UNIVERSIDAD POLITÉNICA DE CARTAGENA

Desarrollo de una aplicación para el transporte colaborativo basada en tecnologías de blockchain

Trabajo Fin de Grado

Autor: Martin Wenceslao Andreo Sánchez
Director: Esteban Egea López

Grado en Ingeniería en Telemática
Curso: 2018/2019

1. INDICE

2. Introducción

Mirando 10 o 20 años atrás, podemos ver que el transporte ha crecido según han pasado los años, debido tanto al aumento de la población, como a las necesidades de las personas, por cuestiones de trabajo, familiares u ocio. De forma paralela, las personas siempre han tenido tendencia al ahorro de dinero, en este caso, en los viajes. La manera más común ha sido siempre viajar con una persona conocida, aprovechando su viaje, de manera que no tenga que contratar una compañía externa que haga de intermediaria a la hora de viajar.

A día de hoy, esto ya es posible debido a aplicaciones como BlaBlaCar o Amovens. Estas aplicaciones permiten al usuario viajar, sin conocer al conductor o pasajeros, al destino que se desee. El viaje, además, sale más barato al compartir los costes del mismo. Sin embargo, estas compañías se llevan una comisión por cada viaje que sea contratado, impidiendo que el coste del viaje sea óptimo.

En este punto habría que plantearse si realmente a los usuarios les interesaría realizar los viajes a través de aplicaciones como BlaBlaCar, o realizarlos a través de aplicaciones descentralizadas, donde no se necesitase un intermediario, pudiendo abaratar el precio del viaje, y no solamente eso, si no dando una mayor fiabilidad a la hora de la gestión de viajes.

La idea principal de este proyecto es realizar una aplicación web para el transporte colaborativo basada en Blockchain.

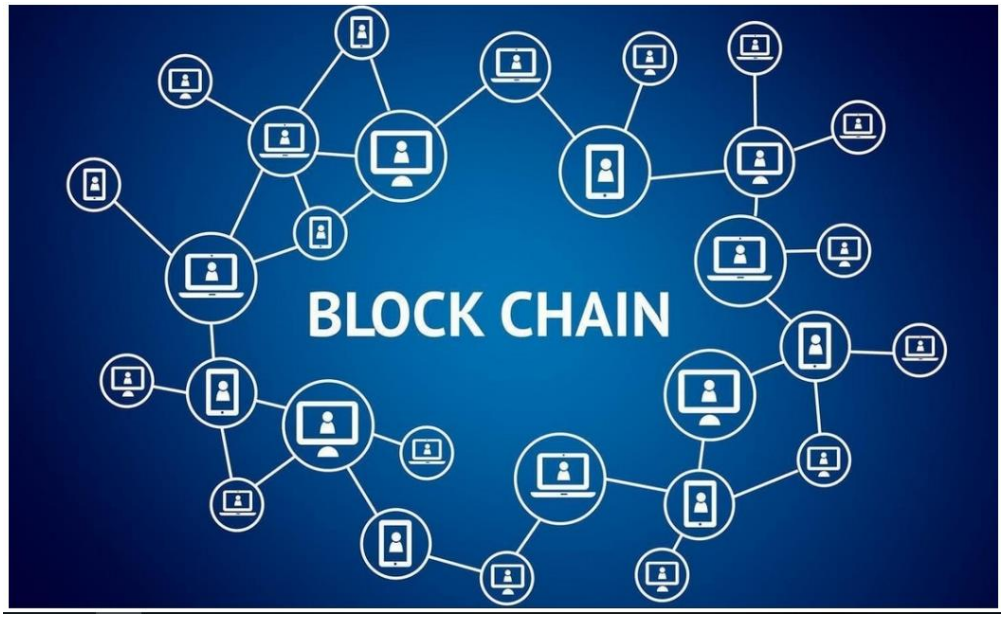
La ventaja principal que tendría esta aplicación es la eliminación del intermediario, permitiendo abaratar el precio de los viajes, apareciendo en este punto el concepto de economía colaborativa.

¿Qué es la economía colaborativa?

La expresión economía colaborativa se basa en utilizar la tecnología en nuestro propio provecho. Gracias a ella, los usuarios se pueden organizar entre ellos para lograr un beneficio gracias por ejemplo a la optimización de recursos**,** es decir, dar salida a bienes que antes no se utilizaban o que no tenían un uso al 100%. Esto permite una mejor oferta para el consumidor final, ya que el consumidor se encuentra con una oferta más amplia entre lo que ofrecen los comercios tradicionales y lo que ofrece la economía colaborativa.

Hoy en día, es posible llevar a cabo la economía colaborativa entre los usuarios gracias a la tecnología de Blockchain.

2.1 ¿QUÉ ES BLOCKCHAIN?



Blockchain es una estructura de datos donde la información se agrupa en bloques a los que se les añade información de bloques anteriores en una línea temporal, consiguiendo que la información de un bloque solo pueda ser editada modificando los siguientes bloques. Esto permite que la estructura de datos *blockchain* pueda hacer que contenga un historial seguro de información.

Blockchain es adecuado cuando se requiere almacenar datos ordenados en el tiempo de forma creciente, sin que puedan modificarse, pretendiendo que la confianza se distribuya en lugar de residir en una entidad certificadora.

En esencia, una red blockchain es solo una base de datos que permite leer y escribir registros, no se puede modificar nada de ella. Todos los registros que se guardan en ella están vinculados entre sí, de manera que sea imposible incluir algo incoherente con el resto de registros incluidos.

A efectos prácticos, permite soportar y garantizar la seguridad de dinero digital.

Las aplicaciones de este sistema de codificación van desde las transacciones financieras o internet de las cosas, pasando por la energía o las administraciones públicas.

2.2 POSIBLES APLICACIONES DE BLOCKCHAIN

Se pueden definir distintas aplicaciones que tiene Blockchain, según el ámbito en que se utilice.

- El primer uso que se le ha dado a la cadena de bloques ha sido en el **sector financiero**, siendo la banca el primer ejemplo de una actividad que, según parece, puede ser desintermediada ya que no sería necesaria.

- En el campo de las **criptomonedas** se usa como notario público no modificable de todo el sistema de transacciones, para evitar el problema de que una moneda se pueda gastar dos veces en cualquier transacción.

- En el campo de las **bases de datos de registro de nombres**, se usa para tener un sistema de registro de nombres con el fin de que un nombre solo pueda ser utilizado para identificar el objeto que lo tiene registrado.

- Se usa en distintos tipos de **transacciones**, haciéndolas más seguras, baratas y rastreables. Por ejemplo puede usarse para sistemas de pago, transacciones bancarias, préstamos, o envío de remesas.

- Interviene en la creación de acuerdos de **smart-contracts** entre pares de usuarios (nodos). El objetivo es permitir a una red de pares administrar sus propios contratos inteligentes creados por los usuarios. Primero se escribe un contrato mediante un código y se sube a Blockchain mediante una transacción.

