# Desarrollo para Sistemas Distribuidos



Equipo: Thanos

Proyecto: Fuel Finder

Repositorio: <https://github.com/sistemawebpe/thanos>

**Integrantes:**

Julio Vizarreta – U20161A693

Fernando Lopez – U201724329

Jesús Villanueva – U201711049

Percy Minalaya – U20181B019

Horario: jueves 7 – 11 PM

Profesor: Héctor Saira

Ciclo: 4to

## Índice

[Desarrollo para Sistemas Distribuidos](#_t24wdjehsrnc)

[Índice](#_db108ghthpbf)

[Introducción](#_ast5fg7qqypl)

[Antecedentes](#_vdveba4rixnk)

[Objetivo del proyecto](#_e7ptcljh7ir1)

[Beneficios del proyecto](#_r0nsfnrl0c1a)

[Descripción del proceso](#_c5txe4bt48dr)

[Arquitectura del sistema](#_11jef75svxhw)

[Desarrollo del sistema](#_wwqgbfjqym8e)

[Conclusiones](#_3424w6c8xq6)

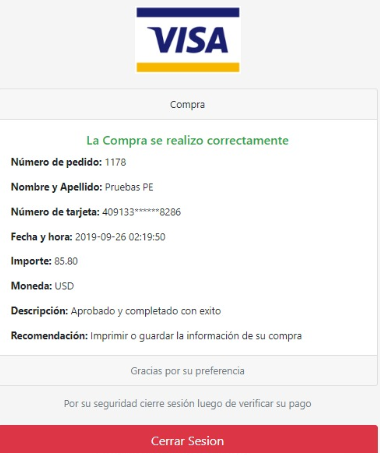
### 

### Introducción

Este proyecto está desarrollado bajo los lineamientos del curso en la cual nos enfocamos sobre servicios SOAP y REST y demostramos los principios de un sistema distribuido que nos permite trabajar cada una de las funciones que se han codificado con el apoyo de cada uno de los tópicos donde se establecen los fundamentos, orientación, Soap, Rest y mensajería.

### Antecedentes

Fuel Finder es una página web que está desarrollada en diferentes lenguajes de programación, por nombrar algunos como, PHP, Java Script, Python. La finalidad de esta solución es que sea utilizada por usuarios de internet que les permita consultar el precio de combustibles de las estaciones del front de la aplicación. Asimismo, lograr registrarse en el sistema de una forma amigable utilizando como dato obligatorio el correo electrónico la cual se almacenará en una base de datos MySQL interconectada dentro de la arquitectura de red siendo independiente a las reglas de negocio dando como sustento el principio de bajo acoplamiento y de esta manera tener habilitada la funcionalidad de comprar paquetes de anuncios publicitarios a través de una API externa de VISA corriendo en otra BD y los usuarios puedan adquirir publicidad a través de la página web de Fuel Finder. Por otro lado, existen los usuarios de las estaciones de combustible, las cuales tienen que registrarse en otra base de datos MySQL y poder ingresar, actualizar y eliminar datos del front, este nivel de acceso le permite a los usuarios de las estaciones de combustible crear nuevas estaciones y modificar los precios de los combustible, tal como se explicó en la diapositiva, poder competir por precio entre diferentes estaciones de servicios, el cual es una de las funcionalidades del sistema. Por otro lado, tenemos a los usuarios finales quienes a través del front puedan ver el mejor precio de combustible y, asimismo, poder adquirir paquetes de publicidad para sus propios negocios, tratando de emular los servicios que brindar Facebook, Outlook, Amazon, entre otros. Finalmente, nuestra aplicación nos permite utilizar colas para transmitir un mensaje de texto confirmando el registro de los usuarios en el sistema y un modal en el front confirmando la transacción de la compra por VISA.



### Objetivo del proyecto

Brindar al usuario final las alternativas de búsqueda de las estaciones de servicios donde se expenda el combustible con el precio mas económico de acuerdo a las condiciones de búsqueda.

