



Instituto Cendrassos

Ciclo de desarrollo de aplicaciones web

GAS FINDER

Proyecto fin de ciclo

Autor:

Dylan Garrido Hurtos

Junio 2020

Índice

Índice	1
Introducción	4
Que es Gas Finder?	4
Cómo nació la idea?	4
Funcionamiento	5
Mejoras futuras	7
Ampliación de los tipos de carburantes	7
Incorporación de surtidores para vehículos eléctricos	7
Realización de una aplicación móvil	8
Trazado de rutas	8
Incorporación de banners publicitarios	8
Sistema de favoritos y autoguardado	8
Funcionamiento Interno	9
Explicación	9
Esquema	9
Manual de desarrollo	10
M01: Sistemas informáticos	10
Introducción	10
Servidor	10
Software	12
Seguridad	13
Backups	14
Backup BDD	14
Backup Proyecto	15
M02: Base de datos	16
Introducción	16
Modelo E-R	16
Modelo Relacional	17
Trigger	18
Consultas	20
Otros	21
M03: Programación	22
Introducción	22
Estructura	22
Código	23
Ejecución	25

Otros	26
M04: Lenguaje de marcas	27
Exportar datos y copias BDD en XML	27
APIS	29
API XML	30
API JSON	31
API CSV	32
Otros	33
M05: Entornos de desarrollo	34
Introducción	34
Diagrama de casos de uso	34
Diagrama de actividades	35
Diagrama de estados	36
Diagrama de secuencia	37
Diagrama de clases	38
Diagrama de objetos	39
TDD (Test driven development)	40
M06: Desarrollo de aplicaciones en entorno cliente	42
Introducción	42
Estructura	42
Código	44
Mapa	44
Filtros	48
Otros	49
M07: Desarrollo de aplicaciones en entorno servidor	50
Introducción	50
Código	50
Migraciones	50
Modelos	52
Vistas	54
Rutas	55
Controladores	56
Otros	58
M08: Despliegamiento de aplicaciones web	59
Introducción	59
Manual	59
Instancia	59
Conexión remota vía SSH	61
Apache	62
PHP	63

MYSQL	64
Configuración servidor	65
Composer	67
Laravel	68
GIT	70
Configuraciones finales	71
Aporte	73
M09: Diseño de interfaces web	74
Introducción	74
Usabilidad y Accesibilidad	74
Usabilidad	74
Accesibilidad	76
Prototipado	78
Wireframes	78
Paleta de colores	82
Logo	82
Iconos	82
Responsive (RWD)	83
CSS GRID	86
CSS3	87
Bootstrap	88
M11: Empresa Iniciativa Emprendedora	89
Valoraciones Personales	90
Código Github	91
Bibliografía / Webgrafía utilizada	92
Laravel y JS	92
Desplegar	92
Programación	93
Base de Datos	93

Introducción

Que es Gas Finder?

Gas Finder es una aplicación web que, en la actualidad, ofrece la oportunidad a sus clientes de conocer los precios más económicos de diversos carburantes a nivel peninsular a través de una web simple e intuitiva, donde podrán ver y comparar los carburantes deseados entre las diversas gasolineras de una localización indicada.

Cómo nació la idea?

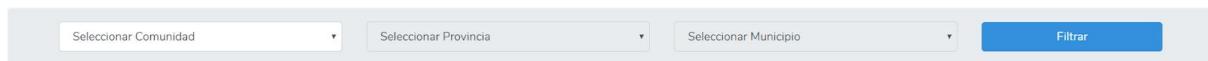
La idea nació poco después de sacarme el carnet de conducir, al ver que estaba invirtiendo más dinero en llenar el depósito que en mis necesidades diarias, teniendo en consideración que mis ingresos provenían de un trabajo de media jornada laboral y realizando más de 6 trayectos en coche al día, lo que suponía un consumo elevado de carburante.

Al final decidí llevar este proyecto adelante, al ver que los diversos comparadores web de gasolina que se pueden encontrar son muy limitados. Ofrecen información desactualizada, pocos filtros o incluso una interfaz plagada de publicidad entre otras cosas. Esto, sumado al alto consumo de carburante como he mencionado anteriormente y mis ingresos limitados, hicieron que naciera Gas Finder.



Funcionamiento

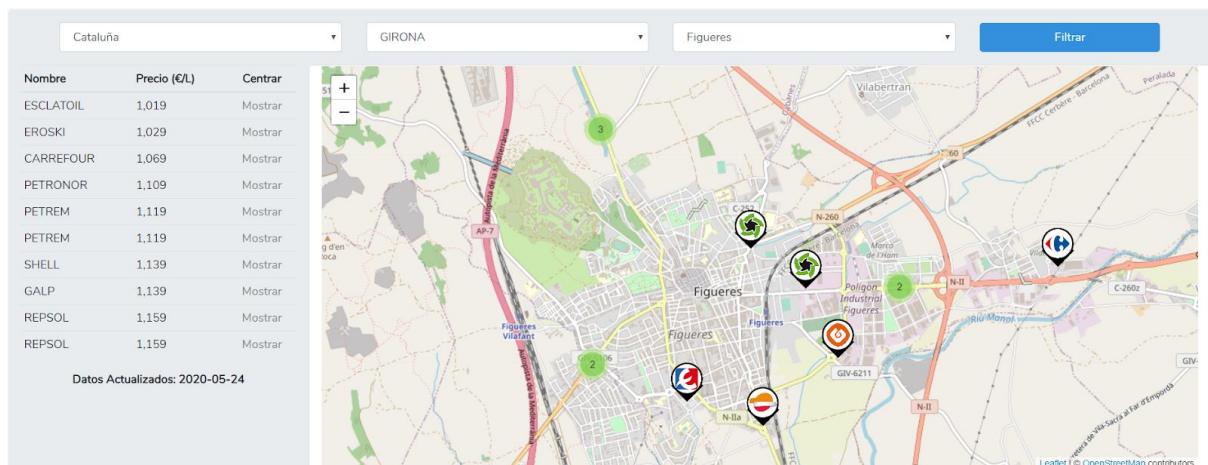
La web, proporciona distintos filtros para poder satisfacer la necesidades de los usuarios de forma más precisa. Pueden filtrar desde comunidades autónomas, hasta municipios, pasando por sus respectivas provincias. Tal y como podemos ver en la imagen inferior:



Por otro lado, pueden seleccionar el tipo de carburante que usen o deseen filtrar, entre ellos encontramos los principales carburantes como la gasolina y el gasoil, pero también podemos encontrar carburantes BIO como el biodiesel y el bioetanol:



A continuación, el usuario puede filtrar las gasolineras con los datos que ha introducido. Esto hará que solo las estaciones de servicio que cumplan los requisitos indicados por el cliente, se indiquen en el mapa.



Al mismo tiempo, podremos observar una tabla lateral en la que nos aparecerán las 10 primeras estaciones de servicio de la zona por orden de más a menos económica, indicando el precio del carburante indicado en euros/Litro. También proporciona un botón llamado “Mostrar” que permite al usuario centrar la gasolinera en el mapa al pulsar-lo.

Finalmente, al hacer clic encima de un marcador de una gasolinera, se abrirá un popup informativo donde se podrá visualizar distintos datos relevantes de esa gasolinera en particular como por ejemplo el nombre, la dirección, el horario y finalmente el precio de la gasolina filtrada:



Mejoras futuras

Mi intención en un futuro, es aumentar el potencial de esta aplicación web añadiendo nuevo contenido y nuevas funcionalidades, para ofrecer un servicio eficaz, compacto y simple donde el usuario pueda encontrar todo tipo de información y opciones de forma fácil e intuitiva.

A continuación se indican algunas de las características planeadas para añadir a Gas Finder, que por desgracia, en el tiempo invertido en este proyecto estas últimas 4 semanas, no ha sido suficiente para poder implementar las siguientes funcionalidades:

1. Ampliación de los tipos de carburantes

La intención es aumentar la cantidad de filtros de carburantes que hay actualmente. Esta ampliación se encuentra en proceso, la aplicación internamente ya gestiona también carburantes como los gases naturales licuados y el nuevo diésel.

2. Incorporación de surtidores para vehículos eléctricos

Otra de las principales características que quiero implementar, es la de añadir la opción de filtrar surtidores eléctricos.

Actualmente en España, solo un 1% de los vehículos en circulación son eléctricos o híbridos lo que, aún supone un nicho pequeño para mi aplicación web. Aun así, es un mercado en auge que va creciendo de forma lenta pero exponencial y en algún momento gran parte de los vehículos serán híbridos o eléctricos.

Aquí podemos ver un artículo que menciona las previsiones para dentro de 10 años, que al final, no dejan de ser previsiones, pero aun así es un sector al que tener en cuenta:

TECNOLOGÍA Y TENDENCIAS

En 2030 el 55% de los coches nuevos serán eléctricos o híbridos

3. Realización de una aplicación móvil

Otro de los sectores a tener en cuenta es el de las aplicaciones móviles. Quien hoy en día no tiene un smartphone? El mundo de las aplicaciones móviles a día de hoy mueven más de 100.000 millones de \$ y tener una aplicación en el móvil en la que puedas revisar donde está el carburante más económico, en ese instante, en la zona donde se encuentra el usuario, es una opción a tener en consideración en un futuro cercano

4. Trazado de rutas

En este caso, la idea se basa en crear una ruta entre la ubicación actual del usuario y la ubicación de la estación de servicio seleccionada por el mismo. Con esta función le daríamos la oportunidad al usuario de poder llegar a la gasolinera deseada de la forma más cómoda y rápida posible.

Como actualmente desconozco la dificultad para realizar dicho apartado, una alternativa relativamente sencilla sería incorporar un botón en la lista de las gasolineras, que al hacer clic abriera Google maps y realizará una ruta automática desde esa aplicación

5. Incorporación de banners publicitarios

Al final, un duro trabajo tiene que reportar algún beneficio propio y si encima de ahorrarme dinero al repostar carburante generar algún ingreso económico pues mejor.

Simplemente sería poner 1 o 2 banners publicitarios (sin fastidiar la estética de la página).

6. Sistema de favoritos y autoguardado

En este apartado me gustaría incorporar un sistema donde un usuario registrado pueda marcar las estaciones de servicio como favoritas (por precio, proximidad,etc) y recibir notificaciones o avisos de cuando esas gasolineras hacen bajadas de precio.

Por otro lado poder almacenar los datos al filtrar, así al volver a logearse la próxima vez se le cargan automáticamente los datos con los parámetros de la última vez que uso la aplicación web.

Hay muchas más ideas, pero las primeras en realizarse serían las anteriores.

Funcionamiento Interno

Explicación

Esta es una breve explicación de cómo funciona la aplicación web, más adelante se profundizará en sus mecánicas más internas en función del módulo formativo.

El funcionamiento de mi aplicación web está basado principalmente en la llamada de una API REST, en la cual se gestiona la información recibida y se sube a una base de datos.

Por otra parte Gas Finder recibe estos datos también vía API, pero en este caso desde la propia base de datos personal, así reducimos el número de peticiones a la API REST pública del estado, lo que podría llevar a que nos limitaran las peticiones si hay un número alto de usuarios.

Finalmente la web se encarga de filtrar las gasolineras recibidas de la BDD para mostrar solo las que el cliente quiere.

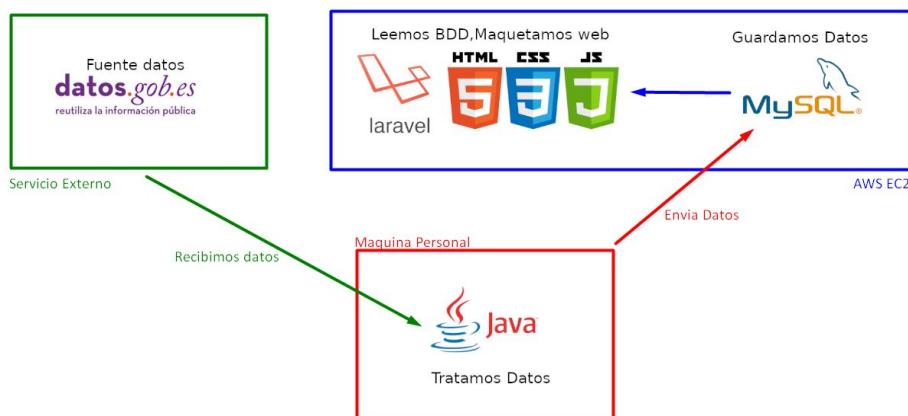
Esquema

A continuación podemos ver un esquema básico del funcionamiento de Gas Finder.

En **Verde** encontramos el servicio externo de datos [GOB](#).

En **Rojo** encontramos mi maquina personal, donde se procesan los datos y se envían al servidor.

En **Azul** encontramos el servidor EC2 de AWS donde tenemos el hosting de la web



Manual de desarrollo

A continuación podremos ver una documentación detallada de cómo he ido elaborando el proyecto, partiendo de las cosas más esenciales como la base de datos o el diseño hasta finalmente su despliegue. Esta parte estará dividida en los diferentes módulos formativos para que quede bien diferenciado que se ha realizado en cada apartado.

M01: Sistemas informáticos

Introducción

Para empezar el manual, comentaremos el tipo de servidor que estoy usando y cómo han sido configuradas sus redes.

También hablaremos de aspectos generales de su configuración y seguridad aunque gran parte de ella sea generada de forma automática.

Servidor

Para poner este proyecto en marcha y a disposición de cualquier usuario en línea, he usado el servicio de EC2 (Elastic Compute Cloud) de AWS (Amazon Web Service).

El servicio de Elastic Compute Cloud es una parte central de la plataforma de cómputo en la nube de la empresa Amazon.com denominada Amazon Web Services. EC2 permite a los usuarios alquilar computadores virtuales en los cuales pueden ejecutar sus propias aplicaciones.

En nuestro caso se nos ha proporcionado un total de 50\$ para invertirlos en el sistema que queramos. Como ya he comentado antes, he destinado el dinero para la implementación de un servidor EC2.

Este servidor no tiene muchas prestaciones, debido a que nuestro nivel de usuario en los servicios de AWS no tiene menos servicios y opciones disponibles. Por ello, nuestro servidor está equipado con un SSD que funciona mediante EBS que consiste en un servicio de almacenamiento de bloque de alto rendimiento con facilidad de uso, diseñado para su uso tanto en cargas de trabajo intensivas de rendimiento, como de transacciones a cualquier escala.

En cuanto a CPU, contamos con un único procesador para la instancia que estamos utilizando, este es de la familia de Intel Xeon.

Por otro lado, en el apartado de memoria ram, podemos encontrar 1GB destinado a nuestro servidor. No es mucha, pero suficiente para poder desplegar nuestro proyecto y que un pequeño grupo de personas puedan estar conectadas de forma simultánea en mi aplicación web.

Revisando el apartado de red, podemos ver que nuestra máquina dispone de una interfaz de red.

Network interfaces eth0

En la que podemos observar que tiene una ip estática y un DNS configurado por defecto.

Network Interface eth0	
Interface ID	eni-08f7efc56070cc5ce
VPC ID	vpc-34d6c74e
Attachment Owner	965877200813
Attachment Status	attached
Attachment Time	Sat May 23 12:05:36 GMT+200 2020
Delete on Terminate	true
Private IP Address	172.31.52.15
Private DNS Name	ip-172-31-52-15.ec2.internal
Public IP Address	-
Source/Dest. Check	true
Description	-
Security Groups	launch-wizard-1
Elastic Fabric Adapter	Disabled

Mencionar también, que cada vez que iniciamos la instancia, se genera de forma automática un DNS público y una ip pública, con la que poder acceder a nuestro sitio web.

Public DNS (IPv4)	ec2-100-25-48-116.compute-1.amazonaws.com
IPv4 Public IP	100.25.48.116

Software

En el apartado de software, comenzaremos por comentar el sistema operativo usado, que en este caso es la versión de Ubuntu 18.04. Que es un sistema operativo de software libre y código abierto, es una distribución de Linux basada en Debian.

La versión 18.04 es la más actual que proporciona EC2 por el momento. Por ello y visto que no hay que tocar apenas parámetros de red, me he decidido a usar esta versión.

En la siguiente lista encontraremos los demás softwares/herramientas/librerías utilizadas en este servidor.

- **MYSQL 7.4**: Es un sistema de gestión de bases de datos relacional comercializado por Oracle Corporation. En mi caso he utilizado la versión
- **PHP 7.4:** Es un lenguaje de programación de propósito general de código del lado del servidor. Esta librería ha sido instalada debido a que el framework que utilizó para realizar la aplicación web, se basa en este código.
- **Apache 2.4:** Es un software de servidor web gratuito y de código abierto para plataformas Unix con el cual se ejecutan el 46% de los sitios web de todo el mundo.
- **Composer 1.1:** Composer es un sistema de gestión de paquetes para programar en PHP el cual provee los formatos estándar necesarios para manejar dependencias y librerías de PHP.
- **Laravel 7:** Es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7. Su filosofía es desarrollar código PHP de forma elegante y simple.
- **Phpmyadmin 5:** Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando un navegador web.

Estos serían la mayoría de softwares/herramientas/librerías utilizadas en el servidor de EC2.

Seguridad

EC2 de AWS incorpora muchos sistemas de seguridad para tener a sus usuarios y sus respectivos datos seguros.

Empezaremos por comentar que EC2 funciona con un sistema llamado VPC (Virtual Private Cloud), que se trata de conjunto de recursos computacionales configurables por demanda al interior de un ambiente de cloud público, el cual provee un cierto nivel de aislamiento entre las diferentes organizaciones que utilizan dichos recursos.

Por otro lado, Amazon Web Services se hace responsable de mantener la estructura donde se almacenan y interactúan los diversos servicios que ofrecen, contratan a auditores profesionales para comprobar de forma constante la eficacia de la seguridad en su sistema en la nube.

A nivel más personal, al crear una instancia en el servicio de EC2, hemos de crear una llave (.pem), que nos permite realizar una conexión de forma segura de forma remota mediante SSH.

Para realizar esta conexión con el servidor, hemos de proporcionar dicha llave, sino, no podremos establecer conexión.

```
C:\Users\Ubuntu@IP-172-31-32-10: ~  
C:\Users\Dylan Garrido\Desktop\ProyectoFinal\Desplegar\Certificado>ssh -i laraveldaw.pem ubuntu@100.25.196.161  
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-1065-aws x86_64)
```

Un punto relacionado con el tema anterior, es que al crear la instancia del servidor, tendremos solamente el puerto 22 TCP de tráfico entrante abierto, solo tendremos este puerto abierto de base para realizar conexiones. Posteriormente abriremos otros puertos para nuestros propósitos personales.

HTTP	TCP	80	::/0	-
SSH	TCP	22	0.0.0.0/0	-

Backups

Otro apartado de seguridad importante, aún que lo comente a parte, se trata de tener copias de seguridad de nuestras aplicaciones y bases de datos, por si en un futuro surge algún problema, poder restaurar dichas copias y volver a la normalidad lo antes posible.

Dichas copias de seguridad han sido solicitados en formato de script, cosa que en mi servidor procurará que fuera con algún tipo de software más profesional ya que si cometí algún error en mi código podría quedarme sin datos. Por este motivo no lo he implementado en el servidor, sino que mantenido dichos scripts en mi máquina personal.

Backup BDD

Para realizar los respaldos de la base de datos, he utilizado mysqldump en un script, el cual permite realizar copias de las bdd con un montón de opciones. En mi caso, he realizado una para la tabla usuarios (la única realmente importante) y otro script para la base de datos en general, por si algún casual no pudiera obtener los datos de la API REST.

Aquí podemos ver unos ejemplos del código:

```
mysqldump -u root projectedaw users > "C:\Users\Dylan
Garrido\Desktop\ProjecteFinal\Sistemes Informatics\Backups
Scripts\BDD\backupUsers.sql"
```

Y aquí del resultado:

```
LOCK TABLES `users` WRITE;
/*!40000 ALTER TABLE `users` DISABLE KEYS */;
INSERT INTO `users` VALUES (1,'Dylan Garrido','dylanhurtos@gmail.com',NULL,'$2y$10$heORNF
/*!40000 ALTER TABLE `users` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME ZONE=@OLD TIME ZONE */;
```

Aquí el script ejecutable:

Nombre	Fecha de modificación	Tipo	Tamaño
backupAll	27/05/2020 22:50	Archivo SQL	3.279 KB
BackupAllBDD	27/05/2020 22:49	Documento de tex...	1 KB
BackupAllBDDScript	27/05/2020 22:50	Archivo por lotes ...	1 KB
backupUsers	27/05/2020 22:51	Archivo SQL	3 KB
BackupUsersBDD	27/05/2020 22:48	Documento de tex...	1 KB
BackupUsersBDDScript	27/05/2020 22:48	Archivo por lotes ...	1 KB

Backup Proyecto

Por otro lado tendríamos el respaldo del proyecto entero, por si se diera el caso de que borrarse alguna cosa que no debía o bien, el servidor se estropeará poder tener alguna copia actualizada.

Aquí podemos observar el código del script que mediante robocopy colona el proyecto de un lugar a otro y luego con compact lo comprime:

```
ROBOCOPY C:\xampp\htdocs\ProjecteFDAW "C:\Users\Dylan
Garrido\Desktop\ProjecteFinal\Sistemes Informatics\Backups Scripts\Projecte" /e /mir
/np /log:backup_log.txt
compact /c ProjecteFDAW
```

A continuación podemos ver el script ejecutable y el zip que se genera.

Nombre	Fecha de modificación	Tipo	Tamaño
BackupProjecte	27/05/2020 23:26	Documento de tex...	1 KB
BackupProjecteScript	27/05/2020 23:26	Archivo por lotes ...	1 KB
publish	27/05/2020 23:16	Archivo WinRAR ZIP	46.079 KB

Con esto concluiríamos el apartado de sistemas informáticos, donde os he mostrado el tipo de maquinaria usada y los sistemas de seguridad implementados tanto de forma automática como manual.

