

APLICACIÓN WEB PARA BUSCAR APARCAMIENTO Y COMPARTIR COCHE

TRABAJO FIN DE GRADO



19 de julio de 2017

CHAMIT OUADI – X4788351G

UNIVERSIDAD DE ALICANTE

Tabla de contenido

[INTRODUCCIÓN 2](#_Toc488408630)

[**Ventajas de utilizar un framework como CodeIgniter** 4](#_Toc488408631)

[OBJETIVOS 5](#_Toc488408632)

[Requisitos funcionales 6](#_Toc488408633)

[Requisitos no funcionales Los requisitos no funcionales serán: 6](#_Toc488408634)

[TECNOLOGÍAS Y HERRAMIENTAS EMPLEADAS 7](#_Toc488408635)

[phpMyAdmin 7](#_Toc488408636)

[Codigneiter 7](#_Toc488408637)

[Github 7](#_Toc488408638)

[SISTEMAS DE INFORMACIÓN 8](#_Toc488408639)

[Esquema E-R 8](#_Toc488408640)

[Modelo Relacional 9](#_Toc488408641)

[Diccionario de Datos 13](#_Toc488408642)

# INTRODUCCIÓN

El proyecto consiste en la creación de una aplicación web, para compartir coche y aparcar en la universidad de Alicante, con el fin de facilitar a los usuarios, encontrar un aparcamiento libre, y para poder compartir coche.

Se ha utilizado el framework codigneiter, para llevar acabo el desarrollo. ¿Qué es codigneiter ? y porque ¿he desarrollado en este framework?

Codeigniter es un framework para el desarrollo de aplicaciones en php que utiliza el **MVC**. Permite a los programadores Web mejorar la forma de trabajar y hacerlo a mayor velocidad.

Al igual que cualquier framework está pensado para gente que tiene un dominio, al menos medio, del lenguaje de programación PHP. Siempre hay que controlar PHP con alto nivel para empezar a trabajar de forma eficiente con este framework (o cualquier otro).

¿Qué es MVC?

El Modelo Vista Controlador es un estilo de programación en el que la aplicación está dividida en 3 capas:

* Modelo: es dónde se procesa y obtienen los datos, la conexión con la bd.
* Vista: presenta los datos en pantalla, es donde va el código HTML.
* Controlador: controla los datos, dicho de forma rápida obtiene datos de un modelo, los procesa, y se los pasa a la vista.



**Ventajas de utilizar un framework como CodeIgniter**

* Las páginas se procesan más rapido, el núcleo de CodeIgniter es bastante ligero.
* Es sencillo de instalar, basta con subir los archivos al ftp y tocar un archivo de configuración para definir el acceso a la bd.
* Reutilización de código, desarrollo ágil.
* Existe abundante documentación en la red.
* Facilidad de edición del código ya creado.
* Facilidad para crear nuevos módulos, páginas o funcionalidades.
* Acceso a librerías públicas y clases. Entre otras, hay librerías para el login, paginador, calendarios, fechas,….
* Estandarización del código. Fundamental cuando hay que tocar código hecho por otra persona o cuando trabaja más de una persona en un mismo proyecto.
* URLs amigables con SEO. Hoy en día creo que nadie duda de la importancia del posicionamiento web.
* Separación de la lógica y arquitectura de la web, el MVC.
* CodeIgniter es bastante menos rígido que otros frameworks. Define una manera de trabajar, pero podemos seguirla o no(esto puede convertirse en un inconveniente también)
* Cualquier servidor que soporte PHP+MySQL sirve para CodeIgniter.
* CodeIgniter se encuentra bajo una licencia open source, es código libre.
* CodeIgniter usa una versión modificada del Patrón de Base de Datos [Active Record](http://codeigniter.com/user_guide/database/active_record.html). Este patrón permite obtener, insertar y actualizar información in tu base de datos con mínima codificación. Permite queries más seguras, ya que los valores son escapadas automáticamente por el sistema.

