Proyecto Final: Diplomado de Desarrollo de Aplicaciones Móviles

Ing. David Arturo Martínez Guzmán

Facultad de Ingeniería

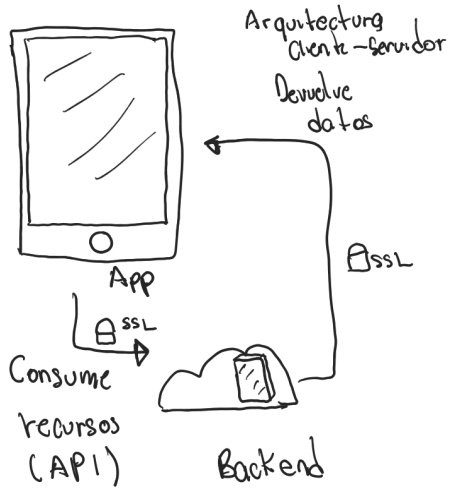
iOS Development Lab

Entrega: 11 de Agosto de 2019

Problema.
Actualmente me dedico al área de desarrollo, sin embargo, me gustan mucho los temas de emprendimiento.
Dado el contexto anterior, me di cuenta de la debilidad que existe en el mercado mexicano respecto a la venta de accesorios y dispositivos de marcas tecnológicas reconocidas en China u otros países del continente Asiático y Europeo.
Un claro ejemplo son las marcas tecnológicas Xiaomi y OnePlus.
Es muy difícil en México conseguir accesorios para dispositivos de las marcas mencionadas anteriormente, así como los mismos dispositivos.
Así como lo anterior, en el mercado hay muchas áreas de oportunidades de venta que quieren explotar emprendedores o pequeñas/medianas empresas.
El obstáculo más difícil es llevar acabo un proceso de logística de venta de artículos/servicios con la menor inversión posible, y esto es lo que nos ofrece internet.

Solución.
De acuerdo a un caso no aislado como el anterior, usando como medio internet y soluciones de software se puede llevar acabo la tarea de venta de artículos/servicios con la menor inversión posible y sin depender de terceros.
Es por ello, que ofrecer una solución integral mediante una aplicación móvil y una solución web se podría resolver el problema anterior.
Dato el contexto del diplomado, la solución se centró en el desarrollo de una aplicación móvil para iOS que consistió en un carrito de ventas genérico adaptable a cualquier negocio, con el fin de poder comercializarlo en un futuro cómo una solución integral.

Arquitectura.
En la solución se identificaron dos componentes claros: la aplicación móvil para el usuario final y con ello implicaba un componte externo que dotara de información la aplicación y permitiera los procesos de compra de artículos.
Con base en lo anterior, se optó por usar una arquitectura de cliente - servidor, donde nuestro cliente iba a ser la aplicación móvil y el servidor iba a ser una API o servicio web.
Los componentes quedaron de la siguiente manera:



Soluciones desarrolladas.

Aplicación móvil.

Se desarrollo una aplicación móvil con las siguientes características:

- Lenguaje de programación: Swift 4.2
- IDE: Xcode 10.2.1
- Aplicación sólo disponible para dispositivos iOS.

Para llevar acabo la solución se resolvió mediante el patrón de diseño MVC (Modelo-Vista-Controlador).

Se hicieron uso de patrones de diseño como: patrón delegado, patrón decorador, patrón singleton para poder hacer más entendible el código y escalable.

Para evitar las dependencias de terceras, no se uso ninguna, únicamente se usaron algunas extensiones adicionales a las clases principales de UIKit para dotar de mayores funcionalidades dichos componentes.

Se trato de modularizar la aplicación separando componentes que en un futuro sean adaptables a una solución personalizada, se hizo mediante la separación de vistas creadas con XIB y su controlador correspondiente.

La mayor carga de procesamiento y validaciones se dejaron del lado del servidor para hacer una aplicación óptima.

