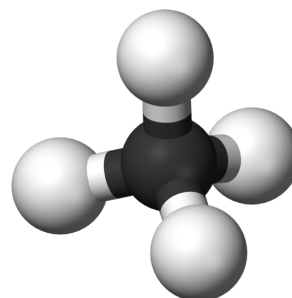


UNIVERSITÉ D'ANGERS



FACULTÉ
DES SCIENCES
*Unité de formation
et de recherche*
DÉPARTEMENT
INFORMATIQUE



FACULTÉ DES SCIENCES

DÉPARTEMENT INFORMATIQUE

ANNÉE 2020/2021

QuChemPedIA

Chefs de projet :
BOUDET Alexandre
PÉRICHET Thomas
RAFIKI Younes

Professeurs référents :
DA MOTA BENOÎT
GENEST DAVID

Client :
CAUCHY THOMAS

07 Février 2021

Nous tenons à remercier toutes les personnes qui ont contribué au projet.

Tout d'abord, nous adressons nos remerciements à nos professeurs référents, **M. Da Mota** et **M. Genest** qui nous ont beaucoup aidés pour la compréhension et l'analyse du sujet mais aussi pour la partie technique du projet, grâce à leur expérience.

Ensuite, nous voulons remercier notre client, **M. Cauchy** qui a su se rendre disponible et répondre à nos questions.

Un grand merci à **Mme Coudray** pour l'enseignement qu'elle nous a apporté sur cette matière.

Nous tenons également à remercier les Master 1 Informatique qui ont travaillé avec nous. Ils ont su répondre aux tâches que nous leur donnions.

Table des matières

1	Introduction	4
1.1	Contexte	4
1.2	Exigence des clients	5
2	Choix des technologies	6
2.1	Framework	6
2.2	Outils de test	6
2.2.1	Pytest	6
2.3	Gestionnaire de version	7
2.4	Intégration Continue	7
3	Management de projet	8
3.1	Organisation	8
3.2	Difficultés rencontrées	10
4	Architecture du projet	11
4.1	Web	12
4.2	API de Consultation	16
4.3	API d'Administration	19
4.4	ElasticSearch	20
5	Déploiement	21
5.1	Machine virtuelle	21
5.2	Ansible	21
6	Documentation	23
6.1	Documentation du code	23
6.2	Documentation du projet	24
7	Conclusion	25

1 Introduction

1.1 Contexte

Le projet QuChemPedIA est un projet de grande envergure regroupant plusieurs acteurs de différents axes qui, durant ces deux dernières années, ont eu pour but de répondre à une demande bien précise qui était de proposer une alternative plus performante, polyvalente et collaborative aux rares solutions existantes de consultation de calcul sur des molécules scientifiques.

L'objectif étant de répondre à des problématiques complexes de calcul et apporter des fonctionnalités intuitives pour faciliter la navigation aux utilisateurs non informaticiens, tout en préservant la qualité des informations contenues sur le site.

En effet, les plateformes de consultations en ligne de molécules existes, néanmoins, aucune ne correspondait entièrement aux besoins des chimistes, d'où l'idée de QuChemPedIA, une plateforme collaborative, permettant aux chimistes de contribuer, de rechercher et de comparer leurs travaux sur les molécules quantiques avec d'autres sur le site.

L'interface web présente plusieurs fonctionnalités qui sont les suivantes :

- Importation/Suppression de nouvelles molécules dans la base de données
- Recherche de molécules dans la base de données
- Calcul de molécules
- Visualisation détaillée d'une molécule

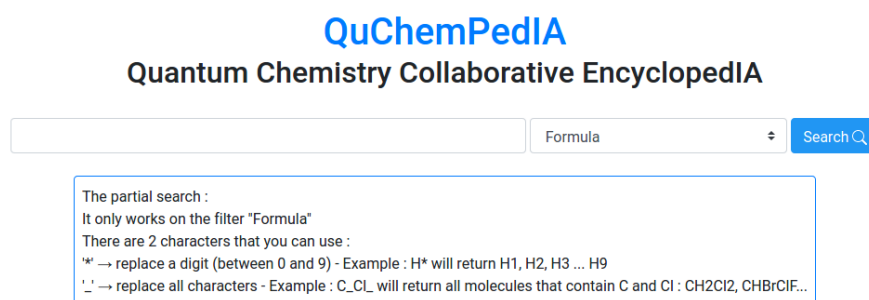


FIGURE 1 – La page d'accueil du site QuChemPedIA

L'année dernière, nous avons eu ce projet à réaliser en collaboration avec les Master 2 qui étaient nos chefs de projet. Mais durant le développement, il y a eu des problèmes comme :

- Un problème de complexité au niveau du *front-end* du site ;
- Une architecture de projet non conventionnelle qui porte préjudice à la maintenabilité du site web ;
- Des soucis de configuration d'environnement sur la machine virtuelle de test ;
- Un manque de communication avec les chefs de projet ;
- Un manque de compétences des chefs de projet sur les technologies utilisées ;

La réalisation de ce projet avait commencé il y a quelques années et chaque contributeur de QuChemPedIA ajoutaient des nouvelles technologies sans que le projet d'origine ne soit terminé. Cela a mené à des cohabitations de plusieurs frameworks qui ont rendues le projet très compliqué à reprendre pour de futurs étudiants.

1.2 Exigence des clients

Le client, Thomas Cauchy, ainsi que Benoit Da Mota ont opté pour une refonte intégrale du projet et des technologies ainsi qu'une baisse des fonctionnalités (Plus de création/-gestion d'utilisateurs, des changements mineurs).

Notre objectif cette année sera donc d'avoir **un site testé et fonctionnel. Permettant ainsi aux futurs étudiants de pouvoir continuer à développer ce projet sans difficultés**, tout en utilisant des technologies faciles à appréhender et pérennes. Il faut également que le projet soit déployé et déployable simplement.

Un premier cahier des charges a donc été réalisé :

- La refonte du *front-end* : Garder la même charte graphique que l'ancien site avec comme contrainte le choix d'une technologie plus intuitive ;
- Créer un système de "recherche partielle" : Lorsqu'un utilisateur saisit une recherche ;
- Réaliser une API REST (pour *Application Programming Interface Representational State Transfer*) pour la consultation des données : Cette API doit être accessible par tout le monde (et donc publique) ;
- Réaliser une API REST afin d'insérer ou supprimer des molécules dans la base de données : Cette API doit être accessible uniquement par certaines personnes (et donc privée). Cette fonction n'est pas obligatoire mais nous avons décidé de la développer malgré tout ;
- Mettre en place l'intégration continue : Il faut utiliser un outil fiable ;
- Et enfin documenter : Pour guider les personnes qui vont reprendre le projet QuChemPedIA ;

2 Choix des technologies

2.1 Framework

Le choix des technologies s'avère cruciale pour le projet. En effet, cela va déterminer l'aboutissement du projet. Tout d'abord, nous devons choisir un langage identique pour chaque grande partie du projet afin d'assurer une cohérence globale. Les années précédentes, le projet était réalisé en *Python*. Nous avons décidé de ré-utiliser ce langage car il est simple et pratique pour concevoir des API dynamique (au même titre que *Express JS*).

