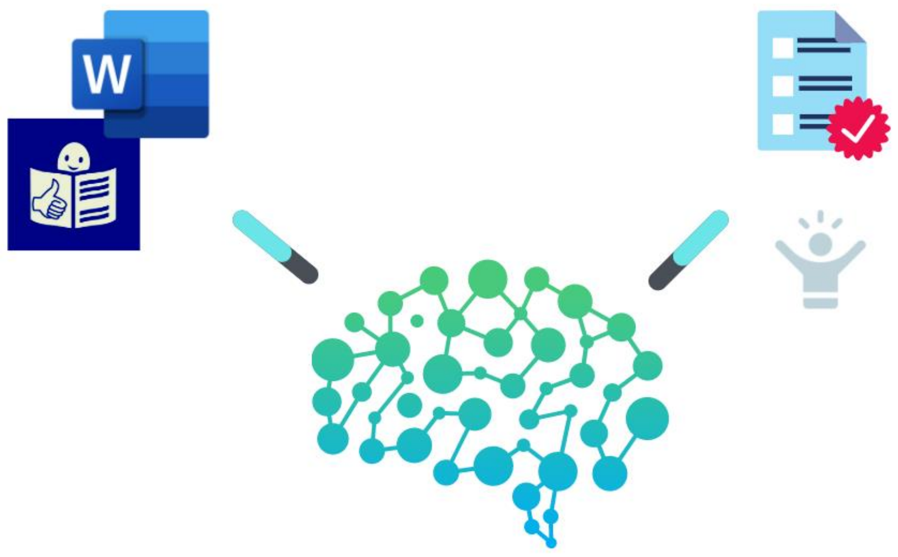
Team 35 : PICTOFALC

Pi² Project Closure Report:



PICTOFALC

An add-in to illustrate your FALC documents and promote accessibility

Project partner:

Microsoft

Philippe Trotin

Team n°35 members :

Nicolas Carval
Bruno Pincet
Chloé Tempo
Laurine Sallé
Louis Tempé

Table of Contents

Objectives Set versus Results Achieved

Project Initial Objectives

Microsoft France and UCANSS want to have a tool to easily generate illustrations for documents written using the FALC method (FALC in French stands for Facile A Lire et à Comprendre) and to facilitate reading for people with intellectual disabilities. The idea is to rely on AI to analyze a sentence, isolate its key words and use a library of pictograms to offer the writer a simple and ergonomic tool to illustrate FALC documents.

The Auto FALC project was proposed by Microsoft France and UCANSS and is carried out by team 35, made up of 5 engineering students in M1 “Data & IA” at ESILV. It consists of designing a tool to automatically generate illustrations for a FALC text.

It is therefore important to define what the term “FALC” means. Generally, people with intellectual disabilities have difficulty reading texts because of their complexity and non-intuitive formatting. The FALC is here to answer this problem. It was created in 2009, through a European project entitled "Pathways", it is a set of rules for writing documents (which relate to both the content of the text, and its form), allowing people with intellectual disabilities to better understand and read documents.

Moreover, a FALC text is even more understandable when it is illustrated by drawings, pictograms or photos. However, today it is still too difficult for its designers to quickly find suitable illustrations.

The objective of the project is therefore to facilitate the illustration of the documents thanks to AI, by automatically proposing pictograms adapted to a text written in FALC and thus facilitating the work of illustrators.

Mr. Trotin, our partner within Microsoft, gave us the freedom to choose the form in which we wish to develop our application, in the form of an Add-In in Microsoft Word or in the form of an external application. Also, we could process text in the language of our choice, in French or in English. The expected result is a prototype, which should at least work for some predefined texts.

Results Achieved

We have chosen, to create an Add-In integrated directly into Word, easy to install, which will analyze the text written in FALC, and will automatically offer pictograms. The person working on the translation of a text into FALC can then choose and place the pictograms that suit him, and this will allow him to save considerable time on the search for illustrations. We also managed to produce a result in French, and our prototype is functional on a selection of texts of our choice.

Our project is currently at the prototype stage, which is why we have not yet succeeded in making it work for all the texts that exist, because our database only contains a number limited in size, so only a sample of words can be illustrated.

Also, we have not yet deployed our project. We preferred to focus on optimizing our program for keyword detection.

During the meetings that we regularly had with our partner, we presented our progress and discussed about possible improvements. The partner's objectives did not change during the project.

List of Deliverables

The main deliverable required by the partner was a functional MVP. Working for 2-3 examples in order to testify for the value it could create. Thus, here are the different deliverables we provided:

- GitHub link to the repository (code, reports...)
- MVP/Proof of Concept
- Video of functional usage

Methodological review

We mainly organized ourselves in an agile way by establishing one-week sprints (or 2 weeks depending on the scope of the tasks).

From the beginning of the project, we agreed to use the Pi2 slot on Tuesday morning each week to go a meeting. So, every Tuesday morning, we presented our progresses to each other, we discussed the problems encountered, then we took time to move forward and plan the next actions. Also, we progressed on our own, or in pairs during the week, and we used a conversation on Messenger to communicate our progress during the week.

Regarding the distribution of tasks, the first 2 weeks of the project consisted mainly of defining objectives, research, first tests, and brainstorming of leads and ideas. We therefore each worked on all the themes, so that each person could bring something to the themes and know their affinities.

Around the end of the first month, we divided into 3 teams, corresponding to the 3 main axes of the project that we defined:

- The first team worked on the Add-In, the user interface visible from Microsoft Word. It consisted of Nicolas and Bruno.
- The second team worked on the key word extraction (Artificial Intelligence) part. Laurine was in charge of this part, and other team members also participated.
- The last part is the creation of the database, its connection with the project and its population. Chloe and Louis oversaw the game.

We worked with this distribution until about the Christmas holidays. At that time, each part was pretty much developed individually, so the challenge was to connect them together so that they worked within the same project, while improving the functionality of each. So, we worked in a more transverse way between the different parties, taking advantage of the Tuesday morning meetings to try to solve the problems and improve the functionalities together.

