

Projet NoSQL & PYTHON

Groupe 1

Sommaire

1. LE PROJET

- a. Contexte
- b. Objectif
- c. Equipe
- d. Phasing
- e. Outils

2. LES DONNEES

- a. SQL
- b. NoSQL

3. L'API PYTHON

- a. Application
- b. Code

4. LES PERSPECTIVES

5. ANNEXES

Le projet DIGISCHOOLS



A l'école, nous avons des
données en SQL et nous aussi
on veut faire du Big Data en
NoSQL!

Adressons-nous à Diginamic !

Objectifs

- L'interprétation de l'expression des besoins
- L'adéquation des spécifications techniques
- L'explication et l'argumentation de la solution adoptée
- Le bon fonctionnement de la réalisation

Le projet - Equipe



Christophe G.
Le yūdansha



Robin H.
Le guide

DEV



Zeyneb M.
2n mother



Kévin T.
Snake charmer

2024-07-26



Aurélie M.
Queen Git

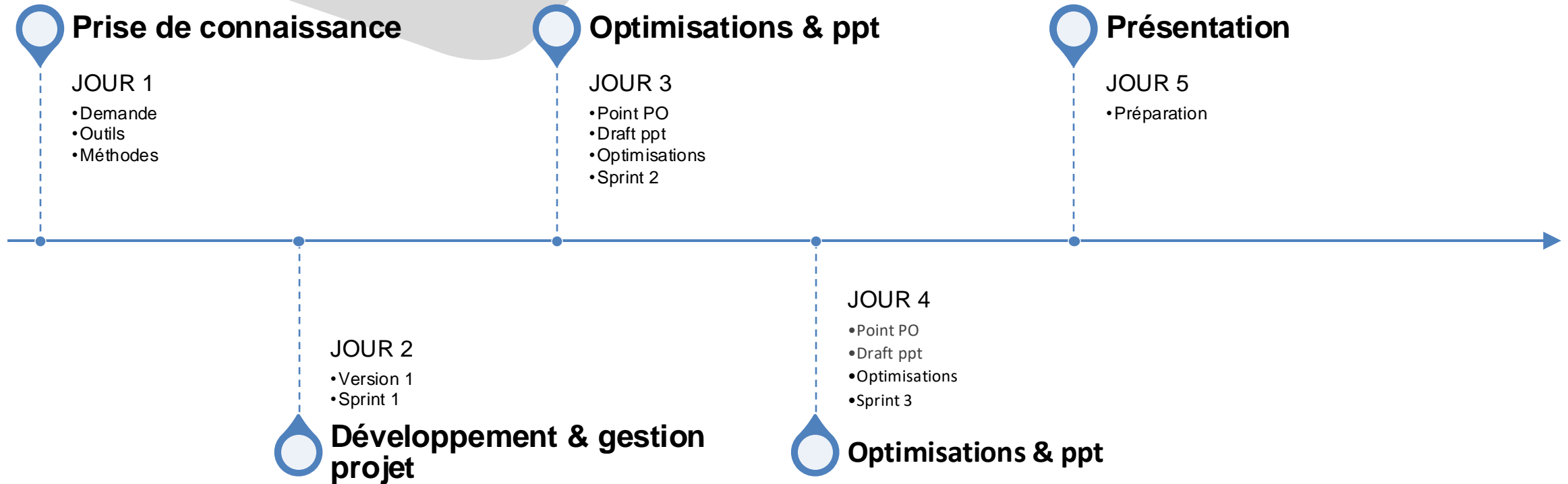
ITIL



Laurent E.
Chaos manager

CDP

Le projet - Phasing



Le projet - Phasing

Jira Planification

JUIL.

2223242526272829

Ta...Ta...Ta...

🔄 Tableau Sprint 1

SPRINT FERMÉ

Mise en place MongoDB + FastAPI

Début du sprint2024/07/23Fin du sprint2024/07/23

🔍

LEAMKTOM

👤

🔊 Donner votre avis

🔗 Partager

🔍 Filtre

👤 Groupe

⋮ Plus

#	Clé	📄 Résumé	🔄 État	🔄 Sprint	@ Assigné(e)	+
	PF-1	Import .sql dans MySQL	TERMINÉ(E)	👍 Tableau Sprint 1	LE Laurent EUD	
	PF-2	Export données MySQL en json / .csv	TERMINÉ(E)	👍 Tableau Sprint 1	LE Laurent EUD	
	PF-3	Import json / .csv dans Mongo	TERMINÉ(E)	👍 Tableau Sprint 1	LE Laurent EUD	
<input type="checkbox"/>	PF-4	Créer un projet FastAPI / pymongo	TERMINÉ(E)	👍 Tableau Sprint 1	KT Kevin Tartou	
<input type="checkbox"/>	PF-5	Endpoint liste des professeurs	TERMINÉ(E)	👍 Tableau Sprint 1	KT Kevin Tartou	
<input type="checkbox"/>	PF-6	Endpoint liste des élèves par classe	TERMINÉ(E)	👍 Tableau Sprint 1	KT Kevin Tartou	
<input type="checkbox"/>	PF-7	Endpoint liste des notes (d'un élève donné)	TERMINÉ(E)	👍 Tableau Sprint 1	OM MERIBAI	
<input type="checkbox"/>	PF-8	Endpoint liste des élèves et leurs notes (d'un prof donné)	TERMINÉ(E)	👍 Tableau Sprint 1	KT Kevin Tartou	

