

**Lita FIAHAU**

L3 Informatique T7

**Grâce FILITIKA**

L4 Informatique T7



# **Rapport de projet tuteuré**

Parseur d'un fichier de références  
bibliographiques



Tuteur de projet: **Mme.Tania Richmond**

# Sommaire

Introduction	3
Technologies utilisées	4
Langages de programmation	4
Les outils utilisés	5
Cahier des charges	6
Étapes du projet	8
Conceptualisation du projet	8
Réalisation	9
Exemple d'utilisation	10
Continuité	16
Conclusion	17
Références bibliographiques	18

# Introduction

Dans le cadre de notre Projet Tuteuré du semestre 5 et 6 de la formation Licence Informatique TREC 7, nous avons effectué un projet tuteuré sous le tutorat de Mme Tania Richmond, .

L'objectif de notre projet est de réaliser un outil permettant de parser et automatiser la gestion d'un fichier .bib de références bibliographiques pour la partie pratique. Parseur est un terme provenant de l'anglais "parser" désignant un outil logiciel permettant de parcourir un document et d'en extraire des informations.

Il s'agit donc dans ce projet d'automatiser ce processus d'extraction d'informations. Dans ce projet, nous avons travaillé qu'avec des fichiers bibliographiques d'extension .bib.

Cependant, un fichier .bib comporte plusieurs informations sur différents types de document, par exemple sur un article, un journal, etc.

Notre outil devra convenir à n'importe quelle personne souhaitant récupérer des informations sur un fichier de références bibliographiques.

Il s'agira dans ce rapport de vous présenter le contexte de ce projet ainsi que les outils qui ont été utilisés.

# Technologies utilisées

## Langage de programmation

Pour ce projet, nous avons choisi d'utiliser le langage python, qui nous convenait le plus et le plus simple pour nous pour coder.



Nous avons aussi utilisé dans nos fonctions la bibliothèque Python appelée Pybtex qui simplifie la manipulation des références bibliographiques au format BibTeX. Pybtex permet de lire, d'écrire et de traiter des fichiers BibTeX, ainsi que d'effectuer des opérations sur les données bibliographiques contenues dans ces fichiers. Par exemple, l'extraction des informations spécifiques à partir de fichiers bib, telles que les noms des auteurs, les titres des publications, les années de publication, etc.

## Les outils utilisés

Pour notre projet, nous avons utilisé différents outils:

- **DISCORD:**

Discord est une application de messagerie instantanée gratuite. Cette application nous a permis de dialoguer et de s'envoyer des fichiers.



- **Visual Studio Code:**

Visual Studio Code est un éditeur de code extensible. Il nous a permis d'avoir un environnement de développement afin d'exploiter notre code. De plus, ce logiciel possède un debugger intégré.