Pour l'API de consultation ainsi que l'API d'insertion et suppression (ou administration), nous avons choisi d'utiliser *Flask*. C'est un framework *Python* très simple à prendre en main et beaucoup plus minimaliste que son grand frère *Django*. Au départ, nous avons choisi d'utiliser *Django* en accord avec le projet de l'année dernière. Mais nous nous sommes rendus compte qu'il était beaucoup trop lourd et beaucoup trop gourmand pour réaliser des API pour ce projet.

Nous avons hésité à utiliser *Vue.js*, un framework *front-end* permettant de construire une interface utilisateur. Mais *Vue.js* est assez long à prendre en main et complexe. Nous avons donc décidé d'utiliser uniquement de l'*HTML*, du *JavaScript* ainsi que des librairies *CSS* comme *Bootstrap* etc.

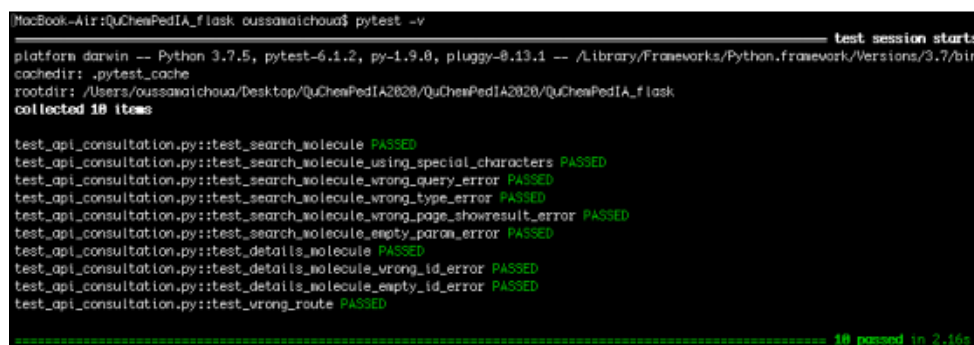
Pour la base de données, nous sommes restés sur *ElasticSearch*, un logiciel permettant l'indexation et la recherche de données s'effectuant grâce à une API REST dans un format JSON. Utiliser une base de données relationnelle n'est pas une bonne idée au vu du volume important de données que nous aurons à stocker.

2.2 Outils de test

2.2.1 Pytest

Pytest est la librairie standard consacrée aux tests unitaires sur des applications Python. La rédaction des tests est assez intuitive et facile à implémenter.

Sur QuChemPedIA les deux API ont été testées avec Pytest, toutes les méthodes possèdent des tests associés en vérifiant le traitement et le code status retourné par l'API.



```
MacBook-Air:QuChemPedIA_flask oussamaichoua$ pytest -v
platform darwin -- Python 3.7.5, pytest-6.1.2, py-1.9.0, pluggy-0.13.1 -- /Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7
cachedir: .pytest_cache
rootdir: /Users/oussamaichoua/Desktop/QuChemPedIA2020/QuChemPedIA2020/QuChemPedIA_flask
collected 10 items

test_api_consultation.py::test_search_molecule PASSED
test_api_consultation.py::test_search_molecule_using_special_characters PASSED
test_api_consultation.py::test_search_molecule_wrong_query_error PASSED
test_api_consultation.py::test_search_molecule_wrong_type_error PASSED
test_api_consultation.py::test_search_molecule_wrong_page_showresult_error PASSED
test_api_consultation.py::test_search_molecule_empty_param_error PASSED
test_api_consultation.py::test_details_molecule PASSED
test_api_consultation.py::test_details_molecule_wrong_id_error PASSED
test_api_consultation.py::test_details_molecule_empty_id_error PASSED
test_api_consultation.py::test_wrong_route PASSED

===== 10 passed in 2.16s =====
```

FIGURE 2 – Exemple de tests API Consultation

2.3 Gestionnaire de version

Pour versionner QuChemPedIA nous avons utilisé le classique et indétronable git accompagné de son interface graphique GitHub. Nous avons pu rendre flexible le développement du projet en utilisant une architecture spécifique mais simplifiée se rapprochant du célèbre **GitFlow** avec une branche *master* et une branche *dev*.

Ensuite chaque développeur a pu créer une branche depuis la branche *dev*, ce système permet de disposer de branche stable avec un code fonctionnel. Généralement ce paradigme peut être complété avec l'implémentation de l'intégration continue.

2.4 Intégration Continue

Concernant l'intégration continue nous avons réfléchi de manière à ne pas ajouter trop de nouvelles technologies au projet. C'est pourquoi nous nous sommes renseignés pour homogénéiser un maximum certains traitements.

En effet GitHub propose une section dédiée à l'intégration continue ainsi que du déploiement continu. Dans l'idée il faudrait pouvoir tester les deux API et pourquoi pas dans un futur proche intégrer des tests End-to-End pour le front-end. Cela permettrait d'avoir la totalité du projet tester si pour X raison les tests ne passent pas nous serons avertis dès qu'un changement a été push sur le projet.

C'est pour cela que nous avons retenu l'intégration continue par le biais de GitHub, elle sera donc hébergée au même endroit que le projet QuChemPedIA.

Sur GitHub la partie CI se situe dans le repository dans la section *Actions* et une instance de CI est appelée un *Workflows*.

A ce jour l'intégration continue est en place mais le script d'exécution des tests n'est pas effectif. Il reste à le modifier pour qu'à chaque fois qu'une modification a été poussée sur une branche spécifique les tests soient lancés.

Il serait une bonne chose d'utiliser tout le pouvoir qu'offre la CI/CD de GitHub. En imaginant qu'à chaque fois que le projet est mis à jour et que les tests sont au verts on puisse créer un script qui déploie automatiquement en production le projet dans son entièreté. Il suffirait de lancer toutes les recettes Ansible stockées sur les VM et ces dernières s'occuperont de clone la nouvelle version du projet.

3 Management de projet

3.1 Organisation

Afin de mener à bien ce projet, nous étions 3 étudiants de Master 2 actant comme des chefs de projet et 5 étudiants de Master 1 incarnant le rôle de développeur.

Au début du projet, afin de leur expliquer le sujet, nous leur avons fourni un document que nous avons écrit expliquant les fonctionnalités et le but de l'application QuChemPe-dIA.

