AI - REPORT

Artificial Intelligence project – ECAM January 2023 – 5MIN

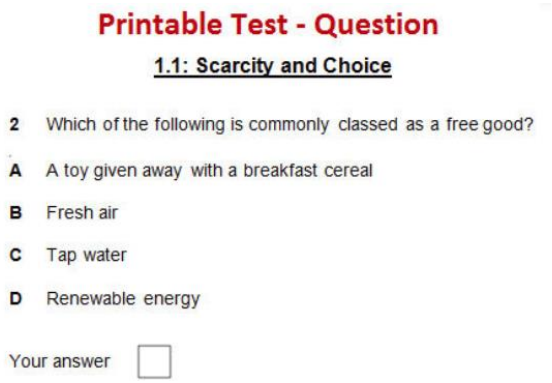
Florian Lebecque
Louis-Antoine Devos
Sarah Liettefti Lens

Git link: https://github.com/FlorianLebecque/OCR-Fusion

Performance

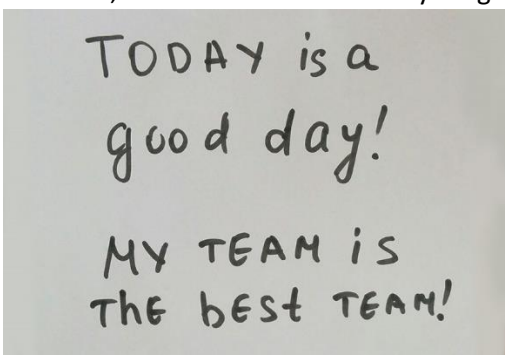
Our goal is to offer the best text recognition application by any means. So, our OCR application can run with different machine learning models.

The most common way to build an OCR application is by using Tesseract, which is an Open-Source OCR Engine. His main advantage is it can run locally. It works well for recognizing printed characters, result below:




```
Printable Test - Question
1.1: Scarcity and Choice
2 Which ofthe following is commonly classed as a free good?
A toy given away with a breakfast cereal
Fresh air
oUu F
Tap water
Renewable energy
Your answer |
```

However, it is not very good with handwritten characters, result below:




```
"TODAY isa
qood day!
MY TEAN TS
The best TERN!
```

Ergonomics and functionalities

The frontend is composed of two pages. The first one is the home page (index.html) where the user can upload images with writing and retrieve the text from it. There are several functionalities in this page:

- Upload images
- Take pictures
- Choose one or more algorithms to use to analyze the text
- Configure a session name: it will associate the result with this name, it is a simple way to create “user profiles” without connection.
- Validate those configurations and send it to the API
- Display the image and the editable text resulted from it for each algorithm selected
- Modify the result text and save the modifications
- Go to the history page

The second page is the history of processed files and results. The functionalities are:

- Set the session name we are looking for
- For the session name set, display all its history in a table with image name, a preview of the result, the algorithm used and a button to display it.

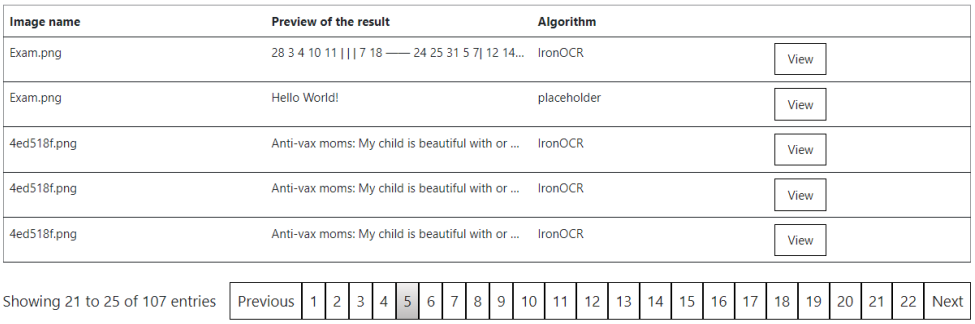


Image name	Preview of the result	Algorithm	
Exam.png	28 3 4 10 11 | | 7 18 —— 24 25 31 5 7| 12 14...	IronOCR	View
Exam.png	Hello World!	placeholder	View
4ed518f.png	Anti-vax moms: My child is beautiful with or ...	IronOCR	View
4ed518f.png	Anti-vax moms: My child is beautiful with or ...	IronOCR	View
4ed518f.png	Anti-vax moms: My child is beautiful with or ...	IronOCR	View

- Browse the result of a processed files, its editable text and save changes

The web interface is configured by 3 classes (<ParametersBuilder>, <Builder> and <OcrAPI>) managed by a script in JavaScript.