# OBJETIVOS

El proyecto nace con la idea de crear una aplicación web dedicada a todos los usuarios de la universidad de Alicante, que sirve para buscar aparcamiento de forma cómoda, ya que la universidad consta de varios parking, cada uno se identifica por una referencia y está ubicado en una zona, por lo que resulta difícil, encontrar un hueco en una zona predeterminada, según el interés de los usuarios, y como resultado, perder tiempo, gastos de energía etc. Y también poder compartir coche, para ir y venir de la universidad para aquellos usuarios que tienen dificultades de ir en bus universitario, o poder ahorrar gastos etc.

Algunas características de la aplicación:

* Gestión de los aparcamientos. cada usuario podrá ver los aparcamientos libres para el acceso.
* Gestión de los coches, cada usuario puede ver los coches disponibles para su ruta.
* Los usuarios tendrán acceso a los comentarios sobre cualquier coche.

# Requisitos funcionales

Los principales requisitos funcionales de la aplicación son:

* Cada usuario registrado es dado de alta, también existe el caso de dar de alta al coche si ese usuario es dueño de un coche en la cual comparte los viajes a la universidad.
* Cada usuario puede dar de alta a una notificación de compartir su coche cuando tiene plazas libres.
* Cada usuario puede buscar aparcamiento, poniendo la referencia del aparcamiento y ver las zonas libre.
* Cada usuario cuando sale del aparcamiento, debe actualizar la zona y ponerla como libre.
* Cuando usuario que se aparca, debe poner como ocupada la zona.
* Cada usuario puede comentar un coche cuando la comparte, así se lo permite a los demás usuarios conocer a las situaciones tanto del dueño como el coche .

# Requisitos no funcionales Los requisitos no funcionales serán:

* Fiabilidad. La aplicación debe ser fiable en los datos que guarde. Si en algún momento no se puede acceder a internet, la aplicación deberá proporcionar otra forma alternativa de acceder a esos datos (guardados localmente).
* Usabilidad. La usabilidad en la aplicación será esencial ya que la mayoría de los usuarios que utilizarán la aplicación no estarán relacionados con el entorno informático. La tasa de errores cometidos por el usuario deberá ser menor. El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final. El sistema debe poseer interfaces gráficas bien formadas.
* Eficiencia. El sistema debe ser capaz de procesar N transacciones por segundo. Toda funcionalidad del sistema y transacción de negocio debe responder al usuario en menor tiempo posible. El sistema debe ser capaz de operar adecuadamente con hasta N usuarios con sesiones concurrentes. Los datos modificados en la base de datos deben ser actualizados para todos los usuarios que acceden en menor tiempo.
* Dependibilidad. El sistema debe tener una disponibilidad del 99,99% de las veces en que un usuario intente accederlo. El tiempo para iniciar o reiniciar el sistema no podrá ser mayor. La tasa de tiempos de falla del sistema no podrá ser mayor. El promedio de duración de fallas no podrá ser mayor. La probabilidad de falla del Sistema no podrá ser mayor.

# TECNOLOGÍAS Y HERRAMIENTAS EMPLEADAS

Para llevar a cabo el proyecto he utilizado diversas tecnologías y herramientas seleccionadas para cumplir con ciertos propósitos concretos.

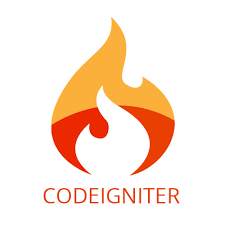
Para la realización de la aplicación, he utilizado el lenguaje de programación PHP en el lado Backend y el framework bootstrap, HTML, CSS, JAVASCRIPT en lado de FrontEnd.

Por último, para realizar la base de datos he utilizado SQL. La herramienta utilizada para manejar la base de datos y los procedimientos es phpMyAdmin.

## phpMyAdmin

phpMyAdmin es la herramienta que he utilizado para el manejo de la base de datos MySQL está alojado en el servidor junto con la página web. La versión de MySQL es la 10.1.25-MariaDB - mariadb.org binary distribution y la versión de phpMyAdmin utilizada es la 4.7.0 y está alojada sobre un servidor Apache/2.4.26 (Win32) OpenSSL/1.0.2l PHP/7.1.7.

