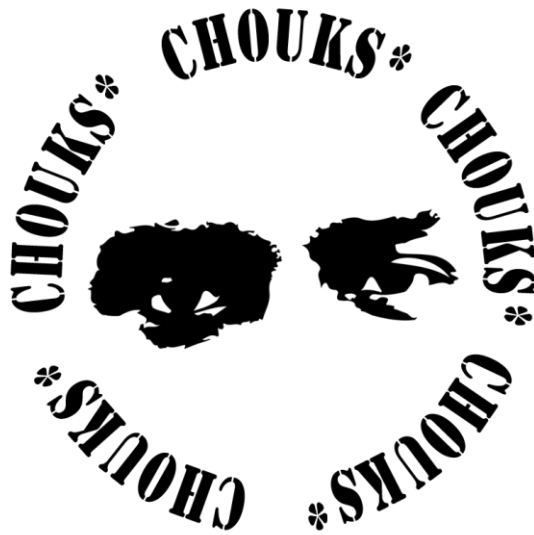


Projet de bases de données Magazine people



Choukar Ayman, Bentounsi Touhami-Alexandre

1 – Présentation et manuel utilisateur



The screenshot shows the homepage of 'CHOUKS MAGAZINE'. At the top, the title 'CHOUKS MAGAZINE' is displayed in large, bold, black letters. To its right is a circular logo with the word 'CHOUKS' repeated around the perimeter and a central graphic. Below the title, a horizontal navigation menu contains the following links: ACCUEIL, CÉLÉBRITÉS, REVUE, FILM, ALBUM, DÉFILÉ, PHOTO, and ADMIN. A search bar is positioned below the menu, with the placeholder text 'Entrez vos mots-clés pour effectuer une recherche.' and a blue 'Rechercher' button. Below the search bar, the text 'Résultat de la recherche' is shown, followed by the message 'Aucune célébrité n'a été trouvée dans la base de données.'

Chouks Magazine est le site que l'on a créé qui aborde le thème du magazine people afin de retracer les activités de célébrités (vip) appartenant au monde du cinéma, de la mode, ou bien de la musique, il vous permettra de découvrir des informations personnels, connues et les liens qu'ont les célébrités entre elles.

Pour lancer le site il faut mettre cette adresse au lancement :
http://127.0.0.1:5000/accueil

Toutes les pages contiennent le même entête avec le titre du site, le logo, et le nom des différentes pages sur lesquelles il est possible de naviguer :

- **La page accueil** a une barre de recherche en plus qui va permettre de retrouver une célébrité ou un album grâce au mot clé qui sera rentré, il n'est pas obligatoire de taper un nom entier, quelques lettres suffiront.
- **La page célébrités** affiche toutes les célébrités sous forme de carte avec leur nom, afin d'avoir plus de détails sur la célébrité. Il suffit de cliquer sur la carte et il y aura alors le sexe, la nationalité, la date de naissance, les albums, les photos, les films, les défilés, les mariages et liaisons que le VIP a pu avoir durant ça vie, s'il y en a, sinon il sera affiché qu'aucune information n'a été trouvé.
- **La page Revue** affichera toutes les revues en donnant leur numéro et leur date de parution, en cliquant sur une des revues vous aurez tous les articles qui la compose, et en cliquant sur un article vous aurez son titre, la page et la revue dans lequel il est présent, le résumé et la date de parution.