Risk Management

The first risk we identify is that the application is not easy enough to install. Indeed, our application is intended for the general public, not necessarily mastering all the computer tools. It is therefore imperative for the installation to be accessible to everyone, in order not to lose users.

- In order to allow an easier installation, we opted for an application directly in the form of an Add-in in Microsoft Word, instead of having an independent application to be installed separately. However, since we haven't made it to the deployment of the application, we have not yet demonstrated the ease of installation. We base this observation on other Add-ins in Word that we have been able to install.

The second risk that we identify is the difficulty of using the application. Indeed, for the same reasons as the installation, we must ensure that the application is intuitive, clear, and does not require a mastery of computer tools.

- The choice to create an Add-in in word responds to this risk, because it allows very easy use, directly from the word processing tool. Also, our application meets the needs of users with very few features, which allows the user to easily take control of the application. Finally, we made sure to design a very intuitive, minimalist and clear display, so that everyone can understand it.

Another risk that we have identified is the application performance. To satisfy users, the keywords and images returned by the Add-in must be relevant.

- For this, during the keyword research phase, we used several libraries to guarantee the best result. For example, the keyword extraction libraries have better results in English, we decided to use 2 of them, translating the text into English. Also, we used a spelling error detection library, and another library that removes words that are too similar. After many tests and adjustments of the results, we chose the libraries that worked best for our use.

There is also the risk of not finding an image related to the keyword in the database.

- For this, we must ensure that we have the most complete database possible, but if the word really does not have an image, we must not return this keyword. We have not yet integrated this case into our code, because we are on a demonstration model where this problem will not appear (our database was adapted for the texts that we have chosen to show).

Constraints

- Accessibility: The first constraint we identified concerns the accessibility of our application. Indeed, our application will be used by people with intellectual disabilities, accompanied by educators. Our interface must therefore be very clear and intuitive, to facilitate use for our target users. Also, the images we return must be very simple and understandable. This is why we chose to use icons.
- Cost: The cost constraint for us, was to find free platforms that we can work with (cloud, translation libraries).
- Language: Most of the keyword extraction libraries fit English, but our text is in French. So it is a challenge for us to make the program work with French text.

- Missing illustration: One of the problems that a user could find is also the fact that it's very difficult to find illustrations for all the words of the French language. In fact, the AI will sometimes find word which are considered as keywords but can't be illustrate as "vie" or "Histoire" which are global concept.
- Legal considerations: We must consider the legislations, and use free of right icons

Technical Review

Our project is divided into 3 Parts:

- Word Add-In
- AI
- Database

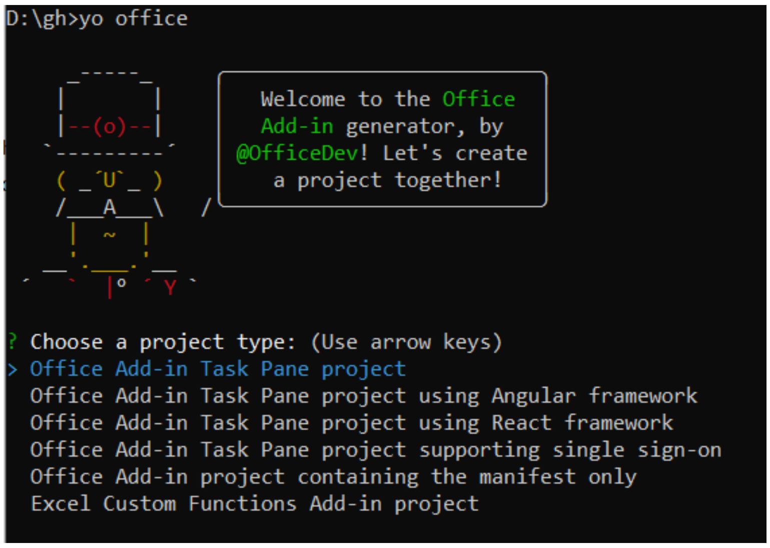
⇨ Add-In

It corresponds to the front end, built with HTML5, CSS and JavaScript. It enables users to interact with our software.

To create the Add-In, we followed Microsoft documentations:

- https://docs.microsoft.com/fr-fr/office/dev/add-ins/develop/yeoman-generator-overview
- https://docs.microsoft.com/en-us/office/dev/add-ins/develop/develop-overview

Thus, we generated the main structure using the Yeoman Generator. The Yeoman Generator for Office Add-ins is an interactive Node.js-based command line tool that creates Office Add-in development projects.



```
D:\gh>yo office

                    Welcome to the Office
                    Add-in generator, by
                    @OfficeDev! Let's create
                    a project together!

? Choose a project type: (Use arrow keys)
> Office Add-in Task Pane project
  Office Add-in Task Pane project using Angular framework
  Office Add-in Task Pane project using React framework
  Office Add-in Task Pane project supporting single sign-on
  Office Add-in project containing the manifest only
  Excel Custom Functions Add-in project
```

Yeoman office Add-In generator

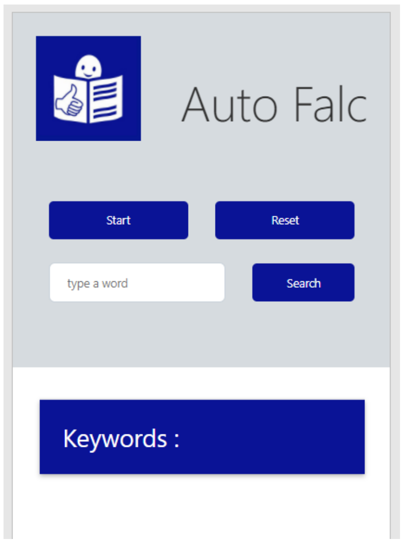
In order to respect the initial expected implementation of our Add-In, we made several choices (usage, design, responsiveness).