Le projet - Les outils

Versionning

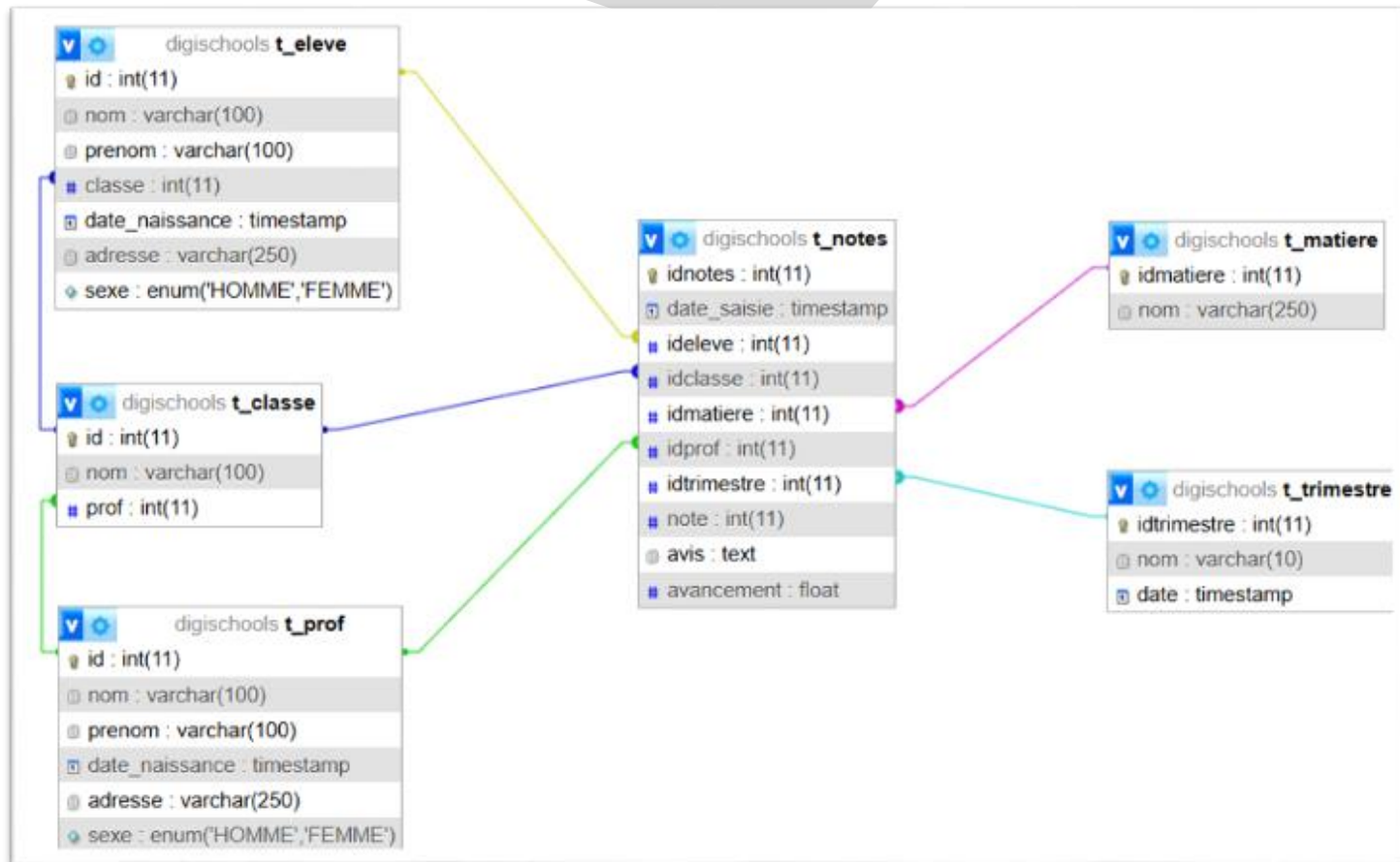
BDD SQL	BDD NoSQL	Coding	Gestion de projet
<p>XAMPP Version 8.2.12</p> <ul style="list-style-type: none">+ Apache 2.4.58+ MariaDB 10.4.32+ PHP 8.2.12 (VS16 X86 64bit thread safe) + PEAR+ phpMyAdmin 5.2.1+ OpenSSL 1.1.1p+ ADOdb 518a+ Mercury Mail Transport System v4.63+ FileZilla FTP Server 0.9.41+ Webalizer 2.23-04+ Strawberry Perl 5.32.1.1 Portable+ Tomcat 8.5.96+ XAMPP Control Panel Version 3.3.0.+ XAMPP mailToDisk 1.0 <p>Serveur de base de données</p> <ul style="list-style-type: none">• Serveur : 127.0.0.1 via TCP/IP• Type de serveur : MariaDB• Connexion au serveur : SSL n'est pas utilisé• Version du serveur : 10.4.32-MariaDB - mariadb.org binary distribution• Version du protocole : 10• Utilisateur : root@localhost• Jeu de caractères du serveur : UTF-8 Unicode (utf8mb4) <p>Serveur Web</p> <ul style="list-style-type: none">• Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12• Version du client de base de données : libmysql - mysqlnd 8.2.12• Extension PHP : mysqli curl mbstring• Version de PHP : 8.2.12 <p>PhpMyAdmin</p> <ul style="list-style-type: none">• Version : 5.2.1 (à jour)	<p>MongoDB Atlas 7.0.12</p> <p>MongoDB Compass 1.43.4</p> <p>MongoDB Shell v2.2.10</p>	<p>PyCharm 2024.1.2 (Community Edition)</p> <p>Python 3.12.4</p> <p>pip 24.1.2</p> <p>uvicorn 0.30.3</p> <p>FastAPI CLI version: 0.0.4</p> <p>dotenv, version 1.0.1</p> <p>Pymongo 4.8.0</p>	<p>Jira Free</p> <p>GitHub Free Plan</p> <p>Powerpoint Microsoft 365</p>