## Codigneiter

**CodeIgniter** es un entorno de desarrollo abierto que permite crear webs dinámicas con [PHP](https://www.ecured.cu/PHP). Su principal objetivo es ayudar a que los desarrolladores, puedan realizar proyectos mucho más rápido que creando toda la estructura desde cero, proveyendo un rico juego de librerías para tareas comúnmente necesarias, así como una interface simple y estructura lógica para acceder a esas librerías.

## Github

GitHub es una plataforma de **desarrollo colaborativo de software** para alojar proyectos utilizando el sistema de control de versiones [Git](http://es.wikipedia.org/wiki/Git).

Sirve para alojar un repositorio de código y brinda **herramientas** muy útiles para el **trabajo en equipo**, dentro de un proyecto.

Además de todo eso, se ofrecen varias herramientas útiles para el **trabajo en equipo**

* Una wiki para el mantenimiento de las distintas versiones de las páginas.
* Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
* Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
* Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio

## Balsamiq

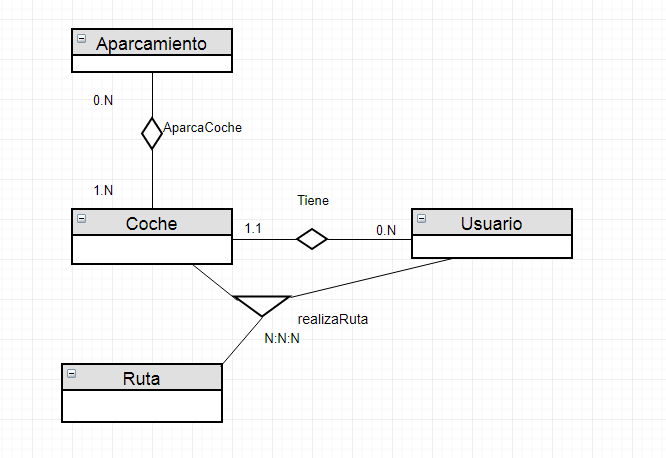
He utilizado Balsamiq Mockups (versión escritorio). Balsamiq es la herramienta que he utilizado para realizar los mockups1 de las pantallas que va a tener mi aplicación. Con el objetivo claro de conseguir un enfoque sobre el diseño básico en etapas más tempranas, es decir, antes de comenzar a diseñar las pantallas ya sé cuál es el diseño que van a tener.

# SISTEMAS DE INFORMACIÓN

Para la realización del proyecto he creado una base de datos mysql utilizando phpmyadmin. La principal función de la base de datos es que sirva como mecanismo de persistencia de datos. Para poder almacenar y consultar toda la información referente a la aplicación independientemente de que estemos utilizando la aplicación Android, iOS o la página web.

## Esquema E-R

Para visualizar la base de datos de una forma más clara he tomado una captura un esquema Entidad-Relación que representa las entidades de la base de datos así como las relaciones entre las distintas entidades



## 

## Modelo Relacional

El modelo relacional nos aporta de una forma sencilla e intuitiva la representación de la base de datos.

**Usuario**

{

- Correo Varchar

- Nombre Varchar

- Apellidos Varchar

- Edad Int

- Contraseña Varchar

- Telefono Varchar

- Salt Varchar

- Foto Varchar

- Universidad Varchar

- Tipo Int

C.P.: Nif

V.N.N.: Correo, Nombre, Apellidos, Edad, Contraseña, Teléfono, Salt, foto, Universidad, tipo.

}

**Coche**

{

- Matricula Varchar

- Modelo Varchar

- Color Varchar

- Acientos Int

- Usuario Varchar

- Marca Varchar

- Categoria Varchar

C.P.: Matricula

C.aj.: Usuario -> Usuario

VNN.: Modelo, Color, Asientos, Usuario, Marcia, Categoría.

}

**Ruta**

{

- Id int

- Plazas Int

- PlazasOcupadas Int

- Origen Varchar

- Destino Varchar

- Detalles Varchar

- Precio Varchar

- FechaPublicacion Varchar

C.P.: Id

V.N.N.: Plazas, PlazasOcupadas, Origen, Destino, Precio.