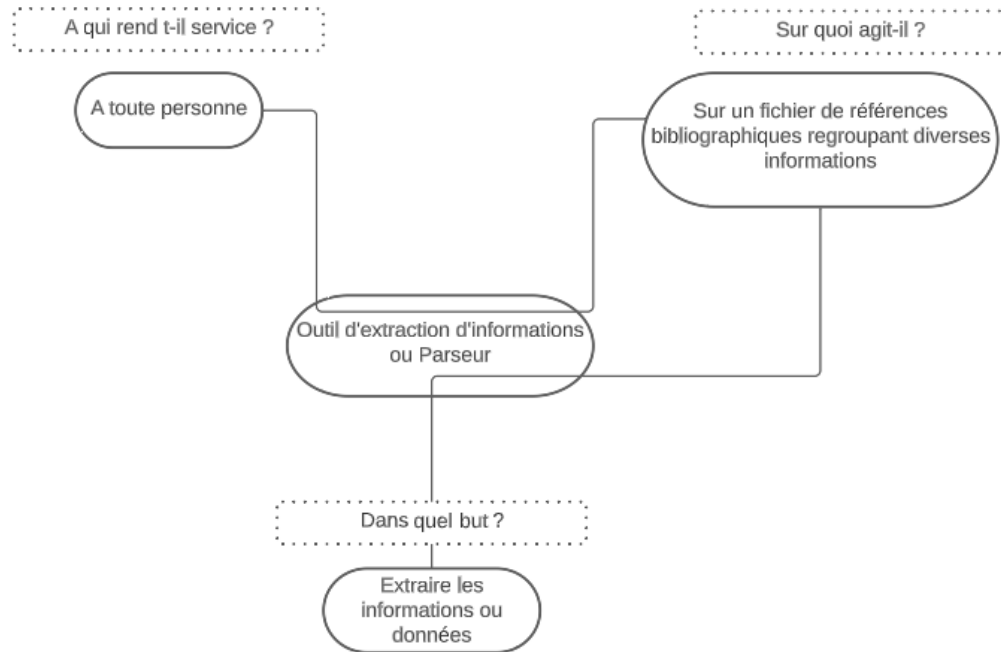
- **Github:**

GitHub est un service d'hébergement Open-Source, permettant aux programmeurs et aux développeurs de partager le code informatique de leurs projets afin de travailler dessus de façon collaborative. On peut le considérer comme un Cloud dédié au code informatique.



## Cahier des charges

### Diagramme bête à corne:



*Diagramme bête à corne*

Au centre nous avons notre outil d'extraction d'informations et celui-ci nous incite à nous poser trois questions principales:

- A qui rend-t-il service ?
- Sur quoi agit-il ?
- Dans quel but ?

### Architecture de l'outil:



*Architecture du parseur*

Grâce au fichier python nous avons extrait les données du fichier bib et créé un fichier texte contenant ces dernières.

## Composition d'un fichier .bib:

Type de document

Nom de la référence

Auteurs : séparés par "and" sous la forme  
Nom, Prénom ou Prénom Nom

Caractère LaTeX

Champs pour HAL

```
@article{ijci2015,  
  author = {Farhi, Laurence and Jaigu, Anne},  
  title = {Exemple de r\ef\erence},  
  journal = {Titre de la revue},  
  abstract = {abstract simple},  
  volume = {8},  
  pages = {77-88},  
  year = {2015},  
  doi = {10.1007/s00186-012-0389-2},  
  pdf = {http://www.irisa.fr/sumo/Publis/PDF/jdedes-opacity-2014.pdf},  
  keywords = {mot1 ; mot2},  
  x-audience = {international},  
  x-language = {en},  
}
```

Organisation d'un fichier bib

# Étapes du projet

## Conceptualisation du projet

La partie la plus importante du projet était la récupération des données d'un fichier de références bibliographiques et ensuite écrire toutes ces informations dans un fichier texte créé ou existant.

Après la réalisation de cette partie et avec l'aide de notre tuteur, Mme. Tania Richmond, il a été convenu de réaliser d'autres fonctions qui permettent de faire une sorte de tri c'est-à-dire que l'utilisateur pourra choisir s'il veut trier les données :

- Dans l'ordre alphabétique en fonction du nom de l'auteur
- Dans l'ordre alphabétique en fonction du type de document
- Dans l'ordre chronologique en fonction des dates (du plus ancien au plus récent)

Ainsi, l'utilisateur pourra exécuter une de ces fonctions en fonction de ses préférences et avoir son fichier texte avec toutes les informations souhaitées triées ou pas.



## Réalisation

Pour la phase de réalisation, nous avons passé quelques semaines à comprendre le but du parseur, la composition d'un fichier bib et comment pourrions nous extraire les informations dont nous avons besoin. Pour cela, nous avons effectué diverses recherches sur Internet.

Tout d'abord nous avons récupéré tout le contenu du fichier bib grâce à la commande:

`pybtex.database.parse_string(fichier_bib.read(),"",bib_format='bibtex')` et sur le terminal de vscode, cela nous a retourné une liste de données avec toutes les informations de chaque document dont voici un exemple ci-dessous:

```
BibliographyData(
  entries=OrderedCaseInsensitiveDict([
    ('Bar94', Entry('article',
      fields=[
        ('Title', 'Some new {NP}-complete coding problems'),
        ('Journal', 'Problemy Peredachi Informatsii'),
        ('Year', '1994'),
        ('Number', '3'),
        ('Pages', '23-28'),
        ('Volume', '30'),
        ('Abstract', 'We prove the NP-completeness of the basic decision problems for ternary linear codes. In particular, we prove that finding out whether a ternary linear code contains a vector of weight equal to the code length is NP-complete. In addition, we prove the hardness of minimum distance decoding for linear product codes with nontrivial factors.'),
        ('Language', 'Russian'),
        ('Publisher', 'Russian Academy of Sciences, Branch of Informatics, Computer Equipment and Automatization'),
        persons=OrderedCaseInsensitiveDict([('Author', [Person('Barg, S')]))]),
    ('McE178', Entry('article',
      fields=[
        ('Title', 'On the inherent intractability of certain coding problems'),
        ('Journal', 'IEEE Transactions on Information Theory'),
        ('Year', '1978'),
        ('Month', 'May'),
        ('Number', '3'),
        ('Pages', '384-386'),
        ('Volume', '24'),
        ('Abstract', 'The fact that the general decoding problem for linear codes and the general problem of finding the weights of a linear code are both NP-complete is shown. This strongly suggests, but does not rigorously imply, that no algorithm for either of these problems which lulls in polynomial time exists.'),
        persons=OrderedCaseInsensitiveDict([('Author', [Person('Berlekamp, Elwyn R.'), Person('McEliece, Robert James'), Person('van Tilborg, Henk C. A.')]))]),
    ('BCS13', Entry('incollection',
      fields=[
        ('Title', '{McBits}: Fast Constant-Time Code-Based Cryptography'),
        ('Booktitle', 'Cryptographic Hardware and Embedded Systems (CHES 2013)'),
        ('Publisher', 'Springer'),
        ('Year', '2013'),
        ('Address', 'Berlin, Heidelberg'),
```

*Capture d'écran du terminal affichant les informations du fichier bib.*

Ensuite, nous avons parcouru chaque élément de cette liste et nous avons procédé à la mise en page pour le fichier texte. En effet, lorsque nous avons récupéré le premier élément, celui-ci correspond à la clé d'entrée appelée aussi nom de la référence, ainsi que le deuxième élément qui est le type du document, etc...

Après avoir récupéré chaque champ du document, ces derniers sont ensuite stockés dans une liste initialement vide. Puis, chaque élément contenu dans la liste est parcouru et est ajouté un par un dans le fichier texte.

## Exemple d'utilisation

Tout d'abord nous avons un fichier bib avec plusieurs informations sur différents types de documents :

```
@Article{Bar94,
  Title       = {Some new {NP}-complete coding problems},
  Author      = {Barg, S},
  Journal     = {Problemy Peredachi Informatsii},
  Year       = {1994},
  Number     = {3},
  Pages      = {23--28},
  Volume     = {30},

  Abstract    = {We prove the NP-completeness of the basic decision problems for ternary linear codes. In particular, we prove that fin
  Language    = {Russian},
  Publisher   = {Russian Academy of Sciences, Branch of Informatics, Computer Equipment and Automatization}
}

@Article{BMcEVT78,
  Title       = {On the inherent intractability of certain coding problems},
  Author      = {Berlekamp, Elwyn R. and McEliece, Robert James and van Tilborg, Henk C. A.},
  Journal     = {IEEE Transactions on Information Theory},
  Year       = {1978},

  Month      = {May},
  Number     = {3},
  Pages      = {384-386},
  Volume     = {24},

  Abstract    = {The fact that the general decoding problem for linear codes and the general problem of finding the weights of a linear
}

@InCollection{BCS13,
  Title       = {{McBits}: Fast Constant-Time Code-Based Cryptography},
  Author      = {Bernstein, Daniel J. and Chou, Tung and Schwabe, Peter},
  Booktitle   = {Cryptographic Hardware and Embedded Systems (CHES 2013)},
  Publisher   = {Cryptology}
```

### Fichier .bib

Ensuite, on lance le code python sans tri, avec en paramètres le nom du fichier bib et le nom du fichier texte.

```
parseur.py > ...
1  # installer d'abord la bibliothèque pybtex sur vscode avec la commande "pip install pybtex"
2  # importer la bibliothèque
3  import pybtex.database
4
5  contenu = []
6
7  # Ouvrir le fichier .bib
8  with (open('sca_cbc.bib', 'r') as fichier_bib, open('parseur.txt', "w") as fichier_txt):
9      # Charger les données bibliographiques
10     bib_data = pybtex.database.parse_string(fichier_bib.read(), bib_format='bibtex')
11
12     # Parcourir chaque entrée bibliographique
13     for entry_key in bib_data.entries:
14         entry = bib_data.entries[entry_key]
15         authors = entry.persons.get("author", [])
16
17         # Affichage des données
18         contenu.append("Cle d'entree: " + entry_key)
19
20         contenu.append("Type: " + entry.type.capitalize())
21
22         contenu.append("Champs: ")
23
24         for field in entry.fields:
25             contenu.append("  "+ field+ ": " + entry.fields[field])
26         contenu.append("  Auteur(s): ")
27         for author in authors:
28             contenu.append("    " + str(author))
29         contenu.append("\n" + "---" + "\n")
30
31     for element in contenu:
32         fichier_txt.write(element+"\n")
33     fichier_txt.close()
34
35
36
```