LES DONNEES

Les données - SQL

➤ Modèle de la base fournie - digischools.sql



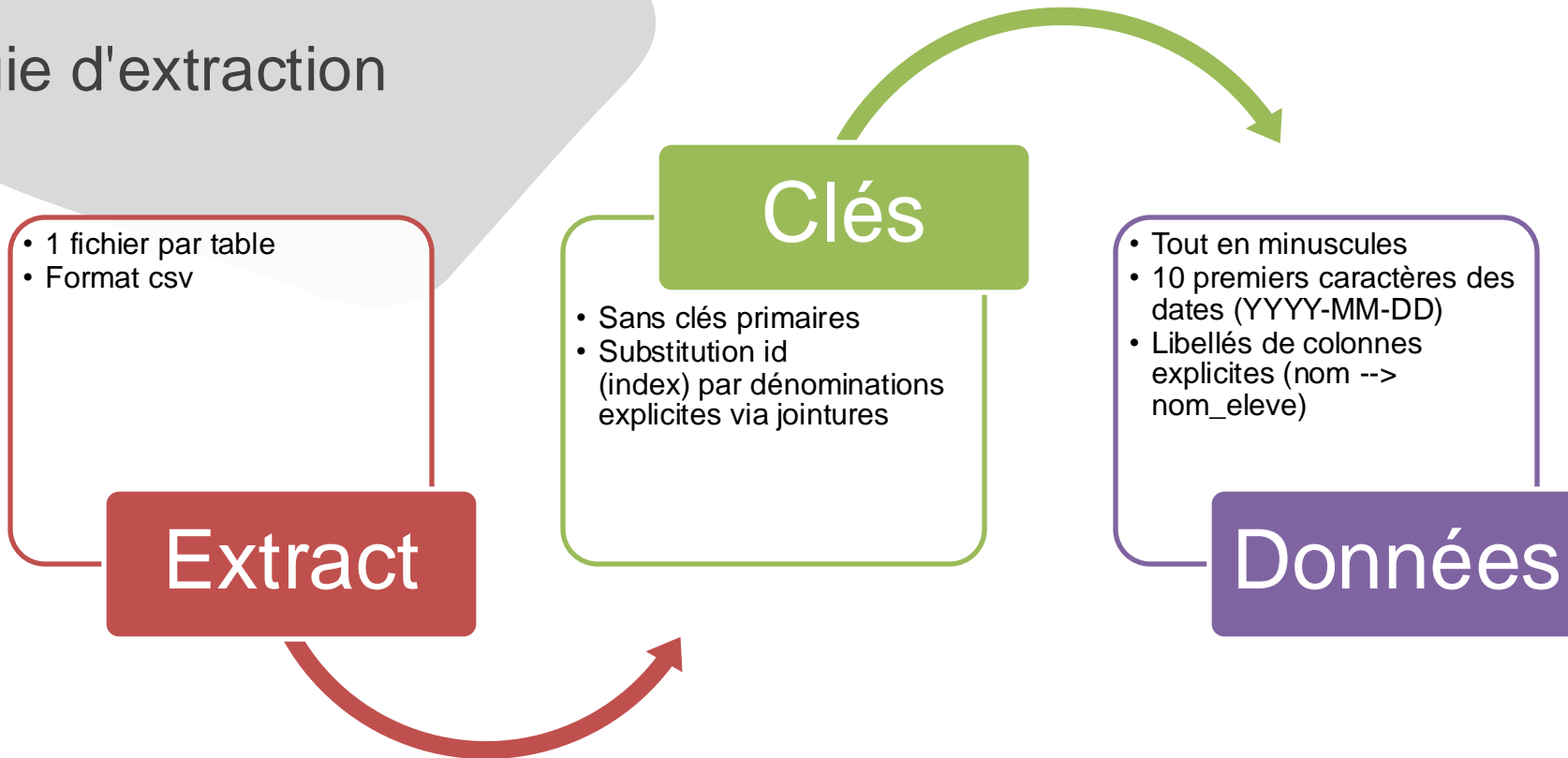
6 tables :

- 1 centrale (t_notes)
- 2 "périphériques" (t_matière & t_trimestre)
- 3 "liées" (t_eleve, t_classe & t_prof)

Lignes	Type	Interclassement	Taille	Perte
72	InnoDB	Latin_swedish_ci	208,0 kio	0

Les données - SQL

➤ Stratégie d'extraction



➤ Exemple sur "t_notes" (tous les cas en annexe 1)

```
SELECT substr(date_saisie,1,10) as 'date_saisie', lower(t_eleve.nom) as 'nom_eleve', lower(t_eleve.prenom) as 'prenom_eleve', lower(t_classe.nom) as 'nom_classe', lower(t_matiere.nom) as 'nom_matiere', lower(t_prof.nom) as 'nom_prof', lower(t_prof.prenom) as 'prenom_prof', lower(t_trimestre.nom) as 'nom_trimestre', note, lower(avis) as 'avis', avancement FROM t_notes JOIN t_eleve on t_notes.idleve = t_eleve.id JOIN t_classe on t_notes.idclasse = t_classe.id JOIN t_matiere on t_notes.idmatiere = t_matiere.idmatiere JOIN t_prof on t_notes.idprof = t_prof.id JOIN t_trimestre on t_notes.idtrimestre = t_trimestre.idtrimestre;
```

Les données - NoSQL

- Stratégie d'injection des données dans une BDD NoSQL orientée documents (MongoDB)
 - Création d'un user pour partage de connexion sur Atlas
 - Création via Compass d'une base de données "digischools"
 - Création des collections représentant les 6 tables originales
 - Import du fichier csv ad hoc dans chaque collection

The screenshot shows the MongoDB Compass interface for a database named 'digischools'. On the left, a sidebar lists the collections: 'classe', 'eleve', 'matiere', 'notes', and 'trimestre'. The main panel displays details for the 'classe' collection, including its storage size (4.10 kB), number of documents (5), average document size (87.00 B), and index information. Below this, a table lists the other collections: 'eleve' (53 documents, 203.00 B avg size), 'matiere' (5 documents, 53.00 B avg size), and 'notes' (3 documents, 283.00 B avg size). On the right, the 'Documents' tab is active, showing a query editor with a sample document for a teacher.

Collection	Storage size	Documents	Avg. document size	Indexes	Total index size
classe	4.10 kB	5	87.00 B	1	4.10 kB
eleve	4.10 kB	53	203.00 B	1	4.10 kB
matiere	4.10 kB	5	53.00 B	1	4.10 kB
notes	4.10 kB	3	283.00 B	1	4.10 kB

```
{
  "_id": ObjectId('66a2232d9bf358135aac77c5'),
  "nom_prof": "germain",
  "prenom_prof": "christophe",
  "date_naissance_prof": "1971-01-02T00:00:00.000+00:00",
  "adresse_prof": "15 rue du printemps 59000 lille",
  "sexe_prof": "homme"
}
```

Les données - NoSQL

- Mise en place de schémas JSON via MongoDB Compass
 - Objectif de validation et de renforcement de la cohérence

- Exemple sur la collection "notes" (tous les cas en annexe 2)

cluster0.3qz2vxx.mongodb.net > digischools > notes

Documents 3 Aggregations Schema Indexes 1 Validation

Validation Action ⓘ Error Validation Level ⓘ Strict

```
1 {
2   $jsonSchema: {
3     bsonType: 'object',
4     required: [
5       'date_saisie',
6       'nom_eleve',
7       'prenom_eleve',
8       'nom_classe',
9       'nom_matiere',
10      'nom_prof',
11      'prenom_prof',
12      'nom_trimestre',
13      'note'
14    ],
15    properties: {
16      date_saisie: {
17        bsonType: 'date',
18        description: 'La date de saisie est obligatoire.'
19      },
20      nom_eleve: {
21        bsonType: 'string',
22        description: 'Le nom de l\'élève est obligatoire et doit être une chaîne de caractères.'
23      },
24      prenom_eleve: {
25        bsonType: 'string',
26        description: 'Le prénom de l\'élève est obligatoire et doit être une chaîne de caractères.'
27      },
```

```
28      nom_classe: {
29        bsonType: 'string',
30        enum: [
31          'cp',
32          'ce1',
33          'ce2',
34          'cm1',
35          'cm2'
36        ],
37        description: 'Le nom de la classe est obligatoire et doit être cp, ce1, ce2, cm1 ou cm2.'
38      },
39      nom_matiere: {
40        bsonType: 'string',
41        description: 'Le nom de la matière est obligatoire et doit être une chaîne de caractères.'
42      },
43      nom_prof: {
44        bsonType: 'string',
45        description: 'Le nom du (de la) professeur est obligatoire.'
46      },
47      prenom_prof: {
48        bsonType: 'string',
49        description: 'Le prénom du (de la) professeur est obligatoire.'
50      },
51      nom_trimestre: {
52        bsonType: 'string',
53        enum: [
54          'trim01',
55          'trim02',
56          'trim03',
57          'trim04'
58        ],
59        description: 'Le nom du trimestre est obligatoire et doit être trim01, 02, 03 ou 04.'
60      },
```

```
61      note: {
62        bsonType: 'int',
63        minimum: 0,
64        maximum: 20,
65        description: 'La note est obligatoire et doit être comprise entre 0 et 20.'
66      },
67      avis: {
68        bsonType: 'string',
69        description: 'L\'avis est optionnel et doit être une chaîne de caractères.'
70      },
71      avancement: {
72        bsonType: 'int',
73        description: 'L\'avancement est optionnel et doit être un entier.'
74      }
75    }
76  }
77 }
```