- **La page Film** affichera les films avec leur nom, le réalisateur, en cliquant sur un film vous aurez en plus son visa, la date de réalisation, les acteurs jouant dedans et le réalisateur.
- **La page Album** affiche tous les albums en donnant leur nom et l'artiste qui l'a créé, puis lorsque l'on clique sur un album, il y'a en plus la date de réalisation et le nom de la maison de disque.
- **La page Défilé** affiche les défilés en donnant le lieu et la date de l'événement, et lorsque l'on clique sur un défilé on peut retrouver en plus les mannequins qui y participent et le créateur.
- **La page photo** permet d'afficher toutes les photos de la base de données (pas fonctionnelle car nous avons eu des soucis pour importer les photos dans la base de données).
- **La page Admin** exige une connexion avec un nom d'utilisateur et un mot de passe. C'est une page privée qui permet de gérer toutes les données afin de rajouter ou modifier les informations présentent sur le site tel qu'un vip, les mariages et liaison, les activités (film, album, défilé), les liens entre les artistes et leur activité (acting, mannequinat, musique), les articles et les photos présent sur le site. Elle donne également des statistiques comme le nombre total de célébrités, le nombre d'articles ou bien les photos publiées.
- **Note à l'attention de l'utilisateur pour la page Admin :**

Lors de l'ajout d'élément il faut porter attention à plusieurs points :

- Le format de la date doit être le suivant lors de l'ajout et de la mise à jour : JJ-MM-ANNEES
- Si une mise à jour doit être faites sur les données déjà présente, il faut bien remplir tous les éléments et bien remettre les bonnes informations si elles ne sont pas modifiées.
- Pour accéder à la page voici les logs :
Nom d'utilisateur : admin
Mot de passe : 123

2 – Premières étapes

A – Schéma Relationnel

Vip (id_vip, nom, prénom, sexe, nationalité, date_naissance, date_décès, id_mariage)

Detail_mariage(id_mariage, date_mariage, lieu, date_separation, circonstance, id_vip_1, id_vip_2)

Film (Visa, date_real, titre, id_vip)

Acteur (id_acteur, date_1^{er}_role, id_vip)

Mannequin (id_mannequin, taille, agence, id_vip)

Défilé (id_defile, lieu, date, id_vip)

Artiste (id_artiste, specialite, id_vip)

Album (id_album, titre, date_rea, nommaisondisque, id_vip)

Photo(id_photo, version_numerise, date_photo, nom_photographe, agence_photo, circonstance_photo)

Article (id_article, titre, num_page_article, resume, numero_revue)
Revue (numero_revue, date_parution)

Liaison (id_vip, id_liaison_vip, date_annonce)

Acting (id_acteur, visa)

Mannequinat (id_mannequin, id_defile)

Musique (id_artiste, id_album)

Présent (id_photo, id_vip)

Utilisé (id_photo, id_article)

Apparaît (id_photo, numero_revue, num_page_photo)

FK : Vip(id_mariage) fait référence à Detail_mariage(id_mariage)

Detail_mariage(id_vip_2) fait référence à Vip(id_vip)

Detail_mariage(id_vip_1) fait référence à Vip(id_vip)

Film(id_vip) fait référence à Vip(id_vip)

Acteur(id_vip) fait référence à Vip(id_vip)

Mannequin(id_vip) fait référence à Vip(id_vip)

Défilé(id_vip) fait référence à Vip(id_vip)

Artiste(id_vip) fait référence à Vip(id_vip)

Album(id_vip) fait référence à Vip(id_vip)

Article(numero_revue) fait référence à Revue(numero_revue)

Liaison(id_vip) fait référence à Vip(id_vip)

Liaison(id_liason_vip) fait référence à Vip(id_vip)

Acting(id_acteur) fait référence à Acteur(id_acteur)

Acting(visa) fait référence à Film(visa)

Mannequinat(id_mannequin) fait référence à Mannequin(id_mannequin)

Mannequinat(id_defile) fait référence à défilé(id_defile)

Musique(id_artiste) fait référence à Artiste(id_artiste)

Musique(id_album) fait référence à Album(id_album)

Présent(id_photo) fait référence à Photo(id_photo)

Présent(id_vip) fait référence à Vip(id_vip)

Utilisé(id_photo) fait référence à Photo(id_photo)

Utilisé(id_article) fait référence à Article(id_photo)

Apparaît(id_photo) fait référence à Photo(id_photo)

Dans un premier temps on a eu la première feuille du cahier des charges du « client » qui nous donnait ses envies générales concernant les célébrités et leurs informations personnels qui devaient apparaître, son envie d'avoir les mariages et les liaisons, et aussi rattacher les célébrités à leurs domaines respectifs, à savoir l'acting/réalisation, la musique et le mannequinat.

