Servicios WEB

localhost

8080

localhost:8080
localhost:8080/vistas/protegida_admin.jsp
localhost:8080/vistas/protegida_usuario.jsp
...

8000

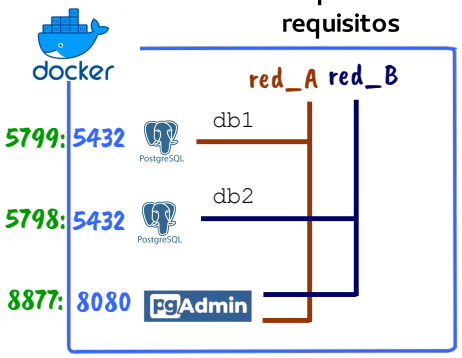
localhost:8000
localhost:8000/apartado1
...

8888

localhost:8888
localhost:8888/login
localhost:8888/registro
...

CONEXIÓN CON BASE DE DATOS DEL SERVIDOR FLASK:
`HOST: localhost`
`PORT: 5798`

1-2 redes dependiendo de los requisitos



PODEMOS CONECTARNOS A LAS BASES DE DATOS DESDE PGADMIN (DOCKER) ACCEDIENDO DESDE UN NAVEGADOR LOCAL:
`http://localhost:8877`
** Es necesario que pgadmin se encuentre en ambas redes, estableciendo las siguientes configuraciones:

9669

PODEMOS CONECTARNOS A LAS BASES DE DATOS DESDE PGADMIN (EXTERNO) ACCEDIENDO DESDE UN NAVEGADOR LOCAL:
`http://localhost:9669`

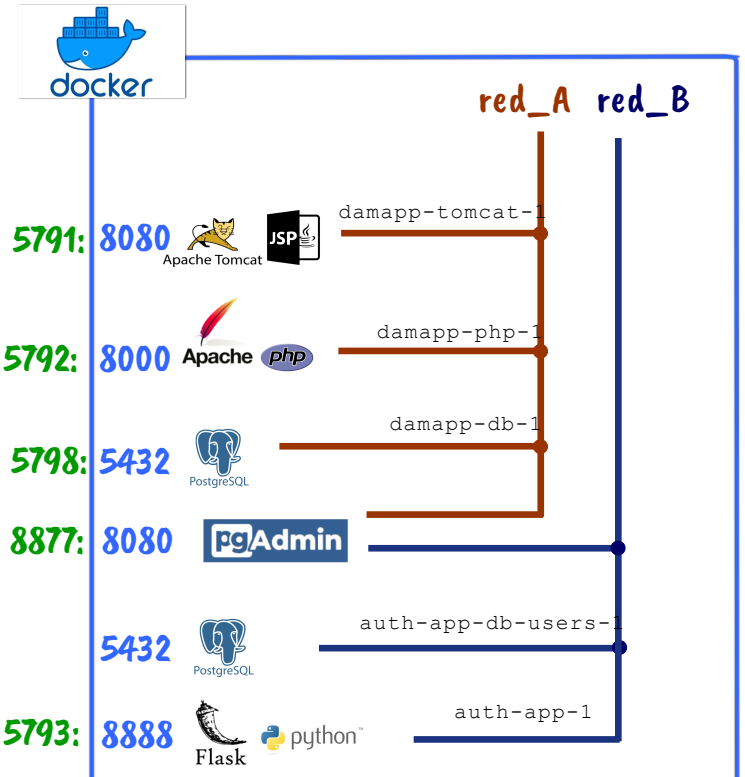
```
BASE DE DATOS 1
HOST: localhost
PORT: 5797

BASE DE DATOS 2
HOST: localhost
PORT:5798
```

localhost

Servicios WEB

DESDE LOCALHOST

Para acceder a cualquier recurso desde fuera de docker, podemos usar sus puertos expuestos. Por ejemplo, puedo acceder desde un cliente pgAdmin a la base de datos `damapp-db-1` expuesta en el puerto 5798, pero no podría acceder a la base de datos de `auth-app-db-users-1`, ya que no tiene un puerto expuesto.



DENTRO DE DOCKER

Para acceder de un recurso a otro podemos usar el nombre del host y el puerto interno, siempre y cuando se encuentren en la misma red