Les données - NoSQL

➤ Validation du bon fonctionnement des schémas

```
>_MONGOSH
```

```
> db.classe.insertOne({"nom_classe": "AB", "nom_prof": 123, "prenom_prof":456})
```

```
✖ ▶ MongoServerError: Document failed validation
```

```
9  ▼ properties: {
10 ▼   nom_classe: {
11     bsonType: 'string',
12 ▼     'enum': [
13       'cp',
14       'ce1',
15       'ce2',
16       'cm1',
17       'cm2'
18     ],
19     description: 'Le nom de la classe est obligatoire et doit être cp, ce1, ce2, cm1 ou cm2.'
20   },
21 ▼   nom_prof: {
22     bsonType: 'string',
23     description: 'Le nom du (de la professeur) est obligatoire et doit être une chaîne de caractères.'
24   },
25 ▼   prenom_professeur: {
26     bsonType: 'string',
27     description: 'Le prénom du (de la) professeur est obligatoire et doit être une chaîne de caractères.'
```

```
reason: 'value was not found in enum',
consideredValue: 'AB'
```

```
reason: 'type did not match',
consideredValue: 123,
consideredType: 'int'
```

```
reason: 'type did not match',
consideredValue: 456,
consideredType: 'int'
```



L'API PYTHON

FastAPI <small>0.1.0</small> <small>OAS 3.1</small>		
<small>/openapi.json</small>		
Note		
GET	/note	Obtenir Note
PATCH	/note	Mise A Jour Note
DELETE	/note	Suppression Note
POST	/nouvelle_note	Ajout Note
POST	/nouvelle_note2	Ajout Note2
PATCH	/note2	Mise A Jour Note2
Professeurs		
GET	/professeurs/	Obtenir Liste Profs
GET	/notes/	Obtenir Notes Par Prof
Elevés		
GET	/all_elevés/	Obtenir Tous Elevés
GET	/elevés_par_classe/	Obtenir Elevés Par Classe
GET	/notes_par_elevé/	Obtenir Notes Elevé
GET	/elevés	Obtenir Elevé
POST	/elevés	Ajout Elevé
DELETE	/elevés	Suppression Elevé
PATCH	/adresse_elevés	Mise A Jour Adresse

L'API Python - Application

Application overview

DEMANDES DU PROJET / FONCTIONS ADDITIONNELLES

*FastAPI: Développement, design,
documentation et utilisation d'APIs REST*

(R) Récupérer une note d'un élève

(U) Mettre à jour une note d'un élève

(D) Supprimer une note d'un élève

(C) Ajouter une note à un élève

(C) Ajouter une note à un élève (2nd version)

(U) Mettre à jour une note d'un élève (2nd version)

Récupérer la liste des professeurs

Récupérer les élèves et leurs notes selon un professeur

Récupérer la liste des élèves (organisée par classe)

Récupérer la liste des élèves selon le choix d'une classe

Récupérer les notes d'un élève

(R) Récupérer un élève

(C) Ajouter une note à un élève

(D) Supprimer un élève

(U) Mettre à jour l'adresse d'un élève

2024-07-26

L'API Python - POST

Version 1- UI friendly

POST /nouvelle_note Ajout Note

Parameters

Name	Description
date_saisie ★ required (query)	<input type="text" value="date_saisie"/>
nom_eleve ★ required (query)	<input type="text" value="nom_eleve"/>
prenom_eleve ★ required (query)	<input type="text" value="prenom_eleve"/>
nom_classe ★ required (query)	<input type="text" value="nom_classe"/>
nom_matiere ★ required (query)	<input type="text" value="nom_matiere"/>
nom_prof ★ required (query)	<input type="text" value="nom_prof"/>
prenom_prof ★ required (query)	<input type="text" value="prenom_prof"/>
nom_trimestre ★ required (query)	<input type="text" value="nom_trimestre"/>
note ★ required (query)	<input type="text" value="note"/>
avis ★ required (query)	<input type="text" value="avis"/>
avancement ★ required (query)	<input type="text" value="avancement"/>

Version 2- Front dev friendly

POST /nouvelle_note2 Ajout Note2

Parameters

No parameters

Request body required

```
{
  "date_saisie": "2024-07-26",
  "nom_eleve": "string",
  "prenom_eleve": "string",
  "nom_classe": "string",
  "nom_matiere": "string",
  "nom_prof": "string",
  "prenom_prof": "string",
  "nom_trimestre": "string",
  "note": 0,
  "avis": "string",
  "avancement": 0
}
```

L'API Python - POST

Version 1- UI friendly

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/nouvelle_note?date_saisie=2024-07-26&nom_eleve=' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
http://127.0.0.1:8000/nouvelle_note?
date_saisie=2024-07-26&nom_eleve=Eude&prenom_eleve=Laurent&nom_classe=
```

Server response

Code	Details
200	<p>Response body</p> <pre>"documents insérés"</pre> <p>Response headers</p> <pre>content-length: 21 content-type: application/json date: Fri, 26 Jul 2024 08:03:44 GMT server: uvicorn</pre>

Responses

Code	Description
200	<p>Successful Response</p> <p>Media type</p> <p><input type="text" value="application/json"/></p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>"string"</pre>
422	<p>Validation Error</p> <p>Media type</p>

Version 2- Front dev friendly

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/nouvelle_note2' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "date_saisie": "2024-07-26",
    "nom_eleve": "Tartour",
    "prenom_eleve": "Kevin",
    "nom_classe": "CP",
    "nom_matiere": "Python",
    "nom_prof": "Hotton",
    "prenom_prof": "Robin",
    "nom_trimestre": "TRIM04",
    "note": 10,
    "avis": "RAS",
    "avancement": 0
  }'
```