- The <ParametersBuilder> makes adding algorithms choices dynamically to the interface possible. Some algorithms need the user to set parameters to be efficient, the class will manage the different display.
- The <Builder> creates templates that can be reused at different place in the project. It builds the algorithm checkbox dynamically thanks to <ParametersBuilder> but also the Card that will the display the result of OCR.
- <OcrAPI> use the previous classes to browse data, makes calls to the API and display the information. This is where all the buttons, forms, and much more are managed.

Architectures

Our application uses a frontend connected to an API backend. All the computation and history management are done on the API side.

API

The API is composed of three main controllers.

Algorithms	It goals is to tells what kind of OCR algorithms are implemented in the API
Image	Offer a way to get the images that have been uploaded on the backend
Ocr	Main controller that let a user use OCR capability of the API with a specified algorithm

The API documentation can be found on the git repository.

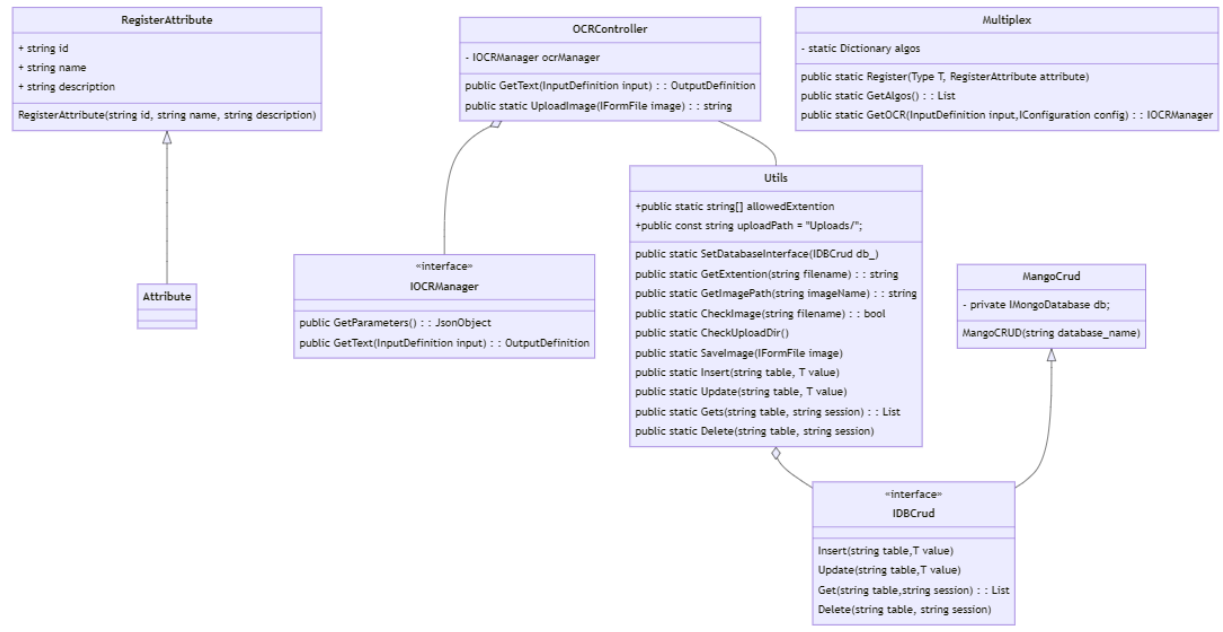
Backend architecture

The backend is build to be able to use many different OCR algorithms. We have a controller <OCRControler> on witch we inject an implementation of the <IOCRManager> with the help of a multiplexer <Multiplex>.

The multiplex returns an implementation based on the internal settings of the API and the requested algorithm asked by the user on the API call.

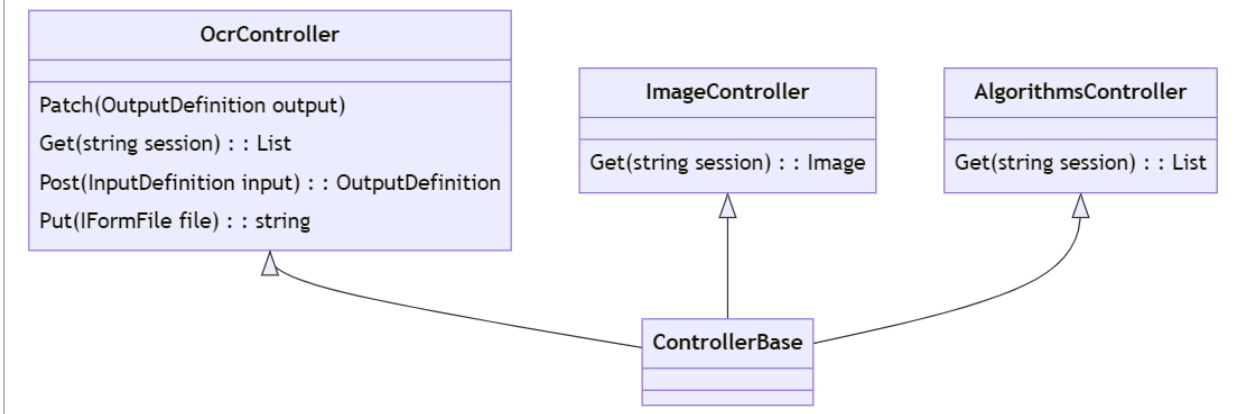
On the startup, we scan all the classes, and we find all the one that have the <RegisterAttribute>, those classes are registered as an algorithms into the multiplexer and are ready to be used by the API.

