



Universidad
Rafael Landívar
Tradición Jesuita en Guatemala

incyt

Instituto de investigación y proyección
sobre ciencia y tecnología

Documentación Técnica

PROYECTO "Guatemala en Datos"

Versión 1.0

Guatemala, 2021

Tabla de contenido

Prerrequisitos	3
Código de Docker	3
Node	3
Yarn	3
Código de Ejecución	3
Código de Docker-compose	4
Instrucciones para ejecutar el código	5
Pasos siguientes	9
Errores comunes	11
Cannot find module.....	11
Error en ejecución de yarn	11
Espacio insuficiente	11
Código de generación de contraseña cifrada.....	12
Librerías.....	12
Variables.....	12
Métodos	12
hashIt()	12
password()	12
Ejecución del código.....	13

Our World In Data Grapher es un proyecto desarrollado por la organización del mismo nombre para la creación de visualizaciones interactivas de datos que pueden ser incluidas en artículos científicos. El siguiente documento tiene como fin ser utilizado a manera de guía para la preparación de una versión contenerizada del graficador para su fácil uso e implementación. Adicionalmente, se describe el código realizado para la generación de la contraseña cifrada, el cual es parte del proceso de la actualización de contraseña.

Prerrequisitos

Para el correcto funcionamiento de la aplicación Our World In Data Grapher, es necesario contar con lo siguiente:

1. Docker instalado
2. Docker-compose instalado
3. (Opcional) Visual Studio Code
4. Tener descargado y descomprimido el código de owid-grapher.

Código de Docker

El código de Docker para la creación del contenedor se detalla por secciones a continuación.

Node

```
FROM node:12
```

La imagen es creada a partir de NodeJS versión 12.x.

```
# Install curl
RUN apt-get install -y git wget curl
```

Se instala Git en el contenedor.

```
COPY ./owid-grapher-master .
```

Se copia la carpeta perteneciente al repositorio, en este caso “owid-grapher-master”, en la carpeta raíz del contenedor. (**Nota: si la carpeta descargada se llama de diferente manera, modificarlo**)

Yarn

```
RUN yarn
```

Se corre el comando yarn para instalar las dependencias del proyecto.

```
RUN cp .env.example .env
```

Se copia el contenido del archivo .env.example en el archivo .env.

Código de Ejecución

```
CMD ["sh", "-c", "yarn startTscServer" ]
```

Se inicia el graficador mediante la instrucción yarn startTscServer.

Se establece que puertos del contenedor estarán expuestos (3030).

Código de Docker-compose

```
db:
  image: mysql
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: "lfarfan"
    MYSQL_DATABASE: owid
  ports:
    - 3306:3306
  volumes:
    - db-config:/etc/mysql
    - db-data:/var/lib/mysql
    - ./mysql/backup/files:/data_backup/data
  command: --default-authentication-plugin=mysql_native_password
```

Se define el servicio de base de datos, el cual utiliza una imagen de MySQL y crea la base de datos inicial llamada owid. También se indica que se expondrá el 3306 del contenedor en el 3306 de la máquina anfitrión. Luego se indican las carpetas donde se almacenarán los datos de la base de datos dentro del contenedor, y la carpeta donde se almacenarán los backups, en caso de realizar uno. Por último, se le indica el comando que se correrá al iniciar el contenedor.

```
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  links:
    - 'db'
  ports:
    - '8080:80'
  environment:
    MYSQL_ROOT_PASSWORD: lfarfan
    PMA_HOST: db
    UPLOAD_LIMIT: 300M
  restart: on-failure
```

Se define el servicio “phpmyadmin”, necesario para manejar la base de datos de mejor manera. Se utiliza la imagen oficial de phpmyadmin. Luego se le indica que debe crear una conexión con el servicio de “db” y que se mostrará en el puerto 8080 del anfitrión (se puede utilizar cualquier otro puerto si lo desea). Después se le debe indicar la contraseña de root de la base de datos, la cual es la indicada en el servicio anterior, se le indica que el host de base de datos será el servicio de db y se establece que el tamaño máximo de archivos a importar es de 300Mb (esto debido a un paso que hay que hacer más adelante). Por último, se le indica que en caso de fallo se debe reiniciar.

```
webapp:
  network_mode: host
  build:
    context: .
    dockerfile: Dockerfile
  ports:
    - 3030:3030
  depends_on:
    - phpmyadmin
```