Request URL

```
http://127.0.0.1:8000/nouvelle_note2
```

Server response

Code	Details
200	<p>Response body</p> <pre>"documents insérés"</pre> <p>Response headers</p> <pre>content-length: 21 content-type: application/json date: Fri, 26 Jul 2024 08:01:19 GMT server: uvicorn</pre>

Responses

L'API Python - POST

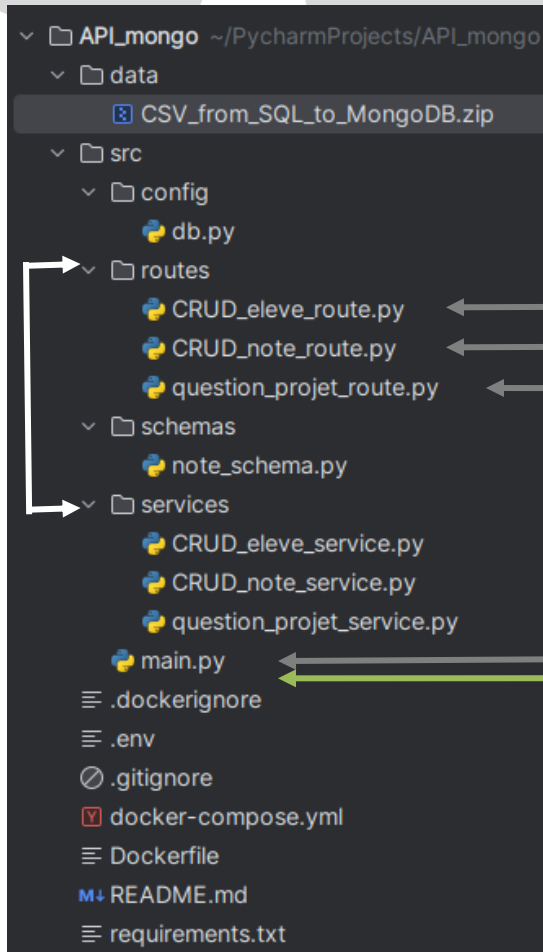
Version 1- UI friendly

```
_id: ObjectId('66a35860515f4b572fd22c')
date_saisie : 2024-07-26T00:00:00.000+00:00
nom_eleve : "eude"
prenom_eleve : "laurent"
nom_classe : "cp"
nom_matiere : "python"
nom_prof : "germain"
prenom_prof : "christophe"
nom_trimestre : "trim04"
note : 20
avis : "ras"
avancement : 0
```

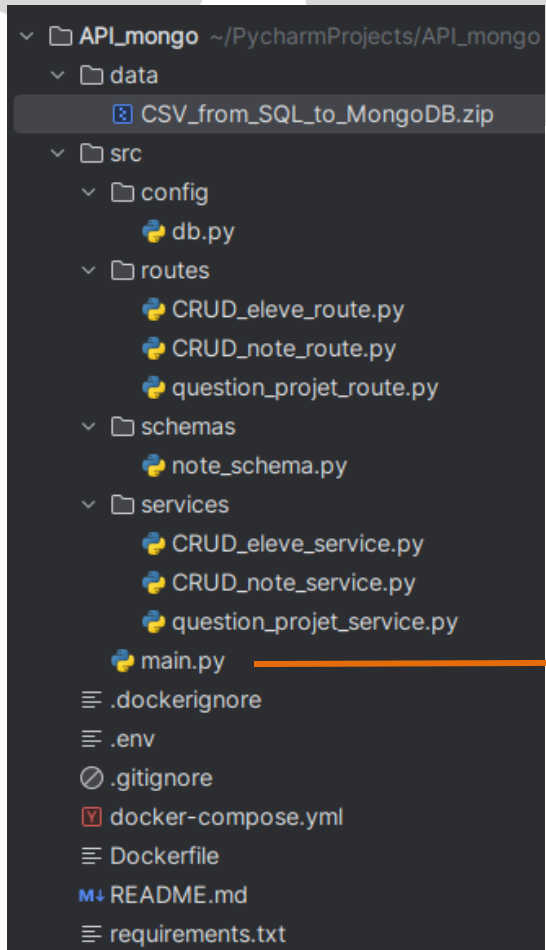
Version 2- Front dev friendly

```
_id: ObjectId('66a357cf515f4b572fd22b')
date_saisie : 2024-07-26T00:00:00.000+00:00
nom_eleve : "tartour"
prenom_eleve : "kevin"
nom_classe : "cp"
nom_matiere : "python"
nom_prof : "hotton"
prenom_prof : "robin"
nom_trimestre : "trim04"
note : 10
avis : "ras"
avancement : 0
```

