# Despliegue e Instalación del KAVAC para Producción.

| Versión de la aplicación | Documento | Versión del documento | Elaborado por | Fecha de Elaboración |
|---|---|---|---|---|
| kavac v-1.0.3 | Manual de despliegue e instalación para producción | 1.0.3 | Laura Colina<br>Dhionel Diaz<br>Dianicth Monsalve<br>Lully Troconis | 10-09-2023 |

KAVAC
Sistema de Gestión de Recursos

# Licencia

**Nombre del producto:** KAVAC ‘Sistema de Gestión de Recursos’

**Nombre del licenciante y año:** Licencia Combinada de Software y Contenidos de la Fundación CENDITEL Versión 1.0.3.

**Créditos:**

**Analistas**


- Julie Vera (jvera@cenditel.gob.ve)
- María Laura González (mgonzalez@cenditel.gob.ve)
- María Morales (mmorales@cenditel.gob.ve)
- María Rujano (mrujano@cenditel.gob.ve)
- Mariangel Molero (mmolero@cenditel.gob.ve)
- Francisco Berbesí (fberbesi@cenditel.gob.ve)
- Luis Ramírez (lgramirez@cenditel.gob.ve
- Hildayra Colmenares (hcolmenares@cenditel.gob.ve )

- Kleivymar Montilla (kmontilla@cenditel.gob.ve)
- Alberto Gil (argil@cenditel.gob.ve )
- Reina Castellanos (recastellanos@cenditel.gob.ve)

**Manuales de usuario y despliegue en producción**

- Luis Ramírez (lgramirez@cenditel.gob.ve)
- Marilyn Caballero (mcaballero@cenditel.gob.ve)
- Dhionel Diaz (ddiaz@cenditel.gob.ve)
- Laura Colina (lcolina@cenditel.gob.ve)
- Dianicth Monsalve (dsmonsalve@cenditel.gob.ve)
- Lully Troconis (ltroconis@cenditel.gob.ve)

**Desarrolladores**

- William Paéz (wpaez@cenditel.gob.ve)
- Henry Paredes (hparedes@cenditel.gob.ve)
- Juan Rosas (jrosas@cenditel.gob.ve)
- Yennifer Ramírez (yramirez@cenditel.gob.ve)
- Pedro Buitrago (pbuitrago@cenditel.gob.ve)
- Angelo Osorio (adosorio@cenditel.gob.ve)
- José Puentes (jpuentes@cenditel.gob.ve)
- Daniel Contreras (dcontreras@cenditel.gob.ve)

- Miguel Narváez (mnarvaez@cenditel.gob.ve)
- Francisco Escala (fescala@cenditel.gob.ve)
- Francisco Ruiz (fjpenya@cenditel.gob.ve)
- José Briceño (jbrincenyo@cenditel.gob.ve)
- Fabian Palmera (fpalmera@cenditel.gob.ve )
- Manuel Zambrano (mzambrano@cenditel.gob.ve )
- José Puentes (jpuentes@cenditel.gob.ve )
- Juan Vizcarrondo (jvizcarrondo@cenditel.gob.ve )
- Oscar J. González (ojgonzalez@cenditel.gob.ve )
- Pedro Contreras (pmcontreras@cenditel.gob.ve )
- Marco Ocanto - CNTI (sanchezmarco8882@gmail.com )
- Jonathan Alvarado - CNTI (jonathanalvarado1407@gmail.com )
- Natanael Rojo (ndrojo@cenditel.gob.ve)

**Líder de proyecto / Diseño / Desarrollo / Autor / Director de Desarrollo (2021-2023)**

- Roldan Vargas (rvargas@cenditel.gob.ve)

**Director de Desarrollo (2018-2019) / Desarrollador (2020-2022)**

- Argenis Osorio (aosorio@cenditel.gob.ve)

**Directora de Desarrollo (2020)**

- Laura Colina (lcolina@cenditel.gob.ve)

**Presidente**

- Oscar González (ogonzalez@cenditel.gob.ve)

**Colaborador**

- Santiago Roca (sroca@cenditel.gob.ve)

**PUBLICADO POR EL CENTRO NACIONAL DE DESARROLLO E INVESTIGACIÓN EN TECNOLOGÍAS LIBRES (2023 CENDITEL)**

# Despliegue e Instalación del KAVAC para Producción

KAVAC es un ERP (Enterprise Resource Planning) un sistema de información que integra los procesos administrativos, de gestión, planificación, control y seguimiento de los recursos (económicos, físicos, insumos, humano) de cualquier organización. Un sistema que incorpore además funcionalidades asociadas a la gestión de proveedores y clientes (CRM).

El sistema está diseñado de forma modular, por lo que las organizaciones podrán utilizar los módulos de gestión que así requieran, además de facilitar la escalabilidad del sistema, la tecnología y metodología de desarrollo permite agregar nuevos módulos ajustados a las necesidades de la organización.

El sistema cuenta con herramientas de seguridad de la información que garanticen la integridad y confidencialidad de los datos generados por la organización, además de integrar la firma electrónica para asegurar la autenticidad de los documentos y archivos generados por el sistema.

KAVAC es una aplicación web desarrollada con el marco de trabajo del Framework Lavarel, bajo el Lenguaje de Programación PHP 8.2, parte importante de sus funcionalidades requiere el procesamiento de tareas programadas y disparadores de evento. En el presente manual se describe el esquema mínimo recomendado para desplegar Kavac en su fase inicial de su puesta en producción, sin embargo la aplicación permite un esquema escalable en el que se pueden colocar uno o más servidores

proxy, uno o más servidores HTTP separados de los servidores de aplicación, todo dependerá de las necesidades de la institución.

