

# OCR APPLICATION – REPORT

Artificial Intelligence project – ECAM January 2023 – 5MIN

Florian Lebecque  
Louis-Antoine Devos  
Sarah Liettefti Lens

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

Optical Character Recognition (OCR) is a technology that enables the recognition and conversion of printed or handwritten text into a digital and editable format. The use of Artificial Intelligence (AI) and Machine Learning (ML) has greatly improved the accuracy and efficiency of OCR. These technologies will be explored through the creation of an OCR application.

## Performance and AI

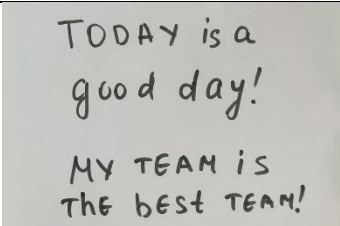
The goal is to offer the best text recognition application possible. Therefore, we designed an OCR application that can run with three different machine learning models: Tesseract with the IronOCR library, Google Vision, and Azure cognitive services.

The first one we used to build our OCR application is **Tesseract**, an Open-Source OCR Engine, whose main advantages are that it can run locally and recognize several languages in printed characters. It was used through the integration of the library **IronOCR** in a .NET C# application, that wraps the Tesseract engine with a C# API and make it easy to use its powerful capabilities.

As it can be seen in the comparison below, Tesseract is more efficient with printed characters than handwritten one.

<p><b>Printable Test - Question</b></p> <p><b><u>1.1: Scarcity and Choice</u></b></p> <p>2 Which of the following is commonly classed as a free good?</p> <p>A A toy given away with a breakfast cereal</p> <p>B Fresh air</p> <p>C Tap water</p> <p>D Renewable energy</p> <p>Your answer <input type="checkbox"/></p>	<p>Printable Test - Question</p> <p>1.1: Scarcity and Choice</p> <p>2 Which of the following is commonly classed as a free good?</p> <p>A toy given away with a breakfast cereal</p> <p>Fresh air</p> <p>oUu F</p> <p>Tap water</p> <p>Renewable energy</p> <p>Your answer  </p>
<p>TODAY is a good day!</p> <p>MY TEAM is THE BEST TEAM!</p>	<p>TODAY isa good day!</p> <p>MY TEAN TS</p> <p>The best TERN!</p>

On the other hand, the OCR API of **Google Vision** gives good results in both handwritten and printed characters recognition.

<p><b>Printable Test - Question</b></p> <p><u>1.1: Scarcity and Choice</u></p> <p>2 Which of the following is commonly classed as a free good?</p> <p>A A toy given away with a breakfast cereal</p> <p>B Fresh air</p> <p>C Tap water</p> <p>D Renewable energy</p> <p>Your answer <input type="checkbox"/></p>	<p>Which of the following is commonly classed as a free good?</p> <p>A toy given away with a breakfast cereal</p> <p>Fresh air</p> <p>C</p> <p>Tap water</p> <p>D Renewable energy</p> <p>Printable Test - Question</p> <p>1.1: Scarcity and Choice</p> <p>A</p> <p>B</p> <p>Your answer</p>
	<p>TODAY is a good day!</p> <p>MY TEAM is</p> <p>The best TEAM!</p>

Google's opponent, Microsoft, also has an OCR API offered by **Azure Cognitive Services**. It will be interesting to compare those two rivals and giant of the technology. This model works better with typed text than handwritten one.

The disadvantage of the last two models is the dependency with an API that prevent to run it locally and forcers us to use a private API key from a subscription account. For Azure, it is free for moderate use, and Google only costs few cents per thousands of requests.

Some of the model's parameters can be configured to improve the results. Among these there is the language of the text, the strategy used (fast or accurate) and others. It is also always possible to manually correct the output and save the results.