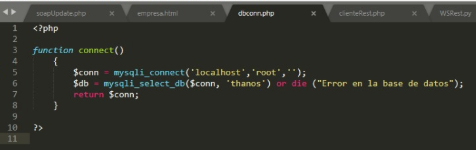
### Beneficios del proyecto

Este proyecto tiene como beneficio los siguientes principios:

**Servicio de abstracción:** *“Los contratos de servicio solo contienen información esencial y la información sobre servicios es limitado a lo que se publica en los contratos de servicio”*. De acuerdo con la teoría y llevado a la práctica dentro de la codificación los valores de las tablas y conexiones a la base de datos las mantenemos ocultas ya que el único que tiene conexión directa con la base de datos es el servicios y este recibe la información a través de variables sin la necesidad de exponer datos innecesarios que puedan vulnerar la seguridad e integridad del sistema.

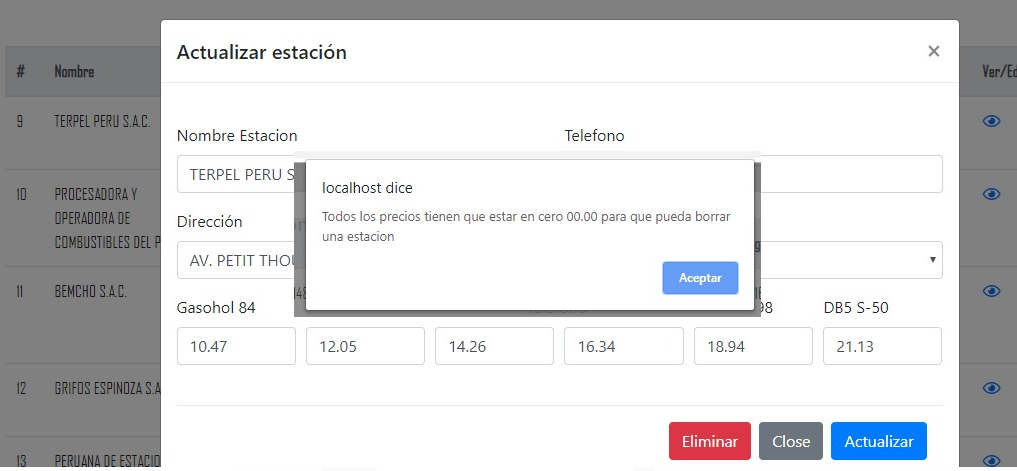
**Servicio de bajo acoplamiento:** “*Los contratos de servicio imponen un bajo requisito de acoplamiento del consumidor y están desacoplados de su entorno*”. Esto quiere decir que dentro de la lógica del negocio se pueden efectuar 3 tipo de operaciones, por ejemplo. Por un lado, se puede mostrar el precio de las estaciones en el front de la página sin necesidad de realizar login. Por otro lado, si una de las reglas del negocio se ven comprometidas el sistema trabaja de forma independiente para el registro y la búsqueda avanzada como la compra de la publicidad dándole continuidad al principio SOAP de bajo acoplamiento debido que cada servicio SOAP trabaja sobre una base de datos independiente.

**Servicio de reusabilidad:** *“Los servicios contienen y expresan lógica agnóstica y se puede posicionar como recursos empresariales reutilizables”*. Por ejemplo, dentro de nuestra codificación estamos utilizando una variable de conexión a la base de datos la cual es invocada desde el web services a través de PHP en cada



**Servicio estandarizado de contrato:** *"Los servicios dentro del mismo inventario de servicios cumplen con los mismos estándares de diseño del contrato".* Por esta razón dentro de la de las reglas del negocio se han considerado la siguiente estandarización:

* Para poder eliminar una estación de servicio esta debe tener los campos de precios en valor cero, de lo contrario el sistema no permite la desactivación o borrado de los datos en el SQL.



* Todos los usuarios deben tener el formato de correo, ingresar contraseña y validar la misma para la creación de la cuenta de acceso, caso contrario el sistema no permitirá el registro y solicitará que se ingresen estos valores obligatorios como regla de estandarización del negocio.



* La clave de los usuarios se encuentra encriptada a nivel de la base de datos para resguardar la información sensible del sistema.



### Arquitectura de negocio (procesos)

Descripción del proceso: El administrador de la página web puede generar reportes automatizados que permitan obtener las ventas del día, esta funcionalidad la conocemos con el Utilities del sistema distribuido, estos reportes automatizados nos muestran las ventas realizadas a través de la plataforma de visa con la finalidad de llevar la estadística de las transacciones.

