

# Core Engine

[Website](#) · [GitHub](#) · [Documentation](#) · [Guide](#)

Swiss AI Center contributors

This work is licensed under the [AGPL 3.0](#) license.

**swiss  center**

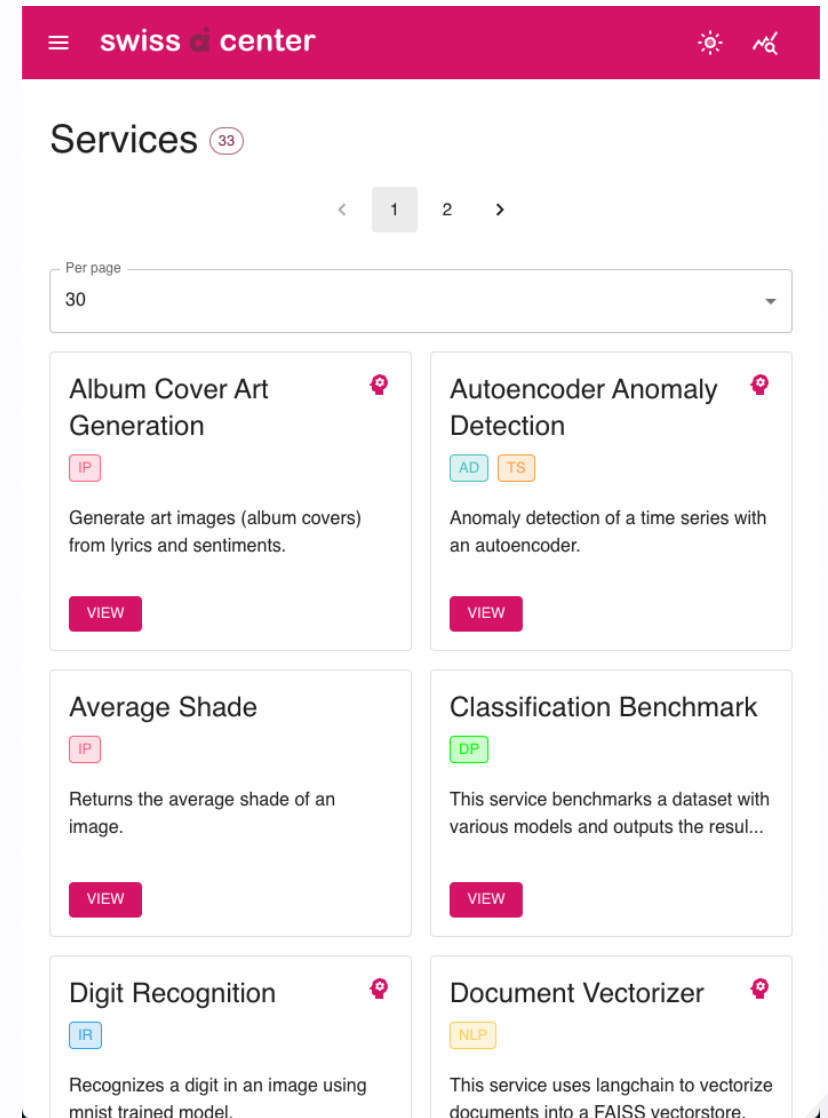
# INTRODUCTION

# SWISS AI CENTER

- **Five schools from the HES-SO** (HEIG-VD, HEIA-FR, HE-Arc, HEVS and HEPIA)
- Project called **Centre Suisse d'Intelligence Artificiel à destination des PME (CSIA-PME)**, also known as the **Swiss AI Center**.
- The center's mission is to **accelerate the adoption of artificial intelligence in the digital transition of Swiss SMEs**.

