CEIoT - FIUBA

TP final Desarrollo Aplicaciones Web

2da Cohorte 2020

Docentes

Agustin Bassi
(jagustinbassi@gmail.com)

Brian Ducca
(brian.ducca@gmail.com)

Ernesto Gigliotti
(ernestogigliotti@gmail.com)

Documento con las actividades y pasos necesarios para realizar y entregar el Trabajo Final de la Asignatura Desarrollo de Aplicaciones Web.

Introducción al documento

Con este proyecto concluye la materia de Desarrollo de aplicaciones Web. Los criterios de evaluación que se tendrán en cuenta para la elaboración de la nota final son:

- Documentación y estructuración del proyecto.
- Claridad de programación, documentación de código.
- Resolución de las consignas aplicando los conocimientos vistos en clase.

Antes de comenzar a realizar las actividades requeridas, es conveniente realizar una lectura total del documento, a fin de tener contexto y consideraciones durante todo el desarrollo del proyecto.

> *Será conveniente ir versionando (haciendo commits) el proyecto a medida que se van realizando cambios en el mismo y no muchos cambios en un commit solo.*

Objetivos

El objetivo de este trabajo es finalizar la asignatura Desarrollo de Aplicaciones Web. Para lograrlo, será necesario cumplir con los siguientes objetivos:

- Crear un nuevo repositorio para el trabajo final de la materia.
- Crear un archivo README dentro del repositorio del proyecto con toda la información necesaria de contexto, así como también para poder correrlo.
- Agregar a la aplicación las funcionalidades descritas en el punto 5.
- Correr el proyecto desarrollado con la herramienta Docker Compose.
- Entregar el repositorio con todo lo necesario para correr la aplicación con un solo comando.

Actividades a realizar

A continuación se describen las actividades a realizar para completar el proyecto.

1. Crear estructura para el proyecto

El primer paso es crear un directorio para el proyecto, con el siguiente comando.

```
mkdir daw_tp_final && cd daw_tp_final
```

Agregar un archivo README.md que contendrá la documentación técnica del proyecto.

```
touch README.md
```

Agregar al archivo README.md el siguiente contenido que servirá como plantilla.

````
Autor: Nombre - 202X
# Introduccion

El proyecto es ...

# Correr la aplicación

Para correr la aplicación es necesario ejecutar el siguiente comando:

```sh
command_to_run
```

# Contribuir

Para contribuir realizar un pull request con las sugerencias.

# Licencia

GPL
````

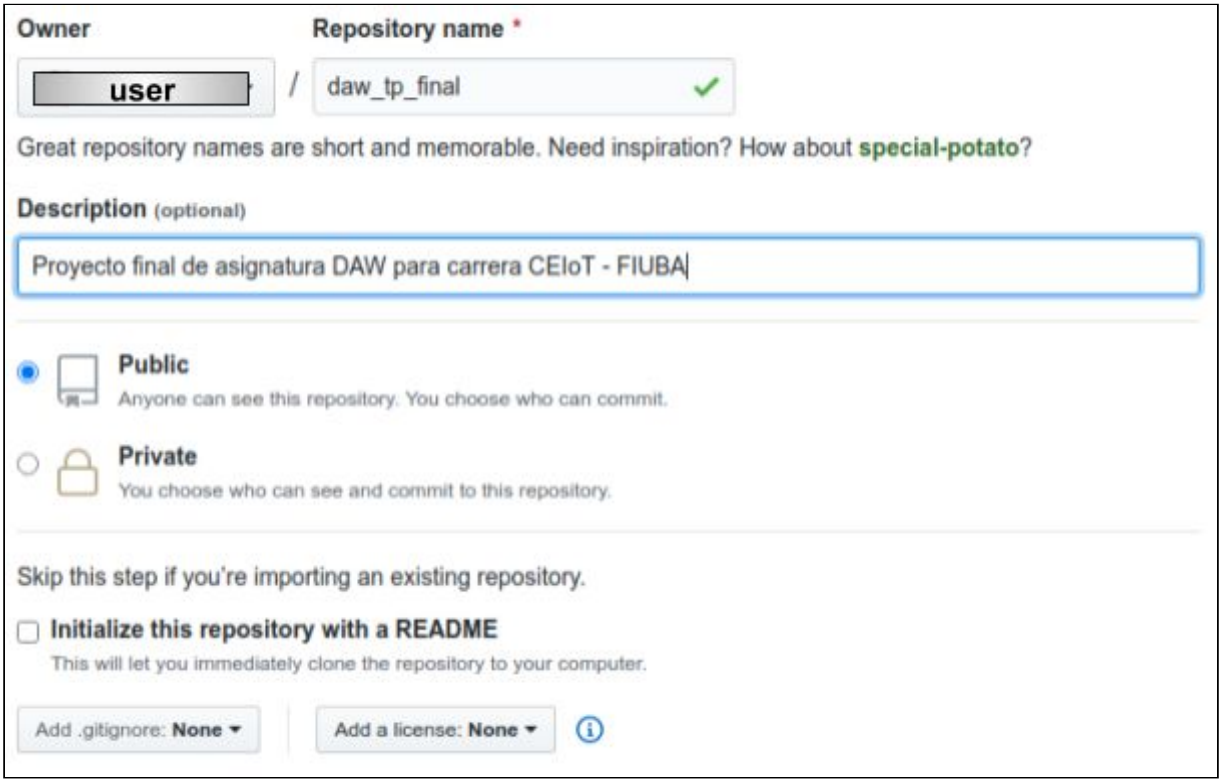
> **TIP**: En Visual Studio Code haciendo click derecho sobre la pestaña del archivo y seleccionando la opción Show Preview se puede visualizar el contenido formateado del archivo README.md.