### Fichier parseur.py

Après l'exécution du code, un fichier texte est créé s'il n'existe pas sinon il écrit dans le fichier existant déjà dans le même répertoire que le fichier du code.

Nom	Modifié le	Type	Taille
.vscode	05/04/2023 23:56	Dossier de fichiers	
__pycache__	03/05/2023 00:48	Dossier de fichiers	
ProjetTuteur	24/05/2023 13:57	Dossier de fichiers	
venv	10/05/2023 16:36	Dossier de fichiers	
bib_data	31/05/2023 09:54	Fichier PNG	43 Ko
parseur	31/05/2023 10:01	Fichier source Pyth...	2 Ko
parseur	31/05/2023 10:33	Document texte	27 Ko

### Création du fichier texte parseur.txt dans le répertoire

```

parseur - Bloc-notes
Fichier  Modifier  Affichage

Cle d'entree: Bar94
Type: Article
Champs:
  Title: Some new {NP}-complete coding problems
  Journal: Problemy Peredachi Informatsii
  Year: 1994
  Number: 3
  Pages: 23--28
  Volume: 30
  Abstract: We prove the NP-completeness of the basic decision problems for ternary linear codes. In particular, we prove that finding out whether a ternary linear code contains a vector
  Language: Russian
  Publisher: Russian Academy of Sciences, Branch of Informatics, Computer Equipment and Automatization
  Auteur(s):
    Barg, S
---

Cle d'entree: BMcEvT78
Type: Article
Champs:
  Title: On the inherent intractability of certain coding problems
  Journal: IEEE Transactions on Information Theory
  Year: 1978
  Month: May
  Number: 3
  Pages: 384-386
  Volume: 24
  Abstract: The fact that the general decoding problem for linear codes and the general problem of finding the weights of a linear code are both NP-complete is shown. This strongly suggests
  Auteur(s):
    Berlekamp, Elwyn R.
    McEliece, Robert James
    van Tilborg, Henk C. A.
---

Cle d'entree: BCS13
Type: Incollection
Champs:
  Title: /McRittel: Fast Constant-Time Code-Based Counterparty

```

### Fichier parseur.txt










L'utilisateur pourra lancer aussi le code "trier\_par\_type.py" qui va lui permettre de retourner d'avoir un fichier texte avec toutes les informations sur les documents mais trier en fonction du type de document. Par exemple, les documents de type article seront affichés avant les documents de type journal.

```

trier_par_type.py > ...
1  # installer d'abord la bibliothèque pybtex sur vscode avec la commande "pip install pybtex"
2  # importer la bibliothèque
3  import pybtex.database
4
5  contenu=[]
6
7  # Ouvrir le fichier .bib en lecture
8  with open('sca_cbc.bib', 'r') as fichier_bib:
9      # Charger les données bibliographiques
10     bib_data = pybtex.database.parse_string(fichier_bib.read(), bib_format='bibtex')
11
12  # Fonction pour le tri par type
13  def trier_par_type(entry_key):
14      entry = bib_data.entries[entry_key]
15      return entry.type
16
17  # Trier les entrées par ordre alphabétique du type d'article
18  donnees_triees = sorted(bib_data.entries, key=trier_par_type)
19
20  # Ouvrir le fichier texte en écriture
21  with open('type.txt', 'w') as fichier_txt:
22      # Parcourir et écrire les entrées triées dans le fichier texte
23      for entry_key in donnees_triees:
24          entry = bib_data.entries[entry_key]
25          authors = entry.persons.get("author", [])
26
27          # Affichage des données
28          contenu.append("Cle d'entree: " + entry_key)
29
30          contenu.append("Type: " + entry.type.capitalize())
31
32          contenu.append("Champs: ")
33
34          for field in entry.fields:
35              contenu.append("    "+ field+ ": " + entry.fields[field])
36          contenu.append("    Auteur(s): ")
37          for author in authors:
38              contenu.append("        " + str(author))
39          contenu.append("\n" + "---" + "\n")
40
41
42  for element in contenu:
43      fichier_txt.write(element+"\n")
44  fichier_txt.close()
45