# FEATURES

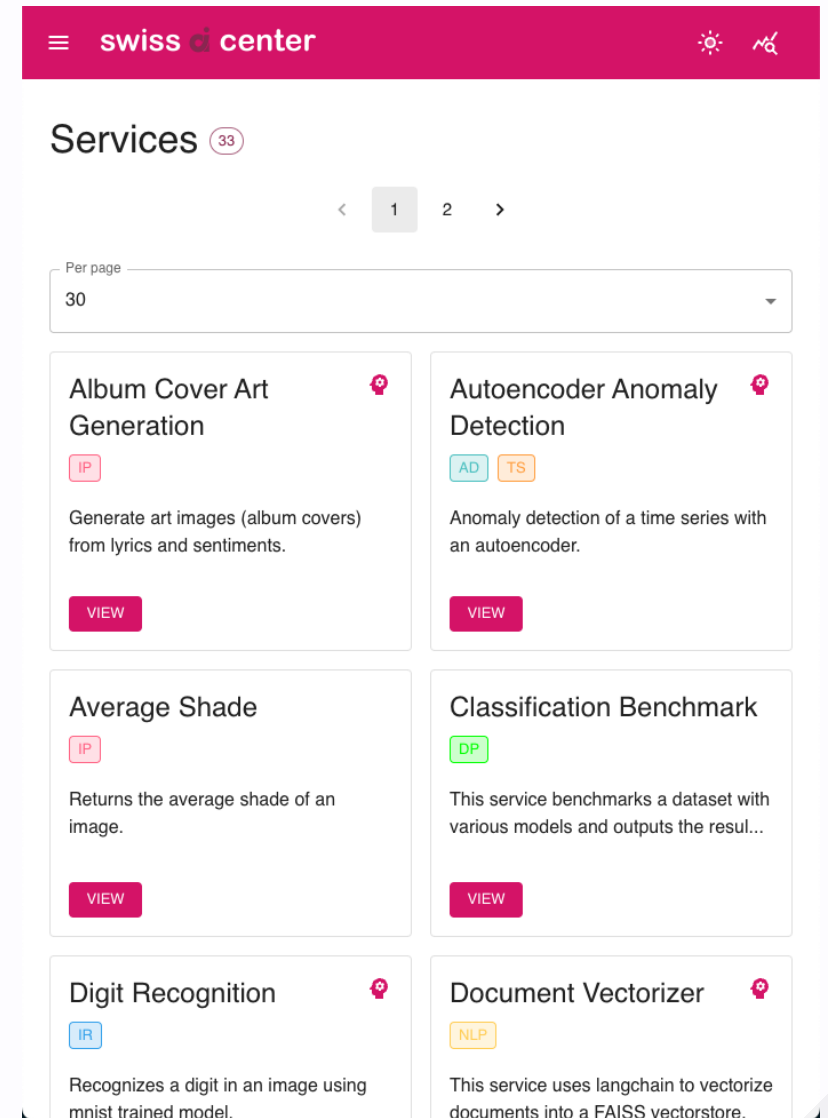
- Centralize **ML services**
- Unify ML services specifications with a **HTTP REST API**
- **Orchestrate** multiple ML services **through pipelines**