Quelques formations étaient nécessaires pour démarrer le projet dans les meilleures conditions. Dans ces dernières nous pouvons retrouver des formations communes aux développeurs telle que git mais aussi des formations spécifiques selon les préférences des M1 et ce sur quoi ils allaient travailler.

Voici la répartition des 5 développeurs que nous avons à disposition :

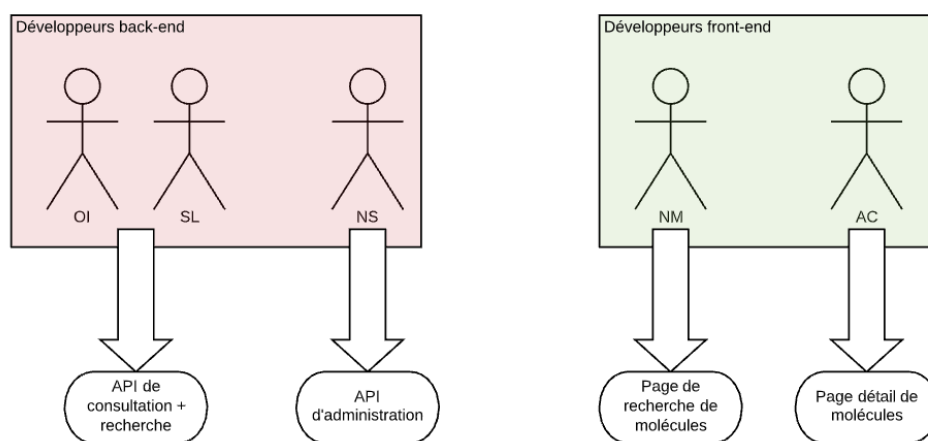


FIGURE 3 – Schéma tâches développeurs

3 développeurs s'occupaient du Back-end pour développer les deux APIs :

- Une API de consultation publique permettant de récupérer les données ;
- Une API d'administration privée permettant de manipuler les données ;

Tandis que les 2 autres développeurs s'occupaient du Front-end réparti en deux parties :

- La recherche de molécule et ses résultats ;
- La page détail des molécules ;

Cette répartition a permis de satisfaire au maximum les préférences de chaque développeur tout en répartissant au mieux les tâches.

Concernant la répartition des tâches des managers de projet, la voici :

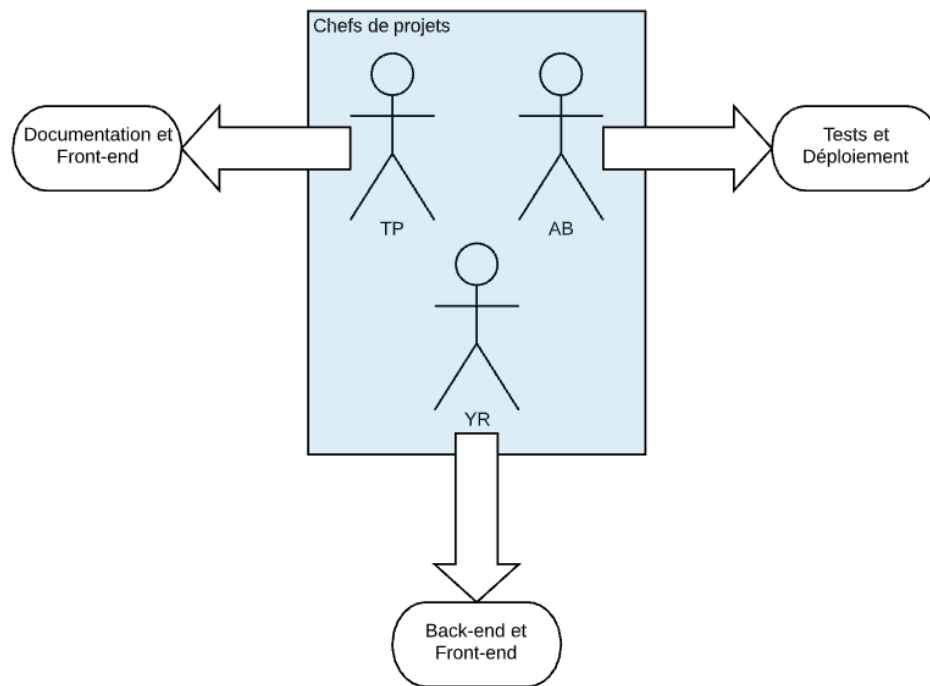


FIGURE 4 – Schéma tâches chefs de projet

Nous nous sommes répartis certaines parties du projet, même si en réalité nous étions plutôt flexible, Thomas s’est occupé principalement de la documentation du code et du Front-End. Younes s’est majoritairement occupé de l’accompagnement des M1 que ce soit sur le Front-End comme sur le Back-End. Et Alexandre a accompagné les M1 pour la mise en place des tests et s’est occupé du déploiement de l’application en publique.

Concernant la communication avec les M1, nous faisons une réunion hebdomadaire en plusieurs parties. Pour chaque développeur nous leur demandons ce qu’ils avaient fait depuis la dernière réunion, ce qu’ils allaient effectuer lors de la séance. Grâce à ces informations, nous pouvions suivre leur avancée et leur donner de nouvelles tâches si nécessaires.

Cependant, avec la situation sanitaire actuelle, la communication fut plus compliquée par rapport aux autres projets. En effet, nous ne pouvions pas effectuer beaucoup de réunion en présentiel, c’est pourquoi nous avons mis en place un serveur *Discord* et nous l’avons configuré afin d’être le plus efficace dans la communication pour pallier la situation sanitaire.

Voici un aperçu de l'organisation des salons Discord :

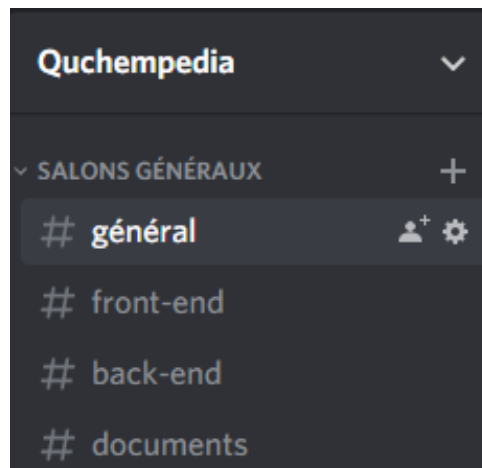


FIGURE 5 – Aperçu des salons Discord

Nous avons un salon pour le front-end et un autre pour le back-end afin d'éviter de mélanger les deux parties, de plus nous avons un salon documents afin de se transférer des documents pour ne pas les perdre.

