BANCO CENTRAL DE RESERVA

Firma Electrónica Automatizada.

GERENCIA DE INNOVACIÓN Y TECNOLOGÍA DE INFORMACIÓN

BANCO CENTRAL DE RESERVA DE EL SALVADOR

2025

Contenido

I. Introducción 1

1.1 Objetivo General 1

1.2 Objetivo específico 1

1.3 Alcance 1

II. SignBox API, primeros pasos 2

2.1. ¿Qué es esta API? 2

2.2. Audiencia dirigida 2

2.3. URL Base de la API 3

2.5 Requisitos Previos y Restricciones 3

III. SignBox API,Configuración 5

3.1. Configuración de Herramientas 5

3.1.1. Instala Docker y sus componentes 5

3.1.2. Inicia el servicio Docker 5

3.1.3. Instalación de SignBox Optimizer 6

IV. Guía de Uso Rápido / Ejemplos Típicos de Errores 9

4.1 Errores recibidos al consumir /api/sign 9

4.2. Logs Enviados a URL webhook configurado en endpoint /api/sign. 10

4.3. El documento no se ha cargado correctamente 10

4.4. Tu usuario de firma electrónica se encuentra bloqueado para firmar 10

4.5. Se sobrepaso tasa de peticiones máximas a procesar. 10

4.6. Credenciales billing ingresadas invalidas 11

4.7. Se excedió el tiempo máximo de espera 11

V. Documentación Signbox API 12

5.1. URL Documentación: 12

5.2. Referencia de Endpoints 12

VI. Manejo de Errores 17

6.1. Códigos de Error Comunes de la API 18

VII. Conceptos Clave y Glosario 19

VIII. Referencias: 20

I. Introducción

La solución de alto rendimiento para firmas masivas, interactivas y de larga duración (LTV). Nuestra mejor opción para firmar un gran número de documentos y transacciones.

SignBox API es una solución empresarial para la firma electrónica automatizada de cualquier tipo de archivo o documento. Un sistema integral que se integra fácilmente con cualquier aplicación empresarial sin afectar la experiencia del usuario y garantizando la seguridad jurídica de sus documentos firmados electrónicamente. En el presente documento técnico se especifica aspectos para la integración del servicio SignBox, manejo de errores y descripción sobre los conocimientos necesarios para que desarrolladores y arquitectos de sistemas puedan integrar exitosamente la solución de firma electrónica de alto rendimiento SignBox API en sus aplicaciones.

1.1 Objetivo General

Proveer una guía técnica detallada y los conocimientos necesarios para que desarrolladores y arquitectos de sistemas puedan integrar exitosamente la solución de firma electrónica de alto rendimiento SignBox API en sus aplicaciones, asegurando la correcta comunicación, el manejo de la carga de trabajo y la garantía de la seguridad de los documentos firmados.

1.2 Objetivo específico

El objetivo específico de esta documentación es permitir que los equipos técnicos de las instituciones de gobierno y clientes del Banco Central de Reserva puedan integrar el servicio de firma electrónica de manera segura, eficiente, robusta y conforme a las normativas aplicables, minimizando los tiempos de desarrollo y depuración.

1.3 Alcance

Este documento técnico abarca los siguientes aspectos clave para la integración del servicio SignBox:

- Descripción de los componentes principales de la solución: SignBox API, SignBox Optimizer, y su interacción.
- Detalles sobre la API RESTful expuesta por SignBox Optimizer, incluyendo endpoints, métodos HTTP, formatos de solicitud y respuesta.
- Aspectos relacionados con el rendimiento y la optimización de la carga de trabajo de firma masiva.
- Requisitos de infraestructura y red (incluyendo consideraciones de firewalls y comunicación bidireccional) para el despliegue y la operación del SignBox Optimizer y la aplicación cliente.

II. SignBox API, primeros pasos

2.1. ¿Qué es esta API?

La solución de alto rendimiento para firmas masivas, interactivas y de larga duración (LTV). Nuestra mejor opción para firmar un gran número de documentos y transacciones.