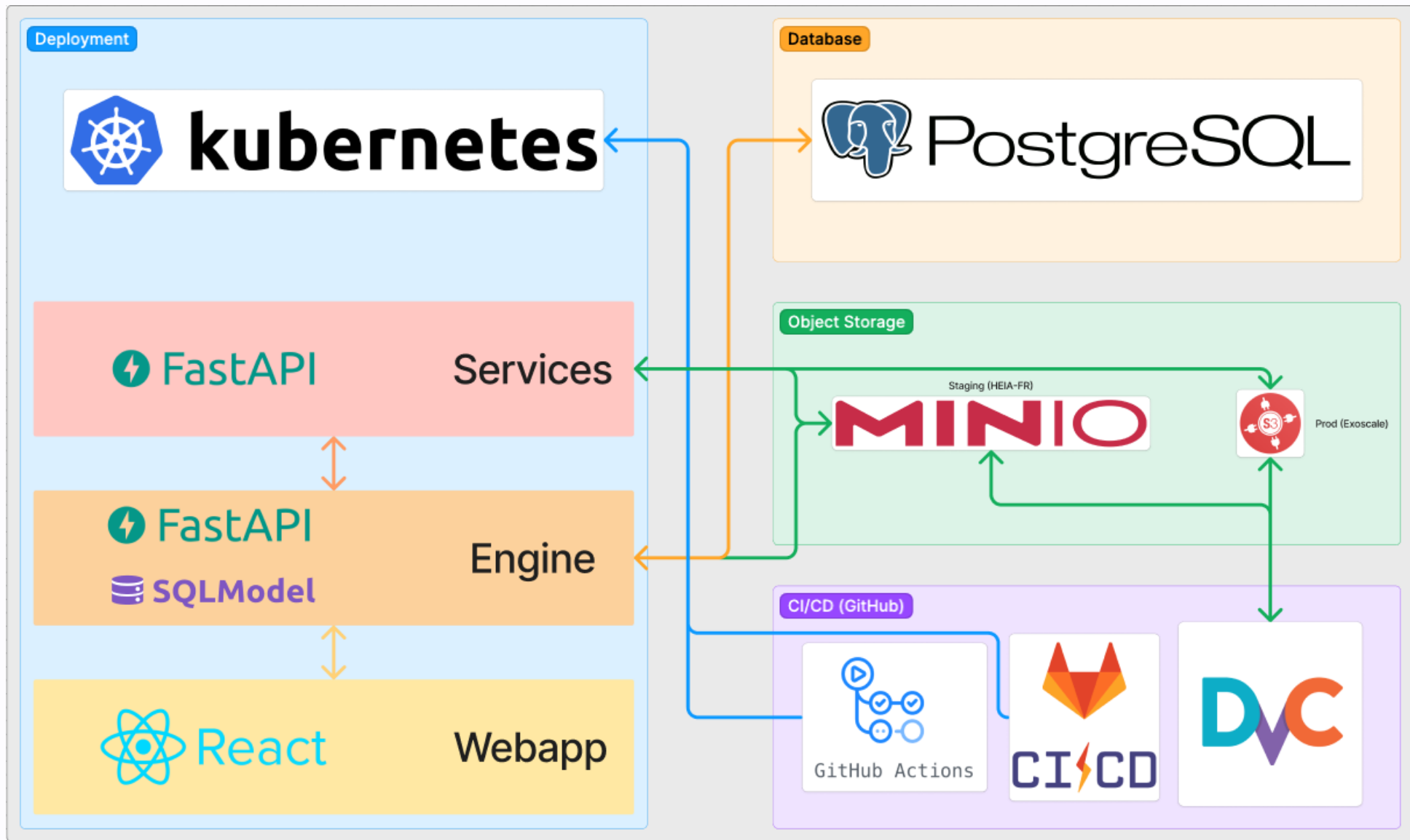
The screenshot shows the 'Services' page of the Swiss AI Center. The header is a dark red bar with the 'swiss ai center' logo and navigation icons. Below the header, the word 'Services' is followed by a count of 33 in a red circle. A pagination control shows '1' selected and '2' as an option. A 'Per page' dropdown menu is set to '30'. The main content area displays six service cards in a 3x2 grid. Each card includes a title, a category tag, a description, and a 'VIEW' button. The services are: Album Cover Art Generation (IP), Autoencoder Anomaly Detection (AD, TS), Average Shade (IP), Classification Benchmark (DP), Digit Recognition (IR), and Document Vectorizer (NLP).

Service Name	Category	Description
Album Cover Art Generation	IP	Generate art images (album covers) from lyrics and sentiments.
Autoencoder Anomaly Detection	AD, TS	Anomaly detection of a time series with an autoencoder.
Average Shade	IP	Returns the average shade of an image.
Classification Benchmark	DP	This service benchmarks a dataset with various models and outputs the result...
Digit Recognition	IR	Recognizes a digit in an image using mnist trained model.
Document Vectorizer	NLP	This service uses langchain to vectorize documents into a FAISS vectorstore.