2. Crear el repositorio del proyecto

Antes de continuar con las actividades será necesario crear un repositorio para el proyecto, agregar el archivo README.md con la descripción hasta el momento y guardarlo como "Initial commit", con los siguientes comandos.

```
git init && git add .
git commit -m "Initial commit"
```

Crear un repositorio para la aplicación en Github llamado **"daw_tp_final"**, agregarle una descripción al mismo, dejarlo como Public e inicializarlo sin README ni licencia. En la siguiente imagen se puede ver un ejemplo de cómo se debe crear el proyecto.



Una vez creado el repositorio, asociar el repositorio creado en Github al repositorio local previamente creado y pushear los commits realizados (hasta ahora solo el README.md).

```
git remote add origin https://github.com/user/daw_tp_final.git
git push -u origin master
```

3. Agregar la estructura existente al proyecto

Actualizar/descargar el repositorio de la materia haciendo un pull con las últimas actualizaciones con el siguiente comando.

```
git pull https://github.com/ernesto-g/daw.git
```

> El comando anterior debe ejecutarse fuera del proyecto daw_tp_final. Se debe realizar en la carpeta donde se trabajó con el repositorio a lo largo de las clases.

El último ejercicio realizado en clases se encuentra dentro de la carpeta **ejercicios/Ejercicios_NodeJs/Ejercicio_12.** Copiar la estructura del último ejercicio dentro de la carpeta del proyecto. La estructura debería quedar de la siguiente manera.

```
├─── css/
├─── db/
├─── images/
├─── js/
├─── src/
├─── ws/
├─── index.html
├─── run_phpadmin.sh
├─── serve_node_app_net.sh
├─── start_mysql.sh
├─── README.md
```

Una vez que el proyecto existente esté dentro ponerlo a correr de la misma manera como en los casos anteriores.

4. Correr el proyecto

A esta altura, dentro de la carpeta del proyecto se encuentra todo lo necesario para correr la aplicación, es decir el código fuente de la aplicación así como también los scripts que levantarán los contenedores de Docker necesarios.

El primer paso será detener todos los contenedores corriendo en la máquina.

```
docker stop $(docker ps -a -q)
```

El siguiente paso será chequear la red de Docker que se utilizará para conectar los contenedores entre sí con el siguiente comando.

```
docker network ls | grep mysql-net
```

Si el comando anterior no arroja info, será necesario crearla con el siguiente comando.

```
docker network create --driver bridge mysql-net
```

Con la red creada será necesario ejecutar el contenedor con la base de datos. Para eso, dentro del directorio raíz del proyecto ejecutar el siguiente comando, pasándole como argumento la red a utilizar y el directorio donde se encuentra la base de datos.

```
./start_mysql.sh mysql-net "$PWD"/db
```

A continuación correr el gestor de base de datos PHPMyAdmin con el siguiente comando, especificando la red de Docker, el nombre del servidor de base de datos y el puerto.

```
./run_phpadmin.sh mysql-net mysql-server 8085
```

Chequear que PHPMyAdmin está funcionando en http://localhost:8085.

Finalmente ejecutar el servidor de NodeJS con la aplicación creada hasta el momento pasando como argumento el directorio actual, especificando que el entry point de la aplicación será ws/index.js, corriendo la aplicación en el puerto localhost 8000 y sobre la red mysql-net, donde se encuentran la base de datos y el administrador de la base de datos.

```
./serve_node_app_net.sh "$PWD" ws/index.js 8000 mysql-net
```

Chequear que la aplicación esté corriendo en http://localhost:8000.

