

# Deep Learning y despliegue en Cloud

# GALAXY CLASSIFIER



THE  
**CREW**



Álvaro Martínez



Juan Pablo Rizzi



Sara Gil



Rocío Ortiz



José Benegas

# INTRODUCCIÓN

# GALAXY CLASSIFIER

Presentación y contexto: Importancia científica y tecnológica.

El modelo predictivo

Back end y front end

Git y estrategia de ramas

Despliegue y demo



# CONTEXTO Y NEGOCIO

Clasificación de galaxias en la era del  
Big Data astronómico

generación de datos > clasificación manual

Relevancia Científica

La morfología galáctica refleja la evolución y  
los procesos físicos internos de las galaxias.

Nuestro Dataset: Galaxy10 DECaLS

- Basado en imágenes de alta resolución del Dark Energy Camera Legacy Survey.
- ~18.000 galaxias clasificadas en 10 clases morfológicas bien diferenciadas.

Motivación Tecnológica / Negocio

- IA aplicada a astronomía: sector en expansión.
- industria cara: reduce tiempo y costes de procesamiento.
- Potenciales clientes: Agencias espaciales, Observatorios e institutos de investigación , Consorcios de telescopios...

# EL MODELO PREDICTIVO

Nuestro Dataset: Galaxy10 DECaLS

1. ~18.000 fotos y 10 categorias a clasificar

Tamaño original imagen 224x224

Archivo H5 de 2.7GB => muy pesado para drive.

2. Dividimos en train/validation/test y redimensionamos las imagenes a 128x128

Desbalanceo de clases => hacemos

3. undersampling de clases mayoritarias y oversampling de las minoritarias hasta equilibrar clases en 1200 imagenes.

Preprocesado:

4. Resize a tamaño adecuado para el modelo (224x224)

Normalización ( /255 )

Aumentación de imagenes con tf.keras.Sequential

MODELO EfficientNetB0

• Descongelamiento de las últimas 100 capas

Dropout => 0.3

• Red Neuronal: 256 + "Relu", 128 + "Relu", 64 + "Relu"

• Capa salida => 10 + "Softmax"

• Learning rate => Adam (1e-5)

• loss="sparse\_categorical\_crossentropy"

• metrics=> "accuracy"