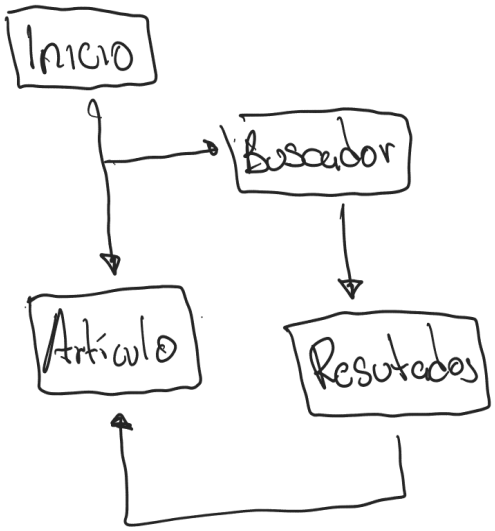
En temas de seguridad, se explicará más adelante los recursos usados para el procesamiento de pagos.

Para la persistencia de datos, se usó el cache temporal para la carga de imagenes dado que será muy frecuente por el tópico de la aplicación, y para almacenar información del usuario se dicidió usar el almacenamiento permanente en archivos por temas de facilidad y dada la complejidad de la aplicación que no requeria implementaciones complejas.

La aplicación móvil consiste en un carrito de compras con 4 secciones principales que son las siguientes:

Sección Inicio:

El flujo en dicha sección es el siguiente:

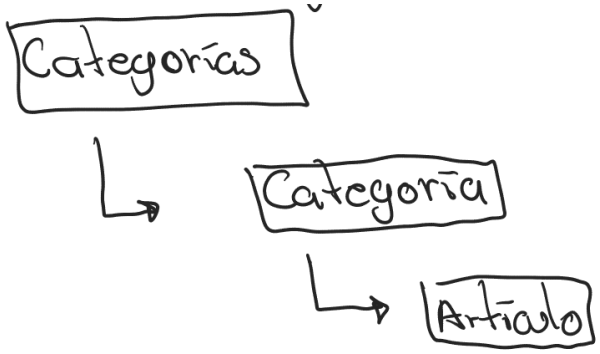


Al iniciar la aplicación, consume un servicio web para mostrar los artículos más vendidos o más populares. Con ello existen 2 flujos posteriores:

- *Buscador*
  En dicho flujo, se puede realizar la búsqueda de un artículo, para ello el texto ingresado debe ser mayor o igual a 3 caracteres para que se realice una petición al servicio web de búsqueda y devuelva solamente 3 resultados.
  Con ello se genera un nuevo flujo:
  - *Resultados de búsqueda*
    En los resultados de búsqueda, se consume el servicio para obtener todos los resultados de la búsqued y mostrarlos.
    Da ahí, el flujo final puede ser de nuevo un artículo individual.
- *Artículo individual*
  Se presenta toda la información de un artículo individual con la posibilidad de agregarlo al carrito de compras.

Sección Categorías:

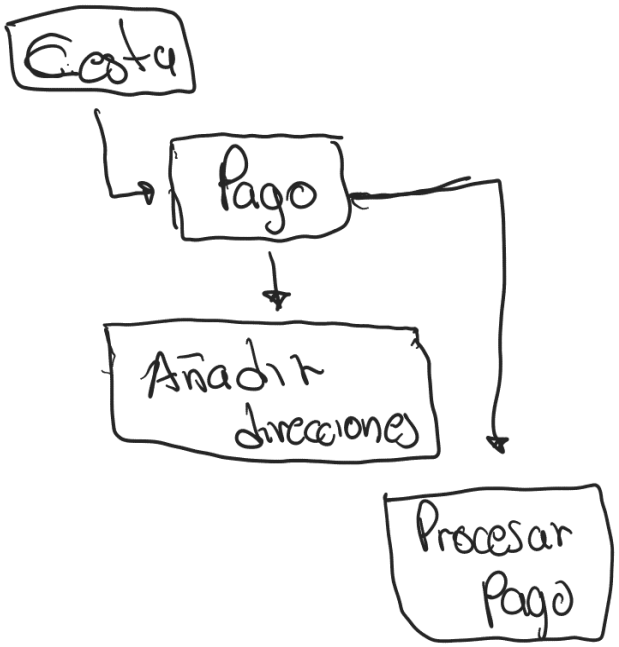
En está categoría existe unicamente un flujo lineal que es el siguiente:



Como se muestra en la anterior imagen, existe un único flujo, en la primer pantalla se muestran toda las categorías de la aplicación, al seleccionar una de ellas se consume un recurso y se muestra una nueva pantalla con todos los articulos que se obtuvieron del servicio.