Se define el servicio “webapp”. Primero se le indica que la red que utilizará será la del host. Esto es necesario para que la plataforma se muestre correctamente. Luego se indica el archivo Dockerfile que usará para la creación del contenedor. Este archivo es el indicado anteriormente. Después se indica qué puerto utilizará, el seleccionado es el 3030. Por último, se le indica que depende de phpmyadmin, esto únicamente para que el servicio de phpmyadmin comience antes que el “webapp”.

```
volumes:
  db-config:
  db-data:
```

Por último, se definen los volúmenes que se utilizan en la base de datos.

Instrucciones para ejecutar el código

****Importante: Utilizar sudo para todos los comandos con Docker-compose.**

1. Copiar el archivo Dockerfile y docker-compose.yml en la misma carpeta donde se encuentra la carpeta descomprimida del repositorio de owid-grapher.
2. De ser necesario, se debe cambiar el siguiente comando en el Dockerfile a la ruta de la carpeta donde se encuentra el repositorio copiado.

```
COPY ./owid-grapher-master .
```

3. Cambiar la contraseña de la base de datos en el archivo .env.example, dentro de la carpeta descomprimida del repositorio. Esta contraseña debe coincidir con la indicada en el servicio de base de datos del archivo Docker-compose.

```
DB_PASS='lfarfan'
```

4. Ejecutar el siguiente comando desde la línea de comando dentro de la carpeta principal:

docker-compose up

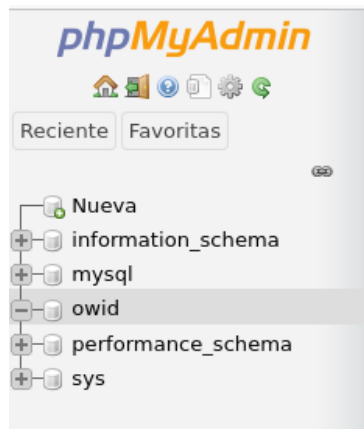
Este comando ejecuta el archivo docker-compose.yml creando los contenedores de Docker y ejecutándolos. Al finalizar, se mostrará el siguiente mensaje dentro de la terminal:

```
webapp_1 | 5:02:52 AM - Building project '/tsconfig.json'...
webapp_1 |
webapp_1 |
webapp_1 | 5:02:52 AM - Found 0 errors. Watching for file changes.
```

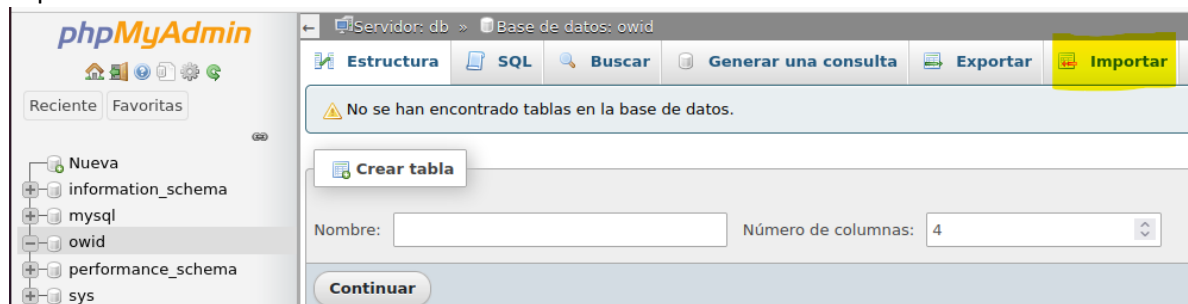
5. Una vez todos los contenedores se encuentren levantados, se deben importar los datos necesarios en la base de datos para que la plataforma funcione. (Esto solo se realiza la primera vez que se crean los contenedores)
 - a. Primero hay que ir al navegador e ingresar a localhost:<puerto de phpmyadmin>



- b. Luego hay que ingresar como root, y la contraseña de base de datos que se definió. Se observará que está creada la base de datos owid pero sin ninguna tabla.