	Efficiency printed characters	Efficiency handwritten characters	Price	Component
<i>IronOCR</i>	Good	Bad	Free	Tesseract Engine and API
<i>Google Vision</i>	Good	Good	1,50 \$ / 5 000 000	Private API
<i>Azure Cognitive Services</i>	Good	Bad	Free for moderate use	Private API

## 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
test.png	hello,World,I,How,Is,your,day,going,?	<a href="#">View</a>
7382939.jpg	ie EG ee Loecscins — etnies age ee ee Tablea...	<a href="#">View</a>
test.png	hello,World,I,How,Is,your,day,going,?	<a href="#">View</a>
pointe-epee.jpg	.— Vis de réglage-Course allumoge l——— ...	<a href="#">View</a>
IMG20221129140623.jpg	vel enna RANT ALY Maas NOM D'HOTE	IronOCR <a href="#">View</a>

Showing 1 to 5 of 164 entries

Previous 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26  
27 28 29 30 31 32 33 Next

- 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.

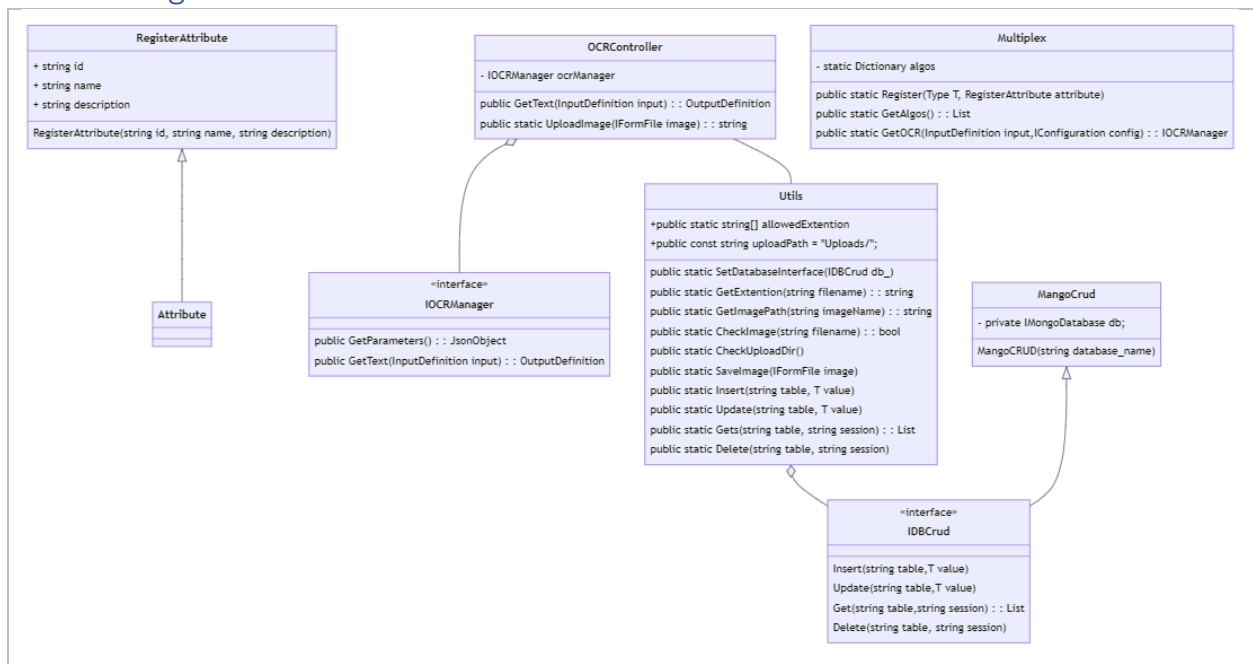
### Backend architecture

The backend is built to be able to use many different OCR algorithms. We have a controller <OCRController> on which 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.

