Construyendo software resiliente y escalable con MLOps

PyCON ES 2024 (Vigo)

https://github.com/alice-biometrics/pycones24

¿Quién Soy?

Daniel Pérez Cabo

- Ingeniero de telecomunicación (UVigo)
- Research ML Engineer

LinkedIn

¿Qué hago?

Tech Lead @ Core-Tech en Alice

Solución de verificación de identidad ágil y segura

¿Qué hacemos?

- Reconocimiento Facial
- Prueba de vida
- Reconocimiento de texto
- Análisis forense de documentos

¿Cual es genuino?

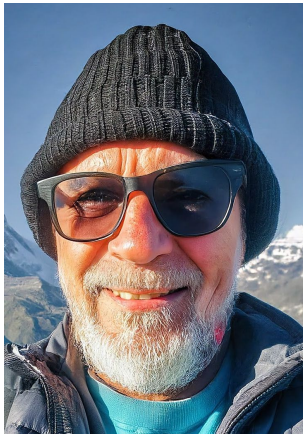
¿Cual es genuino?

¿Cual es genuino?

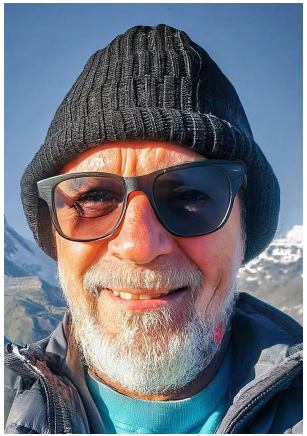
Fake

Genuino

¿Cual es genuino?



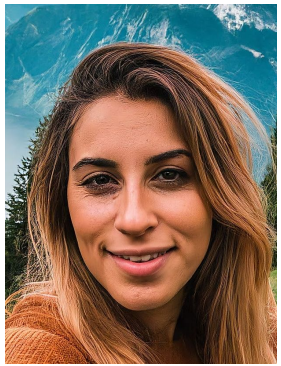
¿Cual es genuino?



Fake

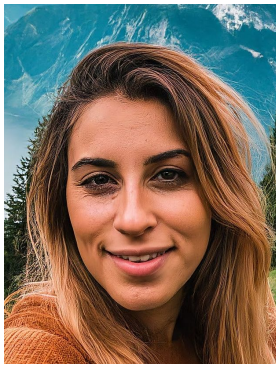
Genuino

¿Cual es genuino?



¿Cual es genuino?

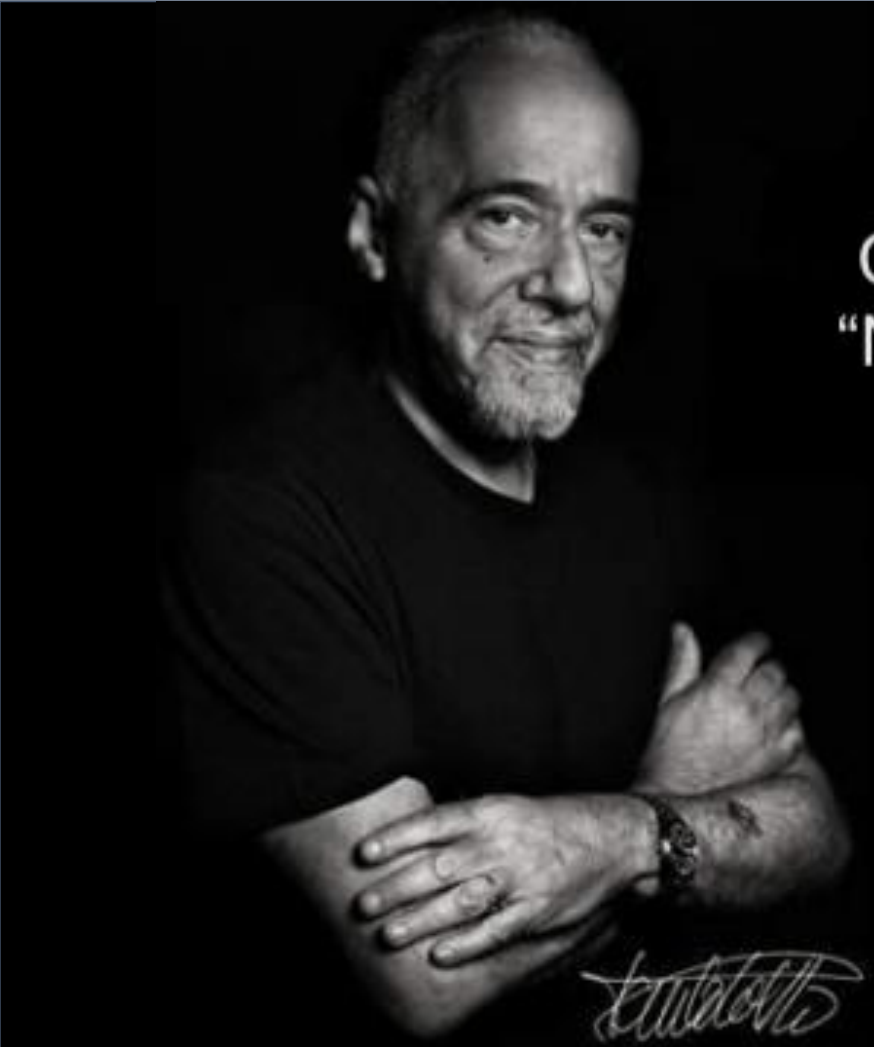
Fake



Fake

Pero …

… en qué nos afecta la escalabilidad y la resiliencia del software



Complejidad Accidental:
“No tenemos tiempo para perderlo con mierdas”

~Paulo Coelho


UX


DX
UX

DX Dev eXperience

UX

DX Dev eXperience

UX

REX

DX Dev eXperience

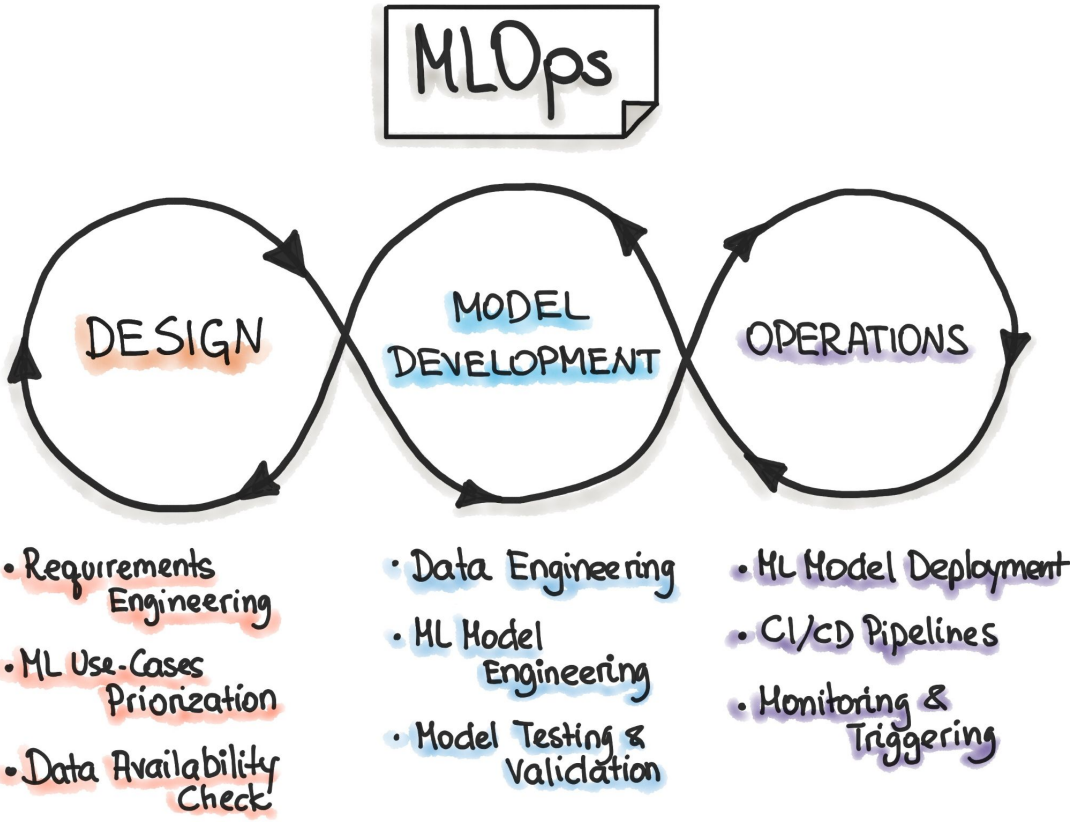
UX

REX Research & Experimentation eXperience

DX
REX

UX

Cíclico y costoso


MLOps
DESIGN
MODEL DEVELOPMENT
OPERATIONS
• Requirements Engineering
• ML Use-Cases Priorization
• Data Availability Check
• Data Engineering
• ML Model Engineering
• Model Testing & Validation
• ML Model Deployment
• CI/CD Pipelines
• Monitoring & Triggering

Objetivos

⬆️ Escalabilidad ⬆️ Resiliencia ⬇️ Coste

Objetivos

⬆️ Escalabilidad ⬆️ Resiliencia ⬇️ Coste

La **evaluación incorrecta** de nuestros modelos puede generar un **alto coste económico** 💸.

