VPOS 2.O

Colaborando para construir tu negocio en internet

Especificaciones Técnicas

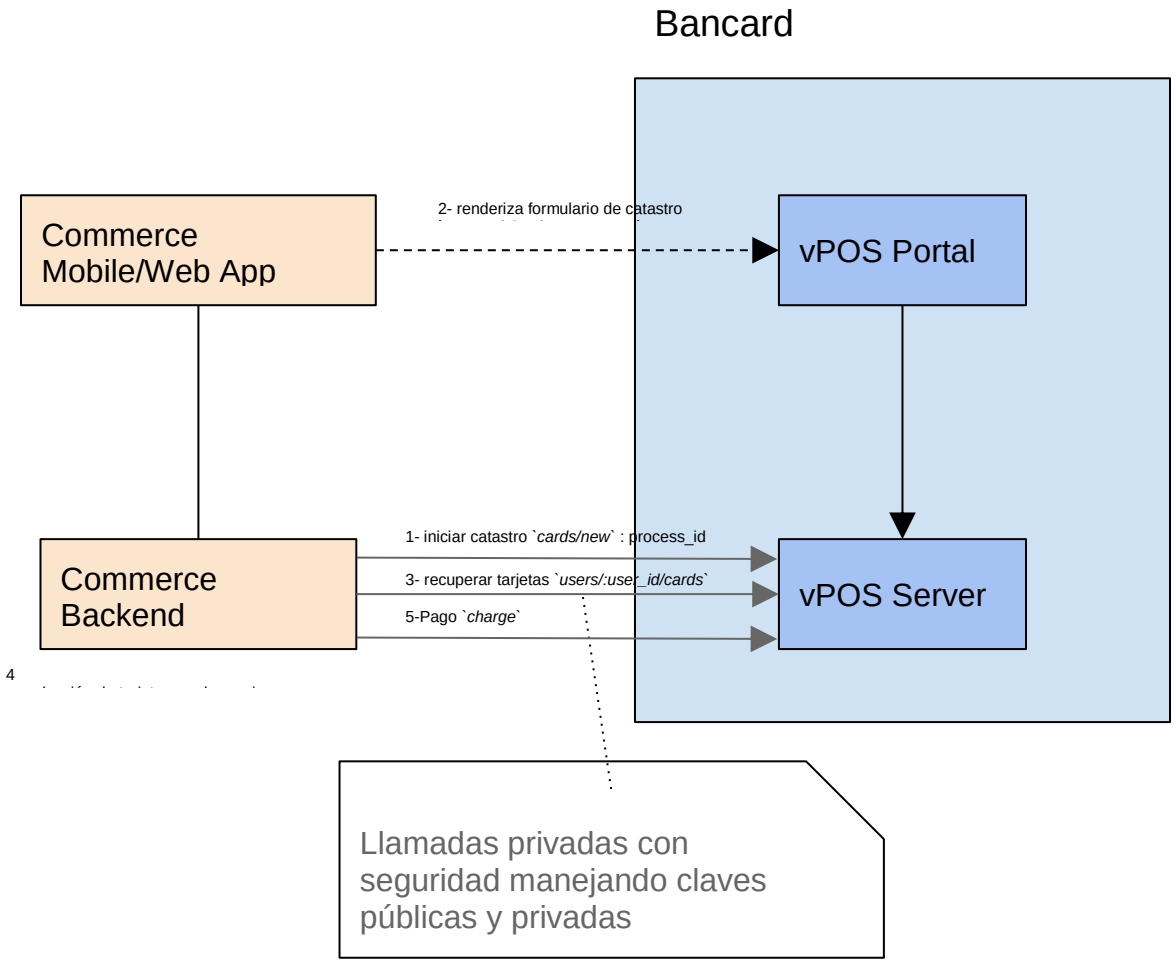
Bancard

Definiciones - Operaciones vPOS 2.0

Introducción

Este documento define las operaciones de vPOS 2.0. Estas operaciones le permiten a un comercio que un usuario pueda catastrar una tarjeta de crédito en la misma página del e-commerce, permitiendo utilizarla en futuros pagos sin el manejo de datos sensibles ni reingresar los datos de la tarjeta.