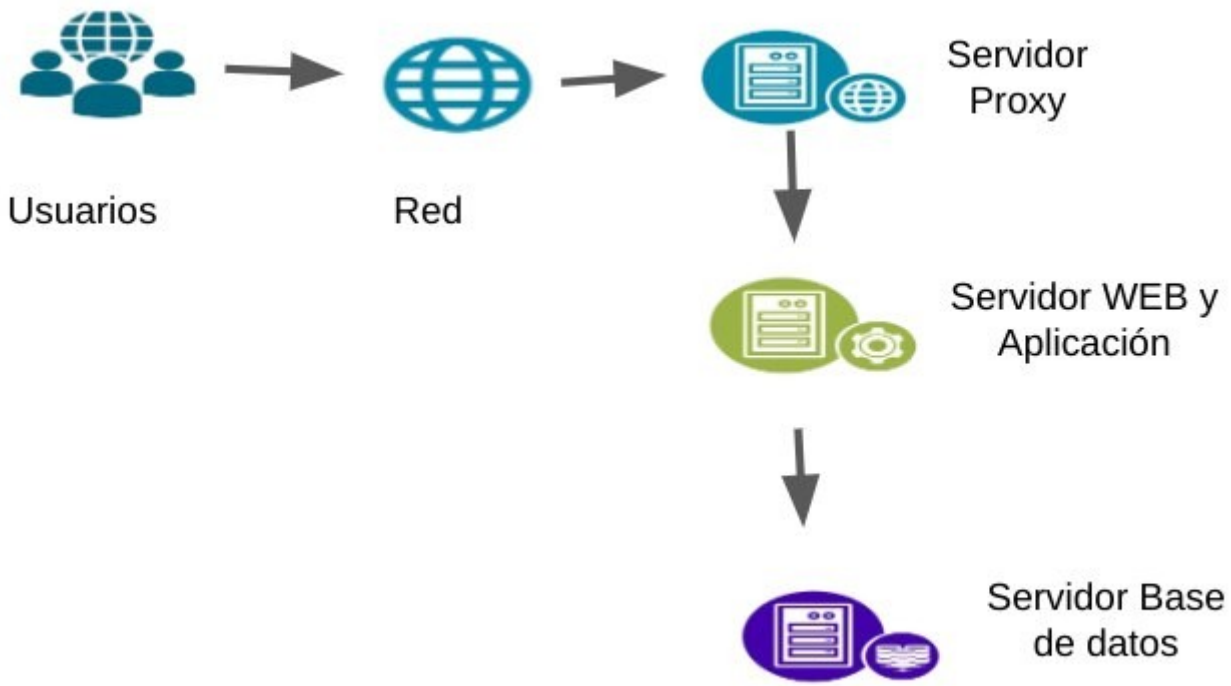
**Nivel 0 - Base de datos:** sistema basado en PostgreSQL con uno o más servidores organizados de forma que la complejidad interna del sistema sea transparente a la aplicación. Permitiendo la optimización y mejoras en cuanto al tiempo de respuesta y en la capacidad de cálculo. Se debe tomar en cuenta la adaptación de ser necesario para la gestión de grandes volúmenes de datos pero no limitativa.

**Nivel 1 - Aplicación:** uno o más servidores para la aplicación Kavac desarrollada en Lavarel. La aplicación se conecta al servidor de Base de datos y al servidor de capa web mediante un motor de aplicaciones como PHP-FPM o UWSGI.Si coloca dos o más servidores de aplicación se deben sincronizar el directorio /storage.

**Nivel 2 – Capa web - HTTP:** dos o más servidores web que se conectan a los servidores de aplicación, para esta capa se puede usar servidores web como Ngnix o Apache.

**Nivel 3 – Capa Proxy Reverso:** dos o más servidores proxy reverso con Nginx que se comunican a los servidores web.

A continuación se muestra el esquema recomendado para el despliegue en producción del sistema Kavac, en el que se unifica la capa web y la capa de aplicación como se puede observar en el siguiente esquema:




Esquema. Despliegue de 3 niveles o capas

El despliegue requiere el registro de nombres de dominio para los servidores de aplicación y el sistema de base de datos. Para mantener la seguridad y confiabilidad del despliegue, los servidores deberán estar en una o varias redes privadas y los nombres de dominio asociados a ellos corresponderá también a una o varias zonas DNS privadas. Será también necesario el registrar un dominio DNS público que apunte a las direcciones IP donde los proxy atienden peticiones HTTP o HTTPS.

Adicionalmente, se requiere de otras configuraciones de redes entre los servidores.

Las pruebas no funcionales han determinado que los requerimientos mínimos de hardware para la puesta en producción del Kavac en el esquema recomendado para la fase inicial son los siguientes:

- Servidor Proxy reverso (1 GB de RAM y 1 CPU).
- Servidor de aplicación-web (4GB de RAM y 2 CPUs).
- Servidor de Base de Datos (2 GB de RAM y 2 CPUs).

**Los componentes de software principales recomendados son los siguientes:**

- **Servidor Proxy Reverso:**
  - Sistema Operativo GNU/Linux Debian 12 - bookworm x86-64.
  - Nginx.

- **Servidor de Aplicación-web:**
  - Sistema Operativo GNU/Linux Debian 12 - bookworm x86-64.
  - Aplicación web Kavac (Proporcionada a través de un .zip).
  - Motor de aplicación: uWSGI,  uwsgi-plugin-php.
  - php8.2 php8.2-gd php8.2-mbstring php8.2-xml php-tokenizer php8.2-zip php8.2-pgsql php8.2-cli php8.2-curl zip unzip php-redis build-essential libssl-dev.

- composer.
- NVM.
- node js version 14.18.2
- npm.
- Nginx.

- **Servidor Base de Datos:**
  - Sistema Operativo GNU/Linux Debian 12 - x86-64.
  - PostgreSQL 15.
  - Redis.

