

Proyecto final IABD

Kevin Cerro Rabanal

1. Objetivo

Desarrollar un proyecto con Flask, AWS y Azure que integre los conocimientos aprendidos en el curso.

En este caso, una herramienta online (SaaS) que integra diferentes herramientas de la nube de Azure y AWS.

- Texto a voz
- Voz a texto
- Imagen a texto => OCR
- Análisis de imágenes

2. Control de versiones

Para el control de versiones se utiliza un repositorio de Git en Github.

<https://github.com/kevincerro/iabd-proyecto-final>

The screenshot shows the GitHub repository page for 'kevincerro/iabd-proyecto-final'. The repository is public and has 58 commits. The main branch is 'master'. The repository contains several files and directories, including 'src', 'static', 'templates/dashboard', '.env', '.flaskenv', '.gitignore', 'README.md', 'docker-compose.yaml', 'main.py', 'package-lock.json', 'package.json', 'requirements.txt', and 'serverless.yml'. The repository is also linked to a pull request for 'feature/aws-lambda'.

File	Commit Message	Time Ago
src	Add image analysis	35 minutes ago
static	Remove not needed css property	2 days ago
templates/dashboard	Add image analysis	35 minutes ago
.env	Add azure vars to env	4 hours ago
.flaskenv	Fix local dev host	5 days ago
.gitignore	Configure serverless	13 days ago
README.md	Use docker as local dev environment	13 days ago
docker-compose.yaml	Deprecate docker app container	8 days ago
main.py	Add image analysis	35 minutes ago
package-lock.json	Update npm packages	yesterday
package.json	Update npm packages	yesterday
requirements.txt	Add lib for azure speech	4 hours ago
serverless.yml	Add serverless env vars for azure	4 hours ago

About
Proyecto final del Curso de Especialización en Inteligencia Artificial y Big Data
0 stars
1 watching
0 forks

Releases
No releases published
[Create a new release](#)

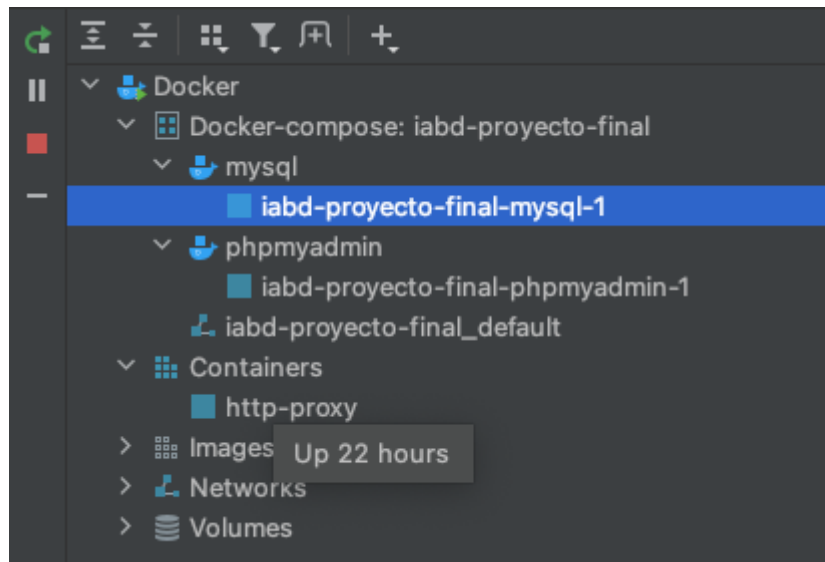
Packages
No packages published
[Publish your first package](#)

Languages
Twig 55.2% Python 42.5% JavaScript 1.6% Shell 0.7%

3. Entorno de desarrollo

Para desarrollar en local, se utiliza “[docker](#)” y “[docker-compose](#)”

Esto permite que cualquier persona pueda replicar el mismo entorno en su ordenador.