Classes diagrams

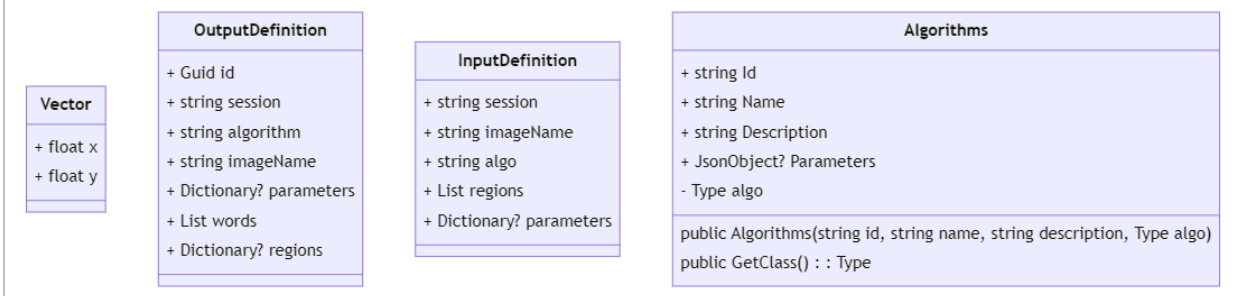


Main part of the progams, those classes have most of the core logic.

- The <Utils> class has an implementation of the database interface <IDBCrud> here we use a MangoDB <MangoCrud>. The Utils class is mainly used to interface with the database and other simple function.
- The <Mutltiplex> class main's goal is to return the registered Algorithm
- The <RegisterAttribute> is an attribute that can be added to classes, those classes must implement <IOCRManager> interface and will be registered to the Multiplexer
- The <OCRController> goal's is to interface with the API and add an abstraction layer between the API and the OCR algorithm logic.