- Beautiful **frontend to visualize** services and pipelines
- Extensive **documentation** available
- Best practices regarding software development (code reviews, CI/CD)
- **Open source**



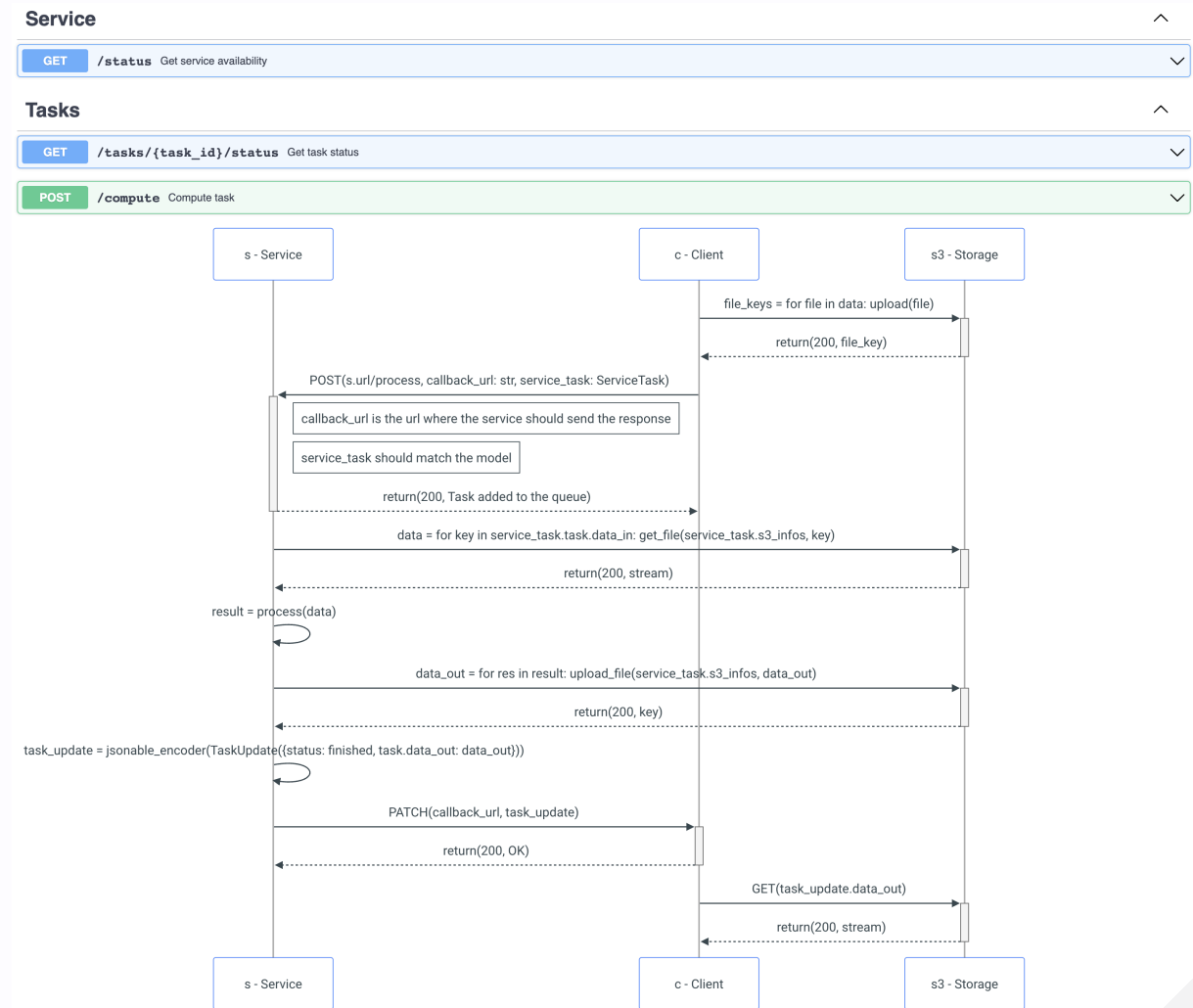
# INFRASTRUCTURE





# **SERVICE SPECIFICATION**

- Can be in **any language** that can implement a **REST API**
- Must have the **required routes** to be “engine” compliant
- The `/compute` route must accept the **“Task” model**
- Can have its own routes (for specific purposes)