Arquitectura Planteada



Se plantea un e-commerce genérico con un backend (Commerce Backend), frontend web (Commerce Web App) y mobile (Commerce Mobile App). Las comercios pueden acceder a la API Rest de vPOS (vPOS Service) y al portal de vPOS (vPOS Portal) ambos dos instalados en Bancard cumpliendo las normas PCI.

La comunicación entre el backend del e-commerce y la api de vPOS se autentica por clave pública y privada. También al estar el portal de vPOS en la infraestructura de Bancard, y esté cumplir las normas PCI, se pueden ingresar número de tarjetas de crédito en el portal.

Formulario embebido en el sitio del comercio

El comercio tendrá la facilidad de embeber el formulario de pago en su sitio con estilos customizados a travez de un iframe.

vPOS 2.0 cuenta con un JavaScript para hacer la integración más fácil para el comercio, además cuenta con un panel de personalización del iframe en el portal de comercio de Bancard que hará mucho más fácil la customizacion del iframe.

Una vez obtenido el **process_id**, el usuario podrá incluir en su e-commerce un formulario de checkout embebido, de esta forma la compra se podrá finalizar en su propia aplicación. Para esto podrá utilizar la librería JavaScript como se indica en el siguiente repositorio de código.

El JavaScript para iframe de pago anonimo se encuentra publicado:
src=" Environment/checkout/javascript/dist/bancard-checkout-1.0.0.js">

Environment
- Producción - https://vpos.infonet.com.py
- Staging - https://vpos.infonet.com.py:8888

Para levantar el iframe:

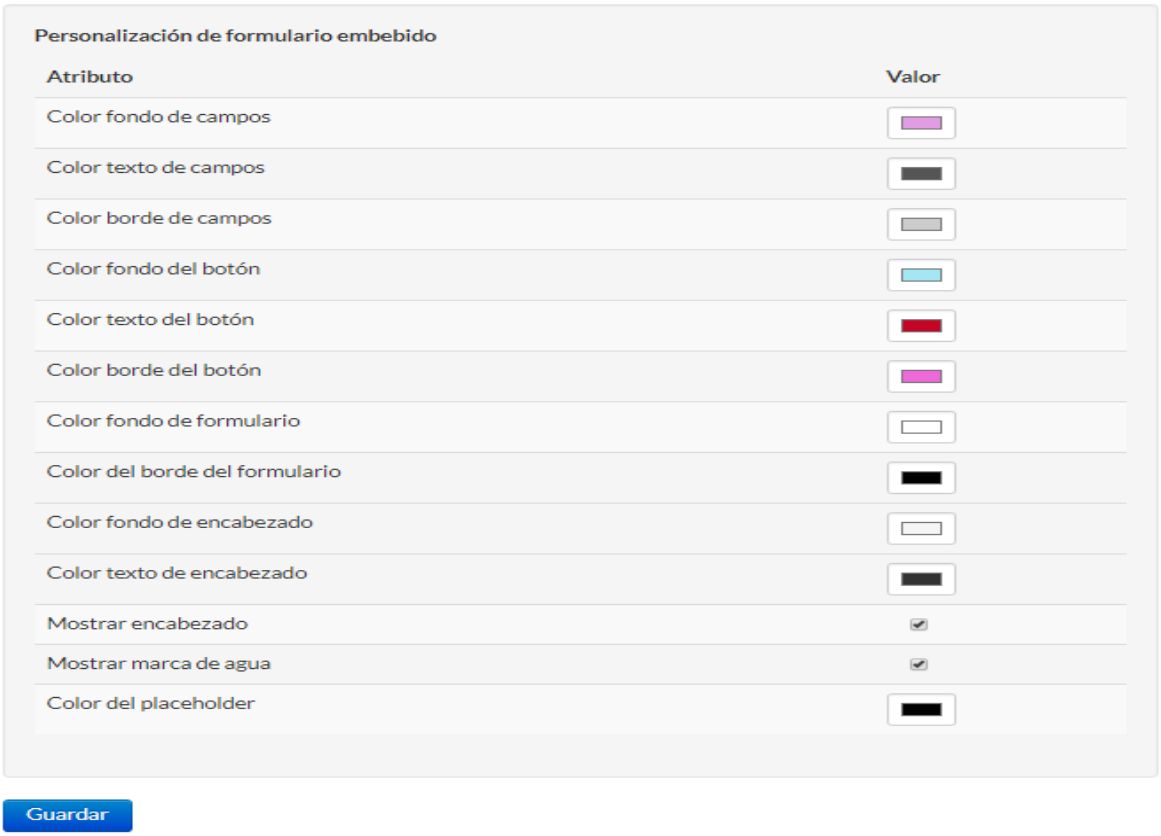
```
window.onload = function() {
    Bancard.Checkout.createForm('iframe-container', process_id ', styles);
  };
```