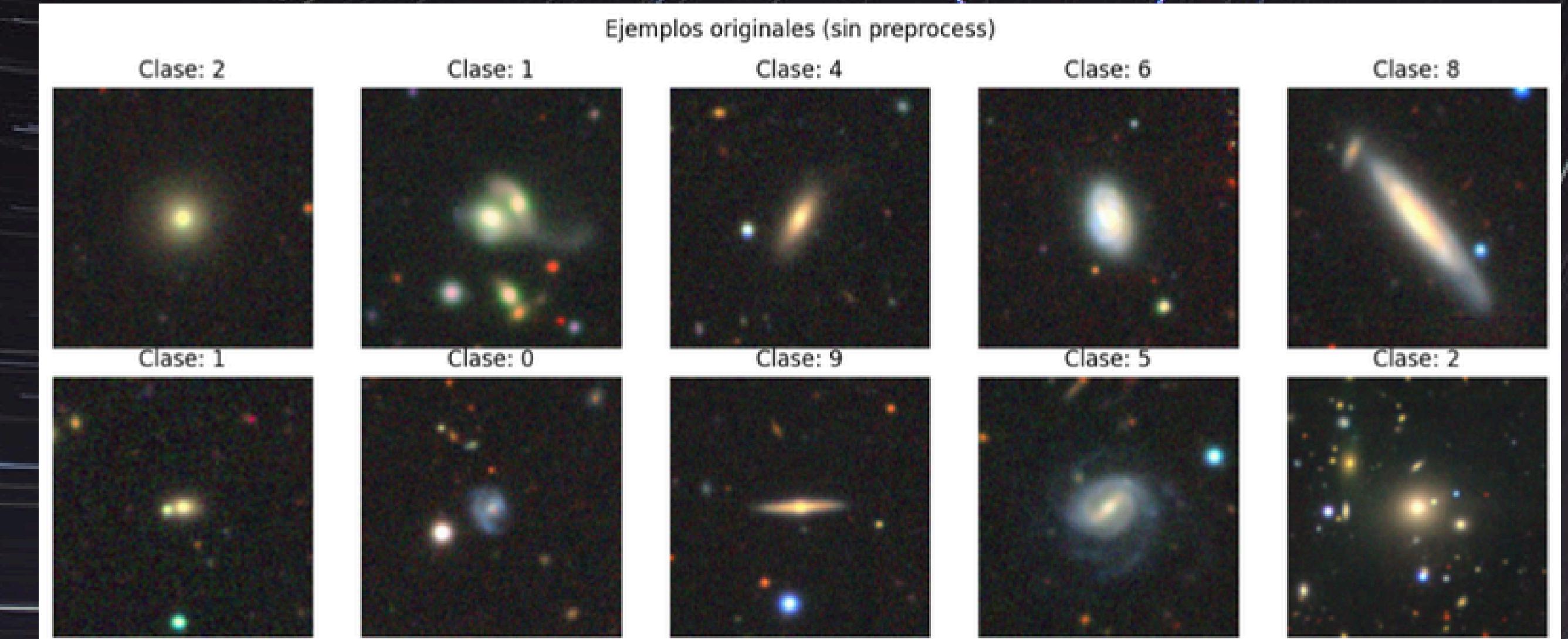
• Early Stop => patience 10,