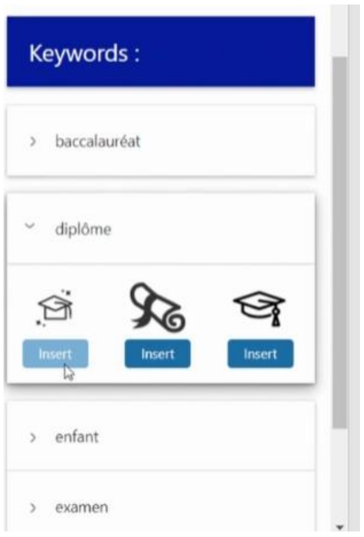
At first, we wanted to design the side page as simple as possible. With only two buttons, for starting and resetting the program. However, after several tries, we changed our mind and decided to implement in addition to previous features a search bar to allow the user to type in a word that would have been missed by our AI.

Then after launching our program and retrieving a list of key words with corresponding images, we noticed that it wasn't intuitive for the user to find back the words in the original text, to copy past at the right place. We thus, implemented a highlighting functionality of key words detected.

In the end, our supervisor suggested in order to show the different images related to the words. We put in place sections for each keyword, with maximum 3 images displayed in it.

This really simplified and made our software more accessible and intuitive.





Add-In design

In Conclusion, this part of the project was made iteratively, by sequences, involving the client (our supervisor) in order to pivot the expected result. There were two people in the team fully dedicated to this, part. It allowed to better think of implementations, to do pair programming, and to respect deadlines.

⇨ AI

To create the keyword extraction algorithm from a text, we used the Python language, which has many libraries that can be used to solve our problem.

First, we tested several models of keyword extraction, in order to choose the best one for our project. We tested the performance of these libraries, both in French and in English, and found that the models are more adapted to English. Finally, we chose 2 models: **KeyBert** and **Yake**.

Python AI libraries used

KeyBert is a minimal and easy-to-use keyword extraction technique that uses BERT-embeddings and simple cosine similarity to find the sub-phrases in a document that are the most similar to the document itself. The most similar words could then be identified as the words that best describe the entire document.

Yet Another Keyword Extractor or Yake is an unsupervised approach for automatic keyword extraction using text features. It uses text statistical features extracted from single documents to select the most important keywords of a text. Yake! does not rely on dictionaries nor thesauri, neither is trained against any corpora

These 2 keyword extraction models require the use of English. So we used the **Azure Cognitive Services** Translation API to process the text from the English translation.

Because we use 2 keyword extraction libraries, we then select a certain number of keywords (given the size of the text entered), according to their relevance. Then we translate the words back into French to display them in the Add-in.

At this stage, our code is functional, but we find several points of improvement to be solved.

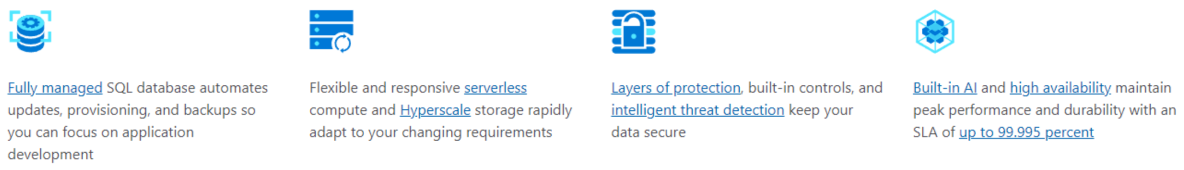
- ✓ First of all, we notice that some displayed words are misspelled. For example, we once got the word “reconnui” instead of “reconnu”. This can happen during a translation. We therefore used a library that will correct these problems, by finding the correctly spelled word closest to the given word. The library we use is **Autocorrect**.
- ✓ Also, we find that the algorithm returns words belonging to the same family. For example, "élève", "élèves". As an improvement, we agreed that it would be good to keep only one of the words. We do this thanks to the **thefuzz** library, which will allow us to group the words belonging to the same family, and then we will choose only one.
- ✓ Finally, since we went through an English translation, the words returned do not all correspond exactly to the words written in the text. However, we highlight in the text the matches with the returned words. To maximize these correspondences, we decided to keep only the stem of the words. For this, we use the **treetaggerwrapper** library.

These improvements allow us to have a better performance in the returned words.

⇨ Database

Concerning the database, after many comparisons, we finally chose a SQL database, as it is a relation-based Database type which was perfect in our case and can handle large amount of data.

Part of the Azure SQL family, Azure SQL Database is an always-up-to-date relational database service built for the cloud. It automatically scales to meet app requirements and keeps them running with up to 99.995% availability.



Fully managed SQL database automates updates, provisioning, and backups so you can focus on application development

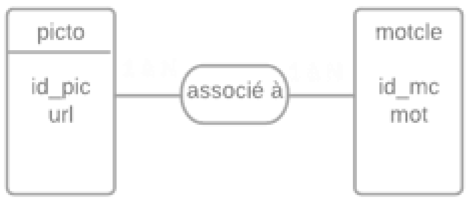
Flexible and responsive serverless compute and Hyperscale storage rapidly adapt to your changing requirements

Layers of protection, built-in controls, and intelligent threat detection keep your data secure

Built-in AI and high availability maintain peak performance and durability with an SLA of up to 99.995 percent

The database is stored on an Azure server, from Microsoft and is accessible from everywhere and 24 hours a day. The database schema is composed of 3 tables:

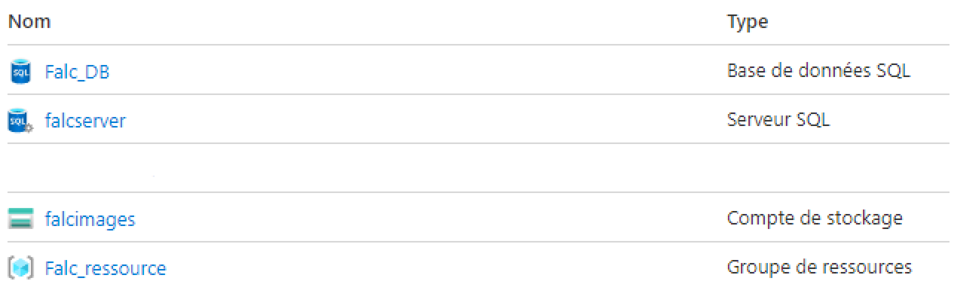
- The first one containing the keywords
- Another one containing the ID of the pictograms
- And finally, a joining table associating the previous tables.



database schema

To send pictograms to the Add-In, we just have to get the ID of the pictograms from the SQL database and add it at the end of our Microsoft Azure blob container storage link.