Estas son sólo unas de las pocas aplicaciones donde puede usarse Blockchain.

En un futuro, seguirá creciendo en distintos ámbitos, como las entidades públicas, que hasta ahora, son incapaces de dar total respuesta a la demanda de la sociedad actual. Algunos países ya están explorando esta tecnología en el **registro de títulos de la propiedad, vehículos, licencias, subvenciones o registros sanitarios**. También el **voto electrónico** está esperando la oportunidad de ser viable gracias a su seguridad, cosa que ahorraría mucho tiempo a los ciudadanos, y dinero a la administración.

En definitiva, las aplicaciones de Blockchain pueden ser innumerables, e irán creciendo conforme se vaya ahondando en esta tecnología.

2.3 VENTAJAS DE LA APLICACIÓN

Las ventajas del uso de la economía colaborativa para este proyecto son muchas. Estas son algunas de las principales.

- **Ahorro de dinero.** El punto fuerte de esta aplicación es **eliminar el intermediario**, que siempre se lleva una comisión por cada viaje que se contrate al usar este servicio. El uso de Blockchain permite la interacción directa entre usuarios. Los conductores, pueden **reducir el coste** de su viaje, ya que llenando las plazas no deben asumir ellos solos el total del viaje, si no que el coste será compartido. También los pasajeros pueden ahorrar dinero a la hora de realizar los viajes en lugar de usar otro tipo de transporte.

- **Cuidado del medio ambiente.** El transporte colectivo contribuye a reducir la contaminación del medio ambiente. Es mejor un coche que lleve a cuatro personas que cuatro coches llevando a una persona. Además, se evita el uso abusivo de coches en los viajes, pudiendo disminuir los atascos.

- **El factor humano.** La aplicación permite aumentar las relaciones sociales, el diálogo y la solidaridad entre los usuarios que la utilicen.

- Interviene también la **fiabilidad**, ya que al estar basada en *Smart-contracts*, habrá que cumplir una serie de normas ya definidas que no podrán romperse, respetando también la confidencialidad los datos.

- Además, al estar descentralizada la información hace sumamente difícil los ataques maliciosos, aumentando la **seguridad**. También interviene la **privacidad**, debido a la encriptación de toda la información.

- **Transparencia**. Cualquier modificación puede ser trazada y vista públicamente.

A pesar de todas estas ventajas, finalmente, como se explica más adelante, el proyecto se acabó desarrollando como una aplicación web convencional.

3. Tecnologías utilizadas

Podemos diferenciar en dos las tecnologías utilizadas en este proyecto, las tecnologías de *Blockchain* y las *tecnologías Web.*

Las tecnologías/herramientas que se han utilizado con Blockchain principalmente han sido ***ethereum***, ***solidity*** y ***web3*****.**

3.1 Blockchain

3.1.1 ETHEREUM

- El propósito inicial que tiene *Ethereum* es descentralizar la web.

Tiene una criptomoneda propia que se llama ***ether***, y permite intercambiar *ether* entre cuentas diferentes, a cambio de, por ejemplo, usar servicios.

Una de las principales ventajas que tiene *Ethereum,* como se dijo antes, es la de **eliminar el intermediario**, permitiendo a los usuarios interactuar entre ellos directamente.

Además, la **información personal** de los usuarios no corre riesgo en esta plataforma, ya que toda la información privada es confidencial.

En *Ethereum*, los desarrolladores pueden escribir la lógica de negocio y acuerdos en forma de *contratos inteligentes*, que se ejecutan automáticamente cuando sus condiciones son satisfechas por ambas partes. Estos contratos pueden almacenar datos, enviar y recibir transacciones, e incluso interactuar con otros contratos. Los programas que realizan *contratos inteligentes* son escritos en lenguajes de programación de alto nivel como ***Solidity*****.**

Ethereum tiene la capacidad de **reducir los costes**, asegurando la confianza entre la interacción de los *Smart-contracts*.

3.1.2 SMART-CONTRACTS

Un **Smart-contract** es un programa informático que facilita, hace cumplir y ejecuta acuerdos registrados entre dos o más partes. Ayudan en la definición y negociación de acuerdos obligados a cumplir una serie de condiciones.

Un contrato inteligente vive en un sistema **no controlado** por ninguna de las partes. Este sistema ejecuta un contrato que interactúa con activos reales, de manera autónoma y automática, **sin intermediarios** ni mediadores. Evitan la interpretación, al no ser verbal ni escrito en los lenguajes que hablamos. Además, normalmente se componen de una interfaz de usuario.

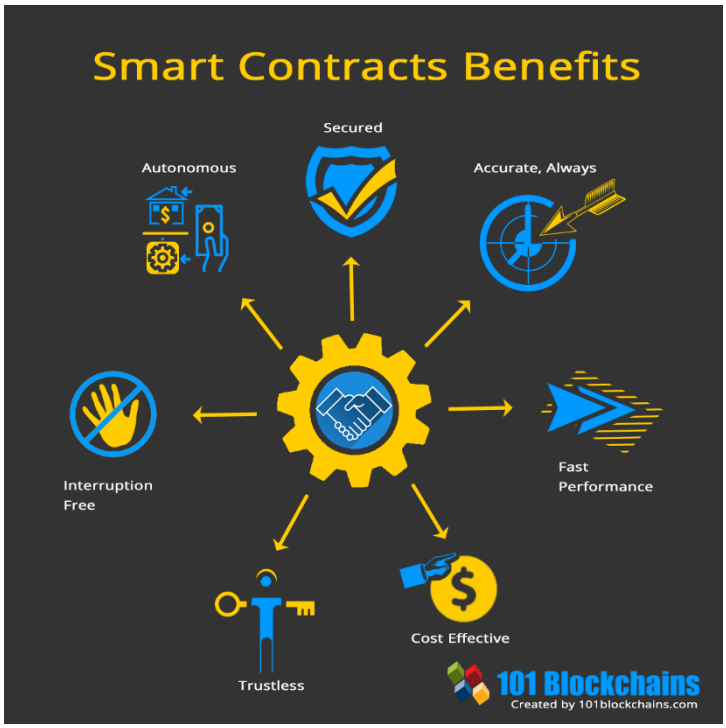
Reducen costes de transacción asociados a la contratación. Se pueden realizar en cualquier transacción que requiera un acuerdo registrado entre partes, como, por ejemplo, la contratación de productos financieros o de seguros, operaciones de compra y venta instrumentos financieros.

Tiene validez, sin depender de autoridades, debido a un **código visible** para todos e **inmutable** por existir sobre *Blockchain*, la cual le da ese carácter descentralizado y transparente.

Para garantizar el sostenimiento de todo el ecosistema de Dapps (Aplicaciones descentralizadas), cualquier ejecución de órdenes en *Blockchain* conlleva gastar criptomoneda. Este coste, denominado **Gas**, depende precisamente de qué operaciones se pretenden ejecutar en dicho Smart-contract, y sirve de incentivo económico para evitar la saturación la red, manteniendo los nodos necesarios para asegurar un servicio permanente y seguro.

