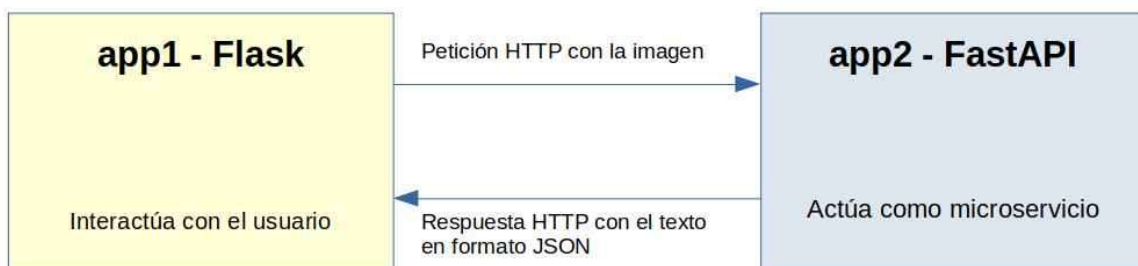


UD3.PR1 - Actividad

En esta actividad se trata de crear dos aplicaciones web, desglosando la aplicación en Flask (creada de forma guiada) en dos aplicaciones. El esquema quedaría del siguiente modo:



Primero tendrás que familiarizarte con el framework FastAPI. En los siguientes enlaces puedes encontrar información para instalar las dependencias y saber lo básico:

<https://fastapi.tiangolo.com/>

<https://fastapi.tiangolo.com/tutorial/request-files/>

A continuación, se describen las especificaciones de cada aplicación:

APP1 - Flask

- Conserva el HTML de la app original, incluyendo la subida de imágenes.
- No contiene la parte de extracción de texto de la imagen.
- Conserva la conversión a audio y la reproducción de sonido.
- Ejecutada desde un entorno virtual de Python, en el puerto 5000.
- Realiza peticiones HTTP con la librería requests a la APP2.
- Convierte el JSON recibido y lo representa en results.html.

APP2 - FastAPI

- Expone solo un endpoint que recibirá una petición HTTP con una imagen, y devolverá el texto extraído en formato JSON.
- La aplicación estará dentro de un contenedor Docker, que se levantará en el puerto 5050.

Se pide:

1. Desarrollar/adaptar correctamente las dos aplicaciones, según los requisitos que se han establecido:

app1: **1 punto**

app2: **2 puntos**

2. Subida de todo el código ordenado a un repositorio de Github. Has de proporcionar el enlace a dicho repositorio y documentar los comandos que has ejecutado (**1 punto**)

3. Subida de la imagen de la app2 a Docker Hub. Has de proporcionar el enlace al repositorio y documentar los comandos que has ejecutado (**1 punto**).

4. Creación de un volumen Docker para almacenar las imágenes que gestiona la app2 y no se pierdan cuando se elimine el contenedor. Documenta las decisiones tomadas y la configuración que has utilizado. (**1 punto**).

5. Crear un fichero de Docker Compose para que la app2 se comunique con el servidor de aplicaciones uvicorn, y éste a su vez con el servidor web Nginx. Te puedes basar en este [enlace](#) (**2 puntos**).

6. Documenta con capturas de pantalla y explicaciones que el sistema está funcionando según los requisitos (**2 puntos**).

Los puntos 4 y 5 no son imprescindibles para que el sistema funcione, se trata de puntos de profundización.

GitHub: *(Documentado con capturas de pantalla y explicaciones)*

He creado varias versiones con el código que iba probando.

Repositorio GitHub: <https://github.com/Zurichk/PracticalIA-FastApi-Flask-Docker.git>

```
git init
```

```
git add .
```

```
git commit -m "primera prueba"
```

```
git remote add origin https://github.com/Zurichk/PracticalIA-FastApi-Flask-Docker.git
```

```
git checkout -b master
```

```
git push -u origin master
```

```
zurich@Zur-Portatil MINGW64 ~/Documents/EntornosPython/Docker2 (master)
$ git push -u origin master
Enumerating objects: 4750, done.
Counting objects: 100% (4750/4750), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4684/4684), done.
Writing objects: 100% (4750/4750), 15.27 MiB | 2.29 MiB/s, done.
Total 4750 (delta 1650), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1650/1650), done.
To https://github.com/Zurichk/PracticalIA-FastApi-Flask-Docker.git
* [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

zurich@Zur-Portatil MINGW64 ~/Documents/EntornosPython/Docker2 (master)
```

Creando contenedor Docker y cargando app2 *(Documentado con capturas de pantalla y explicaciones)*