• restore\_best\_weights=True

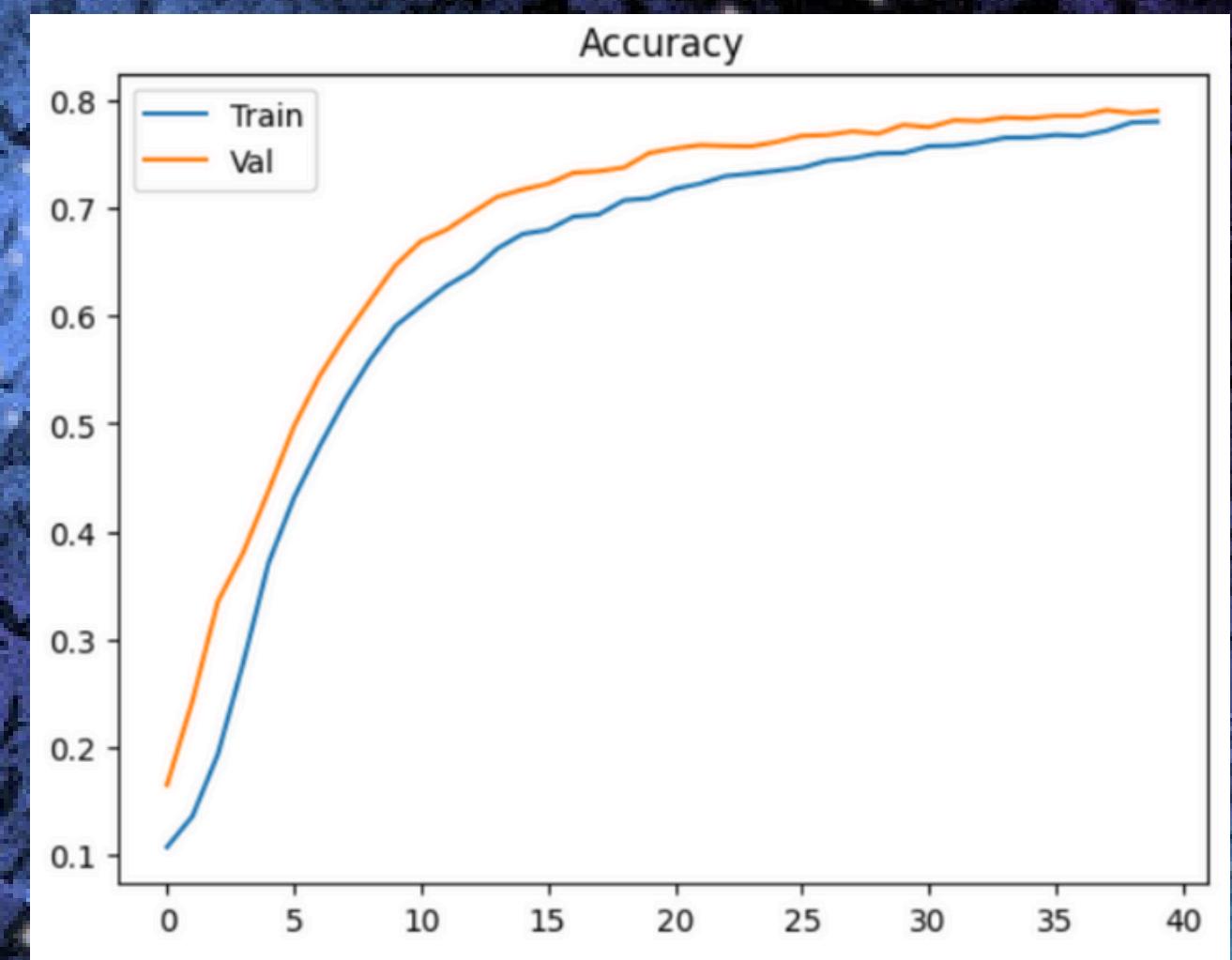