- c. Dar clic en la base de datos, y luego seleccionar la opción “Importar” en el menú superior.



- d. Descargar el script de base de datos del repositorio de owid-grapher desde el link https://files.ourworldindata.org/owid_metadata.sql.gz
- e. Examinar los archivos, seleccionar el archivo recién descargado y dar clic en continuar para ejecutar el script.

Archivo a importar:

El archivo puede ser comprimido (gzip, bzip2, zip) o descomprimido.

Un archivo comprimido tiene que terminar en **.[formato].[compresión]**. Por

Buscar en su ordenador: Examinar... No se ha seleccionado ningún archivo

También puede arrastrar un archivo en cualquier página.

Conjunto de caracteres del archivo: utf-8

- f. Se debe deseleccionar la opción de importación parcial:

Partial import:

☐ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. (This might be a good way to import large files, however it can break transactions.)

Skip this number of queries (for SQL) starting from the first one: 0

- g. Esperar a que termine de ejecutar el archivo

✓ Importación ejecutada exitosamente, 566 consultas ejecutadas. (owid_metadata.sql.gz)

6. Cuando se haya finalizado la ejecución del código anterior y los datos se encuentren importados correctamente en la base de datos, se deberá abrir otra terminal, dentro de la carpeta del archivo Dockerfile y ejecutar el siguiente comando:

docker-compose exec webapp bash

De esta forma, se estará ingresando a la terminal dentro del contenedor del servicio "webapp" como usuario root.

7. Ejecutar la siguiente línea de comando:

yarn startAdminServer

Al finalizar, se mostrará el siguiente mensaje dentro de la terminal:

```
IntersectionObserver not available; interactive embeds won't load on this page
WORDPRESS_DB_NAME is not configured -- continuing without Wordpress DB
WORDPRESS_API_URL is not configured -- continuing without Wordpress API
owid-admin server started on http://localhost:3030
```

Es posible que falle la primera vez. Únicamente se debe reintentar.

8. Abrir otra terminal y ejecutar el mismo comando dentro de la carpeta del archivo Dockerfile:

docker-compose exec webapp bash

9. Ejecutar la siguiente línea de comando:

yarn startWebpackServer

Debe aparecer el siguiente mensaje:

```
root@lfarfan-VirtualBox:/# yarn startWebpackServer
yarn run v1.22.5
$ webpack-dev-server --no-live-reload --config ./itsJustJavascript/webpack.config.js
i ｢wds｣: Project is running at http://localhost:8090/
i ｢wds｣: webpack output is served from /
i ｢wds｣: Content not from webpack is served from public
```

Luego esperar y al finalizar, se mostrará el siguiente mensaje dentro de la terminal:

```
ini-css-extract-plugin} [built]
[../node_modules/css-loader/dist/cjs.js?url=fa
!/site/owid.scss 336 KiB {mini-css-extract-plugin}
[../node_modules/css-loader/dist/cjs.js?url=fa
s-extract-plugin} [built]
[../node_modules/css-loader/dist/cjs.js?url=fa
-css-extract-plugin} [built]
[../node_modules/css-loader/dist/runtime/api.j
i ｢wdm｣: Compiled successfully.
```

Al finalizar estos pasos, se podrá acceder a la plataforma en localhost:3030/admin.



owid-admin

Email address

admin@example.com

Password

.....

Having trouble logging in? Go to [#tech-issues](#).