Automatizar y versionar el despliegue de modelos acelera el flujo de desarrollo, evitando contratiempos y permitiendo concentrarse en la complejidad intrínseca del problema a resolver. 

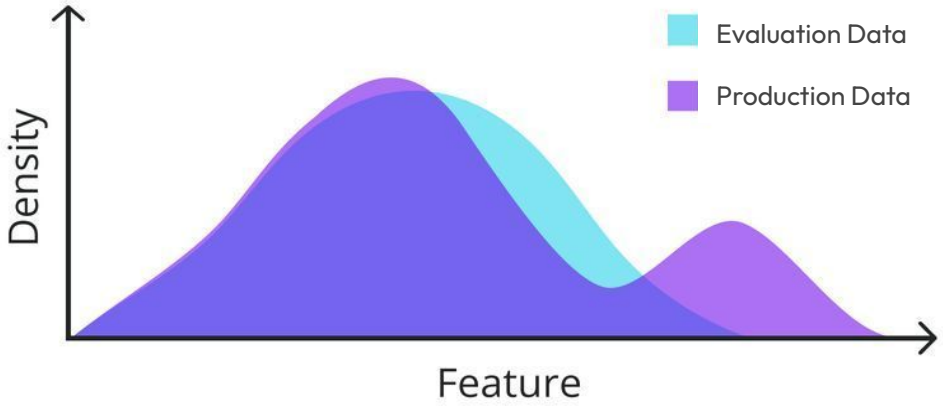
Un buen diseño y arquitectura hace que tus software sea mantenible, flexible y confiable (🙌 también en experimentación y MLOps 🙏).

Test y Validación del Modelo

- Formato final de puesta en producción

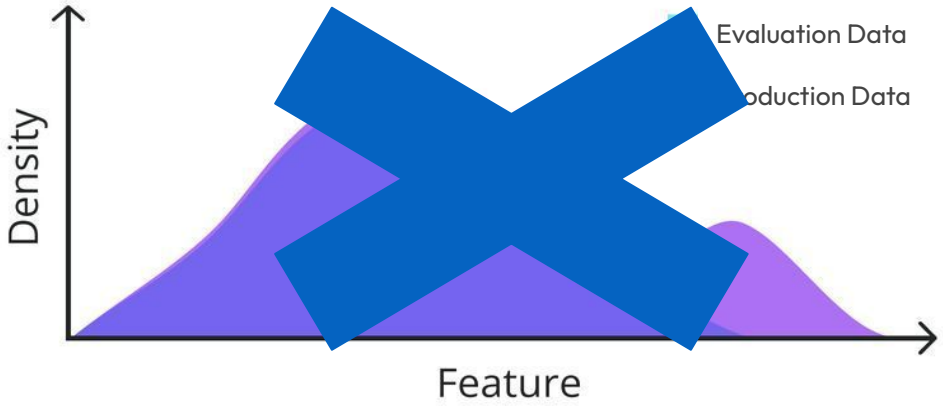
Test y Validación del Modelo

- Formato final de puesta en producción
- Representatividad de producción



Test y Validación del Modelo

- Formato final de puesta en producción
- Representatividad de producción



Evaluación

Producción

Test y Validación del Modelo

- Formato final de puesta en producción
- Representatividad de producción
- Métricas correctas

Problema de Clasificación

Clase	# Muestras Evaluación
🦓	400
🦁	25
🦌	174
🐶	4

Test y Validación del Modelo

- Formato final de puesta en producción
- Representatividad de producción
- Métricas correctas

F1-Score

Accuracy

MSE

Problema de Clasificación

Clase	# Muestras Evaluación
	400
	25
	174
	4

Test y Validación del Modelo

- Formato final de puesta en producción
- Representatividad de producción
- Métricas correctas

Problema de Clasificación

Clase	# Muestras Evaluación
🦓	400
🦁	25
🦌	174
🐶	4

~~Accuracy~~

~~F1-Score~~

Objetivos

⬆️ Escalabilidad ⬆️ Resiliencia ⬇️ Coste

La **evaluación incorrecta** de nuestros modelos puede generar un **alto coste económico** 💸.

Automatizar y versionar el despliegue de modelos acelera el flujo de desarrollo, evitando contratiempos y permitiendo concentrarse en la complejidad intrínseca del problema a resolver. 

Un buen diseño y arquitectura hace que tus software sea mantenible, flexible y confiable (🙌 también en experimentación y MLOps 🙏).

Objetivos

⬆️ Escalabilidad ⬆️ Resiliencia ⬇️ Coste

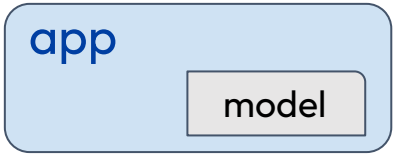
La evaluación incorrecta de nuestros modelos puede generar un alto coste económico 💸.

Automatizar y versionar el despliegue de modelos **acelera el flujo de desarrollo**, evitando contratiempos y permitiendo mantener el **foco** en la **complejidad intrínseca del problema a resolver**. 

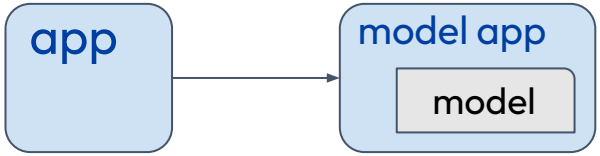
Un buen diseño y arquitectura hace que tus software sea mantenible, flexible y confiable (🙌 también en experimentación y MLOps 🙏).

¿Dónde despliego mi modelo?

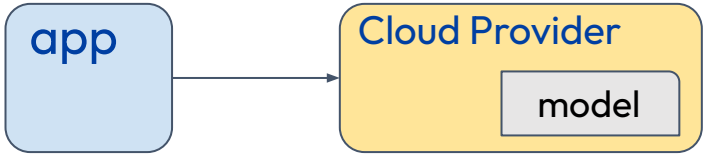
Dentro de mi aplicación



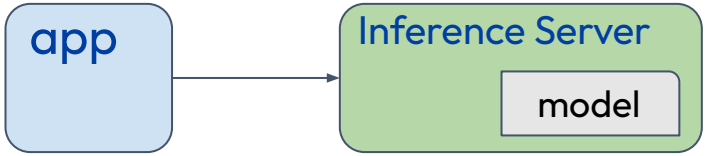
Como una web app independiente



Cloud Provider

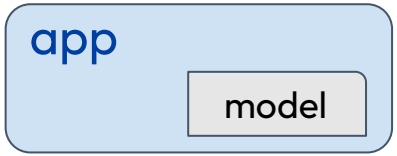


Con un Inference Server



¿Dónde despliego mi modelo?

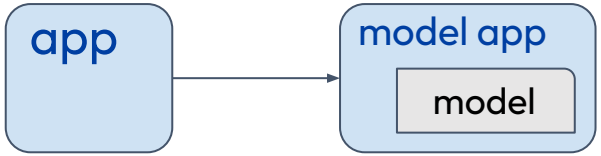
Dentro de mi aplicación



Pros	Cons
• Facilidad de integración • Complejidad mínima • Versión de modelo asociada a versión de aplicación	• Tests complejos • Dependencias extra • Versión de modelo asociada a versión de aplicación • Complejidad de escalado

¿Dónde despliego mi modelo?

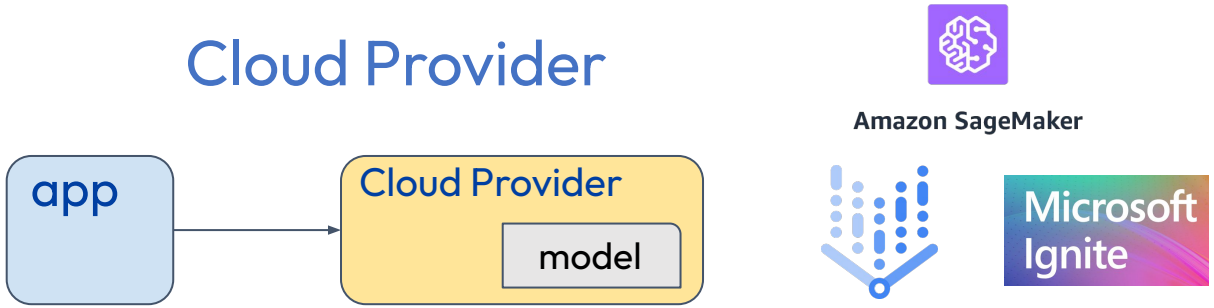
Como un web app independiente



FastAPI Flask django

Pros	Cons
• Flexibilidad • Testeo • Versión de modelo independiente de la aplicación	• Complejidad de escalado • Optimización de recursos (costes) compleja • Mantenimiento

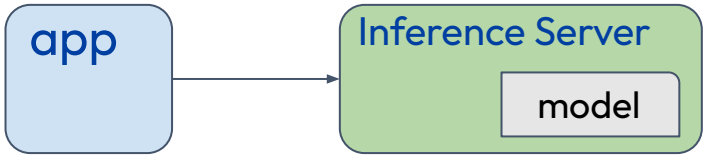
¿Dónde despliego mi modelo?



Pros	Cons
• Facilidad de uso • Auto escalado out of the box • Versión de modelo independiente de la aplicación	• Vendor lock-in • Agilidad y flexibilidad del auto escalado • Dificultad para testear • Difícil de personalizar • Optimización de recursos (costes) compleja

¿Dónde despliego mi modelo?