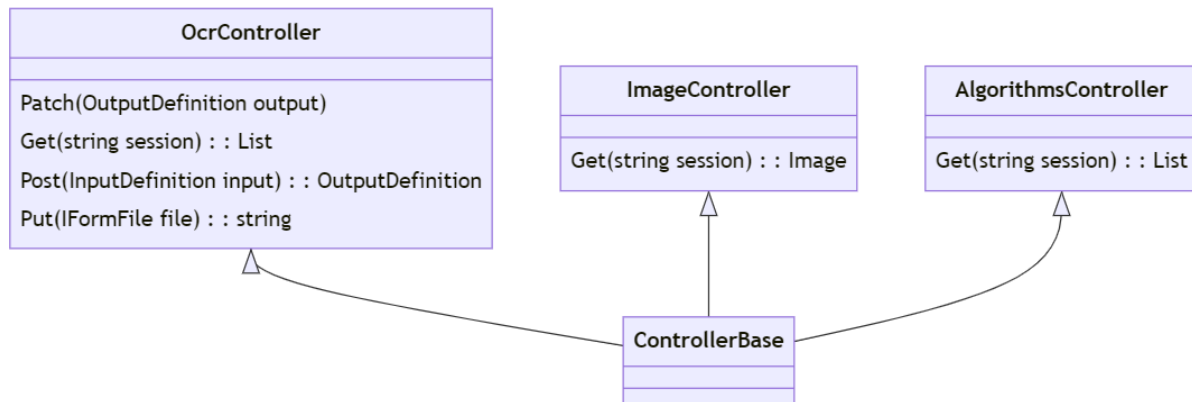
On the startup, we scan all the classes, and we find all the one that have the <RegisterAttribute>, those classes are registered as an algorithm into the multiplexer and are ready to be used by the API.

### Classes diagrams

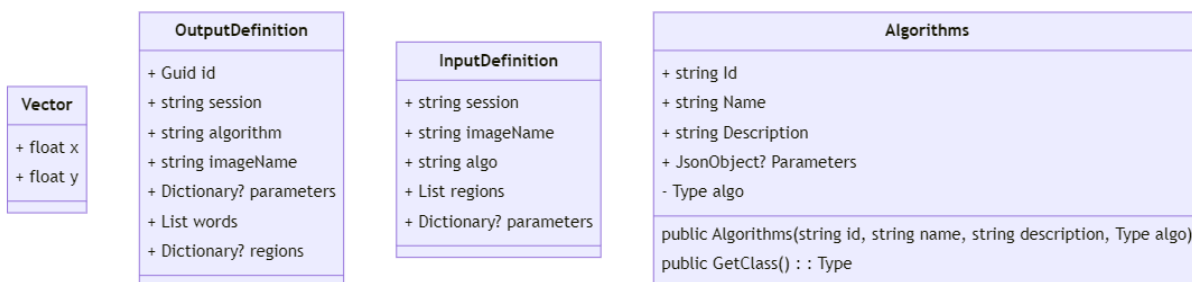


Main part of the programs, those classes have most of the core logic.

- The <Utils> class has an implementation of the database interface <IDBCrud> here we use a MongoDB <MongoCrud>. 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.

