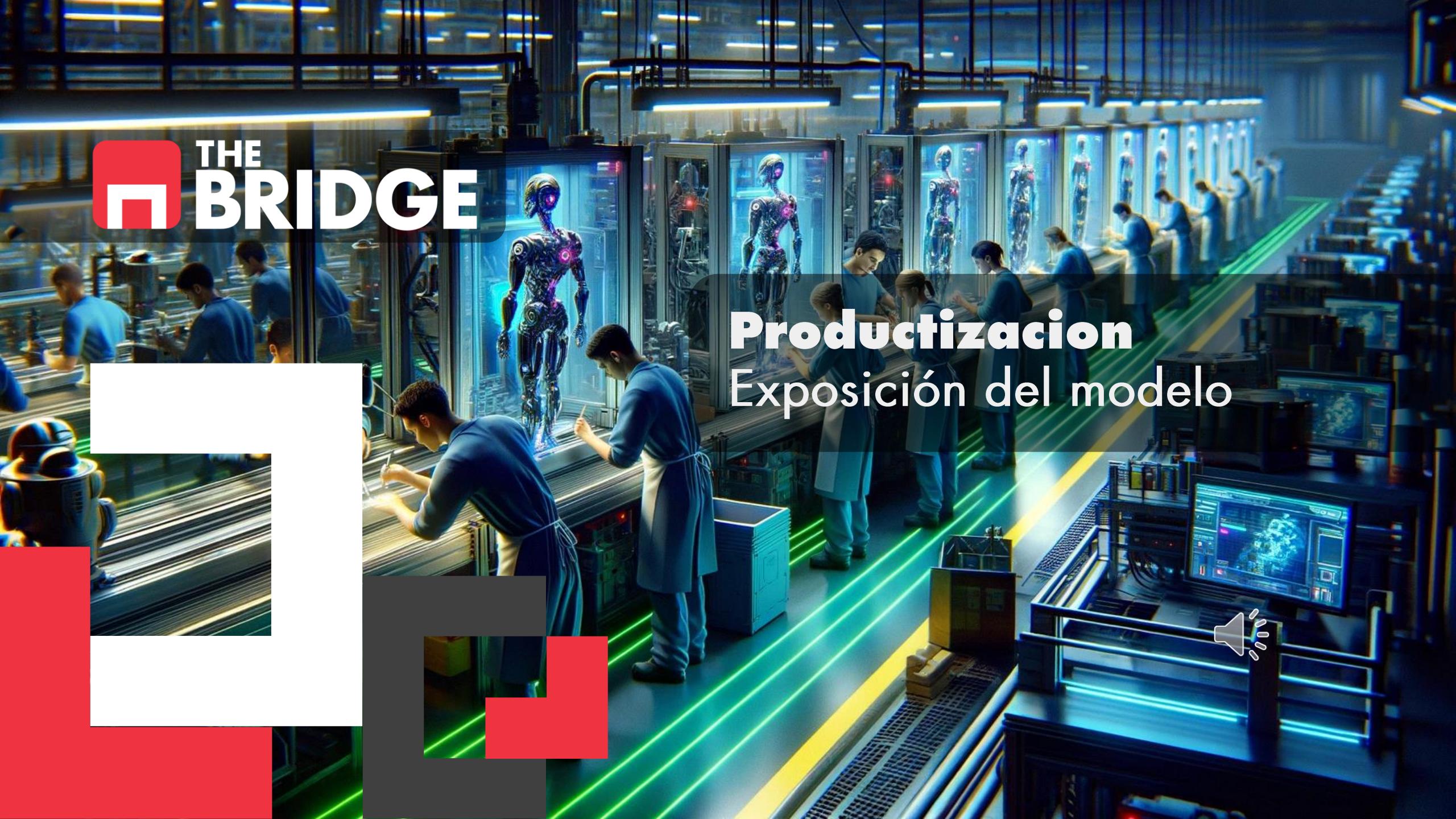


Productizacion

Exposición del modelo

Exposición del modelo

SERVIDOR DEDICADO

GRANJA/CLUSTER
(también CPD)