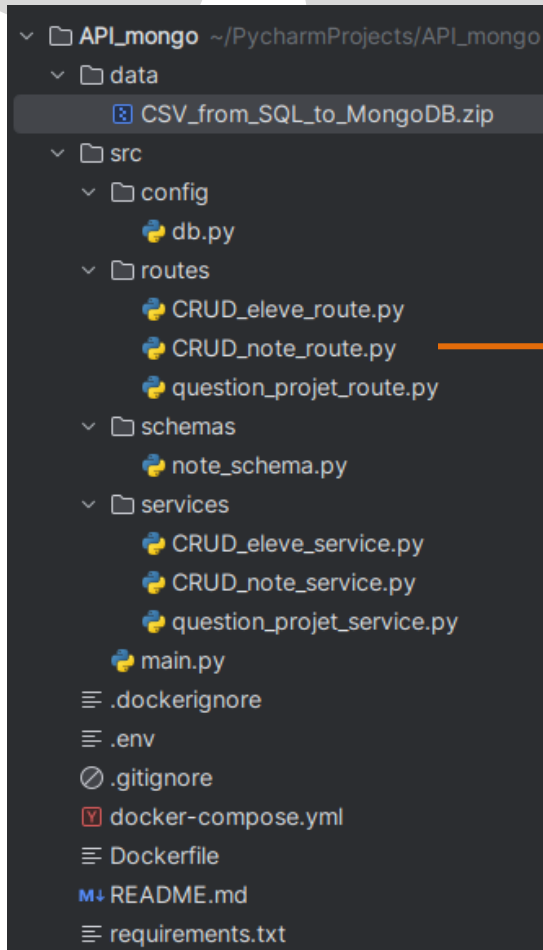
L'API Python - Application



L'API Python - Route



```
load_dotenv()          #pour utiliser la fonction et
app = FastAPI()
app.include_router(CRUD_note_route.router)
app.include_router(question_projet_route.router)
app.include_router(CRUD_eleve_route.router)
```



```
router=APIRouter(tags=["Note"]) #defini le routeur pour cette page, prédéfini le tag pour toutes les for

@router.get("/note")
async def obtenir_note(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere):
    return get_notes(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere)

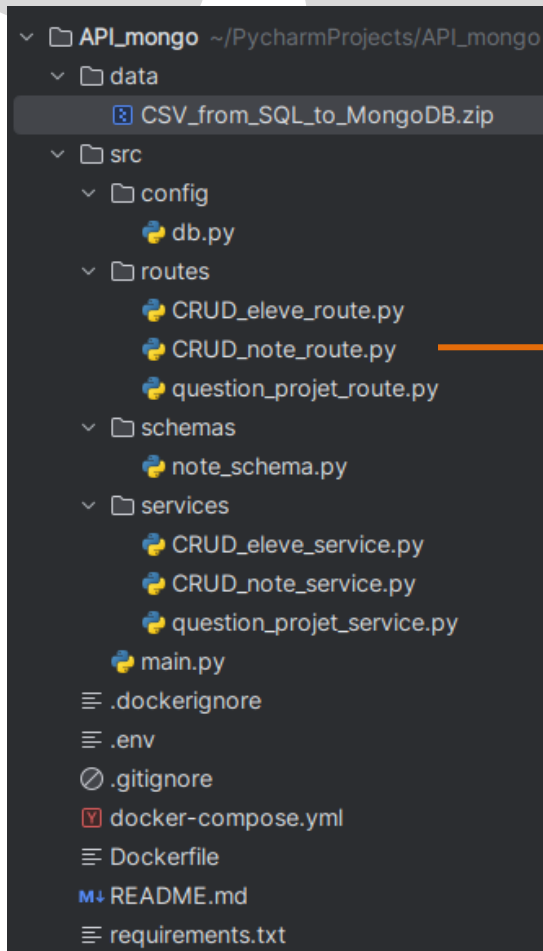
@router.post("/nouvelle_note")
async def ajout_note(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere, nom_prof, prenom_prof, nom_tr)
    return fill_notes(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere, nom_prof, prenom_prof, nom_tr)

@router.post("/nouvelle_note2")
async def ajout_note2(item :NoteCreateSchema):
    return fill_notes2(item)

@router.patch("/note")
async def mise_a_jour_note(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere, new_note):
    return modify_notes(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere, new_note)

@router.patch("/note2")
async def mise_a_jour_note2(find: NoteGetSchema, item: NoteCreateSchema):
    return modify_notes2(find, item)

@router.delete("/note")
async def suppression_note(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere):
    return delete_notes(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere)
```



```
router=APIRouter(tags=["Note"]) #defini le routeur pour cette page, prédéfini le tag pour toutes les for

@router.get("/note")
async def obtenir_note(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere):
    return get_notes(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere)

@router.post("/nouvelle_note")
async def ajout_note(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere):
    return fill_notes(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere)

@router.post("/nouvelle_note2")
async def ajout_note2(item :NoteCreateSchema):
    return fill_notes2(item)

@router.patch("/note")
async def mise_a_jour_note(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere):
    return modify_notes(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere)

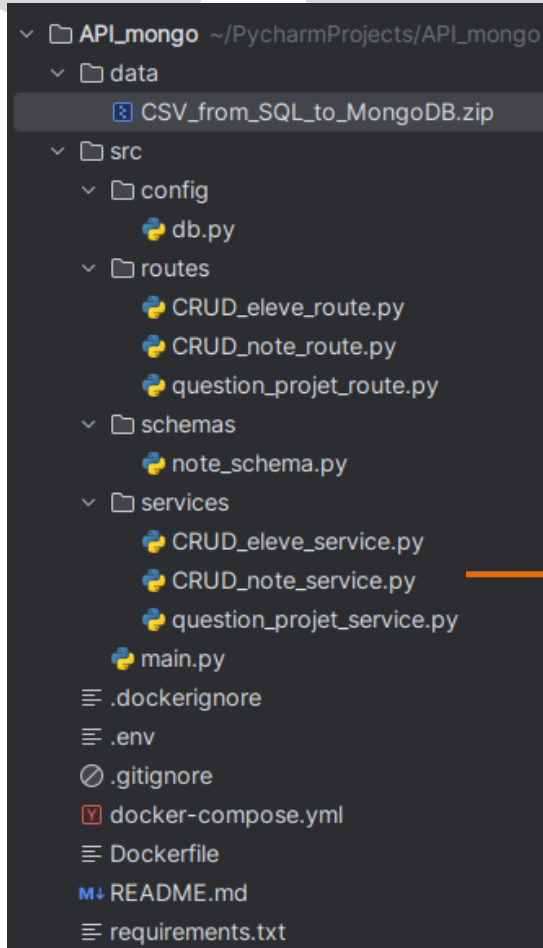
@router.patch("/note2")
async def mise_a_jour_note2(find: NoteGetSchema, item: NoteCreateSchema):
    return modify_notes2(find, item)

@router.delete("/note")
async def suppression_note(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere):
    return delete_notes(date_saisie, nom_eleve, prenom_eleve, nom_classe, nom_matiere)
```

3 usages

```
class NoteCreateSchema(BaseModel):
    date_saisie : date
    nom_eleve: str
    prenom_eleve: str
    nom_classe: Optional[str] = None
    nom_matiere: Optional[str] = None
    nom_prof: Optional[str] = None
    prenom_prof: Optional[str] = None
    nom_trimestre: Optional[str] = None
    note: int
    avis: Optional[str] = None
    avancement: Optional[int] = None
```

L'API Python - functions



```
2 usages
def fill_notes2(item):
    item= dict(item)
    item["date_saisie"] = datetime.strptime(str(item["date_saisie"]), _format: "%Y-%m-%d")
    item["nom_eleve"] = item["nom_eleve"].lower()
    item["prenom_eleve"] = item["prenom_eleve"].lower()
    item["nom_classe"] = item["nom_classe"].lower()
    item["nom_matiere"] = item["nom_matiere"].lower()
    item["nom_prof"] = item["nom_prof"].lower()
    item["prenom_prof"] = item["prenom_prof"].lower()
    item["note"] = int(item["note"])
    item["nom_trimestre"] = item["nom_trimestre"].lower()
    item["avis"] = item["avis"].lower()
    item["avancement"] = int(item["avancement"])

    try:
        mongo_collection.insert_one(dict(item))        #Insert le dictionnaire défini dans document
        return "documents insérés"
    except Exception as e:
        print(f"document non inséré du à: {e}")
```




LES PERSPECTIVES

Les perspectives

- Gérer les erreurs HTTP
- Mettre en place un processus de tests unitaires
- Améliorer la robustesse (inputs)
- Utiliser une base de données plus importante
- Associer une partie front-end
- Proposer des outils de traitements de données (calcul de moyenne, graphiques de répartition...)







