DISEÑO, DESARROLLO E IMPLANTACIÓN DE INFORMACIÓN WEB: DISEÑO E IMPLEMENTACIÓN DE LA CAPA DE PERSISTENCIA DE DATOS

Sistemas de Información

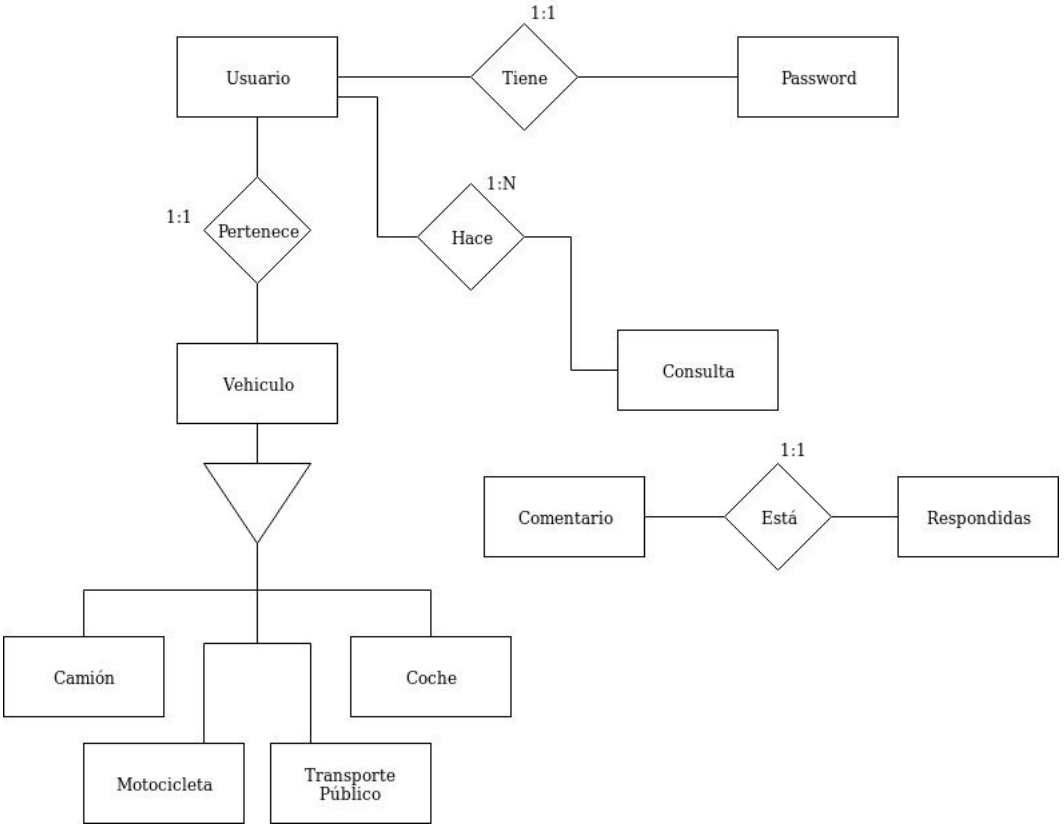
Curso 2019-2020

Raúl Rustarazo Carmona - 715657

Pedro Tamargo Allué - 758267

Juan José Tambo Tambo - 755742

1. Creación de la Base de Datos



Modelo entidad-relación de la Base de Datos

Consulta: id: BIGINT, NOT NULL, AUTOINCREMENT; login: VARCHAR(20), NOT NULL; fecha: DATE, NOT NULL;
origen: VARCHAR(40), NOT NULL; destino: VARCHAT(40), NOT NULL;

Usuario: login: VARCHAR(20), NOT NULL; password: VARBINARY(20), NOT NULL; nombre: VARCHAR(20),
NOT NULL; apellidos: VARCHAR(30), NOT NULL; email: VARCHAR(20), NOT NULL; vehic_id: BIGINT, NOT NULL;

Text

Vehiculo: id: BIGINT, NOTNULL; tipo: VARCHAR(25); segment: VARCHAR(35), NOT NULL;
euro_star: VACRHAR(25), NOT NULL; engine_type:VARCHAR(10), NOT NULL;
fuel: VARCHAR(20), NOT NULL; pollutant: VARCHAR(5), NOT NULL;
emission_factor: FLOAT, NOT NULL;

