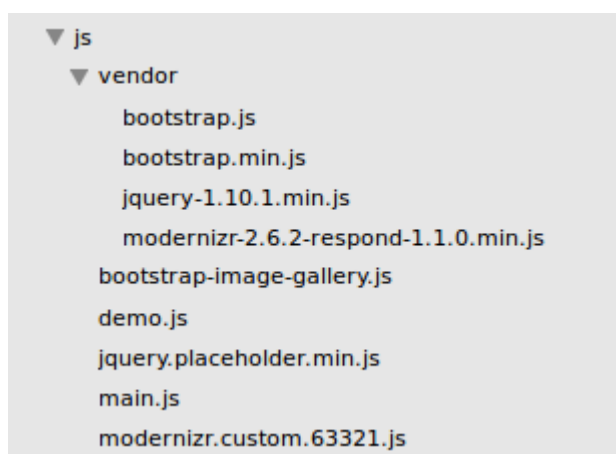


Ilustración 36: Contenido y subcontenido de la carpeta js

Contiene todo el código Javascript. Los archivos destacables son:

- **jquery-1.10.1-min-js:** contiene la librería de JQuery.
- **bootstrap-image-gallery.js:** controla la galería de fotos.
- **main.js:** fichero de personalización para incluir código js.

8.2. Anexo IV: Manual de usuario

A continuación se describe el manual de usuario para utilizar la aplicación.

Registrar usuario

El usuario debe introducir los datos solicitados en este formulario y pulsar el botón “Create”. Todos los datos son requeridos.

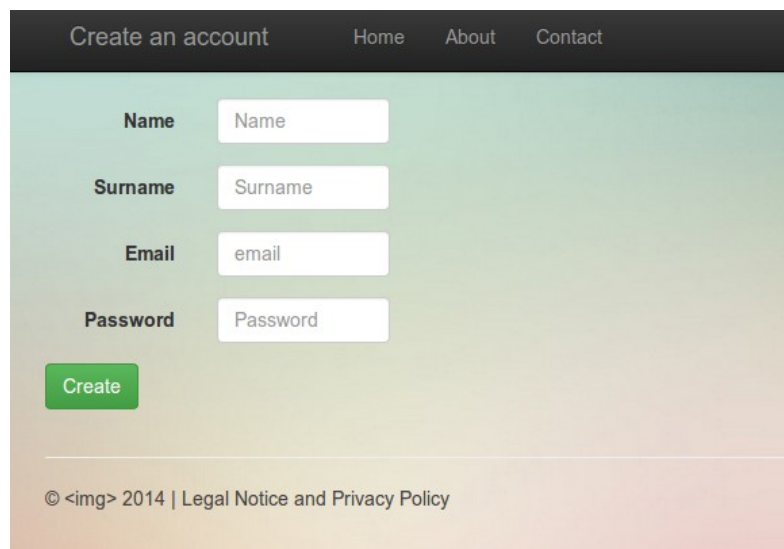
A screenshot of a web application's 'Create an account' page. The page has a dark header with the title 'Create an account' and navigation links 'Home', 'About', and 'Contact'. The main content area has a light green background and contains four input fields labeled 'Name', 'Surname', 'Email', and 'Password'. Each field has a placeholder text matching its label. Below the fields is a green 'Create' button. At the bottom, there is a footer with the text '© 2014 | Legal Notice and Privacy Policy'.

Ilustración 37: Registro de un usuario.

Logar usuario y deslogar

Si el usuario dispone de cuenta podrá logarse introduciendo el email y el password como se muestra en la siguiente imagen de ejemplo:

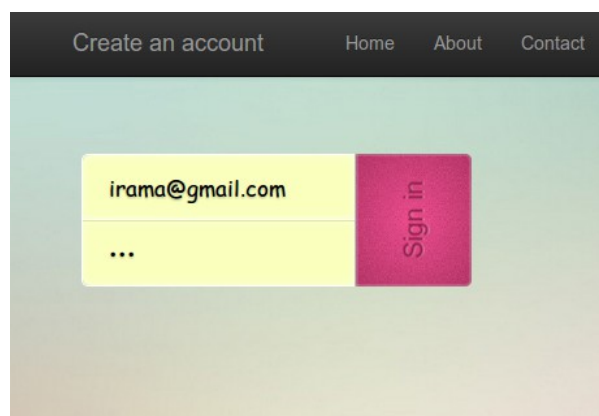
A screenshot of a web application's login page. The page has a dark header with the title 'Create an account' and navigation links 'Home', 'About', and 'Contact'. The main content area has a light green background and contains a login form. The form has two input fields: the first contains the email 'irama@gmail.com' and the second contains three dots '...'. To the right of the input fields is a pink 'Sign in' button.