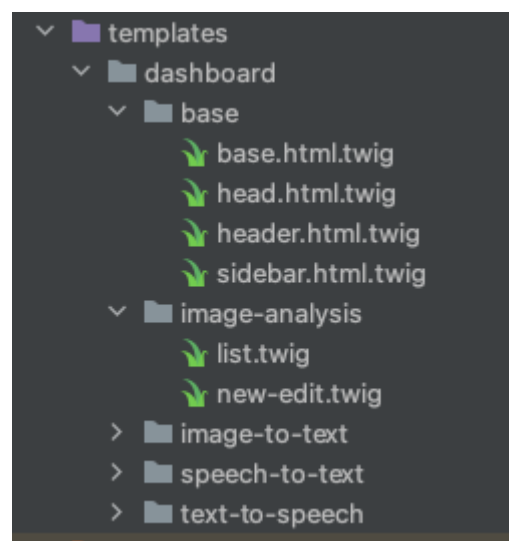
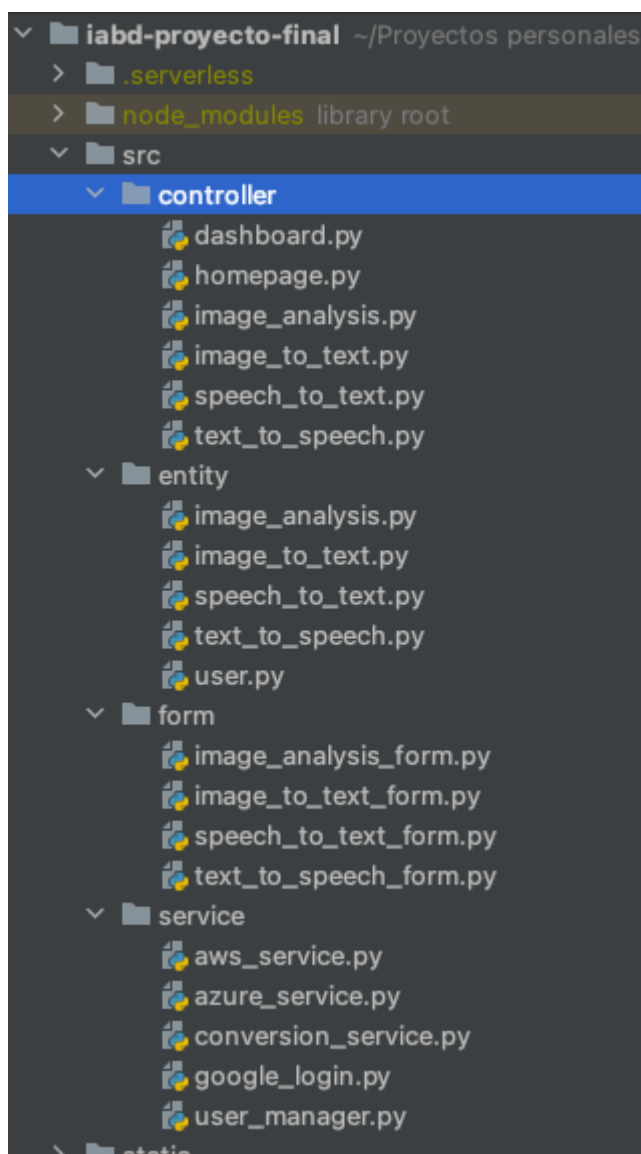


4. Estructura del proyecto

Para organizar el proyecto y facilitar el desarrollo, se investiga sobre las posibles arquitecturas que se pueden usar en Flask

Al final, se opta por utilizar “Blueprints” (<https://flask.palletsprojects.com/en/2.1.x/blueprints/>) El concepto es estructurar el proyecto en controladores, vistas y modelos, esto en otros lenguajes se llama MVC.

En una primera fase del proyecto, se comenzó utilizando una única clase dashboard.py para gestionar todas las rutas del dashboard, pero a medida que el proyecto crecía, este patrón de diseño permitió que se fuera ampliando de forma progresiva y fácil.



Un inconveniente que se ha resuelto es que flask necesita un punto de inicio, es decir el fichero que va a ejecutar por defecto, en este caso “main.py”. Esto implica que hay que cargar todos los demás ficheros del proyecto desde el punto de inicio.

La forma de resolverlo ha sido llamar a todos los controladores, modelos y servicios desde el fichero “main.py”. En otros lenguajes como php se utiliza algo parecido aunque automatizado, llamado autoloader (de composer).

```
# Import models
from src.entity import user
user.init()
from src.entity import text_to_speech
text_to_speech.init()
from src.entity import speech_to_text
speech_to_text.init()
from src.entity import image_to_text
image_to_text.init()
from src.entity import image_analysis
image_analysis.init()

# Import services
from src.service import google_login
google_login.init(app)
from src.service import user_manager
user_manager.init(app)
from src.service import aws_service
aws_service.init(app)
from src.service import azure_service
azure_service.init(app)
```

```
# Import controllers
from src.controller import homepage
app.register_blueprint(homepage.mod)
from src.controller import dashboard
app.register_blueprint(dashboard.mod)
from src.controller import text_to_speech
app.register_blueprint(text_to_speech.mod)
from src.controller import speech_to_text
app.register_blueprint(speech_to_text.mod)
from src.controller import image_to_text
app.register_blueprint(image_to_text.mod)
from src.controller import image_analysis
app.register_blueprint(image_analysis.mod)
```

5. Base de datos y almacenamiento

Los datos del proyecto, dependiendo de su tipo y propósito deben de almacenarse.

