GAUVIN Florian

BTS Services Informatiques aux Organisations
Option SLAM

Lycée Technique Marie Curie
16 Boulevard Jeanne d'Arc

Année scolaire 2020/2021

Création d'un site web réalisant des calculs de Thermochimie

(Avec base de données)

Stage de 1° année de BTS SIO
Du 25 mai 2021 au 02 juillet 2021

REMERCIEMENTS

Durant ma première année de BTS SIO, j'ai effectué un stage à la Faculté de Pharmacie de Marseille.

Je tiens dans un premier temps, à remercier mon tuteur de stage BERGE-LEFRANC David, responsable du secteur du secteur « Service Chimie Physique et Prévention des Risques et Nuisances Technologiques » de la Faculté de Pharmacie de Marseille, de m'avoir accueilli comme stagiaire au sein de l'établissement.

Je tiens à remercier par la suite, Alessandro-Umberto Margueritte, mon binôme et étudiant de master PRNT à la Faculté de Pharmacie, pour les différents renseignements qu'il m'a apportés durant mon stage.

Je remercie tous les professeurs du BTS SIO dont les cours m'ont apporté des connaissances théoriques et pratiques permettant de mener à bien le projet.

Enfin, je tiens à remercier le personnel ainsi que tous les enseignants chercheurs du secteur « Service Chimie Physique et Prévention des Risques et Nuisances Technologiques » pour leur accueil durant ces six semaines de stage. Toutes ces personnes ont contribué, par leur disponibilité, leurs compétences et leur bonne humeur, à rendre mon stage enrichissant et motivant.

Sommaire

Table des matières

Contexte du stage

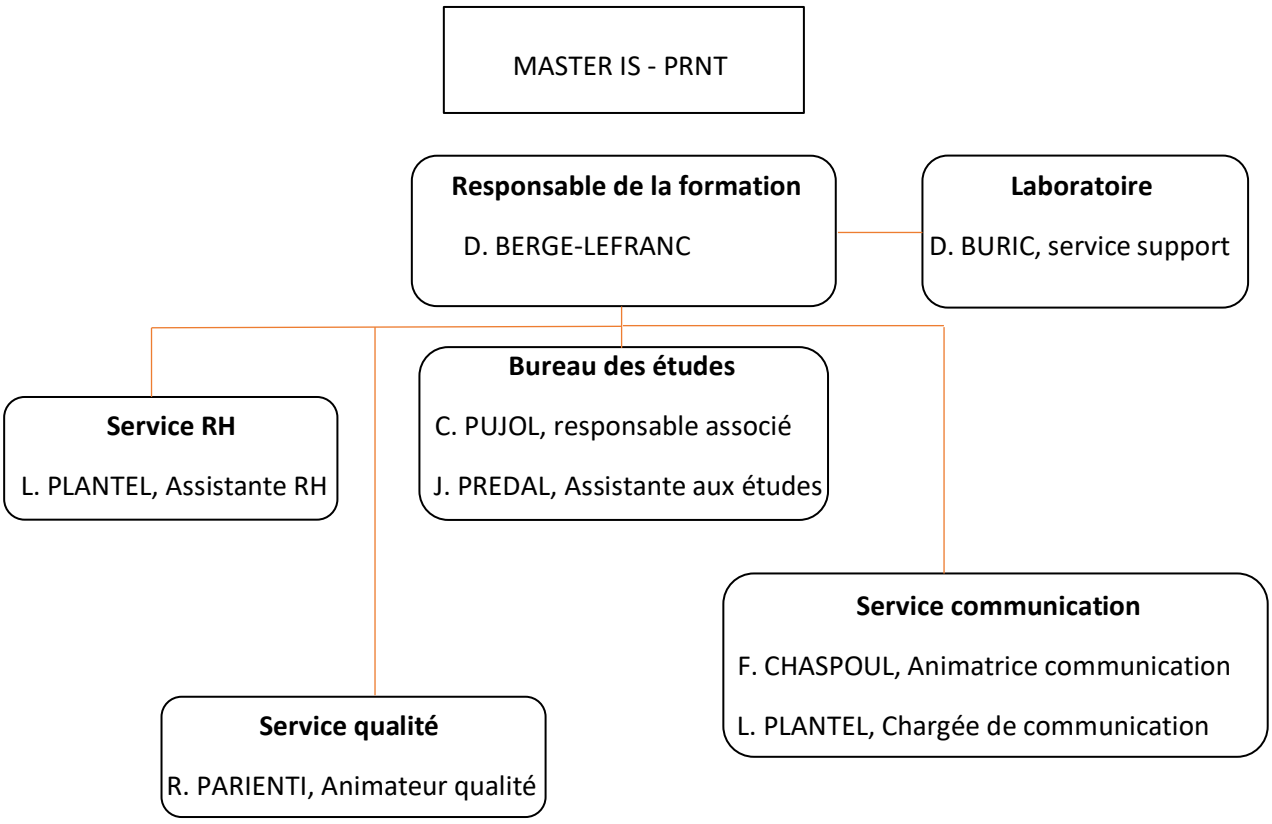
Introduction

Pour commencer, mon stage ne s'est pas réalisé dans une entreprise, mais à la Faculté de pharmacie de Marseille. Plus précisément dans le secteur : « Service Chimie Physique et Prévention des Risques et Nuisances Technologiques ». Les chercheurs réalisant beaucoup de calculs avec différentes méthodes, un projet d'automatisation des calculs a été lancé pour à terme être publié publiquement et aider nombre de chercheurs.

Le but premier de ce stage était donc de réaliser un site web, regroupant plusieurs feuilles de calculs de thermochimie. Cependant, avant de commencer cela, d'autres tâches m'ont été confiées tel que : une veille de sécurité de leurs matériels informatiques et une aide sur leur site web, concernant la mise en place d'une redirection vers un lien PDF.