M02: Base de datos

Introducción

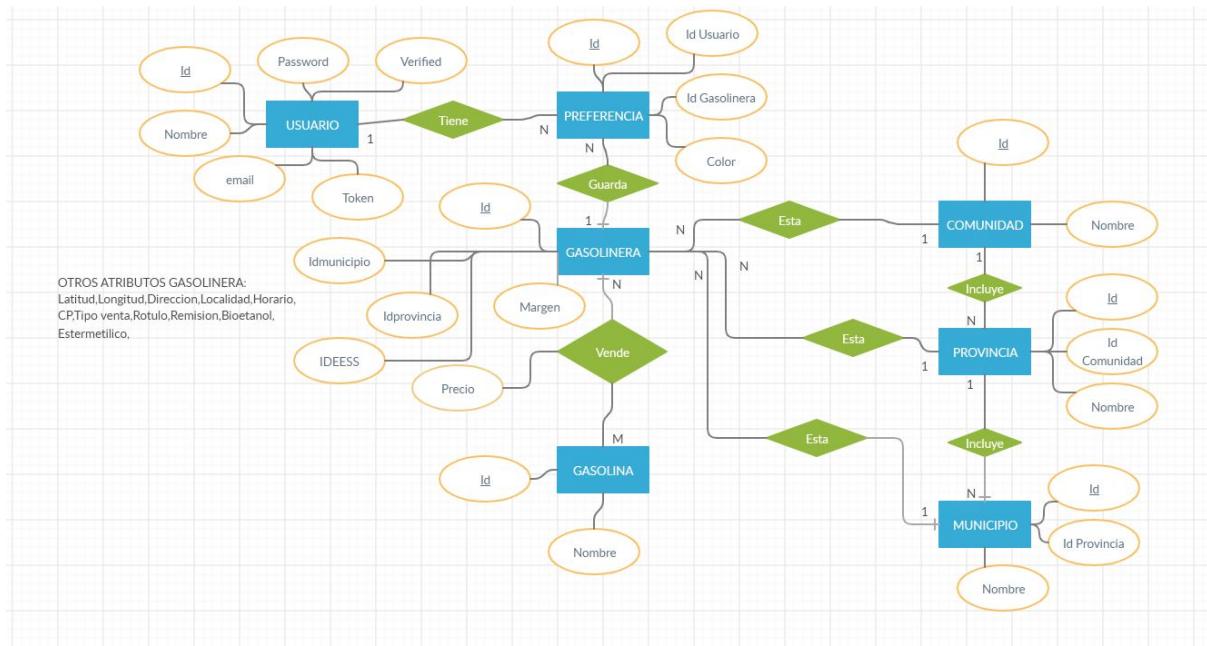
En esta sección comentaré como he planeado la base de datos y alteraciones que he realizado en función del tiempo y las funciones que proporcionaba la aplicación web.

Empezaré por comentar que he utilizado el gestor de base de datos relacional MYSQL (versión 7.4) complementado con la herramienta phpmyadmin (versión 5) con la he podido gestionar diversos apartados de la base de datos a través del navegador web con una interfaz más visual y práctica para el usuario.

Modelo E-R

En primer lugar nos encontramos con la realización del esquema Entidad-Relación en cual facilita la representación de entidades de una base de datos.

En nuestra base de datos podemos encontrar 7 entidades que son las siguientes:



Los usuarios tienen como preferencias determinadas gasolineras donde venden unos carburantes específicos con un precio concreto.

Por otro lado encontramos que las gasolineras se encuentran en una comunidad, municipio y provincia específicas y estas pueden tener muchas gasolineras en su localización.

Modelo Relacional

El modelo relacional representa una base de datos como una colección de relaciones (tablas). Permite una representación lógica y no física lo que equivale a la independencia de los datos.

Mencionar que la herramienta de google drive no permite la personalización de fuentes ni los diversos subrayados, por lo que las claves foráneas, están de color naranja

En cuanto a la transformación del Modelo Entidad-Relación, comentar que la mayoría de relaciones equivalen a 1:N, por lo que la clave primaria del lado “1”, pasa como llave foránea en el lado “N”.

Por otro lado podemos encontrar una relación N:M (Gasolineras/Gasolina) con un atributo de relación. En este caso se crea una tabla nueva, en la que las llaves primarias pasan a ser foráneas en la nueva tablas, pero, que con la unión de las 2 pasan a ser primarias. En esta nueva tabla también aparece el atributo de relación precio.

USUARIO (Id,Nombre,Password,Email,Verified,Token)

PREFERENCIA(Id,Color,Id_usuario,id_gasolinera)

GASOLINERA(Id,CP,Dirección,Horario,Latitud,Longitud,Localidad,Margen,Remisión,Rótulo,
Tipo venta,Bioetanol,Estermetilico,id_comunidad, id_provincia,id_municipio)

GASOLINERA-GASOLINA(Id_gasolinera,Id_gasolina,Precio)

GASOLINA(Id,Nombre)

COMUNIDAD(Id,Nombre)

PROVINCIA(Id,Id_comunidad,Nombre)

MUNICIPIO(Id,Id_provincia,Nombre)

Trigger

Un trigger o disparador es un objeto que se asocia con tablas y se almacena en la base de datos. Su nombre se deriva por el comportamiento que presentan en su funcionamiento, ya que se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado.

En mi caso, he realizado un trigger que registra a los usuarios, el id del elemento borrado y la fecha actual al eliminar un elemento de la tabla de gasolineras, que es la principal tabla de la que se suстраe información en la aplicación web.

Lo primero que he creado ha sido la tabla donde se almacenarán los datos del usuario que ha borrado contenido de la tabla de gasolineras.

Nom	Tipus	Longitud/Valors*	Per defecte	Col·lació	Atributs	Nul	Índex	A_L	Comentaris
id_gasolinera	BIGINT	20	Cap						
fecha_eliminacion	DATE		Cap						
usuario	VARCHAR	50	Cap						
	INT		Cap						
Estructura									
Comentaris de la taula:	Col·lació:			Motor d'emmagatzematge: 					

Como podemos observar en la imagen, en la nueva tabla inserto el id de la gasolinera borrada, por otro lado la fecha (de cuando se ha borrado) y finalmente el usuario que ha realizado tal acción.

A continuación creamos el trigger, con las condiciones que hemos comentado antes:

```
MySQL ha retornat un conjunt buit (és a dir, zero files). (la consulta ha trigat 0,0283 segons.)  

CREATE TRIGGER gasolineras_after_delete AFTER DELETE ON gasolineras FOR EACH ROW BEGIN DECLARE vuser varchar(50); -- Guardamos el nombre del usuario que ha realizado el delete en una variable SELECT USER() INTO vuser; -- Guardamos los datos en una tabla INSERT INTO auditoria_gasolineras ( id_gasolinera, fecha_eliminacion, usuario ) VALUES ( OLD.id, SYSDATE(), vuser ); END;  

[Edita en línia] [ Edita ] [ Crea el codi PHP ]
```

Aquí podemos observar el código utilizado, donde guardamos en una variable el nombre del usuario para luego introducirlo en el insert:

DELIMITER //

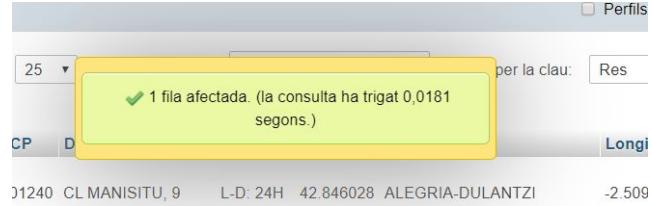
```

CREATE TRIGGER gasolineras_after_delete
AFTER DELETE ON gasolineras FOR EACH ROW
BEGIN
DECLARE vuser varchar(50);
-- Guardamos el nombre del usuario que ha realizado el delete en una variable
SELECT USER() INTO vuser;
-- Guardamos los datos en una tabla
INSERT INTO auditoria_gasolineras
(id_gasolinera,fecha_eliminacion,usuario)
VALUES
( OLD.id,SYSDATE(),vuser );
END; //

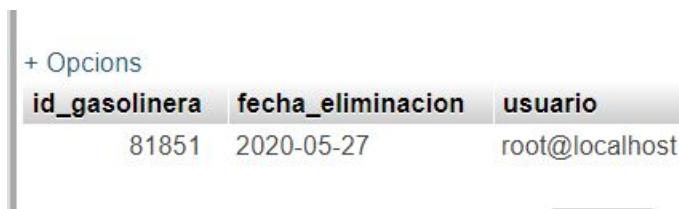
```

DELIMITER ;

A continuación, eliminamos un elemento de la tabla gasolineras para verificar que el trigger funciona correctamente.



Finalmente si nos dirigimos a la nueva tabla que hemos creado anteriormente, podremos ver que se ha registrado todo de forma correcta.



+ Opciones

id_gasolinera	fecha_eliminacion	usuario
81851	2020-05-27	root@localhost

Consultas

Ahora mostraré las 2 consultas que realzado, combinando JOINS con GROUP BY,ORDERS,HAVING etc.

La primera consulta busca en la base de datos mediante diversos JOINS que gasolineras que se encuentren en Cataluña->Girona->Figueres tiene gasolina 95 a un precio inferior a 1,1 ordenadas de forma ascendente por precio.

Aquí podemos ver una captura de la ejecución de la consulta:



The screenshot shows the results of a MySQL query execution. The results are displayed in a table with the following columns: rotulo, PrecioGasolina95, Nombre, p.nombre, and m.nombre. The data is as follows:

rotulo	PrecioGasolina95	Nombre	p.nombre	m.nombre
ESCLATOIL	1,019	Cataluña	GIRONA	Figueres
EROSKI	1,029	Cataluña	GIRONA	Figueres
CARREFOUR	1,069	Cataluña	GIRONA	Figueres

Y a continuación el código de la misma:

```
select g.rotulo,g.PrecioGasolina95,c.Nombre,p.nombre,m.nombre from comunidades c
LEFT JOIN provincias p on c.Id=p.id_comarca LEFT JOIN municipios m on
p.id=m.id_provincia LEFT JOIN gasolineras g ON c.Id=g.IDCCAA and p.id=g.IDProvincia and
m.id=g.IDMunicipio WHERE c.Nombre='Cataluña' and p.nombre='GIRONA' AND
m.nombre='Figueres' AND g.PrecioGasolina95<'1,1' ORDER by g.PrecioGasolina95 asc
```

La segunda consulta, busca en la base de datos las comunidades autónomas que tengan más de 25 gasolineras con gasolina 95 inferior a 1€/L. También nos muestra el total de gasolineras que cumplen los requisitos en cada comunidad. En este caso he utilizado JOINS, GROUP BY y HAVING

En la siguiente imagen vemos la correcta ejecución de la consulta y sus resultados:



Comunidad	Total
Andalucía	101
Aragón	58
Canarias	384
Castilla - La Mancha	94
Castilla y Leon	45
Cataluña	301
Galicia	39
Madrid	55
Murcia	77
Navarra	29
Valencia	173

A continuación podemos observar el código empleado para la consulta:

```
select c.Nombre as 'Comunidad' ,count(*) 'Total' from comunidades c LEFT JOIN
gasolineras g ON c.Id=g.IDCCAA WHERE g.PrecioGasolina95<'1' GROUP BY Comunidad
HAVING Total>25
```

Otros

Como temas a parte mencionar primero que junto a la documentación, puedes encontrar diversas carpetas ordenadas por módulos, en la carpeta de base de datos se puede encontrar un script (backup) de la base de datos entera, con toda la estructura y datos.

Por otro lado comentar que debido a la falta de tiempo la base de datos que se está usando no es 100% fiel al modelo realizado, ya que no existe la relación gasolineras-gasolina.

Si sigo con el proyecto en el futuro, el primer cambio que realizare es el de acabar de pulir la base de datos, ya que a 1 día de entregar el proyecto y con la documentación aún pendiente de acabar no me puedo permitir invertir más tiempo en ello.

M03: Programación

Introducción

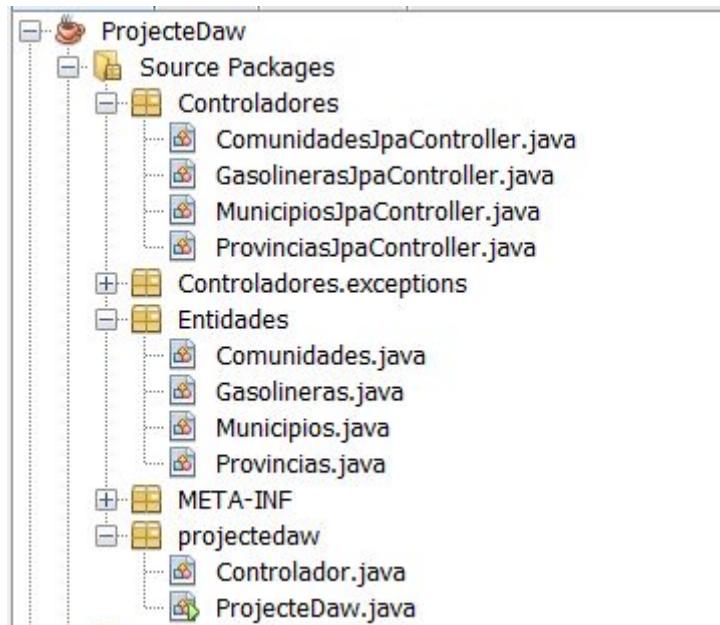
Para la realización del apartado de programación, he utilizado el conocido IDE de Netbeans juntamente con la herramienta de Hibernate, la cual permite interactuar con la base de datos de mysql.

La elección de Netbeans se debe a que proporciona muchas facilidades para trabajar con Hibernate, ya que genera automáticamente todas las entidades y controladores en función de la base de datos que estemos utilizando.

Este programa de java, realiza peticiones a una API REST y mediante la herramienta de Hibernate sube los datos a un servicio mysql propio.

Estructura

Para empezar, mostraré la estructura que podemos encontrar en el proyecto, donde podemos ver un paquete con los controladores, otro con las entidades y finalmente un paquete principal nombrado “projectedaw” en el que podemos encontrar la clase Main del proyecto y una clase complementaria donde se deriva parte del código para tener todo más limpio y estructurado



Código

Por otro lado si entramos en la clase complementaria “Controlador.java” anteriormente mencionada encontraremos una de las primeras funciones que se ejecutan en nuestro programa:

```
public static String peticionHttpGet(String urlParaVisitar) throws Exception {
    // Esto es lo que vamos a devolver
    StringBuilder resultado = new StringBuilder();
    // Crear un objeto de tipo URL
    URL url = new URL(urlParaVisitar);

    // Abrir la conexión e indicar que será de tipo GET
    HttpURLConnection conexion = (HttpURLConnection) url.openConnection();
    conexion.setRequestMethod("GET");
    conexion.setRequestProperty("Content-Type", "application/json; charset=utf-8");
    conexion.setRequestProperty("Accept", "application/json");
    // Búferes para leer
    BufferedReader rd = new BufferedReader(new InputStreamReader(conexion.getInputStream()));
    String linea;
    // Mientras el BufferedReader se pueda leer, agregar contenido a resultado
    while ((linea = rd.readLine()) != null) {
        resultado.append(linea);
    }
    // Cerrar el BufferedReader
    rd.close();
    // Regresar resultado, pero como cadena, no como StringBuilder
    return resultado.toString();
}
```

Esta función, al pasarle una url por parámetro (url de la API REST), se encarga de sustraer toda la información línea por línea con la utilización de un buffer, mientras que con el uso de un bucle while, vamos añadiendo las líneas una detrás de la otra, para al final obtener una sola variable con todo el contenido, que será la que devolverá la función.

Volviendo al documento Main del proyecto podemos encontrar varios apartados importantes en el inicio del código:

- Marcado en **verde(1)**, podemos encontrar la creación de variables de los controladores.
- En **naranja(2)**, podemos encontrar la variable con la url de la API REST.
- En **azul oscuro(3)**, podemos encontrar ArrayList para la gestión de los datos.
- Finalmente en **azul claro(4)**, la llamada a la función para obtener los datos de la url.

```

GasolinerasJpaController CGasolinera = new GasolinerasJpaController();
ComunidadesJpaController CComunidad = new ComunidadesJpaController();
MunicipiosJpaController CMunicipio = new MunicipiosJpaController();
ProvinciasJpaController CProvincia = new ProvinciasJpaController(); 1

String url = "https://sedeaplicaciones.mineco.es/ServiciosRESTCarburantes/PreciosCarburantes/EstacionesTerrestres/"; 2
String jsonString = "";
try {
    ArrayList<String> Comunidad = new ArrayList<String>();
    ArrayList<String> Provincia = new ArrayList<String>();
    ArrayList<String> Municipio = new ArrayList<String>(); 3
    String[] nomComun = {"Andalucia", "Aragon", "Asturias", "Baleares", "Canarias", "Cantabria", "Castilla y Leon", "Castilla - La
    ArrayList<Comunidades> ComunidadObj = new ArrayList<Comunidades>();
    ArrayList<Municipios> MunicipioObj = new ArrayList<Municipios>();
    ArrayList<Provincias> ProvinciaObj = new ArrayList<Provincias>(); 3

    jsonString = Controlador.peticionHttpGet(url); 4

    JSONObject obj = new JSONObject(jsonString);
    String fecha = obj.getString("Fecha");
    Date date1=new SimpleDateFormat("dd/MM/yyyy").parse(fecha);
}

```

Si seguimos revisando el código, podremos observar que antes de insertar datos, lo primero que hacemos es eliminar toda información de las tablas de la base de datos, para no tener problemas de duplicidad de claves primarias o redundancia de datos.

```

CGasolinera.deletetable();
CMunicipio.deletetable();
CProvincia.deletetable();
CComunidad.deletetable();

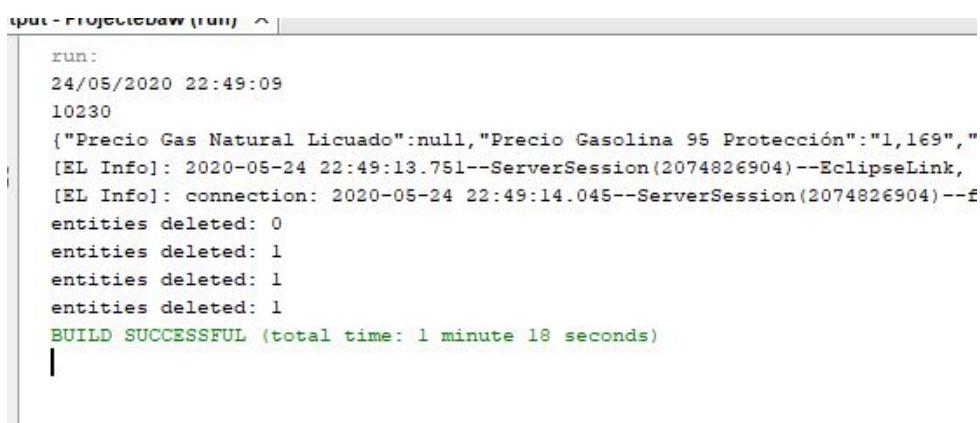
```

Finalmente, con un conjunto de iteradores como los “for” vamos insertando los datos en un orden específico, para que no haya problemas con la relación de las claves primarias y foráneas.

```
for (int i = 0; i < arr.length(); i++)  
{  
    Comunidades com = new Comunidades();  
    Gasolineras gas = new Gasolineras();  
    Municipios mun = new Municipios();  
    Provincias pro = new Provincias();  
  
    //Comunidades  
    if(Comunidad.contains(arr.getJSONObject(i).getString("IDCCAA"))){  
        int posicion=Comunidad.indexOf(arr.getJSONObject(i).getString("IDCCAA"));  
  
        com=ComunidadObj.get(posicion);  
    }  
}
```

Ejecución

Aquí podemos ver un ejemplo de la ejecución del programa:



```
run:  
24/05/2020 22:49:09  
10230  
{"Precio Gas Natural Licuado":null,"Precio Gasolina 95 Protección":"1,169","  
[EL Info]: 2020-05-24 22:49:13.751--ServerSession(2074826904)--EclipseLink,  
[EL Info]: connection: 2020-05-24 22:49:14.045--ServerSession(2074826904)--f  
entities deleted: 0  
entities deleted: 1  
entities deleted: 1  
entities deleted: 1  
BUILD SUCCESSFUL (total time: 1 minute 18 seconds)
```

Otros

Librerías Utilizadas:



Mencionar también, que para poder ejecutar el programa, hay que instalar un certificado de la propia web de la API REST, ya que usan SSL y sin el certificado, el programa no puede comunicarse con la web.

Este equipo > Escritorio > ProyectoFinal > Programación > Certificados			
Nombre	Fecha de modificación	Tipo	Tamaño
certificado	07/05/2020 23:04	Certificado de seg...	2 KB

Para instalar el certificado hay que usar este comando:

```
C:\Users\Dylan Garrido\Desktop>keytool -importcert -file certificado.cer -alias gasofa -keystore "%JAVA_HOME%\lib\security\cacerts"
```

Para el apartado de automatización, he compilado el proyecto de Netbeans en un .jar y posteriormente he creado una tarea programada de windows para que se ejecute 1 vez al día.

Finalmente, quiero indicar que debido a que el servidor cambia de ip constantemente cada vez que se inicia, el programa de netbeans tiene problemas para ejecutar el código apuntando al servidor. Por eso para dejarlo funcional, actualmente apunta a mi base de datos local y posteriormente copio la base de datos local a la desplegada vía ssh.

M04: Lenguaje de marcas

En la sección de este módulo, he realizado copias de seguridad de la base de datos en formato xml de la tabla usuarios, que al mismo tiempo, entiendo que este proceso cuente también como exportar datos en xml, tal y como se indica en la guía del proyecto.

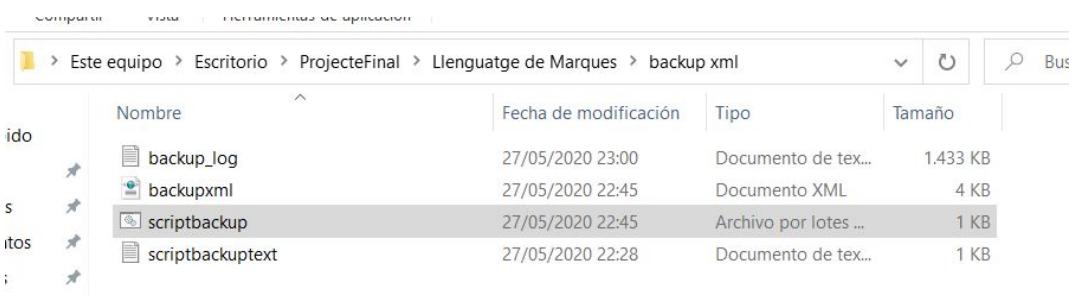
También he realizado un sistema que mediante la url que introduzcas, te devuelve por pantalla los datos de la tabla gasolinera en los distintos formatos mencionados (XML, JSON y CSV).

Exportar datos y copias BDD en XML

Para este apartado, he realizado una exportación (copia de datos) de la tabla usuarios de mi base de datos. Esta tabla es realmente la más importante porque sus datos son los únicos que no podría conseguir de otro modo a diferencia de la tabla gasolineras, provincias, comunidades y municipios, que puedo sustraer sus datos a partir de una API REST.

Para realizar dicha copia de seguridad, he realizado un script que al ejecutarlo utiliza un comando llamado mysqldump (si utilizas windows hay que descargarlo) el cual permite interactuar con la base de datos mysql y realizar una exportación de datos con los parámetros que le indiques.