Por último, en esta instancia se tiene la opción de seleccionar un arículo que nos va a permitir ver la pantalla individual con su información, una vez que la petición a nuestor servicio web nos devuelva el contenido del artículo.
De igual manera, las acciones que podemos ejercer en dicha pantalla es el de agregar el artículo a nuestra cesta de compras y posteriormente nos regresara a la pantalla de los artículos de la categoría selccionada.

Sección Cesta de compras:
Para esta sección, se realizó un flujo casi lineal dado que solo era necesario seguir el proceso de compra validando y asegurando que el usuario en verdad quiere realizar la operación. El flujo en la aplicación es el siguiente:



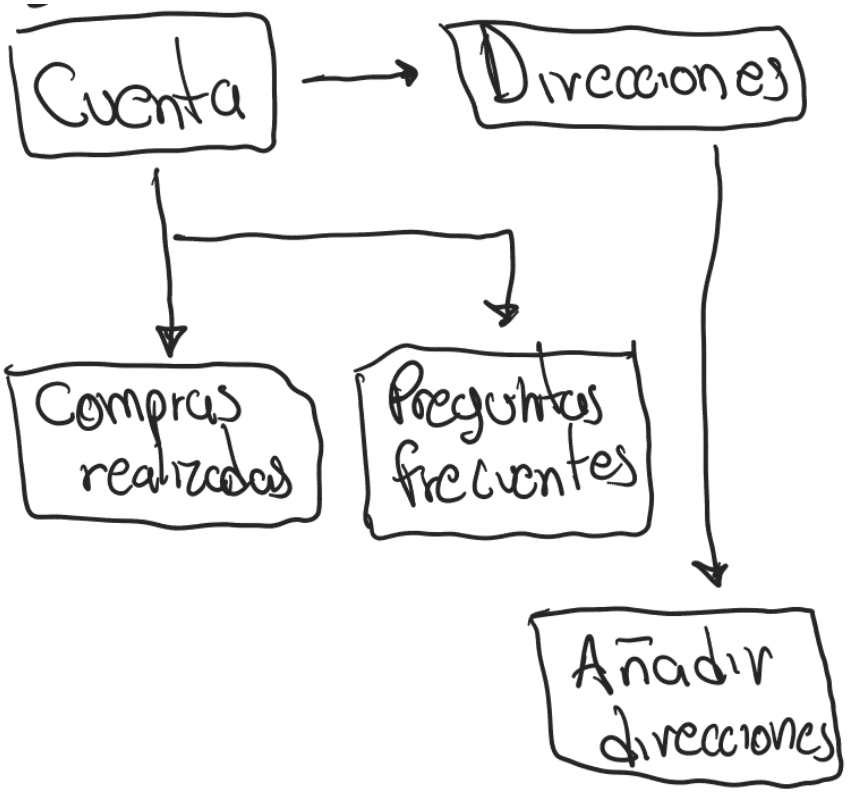
- *Cesta*
  El usuario observará en la vista todos los artículos que ha agregado a su cesta, con la posibilidad de eliminar o editar la cantidad a comprar de cada artículo. Podrá observar el total que lleva en ese momento sin incluir el costo de envío.
  - *Pago*
    Una vez confirmado todo en la cesta, el siguiente flujo es el del pago, donde es necesario llenar un formulario con los datos de dirección de entrega y de la tarjeta de crédito para realizar el cargo.
    Una vez todo procesado correctamente, la aplicación consume un recurso para obtener una llave simetrica para interactuar de forma segura con el servicio web, con la llave se cifra la información y se envía al servicio web. En ese instante, para detener cualquier acción en la aplicación, se presenta la vista de procesar pago.
    - *Añadir direcciones*
      En el proceso de llenar el formulario para la compra, el usuario en caso de no tener una dirección de entrega registrada aún, puede

acceder a esta vistsa mediante un botón para poder así ver y editar sus direcciones.