Con un Inference Server



Torch Serve

NVIDIA
TRITON INFERENCE SERVER

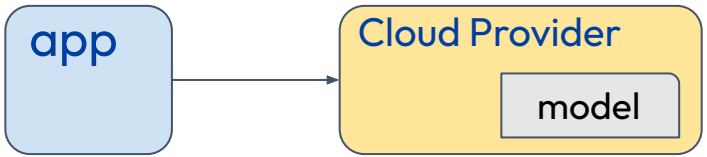
TensorFlow
/serving

Pros	Cons
• Flexibilidad • Personalización • Testeo • Optimización de recursos (costes)	• Testeo • Mantenimiento

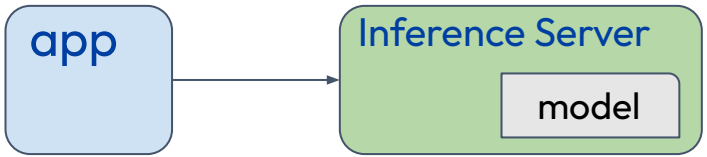
¿Dónde despliego mi modelo?

Nuestro Approach

Anterior



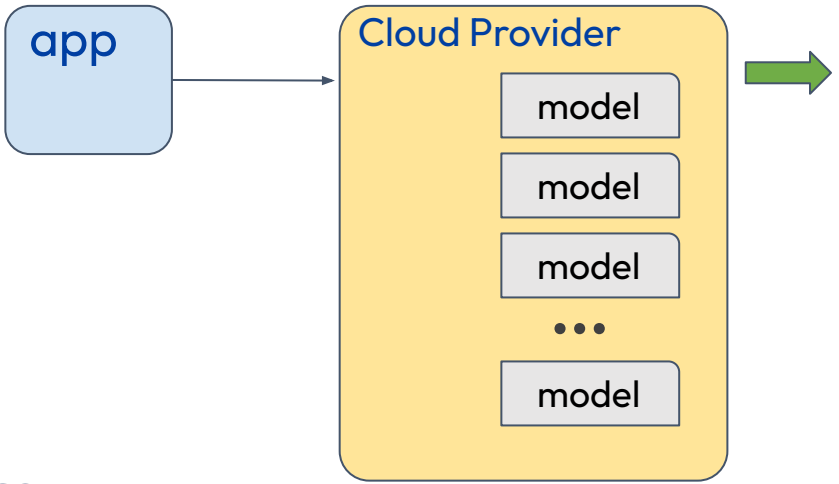
Actual



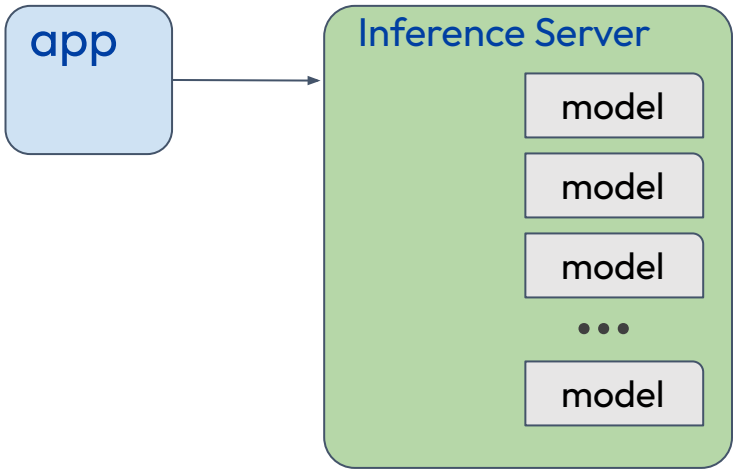
¿Dónde despliego mi modelo?

Nuestro Approach

Anterior



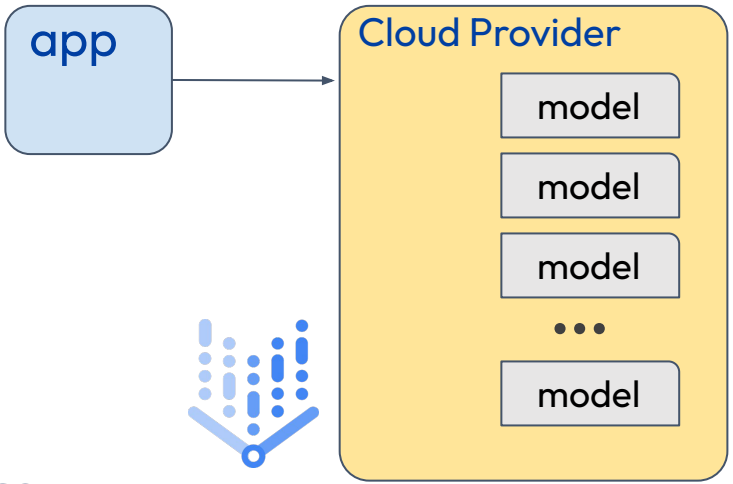
Actual



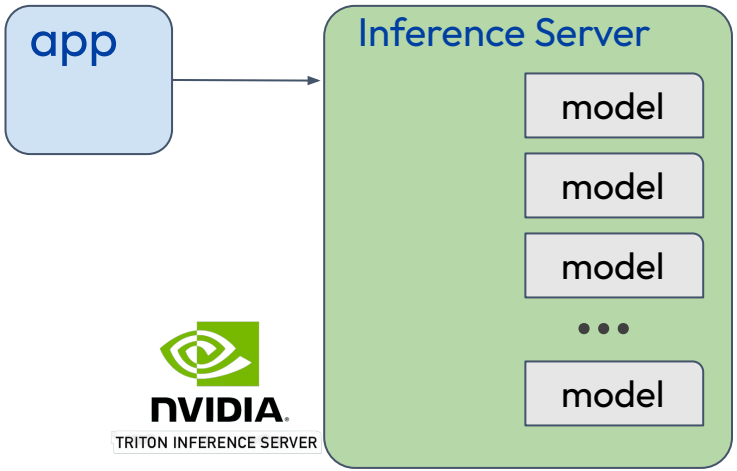
¿Dónde despliego mi modelo?

Nuestro Approach

Anterior



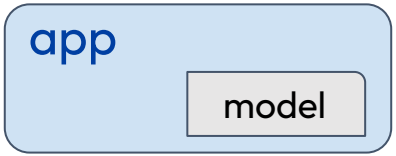
Actual



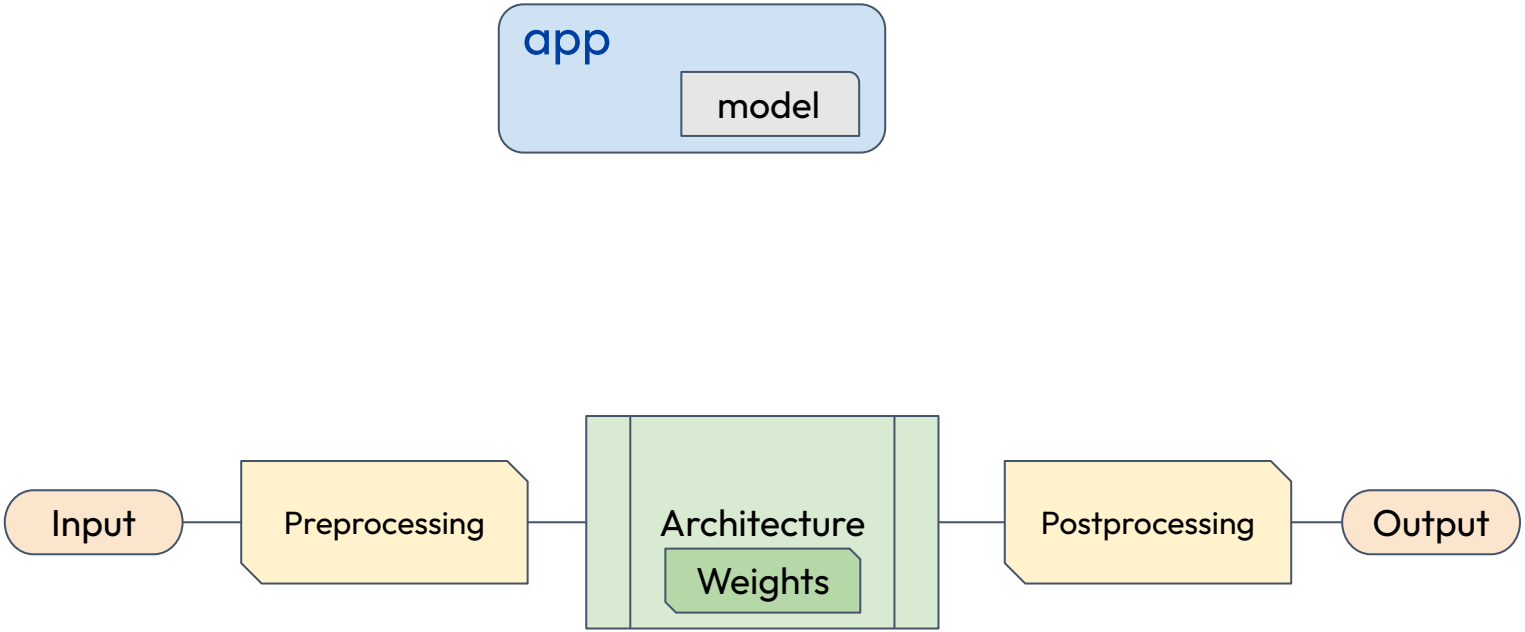
¿Cómo despliego mi modelo?

