

Aplicación web para buscar aparcamiento y compartir coche

Grado en Ingeniería Informática



Trabajo Fin de Grado

Autor:

Ouadi Chamit

Tutor/es:

Estela Saquete Boro

Septiembre 2017



Tabla de contenido

[INTRODUCCION 3](#_Toc488779152)

[RESUMEN 3](#_Toc488779153)

[MOTIVACION Y ESTUDIO DE MERCADO 4](#_Toc488779154)

[ BlaBlaCar 4](#_Toc488779155)

[ Uber 4](#_Toc488779156)

[ Amovens 4](#_Toc488779157)

[ Carpooling 4](#_Toc488779158)

[ Wazypark 5](#_Toc488779159)

[ Parkopedia 5](#_Toc488779160)

[ Aparca&Go 5](#_Toc488779161)

[HERRAMIENTAS, TECNOLOGIAS Y LENGUAJES DE PROGRAMACION EMPLEADOS 6](#_Toc488779162)

[phpMyAdmin 6](#_Toc488779163)

[Github 6](#_Toc488779164)

[Balsamiq 7](#_Toc488779165)

[Codigneiter 7](#_Toc488779166)

[Ventajas de utilizar un framework como CodeIgniter 7](#_Toc488779167)

[REQUESITOS FUNCIONALES Y NO FUNCIONALES 9](#_Toc488779168)

[Requisitos funcionales 9](#_Toc488779169)

[Requisitos no funcionales Los requisitos no funcionales serán: 9](#_Toc488779170)

[DISEÑO 10](#_Toc488779171)

[Mockups 10](#_Toc488779172)

[DISEÑO FINAL 18](#_Toc488779173)

[ARQUETECTURA 19](#_Toc488779174)

[IMPLEMENTACION 20](#_Toc488779175)

[- Front end 20](#_Toc488779176)

[- Back end 20](#_Toc488779177)

[Base de datos 21](#_Toc488779178)

[Esquema conceptual 21](#_Toc488779179)

[Modelo Relacional 22](#_Toc488779180)

[Diccionario de Datos 26](#_Toc488779181)

# INTRODUCCION

## RESUMEN

El proyecto consiste en la creación de una aplicación web, que ayuda a los usuarios a publicar anuncios de coches para compartir coche o buscar aparcamiento en la universidad de Alicante, con el fin de facilitar encontrar un aparcamiento libre, o poder compartir coche para ir y venir de la universidad.

El proyecto nace con la idea de crear una aplicación web dedicada a todos los usuarios de la universidad de Alicante, en concreto a aquellos usuarios que no tienen medios para venir a la universidad o los que no tienen suficiente gasto de los viajes y ayudar a los propietarios de los coches de poder ahorrar gastos. También sirve para buscar aparcamiento de forma cómoda, ya que la universidad consta de varios parkings, 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.

## MOTIVACION Y ESTUDIO DE MERCADO

Siendo alumno de la universidad, siento esa necesidad, y la de los demás usuarios, por lo que me motiva a desarrollar esa aplicación y poder resolver uno de los problemas que afrontan los usuarios de la universidad que se encuentran a día hoy, y poder proveer una herramienta, para poner fin a esa necesidad o por lo menos paliar el problema.

Al analizar la competencia nos hemos dado cuenta de que no existe una aplicación en concreto con el mismo objetivo que gestiona tanto el aparcamiento como compartir coche a la vez. No obstante, hay algunas aplicaciones encontradas que coinciden en algún apartado/sección, y que nos ha servido para mejorar nuestra idea.

Las aplicaciones buscadas han sido, por ejemplo:

* **BlaBlaCar**

Es una red social de viajes de largas y cortas distancias en coche compartido. Permite a los usuarios ponerse en contacto cuando quieren realizar un trayecto común y coinciden para hacerlo el mismo día, así como permitir a los conductores ahorrar el coste del litro de gasolina por cada pasajero.

Los usuarios comparten los gastos del viaje sin que el conductor tenga beneficios, para eso la red social recomienda a los usuarios por cada viaje la aportación de 0.06 euros por cada kilómetro cada uno, y limita la aportación máxima que pueden solicitar los conductores de tal manera que no se superen estos gastos.