Al carecer de puntos centrales, y estar protegidos con **criptografía**, las aplicaciones están a salvo de actividades de carácter fraudulento.

Sin duda, los *Smart-contracts* **aumentarán la velocidad** de la ejecución de las transacciones, lo que se traducirá en la posibilidad de cerrar un mayor volumen de acuerdos con menor riesgo al cumplimiento.



3.1.3 SOLIDITY - WEB3

Solidity es un lenguaje de programación de alto nivel orientado a objetos que permite implementar los *contratos inteligentes* en el EVM.

Este lenguaje está diseñado y compilado en código de bytes para **crear y desarrollar contratos inteligentes** que se ejecuten en la Máquina Virtual Ethereum**.**

Mediante Solidity, los desarrolladores pueden escribir aplicaciones descentralizadas que implementen **automatizaciones en los negocios** a través de los contratos inteligentes, dejando un **registro irrefutable** y autorizado de las transacciones.

En palabras menos técnicas, Solidity sirve para la creación de *Smart-contracts* que permiten que muchas de las partes de un negocio funcionen perfectamente por sí solas y además se lleve un registro de las mismas.

Web3 es una biblioteca ligera escrita en Java 8. Está en una capa superior, ya que necesita integrarse con un cliente o nodo de la red Ethereum.

Web3.js es una API en Javascript compatible con Ethereum. Existe un objeto **web3,** que permite que la aplicación funcione en ***Ethereum*****,** proporcionado por la biblioteca **web3.js**.

3.2 Tecnologías WEB

Finalmente, al acabar realizando el proyecto como una aplicación web convencional, las tecnologías utilizadas para la parte Web han sido **HTML, CSS, PHP, SQL y JS**.

HTML es un lenguaje de marcas. Con él, se hizo un primer diseño estático de la web. Llamamos página estática a aquella cuyos contenidos permanecen siempre igual, mientras que llamamos páginas dinámicas a aquellas cuyo contenido no es el mismo siempre.

HTML permite crear los elementos necesarios con etiquetas, y ponerlas por la página web a un primer nivel, para, posteriormente, con CSS modificar más fácilmente. Se hace uso además de algunas bibliotecas de Bootstrap a la hora de utilizar los fondos y dividir páginas.

CSS permite dar estilos a los elementos de HTML, como, por ejemplo, posicionar elementos, darles color o tamaño. Con esto se evita darle a la página estilos en línea (en el mismo HTML) y además, no replicar código para componentes que sean iguales o muy parecidos.

Java Script permite dar dinamismo a las páginas web. En este proyecto, se utiliza para dar funcionalidad básica a los botones a la hora de redirigir a otras páginas. También interviene en la validación de los campos que debe rellenar el usuario al registrarse o publicar viajes, para no dejarlos vacíos.

PHP es un lenguaje del lado del servidor, pudiendo ser incrustado en HTML.

Se utiliza para generar páginas web dinámicas. Además, permite la conexión con la base de datos, almacenando consultas para obtener, insertar, eliminar o modificar datos de la BBDD, todo ello a través de SQL.

También se utiliza para guardar en sesión los datos que necesitan ser enviados de un documento PHP a otro.

SQL permite realizar las consultas, inserciones, actualizaciones y borrados necesarios en la base de datos

4. Desarrollo del proyecto

4.1 Desarrollo Blockchain

El comienzo de este proyecto fue con la parte *back-end* en S*olidity*.

Para la creación de los *Smart-contracts* fue necesaria la herramienta de **remix** (https://remix.ethereum.org). Es un IDE que sirve como compilador online, que permite desarrollar los Smart-contracts, y posteriormente implementarlos y ejecutarlos.

Se necesita un método en el que los usuarios **oferten rutas** (con origen, destino, fecha, precio [gas/dinero] y plazas disponibles). Cuando un usuario publica una ruta, debe guardarse como un contrato en Blockchain, permitiendo que luego pueda ser firmado por otro usuario (el pasajero).

Tras la creación de un primer archivo solidity, se comenzó con el desarrollo de la función de *Ruta*, donde se le pasaba como parámetros origen, destino, precio, plazas, fecha y hora. Además, la función debía tener el atributo '**payable'**, para permitir las transacciones.

Es necesario tambien, el modificador **onlyOwner**, donde se asigna que el conductor sea el 'msg.sender', es decir, el usuario que esté publicando el viaje y que esté registrado en Blockchain.

El resto de usuarios pueden **buscar rutas y solicitar plaza/s** de esa/s ruta/s siempre que estén disponibles.

Para ello era necesaria la función de **comprar viaje**, (tambien con el atributo payable). Es necesario comprobar que haya plazas disponibles, y se puede hacer la transferencia directamente de una cuenta a otra. Además se debe restar una plaza a la función de Ruta. Para mostrar los viajes disponibles es necesario hacer una comprobación para que coincida el origen, destino, y fecha del viaje.

Finalmente, la idea sobre la aplicación basada en Blockchain se acabó desarrollando como una **aplicación web convencional**, debido a los distintos problemas en el desarrollo con la lógica del contrato a la hora de compilar desde Remix, y también, en la invocación de *Ethereum* desde otros lenguajes con web3. La idea era invocar Solidity en los scripts del servidor.

4.2 DESARROLLO FRONT-END

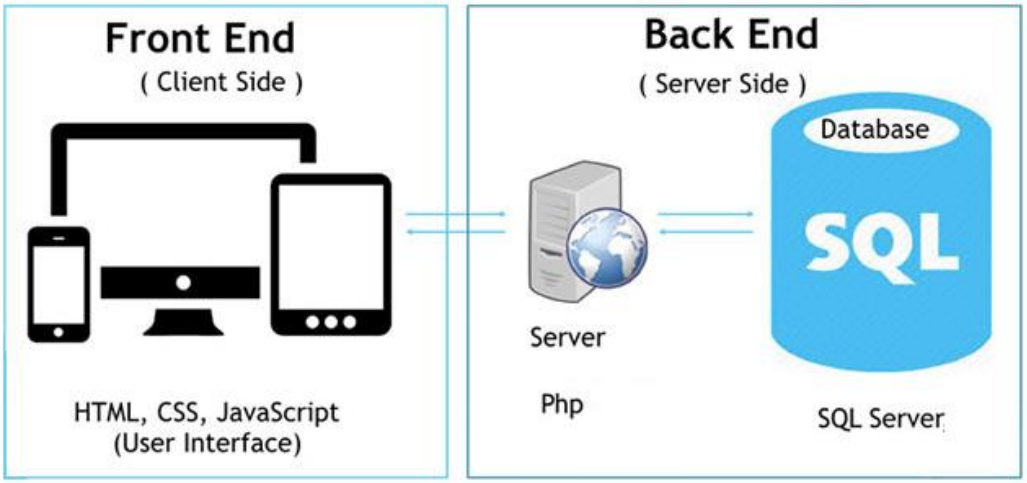
Hubo que comenzar el desarrollo entonces de la interfaz web, mediante **HTML** y **CSS**, para más tarde, mediante **PHP** y **SQL**, darle funcionalidad a la misma con la BBDD.