ANNEXES

Annexe 1

Requêtes SQL (extraction des données de digischools.sql)

t_classe

```
SELECT lower(t_classe.nom) as 'nom_classe', lower(t_prof.nom) as 'nom_prof', lower(t_prof.prenom) as 'prenom_prof' FROM `t_classe`  
JOIN t_prof on t_classe.prof = t_prof.id;
```

t_prof

```
SELECT lower(nom) as 'nom_prof', lower(prenom) as 'prenom_prof', substr(date_naissance,1,10) as 'date_naissance_prof', lower(adresse)  
as 'adresse_prof', lower(sexe) as 'sexe_prof' FROM t_prof;
```

t_eleve

```
select lower(t_eleve.nom) as 'nom_eleve', lower(prenom) as 'prenom_eleve', lower(t_classe.nom) as 'nom_classe', substr(date_naissance,  
e,1,10) as 'date_naissance_eleve', lower(adresse) as 'adresse_eleve', lower(sexe) as 'sexe_eleve' from t_eleve JOIN t_classe on t_eleve  
.classe=t_classe.id;
```

t_matiere

```
select LOWER(nom) as 'nom_matiere' from t_matiere;
```

t_trimestre

```
SELECT LOWER(nom) as 'nom_trimestre', SUBSTR(date,1,10) as 'date_trimestre' from t_trimestre
```

t_notes

```
SELECT substr(date_saisie,1,10) as 'date_saisie', lower(t_eleve.nom) as 'nom_eleve', lower(t_eleve.prenom) as 'prenom_eleve', lower(t_cl  
asse.nom) as 'nom_classe', lower(t_matiere.nom) as 'nom_matiere', lower(t_prof.nom) as 'nom_prof', lower(t_prof.prenom) as 'prenom_pr  
of', lower(t_trimestre.nom) as 'nom_trimestre', note, lower(avis) as 'avis', avancement FROM t_notes JOIN t_eleve on t_notes.ideleve = t_  
eleve.id JOIN t_classe on t_notes.idclasse = t_classe.id JOIN t_matiere on t_notes.idmatiere = t_matiere.idmatiere JOIN t_prof on t_note  
s.idprof = t_prof.id JOIN t_trimestre on t_notes.idtrimestre = t_trimestre.idtrimestre;
```

Annexe 2 - classe

cluster0.3qz2vzx.mongodb.net > digischools > classe

Documents 5

Aggregations

Schema

Indexes 1

Validation

Validation Action ⓘ

Error

Validation Level ⓘ

Strict

```
1 {
2   $jsonSchema: {
3     bsonType: 'object',
4     required: [
5       'nom_classe',
6       'nom_prof',
7       'prenom_professeur'
8     ],
9     properties: {
10      nom_classe: {
11        bsonType: 'string',
12        enum: [
13          'cp',
14          'ce1',
15          'ce2',
16          'cm1',
17          'cm2'
18        ],
19        description: 'Le nom de la classe est obligatoire et doit être cp, ce1, ce2, cm1 ou cm2.'
20      },
21      nom_prof: {
22        bsonType: 'string',
23        description: 'Le nom du (de la professeur) est obligatoire et doit être une chaîne de caractères.'
24      },
25      prenom_professeur: {
26        bsonType: 'string',
27        description: 'Le prénom du (de la) professeur est obligatoire et doit être une chaîne de caractères.'
28      }
29    }
30  }
31 }
```

Annexe 2 - eleve

cluster0.3qz2zvx.mongodb.net > digischools > eleve

Documents 53 Aggregations Schema Indexes 1 Validation

Validation Action ⓘ

Error ▼

Validation Level ⓘ

Strict ▼

```
1 ▼ {
2   $jsonSchema: {
3     bsonType: 'object',
4     required: [
5       'nom_eleve',
6       'prenom_eleve',
7       'nom_classe',
8       'date_naissance_eleve'
9     ],
10    properties: {
11      nom_eleve: {
12        bsonType: 'string',
13        description: 'Le nom de l\'élève est obligatoire et doit être une chaîne de caractères.'
14      },
15      prenom_eleve: {
16        bsonType: 'string',
17        description: 'Le prénom de l\'élève est obligatoire et doit être une chaîne de caractères.'
18      },
19      date_naissance_eleve: {
20        bsonType: 'date',
21        description: 'La date de naissance de l\'élève est obligatoire et doit être une date.'
22      },
23      adresse_eleve: {
24        bsonType: 'string',
25        description: 'L\'adresse de l\'élève est optionnelle.'
26      },
27    }
```

```
27   sexe: {
28     bsonType: 'string',
29     enum: [
30       'homme',
31       'femme'
32     ],
33     description: 'Le sexe de l\'élève est optionnel est doit être homme ou femme.'
34   }
35 }
36 }
37 }
```


Annexe 2 - matiere

cluster0.3qz2zvx.mongodb.net > digischools > matiere

Documents 5 Aggregations Schema Indexes 1 **Validation**

Validation Action ⓘ

Error ▼

Validation Level ⓘ

Strict ▼

```
1 ▼ {
2 ▼   $jsonSchema: {
3     bsonType: 'object',
4     required: [
5       'nom_matiere'
6     ],
7     properties: {
8       nom_matiere: {
9         bsonType: 'string',
10        description: 'Le nom de la matière est obligatoire et doit être une chaîne de caractères.'
11      }
12    }
13  }
14 }
```