Présentation du secteur

Tel que présenté précédemment, mon stage c'est déroulé à la Faculté de Pharmacie de Marseille dans le secteur : « Service Chimie Physique et Prévention des Risques et Nuisances Technologiques ». Ce service est composé de deux domaines, l'enseignement et la recherche, comprenant un personnel de six personnes, dont voici l'organigramme :



Concernant les activités, elles se scindent en deux parties : l'enseignement et gestion du Master formant des étudiants à la prévention des risques afin de garantir la santé de l'Homme et de son environnement ainsi qu'une partie recherche avec une approche chimie physique des problématiques liées à la santé et à l'environnement. Notamment avec plusieurs modèles mathématiques, abordés durant le projet, qui permettent de définir les paramètres énergétiques qui gouvernent les phénomènes de solubilité de pesticides organochlorés.

Présentation du matériel et des logiciel utilisés

Durant ce stage, j'ai pu faire face à de nombreux logiciels. En effet, le projet étant développé en langage python et avec Flask, j'ai dû installer de nombreux logiciels permettant le travail en local. Dans cette partie, je vais donc présenter les différents logiciels que nous avons utilisés pour la mise en place du site web.

Anaconda est un logiciel libre et open source. Celui-ci est très utile pour la programmation en python puisqu'il vise à simplifier la gestion des paquets (les imports) et de déploiement. Il regroupe plus de 250 paquets, les plus populaires auprès des développeurs en langage python. De plus, le navigateur Anaconda est une interface graphique, qui permet aux développeurs de lancer des applications, ainsi que de gérer les librairies conda (les imports) et les environnements sans utiliser de ligne de commande. Ce logiciel est très utile pour les développeurs débutant en python, ce qui m'a permis de comprendre très vite comment coder dans ce langage. De plus, celui-ci permet l'utilisation facile et rapide de l'environnement Flask en local.

Flask est un micro Framework open source permettant le développement web avec le langage python. Celui-ci est dit micro Framework, car il est très léger. En effet, il est composé d'une base simple mais extensible par de nombreuses extensions, qui permettent d'ajouter facilement des fonctionnalités. Malgré la légèreté de celui-ci, il n'en ait pas moins puissant et supporte tout de même des applications lourdes. Flask dispose de plusieurs fonctionnalités, la plus importante que nous avions utilisé pour le projet est le moteur de template, pour du rendu web avec une page HTML que nous verrons plus tard.

WampServer est une plateforme de développement Web permettant de faire fonctionner des scripts PHP sur une machine en local. Celui-ci est un environnement comprenant trois serveurs (Apache, MySQL et MariaDB), un interpréteur de script PHP, ainsi que phpMyAdmin pour l'administration Web des bases de données MySQL. Ce logiciel est particulièrement utile pour la manipulation rapide de bases données avec l'interface graphique de phpMyAdmin. Durant ce projet, nous l'avions utilisé pour la mise en place des bases de données de tous les programmes de calculs avant de les héberger. Cela m'a permis de réaliser plusieurs tests de compatibilité et ainsi rendre les bases de données optimales pour accueillir toutes les données.

Sublime Text est un éditeur de texte prenant en charge 44 langages au total. Celui-ci est très intéressant, car il permet l'utilisation de macro, mais aussi l'ajout de plug-in utile pour coder. Sublime Text a été l'éditeur de texte que j'ai appris à utiliser durant ce stage.

Le travail réalisé

Présentation du travail demandé

Tout d'abord, le projet est effectué en duo avec un étudiant du MASTER-PRNT : Alessandro-Umberto Margueritte, qui a développé la partie calculs de thermochimie en python.

Avant de démarrer, nous avons réalisé une entrevue avec Alessandro-Umberto Margueritte, Monsieur BERGE-LEFRANC David. Cela nous a permis de déterminer l'environnement de travail (web ou application) et les attentes de chacun. Alessandro-Umberto Margueritte ayant déjà commencé à faire les calculs dans le langage python, nous avons convenu que le développement s'effectuerait en python avec Flask et donc sous forme de site web.

De plus, suite à cette entrevue, nous avions déterminé qu'il y avait neuf méthodes de calculs à automatiser. Neuf bases données et neuf IHM différents étaient donc à prévoir sur le site web.

Tâches réalisées

Durant ces six semaines, plusieurs tâches m'ont été confiés. En effet, le projet n'étant pas lancé dans les premières semaines, j'ai dû réaliser des tâches sur place, au sein du secteur.

Veille de sécurité

Dans un premier temps, mon tuteur de stage m'a demandé d'effectuer une veille de sécurité sur les ordinateurs qu'ils utilisaient, c'est-à-dire quatre machines et ainsi par la suite lui envoyer un récapitulatif sous la forme d'un tableau Excel.

Afin de mener à bien ce travail, j'ai tout d'abord effectué une liste d'éléments à vérifier sur toutes les machines tel que : le pare-feu, l'anti-virus ou encore le système d'exploitation. Suite à cela, le jour suivant, j'ai effectué la vérification sur les différents postes de travail des enseignants chercheurs et ainsi remplie le tableau Excel.

	A	B	C	D	E
1		Jeanine PREDAL	Léria PANTEL	Duje Buric	Florence CHASPOUL
2	Type	Windows	Mac	Windows	Windows
3	Version	Windows 7	Version 11.2.3	Windows 10	Windows 10
4	SE à jour	Non	Non	Non	Non
5	Anti-virus	Kaspersky	Non	MacAfee	Windows Defender
6	Anti-virus à jour	Licence expiré	Non	Non opérationnel	Oui
7	Licence office à jou	Oui	Oui	Oui	Oui
8	Pare-feu	Désactivé	Activé	Activé	Activé

Cette veille de sécurité a permis de réaliser un récapitulatif global concernant la cybersécurité dans le secteur. Ce travail a ensuite été transférer à monsieur AGNIEL Florant responsable de l'informatique, pour prévoir une entrevue.

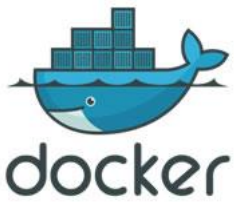
Lors de l'entrevue, plusieurs thèmes concernant la sécurité des machines ont été abordés. En effet, la problématique étant de trouver le moyen le plus fiable de protéger les ordinateurs, des pistes concernant un VPN et un nouveau système d'exploitation ont été retenues.