Azure Blob storage is a Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of structured and unstructured data and is perfectly designed for serving images or documents directly to a browser which is completely adapted for our Add-In.



Nom	Type
Falc_DB	Base de données SQL
falcserver	Serveur SQL
falcimages	Compte de stockage
Falc_ressource	Groupe de ressources

DataBase Azure service used

To facilitate the insertion of the key words and the pictograms into the database, we implemented a simple software in C#. Using the software, which is linked to the Azure SQL database, we can add any word we want, and assign to it one or many pictograms.

Once a keyword is added to the database, it is possible to delete the word, or to unassign any pictogram from it, but also assign new pictograms, and even modify the keyword. Thanks to that software, we do not have to type any 'INSERT INTO ...' SQL nonquery, what makes insertions faster, and easier.

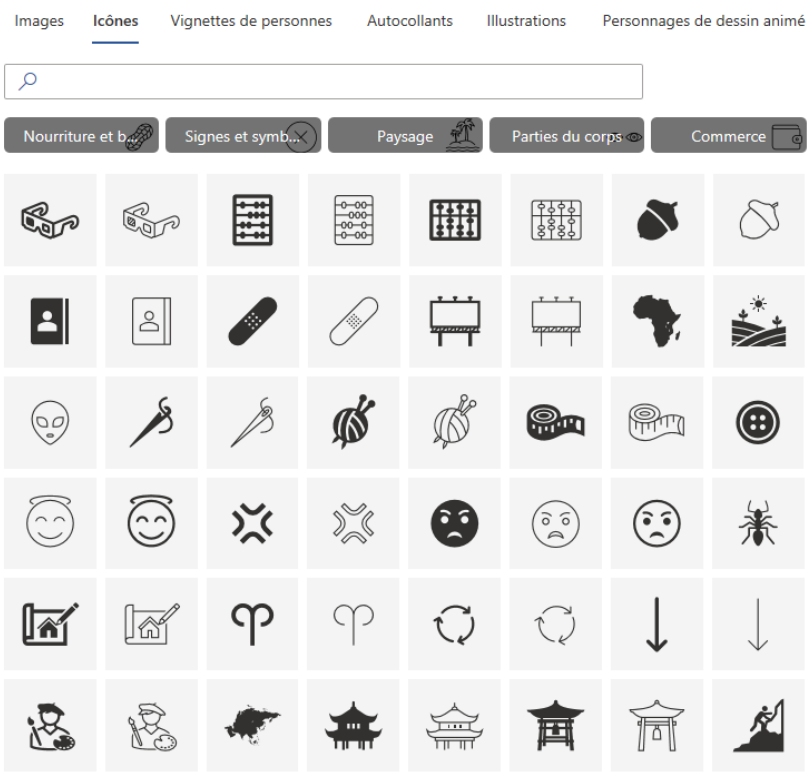
Concerning the deadline for this part, we underestimated the time that it will take to master all the concept of implementing a complete database on a unknow platform. At the beginning, no one of the members of the teams had ever done that, but thanks to the Microsoft documentations we end up succeeding even if it took more weeks.

“Post-Project” tasks

Currently, the main idea of illustrating FALC documents from Word with an AI is functional. However, the objective of proposing a fully online solution is not completely completed. The database part and the add-In part are online ready. However, the AI python part is only available locally and must be launched on the computer of the illustrator. Deploy the application, make it usable by anyone, and easy to install is one the main post project tasks that should be done.

Moreover, the Azure database part have been populated by selecting pictograms by hand according to FALC texts that are used for various proof of concept demonstrations.

It's not sustainable in the long term, and the solution would be to connect an image bank with its keywords, or by labelling the images ourselves, thanks to an AI. Microsoft has already created a pictograms Database; it could be interesting to try to connect this database to this project.