Login

Se puede ingresar al portal con el usuario: **admin@example.com** y la contraseña **admin**

Pasos siguientes

1. **Modificar script inicio.sh para hacer el proceso automático:**
 - a. Ejecutar el comando ***docker ps*** para obtener el nombre completo del contenedor “webapp”. (**Nota: esto se debe hacer con los contenedores corriendo. De lo contrario se puede hacer con el comando Docker ps -a, en el que se mostrarán absolutamente todos los contenedores.**)

```
root@lfarfan-VirtualBox: /home/lfarfan/Esritorio/Proyecto_INCYT# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS
776aed88416   proyectoincyt_webapp "docker-entrypoint.s..." 12 days ago   Up 7 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp
1eacad78b6ff   phpmyadmin/phpmyadmin "docker-entrypoint.s..." 12 days ago   Up 8 seconds   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
ea3b39a5f25a   mysql               "docker-entrypoint.s..." 12 days ago   Up 12 seconds
```

- b. En el ejemplo anterior, el nombre es “proyectoincyt_webapp_1”. Este nombre copiarlo y pegarlo en el archivo “iniciar.sh” en los lugares donde encuentre “NOMBRE_CONTENEDOR”. (**Nota: El nombre del contenedor depende del nombre de la carpeta donde se encuentre. Si se mueve de carpeta o se crean de nuevo los contenedores, el nombre cambiará**)

```
$_ iniciar.sh
1  #!/bin/sh
2  gnome-terminal -e "sudo docker-compose start"
3  read -p "Presione enter para continuar"
4  gnome-terminal -e "docker exec -it proyectoincyt_webapp_1 yarn startAdminServer"
5  read -p "Presione enter para continuar"
6  gnome-terminal -e "docker exec -it proyectoincyt_webapp_1 yarn startWebpackServer"
```

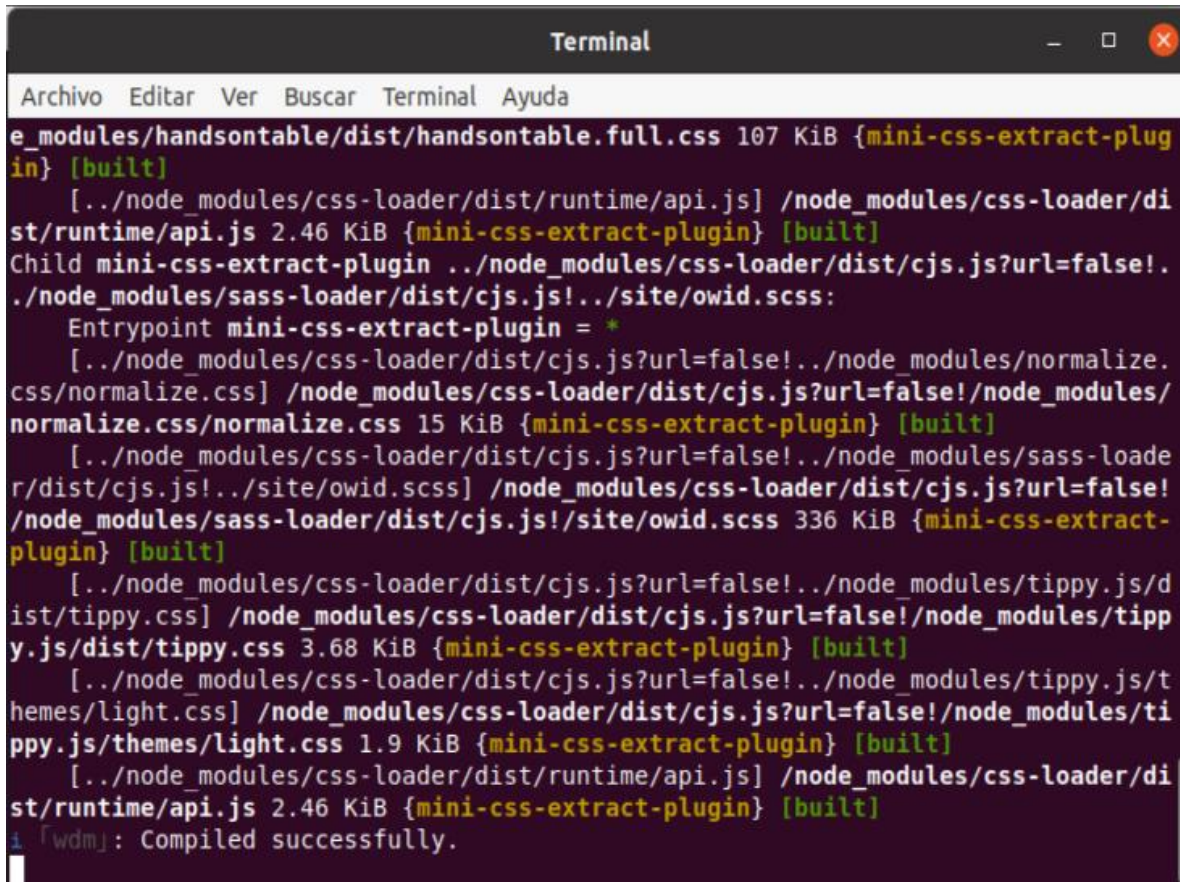
- c. Modificar el archivo “iniciar.sh” para volverlo ejecutable con el comando “***sudo chmod +x inicio.sh***”.
2. Una vez realizado todo esto se podrá ejecutar el comando “***sudo bash iniciar.sh***” y realizar los siguientes pasos:
 - a. Esperar a que los contenedores terminen de iniciar. Automáticamente se cerrará la venta con la terminal.
 - b. Presionar la tecla “***Enter***” para continuar.
 - c. Esperar a que inicie el AdminServer y luego se puede cerrar la ventana. (**Nota: no se cerrará automáticamente la ventana porque es comando que seguirá corriendo**)