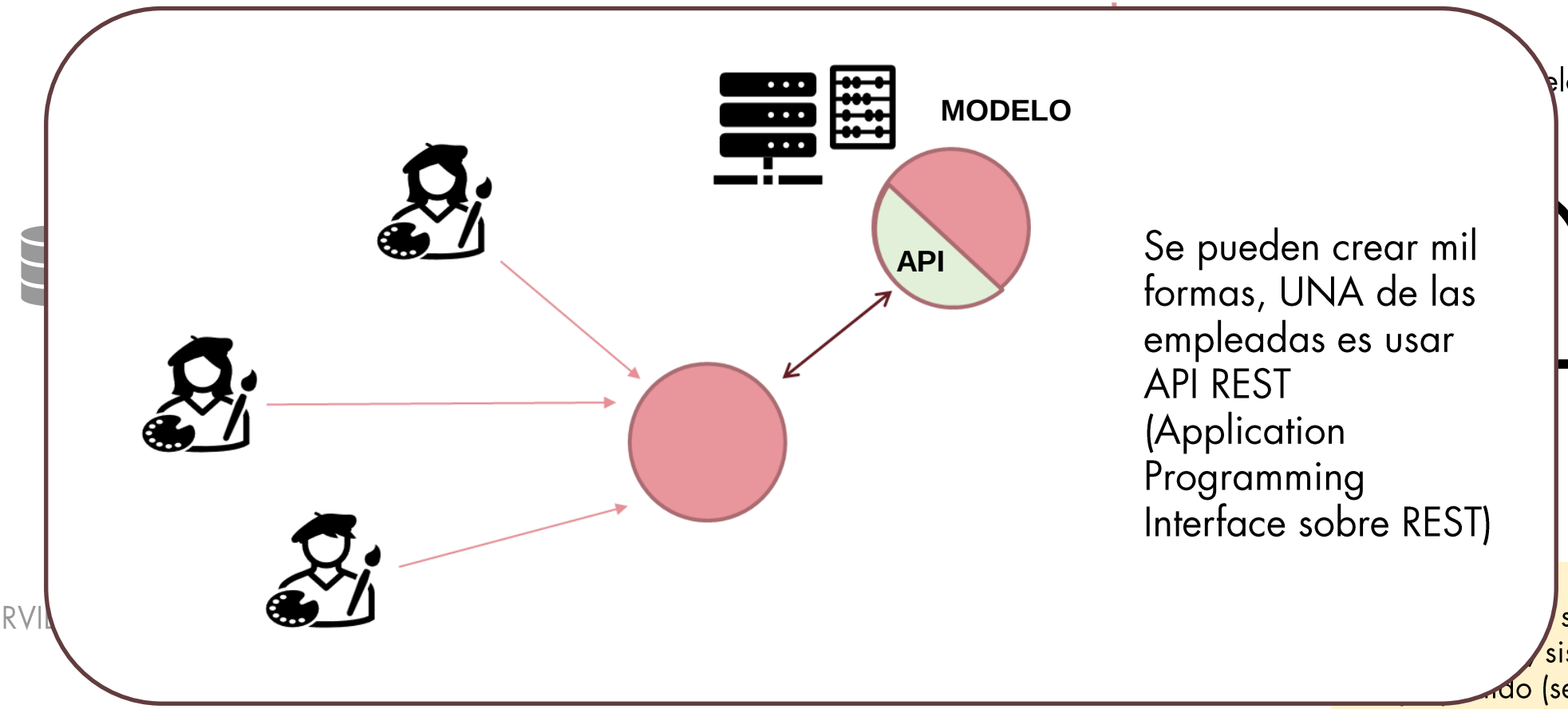
2. Ejecutamos el modelo sobre esos datos (*)

1. Se nos proporcionan los datos en una petición a medida que el cliente navega por la web

3. El resultado se expone/envía al programa/sistema que nos lo haya pedido (se utilizan mecanismos de interconexión como APIs, sockets, etc… Nosotros vamos a ver APIs)

(*) Es habitual tener las "salidas" precalculadas mediante una ejecución batch y que aquí nuestro sistema simplemente "la busque" en el fichero de salida del modelo

Exposición del modelo: Usamos APIs



Se pueden crear mil formas, UNA de las empleadas es usar API REST (Application Programming Interface sobre REST)

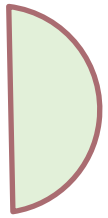
(*) Es habitual tener las "salidas" precalculadas mediante una ejecución batch y que aquí nuestro sistema simplemente "la busque" en el fichero de salida del modelo

1. Se nos proporcionan los datos en una petición a medida que el cliente navega por la web

se expone/envía sistema que nos lo do (se utilizan mecanismos de interconexión como APIs, sockets, etc… Nosotros vamos a ver APIs)

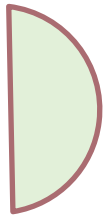
Exposición del modelo: API Rest

- El **"Endpoint"** tiene la pinta de una dirección web (pepito.apis.com:/modelo_a o 112.12.12.113:2345/modelo_b)