SignBox API es una solución empresarial para la firma electrónica automatizada de cualquier tipo de archivo o documento. Un sistema que se integra fácilmente con cualquier aplicación empresarial sin afectar la experiencia del usuario y garantizando la seguridad jurídica de sus documentos firmados electrónicamente. SignBox API es una API web de alto nivel basada en el paradigma RESTful http.

La API se proporciona con SignBox Optimizer, que es un servicio que expone una API RESTful http mediante la cual se permite que las aplicaciones comerciales requieran la firma electrónica de una gran cantidad de documentos.

SignBox Optimizer gestiona la carga de trabajo computacional más costosa del proceso de firma, reduciendo así el tráfico de datos en la red local y aprovechando al máximo la aceleración criptográfica del hardware. Los documentos por firmar se procesan en la capa de negocio del cliente y no se envían a terceros, sino que se envía un hash del documento creado mediante un algoritmo hash. Para entornos que exigen un alto rendimiento, SignBox puede combinarse con un conjunto de SignBox Optimizer.

El sistema ofrece opciones para diversos formatos de firma electrónica, incluyendo sellado de tiempo y validación a largo plazo. Las firmas electrónicas se ejecutan en el Centro de Servicios de Confianza de Namirial, donde las claves de firma se almacenan en un dispositivo cualificado de creación de firmas electrónicas (QSCD).

2.2. Audiencia dirigida

Esta documentación técnica está diseñada y orientada a los siguientes roles profesionales dentro de las instituciones de gobierno y sus equipos técnicos:

Desarrolladores de Software:

Programadores y equipos de desarrollo responsables de escribir el código que interactuará directamente con los endpoints de la API.

Necesidades: Requerirán una comprensión clara de los métodos HTTP, URIs, parámetros de solicitud y respuesta (tipos de datos, formatos), ejemplos de código, y cómo manejar los diferentes escenarios de respuesta.

Administradores de TI / Operaciones:

Personal responsable del despliegue, configuración, monitoreo y mantenimiento continuo de los sistemas que utilizarán la API.

Necesidades: Les interesará la información sobre requisitos de red, gestión de credenciales, límites de tasa, monitoreo de logs, y procedimientos de soporte.

Se asume que la audiencia posee un conocimiento técnico fundamental en:

- Principios y conceptos de las APIs RESTful (métodos HTTP, códigos de estado, idempotencia).
- Formato de intercambio de datos JSON.
- Conceptos básicos de seguridad en APIs (HTTPS/TLS).
- Principios y conceptos de Webhooks y su aplicación.
- Administración básica de sistemas operativos (Linux), incluyendo gestión de servicios, permisos de archivos y configuración de red.
- Conceptos de virtualización y contenedores (como Docker).
- Configuración y monitoreo básico de servicios de red en el servidor (ej. puertos, firewalls).

2.3. URL Base de la API

https://{dominio-firmador}/api

2.4. Autenticación y Autorización

Cómo Obtener y Gestionar Credenciales:

Para obtener las credenciales necesarias para la firma electrónica de documentos, tanto para entornos de prueba como de producción, incluyendo las credenciales de usuario y de billing, los interesados deben ponerse en contacto directo con el Banco Central de Reserva (BCR) de El Salvador a través del correo electrónico psc.bcr@bcr.gob.sv.

El BCR es la entidad encargada de gestionar y proveer el acceso a los sistemas y servicios de firma, y ellos indicarán el proceso específico, los requisitos y los canales de comunicación adecuados para solicitar y recibir dichas credenciales. La gestión y custodia segura de estas credenciales una vez obtenidas es responsabilidad del solicitante.

2.5 Requisitos Previos y Restricciones

2.5.1. Consideraciones de red (firewalls):

Es fundamental asegurar que la comunicación entre el servicio donde se encuentre alojada la aplicación que realiza la firma y el servicio donde esté levantado el API SignBox se garantice de manera bidireccional. Esto cobra especial relevancia en infraestructuras alojadas en la nube (cloud), particularmente si estas se encuentran fuera del país y tienen implementadas restricciones de acceso basadas en la geolocalización de las IPs de origen.