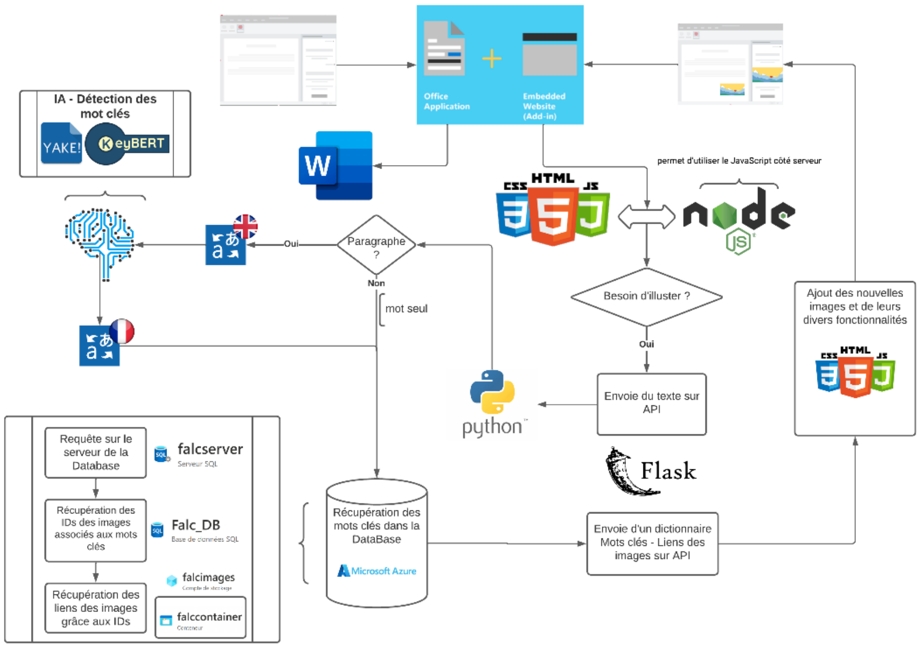
Microsoft pictograms Database

Project completion recommendations

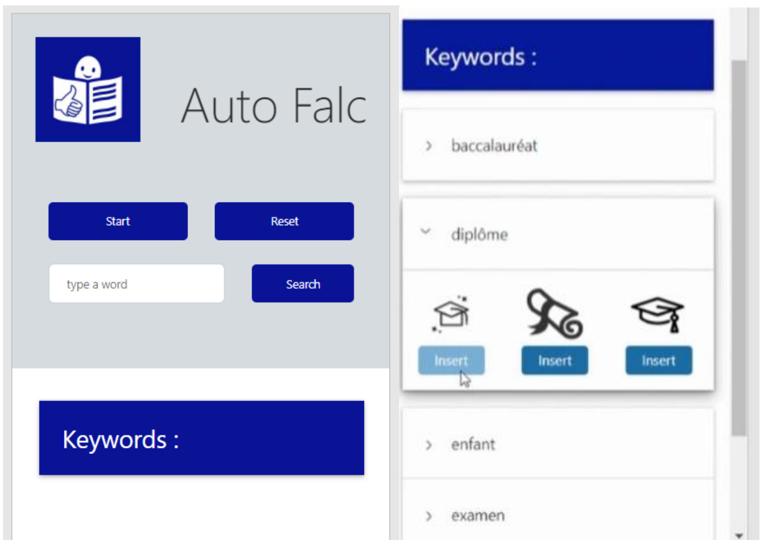
We organized our teamwork in the form of sprints, of one or two weeks. We think it would have been useful to experiment even more with the agile method through this project, for example by making clearer sprint plannings, backlogs, and retrospectives.

Annexes to the closure report

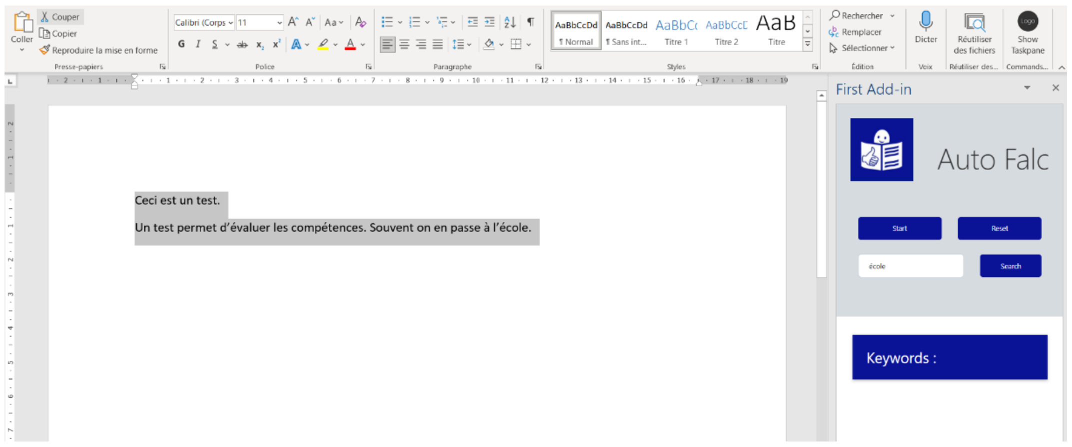
- GitHub link: https://github.com/chlotmpo/Auto_FALC
  - Containing the code, demonstrations, tests
- Functional schema:


IA - Détection des mot clés
YAKE!
KeyBERT
Office Application
Embedded Website (Add-in)
permet d'utiliser le JavaScript côté serveur
HTML
CSS
JS
node
Paragraphe ?
Oui
Non
mot seul
Besoin d'illuster ?
Oui
Envoie du texte sur API
python
Flask
Ajout des nouvelles images et de leurs divers fonctionnalités
Requête sur le serveur de la Database
falcserver
Serveur SQL
Récupération des IDs des images associés aux mots clés
Falc_DB
Base de données SQL
falcimages
Récupération des liens des images grâce aux IDs
falccontainer
Récupération des mots clés dans la DataBase
Microsoft Azure
Envoie d'un dictionnaire Mots clés - Liens des images sur API

- Add-In visualization:

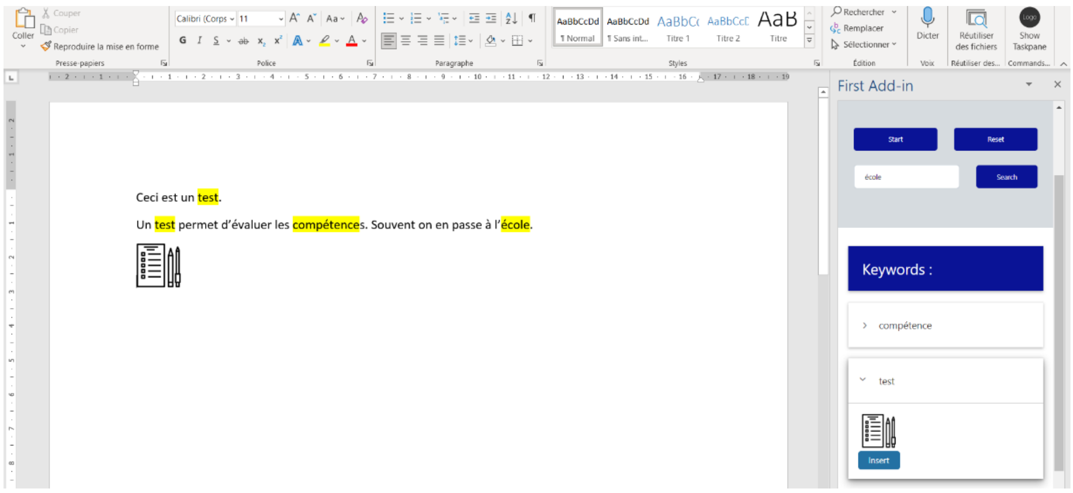


- Result visualization:



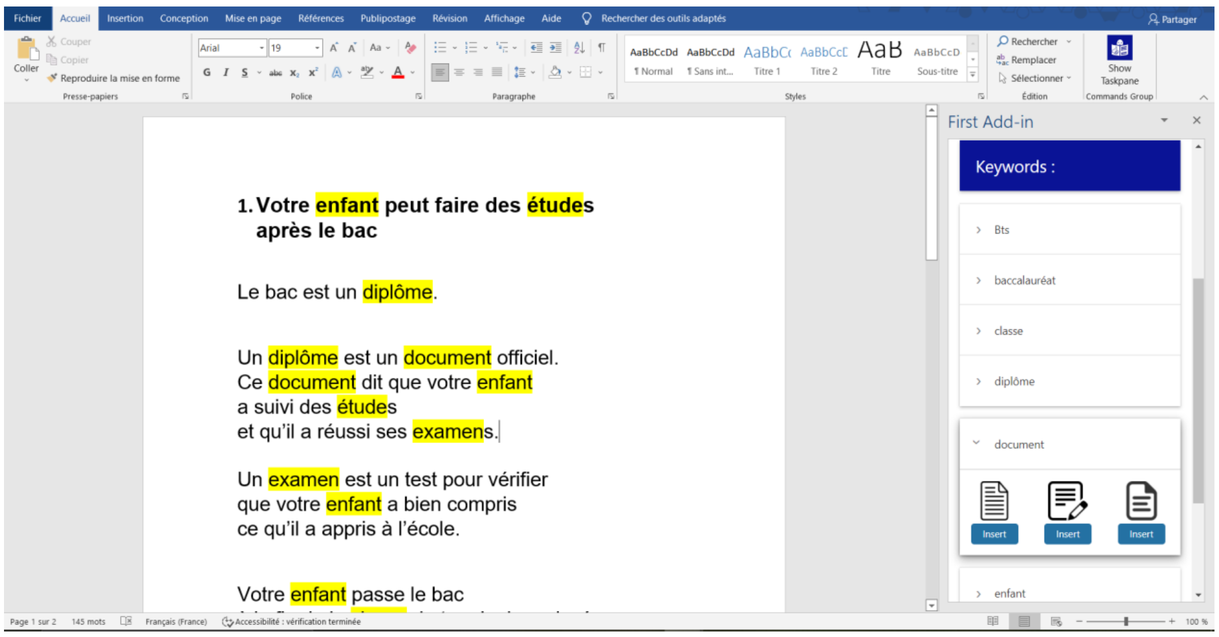
Ceci est un test.

Un test permet d'évaluer les compétences. Souvent on en passe à l'école.



Ceci est un test.

Un test permet d'évaluer les compétences. Souvent on en passe à l'école.



1. Votre enfant peut faire des études après le bac

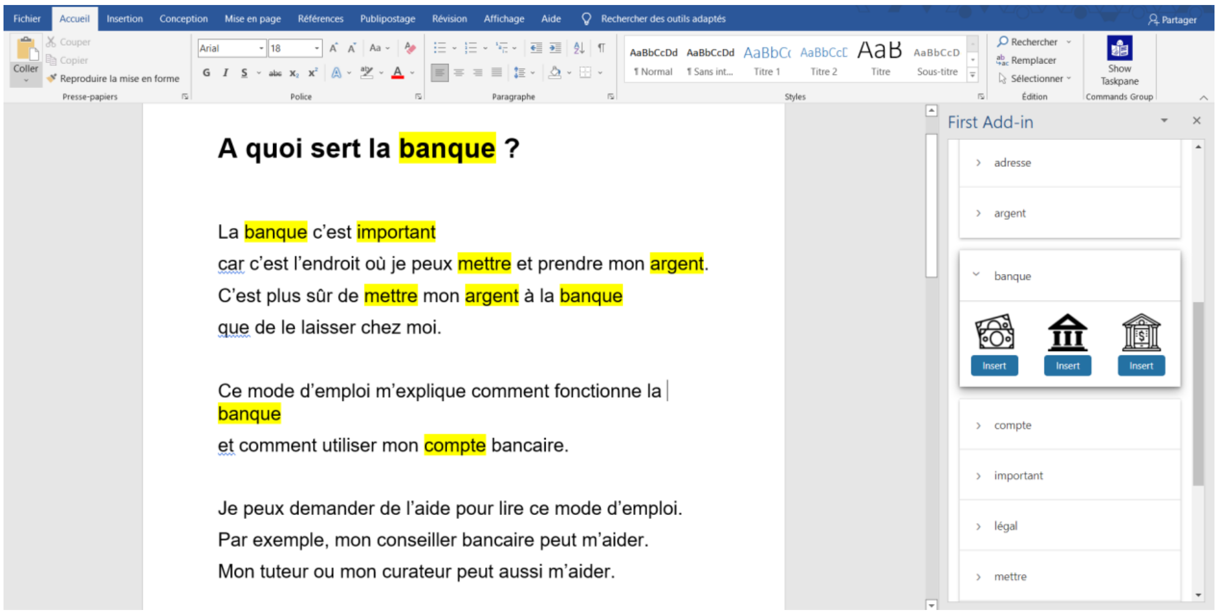
Le bac est un diplôme.

Un diplôme est un document officiel.
Ce document dit que votre enfant
a suivi des études
et qu'il a réussi ses examens.

Un examen est un test pour vérifier
que votre enfant a bien compris
ce qu'il a appris à l'école.