# Despliegue del KAVAC en tres capas.

## Sección I

## 1. Configuración de la capa de base de datos

En esta sección se describe la instalación y configuración de un sistema de Base de Datos PostgreSQL basado en un servidor único, con algunas sugerencias de parámetros para una fase inicial de despliegue de Kavac. Este servidor puede utilizarse para realizar pruebas previas a la puesta en producción, sin embargo para la puesta en producción deberá utilizarse los servicios de un sistema de base de datos que incorpore redundancia y escalabilidad, sistema cuya instalación y configuración está fuera del alcance del presente manual. Este procedimiento es para el caso que no se tenga un servidor de Base de datos, si se tiene solo es necesario crear la Base de Datos, el usuario y definir permisos de acceso a dicha base de datos como se explica más adelante.

En el servidor de base de datos instalar el paquete de postgresql versión 15

```
# apt install postgresql-15
```

La configuración de la base de datos y el usuario de la misma tiene que ser adecuada a la configuración definida en el archivo .env de la aplicación Kavac.

Cambiar el intérprete de comandos al entorno del usuario postgres:

```
# su - postgres
```

Crear el usuario de base de datos kavac y la base de datos kavac:

```
$ createuser -P kavac
Ingrese la contraseña para el nuevo rol:
Ingresarla nuevamente:
```

```
$ createdb kavac -O kavac
```

**Nota:** El usuario y base de datos tiene un nombre que puede cambiar según definiciones de la plataforma y el encargado de la administración de la Base de Datos.

Cerrar el entorno del usuario postgres:

```
$ exit
```

Se debe modificar el archivo de configuración de postgresql para mejorar el manejo de un mayor volumen de conexiones a la base de datos, según lo requerido por la aplicación y criterios del encargado de la administración de la BD, dichas variables se configuran en el archivo /etc/postgresql/15/main/postgresql.conf. Adicionalmente, se debe cambiar los valores de la directiva listen_addresses:

```
listen_addresses = 'localhost, x.x.x.x'
```

**Nota:** Se debe colocar la dirección IP (x.x.x.x) por donde se escucharán las peticiones del servicio postgresql.

Para habilitar al servidor(es) de aplicación acceso a la base de datos es necesario modificar el archivo /etc/postgresql/15/main/pg_hba.conf, colocando la siguiente configuración:

```
# TYPE    DATABASE    USER    ADDRESS        METHOD
host      kavac       kavac   x.x.x.x/32     scram-sha-256
```

La configuración permite que el servidor(es) de aplicación pueda conectarse a la base de datos kavac con el usuario kavac con autenticación scram-sha-256 y con la dirección IP del servidor de aplicaición (x.x.x.x).

Reiniciar y activar el servidor de BD:

```
# systemctl restart postgresql
# systemctl enable postgresql
```

## 2-Instalación y configuración de Redis.

Instalar el paquete de redis

```
# apt install redis
```

Establecer las configuraciones que modifiquen la configuración de redis según lo requerido por la aplicación y criterios del encargado de la administración de la BD. Adicionalmente, se debe cambiar IP por la cual escuchará las peticiones y establecer la contraseña (la contraseña debe ser igual a la del archivo .env)

```
# vim /etc/redis/redis.conf
```

```
bind x.x.x.x 127.0.0.1
...
requirepass u8M6cxOGwVwHyPNL4fO0uG7cb9v9yJ3
```

**Nota:** se debe agregar la IP por donde escuchara el servicio y la contraseña para el servicio, la cual debe ser agregada en el archivo .env de la aplicación.

Reiniciar y activar el servicio

```
# systemctl restart redis
```

## Seccion II

### 1-Configurar la capa de aplicación.

El dominio que se usará en el presente manual como ejemplo será: kavac.gob.ve.

Los procedimientos descritos en esta sección han de realizarse en todos los servidores de la capa aplicación. En caso que se desee configurar el esquema 2, los procedimientos descritos han de realizarse en todos los servidores web+app.

### 1.1-Crear y preparar el directorio donde se hospedara el código fuente

Crear el usuario del sistema por ejemplo kavac

```
# adduser kavac --system --group --home /usr/local/lib/kavac
```

Asignar una shell al usuario de sistema creado

```
# chsh -s /bin/bash kavac
```

Con esto ya tenemos el directorio /usr/local/lib/kavac con el usuario de sistema llamado kavac. Para acceder a la aplicación con:

```
# su - kavac
```

*Por lo tanto a partir de ahora se debe tener en cuenta lo siguiente:*

*$ hace referencia que los comandos a ser ejecutados serán con el usuario de sistema llamado “kavac”.*

*# hace referencia que los comandos a ser ejecutados serán con usuario root.*

**1.2-Enviar el código fuente de la aplicación KAVAC e instalar requerimientos de software**

Para enviar el comprimido al servidor se hace necesario hacer uso de alguna utilidad que proporcione funciones de copia o sincronización de archivos hacia servidores remotos tales como scp o rsync. En el ejemplo a continuación se realiza mediante rsync el cual debe estar instalado tanto en la computadora local como en el servidor remoto. Para descomprimir el archivo se debe instalar unzip en el servidor app-web:

```
# apt install rsync unzip
```

Ejecutar el comando para enviar empaquetado del código fuente desde su computadora local hacia el servidor remoto:

```
$ rsync -avcp kavac-develop.zip mv-web-app@x.x.x.x:
```

Donde,

mv-web-app, es el nombre de usuario regular de la máquina virtual.

x.x.x.x es la IP del servidor por la que se inicia sesión vía ssh.