Para permitir esta comunicación, es necesario identificar las direcciones IP públicas utilizadas por ambos servicios (por ejemplo, IPs Elásticas en el caso de AWS u otras IPs estáticas/públicas asignadas). Estas IPs deben ser explícitamente permitidas en los firewalls, Grupos de Seguridad, o Web Application Firewalls (WAF) que protejan a ambos servicios. Se debe configurar reglas que permitan el tráfico en

los puertos requeridos en ambas direcciones (entrada y salida) entre las IPs autorizadas, asegurando así la fluidez necesaria para la operación de firma.

2.5.2. Requisitos de hardware

CPU: multinúcleo moderno (mínimo 4 núcleos)

RAM: 8 GB

Disco duro: 200 GB

2.5.3. Optimizador de SignBox en Docker

Esta configuración requiere un servidor con un sistema operativo Linux.

Nota Importante sobre la Compatibilidad de Comandos Linux:

Los comandos presentados en esta documentación están optimizados y son directamente compatibles con distribuciones de Linux basadas en Debian y Ubuntu, así como sus derivados más populares. Esto incluye sistemas operativos como:

- Debian (todas las versiones estables)
- Ubuntu (todas las versiones LTS y regulares)
- Linux Mint
- Zorin OS
- Y otras distribuciones que utilizan el sistema de paquetes .deb y el gestor de paquetes apt (o apt-get).

Diferencias que tomar en cuenta si se desea usar una distribución diferente a las mencionadas anteriormente:

Gestores de paquetes: Los comandos para instalar, actualizar o eliminar software varían. Por ejemplo, se utiliza apt en distribuciones Debian/Ubuntu, mientras que dnf o yum se usan en CentOS/Fedora, o pacman en Arch Linux.

Rutas de configuración: Algunos archivos de configuración de servicios o aplicaciones pueden residir en diferentes ubicaciones entre familias de distribuciones.

Herramientas específicas de distribución: Algunas herramientas o scripts pueden ser exclusivos de una familia de distribución particular.

III. SignBox API,Configuración

3.1. Configuración de Herramientas

3.1.1. Instala Docker y sus componentes

- **Actualiza e instala utilidades necesarias**

sudo apt update

sudo apt install -y ca-certificates curl gnupg lsb-release

- **Crea el directorio para almacenar la clave GPG**

sudo mkdir -p /etc/apt/keyrings

- **Descarga la clave GPG oficial de Docker**

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \

sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

- **Actualiza APT e instala Docker y componentes**

sudo apt update

sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

3.1.2. Inicia el servicio Docker

*** Inicia Docker**

sudo systemctl start Docker

- **(Opcional) Habilita Docker al iniciar el sistema**

sudo systemctl enable Docker

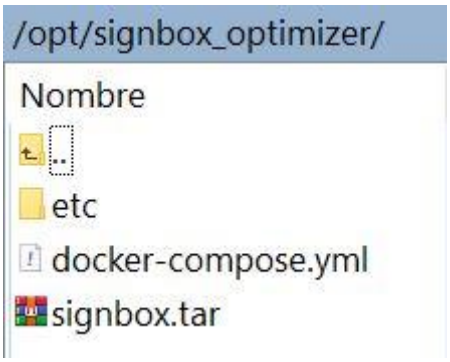
- **Verificar que todo se instaló correctamente corriendo la imagen hello-world de docker.**

sudo docker run hello-world

3.1.3. Instalación de SignBox Optimizer

PASO 1: Extrae y copia el contenido zip de SignBox Optimizer al servidor.

- Extrae todo ***signbox_optimizer_docker.zip*** el contenido en una carpeta local.
- Mueve la carpeta SignBox Optimizer a la ruta /opt en el servidor.
- El resultado debería verse así:



PASO 2: Mapeo de volúmenes (para entornos con un grupo de SignBox Optimizer).