Votre enfant passe le bac

Text 1



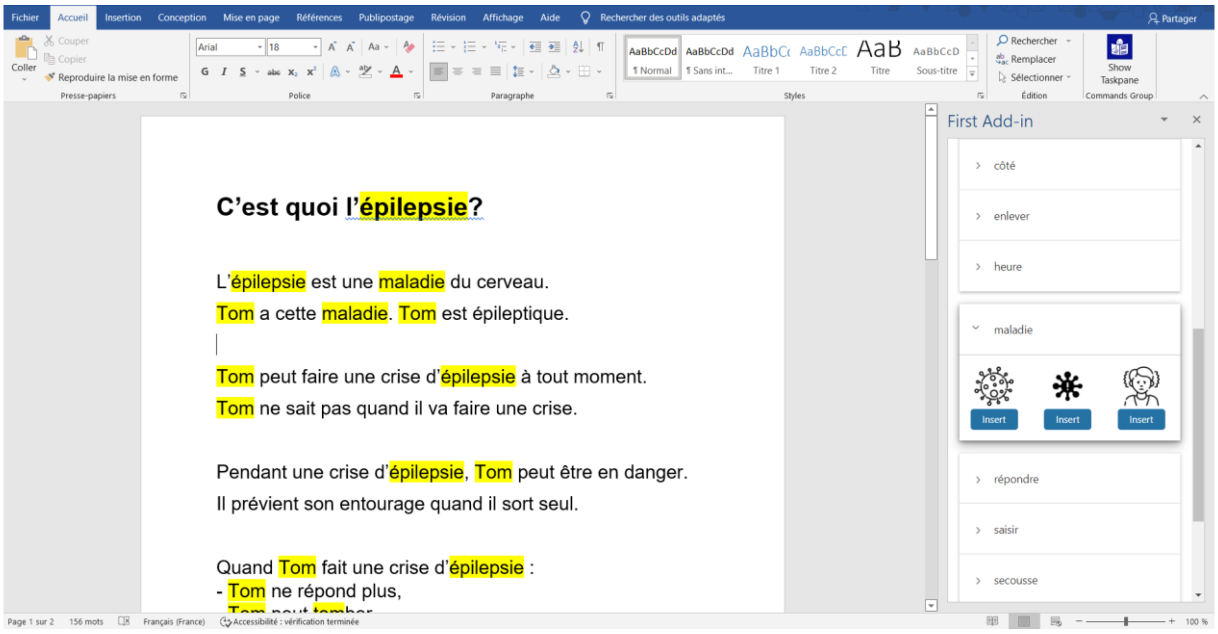
A quoi sert la banque ?

La banque c'est important
car c'est l'endroit où je peux mettre et prendre mon argent.
C'est plus sûr de mettre mon argent à la banque
que de le laisser chez moi.

Ce mode d'emploi m'explique comment fonctionne la
banque
et comment utiliser mon compte bancaire.

Je peux demander de l'aide pour lire ce mode d'emploi.
Par exemple, mon conseiller bancaire peut m'aider.
Mon tuteur ou mon curateur peut aussi m'aider.

Text 2



C'est quoi l'épilepsie?

L'épilepsie est une maladie du cerveau.
Tom a cette maladie. Tom est épileptique.

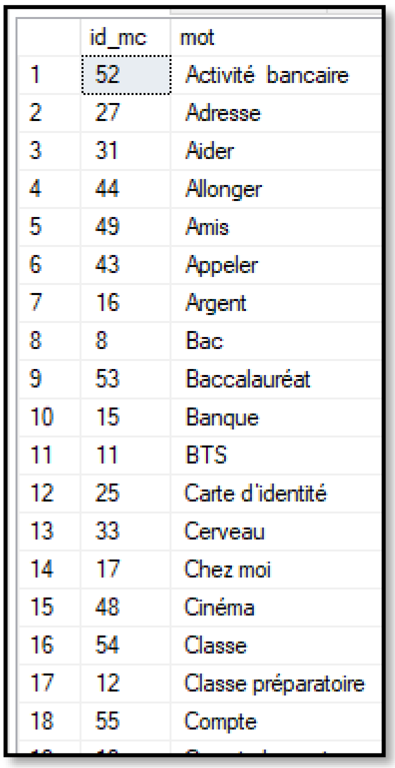
Tom peut faire une crise d'épilepsie à tout moment.
Tom ne sait pas quand il va faire une crise.

Pendant une crise d'épilepsie, Tom peut être en danger.
Il prévient son entourage quand il sort seul.

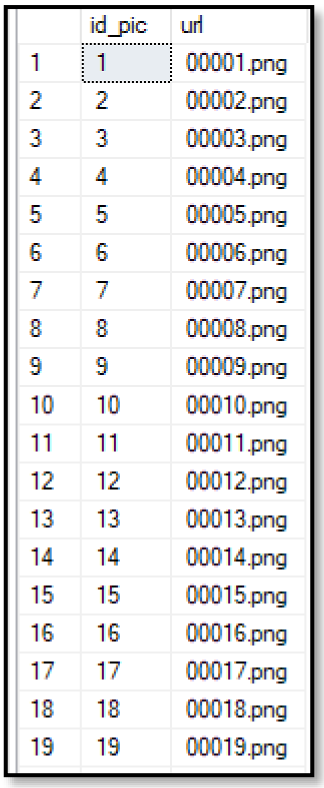
Quand Tom fait une crise d'épilepsie :
- Tom ne répond plus,

Text 3

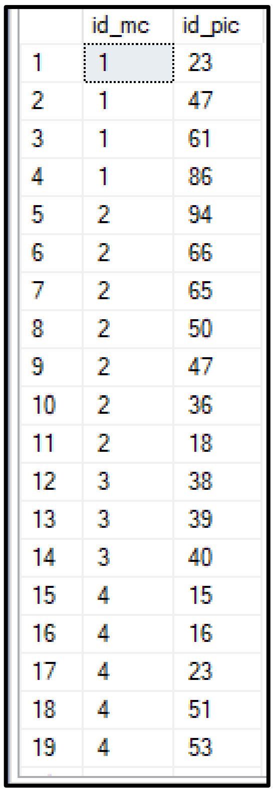
- DataBase tables :



	id_mc	mot
1	52	Activité bancaire
2	27	Adresse
3	31	Aider
4	44	Allonger
5	49	Amis
6	43	Appeler
7	16	Argent
8	8	Bac
9	53	Baccalauréat
10	15	Banque
11	11	BTS
12	25	Carte d'identité
13	33	Cerveau
14	17	Chez moi
15	48	Cinéma
16	54	Classe
17	12	Classe préparatoire
18	55	Compte