Dans cet échange, j'ai pu apporter mes connaissances acquises lors de l'année en cybersécurité, pour donner un avis général sur la situation et conseiller différents outils de VPN et de protection pour un ordinateur tel que : Kaspersky concernant le VPN et l'anti-virus ainsi que Linux pour le système d'exploitation.

Base de données MySQL

Dans les semaines suivantes, le projet commence. Avec mon binôme, nous avons réalisé une visio-conférence pour répartir les tâches et planifier tous les besoins. Alessandro-Umberto n'ayant aucune expérience en base de données et MySQL, c'est naturellement moi qui ai pris en charge cette partie. Lors de l'entrevue, nous avions convenue qu'il serait préférable d'ajouter une colonne « ID » et « Date » dans notre base de données, en plus de celles concernant les valeurs saisies. À ce moment, nous avons fait face à une première difficulté : comment créer notre base de données en local et avec quel outil cela serait le plus optimal.

Dans un premier temps, Alessandro-Umberto a proposé l'utilisation de Docker pour créer notre base de données et la lier à l'application python.



Celui-ci est un logiciel libre utilisé par nombre d'entreprise, il permet de lancer plusieurs applications dans des conteneurs logiciels. Plus simplement, Docker permet de diviser une application en plusieurs parties isolées appelées « conteneur », ceux-ci pouvant êtres des conteneurs MySQL, Python, HTML… Cela est différent de la virtualisation, car Docker s'appuie sur certaines parties de la machine hôte pour son fonctionnement, neutralisant tout problème de compatibilité et de dysfonctionnement de services en cas de panne d'un conteneur.

Je me suis donc renseigné pendant plusieurs jours sur son utilisation, en réalisant des tests et son utilité par rapport à notre projet. En effet, j'ai travaillé sur la création d'un conteneur « phpmyadmin/MySQL ». Malgré l'accès à une documentation, je me suis vite retrouvé en difficulté pour relier les conteneurs entre eux et tout faire fonctionner.

Notre application étant légère, j'en ai déduit après réflexions que ce logiciel ne serait pas indispensable au bon fonctionnement du projet.

Pour créer la base de données en local, je me suis donc orienté vers un logiciel que nous avons utilisé toute l'année en cours : Wampserver. Connaissant déjà son fonctionnement, cela m'a permis de mettre en place rapidement la base de données de tous les programmes, que j'ai appelée « Python ».

Pour commencer, j'ai réalisé neuf tables différentes dans la base de données, correspondant à toutes les méthodes de calculs. Celles-ci sont composées chacune d'une colonne « ID », « Date » ainsi que celles correspondant aux valeurs saisies pour le calcul.

Nous allons voir ici, la table correspondante au premier calcul nommé la méthode « Wilson ».

	ID	Date	deltaH	tempfus	vmH	vmC	ΔI12	ΔI21
Éditer Copier Supprimer	46	2021-06-29 15:52:07	10	2	26	25	79994.05	58537.89
Éditer Copier Supprimer	45	2021-06-29 15:47:56	0.4	25	100	0.25	737434.6	71262.22
Éditer Copier Supprimer	44	2021-06-29 15:47:36	23	24	26	25	79976.63	64594.82
Éditer Copier Supprimer	47	2021-06-30 10:55:12	15	24	65	24	70840.14	60948.35
Éditer Copier Supprimer	48	2021-06-30 13:33:54	15	13	26	24	80699.89	59273.1

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	ID	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT	Modifier Supprimer Plus
2	Date	datetime			Non	CURRENT_TIMESTAMP			Modifier Supprimer Plus
3	deltaH	float			Oui	*NULL*			Modifier Supprimer Plus
4	tempfus	float			Oui	*NULL*			Modifier Supprimer Plus
5	vmH	float			Oui	*NULL*			Modifier Supprimer Plus
6	vmC	float			Oui	*NULL*			Modifier Supprimer Plus
7	ΔI12	double			Oui	*NULL*			Modifier Supprimer Plus
8	ΔI21	double			Oui	*NULL*			Modifier Supprimer Plus

Nous pouvons apercevoir huit champs différents. Pour chaque calcul, j'ai déterminé un ID, clef primaire de la table, qui est incrémenté automatiquement après chaque envoie du formulaire. Cela nous sera utile plus tard lors de l'affichage de la base de données sur la page web. Ensuite, une des exigences étant la date à laquelle le calcul a été fait, j'ai ajouté un champs Date qui est de type date et récupère automatiquement la date du jour lors de l'envoie du calcul.

Concernant les champs des valeurs saisies par l'utilisateur, celles-ci sont récupérées sous forme de nombre à virgule (type float) et déterminer comme null si elles ne sont pas saisies. Pour finir, ΔI12 et ΔI21 sont les résultats déterminés en fin de calcul, que j'ai déterminé en type double pour plus de précision lors du résultat. Le type double est codé sur 64 bits et prend 8 octets de stockage, ce qui est deux fois plus que le type float.

Pour finir, j'ai appliqué cela de la même manière, à toutes les tables des différentes méthodes de calculs. La mise en place total de la base de données étant terminée, j'ai débuté mon travail sur l'insertion des valeurs avec un IHM en langage python.

IHM et requête d'insertion.

Après la mise en place de la base de données, il était nécessaire de travailler sur un IHM (formulaire) pour récupérer les données. Cependant, Alessandro-Umberto ayant déjà commencé à travailler dessus, ma tâche était d'adapter son code à l'insertion de valeurs dans notre base de données.

Pour commencer, je me suis documenté sur comment réaliser un script de connexion à une base de données. Grâce aux cours de développement, j'ai pu comprendre et adapter les scripts de connexion PHP sous langage python. Cependant, travaillant sous python, j'ai dû importer un nouveau package (librairies) : PyMySQL.