On a alors créé les tables au fur et à mesure de la lecture de la feuille afin d'ajouter dans les bonnes tables les informations, puis on a créé les associations entre les tables et les cardinalités.

3 – Fonctionnalités du site

A- Pages Principales :

Dans le main, les pages principales (défile, album, film, revue, photo) sont toutes écrites de la même façon :

```
49 @app.route("/film")
50 def film():
51     with db.connect() as conn:
52         with conn.cursor() as cur:
53             cur.execute("SELECT f.visa , CONCAT(v.prenom, ' ', v.nom) AS realisateur, f.titre , f.date_real , f.id_vip FROM film f JOIN vip v ON f.id_vip = v.id_vip;")
54             lst_film=cur.fetchall()
55         conn.close()
56     return render_template("film.html", Lst_film=lst_film)
57
```

« @app.route(« / nom de la page internet) » afin d'associer la page a la fonction.

« def nom de la fonction () » afin de déclarer la fonction.

« with db.connect » afin de se connecter a la base de données puis de l'arrêter après l'exécution (grâce au with).

« with conn.cursor » afin de pouvoir exécuter les requêtes et arrêter la connexion apres l'exécution (grace au with).

« cur.execute » va exécuter la requête.

« lst_film » = cur.fetchall() » récupère les résultats de la requête SQL sous forme d'une liste de tuples et les stocke dans la variable que l'on a donné.

« conn.close » permet de s'assurer que la connexion avec la base de données est bien terminée.

« return render_template("film.html", lst_film=lst_film) » afin de retourner la page html et utilise la variable que l'on a donné pour afficher les données
Dans les fichiers html de ces pages, l'affichage sera le même

```

<div class="container">
  <div class="row g-4">
    {% if lst_film %}
      {% for film in lst_film %}
        <div class="col-sm-6 col-md-4 col-lg-3">
          <a href="{{ url_for('detailsfilm', visa=film[0]) }}" class="text-decoration-none">
            <div class="card card-celeb">
              
              <div class="celeb-name">
                <h5> Titre : {{ film[2] }} </h5>
                <p><strong> Realisateur : </strong> {{ film[1] }} </p>
              </div>
            </div>
          </a>
        </div>
      {% endfor %}
    {% else %}
      <div class="col-12 text-center">
        <p>Aucun film n'a été trouvée dans la base de données.</p>
      </div>
    {% endif %}
  </div>
</div>

```

On vérifie d'abord que la variable que l'on avait donnée dans le main n'est pas vide, si elle ne l'est pas alors on va afficher tout ce qu'on y trouve. Sinon on affichera que rien n'a été trouvé dans la base de données.

On utilise ensuite nos class de style pour gérer les visuels qui seront sous forme de cartes, et on décide de choisir les informations que l'on mettra selon la requête que nous avons écrite. Ici film[2] et film[1] va donc permettre de voir la 2eme information de notre requête, puis la 1ere.

Et pour chacune de ces pages, on prendra avec la requête la clé primaire qui permettra d'aller sur les données d'une table en particulière (celle qui sera cliqué) et on doit alors créer une page détails pour qui les données vont varier selon la clé primaire

```

@app.route("/film/<int:visa>")
def detailsfilm(visa):
    with db.connect() as conn:
        with conn.cursor() as cur:
            #cur.execute("SELECT * FROM film WHERE visa = %s;", (visa,))
            cur.execute("""SELECT f.*,CONCAT(a.prenom, ' ', a.nom) AS acteur,CONCAT(r.prenom,' ', r.nom) FROM film f
                        JOIN acting ac ON f.visa = ac.visa
                        JOIN acteur act ON ac.id_acteur = act.id_acteur
                        JOIN vip a ON a.id_vip = act.id_vip
                        JOIN vip r ON r.id_vip = f.id_vip
                        WHERE f.visa = %s ;""", (visa,))
            film = cur.fetchall()
            print(film)
        conn.close()
    return render_template("detailsfilm.html", film=film)

```

Pour créer cette page on va utiliser « @app.route("/film/<int:visa>") » car on était sur la page film et le visa change selon le film sur lequel on va cliquer.