En complément à Discord, nous avons également un salon Teams afin de communiquer avec nos professeurs référents et notre client, mais aussi pour les réunions avec les M1 et nos encadrants en même temps.

3.2 Difficultés rencontrées

Il faut savoir que l'expérience de manager de projet fut une grande première pour nous, ce qui fut une difficulté majeur puisque nous ne savions pas comment gérer une équipe, nous nous sommes donc principalement basé sur notre expérience en M1 lorsque nous incarnons les développeurs. Cependant le management peut être compliqué pour certaines personnes, c'est pourquoi nous étions trois et nous avons pu par conséquent compenser les faiblesses de chacun.

De plus, par manque d'expérience, la communication avec tous les acteurs du projet n'a pas été parfaite. Nous nous sommes parfois mal exprimés et nous n'informions pas forcément tous les acteurs de l'avancée du projet par manque de rigueur.

La situation sanitaire a, de plus, compliqué la tâche, l'adaptation n'a pas été facile pour tout le monde, surtout pour la communication puisque nous ne pouvions pas voir en présentiel les développeurs ou nos professeurs référents ce qui est un problème dans un projet.

4 Architecture du projet

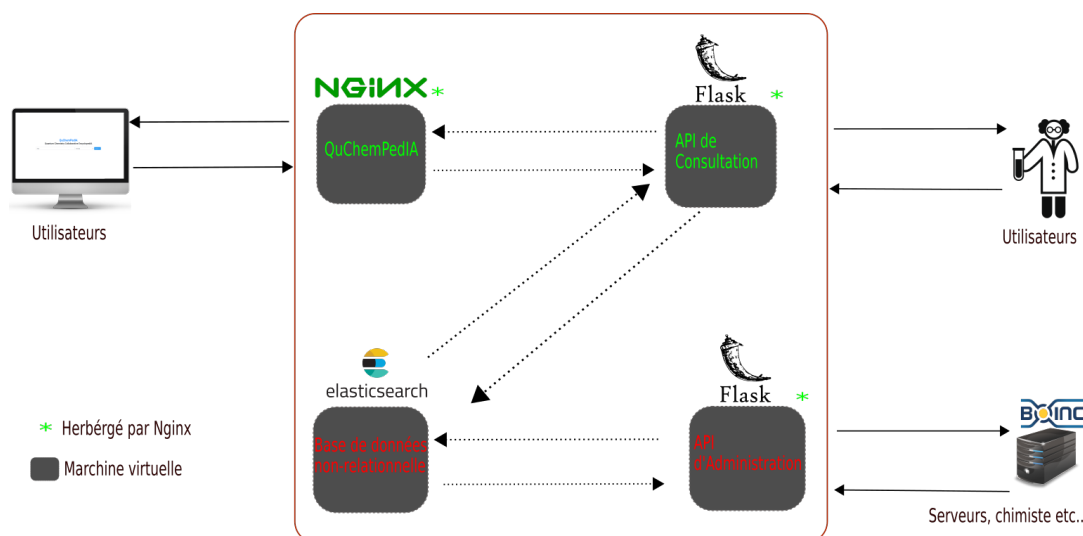


FIGURE 6 – La structure de QuChemPedIA après la refonte

Voici, ci-dessus, la structure globale du projet QuChemPedIA après la refonte. Nous pouvons observer quatre cadres gris dans le grand ensemble rouge. Les cadres gris représentent des machines virtuelles. En effet, nous avons découpé le projet en quatre parties bien distinctes :

- **La partie Front**

C'est la machine virtuelle qui contient toutes les pages HTML de QuChemPedIA. C'est un serveur *Nginx* qui héberge le site web ;

- **La partie Base de données**

C'est la machine virtuelle qui va héberger une base de données de type *Non relationnelle Elasticsearch* ;

- **L'API de Consultation**

C'est l'application qui va faire le lien entre la partie front et la base de données *Elasticsearch* ;

- **L'API d'Administration**

C'est l'application qui va permettre d'ajouter, de consulter ou bien de supprimer des molécules dans la base de données *Elasticsearch* ;

Tout comme la partie Front, l'API de Consultation et d'Administration sont hébergées sur *Nginx*. Pour des raisons de sécurité, nous avons séparé en deux catégories les machines virtuelles :

- La machine qui contient le site web ainsi que l'API de Consultation sont accessibles en publique. En effet, n'importe qui peut accéder à ces deux applications. Un utilisateur pourrait récupérer les résultats d'une recherche via l'API de Consultation sans passer par le site web (afin qu'il puisse utiliser les données récoltées à sa façon) ;
- Afin d'empêcher l'accès à la base de données aux utilisateurs non désirés, nous devons rendre en privée l'API d'Administration ainsi que la base de données *ElasticSearch*.

4.1 Web

Voici l'arborescence de la partie Web :

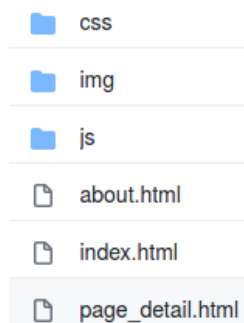


FIGURE 7 – Arborescence du site Web

Nous remarquons qu'on a très peu de contenu. Effectivement, la partie Web n'est guère volumineuse avec seulement trois pages web :

- Une page "*à propos*" qui contient les informations et l'histoire de QuChemPedIA ;
- Une page *index* correspondant à la page d'accueil du site avec le moteur de recherche de molécule ;
- Une page *détail* qui présente les informations détaillées d'une molécule.

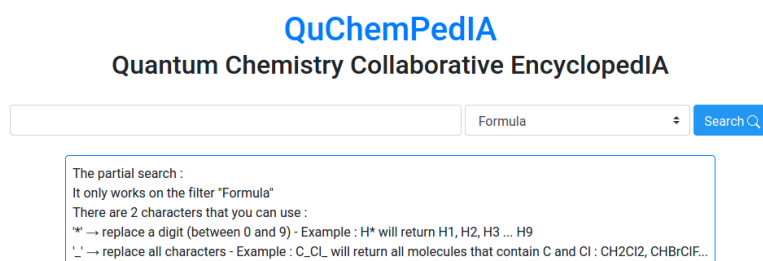


FIGURE 8 – La page d'accueil de QuChemPedIA

Voici la page d'accueil de notre site QuChemPedIA. Nous remarquons que cette interface est divisée en trois zones :

- Une partie composée du titre et du sous-titre ;
- Une partie composée de la barre de recherche ainsi qu'un menu déroulant permettant de choisir le type de recherche (formula, inchi etc.) ;
- Une partie composée d'une zone de texte décrivant comment utiliser la recherche partielle.

Lorsque nous lançons une recherche, nous appelons l'API de Consultation d'afin d'avoir une liste de résultats en Json.