- Por una parte, los datos generados por el proyecto y los usuarios se almacenan en una base de datos Mysql, usando [AWS RDS](#).

Mediante [SQLAlchemy](#), una librería para [Flask](#), es posible definir modelos y gestionar la conexión con la base de datos.

```
# Initialize flask
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = os.environ.get('DATABASE_URL')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

```
# Initialize db
db = SQLAlchemy(app)
```

Por otra parte, los recursos estáticos se almacenan en [AWS S3](#)

La siguiente configuración de serverless crea un bucket en AWS S3

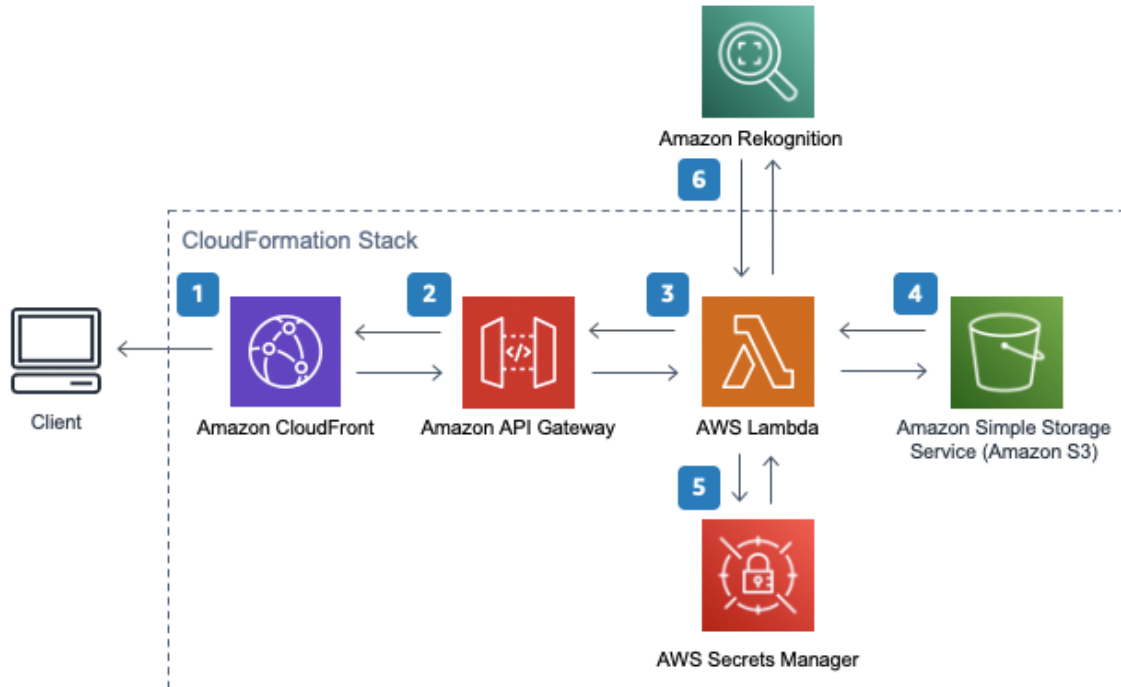
```
constructs:
  uploads:
    type: storage
```

Con este código python, se genera una url pre-firmada para que un usuario suba un fichero a S3. (Más adelante se detalla por qué no se puede subir un fichero de forma directa desde el servidor)