El indicador de que terminó de iniciarse es el siguiente:

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
yarn run v1.22.5
$ node ./itsJustJavascript/adminSiteServer/app.js
IntersectionObserver not available; interactive embeds won't load on this page
WORDPRESS_DB_NAME is not configured -- continuing without Wordpress DB
WORDPRESS_API_URL is not configured -- continuing without Wordpress API
owid-admin server started on http://localhost:3030
```

- d. Una vez cerrada la ventana anterior, presionar de nuevo la tecla “Enter” para continuar.
- e. Por último, esperar a que termine de iniciarse el WebpackServer y luego se puede cerrar la ventana.

El indicador de que terminó de iniciarse es el siguiente:



```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
e_modules/handsontable/dist/handsontable.full.css 107 KiB {mini-css-extract-plugin} [built]
[../node_modules/css-loader/dist/runtime/api.js] /node_modules/css-loader/dist/runtime/api.js 2.46 KiB {mini-css-extract-plugin} [built]
Child mini-css-extract-plugin ../node_modules/css-loader/dist/cjs.js?url=false!../node_modules/sass-loader/dist/cjs.js!../site/owid.scss:
Entrypoint mini-css-extract-plugin = *
[../node_modules/css-loader/dist/cjs.js?url=false!../node_modules/normalize.css/normalize.css] /node_modules/css-loader/dist/cjs.js?url=false!../node_modules/normalize.css/normalize.css 15 KiB {mini-css-extract-plugin} [built]
[../node_modules/css-loader/dist/cjs.js?url=false!../node_modules/sass-loader/dist/cjs.js!../site/owid.scss] /node_modules/css-loader/dist/cjs.js?url=false!../node_modules/sass-loader/dist/cjs.js!../site/owid.scss 336 KiB {mini-css-extract-plugin} [built]
[../node_modules/css-loader/dist/cjs.js?url=false!../node_modules/tippy.js/dist/tippy.css] /node_modules/css-loader/dist/cjs.js?url=false!../node_modules/tippy.js/dist/tippy.css 3.68 KiB {mini-css-extract-plugin} [built]
[../node_modules/css-loader/dist/cjs.js?url=false!../node_modules/tippy.js/themes/light.css] /node_modules/css-loader/dist/cjs.js?url=false!../node_modules/tippy.js/themes/light.css 1.9 KiB {mini-css-extract-plugin} [built]
[../node_modules/css-loader/dist/runtime/api.js] /node_modules/css-loader/dist/runtime/api.js 2.46 KiB {mini-css-extract-plugin} [built]
[wdm]: Compiled successfully.

```

- f. Una vez completado todo el proceso se podrá acceder al portal de owid-grapher.
3. Para detener los contenedores se debe correr el siguiente comando:
 - a. ***docker-compose stop***
 - b. Si se utiliza ***docker-compose down***, se eliminarán los contenedores y se tendrán que volver a construir.
4. Los datos en la base de datos se quedarán almacenados, en los volúmenes creados por el servicio “db”.

Errores comunes

Cannot find module