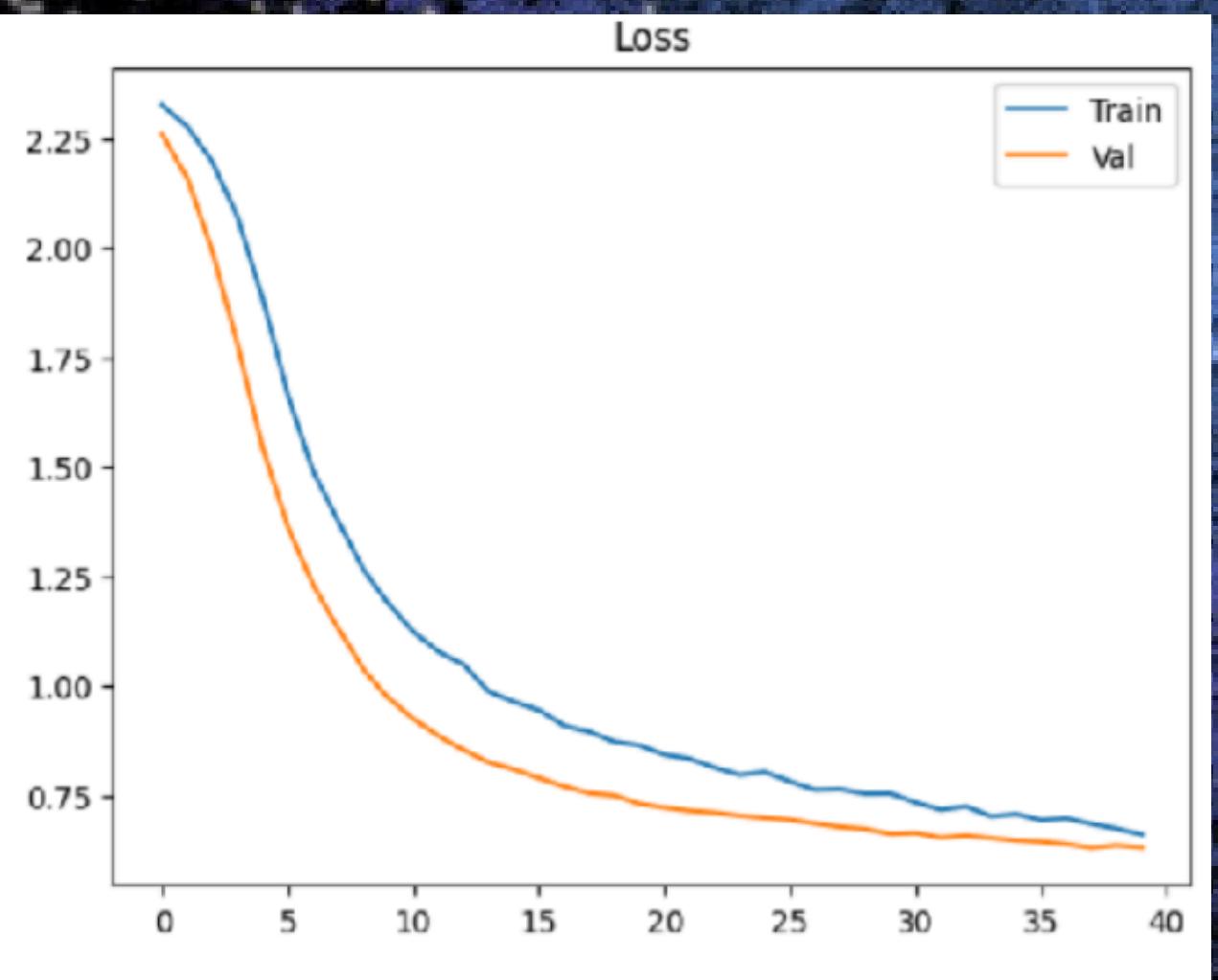
• Epocas= 40