En entornos de alto rendimiento con un grupo de SignBox Optimizer, la configuración y los registros del servicio deben almacenarse en un volumen compartido fuera de los servidores de Optimizer. Estos volúmenes deben definirse en docker-compose.yml un archivo de cada SignBox Optimizer.

cd /opt/signbox_optimizer

Archivo de configuración Docker-compose.yml:

```yaml
version: "3"

services:

    api:
        image: signbox
        command: /opt/bit4id/de/bin/de.exe --console api 0
        volumes:
            - ./etc/settings.ini:/opt/bit4id/de/etc/settings.ini
            - db_storage:/opt/bit4id/de/var/
        environment:
            CRYPTOSVC_URL: http://cryptosvc:4096
            DOCKER_RUNNING: 1
        ports:
            - "8080"
    cryptosvc:
        image: signbox
        command: /opt/bit4id/de/bin/de.exe --console cryptosvc 0
        environment:
            CRYPTOSVC_INTERFACE: cryptosvc
            DOCKER_RUNNING: 1
        volumes:
            - ./etc/tsa/:/opt/bit4id/de/etc/tsa/
            - ./etc/trusted_roots/certs/:/opt/bit4id/de/etc/trusted_roots/certs/
            - ./etc/img/:/opt/bit4id/de/etc/img/
            - ./etc/settings.ini:/opt/bit4id/de/etc/settings.ini
            - db_storage:/opt/bit4id/de/var/
        ports:
            - "4096"
    nginx:
        container_name: nginx
        depends_on:
          - api
          - cryptosvc
        image: nginx
        volumes:
            - ./etc/nginx/de.conf:/etc/nginx/conf.d/de.conf
        ports:
            - 80:80

volumes:
    db_storage:
```

Ilustración 1 Archivo de configuración Docker-compose.yml:

PASO 3: Carga la imagen Docker de SignBox.

Ejecuta los siguientes comandos:

cd /opt/signbox_optimizer

docker image load -i signbox.tar

Elimina el archivo de imagen:

rm -rf /opt/signbox_optimizer/signbox.tar

PASO 4: Iniciar el servicio.

- **Ejecuta los siguientes comandos:**

docker-compose up -d --scale api=4 --scale cryptosvc=8

- Comprueba el estado del servicio:

  Utiliza el comando: **docker-compose ps**, lo cual deberá generar la siguiente salida.

```
docker-compose ps

            Name                           Command               State                Ports
-------------------------------------------------------------------------------------------------------------------
signbox_optimizer_nginx_1              /docker-entrypoint.sh ngin ...   Up      0.0.0.0:443->443/tcp,:::443->443/tcp,
                                                                                 0.0.0.0:80->80/tcp,:::80->80/tcp
signbox_optimizer_signbox_api_1        signbox start                    Up
signbox_optimizer_signbox_cryptosvc_1  signbox start                    Up
```

Ilustración 2 Ejemplo de comprobación de estado del servicio

PASO 5. Actualiza Signbox

cd /opt/signbox_optimizer

docker-compose down

- Enumera todas las imágenes de Docker para ello utilizando el comando **docker images** puede listar todas las imagenes de docker para encontrar el ID o el nombre de la imagen.

```
docker images

REPOSITORY          TAG                 IMAGE ID       CREATED         SIZE
nginx               latest              66f8bdd3810c   7 weeks ago     192MB
signbox             latest              9977e730eb8f   2 months ago    56.2MB
```

- Luego elimine la imagen utilizando los siguientes comandos:

docker rmi nginx:latest

docker rmi signbox:latest

- Cargue la nueva imagen Docker de Signbox.

docker image load -i signbox.tar

docker-compose up –d

IV. Guía de Uso Rápido / Ejemplos Típicos de Errores

4.1 Errores recibidos al consumir /api/sign

- ***Usuario o contraseña invalida***

```
{
    "data": "error:Error('401', 'Invalid credentials',
    '{\"result\": null, \"error\": {\"msg\": \"Invalid
credentials\",
    \"code\":    401, \"error_code\": 1, \"details\": \"\"}}')"
}
```