Celui-ci permet la connexion à une base de données à l'aide d'un « cursor » et donc la mise en place d'une communication. Nous allons donc voir à quoi cela ressemble.

```
import pymysql #Pour utilisation base de donnée
```

Ceci est l'import du package PyMySQL dans l'environnement. Cela est obligatoire dans le script python car il nous permet d'utiliser toutes les commandes associées, qui ne fonctionnerait pas sans celui-ci.

```
connection = pymysql.connect(host="localhost",user="root",passwd="",database="python")
cursor = connection.cursor()
```

Dans ces deux lignes de codes, nous pouvons apercevoir des similitudes avec un script de connexion en PHP :

```
<?php
$dns = 'mysql:host=localhost;dbname=python;charset=utf8';

$utilisateur = 'root';
$motDePasse ='';

$connection = new PDO($dns, $utilisateur, $motDePasse);
?>
```

En effet, la première ligne définie la connexion à la base de données avec une variable « connection ». De plus, dans celle-ci nous déterminons l'IP de connexion (host), ici localhost, ainsi que nos identifiant puis pour finir le nom de la base de données (ici « python »). À cela s'ajoute la commande **pymysql.connect** permettant de déterminer une connexion. En dessous, nous déterminons un curseur, ici nommé « cursor », qui va permettre d'exécuter des requêtes d'insertion…

La connexion avec la base de données étant établie, j'ai commencé à réaliser les requêtes d'insertion. Pour ce faire, je me suis orienté vers des requêtes à paramètres inconnus, vue en cours. Le but est de préparer une requête dans une variable et l'exécuté avec des paramètres, pouvant êtres des valeurs fixes ou des variables.

Dans un premier temps, j'ai regroupé toutes les valeurs saisies ainsi que les deux valeurs de sortie (params[0][0],2…) dans une variable « result » :

```
#Récupération des valeurs dans une variable
result=(deltaH, tempfus, vmH, vmC, {round(params[0][0],2)}, {round(params[0][1],2)})
```

Dans un second temps, je prépare ma requête d'insertion avec des paramètres inconnus :

```
#requête d'insertion avec paramètres inconnus
insert2 = "INSERT INTO programme1(deltaH, tempfus, vmH, vmC, Δl12, Δl21) VALUES(%s, %s, %s, %s, %s, %s);"
```

INSERT INTO programme1(deltaH, tempfus…) VALUES (%s,…) est une requête SQL qui permet d'insérer dans la table programme1 les valeurs de saisies sous forme de paramètre inconnus (%s).

Par la suite, j'exécute cette requête « insert2 » avec la commande cursor.execute en ajoutant comme paramètres, toutes les valeurs contenues dans la variable « result » :

```
#execution requete avec les variables dans result
cursor.execute(insert2, result)

connection.commit()
connection.close()
```

Pour finir, les commandes connection.commit() et connection.close() permettent respectivement d'enregistrer les modifications apportées et « fermer » la connexion à la base de données.

Dans un second temps, après la confection du code et des requêtes, il est nécessaire de réaliser l'IHM

correspondant. Celui-ci comprend les quatre valeurs de saisie : deltaH, tempfus, vmH et vmC et ensuite un bouton d'envoie. Voici le code correspondant :

```
<form class="valeurs" accept-charset="UTF-8" action="/calculate" autocomplete="on" method="POST" >

    <label for="deltaH"><t>ΔHfus</t></label><br/>
    <input class="deltaH" name="deltaH" type="text" value="" required> <br/>

    <label for="tempfus"><t>Tfus</t></label><br/>
    <input class="tempfus" name="tempfus" type="text" value="" required> <br/>

    <label for="vmC"><t>Vm CLD</t></label><br/>
    <input class="vmC" name="vmC" type="text" value="" required> <br/>

    <label for="vmH"><t>Vm H2O</t></label><br/>
    <input class="vmH" name="vmH" type="text" value="" required> <br/>

    <button class="Send" type="submit" value="Calculate"><b>Submit</b></button>

</form>
```

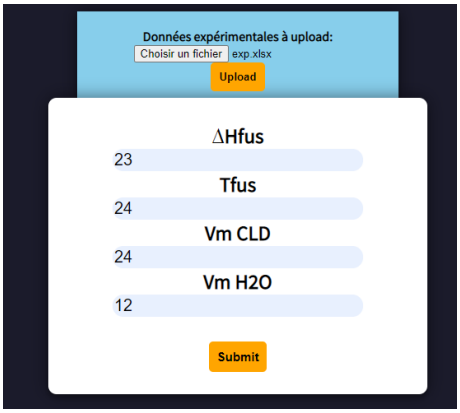
Comme nous pouvons l'apercevoir, ce formulaire contient des zones de texte **(type= « text »)** et un bouton d'envoie **(type= « submit »).** Lors de l'envoie du formulaire, celui-ci renvoie vers la fonction « calculate » **(action= « /calculate »)**, contenue dans le script python qui effectuera le calcul. La partie récupération des valeurs ainsi que le code du calcul ont été développés par Alessandro-Umberto, qui a utilisé des librairies/packages permettant l'affichage d'un graphique en fonction des valeurs saisies.