Exposición del modelo: API Rest

- El **"Endpoint"** tiene la pinta de una dirección web (pepito.apis.com:/modelo_a o 112.12.12.113:2345/modelo_b)



- El **método** (los más utilizados son GET, PUSH, PUT, UPDATE, DELETE…)

Exposición del modelo: API Rest

- El **"Endpoint"** tiene la pinta de una dirección web (pepito.apis.com:/modelo_a o 112.12.12.113:2345/modelo_b)

- El **método** (los más utilizados son GET, PUSH, PUT, UPDATE, DELETE…)

- Parámetros en la "dirección": pepito.apis.com:/modelo_a?param1=12&param2="OCEAN"

Exposición del modelo: API Rest

- El **"Endpoint"** tiene la pinta de una dirección web (pepito.apis.com:/modelo_a o 112.12.12.113:2345/modelo_b)

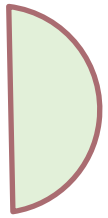
- El **método** (los más utilizados son GET, PUSH, PUT, UPDATE, DELETE…)

- Parámetros en la "dirección": pepito.apis.com:/modelo_a?param1=12&param2="OCEAN"

- Parámetros en el cuerpo de la petición: Otra forma de pasarle los parámetros es crear una petición con un programa y en esa petición incluir un "diccionario" en el que las claves son los nombres de los parámetros ("param1", "param2", y los valores asociados a esas claves, los valores que queremos darle a los parámetros)

Exposición del modelo: API Rest

- El **"Endpoint"** tiene la pinta de una dirección web (pepito.apis.com:/modelo_a o 112.12.12.113:2345/modelo_b)



- El **método** (los más utilizados son GET, PUSH, PUT, UPDATE, DELETE…)

- Parámetros en la "dirección": pepito.apis.com:/modelo_a?param1=12&param2="OCEAN"

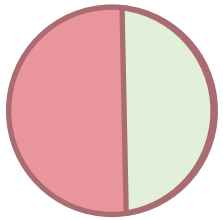
- Parámetros en el cuerpo de la petición: Otra forma de pasarle los parámetros es crear una petición con un programa y en esa petición incluir un "diccionario" en el que las claves son los nombres de los parámetros ("param1", "param2", y los valores asociados a esas claves, los valores que queremos darle a los parámetros)

- Sistema de autenticación (API Key)

Exposición del modelo: API Rest

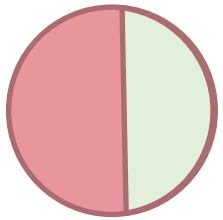
```
import requests

root_path = "http://127.0.0.1:4000" # localhost:5000, it depends

def send_api_requests(endpoint, content=None, params = None, method = "get"):
    url = root_path + "/" + endpoint

    if method == "get":
        response = requests.get(url, json = content, params= params)
    elif method == "post":
        response = requests.post(url, json= content, params = params)

    print("Status code:", response.status_code)
    print("Headers:", response.headers)
    print("Content:", response.text)
    print("Type content:", type(response.text))
    print("Content json:", response.json())

parametros = {'name': 'Pedro'}
peticion = {
    "TV": 230,
    "radio": 12,
    "newspaper": 1200
}
send_api_requests("/predict", content = parametros, method = "post")
```

Exposición del modelo: Construcción de la API



- Alguien tiene desarrollar un programa que conecte las APIs (routing) a las tareas que queremos que se desarrollen

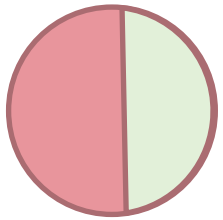
Exposición del modelo: Construcción de la API



- Alguien tiene desarrollar un programa que conecte las APIs (routing) a las tareas que queremos que se desarrollen