Dentro del servidor con usuario root, mover el empaquetado del proyecto a la ruta de hospedaje

```
#  mv /home/mv-web-app/kavac-develop.zip /usr/local/lib/kavac/
```

Iniciar sesión con el usuario kavac creado anteriormente y descomprimir dentro de un directorio llamado kavac

```
# su - kavac
$ unzip -q kavac-develop.zip -d kavac
```

Listar y verificar permisos de usuario kavac al directorio previo

```
$ ls -l
total 187976
drwxr-xr-x 13 kavac       kavac                    4096 feb 27 12:12 kavac
-rw-r--r--  1 mv-web-app mv-web-app 192481429 feb 27 10:57 kavac-develop.zip
```

**1.3-Instalar los requerimientos de software y librerías de la aplicación, dicha información se encuentra en el archivo README del sistema.**

Para volver al superusuario del sistema se debe cerrar la sesión del usurio kavac ejecutando

```
$ exit
```

### 1.3.1-Instalar los requerimientos de software

```
# apt install php8.2 php8.2-gd php8.2-mbstring php8.2-xml php-tokenizer php8.2-zip php8.2-pgsql php8.2-cli php8.2-curl zip unzip php-redis build-essential libssl-dev postgresql-client curl
```

### 1.3.2-Instalar el gestor de paquetes de php composer

Para ello se sigue el procedimiento de instalación directamente desde la página oficial https://getcomposer.org/download/ ya que el hash suele cambiar. A continuación el ejemplo de la instalación del composer:

```
$ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"

$ php -r "if (hash_file('sha384', 'composer-setup.php') === 'e21205b207c3ff031906575712edab6f13eb0b361f2085f1f1237b7126d785e826a450292b6cfd1d64d92e6563bbde02') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"

$ php composer-setup.php

$ php -r "unlink('composer-setup.php');"
```

El composer se encuentra instalado en el directorio del sistema, sin embargo, se desea llamar al gestor desde cualquier otra ruta (instalación global) usando, por ejemplo:

```
$ exit
# cd /usr/local/lib/kavac/kavac
# mv composer.phar /usr/local/bin/composer
```

Una vez instalado composer comprobamos la correcta instalación con el siguiente comando:

```
$ composer --version
```

### 1.3.3-Instalar NVM (Node Version Manager)

Instalar la versión mas reciente de NVM (Node Version Manager) chequeando en el siguiente enlace la versión: https://github.com/nvm-sh/nvm/releases

Si el directorio del usuario “kavac” no tiene los archivos ocultos .profile, .bashrc y .zshrc se deben crear de la siguiente manera:

**a.** Crear archivo .profile

```
$ vim .profile
```

```
# ~/.profile: executed by Bourne-compatible login shells.
  if [ "$BASH" ]; then
   if [ -f ~/.bashrc ]; then
   . ~/.bashrc
 fi
  fi

mesg n 2> /dev/null || true
```

Guardar y cerrar

Instalar la versión mas reciente de nvm, se puede chequear en el siguiente enlace: https://github.com/nvm-sh/nvm/releases.

```
$ curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.39.7/install.sh | bash
```

**b.** Crear archivo .bashrc que es el archivo de inicio individual por shell interactivo

```
$ vim .bashrc
```

Agregamos el contenido para que inicialice NVM

```
# ~/.bashrc: executed by bash(1) for non-login shells.


# Note: PS1 and umask are already set in /etc/profile. You should not
# need this unless you want different defaults for root.
# PS1='${debian_chroot:+($debian_chroot)}\h:\w\$ '
# umask 022


# You may uncomment the following lines if you want `ls' to be colorized:
# export LS_OPTIONS='--color=auto'
# eval "`dircolors`"
# alias ls='ls $LS_OPTIONS'
# alias ll='ls $LS_OPTIONS -l'
# alias l='ls $LS_OPTIONS -lA'
#
# Some more alias to avoid making mistakes:
# alias rm='rm -i'
# alias cp='cp -i'
```

```
# alias mv='mv -i'


export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
```

Guardar y cerrar

**c.** Crear archivo .zshrc

```
$ touch .zshrc
```

Configurar el acceso del nuevo script **NVM**:

```
$ . ~/.profile
```

Instalar la versión de node 14.18.2:

```
$ nvm install 14.18.2
```

Una vez instalado NodeJS comprobamos la correcta instalación:

```
$ node -v
```

Instalar la versión de npm 8.11.0

```
$ npm install -g npm@8.11.0
$ npm -v
```

Copiar archivo de configuración .env que se encuentra dentro del proyecto:

```
$ cd kavac
$ cp .env.example .env
```