- Versionado del modelo (reproducibilidad)

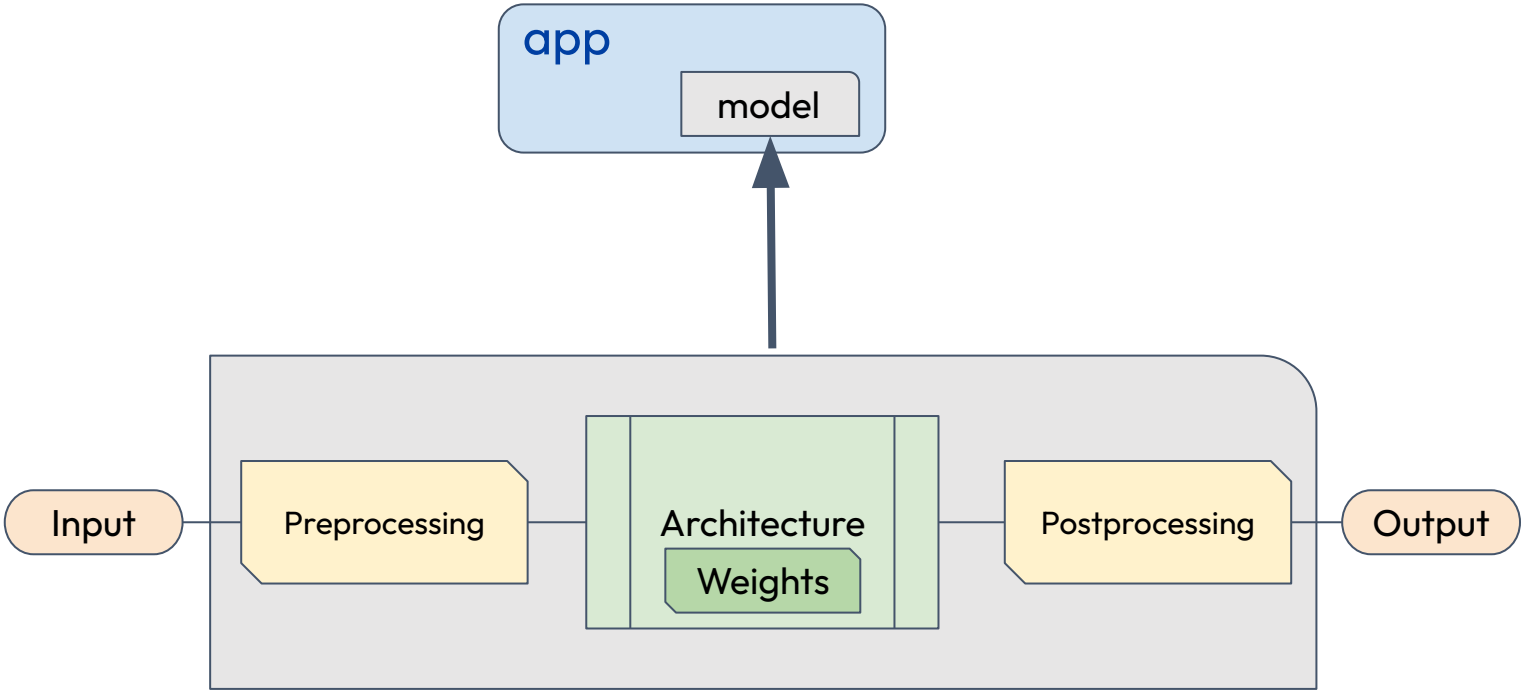
Nuestra definición de modelo



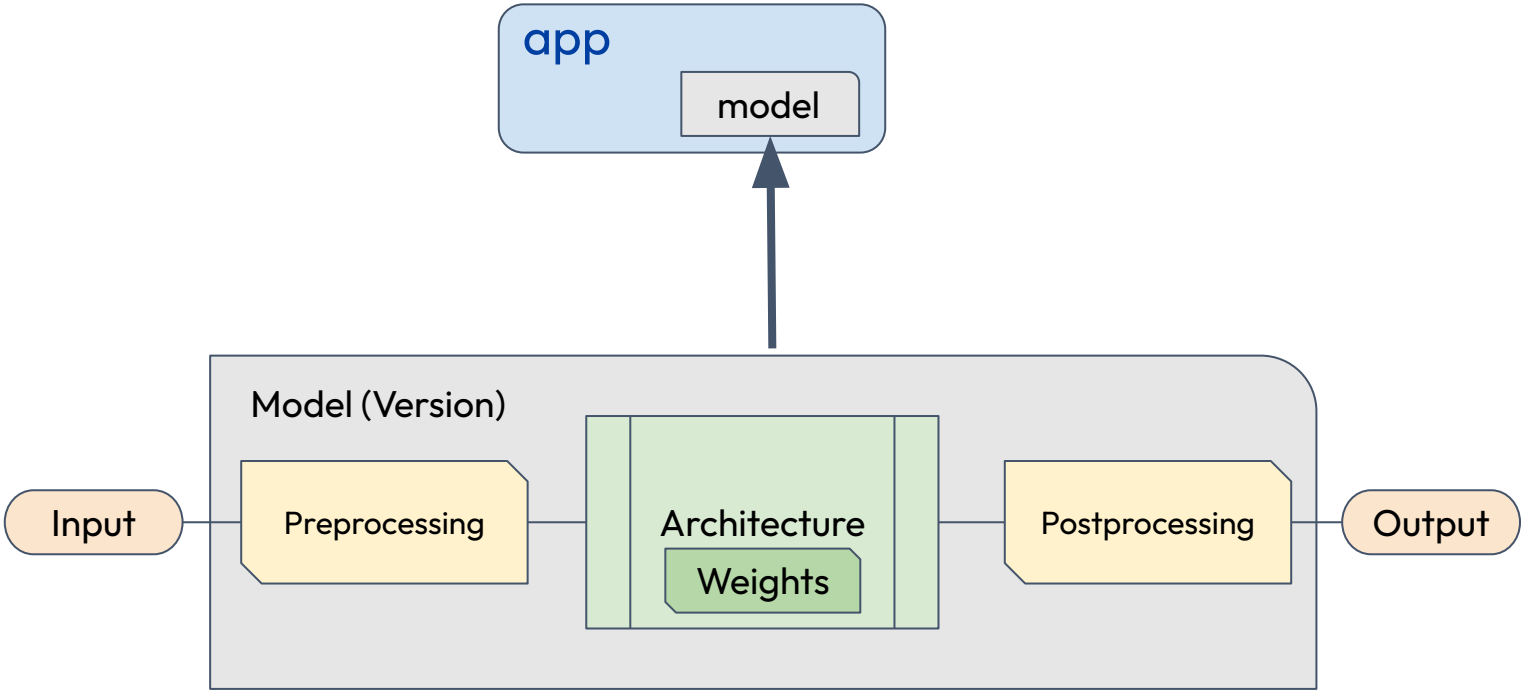
Nuestra definición de modelo



Nuestra definición de modelo



Nuestra definición de modelo

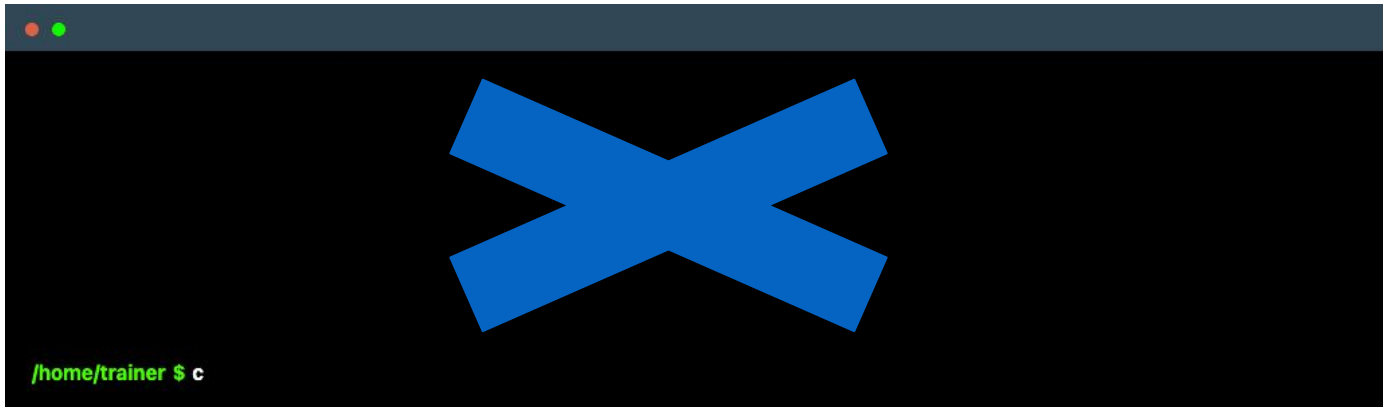


¿Cómo despliego mi modelo?