Ilustración 38: Logar un usuario.

Para deslogar el usuario deberá pulsar el enlace que pone “Logout”.

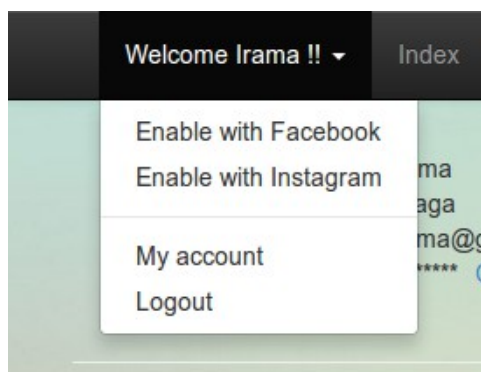


Ilustración 39: Menú que muestra la opción Logout.

Activar red social y ver fotos (Facebook)

1) Activar Facebook:

El usuario debe pulsar “Enable with Facebook” para activar esta red social.

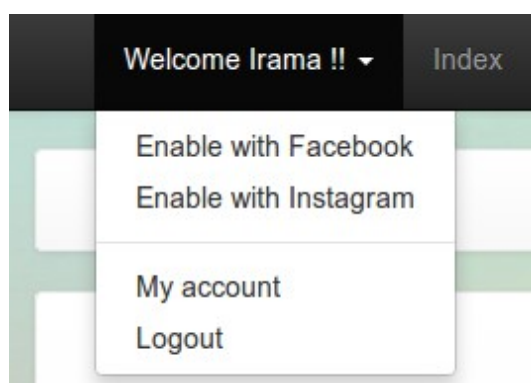


Ilustración 40: Activar red social.

2) Ver fotos

El usuario podrá ver los enlaces a las redes sociales activadas tal como se muestra en la siguiente imagen:

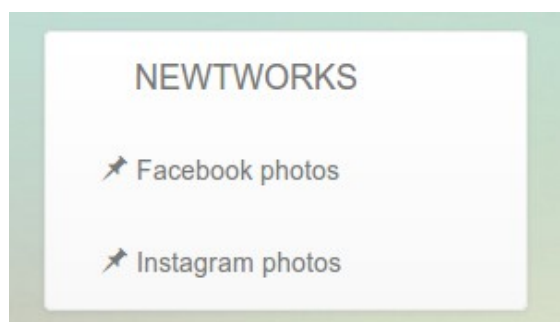


Ilustración 41: Vista de redes sociales activadas.

Al clicar sobre un enlace el usuario podrá ver las fotos de la red social deseada, como muestra la ilustración 42.