Script ejecutable:



	Nombre	Fecha de modificación	Tipo	Tamaño
ido	backup_log	27/05/2020 23:00	Documento de tex...	1.433 KB
s	backupxml	27/05/2020 22:45	Documento XML	4 KB
itos	scriptbackup	27/05/2020 22:45	Archivo por lotes ...	1 KB
i	scriptbackuptext	27/05/2020 22:28	Documento de tex...	1 KB

Aquí podemos ver comando utilizado donde le indicamos el usuario de la base de datos y el parámetro --xml para que realice la copia en este formato:

```
mysqldump --xml -u root projectedaw users > "C:\Users\Dylan
Garrido\Desktop\ProjecteFinal\Llenguatge de Marques\backup xml\backupxml.xml"
```

El resultado de ejecutar dicho script es un documento xml, que si lo abrimos, podremos observar cómo ha creado un backup de la tabla users en formato xml.



```

<?xml version="1.0"?>
- <mysqldump xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <database name="projectedaw">
    - <table_structure name="users">
      <field Comment="" Extra="auto_increment" Key="PRI" Null="NO" Type="bigint(20) unsigned" Field="id"/>
      <field Comment="" Extra="" Key="" Null="NO" Type="varchar(255)" Field="name"/>
      <field Comment="" Extra="" Key="UNI" Null="NO" Type="varchar(255)" Field="email"/>
      <field Comment="" Extra="" Key="" Null="YES" Type="timestamp" Field="email_verified_at" Default="NULL"/>
      <field Comment="" Extra="" Key="" Null="NO" Type="varchar(255)" Field="password"/>
      <field Comment="" Extra="" Key="" Null="YES" Type="varchar(100)" Field="remember_token" Default="NULL"/>
      <field Comment="" Extra="" Key="" Null="YES" Type="timestamp" Field="created_at" Default="NULL"/>
      <field Comment="" Extra="" Key="" Null="YES" Type="timestamp" Field="updated_at" Default="NULL"/>
      <key Comment="" Null="" Index_comment="" Index_type="BTREE" Cardinality="4" Collation="A" Column_name="id" Seq_in_index="1" Key_name="PRIMARY" Non_unique="0" Table="users"/>
      <key Comment="" Null="" Index_comment="" Index_type="BTREE" Cardinality="4" Collation="A" Column_name="email" Seq_in_index="1" Key_name="users_email_unique" Non_unique="0" Table="users"/>
      <options Comment="" Collation="utf8mb4_unicode_ci" Temporary="N" Max_index_length="0" Create_options="" Update_time="2020-05-24 20:35:25" Create_time="2020-05-20 20:29:03" Auto_increment="5" Data_free="0" Index_length="16384" Max_data_length="0" Data_length="16384" Avg_row_length="4096" Rows="4" Row_format="Dynamic" Version="10" Engine="InnoDB" Name="users"/>
    </table_structure>
  ...

```

Como podemos ver ha creado correctamente toda la estructura de datos y a continuación podremos ver algún usuario.

```

</table_structure>
- <table_data name="users">
  - <row>
    <field name="id">1</field>
    <field name="name">Dylan Garrido</field>
    <field name="email">dylanhurtos@gmail.com</field>
    <field name="email_verified_at" xsi:nil="true"/>
    <field name="password">$2y$10$heORNfYXB9TKijunzF2PcurlZ7FbZRQfxWRImub8jQ3EtOF0AjQObq</field>
    <field name="remember_token" xsi:nil="true"/>
    <field name="created_at">2020-05-24 18:23:44</field>
    <field name="updated_at">2020-05-24 18:23:44</field>
  </row>
  - <row>
    <field name="id">2</field>

```

APIS

El apartado de apis que se menciona en la guía del proyecto, ha sido realizado en mi máquina real pero no ha sido incluido en el servidor. Considero que no es necesario que el usuario tenga 3 rutas una para cada formato de la request (JSON, XML,CSV). Ciento es que en la versión final desplegada, hay una ruta api que muestra los datos de la tabla gasolineras en formato JSON.

Mientras tanto, en mi maquina real si que he realizado el apartado que se comentaba en la guía, lo he realizado con node js, con el cual he establecido conexión con mi base de datos y posteriormente con ayuda de algunas librerías, he ido mostrando los datos en los distintos formatos solicitados.

En primer lugar he creado un proyecto node con el siguiente comando:

```
npm init -y
```

Acto seguido, hemos instalado los siguientes módulos y los he instanciado en el proyecto de node.

```
const mysql = require('mysql');
const express = require("express");
const path = require("path");
const js2xmlparser = require("js2xmlparser");
const converter = require('json-2-csv');
```

Después, he creado una variable para la librería de express y otra variable para almacenar el puerto donde quiero que se ejecute el código, en este caso el 8080.

```
const app = express();
const port = process.env.PORT || "8080";
```

Ahora toca el paso de establecer conexión con la base de datos, para ello he creado una constante “connection” que contenga los datos para entrar a mysql. Como por ejemplo el host, el usuario y la base de datos.

```
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'projectedaw'
});
connection.connect((err) => {
  if (err) throw err;
  console.log('Connected!');
});
```

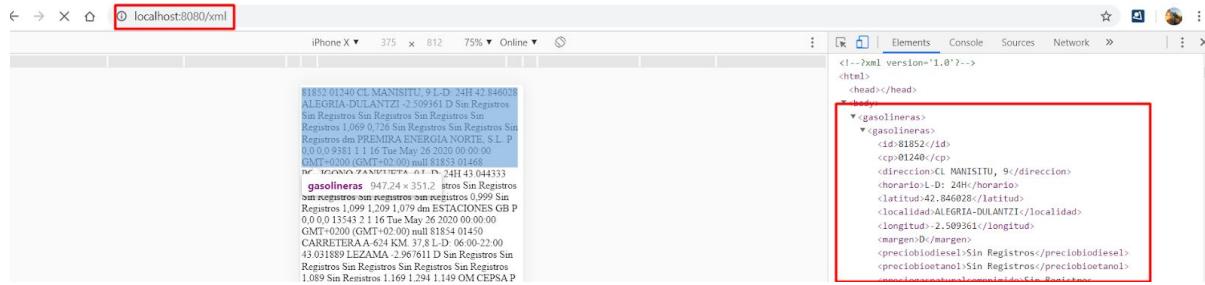
API XML

Para la Api en formato xml, he creado una ruta “/xml”, que cuando se accede al navegador con ella, nos devuelve los datos de las gasolineras en formato xml gracias a la librería “js2xmlparser”, que se encarga de convertir código JSON a XML

```
app.get("/xml", (req, res) => {
  connection.query('SELECT * FROM gasolineras', (err,rows) => {
    if(err) throw err;

    console.log('Datos Recibidos de la BDD');
    res.status(200).send(js2xmlparser.parse("gasolineras", rows));
  });
});
```

A continuación podemos ver una captura, cuando entramos por el navegador con esa ruta.



Como podemos observar, el usuario no ve las etiquetas xml.

Eso se debe a que el navegador las interpreta y muestra su contenido, si nos fijamos en el inspector de elementos, podemos ver como las etiquetas de los datos son xml.

API JSON

Para realizar la api en formato JSON no he tenido muchas complicaciones, ya que el propio node al realizar peticiones a la base de datos te devuelve el contenido en formato JSON, por lo que simplemente he tenido que mostrar el resultado.

```

app.get("/json", (req, res) => {
  connection.query('SELECT * FROM gasolineras', (err,rows) => {
    if(err) throw err;

    console.log('Datos Recibidos de la BDD');
    res.status(200).send(rows);
  });
});
  
```

si entramos a la ruta “/json” podremos ver los datos de las gasolineras en este formato, lo bueno es que al ser el formato por defecto que utiliza node, la página carga con relativa velocidad.

A continuación podemos observar como si entramos al navegador con el nombre de host local y la ruta “/json”, obtendremos los datos de la tabla gasolineras en este formato.

API CSV

Finalmente, para realizar la api que devuelva los datos en formato csv, he tenido que realizar un procedimiento similar al de la api XML.

En primer lugar creamos la ruta y hacemos la query a la base de datos, el resultado que nos devuelva, lo convertimos de JSON a CSV con la librería de “json2csv” y lo mostramos por pantalla.

```
app.get("/csv", (req, res) => {
  connection.query('SELECT * FROM gasolineras', (err,rows) => {
    if(err) throw err;

    converter.json2csv(rows, (err, csv) => {
      if (err) {
        throw err;
      }

      // print CSV string
      res.status(200).send(csv);
    });
  });
});
```

Ahora, si entramos al navegador y introducimos en la barra de búsqueda el host con la ruta “/csv” podremos observar el contenido de la tabla gasolineras en este formato.

Responsive		670	x	412	150%	Online	⋮
localhost:8080/csv							

Otros

Primero de todo mencionar que en la documentación que he entregado, hay una carpeta de este módulo, donde podrás encontrar las capturas del resultado de las apis (por si no se ven bien aquí) y el código de node, por si quieres revisarlo y hacer pruebas.

Por otro lado el generar un RSS en mi proyecto no ha sido posible ya que apenas he utilizado código xml de forma funcional para mi aplicación web.

De todos modos ya hablamos contigo Albert, de que esta parte no era necesaria realizarla.

M05: Entornos de desarrollo

Introducción

A continuación mostraré los diversos diagramas solicitados juntamente con los test unitarios de la página tal y como se pide en la guía del proyecto.

Diagrama de casos de uso

Es una forma de diagrama de comportamiento UML mejorado. El Lenguaje de Modelado Unificado, define una notación gráfica para representar casos de uso llamada modelo de casos de uso.

Como podemos ver el diagrama, se observa las interacciones que puede realizar tanto el sistema como el usuario (Izquierda Usuario y derecha Sistema).

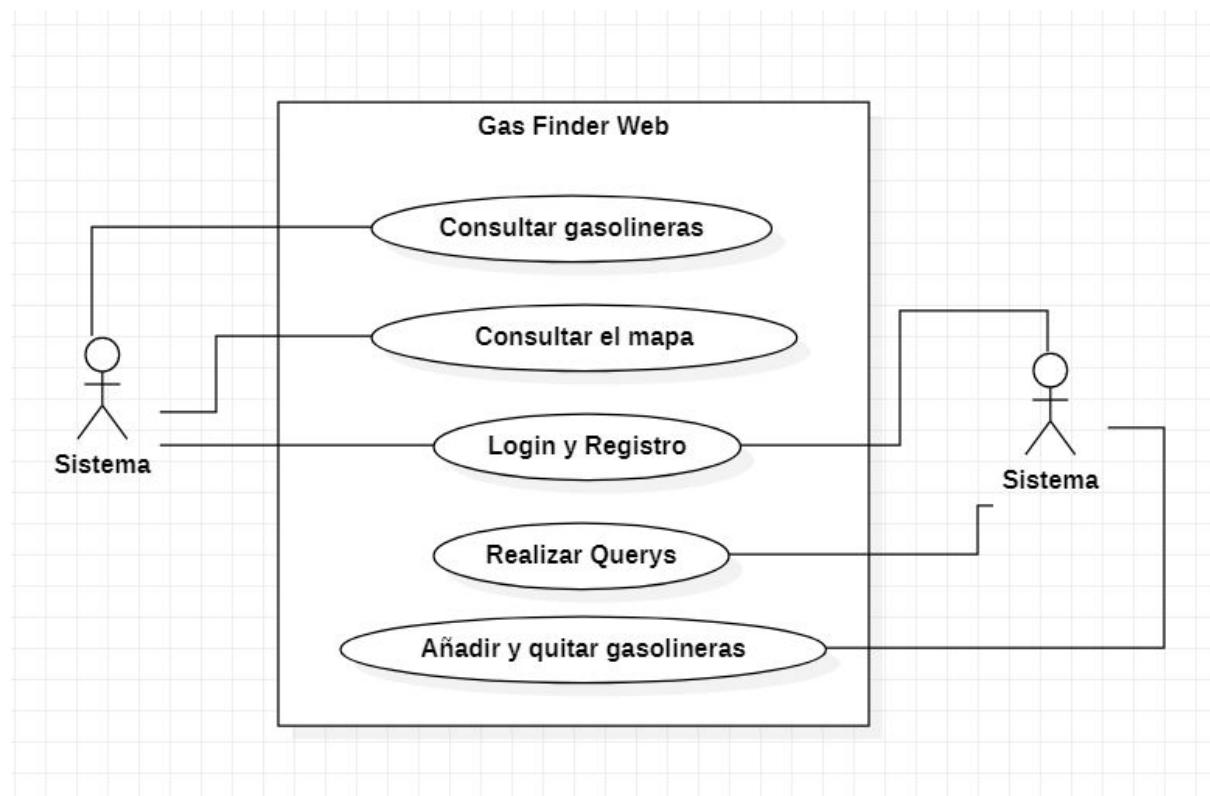


Diagrama de actividades

Es la representación gráfica de un algoritmo o proceso.

En este caso observamos las distintas actividades que puede realizar el usuario y el sistema desde que el cliente entra en la página hasta que visualiza el contenido deseado.

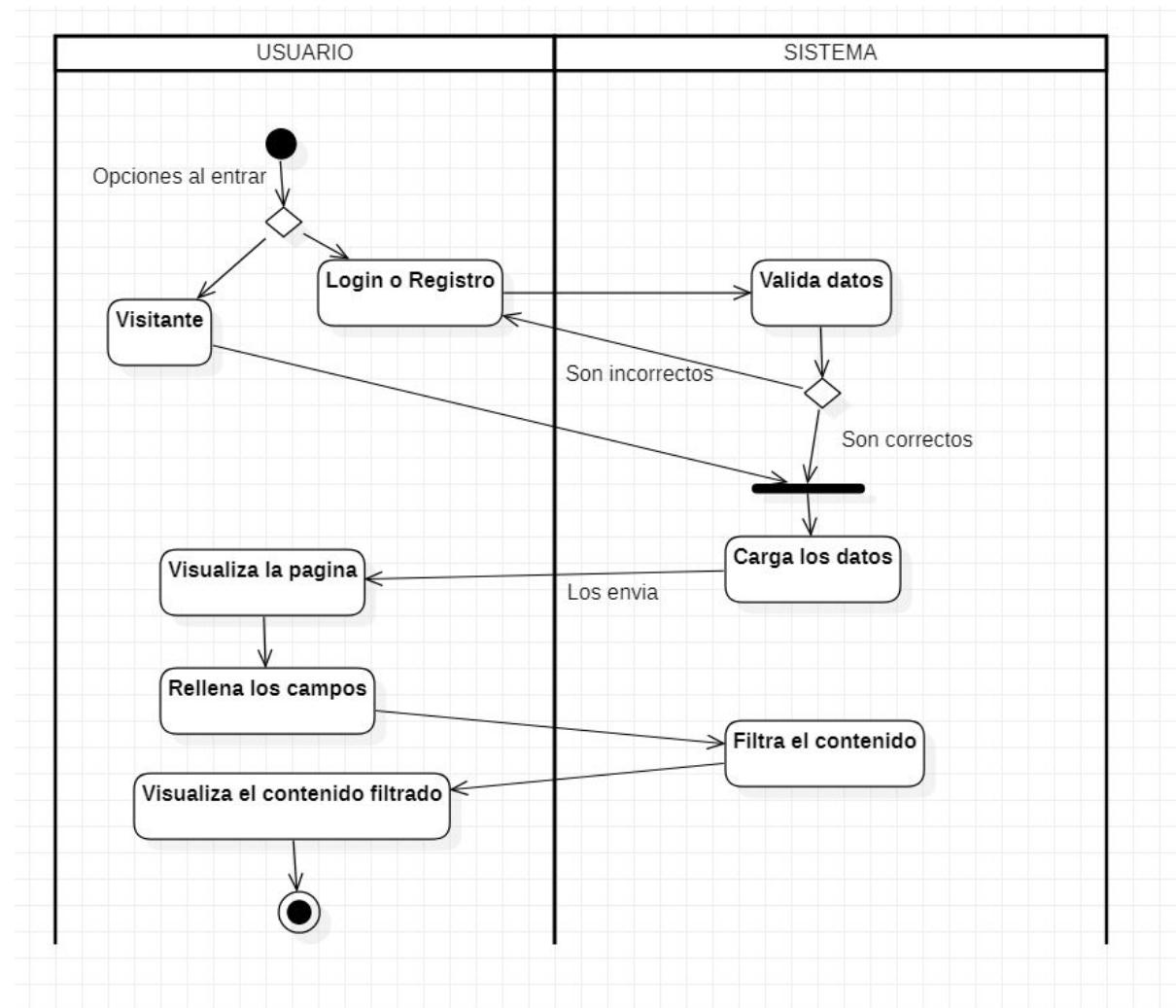


Diagrama de estados

Es un tipo de diagrama utilizado en informática y campos relacionados para describir el comportamiento de los sistemas.

En este diagrama podemos observar los distintos estados de nuestra aplicación web.

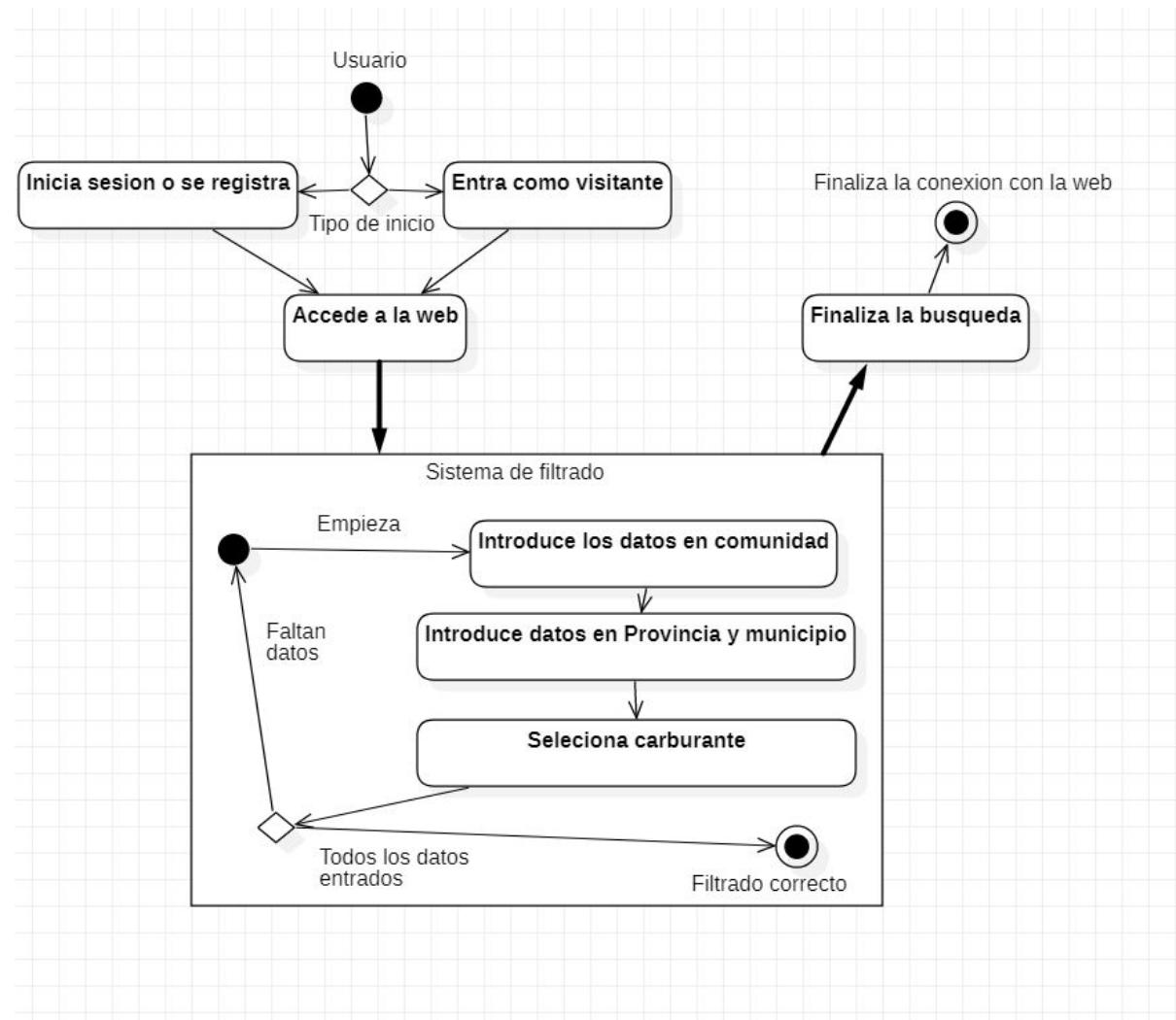


Diagrama de secuencia

Es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML.

Podemos observar las distintas secuencia que se realizan entre el usuario y el sistema.

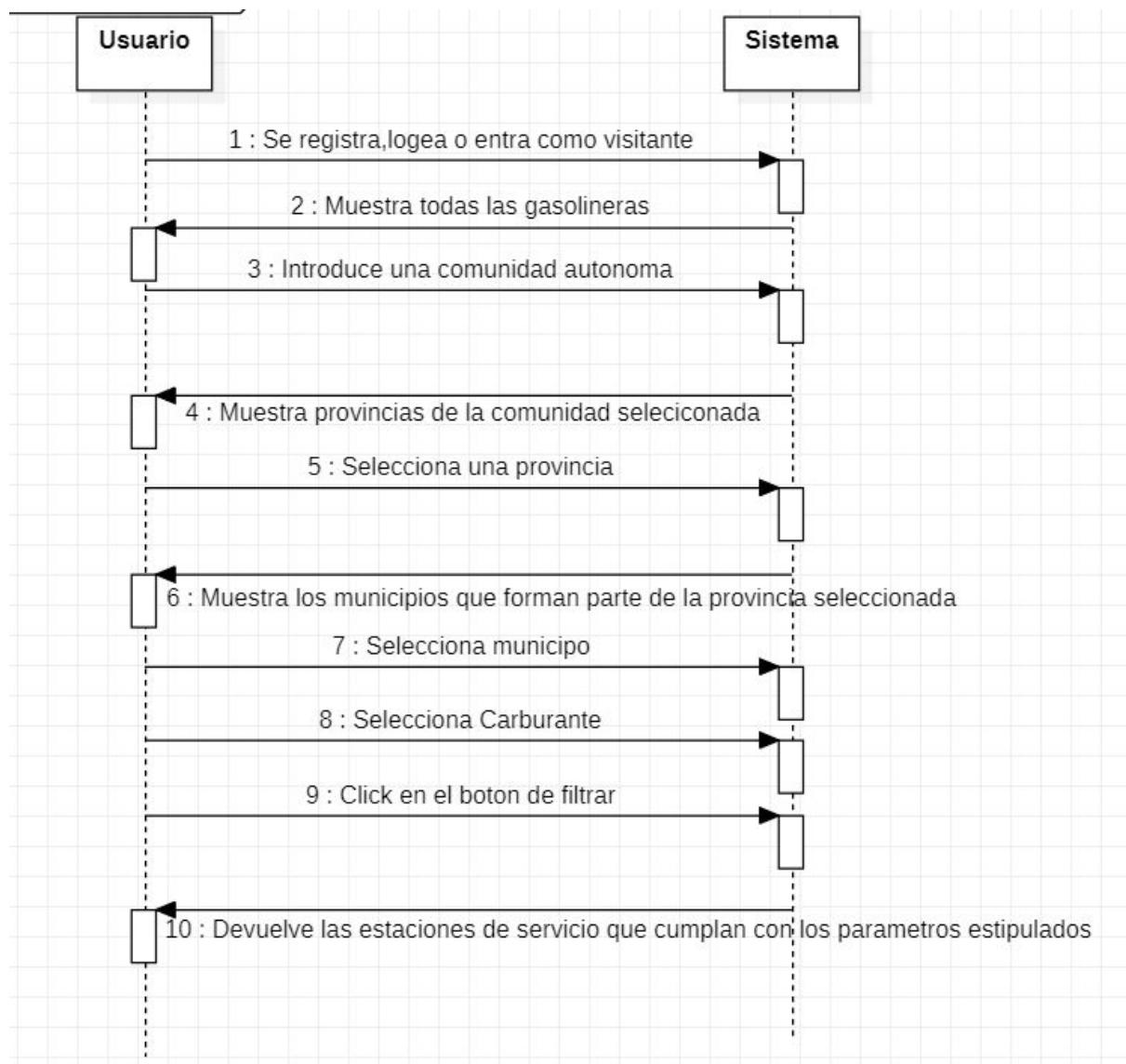


Diagrama de clases

Es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones, y las relaciones entre los objetos.