- Versionado del modelo (reproducibilidad)
- Automatizado (sin errores)

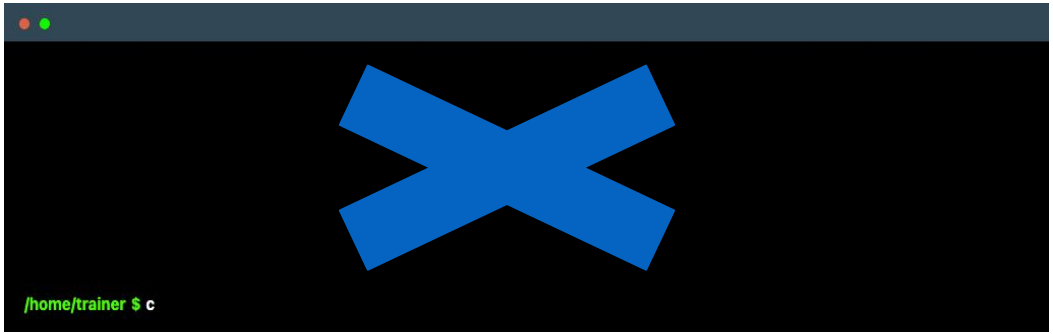
¿Cómo despliego mi modelo?

- Versionado del modelo (reproducibilidad)
- Automatizado (sin errores)



¿Cómo despliego mi modelo?

- Versionado del modelo (reproducibilidad)
- Automatizado (sin errores)
  - Scripting
  - Github Actions
  - MLOps Tools

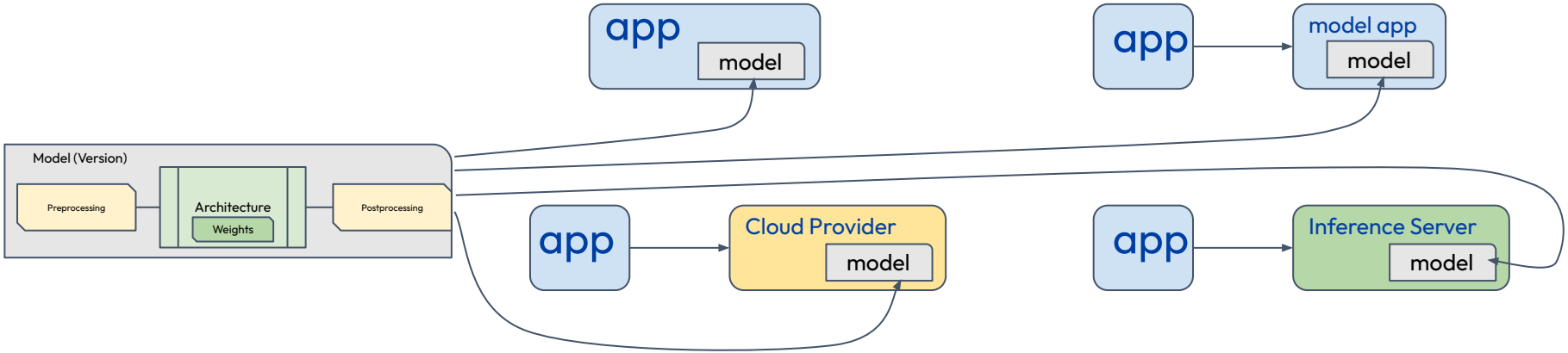


¿Cómo despliego mi modelo?

- Versionado del modelo (reproducibilidad)
- Automatizado (sin errores)
- Transparente para la aplicación (agilidad)

¿Cómo despliego mi modelo?

- Versionado del modelo (reproducibilidad)
- Automatizado (sin errores)
- Transparente para la aplicación (agilidad)



Erase una vez …

Escenario real en el que debido a un problema en la aplicación, ha sido necesario hacer un rollback a una versión anterior

Un modelo que estaba en producción desde hace meses deja de funcionar correctamente

No puede ser… Ayer funcionaba perfectamente y no hemos tocado nada del modelo

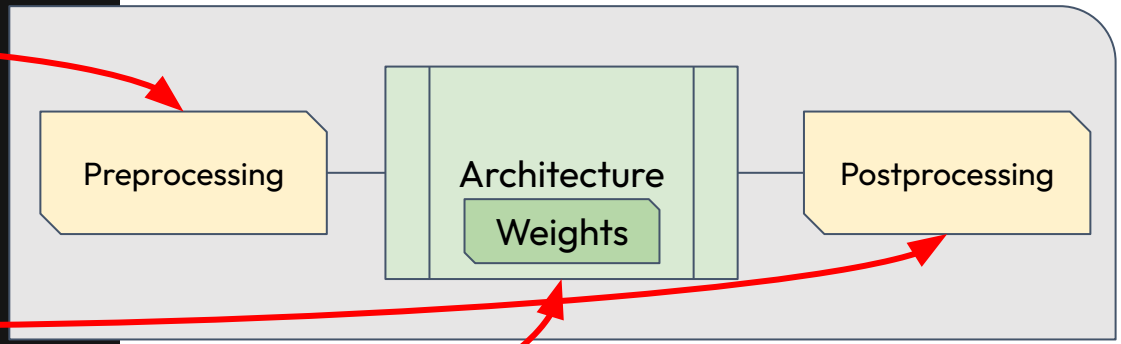
Erase una vez ...

```python
class ClassifyController(UseCase):
    @staticmethod
    def build(model_name: str, preprocessor_name: str, postprocessor_name: str ) -> Classifier:
        if preprocessor_name == "FakeVSReal":
            preprocessor = StandardizeInput()
        elif preprocessor_name == "FakeVSRealV1":
            preprocessor = NormalizeInput(mean=0.5, std=1.0)
        elif preprocessor_name == "FakeVSRealV2":
            preprocessor = NormalizeInput(mean=127.5, std=127.5)
        elif preprocessor_name == "FakeVSRealV2a":
            preprocessor = StandardizeInput2()
        else:
            raise ValueError(f"Unknown preprocessor {preprocessor_name}")

        if postprocessor_name == "FakeVSReal":
            postprocessor = FakeVSRealPostprocessor(th=0.234)
        elif postprocessor_name == "FakeVSRealV1":
            postprocessor = FakeVSRealPostprocessor(th=0.766)
        elif postprocessor_name == "FakeVSRealV2":
            postprocessor = FakeVSRealPostprocessor()
        elif postprocessor_name == "FakeVSRealV2a":
            postprocessor = FakeVSRealPostprocessor(invert=True)
        else:
            raise ValueError(f"Unknown postprocessor {postprocessor_name}")

        if model_name == "FakeVSReal":
            model = load_model("resources/models/fake_vs_real/20220417_fake_vs_real_arch_1")
        elif model_name == "FakeVSRealV1":
            model = load_model("resources/models/fake_vs_real/20220422_fake_vs_real_arch_1")
        elif model_name == "FakeVSRealV2":
            model = load_model("resources/models/fake_vs_real/20220422_fake_vs_real_arch_2")
        elif model_name == "FakeVSRealV2a":
            model = load_model("resources/models/fake_vs_real/20220425_fake_vs_real_arch_2")
        else:
            raise ValueError(f"Unknown postprocessor {model_name}")

        return FakeVSRealClassifier(model=model,
                                    preprocessor=preprocessor,
                                    postprocessor=postprocessor)
```



Erase una vez ...

```python
class ClassifyController(UseCase):
    @staticmethod
    def build(model_name: str, preprocessor_name: str, postprocessor_name: str ) -> Classifier:
        if preprocessor_name == "FakeVSReal":
            preprocessor = StandardizeInput()
        elif preprocessor_name == "FakeVSRealV1":
            preprocessor = NormalizeInput(mean=0.5, std=1.0)
        elif preprocessor_name == "FakeVSRealV2":
            preprocessor = NormalizeInput(mean=127.5, std=127.5)
        elif preprocessor_name == "FakeVSRealV2a":
            preprocessor = StandardizeInput2()
        else:
            raise ValueError(f"Unknown preprocessor {preprocessor_name}")

        if postprocessor_name == "FakeVSReal":
            postprocessor = FakeVSRealPostprocessor(th=0.234)
        elif postprocessor_name == "FakeVSRealV1":
            postprocessor = FakeVSRealPostprocessor(th=0.766)
        elif postprocessor_name == "FakeVSRealV2":
            postprocessor = FakeVSRealPostprocessor()
        elif postprocessor_name == "FakeVSRealV2a":
            postprocessor = FakeVSRealPostprocessor(invert=True)
        else:
            raise ValueError(f"Unknown postprocessor {postprocessor_name}")

        if model_name == "FakeVSReal":
            model = load_model("resources/models/fake_vs_real/20220417_fake_vs_real_arch_1")
        elif model_name == "FakeVSRealV1":
            model = load_model("resources/models/fake_vs_real/20220422_fake_vs_real_arch_1")
        elif model_name == "FakeVSRealV2":
            model = load_model("resources/models/fake_vs_real/20220422_fake_vs_real_arch_2")
        elif model_name == "FakeVSRealV2a":
            model = load_model("resources/models/fake_vs_real/20220425_fake_vs_real_arch_2")
        else:
            raise ValueError(f"Unknown postprocessor {model_name}")

        return FakeVSRealClassifier(model=model,
                                    preprocessor=preprocessor,
                                    postprocessor=postprocessor)
```

Alice

Objetivos

⬆️ Escalabilidad ⬆️ Resiliencia ⬇️ Coste

La evaluación incorrecta de nuestros modelos puede generar un alto coste económico 💸.