Ejemplo de código html

```
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <title>iFrame</title>

    <script        src="https://vpos.infonet.com.py:8888/checkout/javascript/dist/bancard-checkout-1.0.0.js"></script>
  </head>

  <script type="application/javascript">
```

```
    styles = {
      "form-background-color": "#001b60",
      "button-background-color": "#4faed1",
      "button-text-color": "#fcfcfc",
      "button-border-color": "#dddddd",
      "input-background-color": "#fcfcfc",
      "input-text-color": "#111111",
      "input-placeholder-color": "#111111"
    };

    window.onload = function() {
      Bancard.Checkout.createForm('iframe-container', 'WR-YY9JmxsEZV3hpVGA7', styles);
    };

  </script>

  <body>
    <h1 style="text-align: center">iFrame vPos</h1>
    <div style="height: 300px; width: 500px; margin: auto" id="iframe-container"></div>
  </body>

</html>
```

Panel de personalización



Experiencia de compra de un cliente en un sitio con el iframe

Operaciones

Vpos 2.0 contará con la opción de catastro de tarjetas dentro de un iframe de catastro siempre en el ambiente seguro de Bancard cumpliendo con las normas PCI.
Para poder catastrar una tarjeta, el servicio de vpos 2.0 se integra a un motor desarrollado en Bancard al cual llamamos “Tu eres tú”

Tu eres tú

El proyecto **es un sistema de KYC** para clientes de Bancard que permitirá obtener mediante fuentes internas unas series de preguntas relacionados con el cliente, para que este a su vez puedan ser contestadas mediante selecciones múltiples o que el mismo cliente pueda responder, tipeando la respuesta en cuadros de textos. Mediante este proceso podremos medir con el motor si el cliente final es quien dice ser dando un porcentaje de aciertos. Aplicando políticas establecidas y con parámetros exclusivos, podemos aplicar el tipo de complejidad necesaria por tipo de cliente.

Una vez que el motor de Tu eres tu dé una respuesta exitosa sobre el usuario y tarjeta, entonces podrá catastrar su tarjeta.
Para el catastro de tarjeta presentamos una serie de operaciones:

Catastro de Tarjeta

Para catastrar una tarjeta, el comercio deberá invocar a la siguiente operación.

La firma es:

```
POST {environment}/vpos/api/0.3/cards/new
```

Environment
- Producción - https://vpos.infonet.com.py
- Staging - https://vpos.infonet.com.py:8888

El `body` es:

```
{
    "public_key": "kR6oAQoIYCqUZLAivLQgac3lO7mv5bXZ",
    "operation": {
        "token": "69bd9ef382cb47e796ebe9f6b6b850ba",
        "card_id": 1,
        "user_id": 966389,
        "user_cell_phone": 0919876543,
        "user_mail": "gustavo.rolfi@gmail.com",
        "return_url": "http://micomercio.com/resultado/catastro",
    },
    "test_client": true
}
```

Los atributos `card_id`, `user_id`, `user_cell_phone` y `user_mail` son obligatorios y son brindados para asociar el pedido de catastro de tarjeta a un usuario con una referencia interna del comercio.