- *Procesar Pago*
  Es una vista de espera en lo que el servicio web valida los datos. Si todo es correcto, se mostrará una alerta dependiente de si fue exitoso o no el proceso. Una vez concluido el proceso, se regresa a la pantalla principal de esta sección.

Sección Cuenta:

Es la sección con mayor bifurcaciones pero es la más sencilla y en todos sus casos casi es solo de consulta.



- *Cuenta*
  - *Direcciones*
    En dicha pantalla, se podrá observar las direcciones almacenadas en nuestro dispostivo para poder usarlas en el proceso de compra. Es posible realizar un proceso de edición o añadir una nueva dirección siguiente el flujo a la pantalla siguiente.
    - *Añadir direcciones*
      Aquí es donde se podrá editar una dirección existe o bien agregar una nueva.
  - *Compras realizadas*
    Vista de consulta con las operaciones realizadas y el estatus en el que se encuentran para saber cuando vamos a recibir nuestro artículo.
  - *Preguntas frecuentes*

Seccion de consulta para añadir información adicional de nuestro negocio o logistica.

De forma general, la aplicación móvil siempre esta consumiendo nuestra API para obtener la información de los articulos así como para el proceso de los pagos.
Dado que es una solución generica, se decidio no incluir un registro dado que no siempre es necesario, lo que se realizó para cubrir de algún modo ese vacío y saber quien es el responsable de cada operación sin que el usuario tenga que registrarse, se asocio el guid del dispostivo como si fuera una cuenta para poder asociar los datos siempre.
El proceso de pago a más detalle, se describe con el siguiente diagrama.

1. La aplicación realiza una llamada al servicio web, envía como dato el guid del dispositivo.
2. El servicio web, genera la concatenación de una llave simetríca mediante caracteres aleatorios concatenados con el guid del dispostivo. Con lo anterior, genera un hash con la función sha256 y almacena dicho hash para asociar siempre la llave simetrica con el dispostivo. Dicha llave de 16 bytes, la devuelve al dispositivo,
3. El dispostivo una vez que recibe la información, cifra los datos de pago de toda la transacción mediante el algoritmo AES256 usando la llave simetrica y envía lo anterior al servicio para que procese el pago.

4. El servicio web recibe la información y se encarga de realizar las validaciones necesarias, una vez realizado lo anterior, envía todos los datos de la transacción (sin información sensible) para que la apliación la procese, o bien, envía el mensaje de error que se haya generado.
Recordar que durante todo el proceso, existe una comunicación mediante el protocolo de seguridad SSL.

Backend.
Para llevar acabo el desarrollo de la solución se ocuparon las siguientes tecnologías:

- PHP 7.1: Framework Laravel
    - SDK de Openpay para procesar los pagos.
- Base de datos: PostgreSQL
- Protocolo de seguridad: SSL

Al igual que en la aplicación, laravel es un framework que toma como base el patrón de MVC, sin embargo, fue necesario implementar 2 capas más para poder realizar una API.
Para la integración de pagos con tarjetas de crédito, se uso el SDK de openpay que nos ofrece un sandbox de desarrollo para realizar distintas pruebas simulando cargos, una vez simulado cada escenario, se puede desplegar tu aplicación y se comportará tal y como en el sandbox. Es decir, no es necesario el uso de tarjetas reales durante las pruebas, las tarjetas válidas para cada escenario son las siguientes:

Tarjetas para obtener un resultado exitoso:

Numero de tarjeta	Marca	Banco emisor
4111111111111111	Visa	BANAMEX
4242424242424242	Visa	BANCOMER
5555555555554444	MasterCard	BANCO SANTANDER SERFIN (Acepta pago con puntos)
5105105105105100	MasterCard	SCOTIABANK (Acepta pago con puntos)