Automatizar y versionar el despliegue de modelos **acelera el flujo de desarrollo**, evitando contratiempos y permitiendo mantener el **foco** en la **complejidad intrínseca del problema a resolver**. 

Un buen diseño y arquitectura hace que tus software sea mantenible, flexible y confiable (🙌 también en experimentación y MLOps 🙏).

Objetivos

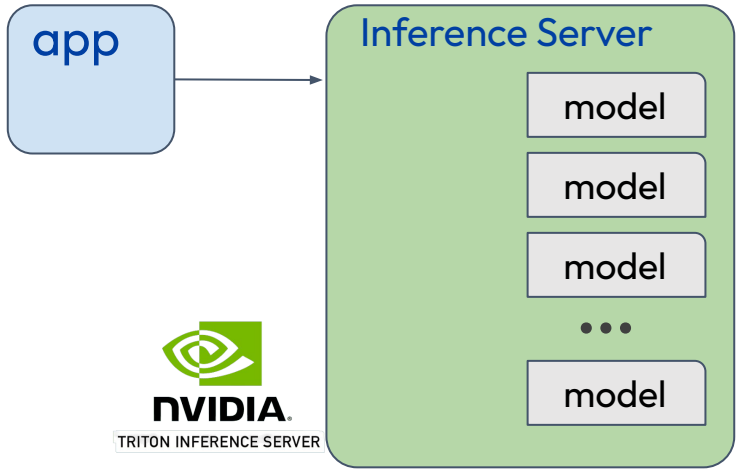
⬆️ Escalabilidad ⬆️ Resiliencia ⬇️ Coste

La evaluación incorrecta de nuestros modelos puede generar un alto coste económico 💸.

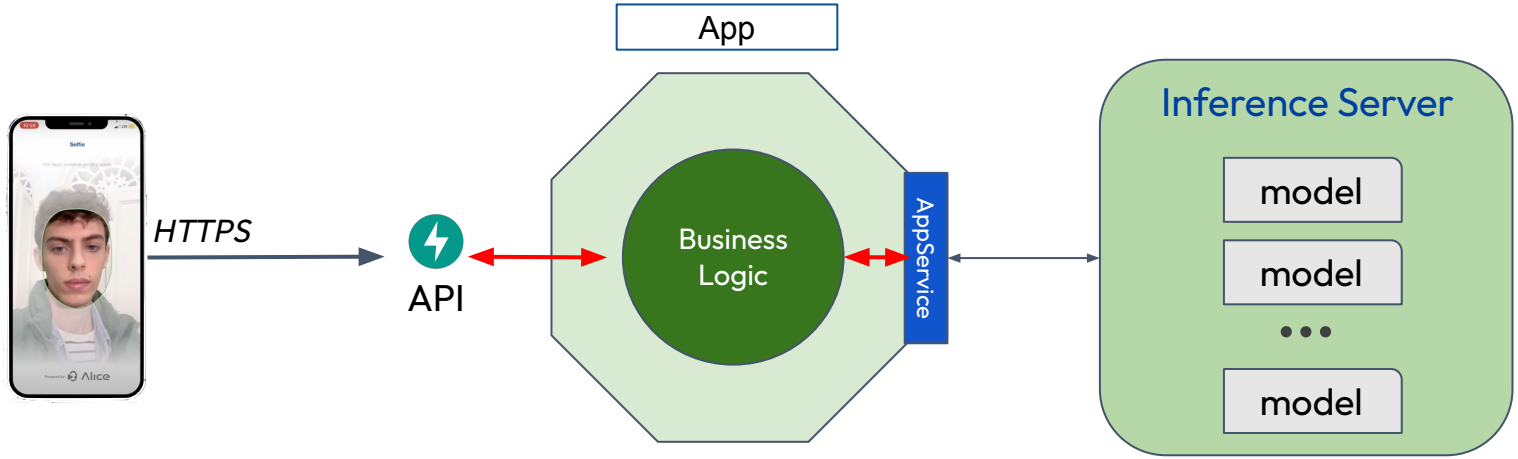
Automatizar y versionar el despliegue de modelos acelera el flujo de desarrollo, evitando contratiempos y permitiendo mantener el foco en la complejidad intrínseca del problema a resolver. 

Un buen diseño y arquitectura hace que tus software sea mantenible, flexible y confiable (🙌 también en **experimentación** y **MLOps** 🙏).

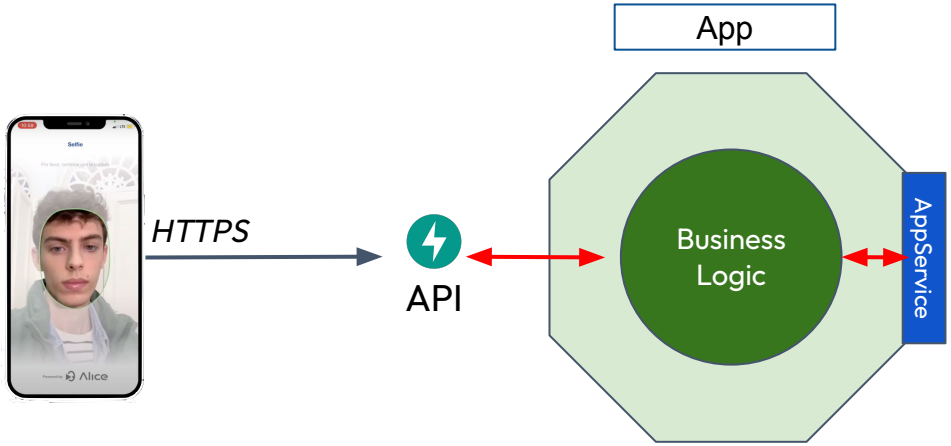
Buen diseño en todas la capas [ml 🙏]



Buen diseño en todas la capas [ml 🙏]



Buen diseño en todas la capas [ml 🙏]



🎯 Dominio

¿Este selfie es fake o es real?

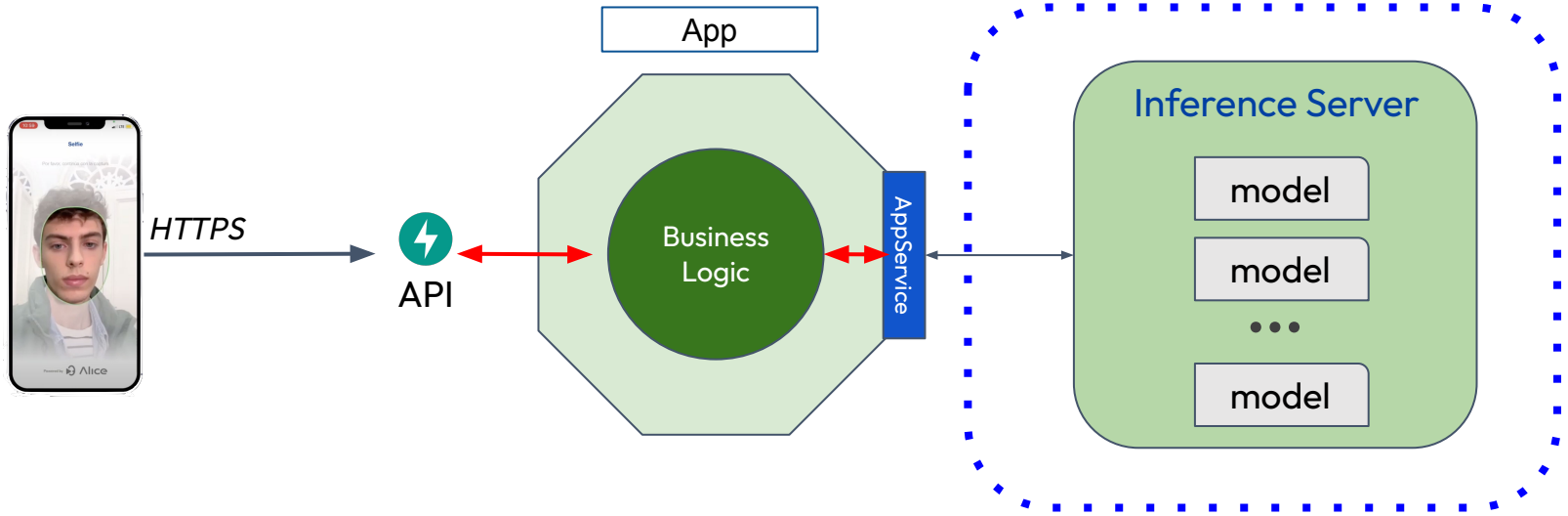
🤝 Lógica de negocio

Ejecución de un modelo de clasificación que diferencia entre selfies fake y reales