El `token` es:

```
md5( private_key + card_id + user_id + "request_new_card" )
```

La respuesta exitosa es:

```
{ "status": "success", "process_id": "i5fn*lx6niQel0QzWK1g" }
```

Invocar al iframe de catastro de tarjeta

El usuario podrá embeber dentro de su propio sitio o app un formulario para el ingreso de información sensible de tarjetas. Bancard creó una librería JavaScript para poder hacerlo de manera simple y transparente. Por más información bancard-checkout-js.

Una vez que se tiene el `process_id` los pasos para realizar la integración son:

1. Incluir bancard-checkout.js
2. Iniciar contenedor con código JavaScript

1. Incluir bancard-checkout.js

Para utilizar la librería *bancard-checkout.js* se debe incluir la misma utilizando, por ejemplo, el siguiente código:

```
<script src="bancard-checkout.js"></script>
```

2. Iniciar contenedor con código JavaScript

Para montar el formulario de catastro en el sitio web, se debe ejecutar `Bancard.Cards.createForm` indicando el id del contenedor, `process_id` y un conjunto de opciones que incluyen los estilos asociados al elemento embebido.

Ejemplo de invocación:

```
window.onload = function() {
  BancardCheckout.Cards.createForm('iframe-container','[PROCESS_ID]', options);
};
```

Para ver información de atributos de customización o más información sobre la integración recomendamos ir a la librería en github

Recuperar Tarjetas catastradas de un usuario

La app de vPOS podrá pedir las tarjetas catastradas por el usuario, para esto deberá invocar a la siguiente operación.

La firma es:

```
POST {environment}/vpos/api/0.3/users/user_id/cards
```

Environment

- Producción - https://vpos.infonet.com.py
- Staging - https://vpos.infonet.com.py:8888

El `body` es:

```
{
    "public_key": "kR6oAQoIYCqUZLAivLQgac3lO7mv5bXZ",
    "operation": {
        "token": "69bd9ef382cb47e796ebe9f6b6b850ba"
    },
    "test_client": true
}
```

El **`user_id`** debe ser el mismo que el comercio ingresó en la operación anterior (POST cards/new)

El `token` es:

```
md5( private_key + user_id + "request_user_cards" )
```

Que retorna las tarjetas que el usuario da de alta:

```
{
  "status": "success"
  "cards": [{
    "alias_token": "c8996fb92427ae41e4649b934ca495991b7852b855",
    "card_masked_number": "5418********0014",
    "expiration_date": "08/21",
    "card_brand": "MasterCard",
    "card_id": 1
  }...]
}
```

El token retornado permite realizar pagos con la tarjeta catastrada en una operación que se especifica a continuación.

Es importante destacar, que el token tiene validez para una sola operación y su tiempo de vida (ttl) es del orden de los minutos.

Pago con token

El comercio podrá establecer un cargo luego de obtener las tarjetas de un usuario, para esto deberá invocar a la siguiente operación.

La firma es:

```
POST {environment}/vpos/api/0.3/charge
```

Environment
- Producción - https://vpos.infonet.com.py
- Staging - https://vpos.infonet.com.py:8888

El `body` es:

```
{
    "public_key": "kR6oAQoIYCqUZLAivLQgac3lO7mv5bXZ",
    "operation": {
        "token": "f9aa075da613ee2b62e6712c1ed537f2",
        "shop_process_id": 60361,
        "amount": "723215.00",
        "number_of_payments": 12,
        "currency": "PYG",
        "additional_data": "",
        "description": "に到着を待 1",
        "alias_token": "c8996fb92427ae41e4649b934ca495991b7852b855"
    },
    "test_client": true
}
```

El `alias_token` es el obtenido al recuperar la lista de tarjetas de un usuario bajo el atributo con el mismo nombre.

El `token` de autenticación para esta operación es:

```
md5( private_key + shop_process_id + "charge" + amount + currency +
alias_token )
```

La respuesta exitosa es (misma respuesta que se recibe actualmente en la confirmación de un *single_buy*):


```
{
    "operation": {
        "token": "17defabc9c99d2d17a596939ffd8b042",
        "shop_process_id": 60361,
        "response": "S",
        "response_details": "Procesado Satisfactoriamente",
        "amount": "723215.00",
        "currency": "PYG",
        "authorization_number": "533916",
        "ticket_number": "270403039",
        "response_code": "00",
        "response_description": "Transaccion aprobada",
        "extended_response_description": null,
        "security_information": {
            "customer_ip": "201.217.159.98",
            "card_source": "L",
            "card_country": "PARAGUAY",
            "version": "0.3",
            "risk_index": 0
        }
    }
}
```


Eliminar tarjeta

Se podrá eliminar una tarjeta a un usuario, para esto se deberá invocar a la siguiente operación.