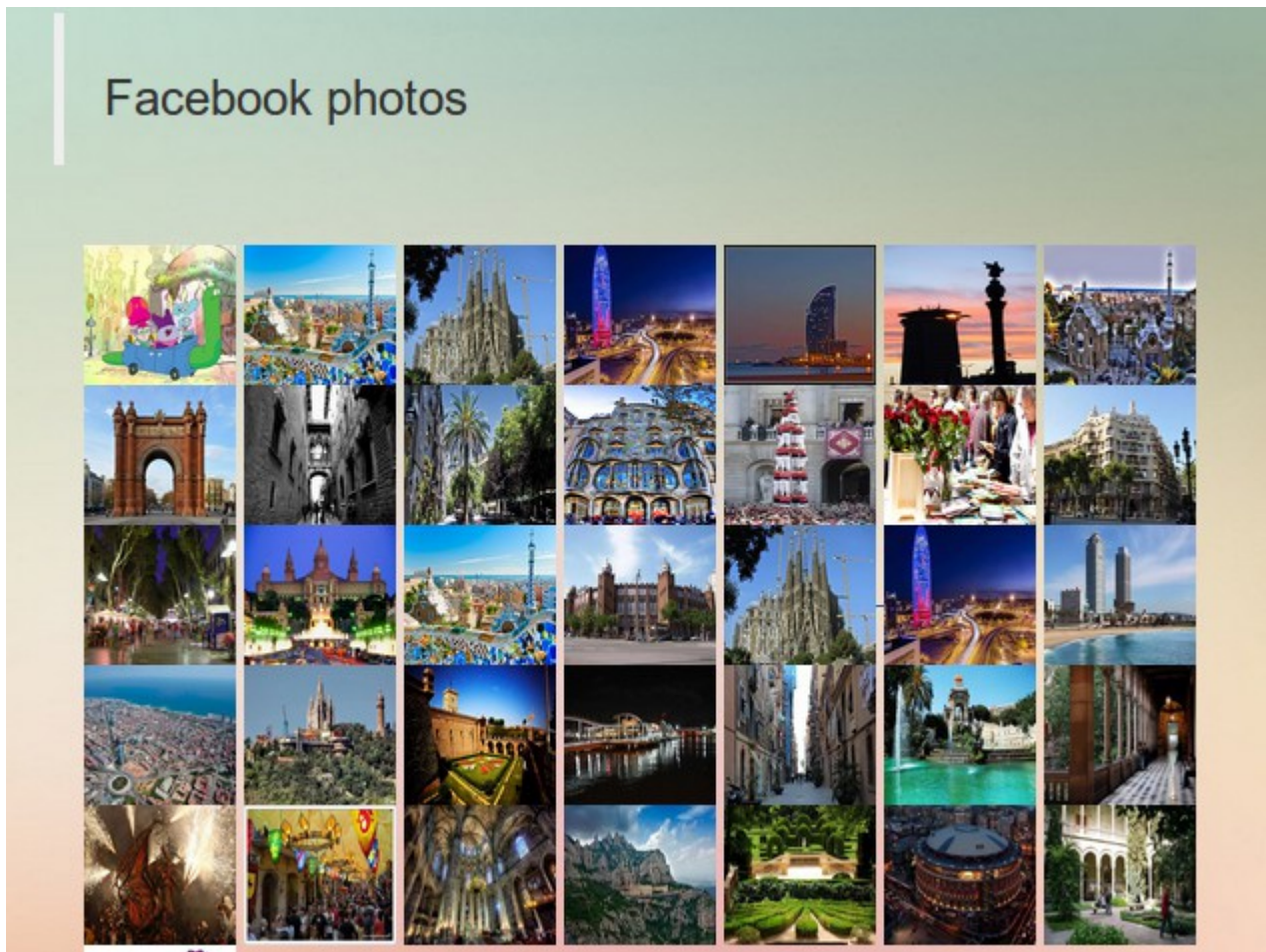


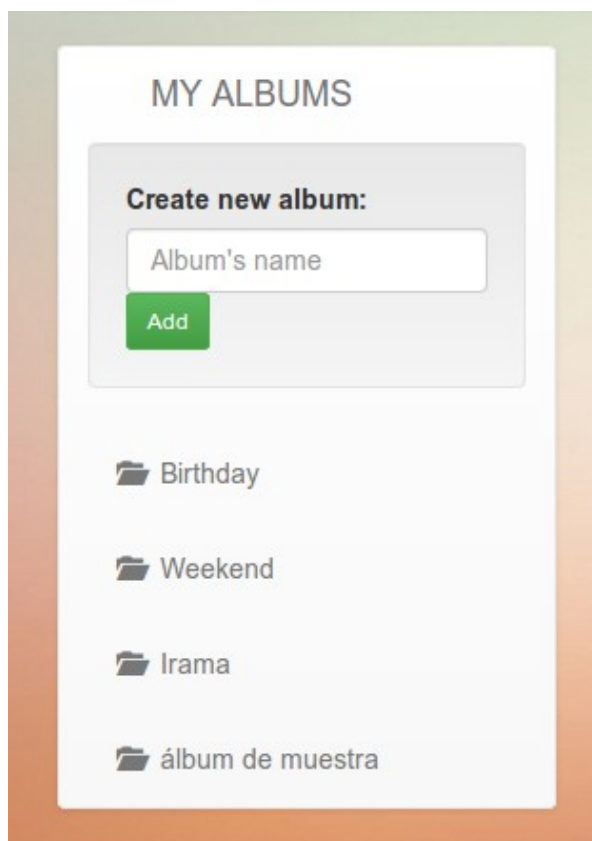
Ilustración 42: Vista de las fotos de Facebook.