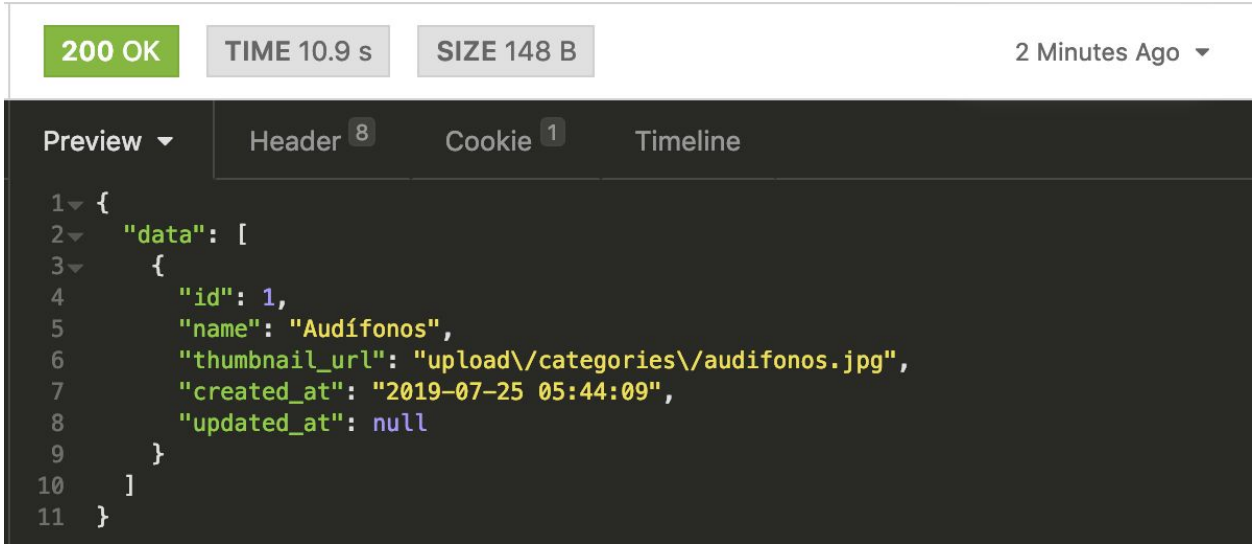
Tarjetas para simular distintas operaciones sin exito:

Número de tarjeta	Error	Descripción
4222222222222220	3001	La tarjeta fue rechazada.
4000000000000069	3002	La tarjeta ha expirado.
4444444444444448	3003	La tarjeta no tiene fondos suficientes.
4000000000000119	3004	La tarjeta ha sido identificada como una tarjeta robada.
4000000000000044	3005	La tarjeta ha sido rechazada por el sistema antifraudes.

Para mayor información se puede consultar la documentación de openpay:
https://www.openpay.mx/docs/testing.html

Endpoint
Se incluyen los servicios principales y que permiten la comunicación con la aplicación.
El uso de cada enpoint es el siguiente:
Base API: https://shop-app-ios.herokuapp.com/api/

GET /category
Obtiene todas las categorías existentes.



GET /product
Devuelve los artículos más vendidos.

200 OK | TIME 165 ms | SIZE 3 KB | 8 Days Ago

Preview | Header 9 | Cookie 2 | Timeline


```
{
  "data": [
    {
      "id": 2,
      "name": "Huawei Honor FlyPods Lite Negro",
      "price": "2490.00",
      "stock": 100,
      "thumbnail_url": "upload\/products\/2\/thumbnail.jpg",
      "active": 1,
      "tags": "huawei, honor, flypods, audífonos",
      "category_id": 1,
      "created_at": "2019-07-23 04:37:03",
      "updated_at": null,
      "description": "1. Compatible con el protocolo de transmisión de audio HWA HD Bluetooth, también compatible con muchos otros protocolos de transmisión Bluetooth, la calidad del sonido es más pura y translúcida. El diseño dinámico del altavoz de 13 mm crea una experiencia auditiva más entretenida. \n2. El único auricular con dos micrófonos MEMS, que coinciden con la tecnología de reducción
```


GET /product/category/idCategory

Devuelve todos los artículos que pertenecen a la categoría especificada.