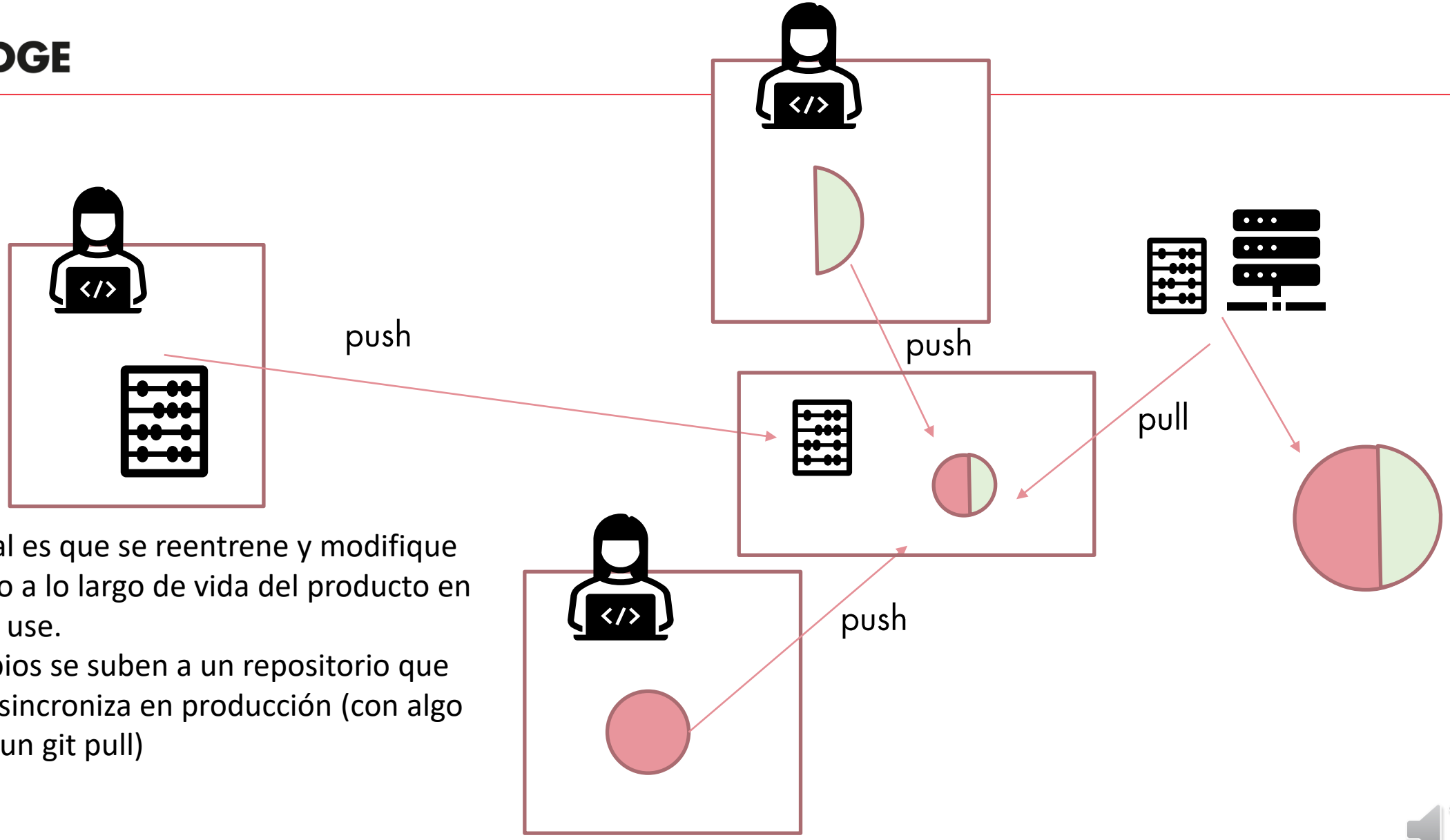
- Alguien además tendrá que desarrollar el código que haga esas tareas que permiten la funcionalidad de la API

Exposición del modelo: Construcción de la API



- Alguien tiene desarrollar un programa que conecte las APIs (routing) a las tareas que queremos que se desarrollen
- Alguien además tendrá que desarrollar el código que haga esas tareas que permiten la funcionalidad de la API
- Es interesante notar que el código de exposición de la API puede estar en un lenguaje y el de desarrollo en otro

```
from flask import Flask, request, jsonify
from data.datos_dummy import books

app = Flask(__name__)
app.config['DEBUG'] = True



@app.route('/', methods=['GET'])
def home():
    return "<h1>Distant Reading Archive</h1><p>This site is a prototype API for distant reading of science fiction
novels.</p>"

# GET Display all the books
@app.route('/api/v1/resources/books/all', methods=['GET'])
def api_all():
    return jsonify(books)

# GET ?id=x -> Display the book with specified id
@app.route('/api/v1/resources/books', methods=['GET'])
def api_id():

    if request.method == 'GET':
        if 'id' in request.args:
            id = int(request.args['id'])
        else:
            return "Error: No id field provided. Please specify an id."

        results = []
        for book in books:
            if book['id'] == id:
                results.append(book)

        return jsonify(results)
```



Lo normal es que se reentrene y modifique el modelo a lo largo de vida del producto en el que se use.
Los cambios se suben a un repositorio que luego se sincroniza en producción (con algo similar a un git pull)

Los desarrolladores del sistema final y de las APIS actúan igual, van haciendo modificaciones que se suben a un repositorio y de ahí se despliegan periódicamente


THE
BRIDGE