<u>Le code complet de la fonction est le suivant :</u>

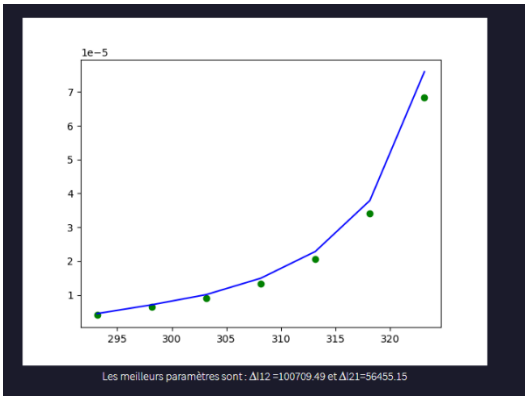
```
@app.route("/calculate", methods=['POST'])
def calculate():
    connection = pymysql.connect(host="localhost",user="root",passwd="",database="python")
    cursor = connection.cursor()

    deltaH = float(request.form['deltaH'])
    tempfus = float(request.form['tempfus'])
    vmH = float(request.form['vmH'])
    vmC = float(request.form['vmC'])

    exp_values=pd.read_excel("./uploads/exp.xlsx", engine='openpyxl')
    xs=exp_values["x"]
    temps=exp_values["T"]
    ydata=xs
    xdata=temps


    def WilsonTot(temps,delta21,delta12):
        def Wilson(x,temps):

            return math.exp((deltaH/8.31)*((1/tempfus)-(1/temps)))-(-math.log(x+(vmH/vmC)*math.exp(-delta12/(8.31*temps))*(1-x))+(1-x)*(((vmH/vmC)*math.exp(-delta12/(8.31*temps)))/(x+(vmH/vmC)*math.exp(-delt

        return [Wilson(x,temp) for x,temp in zip(xs,temps)]

    params = optimize.curve_fit(WilsonTot, xdata, ydata)
```

```
# Generate the figure **without using pyplot**.
fig = Figure()
ax = fig.subplots()
ax.scatter(xdata,ydata,color='green')
ax.plot(xdata, WilsonTot(temps, params[0][0],params[0][1]),color='blue')
# Save it to a temporary buffer.
buf = BytesIO()
fig.savefig(buf, format="png")
encoded_img_data = base64.b64encode(buf.getvalue())
textresult= f'Les meilleurs paramètres sont : Δ112 ={round(params[0][0],2)} et Δ121={round(params[0][1],2)}'
# Embed the result in the html output.

#Récupération des valeurs dans une variable
result=(deltaH, tempfus, vmH, vmC, {round(params[0][0],2)}, {round(params[0][1],2)})

#requête d'insertion avec paramètres inconnus
insert2 = "INSERT INTO programme1(deltaH, tempfus, vmH, vmC, Δ112, Δ121) VALUES(%s, %s, %s, %s, %s, %s);"

#execution requete avec les variables dans result
cursor.execute(insert2, result)

connection.commit()
connection.close()
return render_template("app.php",img_data=encoded_img_data.decode('utf-8'),textresult=textresult)
```

À cela s'ajoute, mon travail sur l'apparence du formulaire. En effet, durant ce projet, je me suis occupé de la partie apparence du site web (détaillé plus tard) et donc par conséquent du style du formulaire :



De plus, nous pouvons apercevoir une partie upload de fichier. Celle-ci est nécessaire, car elle sert à envoyer un fichier Excel contenants les valeurs expérimentales, nécessaire au calcul. Ici, mon travail s'est concentré sur la partie visuelle, pendant que mon binôme travaillait sur la mise en place de l'upload et l'affichage du graphique :



Affichage de la base de données

Dans cette partie, nous allons voir comment l'affichage d'une base de données avec pagination est effectué en python.

Premièrement, avant la mise en place de la pagination il est nécessaire de réaliser un affichage simple de la base de données, comme en PHP. Pour ce faire, une requête SQL est effectuée :

```
cursor.execute("SELECT * FROM programme1")
result = cursor.fetchall()
```

Tel que montré précédemment, nous utilisons ici la méthode de curseur, mais ici en exécutant une requête directement. Pour finir, le résultat est placé dans la variable « result » à l'aide de la méthode **fetchall()** qui permet, comme en php, de stocker sous forme de tableau les valeurs.

Suite à cela, nous renvoyons le résultat de la requête dans le « template » d'affichage :

```
return render_template('TestAff.html', result=result
```

Cela va nous permettre de réaliser un affichage à l'image d'un script PHP, c'est-à-dire avec une boucle « pour », dans la page HTML correspondante.

```
<table class="table table-striped">
<thead>
<tr>
    <th>Date</th>
    <th>deltaH</th>
    <th>tempfus</th>
    <th>vmH</th>
    <th>vmC</th>
    <th>Δl12</th>
    <th>Δl21</th>
    <th>Action</th>
</tr>
</thead>
{% for row in result %}
<tbody>
    <tr>
        <td>{{ row[1] }}</td>
        <td>{{ row[2] }}</td>
        <td>{{ row[3] }}</td>
        <td>{{ row[4] }}</td>
        <td>{{ row[5] }}</td>
        <td>{{ row[6] }}</td>
        <td>{{ row[7] }}</td>
        <td>
        <a href="/delete/{{row[0]}}" class="btn btn-danger btn-delete btn-sm">Delete</a>
    </td>
    </tr>
</tbody>
{% endfor %}
</table>
```

Tout d'abord, l'affichage de la base de données a été effectué dans un tableau. En effet, nous pouvons apercevoir la structure d'une « table » se répétant dans une boucle « for ». Grâce à cela, pour chaque valeur contenue dans la variable result **(for row in result)** nous afficherons la donnée une, puis deux (**(row[1], row[2]…),** jusqu'à sept qui correspondent respectivement à chaque colonne de notre table « programme 1 » dans la base de données.

Dans un second temps, une fois l'affichage fonctionnel, il était nécessaire de mettre en place une pagination pour éviter une surcharge de la page. En effet, celle-ci permet de restreindre le nombre de données à afficher en mettant en œuvre un parcours des valeurs de la table à l'aide de boutons.

Pour commencer, il est nécessaire d'importer plusieurs nouveaux packages :

```
#pagination
from flask_sqlalchemy import SQLAlchemy
from flask import Blueprint
from flask_paginate import Pagination, get_page_parameter
```

SQLAlchemy est un ORM (Object-Relational Mapping) qui permet de synchroniser les classes avec des tables en base de données relationnelles (basée sur SQL). À cela s'ajoute flask_paginate, qui est obligatoire si l'on veut faire une pagination pour créer un paramètre de page (p=2...) ainsi que les boutons « pagination.links » permettant de naviguer entre les pages.

Ensuite, une fois les packages importés, nous déterminons le nombres limite d'affichage ainsi que l'offset. L'offset est très important car c'est grâce à lui que les valeurs de la page une ne seront pas afficher sur la page deux etc.

```python
q = request.args.get('q')
if q:
    search = True
page = request.args.get(get_page_parameter(), type=int, default=1)

limit=10 # perpage
offset = page*limit-limit
```

Tout d'abord, nous déterminons un compteur de page, nommé « q ». Celui-ci est de type entier et a comme valeur par défaut un. Pour ce projet, j'ai déterminé qu'un affichage de dix lignes par page suffirait, je le déclare donc dans une variable « limit ». Ensuite, cette variable nous permet de calculer l'offset qui affichera les dix données suivantes au fur et à mesure.