Comentario: comentarioId: BIGINT, NOTNULL, AUTOINCREMENT; texto: VARCHAR(250), NOT NULL;
email: VARCHAR(20), NOT NULL; nombre: VARCHAR(20), NOT NULL;
apellidos: VARCHAR(30), NOT NULL; respondida: BOOLEAN, NOT NULL;

* El campo Tipo de vehículo unicamente puede tener como valores: "Camión", "Motocicleta", "Transporte público" y "coche"

* En el modelo E-R, se considera password como una entidad, aunque al existir una relación 1:1 con usuario, se ha decidido mantener en una sola entidad

Modelo relacional de la Base de Datos

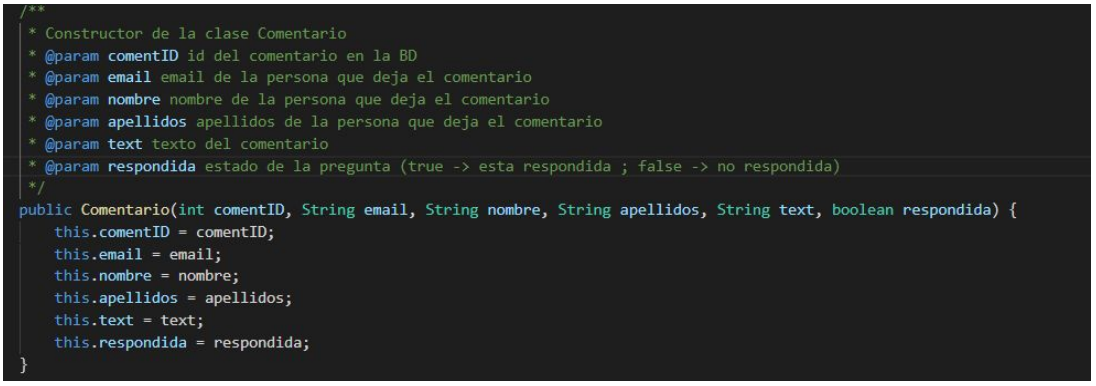
Para la implementación de la BD hemos elegido MariaDB[1], un branch de MySQL, de código abierto, ya que ya habíamos trabajado anteriormente con este SGBD.

Para la implementación en el SGBD MariaDB, hemos partido del esquema relacional y hemos creado las siguientes tablas:

- Vehículo: tabla que mantiene la información de los vehículos con factores de emisión, contaminante, segmento de mercado y combustible que utilizan.
  Como se ha indicado en una restricción del modelo relacional, los únicos tipos de vehículo que se pueden introducir son: camión, coche, trasnporte público y motocicleta.
- Usuario: tabla que mantiene la información de un usuario, identificado por su nickname (login) y que almacena su nombre, apellidos, correo electrónico, y tipo de vehículo.
- Comentario: tabla que mantiene la información acerca de los comentarios escritos en la sección de “Help & Contact”, cada comentario se identifica por un ID único, y tienen información asociada como: el nombre y los apellidos de la persona que hace la consulta, su email y si ha sido respondida o no.
- Consulta: tabla que almacena la información acerca de una ruta consultada por un usuario. Almacena su dirección de origen, su dirección destino, la fecha en la que fue consultada, el usuario que ha hecho la consulta y, un identificador único de consulta.

2. Implementación de clases en la capa de persistencia de datos

Tras la creación de la base de datos, se procede a crear las clases necesarias para la gestión de las tablas en lenguaje Java. Por lo tanto, nos encontraremos las clases Usuario, Comentario, Vehiculo y Consulta, cuyos atributos serán una representación de cada columna de sus respectivas tablas.
Para la documentación de este código se ha optado por el uso de la herramienta Javadoc[2], para lo cual se ha añadido la documentación sobre el propio código, de la siguiente forma:



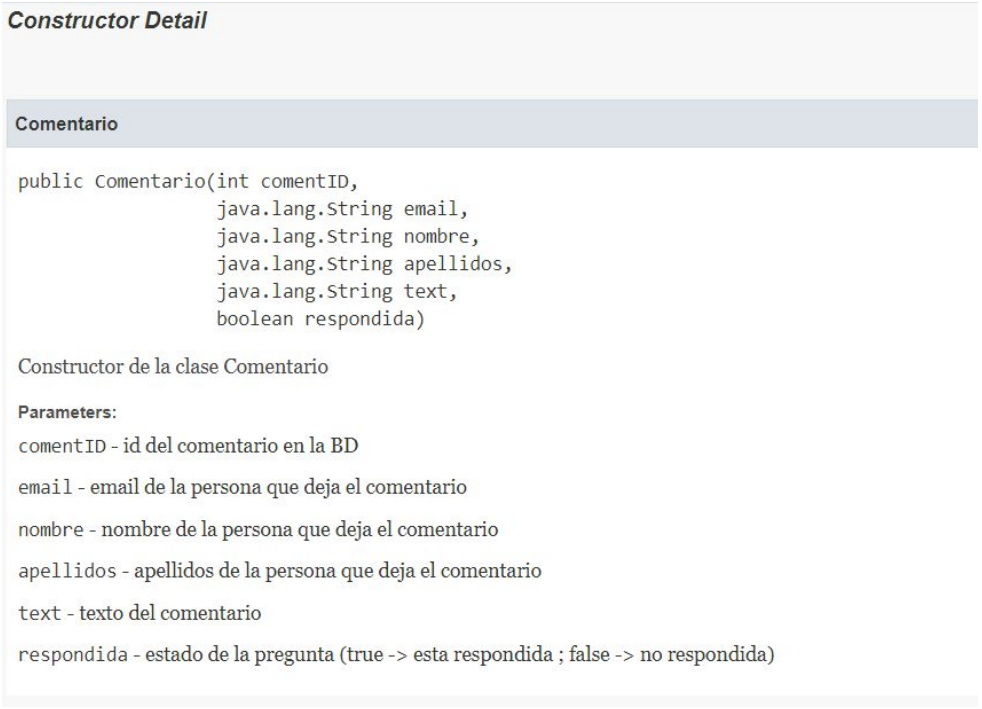
```java
/**
 * Constructor de la clase Comentario
 * @param comentID id del comentario en la BD
 * @param email email de la persona que deja el comentario
 * @param nombre nombre de la persona que deja el comentario
 * @param apellidos apellidos de la persona que deja el comentario
 * @param text texto del comentario
 * @param respondida estado de la pregunta (true -> esta respondida ; false -> no respondida)
 */
public Comentario(int comentID, String email, String nombre, String apellidos, String text, boolean respondida) {
    this.comentID = comentID;
    this.email = email;
    this.nombre = nombre;
    this.apellidos = apellidos;
    this.text = text;
    this.respondida = respondida;
}
```

Captura de pantalla de formato de documentación Javadoc

En la cual se observa que podemos proveer a cada método o constructor, para cada uno de sus parámetros y para el valor de devolución una descripción indicando el comportamiento del mismo.

Para la capa de acceso de a la base de datos se ha utilizado la clase *"ConnectionManager"* provista en el fichero *"PatronDAO_VO.pdf"* de la plataforma Moodle, junto con la clase *"VehiculoDAO"*, encargada de recolectar datos sobre los vehículos, ya sean todos los disponibles o los que cumplan ciertas condiciones, tales como su tipo, su segmento comercial, la tecnología del motor, su tipo de combustible…

Tras la documentación del código, podemos generar un conjunto de ficheros .html para visualizar la documentación de la forma:



Constructor Detail

Comentario

```
public Comentario(int comentID,
                  java.lang.String email,
                  java.lang.String nombre,
                  java.lang.String apellidos,
                  java.lang.String text,
                  boolean respondida)
```

Constructor de la clase Comentario

Parameters:

`comentID` - id del comentario en la BD

`email` - email de la persona que deja el comentario

`nombre` - nombre de la persona que deja el comentario

`apellidos` - apellidos de la persona que deja el comentario

`text` - texto del comentario

`respondida` - estado de la pregunta (true -> esta respondida ; false -> no respondida)

Captura de pantalla de la web generada por Javadoc sobre el ejemplo anterior

Donde se puede apreciar que los comentarios anteriores y la cabecera de la función se han convertido a una documentación en una web, con funciones de búsqueda de métodos, clases…

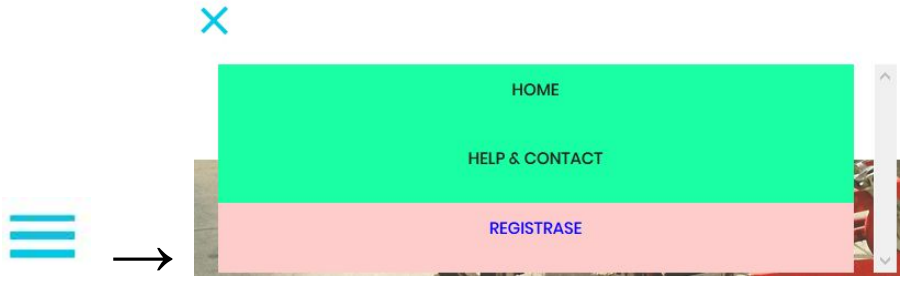
3. Cambios en la vista html de la aplicación Web