Crear Álbum

El usuario debe introducir un nombre para el nuevo álbum y pulsar el botón “Add”.

Ilustración 43: Crear un nuevo álbum de fotos.

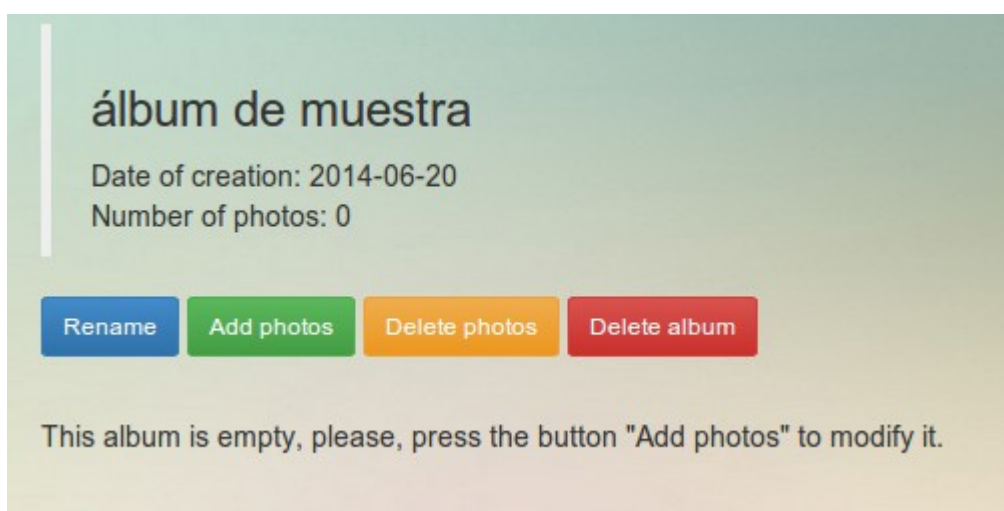
Este álbum se mostrará en la lista de debajo del formulario de la siguiente forma.



The screenshot shows a web interface titled "MY ALBUMS". It features a "Create new album:" section with a text input field labeled "Album's name" and a green "Add" button. Below this, there is a list of existing albums, each represented by a folder icon and a name: "Birthday", "Weekend", "Irama", and "álbum de muestra".

Ilustración 44: Lista de álbums con el nuevo álbum insertado.

Al pulsar el botón “Create” además de crear el álbum nos redirigirá a su página, que inicialmente tendrá este aspecto:



The screenshot shows the page for a newly created album titled "álbum de muestra". It displays the "Date of creation: 2014-06-20" and "Number of photos: 0". Below this information are four buttons: "Rename" (blue), "Add photos" (green), "Delete photos" (orange), and "Delete album" (red). At the bottom, a message states: "This album is empty, please, press the button 'Add photos' to modify it."

Ilustración 45: Nuevo álbum creado.

Añadir fotos a un álbum

Para añadir fotos pulsaremos sobre el botón “Add photos” que podemos ver en la ilustración 36, después elegiremos de que red social añadir las fotos de la siguiente forma:

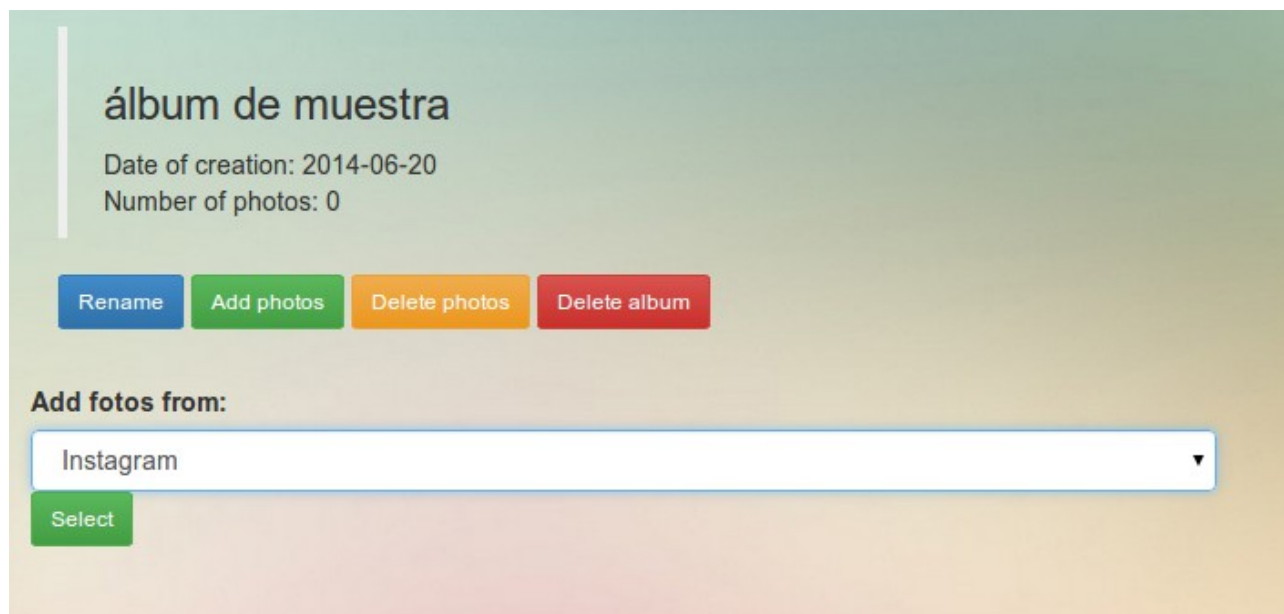


Ilustración 46: Seleccionar red social para añadir sus fotos a un álbum.

Finalmente seleccionaremos las fotos que queramos añadir y pulsaremos el botón “Add selected photos”. Las fotos seleccionadas se mostrarán con un reborde de color azul como se indica en la siguiente imagen:



Ilustración 47: Selección de fotos para añadir a un álbum.

Ver fotos

Pinchando sobre las fotos de nuestro álbum se mostrará la fotografía en grande como muestra la imagen de abajo.