En el archivo .env se debe establecer los parámetros de configuración necesarios bajo los cuales se ejecutará la aplicación

```
APP_NAME=KAVAC
APP_ENV=production
APP_KEY=
APP_DEBUG=false
APP_LOG_LEVEL=error
APP_URL=https://kavac.gob.ve
ASSET_URL=https://kavac.gob.ve
```

```
APP_TIMEZONE='America/Caracas'
APP_DEMO=false
#Colocar en true para servidores de prueba
APP_TESTING=false
#Url del servidor de pruebas (incluyendo subcarpetas si es que esta configurado de tal forma)
APP_TESTING_URL=http://localhost
#Plantilla blade en donde se encuentra el formulario de autenticación de usuario, modificar si se crea un nuevo método de autenticación que requiera un procedimiento distinto
AUTH_VIEW="auth.login"
#Condición que establece el uso de Active Directory como método de autenticación
ACTIVE_DIRECTORY=false
ACTIVE_DIRECTORY_URL=
ACTIVE_DIRECTORY_DN=
ACTIVE_DIRECTORY_BASE_DN=

#Habilita/Deshabilita los respaldos de base de datos desde la interfaz de la aplicación
BACKUP_ENABLED=true

#Indica si la aplicación esta en modo de pruebas unitarias
TEST_UNIT=false
```

```
#Indica si la aplicación permite la auditoria por consola, colocar en false si se requiere no auditar los procesos realziados por consola
APP_AUDIT=true

AUDIT_LIMIT=13

# Indica si hace o no uso del código de la ONAPRE
USE_ONAPRE=false

LOG_CHANNEL=daily
LOG_SLACK_WEBHOOK_URL=

DB_CONNECTION=pgsql
DB_HOST=x.x.x.x
DB_PORT=5432
DB_DATABASE=kavac
DB_USERNAME=kavac
DB_PASSWORD=123456
```

```
BROADCAST_DRIVER=pusher
#Driver para gestionar el cache de la aplicación, los valores permitidos son database o redis
CACHE_DRIVER=redis
#Driver para gestionar las sesiones, los valores permitidos son database o redis
SESSION_DRIVER=redis
#Establecer en default si la variable SESSION_DRIVER se define como redis, de lo contrario dejar en null
SESSION_CONNECTION=null
#Tiempo de vida de la sesión
SESSION_LIFETIME=120
#Indica si las cookies solo se pueden enviar bajo protocolo seguro https
SESSION_SECURE_COOKIE=true
#Establece la bandera HttpOnly
SESSION_HTTP_ONLY=true
#Envía las variables de sesión cifradas
SESSION_ENCRYPT=true
#Valores aceptados: lax, strict
SESSION_SAME_SITE=lax
#Conexión para gestionar las colas, los valores permitidos son database o redis
QUEUE_CONNECTION=redis
```

```
REDIS_HOST=x.x.x.x
#Contraseña del servidor Redis
REDIS_PASSWORD=password
REDIS_PORT=6379
#Cliente redis, los valores permitidos son predis o phpredis
REDIS_CLIENT=phpredis
REDIS_DB=0
REDIS_CACHE_DB=1
REDIS_QUEUE_DB=2
REDIS_QUEUE_PREFIX='q:'
REDIS_QUEUE='default'
REDIS_QUEUE_CONN='queue'

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
```

```
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=kavac@example.com

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

CAPTCHA_COLOR_ONE=false

# longitud de 5 dígitos
PUSHER_APP_ID=anyid
#Longitud de 10 carácteres
PUSHER_APP_KEY=somekey
#Longitud de 12 carácteres
PUSHER_APP_SECRET=somesecret
PUSHER_APP_CLUSTER=mt1
PUSHER_APP_TLS=true
PUSHER_APP_PATH=
```

```
WEBSOCKETS_HOST=kavac.gob.ve
WEBSOCKETS_PORT=443
WEBSOCKETS_SSL_LOCAL_CERT=null
WEBSOCKETS_SSL_LOCAL_PK=null
WEBSOCKETS_SSL_PASSPHRASE=null
#Colocar en true si se requiere la implementación de tls
WEBSOCKETS_TLS=false

BROADCAST_HOST=x.x.x.x
BROADCAST_PORT=6001
BROADCAST_TLS=false
BROADCAST_SCHEME=http

MIX_APP_URL="${APP_URL}"
MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
MIX_WEBSOCKETS_HOST="${WEBSOCKETS_HOST}"
MIX_WEBSOCKETS_PORT="${WEBSOCKETS_PORT}"
```

```
MIX_WEBSOCKETS_TLS="${WEBSOCKETS_TLS}"
# Tiempo de vida de la sesión
MIX_SESSION_LIFETIME="${SESSION_LIFETIME}"
```

**Nota:** En las variables DB_HOST, REDIS_HOST, WEBSOCKETS_HOST el valor x.x.x.x se debe reemplazar por la IP de escucha de cada uno de los servicios (Base de datos, Redis y Websockets). En la variables MAIL_* se debe configurar con los valores de su servicio smtp.

Instalar las dependencias de php de la aplicación

```
$ composer install
```

Generar un identificador único para la aplicación, al generarse, se agrega automáticamente a la variable APP_KEY= del archivo .env anterior ya que por defecto está vacía

```
$ php artisan key:generate
```

Instalar los paquetes necesarios para la gestión reactiva de datos

```
$ npm install
```

Compilar archivos js y css de cada módulo de la aplicación, a su vez la instalación de paquetes de cada módulo

```
$ php artisan module:compile -s -i -p
```

Cabe destacar que luego de haber instalado la dependencias y paquetes, no es necesario ejecutar de nuevo dichos comandos, al menos que haya actualización de los mismos. Si se desea sólo compilar el sistema se puede hacer mediante el siguiente comando:

```
$ php artisan optimize:clear
$ php artisan module:compile -s -p
```

Generar la estructura de la base de datos inicial

```
$ php artisan migrate
```

Cargar la información inicial del sistema

```
$ php artisan db:seed
$ php artisan module:seed
```

### 1.4-Configuración del motor de aplicaciones (uWSGI)

Instalar los siguientes paquetes del repositorio:

```
# apt-get install uwsgi uwsgi-plugin-php
```

Colocar el siguiente contenido en /etc/uwsgi/apps-available/kavac.ini

```
[uwsgi]
plugins=php
processes=2
master=true
cheaper=1
uid=kavac
gid=kavac
log-x-forwarded-for=true
max-requests=521
post-buffering=4096
buffer-size=6144
harakiri=521
harakiri-verbose=true
```

```
socket=x.x.x.x:1597
project_dir = /usr/local/lib/kavac/kavac/public/
chdir=%(project_dir)
; jail our php environment to project_dir
php-docroot = %(project_dir)
php-allowed-docroot = %(project_dir)
; ... and to the .php and .inc extensions
php-allowed-ext = .php
php-allowed-ext = .inc
; and search for index.php and index.inc if required
php-index = index.php
php-index = index.inc
```

**Nota:** En la variable socket se debe colocar la dirección IP (x.x.x.x) por donde escuchará cada servidor de aplicación.