Descripción del proceso: para la automatización de los mensajes de confirmación el usuario inicia sesión con el correo electrónico y recibe un correo electrónico confirmando su acceso al sistema, para ello se hace consultas al query de la base de datos donde se almacena la información requerida. Aquí un mensaje de correo electrónico.



### Arquitectura funcional (sistema)

La arquitectura funcional proporciona el conjunto de patrones utilizado para implementar las funciones de la empresa y las infraestructuras en las que se ejecutan estas funciones. Las áreas de funcionalidad de negocio de Fuel Finder son:

* Estaciones de servicios
* Registro de usuarios y empresas
* Pago Visa
* Mensajería

Todo el sistema está desarrollado sobre diferentes fuentes de programación, las cuales se detallan a continuación:

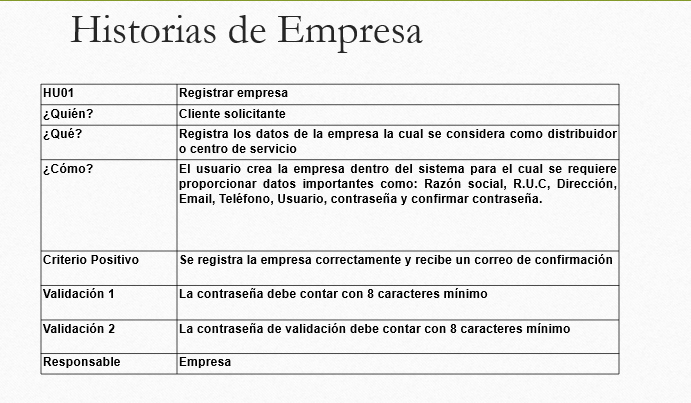
Python 2.7.16

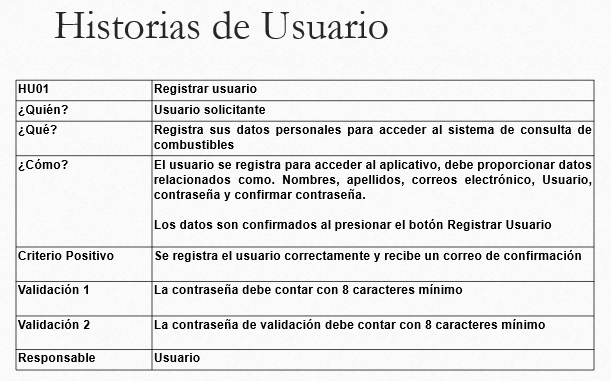
MySQL

Java Script

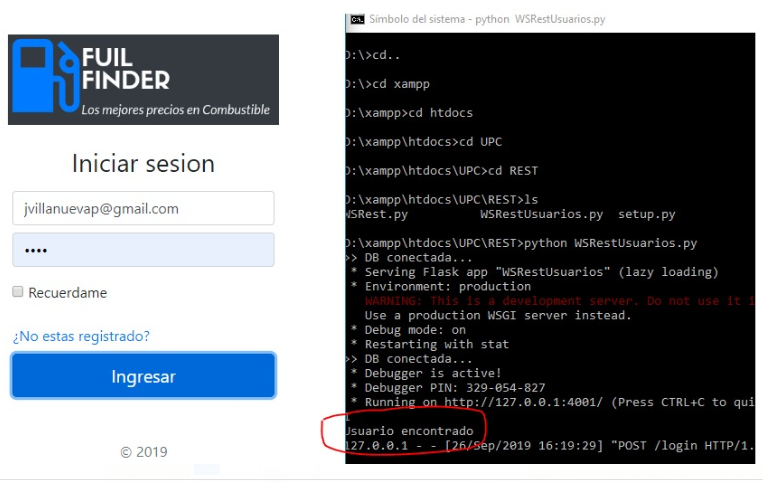
PHP

Para entender mejor el proceso vamos a describir como funciona el proceso del mantenimiento de empresa y usuarios, para ellos se han desarrollado una serie de preguntas claves que nos permiten entender mejor el proceso.