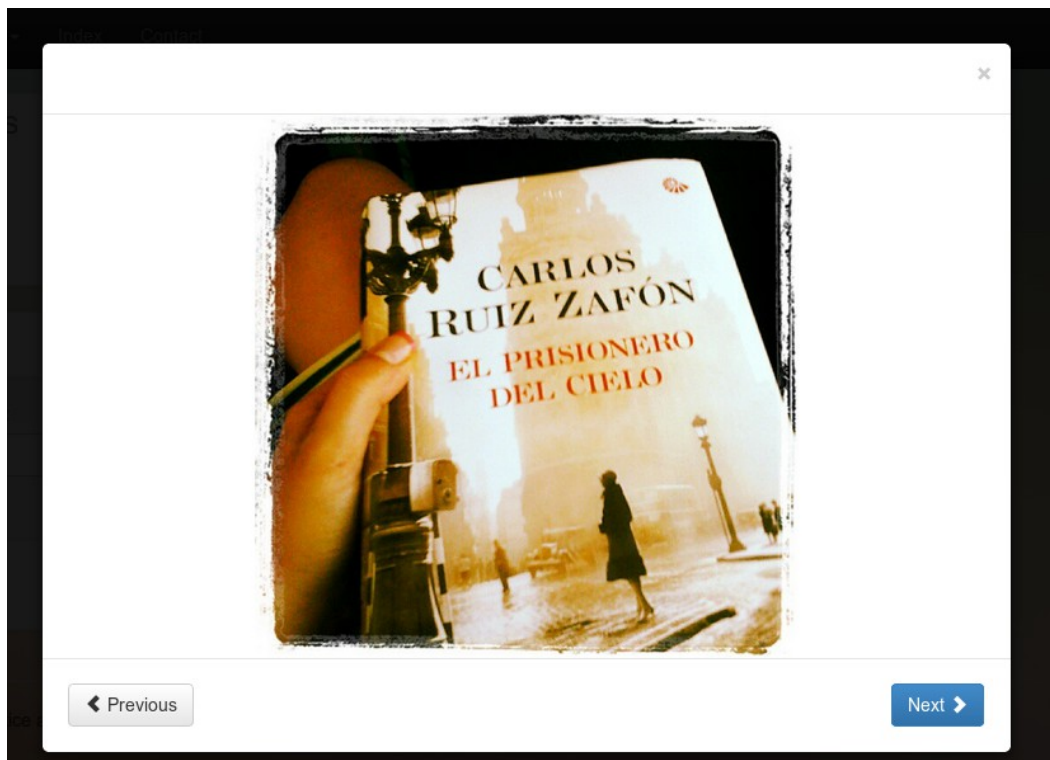


Ilustración 48: Vista principal de una foto.

8.2.1. Borrar fotos

Para borrar las fotos pulsaremos dentro del álbum el botón que dice “Delete photos”. Seleccionaremos las fotos que queramos borrar y pulsaremos el botón “Delete selected photos”.



Ilustración 49: Borrar fotos de un álbum.

Borrar álbum

Para borrar un álbum primero tendremos que entrar en él y pulsar el botón de “Delete album”. Finalmente nos pedirá la confirmación de que lo queremos borrar, tal como se muestra en la siguiente imagen:

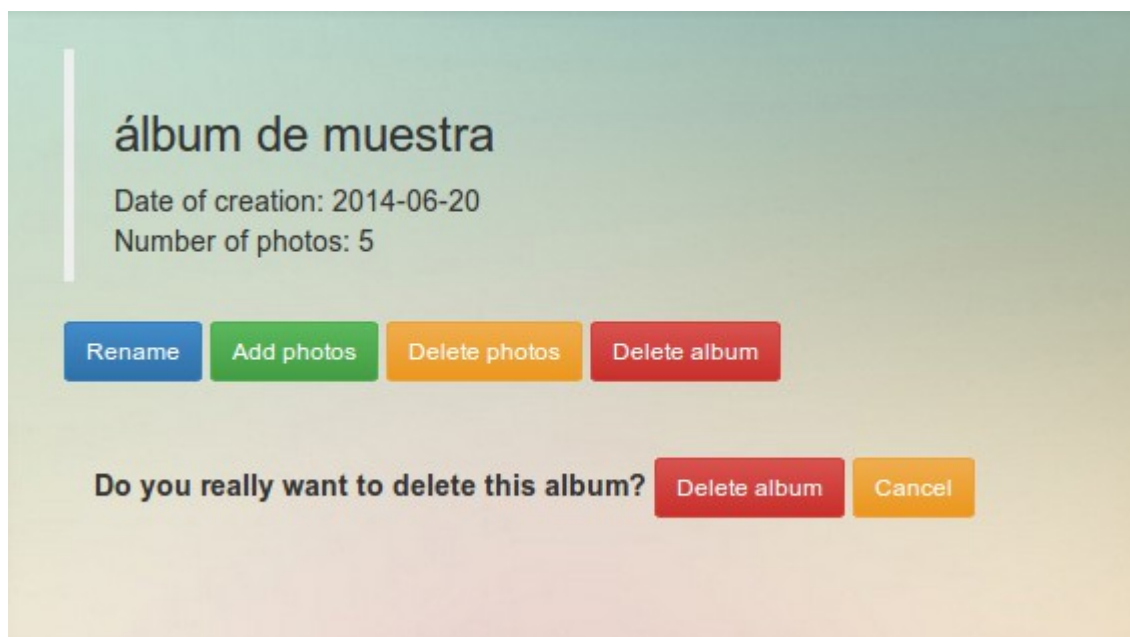


Ilustración 50: Confirmación para borrar un álbum.

Modificar datos de usuario

Para modificar los datos del usuario, este debe acceder al submenú izquierdo de la barra de navegación de la izquierda tal como muestra la figura 45.