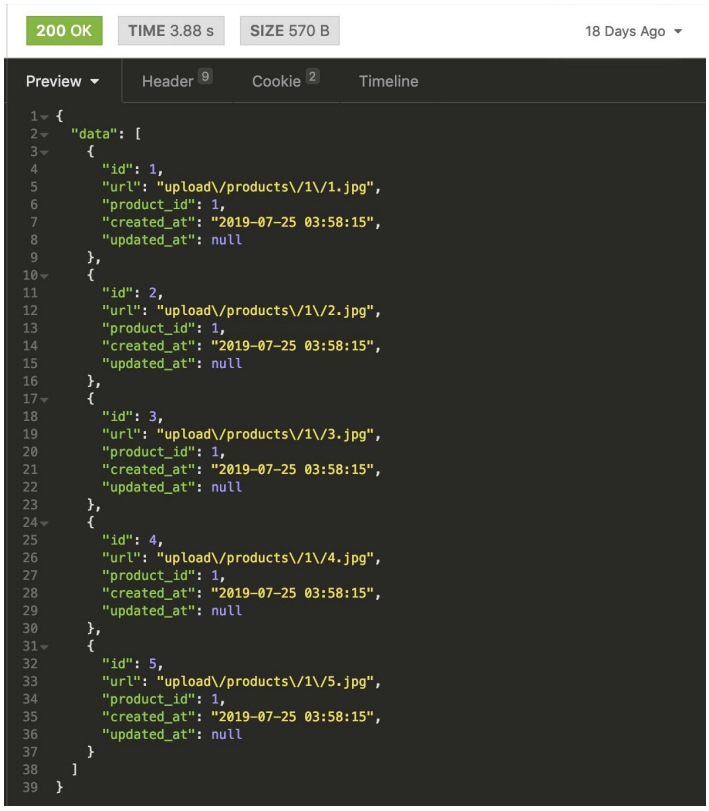
200 OK | TIME 143 ms | SIZE 3 KB | 18 Days Ago

Preview | Header 9 | Cookie 2 | Timeline


```
{
  "data": [
    {
      "id": 1,
      "name": "Xiaomi Redmi AirDots Negro",
      "price": "750.00",
      "stock": 100,
      "thumbnail_url": "upload\/products\/1\/thumbnail.jpg",
      "active": 1,
      "tags": "xiaomi, redmi, airdots, audífonos",
      "category_id": 1,
      "created_at": "2019-07-23 04:29:56",
      "updated_at": null,
      "description": "Conexión inalámbrica: Bluetooth 5.0\nProtocolo Bluetooth: HFP \/ A2DP \/ HSP \/ AVRCP\nTipo de batería: batería de polímero de ión litio\nCapacidad de la batería de un solo auricular: 40mAh\nAuriculares duales tiempo de reproducción continua: alrededor de 4 horas.\nTiempo de carga de los auriculares: aproximadamente 1,5 horas\nTiempo de carga de la caja de carga:
```


GET /product/gallery/idProduct

Obtiene todas las imagenes del producto, galería del producto.