```
def get_presigned_upload_url(file_name: str):
    return s3.generate_presigned_url(
        'put_object',
        Params={
            'Bucket': s3_bucket,
            'Key': UPLOADS_TEMP_DIR + file_name
        },
        ExpiresIn=3600
    )
```

6. Preparando el entorno de producción

El objetivo es desplegar el proyecto a AWS Lambda usando el framework [serverless](https://serverless.com). (<https://iabd.kevincerro.com>)



(Imagen <https://aws.amazon.com/es/solutions/implementations/serverless-image-handler/>)

Una de las particularidades de AWS Lambda es que el sistema de ficheros es de solo lectura, por lo que las siguientes funcionalidades que por norma general se asumen que funcionan por defecto en Lambda no son ciertas:

- Las sesiones y cookies de los servidores web no se pueden almacenar
- No se pueden generar ni almacenar ficheros
- El tamaño de respuesta de una petición HTTP no puede superar los 6MB, por lo que no se pueden enviar ni recibir ficheros pesados por HTTP
- El timeout de API Gateway (proxy de AWS Lambda) es de 29 segundos, si tarda más, la ejecución se interrumpe.

Para resolver estas cuestiones, se emplean diferentes estrategias.

Para enviar y recibir ficheros de los usuarios, se utilizan URL pre-firmadas de AWS S3, esto permite que el usuario pueda recibir o enviar ficheros directamente desde/hacia AWS S3, sin pasar por el servidor.

Ejemplo URL pre-firmada en HTML para que el navegador obtenga un audio desde AWS S3

```
<td>
  <audio controls> == $0
    <source src="https://iabd-proyecto-final-prod-uploadsbucket10132eb4-16imlrxmn9yww.s3...RCXVVK5X7E50P7&Signature=Y1Z3drwhiKhMXNYNlij7FUryZNM%3D&Expires=1654450290" type="audio/mpeg">
    <a href="https://iabd-proyecto-final-prod-uploadsbucket10132eb4-16imlrxmn9yww.s3...RCXVVK5X7E50P7&Signature=Y1Z3drwhiKhMXNYNlij7FUryZNM%3D&Expires=1654450290">Descargar</a>
  </audio>
```

Para que funcionen las sesiones, como no se pueden almacenar en el sistema, se utiliza una base de datos, en este caso Mysql, aunque puede ser Redis, DynamoDb u otras. Gracias a [Flask-session](#), podemos conseguirlo.

```
# Initialize session
app.config['SESSION_TYPE'] = 'sqlalchemy'
app.config['SESSION_USE_SIGNER'] = True
app.config['SESSION_SQLALCHEMY'] = db
app.config['SESSION_SQLALCHEMY_TABLE'] = 'Sessions'
app.config['PERMANENT_SESSION_LIFETIME'] = 604800 # 7 days
Session(app)
```

The screenshot displays a web application interface with a sidebar on the left containing a tree view of the database structure. The main area shows a table named 'Sessions' with columns 'id', 'session_id', 'data', and 'expiry'. The table contains two rows of data. The interface includes search and filter controls at the top and bottom of the table view.

Database Structure (from sidebar):

- iabd
 - Nueva
 - ImageAnalysis
 - ImageToText
 - Sessions
 - SpeechToText
 - TextToSpeech
 - Users
- information_schema

Sessions Table Data:

id	session_id	data	expiry
2	session:97b67639-a5ed-47c0-8545-4f5f97ff69da	[BLOB - 508 B]	2022-06-12 16:30:49
3	session:04bffa8c-ecad-482a-876d-a900562de0c6	[BLOB - 248 B]	2022-06-10 20:03:29

En el fichero "serverless.yml" se detallan los recursos y configuraciones para el despliegue en AWS.

```
plugins:
  - serverless-wsgi
  - serverless-python-requirements
  - serverless-lift