* **Uber**

Uber es una aplicación para compartir viajes rápidos y fiables en cuestión de minutos, de día o de noche. No hace falta aparcar ni esperar taxis o autobuses.

Mediante la cual, comunicas tu ubicación, el destino, el tipo de vehículo y el tiempo de espera estimado. Tras el viaje, lo puedes pagar con tarjeta, efectivo o con tu teléfono móvil y el 20% se lo queda la propia empresa a modo de comisión

* **Amovens**

es similar a Uber en el concepto urbano de la aplicación, pero el objetivo de esta herramienta es encontrar a gente que realice el mismo trayecto que tú a diario y poder crear vínculos de transporte entre las personas, para ir al trabajo o a la universidad, sin que la empresa se lleve ningún tipo de comisión ni facilite modos de pago.

* **Carpooling**

es una aplicación similar a BlaBlaCar. Sirve exactamente para lo mismo, consiste en compartir nuestros viajes en auto con otras personas de forma cotidiana. El carpooling es una práctica popular en Estados Unidos y Europa, donde se realiza de manera organizada para lograr aumentar el número de viajes compartidos y que estos sean concretados con otras personas además de nuestros vecinos y amigos.

Sin embargo ofrece elementos como la elección de viajes solo entre mujeres o fijar un radio de búsqueda a tu alrededor para empezar el viaje.

* [**Wazypark**](http://www.wazypark.com/)

Se trata de una aplicación de inteligencia colectiva en la que mediante geolocalización del móvil en un mapa, podemos saber dónde hay una plaza libre en las calles de nuestra ciudad para aparcar. Wazypark mejora a medida que aumenta el número de usuarios, que deben registrarse ellos mismos tanto como a su vehículo. Cada usuario avisa desde su móvil cuando deja un hueco libre en un aparcamiento, de modo que otro usuario cercano nos pueda detectar y aparcar en la plaza. Disponible para Android e iOS.

* [**Parkopedia**](http://www.parkopedia.es/)

Tiene el mismo espíritu que Wazypark pero con elementos de Foursquare, en la que los usuarios dejan información de zonas de aparcamientos libres en plena calle o de aparcamientos públicos o privados, con los precios por hora o día de cada parking. Su interés reside en que hay información de numerosas ciudades. Tanto en Android como en iOS.

* [**Aparca&Go**](http://www.aparcandgo.com/)

Disponible solo para Android como aplicación, pero se puede usar desde la web. Se ha especializado en la gestión de aparcamientos mediante acuerdos por los que el usuario puede ganar un descuento en su ticket. La aplicación gestiona los pagos -hay que dejar los datos de una tarjeta-, por lo que las molestias son mínimas; además se puede reservar la plaza por adelantado. Un valor fuerte de esta aplicación es la creación de aparcamientos propios y alternativos a los de aeropuertos y estaciones de tren, pero con precios reducidos. De modo que ellos gestionan el transporte hasta el aeropuerto desde sus aparcamientos y de vuelta.

# HERRAMIENTAS, TECNOLOGIAS Y LENGUAJES DE PROGRAMACION EMPLEADOS

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 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.

## 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.

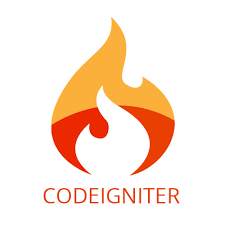
Además de todo eso, se ofrecen varias herramientas útiles para el **trabajo en equipo**

* 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.

## Codigneiter

**CodeIgniter** es un entorno de desarrollo abierto que utiliza el **MVC** y 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.