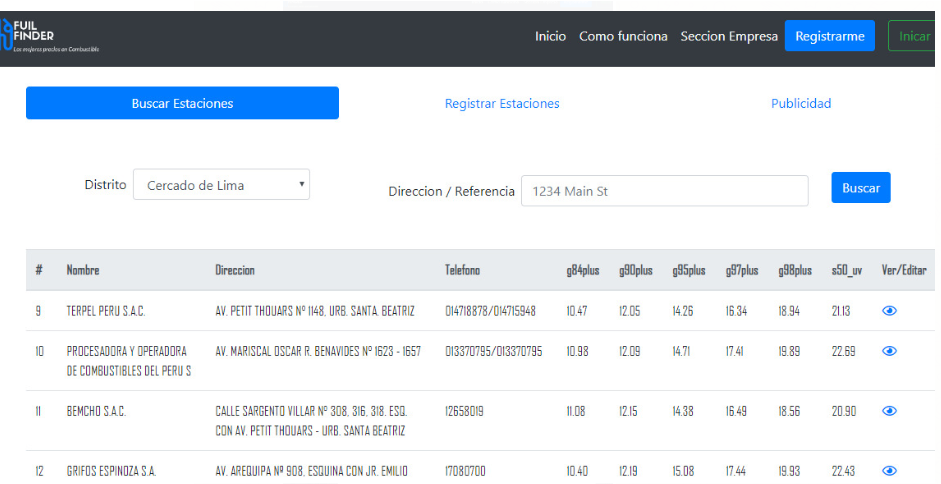




Después de estar creado el usuario se procede a validar el login, para el cual se realizan las pruebas sobre un servidor rest construido en Python 2.7.16 bajo SO Windows. En la siguiente imagen se evidencia que el usuario existe dentro de la base de datos por lo tanto se procede a brindarle acceso al sistema.



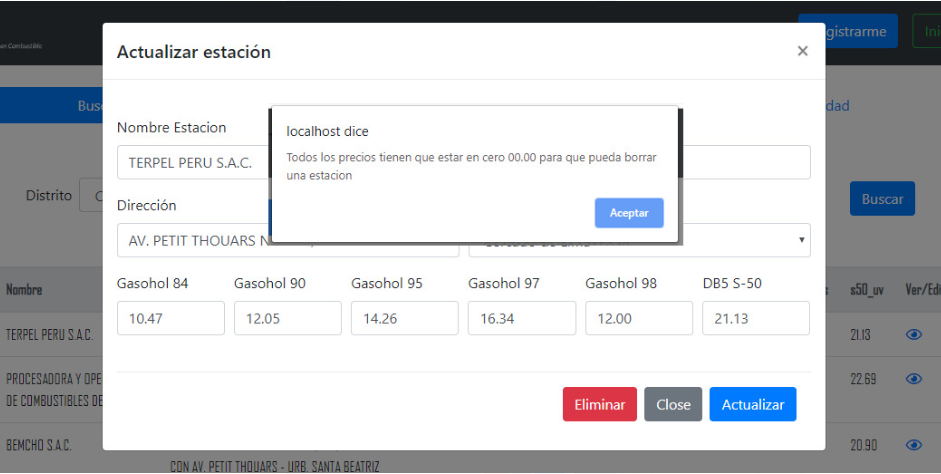
Se entiende que un usuario puede realizar consultas sobre el precio de las estaciones a través del front de la pagina web y se activan ciertos criterios de búsqueda como las listas que se encuentran desarrolladas en SOAP.



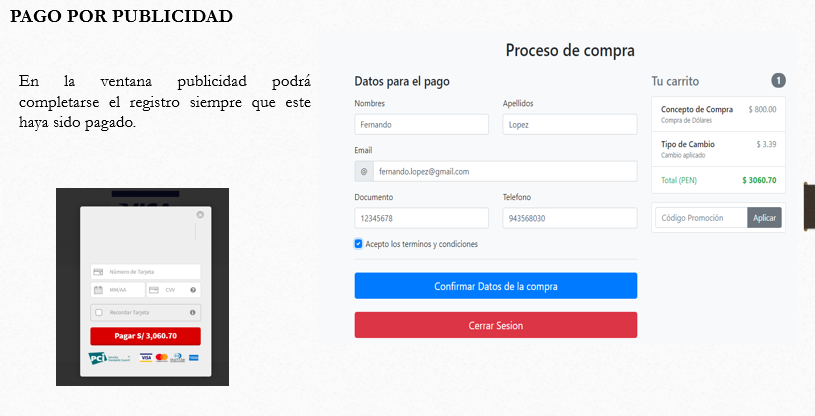
Una empresa puede realizar la actualización de los precios de los combustibles, crear nuevas estaciones de servicios.