# EL MODELO PREDICTIVO

```
tipos_galaxia = {  
    0: "spiral",  
    1: "elliptical",  
    2: "lenticular",  
    3: "irregular",  
    4: "merger",  
    5: "unknown",  
    6: "barred spiral",  
    7: "compact",  
    8: "edge-on",  
    9: "uncertain"
```



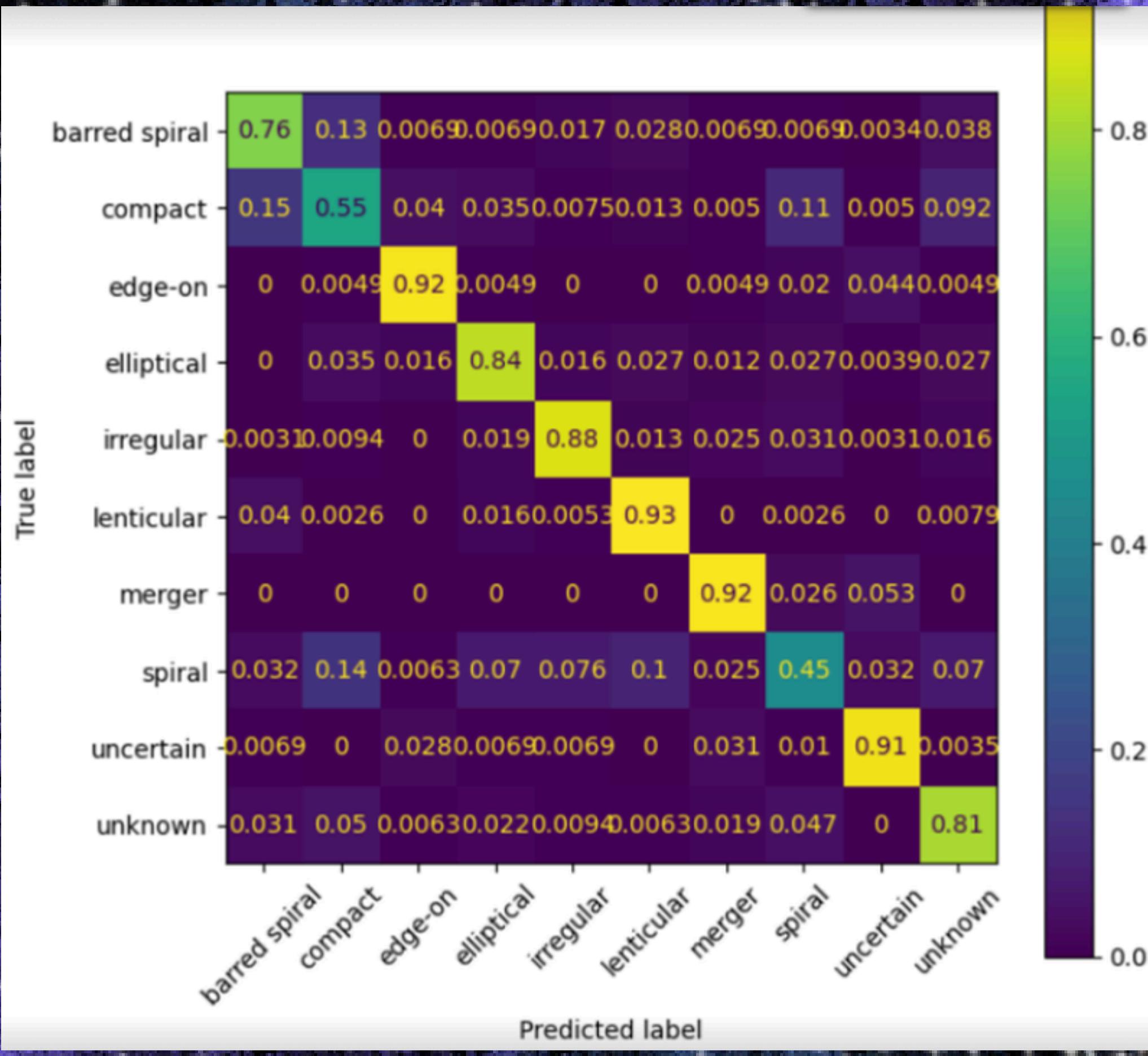
# RESULTADOS DEL MODELO



# RESULTADOS DEL MODELO

		precision	recall	f1-score	support
barred	spiral	0.70	0.76	0.73	290
compact		0.71	0.55	0.62	400
edge-on		0.85	0.92	0.88	204
elliptical		0.82	0.84	0.83	258
irregular		0.90	0.88	0.89	318
lenticular		0.89	0.93	0.91	379
merger		0.50	0.92	0.65	38
spiral		0.45	0.45	0.45	158
uncertain		0.93	0.91	0.92	288
unknown		0.77	0.81	0.79	319
accuracy				0.79	2652
macro avg		0.75	0.79	0.77	2652
weighted avg		0.79	0.79	0.79	2652

# RESULTADOS DEL MODELO



# BASE DE DATOS (POSTGRESQL EN RENDER).

- Almacenamos cada predicción realizada por la API.

Guardamos: nombre de archivo, clase predicha, confianza y fecha.

Usamos **SQLAlchemy** para interactuar con la base de datos de forma sencilla y segura.

# BACKEND (API FLASK):

- API desarrollada con Flask y desplegada en Render.
- Principales endpoints:
  - POST /predict: Recibe una imagen y devuelve la predicción.
  - GET /predictions: Lista todas las predicciones almacenadas.
  - GET /predictions/<id>: Consulta una predicción concreta.
  - DELETE /predictions/reset: Borra todas las predicciones y resetea la BBDD.
  - GET /health: Comprueba el estado de la API.