Pour l'exemple, situons-nous à la page numéro deux. Cela donnera donc 2*10-10 = 10. On aura donc l'affichage de 10 lignes à partir de la ligne numéro 10 à la page deux.

Une fois l'offset et la limite déterminés, nous les appliquons à une requête SQL :

```python
total = len(result)
cur = connection.cursor()
cur.execute("SELECT * FROM programme1 ORDER BY ID DESC LIMIT %s OFFSET %s", (limit, offset))

data = cur.fetchall()
cur.close()

pagination = Pagination(page=page, per_page=limit, total=total, record_name='prog1-BDD')
return render_template('TestAff.html', pagination=pagination,
                       data=data)
```

Tout d'abord, on détermine le nombre de valeurs totales grâce à la fonction len(), qui identifie le nombre de données stockées dans une variable (ici result).

Ensuite, on exécute la requête SQL de sélection des données de manières décroissantes (pour afficher les plus récentes en premières), en y appliquant les variables limit et offset déterminer précédemment. Les données obtenues grâce à celle-ci sont ensuite stockées dans la variable data (précédemment result).

Pour finir, on renvoie toutes les variables dans le template d'affichage, en passant toute celles nécessaires à la pagination dans la variable « pagination ».

Dans la page HTML de l'affichage, on y ajoute donc ceci :

```
    </table>
<br></br>
{{ pagination.links }}
<br></br>
```

Cette fonction de flask_paginate permet la création automatique des boutons de parcours des pages pouvant se situer sur n'importe quelle ligne de la page web. Ici, ils sont placés en dessous du tableau d'affichage, au centre.

Voici donc le résultat :

Date	deltaH	tempfus	vmH	vmC	ΔI12	ΔI21
2021-07-13 17:14:05	23.0	13.0	65.0	25.0	68358.4	60685.95
2021-07-02 15:22:35	15.0	24.0	26.0	25.0	79975.21	61526.96
2021-06-30 13:33:54	15.0	13.0	26.0	24.0	80699.89	59273.1
2021-06-30 10:55:12	15.0	24.0	65.0	24.0	70840.14	60948.35
2021-06-29 15:52:07	10.0	2.0	26.0	25.0	79994.05	58537.89
2021-06-29 15:47:56	0.4	25.0	100.0	0.25	737434.6	71262.22
2021-06-29 15:47:36	23.0	24.0	26.0	25.0	79976.63	64594.82
2021-06-29 10:37:09	10.0	24.0	26.0	24.0	80698.13	67125.69
2021-06-28 14:32:11	23.0	24.0	10.0	25.0	64400.98	54079.85
2021-06-28 11:11:16	23.0	2.0	65.0	25.0	68362.59	58296.32

« 1 2 3 »

Pour une meilleure optimisation de l'affichage, il m'a semblé naturel de pouvoir supprimer un calcul de la table. Je me suis donc permis de concevoir et affiché des boutons de suppression pour chaque ligne, permettant à l'utilisateur de supprimé un calcul non voulu ou erroné.

Voici le code :

```
@app.route('/delete/<string:id>', methods = ['POST','GET'])
def delete(id):
    connection = pymysql.connect(host="localhost",user="root",passwd="",database="python")
    cursorDelete = connection.cursor()
    value_to_delete = "DELETE FROM programme1 where ID = {0}".format(id)
    cursorDelete.execute(value_to_delete)
    connection.commit()
    flash('Calcul supprimé !', 'warning')
    return redirect(url_for('BDD1'))
```

Comme nous pouvons le voir, ce bouton est composé de seulement une requête SQL de suppression. Cependant, il était nécessaire de récupérer l'ID de la bonne ligne. Cette condition est donc remplie par **« where ID = {0} »** récupérant le **row[0]** (colonne ID de la table MySQL) correspondant à la ligne à supprimer.

La fonction exécute ensuite la requête et affiche un message flash « Calcul supprimé ! » de type warning, permettant de prévenir l'utilisateur lors d'une suppression. Celui-ci est affiché dans la page HTML par la formule suivante :

```
{% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
        {% for category, message in messages %}
        <div class="alert alert-warning" role="alert">
        <span>{{ message }}</span>
        </button>
        </div>
        {% endfor %}
    {% endif %}
{% endwith %}
```

Pour finir, voici le résultat final de l'affichage de la table « programme1 » correspondant à la méthode de calcul Wilson :

Date	deltaH	tempfus	vmH	vmC	Δl12	Δl21	Action
2021-07-13 17:14:05	23.0	13.0	65.0	25.0	68358.4	60685.95	Delete
2021-07-02 15:22:35	15.0	24.0	26.0	25.0	79975.21	61526.96	Delete
2021-06-30 13:33:54	15.0	13.0	26.0	24.0	80699.89	59273.1	Delete
2021-06-30 10:55:12	15.0	24.0	65.0	24.0	70840.14	60948.35	Delete
2021-06-29 15:52:07	10.0	2.0	26.0	25.0	79994.05	58537.89	Delete
2021-06-29 15:47:56	0.4	25.0	100.0	0.25	737434.6	71262.22	Delete
2021-06-29 15:47:36	23.0	24.0	26.0	25.0	79976.63	64594.82	Delete
2021-06-29 10:37:09	10.0	24.0	26.0	24.0	80698.13	67125.69	Delete
2021-06-28 14:32:11	23.0	24.0	10.0	25.0	64400.98	54079.85	Delete
2021-06-28 11:11:16	23.0	2.0	65.0	25.0	68362.59	58296.32	Delete

« 1 2 3 »

Calcul supprimé !

Design du site web

Pour finir, une de mes dernières tâches dans ce projet concerne l'agencement du site. Cependant, n'ayant pas de directive concernant le style graphique, j'ai pris l'initiative d'ajouter des éléments me semblant indispensable dans une page web.

