LES TROPHÉES NSI

Édition 2023

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET

Ce document est l'un des livrables à fournir lors du dépôt de votre projet : 4 pages maximum (hors documentation).

Pour accéder à la liste complète des éléments à fournir, consultez la page **Préparer votre participation**.

Vous avez des questions sur le concours ? Vous souhaitez des informations complémentaires pour déposer un projet ? Contactez-nous à **info@trophees-nsi.fr**.

NOM DU PROJET : FaisTonFrançais.fr

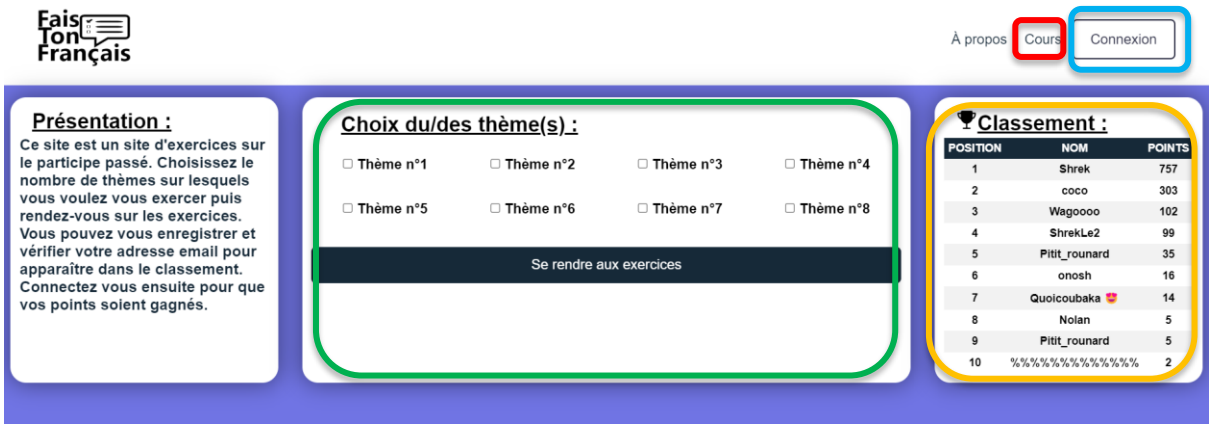
> PRÉSENTATION GÉNÉRALE :

FaisTonFrançais.fr est une application web qui aide les élèves de seconde et de première à réviser la grammaire et la conjugaison pour le Bac de français. C'est un site ludique et compétitif qui transforme les révisions en un jeu.

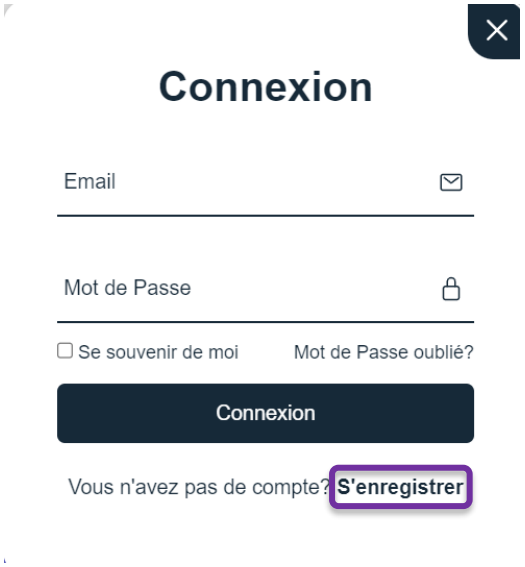
L'idée de ce site vient de l'association inattendue de deux matières : Français et NSI. Ainsi le travail en collaboration d'élèves de seconde en Français et d'élèves de terminale en NSI.

Le site comprend un espace avec les cours classés par thèmes, des exercices générés en fonctions des thèmes choisis et un classement entre tous les joueurs ayant créé un compte. Les élèves pourront faire des exercices sans se connecter, ils n'apparaitront simplement pas dans le classement.

Cette application Web a été développée en Python avec le framework Flask.