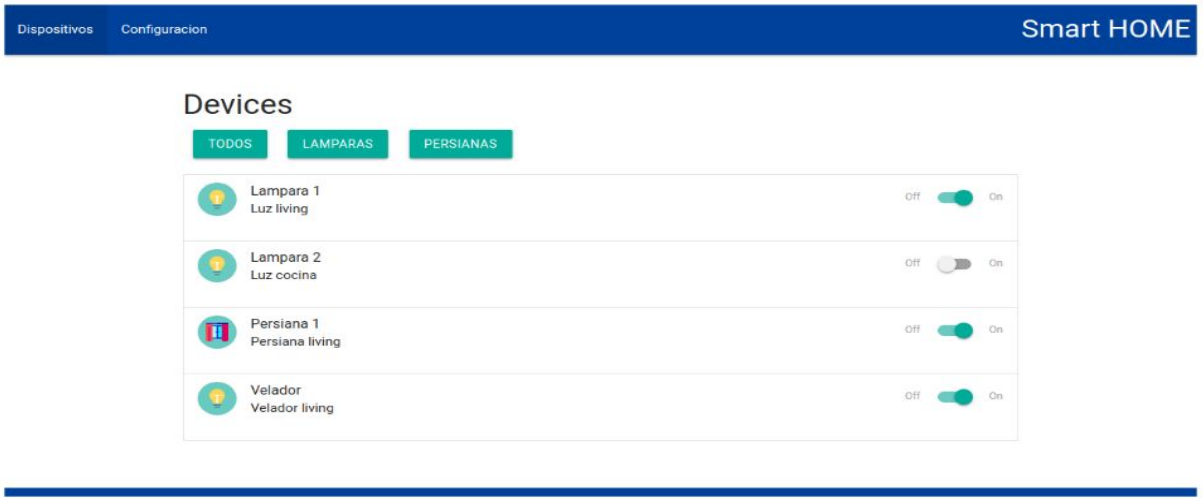
5. Agregar nuevas funcionalidades a la aplicación

Una vez que la aplicación está corriendo, se deberá tomar el sistema “smart home” desarrollado durante la cursada, y agregarle las siguientes características:

1) El sitio web deberá tener un filtro por tipo de dispositivo, para poder listar:

- Solo las lámparas
- Solo las persianas
- Ambos

Para ello se deberán agregar 3 botones en la parte superior de la lista, mediante los cuales se deberá consultar al servicio web que devuelve la lista de dispositivos pero indicando el filtro apropiado según el botón seleccionado.



2) Se deberá modificar el servicio web que devuelve la lista agregando un parámetro “filter” en la URL de la siguiente manera:

`/ws/devices?filter=X`

Siendo X:

- 0 para no filtrar por tipo
- 1 para filtrar el tipo “lámpara”
- 2 para filtrar el tipo “persiana”

De modo que al realizar un request de tipo GET se devuelva una lista filtrada.

3) Al recibir el JSON con la lista de devices, se deberá mostrar en el sitio web.

4) En el backend, se deberá cambiar la consulta SQL según el filtro que se recibió.

6. Entender cómo correr cada contenedor

El siguiente paso será entender los argumentos, variables de entorno, volúmenes, puertos, etc. necesarios para correr cada uno de los contenedores involucrados en el proyecto.

A partir de los scripts **start_myql_server.sh**, **run_phpmyadmin.sh** y **serve_node_app_net.sh**, se pueden establecer los datos requeridos. Si es necesario remitirse al documento de instalación de herramientas con Docker donde se encuentra más información de contexto para correr cada contenedor.

Extraído del script **start_myql_server.sh** se necesitan los siguientes argumentos.

```
docker run --rm \
--detach \
--name $CONTAINER_NAME \
--network $NETWORK_NAME \
--env MYSQL_ROOT_PASSWORD=userpass \
--volume $DATABASE_DIR/dumps:/docker-entrypoint-initdb.d \
--volume $DATABASE_DIR/data:/var/lib/mysql \
mysql:5.7
```

Extraído del script **run_phpmyadmin.sh** se necesitan los siguientes argumentos.

```
docker run --rm \
--detach \
--name $CONTAINER_NAME \
--network $NETWORK_NAME \
--env PMA_HOST=$MYSQL_HOST \
--publish $HOST_PORT:80 \
phpmyadmin/phpmyadmin
```

Extraído del script **run_phpmyadmin.sh** se necesitan los siguientes argumentos.

```
docker run  --rm  \
--interactive \
--name $CONTAINER_NAME \
--network $HOST_NET \
--publish $HOST_PORT:$CONTAINER_PORT \
--volume $HOST_APP_DIR:$CONTAINER_APP_DIR \
abassi/nodejs-server:10.0-dev \
nodemon $CONTAINER_APP_DIR/$ENTRY_POINT
```

Una vez obtenida la información para ejecutar cada contenedor, crear un directorio llamado doc/ y dentro de este un archivo llamado notes.txt; y por cada contenedor anotar los argumentos necesarios para correr, a modo de ejemplo, copiar el siguiente contenido como plantilla para anotar los argumentos de ejecución de cada contenedor.