4.2. Logs Enviados a URL webhook configurado en endpoint /api/sign.

- **URL de documento PDF no es válida**

```
{
            "message": "PDFError",
            "exception": "PDFError",
            "id": "77.122698799670826"
}
```

4.3. El documento no se ha cargado correctamente

```
{
            "message": “NoMatch",
            "exception": "NoMatch",
            "id": "77.122698799670826"
}
```

4.4. Tu usuario de firma electrónica se encuentra bloqueado para firmar

```
{
            "message": "Token locked",
            "exception": "Token locked",
            "id": "77.122698799670826"
}
```

4.5. Se sobrepaso tasa de peticiones máximas a procesar.

```
{
            "message": "Exception: result ",
            "exception": "Exception ",
```

```
            "id": "77.122698799670826"
}
```


4.6. Credenciales billing ingresadas invalidas


```
{
            "message": "Exception: Remote Signature Failed: 401
Unauthorized",
            "exception": "Exception",
            "id": "77.122698799670826"
}
```


4.7. Se excedió el tiempo máximo de espera

(usualmente generado por el desbordamiento de las peticiones al exceder la tasa máxima configurada)


```
{
    "message": "Exception: Remote Signature Failed: User timeout
caused connection failure.",
            "exception": "Exception",
            "id": "77.122698799670826"
}
```

V. Documentación Signbox API

5.1. URL Documentación:

5.1.1 URL Documentación API-Signbox:

https://cdn.developers.uanataca.com/signbox/signbox.html#/default/sign_api_v1_sign_post

5.1.2. URL Documentación Técnica:

https://namirial.atlassian.net/wiki/spaces/BES/pages/245694518/UANATACA+SIGNBOX+API+LEGACY

5.2. Referencia de Endpoints

5.2.1. Endpoint de diagnóstico para servicio de firma:

`[GET] /api/echo?message=Hola Mundo`

- **Descripción:** Endpoint de diagnóstico diseñado para confirmar que el servicio SignBox se encuentra activo y respondiendo adecuadamente en la infraestructura de servidor configurada

- **Autenticación Requerida:** No

- **Parámetros de Consulta:**

Atributo	Tipo	Descripción
message	query params	(string), mensaje de prueba a imprimir.

- **Respuestas Posibles:**

  `Http code: 200 / OK (Éxito)`

5.2.2. Endpoint para firmar documento:

`[POST] /api/sign`

- **Descripción:**

  La API de SignBox requiere que la empresa cliente desarrolle un webhook para gestionar las devoluciones de llamada del servicio. Hay dos devoluciones de llamada en el servicio que deben configurarse en los parámetros url_outy urlbacken la llamada a la API de SIGN .
- **Autenticación Requerida:** Si
- **Parámetros de Consulta:**

Atributo	Tipo	Descripción
file_in	body	Ruta local del archivo a firmar.
url_in	body	Url donde obtener el documento a firmar.
url_out	body	URL donde se envía el archivo firmado.
urlback	body	URL donde se envía el registro del proceso de firma.
env	body	Entorno de referencia para sandbox o producción(sandbox o prod respectivamente)
format	body	Tipos de perfiles de firma admitidos: pades para archivos PDF xades para archivos XML cades para otros archivos
username	body	Nombre de usuario de identidad digital
password	body	Contraseña de identidad digital
pin	body	PIN de identidad digital
level	body	Especifique el nivel de la firma. Debe tenerse en cuenta que LTVLITE es una firma LTV más ligera y eficiente en comparación con el nivel LTV estándar. Los valores permitidos son: BES, T, LTV, LTVLITE for pades BES, T, EPES, XL for xades BES, T, EPES for cades