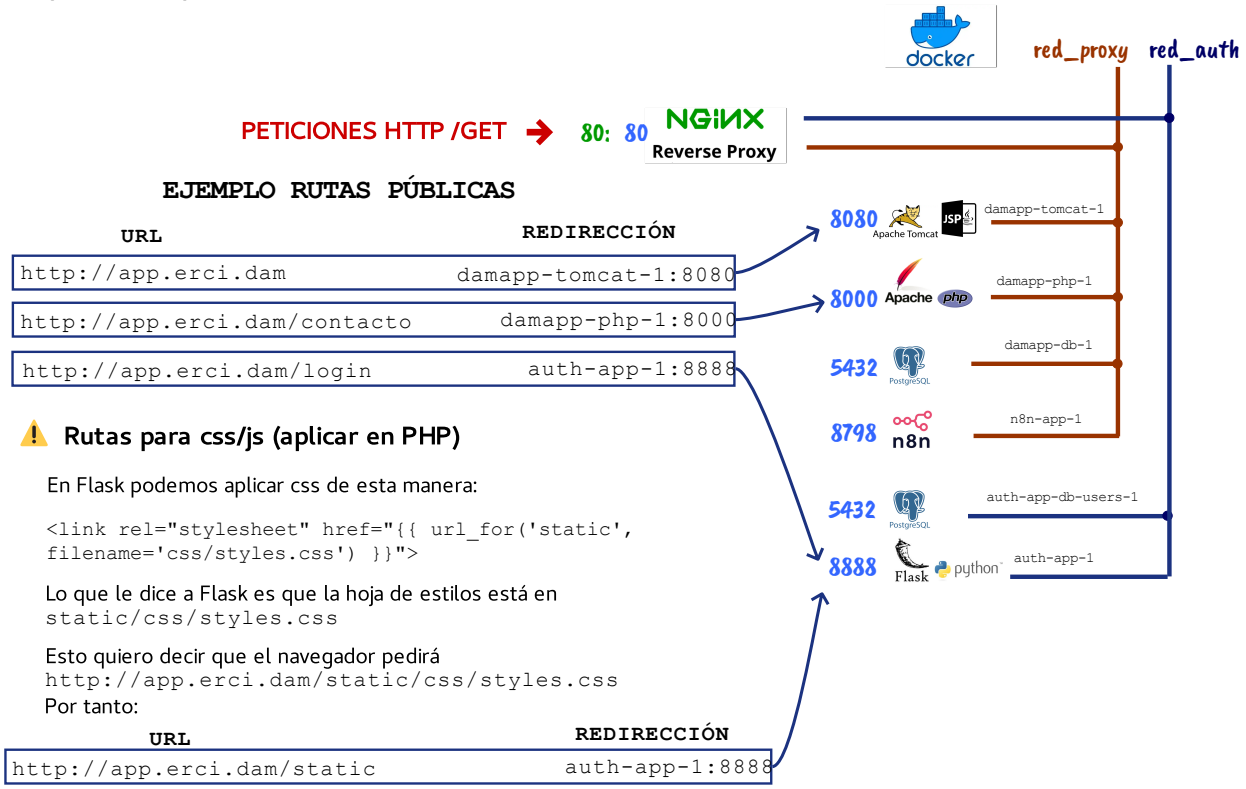
Por ejemplo: `auth-app-1` no puede acceder a la base de datos `damapp-db-1:5432`

Servicios WEB

REVERSE PROXY

Un **reverse proxy** es un servidor que se coloca delante de varias aplicaciones y decide a cuál debe enviar cada petición.

En el siguiente ejemplo lanzamos varias aplicaciones dentro de **Docker**. El único puerto publicado hacia el exterior es el **puerto 80** —y el **81** para administración—. Usamos la URL `http://app.erci.dam`, que ha sido configurada previamente en el DNS local o, para esta práctica, en el archivo `/etc/hosts`.



⚠️ Rutas para css/js (aplicar en PHP)

En Flask podemos aplicar css de esta manera:

```
<link rel="stylesheet" href="{{ url_for('static',
filename='css/styles.css') }}">
```

Lo que le dice a Flask es que la hoja de estilos está en `static/css/styles.css`

Esto quiero decir que el navegador pedirá `http://app.erci.dam/static/css/styles.css`
Por tanto:

URL	REDIRECCIÓN
`http://app.erci.dam/static`	`auth-app-1:8888`

Servicios WEB

REVERSE PROXY

EJEMPLO RUTAS PRIVADAS

5 Endpoint para validación desde reverse proxy.

Respuestas:
- 204: permitido.
- 401: no autenticado.
- 403: autenticado, pero sin rol requerido.

(mirar código facilitado en prácticas)

3

```
location = /_flask_auth_admin {
    internal;

    proxy_pass http://flask-auth-1:8888/auth/verify;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";
    proxy_set_header Cookie $http_cookie;
    proxy_set_header Authorization $http_authorization;
    proxy_set_header X-Required-Roles "admin";
    proxy_set_header X-Original-URI $request_uri;
    proxy_set_header X-Original-Host $host;
    proxy_set_header X-Original-Method $request_method;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location = /_flask_auth_cliente {
    internal;

    proxy_pass http://flask-auth-1:8888/auth/verify;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";
    proxy_set_header Cookie $http_cookie;
    proxy_set_header Authorization $http_authorization;
    proxy_set_header X-Required-Roles "cliente";
    proxy_set_header X-Original-URI $request_uri;
    proxy_set_header X-Original-Host $host;
    proxy_set_header X-Original-Method $request_method;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location @login_redirect {
    return 302 /login?next=$request_uri;
}

location @forbidden {
    return 403;
}
```

4

CONFIGURACIÓN GENERAL DEL PROXY

1 El usuario entra en: http://app.erci.dam/protegida

2 Nginx ve que esa ruta tiene: auth_request /_flask_auth_admin;

3 Antes de enviar la petición a /protegida, Nginx hace una consulta interna a: /_flask_auth_admin

4 Esa location interna llama a Flask: http://flask-auth-1:8888/auth/verify

5 Flask revisa si el usuario está autenticado y si tiene rol admin.

6 Si Flask responde 204: Usuario permitido