Solo puede eliminar una estación siempre y cuando los valores del precio de los combustibles estén en S/. 0.00 caso contrario se impedirá su eliminación.



Respecto al pago por publicidad, esta interactúa directamente sobre el API de Visa cumpliendo el principio de



Respecto a los criterios de aceptación se entiende como las condiciones que nuestro producto de software debe satisfacer para ser aceptado por el usuario, cliente o una empresa. Para Fuel Finder, son estándares preestablecidos o requerimiento que un producto o proyecto debe satisfacer por lo tanto debemos especificar tantos requerimientos funcionales como no funcionales. A continuación, se listan los criterios de aceptación que se han contemplado en el proyecto.

1. Un usuario no puede registrarse sin antes ingresar su correo electrónico
2. La información del registro se almacena en la base de datos Thanos\_Usuarios.
3. Se envía un mensaje de correo informando al usuario que hace login satisfactoriamente.
4. El pago puede efectuarse con tarjeta de crédito.
5. Un usuario no puede registrarse con la misma información de otro usuario.
6. Una estación de servicio no se puede eliminar si no tiene los valores en S/. 0.00

### Arquitectura tecnológica (software)

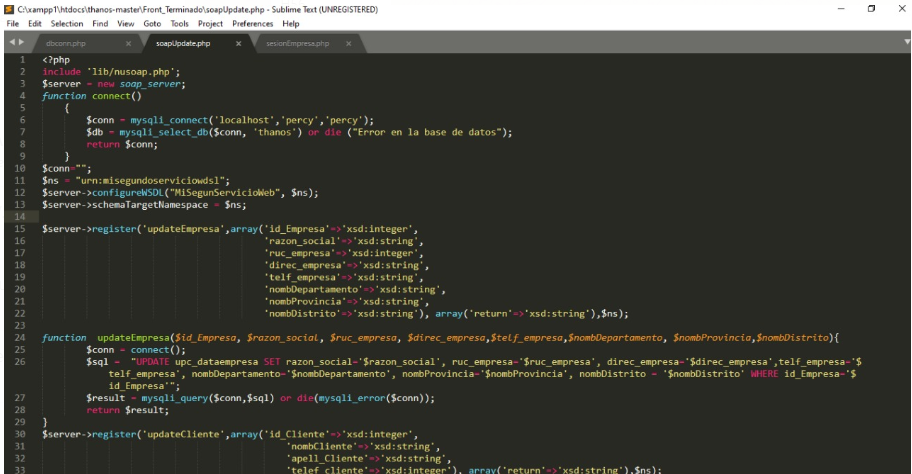
Modele la arquitectura de la solución. Identifique los nodos, esquema de alojamiento de cada nodo, los servicios que expone cada nodo con sus operaciones y las conexiones. Justifique la elección de las tecnologías de desarrollo distribuido a utilizar. Elabore un diagrama de despliegue del sistema utilizando UML o cualquier otra notación libre.