El comienzo fue con el desarrollo de distintas páginas HTML, con la idea básica de tener un registro, un login, una publicación de viajes, contratación de viajes, un resumen de viajes, y la desconexión del usuario.

Después del desarrollo de las páginas, gracias a CSS se pudo mover a gusto los distintos elementos, haciendo una interfaz más intuitiva para el usuario. Además, con JavaScript, se permitía la navegabilidad entre ellas. Para el envío de datos entre las distintas páginas, se usan formularios POST, que recogen y envían los datos necesarios que el usuario introduce por pantalla.

4.3 DESARROLLO BACK-END

El siguiente paso fue, comenzar a interactuar con la BBDD, para ello intervino **PHP**, que permite interactuar a la parte front-end (HTML) con el servidor. Además, fue necesaria la creación de formularios para enviar la información que se recoge de una página web a otra página web, como, por ejemplo, a la hora de registrar usuarios o viajes.



La implementación se realiza de la siguiente manera. Existe una parte back-end, en la que tenemos la BBDD y la parte servidora PHP, ambas relacionadas entre sí.
PHP permite la conexión a la BBDD para, posteriormente, hacer las consultas necesarias, inserciones, actualizaciones o borrados.

Es necesario en la **BBDD** crear una conexión nueva. Para este proyecto, basta con utilizar dos tablas para guardar toda la información necesaria, una tabla llamada ***viajes***, y otra llamada ***usuarios***.

La tabla **viajes** contiene los campos de **id_viaje, origen, destino, conductor, pasajero1, pasajero2, pasajero3, fecha, hora, precio y plazas**.

Id_viaje es un identificador único para cada viaje, lo que permite distinguir los viajes cuando los usuarios vayan a reservar un viaje. Los campos de **origen, destino, fecha, conductor y precio** son obligatorios, para no poder publicar viajes que no contengan esos parámetros. El resto si se permiten que sean opcionales.

La tabla **usuarios** es una tabla para los usuarios, donde se guarda el **nombre, apellido, mail, contraseña y username,** siendo este último obligatorio y único, para poder distinguir a los usuarios. Además, en la tabla de viajes, el campo conductor/pasajeros son rellenados con el campo **username** de la tabla **usuarios**.

Tabla **viajes**

#	Nombre	Tipo de datos	Longitud/Conjunto	Sin signo	Permitir NU...	Rellenar co...	Predeterminado
1	origen	VARCHAR	50	☐	☐	☐	Sin valor predetermin...
2	**id_viaje**	**INT**	**4**	☐	☐	☐	**AUTO_INCREMENT**
3	conductor	VARCHAR	50	☐	☐	☐	Sin valor predetermin...
4	destino	VARCHAR	50	☐	☐	☐	Sin valor predetermin...
5	fecha	DATE		☐	☐	☐	Sin valor predetermin...
6	hora	TIME		☐	☑	☐	NULL
7	pasajero1	VARCHAR	50	☐	☑	☐	NULL
8	pasajero2	VARCHAR	50	☐	☑	☐	NULL
9	precio	INT	4	☐	☐	☐	Sin valor predetermin...
10	plazas	INT	1	☐	☑	☐	NULL
11	pasajero3	VARCHAR	50	☐	☑	☐	NULL

Tabla **usuarios**

#	Nombre	Tipo de datos	Longitud/Conjunto	Sin signo	Permitir NU...	Rellenar co...	Predeterminado
1	nombre	VARCHAR	50	☐	☑	☐	NULL
2	apellido	VARCHAR	50	☐	☑	☐	NULL
3	mail	VARCHAR	50	☐	☑	☐	NULL
4	pass	VARCHAR	50	☐	☑	☐	NULL
5	**username**	**VARCHAR**	**50**	☐	☐	☐	Sin valor predetermi...

4.4 LISTADO ARCHIVOS FRONT-END/BACK-END

A nivel de **PHP/HTML** tenemos los siguientes archivos:

- ***TravelMe*.php**. Es la página principal de la aplicación una vez que el usuario se haya logado. Lo interesante de esta página es que se hace un if en el momento de cargarla, comprobando que el usuario esté logado o no. En caso de no estar logado te reenvía a '*Index_no_reg.php'*, si no, se queda en la misma página. Desde aquí podemos navegar a distintas páginas para **publicar** (*Publicar.php*), **buscar** (*Buscar_Viaje.php*), **desconectarnos** (*Fuera_Sesion.php*) o ver los **viajes disponibles** (*ViajesDisponibles.php*) que tenemos.

- **Buscar.php.** Esta página recoge los datos de '*Buscar_Viaje.php*', y en una tabla muestra los resultados que han coincidido con origen, destino y fecha en la BBDD mediante una consulta a la misma.

  Tiene además un formulario que envía los datos a *'Contratar.php'* con un POST, en caso de que el usuario seleccione un viaje y pulse el botón contratar. Se permite contratar los viajes mediante el elemento *checkbox*. Si el usuario quisiera contratarlo, sin haberse logado (hasta aquí puede navegar) se le redirige a la página de '*Reg.html'* para que se registre antes de contratar el viaje. Si en lugar de mostrar la tabla con los viajes, no hubiese viajes con esas características, redirige a '*SinViaje.html'*. Los datos necesarios, se guardan en variables de sesión para pasarlas entre páginas.

- **Buscar_Viaje.php.** Tiene un formulario para enviar los datos a '*buscar.php*'. Recoge los datos del viaje que quiere buscar (origen-destino-fecha).

- **Contratar.php.** En esta página se recogen los datos necesarios del usuario y de los viajes seleccionados para contratarlos. Al usar las consultas, se hace un UPDATE a la BBDD de la tabla viajes, restando una plaza al viaje contratado, y añadiendo como pasajero al usuario logado que haya contratado el viaje. Al contratarlo, redirige a la página principal '*TravelMe.php'*.

- **Fuera_Sesion.php.** Esta página únicamente elimina todas las variables de sesión que haya guardadas y obliga al usuario a volver a la página principal '*TravelMe.php',* que, al hacer una comprobación para ver si el usuario está logado o no, redirige obligatoriamente a '*Index_no_reg.php'.*

- **Index_no_reg.php.** Básicamente es una copia de '*TravelMe.php*', pero sin las opciones de *desconexión*, ver *los viajes disponibles*, ni *publicar viajes*. Se añade además las opciones de logarte o registrarte pulsando en dos botones distintos.

- **Loga.php.** Viene de la página de '*Login.php'* con los datos de username y password. Se hace una comprobación contra la BBDD, y, si ambos existen y coinciden, se le permite al usuario volver a la página de '*TravelMe.php'*, ya identificado. Si falla, vuelve a '*Login.php'.*

- **Login.php.** Tiene 2 Labels en los que pide *username* y *password*. Si el usuario los rellena, se envían con un POST a '*loga.php'*. Si el usuario no está registrado, se le da la opción de hacerlo redirigiendole a '*Reg.html'*.