identifier	body	Ubicación del token.
tsa_bookmark	body	Nombre del alias del servicio de marca de tiempo. Este campo es obligatorio para firmas con niveles T, LTV o XL.
billing_username	body	Nombre de usuario de la cuenta de facturación
billing_password	body	Contraseña de la cuenta de facturación
reason	body	Razón sobre la firma
location	body	Ubicación geográfica de la firma
npage	body	Número de página donde se coloca la firma de la imagen. La primera página empieza en 0.
otp	body	El código OTP enviado previamente al usuario. Este parámetro solo es obligatorio y está disponible para identidades digitales específicas.
metadata	body	Información de metadatos opcional asociada al documento firmado
metadata_oid	body	Identificador OID asociado a los metadatos
metadata_enveloped	body	Establezca en 0 si los metadatos están en formato DER, o 1 (predeterminado) para la envoltura de cadena de octetos ASN1
img_bookmark	body	Nombre de alias de la imagen de firma correspondiente a una imagen previamente almacenada en el Optimizador. Es necesario crear previamente una plantilla de imagen
img	body	Cadena de imagen preconvertida en Base64
img_size	body	El tamaño de la imagen original en píxeles. (Ejemplo: 200,140) Cualquier discrepancia provocará ajustes incorrectos de la imagen o simplemente no aparecerá.
position	body	Las coordenadas de las esquinas inferior izquierda y superior derecha del rectángulo de la imagen corresponden a una resolución de 72 ppp. Para una página A4, los valores XY varían de 0 a 595 y de 0 a 842

		píxeles, respectivamente. (Ejemplo: 300, 100, 500, 150)
paragraph_format	body	Identificación del texto con los datos asociados del firmante.

- **Respuestas Posibles:**

Http code: 200 / Éxito

```
{
    "id": "String"
}
```

Http code: 200 / Logs de error(enviados a la URL webhook proporcionada en el parametro urlback)

```
{
    "message": "String",
    "exception": "String",
    "id": "String"
}
```

Notas Importantes:

Gestión de cuentas en ambiente sandbox y productivo: Para la realización de pruebas en el ambiente Sandbox, es indispensable gestionar y utilizar un conjunto de credenciales de firma electrónica y de billing específicas para dicho ambiente. Estas credenciales son completamente distintas y no interoperables con las utilizadas en el ambiente productivo. Intentar usar credenciales de producción en el entorno Sandbox, o viceversa, resultará en errores de autenticación.

Manejo de Excepciones: Es crucial considerar el comportamiento de las excepciones al interactuar con el servicio. Las excepciones relacionadas con usuario o contraseña inválidos, al consumir el endpoint **/api/sign**, se devuelven de manera síncrona e inmediata como respuesta directa a la solicitud HTTP. Por otro lado, todas las demás excepciones que puedan ocurrir durante el proceso de firma serán notificadas de forma asíncrona al endpoint proporcionado en el parámetro urlback. Asegúrese de implementar la lógica adecuada en ambos puntos para un manejo de errores robusto.
Comportamiento de Notificaciones Asíncronas (Webhook): El servicio consumirá el webhook proporcionado en el parámetro urlback para la gestión de todas las excepciones que puedan ocurrir durante el proceso de firma (distintas a las de credenciales usuario-constraseña inválido), así como para notificar el éxito de la operación. Este webhook actuará como un endpoint de logs y notificaciones asíncronas para mantener informado a su sistema sobre el resultado final del proceso de firma.

Cuando la operación de firma sea exitosa, el payload enviado al webhook contendrá únicamente el atributo id (identificador de la transacción), indicando que el proceso se completó satisfactoriamente.

Cuando ocurra una excepción (distinta a las de credenciales/usuario inválido), el payload enviado al webhook incluirá los atributos id, message, y exception para proporcionar detalles sobre el error ocurrido durante el procesamiento.

Asegúrese de implementar la lógica adecuada en su endpoint urlback para procesar ambos tipos de notificaciones (éxito y excepción) basándose en la estructura del payload recibido y el contenido de los atributos.