}

**Aparcamiento**

{

- Id Int

- Plazas Int

- Espacio Int

- Universidad Varchar

C.P.: Id

C.aj.: Universidad -> Universidad

V.N.N.: Plazas, Espacio, universidad.

}

**Realiza\_Ruta**

{

- Usuario Varchar

- Coche Varchas

- Ruta

C.P.: (Usuario, Coche, Ruta)

C.aj.: Usuario -> Usuario

C.aj.: Coche -> Coche

C.aj.: Ruta -> Ruta

}

**Comenta**

{

- UsuarioA Varchar

- UsuarioB Varchar

- Comentario Varchar

**C.P.:** (UsuarioA, UsuarioB)

**C. Ajena:** UsuarioA -> Usuario

**C. Ajena:** UsuarioB -> UsuarioB

**V.N.N.:** Comentario

}

**Aparca**

{

- Coche Varchar

- Aparcamiento Varchar

C.P.: (coche, aparcamiento)

C. Ajena: coche -> coche

C. Ajena: aparcamiento -> aparcamiento

}

## Diccionario de Datos

Otra forma de representar la información que contienen las tablas es a través del diccionario de datos, por ejemplo el diccionario de datos de la tabla artículo es la siguiente:

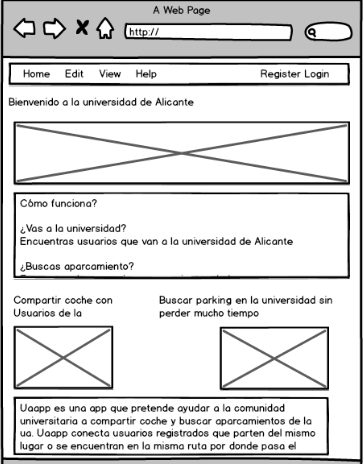
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
|  |  |  |  |  |
| Correo | Varchar | No |  | Correo del usuario |
| Nombre | Varchar | No |  | Nombre del usuario |
| Apellido | Varchar | Sí | Null | Apellido del usuario |
| Edad | Int | Sí | Null | Edad del usuario |
| Contraseña | Varchar | No |  | Contraseña del usuario para login |
| Telefono | Varchar | Sí | Null | Telefono del usuario |
| Salt | Varchar | No |  | Salt generada para crear la contraseña junto con el hash |
| Foto | Varchar | Sí | Null | Foto del usuario |
| Universidad | Varchar | No |  | Universidad donde va el usuario |
| Tipo | Int | No | 0 | 1 indica a un usuario como admin y 0 indica un usuario normal |

Donde podemos observar el nombre de la columnas, el tipo de datos, si es posible que el valor de los campos sea nulo, el valor por defecto que tendrán los campos si no se establecen y el comentario que sirve como nota aclaratoria.

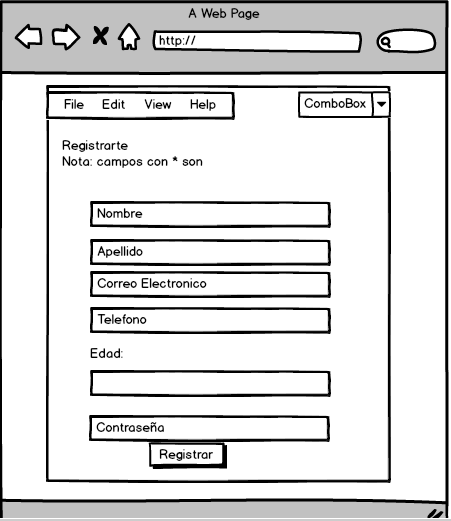
# Mockups

Antes de comenzar a implementar el proyecto, he creado los mockups de la aplicación. Consiguiendo así definir el diseño básico que tendrán las pantallas en fases tempranas del desarrollo .