```
yarn run v1.22.11
$ node ./itsJustJavascript/adminSiteServer/app.js
internal/modules/cjs/loader.js:818
  throw err;
  ^

Error: Cannot find module './itsJustJavascript/adminSiteServer/app.js'
    at Function.Module._resolveFilename (internal/modules/cjs/loader.js:815:15)
    at Function.Module._load (internal/modules/cjs/loader.js:667:27)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:60:1)
    at internal/main/run_main_module.js:17:47 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}
error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
```

El error anterior puede aparecer al ejecutar el comando “yarn startAdminServer”. Esto se debe a que el comando “Docker-compose up” no ha finalizado de ejecutarse. Solo se debe esperar y ejecutarlo nuevamente.

Error en ejecución de yarn

Algunas veces al ejecutar el yarn para obtener las dependencias, se producen errores por problemas como la conexión. En este caso, se debe ejecutar nuevamente el código.

Espacio insuficiente

Si aparece un mensaje mencionando que no se cuenta con espacio suficiente, se debe limpiar las imágenes no utilizadas de Docker y volver a intentar.

Código de generación de contraseña cifrada

La siguiente descripción corresponde al código del proyecto **bcrypt-project**, realizado especialmente para cifrar una contraseña con el fin de poder modificarla en la base de datos posteriormente. El código fue realizado por el equipo y posee las siguientes partes:

Librerías

```
const bcrypt = require("bcrypt");
const readline = require('readline');
var Writable = require('stream').Writable;
```

Se incluyen las siguientes librerías:

- Bcrypt: librería de cifrado por medio de algoritmo bcrypt.
- Readline: librería para la lectura de ingreso de datos en consola.
- Writable: librería para ocultar el ingreso de la contraseña por medio del carácter "*".

Variables

```
algorithm = "bcrypt"
iterations = 12
```

Se definen las variables del algoritmo utilizado y las iteraciones que se utilizan en el cifrado.

Métodos

hashIt()

```
async function hashIt(password) {
  const salt = await bcrypt.genSalt(12);
  const hashed = await bcrypt.hash(password, salt);
  console.log(`\n ${this.algorithm}${hashed}`)
}
```

El método hashIt() recibe la contraseña ingresada en consola. Posteriormente, produce la variable salt con las 12 iteraciones necesarias. Luego, cifra la contraseña por medio del algoritmo bcrypt. Por último, imprime en consola la contraseña cifrada agregándole el algoritmo utilizado.

password()

```
function password() {
  const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
  });

  rl.stdoutMuted = true;

  rl.question('Ingrese la contraseña que desea cifrar: ', (answer) => {
    hashIt(answer);
    rl.close();
  });
}
```

La primera parte del método indica la creación de la interfaz para el ingreso de datos a la consola junto a la opción de ocultar el ingreso de los datos activada. Posteriormente, se imprime la

instrucción en consola para enviar la respuesta al método de `hashIt()` y cerrar el ingreso de datos en la terminal.

```
rl._writeToOutput = function _writeToOutput(stringToWrite) {  
    if (rl.stdoutMuted)  
        rl.output.write("*");  
    else  
        rl.output.write(stringToWrite);  
};  
}
```

La segunda parte del método reemplaza los datos ingresados por el carácter “*” si la opción `stdoutMuted` se encuentra activada.

Ejecución del código

```
password();
```

La última parte inicia la ejecución del código por medio del método de impresión y captura de datos.