VI. Manejo de Errores

- **Estructura de error enviada al Webhook**


```
{
  "message": "String",
  "exception": "String",
  "id": "String"
}
```


- **Caso Especial de Error:** recibido al enviar Usuario o Contraseña invalida al consumir endpoint ***[POST] https://{dominio-firmador}/api/sign***


```
{
  "data": "error:Error('401', 'Invalid credentials',
  '{\"result\": null, \"error\": {\"msg\": \"Invalid credentials\",
  \"code\":   401, \"error_code\": 1, \"details\": \"\"}}')"
}
```

6.1. Códigos de Error Comunes de la API

Mensaje	Razón
ProcessTerminated: process ended with exit code X	Mal funcionamiento del webhook
Token not found	Certificado inexistente
Invalid credentials	Credenciales incorrectas
PasswordProtected	Entrada PDF protegida con contraseña
Error: Pin invalid	Valor de pin incorrecto
Error: Token locked	Credenciales del certificado bloqueadas
Error: Invalid Otp	Código OTP incorrecto o vencido
Exception: Invalid Environment. Supported values are: []	Valor de entorno incorrecto
Exception: Remote Signature Failed: 401 Unauthorized	Credenciales de facturación incorrectas
Exception: Invalid TSA alias	Alias de marca de tiempo incorrecto
Exception: Invalid Image alias	Alias de imagen incorrectos
Exception: Remote Signature Failed: 400 Bad Request	créditos insuficientes
Exception: Need tsa_bookmark for ltv signature	Ejecución de LTV con el sellado de tiempo deshabilitado
Exception: UnknownIssuer	Certificado de cualquier jerarquía faltante
Exception: Time Stamping Failed: 400 Bad Request	Marca de tiempo inexistente

VII. Conceptos Clave y Glosario

Webhook:
Un webhook es una comunicación ligera, basada en eventos, que envía datos automáticamente entre aplicaciones mediante HTTP. Activados por eventos específicos, los webhooks automatizan la comunicación entre interfaces de programación de aplicaciones (API).
Para configurar un webhook, el cliente proporciona una URL única a la API del servidor y especifica el evento que desea conocer. Una vez configurado el webhook, el cliente ya no necesita sondear al servidor; este enviará automáticamente la carga útil relevante a la URL del webhook del cliente cuando se produzca el evento especificado. Los webhooks se conocen a menudo como API inversas o API push , ya que asignan la responsabilidad de la comunicación al servidor, en lugar del cliente.

API REST:

Una API REST es una interfaz de programación de aplicaciones (API) que sigue los principios de diseño del estilo arquitectónico REST. REST es la abreviatura de transferencia de estado representacional y consiste en un conjunto de reglas y directrices sobre cómo crear una API web.

Firewall:
Los firewalls en la nube forman una barrera virtual alrededor de las plataformas, la infraestructura y las aplicaciones en la nube, al igual que los firewalls tradicionales forman una barrera alrededor de la red interna de una organización. Los firewalls en la nube también pueden proteger la infraestructura local.

Docker:
Docker es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y versión ejecutable.

Docker compose:

Docker Compose es una herramienta versátil que te permite definir y gestionar aplicaciones multi-contenedor de forma sencilla. Con Docker Compose, puedes describir la configuración de tu entorno de desarrollo en un archivo YAML, especificando los servicios, volúmenes y redes necesarios para tu aplicación. Luego, con un solo comando, puedes crear y ejecutar todos los contenedores definidos en tu archivo de configuración.

Usuario Billing: Credenciales asociadas con la gestión y el seguimiento del consumo y la facturación del servicio de firma.

VIII. Referencias:

https://cdn.developers.uanataca.com/signbox/signbox.html#/default/sign_api_v1_sign_post. (s.f.). *https://cdn.developers.uanataca.com*. Obtenido de https://cdn.developers.uanataca.com: https://cdn.developers.uanataca.com/signbox/signbox.html#/default/sign_api_v1_sign_post

https://namirial.atlassian.net. (s.f.). Obtenido de https://namirial.atlassian.net: https://namirial.atlassian.net/wiki/spaces/BES/pages/245694518/UANATACA+SIGNBOX+API+LEGACY#SignBox-Optimizer-on-Docker