Creamos u obtenemos los requirements:

```
pip freeze > requirements.txt
```

Creamos el DockerFile

```
Dockerfile
1 FROM python:3.8
2
3 RUN mkdir /usr/src/app/
4 #RUN mkdir /usr/src/resultados/images/
5 WORKDIR /usr/src/app/
6
7 COPY ./requirements.txt /usr/src/app/requirements.txt
8
9 RUN pip install --no-cache-dir --upgrade -r /usr/src/app/requirements.txt
10
11 COPY ./code2 /usr/src/app/
12
13 #RUN apk update && apk add --no-cache python3-dev gcc libc-dev musl-dev linux-headers tesseract-ocr tesseract-ocr-data-spa
14 #RUN apk add -u zlib-dev jpeg-dev gcc musl-dev
15 RUN pip install --upgrade pip
16
17 CMD ["uvicorn", "app2:app", "--host", "0.0.0.0", "--port", "5050"]
18 #CMD uvicorn app2:app --host 0.0.0.0 --port 5050
19
20 #ENTRYPOINT ["uvicorn", "app2:app", "--host", "0.0.0.0", "--port", "5050"]
```

Creamos el docker-compose.yml

```
docker-compose.yml X
C: > Users > zuric > Documents > EntornosPython > Docker2
1 version: "3"
2 services:
3   web:
4     build: .
5     ports:
6       - 5050:5050
7     environment:
8       - DOCKER=yes
```

Y desde el entorno virtual: docker-compose up --build

```
Step 7/8 : RUN pip install --upgrade pip
--> Using cache
--> d41f3041c89e
Step 8/8 : CMD ["uvicorn", "app2:app", "--host", "0.0.0.0", "--port", "5050"]
--> Running in 07c258646a44
Removing intermediate container 07c258646a44
--> 2e6352aba532
Successfully built 2e6352aba532
Successfully tagged docker2_web:latest
Recreating docker2_web_1 ... done
Attaching to docker2_web_1
web_1 | INFO: Started server process [1]
web_1 | INFO: Waiting for application startup.
web_1 | INFO: Application startup complete.
web_1 | INFO: Uvicorn running on http://0.0.0.0:5050 (Press CTRL+C to quit)
web_1 | INFO: 192.168.99.1:51672 - "GET / HTTP/1.1" 200 OK
```

Y probamos que Docker funciona correctamente:

```
{ "Probando": "Contenedor Docker" }
```

Docker Hub:

Subir Imagen a Docker

Docker login

```
C:\Users\zuric\Documents\EntornosPython\Docker2>docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: zurichk
Password:
WARNING! Your password will be stored unencrypted in C:\Users\zuric\.docker\config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded

C:\Users\zuric\Documents\EntornosPython\Docker2>
```

Utiliza docker tag para generar una variante de tu imagen con ese nombre.

docker tag docker2_web zurichk/docker2_web:v1

```
C:\Users\zuric\Documents\EntornosPython\Docker2>docker tag docker2_web zurichk/docker2_web:v1
```

Y la subimos

docker push zurichk/docker2_web:v1

Comprobamos en dockerhub que la imagen esta subida

The screenshot shows the Docker Hub interface for the repository 'zurichk/docker2_web'. The page includes a search bar, navigation tabs (General, Tags, Builds, Collaborators, Webhooks, Settings), and a main content area. The 'General' tab is active, displaying repository details, tags, and scans. The repository is public and has one tag, 'v1', pushed 4 minutes ago. The page also features sections for 'Advanced Image Management', 'Docker commands', and 'Automated Builds'.

Link: https://hub.docker.com/r/zurichk/docker2_web

docker pull zurichk/docker2_web

Creación de un volumen Docker:

Comprobamos que no hay ningún volumen creado

`docker volume ls`

Consultamos las imágenes:

`docker image ls`

Y creamos el volumen

`docker run -d --name docker2 --mount source=mi-volumen,target=/app zurichk/docker2_web:v1`

Nuevamente con `docker volume ls` comprobamos que esta creado

```
C:\Users\zuric\Documents\EntornosPython\Docker2>docker volume ls
DRIVER          VOLUME NAME

C:\Users\zuric\Documents\EntornosPython\Docker2>docker image ls
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
docker2_web     latest      fc9f958a69fa   37 minutes ago  998MB
zurichk/docker2_web v1         fc9f958a69fa   37 minutes ago  998MB
<none>          <none>      2e6352aba532   5 hours ago    998MB
<none>          <none>      b428d5466d2a   5 hours ago    998MB
python          3.8         5c4350efb04f   9 days ago     912MB
training/webapp latest      6fae60ef3446   6 years ago    349MB

C:\Users\zuric\Documents\EntornosPython\Docker2>docker run -d --name docker2 --mount source=mi-volumen,target=/app zurichk/docker2_web:v1
ccf223b9b368ae6de77ff244e1c8dcffa42fdbe2591bdf0a466f3025b0e73f8d

C:\Users\zuric\Documents\EntornosPython\Docker2>docker volume ls
DRIVER          VOLUME NAME
local          mi-volumen
```