- Le lien vers les leçons de chaque thème
- Le joueur peut choisir sur quel(s) thème(s) il veut s'entrainer
- Le classement montre les 10 joueurs ayant le plus de points (les noms ici sont ceux de nos amis béta testeurs)
- Le bouton de connexion ouvre une fenêtre où le joueur peut se connecter avec son email et son mot de passe :



- Si le joueur n'a pas encore de compte, il peut s'enregistre ici

> ORGANISATION DU TRAVAIL :

L'équipe est composée de quatre membres : Hugo, Nolan, Yanis et Baptiste.

Durant toute la durée du projet, nous nous sommes divisés en deux groupes. Le premier composé de Yanis et de Baptiste était chargé de créer les différentes bases de données, de créer les liens entre elles et de coder les fonctions en python qui seraient utilisées par le second groupe. Celui-ci composé de Hugo et Nolan avait pour tâche de créer toute l'architecture de l'application web, les liens entre les pages et l'espace utilisateur.

Voici les tâches de chacun :

- Hugo :
  - Création des leçons des différents thèmes avec l'aide d'une professeure de français.
  - Création des visuels du site web via les langages css et html.
- Nolan :
  - Mise en place des liens entre les pages du site.
  - Gestion de l'envoi des mails.
  - Gestion de la connexion des utilisateurs et de leur enregistrement dans la base de données.
- Yanis :
  - Écriture de la fonction en langage Python et SQL permettant de créer la base de données des exercices à partir des fichiers texte (.txt) fournis par les élèves de seconde.
  - Écriture des fonctions Python permettant de hasher et des sécuriser les mots de passe des utilisateurs.
- Baptiste :
  - Écriture des fonctions permettant d'utiliser sur le site les exercices de la base de données.
  - Création de la base de données des comptes des utilisateurs.

Nous nous retrouvions 6 heures par semaine lors de nos cours de NSI et travaillions quelque fois individuellement depuis chez nous.

Nous avons d'abord codé en local sur l'éditeur VSCodium puis avons mis le site en ligne grâce à pythonanywhere. Les langages informatiques que nous avons utilisés sont Python, SQL, html, css et Javascript. Nous avons eu besoin des modules flask, flask_mail, flask_session, SQLAlchemy et bcrypt.

> LES ÉTAPES DU PROJET :

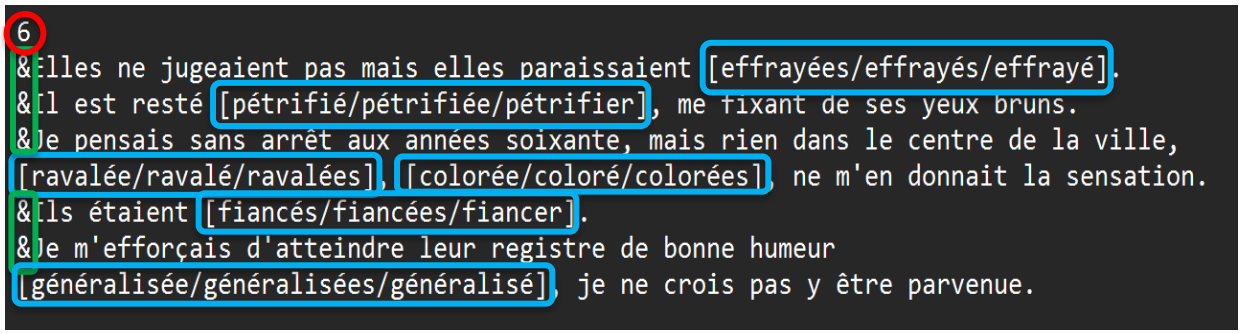
L'idée d'un logiciel de révisions de français est venue de nos professeurs de français et de NSI qui nous ont proposé de participer à ce projet collaboratif auquel nous avons tous les quatre choisis de participer.

Les exercices créés par les élèves de seconde sont des exercices simples de types QCM à partir des phrases du livre « L'Évènement » de Annie Ernaux.

Nous avons d'abord dû passer par une étape préalable avec les secondes afin de leur montrer comment ils devaient écrire leurs fichiers d'exercices pour que nous puissions les traiter correctement. Les fichiers devaient être sous la forme suivante :

```
6
&Elles ne jugeaient pas mais elles paraissaient [effrayées/effrayés/effrayé].
&Il est resté [pétrifié/pétrifiée/pétrifier], me fixant de ses yeux bruns.
&Je pensais sans arrêt aux années soixante, mais rien dans le centre de la ville,
[ravalée/ravalé/ravalées], [colorée/coloré/colorées], ne m'en donnait la sensation.
&Ils étaient [fiancés/fiancées/fiancer].
&Je m'efforçais d'atteindre leur registre de bonne humeur
[généralisée/généralisées/généralisé], je ne crois pas y être parvenue.
```

Trois choses dans l'écriture de ces fichiers nous permettaient de les traiter.