```

*Fichier trier par type.py*

 parseur	31/05/2023 10:01	Fichier source Pyth...	2 Ko
 parseur	31/05/2023 10:33	Document texte	27 Ko
 Publications_TR	09/05/2023 20:21	Fichier source BibT...	28 Ko
 sca_cbc	09/05/2023 20:21	Fichier source BibT...	32 Ko
 Schéma projet tuteuré	28/05/2023 18:00	Microsoft Edge PD...	17 Ko
 test2	25/05/2023 14:23	Fichier source Pyth...	1 Ko
 trier_par_annee	31/05/2023 11:08	Fichier source Pyth...	2 Ko
 trier_par_type	30/05/2023 23:12	Fichier source Pyth...	2 Ko
 type	31/05/2023 13:39	Document texte	27 Ko

*Création du fichier texte type.txt dans le répertoire*

```

Cle d'entree: Bar94
Type: Article
Champs:
  Title: Some new {NP}-complete coding problems
  Journal: Problemy Peredachi Informatsii
  Year: 1994
  Number: 3
  Pages: 23--28
  Volume: 30
  Abstract: We prove the NP-completeness of the basic decision problems for ternary linear codes. In particular, we prove that finding out whether a ternary linear code contains a vector
  Language: Russian
  Publisher: Russian Academy of Sciences, Branch of Informatics, Computer Equipment and Automatization
  Auteur(s):
    Barg, S

---

Cle d'entree: BMCvT78
Type: Article
Champs:
  Title: On the inherent intractability of certain coding problems
  Journal: IEEE Transactions on Information Theory
  Year: 1978
  Month: May
  Number: 3
  Pages: 384-386
  Volume: 24
  Abstract: The fact that the general decoding problem for linear codes and the general problem of finding the weights of a linear code are both NP-complete is shown. This strongly suggests
  Auteur(s):
    Berlekamp, Elwyn R.
    McEliece, Robert James
    van Tilborg, Henk C. A.

---

Cle d'entree: CEVMS16a
Type: Article
Champs:
  Title: Machine Large Keys in Hardware: A Masked Implementation of {McEliece}

```

Fichier type.txt












L'utilisateur pourra également exécuter le code "trier\_par\_annee.py" qui va lui retourner un fichier texte avec toutes les informations sur les documents mais trier en fonction du type de l'année de publication. Par exemple, les documents les plus anciens seront affichés avant les documents récents.

```

trier_par_annee.py > ...
1  import pybtex.database
2
3  contenu=[]
4  # Ouvrir le fichier .bib en lecture
5  with open('sca_cbc.bib', 'r') as fichier_bib:
6      # Charger les données bibliographiques
7      bib_data = pybtex.database.parse_string(fichier_bib.read(), bib_format='bibtex')
8
9  # Fonction de tri personnalisée
10 # Fonction python pour voir si la valeur est composée que de chiffres
11 def trier_par_annee(entry_key):
12     entry = bib_data.entries[entry_key]
13     year = entry.fields.get('year', '')
14     if year.isdigit():
15         return int(year)
16     else:
17         return 0
18
19 # Trier les entrées en fonction des années
20 donnees_triees = sorted(bib_data.entries, key=trier_par_annee)
21
22 # Ouvrir le fichier texte en écriture
23 with open('annee.txt', 'w') as fichier_txt:
24     # Parcourir et écrire les entrées triées dans le fichier texte
25     for entry_key in donnees_triees:
26         entry = bib_data.entries[entry_key]
27         authors = entry.persons.get("author", [])
28
29         # Affichage des données
30         contenu.append("Cle d'entree: " + entry_key)
31
32         contenu.append("Type: " + entry.type.capitalize())
33
34         contenu.append("Champs: ")
35
36         for field in entry.fields:
37             contenu.append("    " + field + ": " + entry.fields[field])
38         contenu.append("    Auteur(s): ")
39         for author in authors:
40             contenu.append("        " + str(author))
41         contenu.append("\n" + "---" + "\n")
42
43     for element in contenu:
44         fichier_txt.write(element+"\n")
45     fichier_txt.close()