- **Publica.php.** Recoge los datos POST de '*Pubicar.php'*, y hace un insert a la BBDD con los mismos. Si ha ido bien redirige a '*Publicado.php'.*

- **Publicado.php.** Crea una tabla con un resumen de los datos del viaje publicado.

- **Publicar.php.** Tiene un formulario a '*publica.php'*, donde pide al usuario rellenar los campos origen, destino, fecha, hora, plazas y precio del viaje.

- **Reg.html.** Es un formulario para enviar a '*registro.php'* por POST toda la información necesaria para el registro de un usuario, los campos son: nombre, apellido, mail, username y contraseña.

- **Registro.php.** Recoge por POST las variables enviadas en '*Reg.html'*, y hace un insert a la BBDD. Muestra un mensaje diciendo que el usuario se ha registrado correctamente y obliga a volver a la página de inicio.

- **SinViaje.html.** Este HTML solo da información diciendo que no hay viajes disponibles en el momento en que el usuario buscó un viaje y no se encontró nada en la BBDD que coincidiese en origen, destino y fecha.

- **SinViajeUsuario.php.** Esta página viene de '*ViajesDisponibles.php'*, en caso de que el usuario no tuviese ningún viaje publicado/contratado. Tan solo muestra un mensaje diciendo que el usuario aún no tiene viajes.

- **ViajesDisponibles.php.** Muestra en dos tablas distintas los viajes que el usuario ha contratado o publicado. Si no hubiese, te reenvía a '*SinViajeUsuario.php'*.

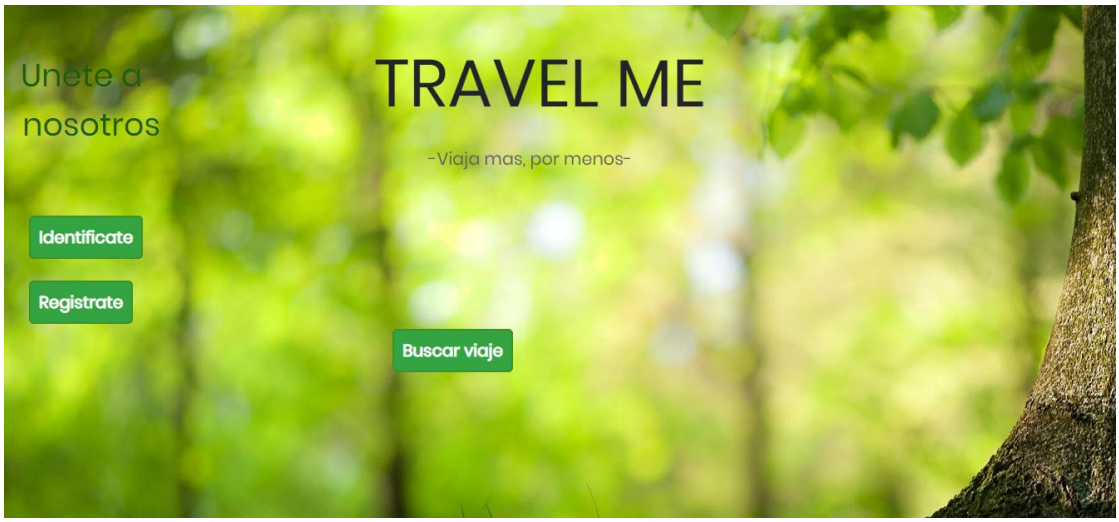
4.5 FLUJOGRAMA NAVEGACIÓN DE USUARIO

El flujo del proyecto es el siguiente:

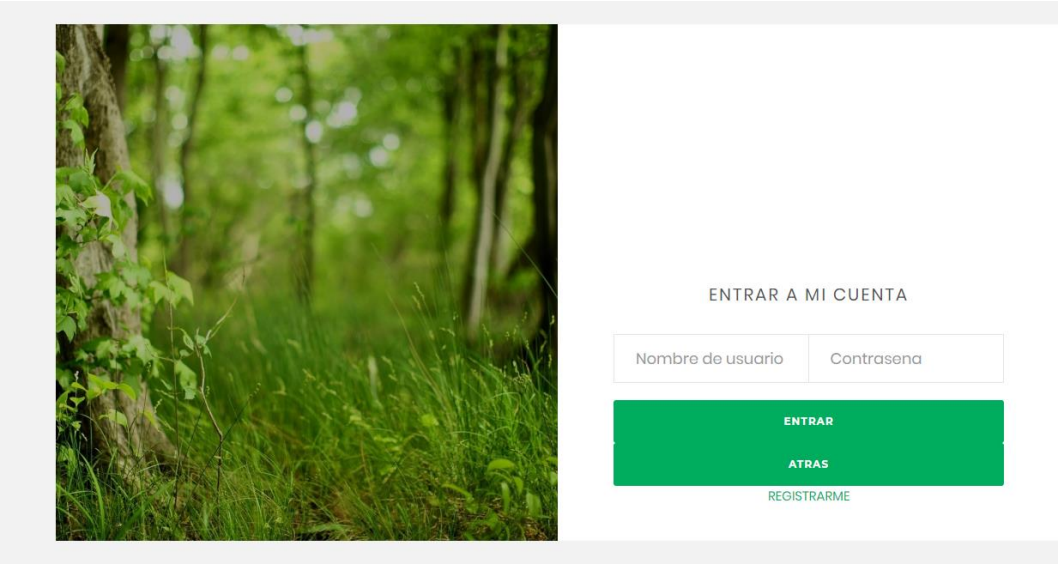
Existe una página principal, que, con PHP, **comprueba si el usuario ha hecho login**. En caso de **no estar logado**, **redirige** a una segunda página principal, bastante similar, pero con menos funcionalidad (no se permite contratar viajes, o ver viajes en los que participe el usuario). Si el usuario **está logado**, puede navegar por distintas páginas, como **buscar** viajes, **contratarlos**, o **ver** los viajes en los que participa como conductor o pasajero.

Para navegar de una página a otra mediante los botones, se utiliza Java Script, y mediante PHP se guarda en sesiones los datos necesarios.