- Le chiffre sur la première ligne indique le thème qui correspond aux questions.
- Le caractère « & » indique le début d'une nouvelle question et permet ainsi de séparer toutes les questions.
- Les propositions de réponse doivent être entre crochets et séparées par de slash. La bonne réponse est placée en premier.

Une fois que nous avions récupéré tous les fichiers faits par les élèves de seconde, nous avons créé et alimenter la base de données avec les exercices. Avant cela, nous avons mené une petite enquête parmi les élèves de seconde pour savoir sous quelle forme ils préfèreraient travailler. Pour la plupart des élèves, avoir des séries de 5 questions était la meilleure chose, c'est donc sur cette forme que nous nous sommes basés pour commencer l'architecture du site.

À partir de ce moment, nous nous sommes séparés en 2 groupes. Hugo et Nolan ont créé le site conformément aux retours des élèves de secondes grâce aux langages Python (avec Flask), html, Javascript et css et Yanis et Baptiste ont créé les bases de données ainsi que les fonctions python qui font le lien entre celles-ci et le site web.

Finalement, nous avons réunis tous nos travaux pour atteindre le rendu final qui, après quelques ajustements, est fonctionnel.

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

Dans la version finale, les joueurs peuvent se connecter avec leur adresse mail et faire les exercices. De plus le système de points et de correction des exercices est fonctionnel. Il reste encore à rendre fonctionnel les boutons « mot de passe oublié » et « supprimer mon compte ».

Nous souhaitons également revoir le système de score pour qu'il prenne en compte le nombre de parties jouées et le pourcentage d'erreur.

Afin de vérifier notre site, nous avons donné le lien à plusieurs de nos amis qui ont rempli le rôle de béta testeurs. Ils nous ont signalé certains bugs dont un qui leur permettait, en actualisant plusieurs fois la page, d'obtenir des points sans faire d'exercice. Nous avons résolu ce problème en créant une nouvelle variable de vérification.

> OUVERTURE :

Une amélioration que nous voudrions faire serait de pouvoir faire des scores et des classements par thème pour que les joueurs puissent voir quel sont leurs points forts et leurs points faibles.

Pour diffuser le projet et atteindre le plus d'élèves possible nous voulons faire une démonstration aux élèves de seconde et de première pour qu'ils utilisent le site dans le cadre de leurs révisions.

Ce projet nous a permis à tous les quatre d'améliorer nos compétences et nous en gardons tous une très bonne expérience

Hugo :

- J'ai bien aimé faire ce projet, j'ai pu pleinement découvrir le langage jinja qui permet de faire le lien entre python et le html. De plus, je me suis familiarisé avec le html et le CSS. Je pense qu'on pourrait rajouter un créateur d'exercices directement intégré au site pour l'améliorer. Je suis content de voir qu'on a réussi à faire ce qu'on avait prévu de faire au début avec en bonus plusieurs améliorations.

Nolan :

- J'ai adoré faire notre projet final. J'en ai plus appris sur le lien entre Flask et HTML grâce aux modules ou encore Jinja. De plus, j'ai expertisé ces multiples langages, ce qui pourrait m'aider dans le futur. J'aurais aimé avoir plus de temps pour finaliser l'application. Par exemple, on ne peut pas confirmer une adresse email après le délai dépassé ou alors, on ne voit pas ses fausses réponses lors de la correction. Malgré quelques regrets, je trouve que le site est complet, joli et fonctionnel. Je suis heureux pour le groupe qu'on ait réussi à faire ce qui était prévu, en plus de quelques améliorations.

Yanis :

- J'ai adoré faire le projet, j'ai pu consolider mes connaissances et en apprendre des nouvelles. Le fait de gérer des données m'as permis de me familiariser avec la conception des bases de données mais aussi sur les requêtes. On aura pu améliorer la coordination en groupe et se répartir les tâches correctement pour mener à bien le projet

Baptiste :

- Le projet était vraiment enrichissant, j'ai pu mieux me familiariser avec les langages web que je maîtrisais peu et améliorer mes compétences en SQL. S'il fallait recommencer, je pense que répartir les tâches plus équitablement serait mieux. Mais malgré ça, ce projet est pour moi un accomplissement.