### Desarrollo del sistema

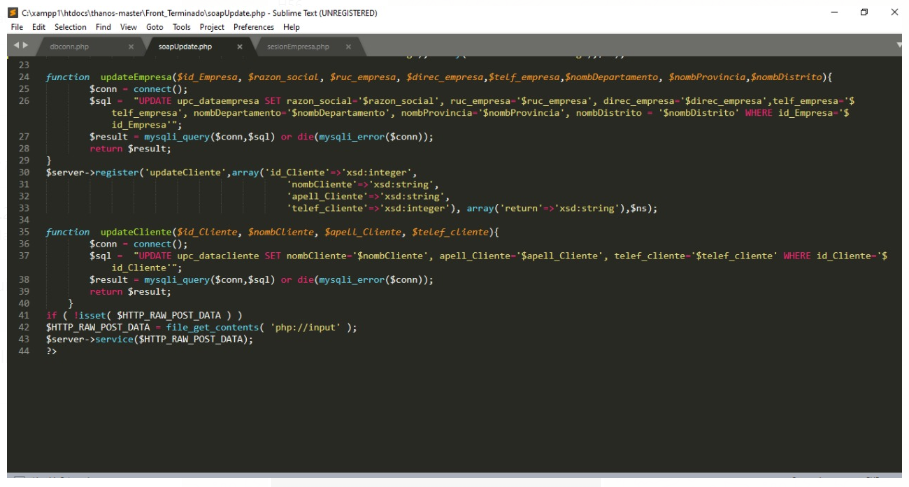
Se deberá ir dejando evidencia de los avances tanto para la entrega parcial como para la final (ej: pantallazos, pruebas, diagramas, enlaces a videos, etc.)

En la programación de los servicios se dejan evidencias de la codificación.

Aquí mostramos la programacion del servicio Soap para actualizar empresa

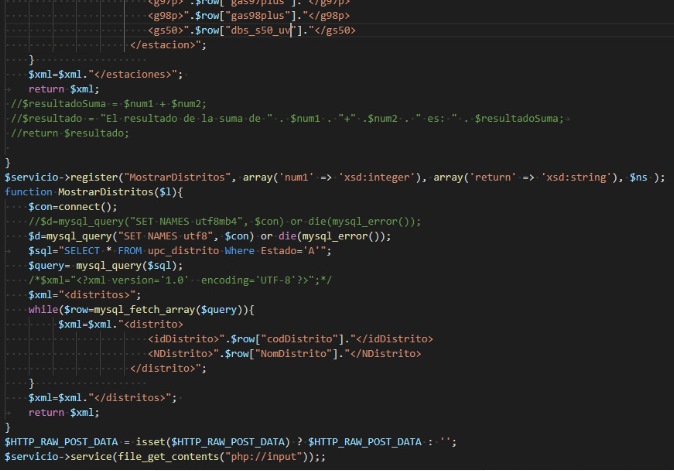


Aquí mostramos la programacion del servicio Soap para actualizar cliente



Aquí mostramos la programacion del servicio Rest para mostrar distrito.





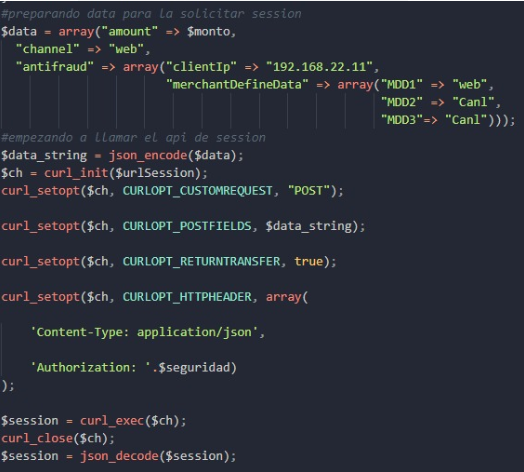
Prueba del servicio Soap que muestra distrito



Aquí mostramos la programacion del Api Rest externo.



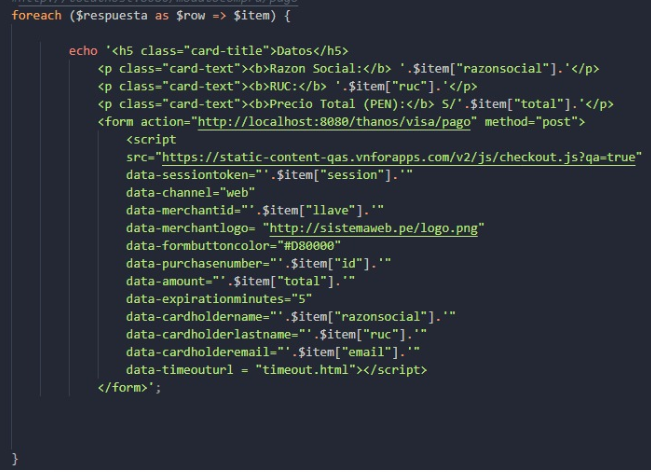
Mostramos la codificacion que prepara la data para solicitar sesion y llamar al api de sesion



Funcion para mostrar resumen de la compra y formulario de visa.