Comenzando el flujo desde el principio, al acceder a la página principal aparece lo siguiente:



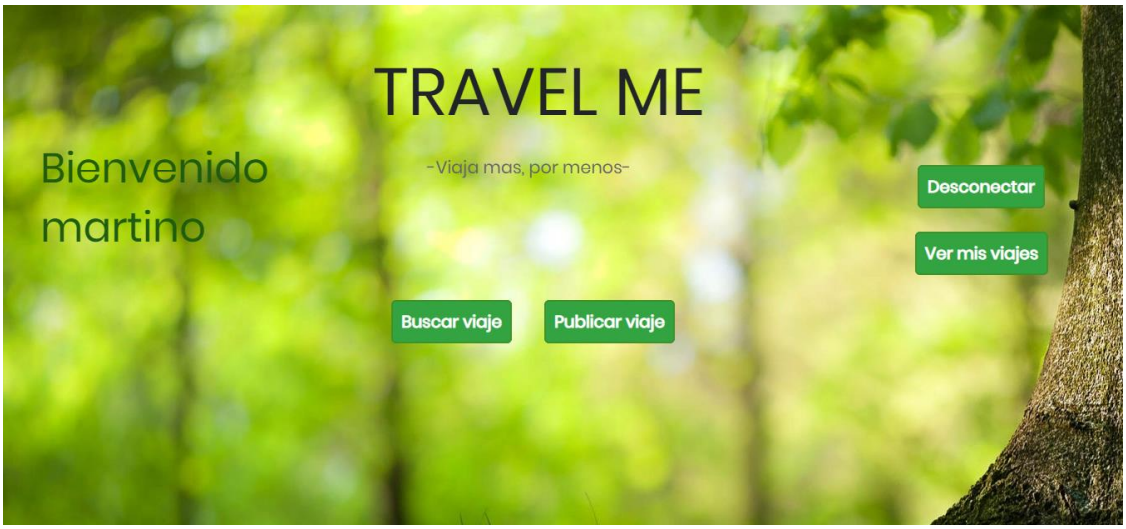
Al pulsar el botón '**Identificate',** redirige a una página para hacer login, en la que pide el nombre de usuario y la contraseña.



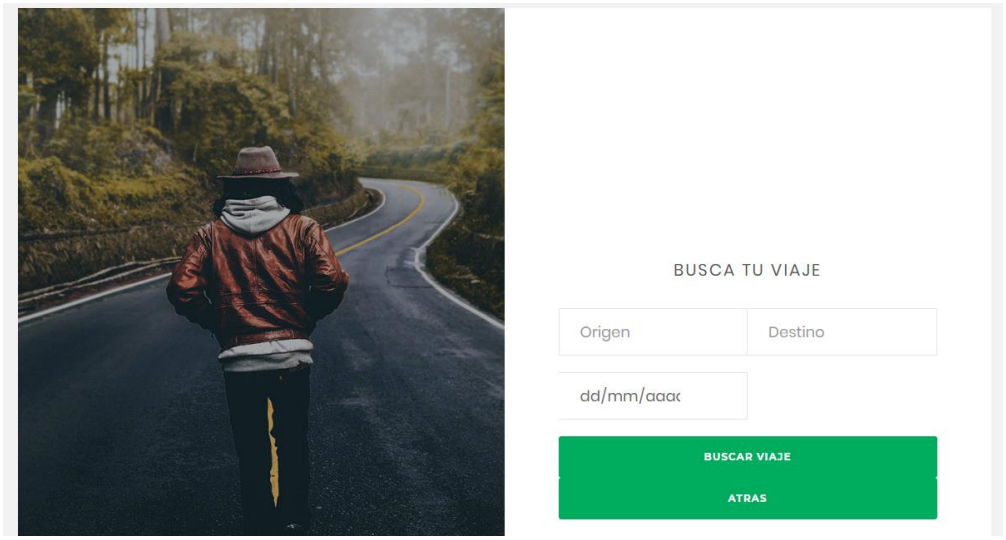
Una vez introducido, comprueba si ambos campos **existen** en la base de datos, y además son **correctos**.

Eso se consigue con la sentencia ***SELECT***, comprobando si existen el par {**usuario-contraseña**}. Esto devuelve una única fila, y, en caso de ser 0, implica que no exista el usuario o que el usuario haya fallado la password. En caso contrario se le permite hacer login.

Si falla, vuelve a pedir login. En caso de logar bien al usuario, **redirige a la página principal**, realizando de nuevo la comprobación para ver si el usuario está logado o no, y, al estarlo, le permite usar las **nuevas funcionalidades**.

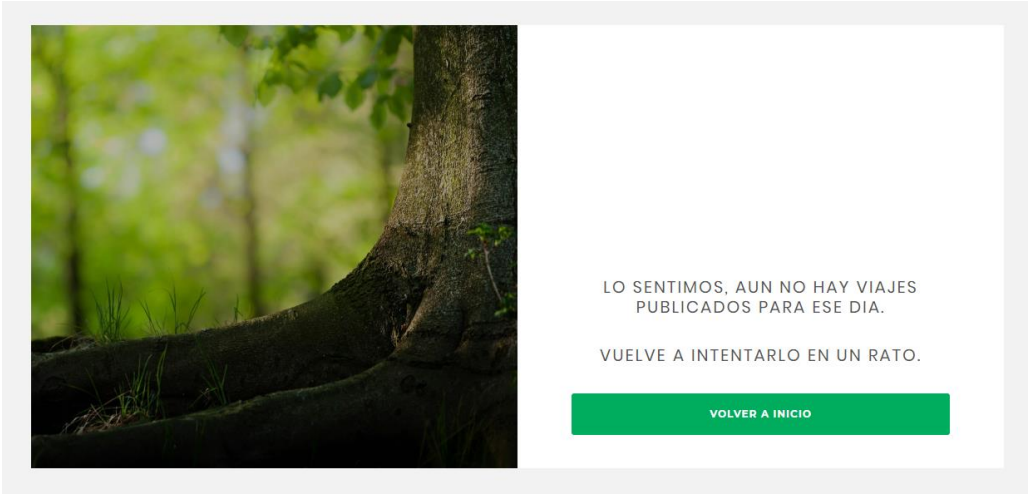


Siguiendo con el flujo, al pulsar '**Buscar viaje**', va a una pantalla donde permite buscar viajes pasándole un origen, un destino, y el día en el que se quiere viajar.

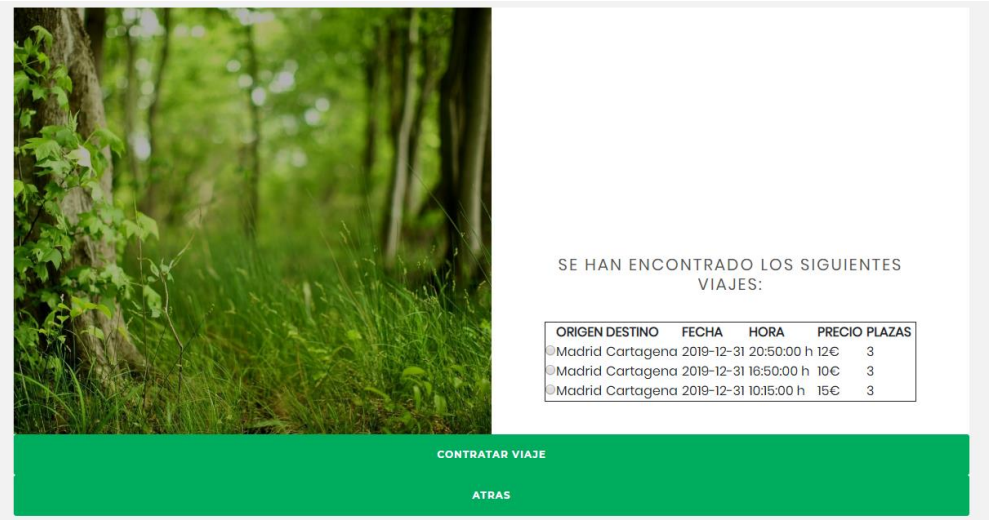


Para la búsqueda de viajes, se necesita ver cuales coinciden en origen, destino y fecha, con los que ya haya guardados en la BBDD. Hay dos opciones, o bien que no haya viaje con esas características, o que si los haya.