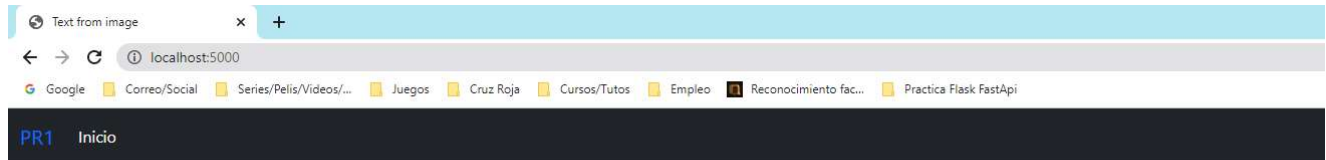
Con `docker volume inspect mi-volumen` lo inspeccionamos

```
C:\Users\zuric\Documents\EntornosPython\Docker2>docker volume inspect mi-volumen
[
  {
    "CreatedAt": "2022-03-28T01:04:44Z",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/mnt/sda1/var/lib/docker/volumes/mi-volumen/_data",
    "Name": "mi-volumen",
    "Options": null,
    "Scope": "local"
  }
]
```

Probamos las aplicaciones:

No he conseguido hacer funcionar las aplicaciones en Docker, ya que no he podido pasar la imagen de Flask a FastApi, he probado File, FileResponse, Streaming, etc y no he conseguido hacerlo, por lo que he optado por mover la imagen a un directorio y cogerlo de ahí, por lo menos para intentar avanzar en la práctica.

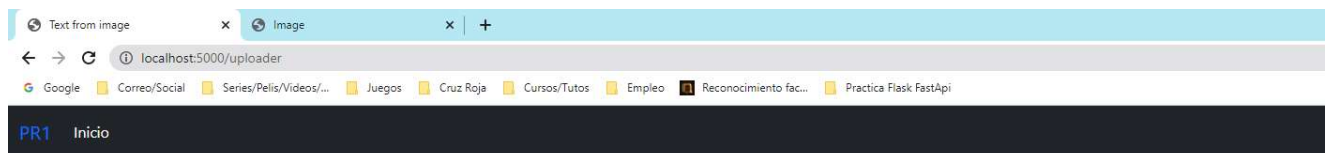
Vamos a <http://localhost:5000/> para cargar la imagen



Extracción de texto de imágenes

Ninguno archivo selec.

Al darle a Transformar cargamos la imagen, a la cual le cambiamos el nombre y el formato. 1.png



Extracción de texto de imágenes

```

('Imagen Subida Correctamente', 'Nombre: 1.png', 'Destino: ..\\resultados\\images')

-----

-----

Inicio
    
```

Automáticamente nos abre otra ventana con la imagen, y esta convertida a texto.



Imagen

Imagen: <PIL.PngImagePlugin.PngImageFile image mode=P size=241x210 at 0x24A70C88190>

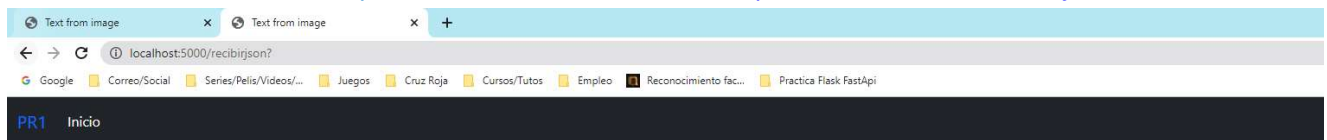
Nombre

Texto: Nombre

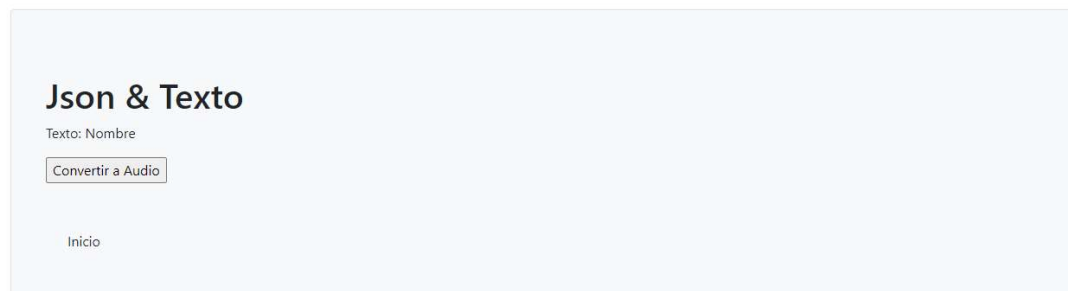
Json: "Nombre\n"