DOCUMENTATION

Instructions pour mise en marche du site web : Dans le terminal :

- Activer l'environnement virtuel grâce à la commande suivante : 'venv\Scripts\activate' (windows) et 'venv/bin/activate' pour (linux)
- Lancer l'application grâce à la commande suivante : flask run

Nous avons utilisé les langages suivants :

- Python
- SQL
- HTML
- CSS
- JavaScript

Nous avons utilisé les modules suivants dans l'environnement virtuel :

- Flask
- Flask-Mail
- Flask-Session
- Flask-SQLAlchemy
- SQLAlchemy
- bcrypt

(Si l'environnement virtuel ne se lance pas, n'hésitez pas à le recréer et à y réinstaller ces modules.)

Nous avons utilisé VS Code chez nous et VS Codium au lycée. Toutes les données sont stockées dans des tables SQL et les données temporaires sont stockées dans un cookie. Un seul cookie est présent mais nécessaire pour le fonctionnement du site.

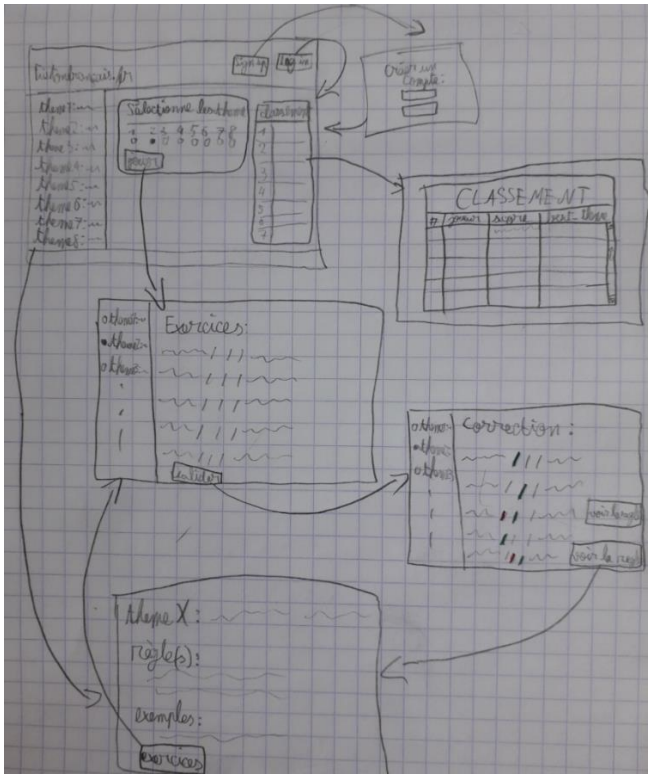
Pour apparaître dans le classement du site, il faut être confirmé. Or ce n'est pas possible en local car le lien de confirmation renvoi sur un site inexistant, donc 3 adresses email ont été créées et confirmées de base pour faire apparaître un classement.

Voici un aperçu de se que nous avons pu faire :

Hugo :

Je me suis occupé de la confection de la page du cours, de la page à propos.

Dans un premier temps, en groupe nous avons déterminé à quoi ressemblerai le site et nous avons opté pour ça :



Par la suite, j'ai créé la page cours en HTML où je me suis inspiré du cours que la professeure de français nous avait donné.

```
</header>
<main>
    <h1>Orthographe : participe passé employé dans un GN ou dans un verbe composé</h1><br>

    <h2>Orthographe des participes passés : ne pas confondre les lettres du radical et les accords</h2><br>

    <h3>1. é/ée/és/ées/er ?</h3><br>

        <li>il a chanté ou il a chanter ? → remplacer par vendu (participe passé) ou vendre (infinitif) </li>

        <p>Réponse : il a vendu → donc c'est un participe passé ; donc on écrit : il a chanté </p>

            <li>é ou ée : voyez si le nom auquel il se rapporte est féminin</li>

            <li>é ou és : voyez si le nom auquel il se rapporte est singulier ou pluriel</li>  <br>
    <h3>2. i/ie/ies/is/it ?</h3><br>

       <li>Pour trouver les lettres du radical (les lettres qui appartiennent au mot) : mettre le mot au féminin </li>

        <p>Ex : il a fini ? finit ? fini ? finis ? → une chose <b>finie</b> (oui)/ finite (non)/ finise (non plus)

        <br>Ex : il a acqui ? acquis ? acquit ? → une moto acquie ? (non)/ <b>acquise</b> ? (oui)/ acquie (non) </p>

            <li>i ou ie : voyez si le nom auquel il se rapporte est féminin</li>

            <li>i ou is : voyez si le nom auquel il se rapporte est singulier ou pluriel</li>



            <li>CONFUSION possible avec des verbes conjugués : </li>
```