**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.

# REQUESITOS FUNCIONALES Y NO FUNCIONALES

## 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:

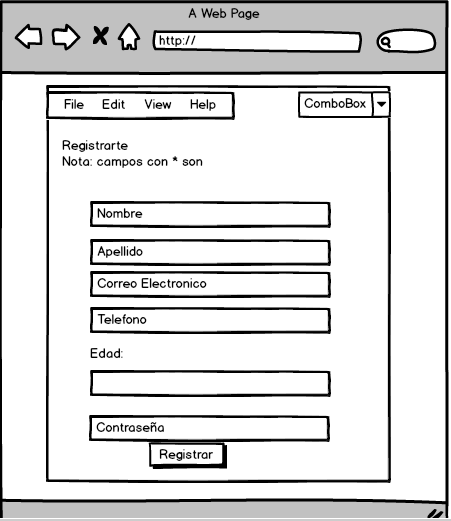
* Fiabilidad. La aplicación debe ser fiable en los datos que guarde. Si en algún momento no se puede acceder a internet, la aplicación deberá proporcionar otra forma alternativa de acceder a esos datos (guardados localmente).
* Usabilidad. La usabilidad en la aplicación será esencial ya que la mayoría 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.

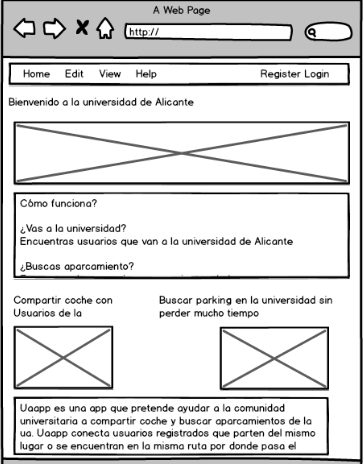
# DISEÑO

## Mockups

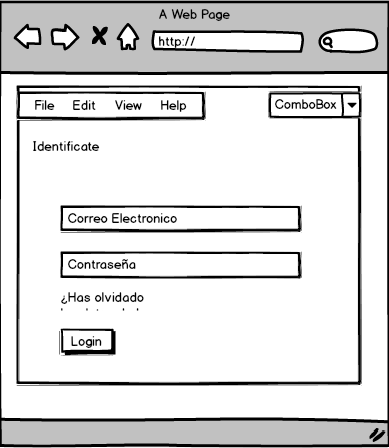
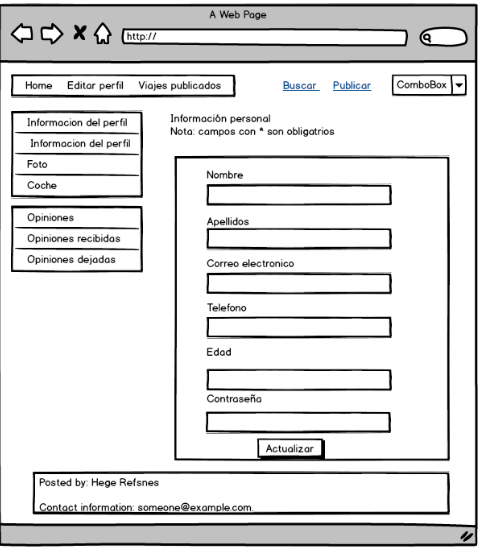
Antes de comenzar a implementar el proyecto, he creado los mockups de la aplicación. Consiguiendo así definir el diseño básico que tendrán las pantallas en fases tempranas del desarrollo .

Por ejemplo vamos a ver el mockup de la pantallas de la aplicación.

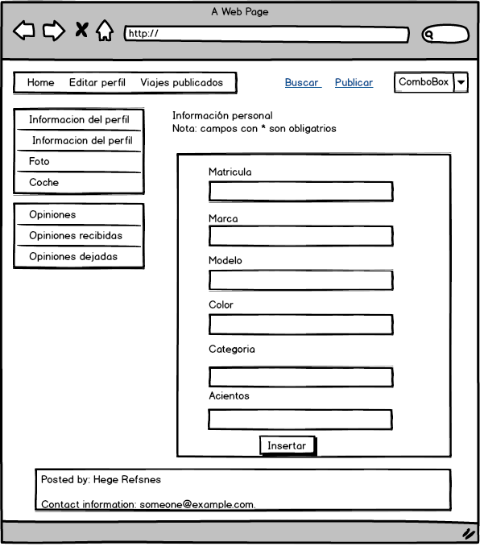
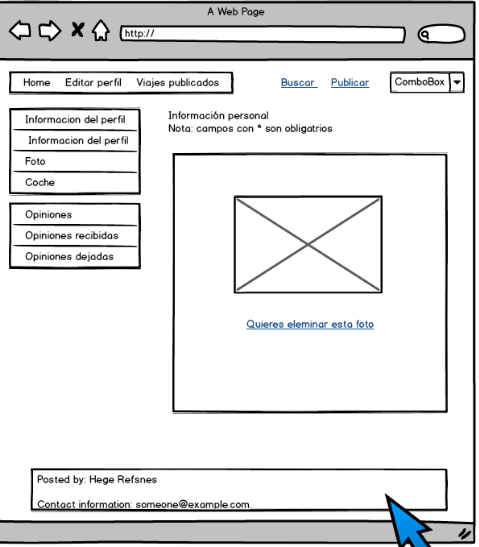