En el caso en que un viaje **no exista** muestra esta página.



Volviendo a buscar un viaje, busco un viaje con mismo origen, destino y fecha:



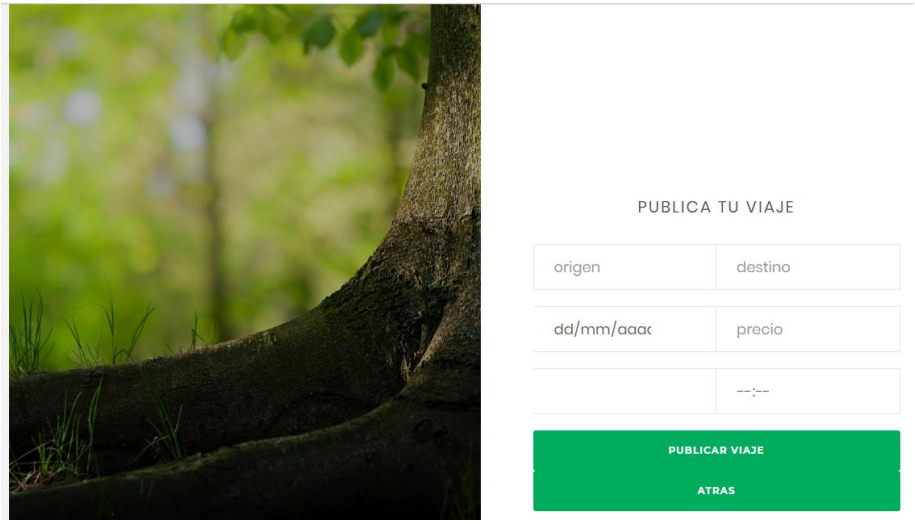
Puede verse que aparecen **3 viajes**, distinguidos por la hora a la que comenzará el viaje, el precio y las plazas disponibles que tiene, que, como máximo son 3.

Existe la opción, al estar logado, de seleccionar uno de ellos y **contratarlo**, o volver a la página anterior a buscar otro viaje.

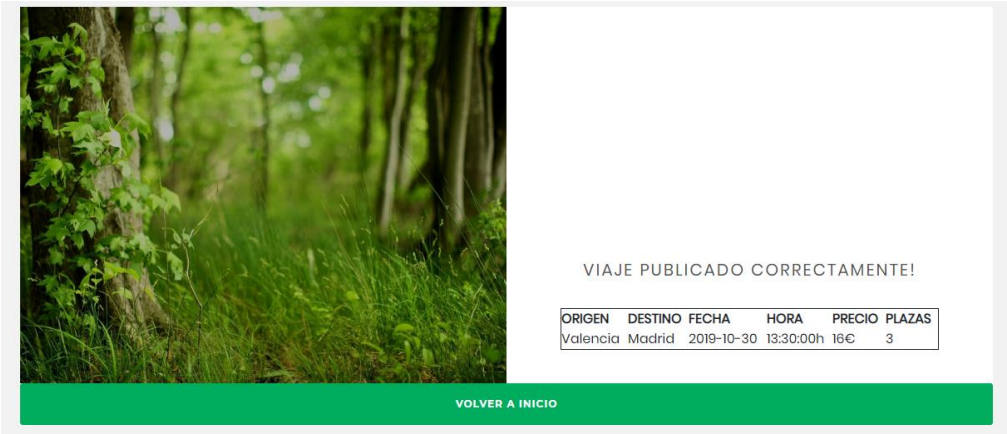
Automáticamente, se le restará una plaza al viaje al contratarlo, asignando al usuario como pasajero a ese viaje. Se escoge el segundo viaje, por ejemplo, cuyo precio es de 10 euros.

En caso de aparecer varios viajes con las mismas características, la forma de distinguirlos al contratarlos es mediante el elemento **radio**, que permite seleccionarlos. Se le asigna al radio el valor de id_**viaje** (campo que se autogenera al crear un viaje), que es **único**, permitiendo distinguirlo así en la BBDD.

Una vez contratado, y de nuevo en la página principal, está también la opción de **publicar viajes**, y, al clickar sobre ella, lleva a la siguiente página:



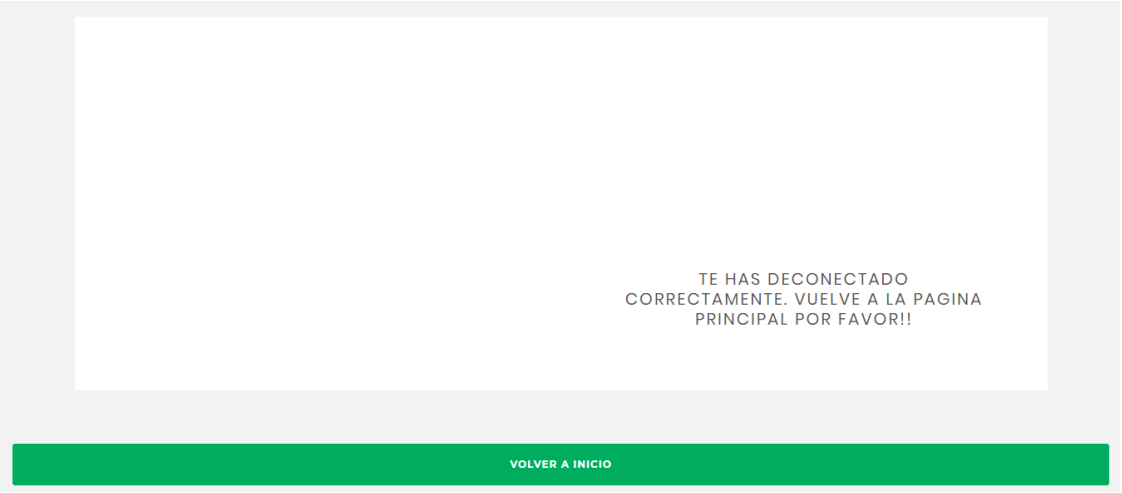
Publicamos (se verá más tarde), un viaje de Valencia a Madrid para el 30 de octubre. Al publicar el viaje, muestra en una tabla un resumen del mismo:



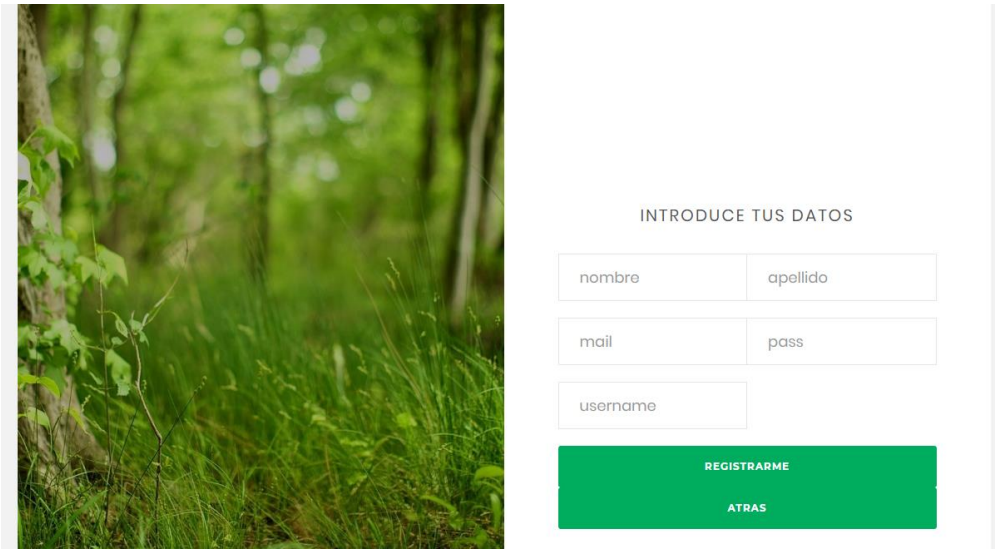
Volviendo a inicio, haciendo click en '**Ver mis viajes**', muestra un resumen de los viajes que el usuario ha publicado (como **conductor**) y los que ha contratado (como **pasajero**).

En este caso, se ve el viaje que se publicó de Valencia a Madrid, y el viaje que se contrató de Madrid a Cartagena.

Por último, el botón **desconectar**, **elimina la sesión**, obligando al usuario a volver a la página de inicio, que, tras hacer la comprobación de login, redirige a la página que permite logarte.



Para finalizar el flujo, queda el botón de '**Registrate**'. Al pulsar sobre él, lleva a la siguiente página, donde pide una serie de datos para poder obtener un usuario:



'Nombre', 'Apellido', 'email', 'Password' y 'Username', siendo este último **único**, para tener una forma de distinguir usuarios con mismo nombre.

5. Conclusiones y líneas futuras

En resumen, con este proyecto lo que se ha pretendido es crear una aplicación web que sea capaz de gestionar viajes con economía colaborativa, sin que haya un intermediario que se lleve comisión. Además, interviene también en la reducción de emisiones al medio ambiente, aprovechando al máximo la capacidad de los coches. Por otra parte, tanto conductor como pasajeros salen ganando, ya que consiguen ahorrar el coste de la gasolina, o del billete de otros medios de transporte que sean más caros, o tiempo, ya que otros medios de transporte puedan ser más lentos.

Este proyecto se podría mejorar, por ejemplo, dándole más funcionalidad al perfil de los usuarios, con opiniones de otros pasajeros/conductores, fotos de los viajes que hayan podido hacer en alguna parada, identificación del coche que usa cada conductor, pidiendo DNI-TLF a los usuarios que se registren… entre otras cosas.

También se podría poner un pequeño resumen de los mejores precios que haya actualmente en los viajes que estén solicitados. También, permitir al usuario al buscar los viajes, ordenarlos por precio o por el número de plazas disponibles que quedan para el viaje.

Otra buena idea, sería añadir ciudades de paso, es decir, no hacer un destino que sea tan solo Madrid>Cartagena, por ejemplo, ya que puede darse el caso de no llenarse las plazas del coche, pero si les pueda interesar a otros usuarios, salir de Albacete para ir a Cartagena, o ser dejados en Albacete desde Madrid. Al estar de paso Albacete, bastaría con hacer una parada y dejar/recoger a los usuarios interesados.

6. Referencias bibliográficas

http://comunicacionsellamaeljuego.com/que-es-blockchain/

https://intereconomia.com/tecnologia/ambitos-se-puede-aplicar-la-blockchain-20170810-1815/

https://uvadoc.uva.es/bitstream/10324/15665/1/TFG-E-141.pdf

https://www.ionos.mx/digitalguide/online-marketing/vender-en-internet/blockchain/

https://medium.com/@i_abel/blockchain-para-dummies-22bd5a52cfe8

https://retina.elpais.com/retina/2017/12/22/tendencias/1513937575_114270.html

https://www.buda.com/blog/posts/que-es-un-smart-contract/

https://www.paradigmadigital.com/techbiz/smart-contracts-algo-nuevo-o-mas-de-lo-mismo/

https://economipedia.com/definiciones/economia-colaborativa.html

https://cysae.com/futuro-blockchain-seis-casos-de-uso-en-la-actualidad/

http://www.vigilancer.es/noticias/oportunidad-blockchain-en-el-presente-y-futuro-de-la-industria

https://cincodias.elpais.com/cincodias/2018/07/30/mercados/1532962048_051932.html

https://www.goodrebels.com/es/como-funciona-un-smart-contract/

https://academy.bit2me.com/que-son-los-smart-contracts/

http://www.mclibre.org/consultar/htmlcss/css/css-posicionamiento-absoluto.html

http://www.mclibre.org/consultar/htmlcss/css/css-tablas-modos-bordes.html

https://www.lawebdelprogramador.com/foros/SQL-Server/627068-Update-con-dos-tablas.html

https://docs.microsoft.com/es-es/sql/analysis-services/multidimensional-models/define-a-many-to-many-relationship-and-many-to-many-relationship-properties?view=sql-server-2017

https://stackoverflow.com/questions/26929057/sql-server-how-to-update-only-one-row-in-database

https://stackoverflow.com/questions/12363047/how-to-update-column-in-a-table-from-another-table-based-on-condition

https://www.php.net/manual/es/function.pg-fetch-array.php

https://www.php.net/manual/es/control-structures.for.php

https://stackoverflow.com/questions/4631224/getting-multiple-checkboxes-names-ids-with-php

https://php.net/manual/es/mysqli.store-result.php

https://www.php.net/manual/es/mysqli.query.php

https://www.w3schools.com/php/func_mysqli_num_rows.asp

https://stackoverflow.com/questions/4620391/mysql-and-php-insert-null-rather-than-empty-string/37067920

https://dzone.com/articles/intro-to-blockchain-with-ethereum-web3j-and-spring

http://www.joseluisestevez.com/index.php/2017/07/16/creando-aplicaciones-la-blockchain-ethereum-usando-java-web3j/

https://github.com/ethereum/wiki/wiki/JavaScript-API

https://es.wikipedia.org/wiki/JavaScript

https://remix.ethereum.org/#optimize=false

https://www.miethereum.com/smart-contracts/solidity/

https://es.wikipedia.org/w/index.php?title=Solidity&action=edit&redlink=1