Crear un enlace simbólico para activar la configuración de uWSGI:

```
# cd /etc/uwsgi/apps-enabled/
# ln -s ../apps-available/kavac.ini
# systemctl restart uwsgi
```

```
# systemctl enable uwsgi
```

Dirigirse con el comando cd a la siguiente ruta /etc/php/8.2/embed/conf.d/ y crear el archivo para agregar las directivas de php.

```
# vim 31-kavac.ini
```

Agregar la información al archivo

```
# Tamaño máximo de datos a través del método POST
post_max_size = 41M
# Tamaño máximo de archivos permitidos para cargar al servidor
upload_max_filesize = 13M
# Número máximo de archivos que se pueden cargar mediante una sola solicitud
max_file_uploads = 53
# Cantidad máxima de memoria que un script puede asignar
memory_limit= 512M
FUNDACIÓN CENTRO NACIONAL DE DESARROLLO E INVESTIGACIÓN EN TECNOLOGÍAS LIBRES (CENDITEL) Avenida Alberto Carnevali, vía La Hechicera, Edificio CENDITEL. Teléfono:
(0274)6574336. Apartado Postal: 078. Código Postal: 5101A. RIF: G 20007349-7. Mérida- Venezuela
# Tiempo máximo de ejecución de cada script, en segundos
```

```
max_execution_time = 300
```

### 1.5-Configurar el procesamiento de colas y notificaciones

La activación del procedimiento de colas y el WebSocket se activarán mediante el gestor de servicios y daemons de los sistemas operativos GNU/Linux llamado Systemd.

#### 1.5.1-Worker (colas)

Crear el daemon en un archivo ubicado en /etc/systemd/system/ llamado kavac-worker@.service para instancias de colas con el siguiente contenido:

```
[Unit]
Description=Obrero procesador de colas de sistema kavac - cola %i
After=network.target

[Service]
Type=exec
Restart=always
RestartSec=233
```

```
TimeoutStopSec=4253
RuntimeMaxSec=28657
ExecStart=/usr/bin/php /usr/local/lib/kavac/kavac/artisan queue:work redis --queue=%i --sleep=3 --tries=3
User=kavac
Group=kavac

[Install]
WantedBy=multi-user.target
```

Recargar el daemon de systemd:

```
# systemctl daemon-reload
```

Activar, iniciar el servicio para la cola default y verificar su status:

```
# systemctl enable kavac-worker@default.service
# systemctl restart kavac-worker@default.service
# systemctl status kavac-worker@default.service
```

Activar, iniciar el servicio para la cola bulk y verificar su status:

```
# systemctl enable kavac-worker@bulk.service
```

```
# systemctl restart kavac-worker@bulk.service
# systemctl status kavac-worker@bulk.service
```

Puede realizar el seguimiento con el comando:

```
# journalctl --unit=kavac-worker@default
```

### 1.5.2-Websocket

Crear el daemon en un archivo ubicado en /etc/systemd/system/ llamado kavac-websockets.service con la siguiente directivas:

```
[Unit]
Description=Obrero gestor de conexiones websocket de sistema kavac
After=network.target

[Service]
Restart=always
RestartSec=233
TimeoutStopSec=4253
ExecStart=/usr/bin/php /usr/local/lib/kavac/kavac/artisan websockets:serve --host x.x.x.x
```

```
User=kavac
Group=kavac

[Install]
WantedBy=multi-user.target
```

**Nota:** En la variable host se debe colocar la dirección IP (x.x.x.x) por donde escuchará cada servicio de websockets, por la cual se establecerá la comunicación entre proxy y websockets.

Recargar el daemon de systemd, activar e iniciar el servicio

```
# systemctl daemon-reload
# systemctl enable kavac-websockets.service
# systemctl start kavac-websockets.service
```

Verificar que el servicio se activó

```
# systemctl status kavac-websockets.service
```

## 2- Instalar y configurar Nginx como servidor web

Configuración del sitio para la aplicación KAVAC.

## 2.1-Instalación y configuración general de Nginx

Instalar nginx desde el repositorio Debian:

```
# apt install nginx
```

Agregar y configurar en /etc/nginx/nginx.conf las siguientes directivas:

```
user www-data;
worker_processes 1;
pid /run/nginx.pid;
.
.
.

http {
    .
    server_tokens off;
    .
    ##
```

```
  # Logging Settings
  ##

  access_log /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;

  log_format main '$http_x_real_ip - $remote_addr - $remote_user [$time_local] $request '

  ' "$status" $body_bytes_sent "$http_referer" '
  ' "$http_user_agent" "$http_x_forwarded_for" ';

 .
 .
 .
}
```

**Nota:** El número de procesos que indica la directiva “worker_processes” es conveniente que sea igual a la cantidad de núcleos o vCPUs que tenga el servidor. Para verificar la cantidad de vCPU utilice el comando lscpu.

### 2.3-Configuración del sitio por defecto de Nginx

Crear un archivo para la configuración por defecto con las siguientes directivas en /etc/nginx/sites-available/limbo:

```
server {
    listen x.x.x.x:80 default_server;
    server_name _;
    return 444;
}
```

**Nota:** Se establece con esta configuración que el servidor web descarte peticiones HTTP que no estén asociadas a uno de los dominios indicados en la directiva server_name de los otros archivos de configuración que estén habilitados. Se debe colocar la dirección IP (x.x.x.x) por donde se escucharán las peticiones de HTTP.

### 2.4-Configuración del sitio para la aplicación KAVAC

La comunicación entre la aplicación y el servidor web se puede establecer mediante un motor de aplicación como UWSGI o FASTCGI (php-fpm). En el presente manual se describe la configuración con UWSGI.