Tout d'abord, suite à une discussion avec Alessandro-Umberto, l'idée d'une barre de navigation a été retenue. Après renseignement, nous nous sommes aperçus qu'il était possible de créer une barre de navigation automatiquement avec python.

Ma tâche ici était donc la réalisation d'une page d'accueil comprenant une barre de navigation, permettant la navigation entre les feuilles de calculs et les base de données. Pour ce faire, il était nécessaire de faire appel à de nouveaux import, flask_nav :

```
#nécessaire navbarre
from flask_nav import Nav
from flask_nav.elements import Navbar, Subgroup, View, Link, Text, Separator
```

Ensuite, vient la création des éléments de la barre de navigation :

```
#éléments navbarre
nav.register_element('my_navbar', Navbar(
    'thenav',
    View('Home Page', 'index'),
    View('Programme 1', 'main'),
    View('Programme 2', 'main'),
    View('Programme 3', 'main'),
))
```

Ici, la barre de navigation se nomme « my_navbar » et est composé de quatre éléments : la page d'accueil et les programmes un, deux et trois. Cependant, n'ayant pas eu le temps de mettre au point qu'une seule feuille de calcul, les deux autres programmes renvoient tout deux vers la page du premier programme.

La « view » est ce qui permet de créer des hyperliens dans la barre de navigation. Celle-ci est composé de deux éléments, son nom et la fonction à laquelle elle renvoie, ici index et main (respectivement la page d'accueil et le programme) :

```
@app.route('/')
def index():

    return render_template('index.php')

@app.route("/prog1")
def main():

    return render_template("app.php")
```

Ensuite, après la création de la première barre de navigation, nous avons décidé d'en créer une deuxième correspondant aux différentes bases de données :

```
#2nd navbar (BDD)
nav.register_element('my_navbar2', Navbar(
    'thenav',
    View('BDD1', 'BDD1'),
    View('BDD2', 'BDD1'),
    View('BDD3', 'BDD1'),
    View('BDD4', 'BDD1')
    ))
```

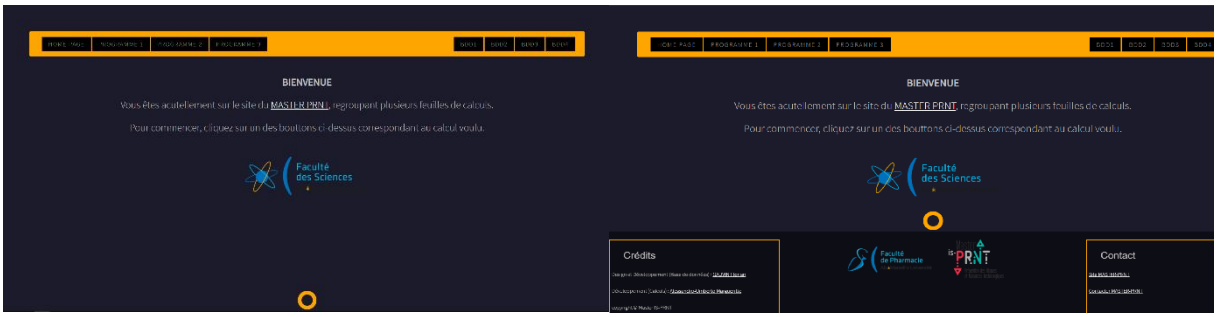
Comme dit plutôt, la méthode Wilson étant la seule totalement fonctionnel, tous les onglets ramènent vers la première table « programme1 ». Pour finir, une fois tout cela mit en place dans le script python, il est nécessaire de les importer dans la page HTML de la manière suivante :

```
<div class="nav-block">
    <div class="nav1">
    {{ nav.my_navbar.render() }}
    </div>
    <div class="nav2">
    {{ nav.my_navbar2.render() }}
    </div>
</div>
```

Comme nous pouvons le voir, il suffit d'écrire **nav.(nom de la barre).render()** pour les afficher.

Par la suite, cela étant terminé et fonctionnel, j'ai travaillé sur l'apparence de la page d'accueil et ajouté un footer (un bas de page), très utile pour ajouter des liens de contacte…

Bien qu'un footer soit normalement directement visible en bas de page, j'ai décidé ici de le dynamiser. En effet, celui-ci est dans un premier temps invisible et possède un « bouton » qui lorsqu'il est survolé, fait apparaître le bas de page.



N’ayant pas eu de directives particulières, je me suis permis d’ajouter des crédits, renvoyant vers le LinkedIn d’Alessandro-Umberto ainsi que le mien. À cela s’ajoute la partie contact, qui elle, renvoie vers le site du master PRNT et leurs formulaires de contact sur leur site web.

Pour finir, la barre de navigation a été ajouté à toutes les pages, celles concernant les feuilles de calculs, mais aussi les bases de données.

Conclusion

Bilan sur le projet

Pour conclure sur le projet, malgré les nombreuses tâches réalisées, celui-ci n'est pas totalement opérationnel. En effet, Alessandro-Umberto étant en stage de master durant ce projet, nous n'avons pas pu investir le maximum des capacités et certaines tâches ne sont pas terminées. Comme dit précédemment, nous avons mis au point qu'une seule méthode de calcul, il en manque donc huit. De plus, étant donné le nombre de programmes différents, il serait judicieux par la suite de remplacer la barre de navigation par un menu déroulant, rendant la page plus clair et moins surchargé par les boutons de navigations. À cela s'ajoute, l'hébergement du site web n'ayant pas encore été déterminer. En effet, pour l'instant plusieurs méthodes ont été proposées, mais pas encore déterminées : hébergement sur les serveurs de la Faculté, hébergement en ligne avec l'hébergeur LWS.

Pour finir, en fin de dernière semaine de stage, nous avons réalisé une visioconférence composée de monsieur BERGE-LEFRANC David (tuteur de stage et directeur du secteur), Duje Buric (enseignant chercheur) et Alessandro-Umberto. Dans cette entrevu, nous avons présenté le projet et discuter des potentielles évolutions concernant la cybersécurité. En effet, ce projet ayant pour but d'être accessible à tous les chercheurs, nous avons déterminé qu'il serait judicieux de créer des sessions utilisateurs et administrateurs permettant de protéger l'accès à la suppression de données des différentes tables.