custom:
  wsgi:
    app: main.app

functions:
  app:
    handler: wsgi_handler.handler
    timeout: 28 # in seconds (API Gateway has a timeout of 29 seconds)
    events:
      - httpApi: '*'
```

Otro aspecto a resolver es cómo servir los recursos estáticos a los usuarios finales, es decir, ficheros CSS, JS, fuentes e Imágenes.

Gracias a serverless y [CloudFront](#), se pueden gestionar las peticiones y dividirlos en 2 tipos: este comportamiento.

- Peticiones a **/static/** => las envía a un bucket de S3 que contiene los recursos estáticos **cacheados**. (Recordemos, CSS, JS, fuente e Imágenes)
- Resto de peticiones las envía a [API Gateway](#) y este a su vez a [AWS Lambda](#)

▼ Response Headers

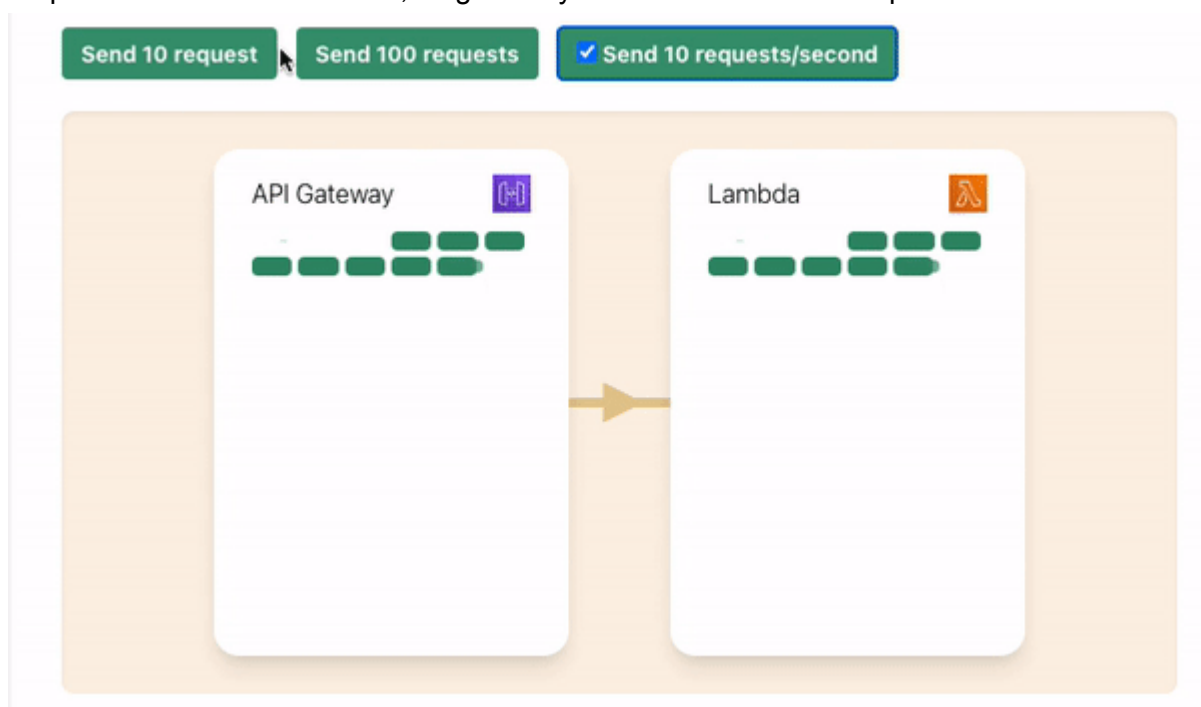
```
age: 3754
content-encoding: br
content-type: text/css
date: Sun, 05 Jun 2022 16:17:30 GMT
etag: W/"e42c16843fdc76ab98cdeca181dfab0"
last-modified: Fri, 03 Jun 2022 22:46:15
server: AmazonS3
vary: Accept-Encoding
via: 1.1 dadbd3993b5303886df72c2fdce172c (CloudFront)
x-amz-cf-id: wEL0NrQ3v3NALApXrYyWGD0MhwLIH4DNqNDQ==
x-amz-cf-pop: MAD51-C2
x-cache: Hit from cloudfront
```


Un inconveniente de que las peticiones pasen por CloudFront y API Gateway (actúan como proxy) es que perdemos cierta información de la petición HTTP original.

Para solucionarlo, utilizamos [ProxyFix](#) para python. Al inicializar la app de flask, añadimos el siguiente código y ya tendremos acceso a la IP original del cliente, el host que ha usado, entre otros.

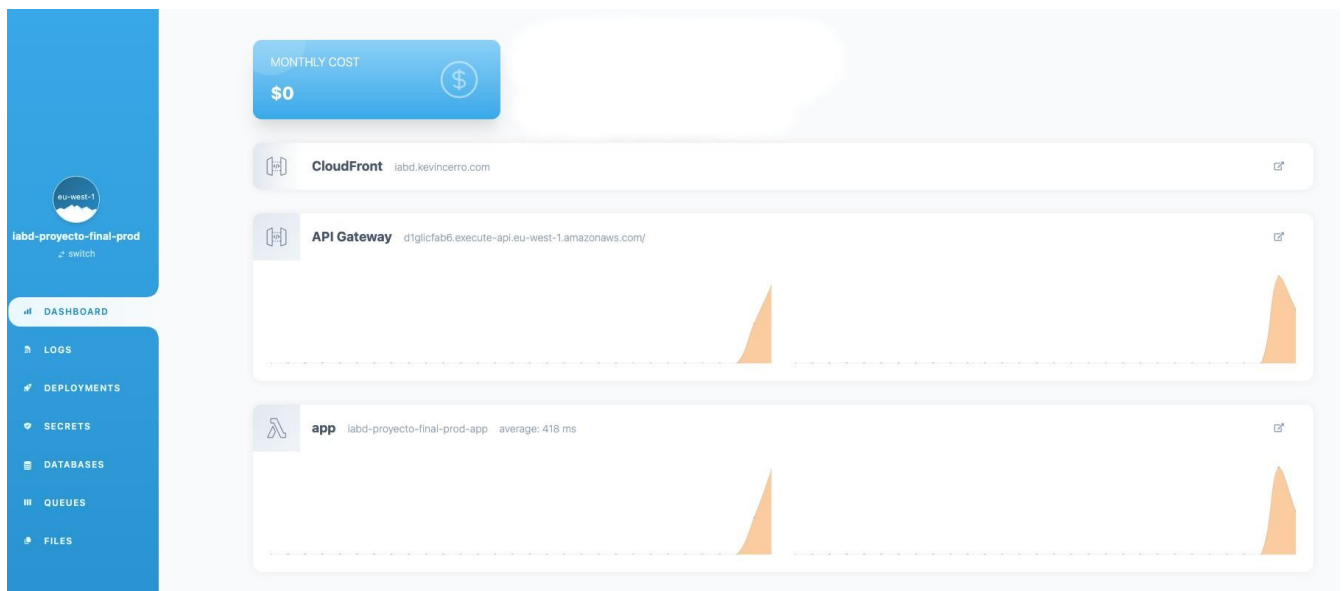
```
# Initialize flask
app = Flask(__name__)
app.wsgi_app = ProxyFix(app.wsgi_app, x_host=1)
```

Esta arquitectura permite escalar a medida que el tráfico web se incrementa o decrementa, simplificando la infraestructura, la gestión y reduciendo los costes operativos.