En este diagrama podemos observar la estructura que sigue Gas Finder, los atributos que utiliza, sus relaciones y operaciones entre otro.

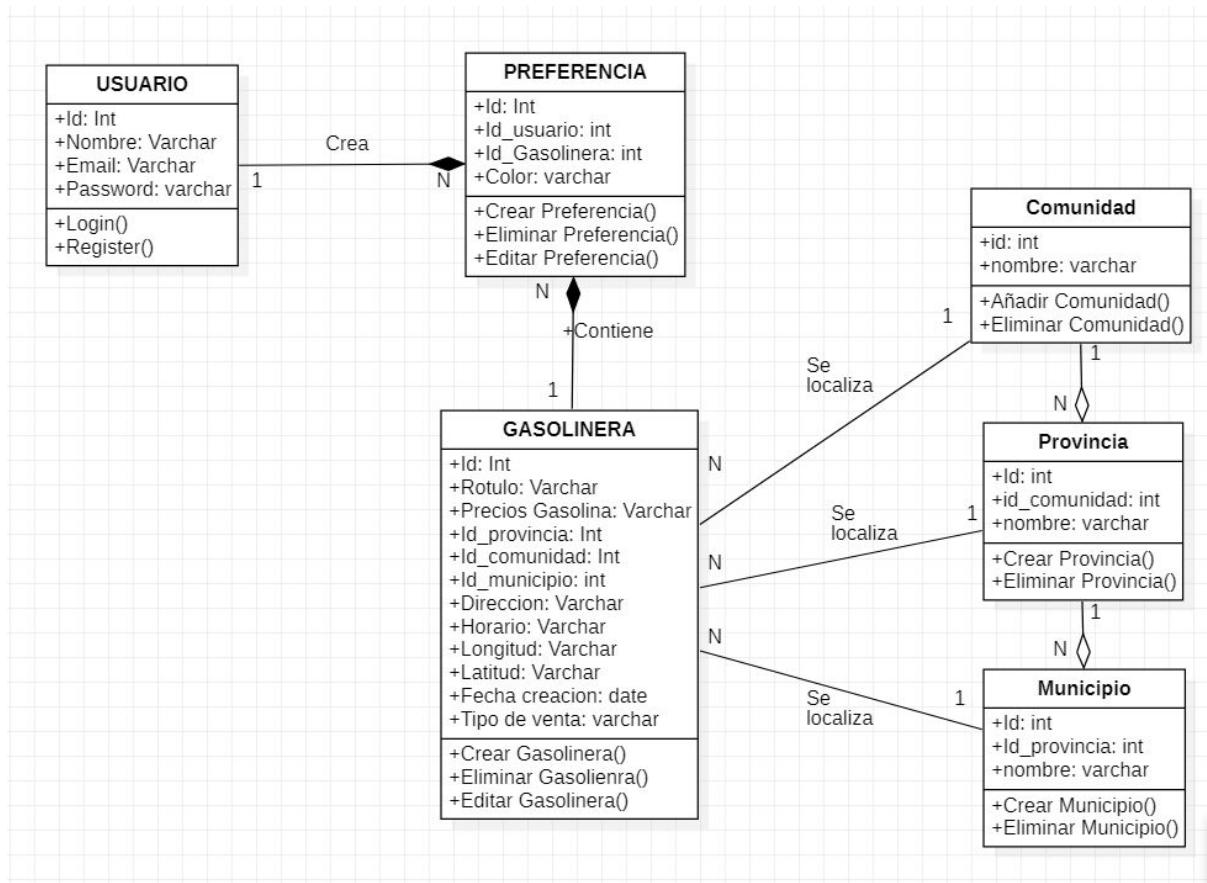
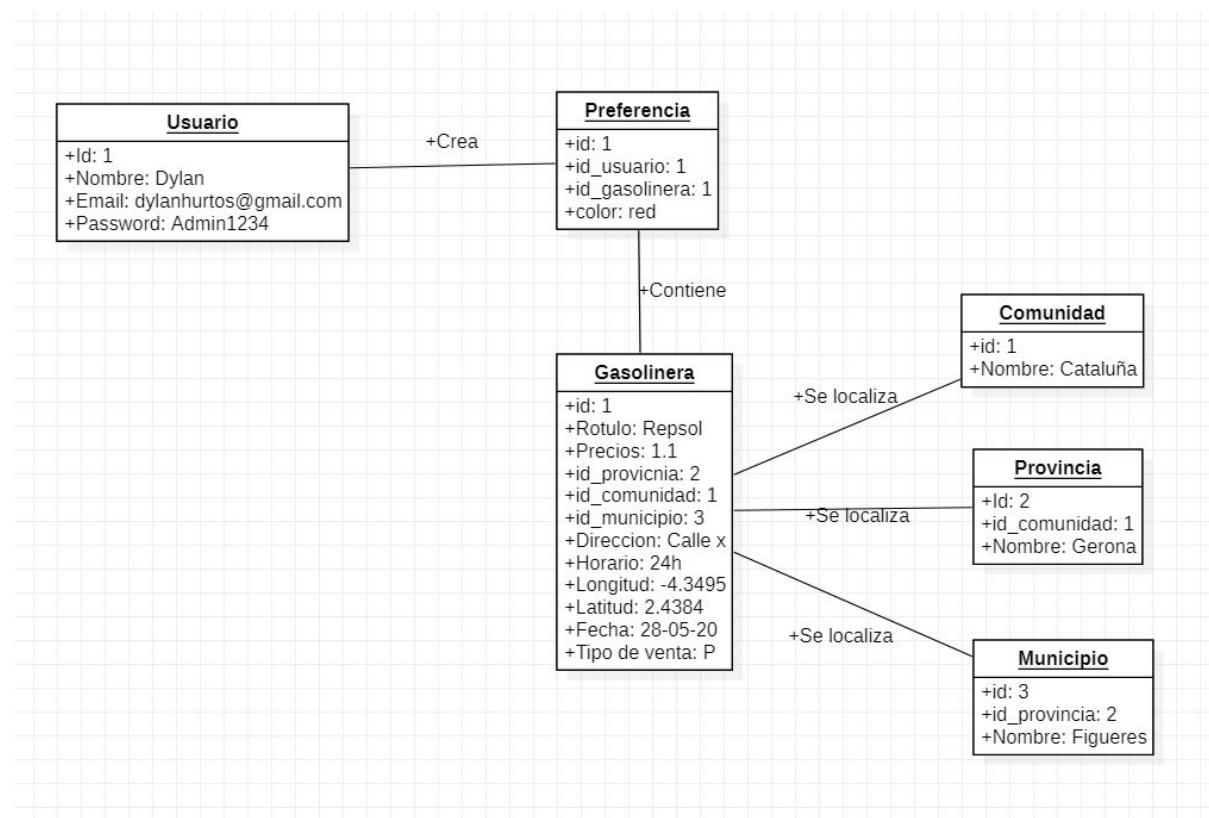


Diagrama de objetos

Es un diagrama que muestra una vista completa o parcial de los objetos de un sistema en un instante de ejecución específico.

Podríamos decir que es una versión del diagrama de clases donde todos los atributos están instanciados en un momento en particular de la aplicación web.



TDD (Test driven development)

Los TDD son un método para realizar pruebas automatizadas del código realizado, con el que podemos revisar si funciona de forma correcta.

Para realizar el TDD en mi caso he utilizado Selenium, que es una API que mediante ciertos comandos permite interactuar con un navegador web de modo que se pueda simular el uso de una aplicación web por parte del usuario final. Es indicada para pruebas de tipo funcionales.

Mi pagina web no tiene muchas funciones ni contenido, por lo que he realizado una simple prueba en la que selecciono una ubicación específica y realizo la simulación de filtrado, en la que podemos ver las estaciones de servicio que forman parte de esa localización.

A continuación podemos ver el código realizado:

```
    @Test
    public void testGooglePage() throws InterruptedException {
        Actions action = new Actions(driver);
        JavascriptExecutor jse = (JavascriptExecutor)driver;

        //Seleccionar Comunidad
        Select comarca = new Select(driver.findElement(By.id("comarca")));
        Thread.sleep(3000);
        comarca.selectByVisibleText("Cataluña");
        Thread.sleep(5000);

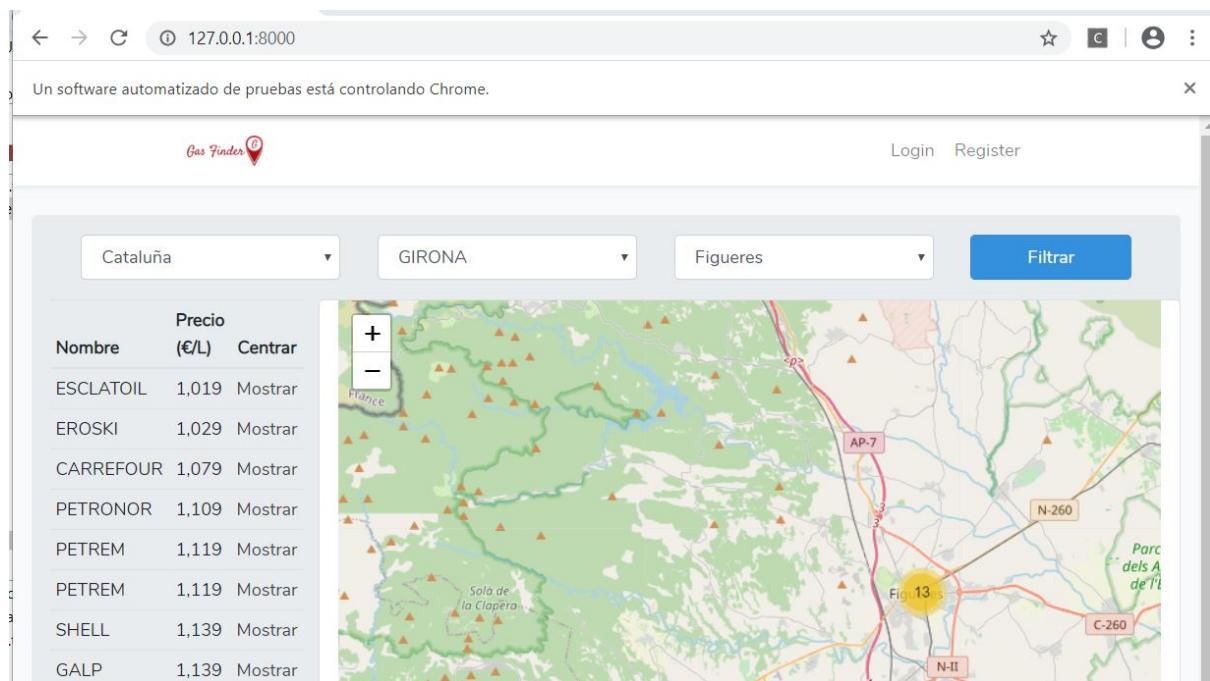
        //Seleccionar provincia
        Select provincia = new Select(driver.findElement(By.id("provincia")));
        Thread.sleep(3000);
        provincia.selectByVisibleText("GIRONA");
        Thread.sleep(5000);

        //Seleccionar municipio
        Select municipio = new Select(driver.findElement(By.id("municipio")));
        Thread.sleep(3000);
        municipio.selectByVisibleText("Figueres");
        Thread.sleep(5000);

        //Buscar y clicar boton por su nombre filtrar
        driver.findElement(By.xpath("//button[text()='Filtrar']")).click();
        Thread.sleep(5000);
    }

    @After
}
```

Y aquí el resultado final que podemos obtener en el navegador:



The screenshot shows a web browser window with the URL 127.0.0.1:8000. A message at the top says "Un software automatizado de pruebas está controlando Chrome." The page title is "Gas Finder". There are three dropdown menus: "Cataluña", "GIRONA", and "Figueres". A blue "Filtrar" button is located to the right of the dropdowns. To the left of the map, there is a table with the following data:

Nombre	Precio (€/L)	Centrar
ESCLATOIL	1,019	Mostrar
EROSKI	1,029	Mostrar
CARREFOUR	1,079	Mostrar
PETRONOR	1,109	Mostrar
PETREM	1,119	Mostrar
PETREM	1,119	Mostrar
SHELL	1,139	Mostrar
GALP	1,139	Mostrar

The map shows the area around Figueres, Spain, with several gas stations marked by red triangles. Roads labeled AP-7, N-260, C-260, and N-II are visible. A yellow circle highlights the location of Figueres.

M06: Desarrollo de aplicaciones en entorno cliente

Introducción

Para la implementación del contenido de este módulo, he utilizado el máximo de elementos que hemos aprendido este año, he intentado alternar código para no usar siempre Javascript puro y combinarlo con otras librerías como JQuery y DOM.

También he utilizado la librería de leaflet, que permite la creación de mapas y marcadores en función de las coordenadas que indiquemos. Básicamente leemos los datos recibidos de la bdd en formato JSON y los gestiono para poder representar todos los datos en pantalla sin que se vea todo muy colapsado, gracias al uso de clúster de puntos, que me permite agrupar los marcadores de zonas muy concurridas, esto sumado a mi toque personal, he conseguido que los separe por provincias.

Estructura

Para realizar la estructuración del contenido, he utilizado una herramienta que proporciona el propio laravel que permite separar el contenido de un documento en secciones para posteriormente al compilarlo, cada sección de código vaya a su lugar correspondiente.

Para la sección de estilos (styles) podemos ver cómo hemos incorporado los diversos estilos de leaflet, ya sea para simplemente crear el mapa o para poder crear los clúster de marcadores.

```
@section('styles')
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.3.1/dist/leaflet.css"
      integrity="sha512-Rksm5RenBEKSKFjgI3a41vrjkw4EVPlJ3+OII65vTjIdo9br1AacEuKOiQ50Fh7coI1bkDwLqdLw3Zg0cRJAQ==">
<link rel="stylesheet" href="https://unpkg.com/leaflet.markercluster@1.4.1/dist/MarkerCluster.css" />
<link rel="stylesheet" href="https://unpkg.com/leaflet.markercluster@1.4.1/dist/MarkerCluster.Default.css" />
```

Por otro lado tenemos el apartado de scripts donde incluiremos todo el apartado de javascript tanto nuestro como externo.

```
@push('scripts')
<!-- Make sure you put this AFTER Leaflet's CSS -->
<script src="https://unpkg.com/leaflet@1.3.1/dist/leaflet.js"
       integrity="sha512-/Nsx9X4Hebav0BvEBuyp3I7od5tA0UzAxs+j83KgC8PU0kgB4XiK4Lfe4y4cgBtaRJQEIFCW+oC506aPT2L1zw=
       crossorigin=""></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet.markercluster/1.4.1/leaflet.markercluster.js">
<script src="{{ asset('js/clustermarkersVAR.js') }}"></script>
<script>

$( "#provincia" ).prop( "disabled", true );
$( "#municipio" ).prop( "disabled", true );


```

Código

Esta sección la voy a dividir en diversos apartados, en función de lo que el código realice. Empezaremos por la mecánicas principales del mapa hasta finalmente llegar a los pequeños detalles.

Mapa

Como ya hemos mencionado antes, para la creación del mapa utilizamos la librería de leaflet, una vez la librería esté incorporada en nuestro código, crearemos un nuevo mapa indicando que las coordenadas se centren en Madrid (para tener el mapa centrado).

También crearemos un array de clúster con 52 clúster de marcadores, que equivale al nombre total de provincias que tiene España:

```
var map = L.map('mapid').setView([40.4165001, -3.7025599], 6);
var baseUrl = "{{ url('/') }}";

L.tileLayer('https://{{baseUrl}}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    minZoom: 6,
    maxZoom: 15,
    attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);

var arrayClusters = new Array(52);

for(var i=0;i<arrayClusters.length;i++){
    arrayClusters[i] = new L.MarkerClusterGroup({disableClusteringAtZoom: 15});
}
```

A continuación, utilizaremos la herramienta de AXIOS, que sería un homólogo al conocido AJAX. Con esta herramienta, llamaremos a una ruta estipulada en laravel que nos devolverá un listado con toda la información de todas las gasolineras en formato JSON.

Recorremos todos los datos uno por uno y en función del ID de la provincia lo asignaremos al clúster del array con el mismo índice que nos devuelva el ID. De este modo nos aseguramos que en cada posición del array sólo contenga marcadores de una sola provincia

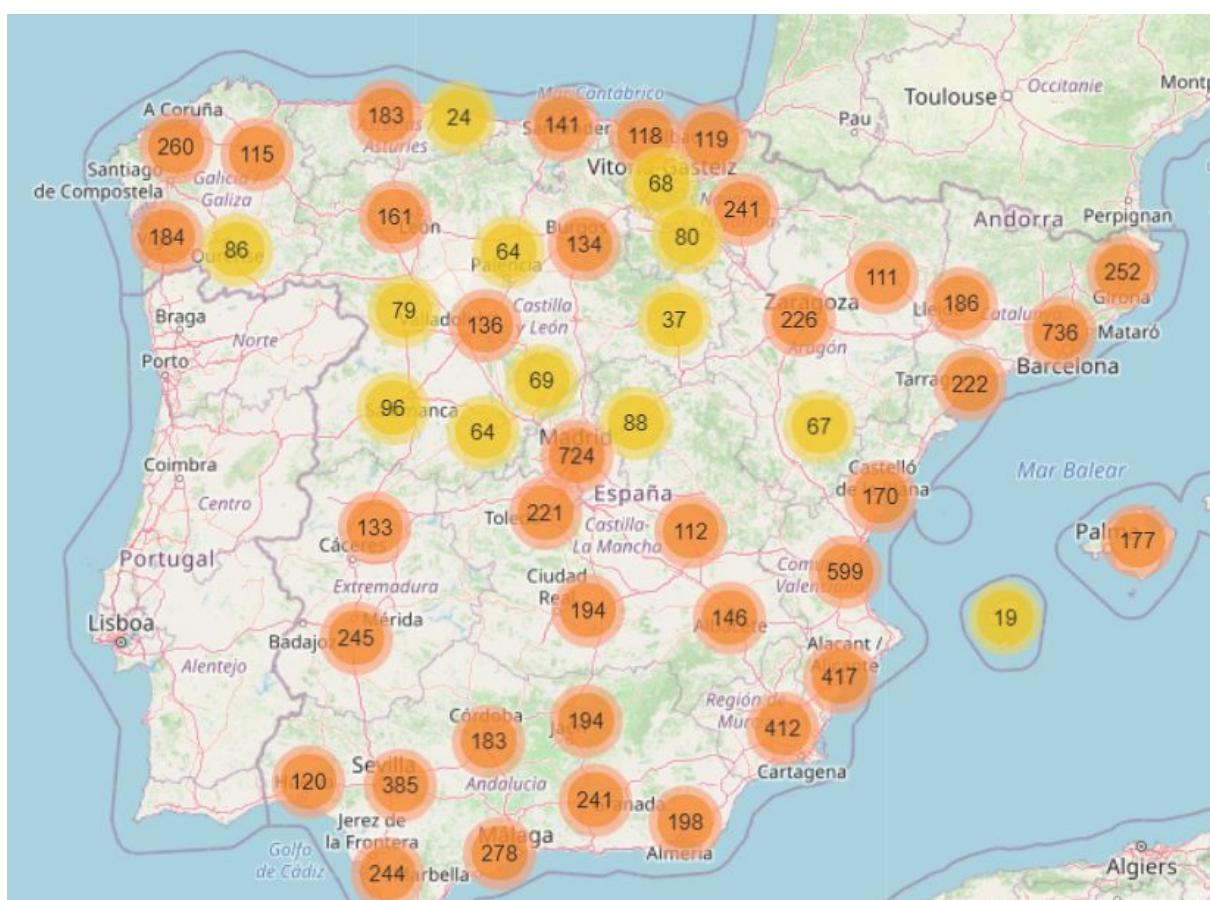
```
axios.get('{{ route('api.gasolineras.index') }}')
.then(function (response) {
    for (var i = 0; i < response.data.features.length; i++) {
        var procesado = parseInt(response.data.features[i].properties.IDProvincia)-1;
        var iconomarca = creaemarker(response.data.features[i].properties.Rotulo);

        //console.log(procesado[0])
        this.arrayClusters[procesado].addLayer(new L.marker([parseFloat(response.data.features[i].properties.Latitud), parseFloat(response.data.features[i].properties.Longitud)]));
    }
})
```

Una vez acabado esto, añadiremos todos los clusters en el mapa lo que hará que el usuario pueda observar ya todas las gasolineras a nivel peninsular.

```
for(var i=0;i<arrayClusters.length;i++){
    arrayClusters[i].addTo(map);
}
```

Aquí podemos observar el resultado del código mencionado anteriormente:



Ahora una vez ya está todo en su sitio toca una de las partes más importantes, donde el usuario pueda usar los distintos filtros para obtener solo los datos de las estaciones de servicio que le interesen.

