**Employee identification**

# Brief description

Develop a web application to proceed photo of EmployeeID tag on the table (and additional employee’s information) and save it to database.

Find the dataset for ML part [**here**](https://drive.google.com/drive/folders/1J-JerIRzq11oyUksi0pGmYQrYSNW3Ub1?usp=sharing).

## Client part

**Functionality:**

* Add new employee information (First and Last Name (str), Age (int), Position (str), Remote (bool) + attach photo of Employee ID tag on the table)
* List existing employees + filtering by parameters (name, position, remote, employee\_id)
* Get employee information (display all employee parameters – name, age, position, remote, ID)

**Tech requirements:**

* FastAPI Swagger UI

## Server part

Consists of two microservices:

* **Service 1**: web application to handle client requests + store data
* **Service 2**: interacts with service 1 only, receives image of employee ID, proceeds it, and generates response of Employee ID in numeric format

At minimum Service 2 can generate numeric Employee ID randomly. At best Service 2 should proceed image using ML model to recognize Employee ID on the photo.

**Requirements to the server app:**

* FastAPI + SQLite
* Both services are standalone applications with REST API support, Service 1 to use port 8800, Service 2 to use port 9000
* Service 2 employee ID response mask: \*\*\*, where \* stands for a digit
* Service 1 new user flow:

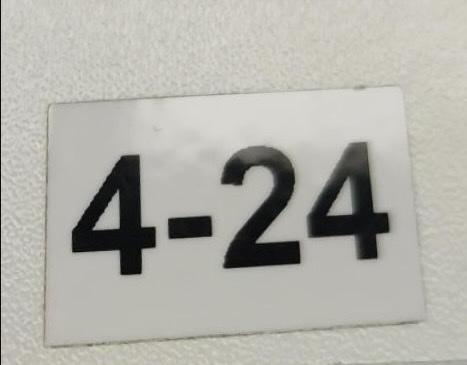
Receive client request -> forward image to Service 2 -> receive numeric employee ID from service 2 -> save employee data to DB -> send response to the client

* Service 1 API endpoints:
  + POST /employes/new
  + GET /employees/list[?name=...][&position=...][&remote=...]
  + GET /employees/{id}

**Requirements to the ML part of the server**

* Prepare and properly preprocess train, validation and test datasets using provided data.
  + *Hints*:
    - In order for the model to learn better, you need to create datasets with a wide variety of photos. Think about how you can increase the variety of photos (for instance, use augmentation).
    - Think also about how you can increase the frequency of occurrence of different numbers.
* Use pre-trained OCR model (from TensorFlow Garden or MMOCR) to recognize the employee id from the image.
* Fine tune the selected OCR model using prepared datasets.
* Write 3 scripts:
  + *train.py* - training the model, the script must accept as input the path to the config file, where you can specify all the training parameters: path to the train and validation datasets, the number of epochs, learning rate, scheduler and other hyperparameters.
  + *test.py* - testing the model, testing on a batch of data and calculating metrics (general accuracy, accuracy of each digit), the script must accept an argument with a path to the directory with photos and ground truth ids.
  + *predict.py* - inference on one photo and print the result.
* Develop a standalone web application with only one endpoint */image* and which can receive image, pass it through the OCR model and return resulting employee id. Use port 9000 for web application.

# Example of Employee ID

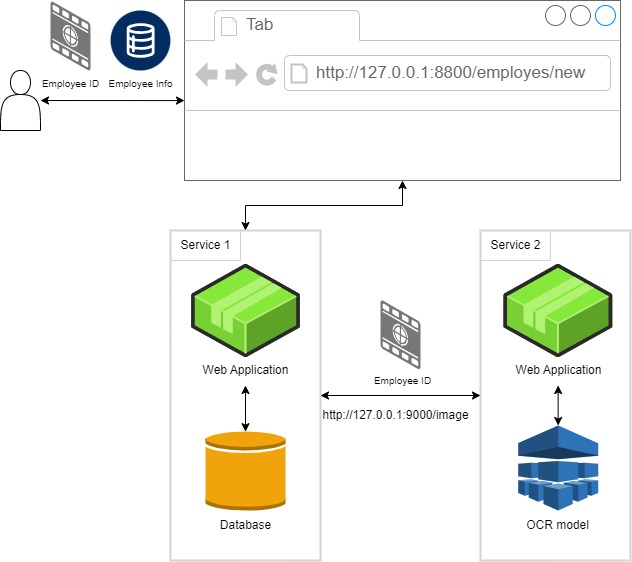


A close-up of a white surface

Description automatically generated A white surface with black numbers

Description automatically generated

# Simplified diagram of entire solution



# What should I do with the final solution?

1. Create repository on GitHub or Gitlab.
2. Push your code, don’t forget to add **README** with the instructions on how to start your application.
3. Grant access to the repository to our reviewer: nickname **andxeg** (the same for GitHub and for Gitlab).
4. Server part and ML part will be evaluated separately, but you should try to do everything.