# DIAGRAMA DE FLUJO

**1.USUARIO**

SUBE IMAGEN  
EN EL FRONT

**2.API FLASK /  
RENDER**

PREDICCIÓN  
CON MODELO

**3.BBDD  
POSTGRESQL**

ALMACENA  
LAS  
PREDICCIONES

**4.API FLASK /  
RENDER**

RESPUESTA DE  
LA API

**5. USUARIO**

RESPUESTA EN  
FRONT

# GIT Y LA ESTRATEGIA DE RAMAS

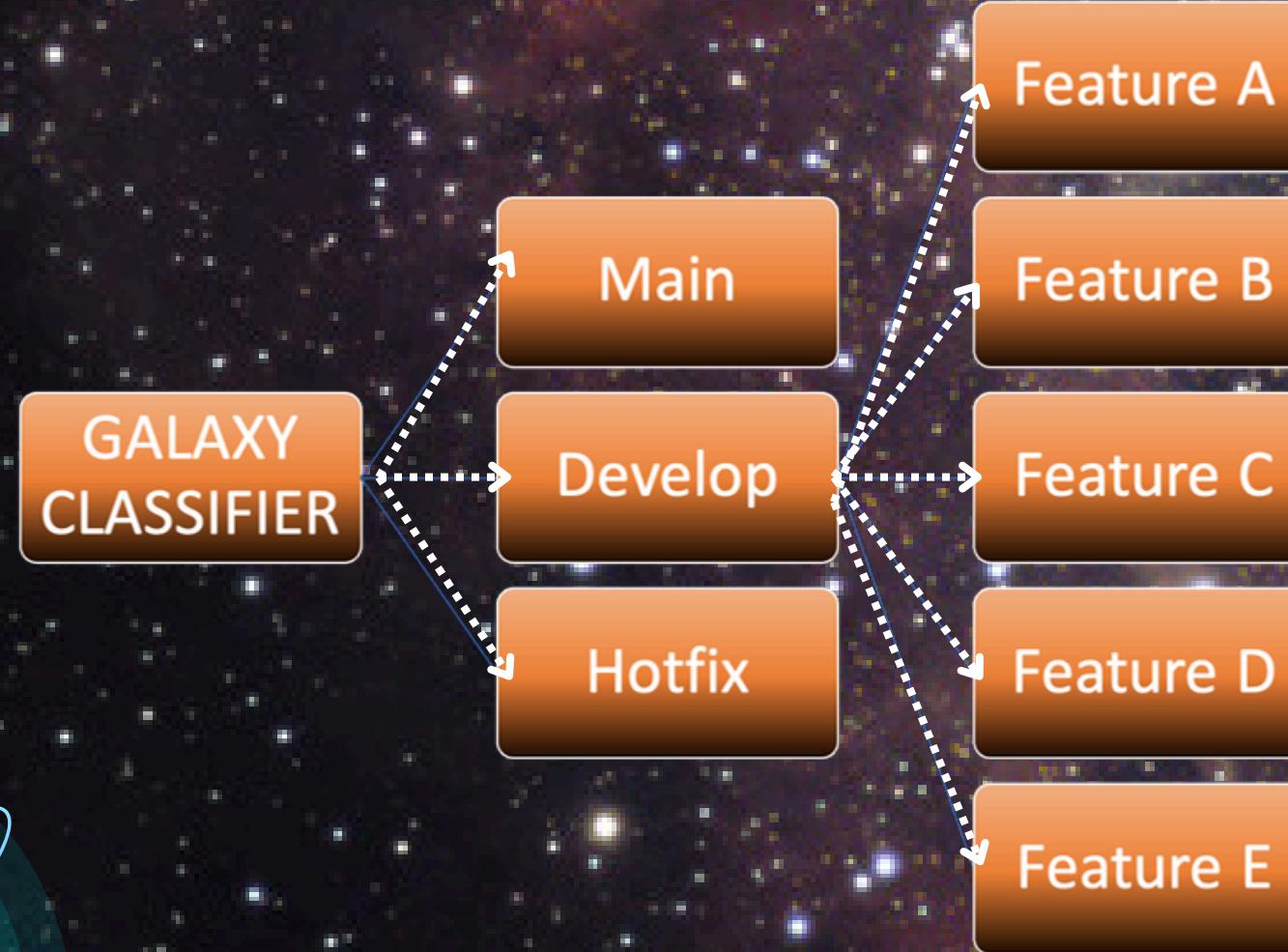
Estrategia de ramas: main, develop, hotfix, feature A/B/C/D/E

Se usan pull requests en github para proponer cambios y revisiones.

Se documentan los comandos usados.

Se hacen capturas de pantalla para evidenciar el proceso.

# GIT Y LA ESTRATEGIA DE RAMAS



- Corregir decimales app rizzijp @develop
- Eliminar ficheros vacíos rizzijp
- Merge pull request #10 from rocio2125/f...
- Create model\_v6\_colab.ipynb rocio2125
- Delete testeo.py rocio2125
- Merge pull request #1 from rocio2125/...
- commit rocio2125
- v3 rocio2125
- Merge branch 'feature/B' rocio2125
- Correccion path de imagen rizzijp
- end point nuevo + cambios en el endpoi...
- Merge pull request #8 from Alvaro-mva...
- upload de Readme Alvaro-mval
- Merge branch 'feature/B' rocio2125
- Se vuelve a subir el modelo en TFLite ...
- Fijar version tensorflow a 2.15.0 rizzijp
- Adaptacion de app.py para modelo TFLite...
- Merge pull request #5 from Alvaro-mva...
- notebook de conversion de formato de ...
- Merge branch 'develop' of https://git...