Durante el desarrollo de esta práctica, se ha ido mejorando de forma notable el diseño de la vista que tendrá nuestra página web para todos los clientes que la usen. Se ha optado por usar una plantilla web gratuita de la muchas que hay disponibles en internet y, poco a poco, hemos procedido a ir adaptando el código que teníamos de nuestra anterior web a la nueva página.
Sobre todo, se ha reutilizado parte del código JavaScript y los ficheros CSS. Se ha analizado el resto del contenido para comprender qué partes podrán reutilizarse. Un ejemplo de la reutilización del código, son los enlaces de la cabecera, que cambian de color si se cambia el ratón sobre ellos.

Ejemplo de sección de navegación: nuestro puntero está sobre la pestaña **REGISTRARSE**. Mientras estamos en la página **HOME**.

Ejemplo: Cuando el usuario ha pulsado para ir a una de las secciones de la web, se mantendrá el color (rojo en este caso) para que siempre sea consciente de dónde está.

Los colores son temporales hasta que se elijan los más adecuados.



Ejemplo: Al reducirse el tamaño de la pantalla, la página web se adapta al tamaño de la pantalla y contrae el menú de navegación en el icono de la izquierda. Esta característica es parte del código JavaScript original.

Aún se están resolviendo algunos problemas respecto a la interfaz y el contenido de los formulario fundamentales para el integrar la base de datos. Como por ejemplo, entre los datos del cliente, precisamos que contenga ciertos datos de su vehículo indispensables para que se puedan hacer la operaciones correctas, tales como el combustible, la tecnología del motor, etc. Para ello, habría que añadir un conjunto de listas desplegables (las cuales contendrán información acerca de distintos campos que aparecen en la tabla Vehículo de la BD) al formulario para registrase.

Además, se han comenzado a utilizar mapas en la página web. Para ello, ha resultado de gran ayuda la librería Javascript de código abierto *Leaflet*[4].
Con esta herramienta se puede introducir marcadores y marcar determinadas zonas en el mapa mediante círculos o polígonos.
Contiene también una serie de plugins con los que se le pueden añadir diferentes funciones al mapa, que usaremos en versiones posteriores de la página.
Se han introducido dos mapas:

- En la página principal, el cual se encarga de mostrar las rutas devueltas correspondientes a las consultas introducidas por el usuario
- En el apartado de ayuda y contacto, en el cual se indica el lugar desde el que trabajan los administradores de la página (Centro Politécnico Superior), mediante un marcador.

4. Implementación sobre servidor

El servidor donde se va a realizar el despliegue del sistema es una Raspberry Pi Model 3, ubicada en casa de unos de los integrantes del equipo. También se aloja en este el servidor Tomcat.
De cara a la conexión al servidor, se ha adquirido el dominio: ecobicizara.tk (se ha elegido este dominio (.tk) ya que son gratuitos durante un periodo máximo de 1 año). También se ha intentado conseguir un certificado SSL para la verificación del dominio mediante *CertBot* [3].

5. Metodología y costes

Para el desarrollo de la práctica, hemos dividido el trabajo entre los integrantes del grupo de la siguiente manera:

- Pedro: Creación y población de tablas para BD , elaboración modelo relacional, configuración del servidor y codificación de clases Java para la BD.

- Juanjo: Introducción de los mapas a la página web, elaboración modelo relacional y modificación de HTML.

- Raúl: Modificación HTML, desarrollo de javascript.

Al dividir desde un principio el trabajo, ha resultado mucho más fácil el desarrollo de esta práctica

Horas de trabajo:

- Raúl: 3 horas (2 h con HTML y 1 con javascript)
- Juanjo: 2 horas (10 minutos relacional, 1 h mapas y 1 hora HTML)
- Pedro: 5 horas (2 h con BD, 1 h modificando servidor, 2 horas con javascript)

6. Bibliografía

[1]:(23/10/2019) https://mariadb.com/
[2]:(25/10/2019)
https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html
[3]:(27/10/2019) https://certbot.eff.org/
[4]:(36/10/2019) https://leafletjs.com/examples/quick-start/