e-Commerce

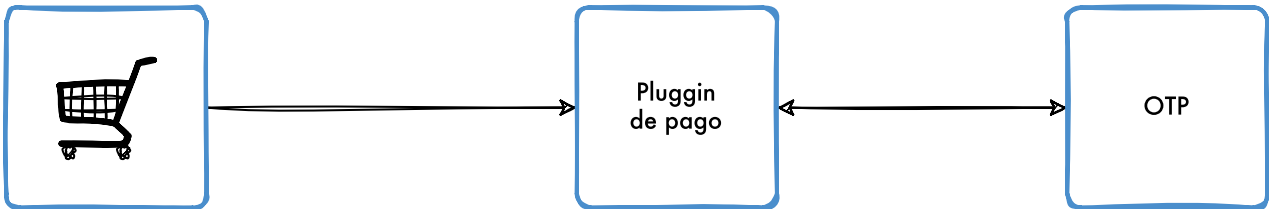
Documentación del desarrollo de plugins en diferentes plataformas e-commerce

🛍️ PLATAFORMAS E-COMMERCE MÀS CONOCIDAS

- VTEX
- Magento
- Shopify
- OpenCart

🎯 CUAL ES EL OBJETIVO DE CREAR UN PLUGGIN DE PAGOS?

La siguiente imagen detalla el flujo general del pluggin de pagos utilizando el OTP:

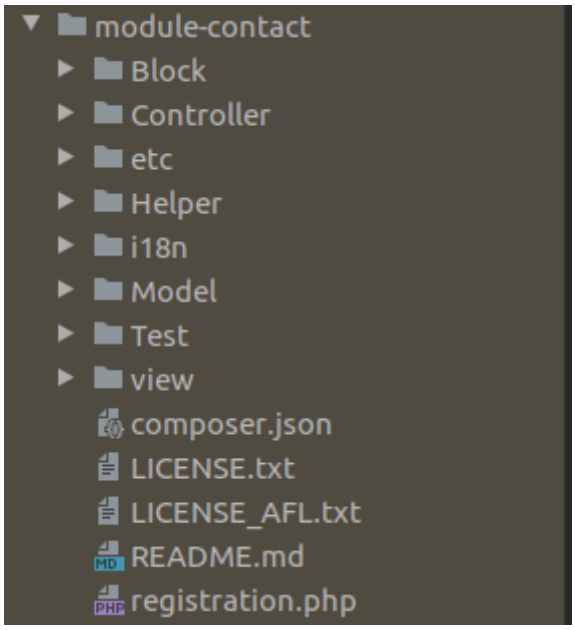


1. El comprador agregará los artículos a su carrito de compras. Para generar la compra, irá al carrito y procederá a generar el pago.
2. El comprador podrá elegir el método de pago (pluggin de pagos desarrollado), e ingresará los datos requeridos
3. Al finalizar clickeará en “Pagar”. Ésto generará una comunicación con OTP, quien aprobará o rechazará la compra.

📌 MAGENTO

Página oficial de desarrolladores: https://devdocs.magento.com/

📂 ESTRUCTURA DE ARCHIVOS



🔷 **Block:** Contiene las clases .php que se encargan de la vista.

Recordemos que Magento en sus módulos utiliza el patrón MVC.

🔷 **Controller:** Contiene las clases .php de los controladores. Son clases que se encargan de llevar las peticiones del usuario a la lógica de programación para que ésta por detrás pueda ejecutar acciones en conjunto con sus demás componentes. Normalmente estas clases responderán a peticiones hechas desde el navegador por POST y GET

🔷 **etc:** Contiene archivos XML que representan estructuras de configuración de nuestros módulos. Estos archivos que nos permiten aplicar configuraciones a nuestro backend en el área de administración.

Aquí se encuentra el archivo module.xml (el mas importante, sin este archivo no existiría un modulo).

🔷 **Helper:** Contiene clases generales que se reutilizan en modulo e incluso en otros módulos, es importante mencionar que estas clases pueden abstraer valores de configuración mediante métodos getter.

🔷 **i18n:** Contiene archivos .CSV que contendrán los paquete de traducción del modulo.

Recordemos que la plataforma de Magento es multilenguaje así que en este directorio van los archivos que pueden aplicar esa configuración de multilenguaje al modulo.

🔷 **Model:** Contiene las clases para las entidades, de recurso, colecciones y aquellas que se encargan de la capa de negocio. Aquí se generan estructuras de clases que representan a las tablas de la base de datos (aquí se construye el modelo de update, insert, delete y selección de datos de una forma no directa por SQL si no utilizando el ORM nativo de Magento).

🔷 **Test:** Contiene componentes con su propia estructura de archivos y carpetas que permiten realizar las pruebas unitarias al módulo que se esta creando.

🔷 **view:** Contiene los archivos que representaran la vista de cara al usuario final, son archivos de plantillas tanto para el frontend y backend del módulo representados por archivos de tipo .phtml y .html así como los archivos estáticos .js y .css.

🔷 **registration.php:** es un archivo php que permite registrar el modulo a la plataforma de Magento para que funcione conforme a lo definido en el modulo.