Ligne permettant de faire le lien entre le HTML et le CSS :

```
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/theme.css') }}">
```

J'ai ensuite fait la mise en page en respectant le désigne que nous avions choisi au début avec le langage CSS.

```css
h1{
    text-align: center;
    font-size: 2.3em;
}

h2 {
    font-size: 1.5em;
}

h3 {
    font-size: 1.3em;
    margin-left: 1.5em;
}

p{
    margin-left: 3em;
}

li{
    margin-left: 4em;
}
```

```css
table,  th, td {

    border: 2px solid rgb(0, 0, 0);
    border-collapse: collapse;
    }

th, td {

    padding: 8px;
    vertical-align: center;
    }


th {

    background-color: rgb(155,155,155);
    font-weight: bold;
  }

td {

    background-color: white;

    }
```

J'ai ensuite fait la même chose avec la page à propos du site :

```html
<main>
    <div class="apropos">
        <p>FaisTonFrançais.fr est un site web qui aide les élèves de seconde est de première à réviser la grammaire et la conjugaison pour le Bac de français.</p>
        <p>C'est un site ludique et compétitif qui transforme les révisions en un jeu.</p>
        <p>L'idée de ce site vient de l'association inattendue de deux matières : Français et NSI.</p>
        <p>Ainsi le travaille en collaboration d'élèves de seconde en Français et d'élèves de terminale en NSI.</p>
        <p>À tes exercices, fait ton français !!</p>
    </div>
```

Nolan :

Je me suis occupé de la relation entre python et html et de la conception des pages web.

Dans cette capture d'écran, on peut voir les différentes utilisations des modules.
La première partie sert à créer un cookie temporaire pour enregistrer des données au fur et à mesure de la navigation.
La deuxième partie sert à envoyer des mails pour les mails de confirmation.
La dernière sert à sécuriser les cookies.

```python
"""Flask-SQLAlchemy"""
app.config['SECRET_KEY'] = [redacted]
app.config["SQLALCHEMY_DATABASE_URI"] = 'sqlite:///session.db'
app.config['SESSION_TYPE'] = 'sqlalchemy'
db = SQLAlchemy(app)
app.config['SESSION_SQLALCHEMY'] = db
sess = Session(app)
app.app_context().push()
#db.create_all()

"""Flask-Mail"""
app.config['MAIL_SERVER'] = 'smtp.gmail.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USE_SSL'] = False
app.config['MAIL_USERNAME'] = 'faistonfrancais@gmail.com'
app.config['MAIL_PASSWORD'] = [redacted]
app.config['MAIL_DEBUG'] = False
app.config['MAIL_SUPPRESS_SEND'] = False
app.config['MAIL_DEFAULT_SENDER'] = ('FaisTonFrançais.fr', 'faistonfrancais@gmail.com')
app.config['MAIL_MAX_EMAILS'] = None
app.config['MAIL_ASCII_ATTACHMENTS'] = False
mail = Mail(app)

"""Flask-urlsafe-serializer"""
s = URLSafeSerializer(app.config['SECRET_KEY'])
```

Voici un exemple de route par laquelle l'utilisateur passe. Ici c'est le route principale, l'accueil du site. session['blabla'] est le cookie de l'utilisateur. On peut récupérer les données envoyées par l'utilisateur dans le request.form, ici on récupère les thèmes pour renvoyer l'utilisateur vers la page d'exercice avec des questions de ces thèmes.