QuChemPedia

CHBrClf

Formula

Search

Found 8 for CHBrClf (formula)

25 entries/page



CHBrClf	Formula: CHBrClf Job Type: FREQ, OPT Charge: 0 Multiplicity: 1 Solvent: Gas	Number heavy atoms: 4 Basis set name: No basis set name List Theory: DFT Ending energy: No ending energy
	Formula: CHBrClf Job Type: OPT, ES Charge: 0 Multiplicity: 1 Solvent: Gas	Number heavy atoms: 4 Basis set name: No basis set name List Theory: DFT Ending energy: No ending energy
	Formula: CHBrClf Job Type: OPT, ES Charge: 0 Multiplicity: 1 Solvent: Gas	Number heavy atoms: 4 Basis set name: No basis set name List Theory: DFT Ending energy: No ending energy
CHBrClf	Formula: CHBrClf Job Type: OPT Charge: 0 Multiplicity: 1 Solvent: Gas	Number heavy atoms: 4 Basis set name: No basis set name List Theory: DFT Ending energy: No ending energy
CHBrClf	Formula: CHBrClf Job Type: TD Charge: 0 Multiplicity: 1 Solvent: Gas	Number heavy atoms: 4 Basis set name: No basis set name List Theory: DFT Ending energy: No ending energy
Formula: CHBrClf		

FIGURE 9 – Résultat de la recherche de *CHBrClf*

Cette page correspond aux résultats de la recherche par l'utilisateur de 'CHBrClf'. Les données récupérées proviennent donc de l'API de Consultation. C'est les fichiers *JavaScript* qui va se charger de convertir le Json obtenu en données lisibles par les utilisateurs.

Nous pouvons :

- Visualiser la liste des molécules trouvées avec quelques caractéristiques (formule, charge, nombre d'atome, solvant etc.).
- Changer le nombre d'affichage des résultats par page ;
- Changer de page etc.

Description
Results

Formule : CHBrClF

Inchi : 1S/CHBrClF/c2-1(3)4/h1H/t1-m/s1

SMILES : [C@@H](Cl)(F)Br

CHBrCl

Computation details		Molecule Details	
Software	Gaussian (09revisionD.01)	Formule	CHBrClF
Computational method	DFT	Charge	0
Functional	PBE1PBE	Spin multiplicity	1
Number of basis set functions	38	Monoisotopic mass	145.89341793199998
Closed shell calculation	True	Original log file	Download
Integration grid	Default		
Solvent	Gas		
Requested SCF convergence on RMS density	1×10 ⁻⁸		
Requested SCF convergence on MAX density	0.000001		
Requested SCF convergence on energy	0.000001		
Temperature	298.15		

FIGURE 10 – La page détail d'une molécule en mode onglet

Voici la page lorsque nous cliquons sur le résultat d'une recherche de molécule. Nous remarquons que la page est disposée en onglet (Description, Results etc.). Chaque onglet affiche les détails caractérisant la molécule.

Molecule

Formule

CHBrCF

Charge

0

Spin multiplicity

1

Monoisotopic mass

145.89341763199998

Inchi :

TS/CHBrCF/c2-1(2)A/h1H(1-)/m1/n1

SMILES :

[C]([H])(Cl)(F)Br

CHBrCl

Computation details

Software

Gaussian (9/revision0.01)

Computational method

DFT

Functional

PBE1PBE

Number of basis set functions

38

Closed shell calculation

True

Integration grid

Default

Solvent

Gas

Requested SCF convergence on RMS density

1x10⁻⁶

Requested SCF convergence on MAX density

0.00001

Requested SCF convergence on energy

0.00001

Temperature

298.15

Original log file

Download

Results

Wavefunction

Total molecular energy

-3137.34999966

HOMO number

34

Calculated energies for the frontier molecular orbitals

Homo-1	Homo	Lumo	Lumo+1
-1516.02	-1515.88	-1515.88	-668.32

Most intense Mulliken atomic charges

Mean = 0.000, Standard Deviation(xsd) = 0.097

Atom	Number	Mulliken partial charges
Cl	2	-0.163
H	1	0.129

Geometry

Nuclear repulsion energy in atomic units

313.86459

Geometry optimization convergence criteria

This calculation is the result of a geometry optimization process.

	Value	Threshold
Maximum Force	0.000371	0.000450
RMS Force	0.000111	0.000300
Maximum Displacement	0.001186	0.001800
RMS Displacement	0.000430	0.001200

Cartesian atomic coordinates

Download

Coordinates are shown in Angstroms.

Atom	X	Y	Z
C	-0.5322	0.5063	0.4076
H	-0.6169	0.5944	1.5200
Cl	-1.8866	-0.7902	-0.0651
F	-0.7696	1.7056	-0.2035
Br	1.2251	-0.2025	-0.0294

FIGURE 11 – La page détail d’une molécule en mode impression

Cette page est la même que la précédente sauf que sa présentation est différente. Le premier est un mode d’affichage en onglet, les détails de chaque catégorie sont séparés alors que celui-ci présente l’ensemble des données dans la même page afin qu’on puisse imprimer tout son contenu.

Pour plus de détails le cahier fonctionnel du projet *QuChemPedIA* est disponible.

4.2 API de Consultation

Voici l'arborescence de notre API de Consultation :

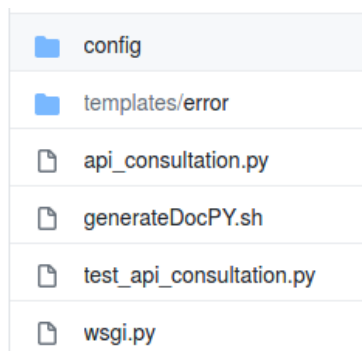


FIGURE 12 – Arborescence de l'API de Consultation

L'API de Consultation va permettre à l'utilisateur de pouvoir rechercher des molécules ou de consulter le détail d'une molécule via des routes. L'API est constitué :

- D'un dossier config qui contient un fichier *app.conf*. Ce fichier va permettre de modifier la localisation de la base de données (*localhost* en développement etc.) ;
- D'un fichier *api_consultation*. C'est ici où sont définis les routes de notre API ainsi que les vues ;
- D'un script permettant de générer automatiquement la documentation de notre API sous format HTML (située à la racine du projet dans le dossier *Documentation/py-Doc*) ;
- D'un fichier qui permet de tester l'API ;
- D'un fichier *wsgi.py* qui permet de déployer notre application sur *Flask* ;

Voici un exemple de route de l'API de Consultation (pour plus de détails se référer à la documentation technique) :

La route `/api/search`

GET <code>/api/search</code> - Retourne une liste de molécules			
Paramètres			
Paramètres	Valeur	Description	Type
q	ex : "h2o", "h*o", "h_"	Recherche de l'utilisateur de molécules	String
type	["formule", "inchi", "smile"]	Type de recherche	String
page	ex : 1,2,3...	Numérotation de la page	Int
showresult	ex : 5, 10, 15...	Nombre de résultat par page	Int
Code retour			
Code retour	Description		
200	La route retourne bien une liste de molécules		
400	Mauvaise requête, les paramètres ne sont pas bons		
404	Aucune molécule n'a été trouvé		
500	Erreur serveur		

Recherche partielle

Une demande du client était d'implémenter une fonctionnalité permettant à l'utilisateur de réaliser des recherches partielles.