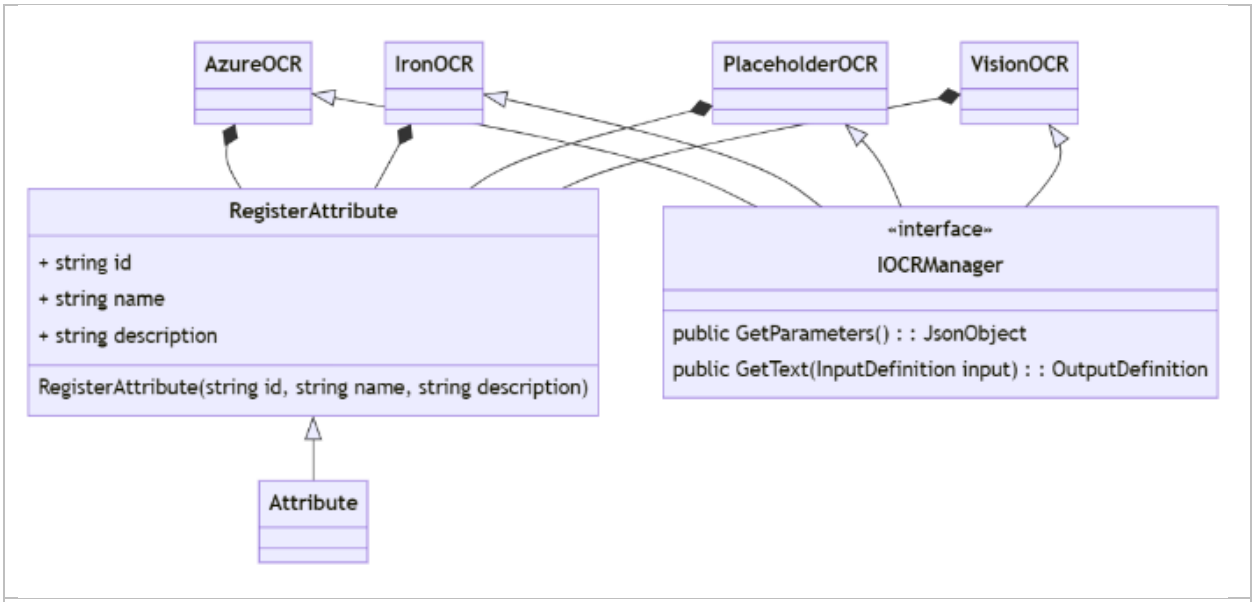


API part of the backend solution. Build in three controllers those extend the <ControllerBase>. They interface with <OCRController> and <Utils> class and manage the API Calls. (See API Documentation)



Main schema of the API, those classes are data object and don't have any logic.

- The <InputDefinition> is the parameters send to the API to trigger and OCR on the specified image.
- The <OutputDefinion> is the output of the API, contains all the data return by the OCR Algorithm
- The <Algorithms> contains all the information and parameters required by an <IOCRManager>



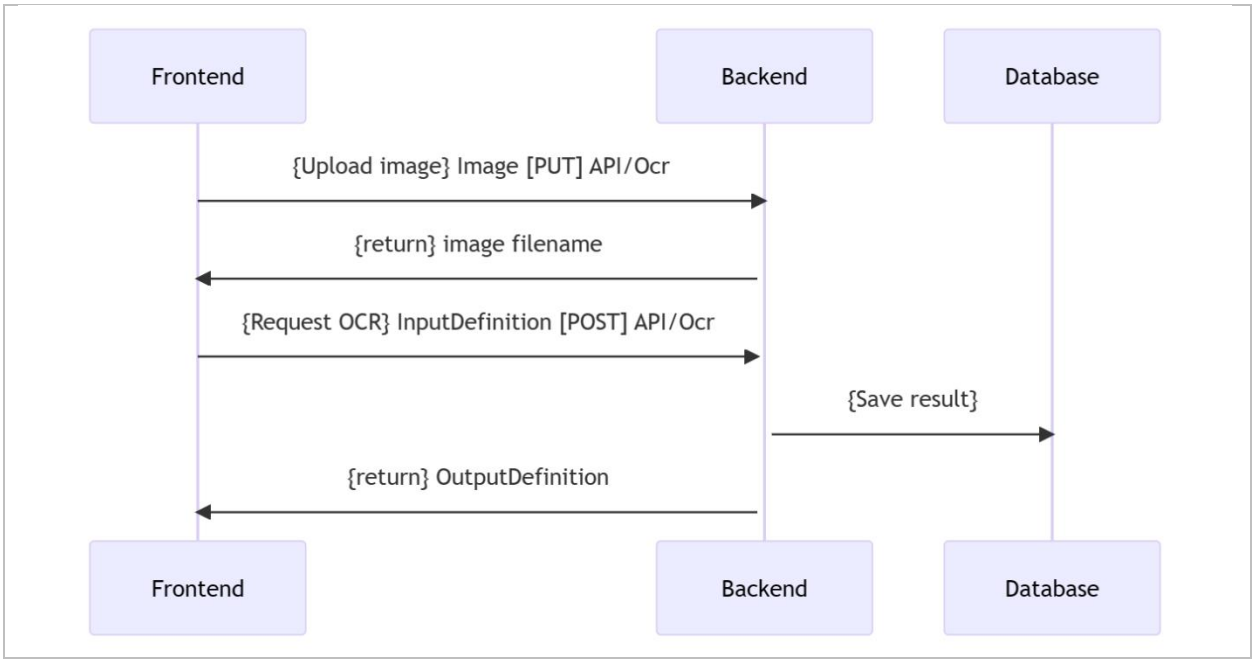
This part is the implementation of the different algorithms, they need to implement the <IOCRManager> interface and have an <RegisterAttribute>. They are then automatically added to the multiplexer and the <AlgorithmsController>.

Folder hierarchy

API Object	Contains all object that are returned by the API
Controllers	All API controllers
Database	Implementation of the <IDBCrud> interface
OCR	Implementation of the <IOCRManager>
Uploads	Contains all the uploaded photo

Sequence diagram

Request an OCR on an image



Conclusion

It is possible to use an API from an existing AI OCR service to implement OCR functionality. This can be a convenient and cost-effective way to incorporate OCR capabilities into a project, as it allows you to leverage the expertise and resources of the API provider. However, it is important to carefully evaluate the capabilities and limitations of the API and ensure that it meets the specific needs of the project. It may also be necessary to consider factors such as the cost of using the API, the terms of service, and any potential legal or ethical issues.