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    session['top10'] = ten_first()
    if request.method == 'POST':
        if 'btn_thèmes' in request.form:
            session['questions'] = []
            session['reponses'] = []
            session['indice_questions'] = 0
            session['scores'] = []
            session['q_selected'] = []
            option1_value = request.form.getlist('option')
            session['themes'] = option1_value
            rep = get_q(5, session['themes'])
            session['questions'].append(rep[0])
            session['reponses'].append(rep[1])
            session['q_selected'].append([e for e in session['reponses'][session['indice_questions']]])
            return render_template('exercices.html')
    else:
        if 'nom_utilisateur' in session:
            return render_template('index.html')
    return render_template('index.html')
```

Voici comment les questions sont affichées. On voit bien que la structure session['blabla'] est réutilisée car c'est le cookie qui peut être utilisé partout dans l'application.

Ici on parcourt le nombre de question et on afficher successivement la début, les propositions (chacune cliquables) et la fin, tout ça dans un formulaire avec un bouton de renvoie de réponses.

```
<div class="exercices">
    <h2>À toi de jouer :</h2>
    {% if session['questions'][session['indice_questions']] is not none %}
    {% if session['questions'][session['indice_questions']]|length == 0 %}
        <p>Il n'y a pas de question</p>
    {% else %}
    <form action="{{ url_for('exercices') }}" method="POST">
        {% for i in range(session['questions'][session['indice_questions']]|length) %}
            <p>{{ session['questions'][session['indice_questions']][i][0] }}
            <label><input type="radio" name="q{{ i }}" value="0"><span>{{ session['questions'][session['indice_questions']][i][1][0] }}</span></label>
            / <label><input type="radio" name="q{{ i }}" value="1"><span>{{ session['questions'][session['indice_questions']][i][1][1] }}</span></label>
            / <label><input type="radio" name="q{{ i }}" value="2"><span>{{ session['questions'][session['indice_questions']][i][1][2] }}</span></label>
            {{ session['questions'][session['indice_questions']][i][2] }}</p>
        {% endfor %}
        <button type="submit" class="btn" name="bouton_correction">Envoyer mes réponses</button>
    </form>
    {% endif %}
    {% endif %}
</div>
```

Enfin voici comment est créé la page de connexion côté html, il contient évidement un formulaire avec des cases à remplir, toutes obligatoires. Le css associé mets en valeur la balise div pour donner un beau résultat. Côté javascript, quand le bouton connexion en haut à droite du site est cliqué, la classe de la div change et le css également, ce qui permet de faire apparaître la page de connexion, c'est la même chose pour la page 's'enregistrer'.

Yanis :

Je me suis occupé de la création de la base de données et le hashage des mots de passe. Dans cette capture d'écran on peut voir trois tables de notre base de données.
La création de la base de données a dû être refaite plusieurs fois car à chaque fois il manquait des attributs pour en faire des tables complète pour faciliter le travail de Baptiste qui allait traiter toutes les données.

```
cur.execute("""CREATE TABLE Question (
    id_q INTEGER PRIMARY KEY AUTOINCREMENT,
    id_theme TEXT,
    debut TEXT,
    reponse LIST,
    fin TEXT);
    """)
cur.execute("""CREATE TABLE Theme(
    id_theme INTEGER PRIMARY KEY,
    nom TEXT);
    """)
cur.execute("""CREATE TABLE Compte(
    id_compte INTEGER PRIMARY KEY AUTOINCREMENT,
    nom_utilisateur NVARCHAR(20),
    email NVARCHAR(255),
    mdp TEXT,
    score INTEGER,
    nb_parties INTEGER,
    confirm BOOLEAN);
    """)
```

Pour pouvoir remplir la base de données nous avions les fichiers de texte des élèves de secondes donc il suffisait de les traiter pour remplir la base de données. La fonction parcours_répertoire a permis de récupérer tous les fichiers qui étaient chacun contenu dans un dossier. De plus cette fonction permet de faire le lien avec obtenir_questions.