- MySQL.
  - Nombre de la imagen.
  - Nombre contenedor.
  - Network.
  - Variables de entorno.
  - Puertos.
  - Volumenes.
- Node App.
  - Nombre de la imagen.
  - Nombre contenedor.
  - Network.
  - Variables de entorno.
  - Puertos.
  - Volumenes.
- PHPMyAdmin.
  - Nombre de la imagen.
  - Nombre contenedor.
  - Network.
  - Variables de entorno.
  - Puertos.
  - Volumenes.

Siempre es útil tener este tipo de archivo para ir realizando anotaciones rápidas, comandos útiles que se utilicen en el proyecto, y demás, con el fin de no tipearlos a mano cada vez que se necesitan.

> Recordar que esta información también puede volcarse al README.md en caso de considerarlo necesario.

7. Correr el proyecto con Docker Compose

Con la información recolectada hasta el momento, será posible crear y completar el archivo docker-compose.yml. Comenzar creando el archivo en la raìz del proyecto con el siguiente comando.

```
touch docker-compose.yml
```

Y agregar el siguiente contenido como plantilla.

```
version: '3'
services:

 service:
   image:
   hostname:
   container_name:
   restart: always
   environment:
     ENV_VAR: value
   volumes:
     - host_dir: container_dir
   networks:
     - net-web-app
   depends_on:
     - container_name
   ports:
     - "host_port:container_port"

networks:
  net-web-app:
    driver: bridge
```

Dentro del archivo docker-compose.yml realizar las siguientes tareas.

- Crear tres servicios llamados mysql-server, node-app y phpmyadmin.
- Completar las imágenes de cada uno de los servicios.
- Poner el hostname y container_name igual al nombre del servicio (ej: node-app).
- Si el contenedor no necesita compartir puertos, volúmenes, borrarlos de cada servicio.
- Agregar las variables de entorno necesarias para cada contenedor.
- En caso que el contenedor no dependa de nadie, borrar depends_on.

> TIP: En el documento de Aplicaciones con Docker Compose listas para utilizar hay información útil y detallada de cómo crear aplicaciones con Docker Compose a través de los contenedores.

Una vez que el archivo esté completo, y se haya revisado detenidamente, iniciar toda la aplicación con el comando a continuación.

```
docker-compose up
```

Si la ejecución no generó ningún inconveniente, en http://localhost:8000 se debería ver la aplicación corriendo y en http://localhost:8085 se debería poder acceder a PHPMyAdmin.

Para detener toda la aplicación ejecutar el comando siguiente.

```
docker-compose down
```

8. Acondicionar el proyecto para su distribución

En la fase final del trabajo final será necesario realizar tareas necesarias para su distribución y documentación. El primero de estos será completar de manera detallada toda la información con documentación técnica del proyecto en el archivo README.md.

> Principalmente se debe explicar qué hace la aplicación y los pasos necesarios para correrla. Pensar que lo más importante de un proyecto para que pueda ser utilizado por otras personas es crear una buena documentación. Ejemplos de buena documentación de README son: proyecto1, proyecto2, proyecto2.

Una vez que el proyecto se considere listo, hacer commits de los cambios y pushearlos al repositorio remoto en Github configurado en el punto 2.

9. Enviar el proyecto

Una vez realizados todos los pasos, enviar por mail el link de la siguiente manera.

Destinatarios	jagustinbassi@gmail.com, ernestogigliotti@gmail.com , brian.ducca@gmail.com
Asunto	CEIOT - DAW - TP Final - Nombre y apellido
Contenido	Hola, Les envío el link del proyecto final de DAW: github.com/usuario/repositorio Saludos! Nombre y apellido.

IMPORTANTE: Recordar que para ejecutar (desde el lado de los evaluadores), el único comando necesario para correr el proyecto debe ser **docker-compose up.** Trabajar la solución para que únicamente se deba ejecutar ese comando.

> **TIP:** Desde otro directorio clonar el repositorio e intentar ejecutar la aplicación con el comando antes mencionado para probar que funcione correctamente antes de realizar la entrega.

¡Felicitaciones!

Si llegaste hasta este punto es porque terminaste el proyecto. Lo evaluaremos lo más rápido posible, y tu nota será reflejada en la planilla de notas de la materia.

Esperamos que la materia haya estado a la altura de tus expectativas y hayas disfrutado el aprendizaje adquirido.

Es muy importante para nosotros que nos des feedback del curso, qué te parecieron las clases, el material, qué recomendaciones nos darías y todo lo que creas relevante, ya que estamos en mejora continua. Por favor completa la encuesta del curso que te llegará por email al finalizar la materia.

Seguiremos en contacto y cualquier consulta no dudes en escribirnos.

Saludos!
Agustin, Ernesto y Brian.