Configuración del sitio para la aplicación KAVAC con UWSGI

Crear el archivo /etc/nginx/sites-available/kavac con el siguiente contenido:

```
upstream kavac-app {
        server unix:/run/uwsgi/app/kavac/socket max_fails=1 fail_timeout=13s;
}

server {
    listen x.x.x.x:80;
    server_name kavac.gob.ve;

    access_log /var/log/nginx/kavac.access.log main;
    error_log /var/log/nginx/kavac.error.log error;

    client_max_body_size 20M;
    client_body_buffer_size 128k;

    set_real_ip_from x.x.x.x;
    real_ip_header X-Real-IP;

    ## Devolver un estatus 204 (No Content) para el favicon.ico
```

```
 location = /favicon.ico {
     return 204;
 }

 location /docs/user {
    alias /usr/local/lib/kavac/kavac/public/docs/user;
}
location / {
    root /usr/local/lib/kavac/kavac/public;
    try_files $uri @kavac;
}

location @kavac {
    include uwsgi_params;
    rewrite ^$ / last;
    uwsgi_param UWSGI_SCHEME  $http_x_forwarded_protocol;
    uwsgi_param HTTPS  $http_x_forwarded_https;
    uwsgi_param REQUEST_SCHEME $http_x_scheme;
    uwsgi_param REQUEST_URI $document_uri;
```

```
        uwsgi_param PATH_INFO /index.php;
        uwsgi_modifier1 14;
        uwsgi_pass kavac-app;
        uwsgi_intercept_errors off;
        uwsgi_read_timeout 521;
        uwsgi_buffers 16 16k;
        uwsgi_buffer_size 32k;
        uwsgi_busy_buffers_size 32k;
    }
    #error_page 404 /404.html;

    # redirect server error pages to the static page /50x.html
    #
     error_page 500 502 503 504 /50x.html;
     location = /50x.html {
     root /srv/www/kavac;
    }
}
```

**Nota:** En la sección upstream de este archivo se establece la conexión al servidor de aplicación y con el Websocket, de igual

manera puede haber más de un servidor aplicación los cuales deben estar sincronizados, en caso de haber más servidores de aplicación en el despliegue, se deberán agregar a esa sección directivas similares a las colocadas en esta oportunidad.

**2.5-Crear el directorio /srv/www/kavac/ y dentro de él, un archivo HTML para la página de error llamado 50x.html**

```
# mkdir -p /srv/www/kavac/
```

50x.html

```
<html>
<head>
<title>Error interno del servidor</title>
</head>
<body>
<h1>Error interno del servidor</h1>
</body>
</html>
```

**2.6-Establecer la activación del sitio kavac de manera automática, creando un enlace simbólico en el directorio sites-enabled:**

```
# cd /etc/nginx/sites-enabled/
```

```
# ln -s ../sites-available/kavac
# ln -s ../sites-available/limbo
# rm default
```

### 2.7-Reiniciar el nginx y verificar que no reporte errores en el arranque:

```
# nginx -t
# systemctl restart nginx
# systemctl enable nginx
```

## Sección III

### 1-Instalar y configurar la capa Proxy

Los procedimientos descritos en esta sección han de realizarse en todos los servidores de la capa proxy.

### 1.1-Instalación y configuración general de nginx

Instalar nginx desde el repositorio Debian:

```
# apt install nginx
```

Agregar y configurar /etc/nginx/nginx.conf con las directivas siguientes:

```
user www-data;
worker_processes 1;
pid /run/nginx.pid;
.
.
.
http {
.
    server_tokens off;
.
    ##
    # SSL Settings
    ##

ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
ssl_prefer_server_ciphers on;
ssl_ciphers     ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-
SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-
ECDSA-AES128-SHA:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-
SHA:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:DHE-DSS-AES256-
```

```
GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:DHE-DSS-AES128-GCM-SHA256:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK;
        ssl_session_cache shared:SSL:10m;
        ssl_session_timeout 5m;

        ##
        # Logging Settings
        ##

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        log_format main      '$http_x_real_ip - $remote_addr - $remote_user [$time_local] $request '

       ' "$status" $body_bytes_sent "$http_referer" '
       ' "$http_user_agent" "$http_x_forwarded_for" ';

   .
```

```
  .
  .
}
```

**Nota:** El número especificado en la directiva “worker_processes” es conveniente que sea igual a la cantidad de núcleos o vCPUs que tenga el servidor.

### 1.2-Configuración del sitio por defecto de Nginx

**Nota:** Con esta configuración nginx va a ignorar las peticiones HTTP o HTTPS cuyo encabezado HOST no sea reconocido por las directivas “server_name” correspondientes a otros archivos de configuración activos, o que no contienen ese encabezado. Para el protocolo HTTPS se usa un certificado en blanco para con ello evitar enviar información a software cliente que esté mal configurado o sea malicioso.

Crear un archivo con las siguientes directivas en /etc/nginx/sites-available/limbo:

```
server {
    listen x.x.x.x:80 default_server;
    server_name _;
```

```
    return 444;
}

server {
    listen x.x.x.x:443 ssl http2 default_server;
    server_name _;
    ssl_certificate             /etc/nginx/ssl_keys/default_blank.crt;
    ssl_certificate_key         /etc/nginx/ssl_keys/default_blank.key;
    keepalive_timeout           70;
    return 444;
}
```

**Nota:** Se establece con esta configuración que el servidor web descarte peticiones HTTP y HTTPS que no estén asociadas a uno de los dominios indicados en la directiva “server_name” de los otros archivos de configuración que estén habilitados. Se debe colocar la dirección IP (x.x.x.x) por donde se escucharán las peticiones de http.

### 1.3-Generar una clave con curva elíptica sin encriptar y en blanco:

```
# mkdir /etc/nginx/ssl_keys
# cd /etc/nginx/ssl_keys
```

```
# openssl ecparam -name secp384r1 -genkey -out default_blank.key
```

Generar una solicitud de certificado en blanco

```
# openssl req -out default_blank.csr -utf8 -new -key default_blank.key -sha512
```

En las preguntas que realiza este último comando, sólo se ha de responder la correspondiente al campo “País”, en donde se puede colocar cualesquiera dos letras. Todos los demás campos se han de dejar vacíos, lo cual requiere responder con un “.” (carácter punto) las preguntas que tengan un valor por defecto no vacío.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----


Country Name (2 letter code) [AU]:VE
```

```
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:.
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (e.g. server FQDN or YOUR name) []:.
Email Address []:.
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Como el certificado en blanco sólo será proporcionado a software cliente mal configurado o malicioso, y con el único objetivo de cerrar la conexión apenas sea completada la negociación SSL, entonces se puede generar de forma auto-firmada:

```
# openssl x509 -req -in default_blank.csr -out default_blank.crt -signkey default_blank.key -days 730 -sha512
```