Annexe 2 - notes

cluster0.3qz2zvx.mongodb.net > digischools > notes

Documents 3 Aggregations Schema Indexes 1 Validation

Validation Action ⓘ Error Validation Level ⓘ Strict

```
1 {
2   $jsonSchema: {
3     bsonType: 'object',
4     required: [
5       'date_saisie',
6       'nom_eleve',
7       'prenom_eleve',
8       'nom_classe',
9       'nom_matiere',
10      'nom_prof',
11      'prenom_prof',
12      'nom_trimestre',
13      'note'
14    ],
15    properties: {
16      date_saisie: {
17        bsonType: 'date',
18        description: 'La date de saisie est obligatoire.'
19      },
20      nom_eleve: {
21        bsonType: 'string',
22        description: 'Le nom de l\'élève est obligatoire et doit être une chaîne de caractères.'
23      },
24      prenom_eleve: {
25        bsonType: 'string',
26        description: 'Le prénom de l\'élève est obligatoire et doit être une chaîne de caractères.'
27      },
```

```
28      nom_classe: {
29        bsonType: 'string',
30        enum: [
31          'cp',
32          'ce1',
33          'ce2',
34          'cm1',
35          'cm2'
36        ],
37        description: 'Le nom de la classe est obligatoire et doit être cp, ce1, ce2, cm1 ou cm2.'
38      },
39      nom_matiere: {
40        bsonType: 'string',
41        description: 'Le nom de la matière est obligatoire et doit être une chaîne de caractères.'
42      },
43      nom_prof: {
44        bsonType: 'string',
45        description: 'Le nom du (de la) professeur est obligatoire.'
46      },
47      prenom_prof: {
48        bsonType: 'string',
49        description: 'Le prénom du (de la) professeur est obligatoire.'
50      },
51      nom_trimestre: {
52        bsonType: 'string',
53        enum: [
54          'trim01',
55          'trim02',
56          'trim03',
57          'trim04'
58        ],
59        description: 'Le nom du trimestre est obligatoire et doit être trim01, 02, 03 ou 04.'
60      },
61      note: {
62        bsonType: 'int',
63        minimum: 0,
64        maximum: 20,
65        description: 'La note est obligatoire et doit être comprise entre 0 et 20.'
66      },
67      avis: {
68        bsonType: 'string',
69        description: 'L\'avis est optionnel et doit être une chaîne de caractères.'
70      },
71      avancement: {
72        bsonType: 'int',
73        description: 'L\'avancement est optionnel et doit être un entier.'
74      }
75    }
76  }
77 }
```

Annexe 2 - prof

cluster0.3qz2zvx.mongodb.net > digischools > prof

Documents 3 Aggregations Schema Indexes 1 Validation

Validation Action ⓘ Error Validation Level ⓘ Strict

```
1 {
2   $jsonSchema: {
3     bsonType: 'object',
4     required: [
5       'nom_professeur',
6       'prenom_professeur'
7     ],
8     properties: {
9       nom_prof: {
10        bsonType: 'string',
11        description: 'Le nom du (de la professeur) est obligatoire et doit être une chaîne de caractères.'
12      },
13       prenom_professeur: {
14        bsonType: 'string',
15        description: 'Le prénom du (de la) professeur est obligatoire et doit être une chaîne de caractères.'
16      },
17       date_naissance_professeur: {
18        bsonType: 'date',
19        description: 'La date de naissance du (de la) professeur est optionnelle.'
20      },
21       adresse_professeur: {
22        bsonType: 'string',
23        description: 'L\'adresse du (de la) professeur est optionnelle.'
24      },
25       sexe: {
26        bsonType: 'string',
27        'enum': [
28          'homme',
29          'femme'
30        ],
31        description: 'Le sexe du (de la) professeur est optionnel est doit être homme ou femme.'
32      }
33    }
34  }
35 }
```

Annexe 2 - trimestre

cluster0.3qz2zvx.mongodb.net > digischools > trimestre

Documents 3

Aggregations

Schema

Indexes 1

Validation

Validation Action ⓘ

Error

Validation Level ⓘ

Strict

```
1 ▼ {
2 ▼   $jsonSchema: {
3     bsonType: 'object',
4     required: [
5       'nom_trimestre'
6     ],
7     properties: {
8       nom_trimestre: {
9         bsonType: 'string',
10        'enum': [
11          'trim01',
12          'trim02',
13          'trim03',
14          'trim04'
15        ],
16        description: 'Le nom du trimestre est obligatoire et doit être trim01, 02, 03 ou 04.'
17      },
18     date_saisie: {
19       bsonType: 'date',
20       description: 'La date du trimestre est optionnelle.'
21     }
22   }
23 }
24 }
```

Récupérer la liste des professeurs

- La fonction `ListProf()` est définie pour récupérer la liste des professeurs. Elle utilise la méthode `find()` de la collection `mongo_collection_prof` pour récupérer tous les documents de la collection. Le premier paramètre de `find()` est un filtre vide `{}`, ce qui signifie qu'il récupérera tous les documents sans aucun filtre spécifique.
- Le deuxième paramètre est un dictionnaire de projection `{"_id": 0, "prenom_prof": 1, "nom_prof": 1}`, qui spécifie les champs que nous voulons inclure (1) ou exclure (0) dans les résultats.
- Ensuite, une liste vide est initialisée. Une boucle est utilisée pour parcourir chaque document de la liste des professeurs récupérés
- Si une exception se produit lors de la récupération des professeurs, elle est capturée et affichée à l'aide de l'instruction `except Exception as e: print(e)`.

- - Récupérer la liste des élèves par classe
- - Récupérer les notes d'un élève
- - Récupérer les élèves et leur note selon un professeur

```
• def list_prof():  
    try:  
        liste_de_prof =  
list(mongo_collection_prof.find({}, {"_id":  
0, "prenom_prof":1, "nom_prof": 1}))  
        liste_de_prof = [d["prenom_prof"]+  
"+d['nom_prof'] for d in liste_de_prof]  
        return liste_de_prof  
    except Exception as e:  
        print(e)  
  
• Route : @router.get("/professeurs/",  
tags=["Professeurs"])  
async def obtenir_liste_profs():  
    # renvoyer nos données  
    return {"L'ensemble des professeurs  
est":list_prof()}
```

Récupérer les élèves et leur note selon un professeur

- À l'intérieur du bloc "try", le code utilise la fonction "find" pour rechercher des notes dans une collection MongoDB.
- La recherche est effectuée avec une condition qui filtre les résultats en fonction du nom et du prénom du professeur fournis en paramètres.
- Les champs à récupérer sont spécifiés dans un dictionnaire en utilisant la syntaxe de requête de MongoDB.
- La liste des résultats est ensuite convertie en une liste de chaînes de caractères formatées contenant le prénom de l'élève, le nom de l'élève, le nom de la matière et la note.
- Enfin, la fonction retourne la liste des notes.

```
def note_choix_prof(nom,prenom):  
    try:  
        liste_notes =  
list(mongo_collection_note.find({"nom_prof": nom.lower(),  
"prenom_prof": prenom.lower()}, {  
        "_id": 0, "prenom_eleve": 1, "nom_eleve": 1,  
"nom_matiere": 1, "note": 1}))  
        liste_notes = [d['prenom_eleve'] + " " + d["nom_eleve"] +  
" " + d["nom_matiere"] + " " + str(d["note"]) for d in  
        liste_notes]  
        return liste_notes  
    except Exception as e:  
        print(e)
```