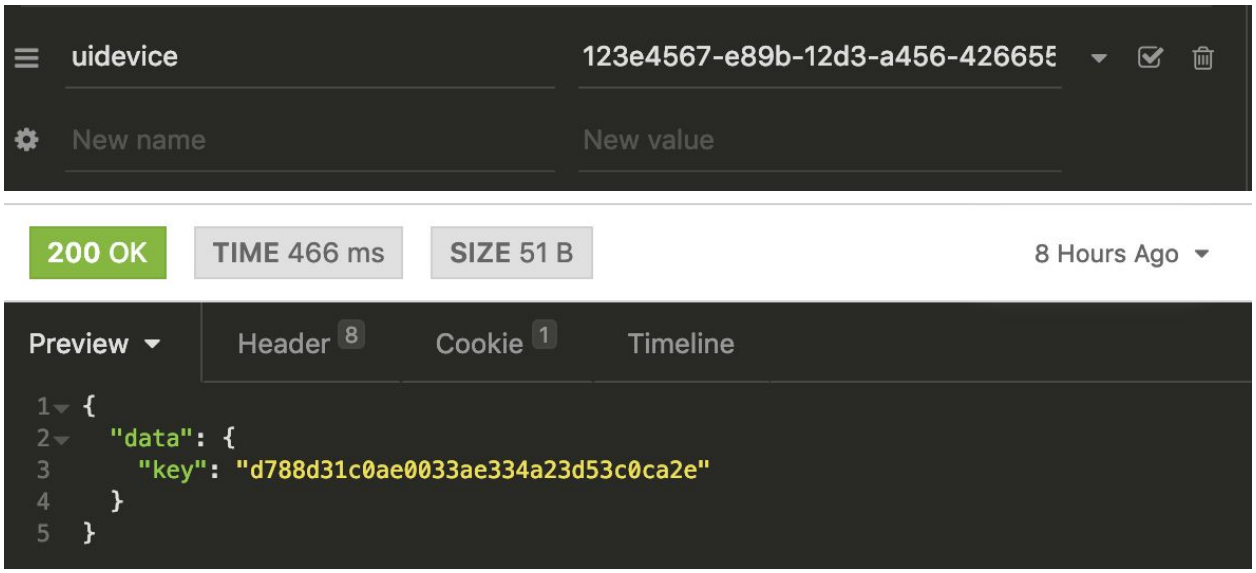


GET /product/search
Servicio para poder realizar búsquedas.
Parámetros GET:
name: Nombre del articulo a buscar.
idCategory: Id de la categoría si se quiere realizar solamente ahí la búsqueda.
results: Número de resultados a mostrar.

GET /promotions
Servicio encargado de devolver toda la información relacionado con una oferta, en la aplicación permite formar un slider.
En caso de que la oferta tenga un producto asociado, incluye al articulo tambien.

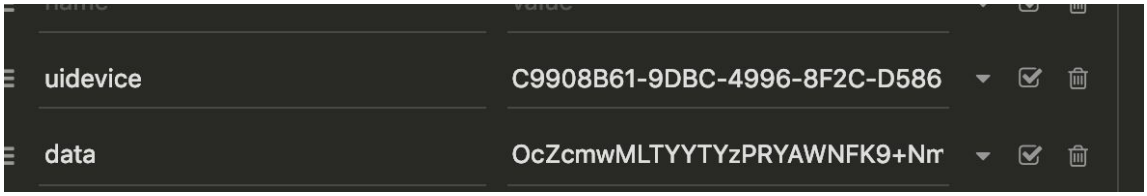
```
{
  "data": [
    {
      "updated_at": "2019-08-09 04:20:13",
      "thumbnail_url": "upload\/offers\/2.jpg",
      "id": 2,
      "created_at": "2019-08-09 04:20:13",
      "product_id": null,
      "product": null
    },
    {
      "updated_at": "2019-08-09 04:20:13",
      "thumbnail_url": "upload\/offers\/3.jpg",
      "id": 3,
      "created_at": "2019-08-09 04:20:13",
      "product_id": null,
      "product": null
    },
    {
      "updated_at": "2019-08-09 04:20:13",
      "thumbnail_url": "upload\/offers\/1.jpg",
      "id": 1,
      "created_at": "2019-08-09 04:20:13",
      "product_id": null,
      "product": null
    }
  ]
}
```