(<https://bref.sh/>)

(<https://serverless-visually-explained.com>)



7. Resultado final

Texto a voz usando Azure y AWS en diferentes idiomas.


 **Text-To-Speech**
Aquí puedes gestionar todas las conversiones de texto a voz.

[+ Nuevo](#)

TEXT-TO-SPEECH

#	Motor	Idioma	Texto	Audio	Fecha de creación
4	Azure	EN	<div>Mostrar texto</div> <div>Hello world</div>	<div>▶ 0:00 / 0:01</div> <div></div> <div>🔊 ⋮</div>	2022-06-05 16:31:28
2	Azure	ES	<div>Mostrar texto</div> <div>Hola mundo</div>	<div>▶ 0:00 / 0:01</div> <div></div> <div>🔊 ⋮</div>	2022-06-05 16:31:16
1	Aws	ES	<div>Mostrar texto</div> <div>Hola mundo</div>	<div>▶ 0:00 / 0:00</div> <div></div> <div>🔊 ⋮</div>	2022-06-05 16:31:09

Voz a texto usando AWS en varios idiomas


 **Speech-To-Text**
Aquí puedes gestionar todas las conversiones de voz a texto.

[+ Nuevo](#)

SPEECH-TO-TEXT

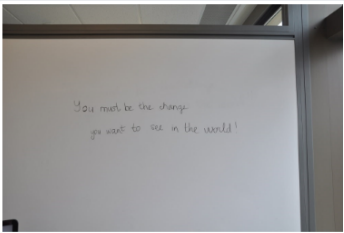
#	Idioma	Audio	Texto	Fecha de creación
1	ES	<div>▶ 0:00 / 0:00</div> <div></div> <div>🔊 ⋮</div>	<div>Mostrar texto</div> <div>Hola mundo.</div>	2022-06-05 17:33:25

Imagen a texto usando AWS y Azure



 **Image-To-Text**
Aquí puedes gestionar todas las conversiones de imagen a texto.

[+ Nuevo](#)

IMAGE-TO-TEXT

#	Motor	Imagen	Texto	Fecha de creación
1	Aws		<div>Mostrar texto</div> <div>You must be the change you want to see in the world!</div>	2022-06-05 17:34:45

Análisis de imágenes usando Azure en varios idiomas

Image Analysis			
Aquí puedes gestionar todos los análisis de imágenes			
+ Nuevo			
IMAGE ANALYSIS			
#	Idioma	Imagen	Resultado
3	EN		<div>Mostrar resultado</div> <div>2022-06-05 16:24:37</div> <div>Mark Zuckerberg, Robert Huth et al. posing for a photo</div>
2	ES		<div>Mostrar resultado</div> <div>2022-06-05 16:24:07</div> <div>Ken Takemoto, Takayo Fischer posa para una fotografía</div>