- `'*'` : Ce caractère correspond à au moins un chiffre. Par exemple, si l'utilisateur souhaite rechercher toutes les molécules comprenant au moins un atome d'hydrogène, il devra écrire `'H*'`.
- `'_'` : Ce caractère correspond à n'importe quel résultat. Par exemple, si l'utilisateur souhaite rechercher toutes les molécules commençant par CH, il devra saisir `'CH_'` qui retournera donc CHN, CHCO₂, CH₂Cl₂ etc. L'utilisateur peut très bien saisir `'CH_Cl_'` qui retournera l'ensemble des molécules commençant par CH et qui inclus Cl.

Exemple d'utilisation avec CURL

Dans les exemples ci-dessous, nous sommes en local et par conséquent l'adresse du lien sera `127.0.0.1 :5000`.

Nous souhaitons rechercher des molécules par la formule `CHN`, nous voulons simplement afficher la première page tout en précisant que nous retournons que deux résultats par page.

- `q = CHN` ;

- type = formula;
- page = 1;
- showresult = 1;

```
1 curl -X GET "http://127.0.0.1:5000/api/search?type=formula&q=chn&page=1&showresult=2"
```

Résultats :

```
1 {
2   "data": [
3     {
4       "charge": 0,
5       "formula": "CHN",
6       "id": "C0t07HUBOSF22BvMoYMo",
7       "inchi": "InChI=1S/CHN/c1-2/h1H",
8       "job_type": "[\"FREQ\", \"OPT\"]",
9       "list_theory": "[\"DFT\"]",
10      "multiplicity": 1,
11      "nb_heavy_atoms": 2,
12      "smi": "N#C",
13      "solvent": "Gas"
14    },
15    {
16      "charge": 0,
17      "formula": "CHN",
18      "id": "XvFO7HUBkjVcihM6oca9",
19      "inchi": "InChI=1S/CHN/c1-2/h1H",
20      "job_type": "[\"OPT_ES\"]",
21      "list_theory": "[\"DFT\"]",
22      "multiplicity": 1,
23      "nb_heavy_atoms": 2,
24      "smi": "[N][CH]",
25      "solvent": "Gas"
26    }
27  ],
28  "total": 16
29 }
```

Si aucune molécule correspond à la recherche :

```
1 {
2   "Error": "Molecule does not exist"
3 }
```

Si les paramètres ont été mal saisis :

```
1 {  
2   "Error": "Something is missing please check your URL"  
3 }
```

Cette route nous retourne donc une liste de résultats correspondant à la recherche effectuée. Il faut savoir que ce n'est pas l'intégralité du Json de la molécule mais seulement les principaux attributs. Si l'utilisateur souhaite avoir le détail d'une molécule, il lui suffit d'utiliser la route ci-dessous avec l'identifiant correspondant. Nous pouvons aussi avoir accès au nombre total de résultat trouvé via l'attribut *total*.

4.3 API d'Administration

Voici l'arborescence de notre API d'Administration :

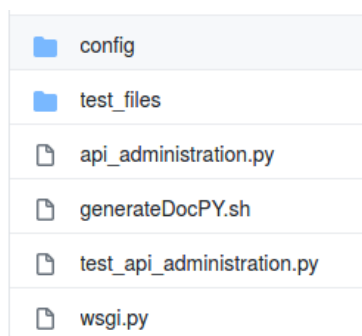


FIGURE 13 – Arborescence de l'API d'Administration

L'API d'administration va permettre à des utilisateurs spécifiques tels que des chimistes ou des professeurs d'administrer la base de données *ElasticSearch* en pouvant :

- Ajouter une molécule et le fichier *.log* correspondant à cette dernière ;
- Supprimer une molécule et le fichier *.log* ;
- Consulter des molécules via son identifiant.

Notre API est constituée des mêmes fichiers que notre API de Consultation avec un dossier *test_files* contenant une molécule à des fins de test.

4.4 Elasticsearch



FIGURE 14 – Logo de Bonsai

Au début de la phase de développement, nous n'avions pas encore monté un serveur Elasticsearch dans une de nos machines virtuelles. Afin d'avoir qu'une seule base de données lors des phases de développements des M1, nous avons décidé d'utiliser une application nommée *Bonsai*. C'est un hébergeur de serveur *ElasticSearch* permettant d'avoir une base de données en ligne disponible et accessible partout dans le monde avec une simple URL.

Avec cette application, nous disposons de 150 Mo ainsi qu'une capacité de stockage de 10 000 documents *Json* mais nous restions limité au niveau du volume étant donné que la base de données accueillera dans un futur proche des millions de molécules.

Nous avons donc créé une base de données *ElasticSearch* au sein de la machine virtuelle correspondante. Nous avons eu beaucoup de difficulté à insérer des molécules dans cette base de données via l'API d'Administration.

En effet, à cause d'une erreur de *Mapping*, nous ne pouvons insérer que quelques types de molécule. Nous sommes donc, pour l'instant, restés sur la base de données *ElasticSearch* de *Bonsai*.

5 Déploiement

Le déploiement des trois applications (API Administration, API Consultation, Web) doit s'effectuer sur des machines virtuelles fournit par M. Da Mota.

Nous avons décidé d'utiliser Ansible pour réaliser un déploiement sous forme de recettes automatisées. Une fois la recette réalisée il suffit de lancer Ansible pour créer l'environnement de production.

5.1 Machine virtuelle

Les machines virtuelles sont hébergées sur une réseau local et peuvent donc communiqué librement entre elles.

L'API de consultation et la partie web QuChemPedIA est accessible à travers une url publique ce qui donne l'accès à l'application de production.

La mise en place de l'environnement de production se fera à l'aide d'Ansible qui permettre un déploiement des différentes applications à la volé.

5.2 Ansible

Pour la réalisation d'une recette avec Ansible nous nous sommes appuyés sur ce tutoriel en français de Grafikart qui explique la base de la configuration.