Para ello, empezamos con recoger los datos que el usuario ha introducido en los filtros, tanto los de localización como los de el tipo de carburante. En este caso para guardar los datos en una variable he utilizado DOM, posteriormente, He modificado las dimensiones del mapa con JQuery para poder mostrar la tabla de resultados.

```
function filtrar(){

    var comarca = document.getElementById('comarca').value;
    var provincia = document.getElementById('provincia').value;
    var municipio = document.getElementById('municipio').value;
    var gasolina = document.querySelector('input[name="Fuel-card"]:checked').value;

    $("#boxlist").addClass( "col-md-3" );
    $("#boxmap").removeClass( "col-md-12" ).addClass( "col-md-9" );
    $("#tablelist").removeClass( "d-none" )
    $('#tablelist > tbody').children().remove();
```

A continuación usamos de nuevo AXIOS para pasar los parámetros recibidos vía URL y así luego recibirlos y tratarlos des del controlador. De igual manera que en la función general, obtendremos las gasolineras (en este caso solo las filtradas) y las mostraremos en el mapa. En este caso también añadiremos los datos en la tabla con JQuery.

```
axios.get("/api/filtro?comarca="+comarca+"&provincia="+provincia+"&municipio="+municipio+"&gasolina="+gasolina)
.then(function (response) {
    for (var i = 0; i < response.data.features.length; i++) {

        if(i<10){
            if(i==0){
                CentrarCluster(response.data.features[i].properties.Latitud,response.data.features[i].properties.Longitud);
                $('#LUP').text("Datos Actualizados: "+(response.data.features[i].properties.created_at).substring(0,10))
            }
            var iconolista='icons/'+response.data.features[i].properties.Rotulo+'.png';
            var iconoerror='icons/OTROS.png';
            for(var k=0;response.data.features[i].properties.length;k++){
                console.log(response.data.features[i].properties[k]);
            }
            $('#tablelist > tbody:last-child').append('<tr><td>'+response.data.features[i].properties.Rotulo+'</td> <td>'+re
        }
        var procesado = parseInt(response.data.features[i].properties.IDProvincia)-1;
        var iconomarca = crearmarker(response.data.features[i].properties.Rotulo);

        this.arrayClusters[procesado].addLayer(new L.marker([parseFloat(response.data.features[i].properties.Latitud), parseFloat(response.data.features[i].properties.Longitud)]));
    }
}).catch(function (error) {
    console.log(error);
});
```

En este apartado de filtro también usaremos un par de funciones extra.

La primera es la de limpiar los clúster anteriores para que no se sobrepongan los datos unos encima de otros.

```
function limpiarlayers(){
    for(var i=0;i<arrayClusters.length;i++){
        arrayClusters[i].clearLayers();
    }
}
```

La otra, que en realidad son 2 pero tienen una función casi similar, es la de centrar la gasolinera o el clúster. Donde pasamos una latitud y longitud y centramos el mapa a esa zona con un zoom determinado

```
function CentrarGasolinera(x,y){
    | map.flyTo([x, y], 15)
}
```

Con esto ya tendríamos el código hecho para poder mostrar las estaciones de servicio a gusto del consumidor.

Filtros

Por otro lado hay algunos detalles más a tratar que quizás no son necesarios pero ofrece más facilidades al usuario, como por ejemplo, al indicar una comunidad autónoma, sólo aparezcan como opciones las provincias que formen parte de esa comunidad y esa misma relación entre provincias y municipios.

Eso se lleva a cabo con esta función de JQuery, donde obtenemos una lista de las provincias y las comparamos con el id de la comunidad autónoma, si coinciden se añade como una opción extra en el selector, una vez acabado se desbloquea el selector para el usuario para que pueda escoger la provincia deseada y de igual manera con los municipios.

```
$( "#provincia" ).prop( "disabled", true );
$( "#municipio" ).prop( "disabled", true );

$( "#comarca" ).change(function() {
    var provinciasjs = @json($provincias);
    var comarcaAct=$("#comarca").val();
    var isEnabled = $('#municipio').prop('disabled');
    $("#provincia").children().slice(1).remove();
    $("#municipio").children().slice(1).remove();
    var o = new Option("Todas", "Todas");
    $(o).html("Todas");
    $("#provincia").append(o);
    for(var d=0;d<provinciasjs.length;d++){
        if(provinciasjs[d].id_comarca==comarcaAct){
            var o = new Option(provinciasjs[d].nombre,provinciasjs[d].id);
            $(o).html(provinciasjs[d].nombre);
            $("#provincia").append(o);
        }
    }
    $("#provincia").prop("disabled", false );
    if(!isEnabled){
        $('#provincia').trigger('change');
    }
});
```

Indicar que al cambiar de comunidad o provincia se eliminan las opciones anteriores y se ejecuta un trigger que simula un cambio, para poder dejar seleccionadas aquellas comunidades que tengan solo una provincia como por ejemplo Cantabria.

Otros

Aquí quiero comentar un par de secciones que pueden ocasionar dudas por su repetición o bien su longitud.

La primera de ellas es la imagen de los marcadores, donde he querido ser detallista y mostrar un ícono personalizado para las principales estaciones de servicio. El problema está en que leaflet no contempla una forma eficiente para poder realizar esto en caso de tener muchos tipos de ícono. Por ese motivo he tenido que crear cada ícono manualmente y eso hace que en mi caso, que he tenido en consideración unas 20-25 estaciones de servicio distintas, tenga bastantes líneas “repetidas” aunque cada una aporte algo distinto.

```
var AGLA = new L.icon({iconUrl: '{{ asset("icons/AGLA.png") }}',iconSize:[50, 50],iconAnchor: [25, 51],popupAnchor: [0, -49],})
var ALCAMPO = new L.icon({iconUrl: '{{ asset("icons/ALCAMPO.png") }}',iconSize:[50, 50],iconAnchor: [25, 51],popupAnchor: [0, -49],})
var AVANZAOIL = new L.icon({iconUrl: '{{ asset("icons/AVANZAOIL.png") }}',iconSize:[50, 50],iconAnchor: [25, 51],popupAnchor: [0, -49],})
var AVIA = new L.icon({iconUrl: '{{ asset("icons/AVIA.png") }}',iconSize:[50, 50],iconAnchor: [25, 51],popupAnchor: [0, -49],})
```

Por otro lado tenemos un tema relacionado con el anterior, que es el de asignar dicho ícono a la gasolinera, para ello utilizo un switch y cotejo el nombre del rótulo con el de las opciones que he creado, si coinciden, añado ese ícono al marcador.

Uno de los principales problemas que he visto que no he podido solucionar (en parte porque no es culpa mía) es que la api muchas veces proporciona nombres distintos para la gasolinera como por ejemplo en el caso de la estación de servicio de “Avanza Oil” en algunas entradas podemos encontrar ese nombre y en otras “Avanza”, que casualmente también es el nombre de otra compañía con estaciones de servicio. En caso de que el nombre sea distinto al convencional, se les añade un ícono por defecto.

```
function crearmarker(rotulo){

    switch (rotulo) {
        case "REPSOL":
            var icon = REPSOL;
            break;
        case "CEPSA":
            var icon = CEPSA;
            break;
```

M07: Desarrollo de aplicaciones en entorno servidor

Introducción

En este módulo formativo he utilizado el famoso framework laravel, para poder realizar código elegante y simple de una forma bien estructurada.

A continuación, explicaré cómo he organizado mi proyecto en laravel desde lo más básico, como podrían ser las migraciones, hasta el uso de los controladores.

Código

Migraciones

Comenzaremos por explicar cómo creamos la base de datos a través de laravel.

Podemos observar las típicas migraciones que crea laravel de forma automática como la de usuarios, hasta las personales hechas por mi, como por ejemplo, las comunidades y las gasolineras entre otros.

El orden de las migraciones es muy importante, hay que tener en cuenta que hay tablas que contienen claves foráneas de otras, y si las hacemos en un orden incorrecto pueden saltar errores.

En mi caso tengo las siguientes migraciones en el orden que vemos en la imagen:

```
▼ migrations
  ▾ 2014_10_12_000000_create_users_table.php
  ▾ 2014_10_12_100000_create_password_resets_table.php
  ▾ 2019_08_19_000000_create_failed_jobs_table.php
  ▾ 2020_12_10_125517_create_comunidades_table.php
  ▾ 2020_12_10_125518_create_provincias_table.php
  ▾ 2020_12_10_125519_create_municipios_table.php
  ▾ 2020_12_10_125521_create_gasolineras_table.php
```

Dentro de las migraciones que he creado podemos observar la función para crear la tabla y sus respectivas variables, donde podemos ver entre otras el “Id” que corresponde a un bigIncrements que equivale a una clave primaria incremental.

```
public function up()
{
    Schema::create('comunidades', function (Blueprint $table) {
        $table->bigIncrements('Id');
        $table->string('Nombre', 255);
        $table->timestamps();
    });
}
```

Todas las migraciones siguen un patrón similar por lo que poner captura de todas las funciones puede ser un poco innecesario, aún así quiero hacer hincapié en toda las claves primarias ya que sin eso, las relaciones entre tablas y sus respectivos datos, no sería posible.

1. Relación migración provincias con las comunidades autónomas y municipios:

```
$table->foreign('id_provincia')->references('id')->on('provincias');
```

2. Relación migración municipios con las provincias:

```
$table->foreign('id_comarca')->references('Id')->on('comunidades');
```

3. Relación migración gasolineras con todas las anteriores:

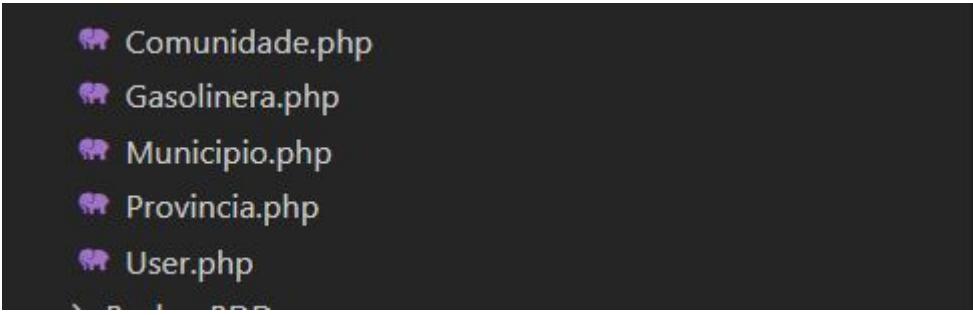
```
$table->foreign('IDMunicipio')->references('id')->on('municipios');
$table->foreign('IDProvincia')->references('id')->on('provincias');
$table->foreign('IDCAA')->references('id')->on('comunidades');
```

Modelos

Son otra parte fundamental de la app ya que tienen la responsabilidad de acceder a los datos, modificarlos, etc.

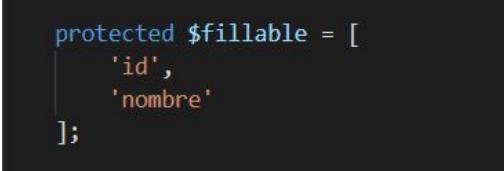
Tengo un modelo para cada tabla con la que tenga interés de realizar acciones como las mencionadas anteriormente.

Aquí podemos ver la estructura de documentos de los modelos:



- Comunidad.php
- Gasolinera.php
- Municipio.php
- Provincia.php
- User.php

En los modelos podemos indicar que campos de la tabla queremos proteger y cuales dar acceso como sería el caso de las comunidades autónomas, donde damos acceso al uso del campo “id” y “nombre”.



```
protected $fillable = [  
    'id',  
    'nombre'  
];
```

También es importante crear la relación de tablas, para poder acceder a los datos de otra tabla con variables de una independiente.

Ejemplo de ello sería si quisiéramos obtener el nombre de la provincia solo con el id de la misma.

A continuación mostraré todas las relaciones entre tablas:

1. Modelo Comunidad con Gasolinera y Provincias.

```
public function gasolineras()
{
    return $this->hasMany('App\Gasolinera');
}
public function provincias()
{
    return $this->hasMany('App\Provincia');
```

2. Modelo Provincias con Comunidades,Municipios y Gasolineras.

```
public function gasolineras()
{
    return $this->hasMany('App\Gasolinera');
}
public function municipios()
{
    return $this->hasMany('App\Municipio');
}
public function comunidades()
{
    return $this->belongsTo('App\Comunidad');
```

3. Modelo Municipios con Provincias y Gasolineras.

```
public function gasolineras()
{
    return $this->hasMany('App\Gasolinera');
}
public function provincias()
{
    return $this->belongsTo('App\Provincia');
```

4. Modelo Gasolineras con Comunidades,Provincias y Municipios.

```
public function comunidades()
{
    return $this->belongsTo('App\Comunidad');
}
public function provincias()
{
    return $this->belongsTo('App\Provincia');
}
public function municipios()
{
    return $this->belongsTo('App\Municipio');
```

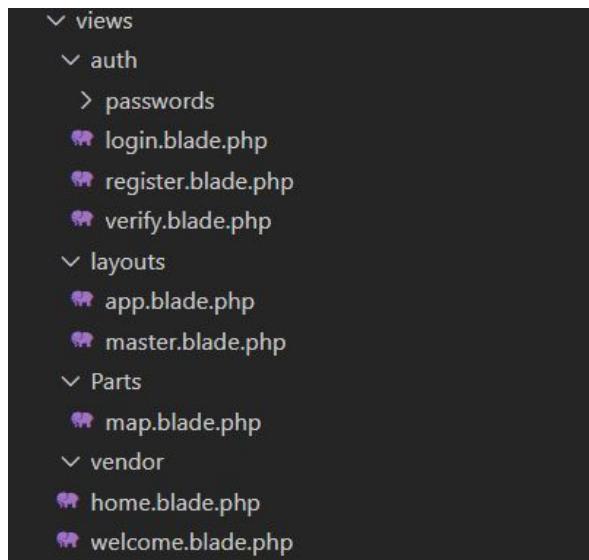
Vistas

Blade es el sistema de plantillas de Laravel, el cual nos permite generar HTML dinámico con una sintaxis mucho más limpia que si usáramos PHP plano.

Mi página, al ser una landing page solo utiliza un blade, sin contar el blade master (app) del que deriva.

También tiene blades para el login y registro, que aun siendo funcionales, aun no les he aplicado un diseño propio debido a que los usuarios en este momento no tienen características especiales en comparación a los invitados.

Aquí podemos ver la estructura de los blades creados:



El app blade, que se crea por defecto no tiene muchas modificaciones simplemente he modificado el logo y el favicon, he añadido yields y staks para poder añadir código en esa sección desde otras plantillas y he incluido algunos estilos y librerías extras.

```

<title>Gas Finder</title>
<!-- Fonts -->
<link rel="dns-prefetch" href="//fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet" type="text/css">
<link rel="icon" href="{{ asset('logo/faviconlogo.png') }}" type="image/x-icon"/>
<!-- Styles -->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<link href="{{ asset('css/app.css') }}" rel="stylesheet">
<link href="{{ asset('css/form.css') }}" rel="stylesheet">
@yield('styles')
</head>
<body>
    <div id="app">
        <nav class="navbar navbar-expand-md navbar-light bg-white shadow-sm">
            <div class="container">
                <a class="navbar-brand w-50" href="{{ url('/') }}>
                     'api.', 'namespace' => 'Api'], function () {
    Route::get('gasolineras', 'GasolineraController@index')->name('gasolineras.index');
    Route::get('filtro', 'GasolineraController@filtro');
});
  
```

En el caso del documento "web.php" podemos encontrar la ruta estática que nos devuelve la vista de la página principal.

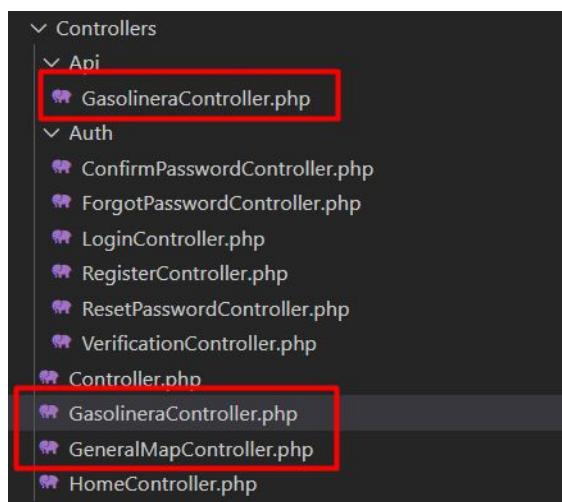
```

Route::get('/', 'GeneralMapController@index');
Auth::routes();
  
```

Controladores

El apartado de los controladores es un mecanismo que nos permite agrupar la lógica de peticiones HTTP relacionadas y de esta forma organizar mejor nuestro código.

Dejando de lado los controladores creados automáticamente por laravel, utilizo 3 controladores tal y como podemos ver en la imagen.



En el controlador de “GasolineraController” (el que no forma parte de la carpeta API) se encarga simplemente de devolver en formato JSON todas las gasolineras de la base de datos tal y como podemos ver en la siguiente imagen.

```
public function index()
{
    return response()->json( Gasolinera::all() );
```

Por otro lado “GeneralMapController” se encarga de devolver a la vista “map” los datos de las comunidades autónomas, provincias y municipios.

```
public function index(Request $request)
{
    $comunidades = Comunidad::all();
    $provincias = Provincia::all();
    $municipios = Municipio::all();
    return view('Parts.map',compact('comunidades','provincias','municipios'));
}
```

Mientras que el apartado “GasolinerasController” que se encuentra en la carpeta API tiene 2 funciones incorporadas, la primera, “index” que se encarga de devolver los datos de todas las gasolineras a través del navegador en formato json.

```
public function index(Request $request)
{
    $Gasolinera = Gasolinera::all();

    $geoJSONdata = $Gasolinera->map(function ($gasolinera) {
        return [
            'type'      => 'Feature',
            'properties' => new GasolineraResource($gasolinera),
            'geometry'   => [
                'type'      => 'Point',
                'coordinates' => [
                    $gasolinera->Longitud,
                    $gasolinera->Latitud,
                ],
            ],
        ];
    });

    return response()->json([
        'type'      => 'FeatureCollection',
        'features' => $geoJSONdata,
    ]);
}
```

Mientras que la función “filtro” se encarga de filtrar las gasolineras en base a los parámetros que ha introducido el usuario en la web. Esta función devolverá también los datos en formato json a través del navegador.

```
public function filtro(Request $request)
{
    $condicions = ['IDCAA' => $request->comarca, 'IDProvincia' => $request->provincia, 'IDMunicipio' => $request->municipio];
    $condicions2 = ['IDCAA' => $request->comarca];
    $condicions3 = ['IDCAA' => $request->comarca, 'IDProvincia' => $request->provincia];

    if($request->comarca=="Todas" and $request->provincia=="Todas" and $request->municipio=="Todas"){
        $Gasolinera = Gasolinera::orderBy($request->gasolina,'asc')->get();
    }
    elseif($request->provincia=="Todas" and $request->municipio=="Todas"){
        $Gasolinera = Gasolinera::where($condicions2)->orderBy($request->gasolina,'asc')->get();
    }
    elseif($request->municipio=="Todas"){
        $Gasolinera = Gasolinera::where($condicions3)->orderBy($request->gasolina,'asc')->get();
    }
    else{
        $Gasolinera = Gasolinera::where($condicions)->orderBy($request->gasolina,'asc')->get();
    }
}
```

Otros

Para finalizar la explicación de cómo he elaborado la sección de este módulo decir que actualmente las vistas , modelos y controladores relacionados con el tema de usuarios están en funcionamiento. Debido al tiempo que se me ha quedado corto no he podido realizar ninguna funcionalidad útil con el sistema de usuarios pero aun así me gustaría comentar las opciones que existían para realizar en el proyecto.

La primera de ellas se trata de realizar un sistema de favoritos entre usuario y gasolinera, es decir, que el usuario pudiera marcar una gasolinera como favorita (por algún motivo que crea conveniente) de esta forma, cuando el usuario realice login, se mostrará una lista alternativa con sus gasolineras favoritas y los precios en ese momento.

Otra de las opciones que tenía planeado, era la de realizar un sistema de notificaciones (cuando estuviera en formato APP) para indicar al usuario cuando bajan los precios o cuando están más altos.

Finalmente, la última opción que había planeado, era tener un sistema de localización fija, donde el usuario indicará su “zona habitual”. Eso haría que al iniciar sesión, en vez de mostrar toda las gasolineras de la península, mostrará directamente las de su municipio o provincia, cosa que daría más velocidad a la aplicación web para cargar las gasolineras y el usuario se ahorraría introducir los datos cada vez.

M08: Despliegamiento de aplicaciones web

Introducción

Para realizar el despliegamiento de Gas Finder, he utilizado la herramienta de EC2 (Elastic Compute Cloud) que proporciona AWS (Amazon Web Services). EC2 permite a los usuarios alquilar computadores virtuales en los cuales pueden ejecutar sus propias aplicaciones.

A continuación mostraré un manual detallado de cómo ha sido todo el proceso para desplegar la aplicación web.

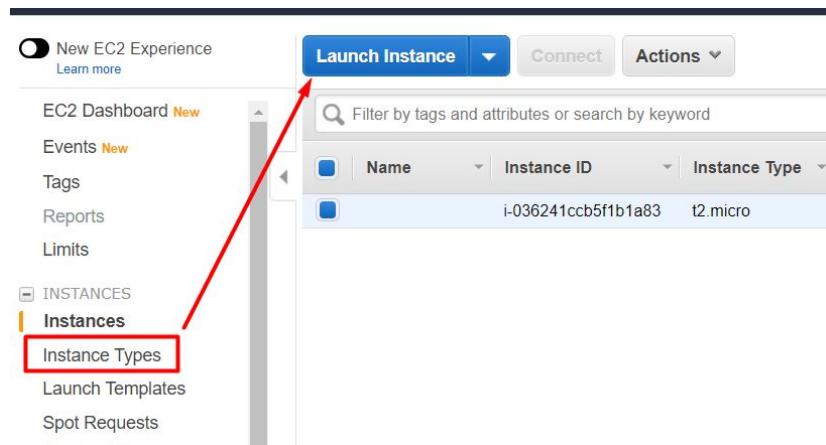
Manual

Instancia

En primer lugar, abriremos la consola de AWS y nos dirigiremos al apartado de servicios. Allí escogemos la opción de EC2 dentro de la categoría de Informática.



A continuación nos dirigiremos al apartado de instancias y crearemos una nueva.



Nos aparecerán distintas opciones de máquinas virtuales. Yo escogí la de Ubuntu 18



Acto seguido, nos aparecerá la opción de escoger una instancia, hay muchas opciones con diversas potencias, pero en nuestro caso solo podemos escoger la gratuita que no ofrece muchos recursos, pero los suficientes para hacer funcionar todo.