Enviar Json

Hacemos click en Enviar Json y nos redirecciona a de nuevo <http://localhost:5000/recibirjson>



Extracción de texto de imágenes



El cual esta desde la app1 recibiendo de app2 (url3 = <http://localhost:5050/jsontexto>)

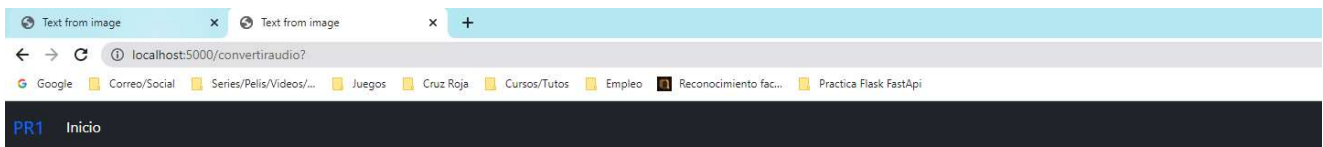
```
@app.route('/recibirjson')
def pasaratexto():
    jsontexto = get(url3).json()
    return render_template('results.html', jsontexto=jsontexto)

@app.route('/convertiraudio')
```

Hacemos click en convertir a Audio, el cual crea el archivo audio.mp3 y nos lo reproduce.

Video demostración del funcionamiento:

<https://drive.google.com/file/d/1Nx-viHGrGn9nOYtjGSI52mxNfQhI0JLV/view?usp=sharing>



Extracción de texto de imágenes



Finalmente, una breve explicación de la configuración o distribución del Entorno:

Compartir VISTA

Este equipo > Documentos > EntornosPython > Docker2

	Nombre	Fecha de modificación	Tipo	Tamaño
ido	.git	28/03/2022 2:41	Carpeta de archivos	
	code	28/03/2022 3:15	Carpeta de archivos	
s	code2	28/03/2022 1:57	Carpeta de archivos	
itos	docker2	14/03/2022 19:28	Carpeta de archivos	
rive (G:)	docker3	27/03/2022 21:52	Carpeta de archivos	
	Instrucciones	27/03/2022 20:52	Carpeta de archivos	
	resultados	27/03/2022 20:45	Carpeta de archivos	
uestra	docker-compose.yml	14/03/2022 19:17	Archivo de origen ...	1 KB
ones	Dockerfile	27/03/2022 22:29	Archivo	1 KB
o	Iniciar app.bat	26/03/2022 22:32	Archivo por lotes ...	1 KB
s	Iniciar app2.bat	26/03/2022 22:33	Archivo por lotes ...	1 KB
itos	README.md	25/03/2022 20:16	Archivo de origen ...	1 KB
	requirements.txt	27/03/2022 20:45	Documento de te...	1 KB

code: código app.py entorno 1: Flask y templates.

code2: código ap2p.py entorno 1: FastApi y templates.

docker2: Entorno virtual para ejecutar Flask.

docker3: Entorno virtual para ejecutar FastApi.

Instrucciones: Simplemente notas que me iba creando para ejecutar entornos etc.

Resultados: donde tenemos imágenes de muestra, o se almacena la imagen cargada y el audio en local.

Docker-compose.yml y Dockerfile, para la creación del contenedor Docker.

Iniciar app.bat Script bash que inicia el entorno virtual 1 y lanza Flask

Iniciar app2.bat Script bash que inicia el entorno virtual 2 y lanza FastApi

Requirements.txt son los requerimientos de todas las librerías necesarias para el funcionamiento de la app2.



GENERALITAT
VALENCIANA
Conselleria de Educació,
Cultura y Esport

TOTS
A UNA
VEU



Carrer d'Illueca, 28
03206 Elx, Espanya
03013224.secret@gva.es
(+34) 966.912.260



Fondo Social Europeo

El FSE invierte en tu futuro

Conclusión:

Una práctica a la que he dedicado mucho tiempo y aun así no he conseguido realizar lo que se proponía, me ha resultado bastante difícil la búsqueda de información, por el simple hecho de que no entendía bien que debía hacer o buscar, ya que antes de empezar no tenía muy claros muchos de los conceptos o comandos que hemos utilizado. Creo que he aprendido bastante a pesar de no obtener buenos resultados.

Comentar también, que en la anterior practica guiada no pude completar la parte final por un problema que tenía con Windows y Docker al utilizar Virtual box y no usar HyperX, ya que los enlaces no se redireccionaban de la forma correcta.