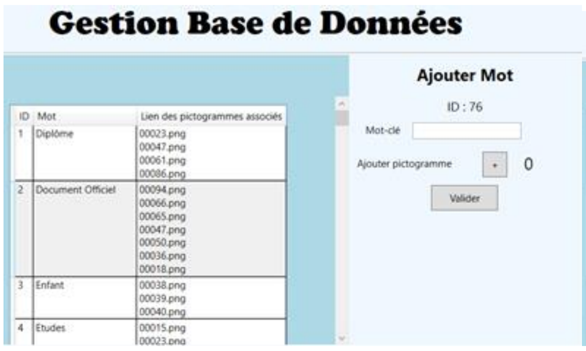


	id_pic	url
1	1	00001.png
2	2	00002.png
3	3	00003.png
4	4	00004.png
5	5	00005.png
6	6	00006.png
7	7	00007.png
8	8	00008.png
9	9	00009.png
10	10	00010.png
11	11	00011.png
12	12	00012.png
13	13	00013.png
14	14	00014.png
15	15	00015.png
16	16	00016.png
17	17	00017.png
18	18	00018.png
19	19	00019.png



	id_mc	id_pic
1	1	23
2	1	47
3	1	61
4	1	86
5	2	94
6	2	66
7	2	65
8	2	50
9	2	47
10	2	36
11	2	18
12	3	38
13	3	39
14	3	40
15	4	15
16	4	16
17	4	23
18	4	51
19	4	53

- C# database management software



Gestion Base de Données

ID	Mot	Lien des pictogrammes associés
1	Diplôme	00023.png 00047.png 00061.png 00086.png
2	Document Officiel	00094.png 00066.png 00065.png 00047.png 00050.png 00036.png 00018.png
3	Enfant	00038.png 00039.png 00040.png
4	Etudes	00015.png 00023.png

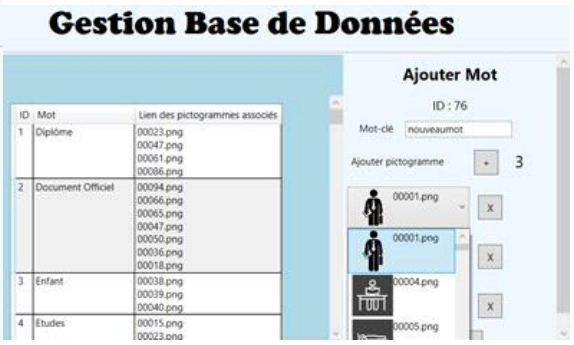
Ajouter Mot

ID : 76

Mot-clé

Ajouter pictogramme + 0

Valider



Gestion Base de Données

ID	Mot	Lien des pictogrammes associés
1	Diplôme	00023.png 00047.png 00061.png 00086.png
2	Document Officiel	00094.png 00066.png 00065.png 00047.png 00050.png 00036.png 00018.png
3	Enfant	00038.png 00039.png 00040.png
4	Etudes	00015.png 00023.png

Ajouter Mot

ID : 76

Mot-clé nouveaumot

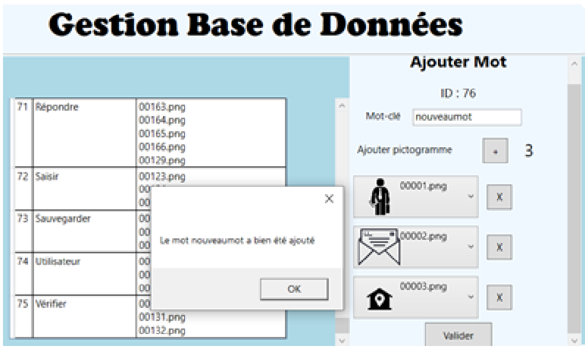
Ajouter pictogramme + 3

00001.png X

00001.png X

00004.png X

00005.png



Gestion Base de Données

71	Répondre	00163.png 00164.png 00165.png 00166.png 00129.png
72	Saisir	00123.png 00 00
73	Sauvegarder	00 00 00
74	Utilisateur	00 00 00
75	Vérifier	00 00131.png 00132.png

Le mot nouveaumot a bien été ajouté

OK

Ajouter Mot

ID : 76

Mot-clé nouveaumot

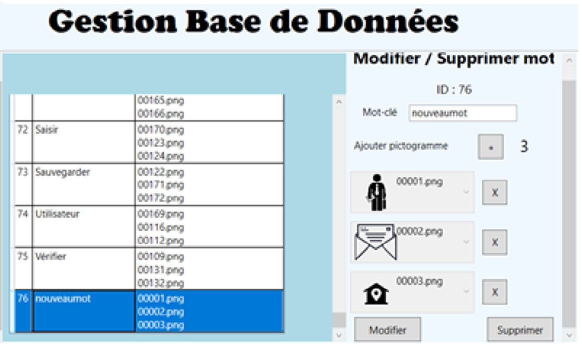
Ajouter pictogramme + 3

00001.png X

00002.png X

00003.png X

Valider



Gestion Base de Données

		00165.png 00166.png
72	Saisir	00170.png 00123.png 00124.png
73	Sauvegarder	00122.png 00171.png 00172.png
74	Utilisateur	00169.png 00116.png 00112.png
75	Vérifier	00109.png 00131.png 00132.png
76	nouveaumot	00001.png 00002.png 00003.png

Modifier / Supprimer mot

ID : 76

Mot-clé nouveaumot

Ajouter pictogramme + 3

00001.png X

00002.png X

00003.png X

Modifier Supprimer