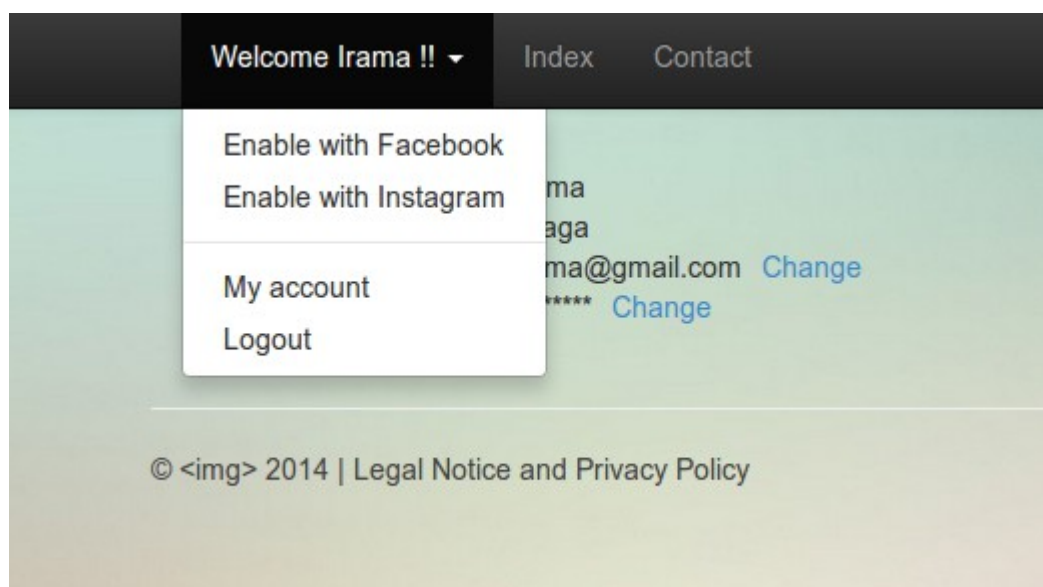


Ilustración 51: Acceso a la configuración de la cuenta de usuario.

Una vez dentro éste podrá cambiar su email o password tal como muestra el siguiente ejemplo:

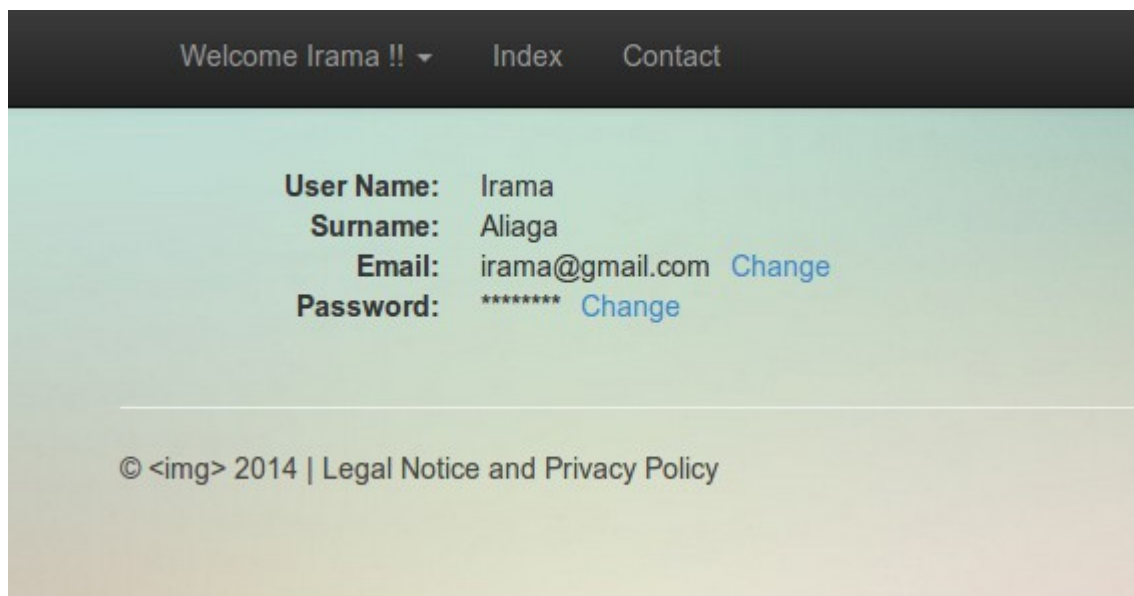


Ilustración 52: Datos del usuario.