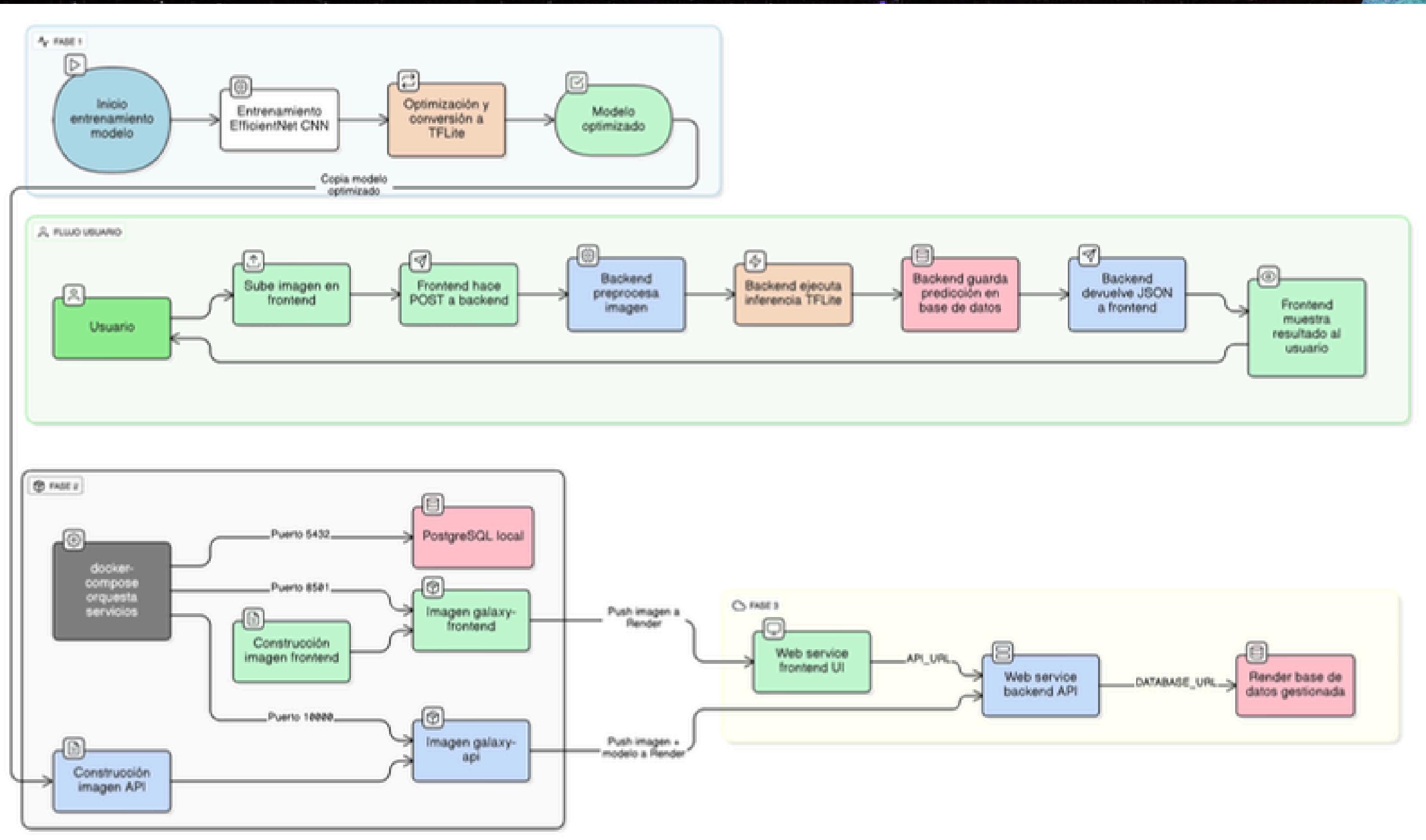
# DESPLIEGUE

**Objetivo:** hacer accesible la API desde cualquier lugar.

**Servidor Cloud:**  Render

- Gestión de base de datos PostgreSQL
- Integración con GitHub
- Webservice a partir del Dockerfile
- Conexión de base y webservices mediante variables de entorno