```
def parcours_repertoire(repertoire):
    """Parcourt tout le répertoire pour trouver les fichiers puis récuupère les questions grâce à obtenir_questions"""
    for nom in os.listdir(repertoire):
        chemin = os.path.join(repertoire, nom)
        if os.path.isdir(chemin):
            parcours_repertoire(chemin)
        elif nom.endswith('.txt'):
            chemin_rel = os.path.relpath(chemin)
            obtenir_questions(chemin_rel)
```

Grâce à la fonction obtenir_questions, tous les fichiers étaient traités de la manière suivante :

- On récupère le thème étant un numéro qui est la première ligne de chaque fichier texte ce qui nous donne le theme_id

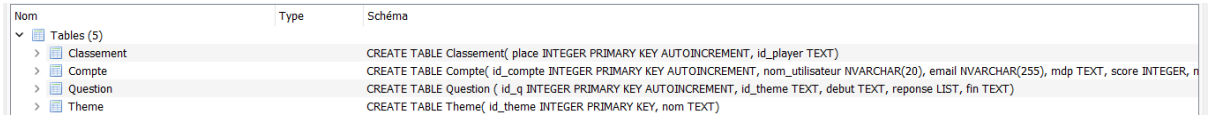
- On cherche une parenthèse ouvrante cela veut dire qu'il y a les propositions et donc à partir de ça on peut trouver la fin et le début et les propositions

Une fois cette recherche effectuer pour la question il manque plus qu'à la stocker dans le tableau de dictionnaire tab_questions

```
def obtenir_questions(fichier):
    '''Récupère les questions du fichier passé en paramètre'''
    f = open(f'{fichier}', 'r', encoding='utf8')
    c = f.read()
    f.close()
    theme_id = c[0]
    c = c.replace('\n', '')
    t = c.split('&')
    for e in t[1:len(t)]:
        compteur = 0
        for carac in e:
            if carac == "[":
                compteur += 1
        if compteur == 1:
            debut = e.split('[')[0]
            propositions = e.split('[')[1].split(']')[0].split('/')
            fin = e.split('[')[1].split(']')[1]
            tab_questions.append({"theme_id": theme_id,'début': debut, 'propositions': propositions, 'fin': fin})
```

Une fois notre tableau de dictionnaire avec toutes nos questions initialisé, toutes les questions peuvent être ajoutées en base de données grâce à la fonction traiter_tab.
Une mauvaise conception au départ de cette fonction nous a valu un problème qui est que les réponses étaient inversées avec la fin dans la base de données.

```
def traiter_tab(tab):
    """Ajoute les questions de tab dans la base de données"""
    for dict in tab:
        conn = sqlite3.connect('francais.db')
        cur = conn.cursor()
        cur.execute("INSERT INTO Question(id_theme,debut,fin,reponse) VALUES(?,?,?,?)",(dict["theme_id"]
                    ,dict["début"],dict["fin"],str(dict["propositions"])))
        conn.commit()
        conn.close()
```



Nom	Type	Schéma
Tables (5)		
Classement		CREATE TABLE Classement(place INTEGER PRIMARY KEY AUTOINCREMENT, id_player TEXT)
Compte		CREATE TABLE Compte(id_compte INTEGER PRIMARY KEY AUTOINCREMENT, nom_utilisateur NVARCHAR(20), email NVARCHAR(255), mdp TEXT, score INTEGER, n
Question		CREATE TABLE Question (id_q INTEGER PRIMARY KEY AUTOINCREMENT, id_theme TEXT, debut TEXT, reponse LIST, fin TEXT)
Theme		CREATE TABLE Theme(id_theme INTEGER PRIMARY KEY, nom TEXT)

id_q	id_theme	debut	reponse	fin
Filtre	Filtre	Filtre	Filtre	Filtre
1	8	J'ai	['pris', 'prit', 'prie']	le boulevard de Magenta et reconnu le magasin ...
2	8	J'ai tendu la main mais elle ne me l'a pas	['donnée', 'donné', 'donner']	.
3	8	La salle d'attente est	['séparée', 'séparer', 'séparé']	en deux boxes contigus.
4	8	J'ai	['suivi', 'suivie', 'suivit']	le long couloir voûté du pavillon Elisa.
5	8	J'ai commencé à corriger les copies que j'avais	['emportées', 'emporté', 'emporter']	.
6	5	La docteure a	['appelé', 'appeler', 'appelée']	mon numéro.
7	5	Je me suis rendu compte que j'avais	['vécu', 'vécus', 'vécue']	ce moment à Lariboisiére de la même façon que...
8	5	J'ai	['descendu', 'descendue', 'descendues']	l'escalier a toute vitesse.
9	5	J'ai	['commencé', 'commencée', 'commencer']	à corriger les copies que j'avais emporté.
10	5	J'ai	['poussé', 'poussée', 'pousser']	la porte 15 et monté les deux étages.
11	6	Elles ne jugeaient pas mais elles paraissaient	['effrayées', 'éffrayés', 'effrayé']	.
12	6	Il est resté	['pétrifié', 'pétrifiée', 'petrifier']	, me fixant de ses yeux bruns.
13	6	Ils étaient	['fiancés', 'fiancées', 'fiancer']	et ne dormaient pas ensemble.
14	6	Je m'efforcais d'atteindre leur registre de bonne ...	['généralisée', 'généralisées', 'généralisé']	,je ne crois pas y être parvenue.
15	3	Impossible de dire pourquoi je m'étais	['rabattue', 'rabattu', 'rabattus']	dans ce beau quartier.
16	3	L'enlacement et la gesticulation de nos corps	['nus', 'nues', 'nue']	me paraissaient une danse de la mort.
17	3	Ils attendaient déjà, assis loin les un des autres, ...	['vêtu', 'vêtus', 'vêtue']	mode et calvitie légère, un jeune homme avec u...
18	3	Comme si j'étais	['retenue', 'retenu', 'retenues']	par quelque chose de très ancien.

Pour la fonction de hashage de mot de passe et de vérification des mots de passes, j'ai utilisé le module Bcrypt qui permet d'ajouter un sel au mot de passe de l'utilisateur puis de le hasher. Pour la vérification il faut hasher le mot de passe rentrer et Bcrypt va s'occuper de le comparer à l'aide de sa fonction checkpw

