Guía de implementación

Para realizar la implementación de este producto, se deben considerar dos aspectos. En primera instancia, se debe configurar el ambiente en el cual se va a desplegar la aplicación. Como segundo punto, la configuración y personalización del código fuente para que funcione y se ajuste a sus necesidades.

Ambiente necesario para desplegar la aplicación

Requisitos del sistema operativo
Necesita la versión de 64 bits de una de estas versiones de Ubuntu:

· Ubuntu Impish 21.10
· Ubuntu hirsuto 21.04
· Ubuntu Focal 20.04 (LTS)
· Ubuntu Bionic 18.04 (LTS)

Instalar Docker Engine
Puede usar el repositorio de Docker para instalar Docker Engine. Puede descargar el archivo .deb para su versión de Linux e instalarlo manualmente. Debe descargar un nuevo archivo cada vez que desee actualizar Docker.

1. Vaya al enlace. Elija su versión de Ubuntu, luego busque pool/stable/, elija amd64, armhf, arm64 o s390x, y descargue el archivo .deb para el Versión del Docker Engine que desea instalar.

2. Instale Docker Engine, cambiando la ruta a continuación a la ruta donde descargó el paquete de Docker.

sudo dpkg -i /ruta/al/paquete.deb

El servicio de Docker se iniciará automáticamente.
Verifique que Docker Engine esté instalado correctamente ejecutando la imagen hello-world.

sudo docker run hello-world

Este comando descarga una imagen de prueba y la ejecuta en un contenedor. Cuando se ejecuta el contenedor, se imprimirá un mensaje y se detendrá. Esto indica que Docker Engine está instalado y funcionando.

Nota sobre permisos Linux:
Se creará el grupo Docker pero no se le agrega ningún usuario. Debe usar sudo para ejecutar los comandos de Docker. Continúe con los pasos posteriores a la instalación de Linux para permitir que los usuarios sin privilegios ejecuten comandos de Docker y para otros pasos de configuración opcionales.

Docker Compose

Para correr el conjunto de contenedores Docker se requiere de Docker Compose.

Docker Compose se basa en Docker Engine, así que asegúrese de tener Docker Engine instalado antes de continuar.

Para instalar docker compose siga este tutorial

Adicionalmente se necesita instalar

1. Yarn
2. Node Version <10
3. Hasura-CLI

NginX

Se debe utilizar un servicio de "reverse proxy" (como Nginx, Caddy, Kong, Traefik, etc.) o las funciones nativas del proveedor de la nube (load balancer) para enrutar el tráfico HTTP/HTTPS hacia la aplicación web y el servicio backend (hasura).
Pueden consultar sobre la instalación de Nginx aquí.

Configuración Nginx
Considere aplicar esta configuración de Nginx en su servidor donde desea ejecutar la aplicación web, para estos ingrese al archivo '/etc/nginx/nginx.conf' y edite su contenido con la siguiente configuración:

```
server {
        listen 80;
        listen 443 ssl;
        server_name pruebas.muniorotina.go.cr;

        location / {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Credentials' 'true';
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers'
                'DNT,X-CustomHeader,Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Ca
                che-Control,Content-Type';

                proxy_pass http://localhost:3000/;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
        }
}
```

```
server {
        listen 80;
        listen 443 ssl;
        server_name graphql-pruebas.muniorotina.go.cr;

        location / {
                proxy_pass http://localhost:8080/;
                proxy_set_header X-Real-IP      $remote_addr;
                proxy_set_header X-Forwarded-For  $proxy_add_x_forwarded_for;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header X-Forward-Proto http;
                proxy_set_header X-Nginx-Proxy true;
                proxy_http_version 1.1;
        }
}
```

Certificados SSL
Se deben de configurar funciones de terminación de SSL para proteger la aplicación web y el API.

Instalacion y ejecucion de la aplicación

Una vez instaladas todas las dependencias indicadas deberá continuar con la instalación y ejecución de la aplicación.

· Clona este repositorio usando git clone https://github.com/edenia/constancias-municipales

· Moverse al directorio apropiado: cd constancias-municipales

· Crear el archivo con las variables de ambiente: cp .env.example .env

· Correr yarn para instalar dependencias.

· Correr make run esto para levantar el proyecto, el cual se encuentra en http://localhost:3000.

Configuración y personalización del código fuente

Esta aplicación web está conformada por un front-end (webapp), el cual se encarga de la comunicación entre el usuario final y el backend (hapi) que funciona como el servidor donde se realizan las acciones de lógica de negocios necesarias para obtener las constancias firmadas.

Front-end (webapp)

A nivel de front-end hay algunos aspectos visuales que se pueden configurar a su gusto para hacer que la aplicación se vea de la forma en la usted lo desea sin necesitar mucho esfuerzo. Dentro de los elementos que puede editar están:

- **Paleta de colores:** Para personalizar la paleta de colores deberá ingresar a "webapp/src/theme/palette.js" donde podrá cambiar los colores primarios y secundarios del sitio, además de agregar cuantas variantes considere necesarias.