Pour conclure, malgré les évolutions du projet possible, une des méthodes de calcules est fonctionnelle et utilisable. De plus, les base de données avec les différentes requêtes d'insertion et d'affichage étant fonctionnel, il serait rapide de les adapter aux dernières feuilles de calculs.

Bilan personnel

Durant ce stage, j'ai pu mettre en pratique les différentes théories et pratique acquise durant ma première année de BTS SIO. Tout d'abord, une des premières difficultés rencontré fu le langage de programmation python. En effet, n'ayant pas pratiqué ce langage durant mon année scolaire, ce projet fu un vrai challenge pour prouver mon adaptabilité à programmer dans de nouveaux langages sans aide de professionnel, en autodidacte. De ce fait, j'ai pu acquérir de nouvelles compétences en python et dans plusieurs nouveaux logiciels de programmation qui me seront utiles dans l'avenir. De plus, ce projet m'a permis d'acquérir de nouvelles compétences en CSS et HTML avec la mise en place d'une barre de navigation et d'un footer dynamique, non vue durant l'année. De plus, une autre difficulté fu concernant le matériel. En effet, étant à distance, j'ai dû utiliser mon ordinateur personnel et installer tous les logiciels nécessaires au bon fonctionnement du site web et ainsi apprendre à les utiliser.

Pour finir, ces six semaines m'ont permis de me rendre compte du travail d'un professionnel. En effet, le projet étant en binôme, cela m'a permis d'acquérir des compétences de travail en équipe mettant en place une répartition des tâches, permettant de mener à bien un projet dans un temps imparti.

BTS SERVICES INFORMATIQUES AUX ORGANISATIONS **SESSION 2021**

ANNEXE 8 : Modèle d'attestation de stage (recto)

LOGO DE L'ORGANISME D'ACCUEIL

ATTESTATION DE STAGE

à remettre à la ou au stagiaire à l'issue du stage

ORGANISME D'ACCUEIL

Nom ou dénomination sociale : Service chimie physique - Prévention des Risques et Nuisances Technologiques

Adresse : Faculté de Pharmacie, 27 boulevard Jean Moulin 13005 MARSEILLE

☏ : ……

Certifie que

LA OU LE STAGIAIRE

Nom : Gauvin…….. Prénom : Florian

Né.e le : 20 / 03 / 2000 Sexe : F ☐ M ☑

Adresse : Chemin de ceinture, lot "Les Jardins de Louisette" n°3, 13720 La Bouilladisse

☏ : 07.61.21.75.01 Mél : gauvin.florian13720@gmail.com

ÉTUDIANT(E) EN BTS Services informatiques aux organisations
option ☐ SISR ☑ SLAM

AU SEIN DE (nom de l'établissement d'enseignement supérieur ou de l'organisme de formation) :

Lycée Marie-Curie Marseille

a effectué un stage prévu dans le cadre de ses études

DURÉE DU STAGE

Dates de début et de fin du stage :
Du 25 / 05 / 2021 **au** 02 / 07 / 2021 .

Représentant une **durée totale** de …6 semaines nombre de semaines / ~~de mois~~ (*rayer la mention inutile).*

La durée totale du stage est appréciée en tenant compte de la présence effective de la ou du stagiaire dans l'organisme, sous réserve des droits à congés et autorisations d'absence prévus à l'article L.124-13 du code de l'éducation (art. L.124-18 du code de l'éducation). Chaque période au moins égale à 7 heures de présence consécutives ou non est considérée comme équivalente à un jour de stage et chaque période au moins égale à 22 jours de présence consécutifs ou non est considérée comme équivalente à un mois.

MONTANT DE LA GRATIFICATION VERSÉE Á LA OU AU STAGIAIRE

La ou le stagiaire a perçu une gratification de stage pour un **montant total** de …….. €

ANNEXE 8 : Modèle d'attestation de stage (verso)

La tutrice ou le tuteur de l'organisation d'accueil certifie que les situations professionnelles, vécues ou observées, présentées par la ou le stagiaire dans son portefeuille de compétences professionnelles listées ci-dessous ont bien été réalisées dans le cadre de son stage.

☒ **OUI** ☐ **NON**

Intitulé de la situation professionnelle	Activité(s) du référentiel concernée(s)
Productions relatives à la mise en place d'un dispositif de veille technologique et à l'étude d'une technologie, d'un composant, d'un outil ou d'une méthode	A1.1.3 , Étude des exigences liées à la qualité attendue d'un service
	A1.2.4 , Détermination des tests nécessaires à la validation d'un service
	A1.3.3 , Accompagnement de la mise en place d'un nouveau service
	A1.3.4 , Déploiement d'un service
	A1.4.1 , Participation à un projet
	A2.1.1 , Accompagnement des utilisateurs dans la prise en main d'un service
	A4.1.3 , Conception ou adaptation d'une base de données
	A4.1.6 , Gestion d'environnements de développement et de test
	A5.1.2 , Recueil d'informations sur une configuration et ses éléments
	A5.2.2 , Veille technologique

L'attestation de stage *est indispensable pour pouvoir, sous réserve du versement d'une cotisation, faire prendre en compte le stage dans les droits à retraite. La législation sur les retraites (loi n°2014-40 du 20 janvier 2014) ouvre aux étudiants* ***dont le stage a été gratifié*** *la possibilité de faire*

FAIT À …La Bouilladisse…….. **LE** 02/07/2021……

*valider celui-ci dans la **limite de deux trimestres**, sous réserve du versement d'une cotisation. **La demande est à faire par l'étudiant(e) dans les deux années** suivant la fin du stage et sur **présentation obligatoire de l'attestation de stage** mentionnant la durée totale du stage et le montant total de la gratification perçue. Les informations précises sur la cotisation à verser et sur la procédure à suivre sont à demander auprès de la Sécurité sociale (code de la Sécurité sociale art. L.351-17 – code de l'éducation art..D.124-9).*

Nom, fonction et signature du représentant de l'organisme d'accueil

Pr. David BERGÉ-LEFRANC, chef de service