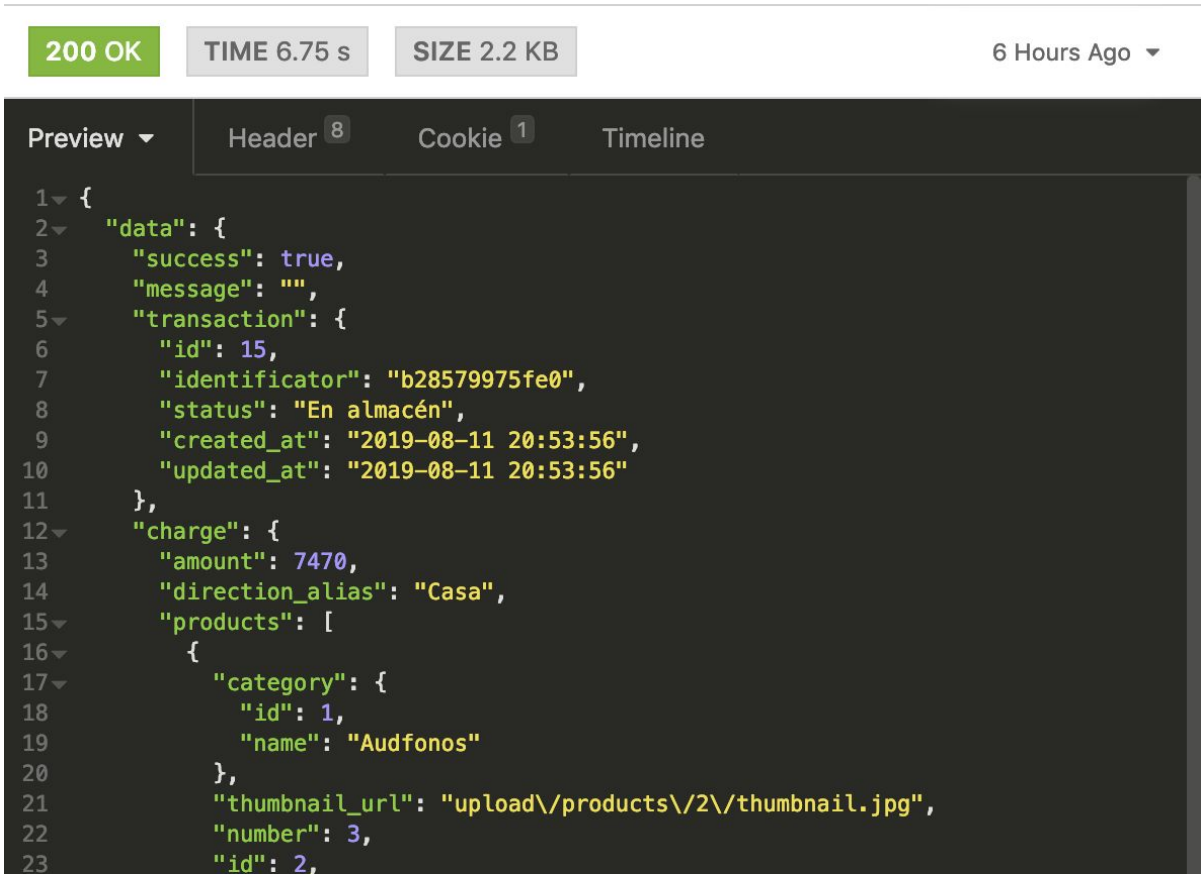

GET /key
Servicio que genera y asocia una llave simetrica para la comunicación con un dispositivo, devuelve dicha llave.
Parámetros:



El siguiente servicio es el encargado de realizar la operación de pago, dado que hay más allá de él y los pasos intermediarios de comunicación, se presenta un servicio simplificado analogo de lo que se realizada en el proceso.

POST /charge
Servicio encargado de realizar el proceso de pago.
Datos envíados





Por temas del curso, la implementación de cada servicio no fue realizada a detalle dado que no compete el backend (PHP).
La comunicación realizada en la aplicación y el backend es usando como tipo de contenido JSON, usando uso de la facilidad de Codable en Swift para poder recibir y enviar información en este formado, así como el tratamiento posterior de la información.
Dada la implementación realizada en PHP, se diseño una clase para consumir la API mediante genericos.