🏭 Infraestructura

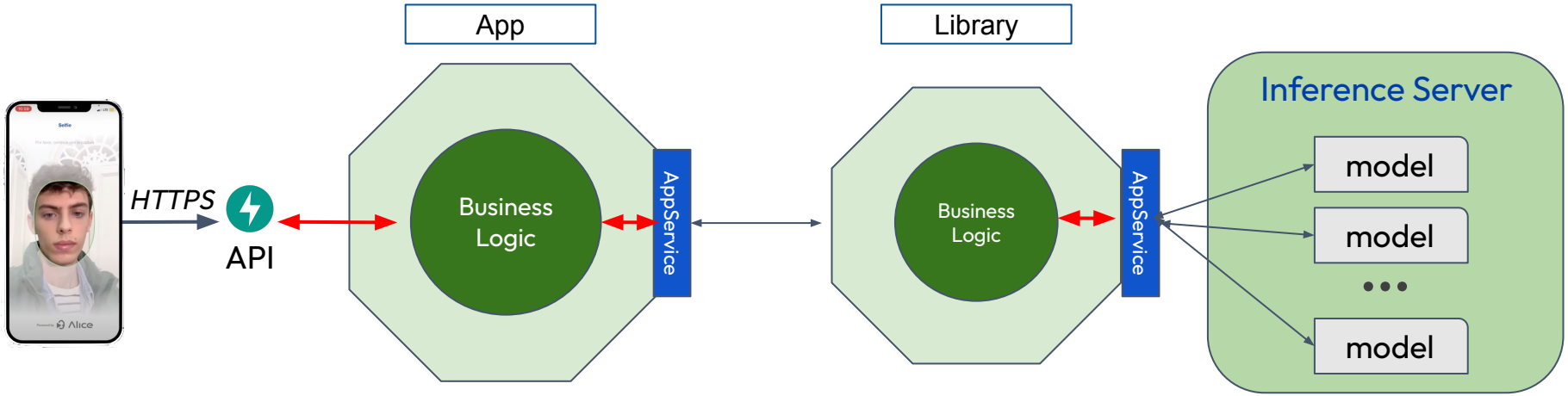
Inferencia del modelo

Buen diseño en todas la capas [ml 🙏]



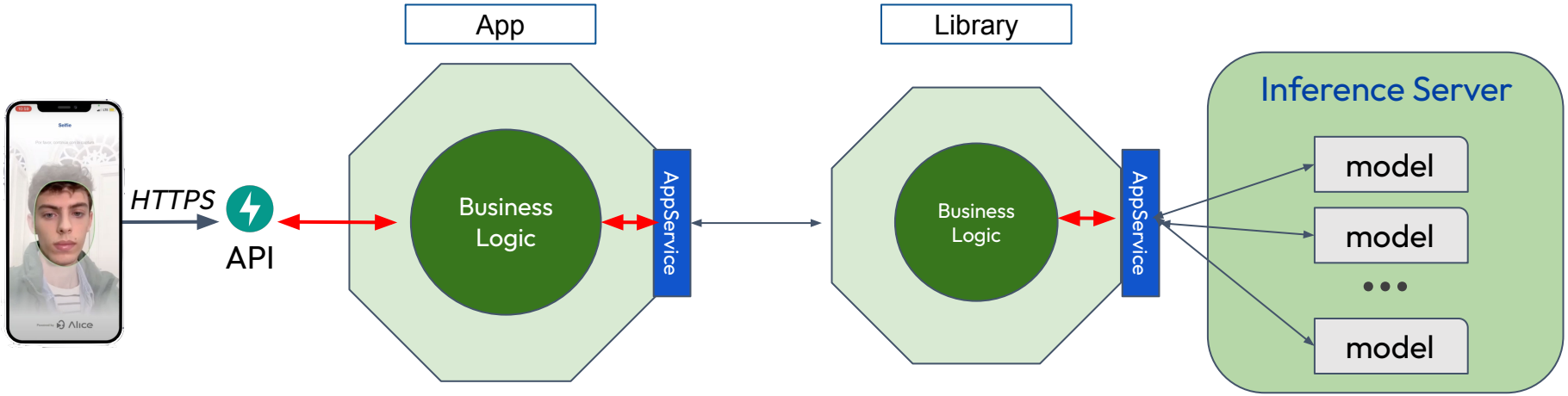
¿ 🎯 ?

Buen diseño en todas la capas [ml 🙏]



Buen diseño en todas la capas [ml 🙏]

Buen diseño en todas la capas [ml 🙏]



Buen diseño en todas la capas [ml 🙏]

🎯 Dominio

Inferencia de un modelo

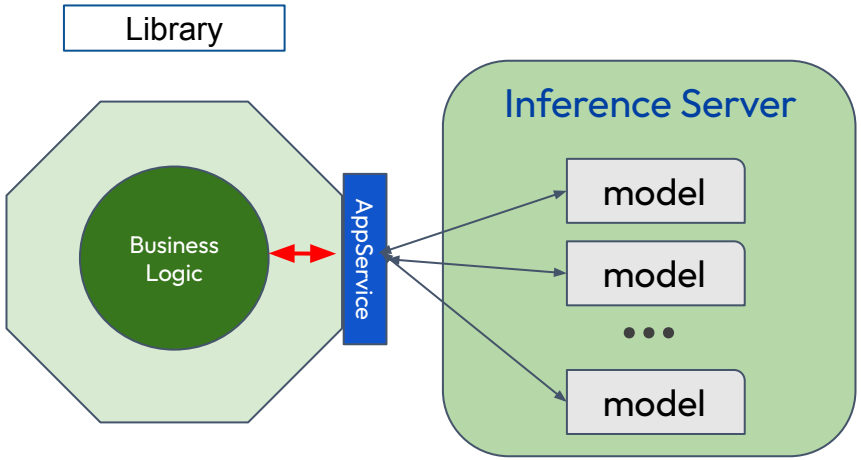
🤝 Lógica de negocio

Tarea de clasificación

Tamaño de entrada (h, w, c)

Normalizado con mean=0.5 y std=0.5

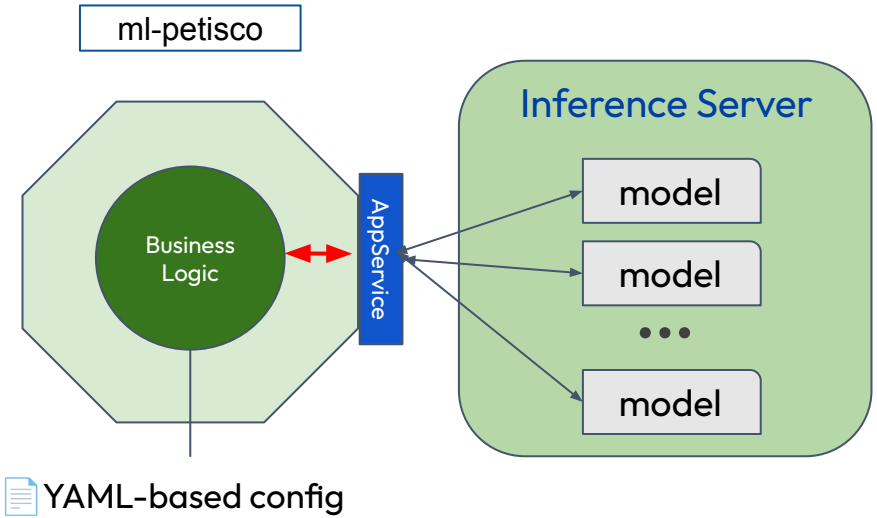
Fake sí salida > 0.5. Si no, es real



🏭 Infraestructura

Ejecuta el modelo X con los recursos Y

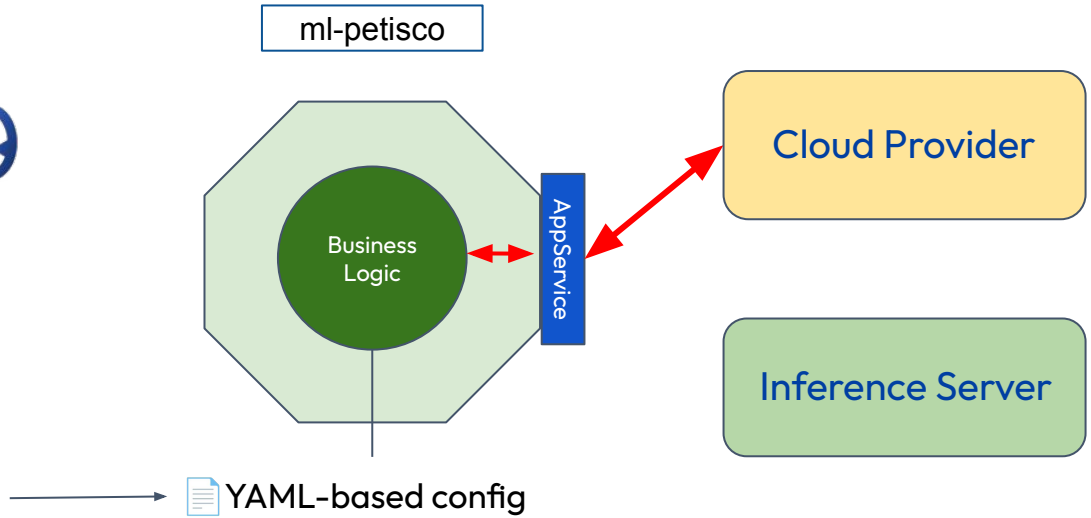
Buen diseño en todas la capas [ml 🙏]