La firma es:

`DELETE` `{environment}/vpos/api/0.3/users/`**`user_id`**`/cards`

Environment
- Producción - https://vpos.infonet.com.py
- Staging - https://vpos.infonet.com.py:8888

El `body` es:

```
{
    "public_key": "kR6oAQoIYCqUZLAivLQgac3lO7mv5bXZ",
    "operation": {
        "token": "f9aa075da613ee2b62e6712c1ed537f2",
        "alias_token": "c8996fb92427ae41e4649b934ca495991b7852b855"
    },
    "test_client": true
}
```

El `alias_token` es el obtenido al recuperar la lista de tarjetas de un usuario bajo el atributo con el mismo nombre.
El **`user_id`** debe ser el mismo que el comercio ingresó en la operación anterior (POST cards/new)

El `token` es:

```
md5( private_key + "delete_card" + user_id + card_token )
```

La respuesta exitosa es:

```
{ "status": "success" }
```

Pago con token Multibuy

El comercio podrá establecer un cargo luego de obtener las tarjetas de un usuario, para esto deberá invocar a la siguiente operación.

La firma es:

```
POST {environment}/vpos/api/0.3/ multi/charge
```

Environment
- Producción - https://vpos.infonet.com.py
- Staging - https://vpos.infonet.com.py:8888

El `body` es:

```
{
    "public_key": "P2kJKsc87PWUXrmtGAhfSgVcs5Jzkr8F",
```

```
    "operation": {
        "token": "a9d3398eb2f4766a32683efa0e3adda5",
        "shop_process_id": 647744,
        "items": [
            {
                "name": "に到着を待 1",
                "store": 232,
                "store_branch": 77,
                "amount": "1.00",
                "currency": "PYG"
            },
            {
                "name": "に到着を待 1",
                "store": 232,
                "store_branch": 77,
                "amount": "2.00",
                "currency": "PYG"
            }

        ],
        "number_of_payments": 1,
        "additional_data": "",
        "alias_token":
"7ac25757a831d16175529183f3a36e804b5dd193a71bab7a6d464d0977a97314"
    }
}
```

El `alias_token` es el obtenido al recuperar la lista de tarjetas de un usuario bajo el atributo con el mismo nombre.

El `token` de autenticación para esta operación es:

```
md5( private_key + shop_process_id + "charge" + total_amount +
number_items )

total_amount es la suma de los amount de cada ítem
number_items es la cantidad de ítem dentro de items
```

La respuesta exitosa es la siguiente

```
{
        "status": "success",
        "confirmation": {
                "token": "5ec6e1b8b35c1f30fbd849b3e5dde5ba",
```

```
        "shop_process_id": 946797,
        "response": "S",
        "response_details": "Procesado Satisfactoriamente",
        "amount_in_us": "0,00",
        "amount_in_gs": "85,00",
        "additional_data": null,
        "number_of_items": 2,
        "items_approved": 2,
        "items": [{
                "amount": "26,00",
                "currency": "PYG",
                "store": "4",
                "store_branch": "46",
                "authorization_id": "270524909",
                "authorization_code": "565183",
                "response_code": "00",
                "response_description": "Transaccion aprobada",
                "extended_response_description": null
            },
            {
                "amount": "59,00",
                "currency": "PYG",
                "store": "4",
                "store_branch": "46",
                "authorization_id": "270524910",
                "authorization_code": "565184",
                "response_code": "00",
                "response_description": "Transaccion aprobada",
                "extended_response_description": null
            }
        ],
        "security_information": {
            "card_source": "L",
            "customer_ip": "127.0.0.1",
            "card_country": "PARAGUAY",
            "version": "0.3",
            "risk_index": 0
        }
    }
}
```