```
def hash_mdp(mdp):
    """Hash le mot de passe"""
    byte = mdp.encode('utf-8')
    salt = bcrypt.gensalt()
    hashed = bcrypt.hashpw(byte, salt)
    return hashed

def verif_mdp(mdp,hashed):
    """Décrypte et renvoie le résultat"""
    userBytes = mdp.encode('utf-8')
    result = bcrypt.checkpw(userBytes, hashed)
    return result
```

Baptiste :

Nous avons utilisé un fichier nommé « modules.py » dans lequel se trouvaient toutes les fonctions pythons qui faisaient le lien entre la base de données et le site web.
Dans ce fichier j'ai codé plusieurs fonctions comme les fonctions « get_q_theme » et « get_q » qui permettent de renvoyer 5 questions des thèmes passés en paramètres.

```python
def get_q_theme(id_theme):
    conn = sqlite3.connect('francais.db')
    cur = conn.cursor()
    res = cur.execute("""SELECT debut,reponse,fin FROM Question WHERE id_theme = ?""",(id_theme,))
    tab = res.fetchall()
    conn.close()
    #print(tab)
    return tab

def get_q(k,id_theme):
    T = []
    for e in id_theme:
        T = T + get_q_theme(e)
    t = []
    for i in range(k):
        a = choice(T)
        T.remove(a)
        t.append(a)
    tableau = []
    for e in t:
        tableau.append(modif(e))
    return tableau
```

La fonction « get_q » créer d'abord un tableau contenant toutes les questions des thèmes choisis grâce à la fonction « get_q_theme ». Puis elle choisit k questions de ce tableau, k valant toujours 5. Enfin elle réécrit correctement les questions choisies grâce à la fonctions « modif » et renvoie les questions modifiées dans un tableau.

J'ai également travaillé sur la modification de la base de données via python, notamment sur les fonction permettant de créer un nouveau compte avec une adresse email, un nom d'utilisateur et un mot de passe et d'augmenter le score des joueurs :

```python
def creer_compte(identifiant,email,mdp):
    conn = sqlite3.connect('francais.db')
    cur = conn.cursor()
    cur.execute("INSERT INTO Compte(nom_utilisateur,email,mdp,score,nb_parties) VALUES(?,?,?,?,?)",(identifiant,email,mdp,0,0))
    conn.commit()
    conn.close()

def ajouter_score(score,id_compte):
    conn = sqlite3.connect('francais.db')
    cur = conn.cursor()
    cur.execute("UPDATE Compte SET score = score + ? AND nb_parties = nb_parties + 1 WHERE id_compte = ?",(score,id_compte))
    conn.commit()
    conn.close()
```

Chacune de ces fonctions utilise le module sqlite3 pour exécuter des commandes SQL Dans le code python.