<https://www.youtube.com/watch?v=DwNapBHypE8>

Voici la documentation d'Ansible :

<https://docs.ansible.com/ansible/latest/index.html>

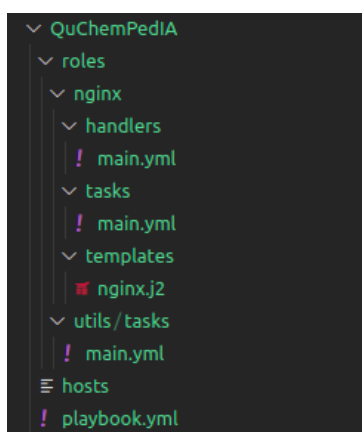


FIGURE 15 – Exemple architecture pour la recette Ansible Web (Statique)

Pour ce qui est de la documentation de la recette elle reste en soit assez intuitive étant donné que c'est juste de la configuration.

Il suffit de retranscrire une installation basique. Par exemple pour les API sous Flask nous avons suivi le tuto (Recherche Google) :

How to serve Flask applications with gunicorn and nginx on Ubuntu - Digital Ocean

hosts

Le fichier hosts contient l'adresse des VM sur lesquelles nous allons déployer. Soit nous indiquons une IP ou alors on peut exécuter la recette en local en indiquant 'localhost'.

playbook.yml

Le playbook est le fichier principal c'est ici que l'on définit les rôles que nous allons utiliser pour procéder à l'installation. Dans l'exemple ci-dessus nous avons deux rôles qui sont 'nginx' et 'utils'.

roles

Le dossier 'rôles' contient chaque grande étape du déploiement. Créer des rôles permet de clarifier l'architecture du projet Ansible.

Chaque rôle doit contenir un dossier 'tasks' avec son fichier 'main.yml' et c'est dans ce dernier que l'on définit les task du rôle en question.

(Facultatif) Nous pouvons définir un dossier 'handlers' qui contient des tasks réutilisable en appelant ce dernier dans le 'main.yml' à l'aide de 'notify'.

(Facultatif) Enfin nous pouvons apercevoir le dossier templates qui contient des fichiers que l'on va pouvoir copier coller depuis la recette vers la VM. Nous pouvons utiliser des variables grâce au moteur de template Jinja intégré à Ansible.

Lancer la recette Ansible

Pour lancer Ansible il faut faire la commande suivante :

```
1 ansible-playbook -i hosts playbook.yml
```

Un exemple de logs de la recette pour l'API d'administration est disponible en Annexe

6 Documentation

L'une des demandes majeure de ce projet était de documenter le code et le projet en général pour que les futurs développeurs puissent reprendre ce projet le plus aisément possible. C'est pourquoi nous avons un dossier "Documentation" dans le projet permettant de regrouper toute la documentation du projet.

6.1 Documentation du code

Concernant le code, nous avons tout d'abord demandé aux développeurs du Back-End d'utiliser la norme PEP8 afin d'améliorer la clarté du code et de l'uniformiser. PEP8 est la norme la plus utilisée pour mettre en forme son code Python.

De plus, tout au long du développement, les M1 ont commenté la majorité de leur code afin d'améliorer la compréhension et la lisibilité. En complément, nous avons ajouté de la documentation générée automatiquement pour le Front-end et le Back-end.

Pour le Front-end nous avons utilisé JSDoc qui nous a permis de générer une page html regroupant toutes les fonctions utilisées.

Voici un aperçu :

```
nextPage(current_page, entree_page)
Pagination - To go to page + 1
Defined in: Pagination.js.

Parameters:
  (int) current_page
  (int) entree_page

-----
pageSelect(pageid, pageid_clone, total_result, entree_page)
To set active page on our two pagination
Defined in: Pagination.js.

Parameters:
  (int) pageid
  (int) pageid_clone
  (int) total_result
  (int) entree_page

-----
pagination(entree_page, result_length, page_number)
Function to set a pagination for the search function
Defined in: Pagination.js.

Parameters:
  (int) entree_page
  (int) result_length
  (int) page_number

-----
plusTen(current_page, entree_page)
Pagination - To go to page + 10
Defined in: Pagination.js.

Parameters:
  (int) current_page
  (int) entree_page

-----
prevPage(current_page, entree_page)
Pagination - To go to page - 1
Defined in: Pagination.js.

Parameters:
  (int) current_page
  (int) entree_page
```

FIGURE 16 – Documentation JSDoc

Au niveau du Back-end, nous avons aussi utilisé une génération automatique de documentation grâce à pydoc.

Voici un aperçu du résultat :



FIGURE 17 – Documentation pydoc

Nous avons décidé d'utiliser JSDoc et pydoc puisque ce sont des générateurs très connus et très simple d'utilisation, pour que les futurs développeurs continuent d'utiliser cette documentation facilement, cela n'ajoute pas une technologie compliquée à utiliser.

Afin de générer ces documentations, nous avons créé des scripts shell, il suffit donc d'exécuter le script "generateDoc.sh" à la racine du projet afin de générer cette documentation qui sera automatiquement placée dans le dossier "Documentation"

6.2 Documentation du projet

Nous avons aussi effectué de la documentation sur le projet en général.

Tout d'abord, nous avons un cahier fonctionnel pour entrer plus dans les détails des fonctionnalités proposées, ce document n'est pas forcément limité aux développeurs mais peut aussi être consulté pour prendre connaissance plus en détails du projet.

Nous avons aussi un cahier technique destiné aux développeurs qui vont reprendre le projet afin de connaître la structure de ce dernier, ses spécificités et de pouvoir reprendre le projet plus facilement.

7 Conclusion

La reprise de QuChemPedIA s'est faite tout naturellement étant donné que c'est un projet pour lequel nous avons déjà contribué l'année dernière. Nous connaissions le sujet ainsi que l'ambition que ce dernier doit avoir. Le changement cette année s'est effectué sur le rôle qui nous a été attribué, nous sommes passés de développeurs à manager de projet. Une transition qui n'a pas été de tous repos au vu des nombreuses difficultés que nous avons rencontrés. Cependant nous avons appris de nos erreurs et nous en ressortons plus avertis.

Concernant la phase technique qui s'est déroulé dans la seconde partie du projet nous avons eu l'occasion de poursuivre les développements et de manipuler les nouvelles technologies utilisées sur QuChemPedIA.