# PIPELINE SPECIFICATION

## A JSON file containing base information:

- Name
- Slug
- Summary
- Description
- Input/Output
- Tags

```
{
  "name": "Face Blur",
  "slug": "face-blur",
  "summary": "Blur the faces in an image",
  "description": "Use Face Detection service to locate the faces in the image and send the bounding boxes to the Image Blur service to get the final result",
  "data_in_fields": [
    {
      "name": "image",
      "type": [
        "image/jpeg",
        "image/png"
      ]
    }
  ],
  "data_out_fields": [
    {
      "name": "result",
      "type": [
        "image/jpeg",
        "image/png"
      ]
    }
  ],
  "tags": [
    {
      "name": "Image Recognition",
      "acronym": "IR"
    },
    {
      "name": "Image Processing",
      "acronym": "IP"
    }
  ],
  "steps": [
    {
      "identifier": "face-detection",
      "needs": [],
      "inputs": ["pipeline.image"],
      "service_slug": "face-detection"
    },
    {
      "identifier": "image-blur",
      "needs": ["face-detection"],
      "condition": "len(face-detection.result['areas']) > 0",
      "inputs": ["pipeline.image", "face-detection.result"],
      "service_slug": "image-blur"
    }
  ]
}
```

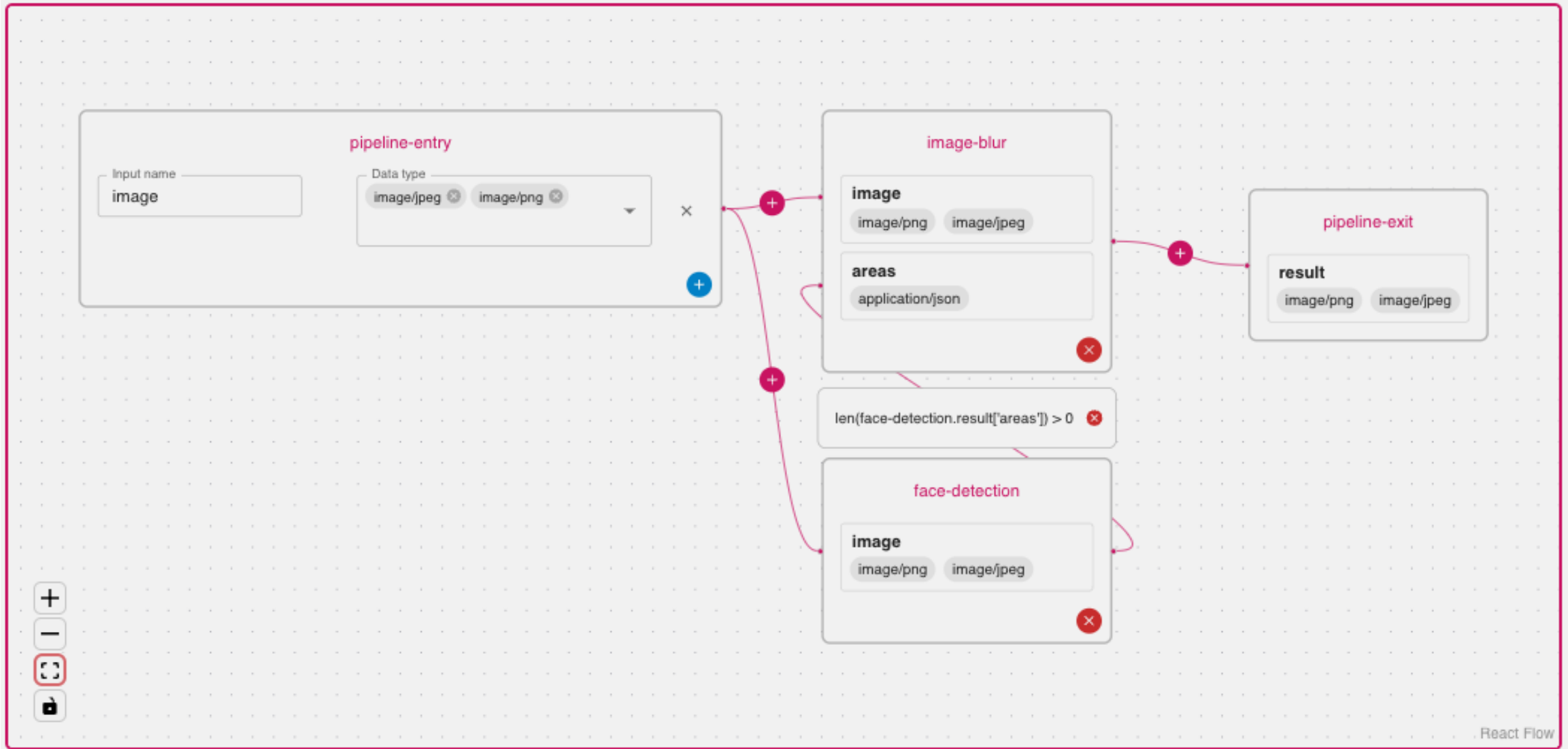
And a list of “Steps” representing the sequel of services to run with the following data:

- **Identifier** (used in the “needs”, “conditions” and “inputs” values)
- **Needs** (used to wait until all the services in the array finished their task)

```
{
  "name": "Face Blur",
  "slug": "face-blur",
  "summary": "Blur the faces in an image",
  "description": "Use Face Detection service to locate the faces in the image and send the bounding boxes to the Image Blur service to get the final result",
  "data_in_fields": [
    {
      "name": "image",
      "type": [
        "image/jpeg",
        "image/png"
      ]
    }
  ],
  "data_out_fields": [
    {
      "name": "result",
      "type": [
        "image/jpeg",
        "image/png"
      ]
    }
  ],
  "tags": [
    {
      "name": "Image Recognition",
      "acronym": "IR"
    },
    {
      "name": "Image Processing",
      "acronym": "IP"
    }
  ],
  "steps": [
    {
      "identifier": "face-detection",
      "needs": [],
      "inputs": ["pipeline.image"],
      "service_slug": "face-detection"
    },
    {
      "identifier": "image-blur",
      "needs": ["face-detection"],
      "condition": "len(face-detection.result['areas']) > 0",
      "inputs": ["pipeline.image", "face-detection.result"],
      "service_slug": "image-blur"
    }
  ]
}
```

- **Condition** ([optional] if this specific step should match a condition before being run)
- **Inputs** (which data should be put in the entry of the service)

```
{
  "name": "Face Blur",
  "slug": "face-blur",
  "summary": "Blur the faces in an image",
  "description": "Use Face Detection service to locate the faces in the image and send the bounding boxes to the Image Blur service to get the final result",
  "data_in_fields": [
    {
      "name": "image",
      "type": [
        "image/jpeg",
        "image/png"
      ]
    }
  ],
  "data_out_fields": [
    {
      "name": "result",
      "type": [
        "image/jpeg",
        "image/png"
      ]
    }
  ],
  "tags": [
    {
      "name": "Image Recognition",
      "acronym": "IR"
    },
    {
      "name": "Image Processing",
      "acronym": "IP"
    }
  ],
  "steps": [
    {
      "identifier": "face-detection",
      "needs": [],
      "inputs": ["pipeline.image"],
      "service_slug": "face-detection"
    },
    {
      "identifier": "image-blur",
      "needs": ["face-detection"],
      "condition": "len(face-detection.result['areas']) > 0",
      "inputs": ["pipeline.image", "face-detection.result"],
      "service_slug": "image-blur"
    }
  ]
}
```



React Flow

# NEXT STEPS



- Pipeline parallelization
- Toy datasets
- Functional tests on service declaration
- And many more...



**ANY QUESTIONS?** 😄

# USEFUL LINKS

- [Official website](#)
- [Documentation](#)
- [Frontend demo](#)
- [Backend demo](#)
- [Guide to MLOps](#)
- [Chatbot](#)
- [GitHub](#)