Et pour afficher toutes les infos on a alors écrit une requête qui prends bien en compte Toutes les infos de la table film, et certaines infos des tables acteur, acting, vip afin de bien retrouver les acteurs qui avaient tous ce visa de film en commun.
Ce raisonnement est à peu près le même pour toutes les pages du site.

B- Page Admin :

```
@app.route('/manage', methods=['GET', 'POST'])
def manage():
    if 'user' not in session:
        return redirect(url_for('log_admin'))
    conn = db.connect()
    cursor = conn.cursor(cursor_factory=RealDictCursor)

    if request.method == 'POST':
        action = request.form.get('action')
        # Ajouter un VIP
        if action == 'add_vip':
            nom = request.form.get('nom')
            prenom = request.form.get('prenom')
            sexe = request.form.get('sexe')
            nationalite = request.form.get('nationalite')
            date_naissance = request.form.get('date_naissance')
            date_deces = request.form.get('date_deces') or None
            cursor.execute("INSERT INTO vip (nom, prenom, sexe, nationalite, date_naissance, date_deces) VALUES (%s, %s, %s, %s, %s, %s)", (nom, prenom, sexe, nationalite, date_naissance, date_deces))
        elif action == 'update_vip':
            id_vip = request.form.get('id_vip')
            nom = request.form.get('nom')
            prenom = request.form.get('prenom')
            sexe = request.form.get('sexe')
            nationalite = request.form.get('nationalite')
            date_naissance = request.form.get('date_naissance')
            date_deces = request.form.get('date_deces') or None
            cursor.execute("UPDATE vip SET nom = %s, prenom = %s, sexe = %s, nationalite = %s, date_naissance = %s, date_deces = %s WHERE id_vip = %s", (nom, prenom, sexe, nationalite, date_naissance, date_deces, id_vip))
        # Supprimer un VIP
        elif action == 'delete_vip':
            id_vip = request.form.get('id_vip')
            cursor.execute("DELETE FROM vip WHERE id_vip = %s", (id_vip,))
        conn.commit()

    cursor.execute("SELECT * FROM vip")
    vip = cursor.fetchall()
    conn.close()
    return render_template('manage.html', vips=vip)
```

Pour la partie admin les fonctions sont aussi assez similaires. On vérifie dans un premier temps si le log est le bon, si ce n'est pas le cas on renvoie l'utilisateur retaper son mot de passe.

Pour l'ajout d'un élément dans la table on utilise méthode de request POST, qui est la plus approprié pour la récupération de données, pour le formulaire qui se fera du côté HTML.

Pour chaque attribut de notre table, sur cet exemple pour la table VIP, on récupère à l'aide d'un formulaire les informations taper sur le site.

Puis on exécute notre requête pour pouvoir ajouter les éléments dans notre table.

Pour la suppression et la mise à jour des informations c'est le même procédé mais avec des requêtes différentes.