- **Logo principal:** Para cambiar el logo principal de la aplicación deberá abrir el archivo. env (si ya lo creo). Dicho archivo se encuentra en la raíz del proyecto. Ahí debe cambiar el valor de la variable "REACT_APP_LOGO" por la url del logo que desea agregar.

- **Imagen de previsualización:** Esta imagen se muestra como parte del contenido de previsualización cuando se comparte el enlace del sitio web por alguna red social o medio de mensajería. Para editar esto debe sobreescribir la imagen "webapp/public/preview-image-constancias.png" por la que usted desee ver. Tenga presente que la nueva imagen contenga exactamente el mismo nombre indicado anteriormente.

- **Contenido de previsualización:** Este contenido se muestra cuando se comparte el enlace del sitio web por alguna red social o medio de mensajería. Para editar el título y la descripción con el contenido que usted considere más apropiado debe ir al archivo **"**webapp/public/index.html".

- **Enlace hacia el sitio web de la organización:** Este valor es configurable desde las variables de ambiente editando la variable "REACT_APP_URL_ORGANIZATION", la intención es utilizar esta url en cualquier lugar del front-end donde sea necesario dirigir al sitio web de la organización.

- **Nombre de la organización en el footer del sitio:** En el footer del sitio se muestra el nombre de la organización a la que pertenece el sitio web. Este valor es configurable desde una variable de ambiente, lo puede modificar cambiando el valor de la variable "REACT_APP_ORGANIZATION_NAME".

- **Enterprise Google recaptcha key:** Este valor también es configurable desde las variables de ambiente. Debe sustituir el valor de la variable "REACT_APP_PUBLIC_RE_CAPTCHA_KEY" por la llave pública del Google Recaptcha Enterprise asociado al dominio donde se va a hospedar el sitio web. Para generar uno puede seguir la documentación de Google **https://cloud.google.com/recaptcha-enterprise/docs**

Backend (hapi)

En el backend también hay valores que son configurables. En este caso, aún más importantes que los del front-end, puesto que son necesarios para poder generar, firmar y enviar las constancias.

- **Llaves de acceso para API de Yaipan:** Yaipan API es un servicio que brinda un método para generar constancias municipales a partir de un número de identificación. La constancia es generada con base en los datos almacenados en la base de datos de este servicio, es decir, la organización debe estar afiliada a los servicios de Yaipan para poder generar dicha constancia. En caso de querer utilizar alguna otra forma para obtener la constancia deberá realizar los cambios correspondientes en el código fuente. Comuníquese con el equipo de Yaipan si necesita más información.

  Si utiliza Yaipan entonces deberá agregar sus credenciales para el método "generador de constancias" en las variables de ambiente "HAPI_YAIPAN_ACCESS_TOKEN" el cual es una llave de acceso que le suministrará Yaipan para poder acceder al servicio y "HAPI_YAIPAN_CONSTACIAS_API_ENDPOINT" en esta variable debe agregar la dirección que le indique Yaipan para poder consumir a dicho método .

- **Límite diario de constancias emitidas:** Este valor le indica al sistema cuál es el límite diario de constancias que permite emitir para un mismo usuario. Puede editarlo con la variable "HAPI_CERTIFICATE_LIMIT".

- **Variables de ambiente necesarias para servicios de sellado electrónico del BCCR:** Para realizar el sellado electrónico del documento emitido por el API de Yaipan, se utilizan los servicios de firma del Banco Central de Costa Rica. Para esto, la organización deberá solicitarle a dicho banco los certificados digitales y credenciales de accesos correspondientes. El método que se utiliza para realizar sellado electrónico de documentos PDF es "RecibaLaSolicitudDeSelladoElectronicoPdf". Desde las variables de ambiente de hapi se deben configurar los siguientes valores: "HAPI_HASH_CREATOR_METHOD", esta variable indica el tipo de método utilizado para crear el hash asociado al documento que se enviará a sellar y este valor debe ser alguno de estos tres 'sha256', sha384, 'sha512'; "HAPI_LOCATION_SEALING_REQUEST", este valor indica el lugar desde el cual se realiza la solicitud de sellado; "HAPI_REASON_SEALING_REQUEST", en esta variable se debe colocar cual es la razón por la cual se desea sellar el documento; "HAPI_DOC_SEALING_TYPE" indica la extensión del tipo de documento que se envía para sellar, ya que se podría firmar no solo documentos en formato pdf, sino también en otros formatos como xml, odf, entre otros. Consulte esta documentación para mayor información https://pyfva.readthedocs.io/en/latest/tutorial.html.