<input checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only
-------------------------------------	-----------------	--------------------------------	---	---	----------

A continuación nos pedirá crear una clave, con ella tendremos acceso a nuestra máquina virtual de forma remota vía ssh. Le indicamos que crearemos una nueva llave y indicaremos el nombre.

Select an existing key pair or create a new key pair x

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair ▼

Key pair name

[Download Key Pair](#)

 You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location**. You will not be able to download the file again after it's created.

[Cancel](#) [Launch Instances](#)

Finalmente nos saldrá una notificación de que nuestra instancia se está iniciando.

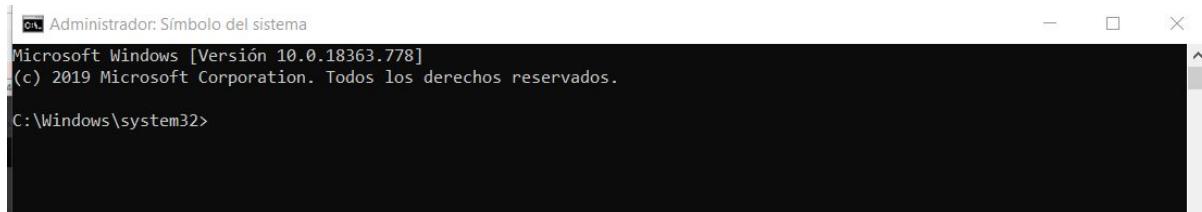
Launch Status

 Your instances are now launching

The following instance launches have been initiated: [i-036241ccb5f1b1a83](#) [View launch log](#)

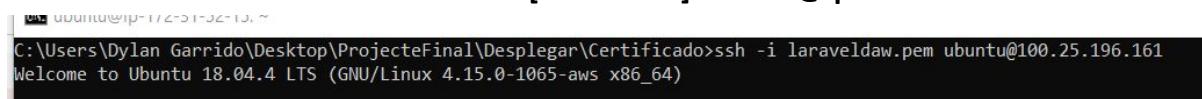
Conexión remota vía SSH

Ahora, podemos conectarnos de forma remota a nuestra máquina virtual vía ssh. Para ello abrimos un terminal en nuestra máquina real para empezar.



Nos dirigimos a la ruta donde hemos guardado el certificado (llave) y ejecutamos el siguiente comando:

```
ssh -i [certificado] usuario@ip
```



Como podemos ver, ya estamos controlando la máquina virtual, así que para empezar la pondremos a punto utilizando el comando:

```
sudo apt-get update
```

```
ubuntu@ip-172-31-52-15:~$ sudo apt-get update
```

Acto seguido, ejecutaremos un upgrade:

```
sudo apt-get upgrade  
Reading package lists... Done  
ubuntu@ip-172-31-52-15:~$ sudo apt-get upgrade
```

Con esto ya tendríamos la máquina actualizada para preparar la instalación de todos los otros componentes.

Apache

Apache es un servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras.

Para instalarlo en nuestra máquina virtual empezaremos por ejecutar el siguiente comando:

```
sudo apt-get install apache2
```

```
Reading package lists... done
ubuntu@ip-172-31-52-15:~$ sudo apt-get install apache2
```

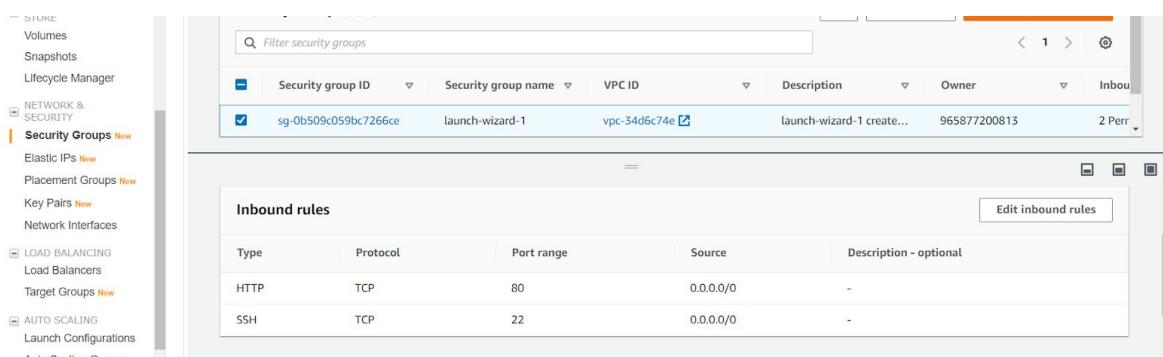
Si queremos ver si la instalación ha sido correcta, ejecutamos un status:

```
sudo service apache2 status
```

```
ubuntu@ip-172-31-52-15:~$ sudo service apache2 start
ubuntu@ip-172-31-52-15:~$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
  Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
            └─apache2-systemd.conf
    Active: active (running) since Sat 2020-05-23 10:18:50 UTC; 31s ago
      Main PID: 11873 (apache2)
        Tasks: 55 (limit: 1152)
       CGroup: /system.slice/apache2.service
               ├─11873 /usr/sbin/apache2 -k start
               ├─11875 /usr/sbin/apache2 -k start
               └─11876 /usr/sbin/apache2 -k start

May 23 10:18:50 ip-172-31-52-15 systemd[1]: Starting The Apache HTTP Server...
May 23 10:18:50 ip-172-31-52-15 systemd[1]: Started The Apache HTTP Server.
ubuntu@ip-172-31-52-15:~$
```

Por otro lado, si queremos tener acceso por el navegador a través del puerto del servidor web, deberemos abrir dicho puerto a través de la configuración de la instancia que se encuentra en la consola de AWS.



PHP

El siguiente paso, es instalar la librería de php:

```
sudo apt-get install php libapache2-mod-php
[May 25 10:18:30 ip-172-31-52-15 systemd[1]: Started
ubuntu@ip-172-31-52-15:~$ sudo apt-get install php
[May 25 10:18:30 ip-172-31-52-15 systemd[1]: Started]
```

Continuaremos instalando más módulos de php

```
sudo apt-get install php-ext-soap
Application key set successfully.
ubuntu@ip-172-31-52-15:~$ sudo apt-get install php-ext-soap
```

También instalaremos php xdebug

```
sudo apt-get install php-xdebug
Processing triggers for libapache2-mod-php7.2 (7.2.24-0ubuntu1)
ubuntu@ip-172-31-52-15:~$ sudo apt-get install php-xdebug
```

MYSQL

Otro servicio importante que debemos instalar es el de mysql, con el que gestionaremos todo el tema de las bases de datos.

Empezaremos con el siguiente comando:

```
sudo apt-get install mysql-server
```

```
ubuntu@ip-172-31-52-15:/var/www/html$ sudo apt-get install mysql-server
```

Uno de los principales problemas que encontraremos es que el usuario por defecto no tiene una contraseña establecida y en muchos casos, aun teniendo una establecida no podemos iniciar sesión por consola. Eso es debido a que el usuario root utiliza a veces un socket en vez de contraseña para iniciar sesión.

Para modificar eso, ejecutaremos el siguiente comando:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password by 'Admin1234'
```

```
mysql> SELECT user,authentication_string,plugin,host FROM mysql.user
   -> ;
+-----+-----+-----+-----+
| user      | authentication_string          | plugin      | host       |
+-----+-----+-----+-----+
| root      | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | auth_socket | localhost  |
| mysql.session | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost  |
| mysql.sys  | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost  |
| debian-sys-maint | *C094BCD0B63123F38F61BDA6A806B0AA5E632D9E | mysql_native_password | localhost  |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Admin1234';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Una vez acabado el paso anterior, podemos probar a iniciar mysql.

Tal y como vemos en la imagen funciona perfectamente.

```
ubuntu@ip-172-31-52-15:/var/www/html$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.30-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

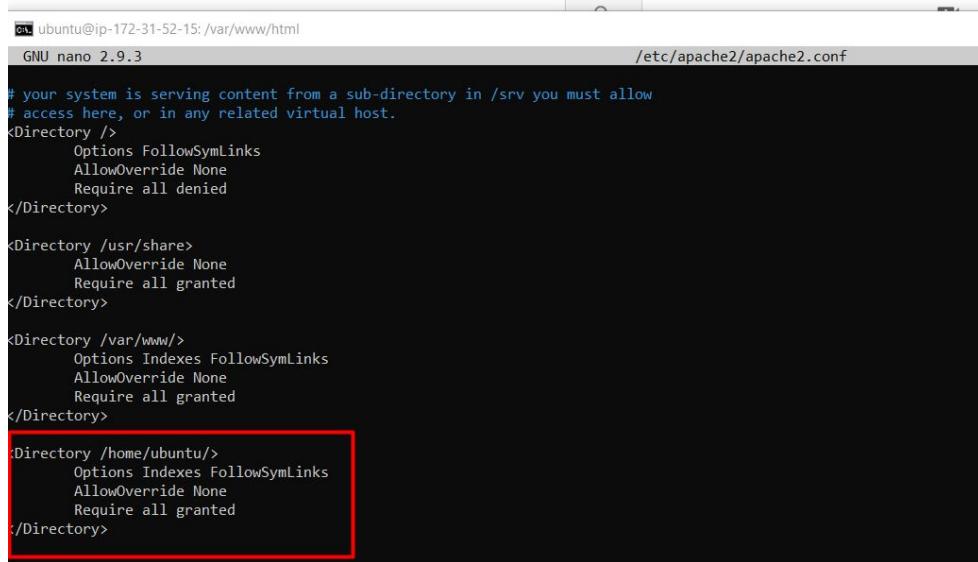
Configuración servidor

Una vez finalizada la instalación de los servicios anteriores activaremos el módulo de rewrite y reiniciamos el servicio de apache.

```
sudo a2enmod rewrite  
sudo service apache2 restart
```

```
ubuntu@ip-172-31-52-15:/var/www/html$ sudo a2enmod rewrite  
Enabling module rewrite.  
To activate the new configuration, you need to run:  
    systemctl restart apache2  
ubuntu@ip-172-31-52-15:/var/www/html$ service apache2 restart
```

A continuación, entraremos en el documento de apache2.conf y añadiremos un nuevo directorio. **Indicar que “AllowOverride” ha de estar en “all”**



```
ubuntu@ip-172-31-52-15:/var/www/html  
GNU nano 2.9.3  
/etc/apache2/apache2.conf  
  
# your system is serving content from a sub-directory in /srv you must allow  
# access here, or in any related virtual host.  
<Directory />  
    Options FollowSymLinks  
    AllowOverride None  
    Require all denied  
</Directory>  
  
<Directory /usr/share>  
    AllowOverride None  
    Require all granted  
</Directory>  
  
<Directory /var/www/>  
    Options Indexes FollowSymLinks  
    AllowOverride None  
    Require all granted  
</Directory>  
  
<Directory /home/ubuntu/>  
    Options Indexes FollowSymLinks  
    AllowOverride All  
    Require all granted  
</Directory>
```

En la mismo documento, comentaremos los tags de User y Group. Indicaremos los nuestros propios de la máquina que estamos utilizando.

```
# These need to be set in /etc/apache2/envvars
#User ${APACHE_RUN_USER}
#Group ${APACHE_RUN_GROUP}
User ubuntu
Group ubuntu
```

Ahora entraremos en el documento de 000-default.conf y modificamos el “DocumentRoot” por la misma ruta que hayamos indicado en el documento de “apache.conf”.

```
GNU nano 2.9.3                               /etc/apache2/sites-enabled/000-default.conf

<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /home/ubuntu
```

Una vez los pasos anteriores estén finalizados, realizamos un reinicio del servicio de apache y revisamos que su funcionamiento sea correcto.

```
sudo service apache2 restart
sudo service apache2 status
```

```
ubuntu@ip-172-31-52-15:/var/www/html$ sudo service apache2 restart
ubuntu@ip-172-31-52-15:/var/www/html$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Drop-In: /lib/systemd/system/apache2.service.d
             └─apache2-systemd.conf
     Active: active (running) since Sat 2020-05-23 11:09:50 UTC; 16s ago
       Process: 29196 ExecStop=/usr/sbin/apachectl stop (code=exited, status=0/SUCCESS)
       Process: 29201 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
     Main PID: 29215 (apache2)
        Tasks: 6 (limit: 1152)
      CGroup: /system.slice/apache2.service
              ├─29215 /usr/sbin/apache2 -k start
              ├─29220 /usr/sbin/apache2 -k start
              ├─29221 /usr/sbin/apache2 -k start
              ├─29222 /usr/sbin/apache2 -k start
              ├─29224 /usr/sbin/apache2 -k start
              └─29226 /usr/sbin/apache2 -k start

May 23 11:09:49 ip-172-31-52-15 systemd[1]: Stopped The Apache HTTP Server.
May 23 11:09:49 ip-172-31-52-15 systemd[1]: Starting The Apache HTTP Server...
May 23 11:09:50 ip-172-31-52-15 systemd[1]: Started The Apache HTTP Server.
ubuntu@ip-172-31-52-15:/var/www/html$
```

Composer

Composer es un sistema de gestión de paquetes para programar en PHP el cual provee los formatos estándar necesarios para manejar dependencias y librerías de PHP.

Para instalarlo, tenemos que introducir los comandos que nos indican en su [web](#) tal y como podemos ver en la imagen inferior.

```
ubuntu@ip-172-31-52-15:~$ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
ubuntu@ip-172-31-52-15:~$ php -r "if (hash_file('sha384', 'composer-setup.php') === 'e0012edf3e80b6978849f5eff0d4b4e4c79ff1609d
fb710ca74755a') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
composer-setup.phpInstaller verified
ubuntu@ip-172-31-52-15:~$ php composer-setup.php
All settings correct for using Composer
Downloading...

Composer (version 1.10.6) successfully installed to: /home/ubuntu/composer.phar
Use it: php composer.phar

ubuntu@ip-172-31-52-15:~$ php -r "unlink('composer-setup.php');"
ubuntu@ip-172-31-52-15:~$
```

Una vez se ejecuten todos los comandos, podemos revisar si la instalación ha sido correcta introduciendo “composer” en el terminal.

si queremos que el composer no afecte solo a un directorio, sino que se instale de forma global utilizaremos el siguiente comando:

```
ubuntu@ip-172-31-52-15:~$ sudo mv composer.phar /usr/local/bin/composer
```

Laravel

Laravel es el framework que utilizaremos para realizar nuestro sitio web. Está desarrollado en php y ofrece un sistema simple y ordenado para programar.

Para empezar su instalación, empezaremos por redistribuir la memoria, ya que al tener un servidor con tan poca memoria RAM, hay que hacer alguna modificaciones primero.

```
ubuntu@ip-172-31-52-15:~$ free -m
      total        used        free      shared  buff/cache   available
Mem:       983         127        608          3        248        711
Swap:        0          0          0
ubuntu@ip-172-31-52-15:~$ sudo /bin/dd if=/dev/zero of=/var/swap.1 bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 13.4288 s, 80.0 MB/s
ubuntu@ip-172-31-52-15:~$ sudo /sbin/mkswap /var/swap.1
mkswap: /var/swap.1: insecure permissions 0644, 0600 suggested.
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=159e9935-2ff1-47b7-b4e2-8e95e3ba49d3
ubuntu@ip-172-31-52-15:~$ sudo /sbin/swapon /var/swap.1
```

Una vez finalizado el paso anterior, ya podremos instalar laravel.

```
ubuntu@ip-172-31-52-15:~$ composer global require laravel/installer
Changed current directory to /home/ubuntu/.config/composer
Using version ^3.1 for laravel/installer
./composer.json has been created
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 16 installs, 0 updates, 0 removals
- Installing symfony/process (v5.0.8): Loading from cache
- Installing symfony/polyfill-ctype (v1.17.0): Downloading (100%)
- Installing symfony/filesystem (v5.0.8): Downloading (100%)
- Installing psr/container (1.0.0): Downloading (100%)
- Installing symfony/service-contracts (v2.0.1): Downloading (100%)
- Installing symfony/polyfill-php73 (v1.17.0): Downloading (100%)
- Installing symfony/polyfill-mbstring (v1.17.0): Downloading (100%)
- Installing symfony/console (v5.0.8): Downloading (100%)
- Installing ralouphie/getallheaders (3.0.3): Downloading (100%)
- Installing psr/http-message (1.0.1): Downloading (100%)
- Installing guzzlehttp/psr7 (1.6.1): Downloading (100%)
- Installing guzzlehttp/promises (v1.3.1): Downloading (100%)
- Installing symfony/polyfill-php72 (v1.17.0): Downloading (100%)
- Installing symfony/polyfill-intl-idn (v1.17.0): Downloading (100%)
- Installing guzzlehttp/guzzle (6.5.3): Downloading (100%)
- Installing laravel/installer (v3.1.0): Downloading (100%)
```

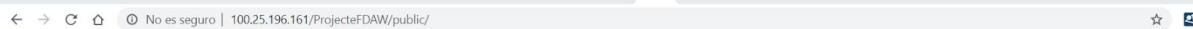
A continuación introduciremos los siguientes comandos para modificar ciertas rutas en la utilización de laravel.

```
ubuntu@ip-172-31-52-15:~$ export PATH="~/config/composer/vendor/bin:$PATH"
ubuntu@ip-172-31-52-15:~$ nano ~/.bashrc
```

Y una vez acabado todo esto, podemos crear un proyecto para revisar que esté todo funcionando correctamente.

```
DISCOVERED package: laravel/laravel [collision]
Package manifest generated successfully.
32 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan key:generate --ansi
Application key set successfully.
ubuntu@ip-172-31-52-15:~$ composer create-project --prefer-dist laravel/laravel ProjecteFDaw
```

Como podemos ver, si entramos por la IP del servidor podemos ver la página por defecto de laravel.



Laravel

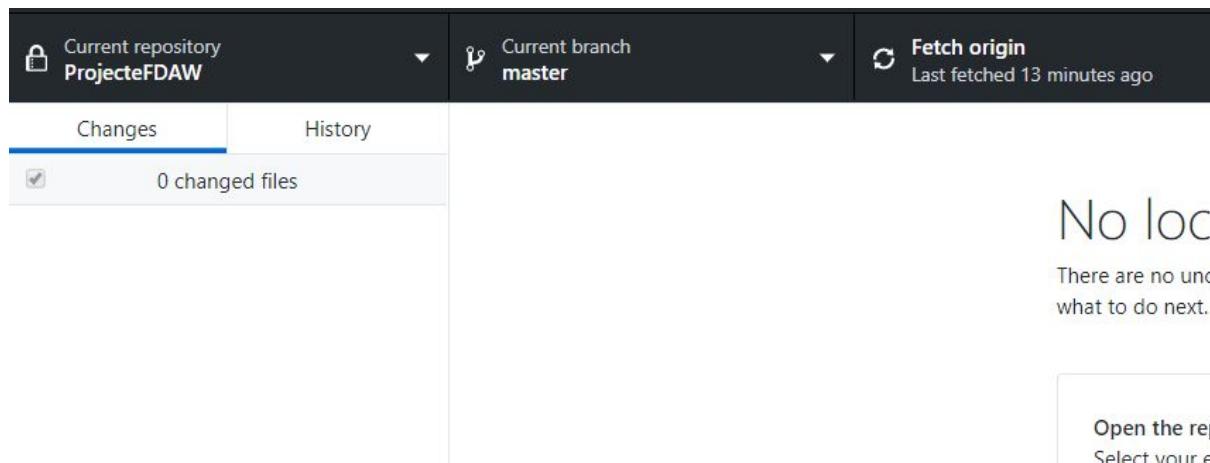
[DOCS](#) [LARACASTS](#) [NEWS](#) [BLOG](#) [NOVA](#) [FORGE](#) [VAPOR](#) [GITHUB](#)

GIT

Git es un software de control de versiones, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

La utilización de git en este proyecto es muy importante, ya que con él, pasamos las modificaciones realizadas desde la máquina real a nuestro servidor.

En la maquina real, uso la versión de Github Desktop para subir las modificaciones al repositorio de github.



Para la utilización de git en nuestro servidor ubuntu, empezaremos por instalarlo.

Sudo apt-get install git-core

```
ubuntu@ip-172-31-52-15:~/ProjecteFDAWBuild/ProjecteFDAW$ apt-get install git-core
```

A continuación clonamos el repositorio con el siguiente comando.

git clone [URL]

```
ubuntu@ip-172-31-52-15:~/ProjecteFDAWBuild/ProjecteFDAW$ git clone https://github.com/dgarrido7/ProjecteFDAW.git
```

Finalmente si hacemos alguna modificación y queremos descargar-la, utilizaremos git pull.

```
ubuntu@ip-172-31-52-15:~/ProjecteFDAWBuild/ProjecteFDAW$ git pull
Username for 'https://github.com': dgarrido7
Password for 'https://dgarrido7@github.com':
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 7 (delta 5), reused 7 (delta 5), pack-reused 0
Unpacking objects: 100% (7/7), done.
From https://github.com/dgarrido7/ProjecteFDAW
  46afad4..935b052 master      -> origin/master
Updating 46afad4..935b052
Fast-forward
 app/Gasolinera.php          | 22 +-----
 app/Http/Controllers/GasolineraController.php |  6 -----
 2 files changed, 1 insertion(+), 27 deletions(-)
ubuntu@ip-172-31-52-15:~/ProjecteFDAWBuild/ProjecteFDAW$
```

Configuraciones finales

Ahora que está todo funcionando, nos dispondremos a hacer algunas modificaciones para dejar todo preparado para cuando clonemos el proyecto en servidor.

Empezaremos por entrar al documento de 000-default.conf y indicaremos el nuevo “DocumentRoot” en función del directorio donde se encuentre el proyecto final.

```
ubuntu@ip-172-31-52-15: ~
GNU nano 2.9.3
/etc/apache2/sites-enabled/000-default.conf

<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /home/ubuntu/ProyectoFDAW/public

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg
```

De igual forma que abrimos anteriormente los puertos para acceder vía HTTP, ahora abriremos los puertos de MYSQL para poder comunicarnos con la base de datos de forma remota.

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
SSH	TCP	22	0.0.0.0/0	-
MYSQL/Aurora	TCP	3306	0.0.0.0/0	-

A continuación accedemos a mysql y creamos la base de datos.

```
ubuntu@ip-172-31-52-15:~/ProyectoFDAWBuild/ProyectoFDAW$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.30-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE projectedaw;
```

En el documento .env de nuestro proyecto, indicamos la nueva base de datos y sus credenciales.