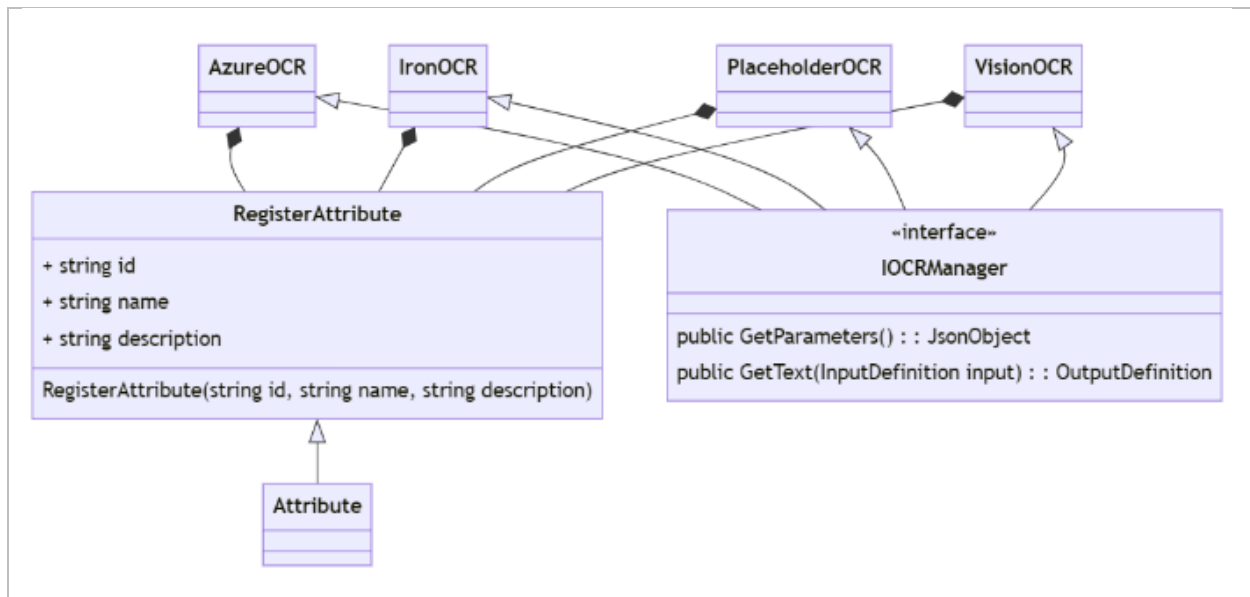


API part of the backend solution. Build in three controllers those extend the <ControllerBase>. They interface with <OCRController> and <Utils> class and manage the API Calls. (See API Documentation)



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>



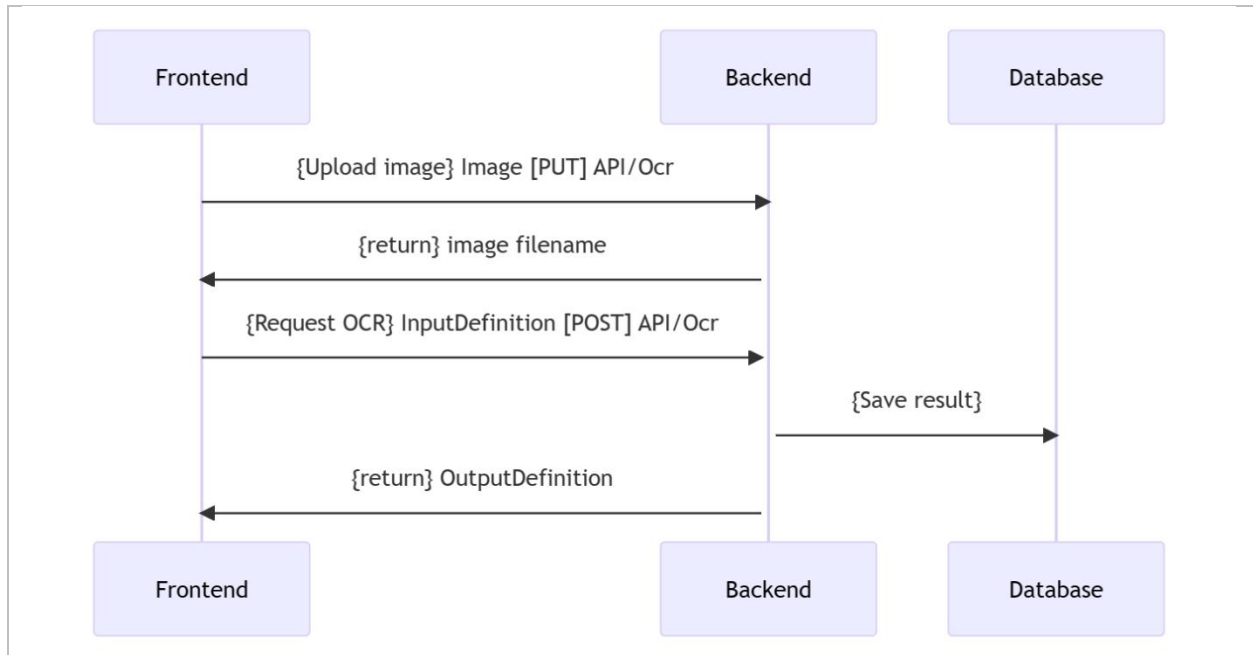
This part is the implementation of the different algorithms, they need to implement the `IOCRManager` interface and have an `RegisterAttribute`. They are then automatically added to the multiplexer and the `AlgorithmsController`.

## Folder hierarchy

API Object	Contains all object that are returned by the API
Controllers	All API controllers
Database	Implementation of the <code>IDBCrud</code> interface
OCR	Implementation of the <code>IOCRManager</code>
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.