  Para terminar debe configurar las variables "HAPI_NEGOCIO" y "HAPI_ENTIDAD". El primero es un valor que identifica el código de la identidad de marca de la entidad que solicita la firma. El segundo valor hace referencia al tipo de usuario que realiza la firma. En este caso siempre se debe asignar 1 por que la función de sellado electrónico solo es válida para entidades jurídicas.

- **Url del servicio firmador (signatureService):** Esta variable contiene el valor de la url del servidor donde corre la aplicacion, por ejemplo “http://127.0.0.1”, dicho valor lo debe agregar en el archivo .env en la variable **“**HAPI_SIGNER_URL”.

- **Credenciales del servidor para envío de correos electrónicos:** La entrega de las constancias se realiza por medio de correo electrónico, por lo que es necesario contar con un servidor de envíos de correos. Para hacer funcionar el envío de correos en la aplicación deberá editar el valor de las variables “HAPI_MAIL_HOST”, “HAPI_MAIL_PORT”, “HAPI_MAIL_USER”, “HAPI_MAIL_PASSWORD”.

- **Credenciales de enterprise Google recaptcha:** Para que el sitio web cuente con un recaptcha, deberá configurar las variables de ambiente asociadas, “HAPI_RE_CAPTCHA_PROJECT_ID”, “HAPI_PUBLIC_RE_CAPTCHA_KEY”. Además de esto, deberá agregar en el archivo “google-credentials.json” ubicado en “hapi/ google-credentials.json” las credenciales generadas en Google recaptcha enterprise.

- **Logo de la organización:** Para cambiar el logo de la organización deberá abrir el archivo .env. Ahí debe cambiar el valor de la variable “HAPI_ORGANIZATION_IMAGE” por la url del logo que desea agregar.

- **Enlace hacia el sitio web de la organización:** Este valor es configurable desde las variables de ambiente editando la variable “HAPI_URL_ORGANIZATION”. La intención es utilizar esta url en cualquier lugar del backend donde sea necesario dirigir al sitio web de la organización.

- **Enlaces a redes sociales:** Como parte de la información de contacto que se envía en el correo electrónico que contiene la constancia firmada, se agregan enlaces hacia las redes sociales de la organización. Puede agregar sus propios enlaces colocando las url en las siguientes variables de ambiente “HAPI_FACEBOOK_LINK”, “HAPI_TWITTER_LINK”, “HAPI_INSTAGRAM_LINK”, ”HAPI_YOUTUBE_LINK”.

Backend (signatureService)

Para realizar la implementación del sellado digital de documentos, se realizó la integración de la librería **pyfva** desarrollada en Python, por lo que se construyó un servicio de backend en dicho lenguaje utilizando la herramienta flask. Como parte del proceso de sellado, el sistema que realiza la solicitud debe autenticarse por medio de los certificados suministrados por el BCCR. Estos valores se manejan como variables de ambiente, las cuales se configuran en el archivo ".env" con los siguientes valores:

· **FVA_HOST:** Indica el host de los servicios del BCCR y su valor debe ser "http://bccr.fva.cr/"

· **STUB_SCHEME:** Esta variable indica el esquema de la url, su valor es "https"

· **STUB_HOST:** Indica el host del conjunto de métodos a consumir y su valor debe ser "firmadorexterno.bccr.fi.cr"

· **RECEPTOR_HOST:** Indica el host receptor de la solicitud y su valor debe ser "http://bccr.fva.cr/"

· **RECEPTOR_CLIENT:** Indica el cliente receptor de la solicitud y su valor debe ser "pyfva.receptor.client"

· **FVA_TESTURLS:** Esta variable se debe enviar en *true* si desea utilizar las urls de pruebas de los servicios del BCCR en caso de querer usar los urls de producción no debe agregar esta variable.

· **REQUESTS_CA_BUNDLE:** Este valor debe contener la ruta donde se encuentra guardado el archivo "ca_nacional_CR.pem", el cual contiene todos los certificados de la CA que autoriza el consumo de los servicios del BCCR.

· **REQUESTS_CA_PATH:** Este valor debe contener la ruta donde se encuentra guardado el archivo "ca_nacional_CR.pem" el cual contiene todos los certificados de la CA que autoriza el consumo de los servicios del BCCR. Ejemplo: '/app/Certificados/ca_nacional_CR.pem'

· **REQUESTS_CERT_PATH:** Esta variable debe contener la ruta donde se encuentra guardado el archivo "bccr_agent.pem", el cual contiene el certificado de agente electrónico de la institución que solicita el uso de los servicios. Ejemplo: '/app/Certificados/bccr_agent.pem'

· **REQUESTS_KEY_PATH:** Esta variable debe contener la ruta donde se encuentra guardado el archivo "bccr_agent_key.pem", el cual contiene la llave de acceso al certificado agente electrónico de la institución que solicita el uso de los servicios. Ejemplo: '/app/Certificados/bccr_agent_key.pem'