Otorgar permisos de sólo lectura para el usuario root (modo 0400) a los archivos default_blank.crt y default_blank.key

```
# chmod 0400 default_blank.crt
# chmod 0400 default_blank.key
```

**1.4-Configurar proxy reverso para el sitio web.**

Crear el archivo /etc/nginx/sites-available/kavac con el siguiente contenido:

```
upstream kavac {
    server      x.x.x.x:80    max_fails=1 fail_timeout=10s;
}

upstream kavac-websocket {
    server      x.x.x.x:6001  max_fails=1 fail_timeout=10s;
}

map $http_upgrade $type_conn {
    default      @web;
    websocket    @ws;
}

server {
    listen x.x.x.x:80;
    server_name kavac.gob.ve;
```

```
access_log /var/log/nginx/kavac.access.log main;
error_log /var/log/nginx/kavac.error.log error;

## Devolver un estatus 204 (No Content) para el favicon.ico
location = /favicon.ico {
  return 204;
 }

 ## Devolver un estatus 204 (No Content) para el robots.txt
location = /robots.txt {
  return 204;
 }

location / {
    return 301 https://kavac.gob.ve$request_uri;
}

#error_page 404 /404.html;
```

```
    # redirect server error pages to the static page /50x.html
    #
    error_page 500 502 504 /50x.html;
    location = /50x.html {
        root /srv/www/kavac;
    }

    error_page 503 /503.html;
    location = /503.html {
        root /srv/www/kavac;
    }

}

server {
     listen x.x.x.x:443 ssl;
     server_name kavac.gob.ve;
```

```
ssl_certificate    /etc/nginx/ssl_keys/kavac.gob.ve.crt;
ssl_certificate_key /etc/nginx/ssl_keys/kavac.gob.ve.key;

access_log    /var/log/nginx/kavac.access-ssl.log main;
error_log    /var/log/nginx/kavac.error-ssl.log error;

client_max_body_size 20M;
client_body_buffer_size 128k;

## Devolver un estatus 204 (No Content) para el robots.txt
location = /robots.txt {
return 204;
}

location ~ ^/ {
    try_files /nonexistent $type_conn;
}
```

```
location ~ ^/app {
    try_files /nonexistent @ws;
}

location @web {
    include        proxy_params;
    proxy_pass      http://kavac;
    proxy_set_header  X-Forwarded-Protocol $scheme;
    proxy_set_header  X-Forwarded-Https $https;
    proxy_read_timeout 90;
    proxy_buffer_size 128k;
    proxy_buffers 4 256k;
    proxy_busy_buffers_size 256k;
}

location @ws {
    proxy_pass     http://kavac-websocket;
    proxy_read_timeout  60;
    proxy_connect_timeout 60;
```

```
    proxy_redirect off;
    # Allow the use of websockets
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}

#error_page 404 /404.html;
# redirect server error pages to the static page /50x.html
#
error_page 500 502 504 /50x.html;
location = /50x.html {
    root /srv/www/kavac;
}

error_page 503 /503.html;
location = /503.html {
```

```
        root /srv/www/kavac;
    }
}
```

**Nota:** Con esta configuración el nginx va a generar la redirección permanente:

origen: http://kavac.gob.ve/(.*)$
destino: https://kavac.gob.ve>/$1

De esa manera el KAVAC solo se accede vía protocolo HTTPS, lo cual requiere tener disponible un certificado digital para el dominio donde será publicado el KAVAC. Ese certificado deberá ser otorgado por una autoridad de certificación reconocida y sus archivos deberán estar disponibles en los servidores proxy con permisos de solo lectura para el usuario root (modo 0400). Sin embargo para efectos de pruebas y de este manual se generan certificados autofirmados con openssl de la siguiente manera:

**1.5-Generar los certificados:**

```
# cd /etc/nginx/ssl_keys
# openssl ecparam -name secp384r1 -genkey -out kavac.gob.ve.key
```

```
# openssl req -out kavac.gob.ve.csr -utf8 -new -key kavac.gob.ve.key -sha512
```

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:VE
State or Province Name (full name) [Some-State]:MERIDA
Locality Name (eg, city) []:MERIDA
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CENDITEL
Organizational Unit Name (eg, section) []:DESARROLLO
Common Name (e.g. server FQDN or YOUR name) []:kavac.gob.ve
Email Address []:lcolina@cenditel.gob.ve
Please enter the following 'extra' attributes
to be sent with your certificate request
```

A challenge password []:

An optional company name []:

*# openssl x509 -req -in kavac.gob.ve.csr -out kavac.gob.ve.crt -signkey kavac.gob.ve.key -days 730 -sha512*

kavac.gob.ve.crt y kavac.gob.ve.key

*# chmod 0400 kavac.gob.ve.crt*

*# chmod 0400 kavac.gob.ve.key*

**1.6-Crear el directorio /srv/www/kavac/ y dentro de él, un archivo HTML para la página de error llamado 50x.html y 503.html**

# mkdir -p /srv/www/kavac/

50x.html

*<html>*

*<head>*

```
<title>Error interno del servidor</title>
</head>
<body>
<h1>Error interno del servidor</h1>
</body>
</html>
```

503.html

```
<html>
<head>
<title>Servicio temporalmente no disponible</title>
</head>
<body>
<h1>Servicio temporalmente no disponible</h1>
</body>
</html>
```

**1.7-Establecer la activación automática del proxy reverso para el sitio KAVAC de manera automática, creando un enlace simbólico en el directorio sites-enabled**:

```
# cd /etc/nginx/sites-enabled/
# ln -s ../sites-available/kavac
# ln -s ../sites-available/limbo
# rm default
```

### 1.8-Verificar que no reporte errores en el arranque y reiniciar el nginx:

```
# nginx -t
# systemctl restart nginx
# systemclt enable nginx
```