On récupère aussi les données de toutes la table dans une liste, pour pouvoir l'utiliser dans le HTML à l'aide de Jinja.

```

11 </li><a href= {{ URI_for( logout ) }} >Deconnexion</a></li>
12 <form method="post">
13   <h3>Ajouter un VIP</h3>
14   <input type="hidden" name="action" value="add_vip">
15   <input type="text" name="nom" placeholder="Nom">
16   <input type="text" name="prenom" placeholder="Prenom">
17   <input type="text" name="sexe" placeholder="Sexe">
18   <input type="text" name="nationalite" placeholder="Nationalité">
19   <input type="text" name="date_naissance" placeholder = "date_naissance">
20   <input type="text" name="date_deces" placeholder="date_deces">
21   <button type="submit">Ajouter</button>
22 </form>
23
24 <form method="post">
25   <h3>Update un VIP</h3>
26   <input type="hidden" name="action" value="update_vip">
27   <input type="number" name="id_vip" placeholder="id_vip">
28   <input type="text" name="nom" placeholder="Nom">
29   <input type="text" name="prenom" placeholder="Prenom">
30   <input type="text" name="sexe" placeholder="Sexe">
31   <input type="text" name="nationalite" placeholder="Nationalité">
32   <input type="text" name="date_naissance" placeholder = "date_naissance">
33   <input type="text" name="date_deces" placeholder="date_deces">
34   <button type="submit">Update</button>
35 </form>

```

Ici on récupère les données avec des formulaires POST.
On différencie l'ajout, l'update et la suppression grâce à cette ligne :

```
<input type="hidden" name="action" value="add_vip">
```

On affiche alors les données dans un tableau, en vérifiant de notre liste ne soit pas vide :

```

<table border="1">
  <thead>
    <tr>
      <th>id_vip</th>
      <th>Nom</th>
      <th>Prenom</th>
      <th>Sexe</th>
      <th>Nationalité</th>
      <th>Date de naissance</th>
      <th>Date de décès</th>
    </tr>
  </thead>
  <tbody>
    {% for vip in vips %}
    <tr>
      <td>{{ (vip.id_vip) }}</td>
      <td>{{ vip.nom }}</td>
      <td>{{ vip.prenom }}</td>
      <td>{{ vip.sexe }}</td>
      <td>{{ vip.nationalite }}</td>
      <td>{{ vip.date_naissance }}</td>
      <td>{{ vip.date_deces }}</td>
      <td>
        <form method="post" style="display:inline;">
          <input type="hidden" name="action" value="delete_vip">
          <input type="hidden" name="id_vip" value="{{ vip.id_vip }}">
          <button type="submit" name = "delete_vip">Supprimer</button>
        </form>
      </td>
    </tr>
    {% endfor %}
  </tbody>
</table>

```

Pour la suppression, on récupère l'ID correspondant à la ligne que l'on veut supprimer.

4 – pistes d'améliorations

Concernant les pistes améliorations visuels, on pourrait afficher une photo de chaque star sur leur carte dans la page célébrités, la couverture officielle de film et d'album pour chaque film et album, et une photo du défile pour chaque défilé.

On pourrait également rajouter un lien qui retournent vers la page de détails de la célébrité sur laquelle on va cliquer quand on voit son nom dans la page détails album, détails film etc. et pareil si on veut cliquer sur la page détails d'un album, d'un film ou d'un défilé en particulier que l'on voit depuis la page détails de la célébrité.

5 – Organisation de travail

Dès le départ nous avons beaucoup aimé le thème sur lequel nous sommes tombés car cela se rapproche beaucoup de nos centres d'intérêt, on a eu beaucoup de temps entre chaque étape ce qui nous a permis de nous organiser et d'avancer petit à petit sans tout faire à la dernière minute.

Lors des deux premières étapes il ne fallait pas beaucoup d'organisation alors nous avons chacun fait une version chez nous, puis on a mis en commun nos 2 schémas et ensuite on s'est appelé via discord tout en regardant le schéma mis en commun et en réfléchissant aux éventuelles erreurs ou améliorations.

Pour la 3eme étape il fallait beaucoup de temps alors on se rejoignais presque tous les jours à la bibliothèque afin de travailler ensemble et de pouvoir avancer sur des taches ou des problèmes tout en faisant en sorte que l'autre comprenne ce qui a été fait en plus, puis après chaque session on mettait en commun le travail effectué.

On avait commencé par la page accueil (sans la recherche) puis admin, puis les autres pages de bases, et on a fini avec la recherche inversée et les pages de détails.