```

Fichier trier par annee.py

	annee	31/05/2023 13:42	Document texte	27 Ko
	bib_data	31/05/2023 09:54	Fichier PNG	43 Ko
	parseur	31/05/2023 10:01	Fichier source Pyth...	2 Ko
	parseur	31/05/2023 10:33	Document texte	27 Ko
	Publications_TR	09/05/2023 20:21	Fichier source BibT...	28 Ko
	sca_cbc	09/05/2023 20:21	Fichier source BibT...	32 Ko
	Schéma projet tuteuré	28/05/2023 18:00	Microsoft Edge PD...	17 Ko
	test2	25/05/2023 14:23	Fichier source Pyth...	1 Ko
	trier_par_annee	31/05/2023 11:08	Fichier source Pyth...	2 Ko
	trier_par_type	30/05/2023 23:12	Fichier source Pyth...	2 Ko
	type	31/05/2023 13:39	Document texte	27 Ko

Création du fichier texte annee.txt dans le répertoire

annee - Bloc-notes

Fichier Modifier Affichage

Cle d'entree: McWS77Book  
 Type: Book  
 Champs:  
 Title: The theory of error-correcting codes  
 Publisher: North-Holland  
 Year: 1977  
 Auteur(s):  
 MacWilliams, Florence Jessie  
 Sloane, Neil James Alexander

---

Cle d'entree: BMcEvT78  
 Type: Article  
 Champs:  
 Title: On the inherent intractability of certain coding problems  
 Journal: IEEE Transactions on Information Theory  
 Year: 1978  
 Month: May  
 Number: 3  
 Pages: 384-386  
 Volume: 24  
 Abstract: The fact that the general decoding problem for linear codes and the general problem of finding the weights of a linear code are both NP-complete is shown. This strongly suggests  
 Auteur(s):  
 Berlekamp, Elwyn R.  
 McEliece, Robert James  
 van Tilborg, Henk C. A.

---

Cle d'entree: McE78  
 Type: Techreport  
 Champs:  
 Title: A public-key cryptosystem based on algebraic coding theory  
 Institution: California Inst. Technol.  
 Year: 1978  
 Address: Pasadena, CA  
 Month: January  
 Number: 11

*Fichier annee.txt*

## Continuité

Dans un premier temps, la chose que l'on pourrait faire par la suite serait par exemple de créer une page web où une personne pourra importer un fichier bib et cliquer sur un bouton "parser" et ainsi pouvoir télécharger son fichier texte contenant toutes les informations.

Nous pourrions ensuite ajouter une liste déroulante où la personne pourra choisir si elle veut trier les informations du fichier bib dans l'ordre alphabétique en fonction du nom de l'auteur ou autres.

Ne disposant pas assez de temps pour la réalisation de cette page, celle-ci pourrait être une idée pour la suite de notre projet.



## Conclusion

Pour conclure, malgré le retard pris par les problèmes d'environnement et l'utilisation de l'outil github nous avons tout de même pu réaliser un code python pouvant renvoyer des résultats similaires à ce que pourrait faire un parseur.

Ce projet nous a permis de découvrir ce qu'est un parseur ainsi que son objectif et sur la composition des fichiers de références bibliographiques et comment les récupérer.

Nous avons su communiquer et nous entraider en équipe, afin de rendre, le mieux possible ce qui a été demandé.

# Références Bibliographiques

<https://doc.archives-ouvertes.fr/x2hal/bibtex-exemple/>

<https://bib.umontreal.ca/ancien-bibtex/debuter/#:~:text=D%C3%A9buter%20avec%20BibTeX-.Qu'est%2Dce%20que%20BibTex%3F,fichier%20portant%20l'extension%20BIB.>

<https://www.overleaf.com/latex/templates/tagged/bibliography>

<https://www.overleaf.com/project/6475f1ee72aa70e09b884b58?&nocdn=true>

<https://github.com/petercorke/pybib/blob/master/BibEntry.py>

<https://pybtex.org/>