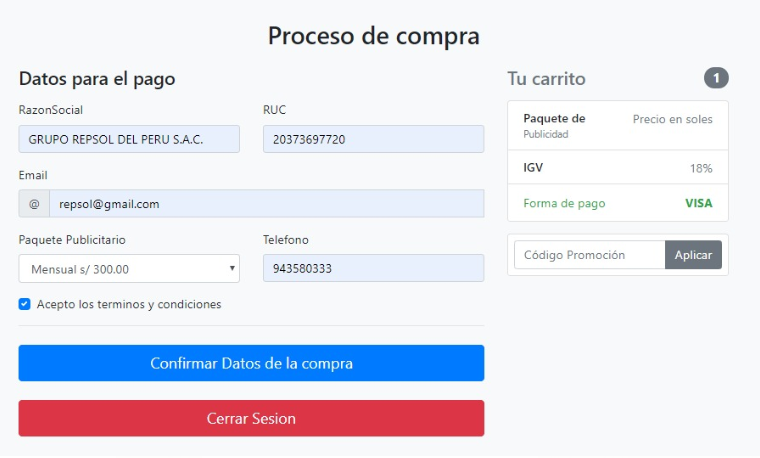


En esta session se muestran los datos que se van a cargar

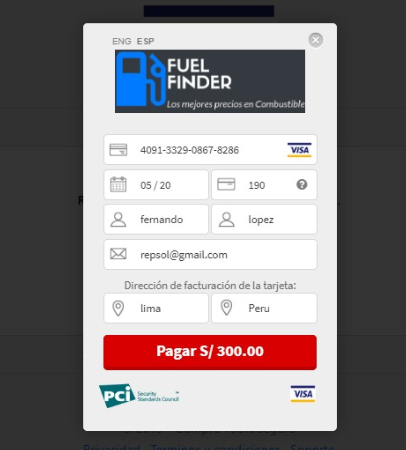


**PROCESO DE COMPRA:**

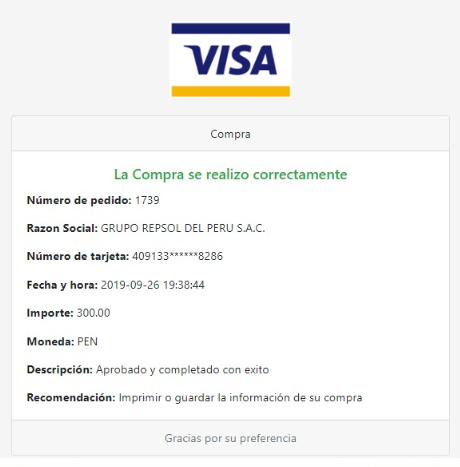
**Paso 1: Proceso de compra**



**Paso 2: Ingresar los datos de la tarjeta para efectuar el pago.**



**Paso 3: Confirmacion de la compra**



Enlace de video Youtube del avance del proyecto:

<https://www.youtube.com/watch?v=GYMv_4ttRPw>

**INTEGRACION DE SISTEMA:**

El sistema se integra a través de dos nodos integrando el sistema a través de MySQl y todos los lenguajes de programación descritos en el documento.

Vamos a presentar el proceso de funcionalidad del proyecto a través de las siguientes imágenes.

### Conclusiones

Al concluir con el desarrollo del sistema integrado que hemos venido trabajando a lo largo de todo el modulo, logramos integrar los diferentes lenguages de programación en el proyecto final del grupo Thanos con el proyecto Fuel Finder . Se puede evidenciar que la construccion de un servicio Soap nos genera mayor curva de aprendizaje y dificultad en la integracion del proyecto. Por otro lado, según los ejercicios realizados en laboratorio a traves de .Net se puede considerar que ID esta mejor preparado para un desarrollo mas agil. Finalmente, el proyecto se integra y funciona correctamente consumiendo api Rest, Soap y Api externas de pago y mensajeria.