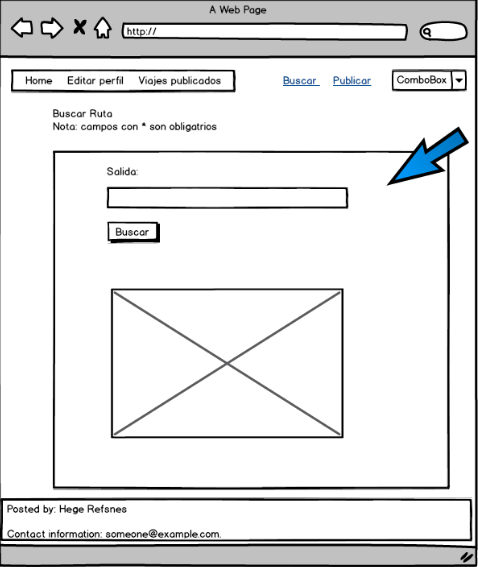


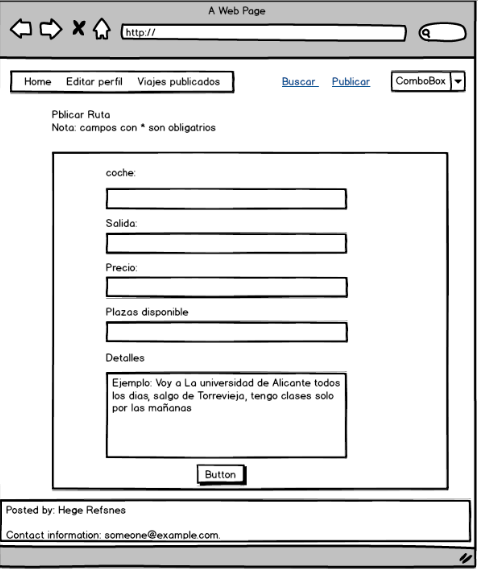
Pantalla Login/Actualizar datos usuario



Pantalla Coche/Foto usuario



Pantalla Publicar anuncio/Buscar coche para compartir



## DISEÑO FINAL

# ARQUETECTURA

Modelo vista controlador

¿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.



# IMPLEMENTACION

## - Front end

Html/css

Bootstrap

Javascript/jsquery

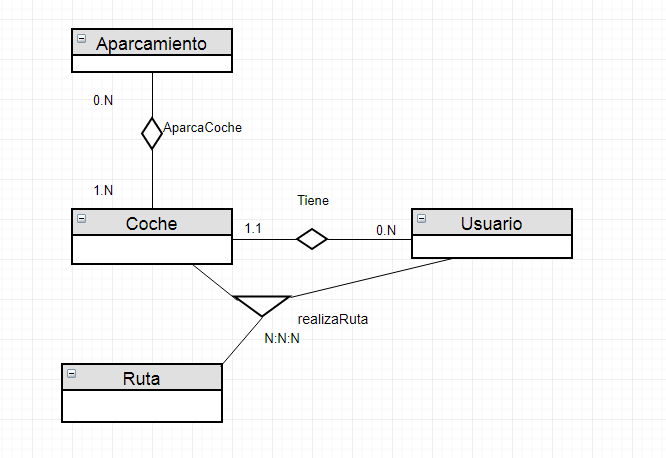
Peticiones Ajax/json

## - Back end

## Base de datos

### Esquema conceptual

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.

SEGURIDAD

APIS UTILIZADAS