Buen diseño en todas la capas [ml 🙏]

ml petisco

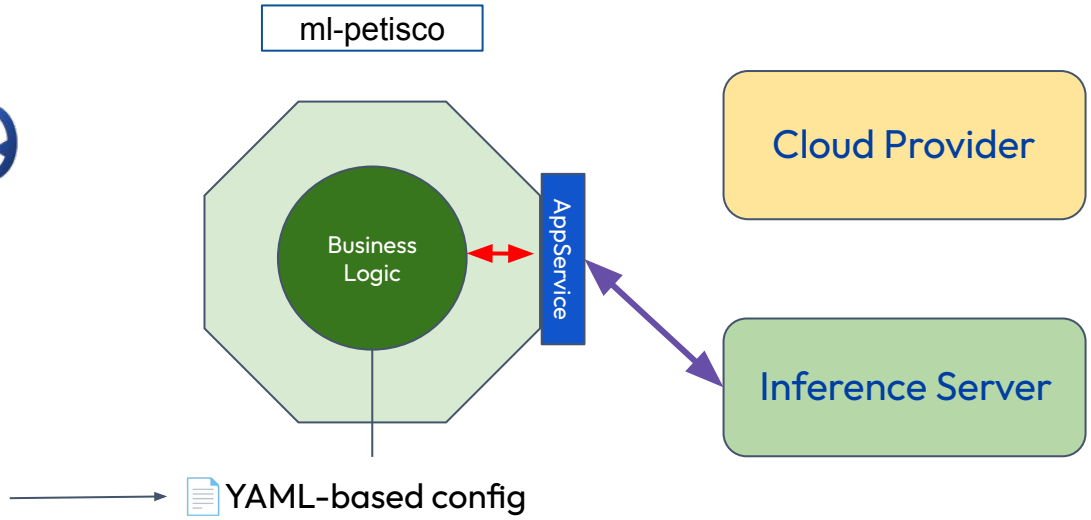
```yaml
- id: selfie_fake_vs_real
  encoder:
    name: BufferMediaImageEncoder
    args:
      image_format: ".png"
      resize_mode: "fix_size"
      size: 224
      resize_type: 1
  model:
    runner:
      type: vertexai
      model: "220784813566623"
    parameters:
      return_with_threshold: True
      postprocess:
        with_softmax: True
  formatter:
    name: ClassificationFormatter
```



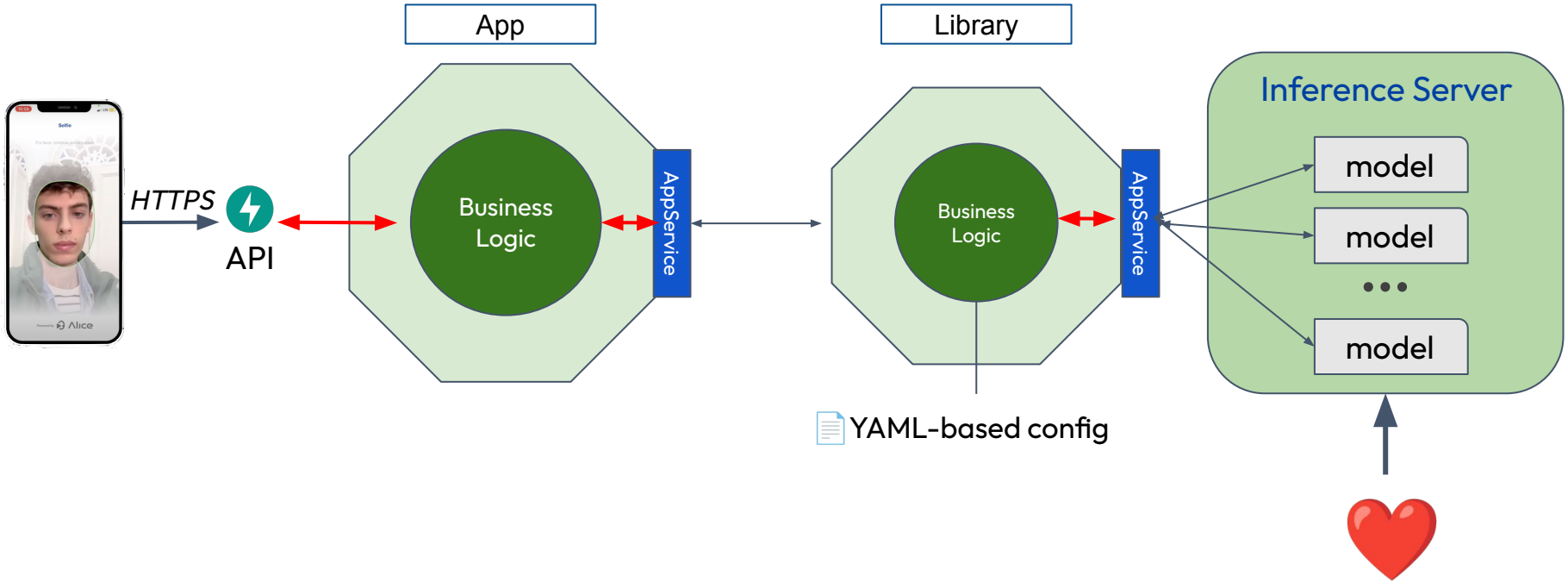
Buen diseño en todas la capas [ml 🙏]

ml petisco

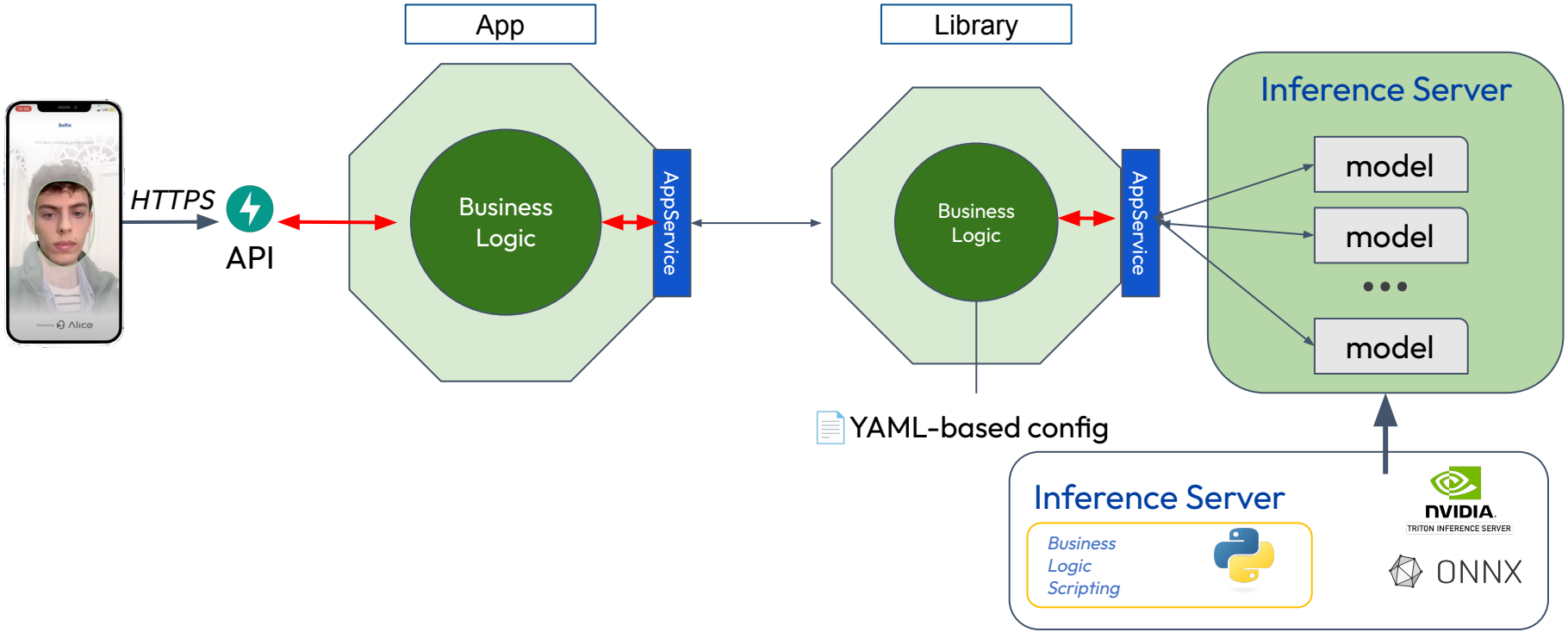
```yaml
- id: selfie_fake_vs_real
  encoder:
    name: BufferMediaImageEncoder
    args:
      image_format: ".png"
      resize_mode: "fix_size"
      size: 224
      resize_type: 1
  model:
    runner:
      type: triton
      model: selfie_fake_vs_real_mbnet
      meta:
        bls_type: classification
        bls_version: "1"
        model_version: "3"
    parameters:
      return_with_threshold: True
      postprocess:
        with_softmax: True
  formatter:
    name: ClassificationFormatter
```



Buen diseño en todas la capas [ml 🙏]



Buen diseño en todas la capas [ml 🙏]



Objetivos

⬆️ Escalabilidad ⬆️ Resiliencia ⬇️ Coste

La **evaluación incorrecta** de nuestros modelos puede generar un **alto coste económico** 💸.

Automatizar y versionar el despliegue de modelos **acelera el flujo de desarrollo**, evitando contratiempos y permitiendo mantener el **foco** en la **complejidad intrínseca del problema a resolver**. 

Un buen diseño y arquitectura hace que tus software sea mantenible, flexible y confiable (🙌 también en **experimentación** y **MLOps** 🙏).


peacock