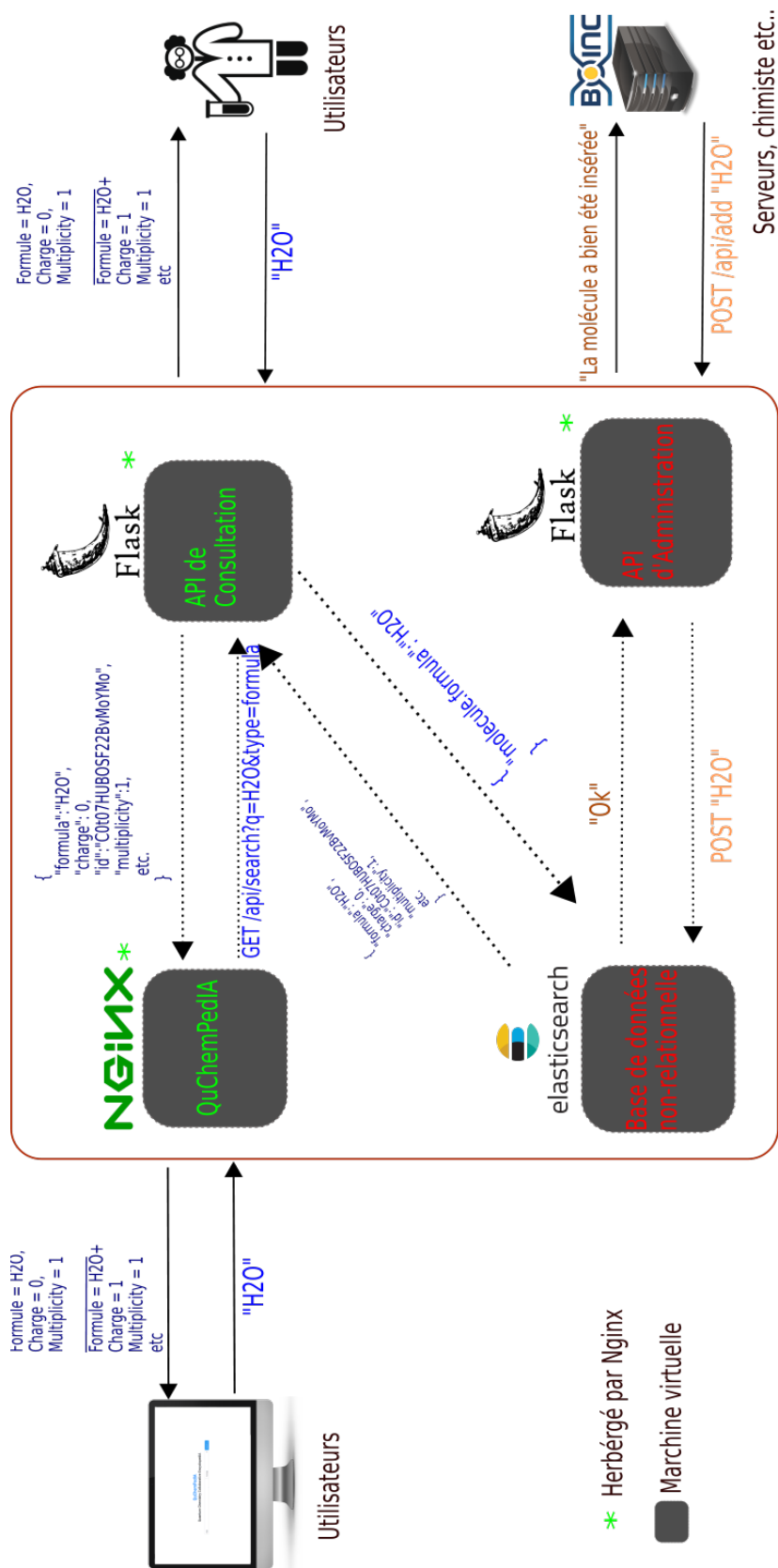
L'application du cours de Management de Projet fut, en somme, très enrichissante car ce cours couvre toutes les phases de développement d'un projet de sa création en passant par ses tests jusqu'à la mise en production de ce dernier. Sans oublier la notion de gestion d'équipe et de la communication entre les parties prenantes.

QuChemPedIA est destiné à évoluer, c'est pourquoi ce projet sera reconduit pour de futurs étudiants, ce qui nous a permis d'accentuer l'importance de la documentation.

Table des figures

1	La page d'accueil du site QuChemPedIA	4
2	Exemple de tests API Consultation	6
3	Schéma tâches développeurs	8
4	Schéma tâches chefs de projet	9
5	Aperçu des salons Discord	10
6	La structure de QuChemPedIA après la refonte	11
7	Arborescence du site Web	12
8	La page d'accueil de QuChemPedIA	12
9	Résultat de la recherche de <i>CHBrClf</i>	13
10	La page détail d'une molécule en mode onglet	14
11	La page détail d'une molécule en mode impression	15
12	Arborescence de l'API de Consultation	16
13	Arborescence de l'API d'Administration	19
14	Logo de Bonsai	20
15	Exemple architecture pour la recette Ansible Web (Statique)	21
16	Documentation JSDoc	23
17	Documentation pydoc	24

Annexe



```

root@quchempedia-pri:~/ansible# ansible-playbook -i hosts playbook.yml

PLAY [Installation de l'API d'Administration] *****
TASK [Gathering Facts] *****
ok: [localhost]

TASK [utils : Installation des paquets] *****
ok: [localhost]

TASK [user : Création d'un utilisateur] *****
ok: [localhost]

TASK [user : Création du dossier .ssh] *****
ok: [localhost]

TASK [user : Génération d'une paire de clé RSA] *****
ok: [localhost]

TASK [user : quchempedia_admin devient sudoers] *****
ok: [localhost]

TASK [nginx : Installation Nginx] *****
ok: [localhost]

TASK [nginx : Démarrage du service Nginx] *****
ok: [localhost]

TASK [nginx : Supprimer default.conf sites-enabled] *****
ok: [localhost]

TASK [nginx : Création du dossier] *****
ok: [localhost]

TASK [nginx : Clonage du site quchempedia.univ-angers.fr depuis https://github.com/younesr19/QuChemPedIA2020.git] ***
changed: [localhost]

TASK [nginx : Création du dossier quchempedia.univ-angers.fr dans le répertoire www] *****
ok: [localhost]

TASK [nginx : Création du dossier logs Nginx] *****
ok: [localhost]

TASK [nginx : Lien vers www] *****
ok: [localhost]

TASK [nginx : Modification des droits du dossier Api_Administration] *****
changed: [localhost]

TASK [nginx : Création de la configuration Nginx] *****
ok: [localhost]

TASK [nginx : Activation du domain quchempedia.univ-angers.fr] *****
ok: [localhost]

TASK [flask : Création d'un environnement virtuel python] *****
changed: [localhost]

TASK [flask : Installation des dépendances Python du projet] *****
ok: [localhost]

TASK [flask : Création du Systemd Unit File] *****
ok: [localhost]

TASK [flask : Démarrage du processs Systemd] *****
changed: [localhost]

PLAY RECAP *****
localhost : ok=21  changed=4  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```