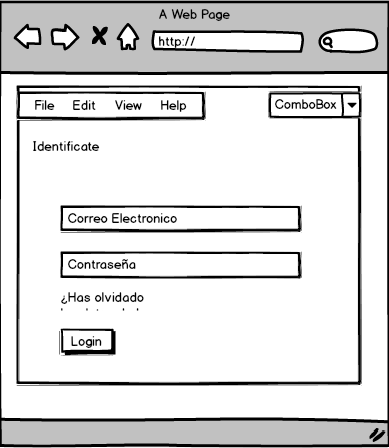
Por ejemplo vamos a ver el mockup de la pantalla principal de la aplicación que es la siguiente:

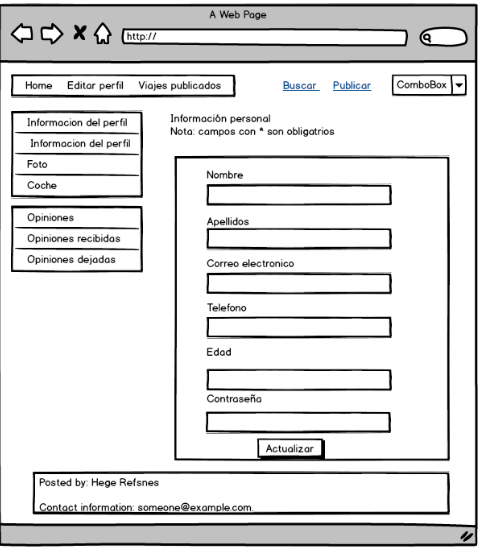


Pantalla de registro

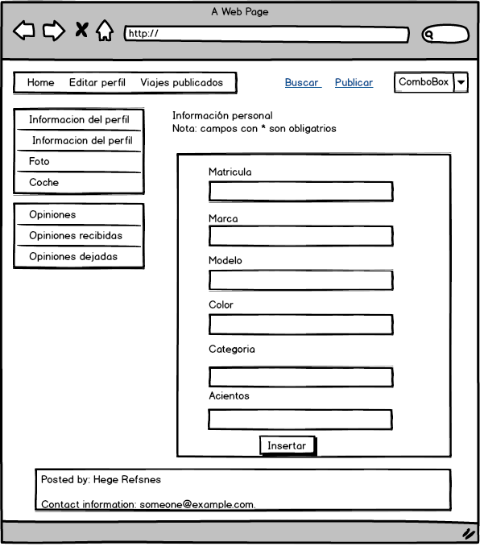


Pantalla Login

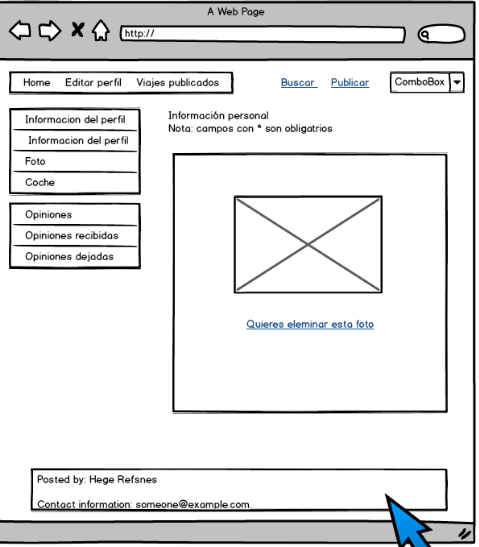


Editar Perfil

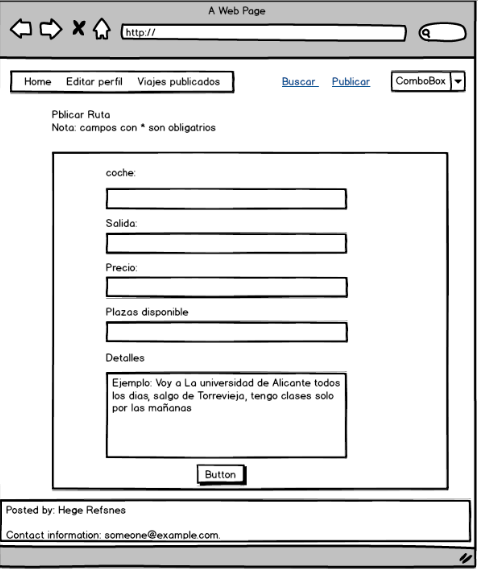
Pantalla insertar Coche



Pantalla insertar Foto usuario



Pantalla publicar ruta



Pantalla buscar ruta