# FASES DEL DESPLIEGUE



```

    <pre>
        > src
            > api
                app.py
            > database
                __init__.py
                db.py
                models.py
            > frontend
                > assets
                    Dockerfile
                    ui.py
                > models
                    modelo_galaxias.keras
                    modelo_galaxias.tflite
                > utils
                    __init__.py
                    .dockerignore
                    .env
                    .gitignore
                    docker-compose.yml
                    Dockerfile
                LICENSE
                README.md
                requirements.txt
    </pre>

```

# SERVICIOS DESPLEGADOS EN RENDER

**Production**

All (3) Services (3) Env Groups (0)

Search resources in Production

Service Name	Status	Runtime	Region	Updated
galaxy-classifier-frontend	✓ Deployed	Docker	Frankfurt	16h
galaxy-classifier-api	✓ Deployed	Docker	Frankfurt	16h
galaxy-postgres-server	✓ Available	PostgreSQL 18	Frankfurt	2d

# RESULTADO DE PREDICCIONES GUARDADAS EN BBDD (POSTGRESQL)

Data Output    Messages    Notifications

Showing rows: 1 to 3    Page No: 1

The screenshot shows a PostgreSQL database interface with a dark theme. At the top, there are tabs for 'Data Output', 'Messages', and 'Notifications'. Below the tabs is a toolbar with various icons for database management. The main area displays a table with the following data:

	<b>Id</b> [PK] integer	<b>timestamp</b> timestamp without time zone	<b>filename</b> character varying (200)	<b>prediction</b> character varying (500)	<b>confidence</b> double precision
1	1	2025-12-09 15:43:26.606206	galaxia.jpg	edge_on	0.8
2	2	2025-12-09 15:43:51.912219	NGC_2787 (1).jpg	barred spiral	0.66
3	3	2025-12-09 15:44:26.016014	Messier_77.jpg	unknown	0.49



# DESPLIEGUE DEMO

URL pública de la API:

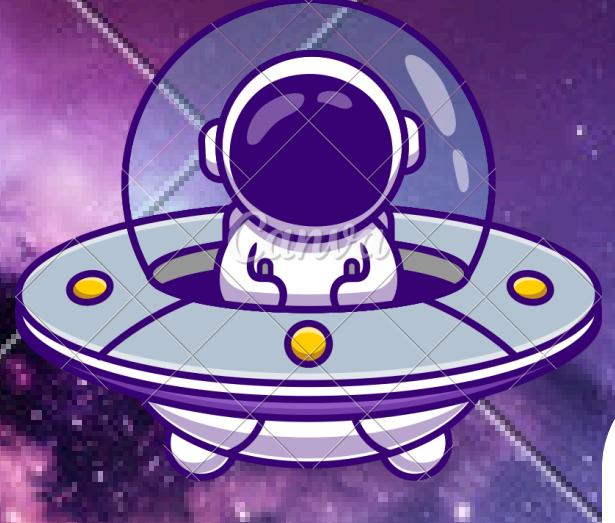
<https://galaxy-classifier-api.onrender.com>

FrontEnd en Streamlit:

<https://galaxy-classifier-frontend.onrender.com/>

# CONCLUSIONES

- Integración de ciencia de datos (Deep learning), frontend, backend y cloud => Aplicación a un caso real.
- El trabajo en equipo y Git facilitaron la colaboración y control de cambios resultando ser una herramienta eficaz para trabajar en grupo.
- La API es accesible, escalable y almacena predicciones en la nube.



# Mejoras futuras

- Mejorar las métricas del modelo => conseguir accuracy > 80%.
- Mejorar el modelo con más datos y técnicas avanzadas => mayor capacidad de computo necesario.
- Aceptar ficheros de muchas imágenes (H5, rar, etc...) con los que alimentar las predicciones.
- Verificación humana de la predicción obtenida.
- Proyecto final de rama DEVELOP a rama MAIN en GitHub.

# GRACIAS