```
LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=projectedaw
DB_USERNAME=root
DB_PASSWORD=Admin1234
```

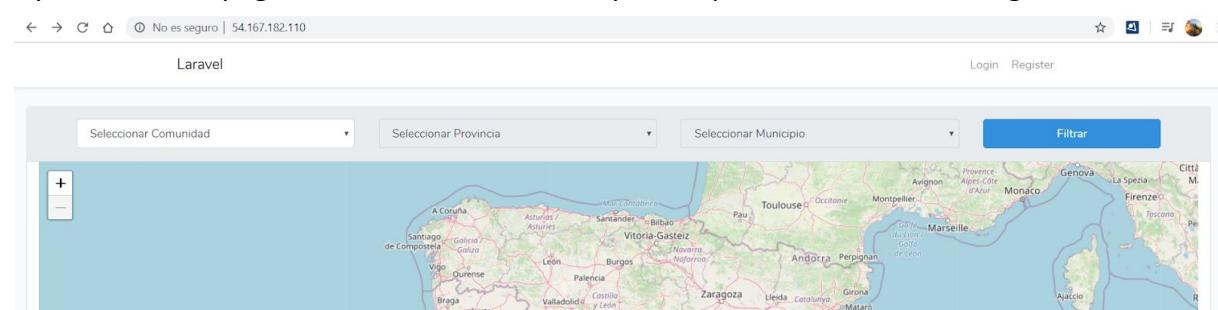
Corremos las migraciones para generar todas las tablas

```
ubuntu@ip-172-31-52-15:~/ProjecteFDAWBuild/ProjecteFDAW$ php artisan migrate:refresh
Migration table not found.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (0.14 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (0.13 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (0.08 seconds)
Migrating: 2020_12_10_125517_create_comunidades_table
Migrated: 2020_12_10_125517_create_comunidades_table (0.07 seconds)
Migrating: 2020_12_10_125518_create_provincias_table
Migrated: 2020_12_10_125518_create_provincias_table (0.24 seconds)
Migrating: 2020_12_10_125519_create_municipios_table
Migrated: 2020_12_10_125519_create_municipios_table (0.25 seconds)
Migrating: 2020_12_10_125521_create_gasolineras_table
Migrated: 2020_12_10_125521_create_gasolineras_table (0.68 seconds)
ubuntu@ip-172-31-52-15:~/ProjecteFDAWBuild/ProjecteFDAW$
```

Generamos la key del proyecto.

```
ubuntu@ip-172-31-52-15:~/ProjecteFDAWBuild/ProjecteFDAW$ sudo php artisan key:generate
Application key set successfully.
```

Y ya tenemos la página en funcionamiento tal y como podemos ver en la imagen inferior.



Aporte

Durante el proceso final de despliegue de la web tenía un problema singular, en el que por algún motivo no se cargaban los iconos de las gasolineras.

Este error se debía a que la carpeta donde se encontraban los iconos, se llamaba “icons” y por casualidades de la vida, apache en su configuración tiene esa ruta apuntando a otro lado.

Para solucionarlo, tenemos que entrar en el documento de alias.conf y comentar la línea que empieza por “Alias” tal y como podemos ver en la imagen.

```
ubuntu@ip-172-31-52-15: /etc/apache2/mods-available
GNU nano 2.9.3                                alias.conf

<IfModule alias_module>
    # Aliases: Add here as many aliases as you need (with no limit). The format is
    # Alias fakename realname
    #
    # Note that if you include a trailing / on fakename then the server will
    # require it to be present in the URL. So "/icons" isn't aliased in this
    # example, only "/icons/". If the fakename is slash-terminated, then the
    # realname must also be slash terminated, and if the fakename omits the
    # trailing slash, the realname must also omit it.
    #
    # We include the /icons/ alias for FancyIndexed directory listings. If
    # you do not use FancyIndexing, you may comment this out.

    # Alias /icons/ "/usr/share/apache2/icons/"

    <Directory "/usr/share/apache2/icons">
        Options FollowSymlinks
        AllowOverride None
        Require all granted
    </Directory>

</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
[ Wrote 24 lines ]
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   M-U Undo
^X Exit       ^R Read File   ^\ Replace    ^U Uncut Text  ^T To Spell   ^L Go To Line M-E Redo
```

M09: Diseño de interfaces web

Introducción

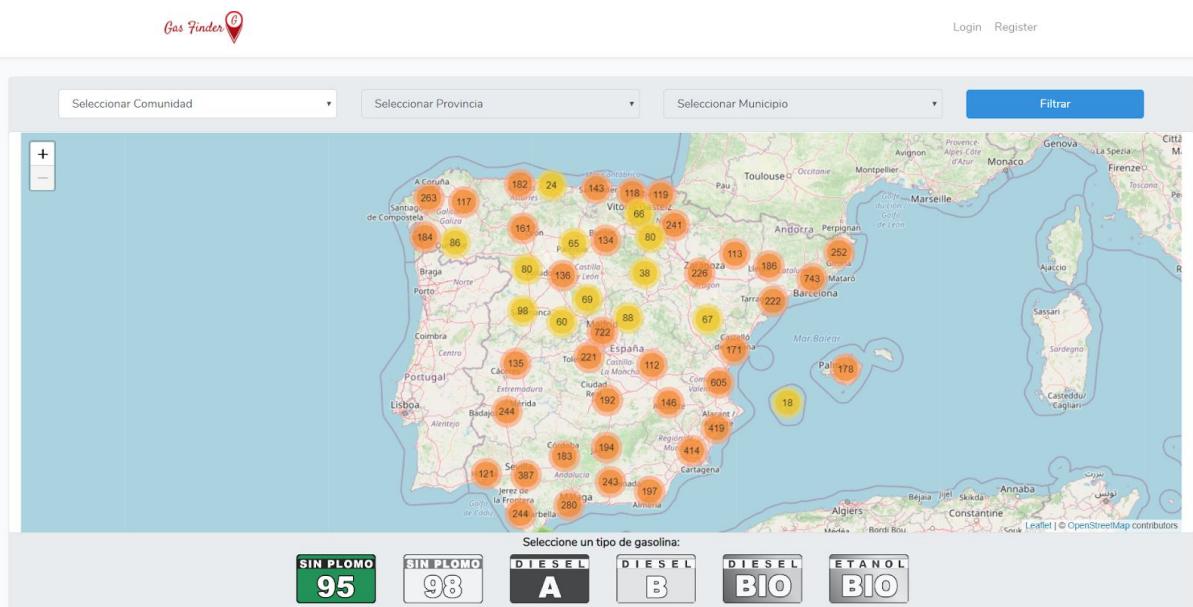
En la sección de este módulo, comentaré varios aspectos visuales del sitio web y parte de su código utilizado realizar la maquetación del sitio.

Usabilidad y Accesibilidad

Usabilidad

Estructuración del contenido:

En mi caso, he realizado una landing page, tipo de página que incluye toda la información en la misma, como en mi caso no hay muchos apartados, considero que es el tipo de página más apropiado para mi aplicación web, permite ver todo el contenido desde una vista general y todos los elementos están colocados de forma jerárquica para que no queden espacios vacíos en la página sin uso y por contra, que no haya contenido innecesario.



Como podemos ver en la imagen, en la parte superior encontramos un pequeño menú, que incluye el logo y los botones de registro.

El formulario está dividido en 2 segmentos, el de localizaciones y el de carburantes. Los he dividido para darles más protagonismo individual, para que el usuario entienda que ambas secciones son importantes y ha de llenar los campos en ambas.

Por otro lado eso también beneficia al mapa, ya que hace que se quede en la zona central y sobre más protagonismo al entrar en la página.

Diseño Limpio

He procurado realizar un diseño que sea cómodo para el usuario, distinguir bien las distintas zonas sin que eso ocasione distorsiones en el diseño.

He utilizado una fuente agradable para la lectura y una gama de colores simple y neutro complementado con las distintas zonas de la aplicación web para tener una zona monótona.

El mapa utilizado, muestra un layout acorde con las tonalidades del mapa, para que no desentoné con el otro contenido.

Control del usuario

Este es uno de los principales puntos a favor de mi aplicación web.

El usuario tiene casi plena libertad para interactuar con lo que quiera, como por ejemplo la interacción con los formularios:



Pero las principales interacciones que puede realizar, son con el mapa, puede moverse por distintas ubicaciones, buscar gasolineras manualmente, consultar sus datos, etc.



Facilitar la interacción

En relación con el apartado anterior, el usuario puede realizar todas las acciones mencionadas de forma fácil e intuitiva, desplegar los menús, hacer zoom en el mapa o incluso informarse de una estación de servicio.

Los procedimientos son sencillos y aptos para cualquier usuario.

Simplificar y sintetizar

Si nos fijamos en el sitio web, es muy simple, tiene lo justo, un navbar para mostrar los botones de login y registro, las 4 opciones de los formularios y el mapa. No le hace falta ni más ni menos, poner más contenido ahora sería innecesario, por ese motivo, la tabla de gasolineras solo la muestro cuando se han seleccionado los parámetros, para que no moleste al usuario mientras navega por el sitio web.

Adaptar la web a todo tipo de dispositivos

Como veremos más adelante, la web es responsive y se adapta a cualquier tipo de dispositivo, ordenadores, tablets y teléfonos. Más adelante mostraré ejemplos de su funcionamiento.

Accesibilidad

Percepción

Mencionar en primer lugar que el sitio web no utilizar contenido multimedia como audios y videos, lo que permite que usuarios con que tengan problemas auditivos puedan utilizar la web con normalidad.

La estructura es correcta y la relación de contraste considero que adecuada, se utiliza texto como alternativa a las imágenes para transmitir información, como la ubicación o el nombre de la estación de servicio.

La página es apta para la mayoría de personas con discapacidades, apenas tiene opciones, la página no tiene scroll, etc.

Operatividad

No hay contenido que requiera un tiempo límite para su visualización, tampoco existe información en movimiento que pueda conllevar a dudas o problemas a leerla.

Por otro lado el tiempo no es parte esencial del evento o actividad presentada por el contenido exceptuando los multimedia sincronizados no interactivos y los eventos en tiempo real.

También tengo en cuenta que cuando expira una sesión autenticada, el usuario puede continuar la actividad sin pérdida de datos tras volver a identificarse.

La página no contiene ningún apartado que pueda causar ataques epilépticos, espasmos o convulsiones entre otros.

A parte los enlaces son descriptivos y su nombre indica la función que realizan.

El contenido está bien encapsulado y separado por secciones.

Comprendibilidad

El idioma de la página se corresponde al target de usuarios que la utilizaran. A parte, el nivel de lectura de la página es óptimo puesto que apenas contiene texto.

Por otro lado se proporcionan etiquetas o instrucciones cuando el contenido requiere la introducción de datos por parte del usuario.

Robustez

En los contenidos implementados mediante el uso de lenguajes de marcas, los elementos tienen las etiquetas de apertura y cierre completos; los elementos están imbricados de acuerdo a sus especificaciones; los elementos no incluidos atributos duplicados y los ID son únicos, excepto cuando las especificaciones específicas de estas características.

Prototipado

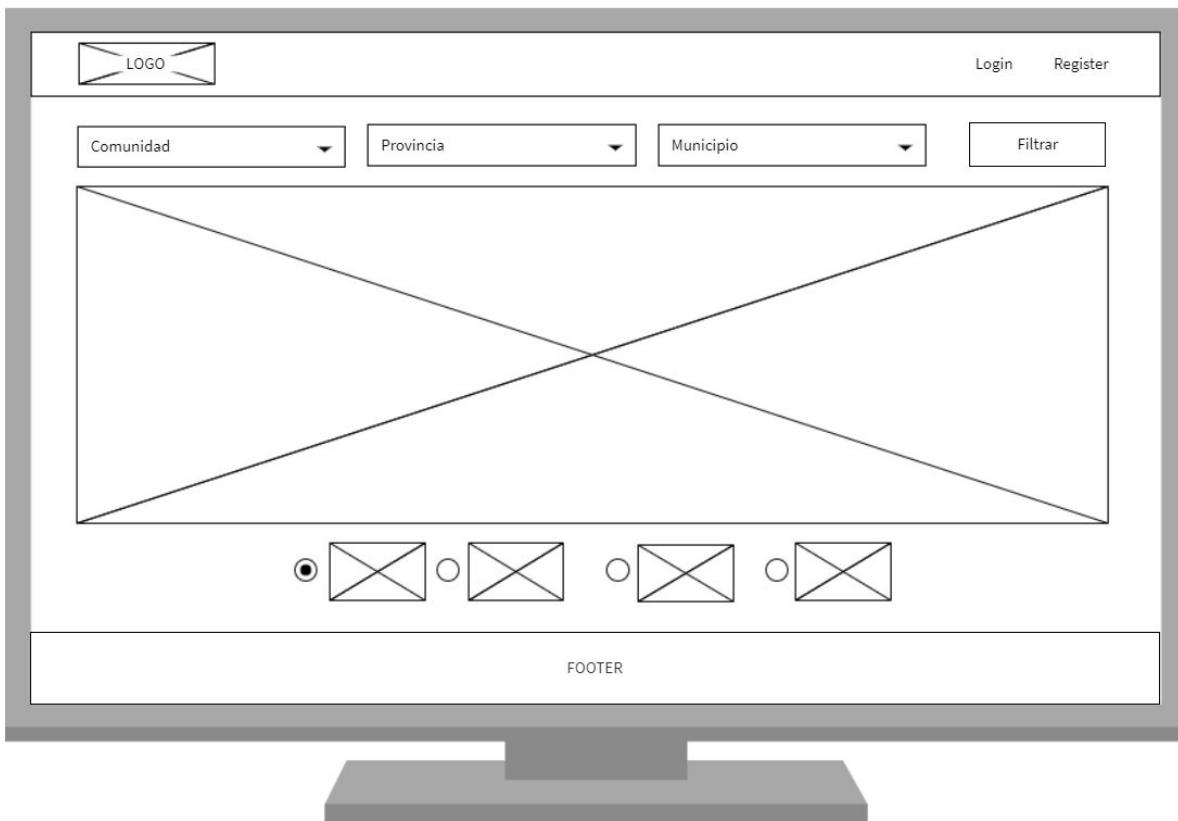
Wireframes

Un wireframe para un sitio web, también conocido como un esquema de página o plano de pantalla, es una guía visual que representa el esqueleto o estructura visual de un sitio web.

Para los wireframes he realizado varios modelos, tanto para ordenador como tablet y teléfono.

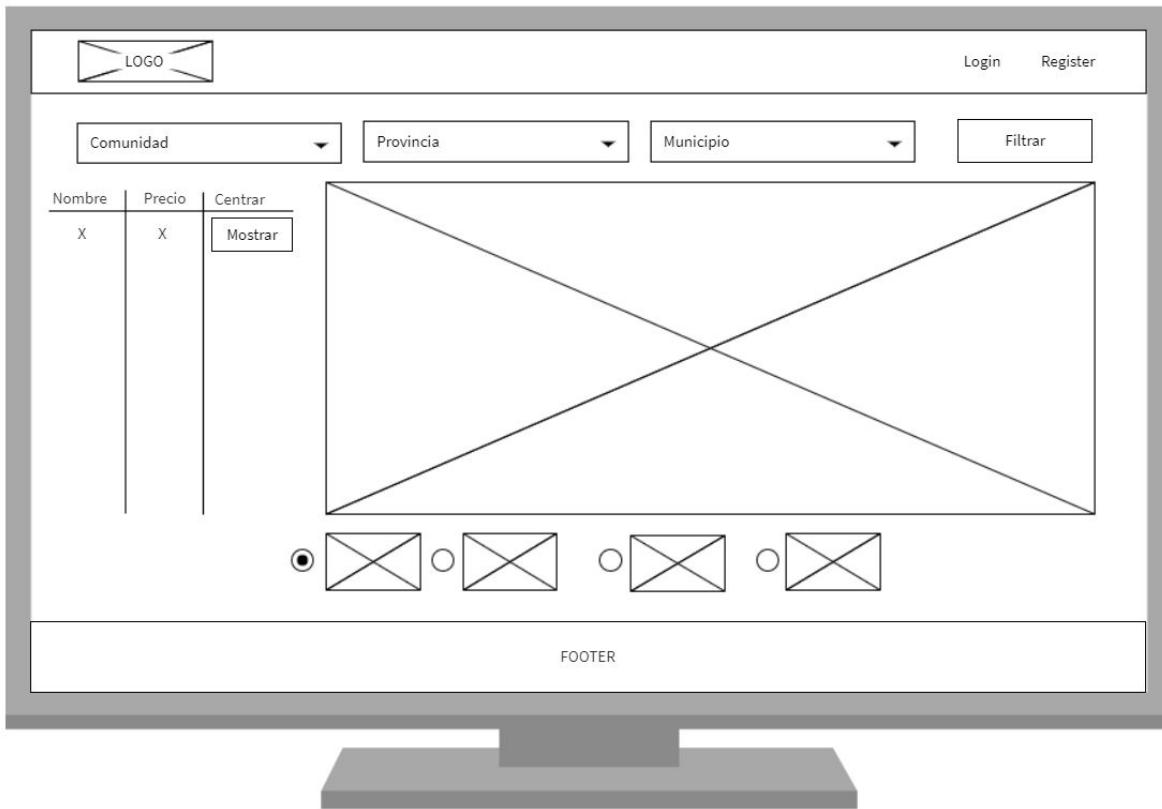
Para cada dispositivo he realizado un modelo antes de usar la herramienta de filtrar y otro modelo para cuando se haya filtrado.

Ordenador antes de filtrar:

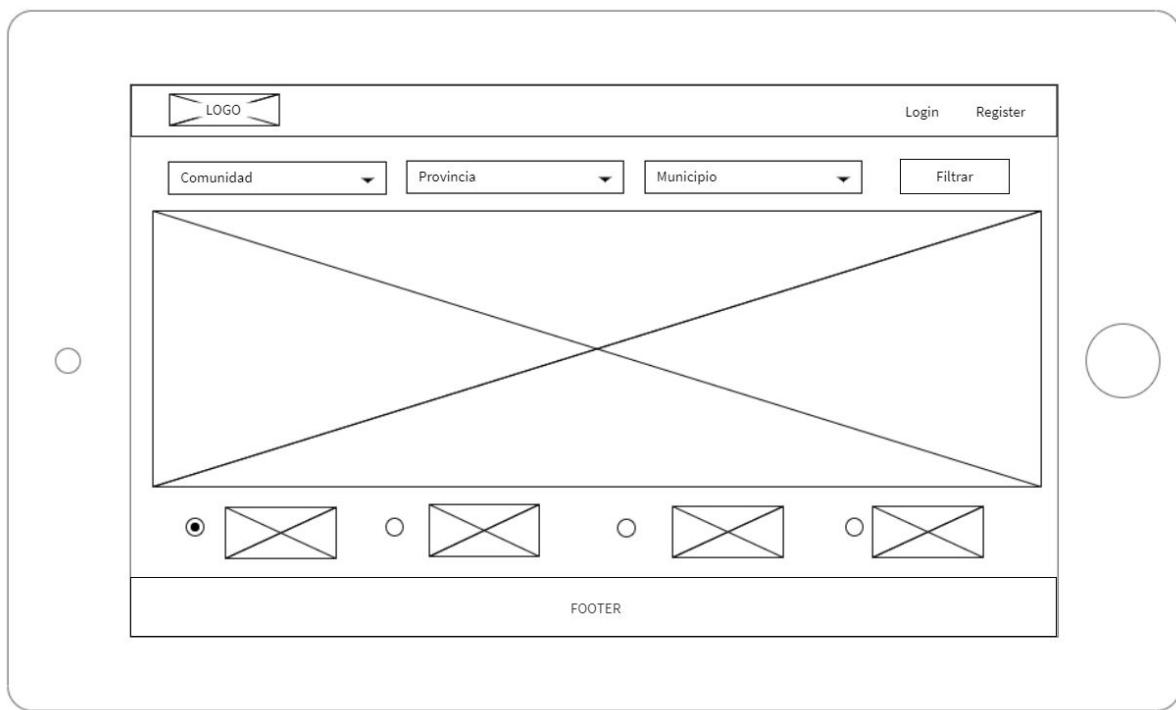


Podemos observar el logo y los botones de registro y login en la parte superior, mientras que si bajamos un poco encontraremos los filtros de localidad.

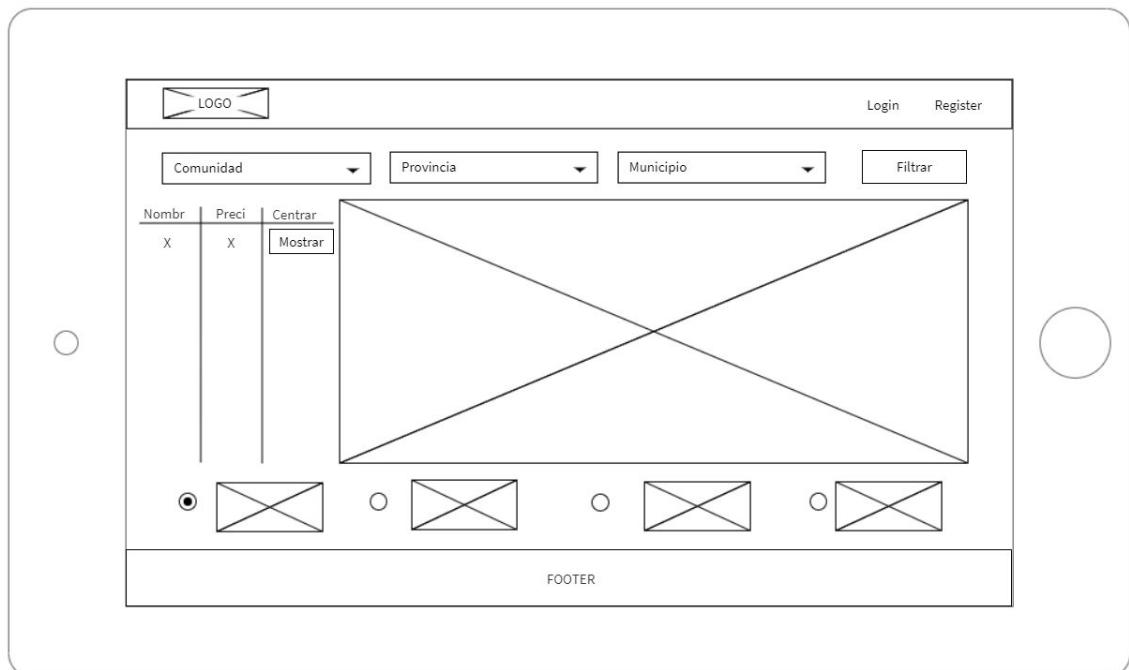
En la parte central encontramos ocupando todo el ancho el mapa, mientras que en la parte inferior encontraremos los filtros de carburantes y el footer.

Ordenador después de filtrar:

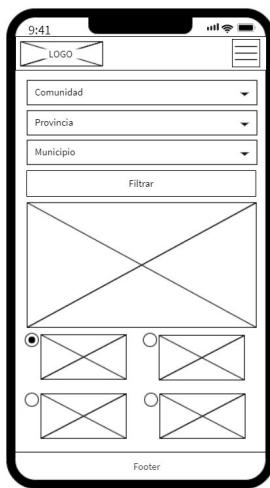
En este segundo modelo, podemos observar que el mapa ha reducido su tamaño $\frac{1}{4}$ dando lugar a una tabla que mostrará los datos de las estaciones de servicio que cumplan los requisitos estipulados por el usuario.

Tablet antes de filtrar:

La principal diferencia respecto a la versión de ordenador es que los filtros de carburantes ahora tienen menos espacio lateral.

Tablet después de filtrar:

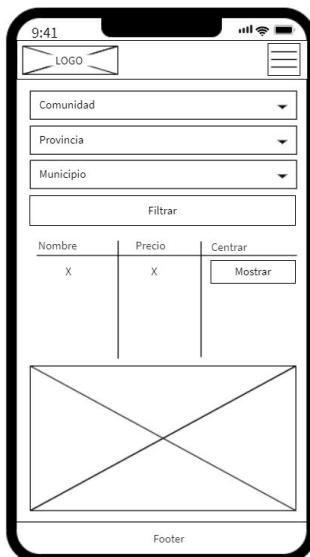
Al igual que en el caso anterior la principal diferencia respecto a la versión de ordenador es que los filtros de carburantes ahora tienen menos espacio lateral. También podremos ver que la tabla lateral es un poco más estrecha.

Móvil antes de filtrar:


En el formato de teléfono si que podemos encontrar más diferencias, empezando por los botones de registro, lo cuales ahora se encuentran en un hamburger menú.

Por otro lado, los filtros de localización ahora se muestran de forma vertical y ocupando todo el ancho de pantalla, mientras que el mapa sigue siendo el contenido central de la aplicación web.

Finalmente los filtros de carburante se distribuyen en 2 columnas.

Móvil antes de filtrar:


Una vez filtrado los resultados, la tabla se mostrará entre el formulario de localización y el mapa, ocupando todo el ancho de la pantalla.

Paleta de colores

La paleta de colores que he utilizado son simples, he escogido una escala de grises de base partiendo del blanco hasta llegar al negro, lo he combinado con el logo rojo y el botón azul para dar variedad y un toque distintivo a la página para que no se vea monótona.



Logo

El logo utilizado combina el nombre de la web con POI (Punto de Interés) el cual hace referencia al mapa y todos los puntos que contiene, el color rojo era para darle variedad a la página.



Iconos

Todos los marcadores que se ven en pantalla han sido creados manualmente. En España existen más de 3000 estaciones de servicio de distintas compañías, así que me he limitado a realizar los iconos de las gasolineras más conocidas y de las que tengo más cerca.

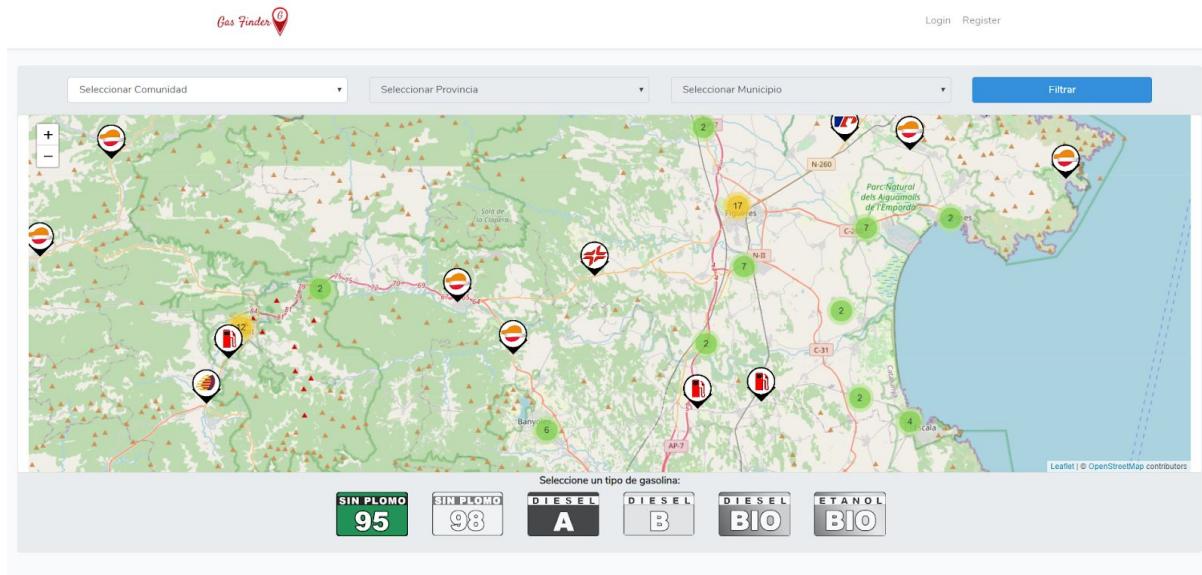


Responsive (RWD)

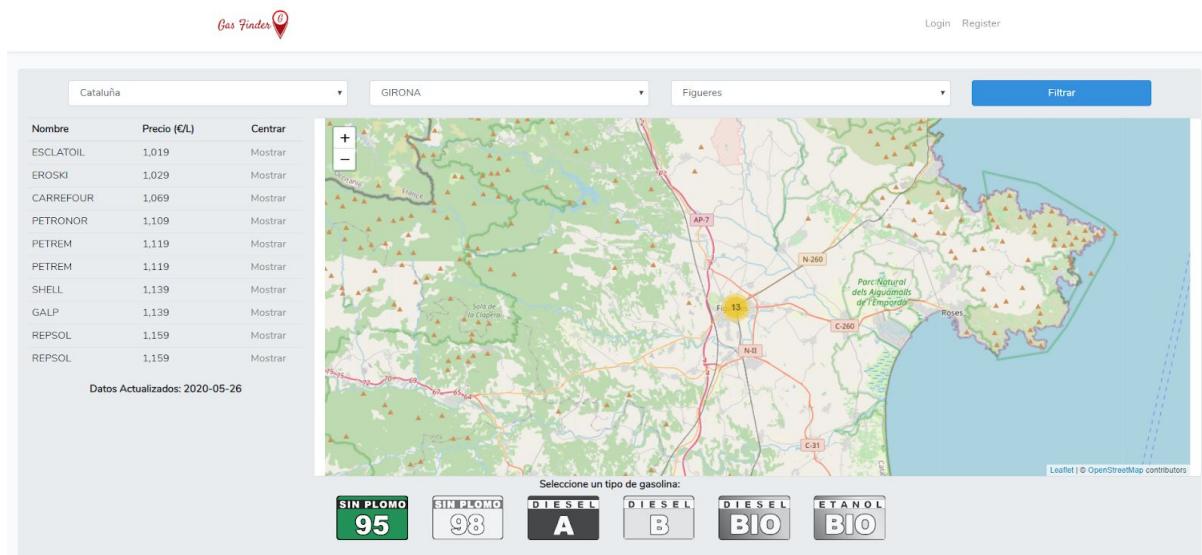
Tal y como hemos visto en los wireframes, la página se adapta a casi cualquier dispositivo, eso se debe al uso de bootstrap juntamente con CSS y GRID.

A continuación podemos ver los resultados.

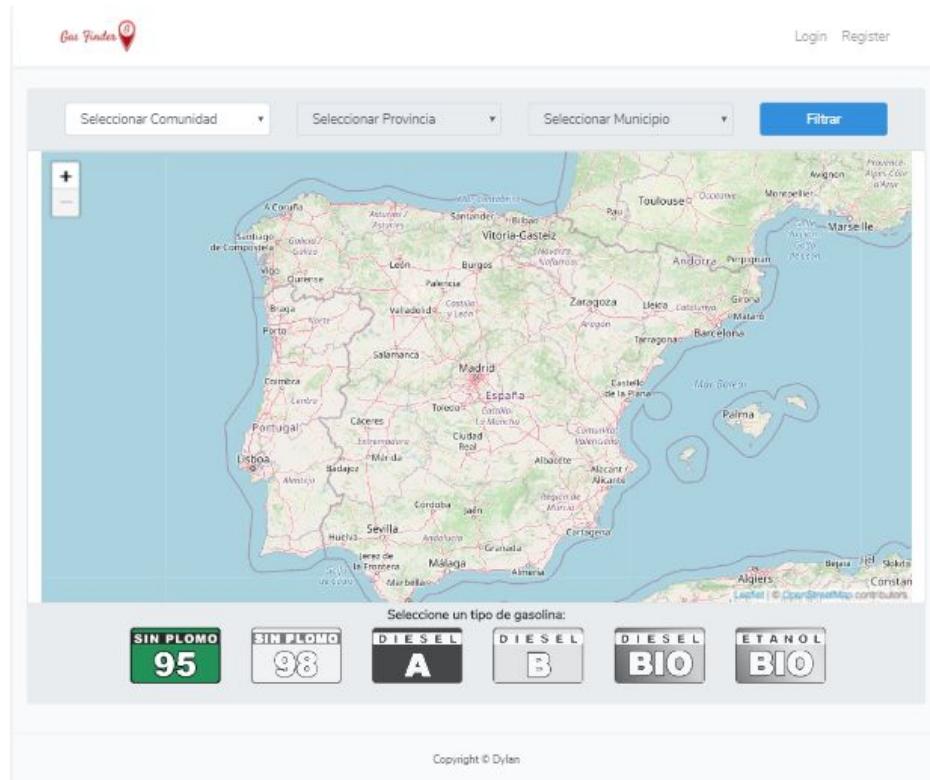
Ordenador antes de filtrar:



Ordenador después de filtrar:



Tablet antes de filtrar:



Gas Finder

Login Register

Seleccionar Comunidad

Seleccionar Provincia

Seleccionar Municipio

Filtrar

SIN PLOMO 95

SIN PLOMO 98

DIESEL A

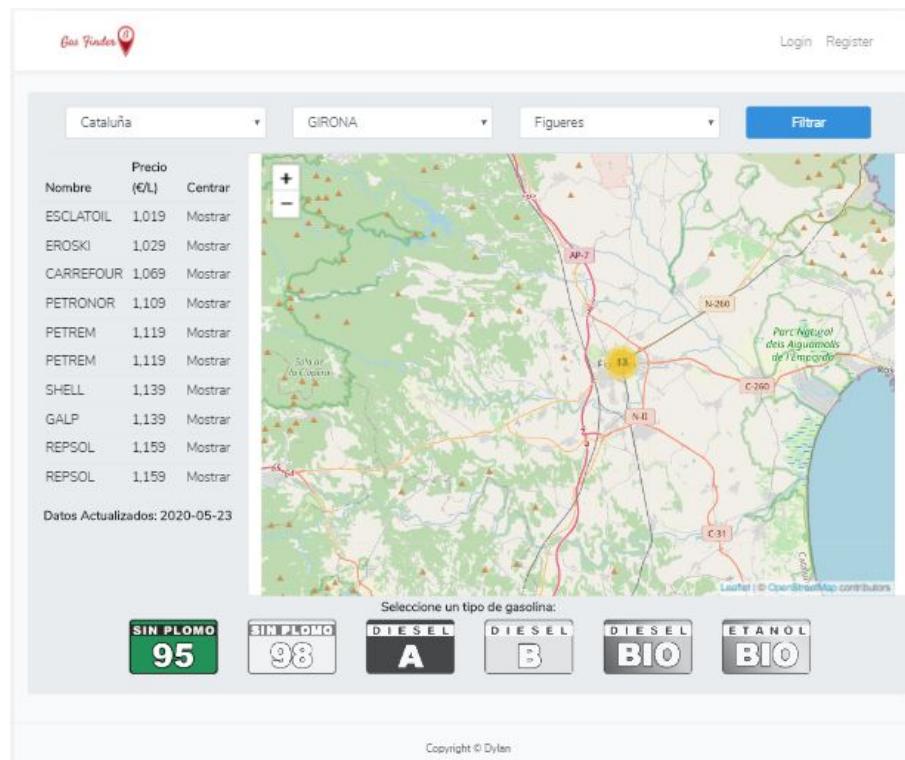
DIESEL B

DIESEL BIO

ETANOL BIO

Copyright © Dylan

Tablet después de filtrar:



Gas Finder

Login Register

Cataluña

GIRONA

Figueres

Filtrar

Nombre	Precio (€/L)	Centrar
ESCLATOIL	1.019	Mostrar
EROSKI	1.029	Mostrar
CARREFOUR	1.069	Mostrar
PETRONOR	1.109	Mostrar
PETREM	1.119	Mostrar
PETREM	1.119	Mostrar
SHELL	1.139	Mostrar
GALP	1.139	Mostrar
REPSOL	1.159	Mostrar
REPSOL	1.159	Mostrar

Datos Actualizados: 2020-05-23

SIN PLOMO 95

SIN PLOMO 98

DIESEL A

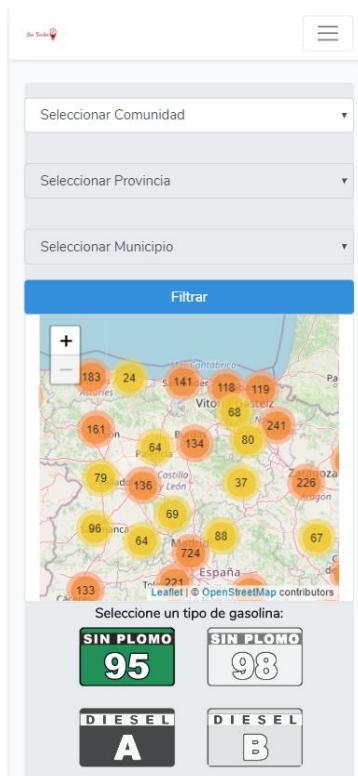
DIESEL B

DIESEL BIO

ETANOL BIO

Copyright © Dylan

Móvil antes de filtrar:



Móvil después de filtrar:



CSS GRID

Para realizar el sitio web, hacer que se adapte a los distintos dispositivos y a las modificaciones como por ejemplo la tabla de gasolineras, que aparece después de utilizar el botón de filtro, he utilizado la herramienta de CSS GRID pero con el framework de Bootstrap, que proporciona clases ya definidas para gestionar los diversos grids.

A continuación podemos observar la utilización de las columnas grid:

```
<div class="col-md-12">
    <div class="jumbotron p-0">

        <div class="row align-items-center justify-content-center">
            <div class="col-md-3 pt-3">
                <div class="form-group">
                    <select id="comarca" class="form-control">
                        <option disabled selected>Seleccionar Comunidad</option>
                        <option value='Todas'>Todas</option>
                        @foreach ($comunidades as $comunidad)
                            <option value='{{ $comunidad->Id }}'>{{ $comunidad->Nombre }}</option>
                        @endforeach
                    </select>
                </div>
            </div>
            <div class="col-md-3 pt-3">
                <div class="form-group">
                    <select id="provincia" class="form-control">
                        <option disabled selected>Seleccionar Provincia</option>
                    </select>
                </div>
            </div>
        </div>
    </div>
```

También podemos ver cómo he realizado con JQuery el cambio de clases para quitar columnas al mapa y dárselas a la tabla de gasolineras

```
$("#boxlist").addClass( "col-md-3" );
$("#boxmap").removeClass( "col-md-12" ).addClass( "col-md-9" );
$("#tablelist").removeClass( "d-none" )
$('#tablelist > tbody').children().remove();
```

CSS3

Gran parte de los estilos utilizados ya vienen predefinidos con laravel, bootstrap o incluso la librería de leaflet.

Aun así, he realizado mi aportación y el formulario de los carburantes lo he realizado vía html/css convencional tal y como podemos ver en la siguiente imagen.

```
1 .cc-selector input{
2     margin:0; padding:0;
3     -webkit-appearance:none;
4     -moz-appearance:none;
5     appearance:none;
6 }
7 .PrecioGasoleoA{background-image:url(..//css/imagenes/DieselA.png);}
8 .PrecioGasoleoB{background-image:url(..//css/imagenes/DieselB.png);}
9 .PrecioGasolina98{background-image:url(..//css/imagenes/Gasolina98.png);}
10 .PrecioGasolina95{background-image:url(..//css/imagenes/Gasolina95.png);}
11 .PrecioBiodiesel{background-image:url(..//css/imagenes/BioDiesel.png);}
12 .PrecioBioetanol{background-image:url(..//css/imagenes/BioEtanol.png);}
13
14 .cc-selector input:active +.drinkcard-cc{opacity: .9;}
15 .cc-selector input:checked +.drinkcard-cc{
16     -webkit-filter: none;
17     -moz-filter: none;
18     filter: none;
19 }
20 .drinkcard-cc{
21     cursor:pointer;
22     background-size:contain;
23     background-repeat:no-repeat;
24 }
```

He creado diversas clases y les he añadido una imágenes de fondo, que realizado en el GIMP y otras clases para que actúen de forma distinta cuando marques un elemento, pases por encima, etc.

Bootstrap

Como ya he mencionado, he realizado la mayor parte de la maquetación con las herramientas que proporciona el framework de bootstrap.

En primer lugar he importado sus estilos.

```
<link href="asset/css/form.css" rel="stylesheet">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384f9cC3eOmAD0YC911o2t3818ptpcFk4l42" crossorigin="anonymous">
```

Y a continuación sus scripts.

```
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENmM7KCkR/rE9/Qpg6oJZBwqHgXWzjyKUOOGdQ" crossorigin="anonymous">
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y2wAeB7OlQ4d8sJ12uJZt9J/lnXp" crossorigin="anonymous">
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8j" crossorigin="anonymous">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

Finalmente he ido maquetando y haciendo pruebas con diversas clases y estilos para finalmente obtener una app web responsive, funcional y útil.

```
<section>
<div class="container-fluid">
    <div class="row">
        <div class="col-md-12">
            <div class="jumbotron p-0">

                <div class="row align-items-center justify-content-center">
                    <div class="col-md-3 pt-3">
                        <div class="form-group">
                            <select id="comarca" class="form-control">
                                <option disabled selected>Seleccionar Comunidad</option>
                                <option value='Todas'>Todas</option>
                            </select>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

M11: Empresa Iniciativa Emprendedora

He considerado diversas opciones y considero que este proyecto se podría vender a diversas empresas para propósitos distintos.

La primera opción sería venderla a una compañía de gasolinera, donde podrían adaptar el código para mostrar únicamente sus estaciones de servicio algo así como un buscador privado.

Por otro lado, una alternativa similar sería que las compañías de estaciones de servicio de publiciten en mi sitio web, donde podrán publicar ofertas y promociones con las que me podría beneficiar cobrándoles una tarifa fija por anuncio por cierto volumen de usuarios.

Otra posible alternativa podría ser para una empresa de marketing o que gestione big data, con la que podrían procesar los datos de los usuarios (gasolineras más buscadas, zona donde se realizan más consultas, etc) hoy en día la información de una persona es muy valiosa y se puede obtener grandes beneficios en caso de tener un alto tráfico en la web. Sino que se lo digan a los de facebook o whatsapp.

Finalmente unas alternativas más a nivel personal, sin empresas de por medio, podrían ser una app de pago o una web app con banners publicitarios.

Valoraciones Personales

Después de 2 años cursando el ciclo superior de desarrollo de aplicaciones web, considero que este proyecto es un reflejo de lo que he llegado a aprender en este periodo.

Es obvio que al proyecto le quedan muchas mejoras para realizar, muchas horas puliendo el código y mejorando funcionalidades para que quede perfecto, pero aun así, el hecho de realizar un proyecto de forma personal y casi independiente (gracias al profesorado por ayudar y aguantarme) es significado de que estos últimos años han sido bien invertidos.

Tengo intención de seguir trabajando en este proyecto para que realmente sea una web / app útil para el consumidor.

También tengo intención de realizar proyectos nuevos y seguir aprendiendo nuevos lenguajes y conceptos que me conviertan en un profesional en el sector.

A nivel profesional, considero que el proyecto ha sido muy útil para potenciar los conocimientos sobre las herramientas y lenguajes que he aprendido estos años, he aprendido a utilizar nuevos mecanismos en laravel, javascript, java y en temas de despliegue . También he aprendido a que las bases de datos, los diseños y diagramas mejor hacerlos al principio, que es cuando tocan, que sino luego hay que cambiar todo y eso suele conllevar problemas.

Código Github

El repositorio de github se encuentra de forma pública en la siguiente URL:

[Repositorio Github](#)

También dejaré el servidor abierto para que se pueda consultar la web cuando sea necesario.

Tener en cuenta que el servidor no tiene muchas prestaciones y eso hace que para cargar los puntos de todas las gasolineras al iniciar tarde un poco. Si hubiera algún problema me avisáis.

[GAS FINDER](#)

Aprovecho a mencionar que todo el código ajeno al proyecto en sí (ejecutables de java, node, scripts, etc) se encuentran en las carpetas adjuntas a la documentación organizadas por módulos.

Bibliografía / Webgrafía utilizada

Solo he incluido los principales links que me han proporcionado información útil

Laravel y JS

<https://laracasts.com/discuss/channels/laravel/laravel-and-leaflet-mapping>

<https://packagist.org/packages/bavix/laravel-admin-leaflet>

<https://www.youtube.com/watch?v=SrKQNE1oevQ>

<https://stackoverflow.com/questions/60113860/how-to-crud-polygon-circle-in-leafletjs-with-laravel>

https://www.google.com/search?q=crud+leaflet+map+laravel&rlz=1C1GCEA_enES899ES899&oq=crud+leaflet+map+laravel&aqs=chrome..69i57j69i61.5091j0j7&sourceid=chrome&ie=UTF-8

<https://laravel.com/docs/4.2/eloquent>

<https://laravel.com/docs/7.x/requests>

<https://stackoverflow.com/questions/27141738/image-handling-in-laravel>

<https://stackoverflow.com/questions/17429427/laravel-eloquent-ordering-results-of-all>

<https://github.com/Leaflet/Leaflet/issues/1026>

<http://bl.ocks.org/uafrazier/d589caa322f1b1e7c651>

Desplegar

<https://www.youtube.com/watch?v=LJITkQxTKbA>

Programación

<https://www.youtube.com/watch?v=ESAUuQfdnRY&feature=youtu.be>

<https://www.oracle.com/technetwork/es/articles/java/api-java-para-json-2251318-esa.html>

<https://www.baeldung.com/java-http-request>

<https://www.baeldung.com/java-convert-inputstream-to-reader>

Base de Datos

(Los manuales del año pasado)

<https://sysgears.com/notes/dump-a-specific-table-or-few-rows-mysql/>

<https://shashankvivek.in/2016/08/09/mysql-dump-in-xml-format/>

<https://bugs.mysql.com/bug.php?id=94719>